

Article

A Fast and Privacy-Preserving Outsourced Approach for K-Means Clustering Based on Symmetric Homomorphic Encryption

Wanqi Tang and Shiwei Xu *

College of Informatics, Huazhong Agricultural University, Wuhan 430070, China;
wanqitang@webmail.hzau.edu.cn

* Correspondence: xushiwei@mail.hzau.edu.com

Abstract

Training a machine learning (ML) model always needs many computing resources, and cloud-based outsourced training is a good solution to address the issue of a computing resources shortage. However, the cloud may be untrustworthy, and it may pose a privacy threat to the training process. Currently, most work makes use of multi-party computation protocols and lattice-based homomorphic encryption algorithms to solve the privacy problem, but these tools are inefficient in communication or computation. Therefore, in this paper, we focus on the k-means and propose a fast and privacy-preserving method for outsourced clustering of k-means models based on symmetric homomorphic encryption (SHE), which is used to encrypt the clustering dataset and model parameters in our scheme. We design an interactive protocol and use various tools to optimize the protocol time overheads. We perform security analysis and detailed evaluation on the performance of our scheme, and the experimental results show that our scheme has better prediction accuracy, as well as lower computation and total overheads.

Keywords: k-means; fast and privacy-preserving; outsourced clustering; symmetric homomorphic encryption

MSC: 68P27



Academic Editor: Cheng-Chi Lee

Received: 3 July 2025

Revised: 1 August 2025

Accepted: 5 August 2025

Published: 8 September 2025

Citation: Tang, W.; Xu, S. A Fast and Privacy-Preserving Outsourced Approach for K-Means Clustering Based on Symmetric Homomorphic Encryption. *Mathematics* **2025**, *13*, 2893. <https://doi.org/10.3390/math13172893>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In modern life, we encounter vast amounts of big data in various fields. Data mining serves as a powerful tool to uncover hidden patterns and extract valuable insights from these massive datasets. However, running data mining algorithms requires significant computational and storage resources. When leveraging cloud-based computing and storage solutions to meet these demands, there is an inherent risk of privacy breaches, as sensitive data may be exposed to unauthorized access or leakage during processing and transmission.

Several privacy-preserving cryptography techniques, such as differential privacy (DP), homomorphic encryption (HE), and secure multi-party computation (SMPC), have been employed to protect sensitive data and algorithm parameters in data mining. The differential privacy technique safeguards individual data privacy by injecting noise into datasets, with high efficiency. However, the model parameters and noise-added data remain in plaintext, leading to weak privacy protection of the whole outsourced process. The technique allows for encrypted data mining but requires multiple nodes to perform ciphertext operations,

resulting in high computational and communication overheads. The homomorphic encryption technique allows for single-node computation to be carried out directly on ciphertext without inter-node communication, yet its adoption is constrained due to its computational inefficiency and restricted operational support in some HE schemes.

In this paper, we focus on the privacy-preserving outsourced clustering of k-means, which is a popular unsupervised machine learning (ML) algorithm used for clustering data into groups (clusters) based on similarity. It partitions a dataset into k distinct, non-overlapping clusters by minimizing the within-cluster variance (e.g., the sum of the squared distances between data points and their cluster centroids).

Similarly to other ML algorithms, various HE-based [1–5] and SMPC-based [6] schemes have already been proposed to implement privacy-preserving outsourced k-means clustering. However, constrained by the inherent characteristics of the adopted cryptographic techniques (i.e., HE and SMPC), the existing solutions suffer from either excessive communication overhead, high computational complexity, or suboptimal prediction accuracy. We will provide a detailed discussion of the related work in the subsequent section.

1.1. Related Work

Secret sharing (SS), which is a classic realization of SMPC, has been widely adopted to ensure secure outsourced computation and [7–9] makes use of different kinds of secret sharing protocols to implement privacy-preserving k-means outsourced clustering. Although SS-based methods offer strong security under standard assumptions, these protocols usually incur relatively heavy computational and communication overheads, due to the secret splitting and frequent interaction between multiple servers.

In addition to SMPC and SS, numerous studies leverage homomorphic encryption (HE) to preserve the privacy of k-means models and associated datasets. For instance, Ref. [3] employs TFHE [10], which replaces non-linear functions with programmable bootstrapping but suffers from extremely slow homomorphic addition and multiplication operations, severely hindering the efficiency of outsourced training. In contrast, works like [4,5] adopt alternative FHE schemes (i.e., BGV [11] and CKKS [12]), offering faster homomorphic operations than TFHE. However, BGV and CKKS also rely on the Ring-LWE problem, resulting in larger ciphertext sizes and slower encryption/decryption speeds. To obtain faster encryption/decryption speeds, researchers [2,7] have made use of Paillier [13], which is a kind of partially homomorphic encryption that only supports addition operations between ciphertexts. However, multiplication/division operations are carried out when distances are calculated; therefore, the researchers [2,7] needed to develop a secure multiparty ciphertext multiplication/division protocol, which increased the communication overhead between multiple servers.

In summary, we provide a hierarchical comparison between our SHE-based (symmetric homomorphic encryption [14]) work and related works including the deployed cryptographic tools, approximated computation, and the communication overhead of each training, based on the experimental results presented in Table 1.

Table 1. Comparison of related works (NOTE: SS and SHE separately denote secret sharing and symmetric homomorphic encryption).

Schemes	[7–9]	[2,7]	[3]	[5]	[4]	Our Work
Num of servers	>1	>1	1	1	1	1
Techniques	SS	Paillier [13]	TFHE [10]	BGV [11]	CKKS [12]	SHE [14]
Comp. overhead	medium	low	high	medium	medium	low
Comm. overhead	medium	medium	high	medium	medium	low

1.2. Example Applications of SHE

SHE was first proposed in [14] and used in the privacy-preserving range query in fog-based IoT. After [14], since SHE provided somewhat efficient (leveled) fully homomorphic encryption, a large amount of research work has used SHE to implement privacy-preserving and efficient federated learning [15–17], community/range/similarity queries in cloud computing, eHealthcare, and IoT [18–21]. Based on these successful applications of SHE, it should be feasible to integrate SHE with privacy-preserving outsourced k-means clustering in order to improve the efficiency of the outsourced scheme.

1.3. Our Contributions

To improve the computation and communication efficiency, we leverage SHE, a (leveled) fully homomorphic encryption scheme supporting efficient additive and multiplicative ciphertext operations, to safeguard the outsourced k-means training process. Our key contributions are as follows:

- During the protocol, we develop the Modified Euclidean Distance (MED) to avoid non-linear operations (e.g., division and square root) and reduce both the computational and communication costs.
- Comprehensive experiments and comparisons were conducted to demonstrate our solution's superiority in prediction accuracy, computational efficiency, and total runtime (including computation and communication).

