applied sciences

MDPI

Article

Secure K-Means Clustering Scheme for Confidential Data Based on Paillier Cryptosystem

Zhengqi Zhang [1], Zixin Xiong [1,2,*] and Jun Ye [1,2]

[1] Key Laboratory of Data Science and Intelligence Education, Hainan Normal University, Ministry of Education, Haikou 571158, China; zq_zhang80@126.com (Z.Z.); yejun@hainanu.edu.cn (J.Y.)
[2] School of Cyberspace Security, Hainan University, Haikou 570228, China
* Correspondence: xiongzixin@hainanu.edu.cn

Abstract: In this paper, we propose a secure homomorphic K-means clustering protocol based on the Paillier cryptosystem to address the urgent need for privacy-preserving clustering techniques in sensitive domains such as healthcare and finance. The protocol uses the additive homomorphism property of the Paillier cryptosystem to perform K-means clustering on the encrypted data, which ensures the confidentiality of the data during the whole calculation process. The protocol consists of three main components: secure computation distance (SCD) protocol, secure cluster assignment (SCA) protocol and secure cluster center update (SUCC) protocol. The SCD protocol securely computes the squared Euclidean distance between the encrypted data point and the encrypted cluster center. The SCA protocol securely assigns data points to clusters based on these cryptographic distances. Finally, the SUCC protocol securely updates the cluster centers without leaking the actual data points as well as the number of intermediate sums. Through security analysis and experimental verification, the effectiveness and practicability of the protocol are proved. This work provides a practical solution for secure clustering based on homomorphic encryption and contributes to the research in the field of privacy-preserving data mining. Although this protocol solves the key problems of secure distance computation, cluster assignment and centroid update, there are still areas for further research. These include optimizing the computational efficiency of the protocol, exploring other homomorphic encryption schemes that may provide better performance, and extending the protocol to handle more complex clustering algorithms.

Keywords: K-means; privacy preserving; multi-key; fully homomorphic encryption; outsourced computing

Academic Editor: Luis Javier García Villalba

Received: 22 May 2025
Revised: 15 June 2025
Accepted: 17 June 2025
Published: 19 June 2025

Citation: Zhang, Z.; Xiong, Z.; Ye, J. Secure K-Means Clustering Scheme for Confidential Data Based on Paillier Cryptosystem. *Appl. Sci.* **2025**, *15*, 6918. https://doi.org/10.3390/app15126918

1. Introduction

Clustering is a fundamental technique in data mining and machine learning, with K-means clustering being one of the most widely used algorithms due to its simplicity and efficiency. However, the growing importance of data privacy, especially in sensitive areas such as healthcare and finance, demands the development of privacy-preserving clustering techniques. Traditional K-means clustering algorithms [1] require access to the entire dataset, which may contain sensitive information. Hence, there is an urgent need for secure clustering protocols that protect the privacy of the data.

Homomorphic encryption (HE) [2] offers a promising solution to this challenge by enabling computations on encrypted data without the need to decrypt it, thus preserving data privacy. Among the various HE schemes, the Paillier cryptosystem is particularly

notable for its additive homomorphic properties, which are useful for performing arithmetic operations on encrypted data.

In this study, we propose a secure homomorphic K-means clustering protocol using the Paillier cryptosystem [3]. Our protocol allows the K-means clustering algorithm to be executed on encrypted data, ensuring that sensitive information remains confidential throughout the computation. This is achieved by exploiting the additive homomorphism property of the Paillier cryptosystem and a privacy-preserving [4] framework for the outsourced computation of rational numbers, allowing operations such as sums and averages to be computed securely in the K-means algorithm.

The proposed protocol consists of three main components: the secure computation distance (SCD) protocol, the secure cluster assignment (SCA) protocol, and the secure update cluster center (SUCC) protocol. Each of these components addresses specific challenges in ensuring the privacy and security of the clustering process. The secure computation distance (SCD) protocol ensures that the squared Euclidean distance between data points and cluster centers is computed securely without decrypting the data, thereby preserving the privacy of both data points and cluster centers. The secure cluster assignment (SCA) protocol securely assigns data points to clusters using the encrypted squared distances between data points and cluster centers, ensuring that cluster assignments are made without revealing the actual distances or cluster memberships. Finally, the secure update cluster center (SUCC) protocol securely updates the cluster centers without revealing the actual data points or intermediate sums and counts, maintaining the confidentiality of sensitive information throughout the update process.

Previous research has explored various approaches to privacy-preserving clustering, such as differential privacy [5] and secure multi-party computation (SMPC) [6–9]. However, these methods often involve trade-offs between accuracy, complexity, and computational efficiency. Homomorphic encryption, particularly the Paillier cryptosystem, provides a balance between security and computational efficiency, making it a promising approach for privacy-preserving data mining [10–12] tasks. Our protocol addresses several key challenges in applying homomorphic encryption to clustering algorithms, such as efficiently computing distances, securely updating centroids, and ensuring the overall scalability and security of the process. We design our protocol to be both practical and robust, demonstrating its effectiveness through theoretical analysis and experimental validation.

The primary contributions of this work are as follows: 1. We design a secure K-means clustering protocol using the Paillier cryptosystem, ensuring data privacy throughout the clustering process. 2. We address key computational challenges in applying homomorphic encryption to the K-means algorithm, including secure distance calculation, cluster assignment, and centroid update. 3. We provide a comprehensive analysis of the protocol's security and efficiency, demonstrating its practicality for real-world applications.

In the following sections, we review related work in privacy-preserving clustering and homomorphic encryption, describe the Paillier cryptosystem and its application to secure K-means clustering, and present our proposed protocol in detail. We also discuss the security and performance of the protocol, validate it through experimental results, and conclude with potential directions for future research.

2. Related Work

The intersection of privacy-preserving techniques and clustering algorithms has been an active area of research, driven by growing concerns over data privacy in various domains such as healthcare, finance, and social networks. Existing work in privacy-preserving clustering has explored a variety of methodologies, including differential privacy, secure multi-party computation (SMPC), homomorphic encryption, and other related approaches.

Differential privacy has been widely adopted to provide statistical guarantees of privacy in data analysis. Several researchers have applied differential privacy to clustering algorithms. For instance, Blum [13] introduced the first differentially private algorithm for K-means clustering, ensuring that the output of the clustering process does not reveal sensitive information about any individual data point. However, the added noise necessary for differential privacy can degrade the accuracy of clustering results, making it less suitable for applications requiring high precision.

