# Fast, Privacy Preserving Linear Regression over Distributed Datasets based on Pre-Distributed Data [*]

Martine De Cock
University of Washington Tacoma
martine@uw.edu

Rafael Dowsley
Karlsruhe Institute of Technology
rafael.dowsley@kit.edu

Anderson C. A. Nascimento
University of Washington Tacoma
andclay@uw.edu

Stacey C Newman
University of Washington Tacoma
newmsc8@uw.edu

## ABSTRACT

This work proposes a protocol for performing linear regression over a dataset that is distributed over multiple parties. The parties will jointly compute a linear regression model without actually sharing their own private datasets. We provide security definitions, a protocol, and security proofs. Our solution is information-theoretically secure and is based on the assumption that a Trusted Initializer pre-distributes random, correlated data to the parties during a setup phase. The actual computation happens later on, during an online phase, and does not involve the trusted initializer. Our online protocol is orders of magnitude faster than previous solutions. In the case where a trusted initializer is not available, we propose a computationally secure two-party protocol based on additive homomorphic encryption that substitutes the trusted initializer. In this case, the online phase remains the same and the offline phase is computationally heavy. However, because the computations in the offline phase happen over random data, the overall problem is embarrassingly parallelizable, making it faster than existing solutions for processors with an appropriate number of cores.

## Keywords

Secure Machine Learning, Private Linear Regression, Unconditional Security, Commodity Based Model

## 1 Introduction

Linear regression is a technique for modelling the relationship between one or more input variables – often called explanatory variables – and a real valued outcome. It is widely used as a tool for statistical analysis and is very popular as a technique to build predictive models in machine learning (see e.g. [30]). Linear regression models owe their popularity to various factors, including the fact that they can be trained efficiently, are intuitive, and fit the data reasonably well for many learning problems. In addition, the high inductive bias caused by the simplicity of the model helps prevent it from overfitting to a specific set of training examples. Furthermore, linear regression models do not require the outcome variable to be linear in terms of the input variables; they are simply linear in terms of the weights associated to each of the input variables.

Techniques for constructing a linear regression model, like other traditional machine learning (ML) methods, require that all training attributes and tuples be accessible to the learning algorithm. As storage and computing becomes increasingly distributed and heterogeneous, data movement and transformations become non-trivial, making conventional ML algorithms difficult to apply. Moreover, multiple parties often cannot or will not share their data due to economic incentives or privacy legislation, creating a dire need for privacy-preserving ML techniques.

The importance of such techniques is immediately apparent in ML applications in security-sensitive industries such as the financial sector and electronic surveillance. In the former, a bank may want to hire an analytics company to mine the data of its customers but, being bound by the customer agreement, cannot simply hand over the data to that company. In the latter, Internet service providers wanting to have a consulting firm do traffic analysis on their logs may be unwilling to disclose details about their customer base in the process. Another prominent example is the healthcare ecosystem. In addition, it is widely acknowledged that big data analytics can revolutionize the healthcare industry, among other things, by optimizing healthcare spending at

AISec'15, October 16 2015, Denver, CO, USA
©2015 ACM. ISBN 978-1-4503-3826-4/15/10 ... 15.00
DOI: http://dx.doi.org/10.1145/2808769.2808774

all levels from patients to hospitals to governments. Important challenges for this reform are leveraging existing large and varied clinical and claims datasets to estimate future healthcare costs and taking measures in care-management that reduce such costs while improving overall population health. However, in practice a major roadblock is that ML for many healthcare tasks (e.g., estimating the risk of hospital readmission) needs data that is split over many different owners – healthcare providers, hospitals, and medical insurance companies – who do not want to or legally cannot share their data with outside entities.

In this paper, we attack the problem of securely computing a linear regression model between two parties that are not allowed to share their data. We propose protocols that securely compute a linear regression model over two separate datasets according to our definition (we later show that our solution can be extended to the case of multiple parties). Our results are information-theoretically secure and work in the commodity based model [3, 5]. This model can provide us with unconditionally secure protocols, that is protocols that do not rely on computational assumptions for ensuring their security. It has been proven that commitments [28, 7, 24], oblivious transfer [3, 2], distributed inner product [12], verifiable secret sharing [25, 13], and oblivious polynomial evaluation [31] are implementable in this setting. [20] presents recent general results on the power of the commodity based model. In the commodity based model, Alice and Bob have correlated data that was pre-distributed at the beginning of the protocol. The pre-distributed data can be seen as data provided by a trusted initializer during an offline setup phase. Note that, in this case, this trusted party does not engage in the protocol after the setup phase and never learns Alice and Bob's inputs.

If a trusted initializer is not available or desirable, we present a protocol where Alice and Bob can simulate the trusted initializer by running a two-party secure computation protocol by themselves during an offline setup phase. The remaining online phase remains the same. In this case, the overall protocol becomes computationally secure.

The online phase of our protocol is extremely efficient, having solely modular additions and multiplications. The offline, pre-processing phase, in the case of the computationally secure protocol, is based on any additive homomorphic public key cryptosystem. Because all the data used during the pre-processing is random, the computations to be performed become embarrassingly parallelizable and gains proportional to the number of available cores can be obtained, making even the costly pre-processing phase practical.

We improve the running time of previous algorithms for secure linear regression from days [19] to seconds in the online phase. Even when considering the time needed for Alice and Bob to perform their pre-processing during the offline phase (in the case of the computationally secure protocol) the overall computing time (offline and online phase) is in the order of minutes.

This paper is structured as follows: after discussing related work (Section 2) and giving preliminaries on the security model in Section 3, we present a high level overview of both our protocols for secure linear regression in Section 4. Next, we provide details on our secure computation of multiplication and inverse of matrices (Section 5 and 7), as well as how we deal with real numbers (Section 6). In Section 8, we summarize how these building blocks fit together in our information-theoretically secure protocol, while in Section 9 we explain how to substitute the trusted initializer and obtain a computationally secure protocol. In Section 10 we present runtime results of both protocols on ten different datasets, with a varying number of instances and features, showing a substantial speed-up compared to existing work. Finally, we sketch how to extend the protocols to more than two parties (Section 11) and how to obtain security against malicious adversaries (Section 12).

## 2 Related Work on Secure Linear Regression

There are many attempts in the literature at obtaining secure linear regression protocols over distributed databases. Most of them clearly do not even aim at obtaining the level of privacy usually required by modern cryptographic protocols (such as Karr et al. [21] and Du et al.[14], see also [29, 22]).

The pioneering work of Hall et al. [19] actually presents a protocol that achieves cryptographic security within the framework of secure two-party protocols and simulation based definitions of security, as proposed by Goldreich in [17]. However, we remark that as some of the protocols they propose rely on approximations, rather than exact computations, the correct framework to be used is that of Feigenbaum et al. [15, 16], instead of Goldreich's. Additionally, Hall et al. [19] uses operations in a finite field and homomorphic encryption as a building block. However, the (interesting version of the) linear regression problem deals with numbers in $\mathbb{R}$, or at least in $\mathbb{Q}$. To cope with this problem, a fixed-point data type and its representation in the finite field are defined in [19]. In such an approach, it is necessary to perform a truncation after each multiplication, but the description of the truncation protocol of [19] has a small (correctable) problem as explained in Appendix A. Finally, the overall computing time for solving the linear regression problem for 51K input vectors, each with 22 features, is two days [19]. The online phase of our protocol solves this problem in a few seconds. Even when considering the running time of the offline phase of our computationally secure protocol, by exploiting its embarrassingly parallelization property, the overall running time is still in the order of minutes for such a number of features and vectors.

In [26], a solution is proposed based on homomorphic encryption and garbled circuits for a scenario where many parties upload their data to a third party responsible for obtaining the regression model (with the help of a Crypto Service Provider, responsible for performing heavy cryptographic operations). The Crypto Service Provider is a semi-honest trusted party that is assumed to not collude with other players and actively engages in the protocol during its execution. In our information theoretical solution the trusted initializer does not engage in the protocol after the setup phase and our online phase is still much faster than the protocol presented in [26]. Even when we add up the offline phase and the online phase running times, in the case of our computationally secure protocol, when multiple cores are available for the offline phase computations, the overall running time is less for our protocol.

Finally, we assessed our secure linear regression by implementing and analyzing the results using ten real datasets. We chose a variety of different datasets based on their number of features and instances. Some of our datasets have millions of vectors. We are unaware of any other work on secure linear regression where real datasets of this size have been analysed before. For example, in [19] and in [26], the real datasets used had thousands of vectors.

# 3 Security Model

## 3.1 Secure Two-Party Computation

We consider honest-but-curious adversaries (i.e., adversaries that follow the protocol instructions but try to learn additional information about the other parties' inputs) and define (exact) secure two-party computation following Goldreich [17].

