A Hybrid Approach to Privacy-Preserving Federated Learning

Stacey Truex[1*] staceytruex@gatech.edu
Nathalie Baracaldo[2] baracald@us.ibm.com
Ali Anwar[2] Ali.Anwar2@ibm.com
Thomas Steinke[2] Thomas.Steinke@ibm.com
Heiko Ludwig[2] hludwig@us.ibm.com
Rui Zhang[2] ruiz@us.ibm.com

[1]Georgia Institute of Technology, [2]IBM Research - Almaden

Abstract

Training machine learning models often requires data from multiple parties. However, in some cases, data owners cannot share their data due to legal or privacy constraints but would still benefit from training a model jointly with multiple parties. Federated learning has arisen as an alternative to allow for the collaborative training of models without the sharing of raw data. However, attacks in the literature have demonstrated that simply maintaining data locally during training processes does not provide strong enough privacy guarantees. We need a federated learning system capable of preventing inference over the messages exchanged between parties during training as well as the final, trained model, considering potential collusion between parties, and ensuring the resulting machine learning model has acceptable predictive accuracy. Currently, existing approaches are either vulnerable to inference or do not scale for a large number of parties, resulting in models with low accuracy. To close this gap, we present a scalable approach that protects against these threats while producing models with high accuracy. Our approach provides formal data privacy guarantees using both differential privacy and secure multiparty computation frameworks. We validate our system with experimental results on two popular and significantly different machine learning algorithms: decision trees and convolutional neural networks. To the best of our knowledge, this presents the first approach to accurately train a neural network in a private, federated fashion. Our experiments demonstrate that our approach outperforms state of the art solutions in accuracy, customizability, and scalability.

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, the popular distributed ML library MLlib from Apache Spark assumes a trusted central node to coordinate distributed learning processes (Meng et al. 2016). Another popular approach is the parameter server (Li et al. 2014), which again requires a fully trusted central node to collect and aggregate parameters from the many nodes learning on their different datasets.

[*]Work done while interning with IBM Research - Almaden.

However, many distributed learning scenarios cannot rely on the absence of trust boundaries, particularly when multiple organizations are involved. In many cases organizations cannot share data, e.g. due to legal restrictions or competition between participants. The area of federated learning (FL) is a response to these more restrictive environments wherein the data holders must be engaged throughout the learning process rather than relying on a trusted third party (Shokri and Shmatikov 2015; Bonawitz et al. 2017).

For example, consider three hospitals H_1, H_2, and H_3 each serving the same city C. Rather than each hospital creating their own predictive model forecasting cancer risks for their patients, the hospitals want to create a model which is learned over the aggregate of data on patients in city C. Similarly we can consider a single service provider S. S collects data from clients both in Europe and the United States. Due to different legislation in the different countries, S does not store data in one central location but rather has two different storage locations. When creating a predictive model forecasting the usage of their services however, S will want to make use of both sets of data.

Despite the increased accuracy that more data provides, data privacy remains a hurdle to the adoption of FL systems in many domains. In addition to eliminating the trusted third party, many distributed learning scenarios must also be concerned with extraction attacks (Shokri et al. 2017) where adversaries have been shown to extract sensitive data used in model training. Even when considering domains without explicit privacy legislation, a demand for data privacy is likely as federation participants may often be competitors who do not want to release data to one another. Thus, any realistic FL framework must consider risks to data privacy.

In our system, data never leaves the participants while privacy is guaranteed using secure multiparty computation 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 which provides formal privacy guarantees and produces models with increased accuracy compared to existing approaches.
- We include a tunable trust parameter which accounts for various trust scenarios while maintaining the increased

accuracy and formal privacy guarantees.

- We provide experimental evaluation of our system with two significantly different ML models: decision trees and convolutional neural networks.
- 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. This is followed by experimental evaluation of our system. Finally, we give an overview of related work and some concluding remarks.