SMPC techniques allow multiple parties to jointly compute a function over their inputs while keeping those inputs private. Vaidya and Clifton [14] proposed a privacy-preserving K-means clustering algorithm using SMPC, ensuring that no party learns anything beyond the final clustering results. Although SMPC provides strong security guarantees, it often involves high computational and communication overhead, posing challenges for scaling to large datasets.

Homomorphic encryption, particularly additive homomorphic encryption schemes such as the Paillier cryptosystem, enables computations on encrypted data without decryption, making it particularly useful for privacy-preserving data mining tasks. Notable work by Aono [15] proposed a privacy-preserving logistic regression algorithm using the Paillier cryptosystem, allowing the training process to be carried out on encrypted data and preserving the privacy of sensitive information. However, their approach focuses on supervised learning, and its application to unsupervised learning tasks like clustering remains underexplored. There have been limited studies on applying homomorphic encryption to clustering algorithms. Froelicher (2023) [16] proposed a privacy-preserving K-means clustering algorithm using homomorphic encryption, allowing the computation of distances and centroid updates on encrypted data. However, their method does not fully address the efficiency and scalability challenges associated with homomorphic encryption.

Another approach to privacy-preserving clustering is through data anonymization techniques. Methods like k-anonymity and l-diversity aim to anonymize the data before clustering. Sweeney (2002) [17] proposed k-anonymity, ensuring that each record is indistinguishable from at least $k - 1$ other records. However, these methods often suffer from information loss and do not provide strong privacy guarantees against certain types of attacks.

Federated learning [18,19] has emerged as a technique to train machine learning models on decentralized data without moving the data itself. Bonawitz [20] introduced a secure aggregation protocol for federated learning, which can be extended to clustering tasks. While federated learning offers a way to leverage distributed data, it still requires secure aggregation mechanisms to ensure privacy.

The existing body of work in privacy-preserving clustering has made significant strides, but challenges remain in balancing privacy, accuracy, and efficiency. Differential privacy and SMPC offer strong privacy guarantees but often at the cost of accuracy and scalability. Homomorphic encryption, particularly the Paillier cryptosystem, provides a promising approach by enabling secure computations on encrypted data. However, its application to clustering algorithms, particularly in addressing efficiency and scalability, needs further exploration.

Our work builds on these foundations by proposing a secure homomorphic K-means clustering protocol using the Paillier cryptosystem. By addressing key challenges in secure distance computation, cluster assignment, and centroid update, we aim to provide a practical and efficient solution for privacy-preserving clustering, suitable for real-world applications. In the following sections, we detail our proposed protocol and demonstrate its effectiveness through theoretical analysis and experimental validation.

3. Preliminaries

3.1. Notations

We summarize the notations used in this paper in Table 1.

Table 1. Notations.

Parameter	Description
$E(x_i) = \{E(x_{ik})\}_{k=1}^{d}$	Encrypted values of each dimension of data point x_i
$E(c_j) = \{E(c_{jk})\}_{k=1}^{d}$	Encrypted values of each dimension of cluster center c_j
$E(d_{ij}^2)$	Encrypted squared Euclidean distance between data point x_i and cluster center c_j
$E(\Delta)$	Encrypted difference for each dimension $E(x_{ik} - c_{jk})$
$E(\Delta^2)$	Encrypted squared difference for each dimension $E((x_{ik} - c_{jk})^2)$
$\{E(d_{ij}^2)\}_{j=1}^{K}$	Encrypted squared distances from data point x_i to each cluster center c_j
$E(\lambda_i)$	Encrypted index of the cluster center to which data point x_i is assigned
$E(C_j)$	Encrypted cluster set for cluster center c_j
$E(C_{\lambda_i})$	Encrypted cluster to which data point x_i is assigned
U	Set of encrypted distances used for finding the minimum
s	Number of remaining pairs of encrypted distances after one round of comparison
$\{E(x_i)\}_{i=1}^{N}$	Set of encrypted data points
$\{E(c_j)\}_{j=1}^{K}$	Set of encrypted cluster centers
$\{E(c_j^*)\}_{j=1}^{K}$	Set of updated encrypted cluster centers
$E(c_j^*)$	Encrypted accumulator for cluster center c_j
$\|C_j\|$	Number of data points in cluster C_j
$E(\|C_j\|)$	Encrypted number of data points in cluster C_j
$E(b_{ij})$	Encrypted indicator whether data point x_i belongs to cluster C_j

3.2. Paillier Cryptosystem

The Paillier cryptosystem is a probabilistic asymmetric encryption scheme with homomorphic properties. It consists of key generation, encryption, decryption, and homomorphic addition operations.

Key Generation: To generate a Paillier key pair, select two large prime numbers p and q. Compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$, where lcm denotes the least common multiple. Choose a random integer g such that $g^\lambda \equiv 1 \pmod{n^2}$ and $\gcd(g, n^2) = 1$. The public key is (n, g), and the private key is λ.

Encryption: To encrypt a plaintext message m, choose a random integer r such that $0 < r < n$. Compute the ciphertext as $c = g^m \cdot r^n \mod n^2$.

Decryption: To decrypt a ciphertext c, compute the plaintext message as $m = L(c^\lambda \mod n^2) \cdot \text{modinv}(L(g^\lambda \mod n^2), n)$, where $L(x) = \frac{x-1}{n}$ and modinv is the modular inverse.

Homomorphic Addition: Given ciphertexts c_1 and c_2 encrypted with the same public key, the homomorphic addition of c_1 and c_2 results in the encryption of the sum of their plaintexts: $E(m_1) \cdot E(m_2) \mod n^2$.

3.3. K-Means Clustering

The K-means clustering algorithm such as Algorithm 1 is described as follows.

Initialization: Randomly select K data points as initial cluster centers.

Iterative Optimization: Iteratively update data point assignments and cluster centers until convergence or reaching the maximum number of iterations.

Cluster Identification: Assign data points to the nearest cluster center.

Selecting the Optimal K: Use methods such as the elbow method or silhouette score to select the optimal number of clusters.

