# Fully Privacy-Preserving and Efficient Clustering Scheme based on Fully Homomorphic Encryption

Mengyu Zhang*, Long Wang*, Xiaoping Zhang†, Yisong Wang†, Wenhou Sun†
* *Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China*
† *Department of Computer Science and Technology, Tsinghua University, Beijing, China*
{mengyu-z23@mails., longwang@, zhxp@, wys19@mails., swh21@mails.}tsinghua.edu.cn

***Abstract*—Fully Homomorphic Encryption is a cryptographic scheme to prevent the privacy leakage of sensitive data. However, one challenge with FHE is the ciphertext comparison widely used in clustering, an important scheme used for data mining and analysis. In this paper, we create a series of ciphertext comparison functions by rewriting the comparison in HE-friendly operations and the Heaviside step function approximated by Chebyshev Polynomials. Furthermore, we solve the challenging ciphertext division through constructing a function whose root is the reciprocal of the divisor and applying Newton's method to approximate the root. Then, we propose a fully privacy-preserving, effective and efficient clustering scheme based on our ciphertext comparison and division. Tests on various datasets show that our algorithm maintains nearly the same classification accuracy as vanilla k-means and significantly outperforms the baseline in terms of accuracy. We then optimize our scheme by batching over multiple records and multithreading, and the results show that our approach has a significant efficiency advantage. Compared with the state-of-the-art FHE-based privacy-preserving clustering scheme (SAC 2018), our algorithm is four orders of magnitude faster than theirs.**

***Index Terms*—Fully Homomorphic Encryption, privacy-preserving, clustering**

## I. Introduction

Fully Homomorphic Encryption (FHE) is a cryptographic primitive that protects the private information contained in data by implementing homomorphic operations on encrypted data. FHE can easily support addition and multiplication, but the ciphertext comparison is challenging, which severely limits its scope of application. Since comparison is widely used in many scenarios, such as clustering, it is necessary to create a ciphertext comparison scheme based on FHE.

Clustering is a significant unsupervised machine learning task that reveals intrinsic patterns in many real-world data mining problems. In many application scenarios, the dataset to be clustered is split vertically, that is, two parties hold different dimensions of the same records. For example, a bank wants to perform cluster analysis on customers, but it is not accurate enough to cluster customers only by using the customer information held by the bank. The government owns citizens' private information, such as income status, which can help the bank obtain more accurate customer clustering results. It should be noted that government data is generally stored on a specific server and cannot be transmitted to any organization or individual in any form. Therefore, the bank (the data owner) needs to send the data to the government (the server), which clusters the joint dataset and returns the results. Since directly providing customer information to the government will result in leakage of private information, a privacy-preserving clustering scheme needs to be proposed. It should be considered that these two organizations typically communicate over a WAN, where network latency and bandwidth are not guaranteed. FHE-based algorithms are more suitable for the above scenarios since they allow non-interactive settings. Therefore, it is necessary to propose an efficient clustering scheme based on FHE.

Some privacy-preserving clustering schemes [1]–[4] based on FHE leak partial private information to service providers during the calculation, such as clusters' sizes or distances between data and clusters' centers. In [5], two non-colluding cloud servers are included in their system architecture, and partial private information is disclosed to the server computing the plaintexts. Jäschke and Armknecht [6] propose a fully privacy-preserving clustering scheme based on FHE, which can ensure that no private information is revealed during the calculation. However, their expensive bitwise ciphertext comparison leads to impractical run times (about 6 days for 400 records).

In this paper, we propose an approximate and fast ciphertext comparison and apply it to achieve a fully privacy-preserving, effective and efficient clustering scheme based on FHE. We choose the approximate homomorphic encryption algorithm CKKS [7], which can encode floating point numbers as our encryption algorithm.

One challenge with CKKS is the comparison in ciphertexts, which is not supported by CKKS directly. As CKKS can only compute arbitrary polynomials, we rewrite the comparison by HE-friendly operations and the Heaviside step function approximated by Chebyshev Polynomials. We propose a series of ciphertext comparison functions, including a function to calculate the maximum/minimum value and a function to return the label of the maximum/minimum value. We apply our ciphertext comparison scheme to the clustering problem to demonstrate its effectiveness and efficiency.

Furthermore, the division involved in the clustering algorithm is also a challenge for CKKS. We construct a function whose root is the reciprocal of the divisor and apply Newton's method to approximate this root. After slightly modifying the clustering algorithm, we implement a fully privacy-preserving clustering algorithm, which leaks no private information to the

ICC 2024 - IEEE International Conference on Communications | 978-1-7281-9054-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICC51166.2024.10622837

server.

We perform our privacy-preserving clustering scheme on several widely used datasets to evaluate the clustering accuracy. Experimental results show that the accuracy of our scheme is nearly equal to that of vanilla k-means on all datasets, and better than that of the state-of-the-art privacy-preserving clustering algorithm [6].

