Fully Homomorphically Encrypted Linear Regression using CKKS

Roberto Carboni, Roland Hostettler, Anders Ahlén, Subhrakanti Dey
Department of Electrical Engineering; Signals and Systems, Uppsala University, Uppsala, Sweden
Email: {roberto.carboni, roland.hostettler, anders.ahlen, subhrakanti.dey}@angstrom.uu.se

***Abstract*—In this paper, we consider the solution of encrypted linear regression using Homomorphic Encryption. We propose a method in which each mathematical operation is performed over encrypted real numbers. This method allows the computation of linear regression in an encrypted environment without the need to modify the original dataset and does not require additional techniques for the calculation of matrix multiplications. The proposed method consists of an iterative method based on a modified Goldschmidt sequence. Numerical results on both synthetic and real data show that the method converges with minimal accuracy loss due to encryption noise, indicating that our approach is well-suited for homomorphically encrypted linear regression.**

***Index Terms*—Linear Regression, Homomorphic Encryption, CKKS, Secure Computing, Machine Learning**

I. Introduction

Predicting a continuous outcome based on one or more predictor variables is fundamental in machine learning and data analysis. Linear regression is a simplistic yet powerful and principled statistical approach to model the relationship between variables [1]. However, performing linear regression over large amounts of data might be expensive in terms of computational cost. To overcome this problem, cloud computing can be used, which involves using shared servers managed by third-party providers. Since the data used in the cloud might be sensitive, it raises the need to provide a privacy-preserving environment. This can be achieved with Homomorphic Encryption (HE), which allows computations on encrypted data without decrypting it [2]. In this way, it is possible to share data with third-party providers while maintaining privacy.

However, HE schemes, such as the Cheon–Kim–Kim–Song (CKKS) scheme [3], have strict constraints, such as the types and number of operations that can be computed on encrypted data. Hence, great care needs to be taken when implementing algorithms using HE. One of the most significant obstacles is the number of sequential operations that can be computed. Typically, HE schemes only support addition and/or (a limited number of) multiplications [4]. Hence, since divisions, and in particular matrix inverses are not supported, one has to resort to iterative algorithms to solve problems such as linear regression. However, classical iterative methods, such as the Gauss–Seidel method [5] require a large number of iterations, see Figure 1.

Financial support by the Swedish Research Council (VR) and the Swedish Agency for Innovation Systems (VINNOVA) is hereby gratefully acknowledged.

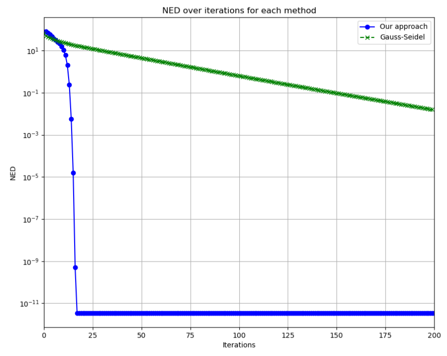


Figure 1: Comparison of the average NED (Normalized Euclidean Distance) over 100 tests for the computation of linear regression using randomized matrices of size 5×5. We can observe that the Gauss–Seidel method continues to converge, but it requires a large number of iterations. In contrast, our approach based on [6] converges quickly to a solution with an acceptable error.

Linear regression using HE has been considered in [7] and [8]. However, these methods do not compute all the operations in an HE environment. In particular, the methods rely on plaintext client-side matrix inversion. Furthermore, in [9] a method for linear regression is proposed based on the Paillier encryption scheme [10]. The Paillier scheme is an additive homomorphic encryption scheme that only supports addition and scalar multiplication, that is, multiplication between an encrypted number and a plaintext number, requiring additional techniques such as additive secret sharing, secure fixed-point arithmetic, and interactive protocols to compute the dot product between matrices. Moreover, the Paillier scheme allows only integer numbers, making it challenging to handle real numbers directly, which can lead to precision loss [9].

In contrast, our approach utilizes the CKKS encryption scheme which supports both addition and multiplication on real or complex numbers. This allows the computation of linear regression in an encrypted environment and does not require additional techniques for the computation of matrix multiplications. In particular, the contributions of this paper are:

- a method that allows the computation of linear regressions in an encrypted domain where each mathematical operation is performed over encrypted real numbers;
- a thorough analysis of the computational complexity of the proposed algorithm;

- evaluation of the method on synthetic and real data.