2. Preliminaries

2.1. K-Means

K-means is an unsupervised learning algorithm that partitions n data points in a dataset into k clusters, where each data point is assigned to the cluster with the nearest centroid. The algorithm proceeds in the following steps:

Firstly, given a dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ ($i \in [1, n]$ and of the n d -dimensional data points into a predefined integer range), k cluster centroids $\mu_1, \mu_2, \dots, \mu_k$ are randomly initialized. Each data point $\mathbf{x}_i$ is then assigned to the nearest cluster by calculating the cluster index $c^{(i)} = \arg \min_j \|\mathbf{x}_i - \mu_j\|^2$, where $j \in \{1, 2, \dots, k\}$, and $c^{(i)}$ denotes the cluster assignment for $\mathbf{x}_i$.

Secondly, each centroid μ_j is recalculated as the mean of all data points assigned to its cluster C_j : $\mu_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$, where C_j is the set of points assigned to cluster j . After the new cluster centroids are updated, each data point $\mathbf{x}_i$ is then reassigned to the nearest new centroid using the same distance calculation as in the first step.

Thirdly, the algorithm iterates between the centroid update and data point assignment until the cluster assignments stabilize or the movement of the centroids falls below a specified threshold.

Upon convergence, the resulting model, defined by the final k centroids, can assign a new data point $\mathbf{x}_{\text{new}}$ by calculating $c_{\text{new}} = \arg \min_j \|\mathbf{x}_{\text{new}} - \mu_j\|^2$, where c_{new} is the predicted cluster index.

2.2. Symmetric Homomorphic Encryption (SHE)

The SHE scheme consists of the following three algorithms:

- $(sk, pp) \leftarrow \text{SHE.KeyGen}(k_0, k_1, k_2)$ is the key generation algorithm, which uses input security parameters k_0, k_1 , and k_2 (satisfying $k_1 \ll k_2 < (k_0/2)$) to randomly generate a secret key sk and public parameters pp . The secret key is $sk = (p, q, \mathcal{R})$, where p and q are two large prime numbers satisfying $|p| = |q| = k_0$, $\mathcal{R}$ is a random number

satisfying $|\mathcal{R}| = k_2$, and the message space $\mathcal{M} = \{m | m \in [-2^{k_1-1}, 2^{k_1-1})\}$. Then, $\mathcal{N} = pq$ is computed, and the public parameters pp are set to $(k_0, k_1, k_2, \mathcal{N})$.

- $\llbracket m \rrbracket_{sk} \leftarrow \text{SHE.Enc}(m, sk)$ is the encryption algorithm, which encrypts a message $m \in \mathcal{M}$ using the secret key $sk = (p, q, \mathcal{R})$. The ciphertext is computed as $\llbracket m \rrbracket_{sk} = \text{SHE.Enc}(m, (p, q, \mathcal{R})) = (r\mathcal{R} + m)(1 + r'p) \bmod \mathcal{N}$, where $r \in \{0, 1\}^{k_2}$ and $r' \in \{0, 1\}^{k_0}$ are randomly sampled. $m \leftarrow \text{SHE.Dec}(\llbracket m \rrbracket_{sk}, sk)$ is the decryption algorithm, which recovers the message m from a ciphertext $\llbracket m \rrbracket_{sk}$ using the secret key sk via the computation $m = (\llbracket m \rrbracket_{sk} \bmod p) \bmod \mathcal{R}$.

Given a plaintext m_0 and two ciphertexts, $\llbracket m_1 \rrbracket_{sk}$ and $\llbracket m_2 \rrbracket_{sk}$ (encrypted under the same key), the scheme supports the following homomorphic operations: $m_0 \oplus \llbracket m_1 \rrbracket_{sk} = \llbracket m_0 + m_1 \rrbracket_{sk}$, $m_0 \otimes \llbracket m_1 \rrbracket_{sk} = \llbracket m_0 m_1 \rrbracket_{sk}$, $\llbracket m_1 \rrbracket_{sk} \oplus \llbracket m_2 \rrbracket_{sk} = \llbracket m_1 + m_2 \rrbracket_{sk}$, and $\llbracket m_1 \rrbracket_{sk} \otimes \llbracket m_2 \rrbracket_{sk} = \llbracket m_1 m_2 \rrbracket_{sk}$, where $\oplus$ and $\otimes$, respectively, represent the homomorphic addition and multiplication. Furthermore, the scheme supports an arbitrary number of homomorphic additions but is limited to $\lfloor (k_0/2k_2) - 1 \rfloor$ sequential homomorphic multiplications.

Although SHE is a symmetric scheme, it can be transformed into an asymmetric one [15] by generating a public key $pk = r_0 \otimes \llbracket 0 \rrbracket_{sk}$, where r_0 is a k_2 -bit random number, and sk serves as the private key. Using this key pair, a message m is encrypted with the public key pk via the operation $\llbracket m \rrbracket_{pk} = m \oplus pk = \llbracket m \rrbracket_{sk}$, and the resulting ciphertext is decrypted with the private key sk using the original decryption algorithm: $m = \text{SHE.Dec}(\llbracket m \rrbracket_{pk}, sk)$.

3. System Design

3.1. System Architecture

Similar to previous works, our system architecture involves two entities: a data owner (DO) and a cloud service provider (CSP), where the DO owns the dataset but lacks sufficient computational resources, while the CSP provides on-demand cloud computing services. Figure 1 illustrates our outsourced k-means clustering protocol, an interactive process between the DO and the CSP. Before the protocol begins, the DO generates a key pair, normalizes the dataset, and encrypts the dataset and initial centroids, providing only the public keys to the CSP. During the protocol, the CSP is responsible for the bulk of the computational tasks, operating on the encrypted dataset and model parameters. However, because SHE does not support the direct comparison of ciphertexts, the CSP must return intermediate results to the DO for any necessary comparisons. The DO then decrypts, compares, re-encrypts, and returns the updated data to the CSP. Finally, upon convergence, the DO receives the encrypted final k-means model parameters from the CSP and decrypts them.

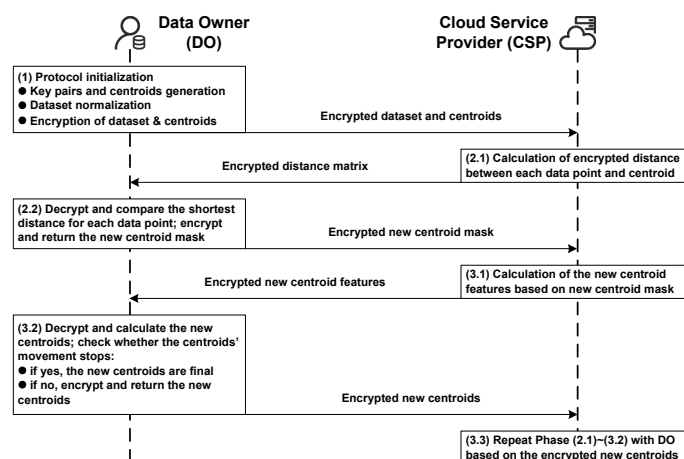


Figure 1. The overview of the outsourced training system architecture.

