# Achieving Secure and Differentially Private Computations in Multiparty Settings

Abbas Acar*, Z. Berkay Celik†, Hidayet Aksu*, A. Selcuk Uluagac*, Patrick McDaniel†

*CPS Security Lab, Department of ECE, Florida International University
{aacar001,haksu,suluagac}@fiu.edu
†SIIS Laboratory, Department of CSE, The Pennsylvania State University
{zbc102, mcdaniel}@cse.psu.edu

***Abstract*—Sharing and working on sensitive data in distributed settings from healthcare to finance is a major challenge due to security and privacy concerns. Secure multiparty computation (SMC) is a viable panacea for this, allowing distributed parties to make computations while the parties learn nothing about their data, but the final result. Although SMC is instrumental in such distributed settings, it does not provide any guarantees not to leak any information about individuals to adversaries. Differential privacy (DP) can be utilized to address this; however, achieving SMC with DP is not a trivial task, either. In this paper, we propose a novel Secure Multiparty Distributed Differentially Private (SM-DDP) protocol to achieve secure and private computations in a multiparty environment. Specifically, with our protocol, we simultaneously achieve SMC and DP in distributed settings focusing on linear regression on horizontally distributed data. That is, parties do not see each others' data and further, can not infer information about individuals from the final constructed statistical model. Any statistical model function that allows independent calculation of local statistics can be computed through our protocol. The protocol implements *homomorphic encryption* for SMC and *functional mechanism* for DP to achieve the desired security and privacy guarantees. In this work, we first introduce the theoretical foundation for the SM-DDP protocol and then evaluate its efficacy and performance on two different datasets. Our results show that one can achieve individual-level privacy through the proposed protocol with distributed DP, which is independently applied by each party in a distributed fashion. Moreover, our results also show that the SM-DDP protocol incurs minimal computational overhead, is scalable, and provides security and privacy guarantees.**

***Keywords*—*Secure computation; differential privacy; multiparty; distributed differential privacy; predictive models; regression***

## I. Introduction

Secure and private computation of statistical models is increasingly used in different operational settings from healthcare [1]–[3] to finance [4] and security sensitive applications [5], [6]. Given the distributed nature of these applications, security and privacy are mostly achieved by utilizing Secure Multiparty Computation (SMC). SMC allows distributed parties to jointly compute an agreed function over their private inputs without revealing those inputs to other parties. Each party learns the final result, but no other information. However, SMC has a major privacy concern for a targeted individual as it does not guarantee that the final result of distributed computation would not leak any information about an individual in a sensitive dataset. Privacy of individuals and their data can be easily violated. [7]–[9]. Therefore, there is a need for a mechanism, where individual parties do not see each others' inputs and further can not infer their data from the final constructed model. Indeed, combining SMC with Differential Privacy (DP) could solve this privacy problem as DP introduces sufficient noise into the final result to prevent any leakage about a single individual.

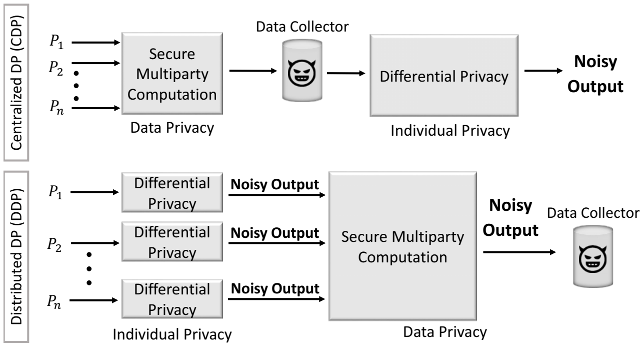

Figure 1. Illustration of secure multiparty computation with distributed and centralized differential privacy methods.

However, combining SMC with DP is not a trivial task. In an ideal case, a trusted data collector[1] can collect the data, aggregate them and add calibrated noise to the results of the queries (predictions) (Centralized DP (CDP) in Fig. 1). However, a trusted party does not exist in many real life scenarios. This technique would easily leak the model of the sensitive data to an untrusted data collector who collects the final model of the data. Even for scenarios with a trusted data collector, relying on the centralized entity makes it a single point of failure for the entire data collection mechanism.

On the other hand, another mechanism involves applying a data sanitization technique (Distributed DP (DDP) in Fig. 1) directly on the local data held by the parties. In this case, the untrusted data collector can not infer individuals' data since sufficient noise is injected by DP to hide the individuals' data. However, this mechanism requires a meticulous analysis since it may lead to a divergent or excessive amount of accumulated noise due to DP at the data collector end. As such, this process may lead to a significant accuracy loss in the final models,

[1] A data collector is either one of the parties or a third party. Every discussion here applies to both of the types.

978-1-5386-1027-5/17 $31.00 © 2017 IEEE
DOI 10.1109/PAC.2017.12

which may cause catastrophic consequences in, for example, the healthcare domain. Therefore, enabling distributed differential privacy on local data with differential privacy guarantees on final results is a challenging problem.

In this paper, we are motivated to provide a solution to this problem. Specifically, we propose a novel protocol for achieving Secure Multiparty Distributed Differentially Private (SM-DDP) computations on sensitive data. The protocol provides the guarantees of both SMC and DP. SMC is provided through Homomorphic Encryption (HE) [10] while DP is provided via Functional Mechanism (FM) [11]. An important characteristic of FM is that it injects noise into the feature matrices (i.e., coefficients of objective function), which can be computed independently by each party in a multiparty computational environment. We explore this feature of FM and apply it to linear regression using our SM-DDP protocol, but it can be applied to the computation of any statistical model function that allows independent calculation from the local statistics. We show that the accumulated noise in our protocol is still bounded and convergent by using the infinite divisibility property of Laplacian distribution [12]. Finally, we evaluated SM-DDP protocol's computational efficacy on linear regression using two real-world datasets. We compare our results with the use of Centralized DP (CDP) in a multiparty setting as in Fig. 1. The intuition is that the distributed setting of DP (DDP), which is proposed in this paper, would cause a greater accuracy loss than the typical client-server setting of SMC systems. However, we show exactly same trade-off can be achieved using the SM-DDP protocol that is presented in Fig. 3. The extensive evaluation results indicate that the proposed SM-DDP protocol yields minimal computational overhead—less than a minute for 20 parties with 32 attributes and 10K samples. The individual parties obtain better accuracy than that would be obtained from a single party model. Finally, SM-DDP is scalable while providing security and privacy guarantees.

**Contributions:** In this paper, we summarize our contributions as follows:

- We proposed a novel Secure Multiparty Distributed Differentially Private (SM-DDP) protocol to achieve secure and differentially private computations in distributed multiparty settings. This protocol can be applied to any statistical model function that allows the calculation of global model from the independent local statistics.
- We implemented the SM-DDP protocol on linear models. We showed that SM-DDP allows parties to compute regression model on pooled data while providing secure computation and differential privacy guarantees.
- We showed that the accumulated noise in our protocol is bounded and convergent. This allows parties to build a model function, which offers the individual-level privacy against an untrusted data collector.
- We evaluated the performance of the proposed protocol using two different datasets. The results demonstrated that the parties compute the models in less than a minute while preserving the security guarantees of SMC and DP.

**Organization:** The rest of the paper is organized as follows: We present the related work in Section II. In Section IV, we give the technical preliminaries about SMC and DP methods that we utilized. Then, in Section III, background about regression analysis and specifically distributed calculation of linear regression is given. In Section V, a novel protocol for SM-DDP computation of a statistical model function $f$ and its application to linear regression is presented. Furthermore, we give the experimental results for the application of our protocol to linear regression in terms of accuracy, scalability, computational overhead, and security trade-offs in Section VI. Finally, we discuss some of the related issues in Section VII and then we conclude the paper in Section VIII.

