A Secure Sum Protocol and Its Application to Privacy-preserving Multi-party Analytics

Shagufta Mehnaz
Dept. of Computer Science,
Purdue University
West Lafayette, IN, USA
smehnaz@purdue.edu

Gowtham Bellala*
C3 IoT
Redwood City, CA, USA
gowtham.bellala@c3iot.com

Elisa Bertino
Dept. of Computer Science,
Purdue University
West Lafayette, IN, USA
bertino@purdue.edu

ABSTRACT

Many enterprises are transitioning towards data-driven business processes. There are numerous situations where multiple parties would like to share data towards a common goal if it were possible to simultaneously protect the privacy and security of the individuals and organizations described in the data. Existing solutions for multi-party analytics that follow the so called Data Lake paradigm have parties transfer their raw data to a trusted third-party (i.e., mediator), which then performs the desired analysis on the global data, and shares the results with the parties. However, such a solution does not fit many applications such as Healthcare, Finance, and the Internet-of-Things, where privacy is a strong concern. Motivated by the increasing demands for data privacy, we study the problem of privacy-preserving multi-party data analytics, where the goal is to enable analytics on multi-party data without compromising the data privacy of each individual party. In this paper, we first propose a secure sum protocol with strong security guarantees. The proposed secure sum protocol is resistant to collusion attacks even with $N-2$ parties colluding, where N denotes the total number of collaborating parties. We then use this protocol to propose two secure gradient descent algorithms, one for horizontally partitioned data, and the other for vertically partitioned data. The proposed framework is generic and applies to a wide class of machine learning problems. We demonstrate our solution for two popular use-cases, regression and classification, and evaluate the performance of the proposed solution in terms of the obtained model accuracy, latency and communication cost. In addition, we perform a scalability analysis to evaluate the performance of the proposed solution as the data size and the number of parties increase.

ACM Reference format:
Shagufta Mehnaz, Gowtham Bellala, and Elisa Bertino. 2017. A Secure Sum Protocol and Its Application to Privacy-preserving Multi-party Analytics. In *Proceedings of SACMAT'17, June 21–23, 2017, Indianapolis, IN, USA, ,* 12 pages.
DOI: http://dx.doi.org/10.1145/3078861.3078869

*This work was done while the author was at Hewlett Packard Labs.

SACMAT'17, June 21–23, 2017, Indianapolis, IN, USA
 ACM ISBN 978-1-4503-4702-0/17/06…$15.00.
DOI: http://dx.doi.org/10.1145/3078861.3078869

1 INTRODUCTION

An important challenge for today's data-driven enterprises is how to extract information from a dataset that can facilitate good business decisions, without sacrificing the privacy of the individuals or organizations whose sensitive details may be contained in the dataset. This challenge is compounded when the analysis involves multiple organizations (or parties) wishing to collaborate, in order to obtain a broader understanding of a topic of mutual interest.

As a motivating example, consider a scenario where a group of hospitals wish to collaborate to improve their collective quality of healthcare. Each hospital already collects a lot of data about its patients, including their demographics, past medical history, lab results, current diagnosis, and prescribed treatment and outcomes. This data contains a wealth of information that if shared across the group could mutually benefit all parties by enabling faster diagnosis and effective treatment for similar cases. However, this data contains extremely sensitive and private information both about the patients and the hospitals. Thus, for a variety of reasons (including regulatory[1]), sharing this sort of data can be problematic. This hinders the desire of the organizations to become more data-driven.

This general class of problem arises when multiple parties, each owning a privacy-sensitive dataset, would like to collectively perform analytics on the union of the datasets while respecting the privacy and security concerns of each individual party. Current commercial solutions require each party to share their raw data or local aggregates with a trusted third party or a mediator under an appropriate confidential agreement, and have the mediator compute and share the results. However, for reasons such as the recent cyber attacks and resulting massive privacy breaches, many organizations are understandably hesitant to share either raw or aggregate data with third parties [1, 2].

In this paper, we introduce a generalized framework that enables training of machine learning models on multi-party, distributed data in a privacy-preserving manner. **The proposed framework assumes an *untrusted mediator*. Under our framework, each party shares only encrypted local results, and then the mediator aggregates these results to obtain the global result**. The encrypted local results are also anonymized so that the mediator cannot associate a local result with the party to which the local result belongs. The proposed framework is generic and applies to several machine learning algorithms.

We first propose a secure sum protocol that serves as a key building block in this framework. While secure sum protocols for

[1] https://www.congress.gov/bill/105th-congress/senate-bill/1368

multiple parties have been widely investigated and applied [3, 4], they are not secure when two or more parties collude. In contrast, our proposed protocol is secure under the honest-but-curious model, and to collusion attacks. We show theoretically that the proposed secure sum protocol is secure with up to $N-2$ parties colluding, where N denotes the total number of parties collaborating in the analysis. We then use the proposed secure sum protocol to develop two secure gradient descent algorithms, one for horizontally partitioned data and the other for vertically partitioned data. This secure gradient descent solution applies to a broad class of machine learning problems.

The proposed solution has several advantages over existing methods for privacy-preserving multi-party analytics, as described below. Most of the existing approaches for Secure Multi-party Computation (SMC) are either theoretical or based on complicated techniques, such as Yao's garbled circuits [5] and oblivious transfer, that do not scale to large datasets. In contrast, our proposed approach uses simple crypto protocols that are practical, efficient, and scale to large datasets while providing strong security guarantees. In addition, most existing SMC based approaches avoid the use of a mediator and rely on peer-to-peer communication. As a result, the communication cost and latency overhead increase significantly (sometimes exponentially) as the number of parties increase. In contrast, our approach can be easily extended to multiple parties. Finally, most existing approaches for privacy preserving data mining and SMC propose secure protocols that are designed to support a specific analytic functionality. For example, a secure protocol, known as additive data perturbation [6], is designed only to support secure clustering and does not apply to other analytic functionality. In contrast, our proposed framework is generic and applies to a wide range of analytic functionality including regression, classification, and clustering. Below are the main contributions of this paper.

- We propose a secure sum protocol that is secure under collusion attacks (even with $N-2$ parties colluding).
- We design two secure gradient descent algorithms, one for horizontally partitioned data, and the other for vertically partitioned data using the proposed secure sum protocol. These algorithms enable privacy-preserving multi-party analytics. Moreover, they are generic and apply to a wide class of machine learning problems such as regression, classification and clustering.
- We have implemented and tested the proposed solution for two analytic use-cases, linear regression and binary classification, using both real-world and synthetic datasets. We evaluate the performance of the proposed approach in terms of accuracy of the resulting model, communication cost, and latency, and compare it with two baseline approaches that use a trusted mediator. We also perform a scalability analysis to determine the performance of the proposed approach as the size of the data, and the number of parties are increased.

2 RELATED WORK

Secure sum protocols are a key building block in privacy-preserving data mining and have been widely studied in the literature. Clifton et al. [3] proposed a round-robin based secure sum protocol under the honest-but-curious model that is based on randomization. However, this solution is not secure when the parties collude. To overcome this limitation, Sheikh et al. [4] proposed an extension that is secure under two colluding parties. However, such a protocol is not secure when more than two parties collude. In contrast, our proposed protocol is secure under the honest-but-curious model with at most $N-2$ colluding parties.

The problem of privacy-preserving, multi-party analytics has been extensively investigated in the areas of privacy-preserving data mining (PPDM) [7] and secure multi-party computation (SMC). Privacy-preserving algorithms for data mining tasks, such as clustering, classification and association rule mining, have been proposed for both horizontally partitioned datasets [8–11] and vertically partitioned datasets [8, 12, 13]. However, these approaches have several drawbacks: they are designed for a specific machine learning algorithm and do not easily generalize to others; they do not scale well to large datasets or multiple parties; they do not provide strong security guarantees. Below, we provide a brief overview of existing approaches broadly classified into three categories.

