A Hybrid Approach to Privacy-Preserving Federated Learning

Stacey Truex
staceytruex@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Nathalie Baracaldo
baracald@us.ibm.com
IBM Research Almaden
San Jose, California

Ali Anwar
Ali.Anwar2@ibm.com
IBM Research Almaden
San Jose, California

Thomas Steinke
Thomas.Steinke@ibm.com
IBM Research Almaden
San Jose, California

Heiko Ludwig
hludwig@us.ibm.com
IBM Research Almaden
San Jose, California

Rui Zhang
ruiz@us.ibm.com
IBM Research Almaden
San Jose, California

Yi Zhou
yi.zhou@ibm.com
IBM Research Almaden
San Jose, California

ABSTRACT

Federated learning facilitates the collaborative training of models without the sharing of raw data. However, recent attacks demonstrate that simply maintaining data locality during training processes does not provide sufficient privacy guarantees. Rather, we need a federated learning system capable of preventing inference over both the messages exchanged during training and the final trained model while ensuring the resulting model also has acceptable predictive accuracy. Existing federated learning approaches either use secure multiparty computation (SMC) which is vulnerable to inference or differential privacy which can lead to low accuracy given a large number of parties with relatively small amounts of data each. In this paper, we present an alternative approach that utilizes both differential privacy and SMC to balance these trade-offs. Combining differential privacy with secure multiparty computation enables us to reduce the growth of noise injection as the number of parties increases without sacrificing privacy while maintaining a pre-defined rate of trust. Our system is therefore a scalable approach that protects against inference threats and produces models with high accuracy. Additionally, our system can be used to train a variety of machine learning models, which we validate with experimental results on 3 different machine learning algorithms. Our experiments demonstrate that our approach out-performs state of the art solutions.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; *Trust frameworks*; • **Computing methodologies → Learning settings**.

AISec'19, November 15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6833-9/19/11…$15.00
https://doi.org/10.1145/3338501.3357370

KEYWORDS

Privacy, Federated Learning, Privacy-Preserving Machine Learning, Differential Privacy, Secure Multiparty Computation

ACM Reference Format:
Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A Hybrid Approach to Privacy-Preserving Federated Learning. In *12th ACM Workshop on Artificial Intelligence and Security (AISec'19), November 15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3338501.3357370

1 INTRODUCTION

In traditional machine learning (ML) environments, training data is centrally held by one organization executing the learning algorithm. Distributed learning systems extend this approach by using a set of learning nodes accessing shared data or having the data sent to the participating nodes from a central node, all of which are fully trusted. For example, MLlib from Apache Spark assumes a trusted central node to coordinate distributed learning processes [28]. Another approach is the parameter server [26], which again requires a fully trusted central node to collect and aggregate parameters from the many nodes learning on their different datasets.

However, some learning scenarios must address less open trust boundaries, particularly when multiple organizations are involved. While a larger dataset improves the performance of a trained model, organizations often cannot share data due to legal restrictions or competition between participants. For example, consider three hospitals with different owners serving the same city. Rather than each hospital creating their own predictive model forecasting cancer risks for their patients, the hospitals want to create a model learned over the whole patient population. However, privacy laws prohibit them from sharing their patients' data. Similarly, a service provider may collect usage data both in Europe and the United States. Due to legislative restrictions, the service provider's data cannot be stored in one central location. When creating a predictive model forecasting service usage, however, all datasets should be used.

The area of federated learning (FL) addresses these more restrictive environments wherein data holders collaborate throughout the learning process rather than relying on a trusted third party to hold

data [6, 39]. Data holders in FL run a machine learning algorithm locally and only exchange model parameters, which are aggregated and redistributed by one or more central entities. However, this approach is not sufficient to provide reasonable data privacy guarantees. We must also consider that information can be inferred from the learning process [30] and that information that can be traced back to its source in the resulting trained model [40].

Some previous work has proposed a trusted aggregator as a way to control privacy exposure [1], [32]. FL schemes using Local Differential Privacy also address the privacy problem [39] but entails adding too much noise to model parameter data from each node, often yielding poor performance of the resulting model.

We propose a novel federated learning system which provides formal privacy guarantees, accounts for various trust scenarios, and produces models with increased accuracy when compared with existing privacy-preserving approaches. Data never leaves the participants and privacy is guaranteed using secure multiparty computation (SMC) and differential privacy. We account for potential inference from individual participants as well as the risk of collusion amongst the participating parties through a customizable trust threshold. Our **contributions** are the following:

- We propose and implement an FL system providing formal privacy guarantees and models with improved accuracy compared to existing approaches.
- We include a tunable trust parameter which accounts for various trust scenarios while maintaining the improved accuracy and formal privacy guarantees.
- We demonstrate that it is possible to use the proposed approach to train a variety of ML models through the experimental evaluation of our system with three significantly different ML models: decision trees, convolutional neural networks and linear support vector machines.
- We include the first federated approach for the private *and* accurate training of a neural network model.

The rest of this paper is organized as follows. We outline the building blocks in our system. We then discuss the various privacy considerations in FL systems followed by outlining our threat model and general system. We then provide experimental evaluation and discussion of the system implementation process. Finally, we give an overview of related work and some concluding remarks.

2 PRELIMINARIES

In this section we introduce building blocks of our approach and explain how various approaches fail to protect data privacy in FL.

2.1 Differential Privacy

Differential privacy (DP) is a rigorous mathematical framework wherein an algorithm may be described as differentially private if and only if the inclusion of a single instance in the training dataset causes only statistically insignificant changes to the algorithm's output. For example, consider private medical information from a particular hospital. The authors in [40] have shown that with access to only a trained ML model, attackers can infer whether or not an individual was a patient at the hospital, violating their right to privacy. DP puts a theoretical limit on the influence of a single individual, thus limiting an attacker's ability to infer such membership. The formal definition for DP is [13]:

Definition 1 (Differential Privacy). *A randomized mechanism $\mathcal{K}$ provides (ϵ, δ)- differential privacy if for any two neighboring database D_1 and D_2 that differ in only a single entry, $\forall S \subseteq Range(\mathcal{K})$,*

$$\Pr(\mathcal{K}(D_1) \in S) \leq e^{\epsilon} \Pr(\mathcal{K}(D_2) \in S) + \delta \quad (1)$$

If $\delta = 0$, $\mathcal{K}$ is said to satisfy ϵ-differential privacy.

To achieve DP, noise is added to the algorithm's output. This noise is proportional to the sensitivity of the output, where sensitivity measures the maximum change of the output due to the inclusion of a single data instance.

Two popular mechanisms for achieving DP are the Laplacian and Gaussian mechanisms. Gaussian is defined by

$$M(D) \triangleq f(D) + N(0, S_f^2 \sigma^2), \quad (2)$$

where $N(0, S_f^2\sigma^2)$ is the normal distribution with mean 0 and standard deviation $S_f\sigma$. A single application of the Gaussian mechanism to function f of sensitivity S_f satisfies (ϵ, δ)-differential privacy if $\delta \geq \frac{5}{4} exp(-(\sigma\epsilon)^2/2)$ and $\epsilon < 1$ [16].

To achieve ϵ-differential privacy, the Laplace mechanism may be used in the same manner by substituting $N(0, S_f^2\sigma^2)$ with random variables drawn from $Lap(S_f/\epsilon)$ [16].