3.2. Security Models and Desired Properties

In our protocol's threat model, the D0 is assumed to be trusted, while the CSP is considered honest-but-curious with respect to the D0's dataset and model parameters. Accordingly, our protocol is designed to uphold the following security properties:

- *Dataset Privacy:* The protocol must ensure the CSP cannot learn the contents of the original clustering dataset provided by the D0.
- *Model Privacy:* The protocol must protect the confidentiality of the k-means model parameters throughout the entire outsourced clustering process. Specifically, CSP should not learn (i) the feature values of the cluster centroids or (ii) the distances between individual data points and these centroids.

To provide these security guarantees, we employ the (leveled) fully homomorphic encryption scheme SHE, which has been proven to be IND-CPA secure [19].

4. Protocol Implementation

In this section, we detail our k-means clustering protocol, aligning with the process illustrated in Figure 1.

4.1. Protocol Initialization

Before the protocol begins, the D0 executes the initialization step, which corresponds to Phase (1) in Figure 1.

Phase (1). The D0's clustering dataset consists of n data points. To begin, the D0 generates a SHE key pair, denoted as (pk_1, sk_1) . Because SHE only supports operations on integers, the D0 must normalize each feature, x_{ij} (for $i \in [1, n], j \in [1, d]$), of the n d -dimensional data points into a predefined integer range $[-2^{b-1}, 2^{b-1}]$, where $b \in \mathbb{Z}^+$.

Next, the D0 randomly generates k d -dimensional cluster centroids $(\mu_1, \mu_2, \dots, \mu_k)$ and similarly normalizes their features, μ_{lj} (for $l \in [1, k], j \in [1, d]$), into the same integer range. To facilitate subtraction between the data point and centroid features, the D0 negates the centroid features (μ_{lj}) , encodes both data features (x_{ij}) and the negated features $(-\mu_{lj})$ as complements, and finally encrypts them with sk_1 to obtain $\llbracket x_{ij} \rrbracket_{sk_1}$ and $\llbracket -\mu_{lj} \rrbracket_{sk_1}$.

Finally, the D0 sends all encrypted features of the data points and cluster centroids ($\llbracket x_{ij} \rrbracket_{sk_1}$ and $\llbracket -\mu_{lj} \rrbracket_{sk_1}$) to the CSP to begin the protocol's first round.

4.2. The First Protocol Round (Distance Calculation and Comparison)

The first protocol round corresponds to Phase (2.1)–(2.2) in Figure 1, where the primary tasks for the CSP and the D0 are distance calculation and comparison, respectively.

Phase (2.1). In this phase, using the encrypted features previously received from the D0, the CSP calculates the distance between each data point and every cluster centroid. Because SHE does not support the square root operation on ciphertexts, we use a Modified Euclidean Distance (MED) metric, which consists solely of addition, subtraction, and multiplication operations. For instance, the MED between the i -th data point x_i and the l -th cluster centroid μ_l is defined as

$$MED_{(x_i, \mu_l)} = \sum_{j \in [1, d]} (x_{ij} - \mu_{lj})^2.$$

To compute this MED, the CSP first homomorphically adds the corresponding encrypted features ($\llbracket x_{ij} \rrbracket_{sk_1}$ and $\llbracket -\mu_{lj} \rrbracket_{sk_1}$), then squares each resulting sum, and finally sums these squared results to obtain the encrypted MED value. As direct ciphertext comparison is not supported by SHE, the CSP then sends all the encrypted MED values back to the D0 for decryption and comparison.

Phase (2.2). Upon receiving the encrypted MEDs from the CSP, the D0 decrypts and compares them to determine the closest cluster centroid for each data point. On one hand, after the comparisons, the D0 counts the number of data points assigned to each centroid and generates a count vector $[cnt_1, cnt_2, \dots, cnt_k]$, where cnt_l is the number of points closest to centroid l , and $\sum_{l=1}^k cnt_l = n$. For example, the vector $[3, 5, 7, 4, 2]$ indicates that 3, 5, 7, 4, and 2 data points are closest to the first, second, third, fourth, and fifth centroids, respectively. This count vector is subsequently used for the centroid update step.

On the other hand, the D0 also generates a centroid mask for each data point i , denoted as a boolean vector $[msk_{i1}, msk_{i2}, \dots, msk_{ik}]$ (e.g., $[0, 0, 1, 0, 0]$ indicates the point is closest to the third centroid). The D0 then encrypts these n mask vectors and sends the resulting ciphertexts $(\llbracket msk_{i1} \rrbracket_{sk_1}, \dots, \llbracket msk_{ik} \rrbracket_{sk_1})$ for $i \in [1, n]$ to the CSP.

4.3. The Second Protocol Round (Centroid Calculation and Update)

The second protocol round corresponds to *Phase (3.1)–(3.3)* in Figure 1. During this round, the CSP and the D0 collaboratively calculate and update the cluster centroids. If the centroids have not yet converged, the CSP and the D0 repeat *Phase (2.1)–(3.2)* using the newly updated centroids.

Phase (3.1). After receiving the n encrypted centroid mask vectors from the D0, the CSP calculates the accumulated features for each new centroid according to the following equation:

$$\llbracket acc\_mu_{lj} \rrbracket_{sk_1} = \sum_{i \in [1, n]} \llbracket msk_{il} \rrbracket_{sk_1} \times \llbracket x_{ij} \rrbracket_{sk_1},$$

where $\llbracket x_{ij} \rrbracket_{sk_1}$ is the encrypted feature of the data point, and $\llbracket msk_{il} \rrbracket_{sk_1}$ is the corresponding encrypted mask value, $i \in [1, n]$, $j \in [1, d]$ and $l \in [1, k]$. Although the new centroid feature is defined as the accumulated feature divided by the number of associated data points, the CSP cannot perform this division directly on ciphertexts, as this operation is not supported by SHE. Therefore, the CSP must send the encrypted accumulated features, $\llbracket acc\_mu_{lj} \rrbracket_{sk_1}$, back to the D0 to perform the final calculation and update of the new centroids.