A two-party computation is specified by an ideal functionality that is a (possibly randomized) mapping $f\colon \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$ from inputs $(a,b)$ to outputs $(c,d)$. Let $f_1(a,b)$ denote the first output of $f$ and $f_2(a,b)$ the second. A two-party protocol is a pair of polynomial-time interactive algorithms $\pi = (\pi_{\mathsf{Alice}}, \pi_{\mathsf{Bob}})$. $\mathsf{Alice}$ executes $\pi_{\mathsf{Alice}}$ with input $a$ and randomness $r_{\mathsf{Alice}}$ and $\mathsf{Bob}$ executes $\pi_{\mathsf{Bob}}$ with input $b$ and randomness $r_{\mathsf{Bob}}$. The execution proceeds in rounds, each party is able to send one message in each round to the other party. The messages are specified by $\pi$, given the party's view, which consists of his input, randomness, and messages exchanged so far. Each party can also terminate, at any point, outputting some value based on his view. Let $\mathsf{view}^{\pi}_{\mathsf{Alice}}(a,b)$ denote $\mathsf{Alice}$'s view of the protocol execution, i.e, her input, her randomness, and all the exchanged messages. Let $\mathsf{view}^{\pi}_{\mathsf{Bob}}(a,b)$ similarly denote $\mathsf{Bob}$'s view of the protocol execution. Let $\mathsf{output}^{\pi}_{\mathsf{Alice}}(a,b)$ and $\mathsf{output}^{\pi}_{\mathsf{Bob}}(a,b)$ denote Alice's and Bob's outputs respectively.

Definition 3.1. *A protocol $\pi$ privately computes $f$ with statistical security if for all possible inputs $(a,b)$ the following properties hold:*

- *Correctness:*

$$\{\mathsf{output}^{\pi}_{\mathsf{Alice}}(a,b), \mathsf{output}^{\pi}_{\mathsf{Bob}}(a,b)\} \equiv \{f(a,b)\}$$

- *Privacy: There are simulators $\mathcal{S}_{\mathsf{Alice}}$ and $\mathcal{S}_{\mathsf{Bob}}$ such that:*

$$\{\mathcal{S}_{\mathsf{Alice}}(a,c), f_2(a,b)\} \overset{s}{\approx} \{\mathsf{view}^{\pi}_{\mathsf{Alice}}(a,b), \mathsf{output}^{\pi}_{\mathsf{Bob}}(a,b)\}$$

$$\{f_1(a,b), \mathcal{S}_{\mathsf{Bob}}(b,d)\} \overset{s}{\approx} \{\mathsf{output}^{\pi}_{\mathsf{Alice}}(a,b), \mathsf{view}^{\pi}_{\mathsf{Bob}}(a,b)\}$$

*where $\overset{s}{\approx}$ denotes statistical indistinguishability.*

The privacy requirement ensures that whatever an honest-but-curious adversary learns from interactions within the protocol can also be learned by an ideal adversary that only learns the input and output of that party.

A very useful paradigm for building private protocols is designing them in a modular way, using the following composition theorem [17]:

Theorem 3.2. *Let $f, g$ be two-party functionalities. Let $\pi^{f|g}$ be a private protocol for computing $f$ using oracle calls to $g$ and suppose that there is a private protocol $\pi^g$ computing $g$. Let $\pi^f$ be the protocol obtained from $\pi^{f|g}$ by independently using one instance of $\pi^g$ for implementing each oracle call to $g$. Then $\pi^f$ privately computes $f$.*

## 3.2 Secure Approximations

Note that the private computation of an approximation $\overline{f}$ of a target functionality $f$ can reveal more information than the target functionality itself. Imagine, for instance, the case where the output of $\overline{f}$ is equal to the output of $f$ in all bits except the least significant one, in which $\overline{f}$ encodes one bit of the input of the other party. To ensure that the approximation $\overline{f}$ does not leak additional information, we use the framework of Feigenbaum et al. [15, 16] for private approximations, using the notation of Kiltz et al. [23]. Only deterministic target functionalities $f$ are considered, but the approximations $\overline{f}$ can be randomized.

Definition 3.3. *The functionality $\overline{f}$ is an $\varepsilon$-approximation of $f$ if for all possible inputs $(a,b)$, $|f(a,b) - \overline{f}(a,b)| < \varepsilon$.*

Definition 3.4. *The functionality $\overline{f}$ is functionally private with respect to $f$ if there is a simulator $\mathcal{S}$ such that for all possible inputs $(a,b)$, $\{\mathcal{S}(f(a,b))\} \overset{s}{\approx} \{\overline{f}(a,b)\}$.*

Note that functional privacy is a property of the functionality $\overline{f}$ itself, and not of any protocol $\pi$ implementing it. It captures the fact that the approximation error is independent from the inputs when conditioned on the output of the exact functionality.

Definition 3.5. *A protocol $\pi$ is a private computation of an $\varepsilon$-approximation with respect to $f$ if $\pi$ privately computes a (possibly randomized) function $\overline{f}$ such that $\overline{f}$ is functionally private with respect to $f$ and is an $\varepsilon$-approximation of $f$.*

## 3.3 Commodity Based Cryptography

In the commodity based cryptography model [3, 2], a trusted initializer (TI) distributes values (i.e., the commodities) to the parties before the start of the protocol execution. The TI has no access to the parties' secret inputs and does not communicate with the parties except for delivering the pre-distributed values during the setup. One main advantage of this model is the high computational efficiency that arises from the fact that the parties often only need to derandomize the pre-computed instances to match their own inputs. Another advantage is the computations are pre-distributed by a trusted initializer, and therefore most protocols yield perfect security. The trusted initializer functionality $\mathcal{F}^{\mathcal{D}}_{TI}$ is parametrized by an algorithm $\mathcal{D}$, which is executed upon initialization to generate the correlated randomness $(P_{\mathsf{Alice}}, P_{\mathsf{Bob}})$ that is distributed to $\mathsf{Alice}$ and $\mathsf{Bob}$ respectively.

# 4 Overview of Our Protocols

Assume that we have a set of training examples (real vectors)

$$(a_1(x_i), a_2(x_i), \ldots, a_m(x_i), y_i)$$

where $a_j(x_i)$ is the value of the input attribute $a_j$ for the training example $x_i$ $(i = 1, \ldots, n)$ and $y_i$ is the associated output. The goal is to leverage these training examples to predict the unknown outcome for a previously unseen input as accurately as possible. To this end, we want to learn a linear function

$$y = \beta_1 a_1(x) + \beta_2 a_2(x) + \ldots + \beta_m a_m(x) + b$$

that best approximates the relation between the input variables $a_1(x), a_2(x), \ldots, a_m(x)$ and the response variable $y$.

Throughout this paper we assume that all variables are real numbers and that we aim to find real values for the parameters $\beta_1, \beta_2, \dots, \beta_m$ and $b$ that minimize the empirical risk function

$$\frac{1}{n}\sum_{i=1}^{n}((\beta_1 a_1(x_i) + \beta_2 a_2(x_i) + \ldots + \beta_m a_m(x_i) + b) - y_i)^2 \quad (1)$$

which is the mean squared error over the training instances. For notational convenience, we switch to the homogenous version of the linear function and we use vector notation, i.e. let

- $\mathbf{x_i} = (a_0(x_i), a_1(x_i), a_2(x_i), \dots, a_m(x_i))$, with $a_0(x_i) = 1$ for all $i \in \{1, \dots, n\}$
- $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_m)$, with $\beta_0 = b$

Using $\langle \boldsymbol{\beta}, \mathbf{x_i} \rangle$ to denote the dot product of $\boldsymbol{\beta}$ and $\mathbf{x_i}$, minimizing (1) amounts to calculating the gradient and comparing it to zero, i.e. solving

$$\frac{2}{n}\sum_{i=1}^{n}(\langle \boldsymbol{\beta}, \mathbf{x_i} \rangle - y_i)\mathbf{x_i} = 0 \quad (2)$$

The solution to (2) is[1]

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{y} \quad (3)$$

with

$$X = \begin{pmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \cdots \\ \mathbf{x_n} \end{pmatrix} \text{ and } \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{pmatrix}$$

The scenarios that we are interested in are those in which the training data is not owned by a single party but is instead distributed across multiple parties who are not willing to disclose it. Our experiments in Section 10 correspond to scenarios in which $X$ is partitioned column-wise across two parties, i.e. Alice and Bob have information about different features of the same instances. However, as will become clear below, our protocols work in all scenarios in which Alice has a share $X_{\mathsf{Alice}}$ and Bob has a share $X_{\mathsf{Bob}}$ such that $X_{\mathsf{Alice}} + X_{\mathsf{Bob}} = X$, regardless of whether the dataset $X$ is sliced column-wise, row-wise, or a mixture of the two. In our experiments we also assume that Bob has the vector $Y$. However, our protocol can also handle the case when $Y$ is distributed over two or more players.

Here we give an overview of our solution. The basic idea is to reduce the problem of securely computing linear regression to the problem of securely computing products of matrices. The protocol for computing products of matrices works only for elements of the matrices belonging to a finite field. Thus, Alice and Bob should be able to map their real valued fixed precision inputs to elements of a finite field (as described in Section 6). Our protocol works in a shared input model in which each party holds some elements of the design matrix. Each party creates its share of the design matrix by mapping their respective real valued inputs to elements of a finite field and putting them on the respective position of the matrix and then filling the remaining positions of the matrix's share with zeros. I.e., the shares $X_{\mathsf{Alice}}$ and $X_{\mathsf{Bob}}$ are such that $X_{\mathsf{Alice}} + X_{\mathsf{Bob}} = X$ where $X$ is the design matrix mapped into the finite field.