## II. Related Work

There have been many works on the secure computation of linear regression over distributed databases [13]–[17]. In these, the threat model is considered as a third party that does not have access to data, but curious about it. However, one of the parties may want to release the model function after computing function securely, which still poses threats to the individuals [7]–[9]. DP copes with this problem as it injects a certain amount of noise to the results of the queries to mask the individuals in the database. Indeed, there have been different works about the DP [18]–[21] and particularly about differentially private linear regression [11], [22]–[27]. However, these works consider DP without SMC. Although they are useful, they only provide privacy guarantees that the output of queries does not carry information about the individuals.

Approaches combining SMC and DP to provide both individual-level privacy and secure computation would be more secure. However, combining DP and SMC is not trivial; indeed, it is a rather challenging task since the application of centralized DP just after SMC in client-server settings would leak the model to an untrusted data collector, which results in a privacy violation of individuals in the database. Applying distributed DP directly on the local data held by the parties is more secure, but if each user independently injects noise randomly, it may lead to an excessive or uncontrollable amount of accumulated noise at the data collector end. Recent works focused on combining SMC and DP [28]–[30], but none of them focused on linear regression. As pointed in [11], the main reason behind this is that the regression analysis involves an optimization problem, which makes it harder to control the required amount of noise, and if the data is also distributed among parties, that makes it much more difficult to control the privacy-accuracy trade-off introduced by DP.In another relevant work [31], a combination of SMC and DP is proposed for aggregate classifiers. However, this approach injects the noise to the optimum model parameter. This resulted in excessive noise in the global model and significant loss in the accuracy. Particularly, the experimental evaluation shows that when the classifier is locally trained, the error rate obtained from locally trained classifiers is higher than the optimum error rates that could be obtained from a centralized approach. However, in our work, we take a different approach from this work. We deploy FM [11], which adds noise to local statistics, which provides the same model as the centralized approach. Lastly, even though a similar idea is proposed in [32], it is not analyzed in detail.

## III. Linear Models

In this section, we start by introducing the linear models. We, then, show how to compute linear regression in a distributed fashion.

### A. Background

Assume a database $D$ consists of $n$ observations $\{x_i, y_i\}_{i=1}^{n}$, where $x_i$ is a vector of $d$ attributes (i.e., $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ and $y_i$ is a scalar response. The aim is to find a *model function* $f : X \rightarrow Y$ that can predict $y_i \in Y$ as close as its actual value using the attributes $x_i \in X$. The type of the regression model is decided by the type of the model function. For instance, in linear regression, the model function is simply a straight line. Model function $f$ takes model coefficients $w = (w_1, w_2, \ldots, w_d)$ and $x_i$ as inputs and outputs a prediction for the value of $y_i$. The deviations between predicted value and the actual response value are calculated through a *loss function* $\ell : Y \times Y \rightarrow \mathbb{R}$. The global value of $w$ over the training data $D$ is calculated by the objective function. We denote the objective function by $\mathcal{L}$ and it is calculated as follows:

$$\mathcal{L}(f, D) = \sum_{i=1}^{n} \ell(f(x_i, w), y_i). \tag{1}$$

### B. Distributed Linear Regression

Regression is a statistical approach that explores the relationships between a set of independent variables called *attributes* and one dependent variable called *response*. In regression, the relationship between the attributes and the response is modeled using a prediction function.

In linear regression, $L_2$-norm of the objective function (i.e., $\ell(f(x_i, w), y_i) = (w \cdot x_i - y_i)^2$) that is minimized in the matrix form as follows:

$$w^* = \arg\min_{w} \mathcal{L}(f, D) = \arg\min_{w} \sum_{i=1}^{m} (w \cdot x_i - y_i)^2, \tag{2}$$

where $m$ is the number of tuples in the database. To calculate the regression in a distributed way, we represent the regression objective by minimizing with the *Maximum likelihood Estimation* (MLE). MLE allows us to obtain the global solution of the Equation 2 as follows[2]:

$$w^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}. \tag{3}$$

We characterize the model parameter $w$ of each party using three parameters:

$$\mathcal{P}_i = \mathbf{X}_i^\top \mathbf{X}_i, \mathcal{V}_i = \mathbf{X}_i^\top \mathbf{Y}_i, \mathcal{O}_i = \mathbf{Y}_i^\top \mathbf{Y}_i \tag{4}$$

Each party computes its *local statistics* $< \mathcal{P}_i, \mathcal{V}_i, \mathcal{O}_i >$ and shares with other parties. Then, the global values of $\mathcal{P}$,$\mathcal{V}$ and $\mathcal{O}$ are computed using the shared local statistics as follows:

$$\mathcal{P} = \mathbf{X}^\top \mathbf{X} = \left[X_{i_1}^\top | \ldots | X_{i_n}^\top\right] \begin{bmatrix} X_{i_1} \\ \vdots \\ X_{i_n} \end{bmatrix} = \sum_{k=1}^{n} \mathbf{X}_{i_k}^\top \mathbf{X}_{i_k} = \sum_{k=1}^{n} \mathcal{P}_k$$

$$\mathcal{V} = \mathbf{X}^\top \mathbf{Y} = \left[X_{i_1}^\top | \ldots | X_{i_n}^\top\right] \begin{bmatrix} Y_{i_1} \\ \vdots \\ Y_{i_n} \end{bmatrix} = \sum_{k=1}^{n} \mathbf{X}_{i_k}^\top \mathbf{Y}_{i_k} = \sum_{k=1}^{n} \mathcal{V}_k$$

$$\mathcal{O} = \mathbf{Y}^\top \mathbf{Y} = \left[Y_{i_1}^\top | \ldots | Y_{i_n}^\top\right] \begin{bmatrix} Y_{i_1} \\ \vdots \\ Y_{i_n} \end{bmatrix} = \sum_{k=1}^{n} \mathbf{Y}_{i_k}^\top \mathbf{Y}_{i_k} = \sum_{k=1}^{n} \mathcal{O}_k,$$

where $n$ is the number of parties in the collaboration. Using this, the global coefficients can be computed as follows:

$$w^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} = \mathcal{P}^{-1} \mathcal{V}. \tag{5}$$

In order to calculate the error of the global function, we rewrite the objective function in Equation 2 in terms of the local statistics (i.e., matrix form) as follows:

$$\begin{aligned} \sum_{i=1}^{m} (w \cdot x_i - y_i)^2 &= (\mathbf{X}w - \mathbf{Y})^\top (\mathbf{X}w - \mathbf{Y}) \\ &= ||(\mathbf{X}w - \mathbf{Y})||^2 \\ &= w^\top \mathbf{X}^\top \mathbf{X} w - 2w^\top \mathbf{X}^\top \mathbf{Y} + \mathbf{Y}^\top \mathbf{Y} \\ &= w^\top \mathcal{P} w - 2w^\top \mathcal{V} + \mathcal{O}, \end{aligned} \tag{6}$$

where $|| \cdot ||$ denotes the Euclidean norm. We note that even though we do not need $\mathcal{O}$ to calculate the global coefficients, it is used for computing the error of the model.

## IV. Technical Preliminaries

Preserving the privacy of the users and data is a long-studied problem in the area of cryptography [16], [18], [22], [30], [34], [35]. As a result of these long-term studies, there are several theoretically well-studied tools that can be employed to protect the data and user privacy such as Secure Multiparty Computation (SMC) [34] and Differential Privacy (DP) [20]. In this section, we introduce the essentials of the secure computation and differential privacy primitives to understand the implementation of SM-DDP algorithms. Particularly, we introduce Homomorphic Encryption (HE) to provide SMC and Functional Mechanism (FM) to provide DP guarantees.

### A. Secure Multiparty Computation

SMC allows the computation of a function with multiple inputs from different users while keeping the users' inputs hidden from each other. For instance, each party $P_i$ in a $n$-party environment holds input $x_i$ learns nothing but the output $f(x_1, ..., x_n)$ of a computation. In the literature, SMC schemes are mostly achieved via either the Yao's garbled circuits [36] or Homomorphic Encryption (HE) [10]. In the following, we use HE to provide guarantees of secure computation.