II. Background

A. Linear Regression

Linear regression is a model that describes a linear relationship between a dependent (noisy) variable $\mathbf{y} \in \mathbb{R}^{d_y}$ and an independent variable (parameters) $\mathbf{x} \in \mathbb{R}^{d_x}$, according to

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{r}, \tag{1}$$

where $\mathbf{H} \in \mathbb{R}^{d_y \times d_x}$ is the observation matrix and $\mathbf{r} \in \mathbb{R}^{d_y}$ is a noise term. The observation matrix is obtained starting from a dataset D defined as a collection of input-output pairs $\{(\mathbf{u}_i, y_i)\}_{i=1}^{N}$ where $\mathbf{u}_i \in \mathbb{R}^{d_x}$ are the regressors (inputs) and $y_i \in \mathbb{R}$ are the corresponding target vectors (outputs) for each sample i. The observation matrix is then constructed as

$$\mathbf{H} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_N\end{bmatrix}^T. \tag{2}$$

One way to estimate the parameters of this model is to use the least squares method [5]. Given the model (1), this is equivalent to solve the system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3}$$

where

$$\mathbf{A} = \mathbf{H}^T\mathbf{H}, \tag{4a}$$

$$\mathbf{b} = \mathbf{H}^T\mathbf{y}, \tag{4b}$$

which has the analytical solution

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}. \tag{5}$$

As shown in (5), the core of this method is to invert the matrix $\mathbf{A} = \mathbf{H}^T\mathbf{H}$ and compute the product $\mathbf{b} = \mathbf{H}^T\mathbf{y}$. Typical approaches for matrix inversion can be computed using one of the classical iterative methods such as Gauss elimination or Cholesky decomposition, see [5].

B. Homomorphic Encryption

In this study, we focus our interest on computing linear regression in an encrypted environment. To achieve this, we use HE [11], a kind of encryption that allows computations on encrypted data without decrypting it. The core of this type of encryption is the use of a *homomorphism*, defined as a function f such that [12]

$$f(a \diamond b) = f(a) \diamond f(b)$$

where the symbol $\diamond$ represents any given operation. It follows that if we apply the inverse function f^{-1} we obtain

$$f^{-1}(f(a \diamond b)) = f^{-1}(f(a) \diamond f(b)) = a \diamond b.$$

In HE, the functions f and f^{-1} represent the encryption and decryption functions encr and decr respectively. Hence

$$\mathrm{encr}(a \diamond b) = \mathrm{encr}(a) \diamond \mathrm{encr}(b)$$

$$\mathrm{decr}(\mathrm{encr}(a \diamond b)) = \mathrm{decr}(\mathrm{encr}(a) \diamond \mathrm{encr}(b)) = a \diamond b.$$

In this work we use the CKKS scheme [3]. This HE scheme is designed for approximate arithmetic with real or complex numbers and supports additions and a limited number of multiplications, making it a so-called *leveled* HE scheme [13]. Furthermore, the scheme is approximate due to the way noise is added during encryption.

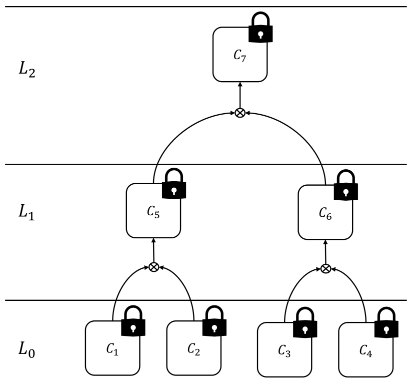


Figure 2: Example of multiplications in layered HE. When a multiplication is computed it consumes one level, and the result will be on a new level. Multiple sequential multiplications consume multiple levels. In this example, the different multiplications consume a total of three levels. Thus, the depth cost of these multiplications is equal to 3.

It follows that the result of an encrypted operation will not be exact, but approximate, that is,

$$\mathrm{decr}(\mathrm{encr}(a) \diamond \mathrm{encr}(b)) \approx a \diamond b$$

Furthermore, the CKKS scheme allows us only to compute additions and multiplications. As discussed above, it is a leveled method, which means that each sequential multiplication *consumes* one level, see Figure 2. The number of available levels is set by specific parameters before encryption.