Phase (3.2). Upon receiving the encrypted accumulated features $\llbracket acc\_mu_{lj} \rrbracket_{sk_1}$, the D0 first decrypts them using sk_1 to obtain the plaintext accumulated features, acc_mu_{lj} . After decryption, the D0 calculates the new centroid feature, new_mu_{lj} , by dividing each accumulated feature acc_mu_{lj} by the corresponding data point count, cnt_l : $new\_mu_{lj} = \frac{acc\_mu_{lj}}{cnt_l}$ (for $j \in [1, d]$, $l \in [1, k]$).

Next, the D0 checks for convergence by comparing the previous centroid features, mu_{lj} , with the new ones, new_mu_{lj} . If $new\_mu_{lj} = mu_{lj}$ for all features, the centroids have converged, the clustering process is complete, and these are the final model parameters. Otherwise, the D0 encrypts the updated centroid features (new_mu_{lj}) with sk_1 and sends the resulting ciphertexts, $\llbracket new\_mu_{lj} \rrbracket_{sk_1}$, to the CSP for the next iteration.

Phase (3.3). Using the newly encrypted centroid features $\llbracket new\_mu_{lj} \rrbracket_{sk_1}$, the CSP and the D0 repeat *Phase (2.1)–(3.2)* until the clustering process converges.

5. Experimental Results and Analysis

The performance of our scheme was evaluated with the D0 deployed on a machine equipped with a 2-core 11th Gen Intel Core i7-1195G7 CPU (operating at 2.92 GHz), 4GB of RAM, and Windows 10. The CSP was hosted on a machine with a 4-core CPU of the same model (2.92 GHz), 16GB of RAM, and Windows 11. Notably, the hardware used in our experiments is less powerful than that of the related works we compare our work against; nevertheless, our scheme demonstrates superior time performance, as will be detailed later.

The datasets used in our experiments, listed in Table 2, were selected from the standard UCI repository [22] to facilitate a direct comparison with the key related works [2–5] and demonstrate our scheme’s feasibility and efficiency. For our experiments, we used seven datasets [23–29], which were partitioned into a 70% training set and a 30% validation set.

Table 2. Specifications of the benchmark datasets used in our experiments.

Datasets	Centroid Num. (k)	Dim. (d)	Data Point Num. (n)
iris [23]	3	4	150
wine [24]	3	13	178
breast cancer [25]	2	30	569
banknote [26]	2	5	1372
musk [27]	2	166	6598
sepsis [28]	2	3	110,204
gas [29]	6	128	13,910

To ensure a sufficient security level, we set the SHE parameters for sk_1 to $(k_0, k_1, k_2) = (1048, 30, 90)$, which offers a security guarantee comparable or superior to that of the related works; a detailed security analysis of these parameters is presented later. For a balance of efficiency and precision, we normalized all dataset features to the integer range $[0, 200]$. This range can be extended up to $[0, 2^{30} - 1]$ with negligible impact on accuracy or performance, as our chosen SHE parameters support the encryption of 30-bit (k_1 -bit) integers.

5.1. Prediction Accuracy and Comparison

Table 3 compares the accuracy of our scheme with the key related works [2–5], each of which utilizes different clustering datasets and evaluation metrics.

Table 3. Accuracy comparisons between our scheme and related works.

Related Works	Datasets	Evaluation Metrics	Accuracy Comparison
(i) [4]	Iris	Mean error	2.95~7.21% (>Our 0%)
(ii) [2]	Self-synthetic	Adjusted Rand Index	49~72.4% ($\approx$ Our 40~80%)
(iii) [3]	FCPS [30]	Misclassification rate	0~3% ($\geq$ Our 0%)
(iv) [5]	Self-synthetic	Relative error	0.65~26.91% (>Our 0%)

(i) The work in [4] uses the mean error between the centroids from ciphertext-based clustering and those from plaintext-based clustering as its accuracy metric. (ii) The authors of [2] employ the Adjusted Rand Index (ARI) [31], which measures the similarity between the clustering result and the ground-truth labels. (iii) Reference [3] utilizes the misclassification rate, which quantifies the fraction of incorrectly clustered data points, as its accuracy metric. (iv) The scheme in [5] is evaluated using the relative error between centroids computed on ciphertexts versus those computed on plaintexts.

Regarding the datasets, some of the compared works utilize self-generated synthetic data [2,5] or the non-UCI FCPS benchmark [3]. In contrast, our experiments use standard UCI datasets with parameters (k, n, d) chosen to be comparable to those in the related works.

As detailed in the notes of Table 3 and summarized in its final column, our scheme’s accuracy is benchmarked against these varied metrics. As the results show, our scheme outperforms the methods in [3–5] and achieves a clustering accuracy comparable to the work in [2].

The superior accuracy of our scheme can be attributed to our choice of cryptographic primitive. The compared works [3–5] rely on lattice-based schemes (e.g., CKKS, TFHE, BGV) that require approximations like polynomial functions to handle non-linear operations. In contrast, our approach utilizes an integer-based SHE scheme combined with data normalization. This method introduces only minimal precision loss by truncating the least significant bits, thereby preserving the accuracy. Furthermore, our interactive

protocol is designed such that all homomorphic computations consist of exact integer operations (additions and multiplications), eliminating the need for approximations and thus minimizing the accuracy degradation.

5.2. Communication Overhead of Our Scheme

Figures 2 and 3 illustrate the communication overhead of our scheme for various datasets, corresponding to the interactive data flows between the D0 and CSP depicted in Figure 1.

In the direction from the D0 to CSP, the communication overhead (illustrated in Figure 2) primarily consists of two components, corresponding to the two transmission steps shown in Figure 1.

- The first component is generated at the end of *Phase (1)*, when the D0 transmits the initial encrypted data package to the CSP, which includes the dataset, initial centroids, and public parameters.
- The second component occurs at the end of *Phase (2.2)* and consists of the encrypted new centroid mask vectors.