1. Offline phase: in the information-theoretically secure protocol, Alice and Bob receive correlated data from the Trusted Initializer. In the case of the computationally secure protocol, they run the protocol described in Section 9.
2. Online Phase:
    (a) The players map their fixed precision real valued inputs to elements of a finite field as described in Section 6 and create the shares of $X$ as described above.
    (b) The players compute over their shares using the protocols for matrix multiplication (described in Section 5) and for computing the inverse of a Covariance Matrix (described in Section 7) in order to obtain shares of the estimated regression coefficient vector.
    (c) The players exchange their shares of the estimated regression coefficient vector and reconstruct it.

After presenting the building blocks in Sections 5, 6 and 7, we reiterate the information-theoretically secure and the computationally secure protocol for linear regression at a more concrete level of detail in Sections 8 and 9 respectively. In presenting our protocols, we first introduce an ideal functionality that captures the behaviour of a secure instance of the protocol in question. Ideal functionalities are always represented by calligraphic letters (e.g. $\mathcal{F}_{\mathsf{DMM}}$ in the case of distributed the matrix multiplication functionality). We then present the protocol and prove that the protocol is as secure as the ideal functionality. Protocols are represented by capital greek letters (e.g. $\Pi_{\mathsf{DMM}}$ in the case of the distributed matrix multiplication protocol).

# 5 Secure Distributed Matrix Multiplication Protocol

Let's first take a look at a straightforward extension of Beaver's protocol for secure multiplication in the commodity based model [4] for matrices. Alice and Bob hold shares of matrices $X \in \mathbb{Z}_q^{n_1 \times n_2}$ and $Y \in \mathbb{Z}_q^{n_2 \times n_3}$ to be multiplied and the goal is to obtain shares of the multiplication result $X \cdot Y \in \mathbb{Z}_q^{n_1 \times n_3}$ in such a way that the result remains hidden from both of them individually. Let $X_{\mathsf{Alice}} \in \mathbb{Z}_q^{n_1 \times n_2}$ and $Y_{\mathsf{Alice}} \in \mathbb{Z}_q^{n_2 \times n_3}$ be Alice's shares of the inputs and $X_{\mathsf{Bob}} \in \mathbb{Z}_q^{n_1 \times n_2}$ and $Y_{\mathsf{Bob}} \in \mathbb{Z}_q^{n_2 \times n_3}$ be Bob's shares. Note that $XY = (X_{\mathsf{Alice}} + X_{\mathsf{Bob}})(Y_{\mathsf{Alice}} + Y_{\mathsf{Bob}}) = X_{\mathsf{Alice}}Y_{\mathsf{Alice}} + X_{\mathsf{Alice}}Y_{\mathsf{Bob}} + X_{\mathsf{Bob}}Y_{\mathsf{Alice}} + X_{\mathsf{Bob}}Y_{\mathsf{Bob}}$. The terms $X_{\mathsf{Alice}}Y_{\mathsf{Alice}}$ and $X_{\mathsf{Bob}}Y_{\mathsf{Bob}}$ can be computed locally, but the computation of the terms $X_{\mathsf{Alice}}Y_{\mathsf{Bob}}$ and $X_{\mathsf{Bob}}Y_{\mathsf{Alice}}$ is more complex. Beaver's protocol solves the problem of computing the last two terms by having the trusted initializer distribute random values $A_{\mathsf{Alice}}, A_{\mathsf{Bob}}, B_{\mathsf{Alice}}, B_{\mathsf{Bob}}$ to the parties and also random shares of the value $A_{\mathsf{Alice}}B_{\mathsf{Bob}} + A_{\mathsf{Bob}}B_{\mathsf{Alice}}$. Then the parties only need to derandomize this pre-distributed instance to the actual input values. The protocol $\Pi_{\mathsf{DMM}}$ is parametrized by the size $q$ of the field and by the dimensions $n_1, n_2, n_3$ of the matrices to be multiplied and works as follows:

1. At the setup, the trusted initializer chooses uniformly random $A_{\mathsf{Alice}}, A_{\mathsf{Bob}} \in \mathbb{Z}_q^{n_1 \times n_2}$, $B_{\mathsf{Alice}}, B_{\mathsf{Bob}} \in \mathbb{Z}_q^{n_2 \times n_3}$ and $T \in \mathbb{Z}_q^{n_1 \times n_3}$, and distributes the values $A_{\mathsf{Alice}}, B_{\mathsf{Alice}}, T$ to Alice and the values $A_{\mathsf{Bob}}, B_{\mathsf{Bob}}, C = (A_{\mathsf{Alice}}B_{\mathsf{Bob}} + A_{\mathsf{Bob}}B_{\mathsf{Alice}} - T)$ to Bob.

[1] Assuming that $X^T X$ is invertible

2. $\mathsf{Bob}$ sends $(X_{\mathsf{Bob}} - A_{\mathsf{Bob}})$ and $(Y_{\mathsf{Bob}} - B_{\mathsf{Bob}})$ to $\mathsf{Alice}$.

3. $\mathsf{Alice}$ chooses a random $T' \in \mathbb{Z}_q^{n_1 \times n_3}$, computes $W = A_{\mathsf{Alice}}(Y_{\mathsf{Bob}} - B_{\mathsf{Bob}}) + (X_{\mathsf{Bob}} - A_{\mathsf{Bob}})B_{\mathsf{Alice}} + X_{\mathsf{Alice}}Y_{\mathsf{Alice}} - T'$ and sends $W$, $(X_{\mathsf{Alice}} - A_{\mathsf{Alice}})$ and $(Y_{\mathsf{Alice}} - B_{\mathsf{Alice}})$ to $\mathsf{Bob}$. $\mathsf{Alice}$ outputs $T + T'$.

4. $\mathsf{Bob}$ outputs $U = (X_{\mathsf{Alice}} - A_{\mathsf{Alice}})Y_{\mathsf{Bob}} + X_{\mathsf{Bob}}(Y_{\mathsf{Alice}} - B_{\mathsf{Alice}}) + X_{\mathsf{Bob}}Y_{\mathsf{Bob}} + W + C$.

This protocol securely implements the ideal distributed matrix multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$ that works as follows: $\mathcal{F}_{\mathsf{DMM}}$ is parametrized by the size $q$ of the field and the dimensions $n_1, n_2, n_3$ of the matrices to be multiplied. Given $\mathsf{Alice}$'s input shares $X_{\mathsf{Alice}} \in \mathbb{Z}_q^{n_1 \times n_2}$ and $Y_{\mathsf{Alice}} \in \mathbb{Z}_q^{n_2 \times n_3}$, and $\mathsf{Bob}$'s input shares $X_{\mathsf{Bob}} \in \mathbb{Z}_q^{n_1 \times n_2}$ and $Y_{\mathsf{Bob}} \in \mathbb{Z}_q^{n_2 \times n_3}$, it computes $V = (X_{\mathsf{Alice}} + X_{\mathsf{Bob}})(Y_{\mathsf{Alice}} + Y_{\mathsf{Bob}})$, chooses a random matrix $R \in \mathbb{Z}_q^{n_1 \times n_3}$ and sends $R$ to $\mathsf{Alice}$ and $V - R$ to $\mathsf{Bob}$.

THEOREM 5.1. *The distributed matrix multiplication protocol* $\Pi_{\mathsf{DMM}}$ *is correct and securely implements the distributed matrix multiplication functionality* $\mathcal{F}_{\mathsf{DMM}}$ *against honest but curious adversaries in the commodity based model.*

The correctness of the protocol can be trivially verified by inspecting the value of $T + T' + U$. The security of this protocol lies in the fact that all values exchanged between the parties are blinded by a value which is completely random in the underlying field from the point of view of the message receiver. This protocol can in fact be proved secure even against malicious parties and in the stronger Universal Composability (UC) framework [8]. The idea is that the simulator simulates an instance of the adversary and an instance of the protocol execution with the adversary, allowing the adversary to communicate with the environment. The leverage used by the simulator is the fact that, in the ideal execution, he is the one simulating the trusted initializer. By simulating the TI, he is able, at the same time, to generate a protocol transcript for the adversary that is indistinguishable from the real protocol execution and also to extract the input of the corrupted parties in order to forward them to the ideal functionality.