Preliminaries

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 (Shokri et al. 2017) have shown that with access to only a drained 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:

Definition 1 (Differential Privacy (Dwork 2008))**.** *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, $\forall S \subseteq Range(\mathcal{K})$,*

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

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

To achieve DP, noise is added to the algorithm's output. This noise is proportional to the sensitivity of the output. 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

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

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 $\delta \geq \frac{4}{5} exp(-(\sigma\epsilon)/2)$ and $\epsilon < 1$ (Dwork, Roth, and others 2014).

To achieve ϵ-differential privacy, the Laplace mechanism may be used in the same manner by simply substituting $N(0, S_f^2\sigma^2)$ with random variables drawn from $Lap(S_f/\epsilon)$ (Dwork, Roth, and others 2014).

When an algorithm requires the use of multiple additive noise mechanisms, the evaluation of the privacy guarantee follows from the basic composition theorem (Dwork et al. 2006; Dwork and Lei 2009) or from advanced composition theorems and their extensions (Dwork, Rothblum, and Vadhan 2010; Dwork and Rothblum 2016; Kairouz, Oh, and Viswanath 2017; Bun and Steinke 2016).

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 type of additive homomorphic scheme is the Paillier cryptosystem (Paillier 1999), a probabilistic encryption scheme based on computations in the group Z_{n^2}, where n is an RSA modulus. In (Damgård and Jurik 2001) 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 set of parties $\mathcal{P}$, where $|\mathcal{P}|$ is less than a pre-defined threshold, is able to decrypt values.

Privacy in Federated Learning

In centralized learning environments a single party P using a dataset D executes some learning algorithm f_M. By contrast, in FL 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 final predictive model. *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 model* refers to the leakage of any participants' data solely from 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.

Privacy of Computation 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 corresponding 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. Any privacy-preserving FL system must account for the risk of inference over these responses.

Privacy-preserving ML approaches addressing privacy of computation 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 (Goldreich 1998). However, as the function output remains unchanged from function execution without privacy, information about individual inputs can still be revealed.

Privacy of Output Therefore, we also consider privacy of outputs. This refers to the outputs of any intermediate calculations made available to participants as well as the

predictive model. Recent work has shown that given only black-box access to the model through an ML as a service API, an attacker can still make inferences of the training data (Shokri et al. 2017). Not only should an FL system prevent such outsider attacks, but there should also be a consideration for insiders. That is, participant P_i should not be able to infer information about D_j when $i \neq j$.

Solutions addressing privacy of output often make use of the DP framework discussed in Preliminaries.

An End-to-End Approach with Trust

Threat Model

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

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 (Goldreich 2009) and application to data mining in (Lindell and Pinkas 2000). 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 it receives.

Colluding Parties Our work additionally considers the threat of collusion among parties, including collusion with the aggregator. Our system includes a trust parameter t which indicates a minimum number of non-colluding parties. Additionally, in contrast to the aggregator, parties in $\mathcal{P}$ may deviate from the protocol execution.

Outsiders We also consider potential attacks from adversaries outside of the system. Our work ensures that any adversary monitoring communication between the parties 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 be deployed as a service, remaining resilient to inference.

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

Communication and system set up We assume secure channels between each party and the aggregator. This allows the aggregator to authenticate incoming messages, preventing an adversary from injecting their own responses.

We additionally make use of the *threshold variant of the Paillier encryption scheme* from (Damgård and Jurik 2001) assuming secure key distribution. Our use of the threshold variant of the Paillier system ensures that any set of $n - t$ or fewer parties cannot decrypt ciphertexts. This ensures the privacy of individual messages sent to the aggregator.

Our Approach

We propose an FL system that addresses privacy of computation, privacy of outputs, *and* trust. We combine methods from SMC and DP to develop protocols that guarantee privacy without sacrificing accuracy.