(1) **Anonymization-based strategies [8]:** These approaches partition attributes in a given database table into two sets – those containing Personally Identifiable Information (PII) and the rest that do not, and remove the set of PII (e.g., 'Name', 'Social Security Number'). Attributes such as "Zip code", "Age", "Gender", etc. which can identify an individual when used in combination are anonymized using techniques such as k-anonymity [14], l-diversity, etc. However, numerous studies have shown that distinction of an attribute into "identifying" and "non-identifying" is difficult and non-trivial as this distinction may change depending on what prior or external information is available to the attacker [15]. One famous example of de-anonymization attack is the Netflix fiasco, where attackers were able to de-anonymize the Netflix dataset based on the published movie ratings, and externally available datasets. Similar attacks have been demonstrated on several anonymized databases [16].
(2) **Secure Multi-party Computation (SMC)-based strategies [9]:** These approaches typically consider the entire data (all attributes) as private, and use cryptographic protocols such as homomorphic encryption, Yao's garbled circuits, etc. Most SMC-based strategies rely on peer-to-peer communication and are usually defined in 2-party scenarios, with extension to multi-party scenarios often resulting in significant communication overhead. While SMC-based strategies provide strong security guarantees, they are based on complicated techniques that are slow and do not scale for large datasets. Similarly, while the recent results on fully homomorphic encryption are promising, they are far from practical [17].
(3) **Randomization-based strategies [8]:** These approaches are based on randomization techniques, such as additive data perturbation and random subspace projection, that masks the underlying data while preserving the statistical properties of the overall dataset [18]. While these approaches are fast and efficient, they do not provide strong security guarantees, and are often susceptible to attacks [19].

Our proposed solution belongs to the second category. It is secure under the honest-but-curious model and against collusion attacks. It uses an untrusted mediator and leverages a simple, and efficient secure sum protocol that scales well to both large datasets and multiple parties. In addition, the proposed framework is applicable to a wide range of machine learning problems.

3 PRELIMINARIES

3.1 Data Partitioning

Let $\{X^{(i)}, y^{(i)}\}_{i=1}^{m}$ denote a dataset consisting of m data samples where $\{X^{(i)}, y^{(i)}\}$ corresponds to the ith sample, $X^{(i)} = (x_1, \cdots, x_n)$ represents an attribute vector with n explanatory (or independent) variables and $y^{(i)}$ represents the dependent variable. Let $X_j^{(i)}$ denote the jth attribute (column) in $X^{(i)}$. In this paper, we consider the setting where the dataset is partitioned across multiple parties (> 2). We consider two scenarios. In the first, the data is horizontally partitioned across parties, i.e., each party owns all the attributes for a subset of data samples. In the second, the data is vertically partitioned across parties, i.e., each party owns a subset of attributes for all data samples.

Figure 1 shows a toy example consisting of two independent attributes x_1, x_2, and a dependent variable y. Figure 1(a) shows an example of horizontal partitioning where each party owns a subset of data samples, while Figure 1(b) shows an example of vertical partitioning where each party owns a subset of attributes for all data samples.

Heart rate (x_1)	Calcium score (x_2)	Length of stay (y)
78	408	20
72	159	8
89	211	13
77	190	9

(a)

Heart rate (x_1)	Calcium score (x_2)	Length of stay (y)
78	408	20
72	159	8
89	211	13
77	190	9

(b)

Figure 1: (a) Horizontal, and (b) Vertical partitioning

3.2 ElGamal Cryptosystem

The ElGamal cryptosystem [20] consists of the following key generation, encryption and decryption algorithms:

• Key generation: Given a security parameter k, it publishes a multiplicative cyclic group $\mathbf{G}$ of prime order q with a generator g, such that discrete logarithm problem over the group $\mathbf{G}$ is hard. As a next step, it selects a secret key r randomly from $\mathbf{Z_q^*} = \{1, 2, \ldots, q-1\}$ and computes a public key $y = g^r$. The secret key r is kept private while the public key y can be known to everyone.

• Encryption: Given a message $m \in \mathbf{G}$ and a public key y, it selects an integer d randomly from $\mathbf{Z_q^*}$ and outputs a ciphertext $C = E_y[m] = (A, B)$, where $A = g^d$, $B = my^d$.

• Decryption: Given a ciphertext (A, B), and a secret key r, it outputs the plain text $m = D_r[C] = D_r[(A, B)] = \frac{B}{A^r}$.

Since the encryption algorithm selects integer d randomly, ElGamal is a non-deterministic cryptosystem.

4 SECURE SUM PROTOCOL

A secure sum protocol enables multiple parties to compute the sum of their individual values while preserving these values' privacy. Our proposed protocol leverages collusion-resistant anonymization [21] in order to provide privacy to the values of the parties. In the following, we outline the threat model, present the protocol, discuss the performance of the protocol in terms of accuracy and communication cost, and explain some of its important properties.

4.1 Threat Model

We consider an honest-but-curious adversary model, i.e., an adversary will follow the protocol but will try to acquire as much information as possible about the private values during the sum computation. The adversary has control over the untrusted mediator and at most $N-2$ parties. The collusion-resistant anonymization is achieved by randomly permuting the input values (or, the segments of the input values) submitted by the parties. This anonymization guarantees that the mediator along with the $N-2$ colluding parties will not be able to breach the anonymity of an honest party's value. Note that, at the end of our secure sum protocol, the sum value is shared across all the parties including the mediator.

4.2 Protocol

The following is the list of our assumptions on the parties and the mediator:

i. Each party P_i has a secret key and a public key, i.e., R_i, and Y_i, respectively. Also, the mediator M's secret and public keys are R_M and Y_M, respectively. The public keys of all parties and the mediator are known to all entities.

ii. The encryption and decryption operations performed by the parties always follow a particular order. For simplicity, we assume that the order is the same as the order of the parties' public keys in the list that is shared by all.

Overview of the protocol: The proposed protocol has three distinct phases: (I) input preparation, (II) anonymization and (III) sum computation. In phase (I), each party prepares its input values for submission to the mediator. The first step of this phase is to shard the individual values into a number of segments, while in the second step the value segments are recursively encrypted with the public keys of the mediator and the N parties. In phase (II), the mediator sends the set of all prepared value segments to the Nth party (the party to which the last shared public key belongs to). The Nth party performs decryption and shuffling (i.e., re-ordering of the segments) on the value segments prepared in phase (I) and sends the randomly shuffled segments to the $(N-1)th$ party. The $(N-1)th$ party then further decrypts and shuffles the value segments, and this process continues until the segments are decrypted and shuffled by the $1st$ party. Finally, the mediator receives anonymized value segments from the $1st$ party with only one layer of encryption with the mediator's public key. In phase (III), the mediator decrypts the value segments using its own secret key and computes the sum.

The following is a formal representation of the protocol phases:

(I) **Input Preparation**: This phase consists of two steps-

(a) Segmentation: Each party P_i shards its input value into a number of segments. For instance, in order to shard its value S_i into s segments, P_i generates secret shares $\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{is}$ uniformly randomly such that $\sum_{j=1}^{s} \alpha_{ij} = 1$ and for $1 \leq j \leq s$, $\alpha_{ij} > 0$. These shares are used to divide P_i's value S_i into s segments $\alpha_{i1}.S_i, \alpha_{i2}.S_i, \ldots, \alpha_{is}.S_i$ (denoted by elements of set $\mathbf{S_i} = \{S_{i1}, S_{i2}, \ldots, S_{is}\}$, respectively) such that $S_i = \sum_{j=1}^{s} S_{ij}$.

(b) Encryption: Each party P_i encrypts elements in set $\mathbf{S_i}$.

i. At first, P_i encrypts the elements in $\mathbf{S_i}$ using the mediator's public key that results in $\mathbf{S_i'} = \{S_{i1}', S_{i2}', \ldots, S_{is}'\}$ where $S_{ij}' = E_{Y_M}[S_{ij}]$ or, $\mathbf{S_i'}[j] = E_{Y_M}\mathbf{S_i}[j]$.

ii. As the next step, P_i employs the public keys of all the parties to the elements in $\mathbf{S_i'}$ and generates $\mathbf{S_i''}$ such that $\mathbf{S_i''}[j] = [E_{Y_k}[\mathbf{S_i'}[j]]]_{k=1 to N}$.

At the end of this phase, each party P_i sends the prepared value segments $\mathbf{S_i''}$ to the mediator M.

(II) **Anonymization**: The mediator M receives all prepared value segments $\mathbf{\Psi} = \{\mathbf{S_1''}, \mathbf{S_2''}, \ldots, \mathbf{S_N''}\}$. Note that the size of the set $\mathbf{\Psi}$ is $(N * s)$ since each party shards its value into s segments. Since

the encryption order is from 1 to N, the appropriate decryption order is from N to 1. Therefore, the mediator M sends the set $\mathbf{\Psi}$ to party P_N at the first iteration, which then updates $\mathbf{\Psi}$ by decrypting and randomly shuffling and sends the updated $\mathbf{\Psi}$ to party P_{N-1} for the next iteration. The iterations terminate when the mediator M receives the updated value segments from party P_1. The following explains the steps in the $(N-i+1)th$ iteration of the anonymization phase.

