PP-STAT: An Efficient Privacy-Preserving Statistical Analysis Framework using Homomorphic Encryption

Hyunmin Choi
NAVER Cloud
Sungkyunkwan University
Seoul, South Korea
hyunmin.choi@{navercorp.com,g.skku.edu}

Abstract

With the widespread adoption of cloud computing, the need for outsourcing statistical analysis to third-party platforms is growing rapidly. However, handling sensitive data such as medical records and financial information in cloud environments raises serious privacy concerns. In this paper, we present *PP-STAT*, a novel and efficient Homomorphic Encryption (HE)-based framework for privacy-preserving statistical analysis. HE enables computations to be performed directly on encrypted data without revealing the underlying plaintext. *PP-STAT* supports advanced statistical measures, including Z-score normalization, skewness, kurtosis, coefficient of variation, and Pearson correlation coefficient, all computed securely over encrypted data. To improve efficiency, *PP-STAT* introduces two key optimizations: (1) a Chebyshev-based approximation strategy for initializing inverse square root operations, and (2) a pre-normalization scaling technique that reduces multiplicative depth by folding constant scaling factors into mean and variance computations. These techniques significantly lower computational overhead and minimize the number of expensive bootstrapping procedures. Our evaluation on real-world datasets demonstrates that *PP-STAT* achieves high numerical accuracy, with mean relative error (MRE) below 2.4×10^{-4}. Notably, the encrypted Pearson correlation between the `smoker` attribute and `charges` reaches 0.7873, with an MRE of 2.86×10^{-4}. These results confirm the practical utility of *PP-STAT* for secure and precise statistical analysis in privacy-sensitive domains.

CCS Concepts

• **Security and privacy → Domain-specific security and privacy architectures**; • **Mathematics of computing → Probability and statistics**.

Keywords

Homomorphic Encryption, CKKS scheme, Statistical analysis, Privacy-preserving computation

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761194

ACM Reference Format:
Hyunmin Choi. 2025. *PP-STAT*: An Efficient Privacy-Preserving Statistical Analysis Framework using Homomorphic Encryption. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746252.3761194

1 Introduction

Statistical analysis services are increasingly deployed across domains such as finance, education, and healthcare. These services rely on core statistical measures—such as Z-score normalization, coefficient of variation, skewness, kurtosis, and Pearson correlation coefficient—to extract meaningful patterns from data [4, 10, 19, 27, 28].

To scale these services, cloud computing has become a de facto solution. However, outsourcing sensitive data to public cloud platforms raises serious privacy concerns, including the risk of unauthorized access or data leakage. To mitigate such threats, homomorphic encryption (HE) enables computation directly over encrypted data without decryption, supporting a variety of privacy-preserving machine learning and statistical workloads [17, 18, 23, 26].

Among existing HE schemes, the Cheon-Kim-Kim-Song (CKKS) scheme [14] is particularly suited for real-valued computations, making it a practical foundation for encrypted analytics. However, despite its flexibility, CKKS poses significant implementation challenges: it only supports polynomial operations (addition, multiplication, and rotation), requiring complex polynomial approximations for non-linear functions. As a result, managing ciphertext noise and computational depth becomes a critical bottleneck in large-scale HE-based statistical analysis.

Despite growing interest, existing HE-based frameworks for data analysis are often closed-source or limited in scope, making practical adoption difficult. To address these limitations, we present *PP-STAT*, an efficient and scalable open-source framework for encrypted statistical analysis. *PP-STAT* implements five foundational statistical operations—Z-score normalization, coefficient of variation, skewness, kurtosis, and Pearson correlation coefficient—executed securely under the CKKS scheme. Many statistical measures in *PP-STAT* rely on the inverse square root operation, which is non-trivial to implement efficiently under HE. Existing methods such as Goldschmidt or Newton's method [11] have been adapted for HE, but typically assume fixed initial guesses. For example, Lee et al. [26] use $y_0 = 1$ as the initial value in Newton's method, which leads to slow convergence and reduced accuracy for small x. Panda et al. [30] improved initialization via a homomorphic *sign* function (Pivot-Tangent method), but the approach requires six bootstrapping operations.

To overcome these limitations, we propose *CryptoInvSqrt*, a Chebyshev-based initialization method for Newton's iteration under HE. It achieves faster convergence while significantly reducing bootstrapping overhead. Specifically, *CryptoInvSqrt* requires only two bootstrapping operations, which is 2.5× and 3× fewer than those used in Lee et al. [26] and Panda et al. [30], respectively. On the interval $[0.001, 100]$, *CryptoInvSqrt* achieves a mean relative error (MRE) of 5.08×10^{-5}, outperforming Pivot-Tangent (MRE: 3.73×10^{-4}) while using less than half its bootstrapping operations. We further evaluate *PP-STAT* on two real-world datasets: *Adult* and *Insurance*. When computing the encrypted Pearson correlation coefficient between the `smoker` and `charges` attributes in the *Insurance* dataset, *PP-STAT* produces a value of 0.7873, with a mean relative error of 2.25×10^{-4}. This result demonstrates that accurate and meaningful statistical analysis can be conducted directly over encrypted data without revealing sensitive information.

Our contributions are as follows:

- We present *PP-STAT*, an efficient and scalable HE-based framework that supports five advanced statistical operations: Z-score normalization, kurtosis, skewness, coefficient of variation, and Pearson correlation coefficient—over encrypted data.
- We propose *CryptoInvSqrt*, a Chebyshev-based initialization for Newton's method that reduces bootstrapping to two rounds with higher accuracy and faster runtime than prior work. We also introduce pre-normalization scaling, enabling higher-degree Chebyshev polynomials within fixed multiplicative depth to improve accuracy without extra cost.
- We empirically demonstrate that *PP-STAT* achieves high numerical accuracy on large-scale datasets. For example, Z-score normalization over one million records in the domain $[0, 100]$ yields an MRE of 4.18×10^{-5}.
- We validate the practical utility of *PP-STAT* through experiments on two real-world datasets, *Adult* and *Insurance*. In the *Insurance* dataset, it identifies a strong correlation (Pearson correlation coefficient = 0.7873) between `smoker` and `charges`, with an MRE of 2.25×10^{-4}.
- The full implementation of *PP-STAT* is publicly available at https://github.com/hm-choi/pp-stat. All experiments are fully reproducible and serve as a resource for further research and deployment.

2 Related Work

2.1 Inverse Square Root over HE

Cheon et al. [16] proposed HE-based methods for computing inverse and square root values, using Goldschmidt's division algorithm [21] and Wilkes's algorithm [31], respectively. These approaches rely solely on addition and multiplication, but do not provide a unified method for inverse n-th root computation, which is frequently required in statistical analysis. Lee et al. [26] introduced HEaaN-STAT, a framework supporting various statistical operations under HE. For inverse n-th root, it applies Newton's iteration with a fixed initial value, but does not optimize the initialization for convergence or depth reduction. Panda et al. [30] proposed the Pivot-Tangent method, which estimates a better initial point for Newton's iteration by evaluating a sign function. While this improves convergence, the sign function is computationally expensive under HE due to its non-linear nature. In contrast, our approach leverages Chebyshev approximation to generate accurate initial guesses. This enables efficient inverse square root computation with fewer Newton iterations and significantly reduced bootstrapping overhead.