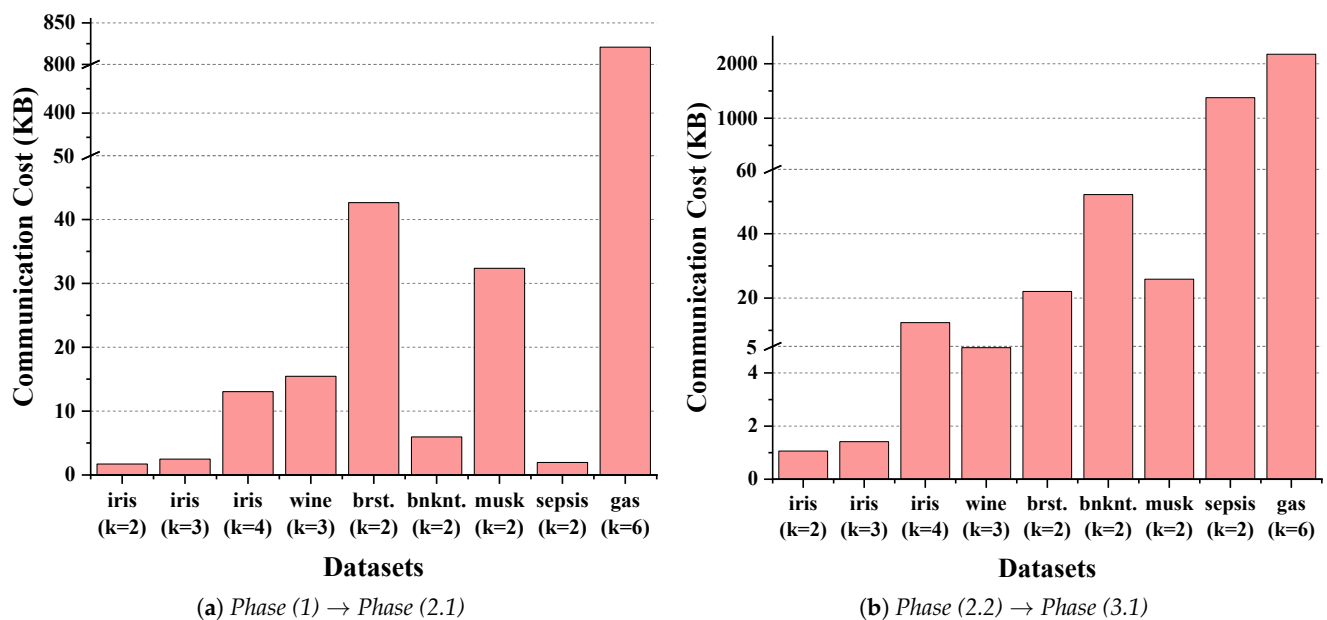


Figure 2. Communication overhead from D0 to CSP.

Conversely, in the direction from the CSP to D0, the communication overhead (Figure 3) is also composed of two main parts, reflecting the two return transmissions in the protocol.

- The first part is the encrypted distance matrix transmitted at the end of *Phase (2.1)*, which the D0 uses for comparison.
- The second part, sent at the end of *Phase (3.1)*, consists of the encrypted accumulated centroid features, which are required for the new centroid calculation.

In summary, all data exchanged between the parties are encrypted, and the resulting communication overhead is of a manageable size. Under typical network conditions, this communication overhead has a negligible impact on the total protocol execution time, a point we will elaborate on in the following subsection.

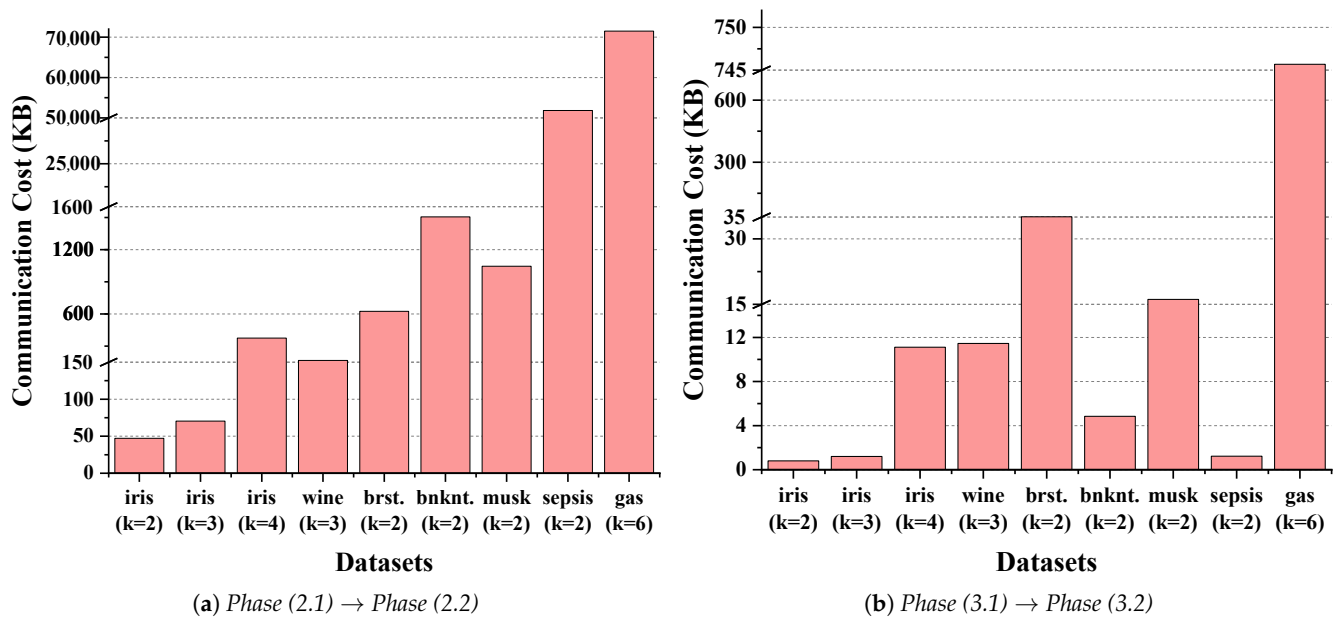


Figure 3. Communication overhead from CSP to D0.

5.3. Time Performance and Comparison

In this subsection, we evaluate the time performance of our protocol by analyzing three key aspects: the computation time for the D0 and CSP, the theoretical computational complexity, and a comparison of the total execution time against the related works. To ensure reliability, all the reported time overheads are the average of 10 independent runs. Our analysis of the computational complexity is presented alongside the empirical computation times for each protocol phase. As plaintext operations are negligible in comparison, this complexity analysis focuses primarily on operations involving ciphertexts.

The empirical computation times for our scheme are presented in Figure 4, which breaks down the performance of both the D0 and the CSP. The D0's computation time, detailed in Figure 4a, is divided into three parts corresponding to its three phases of activity in the protocol.

- *Phase (1).* In this phase, the computation time is the time overhead of protocol initialization, mainly including the key pair generation, dataset normalization, and encryption of the dataset and initial centroids. The initial setup requires the D0 to encrypt $n \cdot d$ data point features and $k \cdot d$ centroid features. As typically $n \gg k$, the complexity is dominated by dataset encryption, resulting in the initial computational complexity $O(n \cdot d)$.
- *Phase (2.2).* This phase involves two main operations: comparing distances to find the nearest centroid for each data point and generating the new centroid masks. During this process, the D0 mainly performs $O(n \cdot k)$ plaintext comparisons to find the nearest centroid for each point and encrypts and returns $n \times k$ mask values. The complexity of this phase is dominated by the $O(n \cdot k)$ cryptographic operations.
- *Phase (3.2).* In this phase, the D0 mainly needs to perform division operation of the new (accumulated) centroid feature by the number of data points that are closest to the corresponding centroid and check for convergence. Therefore, $k \times d$ decryption of encrypted accumulated features and $k \times d$ encryption of the new centroid features are needed, and the complexity of this phase is thus $O(k \cdot d)$.