# 6 Dealing with Real Numbers

The security proof of the (matrix) multiplication protocol presented in the last section essentially relies on the fact that the blinding factors are uniformly random in $\mathbb{Z}_q$. If one tries to design similar protocols working directly with integers or real numbers there would be a problem, since it is not possible to sample uniformly in $\mathbb{Z}$ or $\mathbb{R}$. Similarly, in protocols that use homomorphic encryption as building blocks, the encryption is normally done for messages which are members of a finite group. But in secure protocols for functionalities such as linear regression, one needs to deal with inputs which are real numbers. Thus it is necessary to develop a way to approximate the computations on real numbers by using building blocks which work on fields $\mathbb{Z}_q$.
We adapt the method of Catrina and Saxena [9] with a fixed-point representation. Let $k, e$ and $f$ be integers such that $k > 0$, $f \geq 0$ and $e = k - f \geq 0$. Let $\mathbb{Z}_{\langle k \rangle}$ denote the set $\{x \in \mathbb{Z} : -2^{k-1} + 1 \leq x \leq 2^{k-1} - 1\}$. The fixed-point data type with $k$ bits, resolution $2^{-f}$, and range $2^e$ is the set $\mathbb{Q}_{\langle k,f \rangle} = \{\tilde{x} \in \mathbb{Q} : \tilde{x} = \hat{x}2^{-f}, \hat{x} \in \mathbb{Z}_{\langle k \rangle}\}$. The signed integers in $\mathbb{Z}_{\langle k \rangle}$ are then encoded in the field $\mathbb{Z}_q$ (with $q > 2^k$) using the function

$$g\colon \mathbb{Z}_{\langle k \rangle} \to \mathbb{Z}_q, g(\hat{x}) = \hat{x} \mod q.$$

In secure computation protocols using secret sharing techniques, the values in $\mathbb{Z}_q$ are actually shared between the two parties. Using this encoding, we have that $\hat{x} + \hat{y} = g^{-1}(g(\hat{x}) + g(\hat{y}))$, where the second addition is in $\mathbb{Z}_q$, i.e., we can compute the addition for signed integers in $\mathbb{Z}_{\langle k \rangle}$ by using the arithmetic in $\mathbb{Z}_q$. The same holds for subtraction and multiplication.
For the fixed-point data type we can do additions using the fact that $\tilde{w} = \tilde{x} + \tilde{y} = (\hat{x} + \hat{y})2^{-f}$, so we can trivially obtain the representation of $\tilde{w}$ with resolution $2^{-f}$ by computing $\hat{w} = \hat{x} + \hat{y}$, i.e., we can do the addition of the fixed-point type by using the addition in $\mathbb{Z}_{\langle k \rangle}$, which itself can be done by performing the addition in $\mathbb{Z}_q$. The same holds for subtraction. But for multiplication we have that $\tilde{w} = \tilde{x}\tilde{y} = \hat{x}\hat{y}2^{-2f}$, and therefore if we perform the multiplication in $\mathbb{Z}_q$, we will obtain (if no overflow occurred) the representation of $\tilde{w}$ with resolution $2^{-2f}$. Such representation uses $\mathbb{Z}_{\langle k+f \rangle}$ instead of the original $\mathbb{Z}_{\langle k \rangle}$. In order to have the size of the signed integers representation be independent from the amount of multiplication operations performed with the fixed-point data, we need to scale the resolution of $\tilde{w}$ down to $2^{-f}$. For that purpose we use a slightly modified version of the truncation protocol of Catrina and Saxena [9].
The central idea of this truncation protocol is to reveal the number $w$ to be truncated to one of the parties, but blinded by a factor $r$ which is from a domain exponentially bigger than the domain of the value $w$ and thus statistically hides $w$. The value $r$ is generated so that the parties have shares of both $r$ and $r'$ (which represents the $f$ least significant bits of $r$). Then $\mathsf{Bob}$ can reveal $w + r$ to $\mathsf{Alice}$ and they can compute shares of the truncated value by using local computations. In more detail, let $\lambda$ be a security parameter and let the field $\mathbb{Z}_q$ in which the signed integers are encoded be such that $q > 2^{k+f+\lambda+1}$. Note that the multiplicative inverse of $2^f$ in $\mathbb{Z}_q$ is $((q+1)/2)^f$ and let $F^{-1}$ denote it. The parties have, as inputs, shares $w_{\mathsf{Alice}}, w_{\mathsf{Bob}} \in \mathbb{Z}_q$ such that $w = (w_{\mathsf{Alice}} + w_{\mathsf{Bob}}) \in \{0, 1, \ldots, 2^{k+f-1} - 1\} \cup \{q - 2^{k+f-1} + 1, \ldots, q - 1\}$. The protocol $\Pi_{\mathsf{Trunc}}$ for truncating the output works as follows:

1. At the setup, the trusted initializer chooses uniformly random $r' \in \{0,1\}^f$ and $r'' \in \{0,1\}^{\lambda+k}$ and computes $r = r''2^f + r'$. He also chooses uniformly random $r'_{\mathsf{Bob}}, r_{\mathsf{Bob}} \in \mathbb{Z}_q$ and then sets $r'_{\mathsf{Alice}} = r' - r'_{\mathsf{Bob}}$ and $r_{\mathsf{Alice}} = r - r_{\mathsf{Bob}}$. He sends $r'_{\mathsf{Alice}}, r_{\mathsf{Alice}}$ to $\mathsf{Alice}$ and $r'_{\mathsf{Bob}}, r_{\mathsf{Bob}}$ to $\mathsf{Bob}$.

2. $\mathsf{Bob}$ sends $z_{\mathsf{Bob}} = (w_{\mathsf{Bob}} + r_{\mathsf{Bob}})$ to $\mathsf{Alice}$ and outputs $(w_{\mathsf{Bob}} + r'_{\mathsf{Bob}})F^{-1}$.

3. $\mathsf{Alice}$ computes $c = z_{\mathsf{Bob}} + w_{\mathsf{Alice}} + r_{\mathsf{Alice}} + 2^{k+f-1}$ and $c' = c \mod 2^f$. Then she outputs $(w_{\mathsf{Alice}} + r'_{\mathsf{Alice}} - c')F^{-1}$.

This protocol securely implements the functionality $\mathcal{F}_{\mathsf{Trunc}}$ that captures the approximate truncation without leakage. $\mathcal{F}_{\mathsf{Trunc}}$ is parametrized by the size $q$ of the field and reduces the resolution by $2^{-f}$. Given $\mathsf{Alice}$ and $\mathsf{Bob}$'s shares of the input, $w_{\mathsf{Alice}}, w_{\mathsf{Bob}} \in \mathbb{Z}_q$, and a blinding factor $r'_{\mathsf{Bob}} \in \mathbb{Z}_q$ from $\mathsf{Bob}$, it computes $\hat{w} = g^{-1}(w_{\mathsf{Alice}} + w_{\mathsf{Bob}} \mod q)$ and samples $u$ such that $\Pr[u = 1] = (\hat{w} \mod 2^f)/2^f$. Then it computes $v = (w_{\mathsf{Alice}} - (w_{\mathsf{Alice}} + w_{\mathsf{Bob}} \mod 2^f) - r'_{\mathsf{Bob}})F^{-1} + u$ and sends it to Alice (Bob's output is $(w_{\mathsf{Bob}} + r'_{\mathsf{Bob}})F^{-1}$ and can be locally computed).

THEOREM 6.1. *The truncation protocol $\Pi_{\mathsf{Trunc}}$ privately computes the approximate truncation functionality $\mathcal{F}_{\mathsf{Trunc}}$.*

PROOF. **Correctness**: Let $\hat{w} = g^{-1}(w_{\mathsf{Alice}} + w_{\mathsf{Bob}} \mod q)$. We have that the value $\hat{w} \in \{-2^{k+f-1}+1, -2^{k+f-1}+2, \ldots, 2^{k+f-1}-1\}$. Let $b = \hat{w} + 2^{k+f-1}$ and let $b' = b \mod 2^f$. We have that $b \in \{1, \ldots, 2^{k+f}-1\}$ and since $k > 0$ also that

$$b' = b \mod 2^f = \hat{w} + 2^{k+f-1} \mod 2^f = \hat{w} \mod 2^f.$$

Since $r < 2^{k+f+\lambda}$ and $q > 2^{k+f+\lambda+1}$ we have that $c = b + r$ and thus

$$c' = (b' + r') \mod 2^f = b' + r' - u2^f$$

where $u \in \{0,1\}$ and $\Pr[u = 1] = \Pr[r' \geq 2^f - b'] = (\hat{w} \mod 2^f)/2^f$ with the probability over the choices of random $r'$. Hence

$$c' - r'_{\mathsf{Alice}} - r'_{\mathsf{Bob}} = g(\hat{w} \mod 2^f - u2^f),$$

$$w_{\mathsf{Alice}} + w_{\mathsf{Bob}} + r'_{\mathsf{Alice}} + r'_{\mathsf{Bob}} - c' = g(\hat{w} - (\hat{w} \mod 2^f) + u2^f)$$

$$= g\left(\left\lfloor \frac{\hat{w}}{2^f} \right\rfloor 2^f + u2^f\right),$$

$$(w_{\mathsf{Alice}} + w_{\mathsf{Bob}} + r'_{\mathsf{Alice}} + r'_{\mathsf{Bob}} - c')F^{-1} = g\left(\left\lfloor \frac{\hat{w}}{2^f} \right\rfloor + u\right),$$

and so the shares output by the parties $(w_{\mathsf{Alice}} + r'_{\mathsf{Alice}} - c')F^{-1}$ and $(w_{\mathsf{Bob}} + r'_{\mathsf{Bob}})F^{-1}$ are correct.