**Homomorphic Encryption (HE)-** HE provides an ability to evaluate the functions directly on the encrypted data while keeping the data confidential. The primary advantage of the HE is that it does not require any interaction between the parties other than the data exchange. That is, there is

[2] A unique solution only exists if $(\mathbf{X}^\top \mathbf{X})^{-1}$ is non-singular. In other cases, there are techniques for solving Equation 2 [33]; however, it is out of the scope of this paper.

$$\begin{tikzcd}
c_1, c_2 \arrow[r, "Eval(...)"] & f(c_1, c_2) \arrow[d, "Dec_{sk}(...)"] \\
m_1, m_2 \arrow[u, "Enc_{pk}(...)"] \arrow[r, "f(...)"'] & f(m_1, m_2)
\end{tikzcd}$$

Figure 2. HE operations of encryption, evaluation, and decryption (*pk* is the public key, *sk* is the secret key, and *f* is the function desired to be computed).

no additional communication complexity. However, it may introduce computational overhead on large plaintexts. Recent works improved its performance significantly by introducing new techniques like single instruction, multiple data (SIMD) operations [37] or using different mathematical assumptions like learning with errors LWE [38], [39] (see [40] for a recent survey about HE).

An HE scheme is primarily characterized by four operations: key generation (*KeyGen*), encryption (*Enc*), decryption (*Dec*), and evaluation (*Eval*). *KeyGen* is the operation that is used to generate a secret and public key pair for the asymmetric version of HE or a single key for the symmetric version. *KeyGen*, *Enc* and *Dec* are similar to the ones used in conventional encryption schemes. However, *Eval* is an HE-specific operation, which takes ciphertexts as input and outputs a ciphertext corresponding to a functioned plaintext. Fig. 2 illustrates a commutative diagram depicting the relationship among the four major operations. The simplified version of the diagram shows only one homomorphic encryption with two ciphertexts [41].

### *B. Differential Privacy (DP)*

DP is a statistical disclosure control technique ensuring that the outputs of queries do not leak information about the individuals found in a dataset. It injects a certain amount of noise into the replies of the queries so that while it is not possible to infer an individual-level leak, the output of the query is still "almost" the same. In other words, query results of a data release algorithm for two closely similar data sets give the same answer. The formal definition of $\varepsilon-$differential privacy is formulated as follows [42]:

**Definition 1.** *A randomized algorithm $\mathcal{M}$ is $\varepsilon$-differentially private if for all data sets D and $D'$ differing on at most one element and all $S \subseteq Range(\mathcal{M})$,*

$$Pr[\mathcal{M}(D) \in S] \leq \exp(\varepsilon) \times Pr[\mathcal{M}(D') \in S], \qquad (7)$$

*where $Range(\mathcal{M})$ shows all possible outputs of the function (query), $f$.*

The definition states that two adjacent sets $D$ and $D'$, which differs at most one element, act approximately the same against a query[3] defined by a given mechanism $M$. $\varepsilon$ can be considered as the degree of the privacy guarantee and the amount of information which can be learned from a result of a single query is bounded by $\exp(\varepsilon)$. Since $\varepsilon$ is too small, its guarantee is preserved for consecutive queries. Differential privacy works on the release mechanism and does not modify data or the format of the data in any way.

The parameter $\varepsilon$, called *privacy budget*, is the main parameter to tune the balance between privacy and accuracy. Decreasing $\varepsilon$ increases the privacy guarantees while decreasing the accuracy. The common mechanism to control the amount of noise that needs to be added is *Laplace Mechanism* (LM). In this case, the noise is drawn from a Laplace Distribution. The probability density function of LM is as follows:

$$Lap(x|b) = \frac{1}{2b} exp\left(-\frac{|x|}{b}\right), \qquad (8)$$

for scale $b$ and center 0. It is shown that LM preserves $\varepsilon$-differential privacy [42].

**Definition 2.** *Given any function $f : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathbb{R}^k$, the mechanism is a Laplace Mechanism $\mathcal{M}$ if:*

$$\mathcal{M}(x) = f(x) + \eta, \qquad (9)$$

*where $x \in \mathcal{X}$ and $\eta$ is a vector of independent and identically distributed random variables drawn from Lap($\Delta f/\varepsilon$).*

In addition to the $\varepsilon$, *sensitivity* is another important parameter in DP to determine the optimum noise amount. It is defined as follows:

**Definition 3.** *For a function $f : D \rightarrow R^k$, sensitivity of $f$ is*

$$\Delta f = \max_{D,D'} \parallel f(D) - f(D') \parallel \qquad (10)$$

*for all $D, D'$ differing in at most one element.*

The sensitivity shows the maximum number of elements that can change in two different queries.

**Functional Mechanism (FM)-** FM is an algorithm that is used to provide differential privacy guarantees for a set of linear models [11]. It is an extension of the Laplace Mechanism. The goal of the algorithm is injecting the noise to the polynomial coefficients of a model's objective function. This is accomplished with the mechanism of *objective perturbation* [22]. The optimization of the noisy objective function gives new model parameters that ensure the $\varepsilon$-privacy of each element in a database. Algorithm 1 [11] presents the functional mechanism.

**Algorithm 1** [11] Functional Mechanism ($D$, $\mathcal{L}$, $\varepsilon$)

```
Input: Let $\mathcal{L}(f,D) = \sum_{j=1}^{J} \sum_{\phi \in \Phi_j} \sum_{i=1}^{n} \lambda_{\phi_i} \phi(w)$
Set $\Delta = 2 \max_w \sum_{i=1}^{n} ||\lambda_{\phi_i}||_1$
for each $j \in \{0, ..., J\}$ do
    for each $\phi \in \Phi_j$ do
        $\lambda_\phi = \sum_{i=1}^{n} \lambda_{\phi_i} + Lap(\frac{\Delta}{\varepsilon})$    ▷ noise inject
    end for
end for
Compute new $w^* = \arg\min_w \mathcal{L}(f,D)$    ▷ optimize
return $w^*$
```

[3] The queries or functions correspond to the predictions in the statistical models.

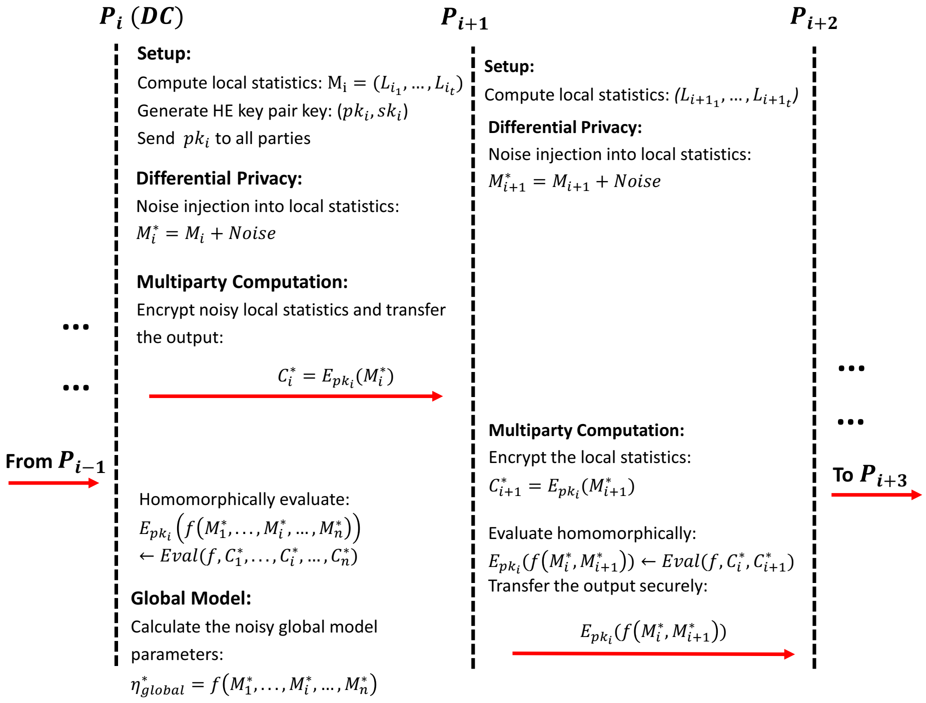
$P_i$ $(DC)$
$P_{i+1}$
$P_{i+2}$
Setup:
Compute local statistics: $M_i = (L_{i_1}, \ldots, L_{i_t})$
Generate HE key pair key: $(pk_i, sk_i)$
Send $pk_i$ to all parties
Differential Privacy:
Noise injection into local statistics:
$M_i^* = M_i + Noise$
Multiparty Computation:
Encrypt noisy local statistics and transfer the output:
$C_i^* = E_{pk_i}(M_i^*)$
From $P_{i-1}$
Homomorphically evaluate:
$E_{pk_i}\left(f\left(M_1^*, \ldots, M_i^*, \ldots, M_n^*\right)\right)$
$\leftarrow Eval(f, C_1^*, \ldots, C_i^*, \ldots, C_n^*)$
Global Model:
Calculate the noisy global model parameters:
$\eta_{global}^* = f\left(M_1^*, \ldots, M_i^*, \ldots, M_n^*\right)$
Setup:
Compute local statistics: $(L_{i+1_1}, \ldots, L_{i+1_t})$
Differential Privacy:
Noise injection into local statistics:
$M_{i+1}^* = M_{i+1} + Noise$
Multiparty Computation:
Encrypt the local statistics:
$C_{i+1}^* = E_{pk_i}(M_{i+1}^*)$
Evaluate homomorphically:
$E_{pk_i}(f(M_i^*, M_{i+1}^*)) \leftarrow Eval(f, C_i^*, C_{i+1}^*)$
Transfer the output securely:
$E_{pk_i}(f(M_i^*, M_{i+1}^*))$
To $P_{i+3}$