(a) Party P_i receives $\mathbf{\Psi}$ either from mediator (if $i = N$), or from party P_{i+1} (otherwise).
(b) P_i strips off one layer of encryption from $\mathbf{\Psi}$ as follows: $\mathbf{\Psi}[k] = D_{R_i}[\mathbf{\Psi}[k]]$ for $1 \leq k \leq (N * s)$.
(c) P_i randomly re-orders the elements in $\mathbf{\Psi}$ by using a random shuffle function π and obtains a randomized set of value segments $\mathbf{\Psi} = \mathbf{\Psi}[\pi(k)]$ for $1 \leq k \leq (N * s)$.
(d) P_i sends decrypted and shuffled $\mathbf{\Psi}$ to the mediator M (if $i = 1$) or to the party P_{i-1} (otherwise).

At the end of N iterations, the mediator M receives the anonymized value segments that has only one layer of encryption with public key Y_M.

(III) **Sum Computation**: M computes the sum as follows:
(a) M decrypts the value segments as $\mathbf{\Psi}[k] = D_{R_M}[\mathbf{\Psi}[k]]$ for $1 \leq k \leq (N * s)$.
(b) M computes the sum of the elements in set $\mathbf{\Psi}$ to obtain the global sum S_G.
(c) M sends the global sum S_G to all the parties.

4.3 Performance

4.3.1 Accuracy. This protocol does not compromise the accuracy of the resulting sum, i.e., the securely computed sum is same as the sum computed over the plain text values.

4.3.2 Communication cost. In phase (I), each party sends its prepared input which results in a number N of communications. In phase (II), the value segments are sent from the mediator to the Nth party and then to the other parties sequentially which results in a number $(N + 1)$ of communications. In phase (III), the mediator sends the global sum S_G to all the parties which results in N additional communications. Therefore, the number of communications required for each global sum computation is $(3N + 1)$, i.e., $O(N)$.

4.4 Advantage of Value Segmentation

In phase (I), each party P_i shards its value S_i into s number of segments. In order to understand the advantage of value segmentation, consider the case where there are only two honest parties (let those be P_1 and P_2) among N, and the rest $N-2$ parties are colluding with the mediator M. With an ordinary background knowledge that P_1's value should be larger than that of P_2's, it is easy to identify S_1 and S_2 from the honest parties' values. However, with value segmentation, the task of extracting exact input values of the honest parties becomes more complex (see Section 5.2 for more details).

4.5 Protocol Parallelization

Note that, in phase (II), the value segments are anonymized sequentially, i.e., while one party is anonymizing the value segments, the other $(N - 1)$ parties are not able to work on the same segments simultaneously. However, if the parties want to compute D sums at a time, i.e., their input for sum computation has D dimensions, the protocol is N-parallelizable provided that $D \geq N$. For example, if $N = 4$, and $D = 40$, we can form a unique order of parties for each 10 dimensions, and the value segments in those 10 dimensions can be encrypted and decrypted according to a particular order. In the above example, the four unique orders could be $[P_1, P_2, P_3, P_4]$, $[P_2, P_3, P_4, P_1]$, $[P_3, P_4, P_1, P_2]$, and $[P_4, P_1, P_2, P_3]$ so that at a given time all the parties can parallelly take part in anonymizing 10 dimensions without any collision.

5 SECURITY ANALYSIS OF THE PROPOSED SECURE SUM PROTOCOL

5.1 Privacy Guarantee

In phase (I), each party submits the encrypted value segments to the mediator. In phase (II), each party has access to a set of encrypted value segments which have N layers to 1 layer of encryption(s) (in N iterations, respectively) even after striping off one layer of encryption with its own secret key. Therefore, it is not possible for a party to reveal the plain text of a value segment even if $(N - 1)$ parties collude - which ensures confidentiality during the anonymization phase. In phase (III), the mediator has access to the plain text of the value segments. However, it is not possible for the mediator to identify any association between a value segment and an honest party as long as at most $(N - 2)$ parties collude with the mediator.

5.2 Collusion Resistance

Proposition: The anonymized value segments $\mathbf{\Psi}$ that the mediator M receives from party P_1 at the end of phase (II) is computationally indistinguishable as long as there are H honest parties such that $H \geq 2$.

Proof: Let us assume a scenario where the mediator M collaborates with malicious parties $P_{MAL} \subseteq P$ such that $|P|$ - $|P_{MAL}| = H$. To distinguish honest parties' value segments from the anonymized set, parties in P_{MAL} could assist the mediator M by excluding their value segments Ψ_{MAL}. However, if the number of value segments for each party is s, and there are H honest parties, the probability that the mediator M is able to associate a value segment $S_D \in \{\Psi - \Psi_{MAL}\}$ with an honest party is $\frac{s}{s*H}$, or $1/H$. Since $H \geq 2$, $1/H$ is at most $1/2$. This implies that value segments in $\{\Psi - \Psi_{MAL}\}$ are indistinguishable, i.e., each segment is equally likely to belong to each non-colluding party.

6 PRIVACY-PRESERVING MULTI-PARTY ANALYTICS ON HORIZONTALLY PARTITIONED DATA

6.1 Problem Setup

In this section, we consider the set-up where the data is horizontally partitioned across multiple parties, and the goal is to train a machine learning model on the global data while ensuring the data privacy of each individual party.

At the end of the analytics, the final model is shared across all the parties including the mediator. However, the mediator along with the colluding parties is unable to learn the data or local results of the honest parties.

6.2 Key Observations

We begin by noting that most machine learning algorithms can be formulated as an optimization problem, with the goal of minimizing a cost (or objective) function as shown below:

$$\min_{\theta_0, \theta_1, \ldots, \theta_n} J(\theta_0, \theta_1, \ldots, \theta_n) \tag{1}$$

While some optimization problems may have a closed form solution, most optimization problems rely on gradient descent, which sometimes tends to be more efficient than a closed form solution. As an example, the linear regression problem has a closed form solution that involves matrix inversion, which is a costly operation when the size of the dataset is large, and hence gradient descent is commonly used to solve the linear regression problem. Gradient descent is a simple, iterative algorithm as described below.

- Initialize the model parameters $\{\theta_0^0, \theta_1^0, \cdots, \theta_n^0\}$.
- In each iteration $t+1$, update parameter $\theta_j^{(t+1)}$, for $j = 1, 2, \cdots, n$ until termination criteria is satisfied.

$$\theta_j^{(t+1)} = \theta_j^t - \alpha \frac{\partial}{\partial \theta_j^t} J(\theta_0, \theta_1, \cdots, \theta_n) \tag{2}$$

Each step in the gradient descent algorithm involves updating all the parameters based on the current gradient value, where α in (2) denotes the learning rate that controls the rate of convergence. The update step is terminated upon convergence.

In most machine learning algorithms, the cost function J in (1) and its partial derivative $\frac{\partial J}{\partial \theta_j}$ in (2) involve a summation over the training data $\{X^{(i)}, y^{(i)}\}_{i=1}^m$ as shown below.

$$J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{m} \sum_{i=1}^{m} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(i)}, y^{(i)}\}) \tag{3}$$

$$\frac{\partial}{\partial \theta_j^t} J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_j^t} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(i)}, y^{(i)}\}) \tag{4}$$

Below, we provide two examples to demonstrate this.

(1) *Linear Regression:* In linear regression, given n independent variables (or attributes), the hypothesis function to estimate the dependent variable is given by

$$h_\theta(X) = \sum_{j=1}^{n} \theta_j x_j, \tag{5}$$

where $X = (x_1, \cdots, x_n)$ denotes the feature vector with the first feature typically corresponding to a constant (or 1). The goal of linear regression is to estimate the parameters θ_j that best fit the training data. The cost function J and its partial derivative $\frac{\partial J}{\partial \theta}$ for linear regression are given by:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(X^{(i)}) - y^{(i)})^2 \tag{6}$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \ldots \theta_n) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(X^{(i)}) - y^{(i)}) X_j^{(i)} \tag{7}$$

(2) *Binary Classification:* Logistic regression is widely used for binary classification, where given a data sample $X = (x_1, x_2, \cdots, x_n)$, the output variable is modeled as

$$h_\theta(X) = \frac{1}{1 + e^{-\sum_{j=1}^{n} \theta_j x_j}} \tag{8}$$

The cost function J and its partial derivative $\frac{\partial J}{\partial \theta}$ for logistic regression are given by:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} log(h_\theta(X^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(X^{(i)}))] \tag{9}$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \ldots \theta_n) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(X^{(i)}) - y^{(i)}) X_j^{(i)} \tag{10}$$

