applied sciences

MDPI

Article

A Parallel Privacy-Preserving k-Means Clustering Algorithm for Encrypted Databases in Cloud Computing

Youngho Song [1], Hyeong-Jin Kim [1], Hyun-Jo Lee [1] and Jae-Woo Chang [2,*]

[1] Department of Computer Engineering, Jeonbuk National University, Jeonju-si 54896, Republic of Korea; songyoungho@jbnu.ac.kr (Y.S.); yeon_hui4@jbnu.ac.kr (H.-J.K.); o2near@jbnu.ac.kr (H.-J.L.)

[2] Department of Computer Science & Artificial Intelligence, Jeonbuk National University, Jeonju-si 54896, Republic of Korea

* Correspondence: jwchang@jbnu.ac.kr

Abstract: With the development of cloud computing, interest in database outsourcing has recently increased. However, when the database is outsourced, there is a problem in that the information of the data owner is exposed to internal and external attackers. Therefore, in this paper, we propose decimal-based encryption operation protocols that support privacy preservation. The proposed protocols improve the operational efficiency compared with binary-based encryption operation protocols by eliminating the need for repetitive operations based on bit length. In addition, we propose a privacy-preserving k-means clustering algorithm using decimal-based encryption operation protocols. The proposed k-means clustering algorithm utilizes efficient decimal-based protocols that enhance the efficiency of the encryption operations. To provide high query processing performance, we also propose a parallel k-means clustering algorithm that supports thread-based parallel processing by using a random value pool. Meanwhile, a security analysis of both the proposed k-means clustering algorithm and the proposed parallel algorithm was performed to prove their data protection, query protection, and access pattern protection capabilities. Through our performance analysis, the proposed k-means clustering algorithm shows about 10~13 times better performance compared with the existing algorithms.

Keywords: secure protocol; privacy-preserving k-Means clustering algorithm; encrypted database; database outsourcing; cloud computing

Citation: Song, Y.; Kim, H.-J.; Lee, H.-J.; Chang, J.-W. A Parallel Privacy-Preserving k-Means Clustering Algorithm for Encrypted Databases in Cloud Computing. *Appl. Sci.* **2024**, *14*, 835. https://doi.org/10.3390/app14020835

Academic Editor: José Salvador Sánchez Garreta

Received: 12 December 2023
Revised: 11 January 2024
Accepted: 17 January 2024
Published: 18 January 2024

1. Introduction

With the advancement of cloud computing, database outsourcing has become increasingly popular [1]. Database outsourcing refers to the delegation of database management by the data owner (DO) to a specialized entity (e.g., the cloud). This allows the data owner to make savings on the computational and human resources required for managing their database, enabling investment in the improvement and development of service quality. Cloud services not only store outsourced databases but also provide query processing and data mining services for extracting meaningful information based on the data [2–5]. Additionally, data owners can benefit from cost savings by dynamically utilizing computational resources as needed.

However, outsourcing databases poses a security challenge, i.e., exposing the database to potential internal and external attacks [6]. Consequently, protecting the database becomes crucial for data owners, given that databases may contain sensitive information and are valuable assets [7]. Users receiving services may also face privacy concerns if their query content sent to the cloud is leaked [8], revealing personal information such as preferences and tendencies [9,10].

Previous strategies modify plaintexts to their substituted data and outsource them to a cloud [11–16]. However, these previous strategies cannot completely protect both the data and queries because they are weak to various attacks. To tackle this problem, recent strategies encrypt the original data and outsource them to the cloud [17–23]. Therefore, research has been conducted on data protection, query protection to prevent query content exposure, and result protection to avoid exposing query results.

The k-means clustering algorithm is a prominent data mining technique that identifies patterns in a dataset by calculating distances between unclassified data points and centroids, assigning them to the closest clusters. The k-means clustering algorithm is used for applications in various fields, including pattern analysis, machine learning, image analysis, text mining, and search engines. Security-enhanced k-means clustering algorithms have been proposed. First, D. Liu et al. [24] proposed a k-means clustering algorithm using homomorphic encryption in an outsourcing environment. However, their homomorphic encryption system is vulnerable to chosen ciphertext attacks, and the information about the selected indexes is exposed, preventing access pattern protection. Then, F. Rao et al. [25] proposed a k-means clustering algorithm using homomorphic encryption supporting addition. Their study, which is based on encryption operation protocols, not only protects data but also supports query protection. It enables access pattern protection by not revealing which cluster data belong to. However, its drawbacks include significant performance variations due to the arbitrary initialization of centroids and high query processing costs associated with using binary array-based encryption operation protocols.