When an algorithm requires multiple additive noise mechanisms, the evaluation of the privacy guarantee follows from the basic composition theorem [14, 15] or from advanced composition theorems and their extensions [7, 17, 18, 23].

2.2 Threshold Homomorphic Encryption

An additively homomorphic encryption scheme is one wherein the following property is guaranteed:

$$Enc(m_1) \circ Enc(m_2) = Enc(m_1 + m_2),$$

for some predefined function $\circ$. Such schemes are popular in privacy-preserving data analytics as untrusted parties can perform operations on encrypted values.

One such additive homomorphic scheme is the Paillier cryptosystem [31], a probabilistic encryption scheme based on computations in the group $\mathbb{Z}^*_{n^2}$, where n is an RSA modulus. In [11] the authors extend this encryption scheme and propose a threshold variant. In the threshold variant, a set of participants is able to share the secret key such that no subset of the parties smaller than a pre-defined threshold is able to decrypt values.

2.3 Privacy in Federated Learning

In centralized learning environments a single party P using a dataset D executes some learning algorithm f_M resulting in a model M where $f_M(D) = M$. In this case P has access to the complete dataset D. By contrast, in a federated learning environment, multiple parties $P_1, P_2, ..., P_n$, each have their own dataset $D_1, D_2, ..., D_n$, respectively. The goal is then to learn a model using all of the datasets.

We must consider two potential threats to data privacy in such an FL environment: (1) inference during the learning process and (2) inference over the outputs. *Inference during the learning process* refers to any participant in the federation inferring information about another participant's private dataset given the data exchanged during

the execution of f_M. *Inference over the outputs* refers to the leakage of any participants' data from intermediate outputs as well as M.

We consider two types of inference attacks: insider and outsider. *Insider attacks* include those launched by participants in the FL system, including both data holders as well as any third parties, while *outsider attacks* include those launched both by eavesdroppers to the communication between participants and by users of the final predictive model when deployed as a service.

2.3.1 Inference during the learning process. Let us consider f_M as the combination of computational operations and a set of queries $Q_1, Q_2, ..., Q_k$. That is, for each step s in f_M requiring knowledge of the parties' data there is a query Q_s. In the execution of f_M each party P_i must respond to each such query Q_s with appropriate information on D_i. The types of queries are highly dependent on f_M. For example, to build a decision tree, a query may request the number of instances in D_i matching a certain criteria. In contrast, to train an SVM or neural network a query would request model parameters after a certain number of training iterations. Any privacy-preserving FL system must account for the risk of inference over the responses to these queries.

Privacy-preserving ML approaches addressing this risk often do so by using secure multiparty computation (SMC). Generally, SMC protocols allow n parties to obtain the output of a function over their n inputs while preventing knowledge of anything other than this output [20]. Unfortunately, approaches exclusively using secure multiparty computation remain vulnerable to inference over the output. As the function output remains unchanged from function execution without privacy, the output can reveal information about individual inputs. Therefore, we must also consider potential inference over outputs.

2.3.2 Inference over the outputs. This refers to intermediate outputs available to participants as well as the predictive model. Recent work shows that given only black-box access to the model through an ML as a service API, an attacker can still make training data inferences [40]. An FL system should prevent such outsider attacks while also considering insiders. That is, participant P_i should not be able to infer information about D_j when $i \neq j$ as shown in [30].

Solutions addressing privacy of output often make use of the DP framework discussed in Preliminaries. As a mechanism satisfying differential privacy guarantees that if an individual contained in a given dataset is removed, no outputs would become significantly more or less likely [13], a learning algorithm f_M which is theoretically proven to be ϵ-differentially private is guaranteed to have a certain privacy of output quantified by the ϵ privacy parameter.

In the federated learning setting it is important to note that the definition of neighboring databases is consistent with the usual DP definition – that is, privacy is provided at the individual record level, not the party level (which may represent many individuals).

3 AN END-TO-END APPROACH WITH TRUST

3.1 Threat Model

We propose a system wherein n data parties use an ML service for FL. We refer to this service as the *aggregator*. Our system is designed to withstand three potential adversaries: (1) the aggregator, (2) the data parties, and (3) outsiders.

3.1.1 Honest-But-Curious Aggregator. The honest-but-curious or semi-honest adversarial model is commonly used in the field of SMC since its introduction in [3] and application to data mining in [27]. Honest-but-curious adversaries follow the protocol instructions correctly but will try to learn additional information. Therefore, the aggregator *will not* vary from the predetermined ML algorithm but *will* attempt to infer private information using all data received throughout the protocol execution.

3.1.2 Colluding Parties. Our work also considers the threat of collusion among parties, including the aggregator, through the trust parameter t which is the minimum number of non-colluding parties. Additionally, in contrast to the aggregator, we consider scenarios in which parties in $\mathcal{P}$ may deviate from the protocol execution to achieve additional information on data held by honest parties.

3.1.3 Outsiders. We also consider potential attacks from adversaries outside of the system. Our work ensures that any adversary monitoring communications during training cannot infer the private data of the participants. We also consider users of the final model as potential adversaries. A predictive model output from our system may therefore be deployed as a service, remaining resilient to inference against adversaries who may be users of the service.

We now detail the assumptions made in our system to more concretely formulate our threat model.

3.1.4 Communication. We assume secure channels between each party and the aggregator. This allows the aggregator to authenticate incoming messages and prevents an adversary, whether they be an outsider or malicious data party, from injecting their own responses.

3.1.5 System set up. We additionally make use of the *threshold variant of the Paillier encryption scheme* from [11] assuming secure key distribution. It is sufficient within our system to say that semantic security of encrypted communication is equivalent to the decisional composite residuosity assumption. For further discussion we direct the reader to [11]. Our use of the threshold variant of the Paillier system ensures that any set of $n - t$ or fewer parties cannot decrypt ciphertexts. Within the context of our FL system, this ensures the privacy of individual messages sent to the aggregator.

3.2 Proposed Approach

We propose an FL system that addresses risk of inference during the learning process, risk of inference over the outputs, *and* trust. We combine methods from SMC and DP to develop protocols that guarantee privacy without sacrificing accuracy.

We consider the following scenario. There exists a set of n parties $\mathcal{P} = P_1, P_2, ..., P_n$, a set of disjoint datasets $D_1, D_2, ..., D_n$ belonging to the respective parties and adhering to the same structure, and an aggregator $\mathcal{A}$. Our system takes as additional input three parameters: f_M, ϵ, and t. f_M specifies the training algorithm, ϵ is the privacy guarantee against inference, and t specifies the minimum number of honest, non-colluding parties.

The aggregator $\mathcal{A}$ runs the learning algorithm f_M consisting of k or fewer linear queries $Q_1, Q_2, ..., Q_k$, each requiring information from the n datasets. This information may include model parameters after some local learning on each dataset or may be

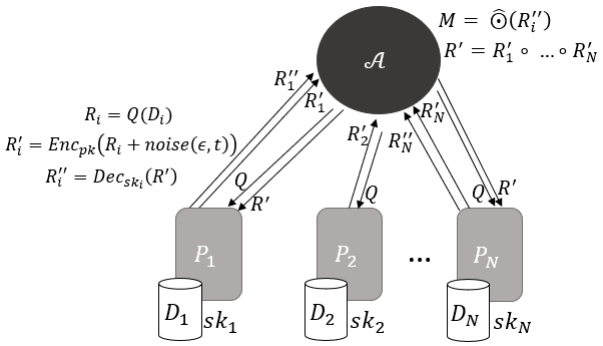

Figure 1: System Overview

more traditionally queried information such as how many individuals match a set of criterion. For each algorithm deployed into our system, any step s requiring such information reflective of some analysis on the private, local datasets must be represented by a corresponding query Q_s. Figure 1 shows an outline of a step s in f_M. Using secure channels between $\mathcal{A}$ and each of the parties, the aggregator will send query Q_s. Each party will then calculate a response using their respective datasets.