As noted in the above two examples, and as shown in Equations (3) and (4), both the objective function J and the partial derivative $\frac{\partial J}{\partial \theta}$ in the gradient descent algorithm are evaluated on the training data $\{X^{(i)}, y^{(i)}\}_{i=1}^m$. In the case of horizontally partitioned data, the m samples are distributed across multiple parties. Let m_i denote the number of samples with party P_i, where $\sum_{i=1}^{N} m_i = m$. Equations 3 and 4 can then be re-written as:

$$\begin{aligned} J(\theta_0, \theta_1, \cdots, \theta_n) &= \frac{1}{m} \sum_{i=1}^{m} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(i)}, y^{(i)}\}) \\ &= \frac{1}{m} \sum_{i=1}^{N} \sum_{k=1}^{m_i} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(k)}, y^{(k)}\}) \end{aligned} \tag{11}$$

and

$$\begin{aligned} \frac{\partial}{\partial \theta_j^t} J(\theta_0, \theta_1, \cdots, \theta_n) &= \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_j^t} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(i)}, y^{(i)}\}) \\ &= \frac{1}{m} \sum_{i=1}^{N} \sum_{k=1}^{m_i} \frac{\partial}{\partial \theta_j^t} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(k)}, y^{(k)}\}) \end{aligned} \tag{12}$$

Note that the inner summation $\sum_{k=1}^{m_i} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(k)}, y^{(k)}\})$ and $\sum_{k=1}^{m_i} \frac{\partial}{\partial \theta_j^t} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(k)}, y^{(k)}\})$ in each of those equations denote computations that are based on data belonging to a single party, and thus can be completely evaluated by the party. On the other hand, the outer summation involves data across parties, and thus this summation should be computed in a secure, privacy-preserving manner. Moreover, note that while the inner summation may involve complex and non-linear functions, the outer summation is simply over N scalars, each generated by a party. **In other words, irrespective of the machine learning algorithm and its cost function, the outer summation always involves a simple sum over N scalars, which can be computed using a secure sum protocol.**

6.3 Secure Gradient Descent Algorithm

Algorithm 1 describes our privacy-preserving horizontally partitioned multi-party data optimization algorithm for a generic machine learning problem with a cost function J and gradient function $\frac{\partial J}{\partial \theta}$.

At the end of the gradient descent algorithm, each party has the model parameters $\{\theta_1, \theta_2, \cdots, \theta_n\}$, which can be used by the party to predict/classify any new data samples it collects. Finally, Algorithm 1 can be naturally extended to other variants of gradient descent such as the mini-batch and stochastic gradient descent.

1: M initializes $\{\theta_1, \cdots, \theta_n\}$
2: M sets convergence to false
3: M sets J to ∞
4: **for** each gradient descent iteration until convergence is true **do**
5: M sends $\{\theta_1, \ldots, \theta_n\}$ to all parties
6: **for** $i = 1, \cdots, N$ **do**
7: **for** each dimension $j = 1, \ldots, n$ **do**
8: P_i computes local gradient $LG_{ij} = \sum_{k=1}^{m_i} \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(k)}, y^{(k)}\})$
9: **end for**
10: P_i computes local cost value $LJ_i = \sum_{k=1}^{m_i} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(k)}, y^{(k)}\})$
11: P_i shards each element in $\{LJ_i, LG_{i1}, \cdots, LG_{in}\}$ into two segments
12: P_i encrypts the shards $\{\{LJ_i\}_1, \{LJ_i\}_2, \{LG_{i1}\}_1, \{LG_{i1}\}_2, \cdots, \{LG_{in}\}_1, \{LG_{in}\}_2\}$ with the public keys of M, and P_is in order of $i = 1, \ldots, N$
13: P_i sends $\{E[\{LJ_i\}_1], E[\{LJ_i\}_2], E[\{LG_{i1}\}_1], E[\{LG_{i1}\}_2], \cdots, E[\{LG_{in}\}_1], E[\{LG_{in}\}_2]\}$ to mediator M
14: **end for**
15: M receives $\{E[\{LJ_i\}_1], E[\{LJ_i\}_2], E[\{LG_{i1}\}_1], E[\{LG_{i1}\}_2], \cdots, E[\{LG_{in}\}_1], E[\{LG_{in}\}_2]\}$ from all P_i
16: M sends the encrypted shards to party P_N
17: **for** $i = N, N-1, \cdots, 2$ **do**
18: P_i decrypts and shuffles the shards
19: P_i sends the decrypted and shuffled shards to party P_{i-1}
20: **end for**
21: P_1 decrypts and shuffles the shards
22: P_1 sends the anonymized shards to mediator M
23: M decrypts and extracts local gradients for each dimension $j = 1, \ldots, n$
24: M computes global gradient $G_j = \sum_{i=1}^{N} \sum_{k=1}^{2} \{LG_{ij}\}_k$ for each dimension $j = 1, \ldots, n$
25: **for** each dimension $j = 1, \cdots, n$ **do**
26: M updates $\theta_j = \theta_j - \alpha G_j$
27: **end for**
28: M computes $J_{new} = \sum_{i=1}^{N} \sum_{k=1}^{2} \{LJ_i\}_k$
29: **if** $J - J_{new} \leq \epsilon$ **then**
30: convergence is true
31: **end if**
32: **end for**

Algorithm 1: Algorithm for privacy-preserving horizontally partitioned multi-party data optimization for a generic machine learning algorithm.

7 PRIVACY-PRESERVING MULTI-PARTY ANALYTICS ON VERTICALLY PARTITIONED DATA

7.1 Problem Setup

The complexity of privacy-preserving analytics greatly increases when dealing with vertically partitioned data. In contrast to horizontal partitioning of data, vertical partitioning raises several unique questions with respect to the way data is processed, and results are obtained and shared. Below, we discuss two such questions and our assumptions.

i. In problems like classification and regression, there is a dependent variable y, i.e., the variable to be modeled. An important question in these problems is the location of this dependent variable. There are two possibilities: the dependent variable y is known to all parties, or it is private and belongs to some party. This impacts the way the model is built and evaluated. Both cases are realistic and model different situations. In this paper, we focus on the former where the dependent variable y is known to all parties.

ii. The second question concerns how the final model is shared among parties. One possibility is to let all the parties know the obtained model - but this may often reveal too much information and fail to comply with the privacy constraints. An alternate fully secure solution is to split the model among the parties. We design our protocol for the latter (fully secure) approach, but it can be easily extended to the former approach. Specifically, we split the model among parties, where each participant has model parameters only for the attributes it owns. However, the downside to this approach is that a secure protocol has to be run each time the model needs to be used on a new data point. If this performance penalty has to be avoided, one can resort to the former approach by sharing the global model with all the parties at the end of our secure optimization protocol.

7.2 Key Observations

Our proposed secure solution is based on two key observations. First, we note that several popular machine learning models, such as linear regression, ridge regression, LASSO, SVM and logistic regression, model the dependent variable y as $h_\theta(X) = f(\langle \Theta, X \rangle)$, i.e., as a function of the inner product between the model parameters $\Theta = (\theta_1, \theta_2, \cdots, \theta_n)$ and the attribute vector $X = (x_1, x_2, \cdots, x_n)$. In vertically partitioned data, since the attributes are distributed across multiple parties, we can use a secure sum protocol to compute $\langle \Theta, X \rangle$. Second, note from Equations 6, 7, 9, and 10 that given the value of $h_\theta(X^{(i)})$ for all training samples $i \in [1, m]$, the objective function can be evaluated by each party independently (as we assumed the dependent variable $y^{(i)}$ to be known to all parties). Similarly, each parameter θ_j can be updated by the party that owns the jth attribute. Hence, both the cost function and the update step can be computed independently by the parties once $h_\theta(X^{(i)})$ is known, which in turn can be computed using a secure sum protocol.

7.3 Secure Gradient Descent Algorithm

Algorithm 2 describes our privacy-preserving vertically partitioned multi-party data optimization algorithm for a generic machine learning problem which models the dependent variable as a function of $\langle \Theta, X \rangle$, and has a cost function J with gradient function $\frac{\partial J}{\partial \theta}$.

At the end of the proposed algorithm, each party learns the final model parameters only for the attributes it owns. Hence, in order to use the learned model to predict/classify any new data samples, the parties should initiate a secure sum protocol.

Algorithm 2 can be extended to other variants of gradient descent. For example, in the case of mini-batch gradient descent, the parties may decide a-priori on the batch size b, and random seed values. Then, in Steps 8-10, each party will compute the local h_θ only for a small subset of size b from their data (rather than the entire dataset). Also, in Steps 28 and 31, the parties will compute the gradient value and the objective function only using the selected sub-samples, and normalize it accordingly (i.e., normalize with b rather than m). Note that the use of mini-batch gradient descent will significantly improve the overall performance of our algorithm. We could similarly use stochastic gradient descent where only one random sample is used in each iteration of the gradient descent.

