Using Robust Estimation Theory to Design Efficient Secure Multiparty Linear Regression

Fida K. Dankar
Sidra medical and Research Center
Doha, Qatar
fdankar@sidra.org

Sabri Boughorbel
Sidra medical and Research Center
Doha, Qatar
sboughorbel@sidra.org

Radja Badji
Sidra medical and Research Center
Doha, Qatar
rbadji@sidra.org

ABSTRACT
Various biomedical research studies, such as large-population studies and studies on rare diseases, require sharing of data across multiple sources or institutions. In fact, data sharing will enable the collection of more cases for analysis and thus increase the statistical power of the study. However, combining data from various sources poses privacy risks. A number of protocols have been proposed in the literature to address the privacy concerns; but these protocols do not fully deliver either on privacy or complexity. The main reason lies in the methodology used to design these secure algorithms. It is based on translating regular algorithms into secure versions using cryptographic procedures and tricks rather than on establishing robust theory for designing secure and communication free distributed algorithms. In this paper, we use well-established theoretical results to design a secure and low communication linear regression protocol. The method used is comprehensive and can be generalized to other estimators.

CCS Concepts
• Security and Privacy→Cryptography • Security and Privacy→Security services→Privacy-preserving protocols.

Keywords
Data sharing; secure multiparty computation; linear regression; information privacy.

1. INTRODUCTION
To advance large-scale biomedical studies, and to strengthen the statistical power of research studies (such as complex associations), researchers frequently need to share data with colleagues from around the world. Such data sharing activities are contingent on the protection of patient anonymity. Traditionally, researchers would strip the data from its identifying information - such as names and unique IDs- before sharing it with each other. However, as many recent studies have shown [1]–[3], it is now possible to deduce the identity of research participants from genomic and/or clinical data that was considered anonymized.

In the face of these growing privacy concerns, researchers are looking more and more at cryptographic secure computations (also known as secure multiparty computations).

These secure techniques enable data holders to compute a function over their data while keeping these data private. Only the final result of the computation is revealed to all parties.

Although secure computation protocols exist for multiple data mining/ statistical functions, these protocols are mostly inappropriate for real world application due to their computational overhead. Communication cost is the main factor driving this inefficiency (extensive message passing between the different concerned parties is the main bottleneck of existing secure computations). The main reason lies in the methodology used to design these secure algorithms. It is based on translating regular algorithms into secure versions using cryptographic procedures and tricks rather than on establishing robust theory for designing ***secure*** and ***communication free***- distributed algorithms.

There is considerable research devoted to designing distributed algorithms that are ***communication free***. In such algorithms, data is partitioned into multiple subsets, subsets are stored on different machines, processes run on the different subsets simultaneously and the final outcome is calculated as a function of the individual ones, thus limiting communication to the final step. In this paper, we import some of this theory into the design of a ***secure*** and ***low communication*** linear regression protocol. The method used is comprehensive and can be generalized to other estimators (geometric median, principal component analysis, etc). The theory is described in the next section followed by a description of the secure protocol. The paper concludes with a discussion of the limitations and future directions.

2. THEORY
We introduce the classical setting of a linear regression problem. Let $X = \{x_{i,j}\}$ be an $n \times p$ matrix of features and $Y = (y_1, \dots, y_n)^T$ a corresponding $n \times 1$ response vector, where n is the number of samples and p is the number of features. The linear regression problem consists of solving the linear model, $Y = X\beta + \epsilon$ [4]. To improve the computational efficiency of the linear regression problem, Wang et al [5] designed a distributed linear regression algorithm. The algorithm, referred to as *message*, partitions the dataset into multiple subsets and processes these subsets in parallel. Formally, the data (X, Y) is divided horizontally into k subsets $\{(X^1, Y^1); \dots; (X^k, Y^k)\}$, with $X^i = (x_1^i, \dots, x_p^i)$ the $n_i \times p$ feature matrix for subset i and $Y^i = (y_1^i, \dots, y_{n_i}^i)^T$ the corresponding $n_i \times 1$ response vector. The algorithm then executes the following two steps:

1. In the first step, the feature selection vector for each subset is calculated separately using the Lasso algorithm [6]. After that, the overall model is obtained by including the features selected by the majority of the subsets. Thus, if $\gamma^i = \{\gamma_1^i, \dots, \gamma_p^i\}$ is the feature

selection vector for site i, (with $\gamma_j^i = 1$ if feature j is included and 0 otherwise), then the overall vector is obtained by:
$\gamma = \{mode\{\gamma_1^1, \dots, \gamma_1^k\}, \dots, mode\{\gamma_k^1, \dots, \gamma_k^k\}\}$

2. In the second step, the coefficients of the selected features are estimated separately for each subset, and the final result is obtained by averaging the coefficient estimates of each feature. Thus, if $\beta^i = \{\beta_1^i, \dots, \beta_p^i\}$ is the feature coefficients vector for site i, then the overall vector is obtained by:
$\gamma = \{Average\{\beta_1^1, \dots, \beta_1^k\}, \dots, Average\{\beta_k^1, \dots, \beta_k^k\}\}$

The *message* algorithm fairs well not only in terms of computational time, but also in terms of feature selection performance. In fact, the authors prove that under certain assumptions (that will be detailed in the discussion section) the feature inclusion vector converges exponentially to the optimal model [5]. A result that is better than applying Lasso to the whole dataset.

Privacy is not a concern in the *message* algorithm. In fact, sharing of intermediate and local results between the different subsets is supported in the algorithm. Prior work, [7], demonstrated through multiple scenarios, how identifying information can be inferred from local and intermediate statistics using various different attacks.

This paper uses the theory behind the *message* algorithm to formulate an efficient and private linear regression algorithm. The proposed algorithm performs efficiently with regards to *feature selection* and *model calculation* under certain assumptions. The algorithm is presented in the next section and the assumptions are discussed in Section 5. The paper concludes with a discussion of future work.

3. SECURE PROTOCOL

We assume that a dataset (X, Y) is horizontally distributed among $k \geq 2$ data holders (or sites) $S_1, \dots, S_k$. The different sites are interested in cooperatively performing linear regression on the union of their datasets, however they are not willing to share their data. Only the final result of the computation should be revealed to all parties. The secure algorithm is depicted in Algorithms1 and 2, it requires a semi-trusted third party whose role is the generation of keys for a public encryption cryptosystem as well as the decryption of some results.

Paillier cryptosystem [8] will be used due to its nice homomorphic properties. These properties will be introduced in the next subsection followed by a presentation of the secure protocol.

3.1 Paillier Public Cryptosystem

In public key cryptosystems, The ciphertext (the encryption) c of a message m (usually an integer) is obtained by applying an encryption function: $c = Enc_{pk}(m)$, where pk is the public encryption key. The ciphertext c can be decrypted using another key sk (referred to as the secret key) and a decryption function: $m = Dec_{sk}(c)$. Paillier public cryptosystem is additively homomorphic, as such the sum of two messages can be obtained from their respective cyphertexts. For Paillier, this translates to $Enc_{pk}(m_1 + m_2) = Enc_{pk}(m_1) \times Enc_{pk}(m_2)$ [8]. Moreover, Paillier allows a limited form of homomorphic multiplication, in that we can multiply an encrypted message by a plaintext. It is done as follows: $Enc(m_1)^{m_2} = Enc(m_1 m_2)$.

3.2 Protocol

The main algorithm is described in Algorithm 1; it can be roughly divided into 4 steps:

1. The third party generates the keys for the Paillier cryptosystem and propagates the public key to all parties
2. Each party calculates its local feature selection vector via Lasso, the overall feature selection vector is then calculated via a secure mode protocol (Algorithm 2). The secure mode protocol retains all features that have an overall inclusion probability greater than ½. The calculated mode vector is then propagated to the different parties without leaking any information about individual feature selection vectors.

```
Initialization
1: Input
   $n, p, k, \{S_1, \dots, S_k\}, \{(X^1, Y^1); \dots; (X^k, Y^k)\}, \{n_1, \dots, n_k\}$ # n is the total sample size, p is the number of features, k is the number of sites, $\{S_1, \dots, S_k\}$ are the different sites, $X^i, Y^i$ represent the data set held by site i, and $n_i$ is the sample size at site i.

Third party
2: Propagates a public encryption key to all sites
Each party calculates
3: $\gamma^i \in \{0,1\}^p$ # the feature inclusion model via lasso
All parties calculate
4: $\gamma = mode(\gamma^i, i \in \{1, \dots, k\})$ via Secure mode protocol
Each party calculates
5: $\beta^i = (X^{i^T} X^i)^{-1} X^{i^T} Y^i$ # the estimated coefficients for features $\gamma$
Each party calculates
6: $Enc(\frac{\beta^i}{k})$
All parties calculate
7: $Enc(\beta) = Enc\left(\sum_{i=1}^{k} \frac{\beta^i}{k}\right) = \prod_{i=1}^{k} Enc(\frac{\beta^i}{k})$
   #Calculation is done sequentially: party 1 calculates
   $Enc(\frac{\beta^1}{k})$ and sends it to party 2, party 2 calculates
   $Enc\left(\frac{\beta^1}{k}\right) * Enc(\frac{\beta^2}{k})$ and sends it to party 3, and so on ...
8: $Enc(\beta)$ is sent to the third party

Third party
9: Decrypts and propagates $\beta$
```

Algorithm 1. Main algorithm

3. After receiving the overall feature selection vector, each party calculates the coefficients of the selected features separately. The encrypted average of these features is then securely computed using Paillier encryption as follows: each site i calculates $Enc(\frac{\beta^i}{m})$, then all sites calculate $Enc\left(\sum_{i=1}^{k}\frac{\beta^i}{k}\right)=\prod_{i=1}^{k}Enc(\frac{\beta^i}{k})$ sequentially. The encrypted result is sent to the third party.
4. The third party decrypts and propagates the estimated feature coefficients.

The secure mode protocol is described in Algorithm 2, it computes the mode of the k feature inclusion vectors without revealing any information about the different sites (in other words, the inclusion information for the sites is kept confidential, only the mode is revealed to the different parties).

```
Initialization
1: $\theta = \{\frac{1}{2}\}^p$

Each party $i$
2: Generates a random positive integer $x_i$ # denote by $x = \prod_{i=1}^{k} x_i$
Each party calculates
3: $w^i = (\gamma^i - \theta)$
4: $Enc(w^i)$

All parties calculate
5: $Enc(w) = \sum_{i=1}^{k} Enc(w^i)$ # note if w is positive at position j ($w[j] > 0$) this implies that the majority of sites had 1 at the j position in their feature inclusion model
Sequentially the parties calculate
6: $Enc(wx) = \{Enc(w[1])^x, \dots, Enc(w[p])^x\}$ # party1 calculates $Enc(wx_1) = \{Enc(w[1])^{x_1}, \dots, Enc(w[p])^{x_1}\}$ and sends it to party2, party 2 calculates $Enc(wx_1x_2) = \{Enc(wx_1)^{x_2}, \dots, Enc(wx_1)^{x_2}\}$ and so on.
7: $Enc(wx)$ is sent to third party

Third party
8: Decrypts $Enc(wx)$
9: Propagates $wx$
Each site
10: Calculate γ as follows $\begin{cases}\gamma[j] = 1 \text{ if } W[j] > 0\\ \gamma[j] = 0 \text{ otherwise}\end{cases}$
```

Algorithm 2. Secure mode protocol

If $\partial = \sum_{i=1}^{k}\gamma^i$ is the sum of the feature inclusion vectors of the different parties, then $w = \partial - \{\left\lfloor\frac{k}{2}\right\rfloor\}^p$ would indicate whether the feature should be included or not through its sign: if w is positive at position j ($w[j] > 0$) this implies that the majority of sites had 1 at the jth position in their feature inclusion model, and thus the overall feature inclusion vector should be set as $\gamma[j] = 1$, otherwise it is set to 0. Thus, in order to calculate the feature inclusion vector, it is enough to securely calculate w. However, as w presents aggregated information about the sites (the number of sites that include every feature), our secure protocol will instead calculate an obfuscated version: wx, where x is a positive integer whose factors are distributed among the different parties: $x = \prod_{i=1}^{k} x_i$, where x_i is held by site i and is kept confidential. As x is positive, wx would still indicate whether the feature should be included or not through its sign. The protocol is explained in detail in Algorithm2.