The participant will use a differential privacy mechanism that depends on the algorithm f_M to add the appropriate amount of noise according to the privacy budget allocated to that step, the sensitivity of Q_s, and the level of trust in the system. The noisy response is then encrypted using the threshold variant of the Paillier cryptosystem and sent to $\mathcal{A}$. Homomorphic properties then allow $\mathcal{A}$ to aggregate the individual responses. $\mathcal{A}$ subsequently queries at least $n - t + 1$ data parties to decrypt the aggregate value and updates the model M. At the conclusion of f_M, the model M is exposed to all participants. This process is outlined in Algorithm 1.

We consider trust with respect to collusion in two steps: (1) in the addition of noise and (2) in the threshold setting of the encryption scheme. The more participants colluding, the more knowledge which is available to infer the data of an honest participant. Therefore, the noise introduced by an honest participant must account for collusion. The use of homomorphic encryption however allows for significant increases in accuracy (over local privacy approaches). We now detail this strategy for FL.

3.3 Reducing Noise with SMC

A key component to our system is the ability to reduce noise by leveraging the SMC framework while considering a customizable trust parameter.

Specifically, let σ_s and S_s respectively be the noise parameter and sensitivity to step Q_s allocated a budget ϵ_s in the learning algorithm f_M. In a traditional application of differential privacy to federated learning, each party will use the Gaussian mechanism to add $N(0, S_s^2\sigma_s^2)$ noise to their response $r_{i,s}$ when queried by $\mathcal{A}$ at step Q_s. This guarantees the privacy of each $r_{i,s}$.

If, however, each $r_{i,s}$ is encrypted using the scheme proposed in [11] with a threshold setting of $\bar{t} = n - t + 1$, the noise may be reduced by a factor of $t - 1$. Rather than returning $Q_s(D_i) + N(0, S_s^2\sigma_s^2)$, each party may return $Enc(Q_s(D_i) + N(0, S_s^2 \frac{\sigma_s^2}{t-1}))$.

Note that when $\mathcal{A}$ aggregates these responses the value that is eventually decrypted and exposed will be $\sum_{i=1}^{n} Q_s(P_i) + Y_i$ where

Algorithm 1 Private Federated Learning

Input: ML algorithm f_M; set of data parties $\mathcal{P}$ of size N, with each $P_i \in \mathcal{P}$ holding a private dataset D_i and a portion of the secret key sk_i; minimum number of honest, non-colluding parties t; privacy guarantee ϵ
$\bar{t} = n - t + 1$
for each $Q_s \in f_M$ **do**
  for each $P_i \in \mathcal{P}$ **do**
    $\mathcal{A}$ asynchronously queries P_i with Q_s
    P_i sends $r_{i,s} = Enc_{pk}(Q_s(D_i) + noise(\epsilon, t))$
  end for
  $\mathcal{A}$ aggregates $Enc_{pk}(r_s) \leftarrow r_{1,s} \circ r_{2,s} \circ \ldots \circ r_{N,s}$
  $\mathcal{A}$ selects $\mathcal{P}_{dec} \subseteq \mathcal{P}$ such that $|\mathcal{P}_{dec}| = \bar{t}$
  for each $P_i \in \mathcal{P}_{dec}$ **do**
    $\mathcal{A}$ asynchronously queries P_i with $Enc_{pk}(r_s)$
    $\mathcal{A}$ receives partial decryption of r_s from P_i using sk_i
  end for
  $\mathcal{A}$ computes r_s from partial decryptions
  $\mathcal{A}$ updates M with r_s
end for
return M

each Y_i is drawn from the Gaussian distribution with standard deviation $S_s \frac{\sigma_s}{\sqrt{t-1}}$. This is equivalent to $N(0, S_s^2 \frac{n\sigma_s^2}{t-1}) \sum_{i=1}^{n} Q_s(D_i)$. Since we know that $t - 1 < n$, the noise included in the decrypted value is strictly greater than that required to satisfy DP. Additionally, the encryption scheme guarantees that the maximum number of colluders, $\bar{t}$, cannot decrypt values of honest parties.

Given this approach, we are able to maintain the customizable nature of our system with the trust parameter t and the formal privacy guarantees of the DP framework while decreasing the amount of noise *for each query response* leading to more accurate ML models.

4 EXPERIMENTAL EVALUATION

In this section we empirically demonstrate how to apply our approach to train three distinct learning models: decision trees (DT), convolutional neural networks (CNN) and linear Support Vector Machines (SVM). We additionally provide analysis on the impact of certain settings on the performance of our approach.

4.1 Decision Trees

We first consider DT learning using the ID3 algorithm. In this scenario, each dataset D_i owned by some $P_i \in \mathcal{P}$ contains a set of instances described by the same set of categorical features $\mathcal{F}$ and a class attribute C. The aggregator initializes the DT model with a root node. Then, the feature $F \in \mathcal{F}$ that maximizes information gain is chosen based on counts queried from each party in $\mathcal{P}$ and child nodes are generated for each possible value of F. The feature F is then removed from $\mathcal{F}$. This process continues recursively for each child node until either (a) there are no more features in $\mathcal{F}$, (b) a pre-determined max-depth is reached, or (c) responses are too noisy to be deemed meaningful. This process is specifically detailed as algorithmic pseudocode in Section 5.1.

There are two types of participant queries in private, federated DT learning: *counts* and *class_counts*. For executing these queries $\mathcal{A}$ first divides the entire privacy budget ϵ equally between each layer of the tree. According to the composition property of differential privacy, because different nodes within the same layer are evaluated on disjoint subsets of the datasets, they do not accumulate privacy loss and therefore the budget allocated to a single layer is not divided further. Within each node, half of the budget (ϵ_1) is allocated to determining total counts and half is allocated to either class

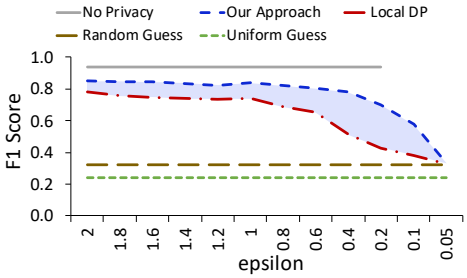

Figure 2: Effect of privacy budgets on the overall F1-score for Decision Trees

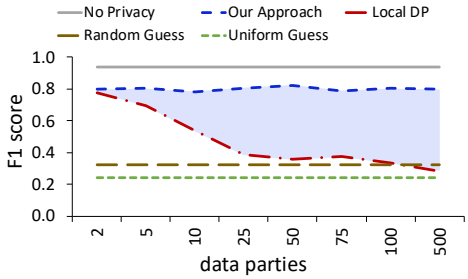

Figure 3: Effect of increasing number of parties on the overall F1-score for Decision Trees

counts (done at the leaf nodes) or evaluating attributes (done at internal nodes). For internal nodes, each feature is evaluated for potential splitting against the same dataset. The budget allocated to evaluating attributes must therefore be divided amongst each feature (ϵ_2). In all experiments the max depth is set to $d = \frac{|\mathcal{F}|}{2}$.