2.2 Statistical Analysis over HE

Several frameworks have been developed for encrypted machine learning. HElayers [2] supports neural networks and decision trees using a tensor-based abstraction. TenSEAL [8] enables encrypted tensor computations for logistic regression and convolutional networks. UniHENN [17] proposes an HE-friendly CNN inference architecture with approximation and batching strategies. While effective for ML workloads, these systems provide limited support for general-purpose statistical analysis. In contrast, *PP-STAT* focuses on foundational measures such as Z-score normalization, skewness, and kurtosis, and incorporates Chebyshev-based inverse square root approximation—an optimization not available in existing frameworks.

3 Background

3.1 Homomorphic Encryption (HE)

Homomorphic Encryption (HE) is an encryption scheme that allows computation on encrypted data without decryption. Following Gentry's blueprint of HE in 2009 [20], there have been many studies about HE. The Cheon-Kim-Kim-Song (CKKS) scheme [14] is the fourth generation of HE that supports the encryption of a vector of real or complex numbers with predefined size. The CKKS scheme supports approximate arithmetic operations on real or complex vectors, including addition (Add), multiplication (Mul), and rotation (Rot), where Rot refers to cyclic slot-wise permutation. While rotation (Rot) is supported by CKKS and used implicitly during ciphertext evaluation (e.g., in bootstrapping), it is not discussed explicitly in this paper. The allowed vector size is called the *number of slots*. The allowed maximum number of multiplication of CKKS is called *depth*, which is predefined in the key generation. If the number of multiplications exceeds the *depth*, then the decryption result is not guaranteed. The remaining allowed number of multiplication in a ciphertext is called *level* and denoted as L. The scale factor Δ guarantees the precision of the ciphertext. Add (C) and Mul (C) denote element-wise operations between two ciphertexts, e.g., $Add(C(\mathbf{v_1}), C(\mathbf{v_2})) = C(\mathbf{v_1} \oplus \mathbf{v_2})$. Add (P) and Mul (P) operate between a ciphertext and a plaintext vector (or a constant).

3.2 Bootstrapping in the CKKS Scheme

If a ciphertext's level reaches zero, further multiplication operations cannot be performed. To overcome this limitation, Gentry [20] proposed *bootstrapping*, a technique that refreshes a ciphertext's level back to its initial ciphertext. Following Cheon [13], numerous studies have been conducted to optimize the bootstrapping process in the CKKS scheme [5, 6, 12].

However, bootstrapping in the CKKS scheme remains computationally expensive and time-consuming, as it involves encrypted Fourier transform operations.

Table 1 summarizes the runtime (in milliseconds) of addition, multiplication, and bootstrapping operations in the Lattigo [1] library. We set the ring degree to $N = 2^{16}$ with a scale factor of $\Delta = 2^{40}$. The total modulus size $\log_2(PQ)$ is set to 1443 bits, and the number of slots is 32,768.

Each value represents the mean runtime over ten trials, with the standard deviation shown in parentheses.

Table 1: Runtime comparison of HE operations in Lattigo (depth = 11). All values are in milliseconds, averaged over ten trials.

Operation	Addition	Multiplication	Bootstrapping
Runtime (ms)	6.41 (1.27)	196.35 (6.69)	43478.77 (302.74)

As shown in Table 1, bootstrapping requires more than 221.44× the runtime of multiplication.

3.3 Newton's Method

The Newton's Method (Newton-Raphson Method [3]) is an incremental algorithm to find a root α of a given function f. It starts with an initial value x_0, calculate the updating equation $x_{n+1} = x_n - f(x_n)/f'(x_n)$ until the error between the estimated value x_{n+1} and the real value α. In this section, we only consider Newton's method to find an inverse square root. It is described in Algorithm 1).

Algorithm 1 *Newton's method for the inverse square root operation*

1: **Input:** An initial point x_0, y_0 is an initial approximation at x_0, d is the number of iterations
2: **Output:** An approximation y_d.
3: **for** $i = 1$ **to** d **do**
4: $\quad y_i \leftarrow 0.5 \cdot y_{i-1} \cdot (3 - x_0 \cdot y_{i-1}^2)$
5: **end for**
6: **return** y_d

Newton's method converges very fast when the initial value x_0 is close to the real value α, but if x_0 is far from α, then it may diverge. Thus, it is important to find a good initial value x_0 for finding the suitable estimated result of the Newton's method.

3.4 Polynomial Approximation

The CKKS scheme allows arithmetic operations such as addition and multiplication, but only supports polynomial operations. Non-polynomial functions, such as square root or absolute value, cannot be directly supported in CKKS. Many previous approaches about implementing non-polynomial type of function use the approximation methods to convert it to a polynomial function. For instance, the approximated polynomial of the sign function can be obtained using the minimax approximation methods [15, 24, 25], and sine function used in the modulus evaluation in the bootstrapping obtained by Chebyshev approximation [12, 22].

3.5 Approximation Algorithm for Inverse Square Root

To find an inverse square root of a given encrypted value, authors in [29] obtained a square root using Newton's method first with Goldschmidt's algorithm to get the inverse of it. However, for fast convergence, Goldschmidt's algorithm requires a suitable initial point of $1/\sqrt{x}$.

Panda et al [30] suggest a novel method *Pivot-Tangent*, a 2-line approximation to find a polynomial of an approximation of inverse square root using the pivot point and sign function. It overcomes the challenge of finding a good initial point for faster Newton's iteration, they split a domain $[a, b]$ as two intervals $[a, P]$ and $[P, b]$ as a pivot point P, and combine them using a function $\beta(x) = step(P/(b-a), x/(b-a))$ where $step(x) = (1 + sign(x))/2$. The initial point $h(x)$ of x using the Newton's method can be derived as follows:

$h(x) = (1 - \beta(x)) \cdot L_1(x) + \beta(x) \cdot L_2(x)$ where

$$L_1(x) = -0.5 \cdot k_2 \cdot x_1^{-1.5} \cdot x + 1.5 \cdot k_2/\sqrt{x_1}$$
$$L_2(x) = -0.5 \cdot k_2 \cdot x_2^{-1.5} \cdot x + 1.5 \cdot k_2/\sqrt{x_2}$$

In [30] a method to find the pivot point P and parameters k_1, k_2, x_1, x_2 and some examples of them are suggested. The convergence time of the algorithm is about half of them in [29] using both Newton's method and Goldschmidt's algorithm.

4 Overview of *PP-STAT*

We present *PP-STAT*, a HE-based framework for privacy-preserving statistical analysis over encrypted client data. An overview of the system architecture is shown in Figure 1.

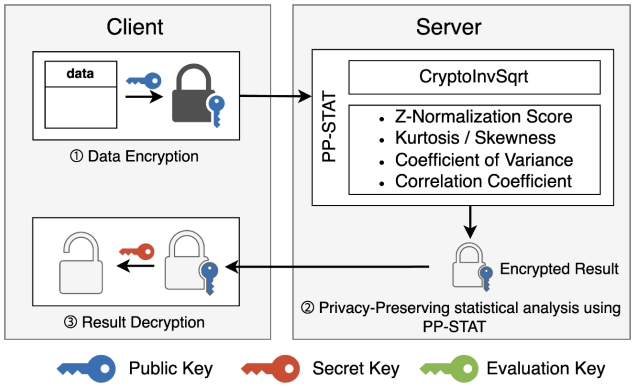

Figure 1: System overview of *PP-STAT*.