**Security**: The only message exchanged is $z_{\mathsf{Bob}} = (w_{\mathsf{Bob}} + r_{\mathsf{Bob}})$ that reveals $c = b + r$ to Alice, but since $r$ is a uniformly random $(k+f+\lambda)$-bits integer and $b$ is a $(k+f)$-bits integer, we have that the statistical distance between $c$ and $r$ is at most $2^{-\lambda}$, i.e., $c$ is statistically indistinguishable from a uniformly random value. □

THEOREM 6.2. *$\mathcal{F}_{\mathsf{Trunc}}$ is an 1-approximation*[2] *and is functionally private with respect to an exact truncation functionality that computes the truncation using the floor function.*

PROOF. The only difference between the two functionalities is that in the approximate truncation an error factor $u$ is present in the shared output. But note that $u \in \{0,1\}$ and $\Pr[u = 1] = (\hat{w} \mod 2^f)/2^f$, but $u$ is independent from the specific shares $w_{\mathsf{Alice}}$, $w_{\mathsf{Bob}}$ used to encode $g(\hat{w})$. Thus the protocol rounds $\hat{w}/2^f$ to the nearest integer with probability $1 - \alpha$, where $\alpha$ is the distance between $\hat{w}/2^f$ and the nearest integer. □

We should mention that in the case of matrix multiplication the truncations only have to be performed in the elements of the resulting matrix instead of once per element multiplication, which would be less efficient and also increase the error due to truncation.

# 7 Computing the Inverse of a Covariance Matrix

In order to be able to compute the linear regression model from a design matrix and the response vector we need to compute the inverse of the covariance matrix. Let $A$ be the covariance matrix. In order to compute $A^{-1}$ we use a generalization for matrices of the Newton-Raphson division method.

The algorithms for division of fixed-point numbers are divided in two main classes: digit recurrence (subtractive division) and functional iteration (multiplicative division). The Newton-Raphson division method is from the functional iteration class, which is more amenable to secure implementation and converges faster. Additionally this method is self correcting, i.e., truncation errors in one iteration decrease quadratically in the next iterations. The inverse of a number $a$ is computed by defining the function $h(x) = x^{-1} - a$ and then applying the Newton-Raphson method for finding successively better approximations to the roots of $h(x)$. The iterations follow the form:

$$x_{s+1} = x_s(2 - ax_s).$$

This algorithm can be generalized for computing the inverse of the matrix $A$. A numerical stable iteration for computing $A^{-1}$ is [19, 18]:

$$\begin{aligned} c &= \mathsf{trace}(A) \\ X_0 &= c^{-1}I \\ X_{s+1} &= X_s(2 - AX_s) \end{aligned}$$

where $I$ is the identity matrix with the same dimensions as $A$. Note that $A$ is a covariance matrix and thus it is positive semi-definite and the trace of $A$ dominates the largest eigenvalue of $A$. It is convenient to use $c = \mathsf{trace}(A)$ because the trace of $A$ can be computed locally by parties that have shares of $A$. To compute $c^{-1}$ the Newton-Raphson is also used with $x_0$ set to an arbitrarily small value, as the convergence happens if the magnitude of the initial value is smaller than that of $c^{-1}$.

Note that, in our case, we use this method to securely compute the inverse of the covariance matrix, i.e, each party has a share of the covariance matrix as input and should receive, as output, random shares of its inverse, but no additional information should be learned by the parties. Hence we cannot perform a test after each iteration in order to check if the values already converged with resolution $2^{-f}$ (and thus stop the iterations at the optimal point) because this would leak information about the input based on how many iterations were performed. We have to use an upper bound $\ell$ on the number of iterations such that all possible covariance matrices converge with resolution $2^{-f}$ in $\ell$ iterations. A very conservative upper bound is $\ell = 2k$ [19]; further studies will be done to try to determine a tighter upper bound.

The protocol to securely compute the inverse of a shared covariance matrix is parametrized by the size $q$ of the field. Let $A \in \mathbb{Z}_q^{n \times n}$ be the encoding in $\mathbb{Z}_q$ of a covariance matrix where the elements are fixed-point numbers. Alice has input $A_{\mathsf{Alice}} \in \mathbb{Z}_q^{n \times n}$ and Bob has input $A_{\mathsf{Bob}} \in \mathbb{Z}_q^{n \times n}$ such that $A_{\mathsf{Alice}} + A_{\mathsf{Bob}} = A$. Then the protocol proceeds as follows:

1. Alice and Bob locally compute shares of $c = \mathsf{trace}(A)$, i.e., $c_{\mathsf{Alice}} = \sum_{i=1}^{n} A_{\mathsf{Alice}}[i,i]$ and $c_{\mathsf{Bob}} = \sum_{i=1}^{n} A_{\mathsf{Bob}}[i,i]$

2. Alice and Bob use the Newton-Raphson division method

[2] in the representation, $2^{-f}$ in the fixed-point data type

to compute shares $c^{-1}_{\mathsf{Alice}}$ and $c^{-1}_{\mathsf{Bob}}$ of $c^{-1}$ with resolution $2^{-f}$. The subtractions can be performed locally and the multiplications using the distributed (matrix) multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$ followed by the approximate truncation functionality $\mathcal{F}_{\mathsf{Trunc}}$.

3. Alice and Bob use the generalized Newton-Raphson method to obtain shares $A^{-1}_{\mathsf{Alice}}$ and $A^{-1}_{\mathsf{Bob}}$ of $A^{-1}$ with resolution $2^{-f}$ for the elements. The subtractions can be performed locally and the multiplications using the distributed matrix functionality $\mathcal{F}_{\mathsf{DMM}}$ followed by the approximate truncation functionality $\mathcal{F}_{\mathsf{Trunc}}$.

We emphasize that the truncation used has some intrinsic error, but the Newton-Raphson method's self-correcting properties compensate for that.

## 8 Computing the Linear Regression Coefficients

We consider the setting in which there is a design matrix $\tilde{X}$ and a response vector $\tilde{y}$. We are interested in analyzing the statistical regression model

$$\tilde{y} = \tilde{X}\tilde{\beta} + \epsilon,$$

and therefore want to compute the estimated regression coefficient vector

$$\overline{\beta} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\tilde{y}$$

The design matrix is a $n \times m$ matrix where the elements are of the fixed-point data type with precision $2^{-f}$ and range $2^{k-f}$ (i.e., $\tilde{X} \in \mathbb{Q}^{n\times m}_{\langle k,f\rangle}$) and the response vector $\tilde{y} \in \mathbb{Q}^{n}_{\langle k,f\rangle}$. Let $\hat{X} \in \mathbb{Z}^{n\times m}_{\langle k\rangle}$ be the element-wise representation of $\tilde{X}$ as signed integers and let $X \in \mathbb{Z}_q^{n\times m}$ be the element-wise encoding of $\hat{X}$ as elements of the field $\mathbb{Z}_q$. Define $\hat{y}$ and $y$ in the same way from $\tilde{y}$.

It is assumed that the parties Alice and Bob hold shares of $X$ and $y$. Alice and Bob can then use the protocols for matrix multiplication, truncation and covariance matrix inversion that were described in the previous sections in order to compute shares of

$$\beta = (X^TX)^{-1}X^Ty$$

Then they only need to reveal their final shares and convert the result back to the fixed-point data type in order to get $\overline{\beta}$.

In more details:

1. Online Phase:
   (a) The players map their fixed precision real valued inputs to elements of a finite field as described in Section 6 and create the shares of $X$ as described above.
   (b) The players compute $X^TX$ by using the matrix multiplication protocol $\Pi_{\mathsf{DMM}}$ (described in Section 5). Once the multiplication is finished they ran the truncation protocol $\Pi_{\mathsf{Trunc}}$ for each element of the matrix $X^TX$.
   (c) Alice and Bob compute the inverse of $(X^TX)$ by running the protocol for computing the inverse of a covariance matrix (described in Section 7). Within the covariance matrix inversion protocol there are several calls to the matrix multiplication and truncation protocols.
   (d) Alice and Bob run the matrix multiplication and the truncation protocols twice to obtain $(X^TX)^{-1}X^T$ and finally $(X^TX)^{-1}X^Ty$.
   (e) The players exchange their shares of the estimated regression coefficient vector and reconstruct it.
   (f) The coefficients $\beta$ obtained by the players are mapped back from finite field elements to real values with finite precision.

The security of the composed protocol follows from the composition theorem 3.2 using the facts that $\Pi_{\mathsf{DMM}}$ securely implements the distributed matrix multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$ and $\Pi_{\mathsf{Trunc}}$ privately computes the approximate truncation functionality $\mathcal{F}_{\mathsf{Trunc}}$. It is assumed that a big enough $k$ is used so that no overflow occurs and hence the correctness of the protocol follows. The final protocol implements the linear regression functionality $\mathcal{F}_{\mathsf{Reg}}$ that upon getting the shares $X_{\mathsf{Alice}}$ and $X_{\mathsf{Bob}}$ of the design matrix $X$ and $y_{\mathsf{Alice}}$ and $y_{\mathsf{Bob}}$ of the response vector $y$, compute $\beta = (X^TX)^{-1}X^Ty$ and output $\beta$ to Alice and Bob.