3.3 Protocol Modification

It is important to note that the algorithm can be slightly changed to operate without the need of a third party. In such case, a ***threshold*** Paillier cryptosystem can replace the third party [9]–[11]. In a threshold cryptosystem, the secret decryption key is distributed among a preset number, t, of entities (the data holders in our case). For the decryption to occur, each of the t entities has to perform its share of the decryption. The decryption shares are then combined to obtain the final result. This way, the protocol will be robust against the corruption of at most $t-1$ entities [12] (note that the current version works under the assumptions that all sites are non-corruptible, because if one party is corrupted, then they can communicate with the third party to decrypt intermediate results).

The threshold Paillier cryptosystem can be set up through a trusted party that will generate and distribute the public and secret keys. The trusted party can then erase all information pertaining to the key generation. If no such trusted party is available, the keys can be generated using secure multiparty computations [13]. Although this requires more computation overhead from each data owner, it only has to be done once. As such, it is an acceptable tradeoff.

4. COMPLEXITY

In this section, we evaluate the ***additional*** computational burden of the algorithm incurred to achieve security. This burden is evaluated on each participating party. The complexity will be expressed in terms of the number of messages and in terms of basic functional units. These units are homomorphic multiplication (HM) and homomorphic addition (HA). Assuming an instance of Paillier modulus m^2 [8], HA is equivalent to multiplying two integers modulo m^2, and HM is equivalent to computing an exponentiation modulo m^2 where the exponent is at most m. As such, an HM operation is equivalent to $\log(m)$ HA. With these considerations, it follows that a message encryption is dominated by 2HM, while a decryption is dominated by HM.

The algorithm performs 2 encryptions per party (step 6 of main algorithm and step 4 of secure protocol), and one homomorphic multiplication per party (step 6 in secure protocol). This amount to a total of 5 HM per party. On the other hand, the number of message passing per party is constant, amounting to 5 to 7 messages.

Prior secure linear regression algorithms bear much higher computational burdens on the participants [7], [12], [14]–[16]. The best algorithm [12] requires $O(k\frac{p^2}{2}+\frac{p^3}{3})$ messages per party and $O(\frac{p^4}{4})$ HM operations per party.

5. DISCUSSION

Algorithm consistency for regression or classification is a well-studied property in the field of statistical learning. It gives a theoretical guarantee that an algorithm converges to a optimal algorithm when the number of training samples becomes very large. In practical situations, the number of samples is not that large and therefore the (theoretical) consistency property is not of critical importance, what's more critical is the validation of the proposed algorithms on multiple diverse datasets as it provides confidence in their performances on real datasets and under different settings.

In general, both consistency and good performance on experimental datasets are well-desired properties for regression algorithms. The algorithm *message* (that was used to derive our algorithm) was tried in various experiments using multiple datasets in [5], and it showed good consistency and good performance. Nonetheless, in what follows, we discuss the required assumptions to ensure the ***theoretical*** property of consistency. There are four assumptions for the consistency of the selection algorithm [5]:

- A.1 Consistency condition for estimation: This condition is usually required for high dimensional model regression.
- A.2 Conditions on model parameters. It imposes a restriction on model noise, parameter estimate and the number of selected features.
- A.3 (Lasso) The strong irrepresentable condition.
- A.4 The sparse Riesz condition.

Where A.2 is a basic assumption that is required for high-dimensional model estimation [17], [18], A.3 and A.4 are the specific conditions for model selection consistency in Lasso.

In [5] a theoretical analysis of assumptions A.1, A.3 and A.4 is proposed. This analysis indicates that it is possible to ensure the validity of these three assumptions under the assumption that $n \geq k * (A + kC)$ where n is the total number of samples; k is the number of sites and the number of samples per site is assumed to be equal ($n_i = \frac{n}{k}$). A and B are complex terms that depend on the training data and model parameters. The previous equation guarantees that the three assumptions are valid in probability provided that k and n are chosen such that $k = O(n)$.

The experimental results in the paper that introduced *message* algorithm are obtained for both synthetic data and real-world data [5]. For the synthetic data, the chosen values for k is 200 and the sample size n is varied between 2000 and 10000. The number of features p is 10000, k =50 and n varied between 2000 and 10000. For the real-world data (household electric power consumption dataset), the sample size is defined as n =2 million for training and n =75259 for testing. The number of sites is $k = 200$. For the second real-world data (HIGGS classification dataset), n =11 million and k =1000. For the previously described experimental settings, the median selection algorithm showed very good results. Table 1 summarizes the values of n and k chosen in [5].