To overcome the problem of a limited number of multiplications, Gentry [2] proposed a *bootstrapping* method, which has also been implemented for CKKS [14]. However, bootstrapping is an expensive operation which significantly increases the computational complexity of the algorithm. Thus, it is often desirable to avoid bootstrapping in practice, and we shall not consider it in this work.

III. Method

To solve (3) one can use iterative numerical methods, such as Gauss elimination, Cholesky decomposition, or gradient descent. However, these methods typically require a relatively large number of multiplications and divisions, rendering them not feasible to use together with CKKS. Moreover, these methods require divisions and other operations such as comparisons or absolute values that require a large number of levels [5], [15]. For these reasons, implementing linear regression in the HE environment is challenging. Instead, we propose the use of a modified Goldschmidt algorithm, first proposed in [6].

It is important to notice that in a HE environment, it is not possible to use the result of a comparison between values to terminate an algorithm, since the result of this comparison would be an encrypted number and cannot be used for a stopping criterion. Thus, no comparisons are used to stop the algorithm upon reaching a certain accuracy. Instead, the

Algorithm 1 Modified Goldschmidt Algorithm

```
Function: $\mathrm{GoldschmidtAlg}$
Input: $Z_0$, $A_0$, $n_{iter}$
for $i = 1$ to $n_{iter}$ do
    $Z_i \leftarrow Z_{i-1} + Z_{i-1}A_{i-1}$
    $A_i \leftarrow A^2_{i-1}$
end for
return $Z_i$
```

algorithm is run for a predefined number n_{iter} iterations, where n_{iter} is the maximum number of iterations that can be computed. This maximum is determined by specific encryption parameters and may vary accordingly.

A. Linear Regression using Modified Goldschmidt Algorithm

The proposed method solves the system (5) by numerically finding the inverse $\mathbf{A}^{-1}$. In particular, our approach is based on the modified Goldschmidt's algorithm proposed in [6] and shown in Alg. 1. Our approach consists of three steps.

The first step is the *Initialization*, in which we initialize the variables for the iterative method. In this step we need the inverse of the scalar λ, see Alg. 2, which can also be computed using the modified Goldschmidt algorithm in Alg. 1, initializing it with an initial guess g. In the second step, the matrix inverse is computed again using Alg. 1. In the third step, the last multiplication is performed for the parameter estimation. This is summarized in Alg. 2.

Note how in Alg. 2 all the operations are HE-friendly, meaning that they can be easily computed in an HE environment.

B. Analysis of Multiplicative Depth

In an HE environment, the number of levels, and thus the number of sequential multiplications, must be predetermined. Therefore, it is important to analyze the depth cost of our approach.

First, note that a matrix multiplication, despite involving multiple scalar multiplications, requires only one level, as the scalar multiplications are performed in parallel, see Figure 2.

a) Step 1. Initialization: From Alg. 2 it can be seen that, in this step, the number of matrix multiplications is fixed. In this step, we need one matrix-matrix multiplication for the computation of $\mathbf{H}^T\mathbf{H}$. Then we need one matrix-vector multiplication for $\mathbf{H}^T\mathbf{y}$, one matrix-scalar multiplication to compute $\lambda_{inv}\mathbf{A}$, and one matrix-matrix element-wise multiplication ($\mathbf{A} \circ \mathbf{I}$ to compute the trace of the matrix $\mathbf{A}$). Note that this yields a total of 3 sequential matrix multiplications, as the computation of $\mathbf{A}$ and $\mathbf{b}$ (4) consumes only one level since they are computed in parallel, as shown in the example in Figure 2 where, at level L_0, the multiplication between c_1 and c_2, and between c_3 and c_4, consume only one level.