Therefore, this paper proposes a solution to address these issues by introducing a decimal-based encryption operation protocol. The proposed protocol enhances the operational efficiency by eliminating the need for repetitive operations of bit length, distinct from binary-based encryption operation protocols. Furthermore, based on the decimal-based operation protocol, we propose a k-means clustering algorithm that supports information protection in cloud computing. The research contributions of this paper are as follows:

- This paper proposes decimal-based encryption operation protocols, like ASMIN and ASMINn. The proposed protocols address the challenges of the existing binary-based encryption operation protocols, where the performance degradation is proportional to the data size. The proposed protocols overcome the limitation, providing excellent processing performance independently of data size.
- This paper proposes a privacy-preserving k-means clustering algorithm that utilizes the proposed decimal-based encryption operation protocols. While providing superior processing performance, the proposed k-means clustering algorithm ensures that data, queries, and data access patterns are safeguarded in order to protect original databases and user information in cloud computing.
- This paper proposes a privacy-preserving parallel k-means clustering algorithm. To the best of our knowledge, this is the first work to study a privacy-preserving parallel k-means clustering algorithm. To perform parallel processing, we utilize a thread pool to prevent data bottlenecks and parallelize encryption operation protocols for efficient support of k-means clustering. However, when parallelizing homomorphic encryption techniques, there is the issue of the increased operational overhead per core, leading to bottlenecks and performance degradation. To mitigate this bottleneck, we use a random value pool to reduce the computational cost by preprocessing operations that generate random values for hiding data and encrypting them in the encryption operation protocol.

This paper is structured as follows: In Section 2, we explain the Paillier cryptosystem and the adversarial attack model. Section 3 describes the overall system architecture and the newly proposed protocol. Section 4 presents the newly proposed k-means clustering algorithm, and Section 5 proposes the parallel k-means clustering algorithm. In Section 6, we provide a security analysis of the proposed k-means clustering algorithm. Section 7 shows the performance evaluation of the proposed k-means clustering algorithm. In Section 8, we discuss the performance evaluation. Finally, in Section 9, we conclude the paper and discuss future research directions.

2. Background and Related Research

2.1. Background Knowledge

The Paillier cryptosystem [26] is a prominent additive homomorphic encryption technique characterized by a probabilistic encryption scheme where the same value results in different ciphertexts each time it is encrypted. In the Paillier encryption system, the encryption key (public key) pk is given as (N, g), where N is the product of two large prime numbers (e.g., p, q) and g is a randomly chosen integer in $Z_N{}^2$. On the other hand, the decryption key (secure key) sk in the Paillier encryption system is given as (λ, μ), where λ is the least common multiple (LCM) of p, q, μ is L (g^{λ} mod N^2))$^{-1}$ mod N, and $L(x) = \frac{x-1}{N}$. The Paillier encryption system exhibits the unique property of computing ciphertexts corresponding to the addition of plaintexts through operations between ciphertexts without the need for a decryption process.

$$E(m_1 + m_2) = E(m_1) \times E(m_2)\ mod\ N^2 \tag{1}$$

- Homomorphic addition: The multiplication of two ciphertexts $E(m_1)$ and $E(m_2)$ generates the ciphertext of the sum of their plaintexts m_1 and m_2 (Equation (1)).
- Homomorphic multiplication: The m_2-th power of ciphertext $E(m_1)$ generates the ciphertext of the multiplication of m_1 and m_2 (Equation (2)).

$$E(m_1 \times m_2) = E(m_1)^{m_2}\ mod\ N^2 \tag{2}$$

- Semantic security: Encryptions of the same plaintexts generate different ciphertexts in the same public key (Equation (3)).

$$m_1 = m_2 \nRightarrow E(m_1) = E(m_2) \tag{3}$$

The adversarial attack model in an outsourced database environment can be categorized into two attack models: the semi-honest attack model and the malicious attack model [27]. The semi-honest (or honest-but-curious) attack model implies that the cloud executes the assigned protocol honestly but may attempt to gain additional information about the data owner and query requester based on the information acquired during the protocol's execution. The malicious attack model, on the other hand, refers to the cloud deviating from the given protocol and attempting to acquire information with malicious intent. Therefore, when verifying the security of a specific protocol or algorithm against a malicious attack model, it can be demonstrated that the protocol is also secure against other attack models. However, protocols secure against malicious attack models often pose challenges in terms of implementation and usage due to the high costs involved, making them difficult to apply in practical environments. On the contrary, protocols secure against the semi-honest attack model are not only applicable in real-world scenarios but also serve as a foundation for designing protocols secure against malicious attack models. Hence, in this paper, we conduct research considering the semi-honest attack model as in previous studies [17,28–30].