PP-STAT supports advanced statistical operations including Z-score normalization, skewness, kurtosis, coefficient of variation, and Pearson correlation coefficient. These operations require inverse or inverse square root operations, which are not natively supported by the CKKS scheme. To address this, *PP-STAT* introduces an optimized Chebyshev polynomial approximation—*CryptoInvSqrt* for computing both $1/x$ and $1/\sqrt{x}$—and reuses them across all statistical functions to reduce depth and improve computational efficiency.

The system follows a standard client-server model:

① The client encrypts sensitive input data using the public key and sends it to the server along with evaluation keys. The secret key is kept private and never leaves the client.
② The server performs statistical operations over the encrypted data using *PP-STAT*'s supported operations.
③ The client receives the encrypted results and decrypts them locally to obtain the final statistical outputs.

Section 5 provides the technical details of the *CryptoInvSqrt* construction. Section 6 presents the algorithms used to compute the five supported statistical measures.

5 HE-Based Inverse and Inverse Square Root Computation

In this work, we define a unified function *CryptoInvSqrt* to represent the n-th inverse root operation of the form $x^{-1/n}$. This general form covers both inverse ($n = 1$) and inverse square root ($n = 2$), and is used throughout the paper for consistent construction and optimization under HE. The operation serves as a core computational primitives for several statistical functions in *PP-STAT*, including Z-score normalization, covariance, kurtosis, skewness, and Pearson correlation coefficient.

5.1 HE implementation of the inverse square root

CryptoInvSqrt is designed to estimate suitable initial values for Newton's method in computing inverse n-th square roots under encryption. Notably, the inverse operation corresponds to the case where $n = 1$. Algorithm 2 describes a HE implementation of Newton's method for approximating the inverse n-th square root of an encrypted input. The accuracy of this method critically depends on the initial guess y_0, which should closely approximate the true inverse n-th root of the encrypted input x. However, since x is encrypted in the HE setting, selecting a suitable y_0 is challenging.

Algorithm 2 *HE-based Newton's method for inverse n-th square root*

Input:
- ct_x: Ciphertext of x.
- ct_0: Ciphertext of y_0.
- d: Number of iterations

Output:
- ct_d: Approximation ciphertext.

if $n \geq 2$ **then**
  $ct_x \leftarrow Mul(P)(ct_x, 1/n)$
end if
for $i = 1$ **to** d **do**
  $tmp_a \leftarrow Mul(P)(ct_{i-1}, (n+1)/n)$
  $tmp_b \leftarrow Mul(C)(ct_x, ct_{i-1})$
  for $j = 1$ **to** n **do**
    $tmp_b \leftarrow Mul(C)(tmp_b, ct_{i-1})$
  end for
  $ct_i \leftarrow Sub(C)(tmp_a, tmp_b)$
end for
return ct_d

Lee et al. [26] use a fixed constant for the initial value y_0 (e.g., $y_0 = 1$), which leads to slow convergence. Their method requires 25 and 21 Newton iterations for inverse n-th root with $n = 1$ and $n = 2$, respectively. This results in deep computation graphs and a total of five bootstrapping operations to maintain noise levels.

Panda et al. [30] improve the initialization by applying a homomorphic *sign* function, but their method still consumes six bootstrapping operations due to the additional depth required for polynomial evaluations and iterative refinement.

In contrast, *CryptoInvSqrt* approximates y_0 using a Chebyshev polynomial, which enables fast convergence in just 6 Newton iterations. Our method requires only two bootstrapping operations in total—less than half of Lee et al.'s and one-third of Panda et al.'s—while achieving higher accuracy and faster runtime. See Experiment 1 in Section 7.

Algorithm 3 Initial value estimation for inverse n-th square root using Chebyshev approximation

Input:
- ct_x: Ciphertext of x (assumed in $[0, 1]$)
- d: Degree of Chebyshev approximation
- n: Root degree (e.g., 1 for Inv, 2 for InvSqrt)

Output:
- ct_0: Initial approximation ciphertext

$ct_0 \leftarrow \text{ChebyshevApprox}(ct_x, d)$
$ct_0 \leftarrow \text{Bootstrapping}(ct_0)$
if $n == 2$ **then**
  $ct_0 \leftarrow \text{Mul}(ct_0, ct_0)$
end if
return ct_0

In Algorithm 3, a unified initial approximation is used for both inverse and inverse square root computations. Specifically, $\text{Inv}(x)$ can be computed by squaring $\text{InvSqrt}(x)$, eliminating the need for separate approximations.

5.2 Domain Mapping for *CryptoInvSqrt*

Since Chebyshev approximation is defined on the interval $[-1, 1]$, we set the function $F(x)$ as a shifted version of the inverse square root function to match this domain. The function is given as:

$$F(x) = \begin{cases} \frac{1}{\sqrt{x+1}} & \text{if } x > -1, \\ 0 & \text{if } x \leq -1. \end{cases} \tag{1}$$

This definition ensures that $F(x)$ is continuous on $[-1, 1]$ and can be approximated using Chebyshev polynomials. The shift by +1 maps the original domain $[0, 2]$ of the inverse square root to the standard Chebyshev domain. A polynomial $p_1(x)$ of fixed degree can be constructed to approximate $F(x)$ over the domain $[-1, 1]$ using Chebyshev approximation. This transformation shifts the Chebyshev approximation back to the original inverse square root domain. As a result, $\text{AppInvSqrt}(x)$ closely approximates $1/\sqrt{x}$ over $x \in [0, 2]$, while remaining Chebyshev-compatible in the computational pipeline.

6 Statistical Operations with *CryptoInvSqrt*

PP-STAT supports five advanced statistical operations: Z-score normalization, kurtosis, skewness, Pearson correlation coefficient, and coefficient of variation, as introduced in Section 1. All these measures fundamentally rely on inverse or inverse square root operations, which are implemented using the optimized *CryptoInvSqrt* method presented in Section 5. In this section, we detail the HE-based constructions of these operations and describe how our optimizations reduce multiplicative depth and improve runtime. For brevity, we omit the full description of skewness, as its computational structure closely mirrors that of kurtosis.