1: **for** i = 1,. . ., N **do**
2: P_i initializes θ_js for the jth attributes that belong to P_i
3: P_i sets convergence to false
4: P_i sets J to ∞
5: **end for**
6: **for** each gradient descent iteration until convergence is true **do**
7: **for** i = 1,. . ., N **do**
8: **for** r = 1,. . ., m **do**
9: P_i computes local h_θ, $Lh_i^r = \sum_{j=1}^{n_i} \theta_j x_j$ for each training sample r
10: **end for**
11: P_i shards each element in $\{Lh_i^1, Lh_i^2, \cdots, Lh_i^m\}$ into two segments
12: P_i encrypts the shards $\{Lh_i^1\}_1$, $\{Lh_i^1\}_2, \cdots, \{Lh_i^m\}_1$, $\{Lh_i^m\}_2\}$ with the public keys of M, and P_is in order of i = 1,. . ., N
13: P_i sends $E[\{Lh_i^1\}_1]$, $E[\{Lh_i^1\}_2], \cdots, E[\{Lh_i^m\}_1]$, $E[\{Lh_i^m\}_2]\}$ to the mediator M
14: **end for**
15: M receives $E[\{Lh_i^1\}_1]$, $E[\{Lh_i^1\}_2]$, . . ., $E[\{Lh_i^m\}_1]$, $E[\{Lh_i^m\}_2]\}$ from all P_i
16: M sends the encrypted shards to party P_N
17: **for** i = N, $N-1$,. . ., 2 **do**
18: P_i decrypts and shuffles the shards
19: P_i sends the decrypted and shuffled shards to party P_{i-1}
20: **end for**
21: P_1 decrypts and shuffles the shards
22: P_1 sends the anonymized shards to mediator M
23: M decrypts and extracts local h_θ for each row i = 1,. . ., m
24: M computes h_θ for each row i = 1,. . ., m
25: M sends h_θ values to all P_i
26: **for** i = 1,. . ., N **do**
27: **for** each dimension j that belongs to P_i **do**
28: P_i updates $\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{k=1}^{m} \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \cdots, \theta_n | \{X^{(k)}, y^{(k)}\})$
29: **end for**
30: **end for**
31: P_i computes $J_{new} = \frac{1}{m} \sum_{k=1}^{m} J(\theta_1, \theta_2, \cdots, \theta_n | \{X^{(k)}, y^{(k)}\})$, using the h_θ values
32: **if** J - $J_{new} \leq \epsilon$ **then**
33: convergence is true
34: **end if**
35: **end for**

Algorithm 2: Algorithm for privacy-preserving vertically partitioned multi-party data optimization for a generic machine learning algorithm.

8 DISCUSSION ON SECURE GRADIENT DESCENT ALGORITHMS

This section discusses some key properties of the Algorithms 1 and 2 in the following:

(1) In Step 11 of the algorithms, we shard the data into two segments. In general, one may shard the data into s segments, where $s > 1$.

(2) Steps 17-22 of the algorithms involve a sequential process, where each party decrypts the shards, shuffles them, and sends them to the next party. This results in only one party being active at any given time, with the remaining $N-1$ parties being idle (idle CPU cycles). However, we can parallelize the gradient descent operation to improve the overall CPU usage, by using a different encryption ordering for each dimension in Step 12 of the algorithms.

- In Algorithm 1, rather than encrypting all the local gradients using the same encryption order, say we encrypt all $\{LG_{i1}\}_1$, $\{LG_{i1}\}_2$ using the parties' public keys in order $i = 1, \cdots, N$; all $\{LG_{i2}\}_1$, $\{LG_{i2}\}_2$ using order $i = 2, \cdots, N, 1$; all $\{LG_{i3}\}_1$, $\{LG_{i3}\}_2$ using order $i = 3, \cdots, N, 1, 2$, etc. Then, the mediator can invoke parallel decryption cycles in Steps 17-22 by sending $E[\{LG_{i1}\}_1]$, $E[\{LG_{i1}\}_2]$ to party N first; $E[\{LG_{i2}\}_1]$, $E[\{LG_{i2}\}_2]$ to party 1 first, $E[\{LG_{i3}\}_1]$, $E[\{LG_{i3}\}_2]$ to party 2 first etc.
- In Algorithm 2, rather than encrypting all the local h_θ's using the same encryption order, say we encrypt the local sum of first data sample of all parties $\{Lh_i^1\}_1$, $\{Lh_i^1\}_2$ using the parties' public keys in order $i = 1, \cdots, N$; the local sum of second data sample $\{Lh_i^2\}_1$, $\{Lh_i^2\}_2$ using order $i = 2, \cdots, N, 1$; the third data sample $\{Lh_i^3\}_1$, $\{Lh_i^3\}_2$ using order $i = 3, \cdots, N, 1, 2$, etc. Then, the mediator can invoke parallel decryption cycles in Steps 17-22 by sending $E[\{Lh_i^1\}_1]$, $E[\{Lh_i^1\}_2]$ to party N first; $E[\{Lh_i^2\}_1]$, $E[\{Lh_i^2\}_2]$ to party 1 first, $E[\{Lh_i^3\}_1]$, $E[\{Lh_i^3\}_2]$ to party 2 first etc.

This improves the CPU utilization of all the parties, thereby reducing the overall execution time.

(3) Data segmentation (or data sharding) used in Step 11 of the algorithms makes the protocols robust to prior knowledge attacks. We could replace the data sharding solution with alternate approaches that mask the structure of the data. An alternate approach is to have the parties determine a-priori a large random number R, so that each party P_i generates s random numbers $\{r_{i1}, r_{i2}, \cdots, r_{is}\}$ from $[-R, R]$ such that $\sum_{j=1}^{s} r_{ij} = Lh_i^r$ (in Algorithm 2). The local h_θ for all other data samples $r \in [1, m]$ can also be masked using a similar approach.

9 PERFORMANCE EVALUATION

9.1 Experiment Setup

We used two testbeds in this analysis. The first testbed is a cluster of 9 Virtual Machines (VM) on a HPE Cloud openstack instance. Each VM is a single core, Intel Xeon machine with 8GB of RAM and 60GB storage running Debian GNU/Linux. The second testbed is an Amazon AWS EC2 cluster of 4 *t2.micro* instances with 1 GB of RAM and EBS-only storage. We implemented our solution in Python v2.7.3, where we used ZeroMQ for communication between parties/machines. We used an open source implementation of the ElGamal cryptosystem[2].

We performed three sets of experimental analysis. In Section 9.2, we evaluate the computational performance of the proposed secure sum protocol, measured by the latency (total wall-clock time) and the total communication cost. We evaluate and compare non-parallelized and parallelized versions of this protocol.

In Section 9.3, we evaluate our secure gradient descent protocol for two analytic use-cases: linear regression and binary classification (logistic regression) with horizontal data partitioning. We also evaluate our protocol with vertically partitioned data for the linear regression use-case. We compare the proposed solution scenario, i.e., distributed untrusted (DU) (where parties execute gradient descent locally and share encrypted intermediate results with an

[2]https://github.com/RyanRiddle/elgamal

untrusted mediator) with two baselines: (i) centralized trusted (CT), where the parties transfer their entire raw data to a trusted mediator to perform the analysis, and (ii) distributed trusted (DT), where parties execute gradient descent locally and share intermediate results with a trusted mediator to aggregate. We evaluate these solutions on several real datasets under three different metrics: accuracy or quality of the resulting machine learning model, latency, and total communication cost.

Finally, in Section 9.4, we perform a scalability analysis to evaluate the computational performance of the proposed gradient descent solution and compare it with the two baselines. We use a synthetic dataset for this analysis, where we vary the number of instances as 1K, 10K, 100K, and the number of parties as 2, 4, and 8.

All the results shown in this section are averaged over 5 repetitions of each experiment. While the experiments presented in Section 9.3 are performed on Amazon AWS EC2 instances, the HPE Cloud openstack cluster has been used for the experiments in Sections 9.2 and 9.4. Independent of testbeds and analytic use-cases, the experiments demonstrate that the DU scenario outperforms the baseline CT scenario for large datasets. Although the DT scenario results in lower communication cost than that of the DU scenario, the DT scenario has weak privacy guarantees as the parties share their intermediate results with a trusted mediator in plain text. Moreover, as the dataset size increases, the difference between the latencies of the DT and DU scenarios reduces as shown later in this section.