Definition 1. *Assuming α_i is the input parameter of cloud C_i, $\prod_i$ ($\rho(\alpha)$) is the execution image of C_i for the protocol ρ. If the simulating execution image $\prod S_i$ ($\rho(\alpha)$) is indistinguishable from $\prod_i$ ($\rho(\alpha)$), the protocol ρ is a secure protocol under the semi-honest attack model.*

2.2. Related Work

The k-means clustering algorithm is one of the representative data mining techniques, identifying the characteristics of a dataset by calculating distances between unclassified data points and centroids, incorporating them into the closest clusters. This algorithm is used for applications in various fields such as pattern analysis, machine learning, image analysis, text mining, and search engines. Security-enhanced k-means clustering algorithms have been proposed. First, D. Liu et al. [24] proposed a k-means clustering algorithm utilizing homomorphic encryption in an outsourcing environment. The algorithm constructs an index that allows for comparisons in the encrypted state by leveraging the characteristics of their self-developed homomorphic encryption technique. This index facilitates comparisons without data leakage. However, the homomorphic encryption system used in their study is vulnerable to chosen ciphertext attacks. Additionally, although encrypted, the information about the selected indexes is exposed, allowing for the determination of intermediate results, posing the problem of revealing access patterns. Then, F. Rao et al. [25] proposed the k-means clustering algorithm using a homomorphic encryption system supporting addition. The algorithm calculates distances for the entire dataset, assigns each data point to the nearest centroid, and updates the centroids through merging encrypted data belonging to the same cluster. The SSED protocol [25], which calculates Euclidean distances, is utilized to compute distances between the previous and new centroids. If the calculated distance is smaller than the query, the centroid is returned; otherwise, the process is repeated. Their study is based on encryption operation protocols, providing data protection and supporting query protection. Since distance calculations for the entire dataset do not expose the cluster to which each data point belongs, access patterns can also be protected. However, their study has drawbacks, including the arbitrary initialization of centroids leading to significant performance variations and the use of binary array-based encryption operation protocols, resulting in high query processing costs. Finally, Y. Yang et al. [31] proposed a privacy-preserving smart IoT-based healthcare big data storage system with self-adaptive access control. The aim is to ensure the security of patients' healthcare data, realize access control for normal and emergency scenarios, and support smart deduplication to save storage space in the big data storage system. The medical files generated by the healthcare IoT network are encrypted and transferred to the storage system, which can be securely shared among the healthcare staff from different medical domains by leveraging a cross-domain access control policy.

Table 1 summarizes the existing studies based on their characteristics. We compare them with regard to three major characteristics, i.e., hiding access patterns, computational overhead, and security risk. First, F. Rao et al.'s work [25] and our work can protect data access patterns, while D. Liu et al.'s work [24] and Y. Yang et al.'s work [31] cannot protect them. Second, our work requires moderate computational overhead because we use decimal-based encryption operations, while D. Liu et al.'s work [24] and F. Rao et al.'s work [25] have high computational overhead due to the use of binary-based encryption operations. Finally, F. Rao et al.'s work [25] and our work have a low security risk with regard to security because they protect sensitive data, users' queries, and data access patterns, while D. Liu et al.'s work [24] and Y. Yang et al.'s work [31] have a high security risk because they protect both sensitive data and users' queries.

Table 1. Comparison of the existing studies.

Schemes \ Features	Data Privacy	Query Privacy	Hiding Data Access Pattern	Index	Computational Overhead	Encryption	User Involvement Computation	Security Risk
D. Liu et al. [24]	Supported	Supported	Not supported	Order-preserving index	High	Homomorphic encryption	Partially involved	High
F. Rao et al. [25]	Supported	Supported	Supported	None	High	Homomorphic encryption supporting addition	Not involved	Low
Y. Yang et al. [31]	Supported	Supported	Not supported	None	High	Attribute-based encryption	Not involved	High
Proposed	Supported	Supported	Supported	Kd-tree	Moderate	Paillier encryption	Not involved	Low

3. Overall System Architecture

3.1. System Architecture

In the outsourcing environment considered in this paper, there exist non-colluding clouds C_A and C_B. C_A and C_B adopt the semi-honest attack model, which means they perform query processing honestly but attempt to infer the original data and user preferences based on the information generated during the query processing. However, they do not collude with each other to exchange data and information. The characteristic of this attack model is that the attacker leaves no trace, making it difficult to determine when the attack occurred. As passive attacks have the difficulty of detection and recovery, preventing attacks through precaution and protection is crucial. Indeed, the assumption of the semi-honest attack model on clouds has been widely utilized in various fields dealing with similar issues [17,30,32].