Then, we optimize our algorithm through batching and multithreading, and test the run time of our scheme. To demonstrate the superior efficiency of our algorithm, we compare the run time of our optimized algorithm to a state-of-the-art FHE-based privacy-preserving clustering scheme [6]. The results show that our algorithm is four orders of magnitude faster than theirs. Our solution allows data owners with large datasets and high-efficiency requirements to enjoy outsourced clustering services with complete privacy protection.

The main contributions of this work are summarized as follows:

- We solve ciphertext comparison and division respectively through polynomial approximation and construction of the iterative formula.
- Applying the ciphertext comparison and division schemes and making minor changes to the clustering algorithm, we achieve a fully privacy-preserving and efficient clustering scheme.
- We test our algorithm on various datasets and compare our work with a state-of-the-art privacy-preserving clustering scheme. The results show that our work has a significant efficiency advantage while maintaining nearly the same classification accuracy as vanilla k-means.

## II. Problem Statement

In this section, we present related works, background knowledge, system architecture, and threat model.

### A. Related Works

In recent years, privacy-preserving clustering has been extensively studied. Differential privacy [8]–[13] is used to preserve privacy in clustering. Some literature [14]–[17] leverages Partially Homomorphic Encryption (PHE) and two non-colluding clouds, designing interactive protocols between them. However, communication cost in these works is high. Some previous works [18], [19] apply Multiparty Computation (MPC), which is sensitive to network bandwidth [20].

Focusing on FHE, some works [1]–[4] leak intermediate values to the server, such as clusters' sizes or distances between data and clusters' centers. In [5], the system structure includes two non-colluding cloud servers, and some private information will be leaked to the server that calculates plaintext. Different from these works, our algorithm is fully secure, which means no private information is leaked to the server.

Jäschke and Armknecht [6] propose a fully privacy-preserving k-means clustering scheme based on FHE, which allows data owners to outsource clustering without revealing private information. To improve efficiency, they use approximate comparison instead of exact comparison, by truncating several least significant bits in ciphertexts. However, the total run time of the dataset with 400 records is 25.79 days, which is far from practical for large datasets. Different from the bitwise approximate comparison presented in [6], the more efficient approximate comparison proposed in our work is achieved by approximating the Heaviside step function with a polynomial function. The results show that our algorithm is five orders of magnitude faster than theirs.

### B. Background Knowledge

*1) Approximate Homomorphic Encryption CKKS:* We choose the approximate homomorphic encryption scheme CKKS [7] as our encryption scheme. Let $N$ be a power of two and $R = \mathbb{Z}[X]/(X^N + 1)$ be a ring of integers. A message $\mathbf{m} \in \mathbb{C}^{N/2}$ is firstly encoded into a polynomial $m \in R$ called plaintext. Let $L$ be a level parameter, $q_l = 2^l$ for $1 \leq l \leq L$ and $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ be a modulo-q quotient ring of $R$. $[\cdot]_q$ denotes a modulo $q$ operation on each element of $R_q$. The distribution $X_{\mathrm{s}} = HWT(h)$ outputs a polynomial with $\{-1, 0, 1\}$-coefficients, whose Hamming weight is $h$. And the distributions $X_{\mathrm{enc}}$ and $X_{\mathrm{e}}$ are the discrete Gaussian distribution. Set the secret key as $sk \leftarrow (1, s)$ and the public key as $pk \leftarrow ([-a \cdot s + e]_{q_L}, a)$, with $s \leftarrow X_{\mathrm{s}}$, $a \leftarrow U(R_{q_L})$, $e \leftarrow X_{\mathrm{e}}$. The plaintext $m$ is encrypted by computing $c = [v \cdot pk + (m + e_0, e_1)]_{q_L}$, where $v \leftarrow X_{\mathrm{enc}}$ and $e_0, e_1 \leftarrow X_{\mathrm{e}}$. Decrypt the ciphertext $c = (c_0, c_1)$ by computing $m' = [c_0 + c_1 \cdot s]_{q_l}$.

*2) Chebyshev Polynomials:* We choose Chebyshev polynomials to approximate functions. The roots of the first class of Chebyshev polynomials are widely used in polynomial interpolation. The corresponding interpolation polynomial minimizes the Runge phenomenon and provides the best consistent approximation of the polynomial over continuous functions. Therefore, the first class of Chebyshev polynomials has significant applications in approximation theory. The Chebyshev polynomials [21] are defined by

$$T_n(x) = \cos(n \arccos x), \quad -1 \leq x \leq 1, \ n = 0, 1, \ldots \quad (1)$$

### C. System Architecture

In this section, we describe the system architecture shown in Figure 1. There are two actors, where one is the data owner and the other is the server. We focus on a vertically partitioned scenario, where the data owner and the server hold different dimensions of the same records. We denote the joint dataset as $D$, which contains $n$ records of $m$ dimensions. Suppose the data owner holds a dataset $D_1$ which contains $n$ records of $m_1$ dimensions, and the server owns a dataset $D_2$ containing the same $n$ records of $m_2$ dimensions, where $m = m_1 + m_2$. Previous works [22], [23] have proposed the private set intersection technique, so we can assume that $D_1$ and $D_2$ are already aligned. Therefore, $D = D_1 || D_2$.