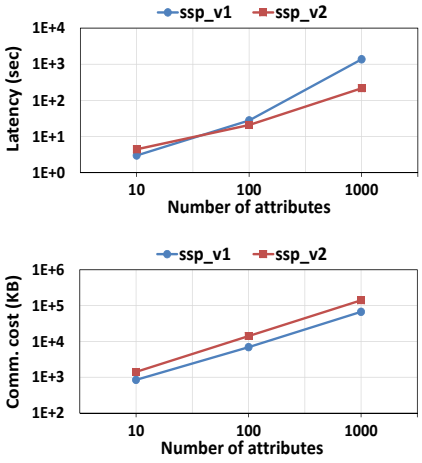

Figure 2: Latency and communication cost comparison between non-parallelized (ssp_{v1}) and parallelized (ssp_{v2}) secure sum

9.2 Experimental Analysis of Secure Sum Protocol

We implement non-parallelized and parallelized versions of the proposed secure sum protocol, and evaluate their performance measured by latency and communication cost. For all the experiments in this section, we perform secure linear regression on a dataset consisting of 100K instances, and partitioned horizontally between 2 parties. As noted in Section 4.5, the secure sum protocol can be parallelized to compute D secure sums at the same time. This parallelization is useful for secure gradient descent solutions as such solutions require to compute the global gradient for each independent attribute. In this experiment, we compare the latency and communication cost between the non-parallelized (ssp_{v1}) and the parallelized (ssp_{v2}) versions of our secure sum protocol, where we vary the number of attributes as 10, 100, and 1000. In the case of ssp_{v2}, the order of public keys' used for secure global gradient computation of half of the attributes is $[P_1, P_2]$ whereas the order $[P_2, P_1]$ is used for the rest half attributes. As shown in Figure. 2, the parallelized version improves the latency significantly when compared to the non-parallelized version as the number of attributes increases. However, the parallelized version incurs a higher communication overhead as the number of communications increases due to parallelization. Note that, in the following experiments, the results with the DU scenario represent the non-parallelized version of the secure sum protocol.

Dataset name	# of Instances	# of Attributes	Use-case
Bike Sharing [22]	17389	16	Linear Regression
YearPredictionMSD [23]	515345	90	Linear Regression
Phishing Websites [23]	2456	30	Binary Classification
SUSY [24]	5000000	18	Binary Classification

Table 1: Datasets from UCI repository

9.3 Experimental Analysis of Secure Gradient Descent Algorithms

We present the simulation results on four datasets from the UCI repository as shown in Table 1. We use 500,000 randomly selected instances from the SUSY dataset while all the instances from the other datasets are used. For each dataset, we assume 70% of the data as training data, which is distributed uniformly across 3 parties. For example, in the case of the YearPredictionMSD dataset, each party owns 120,247 instances under horizontal partitioning and 30 columns under vertical partitioning.

9.3.1 Linear Regression on Horizontally Partitioned Data. The number of gradient descent iterations for linear regression on the horizontally partitioned Bike Sharing and YearPredictionMSD datasets are 150 and 25, respectively. Figure 3(a) and 3(b) show comparisons among the CT, DT, and DU scenarios in terms of their latency and communication cost, respectively. Though the latency of the CT scenario is minimum for the smaller Bike Sharing dataset, regression analysis on the larger YearPredictionMSD dataset results in the maximum latency for this scenario. The latency in the CT scenario consists of both the communication time to transfer all the data to the mediator, and the computation time to execute linear regression centrally. In contrast to CT, the cost of the DT and DU scenarios depend mostly on the number of iterations and less on the size of the dataset. In the case of large datasets like YearPredictionMSD, the difference between the latencies of the DT and DU scenarios decrease since in both scenarios the local gradient computation takes significant amount of time (due to the size of dataset) and the latency for running the secure sum protocol in the DU scenario becomes insignificant. The communication cost for the CT scenario is also the maximum among all for the YearPredictionMSD dataset.

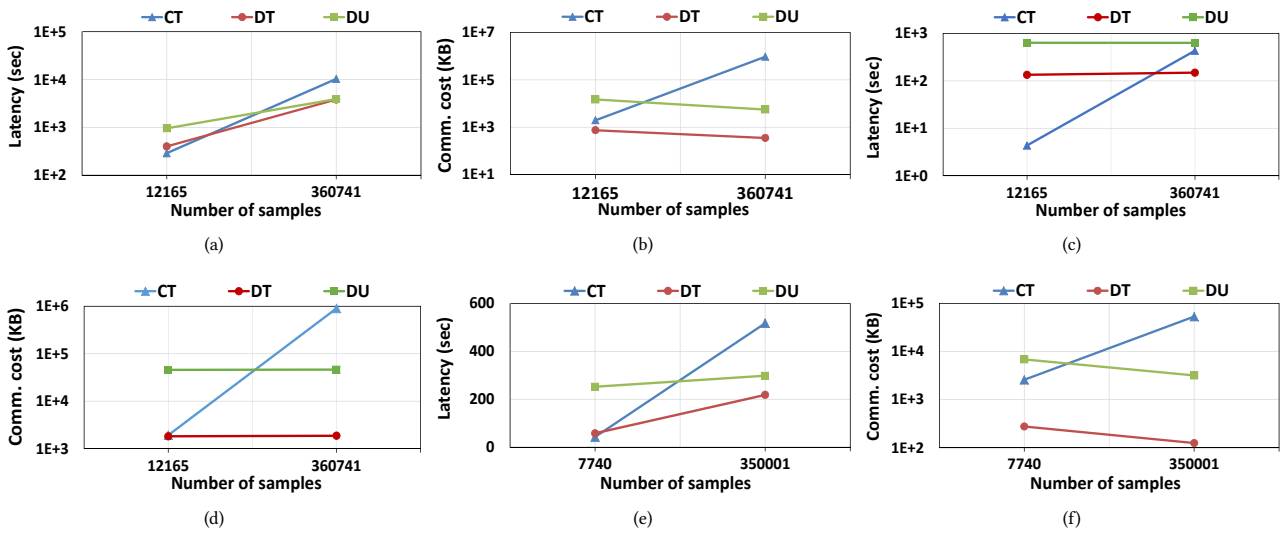

Figure 3: Comparison among the CT, DT, and DU scenarios for analytics on real datasets: (a), (b) latency and communication cost for linear regression on horizontally partitioned data; (c), (d) latency and communication cost for linear regression on vertically partitioned data; (e), (f) latency and communication cost for logistic regression on horizontally partitioned data

We assess the accuracy of the CT, DT, and DU scenarios using the Root Mean Square Error (RMSE) measure. The experiment shows that the DU scenario has the same accuracy as the two baselines. For example, the RMSE calculated for YearPredictionMSD test dataset is 1040.9 for all the scenarios. Since all the scenarios have same accuracy, we omit the accuracy comparison in later experiments.

9.3.2 Linear Regression on Vertically Partitioned Data. In the case of vertical partitioning, we would need a secure sum for each sample as described in Section 7.2. In order to reduce the overhead in latency and communication cost, we randomly select a set of subsamples from the large dataset. In other words, we implement mini-batch gradient descent. For both the Bike Sharing and YearPredictionMSD datasets, a mini-batch of 100 samples is used for computing the gradients in each step. The number of gradient descent iterations for both datasets is 100. Figure 3(c) and 3(d) show that while the latency and communication cost for the DT and DU scenarios are similar for both datasets, these metrics increase linearly with the increase in the size of the dataset for the CT scenario.

9.3.3 Logistic Regression on Horizontally Partitioned Data. The number of gradient descent iterations for logistic regression on the horizontally partitioned Phishing Websites and SUSY datasets are 38 and 25, respectively. As shown in Figure 3(e), the CT scenario result in the highest latency for the large SUSY dataset while the difference between the latencies of the DT and DU scenarios decreases. Figure 3(f) shows the results of the comparison among the CT, DT, and DU scenarios in terms of communication cost which are similar to the results in Figure 3(b).

9.4 Scalability Analysis of Secure Gradient Descent Algorithms

In order to perform the scalability analysis, we use synthetic datasets for the linear regression use-case. The datasets are generated using the `make_regression`[3] function. We use a life sciences dataset[4] for the binary classification use-case.

9.4.1 Linear Regression on Horizontally Partitioned Data. In the following, we present experiment results for linear regression on horizontally partitioned data with variable number of samples and parties.

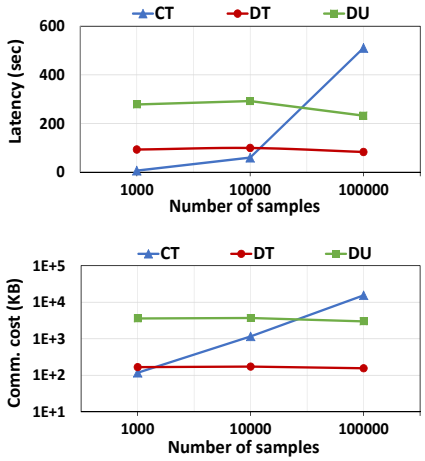

Figure 4: Comparison among the CT, DT, and DU scenarios in terms of latency and communication cost for linear regression on horizontally partitioned data (variable number of samples)

[3] http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_regression.html
[4] ds1.10, http://komarix.org/ac/ds/)

• *Variable Number of Samples:* Figure 4 shows the comparison among the CT, DT, and DU scenarios in terms of latency and communication cost for linear regression on horizontally partitioned data with variable number of samples. The experiment is with three sets of data of sample sizes 1K, 10K, and 100K, where each sample consists of 5 input attributes. Two parties participate in the analytics (each owns half of the samples) and the numbers of gradient descent iterations required for convergence for the three datasets are 91, 94, and 85, respectively, and are equal for all the CT, DT, and DU scenarios.