6.1 Pre-normalization Scaling: Reducing Depth via Constant Folding

To ensure stable bootstrapping and accurate polynomial approximation, inputs to *CryptoInvSqrt* must lie within a bounded domain—typically $[-1, 1]$ for bootstrapping and $[0, 2]$ for Chebyshev approximation. However, intermediate results such as variance and mean often exceed these ranges in practical applications. For example, the standard deviation in Z-score normalization can exceed 2, placing its inverse square root outside the approximable range of Chebyshev polynomials. To address this, we rescale intermediate values by multiplying the input ciphertexts with a constant factor $1/B$, where B is a predefined bound. Unlike the conventional normalization method—which applies scaling after computing statistics such as variance and thus consumes an extra multiplicative level—our *pre-normalization scaling* embeds the constant factor directly into the computation of mean and variance. This eliminates the additional multiplicative level overhead without compromising numerical correctness. Since computing variance already requires two levels (due to squaring and averaging), our method keeps the total depth prior to applying *CryptoInvSqrt* at two instead of three. This transformation is shown in Equation 2.

$$B \cdot \mathrm{Var}(X) = B \cdot \left(E[X^2] - E[X]^2\right) \tag{2}$$

$$= B \cdot \left(\sum \left(X/\sqrt{N}\right)^2 - \left(\sum X/N\right)^2\right) \tag{3}$$

$$= \sum \left(X/\sqrt{N/B}\right)^2 - \left(\sum X/(N/\sqrt{B})\right)^2 \tag{4}$$

This manipulation folds the constant B into the normalization coefficients, allowing it to be applied earlier via plaintext multiplication. As a result, we reduce the cost of computing variance from three multiplicative levels to two before applying *CryptoInvSqrt*. To reflect this rescaling in the approximation, we modify the target function for inverse square root operation as:

$$F(x, B) = \begin{cases} \frac{1}{\sqrt{B}\cdot\sqrt{x+1}} & \text{if } x \geq -1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

We define $\texttt{CryptoInvSqrt}(x, B)$ as the Chebyshev approximation of $F(x, B)$ in Equation 5, ensuring numerical compatibility with the scaled variance. This strategy can also be applied to compute the square root of variance, as used in the coefficient of variation. For that, we define a scaled square root function:

$$F(x, B) = \begin{cases} \sqrt{B} \cdot \sqrt{x+1} & \text{if } x \geq -1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Let $\texttt{CryptoSqrt}(x, B)$ denote the Chebyshev polynomial that approximates Equation 6. We also apply this pre-normalization scaling technique to the mean operation:

$$\mathrm{Mean}(x, B) := \sum \left(\frac{x_i}{B \cdot N}\right) = \frac{1}{B} \cdot \mathrm{Mean}(x)$$

$$\mathrm{Variance}(x, B) := \sum \left(\left(\frac{x_i}{B \cdot \sqrt{N}}\right)^2\right) - \left(\sum \left(\frac{x_i}{B \cdot N}\right)\right)^2 = \frac{1}{B^2} \cdot \mathrm{Var}(x)$$

This constant-folding strategy reduces multiplicative depth by one level and is applicable to both mean and variance. An additional benefit of this method is that it enables the use of higher-degree Chebyshev polynomials under the same depth constraint. For example, in Z-score normalization, the maximum allowable Chebyshev degree is $2^8 - 1$ under depth 11 without pre-normalization scaling. With this technique, we can raise the polynomial degree to $2^9 - 1$ without increasing the number of bootstrapping rounds, thereby improving approximation accuracy without incurring additional computational cost.

6.2 Z-Score Normalization (ZNorm)

Z-score normalization is a standard statistical technique used to standardize values across datasets by removing the influence of differing means and variances. It is computed as the difference between a value and the mean, divided by the standard deviation:

$$\mathrm{ZNorm}(X) = \frac{X - \mu}{\sigma}$$

Given that the standard deviation is the square root of the variance, computing its reciprocal reduces to evaluating the inverse square root of the variance. This can be efficiently approximated using *CryptoInvSqrt*, enabling low-depth Z-score normalization under HE. We apply a scaling factor B during the variance operation to fit the input into the valid approximation range. The corresponding correction factor B^2 is passed to `CryptoInvSqrt` to ensure numerical consistency. The complete procedure is presented in Algorithm 4.

Algorithm 4 HE-based Z-score Normalization (ZNorm)

Input: ct_X: Ciphertext of input vector X
Output: ct_{znorm}: Ciphertext of $\mathrm{ZNorm}(X)$

1: $ct_\mu \leftarrow \texttt{Mean}(ct_X)$
2: $ct_{var} \leftarrow \texttt{Variance}(ct_X, B)$
3: $ct_b \leftarrow \texttt{CryptoInvSqrt}(ct_{var}, B^2)$
4: $ct_a \leftarrow \texttt{Sub}(C)(ct_X, ct_\mu)$
5: $ct_{znorm} \leftarrow \texttt{Mul}(C)(ct_a, ct_b)$
6: **return** ct_{znorm}

6.3 Kurtosis (Kurt)

Kurtosis is a statistical measure that quantifies the heaviness of the tails of a distribution relative to a normal distribution. It is defined as the ratio of the fourth central moment to the square of the variance:

$$\mathrm{Kurt}[X] = \frac{E[(x - E[X])^4]}{(E[(x - E[X])^2])^2}$$

In HE, we compute the fourth central moment and the square of the variance homomorphically. The denominator is strictly non-negative, allowing us to apply an inverse square root approximation to its square for better depth efficiency. Algorithm 5 outlines the full procedure.

Algorithm 5 HE-based Kurtosis (Kurt)

Input: ct_X: Ciphertext of input vector X
Output: ct_{kurt}: Ciphertext of Kurt(X)

$ct_\mu \leftarrow \texttt{Mean}(ct_X)$
$ct_{x1} \leftarrow \texttt{Sub}(ct_X, ct_\mu)$
$ct_{x2} \leftarrow \texttt{Mul}(C)(ct_{x1}, ct_{x1})$
$ct_{x4} \leftarrow \texttt{Mul}(C)(ct_{x2}, ct_{x2})$
$ct_{numerator} \leftarrow \texttt{Mean}(ct_{x4})$
$ct_{var} \leftarrow \texttt{Variance}(ct_X, B)$
$ct_{inv} \leftarrow \texttt{CryptoInvSqrt}(ct_{var}, B^2)$
$ct_{inv2} \leftarrow \texttt{Mul}(C)(ct_{inv}, ct_{inv})$
$ct_{inv4} \leftarrow \texttt{Mul}(C)(ct_{inv2}, ct_{inv2})$
$ct_{kurt} \leftarrow \texttt{Mul}(C)(ct_{numerator}, ct_{inv4})$
return ct_{kurt}

6.4 Coefficient of Variation (CV)

The coefficient of variation (CV) is a statistical measure defined as the ratio of the standard deviation to the mean of a given dataset. Unlike other statistical operations, CV involves a denominator—the mean—that can be negative. This poses a challenge for polynomial approximations such as `CryptoInvSqrt`, which are typically defined over non-negative domains. To address this, we define the inverse of the mean as $1/\mu = \mathit{CryptoInvSqrt}(\mu^2)$. However, since the square of the inverse square root is always positive, this approach does not produce the correct sign when the mean is negative. To handle this, we first extract the sign of the mean using a sign function and then apply it after the inverse operation. This enables compatibility with the *CryptoInvSqrt* function while preserving the correct output sign. The CV is then computed as follows, where Std denotes the standard deviation:

$$\mathrm{CV}(x) = \mathrm{Std}(x) \cdot \frac{\mathrm{sign}(\mu)}{|\mu|} = \mathrm{Std}(x) \cdot \mathit{CryptoInvSqrt}(\mu^2) \cdot \mathrm{sign}(\mu)$$

Algorithm 6 summarizes the implementation.