Figure 1 illustrates the overall system architecture of this paper, consisting of a Data Owner (DO), an Authenticated User (AU), Cloud A (C_A), and Cloud B (C_B). The DO possesses an original database (data) consisting of n records $data_i$ ($1 \leq i \leq n$). Each record is composed of m attributes or columns, and the j-th attribute of the i-th record is denoted as $data_{i,j}$ ($1 \leq i \leq n$, $1 \leq j \leq m$). To support indexing for this database, the DO performs kd-tree-based data indexing. In this case, the level of the kd-tree is denoted as h, the number of leaf nodes is 2h-1, and the number of data points that a leaf node can store (FanOut) is denoted as F. The leaf nodes of the kd-tree store the lower bounds ($lb_{z,j}$) and upper bounds ($ub_{z,j}$) for each dimension that the node is responsible for ($1 \leq z \leq 2^{h-1}$, $1 \leq j \leq m$) as well as the ids of the original data points within the range of that leaf node.

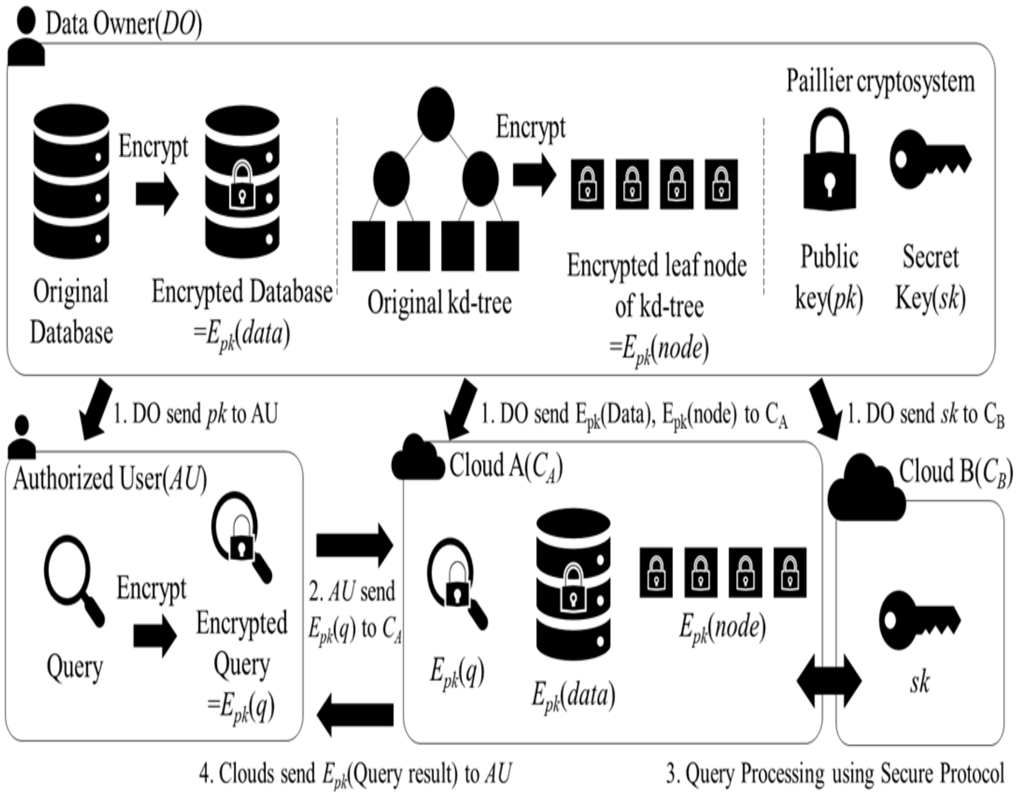

Figure 1. Overall system architecture.

The encryption of the database utilizes the Paillier cryptosystem [26]. To achieve this, the DO generates a pair of keys: the public key (encryption key, pk) and the private key (decryption key, sk). Using the encryption key, the DO encrypts the database. The encryption of the database is performed on a dimension-by-dimension basis for each record, resulting in the creation of ciphertext for the original database (i.e., $E(data_{i,j})$ for $1 \leq I \leq n$

and $1 \leq j \leq m$). Additionally, the DO encrypts the lower and upper bounds for each dimension of the leaf nodes in the constructed kd-tree. The encryption of the kd-tree is conducted on a dimension-by-dimension basis for the lower and upper bounds of each leaf node. This process leads to the creation of ciphertext for the kd-tree leaf nodes (i.e., $E(lb_{z,j})$ and $E(ub_{z,j})$ for $1 \leq z \leq 2^{h-1}$ and $1 \leq j \leq m$).