Figure 3. Secure Multiparty Distributed Differentially Private (SM-DDP) protocol for the computation of a linear model coefficients. The parties create a ring topology and the Data Collector (DC) initiates the protocol. The protocol can be applied to any statistical model function that allows independent calculation of local statistics.

As illustrated in Algorithm 1, FM takes a dataset $D$, the polynomial representation of the objective function $L$, and the privacy budget $\varepsilon$ as inputs and it returns the differentially private model coefficients $w^*$. It firstly injects noise drawn from a Laplacian distribution ($Lap(\frac{\Delta}{\varepsilon})$) into all the coefficients $\lambda_{\phi_i}$ of the polynomial representation of the objective function and then the optimization is performed using noisy coefficients. It is shown that it satisfies $\varepsilon$-differential privacy [11] i.e., the predictions using $w^*$ does not leak any information about an individual in the database data. For example, if we have a quadratic objective function in the matrix form of $w^\top \mathcal{P} w + w^\top \mathcal{V} + \mathcal{O}$, where $\mathcal{P}$, $\mathcal{V}$, and $\mathcal{O}$ are the coefficients of the polynomial representation of the objective function. FM firstly injects noise into the coefficients, which results in $w^\top \mathcal{P}^* w + w^\top \mathcal{V}^* + \mathcal{O}^*$. Then, the optimization problem (i.e., $w^* = \arg\min_w \mathcal{L}(f, D)$) is solved using $\mathcal{P}^*$, $\mathcal{V}^*$, and $\mathcal{O}^*$.

## V. Secure and Differentially-private Distributed Computations

In this section, we propose a novel protocol for secure multiparty distributed and differentially private (SM-DDP) computations through the use of homomorphic encryption (HM) and functional mechanism (FM). We evaluate its application to linear regression and discuss its extension to the logistic regression that can be used in supervised classification.

Consider $n$ parties $P_1, \ldots, P_n$, where each has private horizontally distributed database $D_1, \ldots, D_n$. Each database consists of a certain number of tuples in the format of $t_i = (x_i, y_i)$. The parties would like to jointly build a linear model of the pooled database $f(D)$, where $D = \cup_{i=1}^{n} D_i$ so that the security guarantees of both SMC and DP are preserved. Before running the protocol, each party in the collaboration agrees on the function to be computed and compute a collection of local statistics $M_i = (L_{i_1}, \ldots, L_{i_t})$. We assume the linear model can be computed using the local statistics generated by each party independently i.e., $\eta_{global} = f(M_1, \ldots, M_i, \ldots, M_n)$. We define the guarantees and goals of our protocol as follows:

- *Individual privacy:* No information leaks about the individuals in the private databases held by the parties, i.e., tuples $t_i$ is not leaked.
- *Data privacy:* Information about the statistics of the data does not leak in the databases held by the parties, i.e., the statistics about the data $M_i$ is not leaked.
- *Correctness:* The parties receive the correct output of the model.

We note that using SMC only would violate the individual privacy while using DP only violates the data privacy. In our combined protocol, we achieve individual privacy through FM and data privacy through HE and since all operations in the protocol are deterministic, the correctness is satisfied by design. We note that we assume there is a secure channel between parties to exchange messages.

Fig. 3 illustrates our protocol to be able to perform SM-DDP computations. It is initiated by one of the parties called *data collector* (DC). In the setup phase, DC generates a key pair $(pk_i, sk_i)$ and computes its own local statistics $M_i$ independent from other parties. Then, in the next phase, DC applies DP by injecting (adding) noise drawn from a random distribution that satisfies $\varepsilon$-differential privacy into its local statistics. The encryption of the noisy local statistics is transmitted to the

**Algorithm 2** Computation of Linear Regression using SM-DDP protocol

```
Input: Each party holds a database in the format of $D_i = (x_i, y_i)_{i=1}^n$ i.e., horizontally partitioned
       The global privacy budget $\varepsilon$.
Output: The differentially private global regression model of $D = \cup_{i=1}^n D_i$

Setup: Runs at the party $P_i$ (DC)
$(pk_i, sk_i) \leftarrow KeyGen()$        ▷ generate the key pair of HE
$\eta_{max}$, $\eta_{min} \leftarrow ComputeMinMax(D)$        ▷ calculate the global max and min of each attribute via [43]
$\Delta \leftarrow 2(d+1)^2$        ▷ calculate the global sensitivity, $d$ is the number of attributes

Secure Regression Protocol: each party $P_j$ runs locally
Input: Received aggregate statistics for all previous parties as:
    $\xi$: $E_{pk_i}(\sum_{k=1}^{j-1} \mathcal{P}_k^*)$
    $\kappa$: $E_{pk_i}(\sum_{k=1}^{j-1} \mathcal{V}_k^*)$
    $\delta$: $E_{pk_i}(\sum_{k=1}^{j-1} \mathcal{O}_k^*)$

$D_j^{norm} \leftarrow (D_j - \eta_{min})/(\eta_{max} - \eta_{min})$        ▷ perform min-max normalization
$\mathcal{P}_j \leftarrow \mathbf{X}_j^\top \mathbf{X}_j$, $\mathcal{V}_j \leftarrow \mathbf{X}_j^\top \mathbf{Y}_j$, and $\mathcal{O}_j \leftarrow \mathbf{Y}_j^\top \mathbf{Y}_j$        ▷ compute local statistics
$\varepsilon_i \leftarrow \alpha\varepsilon$        ▷ compute its share from the global privacy budget
$(\mathcal{P}_j^*, \mathcal{V}_j^*, \mathcal{O}_j^*) \leftarrow FM.NoiseInject(\mathcal{P}_j, \mathcal{V}_j, \mathcal{O}_j)$        ▷ apply FM noise injection
$C_j^* = \big(E_{pk_i}(\mathcal{P}_i^*), E_{pk_i}(\mathcal{V}_i^*), E_{pk_i}(\mathcal{O}_i^*)\big)$        ▷ perform encryption
        ▷ add its own encrypted local statistics to the received aggregate statistics
$E_{pk_i}(\sum_{k=1}^{j} \mathcal{P}_k^*) \leftarrow E_{pk_i}(\mathcal{P}_j^*) + \xi$
$E_{pk_i}(\sum_{k=1}^{j} \mathcal{V}_k^*) \leftarrow E_{pk_i}(\mathcal{V}_j^*) + \kappa$
$E_{pk_i}(\sum_{k=1}^{j} \mathcal{O}_k^*) \leftarrow E_{pk_i}(\mathcal{O}_j^*) + \delta$
Send( $E_{pk_i}(\sum_{k=1}^{j} \mathcal{P}_k^*)$, $E_{pk_i}(\sum_{k=1}^{j} \mathcal{V}_k^*)$, $E_{pk_i}(\sum_{k=1}^{j} \mathcal{O}_k^*)$ ) to $P_{j+1}$        ▷ send updated aggregate statistics to the next party.

Reconstruction: runs at the party $P_i$ (DC)
Input: Received aggregate statistics for all parties as:
    $\xi$: $E_{pk_i}(\sum_{k=1}^{n} \mathcal{P}_k^*)$
    $\kappa$: $E_{pk_i}(\sum_{k=1}^{n} \mathcal{V}_k^*)$
    $\delta$: $E_{pk_i}(\sum_{k=1}^{n} \mathcal{O}_k^*)$

$\mathcal{P}^* \leftarrow D_{sk_i}\Big(\xi\Big)$        ▷ acquire the cleartext
$\mathcal{V}^* \leftarrow D_{sk_i}\Big(\kappa\Big)$        ▷ acquire the cleartext
$\mathcal{O}^* \leftarrow D_{sk_i}\Big(\delta\Big)$        ▷ acquire the cleartext
$(\mathcal{P}^*, \mathcal{V}^*, \mathcal{O}^*) \leftarrow FM.Optimize(\mathcal{P}^*, \mathcal{V}^*, \mathcal{O}^*)$        ▷ apply optimization
$w^* \leftarrow \mathcal{P}^{*-1}\mathcal{V}^*$ (i.e., $w^* = \arg\min_w w^\top \mathcal{P}^* w + w^\top \mathcal{V}^* + \mathcal{O}^*$)        ▷ compute the global parameters
$Err \leftarrow w^{*\top} \mathcal{P}^* w^* + w^{*\top} \mathcal{V}^* + \mathcal{O}^*$
Publish( $w^*$, $Err$ ) to all parties.
        ▷ Use of Model:
$f(x_i, w^*) \leftarrow \sum_{i=1}^n x_i {w^*}_i$ for an input $x_i \in \mathbf{X}_i$        ▷ computes the normalized predictions
$y_{pred} \leftarrow f(x_i, w^*)(\eta_{max} - \eta_{min}) + \eta_{min}$        ▷ perform de-normalization to get actual values
```