Algorithm 6 HE-based Coefficient of Variation (CV)

Input: ct_X: Ciphertext of input vector X
Output: ct_{cv}: Ciphertext of CV(X)

$ct_\mu \leftarrow \texttt{Mean}(ct_X, B)$
$ct_{sign} \leftarrow \texttt{Sign}(ct_\mu)$
$ct_{\mu\_sqr} \leftarrow \texttt{Mul}(C)(ct_\mu, ct_\mu)$
$ct_{\mu\_inv} \leftarrow \texttt{CryptoInvSqrt}(ct_{\mu\_sqr}, B)$
$ct_{var} \leftarrow \texttt{Variance}(ct_X, B)$
$ct_{std} \leftarrow \texttt{CryptoSqrt}(ct_{var}, B)$
$ct_{tmp} \leftarrow \texttt{Mul}(C)(ct_{std}, ct_{\mu\_inv})$
$ct_{cv} \leftarrow \texttt{Mul}(C)(ct_{tmp}, ct_{sign})$
return ct_{cv}

6.5 Pearson Correlation Coefficient (PCC)

The Pearson correlation coefficient (PCC) is a statistical measure of linear dependence between two random variables. It is defined as the ratio of their covariance to the product of their standard deviations:

$$\mathrm{PCC}(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)} \cdot \sqrt{\mathrm{Var}(Y)}}$$

Lee et al. [26] proposed a homomorphic algorithm for computing PCC using the CKKS scheme. In our implementation, we compute the centered versions of X and Y, then apply mean, multiplication, and inverse square root operations homomorphically. The inverse square roots are approximated using *CryptoInvSqrt*, as in other statistical functions. The overall procedure follows Algorithm 3 in [26] for detailed pseudocode.

7 Experiment

7.1 Experimental Setup

We evaluate the precision and computational efficiency of *PP-STAT* under standard CKKS homomorphic encryption settings. All experiments are implemented using Lattigo v6 [1], an open-source HE library written in Go that supports approximate arithmetic over real numbers. For a fair and consistent comparison, we reimplemented the inverse n-th square root algorithms proposed in *Pivot-Tangent* and *HEaaN-STAT* within the same Lattigo environment.

Hardware Configuration. All experiments were conducted on a server equipped with an Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz processor, 64GB RAM, and 500GB SSD. All implementations were executed in single-thread mode to ensure consistency across all evaluations.

Parameter Settings. We set the ring dimension (polynomial degree) to $N = 2^{16}$ with a maximum depth of 27 and the scale factor to $\Delta = 2^{40}$. The total modulus size $\log_2(PQ)$ is set to 1443 bits, which supports up to 11 levels of multiplicative depth before requiring bootstrapping. For bootstrapping, $\log_2(PQ)$ is set to 744 bits. The number of slots is 32, 768, half of N. This configuration satisfies the 128-bit security level under the standard RLWE hardness assumptions [9].

7.2 Experiment 1: Performance Comparison of Inverse Square Root Operation

The inverse square root operation is a critical component in computing fundamental statistical measures such as Z-score normalization, kurtosis, skewness, and the Pearson correlation coefficient in *PP-STAT*. Given its central role, we evaluate the accuracy and efficiency of the proposed *CryptoInvSqrt* method in comparison to existing techniques, specifically those proposed in *HEaaN-STAT* and *Pivot-Tangent*. In this experiment, we measure the MRE of inverse square root operations across three methods: *HEaaN-STAT*, *Pivot-Tangent*, and our proposed *CryptoInvSqrt*. Panda et al. [30] utilize a function $G_3^{(7)}(x)$ as a *sign* function to generate a suitable initial value for Newton's method. $G_3^{(7)}(x)$ denotes the 7-fold composition of the degree-7 polynomial $g_3(x)$:

$$g_3(x) = \frac{35}{16}x - \frac{35}{16}x^3 + \frac{21}{16}x^5 - \frac{5}{16}x^7.$$

However, $G_3^{(7)}(x)$ exhibits instability in the narrow interval $|x| \leq 0.01$, leading to poor approximation behavior near zero. To address this, we adopt a minimax-optimized step function [24], which, while requiring 5 additional multiplicative levels, achieves significantly higher precision over the entire input domain. Figure 2 compares $G_3^{(7)}(x)$ (green) with the minimax approximation (blue).

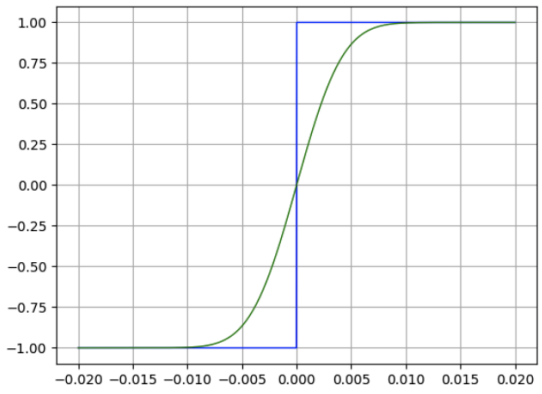

Figure 2: Comparison of step functions: $G_3^{(7)}(x)$ (green) vs. minimax approximation (blue).

Both *Pivot-Tangent* and *CryptoInvSqrt* are designed to operate directly over the input domain $[0.001, 100.0]$ without any preconditioning. In contrast, the original *HEaaN-STAT* method requires inputs in the interval $(0, 1]$. To enable a fair comparison, we scale the inputs for *HEaaN-STAT* by a factor of $1/100$, and subsequently rescale the outputs by $\sqrt{100}$ to align with the full domain of the inverse square root function. To assess approximation accuracy across a wide dynamic range, we evaluate the MRE over 32,768 real-valued inputs sampled from $[0.001, 100]$. The domain is partitioned into two subranges to capture both low- and high-value behaviors: 16,384 inputs in $[0.001, 1.0]$ (spacing $\approx 6.10 \times 10^{-5}$), and 16,384 in $[1.0, 100]$ (spacing $\approx 6.04 \times 10^{-3}$).

Table 2 summarizes the evaluation results. Here, #BTS denotes the number of bootstrapping operations, and Runtime (s) is measured using single-core execution. All values are averaged over 10 runs, with standard deviations shown in parentheses. Unlike *Pivot-Tangent*, which supports only the narrow input domain $[0, 1]$, *HEaaN-STAT* requires normalization when operating over broader ranges. To ensure compatibility, we scale the input data by a constant factor of $1/B$, where $B = 100$. This same factor is also applied inside *CryptoInvSqrt* to maintain consistency with the transformed domain.

Table 2: Performance comparison of inverse square root operation over the input domain $[0.001, 100]$. The parameter B denotes the constant scaling factor. All values are averaged over ten runs; values in parentheses represent standard deviations.