Dataset. We conduct a number of experiments using the Nursery dataset from the UCI Machine Learning Repository [12]. This dataset contains 8 categorical attributes about 12,960 nursery school applications. The target attribute has five distinct classes with the following distribution: 33.333%, 0.015%, 2.531%, 32.917%, 31.204%.

Comparison Methods. To put model performance into context, we compare with two different random baselines and two current FL approaches. Random baselines enable us to characterize when a particular approach is no longer learning meaningful information while the FL approaches visualize relative performance cost.

(1) Uniform Guess. In this approach, class predictions are randomly sampled with a $\frac{1}{|C|}$ chance for each class.
(2) Random Guess. Random guess improves upon Uniform Guess with consideration of class value distribution in the training data. At test time, each prediction is sampled from the set of training class labels.
(3) Local DP. In the local approach, parties add noise to protect the privacy of their own data in isolation. The amount of noise necessary to provide ϵ-differential privacy to each dataset is defined in [5].
(4) No Privacy. This is the result of running the distributed learning algorithm without any privacy guarantee.

4.1.1 ***Variation in Settings.*** We now look at how different settings impact results.

Privacy Budget. We first look at the impact of the privacy budget on performance in our system. To isolate the impact of the privacy budget we set the number of parties, $|\mathcal{P}|$, to 10 and assume no collusion. We consider budget values between 0.05 and 2.0. Recall from Preliminaries that for a mechanism to be ϵ-differentially private the amount of noise added will be inversely proportional to value of ϵ. In other words, the smaller the ϵ value, the smaller the privacy budget, and the more noise added to each query.

We can see in Figure 2 that our approach maintains an F1-score above 0.8 for privacy budgets as small as 0.4. Once the budget dips below 0.4 we see the noise begins to overwhelm the information being provided which can have one of two outcomes: (1) learning pre-maturely haults or (2) learning become inaccurate. This results in degraded performance as the budget decreases, which is expected. It is clear that our approach maintains improved performance over the local DP approach for all budgets (until both approaches converge to the random guessing baseline). Particularly as the budget decreases from 1.0 to 0.4 we see our approach maintaining better resilience to the decrease in the privacy budget.

Number of Parties. Another important consideration for FL systems is the ability to maintain accuracy in highly distributed scenarios. That is, when many parties, each with a small amount of data, such as in an IoT scenario, are contributing to the learning.

In Figures 3 and 4 we show the impact that $|\mathcal{P}|$ has on performance. The results are for a fixed overall privacy budget of 0.5 and assume no collusion. For each experiment, the overall dataset was divided into $|\mathcal{P}|$ equal sized partitions.

The results in Figure 3 demonstrate the viability of our system for FL in highly distributed environments while highlighting the shortcomings of the local DP approach. As $|\mathcal{P}|$ increases, the noise in the local DP approach increases proportionally while our approach maintains consistent accuracy. We can see that with as few as 25 parties, the local DP results begin to approach the baseline and even dip below random guessing by 100 participants.

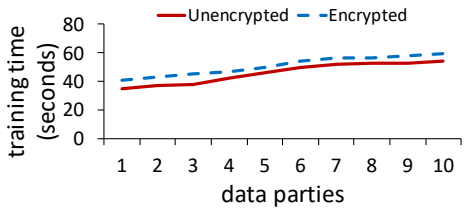

Figure 4: Decision Tree Training Time with Encryption

Another consideration relative to the scaling of participants is the overhead of encryption. Figure 4 highlights the scalability of our system, showing the impact that encryption has on overall training time in our system as the number of parties increases from 1 to 10. While the entire system experiences a steady increase in cost as the number of participants increases, the impact of the encryption remains consistent. Because our system is designed for a distributed scenario, the interactions with the aggregator are done in parallel and therefore the overhead of encryption remains constant as the number of parties increases.

Trust. An important part of our system is the trust parameter. While the definition of a neighboring database within the context of the differential privacy framework considers privacy at the

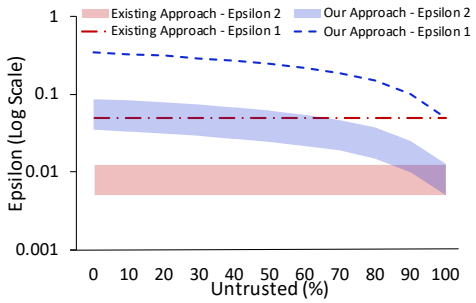

Figure 5: Query Epsilons in Decision Tree Training with Varying Rate of Trust (50 parties). Epsilon 1 is defined as the privacy budget for count queries while Epsilon 2 is used for class counts.

record level, the trust model for adversarial knowledge is considered within the context of the entire system. The trust parameter therefore represents the degree of adversarial knowledge by capturing the maximum number of colluding parties which the system may tolerate. Figure 5 demonstrates how the ϵ values used for both count and distribution queries in private, federated DT learning are impacted by the trust parameter setting when $|\mathcal{P}| = 50$.

In the worst case scenario where a party $P_i \in \mathcal{P}$ assumes that all other $P_j \in \mathcal{P}, i \neq j$ are colluding, our approach converges with existing local DP approaches. In all other scenarios the query ϵ values will be increased in our system leading to more accurate outcomes. Additionally, we believe the aforementioned scenario of no trust is unlikely to exist in real world instances. Let us consider smart phone users as an IoT example. Collusion of all but one party is impractical not only due to scale but also since such a system is likely the be running without many users even knowing. Additionally, on a smaller scale, if there is a set of five parties in the system and one party is concerned that the other four are all colluding, there is no reason for the honest party to continue to participate. We therefore believe that realistic scenarios of FL will see accuracy gains when deploying our system.

4.2 Convolutional Neural Networks

We additionally demonstrate how to use our method to train a distributed differentially private CNN. In our approach, similarly to centrally trained CNNs, each party is sent a model with the same initial structure and randomly initialized parameters. Each party will then conduct one full epoch of learning locally. At the conclusion of each batch, Gaussian noise is introduced according to the norm clipping value c and the privacy parameter σ. Norm clipping allows us to put a bound on the sensitivity of the gradient update. We use the same privacy strategy used in the centralized training approach presented in [1]. Once an entire epoch, or $\frac{1}{b}$ batches where b = batch rate, has completed the final parameters are sent back to $\mathcal{A}$. $\mathcal{A}$ then averages the parameters and sends back an updated model for another epoch of learning. After a predetermined E number of epochs, the final model M is output. This process for the aggregator and data parties are specifically detailed as algorithmic pseudocode in Section 5.2.