In this scenario, in order to get better results, the data owner needs to send $D_1$ to the server which provides cloud computing services, and the server calculates the optimal solution of k-means clustering on the joint dataset $D$. Since the

server is untrusted, the data owner has to encrypt $D_1$ before sending it.

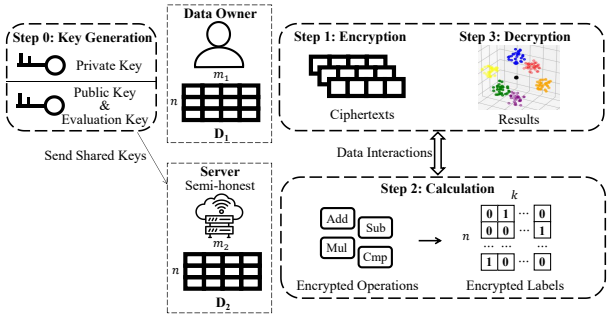


Fig. 1. Our System Architecture

First, the data owner generates the secret key $sk$ kept private by itself as well as the public key $pk$ and the evaluation key $evk$ both shared with the server. Then, the data owner encodes $n$ records into plaintexts and encrypts these plaintexts into ciphertexts. After that, the data owner delivers these ciphertexts and necessary parameters to the server, including $m_1$, the number of clusters denoted by $k$, and the number of iterations for clustering denoted by $T$. After receiving these ciphertexts and parameters, the server performs k-means clustering on the joint dataset, where $D_1$ is in ciphertext and $D_2$ is in plaintext. After the calculation completed, the server returns the clustering result in ciphertexts to the data owner. Finally, the data owner decrypts the ciphertext result and obtains the label of each record through simple calculation.

### D. Threat Model

In this paper, we assume a semi-honest security model, which means that the data owner and the server strictly enforce the protocols but might try to infer more private information. Private information refers to the records in datasets $D_1$ and $D_2$. $m_1, k, T$ are not private, since $m_1$ represents the dimensions of the records held by the data owner, and $k, T$ are only the parameters for clustering. Furthermore, we assume that the data channel is not secure, which means eavesdropping attacks might exist.

In our system, the dataset $D_1$ is homomorphically encrypted locally and then transmitted to the server in ciphertexts for calculation. Therefore, no eavesdropping attack can occur and no private information is leaked to the server. In addition, $D_2$ will not be transmitted to the data owner in any form, so the data owner cannot infer the server's private information.

Baiyu and Micciancio [24] show that the CKKS scheme does not satisfy indistinguishability under chosen plaintext attacks with decryption oracles (IND-CPA$^{\mathrm{D}}$) security. That is, when a passive adversary obtains the plaintext, homomorphic computation, encryption and evaluation output ciphertexts, and decryption results, it can launch a devastating key recovery attack. However, this attack does not occur in our system since the data owner never sends the decryption results to the server.

## III. Privacy-Preserving Clustering Scheme Based on Ciphertext Comparison

In this section, we describe our approximate and fast ciphertext comparison and apply it to achieve a fully privacy-preserving clustering scheme based on FHE.

### A. Ciphertext Comparison

The homomorphic encryption algorithm CKKS can easily support addition, subtraction, and multiplication, but the comparison in ciphertexts is challenging. Since comparison is widely used in many scenarios, such as clustering, it is necessary to propose a fast ciphertext comparison scheme to achieve complete privacy protection in the above scenarios. For fast ciphertext comparison, we trade a little bit of accuracy for a significant increase in efficiency, and propose an approximate and fast ciphertext comparison scheme.

Since the maximum and minimum values in $a$ and $b$ are calculated similarly, we only introduce the calculation of the minimum below. We define the output of the function $\min(a, b)$ as the smaller value of $a$ and $b$, where $a, b \in [0, 1]$. We introduce the Heaviside step function $H(x)$ where $H(x) = 1$ if $x \geq 0$ and $H(x) = 0$ if $x < 0$. Therefore, $\min(a, b)$ is rewritten as

$$\min(a, b) = \frac{a+b}{2} - H(a-b) \cdot \frac{a-b}{2} \tag{2}$$

However, we can not compute $H(x)$ in CKKS directly since it is not a polynomial function. We can only approximate $H(x)$ by a polynomial $f(x)$ in some finite domain. We choose the Chebyshev polynomials as $f(x)$ and $[-1, 1]$ as the domain. We can also handle the case where $a, b$ are not in the range $[0, 1]$. At this point, we need to know the range of $a$ and $b$. Let $a', b'$ be in $[0, M]$ for some constant $M$. We denote the approximate comparison as $\mathrm{approxmin}(a', b')$ and calculate it as follows:

$$\mathrm{approxmin}(a', b') = \frac{a'+b'}{2} - f(\frac{a'-b'}{M}) \cdot \frac{a'-b'}{2} \tag{3}$$

In many cases, it is not enough to return the minimum value, we want the output of the function to be a one-hot encoding vector that labels the minimum value as follows:

$$\mathrm{approxminlabel}(a_1, \ldots, a_n) = \mathrm{minvec} \tag{4}$$

where we expect $\mathrm{minvec_i}$ to be 1 if $a_i$ is the minimum value, and 0 otherwise. And $a_1, \ldots, a_n$ are in $[0, M]$ for some constant $M$.