Method	B	#BTS	MRE	Runtime (s)
HEaaN-STAT	-	5	5.36×10^{-3} (1.05×10^{-7})	225.02 (3.18)
Pivot-Tangent	100.0	6	3.73×10^{-4} (1.68×10^{-9})	273.48 (5.29)
CryptoInvSqrt	100.0	2	5.08×10^{-5} (1.75×10^{-9})	94.73 (1.05)

As shown in Table 2, *CryptoInvSqrt* achieves the highest precision among all methods, with a MRE of 5.08×10^{-5}. While all methods adopt Newton's method for inverse square root approximation, *CryptoInvSqrt* utilizes a Chebyshev-based strategy to determine an optimized initial guess, significantly reducing the number of Newton iterations and associated bootstrapping operations. As shown in Table 1, bootstrapping is approximately 221.44× slower than homomorphic multiplication. Consequently, *CryptoInvSqrt* requires fewer bootstrapping rounds and achieves the fastest runtime. To accommodate the wide input domain $[0.001, 100]$, we set the constant scaling factor B to 100, enabling accurate approximation within the Chebyshev-valid range.

Compared to *Pivot-Tangent*, *CryptoInvSqrt* achieves approximately 7.34× lower MRE while consuming less than half the number of bootstrapping operations. Relative to *HEaaN-STAT*, *CryptoInvSqrt* delivers over 105.51× better accuracy and reduces runtime by a factor of 2.38× (94.73s vs. 225.02s). These results highlight the effectiveness of our Chebyshev-based initialization in reducing both multiplicative depth and latency in homomorphic inverse square root operation.

7.3 Experiment 2: Accuracy on Large-Scale Dataset

In this experiment, we evaluate the MRE of five core statistical operations in *PP-STAT*: Z-score normalization, skewness, kurtosis, PCC, and CV. For scalar-valued metrics (e.g., kurtosis, skewness), MRE corresponds to the relative error between the decrypted result and its plaintext reference. Z-score normalization is applied to raw input distributions with wide dynamic ranges, while the remaining operations are computed over normalized data. Since Chebyshev polynomial approximation suffers from increased error over wide domains, we evaluate Z-score normalization over $[0, 100]$ and the other measures over $[0, 20]$. Each experiment is conducted on one million independently sampled real-valued inputs to simulate large-scale data processing. Table 3 summarizes the accuracy and runtime.

Table 3: Accuracy and efficiency of Z-score normalization evaluated over the domain $[0, 100]$, and the remaining four statistical measures over $[0, 20]$. All values are averaged over 10 trials; values in parentheses denote standard deviations. The parameter B denotes the constant scaling factor used for normalization.

Measure	B	#BTS	MRE	Runtime (s)
ZNorm	100	2	4.18×10^{-5} (6.06×10^{-6})	141.29 (1.55)
Skew	20	2	8.12×10^{-3} (1.41×10^{-2})	154.10 (1.61)
Kurt	20	2	3.73×10^{-4} (6.97×10^{-6})	154.70 (1.34)
CV	20	7	1.25×10^{-4} (9.70×10^{-5})	311.08 (2.94)
PCC	20	4	2.62×10^{-4} (3.21×10^{-5})	289.86 (3.47)

Abbreviations: ZNorm = Z-score normalization, skew = skewness, kurt = kurtosis

Table 3 shows that Z-score normalization achieves high accuracy, with an MRE of 4.18×10^{-5} over the wide input domain $[0, 100]$. To ensure compatibility with this domain, we apply a constant scaling factor of $B = 100$ to align the inputs to the Chebyshev-valid range.

Among the five operations, four—Z-score normalization, skewness, kurtosis, and PCC—require computing the inverse square root of variance. In contrast, the CV additionally involves computing the inverse of the mean. These intermediate values often exceed the valid Chebyshev approximation domain $[0, 2]$, which may lead to

substantial approximation error. To address this, we normalize the variance and mean using a constant scaling factor $B = 20$, applying $1/B^2 = 1/400$ and $1/B = 1/20$, respectively. This transformation ensures compatibility with the approximation domain, while correctness is preserved through the multiplicative homomorphism of CKKS and symmetric post-processing.

Unlike variance, which is always non-negative, the mean can be negative. To correctly compute the inverse in such cases, we adopt a sign-aware approach: we extract the sign using the minimax approximation [25], apply *CryptoInvSqrt* to the absolute value, and reapply the sign. This is implemented using a minimax-optimized polynomial of multiplicative depth 32, requiring three bootstrapping rounds. As a result, CV incurs the highest bootstrapping cost (seven rounds) due to its multi-phase computation, including standard deviation calculation, sign-aware inverse mean approximation, and additional polynomial evaluations. In contrast, the other measures require at most four bootstrapping rounds, reflecting their lower depth complexity and higher runtime efficiency.

7.4 Experiment 3: Evaluation on Real-world Datasets

To evaluate the practical applicability of *PP-STAT* in real-world settings, we conduct experiments on two widely used datasets: the UCI Adult Income Dataset [7] and the Medical Cost Insurance Dataset [4]. For brevity, we refer to them as *Adult* and *Insurance*, respectively.

7.4.1 Dataset Descriptions. The *Adult* dataset contains 48,842 records with 14 attributes, including `age`, `hours-per-week`, and `education-num`. It is commonly used to predict whether an individual's income exceeds $50,000. The *Insurance* dataset consists of 1,338 samples across 7 features such as `age`, `bmi`, `smoker`, and `charges`, and is often used in regression and cost modeling tasks.

Using *PP-STAT*, we compute several statistical measures—including Z-score normalization, skewness, kurtosis, CV, and PCC—on selected features from both datasets. Decrypted outputs are compared to plaintext baselines to assess numerical accuracy.

7.4.2 Evaluation on the UCI Adult Income Dataset. We select three continuous-valued features from the Adult income dataset (*Adult* dataset): `age`, `education-num`, and `hours-per-week`. For each feature, we compute the mean relative error (MRE) for Z-score normalization as well as for four additional statistical measures: skewness, kurtosis (reported as excess kurtosis), CV, and PCC. In addition, we compute the PCC between the feature pairs (`age, hours-per-week`) and (`age, education-num`). We empirically set the constant scaling factor B to 50, based on the observation that the means of the selected features—particularly `hours-per-week` typically fall between 30 and 50. The results are summarized in Table 4. As shown in Table 4, *PP-STAT* maintains high accuracy across all metrics. Z-score normalization, skewness, and kurtosis yield MREs below 3×10^{-3}, while PCC results show errors consistently under 1.4×10^{-4}. CV results vary depending on feature scale, with increased error observed in `education-num`, which has a relatively small mean. This behavior aligns with the known sensitivity of CV to small denominators.

Table 4: Evaluation of statistical operations over the *Adult* dataset. We report MRE for Z-score normalization (ZNorm), kurtosis (Kurt), skewness (Skew), and coefficient of variation (CV) across selected features. PCC denotes the Pearson correlation coefficient between feature pairs. The parameter B denotes the constant scaling factor. All values are averaged over ten trials; values in parentheses indicate standard deviations.