The number of multiplications required for the inversion of λ varies (where λ is the trace of the matrix $\mathbf{A}$ and g is an initial guess for the inverse of λ). This is because we are using the modified Goldschmidt algorithm also to invert

Algorithm 2 Linear Regression

```
Input: $\mathbf{H}$, $\mathbf{y}$, $g$
Output: $\hat{\mathbf{x}}$
Step 1. Initialization
$\mathbf{A} \leftarrow \mathbf{H}^T\mathbf{H}$
$\mathbf{b} \leftarrow \mathbf{H}^T\mathbf{y}$
$\lambda \leftarrow \mathrm{trace}(\mathbf{A})$
$\lambda_{inv} \leftarrow \mathrm{GoldschmidtAlg}(g, 1 - g\lambda, n_\lambda)$
$\mathbf{Z}_0 \leftarrow \lambda_{inv}\mathbf{I}_{n\times n}$
$\mathbf{A}_0 \leftarrow \mathbf{I}_{n\times n} - \lambda_{inv}\mathbf{A}$
Step 2. Matrix Inversion
$\mathbf{A}_{inv} \leftarrow \mathrm{GoldschmidtAlg}(\mathbf{Z}_0, \mathbf{A}_0, n_A)$
Step 3. Parameter Estimation
$\hat{\mathbf{x}} \leftarrow \mathbf{A}_{inv}\mathbf{b}$
return $\hat{\mathbf{x}}$
```

this scalar, and since it is an iterative method, the number of iterations, and thus the number of multiplications, is not constant. Nevertheless, we empirically observed that when the magnitude of the scalar λ is known, then the iterative method can be initialized to require approximately two iterations, and thus two multiplications, to invert the scalar. However, in the following analysis, we will consider the worst case scenario, and we will assume zero knowledge of λ and consider the cost of inverting λ as n_λ.

b) Step 2. Matrix Inversion: In this step, the number of multiplications depends on the number of iterations, which depends on the size of the matrix [6]. Notice that the matrices $\mathbf{Z}_i$ and $\mathbf{A}_i$ are always on the same level. Hence, one iteration consumes one level and this step requires a total of n_A levels.

c) Step 3. Parameter Estimation: In this step only one multiplication is performed, so this requires only one level.

The depth cost analysis for Alg. 2 is summarized in Table I.

TABLE I: Depth Cost Analysis for Alg. 2

Step	Operation	Depth
1	Matrix-Matrix/Vector Multiplications	1
	Matrix-Scalar Multiplications	1
	$1/\lambda$	1+n_λ
	Trace	1
2	Matrix Inversion	n_A
3	Matrix Multiplications	1
	Total	$5 + n_\lambda + n_\mathrm{A}$

IV. Results

A. Setup

The proposed method as represented by Alg. 2 is implemented using the SEAL library [16] through the Python API provided by TenSEAL library [17] with the CKKS scheme. The experiments were run on a system with a Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz, 754 GiB of RAM and AlmaLinux 8.10. For all the experiments, we used a total of 25 levels with 60 bit precision for each of them.

We evaluate the proposed method on both synthetic and real data. In particular, we focus our interest in evaluating the error of the matrix inversion and the parameter estimation.

The error of the matrix inversion is evaluated using the Norm Spectral Error (NSE) and the Natural Distance (d_N) [18]. The NSE and the d_N are defined as

$$NSE = \frac{\|\mathbf{A}^{-1} - \hat{\mathbf{A}}^{-1}\|_2}{\|\mathbf{A}^{-1}\|_2} \times 100, \tag{7}$$

$$d_N(\mathbf{A}^{-1}, \hat{\mathbf{A}}^{-1}) = \sqrt{\sum_{p=1}^{P} (\ln(\nu_p))^2} \tag{8}$$

where $\mathbf{A}^{-1}$ is the true inverted matrix and $\hat{\mathbf{A}}^{-1}$ is the approximated inverted matrix using Alg. 1, and ν_p are the eigenvalues of $\mathbf{A}\hat{\mathbf{A}}^{-1}$. For the parameter estimation, we used the Normalized Euclidean Distance (NED) defined as

$$NED = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2} \times 100, \tag{9}$$

where $\mathbf{x}$ is the vector of target values defined in (1), and $\hat{\mathbf{x}}$ is the estimated value defined in (5), i.e., the values obtained from the parameter estimation using Alg. 2.

B. Synthetic data

We start by creating three random vectors $\mathbf{v} \in [-1, 1]^{d_y}$, $\mathbf{x} \in [0, 1]^{d_x}$ and $\mathbf{r}$, where $\mathbf{v}$ and $\mathbf{x}$ are sampled from a uniform distribution and $\mathbf{r}$ is sampled from a normal distribution with standard deviation $\sigma = 10^{-3}$. We then construct the vectors $\mathbf{u}$ such that, for a given feature v_i, $\mathbf{u}_i = \left[1, v_i, v_i^2, \ldots, v_i^n\right]^T$, where the first element corresponds to the bias term, i.e., $v_i^0 = 1$, and the remaining elements are higher-order powers of the feature v_i. Thus, the matrix $\mathbf{H}$ is formed according to (2), so that $\mathbf{H}$ consists of the polynomial expansions of the input data, one row for each observation. We then compute $\mathbf{y}$ using (1). Finally, we construct $\mathbf{A}$ and $\mathbf{b}$ using (4).