Since the server can not obtain any intermediate results during the ciphertext calculation process, it is necessary to compare the $n$ numbers pairwise to find the minimum. Clearly, if $a_i$ is the minimum, then for any $1 \leq j \leq n$ and $j \neq i$, there are $f(\frac{1}{M}(a_j - a_i)) = 1$. If $a_i$ is not the minimum, then there exists $1 \leq j \leq n$ and $j \neq i$, such that $f(\frac{1}{M}(a_j - a_i)) = 0$.

Therefore, to compute $\mathrm{minvec_i}$, we compare the $i$-th number to the other $n-1$ numbers and multiply the results together, to determine whether it is the minimum:

$$\text{minvec}_\text{i} = \prod_{j=1, j\neq i}^{n} f(\frac{a_j - a_i}{M}) \tag{5}$$

CKKS can directly support the series of comparison functions proposed above, since they only contain addition, subtraction, and multiplication.

### B. Privacy-Preserving Lloyd's Algorithm

To demonstrate the effectiveness and efficiency of our approximate and fast ciphertext comparison scheme, we apply it to Lloyd's algorithm [25] that can solve k-means clustering problem. The reasons why we choose Lloyd's algorithm are (1) it is simple to implement; (2) it has effective local fine-tuning capability [26]. Our ciphertext comparison scheme can also be applied to other algorithms. After making minor changes to Lloyd's algorithm, we implement a fully privacy-preserving clustering algorithm.

*1) Preliminaries:* Suppose the dataset contains $n$ records $(r_1, \ldots, r_n)$, where each record $r_i$ is a real vector of dimension $m$ ($r_i = (r_{i1}, \ldots, r_{im}) \in \mathbb{R}^m$). The Lloyd's algorithm divides the dataset into $k$ clusters, where each cluster is a set of records. The centroids of the clusters, $c_1, \ldots, c_k (c_i \in \mathbb{R}^m)$, are the means of the records in the clusters.

**Algorithm 1:** The Lloyd's algorithm

Input $n$, $m$, $k$, $T$, $r_1, \ldots, r_n$
Initialize centroids $c_1, \ldots, c_k$ randomly
**repeat** $T$ **times**
  Calculate distance of each record to each centroid
  $$d_{ij} = \sum_{t=1}^{m} (r_{it} - c_{jt})^2 \quad (i = 1, \ldots, n, j = 1, \ldots, k)$$
  Find the closest centroid to each record
  $$\text{label}_i = \arg\min_{j=1}^{k} d_{ij} \quad (i = 1, \ldots, n)$$
  Count the number of records in each cluster
  $$\text{cnt}_j = \sum_{i=1}^{n} [\text{label}_i = j] \quad (j = 1, \ldots, k)$$
  Update centroids
  $$c_j = \frac{\sum_{i=1}^{n} r_i [\text{label}_i = j]}{\text{cnt}_j} \quad (j = 1, \ldots, k)$$
Output label and $c_1, \ldots, c_k$

If $\text{label}_i = j$, then $[\text{label}_i = j] = 1$, otherwise $[\text{label}_i = j] = 0$.

Algorithm 1 is the complete Lloyd's algorithm. According to our threat model in Section II-D, the data owner needs to send ciphertext records to the server, and Steps 4 to 7 must be computed in ciphertexts.

*2) Distance Calculation in Step 4:* As described in Section II-C, we focus on the vertical partition scenario. That is, for each record $r_i$, the data owner holds $r_{i1}, \ldots, r_{im_1}$ and the server holds $r_{i(m_1+1)}, \ldots, r_{im}$. To protect private information, the data owner needs to encrypt $r_{i1}, \ldots, r_{im_1}$ $(1 \leq i \leq n)$ and send ciphertext records to the server. Therefore, in Step 4, $r_{i1}, \ldots, r_{im_1}$ are in ciphertexts while $r_{i(m_1+1)}, \ldots, r_{im}$ are in plaintexts, where $1 \leq i \leq n$.

However, this situation does not affect the calculation of the distance between each record and each centroid, since CKKS supports addition, subtraction and multiplication of plaintext and ciphertext. The output of operating plaintext and ciphertext is in ciphertexts, thus each element in the distance matrix $d$ is in ciphertexts.

In the distance matrix $d$, $d_{ij}$ represents the squared Euclidean distance between $r_i$ and $c_j$, where $1 \leq i \leq n$ and $1 \leq j \leq k$. Since $d_{ij}$ is only used to compare with some other elements in $d$, we choose to compute the squared Euclidean distance to simplify the calculation.