## 9 Substituting the Trusted Initializer

If a trusted initializer is not desired, it is possible to obtain a solution where the parties, during the setup phase, compute the correlated data themselves. The idea is to use the homomorphic properties of the Paillier's encryption scheme [27]. For two large prime numbers $p$ and $q$, the secret key of Paillier's cryptosystem is $\mathsf{sk} = (p, q)$. The corresponding public key is $\mathsf{pk} = N = pq$ and the encryption of a message $x \in \mathbb{Z}_N$ is done by picking a random $r \in \mathbb{Z}^*_{N^2}$ and computing $\mathsf{Enc}(\mathsf{pk}, x) = (N + 1)^x r^N \mod N^2$. The following homomorphic properties of Paillier's encryption scheme are used:

$$\mathsf{Enc}(\mathsf{pk}, x) \cdot \mathsf{Enc}(\mathsf{pk}, y) = \mathsf{Enc}(\mathsf{pk}, x + y \mod N)$$

$$\mathsf{Enc}(\mathsf{pk}, x)^y = \mathsf{Enc}(\mathsf{pk}, xy \mod N).$$

Given two vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ where the second is given in clear and the first is encrypted element-wise (i.e., $\mathsf{Enc}(\mathsf{pk}, x_i)$ are revealed), one can compute a ciphertext corresponding to the inner product:

$$\mathsf{Enc}(\mathsf{pk}, \langle x, y\rangle \mod N) = \prod_{i=1}^{n} \mathsf{Enc}(\mathsf{pk}, x_i)^{y_i}.$$

The idea for computing the necessary correlated data for the distributed matrix multiplication protocol is to use the above fact in order to compute the non-local multiplication terms. Bob has a pair of public/secret keys for Paillier's encryption scheme and sends to Alice the element-wise encryption *under his own public key* of the elements of the column/row that needs to get multiplied. Alice, having the plaintext corresponding to her own values on the appropriate column/row, can compute an encryption of the inner product under Bob's public key. She then adds a random blinding factor and sends the ciphertext to Bob, who can decrypt it, thus yielding distributed shares of the inner product between Alice and Bob.

The protocol is parametrized by the dimensions $n_1, n_2, n_3$ of the matrices to be multiplied. Bob holds a Paillier's secret key $\mathsf{sk}$, whose corresponding public-key is $\mathsf{pk}$. For a matrix $X$, $x[i, j]$ denote the element in the $i$-th row and $j$-th column. The protocol works as follows:

1. Alice chooses uniformly random $A_{\mathsf{Alice}} \in \mathbb{Z}_N^{n_1 \times n_2}$, $B_{\mathsf{Alice}} \in \mathbb{Z}_N^{n_2 \times n_3}$ and $T \in \mathbb{Z}_N^{n_1 \times n_3}$.

2. Bob chooses uniformly random $A_{\mathsf{Bob}} \in \mathbb{Z}_N^{n_1 \times n_2}$ and $B_{\mathsf{Bob}} \in \mathbb{Z}_N^{n_2 \times n_3}$, element-wise encrypts them under his own public key and send the ciphertexts to Alice.

3. For $i = 1, \ldots, n_1$, $j = 1, \ldots, n_3$, Alice computes the ciphertext

$$\tilde{c}[i,j] = \mathsf{Enc}\left(\mathsf{pk}, t[i,j]\right) \cdot \prod_{k=1}^{n_2} \Big( \mathsf{Enc}\left(\mathsf{pk}, b_{\mathsf{Bob}}[k,j]\right)^{a_{\mathsf{Alice}}[i,k]} \cdot \\ \cdot \mathsf{Enc}\left(\mathsf{pk}, a_{\mathsf{Bob}}[i,k]\right)^{b_{\mathsf{Alice}}[k,j]} \Big)$$

and sends them to Bob. Alice outputs $A_{\mathsf{Alice}}$, $B_{\mathsf{Alice}}$ and $T$.

4. Bob decrypts the ciphertexts in order to get the matrix $C = (A_{\mathsf{Alice}} B_{\mathsf{Bob}} + A_{\mathsf{Bob}} B_{\mathsf{Alice}} + T)$. Bob outputs $A_{\mathsf{Bob}}$, $B_{\mathsf{Bob}}$ and $C$.

The security of this protocol follows trivially from the IND-CPA security of Paillier's encryption scheme [27].
Note that the values $r$ and $r'$ that are distributed by the trusted initializer for performing the truncation protocol can be trivially computed by the parties themselves using distributed multiplications.

## 10 Experiments

We assessed our secure linear regression algorithm by implementing it and analyzing the results using ten real datasets. We chose a variety of different datasets based on the number of features and the number of instances (see Section 10.1). We used C++ as our programming language which we augmented with the BOOST libraries for functionality such as lexical_cast for type casting and asio for work with sockets. We also made use of the GMP and NTL libraries within C++ to implement our protocols. We built our system on top of a Microsoft Azure G4 series machine with Intel Xeon processor E5 v3 family, 224GB RAM size, 3072GB of disk size and 16 cores. Finally, we chose Ubuntu 12.04 as our operating system. We have merged the matrix multiplication and truncation protocols within one protocol for implementation purposes.
The online phase (Section 10.2) is very fast and capable of handling millions of records within less than an hour, which is a huge improvement to the previous results. We only use addition and multiplication of matrices on our online phase which makes it simple and easy to manage.
In the case when a trusted initializer is not desired one can use our computationally secure protocol, at the cost of having a costier offline phase (Section 10.3). However, because Alice and Bob only work over random inputs during the offline phase, the encryption, decryption and mathematical operations are all embarassingly parallelizable.

### 10.1 Datasets

All our datasets are contained within the UCI repository[3], with the exception of the State Inpatient Database (WA) which is provided by HCUP[4]. The UCI repository includes 48 regression task datasets from which we chose 9. Our datasets range in size from 395 instances to over 4 million and from 7 attributes to 367, and are summarized in Table 1.

**Gas Sensor Array Under Dynamic Gas Mixtures** This dataset represents data from 16 chemical sensors exposed to ethylene and CO mixtures at various concentration levels in the air. We added together the concentration of ethylene and the concentration of CO to create one continuous response variable of gas concentration and removed the time feature from the dataset. We then designated the first 8 sensor readings to Alice and the second 8 to Bob. This left us with a total of 4,208,261 sets of 16 sensor readings to different total concentrations of ethylene and CO.

**Communities and Crime** We used 122 attributes describing 1,993 communities and their law enforcement departments in 1990 to create this dataset. The goal with this dataset is to predict the number of violent crimes per capita for each community. All missing values present in the dataset (of which there were 36,850 distributed throughout 1,675 different communities and 22 different attributes) were replaced with 0s. These missing values were largely relevant to the communities' police departments. We also removed 5 variables that were present in the original data but described by the UCI documentation as non-predictive, namely state, county, community, community name, and fold. The final 122 attributes were then divided in half between Alice and Bob.

**Auto MPG** This dataset contains attributes describing 398 automobiles in attempt to predict MPG (miles per gallon) for each. We removed the car name attribute which was present in the original data and were left with 7 predictive features. We then replaced the 6 missing horsepower values with 0s. In the end we designated the cylinders, displacement, and horsepower features to Alice and the weight, acceleration, model year, and origin features to Bob.

**BlogFeedback** In an attempt to predict the number of comments a blog post will receive in the upcoming 24 hours, this dataset originally had 280 attributes. Since our complete dataset must be linearly independent, to enable the inversion of $X^T X$ required in our protocol, we removed 57 of these original attributes leaving us with 223 predictors describing 52,397 blog posts. An example of such a feature would be the binary indicator of whether or not a blog post was published on a Sunday. There are binary indicators of whether publication occurred on any of the other days of the week and therefore this feature, publication on a Sunday, is linearly dependent on the other six. Finally, the dataset was divided column wise, designating 111 attributes to Alice and the other 112 to Bob.

**Wine Quality** This dataset takes 11 attributes related to the white variant of Portuguese "Vinho Verde" wine which are used to predict the quality score of 4,897 wines. We designated the fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, and free sulfur

[3] UC Irvine Machine Learning Repository
`https://archive.ics.uci.edu/ml/datasets.html`

[4] `http://www.ahrq.gov/research/data/hcup/`

dioxide features to Alice and the total sulfur dioxide, density, pH, sulphates, and alcohol features to Bob.

**Bike Sharing** In this dataset we took attributes describing a certain hour and day and attempted to predict the number of users for a bike sharing system. We removed the record index which was present in the original data as well as the count of casual users and the count of registered users and targeted the total rental bikes used ($casual + registered$) for our prediction. We were left with 13 predictors of 17,379 hour/day combinations which were used to model bike use. Alice received information on the dates, seasons, years, months, hours, and holidays while Bob was given information on weekdays, working days, weather situations, temperatures, feel temperatures, humidities, and windspeeds.