Within our private, federated NN learning system, if $\sigma = \sqrt{2 \cdot \log \frac{1.25}{\delta}}/\epsilon$ then by [16] our approach is (ϵ, δ)-differentially private with respect

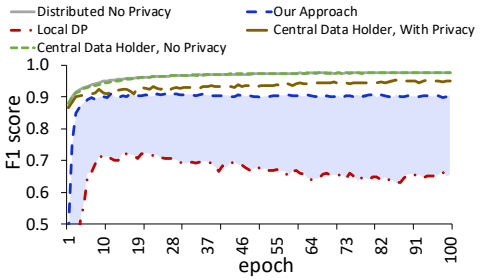

Figure 6: Convolutional Neural Network Training with MNIST Data (10 parties and σ = 8, (ϵ, δ) = (0.5, 10^{-5}))

to each randomly sampled batch. Using the moments accountant in [1], our approach is $(O(b\epsilon\sqrt{E/b}), \delta)$-DP overall.

Dataset and Model Structure. For our CNN experiments we use the publicly available MNIST dataset. This includes 60,000 training instances of handwritten digits and 10,000 testing instances. Each example is a 28x28 grey-scale image of a digit between 0 and 9 [24].

We use a model structure similar to that in [1]. Our model is a feedforward neural network with 2 internal layers of ReLU units and a softmax layer of 10 classes with cross-entropy loss. The first layer contains 60 units and the second layer contains 1000. We set the norm clipping to 4.0, learning rate to 0.1 and batch rate to 0.01. We use Keras with a Tensorflow backend.

Comparison Methods. To the best of our knowledge, this paper presents the first approach to accurately train a neural network in a private federated fashion without reliance on any public or non-protected data. We therefore compare our approach with the following baselines:

(1) Central Data Holder, No Privacy. In this approach all the data is centrally held by one party and no privacy is considered in the learning process.
(2) Central Data Holder, With Privacy. While all the data is still centrally held by one entity, this data holder now conducts privacy-preserving learning. This is representative of the scenario in [1].
(3) Distributed, No Privacy. In this approach the data is distributed to multiple parties, but the parties do not add noise during the learning process.
(4) Local DP. Parties add noise to protect the privacy of their own data in isolation, adapting from [1] and [39].

Figure 6 shows results with 10 parties conducting 100 epochs of training with the privacy parameter σ set to 8.0, the "large noise" setting in [1]. Note that Central Data Holder, No Privacy and Distributed Data, No Privacy achieve similar results and thus overlap. Our model is able to achieve an F1-score in this setting of 0.9. While this is lower than the central data holder setting where an F1-score of approximately 0.95 is achieved, our approach again significantly out-performs the local approach which only reaches 0.723. Additionally, we see a drop off in the performance of the local approach early on as updates become overwhelmed by noise.

We additionally experiment with $\sigma = 4$ and $\sigma = 2$ as was done in [1]. When $\sigma = 4$ $((\epsilon, \delta) = (2, 10^{-5}))$ the central data holder with privacy is able to reach an F1 score of 0.960, the local approach reaches 0.864, and our approach results in an F1-score of 0.957.

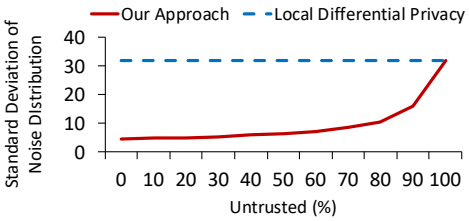

Figure 7: Degree of Noise in Convolutional Neural Network Training with Varying Rate of Trust

When $\sigma = 2$ ($(\epsilon, \delta) = (8, 10^{-5})$) those scores become 0.973, 0.937, and 0.963 respectively. We can see that our approach therefore demonstrates the most gain with larger σ values which translates to tighter privacy guarantees.

Figure 7 again shows how the standard deviation of noise is significantly decreased in our system for most scenarios.

Our experiments demonstrate that the encryption time for one parameter at a party P_i takes approximately 0.001095 sec while decryption between $\mathcal{A}$ and $\mathcal{P}_{dec}$ takes 0.007112 sec. While each parameter requires encryption and decryption, these processes can be done completely in parallel. Therefore overhead remains relatively constant as both $|\mathcal{P}|$ and the number of parameters increase.

Overall, we have demonstrated that our system provides significant accuracy gains for FL compared with local DP in plausible, real world scenarios and scales well.

4.3 Support Vector Machines (SVM)

We also demonstrate and assess our approach when solving a classic ℓ_2-regularized binary linear SVM problem with hinge loss.

To train the linear SVM in a private distributed fashion, the aggregator distributes a model with the same weight vector w to all parties. Each party then runs a predefined number of epochs to learn locally. To apply differential privacy in this setting, we first perform norm clipping on the feature vector x to obtain a bound on the sensitivity of the gradient update. Then, Gaussian noise is added to the gradient according to [25]. After each party completes its local training, the final noisy encrypted weights are sent back to the aggregator. The aggregator averages the encrypted weights and sends back an updated model with a new weight vector for another epoch of learning. Training ends after a predefined number of epochs. The detailed process is presented in Section 5.3.

Dataset. We use the publicly available 'gisette' dataset, which was used for NIPS 2003 Feature Selection Challenge [9]. This dataset has 6,000 training and 1,000 testing samples with 5,000 features.

Comparison Methods. We contrast the performance of our approach against other ways to train the model.

(1) Central, No Privacy. Centralized training without privacy.
(2) Central DP. Centralized training with DP.
(3) Distributed, No Privacy. Model trained through federated learning without privacy.
(4) Local DP. In this approach each party adds enough noise independently to protect their data according to [5].

In these experiments, the learning rate was set to 0.01 for all settings. We used 100 epochs for all approaches. Additionally, for

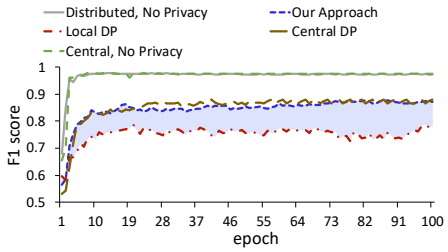

Figure 8: Linear SVM Training(10 parties and (ϵ, δ) = (5, 0.0059))

FL methods, each party runs 10 epochs locally. We used 10 non-colluding parties. Using σ=4, we report findings to achieve $(\epsilon, \delta) = (5, 0.0059)$ according to [7].

Figure 8 shows F1-scores for the evaluated training methods. Central, No Privacy and Distributed, No Privacy perform similarly with F1-scores around 0.99 after fewer than 10 epochs due to the balanced nature of the dataset. Among the privacy-preserving approaches, Central DP introduces the least noise and achieves the highest F1-score. Among private FL methods, our approach achieves an F1-score over 0.87 which is almost equal to Central DP and significantly out-performs Local DP after 100 epochs.

We also evaluated our system in a lower trust setting with only half of the parties trusted as non-colluding. Our approach again outperformed Local DP. Specifically, after 100 epochs, our approach reached an F1-score of 0.85, while the Local DP achieves only 0.75.

These experimental results show that our approach consistently out-performs state of the art methods to train different ML models in a private FL fashion. We similarly showed that our approach consistently out-performs baselines such as random guessing while remaining reasonably close to non-private settings.

5 SYSTEM IMPLEMENTATION

The development and deployment of new machine learning training algorithms to our system requires the training process be first broken down into a set of queries in which meaningful aggregation may be done via summation. Each query must then be analyzed for its privacy impact and and designated a portion of the overall privacy budget. Additionally, support must exist at each party for each type of query required by the private training algorithm. We will now provide implementation details for each of the model types evaluated, Decision Trees, Neural Networks, and Support Vector Machines, with additional discussion on the applicability of our framework to machine learning algorithms for other model types.