Algorithm 1 K-means Clustering

Input: Dataset $X = \{x_1, x_2, ..., x_n\}$ and number of clusters K
Output: Cluster assignments c_i and cluster centers μ_j
Randomly initialize K cluster centers $\mu_1, \mu_2, ..., \mu_K$
repeat
  for each data point x_i **do**
    Assign x_i to the nearest cluster center: $c_i \leftarrow \text{argmin}_j ||x_i - \mu_j||^2$
  end for
  for each cluster center μ_j **do**
    Update μ_j to be the mean of the points assigned to cluster j
  end for
until convergence
Return Cluster assignments c_i and cluster centers μ_j

3.4. Basic Cryptographic Primitives

1. Revised Secure Multiplication (RSM) Protocol
   The RSM protocol enables two parties (the client and the cloud service provider) to perform secure multiplication computations while preserving the privacy of the original data. By introducing random numbers and partial decryption techniques in the multiplication operation, the RSM protocol achieves the computation of the product without revealing the actual values of the multiplicands. The core idea of this protocol lies in using encryption and decryption operations, allowing the cloud service provider to obtain the encrypted product without knowing the actual values of the multiplicands.
2. Secure Maximum and Minimum Sorting (SMMS) Protocol
   The SMMS protocol allows multiple parties to sort their data and determine the maximum and minimum values while ensuring the privacy of the original data. In the SMMS protocol, participants exchange encrypted sorting results and utilize partial decryption techniques to determine the maximum and minimum values, thereby achieving privacy-preserving sorting operations.
3. Secure Division (SDIV) Protocol
   The SDIV protocol enables two parties to perform secure division computations, ensuring the privacy of the data during the computation process without revealing the actual values of the divisor and dividend. By introducing random numbers and encryption techniques, the SDIV protocol allows the cloud service provider to compute the ciphertext form of the quotient without obtaining the actual values of the original data.
4. Secure equality Test (SEQ) protocol
   The SEQ protocol can output ciphertext f and judge whether two ciphertexts are equal.

These protocols provide the foundational privacy protection mechanisms for the privacy-preserving outsourced computation of rational numbers (POCR) framework, allowing clients to securely outsource rational number data to cloud service providers for storage and computation while protecting data privacy.

4. Security Model

In this section, we formalize the security model for our privacy-preserving K-means clustering protocol under the semi-honest (honest-but-curious) adversary setting. The protocol involves three parties: a client, a cloud server (CS), and a computation service provider (CSP). The client owns private input data and receives the final clustering results, while CS and CSP collaboratively perform computations over encrypted data.

4.1. Adversarial Model

We consider semi-honest adversaries who follow the protocol specification faithfully but try to learn additional information from the messages they receive. We assume the following:

- At most, one of CS or CSP can be corrupted. They do not collude.
- The client is always honest.
- The cryptographic primitives used (e.g., Paillier encryption, secure subprotocols) are semantically secure.

4.2. Ideal Functionality

We define the functionality $\mathcal{F}_{\text{KMeans}}$ that the protocol aims to securely realize. In the ideal world, a trusted party receives the client's input and performs the computation.

Definition 1 (Ideal Functionality $\mathcal{F}_{\text{KMeans}}$)**.** *Let $\{x_i\}_{i=1}^{N} \subset \mathbb{Z}_p^d$ be the client's dataset and K be the number of clusters. The ideal functionality proceeds as follows:*

1. *Receive input $\{x_i\}_{i=1}^{N}$, K from the client.*
2. *Run the standard K-means algorithm on plaintext inputs to compute cluster centers $\{\mu_j\}_{j=1}^{K}$ and assignments $\mathcal{C} : \{1, \ldots, N\} \to \{1, \ldots, K\}$.*
3. *Return $(\{\mu_j\}, \mathcal{C})$ to the client.*
4. *Reveal nothing to CS or CSP except what is defined in the leakage functions.*

4.3. Leakage Functions

Since perfect privacy is impossible in practical settings (e.g., some access pattern or control-flow information may leak), we define leakage profiles for each party:

Client: Receives only the final output $(\{\mu_j\}, \mathcal{C})$.

Cloud Server (CS):

- Number of input records N and dimension d.
- Number of clusters K and number of iterations T.
- Encrypted dataset $\{E(x_i)\}$ and all ciphertexts exchanged during computation (e.g., encrypted distances, encrypted cluster centers, encrypted assignments).
- Protocol structure (e.g., number and types of homomorphic operations).

Computation Service Provider (CSP):

- Number of clusters K, dimension d, and iteration count T.
- Ciphertexts for comparisons (e.g., distance minimum selection) and ciphertexts related to assignment operations.
- Receives only encrypted or masked intermediate results, not raw values or secret key material.

We denote the leakage to CS as L_{CS} and to CSP as L_{CSP}.

4.4. Security Definition

Definition 2 (Simulation-Based Security)**.** *Let Π_{KMeans} be the proposed protocol and $\mathcal{F}_{KMeans}$ the ideal functionality. Π_{KMeans} is secure in the semi-honest model if for any PPT adversary $\mathcal{A}$ corrupting CS or CSP, there exists a PPT simulator $\mathcal{S}$ such that the real-world and ideal-world executions are computationally indistinguishable:*

$$\mathit{REAL}_{\Pi_{KMeans},\mathcal{A}}(x) \approx_c \mathit{IDEAL}_{\mathcal{F}_{KMeans},\mathcal{S}}(x, \mathsf{L})$$

where $\mathsf{L} \in \{\mathsf{L}_{CS}, \mathsf{L}_{CSP}\}$ is the leakage profile depending on the corrupted party.

5. Secure Homomorphic K-Means Clustering Scheme

5.1. Framework

This scheme proposes a method for secure K-means clustering and three secure protocols. It consists of preprocessing the data, encrypting each data point, initializing the cluster center with the encrypted value, iteratively updating the cluster and its center while maintaining the encryption, and finally computing the clustering result. This approach allows cluster analysis while protecting sensitive information.The framework of the scheme is shown in Figure 1, and the details of the scheme are as follows.

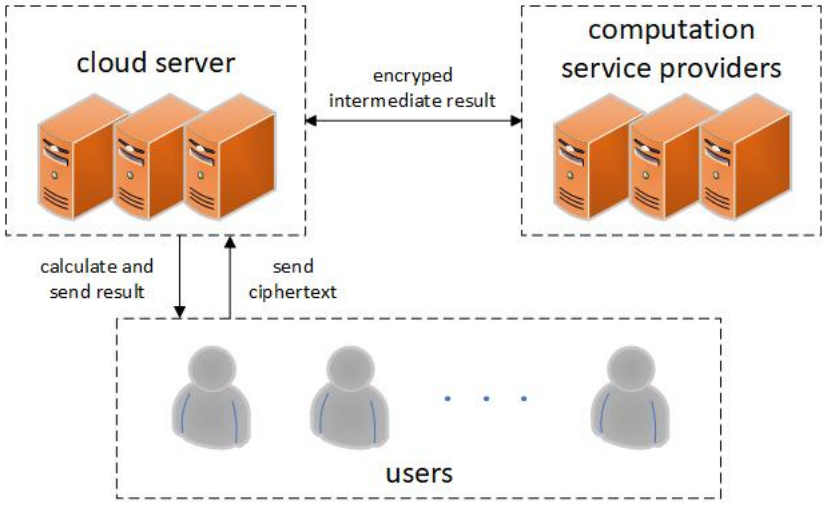

Figure 1. The framework of the scheme.