Measure	Feature(s)	B	Output	MRE	Runtime (s)
ZNorm	AGE	50	-	2.47×10^{-5} (3.39×10^{-21})	110.18 (2.42)
	EDU	50	-	1.02×10^{-4} (1.36×10^{-20})	110.03 (1.40)
	HPW	50	-	7.62×10^{-5} (1.36×10^{-20})	109.73 (1.71)
Skew	AGE	50	0.5576	7.92×10^{-5} (1.36×10^{-20})	113.92 (1.83)
	EDU	50	-0.3165	2.50×10^{-4} (0.00)	112.32 (2.02)
	HPW	50	0.2387	1.38×10^{-4} (0.00)	111.81 (1.68)
Kurt	AGE	50	-0.1844	4.81×10^{-3} (0.00)	113.20 (2.07)
	EDU	50	0.6256	2.14×10^{-3} (0.00)	113.02 (1.58)
	HPW	50	2.9506	7.76×10^{-4} (0.00)	112.36 (1.52)
CV	AGE	50	0.3548	4.39×10^{-5} (6.78×10^{-21})	297.60 (3.84)
	EDU	50	0.2551	1.10×10^{-3} (2.17×10^{-19})	295.72 (3.92)
	HPW	50	0.3065	4.82×10^{-5} (6.78×10^{-21})	295.68 (3.08)
PCC	AGE vs HPW	50	0.0716	1.40×10^{-4} (2.71×10^{-20})	222.39 (3.05)
	AGE vs EDU	50	0.0309	1.01×10^{-4} (0.00)	222.38 (3.35)

Abbreviations: AGE = age, EDU = education-num, HPW = hours-per-week

7.4.3 Evaluation on the Medical Cost Insurance Dataset. To assess the applicability of *PP-STAT* to real-world privacy-preserving statistical analysis tasks, we conduct experiments on the Medical Cost Insurance Dataset (*Insurance* dataset), which includes sensitive personal and medical information often used in actuarial studies. We select three representative features: `age`, `bmi`, and `smoker`, which are frequently analyzed in medical cost prediction. Categorical or discrete features such as `sex`, `children`, and `region` are excluded to simplify encrypted operation and ensure reproducibility. Both `age` and `bmi` are continuous variables ranging from 18 to 64 and 15.96 to 53.13, respectively. The `smoker` attribute is binary, and we map it to numerical values (yes $\rightarrow$ 1, no $\rightarrow$ 0) to enable encrypted computation of PCC. The target variable `charges` ranges from 1121.87 to 63770.43; we scale it by a factor of 1/1000 before encryption to match the CKKS dynamic range. Since the PCC is scale-invariant, this transformation does not affect the final result. This configuration enables privacy-preserving statistical analysis on encrypted features and target variables, allowing us to extract meaningful patterns without exposing sensitive data. We set $B = 100$ for Z-score normalization, and $B = 20$ for all other evaluations. Table 5 summarizes the evaluation results. As shown in Table 5, *PP-STAT* achieves high accuracy across all evaluated statistical functions. Z-score normalization, skewness, and CV produce MREs below 8×10^{-4}, highlighting the framework's robustness even for high-variance numerical attributes. Pearson correlation values involving `age` and `bmi` also maintain low error. In contrast, the PCC involving the binary `smoker` feature shows higher error due to its limited dynamic range, which affects the stability of the denominator in correlation operation. As shown in Table 5, all relative errors lower than 3×10^{-4}, demonstrating the high numerical precision of *PP-STAT* in encrypted computation. The skewness of *charges* is measured as 1.5143, indicating a right-skewed distribution. This

Table 5: Evaluation of statistical metrics using *PP-STAT* over the *Insurance* dataset. We report the mean relative error (MRE) for each statistical function compared to the plaintext result. PCC denotes the PCC between the target feature (charges) and selected predictors. The parameter B denotes the constant scaling factor. Kurtosis is reported as excess kurtosis (i.e., normal kurtosis minus 3).

Measure	Feature(s)	B	Output	MRE	Runtime (s)
ZNorm	charges	100	-	3.81×10^{-5} (0.00)	108.26 (3.34)
Skew	charges	20	1.5143	8.67×10^{-5} (1.36×10^{-20})	105.52 (2.36)
Kurt	charges	20	1.5966	6.08×10^{-4} (1.08×10^{-19})	104.94 (2.45)
CV	charges	20	0.9123	5.54×10^{-5} (6.78×10^{-21})	279.02 (5.11)
PCC	AGE vs charges	20	0.2990	2.22×10^{-4} (0.00)	207.87 (4.73)
	BMI vs charges	20	0.1983	7.33×10^{-5} (0.00)	207.85 (3.64)
	SMOKER vs charges	20	0.7873	2.86×10^{-4} (5.42×10^{-20})	209.98 (2.07)

suggests that most insurance fees are concentrated near the lower range, while a small subset of individuals are charged substantially higher fees. The excess kurtosis is 1.5966, implying a leptokurtic distribution with heavy tails. This reflects the presence of extreme outliers—individuals who incur unusually high charges, a common pattern in real-world insurance data. The CV is 0.9123, suggesting that the standard deviation is nearly as large as the mean. This highlights the substantial variability in insurance fees across the population, and underscores the need for secure statistical tools in sensitive domains. Z-score normalization achieves MRE well below 10^{-4}, validating *PP-STAT* 's ability to support encrypted feature scaling with high precision. Figure 3 illustrates the kernel density estimation (KDE) of *charges*, visually confirming the right-skewness and heavy-tailed structure observed in our numerical evaluation.

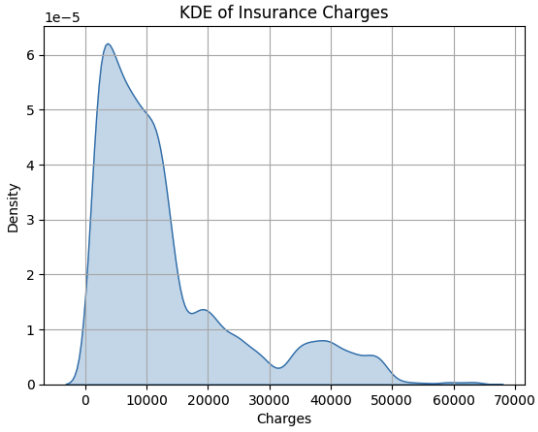

Figure 3: Kernel density estimation (KDE) of *charges* in the *insurance* dataset.

In the PCC evaluation, the correlation between `smoker` and `charges` is 0.7873, indicating a strong positive association. This indicates that smoking status is a dominant factor affecting medical costs. The PCC between `age` and `charges` is 0.2990, reflecting a moderate positive correlation—insurance fees tend to increase with age. Meanwhile, the PCC for `bmi` is 0.1983, suggesting a weaker but still positive relationship. These results demonstrate that *PP-STAT* enables accurate and meaningful higher-order statistical analysis on encrypted medical cost data. The ability to perform normalization, distributional analysis, and correlation estimation directly in the encrypted domain underscores the practical value of *PP-STAT* in privacy-preserving healthcare analytics.