5.1 Application to Private Decision Tree Training

DT learning follows these steps: (1) determine the feature which best splits the training data, (2) split the training data into subsets according to the chosen feature, (3) repeat steps (1) and (2) for each subset. This is repeated until the subsets have reached a predetermined level of uniformity with respect to the target variable.

To conduct private decision tree learning in our system we first address step (1): determining the best feature on which to split the data. We define the "best" feature as the feature which maximizes information gain. This is the same metric used in the ID3 [34],

Algorithm 2 Private Decision Tree Learning

Input: Set of data parties $\mathcal{P}$; minimum number of honest, non-colluding parties t; privacy guarantee ϵ; attribute set $\mathcal{F}$; class attribute C; max tree depth d; public key pk
$\bar{t} = n - t + 1$
$\epsilon_1 = \frac{\epsilon}{2(d+1)}$
Define current splits, $\mathcal{S} = \emptyset$, for root node
M = BuildTree($\mathcal{S}$, $\mathcal{P}$, t, ϵ_1, $\mathcal{F}$, C, d, pk)
return M
procedure BuildTree($\mathcal{S}$, $\mathcal{P}$, t, ϵ_1, $\mathcal{F}$, C, d, pk)
  $f = \max_{F \in \mathcal{F}} |F|$
  Asynchronously query $\mathcal{P}$: *counts*($\mathcal{S}$, ϵ_1, t)
  N = decrypted aggregate of noisy counts
  if $\mathcal{F} = \emptyset$ or $d = 0$ or $\frac{N}{f|C|} < \frac{\sqrt{2}}{\epsilon_1}$ **then**
    Asynchronously query $\mathcal{P}$: *class_counts*($\mathcal{S}$, ϵ_1, t)
    N_c = vector of decrypted, noisy class counts
    return node labeled with $\arg\max_c N_c$
  else
    $\epsilon_2 = \frac{\epsilon_1}{2|\mathcal{F}|}$
    for each $F \in \mathcal{F}$ **do**
      for each $f_i \in F$ **do**
        Update set of split values to send to child node: $S_i = S + \{F = f_i\}$
        Asynchronously query $\mathcal{P}$: *counts*($\mathcal{S}_i$, ϵ_2, t)
        and *class_counts*($\mathcal{S}_i$, ϵ_2, t)
        N'^F_i = aggregate of *counts*
        $N'^F_{i,c}$ = element-wise aggregate of *class_counts*
        Recover N^F_i from $\bar{t}$ partial decryptions of N'^F_i
        Recover $N^F_{i,c}$ from $\bar{t}$ partial decryptions of $N'^F_{i,c}$
      end for
      $V_F = \sum_{i=1}^{|F|} \sum_{c=1}^{|C|} N^F_{i,c} \cdot \log \frac{N^F_{i,c}}{N^F_i}$
    end for
    $\bar{F} = \arg\max_F V_F$
    Create root node M with label $\bar{F}$
    for each $f_i \in \bar{F}$ **do**
      $S_i = S + \{F = f_i\}$
      M_i = *BuildTree*(S_i, $\mathcal{P}$, t, ϵ_1, $\mathcal{F} \setminus \bar{F}$, C, $d-1$, pk)
      Set M_i as child of M with edge f_i
    end for
    return M
  end if
end procedure

C4.5 [35] and C5.0 [36] tree training algorithms. Information gain for a candidate feature f quantifies the difference between the entropy of the current data with the weighted sum of the entropy values for each of the data subsets which would be generated if f were to be chosen as the splitting feature. Entropy for a dataset (or subset) D is computed via the following equation:

$$Entropy(D) = \sum_{i=1}^{|C|} p_i \log_2 p_i \tag{3}$$

where C is the set of potential class values and p_i indicates the probability that a random instance in D is of class i. Therefore, the selection of the "best" feature on which to switch can be chosen via determining class probabilities which in turn may be computed via counts. Queries to the parties from the aggregator are therefore counts and class counts, known to have a sensitivity of 1.

Given the ability to query parties for class counts the aggregator may then control the iterative learning process. To ensure this process is differentially private according to a pre-defined privacy budget, we follow the approach from [19] to divide the budget for each iteration and set a fixed number of iterations rather than a purity test as a stopping condition. The algorithm will also stop if counts appear too small relative to the degree of noise to provide meaningful information. The resulting private algorithm deployed in our system is detailed in Algorithm 2.

5.2 Application to Private Neural Network Training

The process of deploying our system for neural network learning is distinct from the process outlined in the previous section for decision tree learning. In central neural network training, after a randomly initialized model of pre-defined structure is created, the following process is used: (1) the dataset D is shuffled and then equally divided into batches, (2) each batch is passed through the model iteratively, (3) a loss function $\mathcal{L}$ is used to compute the error of the model on each batch, (4) errors are then propagated back through the network where an optimizer such as Stochastic Gradient Descent (SGD) is used to update network weights before processing the next batch. Steps (1) through (4) constitute one epoch of learning and are repeated until the model *converges* (stops demonstrating improved performance).

In our system we equate one query to the data parties as one epoch of local learning. That is, each party conducts steps (1) through (4) for one iteration and then sends an updated model to the aggregator. The aggregator then averages the new model weights provided by each party. An updated model is then sent along with a new query for another epoch of learning to each party.

Algorithm 3 Private CNN Learning: Aggregator

Input: Set of data parties $\mathcal{P}$; minimum number of honest, non-colluding parties t; noise parameter σ; learning rate η; sampling probability b; loss function $\mathcal{L}$; clipping value c; number of epochs E; public key pk
$\bar{t} = n - t + 1$
Initialize model M with random weights θ;
for each $e \in [E]$ **do**
  Asynchronously query $\mathcal{P}$:
    train_epoch(M, η, b, $\mathcal{L}$, c, σ, t)
  θ_e = decrypted aggregate, noisy parameters from $\mathcal{P}$
  $M \leftarrow \theta_e$
end for
return M

Each epoch receives the noise parameter σ and cost to the overall privacy budget is determined through a separate privacy accountant utility. Just as the decision tree stopping condition was replaced with a pre-set depth the neural network stopping condition of convergence is replaced with a pre-defined number of epochs E. This process from the aggregator perspective is outlined Algorithm 3.

At each data party we deploy code to support the process detailed in Algorithm 4. To conduct a complete epoch of learning we follow the approach proposed in [1] for private centralized neural network learning. This requires a number of changes to the traditional learning approach. Rather than shuffling the dataset into equal sized batches, a batch is randomly sampled for processing with sampling probability b. An epoch then becomes defined as the number of batch iterations required to process $|D_i|$ instances. Additionally, parameter updates determined through the loss function $\mathcal{L}$ are clipped to define the sensitivity of the neural network learning to individual training instances. Noise is then added to the weight updates. Once an entire epoch is completed the updated weights can be sent back to the aggregator.