*3) Substituting the Label Array with One-Hot Encoding Vectors:* In Step 5, we store the index of the closest centroid to each record in $\text{label}$. Considering Steps 6 and 7, where each element of $\text{label}$ is used as $k$ indicator variables for the $k$ clusters, it is natural to construct a one-hot encoding vector for each record. We denote $\text{belongs}_i$ as the one-hot vector for record $r_i$:

$$\text{belongs}_{ij} = \begin{cases} 1, & j = \text{label}_i \\ & (\min\{d_{i1}, \ldots, d_{ik}\} = d_{ij}) \\ 0, & j \neq \text{label}_i \\ & (d_{ij} \text{ is not the minimum}) \end{cases} \tag{6}$$

where $d_{ij}$ represents the squared Euclidean distance between $r_i$ and $c_j$. In the above equation, $i$ $(1 \leq i \leq n)$ is the index of the record, and $j$ $(1 \leq j \leq k)$ is the index of the cluster. $\text{belongs}_{ij}$ equals one only when $c_j$ is the closest centroid to $r_i$. We can calculate $\text{belongs}_i$ according to Equation 4 as follows:

$$\text{belongs}_i = \text{approxminlabel}(d_{i1}, \ldots, d_{ij}) \tag{7}$$

After this transformation, Steps 6 and 7 could be immediately rewritten as follows:

$$\text{Step 6:} \quad \text{cnt}_j = \sum_{i=1}^{n} \text{belongs}_{ij} \tag{8}$$

$$\text{Step 7:} \quad c_j = \frac{\sum_{i=1}^{n} r_i \text{belongs}_{ij}}{\text{cnt}_j} \tag{9}$$

*4) Ciphertext Division in Step 7:* In Step 7, we need to divide the sum $\sum_{i=1}^{n} r_i \cdot \text{belongs}_{ij}$ by the count $\text{cnt}_j$ to get the new centroid $c_j$, where the ciphertext division is not supported by CKKS. We apply Newton's method [27] to approximate $\frac{1}{\text{cnt}_j}$. Construct $g(x) = \frac{1}{x} - \text{cnt}_j$ and approximate $\frac{1}{\text{cnt}_j}$ (the root of $g(x)$) by the following iterative formula:

$$x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)} = x_i(2 - \text{cnt}_j \cdot x_i) \tag{10}$$

We set $x_0$ to $\frac{2}{n}$, where $n$ represents the number of records contained in the dataset $D$. The above method of calculating division only uses subtractions and multiplications in ciphertexts, which are directly supported by CKKS.

*5) Converting the One-Hot Vectors to Label in Plaintexts:* The label array has been substituted with the belongs array in Step 5. After finishing the algorithm, we need to convert the belongs array back to the label array. Since errors exist in the computation of the belongs array, we extract the label array in an error-tolerant manner: the server sends the whole belongs array to the data owner, and the data owner finds the index of the maximum element in $\text{belongs}_{ij}$ for each record $i$ as $\text{label}_i$:

$$\text{label}_i = \arg\max_{j=1}^{k} \text{belongs}_{ij} \tag{11}$$

Extracting the label array according to Equation 11 can significantly reduce the impact of the errors in the belongs array on the classification accuracy. The reason is that, for each record $r_i$, when $c_j$ is the centroid closest to $r_i$, $\text{belongs}_{ij}$ is not required to be equal to 1. As long as $\text{belongs}_{ij}$ is greater than $\text{belongs}_{iq}$ $(1 \le q \le k, q \ne j)$, the data owner can obtain the correct $\text{label}_i$ according to Equation 11.

### C. The Complete Algorithm of Privacy-Preserving Lloyd's

**Algorithm 2:** Privacy-preserving Lloyd's algorithm

1 **Input** $n$, $m = m_1 + m_2$, $k$, $T$, $D = D_1 || D_2$
2 Initialize centroids $c_1, \ldots, c_k$ randomly in $[0,1]^m$
3 **repeat** $T$ **times**
4 Calculate distance of each record to each centroid
$d_{ij} = \sum_{t=1}^{m} (r_{it} - c_{jt})^2 \quad (i = 1, \ldots, n, j = 1, \ldots, k)$
5 Calculate the belongs array
$\text{belongs}_i = \text{approxminlabel}(d_{i1}, \ldots, d_{ij}) \quad (i = 1, \ldots, n)$
6 Count the number of records in each cluster
$\text{cnt}_j = \sum_{i=1}^{n} \text{belongs}_{ij} \quad (j = 1, \ldots, k)$
7 Update centroids
$c_j = \sum_{i=1}^{n} r_i \cdot \text{belongs}_{ij} \cdot \frac{1}{\text{cnt}_j} \quad (j = 1, \ldots, k)$
8 Send belongs array (in ciphertexts), and the data owner extracts label array from belongs array

We summarize the above steps and describe the complete algorithm of privacy-preserving Lloyd's (Algorithm 2). In Step 4, calculate the squared Euclidean distance between each record $r_i$ and each centroid $c_j$, denoted as $d_{ij}$. In Step 5, the server computes $\text{belongs}_i$ according to Equation 5, where $\text{belongs}_{ij} = 1$ indicates that $c_j$ is the centroid closest to $r_i$, and $\text{belongs}_{ij} = 0$ otherwise. In Step 6, count the number of records in each cluster.