3.2. Secure Protocol

The existing research [9,10,17] performs query processing and data mining using encryption operation protocols that support bitwise operations, multiplication, comparison, and overlapping area checks. However, a drawback of the encryption operation protocols used in the existing research is the performance degradation in processing depending on the data domain, as they perform operations through the binary numeral system. Figure 2 illustrates the execution process of the SMIN protocol, the typical MIN operation proposed in [17]. The SMIN protocol transforms E(5) and E(3) into binary ciphertext arrays for the operation. Assuming a bit length of 4 in the figure, E(5) is transformed to {E(0), E(1), E(0), E(1)}, and E(3) is transformed into {E(0), E(0), E(1), E(1)}. Since the SMIN protocol's execution steps are composed of bit operations, four bit operations are performed at each step. As a result, the existing SMIN protocol requires a total of 40 operations to perform the MIN operation on data with a domain of 0 to 24 (=16). Assuming a data domain of 29 for real data, the required domain for distance operations is 29 × 29 = 218. In such an environment, executing the SMIN protocol requires a total of 18 × 18 = 324 Paillier encryption addition and multiplication operations.

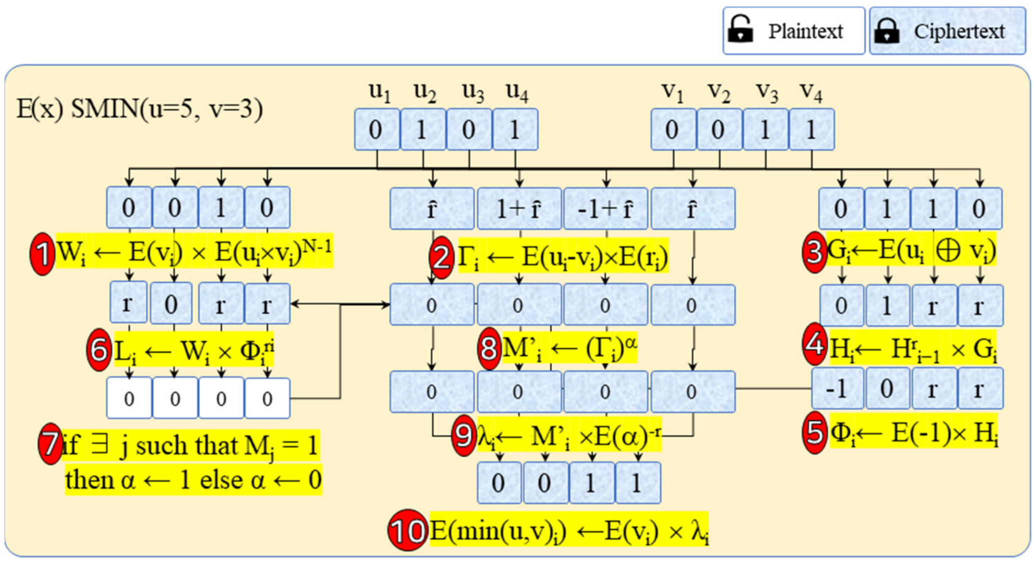

Figure 2. Execution process of the SMIN protocol.

On the other hand, the execution process of the ASMIN protocol proposed in this paper is depicted in Figure 3. The ASMIN protocol performs the MIN operation using arithmetic operations based on decimal numbers (i.e., Paillier addition and multiplication properties). The proposed ASMIN protocol securely calculates the smaller of two inputs through a total of seven Paillier encryption addition and multiplication operations. Furthermore, since the proposed ASMIN protocol represents data in decimal numbers, it offers the advantage of consistent performance regardless of the data domain.