Algorithm 4 Private CNN Learning: Data Party P_i

procedure TRAIN_EPOCH(M, η, b, $\mathcal{L}$, c, σ, t)
  θ = parameters of M
  for $j \in \{1, 2, \ldots, \frac{1}{b}\}$ **do**
    Randomly sample $D_{i,j}$ from D_i w/ probability b
    for each $d \in D_{i,j}$ **do**
      $\mathrm{g_j}(d) \leftarrow \nabla_\theta \mathcal{L}(\theta, d)$
      $\bar{\mathrm{g}}_\mathrm{j}(d) \leftarrow \mathrm{g}_j(d) / \max\left(1, \frac{||\mathrm{g}_j(d)||_2}{c}\right)$
    end for
    $\tilde{\mathrm{g}}_\mathrm{j} \leftarrow \frac{1}{|D_{i,j}|}\left(\sum_{\forall d} \bar{\mathrm{g}}_\mathrm{j}(d) + \mathcal{N}\left(0, c^2 \cdot \frac{\sigma^2}{t-1}\right)\right)$
    $\theta \leftarrow \theta - \eta \tilde{\mathrm{g}}_\mathrm{j}$
    $M \leftarrow \theta$
  end for
  return $Enc_{pk}(\theta)$
end procedure

5.3 Application to Private Support Vector Machine Training

Finally, we focus on the classic ℓ_2-regularized binary linear SVM problem with hinge loss, which is given in the following form:

$$\mathcal{L}(w) := \frac{1}{|D|} \sum_{\forall (x_i, y_i) \in D} \max\{0, 1 - y_i \langle w, x_i \rangle\} + \lambda \|w\|_2^2, \quad (4)$$

where $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$ is a feature vector, class label pair, $w \in \mathbb{R}^d$ is the model weight vector, and λ is the regularized coefficient.

From the aggregator perspective, specified in Algorithm 5, the process of SVM training is similar to that of neural network training. Each query to the data parties is defined as K epochs of training. Once query responses are received, model parameters are averaged to generate a new support vector machine model. This new model is then sent to the data parties for another K epochs of training. We again specify a pre-determined number of epochs E to control the number of training iterations.

Algorithm 5 Private SVM Learning: Aggregator

Input: Set of data parties $\mathcal{P}$; minimum number of honest, non-colluding parties t; noise parameter σ; learning rate η; loss function $\mathcal{L}$; clipping value c; number of epochs E; number of epochs per query K, public key pk
$\bar{t} = n - t + 1$
Initialize model M with random weights w;
for each $e \in [E/K]$ **do**
  Asynchronously query $\mathcal{P}$:
    train_epoch(M, η, K, $\mathcal{L}$, c, σ, t)
  θ_e = decrypted aggregate, noisy parameters from $\mathcal{P}$
  $M \leftarrow \theta_e$
end for
return M

To complete an epoch of learning at each data party, we iterate through each instance in the local training dataset D_i. We again deploy a clipping approach to constrain the sensitivity of the updates. The model parameters are then updated according to the loss function $\mathcal{L}$ as well as the noise parameter. The process conducted at each data party for K epochs of training in response to an aggregator query is outlined in Algorithm 6.

5.4 Expanding the Algorithm Repository

Beyond the three models evaluated here, our approach can be used to extend any differentially private machine learning algorithm into a federated learning environment. We demonstrate the flexibility of our system through 3 example algorithms which are of broad interest and significantly different. The task of generating and deploying our system for each algorithm, however, is non-trivial. First, a DP version of the algorithm must be developed. Second, this must be written as a series of queries. Finally, each query must have an appropriate aggregation procedure. Our approach may then be applied for accurate, federated, private results.

Algorithm 6 Private SVM Learning: Data Party P_i

procedure TRAIN_EPOCH(M, η, K, $\mathcal{L}$, c, σ, t)
  w = parameters of M
  for each $(x_i, y_i) \in D$ **do**
    $x_i \leftarrow x_i / \max\left(1, \frac{||x_i||_2}{c}\right)$
  end for
  for $k \in \{1, 2, \ldots, K\}$ **do**
    $\mathrm{g}(D) \leftarrow \nabla_w \mathcal{L}(w, D)$
    $\tilde{\mathrm{g}} \leftarrow \mathrm{g}(D) + \mathcal{N}\left(0, \frac{\sigma^2}{t-1}\right)$
    $w \leftarrow w - \eta \tilde{\mathrm{g}}$
    $M \leftarrow w$
  end for
  return $Enc_{pk}(w)$
end procedure

Due to our choices to use the threshold Paillier cryptosystem in conjunction with an aggregator, rather than a complex SMC protocol run by the parties themselves, we can provide a streamlined interface between the aggregator and the parties. Parties need only answer data queries with encrypted, noisy responses and decryption queries with partial decryption values. Management of the global model and communication with all other parties falls to the aggregator, therefore decreasing the barrier to entry for parties to engage in our federated learning system. Figure 4 demonstrates the impact of this choice as our approach is able to effectively handle the introduction of more parties into the federated learning system without the introduction of increased encryption overhead.

Another issue in the deployment of new machine learning training algorithms is the choice of algorithmic parameters. Key decisions must be made when using our system and many are domain-specific. We aim to inform such decisions with our analysis of trade-offs between privacy, trust and accuracy in Section 4.1.1, but note that the impact will vary depending on the data and the training algorithm chosen. While our system will reduce the amount of noise required to train any federated ML algorithm, questions surrounding what impact various data-specific features will have on the privacy budget are algorithm-specific. For example, Algorithm 2 demonstrates how, in decision tree training, the number of features and classes impact the privacy budget at each level. Similarly, Algorithms 4 and 6 show the role of norm clipping in neural network and SVM learning. In neural networks, this value not only impacts noise but will also have a different impact on learning depending on the size of the network and number of features.

6 RELATED WORK

Our work relates to both the areas of FL as well as privacy-preserving ML. Existing work can be classified into three categories: trusted aggregator, local DP, and cryptographic.

Trusted Aggregator. Approaches in this area trust the aggregator to obtain data in plaintext or add noise. [1] and [22] propose differentially private ML systems, but do not consider a distributed data scenario, thus requiring a central party. In [41], the authors develop

a distributed data mining system with DP but show significant accuracy loss and require a trusted aggregator to add noise.

Recently, [32] presented PATE, an ensemble approach to private learning wherein several "teacher" models are independently trained over local datasets. A trusted aggregator then provides a DP query interface to a "student" model that has unlabelled public data (but no direct access to private data) and obtains labels through queries to the teachers. While we have proposed a federated learning (FL) approach wherein one global model is learned over the aggregate of the parties' datasets, the PATE method develops an ensemble model with independently trained base models using local datasets. Unlike the methods we evaluate, PATE assumes a fully trusted party to aggregate the teachers' labels; focuses on scenarios wherein each party has enough data to train an accurate model, which might not hold, e.g., for cellphone users training a neural network; and assumes access to publicly available data, an assumption not made in our FL system. Models produced from our FL system learn from all available data, leading to more accurate models than the local models trained by each participant in PATE (Figure 4b in [32] demonstrates the need for a lot of parties to achieve reasonable accuracy in such a setting).

Local Differential Privacy. [39] presents a distributed learning system using DP without a central trusted party. However, the DP guarantee is per-parameter and becomes meaningless for models with more than a small number of parameters.

Cryptographic Approaches. [38] presents a protocol to privately aggregate sums over multiple time periods. Their protocol is designed to allow participants to periodically upload encrypted values to an oblivious aggregator with minimum communication costs. Their approach however has participants sending in a stream of statistics and does not address FL or propose an FL system. Additionally, their approach calls for each participant to add noise independently. As our experimental results show, allowing each participant to add noise in this fashion results in models with low accuracy, making this approach is unsuitable for FL. In contrast, our approach reduces the amount of noise injected by each participant by taking advantage of the additive properties of DP and the use of threshold-based homomorphic encryption to produce accurate models that protect individual parties' privacy.