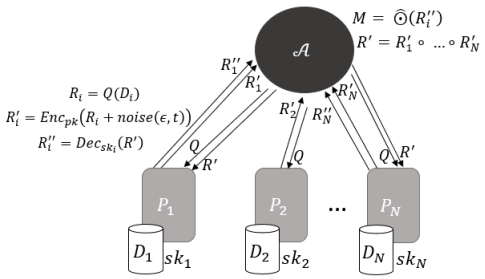

Figure 1: System Overview

Algorithm 1 Private Federated Learning

Input: ML algorithm f_M; set of data parties $\mathcal{P}$, 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 $Enc_{pk}(r_s) \leftarrow r_{1,s} \circ r_{2,s} \circ ... \circ r_{n,s}$
  $\mathcal{A}$ 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

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 algorithm for training the target model M, ϵ is the privacy guarantee against inference, and t specifies the minimum number of honest, non-colluding parties.

An outline of our approach is shown in Figure 1. The aggregator $\mathcal{A}$ is tasked with running 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. Using the secure channel between $\mathcal{A}$ and each of the parties, the aggregator will send query Q_s when it reaches the corresponding step s in f_M. Each participant will then calculate a response using their respective datasets. Before sending the response back, the participant will 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 response is then encrypted and sent to $\mathcal{A}$. Homomorphic properties then allow $\mathcal{A}$ to aggregate the individual responses. $\mathcal{A}$ then queries at least $n-t+1$ data parties to decrypt the aggregate value. Eventually, a model M is produced and 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.

Our use of homomorphic encryption allows for significant increases in accuracy (over local privacy approaches). We now detail this strategy for FL.

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, given an overall privacy budget ϵ, δ, let σ_s and S_s respectively be noise parameter and sensitivity to step Q_s. Using the Gaussian mechanism, each party P_i will then add $N(0, S_s^2\sigma_s^2)$ 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 (Damgård and Jurik 2001) 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 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.

The maximum number of colluders, $\bar{t}$, cannot decrypt a single response sent from an honest P_i to $\mathcal{A}$. Additionally, if each colluder sends a zero value to $\mathcal{A}$, then our noise inclusion becomes $N(0, S_s^2 \frac{t\sigma_s^2}{t-1}) > N(0, S_s^2\sigma_s^2)$, guaranteeing the privacy of the aggregate result.

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.

Experimental Evaluation

In this section we empirically evaluate our system using two distinct learning algorithms: decision trees (DT) and convolutional neural networks (CNN). We additionally provide analysis on the impact of certain settings on performance.

Decision Trees

We first consider DT learning using the ID3 algorithm, introduced in (Quinlan 1986). 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 outlined in Algorithm 2.

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
$\bar{t} = n - t + 1$
$\epsilon_1 = \frac{\epsilon}{2(d+1)}$
M = BuildTree($\emptyset$, $\mathcal{P}$, t, ϵ_1, $\mathcal{F}$, C, d, pk) **return** M
procedure BUILDTREE(M, $\mathcal{S}$, $\mathcal{P}$, 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**
        $S_i = S + \{F = f_i\}$
        Asynchronously query $\mathcal{P}$: *counts*($\mathcal{S}_i$, ϵ_2, t)
        and *class_counts*($\mathcal{S}_i$, ϵ_2, t)
        ${N'}_i^F$ = aggregate of *counts*
        ${N'}_{i,c}^F$ = element-wise aggregate of *class_counts*
        Recover N_i^F from $\bar{t}$ partial decryptions of ${N'}_i^F$
        Recover $N_{i,c}^F$ from $\bar{t}$ partial decryptions of ${N'}_{i,c}^F$
      end for
      $V_F = \sum_{i=1}^{|F|} \sum_{c=1}^{|C|} N_{i,c}^F \cdot \log \frac{N_{i,c}^F}{N_i^F}$
    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, pk)
      Set M_i as child of M with edge f_i
    end for
    return M
  end if
end procedure

There are two types of participant queries in Algorithm 2, *counts* and *class_counts*. For executing these queries $\mathcal{A}$ first divides the entire privacy budget ϵ equally between each layer of the tree including the leaf nodes. 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 does not need to be further divided. Within each node, half of the budget is allocated to determining total counts while half is allocated to class counts (ϵ_1). For internal nodes, however, each feature is evaluated for potential splitting and the budget must 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 (Dheeru and Karra Taniskidou 2017). This dataset contains 8 categorical attributes about 12,960