As shown in Figure 4, the latency of the CT scenario increases as the dataset size increases. The latency of the DU scenario for all sample sizes is higher than the DT scenario and that is attributed to the cryptographic operations in the secure sum protocol of Section 4, i.e., onions of encryprion, and decryption. The communication cost of the CT scenario also increases linearly with the dataset size which is not true for the DT and DU scenarios. Again, the communication cost of the DT and DU scenarios depend on the number of iterations. From this experiment, it is clear that using the CT scenario for larger datasets (in most practical cases datasets are large) is not cost effective. Moreover, even though the DT scenario has lower latency and communication cost when compared with the DU scenario, the latter provides stronger security guarantees.

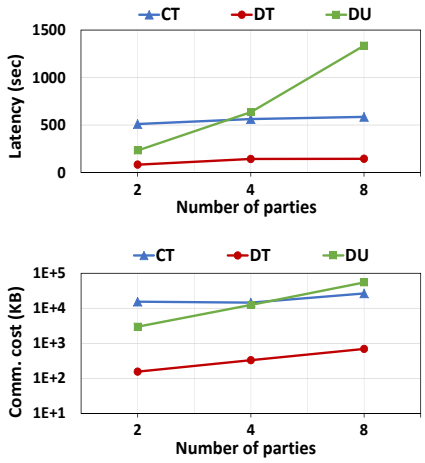

Figure 5: Comparison among the CT, DT, and DU scenarios in terms of latency and communication cost for linear regression on horizontally partitioned data (variable number of parties)

• *Variable Number of Parties:* This experiment has a similar setting as above but varies the number of parties as 2, 4, and 8 instead of varying the number of samples. The experiment is with a dataset of sample size 100K, where each sample consists of 5 input attributes. Two, four, and eight parties participate in the analytics (each owns 50K, 25K, and 12.5K samples, respectively) and the number of gradient descent iterations required for convergence is 85.

As shown in Figure 5, the latency of the DU scenario grows faster than that of the CT and DT scenarios. This is attributed to the increased number of encryptions and decryptions required by the increased number of parties. However, the growth of communication cost is slower than that of latency. Moreover, for larger datasets, e.g., the dataset with 1M samples, the DU scenario would outperform the CT scenario in terms of communication cost even in the case of 8 parties.

9.4.2 Linear Regression on Vertically Partitioned Data. While performing multi-party analytics on vertically partitioned datasets, we use mini-batch gradient descent. In the following, we present experiment results for variable size of mini-batch, variable number of samples, and variable number of parties.

• *Variable Number of Mini-batch Size:* In the following, we assess how the size of mini-batch affects the latency and communication cost of the CT and DT scenarios. The experiment is with a dataset of sample size 100K, where each sample consists of 8 input attributes. Two parties participate in the analytics (each owns 4 input attributes) and the number of gradient descent iterations required for convergence is 89.

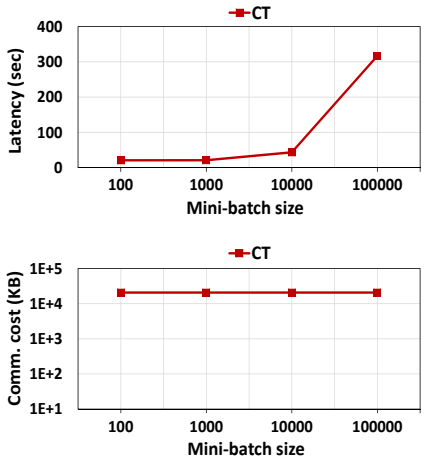

Figure 6: Linear regression on vertically partitioned data for the CT scenario: latency and communication cost comparison for variable mini-batch size

(a) *CT Scenario:* We vary the mini-batch size with 0.1K, 1K, 10K, and compare these with the case of 100K. Figure 6 shows comparisons among different mini-batch sizes in terms of latency and communication cost.

The latency increases with the increasing mini-batch size. This is because in the case of 0.1K subsamples, the mediator randomly selects 0.1K samples for each iteration of gradient descent and updates the θ values according to only those subsamples. However, in the case of 10K subsamples, the mediator needs to run analysis on 10K rows in each iteration which increases the computation time significantly. The communication costs for all mini-batch sizes are similar as the most significant cost component is the initial communication cost to transfer the data to the mediator.

(b) *DT Scenario:* We vary the mini-batch size with 0.1K, 1K, 10K, and compare those with the case of 100K. Figure 7 shows comparisons among different mini-batch sizes in terms of latency and communication cost.

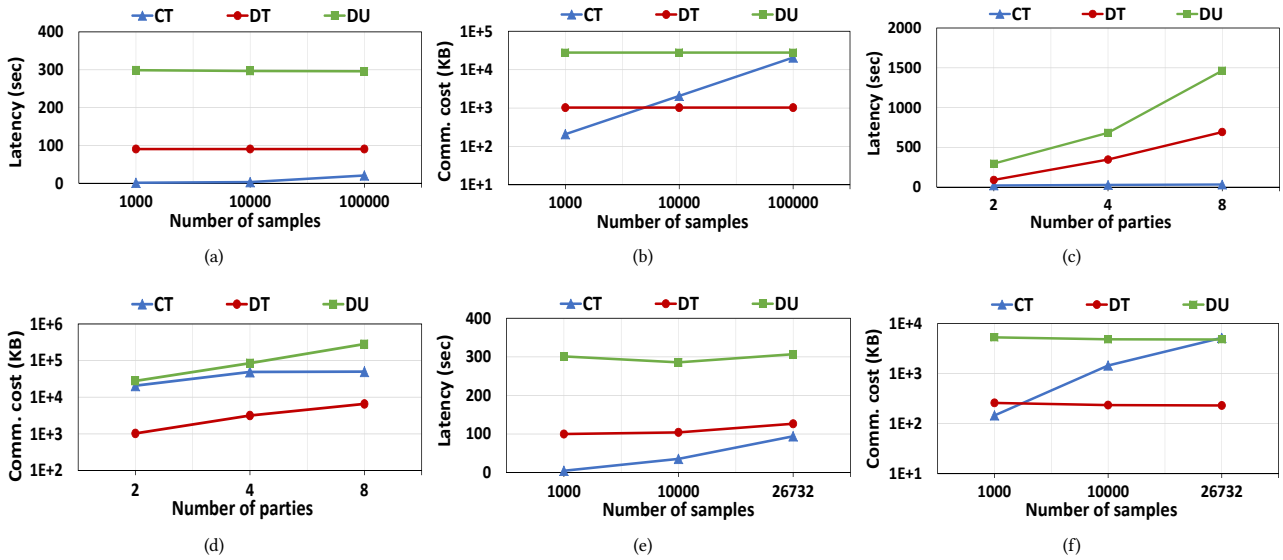

Figure 8: Comparison among the CT, DT, and DU scenarios for scalability experiments: (a), (b) latency and communication cost for variable number of samples [linear regression on vertically partitioned data]; (c), (d) latency and communication cost for variable number of parties [linear regression on vertically partitioned data]; (e), (f) latency and communication cost for variable number of samples [logistic regression on horizontally partitioned data]

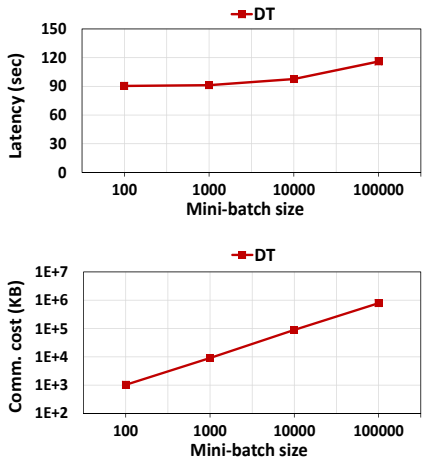

Figure 7: Linear regression on vertically partitioned data for DT scenario: latency and communication cost comparison for variable mini-batch size

The communication cost increases with the increasing mini-batch size. This is because in the case of 0.1K subsamples, the parties need to randomly select 0.1K rows and send the 0.1K local h_θ sums to the mediator at each iteration. The mediator then computes 0.1K sums and updates the θ values accordingly. However, in the case of 10K subsamples, the parties need to send 10K local h_θ sums which increases the communication cost significantly. The increasing number of bytes being transferred with increasing size of mini-batch results in higher latency though the number of gradient descent iterations is fixed for all the cases.