next party $P_{i+1}$. The next party $P_{i+1}$ also computes its local statistics and injects noise into them. The result is encrypted with $pk_i$ and the function is evaluated homomorphically with the inputs of parties $P_i$ and $P_{i+1}$. The protocol is continuous in the same way, where parties are located in a ring topology. At the final step, the securely evaluated function result is used by the party $P_i$ which decrypts it with $sk_i$. In the end, $P_i$ reveals the differentially private global model.

### A. Case Study: Linear Regression

In this subsection, we show how to compute linear regression using our protocol proposed in Fig. 3. Particularly, we use functional mechanism shown in Algorithm 1 by splitting it into two parts: $NoiseInject()$ and $Optimize()$. In $NoiseInject()$, the noise drawn from Laplacian distribution (Equation 8) is injected into each coefficient of the polynomial representation of the objective function. Then, in $Optimize()$, the optimization problem of the objective function is solved by applying regularization and spectral trimming introduced in [11] in order to avoid unbounded noisy objective function. Moreover, in FM, it is assumed that $\sqrt{\sum_{i=1}^{d} x_{id}^2} \leq 1$. Therefore, a secure maximum computation is performed to calculate $\eta_{min}$ and $\eta_{max}$ in setup phase of Algorithm 2, where $\eta_{min}$ (resp. $\eta_{max}$) is vector consists of global minimum (resp. maximum) of each attribute. Before applying FM, each party normalizes its database using the global maximum and minimum values. This guarantees that the local sensitivity of the parties is always same as the global sensitivity as we focus on the horizontally distributed data.

Algorithm 2 illustrates the computation of linear regression algorithm using the protocol presented in Fig. 3. In linear regression, the global model is calculated by simply aggregating locally calculated noisy statistics. While aggregating the local statistics, the noise of each party is aggregated as well. Therefore, it is necessary to make sure the final model will not violate ε-differential privacy nor cause an unbounded noise. Particularly, the noise is injected to each coefficient as follows:

$$\mathcal{P}_i{}^* = \mathcal{P}_i + Lap\Big(\frac{\Delta}{\varepsilon_i}\Big). \qquad (11)$$

Then, when DC computes the global model, the local statistics are summed up as follows:

$$\mathcal{P}^* = \sum_{i=1}^{n} \mathcal{P}_i{}^* = \sum_{i=1}^{n} \left(\mathcal{P}_i + Lap\Big(\frac{\Delta}{\varepsilon_i}\Big)\right) = \mathcal{P} + \sum_{i=1}^{n} Lap\Big(\frac{\Delta}{\varepsilon_i}\Big). \quad (12)$$

Moreover, $\mathcal{V}^*$ and $O^*$ can be computed similarly. In all $\mathcal{P}^*$, $\mathcal{V}^*$, and $O^*$, the noise term is $\sum_{i=1}^{n} Lap\big(\frac{\Delta}{\varepsilon_i}\big)$. In order to make sure that the accumulated noise is also Laplacian distribution, we use the following theorem.

**Theorem 1.** *Let $Y$, $Y_1$, $Y_2$... be non-degenerate and symmetric i.i.d. random variables with variance $\sigma^2 > 0$, and let $\nu_p$ be a geometric random variable with mean $1/p$, independent of the $Y_i$'s. Then, the following statements are equivalent (Proof is given in [12]):*
*(i) $Y$ is stable with respect to geometric summation, i.e., there exist constants $a_p > 0$ and $b_p \in \mathbb{R}$, such that*

$$a_p \sum_{i=1}^{\nu_p} (Y_i + b_p) = Y \quad \forall p \in (0,1) \qquad (13)$$

*(ii) $Y$ possesses the Laplace distribution with mean zero and variance $\nu_2$. Moreover, the constants $a_p$ and $b_p$ must be of the form: $a_p = \sqrt{p}$, $b_p = 0$*

From the theorem above, a Laplace distribution can be calculated by summing up several Laplace distributions in a certain form. In other words, the sequence of partial sums, $a_p \sum_{i=1}^{\nu_p}(Y_i + b_p)$ converges to a Laplace distribution under beta-distributed $a_p$. We addressed requirements of the theorem in Algorithm 2 by multiplying the noise distribution of local parties with a number drawn from the geometric distribution i.e., $a_p \sum_{i=1}^{n} Lap\big(\frac{\Delta}{\varepsilon_i}\big)$, where $a_p$ is a geometric random variable.

## VI. Performance Evaluation

In this section, we give the experimental results for the application of our SM-DDP protocol to linear regression. Table I presents the notations used throughout the experiments. We first demonstrate how we set the parameters that are introduced in the distributed setting. Particularly, the success probability of the geometric random variable $p$ in Equation 13 and $\alpha$ introduced in Algorithm 2 is investigated. After experimentally tuning these two parameters, we test the final protocol with a different dataset without random sampling directly as it is collected. During evaluation, we focus on the following questions: (i) Can we obtain a differentially private global linear regression model from differentially private local statistics? (ii) Does our approach support up to 100 parties? (iii) How long does it take to complete the protocol? (iv) Does it guarantee the security and privacy of both data and individuals? We analyzed and discussed each of these questions in Sections VI-A-VI-D.