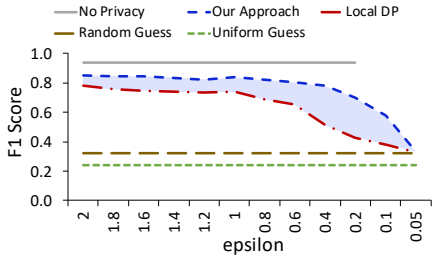

Figure 2: Effect of privacy budgets on the overall F1-score for Decision Trees

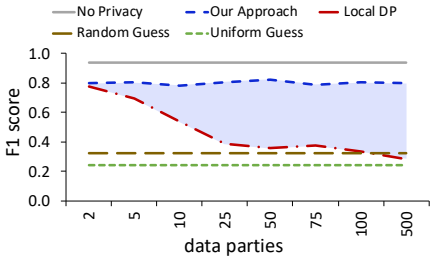

Figure 3: Effect of increasing number of parties on the overall F1-score for Decision Trees

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. Our four baselines are:

1. Uniform Guess. In this approach, class predictions are randomly sampled with a $\frac{1}{|C|}$ chance for each class.
2. Random Guess. The random guess is an improvement on Uniform Guess as it considers the distribution of the class values in the training data. That is, at test time, each class prediction is sampled from the set of training class labels.
3. Local DP. In the local approach, each party adds enough 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 (Blum et al. 2005).
4. No Privacy. This is the result of running the distributed learning algorithm without any privacy guarantee.

Variation in Setting

We now look at how different settings impact results.

Privacy Budget We first look at the impact of the privacy budget on the performance of 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 propostional to value of ϵ. In other words, the smaller the ϵ value, the smaller the privacy budget, and the more noise which will be 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) causes learning to end or (2) causes inaccurate learning. This causes the performance to degrade 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 budget.

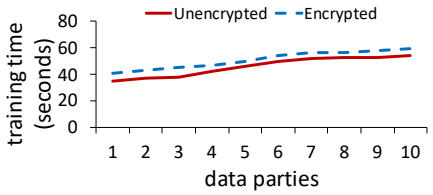

Figure 4: Decision Tree Training Time with Encryption

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, are contributing to the learning. This is particularly relevant in IoT scenarios.

In Figure 3 we demonstrate the impact that the size of $\mathcal{P}$ has on performance. The results of Figure 3 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.

As $|\mathcal{P}|$ increases, the noise in the local DP approach increases proportionally while our approach maintains consistent accuracy. 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. 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.

Another consideration relative to the scaling of participants is the overhead of encryption. Figure 4 shows the impact that encryption has on overall training time in our system as the number of parties increases from 1 to 10. Because our system is designed for a distributed scenario, the overhead of encryption remains constant as the number of parties increases. Each party is able to encrypt query responses (and decrypt aggregations) in parallel due to this distributed setup. Our system therefore generally scales well.

Trust An important part of our system is the inclusion of a trust parameter. Figure 5 demonstrates how the ϵ values used for both count and distribution queries in Algorithm 2 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

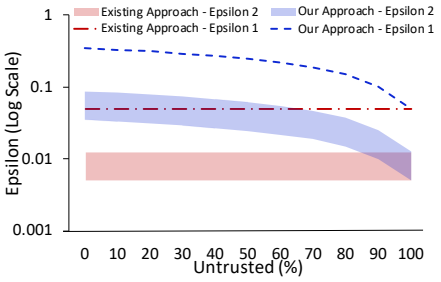

Figure 5: Query Epsilons in Decision Tree Training with Varying Rate of Trust (50 parties). Epsilon 1 and 2 are defined in Algorithm 2.

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

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.

Convolutional Neural Networks

We additionally developed distributed differentially private CNN learning using similar principles. 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 presented in (Abadi et al. 2016). 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 pre-determined E number of epochs, the final model M is output. This process for the aggregator and one data party are outlined in Algorithms 3 and 4, respectively.