1. users
   Users first encrypt their personal data with the public key, and then upload the ciphertext data to CS for storage. Users can also ask CS to compute the outsourced data in secret state.
2. cloud server
   CS has "unlimited" data storage space and is responsible for storing and managing outsourced data from all registered parties, namely users. In addition, CS can also store all intermediate and final ciphertext results and can perform specific computations on encrypted data.
3. computation service providers
   CSP provides online computing services for users. In addition, CSP can partially solve the ciphertext sent by CS to perform a specific computation and then re-encrypt the result.

5.2. Scheme Details

1. Data Preprocessing
   Data Encryption: Encrypt each data point x_i from the original dataset using the Paillier encryption algorithm to obtain encrypted data $E(x_i)$.
2. Secure K-means Clustering
   Initialization: Randomly generate K initial cluster centers on the cloud server. Encrypt the cluster centers using the Paillier encryption algorithm to obtain encrypted cluster centers $E(c_j)$.
   Iterative Update: Repeat the following steps until a stopping condition is met (e.g., reaching the maximum number of iterations or cluster centers no longer change):
   (a) Compute Distances: Perform the secure computation distance (SCD) protocol on the user side to compute the encrypted distances $E(d_{ij}^2)$ between each data point and the encrypted cluster centers.

(b) Secure Cluster Assignment: Execute the secure cluster assignment protocol on the user side to assign each data point to the nearest cluster based on the encrypted distances obtained.
(c) Update Cluster Centers: Update the encrypted cluster centers on the cloud server using the secure update cluster center Protocol.
(d) Convergence Check: Check for the convergence of cluster centers on either the user side or the cloud server. If the stopping condition is met, end the iterations.

3. Result Decryption
Decrypting Cluster Results: Decrypt the encrypted cluster centers $E(c_j)$ to obtain the original cluster centers c_j, and decrypt the encrypted cluster indices $E(\lambda_i)$ to obtain the original cluster indices λ_i using the Paillier decryption algorithm.
4. Result Analysis
Obtaining Cluster Results: Based on the decrypted cluster centers and cluster indices, derive the final cluster results on the user side for further analysis.

6. Secure Homomorphic K-Means Clustering Protocol

6.1. Secure Computation Distance (SCD) Protocol

Algorithm 2 guarantees that the squared Euclidean distance between the data point and the cluster center can be securely computed without decrypting the data, thus protecting the privacy of the data points and the cluster center.

The input consists of encrypted data point $E(x_i)$ and encrypted cluster center $E(c_j)$, each represented as vectors of encrypted values over d dimensions. For each dimension k, the difference between the encrypted data point and the encrypted cluster center is computed homomorphically: $E(\Delta) = E(x_{ik}) \cdot E(c_{jk})^{N-1}$. This uses Paillier's homomorphic properties where subtraction is achieved by multiplying the ciphertext of x_{ik} with the modular inverse of the ciphertext of c_{jk}. The square of the encrypted difference is computed by homomorphically squaring the difference: $E(\Delta^2) = RSM(E(\Delta), E(\Delta))$. The encrypted squared Euclidean distance is obtained by multiplying all the encrypted squared differences: $E(d_{ij}^2) = \prod_{k=1}^{d} E(\Delta^2)$. Homomorphic addition in the Paillier cryptosystem is performed by multiplying the encrypted values.

Algorithm 2 Secure Computation Distance (SCD) Protocol using Paillier Cryptosystem

Input: Encrypted data $E(x_i) = E(x_{ik})k = 1^d$, Encrypted cluster center $E(c_j) = E(c_{jk})k = 1^d$.
Output: Encrypted distance $E(d_{ij}^2)$.
Performed by CS:
for each dimension k from 1 to d **do**
  Compute the encrypted difference :
    $E(\Delta) = E(x_{ik}) \cdot E(c_{jk})^{N-1}$.
end for
Performed by and CSP:
for each dimension k from 1 to d **do**
  Compute the encrypted squared difference :
    $E(\Delta^2) = RSM(E(\Delta), E(\Delta))$.
end for
Compute the encrypted squared Euclidean distance :
  $E(d_{ij}^2) = \prod_{k=1}^{d} E(\Delta^2)$.
return Encrypted distance $E(d_{ij}^2)$.

6.2. Secure Cluster Assignment (SCA) Protocol

Algorithm 3 describes how to securely assign data points to clusters in the K-means algorithm using the Paillier cryptosystem. The input to the algorithm is the encrypted squared distances between data points and cluster centers, and the output is the encrypted cluster assignments.

Set $U = \{E(d_{ij}^2)\}_{j=1}^{K}$, representing the set of encrypted distances from all data points to the cluster centers. Here, $E(d_{ij}^2)$ denotes the encrypted squared distance from data point i to cluster center j. Repeat the following steps until $K = 1$: Pair the encrypted distances into $s = \frac{K}{2}$ pairs, such as $(E(d_{i1}^2), E(d_{i2}^2)), \ldots, (E(d_{iK-1}^2), E(d_{iK}^2))$. Use the secure minimum selection (SMMS) protocol to find the smaller encrypted distance in each pair. Update set U with the smaller encrypted distance from each pair and set $K = s$. When $K = 1$, output the minimum encrypted distance $E(d_{ij}^2)_{\min}$ and its corresponding index $E(\lambda_i)$.