To test Alg. 2, we consider two cases. For each case we generate 10 synthetic samples and execute Alg. 2. We compare the results obtained in the HE environment with the results obtained using Alg. 2 with plaintext data. We then compute the errors using the NSE, d_N, and NED defined above, followed by averaging these errors. For these tests, we used random vectors $\mathbf{v}$ with dimension $d_y = 100$ and $n = 3$ and $n = 4$. The results of the first case are reported in Figure 3a, and the results for the second case are reported in Figure 3b and summarized in Table II. A comparison of the d_N for both tests is reported in Figure 3c.

In Table II, it is shown that the error, for all three metrics, using plaintext data and encrypted data is similar but not the same. This difference arises from the CKKS method, which introduces noise as a security parameter.

From these results, we can see that the method converges. However, as shown in Figure 3, it does not always converge to the lowest error but rather to a slightly higher one. In particular, in some cases, it initially reaches a low error value, but then converges toward a slightly higher one. This behavior is attributed to the noise introduced by the CKKS method, which grows and accumulates at each multiplication.

TABLE II: Experiment Results for the synthetic data

Metric	Type	n=3	n=4
NSE	Encrypted	8.44e-07%	5.08e-06%
	Plaintext	1.01e-12%	3.06e-12%
d_N	Encrypted	8.68e-09	5.19e-08
	Plaintext	1.11e-14	3.72e-14
NED	Encrypted	2.37e-07%	8.65e-07%
	Plaintext	5.64e-13%	3.37e-12%

C. Real data

For the tests over real data, we used the Mauna Loa CO_2 dataset [19]. Specifically, we selected the subset containing the monthly mean of CO_2 expressed in parts per million, which consists of 301 samples. We modeled the dataset generating the matrix $\mathbf{H}$ as defined in (2) with

$$\mathbf{u}_i = [1, t'_i, {t'_i}^2, \sin(\omega t_i), \cos(\omega t_i), \sin(2\omega t_i), \cos(2\omega t_i)]^T,$$

where the t_i represents the time, and t'_i are the terms of the normalized vector $\mathbf{t}' = \frac{\mathbf{t} - \min(\mathbf{t})}{\max(\mathbf{t}) - \min(\mathbf{t})}$ that was used for numerical stability. The angular frequency $\omega = \frac{2\pi}{T_0}$ is defined with $T_0 = 12$, and $\mathbf{y}$ is the target vector that represent the CO_2 concentration level.

The results of the matrix inversion and the parameter estimation are presented in Figure 4 and summarized in Table III. We notice that the error of the linear regression is converging.

TABLE III: Experiment Results for the CO_2 dataset

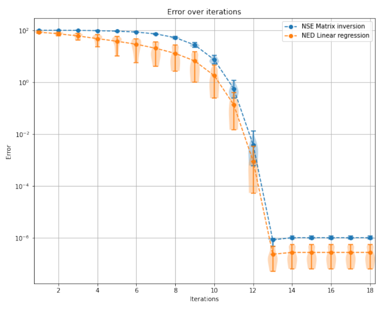
Metric	Type	CO_2
NSE	Encrypted	1.94e-06%
	Plaintext	3.33e-12%
d_N	Encrypted	1.95e-08
	Plaintext	3.77e-14
NED	Encrypted	1.19e-06%
	Plaintext	4.42e-12%

V. Conclusions

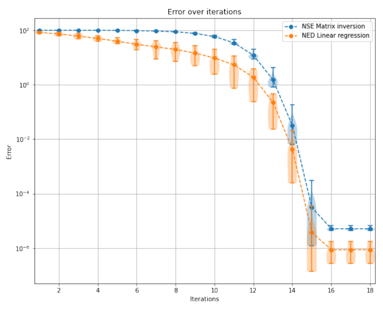
In this work, we have defined a structured methodology for evaluating fully Homomorphically Encrypted linear regression. The proposed method allows the evaluation of a linear regression model where the input data remains encrypted throughout the entire process. We then evaluated the proposed method with both synthetic and real data. Each computational step, including the setup phase, is performed using encrypted operations without requiring data decryption at any stage and without multiparty computation.