When using Algorithm 4, if $\sigma = \sqrt{2 \cdot \log \frac{1.25}{\delta}}/\epsilon$ then by (Dwork, Roth, and others 2014) Algorithm 4 is (ϵ, δ)-differentially private with respect to each randomly sampled batch. Using the moments accountant in (Abadi et al. 2016), Algorithm 4 is shown to be $(O(b\epsilon E\frac{1}{b}), \delta)$-DP overall.

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

Dataset 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 (LeCun et al. 1998).

Model Structure We use a model structure similar to that in (Abadi et al. 2016). 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 use a norm clipping of 4.0, a learning rate of 0.1 and a batch rate of 0.01. We use the Keras with a Tensorflow backend.

Comparison Methods To the best of our knowledge, this paper presents first approach to accurately train a neural network in a private federated fashion. For this reason, we 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 (Abadi et al. 2016).
3. Distributed Data, 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 enough noise to protect the privacy of their own data in isolation, adapting from (Abadi et al. 2016) and (Shokri and Shmatikov 2015).

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 (Abadi et al. 2016). 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 outperforms 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.

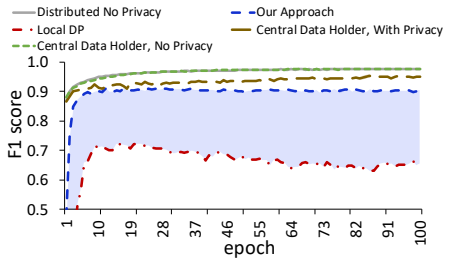

Figure 6: Convolutional Neural Network Training with MNIST Data (10 parties and $\sigma = 8$)

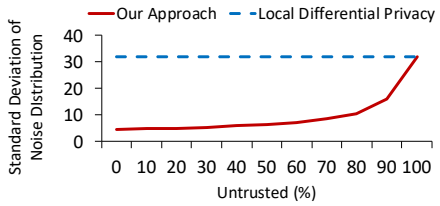

Figure 7: Degree of Noise in Convolutional Neural Network Training with Varying Rate of Trust

We additionally experiment with $\sigma = 4$ and $\sigma = 2$ as was done in (Abadi et al. 2016). When $\sigma = 4$ 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. When σ is set to 2 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 the overhead remains relatively constant as both $|\mathcal{P}|$ and the number of model 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.

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 to add noise. In (Abadi et al. 2016) and (Jagannathan, Pillaipakkamnatt, and Wright 2009) the authors propose differentially private ML systems, however the aggregator is trusted to see the training data. In (Zhang, Li, and Lou 2011), the authors develop a distributed data mining system with DP but show significant accuracy loss and require a trusted aggregator to add noise.

Recently, (Papernot et al. 2018) 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. Unlike the methods we evaluate, PATE assumes a fully trusted party to aggregate the teachers' labels and also 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.

Local Differential Privacy: (Shokri and Shmatikov 2015) present a system for distributed learning using DP without a central trusted party. The authors, however, provide a DP guarantee per-parameter which becomes meaningless for models with more than a small number of parameters.

Cryptographic approaches: (Shi et al. 2011) present 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.

In (Bonawitz et al. 2017, §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 federated learning as a motivation for their work, no complete system is developed with "a detailed study of the integration of differential privacy, secure aggregation, and deep learning" remaining beyond the scope.

(Beimel, Nissim, and Omri 2008) provide a theoretical analysis on how differentially private computations could be done in a federated setting for single instance operations. By comparison, we consider multiple operations to conduct FL and provide empirical evaluation of the FL system. (Narayan and Haeberlen 2012) 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 (Pettai and Laud 2015) 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.

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, protecting against extraction attacks and collusion threats. Through adherence to the DP framework we are able to guarantee the 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 allows us to guarantee any messages exchanged without DP protection are not revealed and therefore do not leak any private information. This allows us to provide end-to-end privacy guarantees with respect to the participants as well as any attackers of the model itself. Given these guarantees, models produced from our system can be safely deployed in a ML as a service environment.