As shown in Figure 4a, most of the computation time overheads of the D0 operations in different phases are less than two dozen seconds, and the total D0 computation time over-

heads are all low (i.e., at most dozens of seconds) based on our different clustering datasets (even for the huge dataset). Based on the theoretical analysis, the total computational complexity of the D0 is $O(n \cdot d)$ when $n \gg k$ is common.

The CSP's computation time, presented in Figure 4b, comprises two parts corresponding to its two active phases in the protocol.

- *Phase (2.1).* CSP performs ciphertext addition and multiplication to calculate the distances between each data point and each centroid. To compute the Modified Euclidean Distance (MED) between one data point and one centroid, the CSP performs operations for each of the d dimensions. This involves d homomorphic multiplications and $2d - 1$ homomorphic additions. Since this must be performed for all n data points and k centroids, the total complexity for this phase is $O(n \cdot k \cdot d)$ homomorphic operations.
- *Phase (3.1).* In this phase, the CSP performs homomorphic multiplications and additions to compute the new accumulated centroid features. The CSP calculates the accumulated features for each new centroid based on the equation in *Phase (3.1)*. For each of the $k \times d$ features, this requires a summation over all n data points, which involves n homomorphic multiplications and $n - 1$ homomorphic additions. Therefore, the total complexity of this phase is also $O(n \cdot k \cdot d)$ homomorphic operations.

Similar to the computation execution time of the D0, the total computation time overheads of the CSP are also low (i.e., less than dozens of seconds for the largest dataset) based on different clustering datasets. The overall computational complexity for the CSP is $O(n \cdot k \cdot d)$, which constitutes the dominant portion of the protocol's entire computation time.

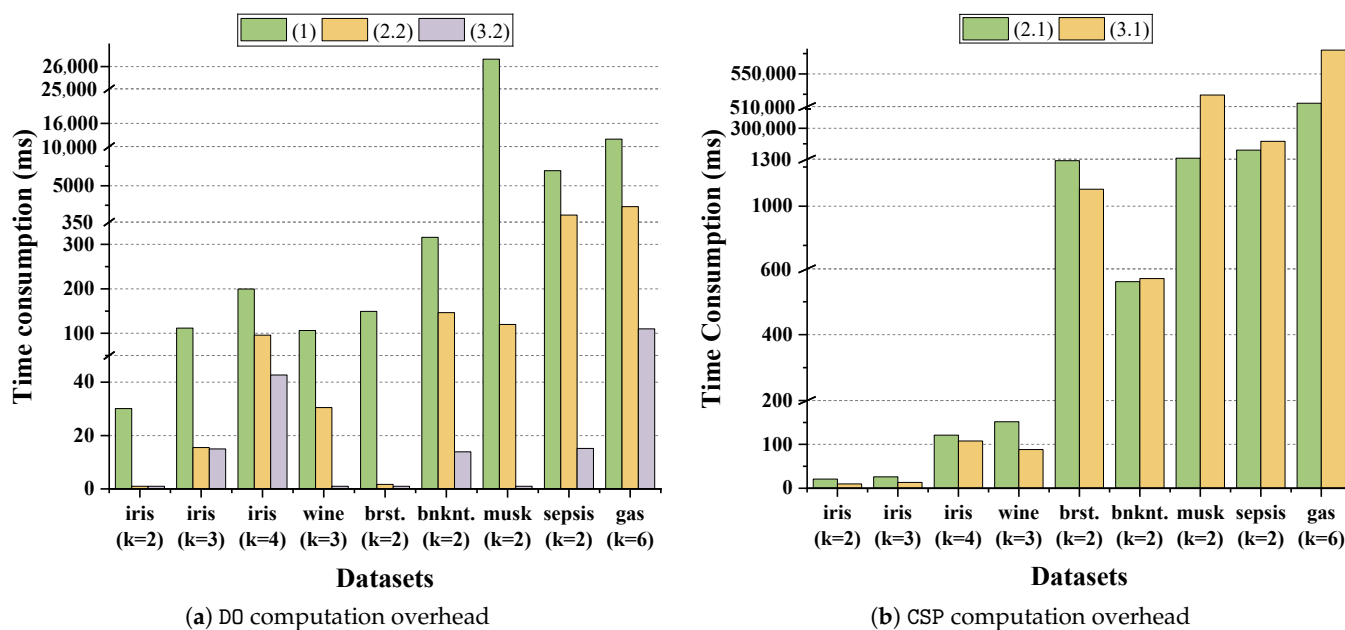


Figure 4. Computation overhead of D0 and CSP.

While the computation times for both parties are low, the interactive nature of our protocol means that the communication overhead could also impact the total execution time. Therefore, to comprehensively evaluate the impact of different network environments on the communication overheads, we simulated a standard Internet environment (100 Mbps bandwidth, 10 ms round-trip delay) to test the total execution time overheads of our protocol (including the computation time overheads and communication time overheads). The experimental results, as shown in Figure 5, demonstrate that our scheme maintains lower total execution time overheads than the related work [4] based on various UCI datasets.

While a poor network environment (e.g., 10 Mbps bandwidth, 30 ms delay) could become a bottleneck with larger datasets, this effect is mitigated by modern high-speed internet. However, Gigabit-bandwidth Internet, which is quite common nowadays, can significantly reduce the transmission time and further reduce the total execution time overhead of our protocol.