In [6, §B] the authors propose the use of multiparty computation to securely aggregate data for FL. The focus of the paper is to present suitable cryptographic techniques to ensure that the aggregation process can take place in mobile environments. While the authors propose FL as motivation, no complete system is developed with "a detailed study of the integration of differential privacy, secure aggregation, and deep learning" remaining beyond the scope.

[4] provides a theoretical analysis on how differentially private computations could be done in a federated setting for single instance operations using either secure function evaluation or the local model with a semi-trusted curator. By comparison, we consider multiple operations to conduct FL and provide empirical evaluation of the FL system. [29] proposes a system to perform differentially private database joins. This approach combines private set intersection with random padding, but cannot be generally applied to FL. In [33] the authors' protocols are tailored to inner join tables and counting the number of values in an array. In contrast, we propose an accurate, private FL system for predictive model training.

Dwork et al. [14] present a distributed noise generation scheme and focus on methods for generating noise from different distributions. This scheme is based on secret sharing, an MPC mechanism that requires extensive exchange of messages and entails a communication overhead not viable in many federated learning settings.

[10] proposes a method to train neural networks in a private collaborative fashion by combining MPC, DP and secret sharing assuming non-colluding honest parties. In contrast, our system prevents privacy leakages even if parties actively collude.

Approaches for the private collection of streaming data, including [2, 8, 21, 37], aim to recover computation when one or more parties go down. Our system, however, enables private federated learning which allows for checkpoints in each epoch of training. The use of threshold cryptography also enables our system to decrypt values when only a subset of the participants is available.

7 CONCLUSION

In this paper, we present a novel approach to perform FL that combines DP and SMC to improve model accuracy while preserving provable privacy guarantees and protecting against extraction attacks and collusion threats. Our approach can be applied to train different ML models in a federated learning fashion for varying trust scenarios. Through adherence to the DP framework we are able to guarantee overall privacy from inference of any model output from our system as well as any intermediate result made available to $\mathcal{A}$ or $\mathcal{P}$. SMC additionally guarantees any messages exchanged without DP protection are not revealed and therefore do not leak any private information. This provides end-to-end privacy guarantees with respect to the participants as well as any attackers of the model itself. Given these guarantees, models produced by our system can be safely deployed to production without infringing on privacy guarantees.

We demonstrated how to apply our approach to train a variety of ML models and showed that it out-performs existing state-of-the art techniques for FL. Our system provides significant gains in accuracy when compared to a naïve application of state-of-the-art differentially private protocols to FL systems.

For a tailored threat model, we propose an end-to-end private federated learning system which uses SMC in combination with DP to produce models with high accuracy. As far as we know this is the first paper to demonstrate that the application of these combined techniques allow us to maintain this high accuracy at a given level of privacy over different learning approaches. In the light of the ongoing social discussion on privacy, this proposed approach provides a novel method for organizations to use ML in applications requiring high model performance while addressing privacy needs and regulatory compliance.

REFERENCES

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

[2] Gergely Ács and Claude Castelluccia. 2011. I have a dream!(differentially private smart metering). In *International Workshop on Information Hiding*. Springer, 118–132.

[3] Donald Beaver. 1991. Foundations of secure interactive computing. In *Annual International Cryptology Conference*. Springer, 377–391.

[4] Amos Beimel, Kobbi Nissim, and Eran Omri. 2008. Distributed Private Data Analysis: Simultaneously Solving How and What. In *Advances in Cryptology – CRYPTO 2008*, David Wagner (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 451–468.

[5] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. 2005. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 128–138.

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1175–1191.

[7] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*. Springer, 635–658.

[8] T. H. Hubert Chan, Elaine Shi, and Dawn Song. 2012. Privacy-Preserving Stream Aggregation with Fault Tolerance. In *Financial Cryptography and Data Security*, Angelos D. Keromytis (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 200–214.

[9] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2 (2011), 27:1–27:27. Issue 3. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[10] Melissa Chase, Ran Gilad-Bachrach, Kim Laine, Kristin E Lauter, and Peter Rindal. 2017. Private Collaborative Neural Network Learning. *IACR Cryptology ePrint Archive* 2017 (2017), 762.

[11] Ivan Damgård and Mats Jurik. 2001. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography (PKC '01)*. Springer-Verlag, London, UK, UK, 119–136. http://dl.acm.org/citation.cfm?id=648118.746742

[12] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI Machine Learning Repository. (2017). http://archive.ics.uci.edu/ml

[13] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*. Springer, 1–19.

[14] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 486–503.

[15] Cynthia Dwork and Jing Lei. 2009. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM, 371–380.

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

[17] Cynthia Dwork and Guy N Rothblum. 2016. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887* (2016).

[18] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 51–60.

[19] Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 493–502.

[20] Oded Goldreich. 1998. Secure multi-party computation. *Manuscript. Preliminary version* 78 (1998).

[21] S. Goryczka and L. Xiong. 2017. A Comprehensive Comparison of Multiparty Secure Additions with Differential Privacy. *IEEE Transactions on Dependable and Secure Computing* 14, 5 (Sep. 2017), 463–477. https://doi.org/10.1109/TDSC.2015.2484326

[22] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N Wright. 2009. A practical differentially private random decision tree classifier. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*. IEEE, 114–121.

[23] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2017. The composition theorem for differential privacy. *IEEE Transactions on Information Theory* 63, 6 (2017), 4037–4049.

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

[25] Jaewoo Lee and Daniel Kifer. 2018. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1656–1665.

[26] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server.. In *OSDI*, Vol. 14. 583–598.

[27] Yehuda Lindell and Benny Pinkas. 2000. Privacy preserving data mining. In *Annual International Cryptology Conference*. Springer, 36–54.

[28] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. 2016. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research* 17, 1 (2016), 1235–1241.

[29] Arjun Narayan and Andreas Haeberlen. 2012. DJoin: Differentially Private Join Queries over Distributed Databases.. In *OSDI*. 149–162.

[30] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Stand-alone and Federated Learning under Passive and Active White-box Inference Attacks. In *Security and Privacy (SP), 2019 IEEE Symposium on*.

[31] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 223–238.

[32] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. *arXiv preprint arXiv:1802.08908* (2018).

[33] Martin Pettai and Peeter Laud. 2015. Combining differential privacy and secure multiparty computation. In *Proceedings of the 31st Annual Computer Security Applications Conference*. ACM, 421–430.

[34] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106.

[35] J Ross Quinlan. 1993. C4. 5: Programming for machine learning. *Morgan Kauffmann* 38 (1993), 48.

[36] J. Ross Quinlan. 2007. C5. (2007). http://rulequest.com

[37] Vibhor Rastogi and Suman Nath. 2010. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 735–746.

[38] Elaine Shi, HTH Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. 2011. Privacy-preserving aggregation of time-series data. In *Annual Network & Distributed System Security Symposium (NDSS)*. Internet Society.

[39] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 1310–1321.

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

[41] Ning Zhang, Ming Li, and Wenjing Lou. 2011. Distributed data mining with differential privacy. In *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 1–5.