TABLE I. Abbreviations and notations used in experiments

| Notation | Description | Range |
|---|---|---|
| DDP | Distributed Differential Privacy | - |
| NoDP | No Differential Privacy | - |
| CDP | Centralized Differential Privacy | - |
| $\varepsilon$ | global privacy budget | $\{0.1,0.2,0.4,0.8,1.6,3.2,6.4,12.8\}$ |
| $\varepsilon_i$ | local privacy budget | $\varepsilon_i = \alpha\varepsilon$ |
| $\alpha$ | local privacy ratio i.e., $\alpha = \varepsilon_i/\varepsilon$ | $\{1,10,100\}$ |
| p | success probability of the geometric random variable, $a_p$ | $\{0.1,0.5,0.9\}$ |
| n | number of parties | $[1,100]$ |
| L | number of levels in HElib | $\{4,6\}$ |
| *nslots* | number of slots in HElib | calculated by HElib |
| s | minimum of *nslots* | $\{8^2,16^2,24^2,32^2,40^2\}$ |

**Dataset-** We used two real-world datasets to evaluate the algorithms of our protocol. Both datasets include highly sensitive data. The first dataset is *Integrated Public Use Microdata Series* (IPUMS) [44]. It contains 370K decennial census records of people living in the US with 14 attributes, 7 of which are demographic information and the rest are working hours per week, the number of years residing in the current location, the number of children, the number of automobiles, and the annual income. The attributes are used to predict the *annual income* of a person. The second dataset is the warfarin dataset collected by the International Warfarin Pharmacogenetics Consortium (IWPC) [45]. The dataset contains clinical and genetic data of patients to predict the stable therapeutic dose of warfarin. Clinical data includes demographics, background, and phenotypic attributes. Genetic data includes genotype variants of CYP2C9 (*1, *2 and *3) and VKORC1 (one of seven single nucleotide polymorphisms in linkage disequilibrium). 21 sites in 9 countries and four continents contributed to the dataset. We used a subset of this dataset wherein patient samples include no missing attributes. Overall, we used 1400 complete patient samples from seven medical institutions. We used IPUMS dataset to experimentally set the parameters of our protocol and we tested the final protocol with the IWPC dataset, where each party corresponds to a medical institution in the dataset.

**Evaluation Metrics-** We applied stratified cross validation to split the dataset into training and test sets. To evaluate the model's prediction accuracy, we used *Mean Squared Error* (MSE) as it is a commonly used metric for linear regression analysis. It is calculated as $\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)$, which gives the average of the squared errors between actual ($y_i$) and predicted ($\hat{y}_i$) values in $n$ data samples. The lower values of MSE shows better predictions. Finally, it is worth mentioning that all the experiments show 100 independent runs and their average is reported in this work.

**Experimental Setup-** To evaluate the computational overhead, we used open-source HE library (HElib) [46], which implements BGV homomorphic cryptosystem [38] and we ran experiments on 16-core Intel Xeon CPU at 1.90 GHz running Linux Server. In BGV, a prior level $L$ should be set before initiating the computation. In addition to the level $L$, HElib also has a parameter *nslots* which defines a number of slots for

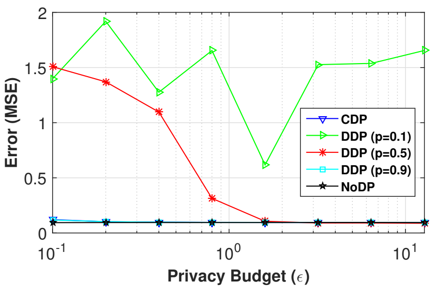

Figure 4. Tuning $p$ . Variation of error is tested for several values of $p$. As a result, $p = 0.1$ is not stable or convergent; $p = 0.5$ is convergent, but error is much higher than CDP for especially small ε values. Hence, we chose $p = 0.9$ as the best case.

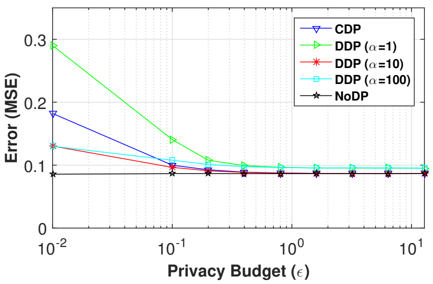

Figure 5. Tuning $\varepsilon_i$. Variation of error is tested for several values of local privacy budget $\varepsilon_i$ for $\alpha = \varepsilon_i/\varepsilon$. For $\alpha = 1$, error is too high for small ε values. For $\alpha = 10$, error is lower than CDP and and it converging to the value as NoDP. For $\alpha = 1$, error is low, but it converges to a value higher than NoDP. Hence, we chose $\alpha = 10$ as the best case.

the utilization of SIMD techniques [37], [47]. HElib allows encrypting multiple messages at one time through its SIMD features by packing the messages into the independent slots of an array. We note that the parameter $L$ affects not only the number of allowed homomorphic operation but also all the other timings and the key size. Therefore, the parameter $L$ should be optimized so that the minimum $L$ is set without failure of the decryption. To do so, we first calculated the table of a number of homomorphic operations for each level $L$ and we used the minimum level for each number of the party.

Furthermore, in our experiments, the data encrypted is the local statistics i.e., not the raw data. The size of the local statistics is considered the same for all the parties. The homomorphic operation computed for linear regression is the element-wise matrix addition. To take advantage of HElib library SIMD features, we converted matrices into arrays and the parameter of minimum number for *nslots* was set to the length of the array for each statistics. This prevents data loss during the conversion. We did not utilize any multi-threading technique during our experiments to see the lower bound of the performance of our protocol. Thus, our results are lower bound and can be improved with the use of any multi-threading technique.

### A. Accuracy Analysis

We evaluate the accuracy-privacy trade-off of distributed evaluation of differential privacy on linear regression. Specifically, we compare our results with the centralized approach. In *Centralized Differential Privacy* (CDP), the accuracy of the regression depends only on the global privacy budget ε. However, in *Distributed Differential Privacy* (DDP), each party has its own local privacy budget $\varepsilon_i$ and DDP is applied independently by each party. We note that this is a particular property of FM. In FM, data is first normalized and the optimum noise amount is only determined by the number of the attributes which is same for all parties. Therefore, the size and the range of the local statistics are same for all the parties; it does not depend on the number of tuples in the local database. Since all parties are identical, we choose the same local privacy budget $\varepsilon_i$ for all the parties. Finally, in our fist three experiments (Fig. 4, 5, and 7), we used IPUMS dataset and split it into parties using random sampling methods. In the last experiment, we used IWPC dataset for accuracy evaluation. We split the dataset based on the given medical institutions (See Fig. 6)

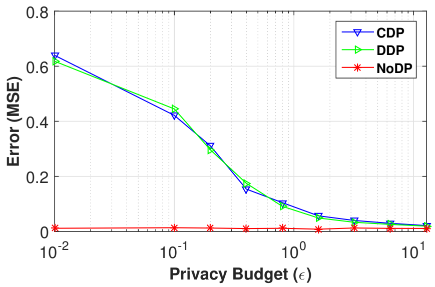

Figure 6. A real test: Warfarin dataset with 7 parties with $\varepsilon_i = n\varepsilon$ and $p = 0.9$. Exactly the same trade-off as the centralized differential privacy is obtained.

The first set of experiments was conducted to analyze the optimum value of $p$, which is a parameter of geometric random variable $a_p$ given in Equation 13. In theory, $a_p$ is required to obtain a Laplace distribution in the global model, thereby it is required to be able to satisfy ε-differential private model. To present the impact of the parameter $p$ on the accumulated global noise, we kept the party number constant for several values of $p$ and various ε values ($\varepsilon_i = \varepsilon$). To do so, each party multiplies the noise drawn from Laplace distribution with a random variable $a_p$, which is a geometric random variable with success probability $p$. We compared the error rates of CDP, DDP, and NoDP algorithms in terms of MSE.

Fig. 4 illustrates the error and privacy budget trade-off for various values of $p$. We varied $p$ from $\{0.1, 0.5, 0.9\}$. We found that DDP with $p = 0.1$ does not converge to a value while increasing the value of ε. However, $p = 0.5$ and $p = 0.9$ converges to the same value as NoDP as it is desired and when $p$ is 0.9, it gives similar results to CDP. In the sequel, we tuned $p = 0.9$ and used it in our experiments.