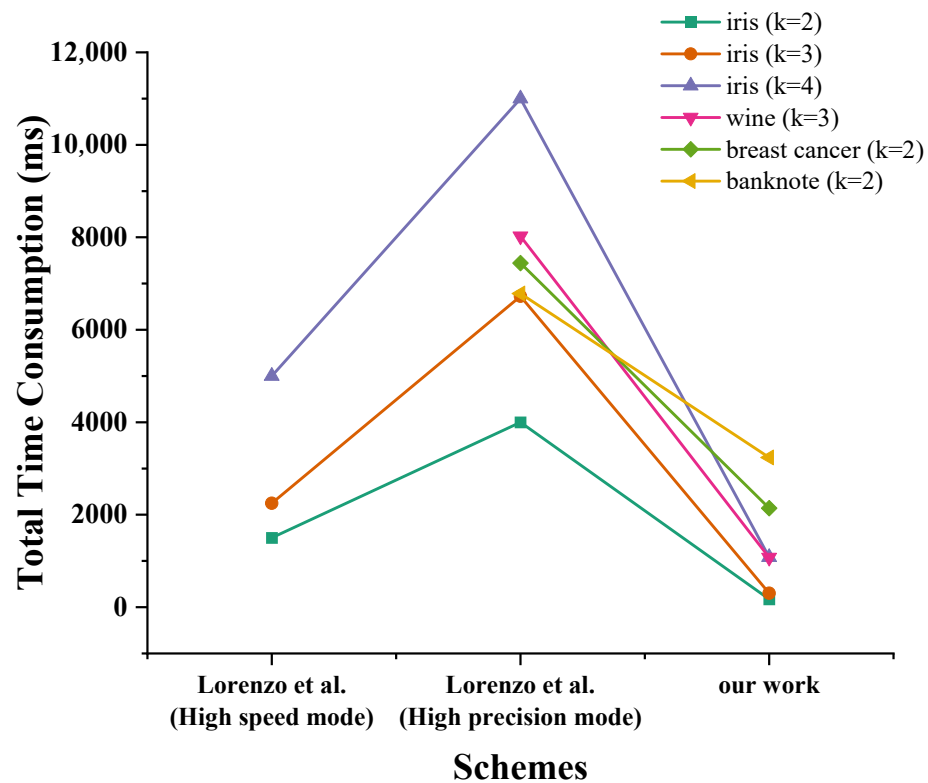


Figure 5. Total time overhead comparison with UCI-based related work (i.e., Lorenzo et al. [4]).

However, since three other related works [2,3,5] used as comparison deployed non-UCI datasets, we selected corresponding UCI datasets having larger dataset parameters, and the total time overhead comparison results are shown in Table 4. As one may see, our work has an overwhelming advantage in the total time overhead over the compared non-UCI-based works, while our work has equal or better accuracy as introduced before.

There are some details that need to be explained in Table 4. When compared to [3], our scheme has converged and stops when the protocol iteration number reaches 5; so, the total time overhead is the multiplication result of our one-iteration runtime and 5. When compared to [5], we chose datasets with larger dimensions (and same for the other parameters) and we still obtained a smaller total time overhead.

5.4. Security Analysis

As proven in [19], the SHE scheme employed in our work satisfies IND-CPA (Indistinguishability under Chosen-Plaintext Attack) security, based on the $(\mathcal{R}, p)$ -decision assumption. Specifically, the ability of the SHE scheme to resist chosen-plaintext attacks and thus achieve IND-CPA security is conditional upon the lengths of the security parameters $\mathcal{R}$ (length k_2) and p (length k_0). On one hand, the security of $\mathcal{R}$, a random number of length k_2 , relies on the infeasibility of a brute-force attack. To render such an attack computationally infeasible (requiring 2^{k_2} operations), a length of $k_2 \geq 90$ is required. On the other hand, the security of the large prime p is tied to the difficulty of the integer factorization problem. Given the public modulus $N = pq$, an attacker must factor N to

recover p . With the currently recommended minimum secure length for N at 1024 bits, the length of p, k_0 , must be at least 512 bits.

Table 4. Total time overhead comparison with non-UCI-based related works.

Comparison Results Between [2] Based on Paillier and Our Work				
Dataset Size		Research [2]		Our work
2000		99,402.12 s $\approx$ 27.61 h		1.14 s
3000		255,790.18 s $\approx$ 71.05 h		1.29 s
4000		377,794.82 s $\approx$ 104.94 h		1.86 s
5000		680,925.54 s $\approx$ 189.15 h		4.76 s
Comparison results between [3] (exact/approximate mode) based on TFHE and our work				
		[3] (Exact mode)	[3] (Approximate mode)	Our work
Runtime per Iteration		873.46 h	15.56 h	0.166 s
Iteration Number		15	40	5
Total Time		545.91 days $\approx$ 17.95 months	25.93 days $\approx$ 0.85 months	0.83 s
Comparison results between [5] (fastest/slowest mode) based on BGV and our work				
d, n	[5] (Fastest Mode)	[5] (Slowest Mode)	Our work	
$d = 2, n = 100$	1479 s	3164 s	$d = 4, n = 100$	0.16 s
$d = 10, n = 100$	150 s	702 s	$d = 13, n = 100$	0.97 s

The security parameters selected for our implementation, $k_2 = 90$ and $k_0 = 1048$, meet or exceed the required minimums of 90 and 512 bits, respectively. Therefore, our configuration of the SHE scheme satisfies the requirements for IND-CPA security.

Given this IND-CPA secure foundation, we now analyze how our protocol achieves the specific privacy properties defined in Section 3.2 under the honest-but-curious model.

- *Dataset Privacy.* The clustering dataset X is encrypted using the secret key sk_1 . Without access to sk_1 , the CSP has no means to decrypt the data features. This directly guarantees the privacy of the dataset.
- *Model Privacy.* Throughout the protocol, all sensitive model parameters are encrypted with sk_1 . This includes initial/updated centroid features, intermediate MED values, centroid masks, and accumulated features. The CSP operates exclusively on these ciphertexts. Given the IND-CPA security of SHE, the CSP cannot infer any information about these parameters, thus ensuring the privacy of the k-means model.

6. Conclusions

In this paper, we proposed a fast and privacy-preserving outsourced k-means clustering scheme based on symmetric homomorphic encryption (SHE). Our scheme enables a data owner (DO) to securely outsource encrypted data to a cloud service provider (CSP) for k-means model training, while guaranteeing the privacy of both the dataset and the model parameters. We developed a Modified Euclidean Distance (MED) metric to avoid costly non-linear operations, thereby reducing both the computational and communication overheads. The experimental results demonstrate that our scheme achieves superior performance compared to the related works in terms of the prediction accuracy, computation time, and total execution overhead. Building on the success of the SHE for efficient privacy-preserving k-means, our future work will focus on integrating this approach with outsourced incremental clustering and classification schemes to support a broader range of application scenarios.