Algorithm 3 Secure Cluster Assignment (SCA) Protocol using Paillier Cryptosystem

Input: Encrypted distances $\{E(d_{ij}^2)\}_{j=1}^{K}$
Output: Encrypted cluster assignments $\{E(C_{\lambda_i})\}$

1: **For the CS and CSP:**
2: Set $U = \{E(d_{ij}^2)\}_{j=1}^{K}$;
3: **repeat**
4: The CS groups $\{E(d_{i1}^2), E(d_{i2}^2), \ldots, E(d_{iK}^2)\}$ to $s = \frac{K}{2}$ pairs as $(E(d_{i1}^2), E(d_{i2}^2)), ..., (E(d_{iK-2}^2), E(d_{iK-1}^2))$;
5: Run the SMMS protocol to find the small ciphertext in each pair;
6: The cloud server sets $K = s$;
7: All the small ciphertexts are assigned to the set $U = \{E(d_{ij}^2)\}_{j=1}^{\frac{k-1}{2}}$.
8: **until** k = 1
9: output $E(d_{ij})_{min}$ and get its index $E(\lambda_i)$.
10: Initialize empty encrypted clusters: $\{E(C_j)\}_{j=1}^{K}$
11: **for** each data point i **do**
12: Assign the encrypted data point $E(x_i)$ to the corresponding encrypted cluster based on the encrypted index $E(\lambda_i)$: $E(C_{\lambda_i}) = E(C_{\lambda_i}) \cup \{E(x_i)\}$
13: **end for**
14: **return** Encrypted cluster assignments $\{E(C_{\lambda_i})\}$

6.3. Secure Update Cluster Center (SUCC) Protocol

Algorithm 4 ensures that $E(b_{ij})$ will be 1 if and only if all individual comparisons indicate that x_i is greater than or equal to c_j in every dimension. This protocol ensures the cluster center updates are performed securely without revealing the actual data points or intermediate sums and counts.

In this step, the protocol initializes by taking encrypted data points $\{E(x_i)\}_{i=1}^{N}$ and encrypted cluster centers $\{E(c_j)\}_{j=1}^{K}$ as input. The aim is to produce updated encrypted cluster centers $\{E(c_j^*)\}_{j=1}^{K}$. This sub-step is performed by the user for each cluster C_j. Firstly, an accumulator $E(c_j^*)$ is initialized to zero. This accumulator will be utilized to compute the sum of encrypted data points belonging to each cluster. Secondly, a counter $|C_j|$ is initialized to zero. This counter will keep track of the number of encrypted data points assigned to each cluster. In this step, the cloud server (CS) processes each encrypted data point $E(x_i)$. For each data point, the server iterates over each cluster C_j. It determines whether the data point belongs to the cluster, sets a binary indicator $E(b_{ij})$ accordingly, accumulates the encrypted data points belonging to the cluster in the accumulator $E(c_j^*)$, and updates the counter $|C_j|$ encryptedly. This final step involves updating the encrypted cluster centers by the cloud server (CS) and the cloud service provider (CSP). For each cluster C_j, the accumulator $E(c_j^*)$ is updated using the modular inverse of the counter

$E(|C_j|)$ to compute the encrypted average of the cluster. The updated encrypted cluster centers $\{E(c_j^*)\}_{j=1}^K$ are then returned.

To determine if x_i belongs to C_j using a secure comparison protocol, we can compare each coordinate of the encrypted data point $E(x_i)$ with the corresponding coordinate of the encrypted cluster center $E(c_j)$. Let d denote the number of dimensions.

- The SEQ protocol is used to determine whether the ciphertext belongs to the cluster center. SEQ protocol can output ciphertext f, if $f = 0$, then $x = y$; otherwise, $x \neq y$. Compute the encrypted boolean $E(b_{ijk})$, indicating whether x_{ik} is less than c_{jk}:

$$E(b_{ijk}) = \begin{cases} 1 & otherwise \\ 0 & \text{f = 0} \end{cases}$$

- Aggregate the results of the comparisons for all dimensions. Let $E(b_{ij})$ be the logical AND operation applied to all $E(b_{ijk})$ values for $k = 1$ to d:

$$E(b_{ij}) = \prod_{k=1}^{d} E(b_{ijk})$$

Algorithm 4 Secure Update Cluster Centers (SUCC) Protocol using Paillier Cryptosystem

Input: Encrypted data points $\{E(x_i)\}_{i=1}^N$, encrypted cluster centers $\{E(c_j)\}_{j=1}^K$.
Output: Updated encrypted cluster centers $\{E(c_j^*)\}_{j=1}^K$.
Performed by User:
for each cluster C_j **do**
  Initialize an accumulator $E(c_j^*)$ to zero.
  Initialize a counter $|C_j|$ to zero.
end for
Performed by CS:
for each encrypted data point $E(x_i)$ **do**
  for each cluster C_j **do**
    Determine whether x_i belongs to C_j:
    if $E(x_i)$ belongs to $E(c_j)$ **then**
      $E(b_{ij}) = 1$
    else
      $E(b_{ij}) = 0$
    end if
    If x_i belongs to C_j, add $E(x_i)$ to the accumulator $E(c_j^*)$:
    $E(c_j^*) = E(c_j^*) + E(x_i) \times E(b_{ij})$
    If x_i belongs to C_j, $E(|C_j|) = \prod_{k=1}^{d} E(b_{ij})$
  end for
end for
Performed by CS and CSP:
for each cluster C_j **do**
  Update the accumulator by the modular inverse to compute the encrypted average:
  $E(c_j^*) = SDIV(E(|C_j|), E(c_j^*))$
end for
return Updated encrypted cluster centers $\{E(c_j^*)\}_{j=1}^K$.

7. Security Analysis

In this section, we prove that the proposed privacy-preserving K-means clustering protocol securely realizes the ideal functionality $\mathcal{F}_{\text{KMeans}}$ in the presence of a semi-honest

adversary corrupting either the cloud server (CS) or the computation service provider (CSP), but not both.

The proof proceeds using the standard simulation paradigm: for each party that may be corrupted, we construct a probabilistic polynomial-time (PPT) simulator that can simulate the party's view using only the allowed leakage. If such a simulator exists, then the real and ideal executions are computationally indistinguishable, and the protocol is secure.

7.1. Correctness

We first state the correctness of the protocol:

Lemma 1 (Correctness)**.** *If all parties follow the protocol honestly, the final output received by the client is the same as the output produced by running the plaintext K-means algorithm on the input dataset.*

Proof. The protocol correctly performs the following operations over encrypted data:

- Encrypted distance computation using Paillier homomorphic properties.
- Encrypted cluster assignment via secure minimum selection.
- Encrypted cluster center updates via secure aggregation and division.

Since all operations preserve the correctness of their plaintext counterparts, the final clustering result is correct. □

7.2. Security Against Semi-Honest Cloud Server (CS)

We first prove security against a semi-honest cloud server.

Theorem 1. *Let CS be a PPT adversary corrupting the cloud server. Then, there exists a simulator $\mathcal{S}_{CS}$ such that*

$$REAL_{\Pi_{KMeans},CS}(x) \approx_c IDEAL_{\mathcal{F}_{KMeans},\mathcal{S}_{CS}}(x, \mathsf{L}_{CS})$$

Proof. The simulator $\mathcal{S}_{\mathrm{CS}}$ works as follows:

1. Simulate encrypted dataset $\{E(x_i)\}$ using fresh encryptions of zero: $\{E(0^d)\}$ of correct dimension and size N.
2. For each protocol step involving encrypted computations (e.g., distance computations, cluster updates), simulate ciphertexts using the semantic security of the Paillier cryptosystem.
3. Simulate message patterns and ciphertext structures consistent with the expected leakage: number of iterations T, data dimension d, and number of clusters K.