We assess the proposed approach extensively and show that it scales well while producing ML models with higher accuracy than existing approaches. If an FL system cannot

both protect privacy and maintain accuracy it is unlikely to be adopted. We demonstrate that our system provides significant gains in accuracy when compared to a naïve application of state-of-the-art differentially private protocols to FL systems. We similarly show that our approach consistently out-performs baselines such as random guessing while remaining reasonably close to non-private settings.

References

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

[Beimel, Nissim, and Omri 2008] Beimel, A.; Nissim, K.; and Omri, E. 2008. Distributed private data analysis: Simultaneously solving how and what. In Wagner, D., ed., *Advances in Cryptology – CRYPTO 2008*, 451–468. Berlin, Heidelberg: Springer Berlin Heidelberg.

[Blum et al. 2005] Blum, A.; Dwork, C.; McSherry, F.; and Nissim, K. 2005. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 128–138. ACM.

[Bonawitz et al. 2017] Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H. B.; Patel, S.; Ramage, D.; Segal, A.; and Seth, K. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1175–1191. ACM.

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

[Damgård and Jurik 2001] Damgård, I., and Jurik, M. 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, 119–136. London, UK, UK: Springer-Verlag.

[Dheeru and Karra Taniskidou 2017] Dheeru, D., and Karra Taniskidou, E. 2017. UCI machine learning repository.

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

[Dwork and Rothblum 2016] Dwork, C., and Rothblum, G. N. 2016. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*.

[Dwork et al. 2006] Dwork, C.; Kenthapadi, K.; McSherry, F.; Mironov, I.; and Naor, M. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 486–503. Springer.

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

[Dwork, Rothblum, and Vadhan 2010] Dwork, C.; Rothblum, G. N.; and Vadhan, S. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 51–60. IEEE.

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

[Goldreich 1998] Goldreich, O. 1998. Secure multi-party computation. *Manuscript. Preliminary version* 78.

[Goldreich 2009] Goldreich, O. 2009. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press.

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

[Kairouz, Oh, and Viswanath 2017] Kairouz, P.; Oh, S.; and Viswanath, P. 2017. The composition theorem for differential privacy. *IEEE Transactions on Information Theory* 63(6):4037–4049.

[LeCun et al. 1998] LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

[Li et al. 2014] Li, M.; Andersen, D. G.; Park, J. W.; Smola, A. J.; Ahmed, A.; Josifovski, V.; Long, J.; Shekita, E. J.; and Su, B.-Y. 2014. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, 583–598.

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

[Meng et al. 2016] Meng, X.; Bradley, J.; Yavuz, B.; Sparks, E.; Venkataraman, S.; Liu, D.; Freeman, J.; Tsai, D.; Amde, M.; Owen, S.; et al. 2016. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research* 17(1):1235–1241.

[Narayan and Haeberlen 2012] Narayan, A., and Haeberlen, A. 2012. Djoin: Differentially private join queries over distributed databases. In *OSDI*, 149–162.

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

[Papernot et al. 2018] Papernot, N.; Song, S.; Mironov, I.; Raghunathan, A.; Talwar, K.; and Erlingsson, Ú. 2018. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*.

[Pettai and Laud 2015] Pettai, M., and Laud, P. 2015. Combining differential privacy and secure multiparty computa-

tion. In *Proceedings of the 31st Annual Computer Security Applications Conference*, 421–430. ACM.

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

[Shi et al. 2011] Shi, E.; Chan, H.; Rieffel, E.; Chow, R.; and Song, D. 2011. Privacy-preserving aggregation of time-series data. In *Annual Network & Distributed System Security Symposium (NDSS)*. Internet Society.

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

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

[Zhang, Li, and Lou 2011] Zhang, N.; Li, M.; and Lou, W. 2011. Distributed data mining with differential privacy. In *Communications (ICC), 2011 IEEE International Conference on*, 1–5. IEEE.