In the second set of experiments, we were interested in finding the optimal local privacy budget $\varepsilon_i$ for a predetermined global privacy budget. In other words, we assume all parties agree on a global privacy budget according to the sensitivity of the dataset, which was indeed calculated by the number of attributes. We denote the ratio of local privacy budget to the global privacy budget as α, i.e., $\alpha = \varepsilon_i/\varepsilon$. We first tried the value of α less than 1, the result of DDP was much worse than CDP. This is because smaller $\varepsilon_i$ means more noise injected locally by each party than the centralized approach. This noise

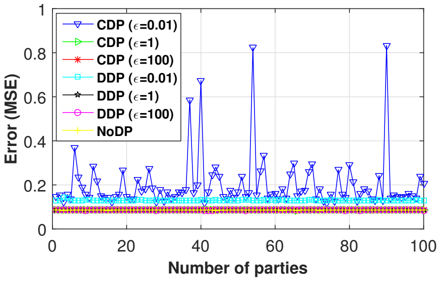

Figure 7. Impact of number of parties in the collaboration for $\varepsilon_i = n\varepsilon$ and $p = 0.9$.

decreases the accuracy significantly. Therefore, we changed $\alpha$ from $\{1, 10, 100\}$ and compared the results with CDP and NoDP mechanisms. The results are presented in Fig. 5. We found that if $\alpha$ is the number of parties, which is 10 in this experiment, the plot gets closer to CDP and the error is converging to NoDP, which is the desired case. Therefore, in the rest of experiments, we set $\alpha = n$, where $n$ is the number of parties.

So far, we tuned the parameters of our approach experimentally. Now, in our last experiment, we evaluated the efficiency of our protocol using the dataset (IWPC dataset) collected from multi sources. We applied DP locally on each party's dataset and calculated the global model and error. Our goal was to see the feasibility of our approach in a real case and test the feasibility of our approach.

In this experiment, we set $\varepsilon_i = n\varepsilon$, $p = 0.9$ as we found in earlier experiments. We compared the performance of CDP, DDP, and NoDP algorithms. Fig. 6 shows MSE rates for varying $\varepsilon$. We found that the same trade-off with CDP can be achieved by applying DP while training the classifiers locally. We note the DDP is also converging to the error of NoDP when $\varepsilon$ approaches infinity as desired.

### B. Scalability Analysis

In this set of experiments, we evaluated the scalability of our proposed protocol. We set $\varepsilon_i = n\varepsilon$, where $n$ is the number of parties; as we found $\alpha = n$ is optimum and for a different number of parties, we split the dataset into the number of parties ($n$) by using random sub-sampling. Then, each party applies DP locally, but we note that the pooled dataset is still the same.

Laplace distribution is infinitely divisible [12]. Therefore, the accumulated error of global model should not be affected by the number of parties. We ran the analysis for some users ranging from 1 to 100 and present the results in Fig. 7. The results demonstrated an interesting point, which is when $\varepsilon = 0.01$, even though CDP is not stable, DDP is. On the other hand, when $\varepsilon$ is 1 or 100, the error rate stays the same even for 100 parties. This means our protocol is scalable even for 100 parties.

### C. Computational Overhead Analysis

In this subsection, we evaluate the computational overhead of linear regression presented in Algorithm 2. We found that DP algorithms do not introduce computational overhead. Therefore, we only evaluate the computational overhead of our SMC algorithm, which consists of three main parts: Key generation of HE, min-max, and regression computation.

Fig. 8 shows the computation time for different dimension sizes. Fig. 8a presents the time for secure computation of finding global min-max of each attribute. It increases quadratically with the number of parties. However, this algorithm runs at the setup phase, so it is performed before initiating the computations. There are two interesting results worth to note. First, the time of secure regression computation increases linearly as a number of parties in the collaboration increases, but with a different slope for dimension, which is illustrated in Fig. 8. The reason for the linear increase is that the number of encryptions and homomorphic evaluations are directly scaled by the number of parties in the group. Second, similar results hold for the overall computation time (see Fig. 8c), but as a minor change since the key generation time shifts the lines in the y-axis and also increases the scale. However, similar to the secure min-max computation, the execution of the key generation algorithm does not require all parties in the group to be online since it occurs in the setup phase. On the other hand, we also note that size of the local database of each party does not have an impact on the total computational time since parties only share the local statistics, which is dependent on the attribute size, instead of the raw data. As can be seen in Fig. 8c, the overall computation of the protocol including both offline and online phases for 20 parties with 32 attributes and 10K samples is less than a minute. Hence, our SM-DDP protocol yields minimal computational overhead.

### D. Security and Privacy Analysis

In this section, we discuss the security and privacy guarantees of SM-DDP protocol given in Fig. 3. As all the communication among the parties is encrypted, the security of the algorithm is simply reduced to the security of underlying HE scheme. A leak can occur only if DC is corrupted since the data is encrypted using the public key generated by DC. However, even in this case, DC will only obtain the noisy local statistics, not the raw data, and at the end of the protocol, DC has only control over the aggregated data while reconstructing the global model and it can not know which party contributed to the result. While the protocol is running, the view of all the other parties consists of homomorphically encrypted data. Therefore, if the given homomorphic encryption scheme is semantically secure, the parties can not distinguish the corresponding plaintexts. So, the computation is private even in the presence of an honest, but curious adversary model presented in [48]. Therefore, data privacy is preserved.

On the other hand, we both showed theoretically (Section V-A) and experimentally (Fig. 6), a differentially private global model can be obtained through the locally applied DP. Therefore, it is not possible that an untrusted data collector can infer information about the individuals. Furthermore, the collaboration comes with a price as the local parties used $\varepsilon_i$ instead of $\varepsilon$. Therefore, the local privacy guarantee is decreased by $\alpha$ (i.e., $\varepsilon_i$ is increased by $\alpha$), even though the global model's guarantee is still the same, meaning that data privacy against an untrusted DC is still preserved and the local privacy guarantee is important only if the underlying SMC is bypassed. Finally, since we set $\alpha$ as the number of parties in the collaboration, each party should take this into consideration while deciding on the global privacy budget.

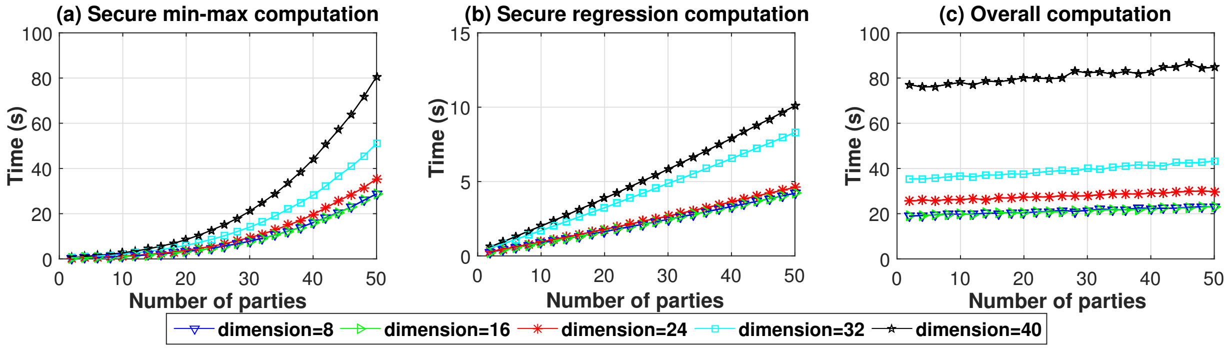

Figure 8. Performance evaluation of SM-DDP computations of linear regression algorithm.