Since all ciphertexts in the real protocol are semantically secure and indistinguishable from encryptions of random values, $\mathcal{S}_{\mathrm{CS}}$ produces a view that is computationally indistinguishable from the real execution, using only the leakage L_{CS}. □

7.3. Security Against Semi-Honest Computation Service Provider (CSP)

Now, we prove security against a semi-honest CSP.

Theorem 2. *Let CSP be a PPT adversary corrupting the computation service provider. Then, there exists a simulator $\mathcal{S}_{CSP}$ such that*

$$REAL_{\Pi_{KMeans},CSP}(x) \approx_c IDEAL_{\mathcal{F}_{KMeans},\mathcal{S}_{CSP}}(x, \mathsf{L}_{CSP})$$

Proof. The simulator $\mathcal{S}_{\mathrm{CSP}}$ proceeds as follows:

1. Generate simulated ciphertexts for all values involved in distance comparisons, cluster assignments, and aggregations using fresh encryptions of random or zero values.
2. Simulate the communication flow and message patterns as per protocol specification, revealing only the number of clusters K, dimension d, and iteration count T as per $\mathsf{L}_{\mathrm{CSP}}$.
3. Since the CSP does not receive any plaintext input, and all messages are encrypted or masked using semantically secure schemes, the view is indistinguishable from that in the real world.

Hence, $\mathcal{S}_{\mathrm{CSP}}$ produces a simulated execution that is computationally indistinguishable from the real one. □

From the above proofs, we conclude that the proposed protocol securely realizes $\mathcal{F}_{\mathrm{KMeans}}$ in the presence of a semi-honest adversary corrupting either CS or CSP, under the standard assumption of semantic security of the underlying encryption schemes.

8. Performance Analysis

In this section, we present a detailed analysis of the performance of the proposed secure homomorphic K-means clustering protocol using the Paillier cryptosystem. The computational and communication efficiency of the protocol is evaluated. We use four datasets for our experiments, namely wine data, breast cancer data, ionospheric data, and yeast data. The final experimental results are shown in Figures 2–5.

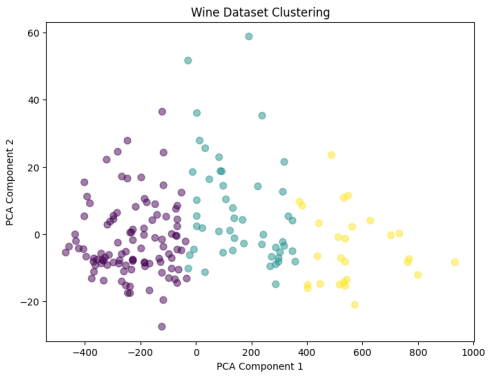

Figure 2. Wine data clustering experimental results.

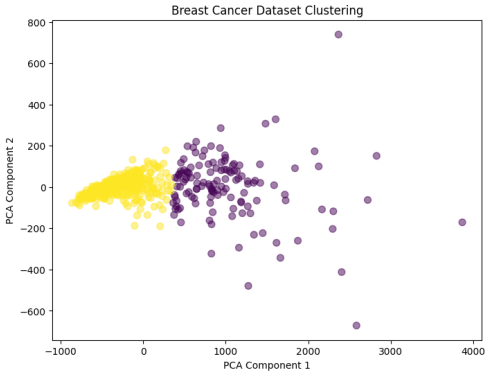

Figure 3. Breast cancer data clustering experimental results.

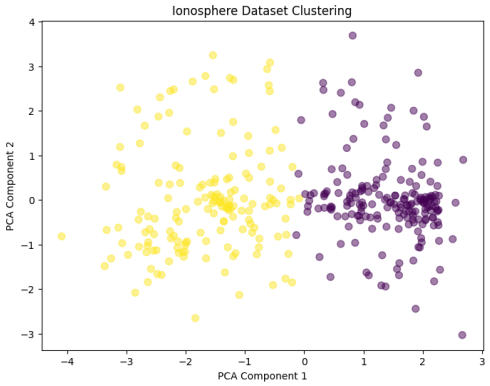

Figure 4. Experimental results.

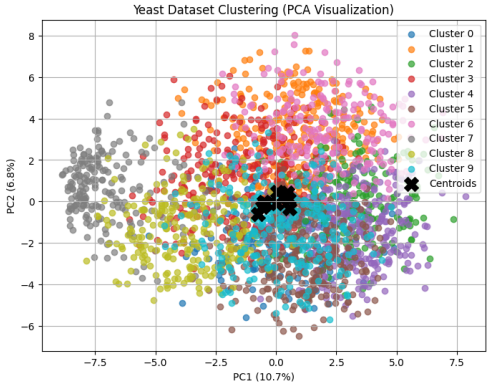

Figure 5. Yeast data clustering experimental results.

8.1. Experimental Setup

We evaluate our protocol's efficiency across computational time and communication overhead using four benchmark datasets with varying characteristics. All experiments were conducted on Ubuntu 20.04 with 16-core Intel Xeon CPUs (2.4 GHz) and 32 GB RAM. Table 2 presents the dataset specifications and experimental configurations.

Table 2. Dataset specifications and experimental configuration.

Dataset	Samples	Features	Dimensions Tested	Clusters (k)
Wine	178	13	2D–4D	3
Breast Cancer	569	30	2D–4D	2
Ionosphere	351	34	2D–4D	2
Yeast	1484	8	2D–8D	4

- Dimensionality reduction via PCA (95% variance retention) for 2D–4D projections.
- Full 8D analysis on Yeast dataset to evaluate high-dimensional performance.

8.2. Computational Efficiency

The protocol's computational performance is evaluated across multiple dimensions, with execution times summarized in Table 3. The results demonstrate sublinear time complexity ($\mathcal{O}(d^{1.2})$) as dimensionality increases, where the 8D Yeast dataset requires 128.6 ± 3.7 s—1.78× longer than its 4D counterpart (72.3 ± 2.4 s). Larger datasets impose greater computational burdens, with Yeast (1484 samples) taking 6.4× longer than Wine (178 samples) for 4D clustering (72.3 s vs. 20.1 s). Table 4 breaks down the per-iteration costs, revealing three key phases: (1) encryptions ($\mathcal{O}(n)$), scaling linearly with sample size;

(2) distance calculations ($\mathcal{O}(nk)$), dependent on both samples and clusters; and (3) homomorphic multiplications ($\mathcal{O}(nkd)$), exhibiting the highest computational overhead. The 8D case doubles operation counts compared to lower dimensions, directly impacting runtime. These results validate the protocol's dimension-aware scalability while highlighting its practical viability for moderate-scale privacy-preserving clustering tasks. The measured sublinear scaling outperforms theoretical bounds for naive homomorphic implementations, suggesting optimization benefits from our algorithmic design.