In Step 7, we convert division into multiplication and subtraction through constructing a function whose root is the reciprocal of the divisor and applying Newton's method to approximate the root. Then, the server computes $k$ new centroids. Step 4 to 7 will be repeated for $T$ iterations. In Step 8, the data owner receives belongs and decrypts it, extracting label from it.

TABLE I
THE DATASETS CHOSEN FROM FCPS [29].

| Dataset | $n$ | $m$ | $k$ | Main Problems |
|---|---|---|---|---|
| Chainlink | 1000 | 3 | 2 | Not linearly separable |
| EngyTime | 4096 | 2 | 2 | Gaussian mixture |
| Hepta | 212 | 3 | 7 | None |
| Lsun | 400 | 2 | 3 | Different variances & shapes |
| Tetra | 400 | 3 | 4 | Almost touching |
| TwoDiamonds | 800 | 2 | 2 | Touching clusters |
| WingNut | 1016 | 2 | 2 | Non-uniform density |

Algorithm 2 uses only addition, subtraction, and multiplication in ciphertexts, which are all supported by CKKS. Also, the algorithm is completely secure, since the data owner only sends encrypted records to the server.

## IV. Experiment Results

In this section, we design multiple experiments to evaluate our fully privacy-preserving clustering scheme.

### A. Experiment Setup

*1) Server Configuration:* We implement our scheme using the Lattigo library [28] (version 4.1.0). We perform the experiments on a Ubuntu-22.04 server with an Intel Core i9-10900 CPU (20 threads) and 128GB RAM.

*2) Datasets:* To evaluate our scheme, we select several datasets from different sources. As shown in Table I, from the fundamental clustering problems suite (FCPS) [29] consisting of datasets with known a priori classifications, we choose seven datasets widely used in [30]–[34]. Besides, we choose five datasets (G2-1-20, G2-2-20, G2-4-20, G2-8-20, G2-16-20) from G2 [26] with different overlaps and dimensions, where $n = 2048$, $m \in \{1, 2, 4, 8, 16\}$ and `stdev=20`.

*3) Parameter Selection:* We choose the following parameters for our experiments:

- Parameters of CKKS: We select a default parameter set in Lattigo called PN16QP1761CI, where $\log N = 16$ and $\log QP = 1761$.
- Degree of Chebyshev Polynomial: 9.
- Number of iterations: 5, 10.

In all experiments, the initial $k$ centroids are uniformly and independently randomly sampled from $[0,1]^m$. We use five different random seeds to run each experiment and average the results.

As described in Section II-C, the joint dataset $D$ is vertically partitioned between the server and the data owner, where the data owner holds $D_1$ and the server holds $D_2$. It is obvious that for the same $D$, the larger $D_1$ leads to the longer run time of the privacy-preserving clustering algorithm, since $D_1$ is encrypted while $D_2$ is in plaintexts when calculating the distances. Note that operations between plaintext and ciphertext are faster than operations between ciphertexts. To make the results more convincing, we test the run times of our algorithm after encrypting all dimensions of the records in all experiments.

TABLE II
ACCURACY AND EFFICIENCY OF OUR ALGORITHM ON G2 AND FCPS.

| Dataset | Accuracy | | Run Time |
|---|---|---|---|
| | k-means | This work | This work |
| G2-1-20 | 99.4% | 99.5% | 173.96 s |
| G2-2-20 | 100.0% | 100.0% | 181.74 s |
| G2-4-20 | 100.0% | 100.0% | 204.72 s |
| G2-8-20 | 100.0% | 100.0% | 273.45 s |
| G2-16-20 | 100.0% | 100.0% | 428.44 s |
| Chainlink | 65.9% | 65.4% | 359.36 s |
| EngyTime | 95.1% | 92.2% | 339.05 s |
| Hepta | 85.8% | 97.6% | 837.54 s |
| Lsun | 76.8% | 78.3% | 352.15 s |
| Tetra | 100.0% | 100.0% | 474.87 s |
| TwoDiamonds | 100.0% | 100.0% | 338.79 s |
| WingNut | 96.4% | 96.3% | 338.66 s |

TABLE III
THE DROP IN CLUSTERING ACCURACY (exact − approximate).

| Work | Hepta | Lsun | Tetra | WingNut |
|---|---|---|---|---|
| Approximate Version in [6] | 4% | 3.5% | 13% | 1.25% |
| This Work | -11.8% | -1.5% | 0% | 0.1% |

### B. The Accuracy of Classification

In this section, we use various datasets to test the classification accuracy of our privacy-preserving clustering algorithm. Then, we compare the accuracy of our scheme with the state-of-the-art clustering algorithm proposed in [6].

We run experiments on datasets G2 and FCPS and calculate the classification accuracy of both our algorithm and vanilla k-means. In this part, we choose the 9th-degree Chebyshev polynomial for experiments and $T = 5$ for G2, and $T = 10$ for FCPS. Since the order of clusters in the experiments and the ground truths may be different, we enumerate full permutations of $\{1, \ldots, k\}$ as the mapping of labels from the algorithm outputs to the ground-truth, and find the one with most correctly classified records.