Author Contributions: Conceptualization, S.X.; Methodology, S.X.; Software, W.T.; Validation, W.T.; Data curation, W.T.; Writing—original draft, W.T.; Writing—review & editing, S.X.; Visualization, W.T.; Supervision, S.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant Nos. 61902285, 62004077), the Open Foundation of Hubei Key Laboratory of Applied Mathematics (Hubei University) (Grant No. HBAM202101), the Major Program (JD) of Hubei Province (2023BAA027), and the Fundamental Research Funds for the Central Universities (Program No. 2662022XXYJ004). We thank Bo Li and the experimental teaching center of College of Informatics, Huazhong Agriculture University for providing the experimental environment and computing resources.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cheon, J.H.; Kim, D.; Park, J.H. Towards a Practical Cluster Analysis over Encrypted Data. In Proceedings of the Selected Areas in Cryptography—SAC 2019, Waterloo, ON, Canada, 12–16 August 2019; Paterson, K.G., Stebila, D., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 227–249.
2. Catak, F.O.; Aydin, I.; Elezaj, O.; Yildirim-Yayilgan, S. Practical Implementation of Privacy Preserving Clustering Methods Using a Partially Homomorphic Encryption Algorithm. *Electronics* **2020**, *9*, 229. [\[CrossRef\]](#)
3. Jaschke, A.; Armknecht, F. Unsupervised Machine Learning on Encrypted Data. In Proceedings of the Selected Areas in Cryptography—SAC 2018, Calgary, AB, Canada, 15–17 August 2018; Cid, C., Jacobson, M.J., Jr., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 453–478.
4. Lorenzo, R. Fast but approximate homomorphic k-means based on masking technique. *Int. J. Inf. Secur.* **2023**, *22*, 1605–1619. [\[CrossRef\]](#)
5. Sakellariou, G.; Gounaris, A. Homomorphically encrypted k-means on cloud-hosted servers with low client-side load. *Computing* **2019**, *101*, 24. [\[CrossRef\]](#)
6. Mohassel, P.; Rosulek, M.; Trieu, N. Practical Privacy-Preserving K-means Clustering. *Proc. Priv. Enhancing Technol.* **2020**, *2020*, 414–433. [\[CrossRef\]](#)
7. Bunn, P.; Ostrovsky, R. Secure two-party k-means clustering. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; CCS '07, pp. 486–497. [\[CrossRef\]](#)
8. Liu, X.; Jiang, Z.L.; Yiu, S.M.; Wang, X.; Tan, C.; Li, Y.; Liu, Z.; Jin, Y.; Fang, J. Outsourcing Two-Party Privacy Preserving K-Means Clustering Protocol in Wireless Sensor Networks. In Proceedings of the 2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN), Shenzhen, China, 16–18 December 2015; pp. 124–133. [\[CrossRef\]](#)
9. Zhang, E.; Li, H.; Huang, Y.; Hong, S.; Zhao, L.; Ji, C. Practical multi-party private collaborative k-means clustering. *Neurocomputing* **2022**, *467*, 256–265. [\[CrossRef\]](#)
10. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast fully homomorphic encryption over the torus. *J. Cryptol.* **2020**, *33*, 34–91. [\[CrossRef\]](#)
11. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, Cambridge, MA, USA, 8–10 January 2012; ITCS '12, pp. 309–325. [\[CrossRef\]](#)
12. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic Encryption for Arithmetic of Approximate Numbers. In Proceedings of the Advances in Cryptology—ASIACRYPT 2017, Hong Kong, China, 3–7 December 2017; Takagi, T., Peyrin, T., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 409–437.
13. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; pp. 223–238.
14. Mahdikhani, H.; Lu, R.; Zheng, Y.; Shao, J.; Ghorbani, A.A. Achieving $O(\log^3 n)$ Communication-Efficient Privacy-Preserving Range Query in Fog-Based IoT. *IEEE Internet Things J.* **2020**, *7*, 5220–5232. [\[CrossRef\]](#)
15. Zhao, J.; Zhu, H.; Wang, F.; Lu, R.; Liu, Z.; Li, H. PVD-FL: A Privacy-Preserving and Verifiable Decentralized Federated Learning Framework. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 2059–2073. [\[CrossRef\]](#)
16. Zhao, J.; Zhu, H.; Wang, F.; Lu, R.; Wang, E.; Li, L.; Li, H. VFLR: An Efficient and Privacy-Preserving Vertical Federated Framework for Logistic Regression. *IEEE Trans. Cloud Comput.* **2023**, *11*, 3326–3340. [\[CrossRef\]](#)
17. Miao, Y.; Liu, Z.; Li, X.; Li, M.; Li, H.; Choo, K.K.R.; Deng, R.H. Robust Asynchronous Federated Learning With Time-Weighted and Stale Model Aggregation. *IEEE Trans. Dependable Secur. Comput.* **2024**, *21*, 2361–2375. [\[CrossRef\]](#)

18. Sun, L.; Zhang, Y.; Zheng, Y.; Song, W.; Lu, R. Towards Efficient and Privacy-Preserving High-Dimensional Range Query in Cloud. *IEEE Trans. Serv. Comput.* **2023**, *16*, 3766–3781. [[CrossRef](#)]
19. Zheng, Y.; Lu, R.; Guan, Y.; Shao, J.; Zhu, H. Efficient and Privacy-Preserving Similarity Range Query over Encrypted Time Series Data. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 2501–2516. [[CrossRef](#)]
20. Miao, Y.; Xu, C.; Zheng, Y.; Liu, X.; Meng, X.; Deng, R.H. Efficient and Secure Spatial Range Query over Large-scale Encrypted Data. In Proceedings of the 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), Hong Kong, China, 18–21 July 2023; pp. 1–11. [[CrossRef](#)]
21. Zheng, Y.; Lu, R.; Guan, Y.; Zhang, S.; Shao, J.; Zhu, H. Efficient and Privacy-Preserving Similarity Query with Access Control in eHealthcare. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 880–893. [[CrossRef](#)]
22. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: <https://ergodicity.net/2013/07/> (accessed on 4 August 2025).
23. Iris. 2025. Available online: <http://archive.ics.uci.edu/dataset/53/iris> (accessed on 4 August 2025).
24. Wine. 2025. Available online: <http://archive.ics.uci.edu/dataset/186/wine+quality> (accessed on 4 August 2025).
25. Breast Cancer. 2025. Available online: <http://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic> (accessed on 4 August 2025).
26. Banknote. 2025. Available online: <https://archive.ics.uci.edu/dataset/267/banknote+authentication> (accessed on 4 August 2025).
27. Musk. 2025. Available online: <https://archive.ics.uci.edu/dataset/75/musk-version-2> (accessed on 4 August 2025).
28. Sepsis. 2025. Available online: <https://archive.ics.uci.edu/dataset/827/sepsis+survival+minimal+clinical+records> (accessed on 4 August 2025).
29. Gas. 2025. Available online: <https://archive.ics.uci.edu/dataset/224/gas+sensor+array+drift+dataset> (accessed on 4 August 2025).
30. Ultsch, A. CLUSTERING WITH SOM: U* C. In Proceedings of the Workshop on Self-Organizing Maps, WSOM'05, Paris, France, 5–8 September 2005; pp. 75–82.
31. Halkidi, M.; Batistakis, Y.; Vazirgiannis, M. Clustering validity checking methods: Part II. *ACM Sigmod Rec.* **2002**, *31*, 19–27. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.