8 Discussion and Future Work

In this work, we focus on the Chebyshev approximation for determining suitable initial values in Newton's iteration. Beyond engineering optimizations, *PP-STAT* introduces key technical innovations, including pre-normalization scaling to reduce multiplicative depth, Chebyshev-based initialization for fast *CryptoInvSqrt* convergence. However, potential vulnerabilities arising from polynomial approximation errors—such as information leakage through output deviation patterns—were not deeply analyzed in this study. In future work, we plan to conduct a comprehensive security analysis of such approximation-based vulnerabilities. Currently, *PP-STAT* supports only five statistical operations. We plan to investigate more optimized operations, such as minimum and maximum. The computation of CV incurs a considerably large runtime due to the use of the sign function, which requires three bootstrapping operations; optimizing the CV algorithm to reduce this cost is therefore necessary. While our current focus is on encrypted statistical analysis, extending *PP-STAT* to privacy-preserving machine learning (ML) pipelines is a natural next step. This includes enabling secure feature selection, model training, and inference on encrypted datasets. At present, *PP-STAT* is implemented using Lattigo, a Go-based HE library that operates in CPU-only environments. Since HE operations are computationally intensive with limited throughput in such settings, we plan to explore hardware acceleration via GPU-based HE libraries and dedicated computing platforms such as FPGAs or ASICs. Furthermore, we aim to extend *PP-STAT* to support secure multi-party computation (MPC), federated analytics, and differential privacy (DP) mechanisms. Such integration will broaden the applicability of *PP-STAT* in collaborative, distributed, and privacy-regulated environments.

9 Conclusion

In this work, we presented *PP-STAT*, a homomorphic encryption (HE)-based framework for privacy-preserving statistical analysis. *PP-STAT* supports five fundamental statistical measures—Z-score normalization, skewness, kurtosis, coefficient of variation, and Pearson correlation coefficient—directly over encrypted data. A key contribution is our optimized inverse square root operation, achieving 7.34× lower mean relative error (MRE) and 2.38× faster runtime compared to prior approaches such as *Pivot-Tangent* and *HEaaN-STAT*. This reduces multiplicative depth and bootstrapping frequency, enabling practical computation of higher-order statistics under HE. We also proposed a *pre-normalization scaling* technique that eliminates the additional multiplicative level without additional cost. We validated *PP-STAT* on two real-world datasets—*Adult* and *Insurance*—and demonstrated accurate and efficient encrypted computation across diverse statistical tasks. These results confirm that *PP-STAT* provides a practical and scalable solution for secure data analysis in privacy-sensitive domains.

GenAI Usage Disclosure

This paper was entirely written by the authors. During the preparation of the manuscript, we used ChatGPT (developed by OpenAI) solely for the purpose of grammar correction, word choice refinement, and improvement of sentence clarity. No content, sentence, or paragraph was generated by the model without substantial author input. All scientific claims, experimental results, and explanations were written and validated by the authors.

References

[1] 2024. Lattigo v6. Online: https://github.com/tuneinsight/lattigo. EPFL-LDS, Tune Insight SA.

[2] Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Guy Moshkowich, Dov Murik, Hayim Shaul, and Omri Soceanu. 2023. HeLayers: A Tile Tensors Framework for Large Neural Networks on Encrypted Data. *Privacy Enhancing Technology Symposium (PETs) 2023* (2023). https://petsymposium.org/popets/2023/popets-2023-0020.php

[3] Saba Akram and Quarrat Ul Ann. 2015. Newton raphson method. *International Journal of Scientific & Engineering Research* 6, 7 (2015), 1748–1752.

[4] Nahida Akter and Ashadun Nobi. 2018. Investigation of the financial stability of S&P 500 using realized volatility and stock returns distribution. *Journal of Risk and Financial Management* 11, 2 (2018), 22.

[5] Youngjin Bae, Jung Hee Cheon, Wonhee Cho, Jaehyung Kim, and Taekyung Kim. 2022. Meta-bts: Bootstrapping precision beyond the limit. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 223–234.

[6] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, and Damien Stehlé. 2024. Bootstrapping bits with CKKS. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 94–123.

[7] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[8] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. 2021. TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. arXiv:2104.03152 [cs.CR]

[9] Jean-Philippe Bossuat, Rosario Cammarota, Ilaria Chillotti, Benjamin R Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, et al. 2024. Security guidelines for implementing homomorphic encryption. *Cryptology ePrint Archive* (2024).

[10] Giovanni Campisi, Luca La Rocca, and Silvia Muzzioli. 2023. Assessing skewness in financial markets. *Statistica Neerlandica* 77, 1 (2023), 48–70.

[11] Gizem S Cetin, Yarkin Doroz, Berk Sunar, and William J Martin. 2015. Arithmetic using word-wise homomorphic encryption. *Cryptology ePrint Archive* (2015).

[12] Hao Chen, Ilaria Chillotti, and Yongsoo Song. 2019. Improved bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 34–54.

[13] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2018. 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, 360–384.

[14] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. 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, 409–437.

[15] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. 2020. 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, 221–256.

[16] Jung Hee Cheon, Dongwoo Kim, Duhyeong Kim, Hun Hee Lee, and Keewoo Lee. 2019. Numerical method for comparison on homomorphically encrypted numbers. In *International conference on the theory and application of cryptology and information security*. Springer, 415–445.

[17] Hyunmin Choi, Jihun Kim, Seungho Kim, Seonhye Park, Jeongyong Park, Wonbin Choi, and Hyoungshick Kim. 2024. UniHENN: Designing Faster and More Versatile Homomorphic Encryption-based CNNs without im2col. *IEEE Access* (2024).

[18] Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. 2019. CHET: an optimizing compiler for fully-homomorphic neural-network inferencing. In *Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation*. 142–156.

[19] Joost CF De Winter, Samuel D Gosling, and Jeff Potter. 2016. Comparing the Pearson and Spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data. *Psychological methods* 21, 3 (2016), 273.

[20] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 169–178.

[21] Robert E Goldschmidt. 1964. *Applications of division by convergence*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[22] Kyoohyung Han and Dohyeong Ki. 2020. Better bootstrapping for approximate homomorphic encryption. In *Cryptographers' Track at the RSA Conference*. Springer, 364–390.

[23] Takumi Ishiyama, Takuya Suzuki, and Hayato Yamana. 2020. Highly accurate CNN inference using approximate activation functions over homomorphic encryption. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 3989–3995.

[24] Eunsang Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. 2022. Optimization of homomorphic comparison algorithm on rns-ckks scheme. *IEEE Access* 10 (2022), 26163–26176.

[25] Eunsang Lee, Joon-Woo Lee, Jong-Seon No, and Young-Sik Kim. 2021. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing* 19, 6 (2021), 3711–3727.

[26] Younho Lee, Jinyeong Seo, Yujin Nam, Jiseok Chae, and Jung Hee Cheon. 2023. HEaaN-STAT: a privacy-preserving statistical analysis toolkit for large-scale numerical, ordinal, and categorical data. *IEEE Transactions on Dependable and Secure Computing* (2023).

[27] Amal Saki Malehi, Fatemeh Pourmotahari, and Kambiz Ahmadi Angali. 2015. Statistical models for the analysis of skewed healthcare cost data: a simulation study. *Health economics review* 5 (2015), 1–16.

[28] Mustafa Pamuk. 2022. The Role of Parental Academic Pressure in Adolescent-Parent Conflict: An Investigation by Gender. *International Online Journal of Educational Sciences* 14, 5 (2022).

[29] Samanvaya Panda. 2021. Principal component analysis using ckks homomorphic encryption scheme. *Cryptology ePrint Archive* (2021).

[30] Samanvaya Panda. 2022. Polynomial approximation of inverse sqrt function for FHE. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*. Springer, 366–376.

[31] Maurice V Wilkes, David J Wheeler, Stanley Gill, and FJ Corbató. 1958. The preparation of programs for an electronic digital computer.