**Student Performance** We used 30 attributes describing 395 students across two schools to create this dataset. The goal with this dataset is to predict the final grade of each student in their math class. We removed two columns from the original dataset – one detailing students' performances in the first period and one detailing their performances in the second period. We identified the student's final grade as our sole response variable. The final 30 attributes were then divided evenly between Alice and Bob.

**YearPredictionMSD** In this dataset we have attributes describing audio features of 515,344 songs and we aim to predict the release year of each song. We kept all 90 features that were present in the original data provided by the UCI repository. In allocating the data we gave Alice the first 45 features and the second 45 to Bob.

**State Inpatient Database (WA)** From the HCUP State Inpatient Database (WA) we extracted attributes describing 25,180 beneficiaries who had at least one hospital admission within the state of Washington during the first nine months of the year between the years 2009 and 2012. The goal with this data is to predict the cost each beneficiary will incur in the final three months of the same year. We extracted demographic, medical, and previous cost information from the original data and replaced any missing values with a 0 value. We then designated the age, gender, race, number of chronic conditions, length of stay, and number of admits attributes to Alice. Bob was given a Boolean matrix of comorbidities as well as previous cost information.

**Relative Location of CT Slices on Axial Axis** In an attempt to predict the relative location of a CT slice on the axial axis of the human body, the original dataset had 384 attributes describing CT images. Since our complete dataset must be linearly independent, we removed 17 of these original attributes leaving us with 367 predictors describing 53,500 CT images. We then divided this dataset column wise, designating 183 attributes to Alice and the other 184 to Bob.

## 10.2 Online Phase

We present in Table 1 the running times for the online phase of our protocol building a predictive linear regression model. Our online phase is very fast, computing a linear regression model for a matrix of over 4 million rows and 16 columns in under one hour. The regression coefficients computed with our secure protocol agree to the 5th decimal digit with regression coefficients computed without any security.

We briefly work out the theoretical computational complexity of computing $\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{y}$ with our online protocol. If our dataset (which is denoted by $X$ in this formula), has $n$ features and $m$ records, then the total runtime for computing the $\boldsymbol{\beta}$ values is $O(mn^2)$ which means that the number of records in the dataset has only a linear effect on the run time of our implementation.

We used NTL for matrix multiplication with modular arithmetic. We also used GMP (the GNU Multi-Precision library) in conjunction with NTL to increase our performance. In the NTL library, the basic algorithm is used (with time complexity $O(n^3)$ when all matrix dimensions are $n$).

Note that $(X^T X)$ is a square matrix with both dimensions equal to $n$, and for datasets in which the number of features is small relative to the number of records, computing $(X^T X)^{-1}$ is very fast and negligible in respect to, for example, computing $X^T X$. Our online phase is faster and independent from the semi-honest trusted party unlike similar implementations, such as Nikolaenko et al.'s implementation [26].

## 10.3 Computationally Secure Offline Phase

In the pre-processing of the computationally secure offline phase of the matrix multiplication protocol $\Pi_{\mathsf{DMM}}$, we use Paillier for encryption and decryption, but any additive homomorphic encryption scheme can be used. The down side of these schemes is that their encryption and decryption times are computationally intensive and, if the given dataset is large, the pre-processing phase can take a long time. This issue can be tackled by noticing that Alice and Bob, during this phase, only work over random inputs and thus one can use heavy parallelization to speed-up the running time.

For a dataset with $m$ records and $n$ features, in order to get coefficients securely and correctly, we use $i = 50$ iterations in the computation of inversion. Overall, we need $5mn+(3i+3)n^2+n+3i$ encryptions and $mn+(i+1)n^2+n+i$ decryptions. We also have two matrix multiplications and 3 matrix additions between encryption and decryption. Each encryption in an Azure VM takes about 0.005 seconds for each core. It is then easy to see that the encryption phase is the bottle-neck of the pre-processing phase and easily parallelizable. Since we have $5mn$ number of encryptions, by multiplying this number to the runtime of a single encryption time divided by number of cores, a good estimate of the pre-processing phase is achievable.

The estimated running time for the offline phase is given in Table 2. These estimated results are a huge improvement when compared to the previous result [19] which took two days given a dataset with only about 50,000 records and are comparable to the total running time presented in [26] in the case of 256 cores.

# 11 Extending the Protocol to Multiple Parties

It is easy to generalize the previous protocol to the case in which the design matrix $\tilde{X}$ and the response vector $\tilde{y}$ are shared among more than two parties.

Let $A$, $B$ and $C$ be random matrices such that $C = AB$ and let the party $P_i$ have shares $A_i$, $B_i$, $C_i$ of these matrices. Such triples of shares will be called matrix multiplication

**Table 1: Actual Time Required (in seconds) for Online Phase of Secure Protocol to Build a Predictive Linear Regression Model**

| Dataset Name | Number of Rows | Number of Columns | Train Time: Data Shared in the Clear | Train Time: Using Proposed Secure Protocol |
|---|---|---|---|---|
| Student Performance | 395 | 30 | 0.3 sec | 11.7 sec |
| Auto MPG | 398 | 7 | 0.09 sec | 1.2 sec |
| Communities and Crime | 1,993 | 122 | 9 sec | 147 sec |
| Wine Quality | 4,897 | 11 | 0.9 sec | 5.2 sec |
| Bike Sharing | 17,379 | 13 | 3.7 sec | 16.5 sec |
| State Inpatient Database (WA) | 25,180 | 36 | 21 sec | 93 sec |
| BlogFeedback | 52,397 | 223 | 1,800 sec | 9,000 sec |
| Relative Location of CT Slices on Axial Axis | 53,500 | 367 | 6,000 sec | 30,000 sec |
| YearPredictionMSD | 515,344 | 90 | 3,800 sec | 18,000 sec |
| Gas Sensor Array Under Dynamic Gas Mixtures | 4,208,261 | 16 | 1,100 sec | 4,500 sec |

**Table 2: Estimated Time Required (in seconds) for Offline Phase of Secure Protocol to Build a Predictive Linear Regression Model**

| Dataset Name | Number of Rows | Number of Columns | Offline Time With 16 Cores | Offline Time With 64 Cores | Offline Time With 256 Cores |
|---|---|---|---|---|---|
| Student Performance | 395 | 30 | 20 sec | 6 sec | 2 sec |
| Auto MPG | 398 | 7 | 4 sec | 1 sec | 0.3 sec |
| Communities and Crime | 1,993 | 122 | 400 sec | 100 sec | 30 sec |
| Wine Quality | 4,897 | 11 | 100 sec | 30 sec | 10 sec |
| Bike Sharing | 17,379 | 13 | 350 sec | 100 sec | 30 sec |
| State Inpatient Database (WA) | 25,180 | 36 | 1,500 sec | 400 sec | 100 sec |
| BlogFeedback | 52,397 | 223 | 15,000 sec | 4,000 sec | 1,000 sec |
| Relative Location of CT Slices on Axial Axis | 53,500 | 367 | 30,000 sec | 10,000 sec | 3,000 sec |
| YearPredictionMSD | 515,344 | 90 | 70,000 sec | 20,000 sec | 6,000 sec |
| Gas Sensor Array Under Dynamic Gas Mixtures | 4,208,261 | 16 | 100,000 sec | 30,000 sec | 10,000 sec |

triples. Given a pre-distributed matrix multiplication triple and two shared matrices $X$ and $Y$, it is possible to compute shares of $Z = XY$ without leaking any information about these values. The idea is for the parties to locally compute shares of $D = X - A$ and $E = Y - B$ and then open the values $D$ and $E$. The parties can compute the shares of $Z$ using the fact that

$$Z = AE + BD + AB + DE = AE + BD + C + DE.$$

For instance, $P_1$ can output $Z_1 = A_1E + B_1D + C_1 + DE$ and all the other parties $Z_i = A_iE + B_iD + C_i$. The correctness can trivially be verified by inspection. The security of this method follows from the fact that, when $D$ and $E$ are opened, the values of $X$ and $Y$ are masked by the random factors $A$ and $B$ from the matrix multiplication triple (and all other operations are local). For the truncation protocol, one of the parties, lets say $P_1$, learns the value to be truncated $w$ blinded by a factor $r$, the other parties only use their shares of $w$ and $r'$ (which represents the least significant bits of $r$) in order to get their outputs. The remaining protocols can be trivially generalized to the case of multiple parties.