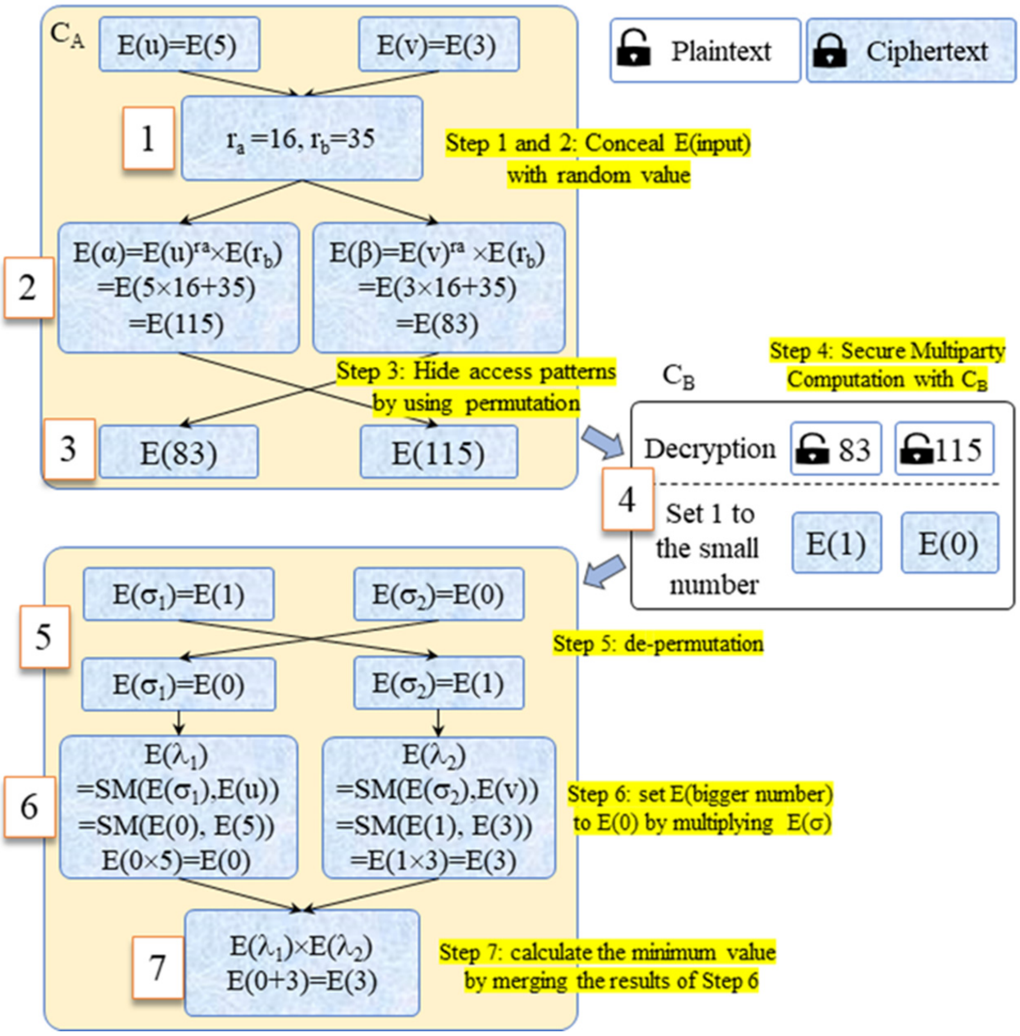

Figure 3. Execution process of the ASMIN protocol.

3.2.1. Advanced Secure MINimum (ASMIN) Protocol

The ASMIN protocol is a protocol that, given two encrypted data points E(u) and E(v), returns the smaller of the two to C_A, utilizing the properties of the Paillier encryption system. The proposed ASMIN protocol hides encrypted data through randomization for comparison. The execution process of the ASMIN protocol is outlined in Algorithm 1. Firstly, C_A selects two random constants (r_a, r_b) from the random value pool. Using the properties of the Paillier encryption system, C_A calculates $E(u)r_a \times E(r_b) = E(u \times r_a + r_b)$ and $E(v \times r_a) \times E(r_b) = E(v \times r_a + r_b)$. Secondly, selecting F, if F is F_0, C_A sends $E(u \times r_a + r_b)$ and $E(v \times r_a + r_b)$ to C_B; if F is F_1, C_A sends $E(v \times r_a + r_b)$ and $E(u \times r_a + r_b)$ to C_B. Thirdly, C_B decrypts the received $E(u \times r_a + r_b)$ and $E(v \times r_a + r_b)$ and performs the comparison. For explanation purposes in this paper, let us assume that the selected F is F_0. In this case, if $u \times r_a + r_b$ is less than or equal to $v \times r_a + r_b$, C_B sends $E(\alpha) = E(1)$ to C_A; otherwise, C_B sends $E(\alpha) = E(0)$ to C_A. Lastly, if F is F_0, C_A performs SM(E(α), E(u)) × SM(E(β), E(v)); if F is F_1, C_A performs SM(E(β), E(u)) × SM(E(α), E(v)). Here, the Secure Multiplication (SM) protocol is the multiplication protocol proposed by Y. Elmehdwi et al. [17]. This protocol calculates the ciphertext $E(\alpha \times \beta)$ with two encrypted data points E(α) and E(β) representing α and β, respectively. Through this process, C_A securely obtains the result of E(u) if $u \leq v$ or E(v) if $u > v$.

Algorithm 1 ASMIN (Advanced Secure MINimum)

Input: E(u), E(v)
Output: E(u) when u $\leq$ v, otherwise E(v)
C_A:
01. select <r_a, E(r_a)>, <r_b, E(r_b)> in the random value pool
02. E(u $\times$ r_a) $\leftarrow$ E(u)ram
03. E(v $\times$ r_a) $\leftarrow$ E(v)r_a
04. E(u $\times$ r_a + r_b) $\leftarrow$ E(u $\times$ r_a) $\times$ E(r_b)
05. E(v $\times$ r_a + r_b) $\leftarrow$ E(v $\times$ r_a) $\times$ E(r_b)
06. if F_0: E(u $\times$ r_a + r_b) then, E(v $\times$ r_a + r_b) send to C_B
07. else if F_1: E(v $\times$ r_a + r_b) then, E(u $\times$ r_a + r_b) send to C_B
C_B:
08. Decrypt E(u $\times$ r_a + r_b), E(v $\times$ r_a + r_b)
09. if (u $\times$ r_a + r_b $\leq$ v $\times$ r_a + r_b) then, <E(α), E(β)> $\leftarrow$ <E(1), E(0)>
10. else then, <E(α), E(β)> $\leftarrow$ <E(0), E(1)>
11. send <E(α), E(β)> to C_A
C_A:
12. if F_0 then, E(result) = SM(E(α), E(u)) $\times$ SM(E(β), E(v))
13. else then, E(result) = SM(E(β)), E(u)) $\times$ SM(E(α), E(v))
14. **return** E(result)
End Algorithm