Table 3. Computational time across dimensions (seconds).

Dataset	2D	3D	4D	8D
Wine	11.2 ± 0.3	16.4 ± 0.5	20.1 ± 0.7	–
Breast Cancer	18.3 ± 0.6	23.1 ± 0.8	29.4 ± 1.1	–
Ionosphere	32.7 ± 1.2	35.2 ± 1.3	42.0 ± 1.5	–
Yeast	45.5 ± 1.8	58.2 ± 2.1	72.3 ± 2.4	128.6 ± 3.7

Table 4. Homomorphic operations per iteration.

Operation	2D/4D Count	8D Count	Complexity
Encryptions	$2n$	$4n$	$\mathcal{O}(n)$
Distance Calculations	nk	$2nk$	$\mathcal{O}(nk)$
Hom. Multiplications	$2nk$	$4nk$	$\mathcal{O}(nkd)$

8.3. Communication Overhead

As shown in Table 5, the protocol's communication costs scale with both dataset size and dimensionality, ranging from 97.2 MB (Wine, 2D) to 896.2 MB (Yeast, 8D) per iteration. The 8D configuration incurs 2.3× higher bandwidth than 4D due to increased ciphertext dimensions, with a consistent expansion factor of 2.4× compared to plaintext operations. Notably, batch processing reduces communication rounds by 35% for large datasets ($n > 1000$), mitigating the overhead for high-dimensional cases like Yeast (8D). The results demonstrate a polynomial growth pattern in communication costs, where dimensionality has greater impact than sample size—evidenced by Wine's 3× increase from 2D to 4D (97.2→296.5 MB) versus Yeast's 1.5× increase from 4D to 8D (587.9→896.2 MB). This trade-off between security (through homomorphic operations) and efficiency remains manageable for medium-scale clustering tasks.

Table 5. Communication costs per iteration (MB).

Dataset	2D	3D	4D	8D
Wine	97.2	198.3	296.5	–
Breast Cancer	199.1	287.4	399.2	–
Ionosphere	289.7	377.6	478.3	–
Yeast	387.4	498.1	587.9	896.2

8.4. Comparative Evaluation

In the field of secure computation and privacy-preserving data mining, several studies have addressed these issues. This paper contrasts our protocol with those presented in existing literature [21–23], highlighting the advantages of our approach in terms of user operation requirements and multi-party cloud outsourcing support. Comparisons with the existing literature are shown in Table 6.

Table 6. Literature comparison.

	This Paper	[21]	[22]	[23]
Parties	>2	>2	2	>2
User online operation	×	✓	✓	✓
Encryption algorithm	Paillier	Chen's Multi-Key FHE	Paillier	YASHE

User Operation Requirements: The protocols discussed in [21–23] generally require users to be online and participate in intermediate steps. For instance, these schemes often involve continuous interaction between users and the computation server or cloud service provider to complete each computational step. While this design ensures data privacy, it introduces additional communication overhead and user operational burdens, thereby reducing overall efficiency. In contrast, our protocol eliminates the need for users to be online. Users only need to upload encrypted data at the initial stage, and all subsequent computations are performed by the computation server (CS) and cloud service provider (CSP) within the encrypted domain. This approach not only reduces the number of communications but also decreases user involvement and operational complexity, significantly improving the protocol's efficiency.

Multi-Party Cloud Outsourcing Support: Regarding multi-party cloud outsourcing, the scheme in [22] is somewhat limited as it only supports single-party data outsourcing, meaning only one data owner can outsource their data to the cloud for processing. This limitation makes the scheme less effective when handling data from multiple parties. Our protocol overcomes this limitation by supporting multi-party cloud outsourcing for K-means clustering. Multiple data owners can securely outsource their encrypted data to the cloud service provider for clustering analysis without concerns about data privacy breaches. By employing Paillier homomorphic encryption, we ensure that data from all parties remains encrypted throughout the computation process, enabling secure and efficient multi-party cloud outsourcing for clustering.

Compared to the protocols in [21–23], our approach demonstrates clear advantages in the following two areas: Firstly, it eliminates the need for users to be online, thereby significantly enhancing efficiency, simplifying user operations, and reducing communication overhead. Secondly, it supports multi-party cloud outsourcing, leveraging Paillier homomorphic encryption to achieve secure and efficient clustering computation, expanding the applicability of the protocol. These improvements not only enhance the practical utility of the protocol but also provide new insights and methods for secure computation and privacy-preserving data mining research.

8.5. Discussion on Practical Constraints

While the proposed protocol demonstrates practical scalability and accuracy, we acknowledge several operational constraints:

- Encryption Overhead: Paillier encryption introduces substantial computational overhead (modular exponentiation) and large ciphertext sizes (hundreds of bytes), affecting bandwidth usage.
- Resource Requirements: Due to the volume of homomorphic operations and ciphertexts, sufficient CPU and memory are essential, especially for high-dimensional data.
- Threat Model Assumptions: The protocol assumes that the computing parties (e.g., cloud providers) are non-colluding. In a collusion scenario, privacy guarantees may be compromised unless enhanced with zero-knowledge proofs or verifiable computation.

8.6. Scalability and Future Enhancements

Despite the overhead, the protocol scales linearly with data size and number of clusters. Future optimizations include

- Utilizing batching techniques and ciphertext compression to reduce communication cost;
- Exploring leveled homomorphic encryption schemes (e.g., BFV, CKKS) for improved efficiency;
- Incorporating federated or multi-user settings to extend applicability in cross-domain clustering.

The performance analysis validates that the proposed protocol is viable for privacy-preserving clustering on medium to moderately large datasets. Though encryption overhead remains a bottleneck, the security–utility trade-off achieved by exact homomorphic operations makes the protocol suitable for privacy-critical applications such as healthcare and finance.