## 12 Securing Against Malicious Adversaries

One possibility for obtaining security against malicious adversaries is to use the compute with MACs approach [6, 11, 10] to protect the shared values from being manipulated by a malicious adversary. In this technique there is an unconditionally secure message authentication code (MAC) associated to each shared secret value. At the beginning of the protocol the parties commit to their inputs by opening the difference between their inputs and some random shared value (that has an associated MAC). Whenever an operation is performed over some shared values, the MAC for the output value is also computed (using the MACs of the input values). This approach prevents the malicious adversary from successfully deviating from the specified protocol instructions (if he deviates, the honest parties will notice it during the checking of the MACs). The only problem for applying this technique is that the truncation protocol used before is probabilistic and so it would not allow propagation of the MACs, but this can be solved by using a deterministic truncation procedure [1] (at the cost of having to perform one execution of a comparison protocol per truncation). More details about security against malicious adversaries will be presented in the full version. See [6, 11, 10] for further details about the compute with MACs technique.

## 13 Conclusion

In this paper we have presented an information-theoretically secure protocol for privacy preserving linear regression in the commodity-based model. The protocol has an offline phase where a trusted authority pre-distributes random correlated data to Alice and Bob and never again engages in the protocol and/or the online phase. The online phase is orders of magnitude faster when compared to previous solutions in the literature [26, 19].

When a trusted authority is not desirable, we propose an offline phase that is computationally secure and based on any additive homomorphic public key cryptosystem. This offline phase is computationally heavy but completely parallelizable, making it practical when a large number of cores is available.

## 14 References

[1] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele. Secure computation on floating point numbers. In *ISOC Network and Distributed System Security Symposium – NDSS 2013*, San Diego, California, USA, Feb. 24–27, 2013. The Internet Society.

[2] D. Beaver. Precomputing oblivious transfer. In D. Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109, Santa Barbara, CA, USA, Aug. 27–31, 1995. Springer, Berlin, Germany.

[3] D. Beaver. Commodity-based cryptography (extended abstract). In *29th Annual ACM Symposium on Theory of Computing*, pages 446–455, El Paso, Texas, USA, May 4–6, 1997. ACM Press.

[4] D. Beaver. One-time tables for two-party computation. In *Computing and Combinatorics*, pages 361–370. Springer, 1998.

[5] D. Beaver. Server-assisted cryptography. In *Proceedings of the 1998 workshop on New security paradigms*, NSPW '98, pages 92–106, New York, NY, USA, 1998. ACM.

[6] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.

[7] C. Blundo, B. Masucci, D. R. Stinson, and R. Wei. Constructions and bounds for unconditionally secure non-interactive commitment schemes. *Des. Codes Cryptography*, 26(1-3):97–110, June 2002.

[8] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, Nevada, USA, Oct. 14–17, 2001. IEEE Computer Society Press.

[9] O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In R. Sion, editor, *FC 2010: 14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 35–50, Tenerife, Canary Islands, Spain, Jan. 25–28, 2010. Springer, Berlin, Germany.

[10] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In J. Crampton, S. Jajodia, and K. Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18, Egham, UK, Sept. 9–13, 2013. Springer, Berlin, Germany.

[11] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, Santa Barbara, CA, USA, Aug. 19–23, 2012. Springer, Berlin, Germany.

[12] R. Dowsley, J. Graaf, D. Marques, and A. C. A. Nascimento. A two-party protocol with trusted initializer for computing the inner product. In Y. Chung and M. Yung, editors, *WISA 10: 11th International Workshop on Information Security Applications*, volume 6513 of *Lecture Notes in Computer Science*, pages 337–350, Jeju Island, Korea, Aug. 24–26, 2010. Springer, Berlin, Germany.

[13] R. Dowsley, J. Müller-Quade, A. Otsuka, G. Hanaoka, H. Imai, and A. C. A. Nascimento. Universally composable and statistically secure verifiable secret sharing scheme based on pre-distributed data. *IEICE Transactions*, 94-A(2):725–734, 2011.

[14] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *In Proceedings of the 4th SIAM International Conference on Data Mining*, pages 222–233, 2004.

[15] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, pages 927–938, 2001.

[16] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright. Secure multiparty computation of approximations. *ACM Transactions on Algorithms*, 2(3):435–472, 2006.

[17] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[18] C.-H. Guo and N. J. Higham. A schur-newton method for the matrix p'th root and its inverse. *SIAM Journal On Matrix Analysis and Applications*, 28(3):788–804, oct 2006.

[19] R. Hall, S. E. Fienberg, and Y. Nardi. Secure multiple linear regression based on homomorphic encryption. *Journal of Official Statistics*, 27(4):669–691, 2011.

[20] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *Theory of Cryptography*, pages 600–620. Springer, 2013.

[21] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter. Secure regression on distributed databases. *Journal of Computational and Graphical Statistics*, 14(2):263–279, 2005.

[22] 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*, 25(1):125, 2009.

[23] E. Kiltz, G. Leander, and J. Malone-Lee. Secure computation of the mean and related statistics. In J. Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 283–302, Cambridge, MA, USA, Feb. 10–12, 2005. Springer, Berlin, Germany.

[24] A. C. A. Nascimento, J. Müller-Quade, A. Otsuka, G. Hanaoka, and H. Imai. Unconditionally secure homomorphic pre-distributed bit commitment and secure two-party computations. In C. Boyd and W. Mao, editors, *ISC 2003: 6th International Conference on Information Security*, volume 2851 of *Lecture Notes in Computer Science*, pages 151–164, Bristol, UK, Oct. 1–3, 2003. Springer, Berlin, Germany.

[25] A. C. A. Nascimento, J. Müller-Quade, A. Otsuka, G. Hanaoka, and H. Imai. Unconditionally non-interactive verifiable secret sharing secure against faulty majorities in the commodity based model. In M. Jakobsson, M. Yung, and J. Zhou, editors, *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 355–368, Yellow Mountain, China, June 8–11, 2004. Springer, Berlin, Germany.

[26] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 334–348. IEEE, 2013.

[27] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Berlin, Germany.

[28] R. L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Preprint available at http://people.csail.mit.edu/rivest/Rivest-commitment.pdf, 1999.

[29] A. P. Sanil, A. F. Karr, X. Lin, and J. P. Reiter. Privacy preserving regression modelling via distributed computation. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 677–682. ACM, 2004.

[30] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[31] R. Tonicelli, A. C. Nascimento, R. Dowsley, J. Müller-Quade, H. Imai, G. Hanaoka, and A. Otsuka. Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. *International Journal of Information Security*, pages 1–12, 2014.

# APPENDIX

## A Problems with the Truncation Protocol of Hall et al. [19]

The protocol of Hall et al. [19] uses operations in a finite field and homomorphic encryption as a building block. A fixed-point data type is defined and also its representation in the finite field. In such approach it is necessary to perform a truncation after each multiplication, but the truncation protocol of [19] is not correct. The truncation protocol of [19] works in two phases. First, for $P \ll q$ and $\hat{x}$ such that $|\hat{x}| < P$, it takes the encoding $x$ of $\hat{x}$ in $\mathbb{Z}_q$ (i.e., $x \in \{0, \dots, P/2-1\} \cup \{q-P/2+1, \dots, q-1\}$) and generates shares of $\hat{x}$ encoded in the smaller field $\mathbb{Z}_P$. Then using the encodings of $\hat{x}$ in both $\mathbb{Z}_q$ and $\mathbb{Z}_P$ it performs the truncation. In their protocol Alice has the private key to an instance of Paillier's encryption scheme, Bob knows the corresponding public key $q$, and has the encryption under $q$ of a value $\hat{x}$ that is encoded as $x \in \mathbb{Z}_q$, denoted $E(x)$. It is assumed that $|\hat{x}| < P$ for $P \ll q$. The parties want to truncate the input that have an additional factor $M$, each obtaining a share of the truncated output. It works as follows:

1. Bob draws $r$ uniformly at random from the set $\{P, \dots, q-1\}$. He then sets $x_{\mathsf{Bob}} = r \mod P$, computes $E(x-r)$ using the homomorphic properties of the Paillier's encryption scheme and sends it to Alice.
2. Alice decrypts the value to obtain $s = x - r \mod q$ and sets $x_{\mathsf{Alice}} = (s-q) \mod P$. She computes $E(x_{\mathsf{Alice}})$ and sends the ciphertext to Bob.
3. Bob uses the homomorphic properties of the Paillier's encryption scheme to compute $E((x_{\mathsf{Alice}}+x_{\mathsf{Bob}}-x)M^{-1}-r')$ for a random $r' \in \mathbb{Z}_q$, sends it to Alice and outputs $\left\lfloor \frac{x_{\mathsf{Bob}}}{M} \right\rfloor - r' \mod q$.
4. Alice decrypts $s' = (x_{\mathsf{Alice}} + x_{\mathsf{Bob}} - x)M^{-1} - r'$ and outputs $\left\lfloor \frac{x_{\mathsf{Alice}}}{M} \right\rfloor - s' \mod q$.

This protocol is not correct since the subprotocol for generating the shares of the secret $\hat{x}$ encoded in $\mathbb{Z}_P$ is not correct. If the blinding factor $r$ is such that $P < x < r$ then equation 5 used in the correctness proof (see [19]) does not hold since the equality $s + r = x + q$ (in the integers) does not hold. This can be solved by adding $P/2$ to $x$ before executing the $P$-sharing subprotocol and then removing $P/2$ accordingly in the end of the subprotocol.