Figure 4 illustrates an example of the ASMIN protocol. Firstly, C_A receives the input values E(u) = E(3) and E(v) = E(7). Secondly, C_A generates the random numbers 6 and 27, and multiplies E(u) = E(3) and E(v) = E(7) by the plaintext exponentiation of 6. Due to the homomorphic property of the Paillier encryption system (Equations (1) and (2)), C_A can calculate E(3 $\times$ 6 + 27) = E(45) and E(7 $\times$ 6 + 27) = E(69). Thirdly, C_A performs the permutation function (i.e., π) and sends the results E(a) = E(69) and E(b) = E(45) to C_B. Fourthly, C_B decrypts E(a) and E(b). Fifthly, CB stores E(1) for the smaller value and E(0) for the larger value, and returns the results to C_A. In this example, since 69 is not smaller than 45, C_B returns E(a) = E(0) and E(b) = E(1) to C_A. Sixthly, C_A performs the inverse permutation function (i.e., π^{-1}), resulting in E(a) = E(1) and E(b) = E(0). Seventhly, C_A performs SM(E(u), E(a)), and SM(E(v), E(b)), then adds the results using the Paillier addition operation, returning E(3) as the result of the ASMIN protocol. Through this process, the ASMIN protocol safely returns the smaller encrypted value.

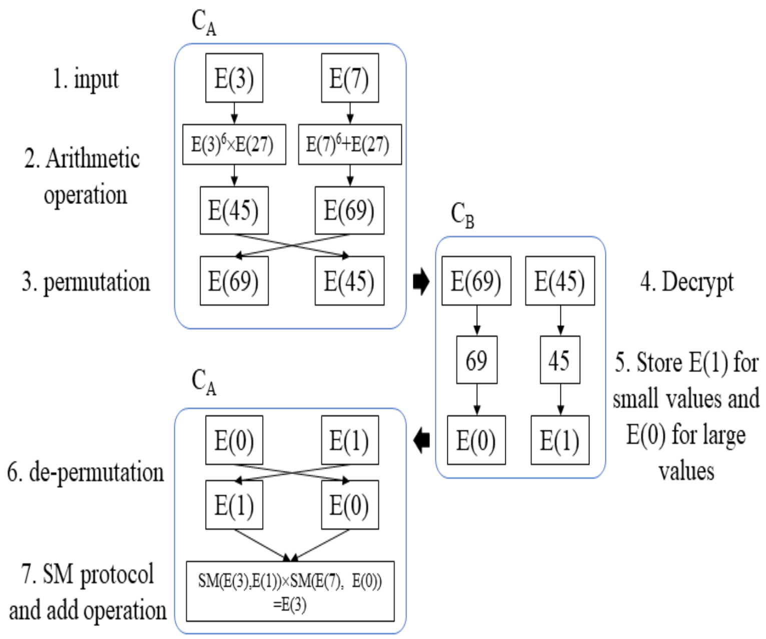

Figure 4. Example of the ASMIN protocol.

3.2.2. Advanced Secure MINimum out of n Numbers ($ASMIN_n$) Protocol

$ASMIN_n$ is a protocol that, given n encrypted values, returns the smallest value to C_A. The ASMINn protocol is based on the previously described ASMIN protocol. The execution process of the $ASMIN_n$ protocol is outlined in Algorithm 2.

Algorithm 2 ASMINn (Advanced Secure MINimum out of n Numbers)

Input: $E(d_1), \ldots, E(d_n)$
Output: $E(d_{min})$
C_A:
$E(d'_i) \leftarrow E(d_i)$ (for $1 \le i \le n$) and num $\leftarrow$ n
for $1 \le i \le \lceil \log_2 n \rceil$
  for $1 \le j \le \lfloor num/2 \rfloor$
    if i = 1 then
      left $\leftarrow 2 \times j - 1$; right = $2 \times j$
    else
      left $\leftarrow 2i(j - 1) + 1$; right = $2ij - 1$
    $E(d'_{left}) \leftarrow ASMIN(E(d'_{left}), E(d'_{right}))$
    num $\leftarrow \lceil num/2 \rceil$
return $E(d_{min}) \leftarrow E(d'_I)$
End Algorithm

---

4. Privacy-Preserving k-Means Clustering Algorithm

In this section, we propose a privacy-preserving k-means clustering algorithm. The proposed k-means clustering algorithm is performed in two stages: the preprocessing phase and the k-means clustering phase.