## VII. Discussion

The preceding analysis showed how to achieve secure multiparty computation and differential privacy in distributed settings focusing on linear regression on horizontally distributed data. That is, parties do not see each others' inputs and further can not infer individuals' data from the final constructed model. A limitation of our algorithm is that we assume parties do not collaborate to learn a target party's input. However, if the party that generates the key pair conspires with the parties that are neighbors of a target in the ring topology, the noisy local statistics ($\xi$, $\kappa$, $\delta$) of the victim can be extracted. More generally, this is known as *active corruption*, where the data collector is an active attacker and has control over the other corrupted parties. Our protocol in Fig. 3 achieves only a collusion threshold of 1, but the distributed DP algorithm that we present here can easily be adapted to work with recent solutions in SMC such as [34], which is secure in the presence of an active adversary corrupting up to $n-1$ of the $n$ parties. To extend our work with these more secure SMC schemes, it suffices to use the noisy output of the functional mechanism instead of using the local statistics directly as input to the underlying SMC algorithm.

In our evaluation, we used HElib, an implementation of the fully homomorphic operation, to compute generic results. It supports both addition and multiplication; however, while computing the linear regression coefficients, we only used the addition operation. The performance of secure computation can be improved by using other libraries such as Paillier cryptosystem [49], which is only additively homomorphic cryptosystem.

Finally, our algorithms can be easily extended to other algorithms such as logistic regression in a supervised classification setting. In logistic regression, each party independently computes a score vector $u_i$ and information matrix $I_i$. Instead of injecting noise to the local statistics as in linear regression, noise can be injected into $u_i$ and $I_i$ vectors. However, the optimization of objective function differs in logistic regression as it requires several iterations. Fortunately, there exist some techniques that let implementing the iterations for computing the secure multi-site logistic regression [7]. Combining this secure multi-site logistic regression algorithm with FM would solve this issue. We defer the detailed application of this method to future work.

## VIII. Conclusion

In this work, we have proposed a novel Secure Multiparty Distributed Differentially Private (SM-DDP) protocol to achieve private computations in a multiparty environment as an application in linear regression. Using homomorphic encryption and functional mechanism, we first presented a protocol to provide the guarantees of secure multiparty computation and differential privacy. Then, we built the algorithms that would allow distributed parties to compute a global model while preserving the privacy of their data and individuals found in the dataset. Any statistical model function that can be independently calculated by sharing the local statistics of the parties can be computed through this protocol. Finally, we evaluated the performance of the proposed protocol on two datasets, namely, warfarin dose and budget predictions. Our findings show that a party can achieve individual-level privacy via our proposed protocol for distributed differential privacy, which is independently applied by each party in a distributed fashion. Moreover, the experiment results demonstrated that the proposed SM-DDP protocol is both feasible and scalable that is its computational overhead is minimal and overall computation time is sub-linear with the number of parties. Indeed, SM-DDP protocol provides security and privacy guarantees while being feasible and scalable. Our future work will extend the algorithms outside the linear models and investigate the accuracy and performance trade-offs of other algorithms. We are also planning to compare the performance of Laplacian mechanism used in FM with other DP mechanisms such as Exponential Mechanism [21] and Sample-and-aggregate [50].

## Acknowledgment

This work was partly supported by US NSF-CAREER-CNS-1453647 and Army Research Laboratory under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## References

[1] E. Kimura *et al.*, "Evaluation of secure computation in a distributed healthcare setting." *Studies in health technology and informatics*, 2016.

[2] Z. B. Celik, D. Lopez-Paz, and P. McDaniel, "Patient-driven privacy control through generalized distillation," in *arXiv:1611.08648*, 2016.

[3] Z. B. Celik, H. Aksu, A. Acar, R. Sheatsley, A. S. Uluagac, and P. D. McDaniel, "Curie: Policy-based secure data exchange," 2017. [Online]. Available: http://arxiv.org/abs/1702.08342

[4] D. Bogdanov, R. Talviste, and J. Willemson, "Deploying secure multi-party computation for financial data analysis," in *Financial Cryptography and Data Security*, 2012.

[5] J. Freudiger *et al.*, "Controlled data sharing for collaborative predictive blacklisting," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2015.

[6] Z. B. Celik *et al.*, "Extending detection with forensic information," in *arXiv:1603.09638*, 2016.

[7] K. El Emam *et al.*, "A secure distributed logistic regression protocol for the detection of rare adverse drug events," *Journal of the American Medical Informatics Association*, 2013.

[8] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *IEE Security and Privacy*. IEEE, 2008.

[9] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith, "Composition attacks and auxiliary information in data privacy," in *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008.

[10] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[11] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: regression analysis under differential privacy," *VLDB*, 2012.

[12] S. Kotz, T. Kozubowski, and K. Podgorski, *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. Springer Science & Business Media, 2012.

[13] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, "Secure regression on distributed databases," *Journal of Computational and Graphical Statistics*, 2005.

[14] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *SIAM International Conference on Data Mining*, 2004.

[15] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, "Privacy-preserving analysis of vertically partitioned data using secure matrix products," *Journal of Official Statistics*, 2009.

[16] A. P. Sanil, A. F. Karr, X. Lin, and J. P. Reiter, "Privacy preserving regression modelling via distributed computation," in *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[17] R. Hall, S. E. Fienberg, and Y. Nardi, "Secure multiple linear regression based on homomorphic encryption," *Journal of Official Statistics*, 2011.

[18] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Theory and Applications of Cryptographic Techniques*, 2006.

[19] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*, 2006.

[20] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation*, 2008.

[21] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Foundations of Computer Science*, 2007.

[22] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, 2011.

[23] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Foundations of Computer Science (FOCS)*, 2014.

[24] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Foundations of Computer Science (FOCS)*. IEEE, 2013.

[25] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing." in *USENIX Security*, 2014.

[26] P. Jain and A. Thakurta, "Differentially private learning with kernels." *ICML*, 2013.

[27] A. Smith, A. Thakurta, and J. Upadhyay, "Is interaction necessary for distributed private learning?" in *IEEE Symposium on Security and Privacy*, 2017.

[28] S. Goryczka, L. Xiong, and V. Sunderam, "Secure multiparty aggregation with differential privacy: A comparative study," in *Joint EDBT/ICDT 2013 Workshops*, 2013.

[29] C. Clifton and B. Anandan, "Challenges and opportunities for security with differential privacy," in *International Conference on Information Systems Security*, 2013.

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

[31] M. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers," in *Advances in Neural Information Processing Systems*, 2010.

[32] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang, "Fast and secure linear regression and biometric authentication with security update." *IACR Cryptology ePrint Archive*, vol. 2015, 2015.

[33] M. Mohri, "Introduction to machine learning lecture 15."

[34] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology–CRYPTO 2012*, 2012.

[35] F. K. Dankar, "Privacy preserving linear regression on distributed databases," *Transactions on Data Privacy*, 2015.

[36] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science*, 1982.

[37] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," in *Designs, codes and cryptography*, 2014.

[38] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "fully homomorphic encryption without bootstrapping," in *Theoretical Computer Science*, 2012.

[39] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," *SIAM Journal on Computing*, 2014.

[40] A. Acar, H. Aksu, A. Selcuk Uluagac, and M. Conti, "A Survey on Homomorphic Encryption Schemes: Theory and Implementation," *ArXiv e-prints*, Apr. 2017.

[41] C. Gentry, "Computing on the edge of chaos: Structure and randomness in encrypted computation." in *Electronic Colloquium on Computational Complexity (ECCC)*, 2014.

[42] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, 2014.

[43] G. S. Çetin, Y. Doröz, B. Sunar, and E. Savaş, "Depth optimized efficient homomorphic sorting," in *International Conference on Cryptology and Information Security in Latin America*, 2015.

[44] IPUMS-International, "Harmonized international census data for social science and health research," https://international.ipums.org/international/.

[45] I. W. P. Consortium *et al.*, "Estimation of the warfarin dose with clinical and pharmacogenetic data," *New England Journal of Medicine*, 2009.

[46] E. C. Report, "An implementation of homomorphic encryption," https://github.com/shaih/HElib, [Online; accessed 01-January-2017].

[47] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *International Workshop on Public Key Cryptography*, 2010.

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

[49] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptologyEUROCRYPT99*. Springer, 1999, pp. 223–238.

[50] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth sensitivity and sampling in private data analysis," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM, 2007, pp. 75–84.