As shown in Table II, the accuracy of our algorithm is nearly equal to that of vanilla k-means for all datasets. Therefore, adding privacy protection to the clustering algorithm has almost no impact on accuracy.

As shown in Table III, we compare our scheme with the approximate version of the algorithm proposed in [6]. Bitwise ciphertext comparison is used in their algorithm. The approximate version improves the efficiency of the comparison by truncating several least significant bits. They demonstrate the accuracy of their approximate algorithm by calculating the difference in classification accuracy between the exact k-means clustering and the approximate version. To obtain comparable results, we calculate the difference in classification accuracy between k-means and our algorithm and compare our difference with theirs in each dataset (Table III). The negative numbers in the second row indicate that the accuracy of our algorithm is better than that of the k-means clustering. Our approximate algorithm suffers a much smaller loss in accuracy than theirs on Hepta and Tetra. To summarize, our privacy-preserving clustering algorithm is effective as it performs well in classification accuracy.

TABLE IV
PERFORMANCE COMPARISON WITH [6] ON LSUN.

| Work | Number of Threads | Run Time |
|---|---|---|
| Jäschke et al. [6] | 1 | 25.79 days |
| This work | 1 | 1116.24 s |
| This work | 20 | 352.15 s |

### C. Performance

In this section, we evaluate the run time of our privacy-preserving clustering scheme and compare it with the state-of-the-art FHE-based clustering scheme [6].

Since the direct implementation of our algorithm is inefficient, we optimize our scheme through batching and multithreading. We exploit the SIMD capabilities of CKKS to parallelize the operations. Denote $N$ as the polynomial modulus degree, so at most $N$ real values can be batched in a single ciphertext. Performing one operation on the ciphertext will result in performing the same function simultaneously on these $N$ values. Steps 4 and 5 in Algorithm 2 calculate the distances and the $\mathrm{belongs}$ array, which can be parallelized $\min(n, N)$ times by simply packing $\min(n, N)$ coordinates into a single ciphertext. Furthermore, we speed up our algorithm by multithreading. Since all operations in our scheme are independent of each cluster and each batch, we can enable multithreading across different batches and different clusters.

We record the run time of our optimized privacy-preserving algorithm (with batching and multithreading) on G2 and FCPS in Table II. In these experiments, we select the 9th-degree Chebyshev polynomial and $T = 5$ for G2, and $T = 10$ for FCPS. It can be seen that our privacy-preserving clustering scheme is efficient.

Then, we compare the run time of our algorithm with that of [6], which is the latest paper that I am aware of that proposes a fully privacy-preserving clustering scheme. Since this work is fully privacy-preserving, we choose it as the baseline. It should be noted that we directly use the run time reported in their original paper. In [6], they evaluate their scheme using a single thread, so we also test the single-threaded run time.

Table IV contains the run time of Algorithm 2 and the algorithm presented in [6] on Lsun. In [6], they only test their algorithm on Lsun, and the fastest version (approximate version) that trades accuracy for efficiency still costs a total run time of 25.79 days. Without multithreading, the run time of our algorithm is 1116.24 s, which is four orders of magnitude faster than theirs. Besides, our approach under full optimizations (batching and multithreading) is 6328x faster than theirs (Table IV). In summary, our privacy-preserving clustering algorithm has significant advantages in computational efficiency.

## V. Conclusions

We rewrite the comparison with HE-friendly operations and the Heaviside step function approximated by Chebyshev Polynomials, and create an approximate and fast ciphertext comparison scheme. We solve ciphertext division through constructing a function whose root is the reciprocal of the divisor and apply Newton's method to approximate the root. After minor modifications to the clustering algorithm, we propose a fully privacy-preserving, effective and efficient clustering scheme based on our ciphertext comparison and division. Experimental results show that our scheme achieves comparable accuracy to vanilla k-means and significantly outperforms the baseline in both accuracy and efficiency. In the future, we will try to use FPGA to accelerate the FHE-based clustering algorithm.

## References


[1] D. Liu, E. Bertino, and X. Yi, "Privacy of outsourced k-means clustering," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 123–134. [Online]. Available: https://doi.org/10.1145/2590296.2590332

[2] N. Almutairi, F. Coenen, and K. Dures, "K-means clustering using homomorphic encryption and an updatable distance matrix: Secure third party data clustering with limited data owner interaction," in *Big Data Analytics and Knowledge Discovery*, L. Bellatreche and S. Chakravarthy, Eds. Cham: Springer International Publishing, 2017, pp. 274–285.

[3] K.-P. Lin, "Privacy-preserving kernel k-means clustering outsourcing with random transformation," *Knowl. Inf. Syst.*, vol. 49, no. 3, p. 885–908, dec 2016. [Online]. Available: https://doi.org/10.1007/s10115-016-0923-2

[4] Y. Huang, Q. Lu, and Y. Xiong, "Collaborative outsourced data mining for secure cloud computing," *J. Networks*, vol. 9, pp. 2655–2664, 2014.