Table 1. Choices of sample sizes (n) and number of sites (k) as in [5].

Experimental data	Number of samples (n)	Number of sites (k)
Synthetic	2000-10000	50,200
household electric power consumption	75259, 2 millions	200
HIGGS classification	11 millions	1000

In summary, the previous experimental results showed that a ratio between the number of samples and sites of about 50 might ensure the nice consistency property.

6. FUTURE WORK

As future work, we plan to investigate the optimal choice of the number of sites as a function of the number of samples to guarantee the trade-off between accurate regression and efficient computation. The consistency assumption gives an inequality condition between the number of sites and samples. We plan to look deeper into this relation and validate the theoretical findings using experimental data.

We also plan to test our algorithm on real data sets and compare it (i) to existing secure computation protocols, and (ii) to the ordinary Lasso technique (applied on the combined dataset) in terms of both *accuracy* and *performance*.

Additionally, we plan to extend our method to other estimators such as principal component analysis. This *may* require designing new secure protocols for dealing with intermediate statistical calculations among the different sites, analogous to the presented secure mode protocol.

7. REFERENCES

[1] Y. Erlich and A. Narayanan, "Routes for breaching and protecting genetic privacy," *Nat. Rev. Genet.*, vol. 15, no. 6, pp. 409–421, 2014.

[2] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, "Privacy in the genomic era," *ACM Comput. Surv. CSUR*, vol. 48, no. 1, p. 6, 2015.

[3] E. Check Hayden, "Researchers wrestle with a privacy problem," *Nature*, vol. 525, no. 7570, pp. 440–442, Sep. 2015.

[4] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*, vol. 821. John Wiley & Sons, 2012.

[5] X. Wang, P. Peng, and D. B. Dunson, "Median Selection Subset Aggregation for Parallel Inference," in *Advances in Neural Information Processing Systems*, 2014, pp. 2195–2203.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. Ser. B Methodol.*, pp. 267–288, 1996.

[7] K. El Emam, S. Samet, L. Arbuckle, R. Tamblyn, C. Earle, and M. Kantarcioglu, "A secure distributed logistic regression protocol for the detection of rare adverse drug events," *J. Am. Med. Inform. Assoc. JAMIA*, vol. 20, no. 3, pp. 453–461, May 2013.

[8] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptology—EUROCRYPT'99*, 1999, pp. 223–238.

[9] Y. Desmedt, "Threshold cryptosystems," in *Advances in Cryptology — AUSCRYPT '92*, J. Seberry and Y. Zheng, Eds. Springer Berlin Heidelberg, 1993, pp. 1–14.

[10] C. Hazay, G. L. Mikkelsen, T. Rabin, and T. Toft, "Efficient rsa key generation and threshold paillier in the two-party setting," in *Topics in Cryptology–CT-RSA 2012*, Springer, 2012, pp. 313–331.

[11] R. Canetti and S. Goldwasser, "An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack," in *Advances in Cryptology—EUROCRYPT'99*, 1999, pp. 90–106.

[12] F. Dankar, "Privacy Preserving Linear Regression on Distributed Databases," *Trans. Data Priv.*, vol. 8, pp. 3–28, 2015.

[13] T. Nishide and K. Sakurai, "Distributed paillier cryptosystem without trusted dealer," in *Information Security Applications*, Springer, 2011, pp. 44–60.

[14] R. Hall, S. E. Fienberg, and Y. Nardi, "Secure multiple linear regression based on homomorphic encryption," *J. Off. Stat.*, vol. 27, no. 4, p. 669, 2011.

[15] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-Preserving Ridge Regression on Hundreds of Millions of Records," 2012.

[16] F. Dankar, R. Brien, C. Adams, and S. Matwin, "Secure Multi-Party linear Regression.," in *EDBT/ICDT Workshops*, 2014, pp. 406–414.

[17] P. Zhao and B. Yu, "On model selection consistency of Lasso," *J. Mach. Learn. Res.*, vol. 7, pp. 2541–2563, 2006.

[18] Y. Kim, S. Kwon, and H. Choi, "Consistent model selection criteria on high dimensions," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 1037–1057, 2012.