- *Variable Number of Samples:* Figure 8(a) and 8(b) show comparisons among the CT, DT, and DU scenarios in terms of latency and communication cost for variable number of samples. The experiment is with three sets of data of sample sizes 1K, 10K, and 100K, where each sample consists of 8 input attributes. Two parties participate in the analytics (each owns 4 input attributes), the size of mini-batch is 0.1K, and the number of gradient descent iterations required for convergence is 89.

  As shown in Figure 8(a), the latency of the CT scenario is less in comparison with both the DT and DU scenarios. The computation time at the mediator is significantly reduced in the CT scenario since the analysis runs on only 0.1K randomly selected subsamples in each iteration. The increase in the latency with increasing size of dataset in the CT scenario is due to the increase in communication time for transferring the data to mediator. Since the latency of the DT and DU scenarios depend on the number of iterations, it remains similar for all sample sizes. However, in terms of communication cost, the CT scenario performs worse with increasing number of samples as shown in Figure 8(b). Therefore, for larger datasets, e.g., dataset with 1M samples, the DU scenario would outperform the CT scenario in terms of communication cost.
- *Variable Number of Parties:* Figure 8(c) and 8(d) show comparisons among the CT, DT, and DU scenarios in terms of latency and communication cost for 2, 4, and 8 parties. The experiment is with a dataset of sample size 100K, where each sample consists of 8 input attributes. Two, four, and eight parties participate in the analytics (each owns 4, 2, and 1 input attribute(s), respectively) and the number of gradient descent iterations required for convergence is 89.

  As shown in Figure 8(c), the latency of the DU scenario grows faster than that of the CT and DT scenarios. This is attributed to

the increased number of encryptions and decryptions required by the increased number of parties. The latency of the CT scenario is significantly less in comparison with the latency of both the DT and DU scenarios for the reason mentioned for the previous experiment. The growth of communication cost for the DU scenario is slower than that of latency (Figure 8(d)). For larger datasets, e.g., dataset with 1M samples, the DU scenario would perform the CT scenario in terms of communication cost even in the case of 8 parties.

9.4.3 Logistic Regression on Horizontally Partitioned Data. In the following, we present experiment results for logistic regression on horizontally partitioned data.

- *Variable Number of Samples:* Figure 8(e) and 8(f) show comparisons among the CT, DT, and DU scenarios in terms of latency and communication cost for variable number of samples. The experiment is with three sets of life sciences datasets- one consisting of ~27K samples, and the two other consisting of randomly selected samples from these ~27K samples and of sizes 1K and 10K, where each sample consists of 10 input attributes. Two parties participate in the analytics (each owns half of the samples) and the numbers of gradient descent iterations required for convergence are 98, 89, and 88, respectively, for the 1K, 10K, and ~27K datasets.

As shown in Figure 8(e), the latency of the CT scenario increases as the dataset size increases. Although the numbers of iterations for sample sizes 10K, and ~27K are almost similar, the increase in the latency for the ~27K case is due to the local gradient computation time. The communication cost of the CT scenario increases linearly with the dataset size which is not true for the DT and DU scenarios (Figure 8(f)). Even with a dataset of ~27K samples, the DU scenario outperforms the CT scenario in terms of communication cost.

10 CONCLUSION AND FUTURE WORK

In this paper, we introduce a generalized framework for privacy-preserving multi-party analytics. We first propose a secure sum protocol that is secure under the honest-but-curious model, and is resistant to collusion attacks as long as there are at least two honest (non-colluding) parties. We then use this protocol to propose two secure gradient descent solutions, one for horizontally paritioned data and the other for vertically partitioned data. The proposed framework is generic and applies to a broad class of machine learning problems. We demonstrate our solution for two popular analytic use-cases, regression and classification, and evaluate its performance with respect to the accuracy of the obtained model, latency and communication cost, on both real and synthetic datasets. We also assess the scalability properties of the proposed solution as the size of the data or the number of parties increase. We compare the proposed solution with two baselines that use a trusted mediator. Our experimental results demonstrate that while the proposed solution does not compromise on the accuracy of the obtained model, it is secure, scales well, and achieves better computational performance (latency and communication cost) compared to the centralized trusted mediator solution as the size of the data increases. As part of the future work we plan to expand our study to evaluate the proposed solution for other machine learning algorithms such as LASSO and SVM, and also extend the scalability analysis to study the effect of increasing data dimensionality on the solution complexity.

11 ACKNOWLEDGMENTS

The work reported in this paper has been partially supported by the Schlumberger Foundation under the Faculty For The Future (FFTF) Fellowship.

REFERENCES

[1] Third-Party Vendors a Weak Link in Security Chain. http://www.esecurityplanet.com/network-security/th ird-party-vendors-a-weak-link-in-security-chain.html
[2] Recent Data Security Breaches Involving Third-Party Vendors. https://www.privacyandsecurityforum.com/ wp-content/uploads/2015/10/25092-Privacy-and-Data-Security-Breach.pdf.
[3] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, Michael Y.: Tools for privacy preserving distributed data mining. ACM Sigkdd Explorations Newsletter 4.2, pp. 28–34 (2002)
[4] Sheikh, R., Kumar, B., Mishra, D. K.: A distributed k-secure sum protocol for secure multi-party computations. arXiv preprint arXiv:1003.4071 (2010)
[5] Yao, Andrew C.: How to generate and exchange secrets. In Proceedings of the 27th Annual Symposium on Foundations of Computer Science. IEEE Computer Society, Washington, DC, USA, 162-167 (1986)
[6] Muralidhar, K., Parsa, R., Sarathy, R.: A General Additive Data Perturbation Method for Database Security. Manage. Sci. 45, 10, 1399-1415 (1999)
[7] Verykios, Vassilios S., Bertino, E., Fovino, Igor N., Provenza, Loredana P., Saygin, Y., Theodoridis, Y.: State-of-the-art in privacy preserving data mining. SIGMOD Rec. 33, 1 , 50-57 (2004)
[8] Aggarwal, C. C., Yu, P. S.: A general survey of privacy-preserving data mining models and algorithms, Privacy-Preserving Data Mining, Advances in Database Systems, vol. 34, Springer, US, pp. 11-52 (2008)
[9] Lindell, Y., Pinkas, B.: Secure Multiparty Computation for Privacy-Preserving Data Mining, Journal of Privacy and Confidentiality: Vol. 1: Iss. 1, Article 5 (2009)
[10] Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. In Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology, Mihir Bellare (Ed.). Springer-Verlag, London, UK, UK, 36-54 (2000)
[11] Kantarcioglu, M., Clifton, C.: Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. IEEE Trans. on Knowl. and Data Eng. 16, 9, 1026-1037 (2004)
[12] Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, USA, 639-644 (2002)
[13] Vaidya, J.: Privacy preserving data mining over vertically partitioned data, Ph.D. dissertation, Purdue University, West Lafayette, Indiana. http://www.cs.purdue.edu/ homes/jsvaidya/thesis.pdf (2004)
[14] Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10, 5, 557-570 (2002)
[15] Narayanan, A., Shmatikov, V.: Myths and fallacies of Personally Identifiable Information. Commun. ACM 53, 6, 24-26 (2010)
[16] Narayanan, A., Shmatikov, V.: De-anonymizing Social Networks. In Proceedings of the 2009 30th IEEE Symposium on Security and Privacy. IEEE Computer Society, Washington, DC, USA, 173-187 (2009)
[17] Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. Dissertation. Stanford University, Stanford, CA, USA (2009)
[18] Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, USA, 217-228 (2002)
[19] Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In Proceedings of the ACM SIGMOD international conference on Management of data. ACM, New York, NY, USA, 37-48 (2005)
[20] El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory, 31(4), pp. 469–472 (1985)
[21] Ashrafi, Mafruz Z., Ng, See K.: Collusion-resistant Anonymous Data Collection Method. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 69–78 (2009)
[22] Fanaee-T, H., Gama, J.: Event labeling combining ensemble detectors and background knowledge, Progress in Artificial Intelligence: pp. 1-15, Springer Berlin Heidelberg (2013)
[23] Lichman, M.: UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science (2013)
[24] Baldi, P., Sadowski, P., Whiteson, D.: Searching for Exotic Particles in High-energy Physics with Deep Learning. Nature Communications 5, (2014)