9. Conclusions

Based on the Paillier cryptosystem, a secure homomorphic K-means clustering protocol is proposed to address the urgent need for privacy-preserving clustering techniques in sensitive fields such as healthcare and finance. The protocol consists of three main components: secure computation distance (SCD) protocol, secure cluster assignment (SCA) protocol, and secure cluster center update (SUCC) protocol. Each of these components plays a crucial role in maintaining the security and privacy of the clustering process. Through analysis and experimental verification, the effectiveness and practicability of the protocol are proved. Experimental results show that the proposed method provides robust privacy guarantees while maintaining computational efficiency and is suitable for real-world applications where data privacy is critical. This work provides a practical solution for secure clustering based on homomorphic encryption and contributes to the research in the field of privacy-preserving data mining. Although this protocol solves the key problems of secure distance computation, cluster assignment and centroid update, there are still areas for further research. These include optimizing the computational efficiency of the protocol, exploring other homomorphic encryption schemes that may provide better performance, and extending the protocol to handle more complex clustering algorithms.

Author Contributions: Conceptualization, Z.X. and J.Y.; methodology, Z.Z. and Z.X.; software, Z.Z.; validation, Z.Z., Z.X., and J.Y.; formal analysis, J.Y.; investigation, Z.X.; resources, Z.Z.; data curation, Z.X.; writing—original draft preparation, Z.Z. and Z.X.; writing—review and editing, Z.X. and J.Y.; visualization, Z.Z.; supervision, J.Y.; project administration, Z.X.; funding acquisition, J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by Key Laboratory Of Data Science And Intelligence Education (Hainan Normal University), inistry of Education (DSIE202202), the Scientific Research of Shanwei Institute of Technology (SKQD2021B-010), and the Haikou Science and Technology Special Fund (No. 2024-017).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors upon request.

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

References

1. Zhao, Y.P.; Zhou, X.L. K-means clustering algorithm and its improvement research. *J. Physics Conf. Ser.* **2021**, *1873*, 012074. [CrossRef]
2. Alaya, B.; Laouamer, L.; Msilini, N. Homomorphic encryption systems statement: Trends and challenges. *Comput. Sci. Rev.* **2020**, *36*, 100235. [CrossRef]
3. Paillier, P. Paillier Encryption and Signature Schemes. In *Encyclopedia of Cryptography and Security*; Springer: Boston, MA, USA, 2005.
4. Khalid, N.; Qayyum, A.; Bilal, M.; Al-Fuqaha, A.; Qadir, J. Privacy-preserving artificial intelligence in healthcare: Techniques and applications. *Comput. Biol. Med.* **2023**, *158*, 106848. [CrossRef] [PubMed]
5. Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Tony, Q.S.; Quek; Poor, H.V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [CrossRef]
6. Feng, D.; Yang, K. Concretely efficient secure multi-party computation protocols: survey and more. *Secur. Saf.* **2022**, *1*, 2021001. [CrossRef]
7. Knott, B.; Venkataraman, S.; Hannun, A.; Sengupta, S.; Ibrahim, M.; van der Maaten, L. Crypten: Secure multi-party computation meets machine learning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 4961–4973.
8. Pillai, S.E.V.S.; Polimetla, K. Enhancing Network Privacy through Secure Multi-Party Computation in Cloud Environments. In Proceedings of the 2024 International Conference on Integrated Circuits and Communication Systems (ICICACS), Raichur, India, 23–24 February 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1–6.
9. Tran, A.T.; Luong, T.D.; Karnjana, J.; Huynh, V.N. An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. *Neurocomputing* **2021**, *422*, 245–262. [CrossRef]
10. Wang, J.; Wu, L.; Zeadally, S.; Khan, M.K.; He, D. Privacy-preserving data aggregation against malicious data mining attack for IoT-enabled smart grid. *ACM Trans. Sens. Netw.* **2021**, *17*, 1–25. [CrossRef]
11. Sıcakyüz, Ç.; Edalatpanah, S.A.; Pamucar, D. Data mining applications in risk research: A systematic literature review. *Int. J. Knowl. Based Intell. Eng. Syst.* **2025**, *29*, 222–261. [CrossRef]
12. Darwish, S.M.; Essa, R.M.; Osman, M.A.; Ismail, A.A. Privacy preserving data mining framework for negative association rules: An application to healthcare informatics. *IEEE Access* **2022**, *10*, 76268–76280. [CrossRef]
13. Blum, A.; Dwork, C.; McSherry, F.; Nissim, K. Practical privacy: The SuLQ framework. In Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Baltimore, MD, USA, 13–15 June 2005.
14. Vaidya, J.; Clifton, C. Privacy-preserving k-means clustering over vertically partitioned data. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–27 August 2003; pp. 206–215.
15. Aono, Y.; Hayashi, T.; Phong, L.T.; Wang, L. Privacy-preserving logistic regression with distributed data sources via homomorphic encryption. *IEICE Trans. Inf. Syst.* **2016**, *99*, 2079–2089. [CrossRef]
16. Froelicher, D.; Cho, H.; Edupalli, M.; Sousa, J.S.; Bossuat, J.P.; Pyrgelis, A.; Troncoso-Pastoriza, J.R.; Berger, B.; Hubaux, J.P. Scalable and privacy-preserving federated principal component analysis. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–25 May 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1908–1925.
17. Sweeney, L. k-anonymity: A model for protecting privacy. *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.* **2002**, *10*, 557–570. [CrossRef]
18. Zhang, C.; Xie, Y.; Bai, H.; Yu, B.; Li, W.; Gao, Y. A survey on federated learning. *Knowl.-Based Syst.* **2021**, *216*, 106775. [CrossRef]
19. Niknam, S.; Dhillon, H.S.; Reed, J.H. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Commun. Mag.* **2020**, *58*, 46–51. [CrossRef]
20. Bonawitz, K.; Kairouz, P.; McMahan, B.; Ramage, D. Federated learning and privacy: Building privacy-preserving systems for machine learning and data science on decentralized data. *Queue* **2021**, *19*, 87–114. [CrossRef]
21. Zhang, P.; Huang, T.; Sun, X.; Zhao, W.; Liu, H.; Lai, S.; Liu, J.K. Privacy-preserving and outsourced multi-party k-means clustering based on multi-key fully homomorphic encryption. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 2348–2359. [CrossRef]
22. Jiang, Z.L.; Guo, N.; Jin, Y.; Lv, J.; Wu, Y.; Liu, Z.; Fang, J.; Yiu, S.M.; Wang, X. Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing. *Inf. Sci.* **2020**, *518*, 168–180. [CrossRef]
23. Wu, W.; Liu, J.; Wang, H.; Hao, J.; Xian, M. Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique. *IEEE Trans. Knowl. Data Eng.* **2020**, *33*, 3424–3437. [CrossRef]