We conducted evaluations using both synthetic and real datasets. The results from these experiments show that the proposed approach is capable of performing linear regression while preserving data privacy.

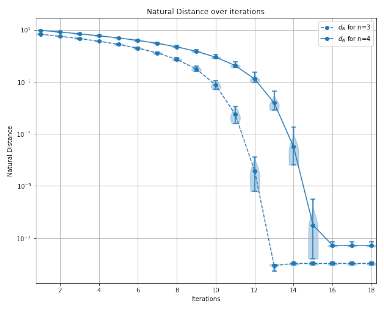
From the depth cost analysis and experimental outcomes, we conclude that having more information about the dataset allows for better parameter selection, leading to higher accuracy. Specifically, we can better manage the number of operations required for the trace and matrix inversion, as well as the use of prior approximation for the trace.



(a)

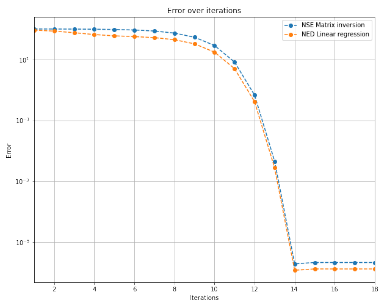


(b)

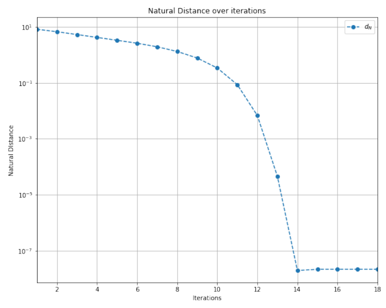


(c)

Figure 3: Comparison of the errors for the two cases using synthetic data. (a) NSE and NED for the case with $n = 3$; (b) NSE and NED for the case with $n = 4$; (c) d_N for both cases.



(a)



(b)

Figure 4: NSE, NED and d_N for the CO_2 dataset.

The next step is to use the trained model from this study and use it for other datasets to further analyze the methods' behavior, and study the optimal number of training iterations needed to achieve high prediction accuracy on new data.

References

[1] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012.

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[3] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer, 2017, pp. 409–437.

[4] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. Fitzek, and N. Aaraj, "Survey on fully homomorphic encryption, theory, and applications," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1572–1609, 2022.

[5] Å. Björck *et al.*, *Numerical methods in matrix computations*. Springer, 2015, vol. 59.

[6] T. M. Ahn, K. H. Lee, J. S. Yoo, and J. W. Yoon, "Cheap and fast iterative matrix inverse in encrypted domain," in *European Symposium on Research in Computer Security*. Springer, 2023, pp. 334–352.

[7] B. Chen and X. Zheng, "Implementing linear regression with homomorphic encryption," *Procedia Computer Science*, vol. 202, pp. 324–329, 2022.

[8] I. Giacomelli, S. Jha, M. Joye, C. D. Page, and K. Yoon, "Privacy-preserving ridge regression with only linearly-homomorphic encryption," in *Applied Cryptography and Network Security: 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings 16*. Springer, 2018, pp. 243–261.

[9] R. Hall, S. E. Fienberg, and Y. Nardi, "Secure multiple linear regression based on homomorphic encryption," *Journal of official statistics*, vol. 27, no. 4, pp. 669–691, 2011.

[10] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238.

[11] R. L. Rivest, L. Adleman, M. L. Dertouzos *et al.*, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[12] V. Shoup, *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[13] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption Without Bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[14] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part I 37*. Springer, 2018, pp. 360–384.

[15] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*. Springer, 2020, pp. 221–256.

[16] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, Jan. 2023, microsoft Research, Redmond, WA.

[17] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," 2021.

[18] M. L. Nordenvaad and L. Svensson, "A MAP based estimator for inverse complex covariance matrices," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 3369–3372.

[19] D. X. Lan and D. R. Keeling, "CO2 Trends," https://gml.noaa.gov/ccgg/trends/ and https://scrippsco2.ucsd.edu/, 2025.