[5] W. Wu, J. Liu, H. Wang, J. Hao, and M. Xian, "Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 10, pp. 3424–3437, 2021.

[6] A. Jäschke and F. Armknecht, "Unsupervised machine learning on encrypted data," in *Selected Areas in Cryptography – SAC 2018*, C. Cid and M. J. Jacobson Jr., Eds. Cham: Springer International Publishing, 2018, pp. 453–478.

[7] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds. Cham: Springer International Publishing, 2017, pp. 409–437.

[8] L. Ni, C. Li, X. Wang, H. Jiang, and J. Yu, "Dp-mcdbscan: Differential privacy preserving multi-core dbscan clustering for network user data," *IEEE Access*, vol. 6, pp. 21 053–21 063, 2018.

[9] M.-F. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang, "Differentially private clustering in high-dimensional Euclidean spaces," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 322–331. [Online]. Available: https://proceedings.mlr.press/v70/balcan17a.html

[10] U. Stemmer, "Locally private k-means clustering," in *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '20. USA: Society for Industrial and Applied Mathematics, 2020, pp. 548–559.

[11] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin, "Differentially private k-means clustering," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 26–37. [Online]. Available: https://doi.org/10.1145/2857705.2857708

[12] Z. Huang and J. Liu, "Optimal differentially private algorithms for k-means clustering," in *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, ser. PODS '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 395–408. [Online]. Available: https://doi.org/10.1145/3196959.3196977

[13] L. Yuan, S. Zhang, G. Zhu, and K. Alinani, "Privacy-preserving mechanism for mixed data clustering with local differential privacy," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 19, p. e6503, 2023.

[14] F.-Y. Rao, B. K. Samanthula, E. Bertino, X. Yi, and D. Liu, "Privacy-preserving and outsourced multi-user k-means clustering," in *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*, 2015, pp. 80–89.

[15] W. Wu, J. Liu, H. Rong, H. Wang, and M. Xian, "Efficient k-nearest neighbor classification over semantically secure hybrid encrypted cloud database," *IEEE Access*, vol. 6, pp. 41 771–41 784, 2018.

[16] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "k-nearest neighbor classification over semantically secure encrypted relational data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1261–1273, 2015.

[17] H. Rong, H.-M. Wang, J. Liu, and M. Xian, "Privacy-preserving k-nearest neighbor computation in multiple cloud environments," *IEEE Access*, vol. 4, pp. 9589–9603, 2016.

[18] W. Wei, C. ming Tang, and Y. Chen, "Efficient privacy-preserving k-means clustering from secret-sharing-based secure three-party computation," *Entropy*, vol. 24, 2022.

[19] P. Mohassel, M. Rosulek, and N. Trieu, "Practical privacy-preserving k-means clustering," Cryptology ePrint Archive, Paper 2019/1158, 2019, https://eprint.iacr.org/2019/1158. [Online]. Available: https://eprint.iacr.org/2019/1158

[20] W.-j. Lu, Z. Huang, Q. Zhang, Y. Wang, and C. Hong, "Squirrel: A scalable secure two-party computation framework for training gradient boosting decision tree," *Cryptology ePrint Archive*, 2023.

[21] X. Wang, H. Chen, and Y. Jiang, "A modified krylov subspace model reduction method based on the chebyshev polynomials," in *2011 Second International Conference on Mechanic Automation and Control Engineering*, 2011, pp. 7123–7126.

[22] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 1–19.

[23] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 789–800.

[24] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers." Springer-Verlag, 2021.

[25] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[26] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied Intelligence*, vol. 48, no. 12, pp. 4743–4759, december 2018. [Online]. Available: https://doi.org/10.1007/s10489-018-1238-7

[27] A. Galántai, "The theory of newton's method," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 25–44, 2000, numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377042700004350

[28] "Lattigo v4," Online: https://github.com/tuneinsight/lattigo, Aug. 2022, ePFL-LDS, Tune Insight SA.

[29] A. Ultsch, "Clustering wih som: U* c," *Proc. Workshop on Self-Organizing Maps*, 01 2005.

[30] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues, "Clustering algorithms: A comparative approach," *PLOS ONE*, vol. 14, no. 1, pp. 1–34, 01 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0210236

[31] A. Ultsch, "Emergence in self organizing feature maps," in *The 6th International Workshop on Self-Organizing Maps (WSOM 2007)*, 2007. [Online]. Available: https://doi.org/10.2390/biecoll-wsom2007-114

[32] C. Lopez, S. Tucker, T. Salameh, and C. Tucker, "An unsupervised machine learning method for discovering patient clusters based on genetic signatures," *Journal of Biomedical Informatics*, vol. 85, pp. 30–39, 2018.

[33] P. A. Estévez and C. J. Figueroa, "Online data visualization using the neural gas network," *Neural Networks*, vol. 19, no. 6, pp. 923–934, 2006, advances in Self Organising Maps - WSOM'05.

[34] F. Li, Y. Qian, J. Wang, C. Dang, and L. Jing, "Clustering ensemble based on sample's stability," *Artificial Intelligence*, vol. 273, pp. 37–55, 2019.