4.1. Preprocessing Phase

In the k-means clustering algorithm, selecting the initial centroids is crucial as it influences the number of iterations required for the algorithm. However, F. Rao et al.'s study [25] suffers from the drawback of significant time disparities in processing the clustering algorithm due to the random initialization of initial centroids. Therefore, the proposed k-means clustering algorithm addresses this issue by performing a preprocessing step to set the initial centroids. The execution steps of the proposed preprocessing phase are outlined in Algorithm 3.

1. Calculate the number of data points (i.e., cnt) to be selected from each node based on the sampling ratio (line 1).
2. Initialize the initial centroids for the preprocessing step (lines 2–5). Here, the Paillier cryptosystem cannot encrypt real numbers, so the centroids (i.e., initial_center) are represented by the sum of centroids (i.e., initial_center.sum) and the count of centroids (i.e., initial_center.cnt).
3. Select and store the sampled data (i.e., E(sample_data)) from each node, amounting the data points to *cnt* (line 7–9).
4. Calculate the distances between E(sample_data) and E(initial_center) using the $SSED_{op}$ protocol and find the minimum distance using the $ASMIN_n$ protocol (lines 10–13). The $SSED_{op}$ protocol is a distance calculation protocol that was proposed by F. Rao et al. [23] for determining distances between centroids and data points.
5. Calculate the difference between E(dist_min) and each encrypted distance, perform random insertion and permutation, and transmit the result to C_B (lines 14–18).
6. C_B decrypts each element of the received $E(\beta)$, sets E(Ui) = E(1) if $\beta = 0$, and sets E(Ui) = E(0) otherwise. C_B then sends E(U) to C_A (lines 19–22).
7. C_A reverses E(U) and stores it in E(V) (line 23).
8. Perform the SM protocol for $E(V_j)$ and $E(sample\_data_{i,l})$, summing the results for each dimension, and add the sample data with a distance of E(dist_min) to the new centroid E(new_center) = <E(new_center.sum), E(new_center.cnt) > (lines 24–28).

9. Use the SETC protocol to calculate the termination condition (i.e., α) between the initial and new centroids. Then, store new_center in initial_center, and if α is 1, return initial_center; otherwise, repeat the clustering protocol from line 10 (lines 29–34). Here, the Secure Minimum out of n Numbers (SETC) protocol, proposed in [25], checks the termination condition of the k-means clustering algorithm. It returns 1 if the difference between the previous and new centroids is less than the threshold provided by the user, and 0 otherwise.

Algorithm 3 Encrypted initial center selection for k-means clustering

Input: E(data), E(node), E(threshold)
Output: E(initial_center) = <E(initial_center1), . . ., E(initial_center$_k$)>, E(initial_center$_i$) = <E(initial_center$_i$.sum), E(initial_center$_i$.cnt)>, E(initial_center$_i$.sum) = <E(initial_center$_i$.sum1), . . ., E(initial_center$_i$.summ)>

C_A:
01. cnt = Fanout/sampling rate//Fanout is #_of data in each node
02. for $1 \le i \le k$
03. E(initial_center$_i$.cnt) $\leftarrow$ E(1)
04. for $1 \le j \le m$
05. generate random number r
06. E(initial_center$_i$.sum$_j$) $\leftarrow$ E(r)
07. for $1 \le i \le$ NumNode
08. for $1 \le j \le$ cnt
09. E(sample_data) $\leftarrow$ E(node$_i$.data$_j$)
10. for $1 \le i \le$ cnt
11. for $1 \le j \le k$
12. E(dist$_j$) = SSED$_{op}$(E(sample_data$_i$), E(initial_center$_i$))
13. E(dist_min) $\leftarrow$ ASMIN$_n$(E(dist$_1$), . . ., E(dist$_k$))
14. for $1 \le i \le k$
15. E(δ_i) $\leftarrow$ E(dist_min) $\times$ E(dist$_i$)N – 1
16. E(δ'_i) $\leftarrow$ E(δ_i)r$_i$
17. E(β) $\leftarrow$ π(E(δ'_i))
18. send E(β) to C_B

C_B:
19. for $1 \le i \le$ cnt
20. if D(E(β_i)) = 0 then E(U$_i$) $\leftarrow$ E(1)
21. else E(U$_i$) $\leftarrow$ E(0)
22. send E(U) to C_A

C_A:
23. E(V) $\leftarrow$ π−1(E(U))
24. for $1 \le j \le k$
25. for $1 \le l \le m$
26. E(V'$_{j,l}$)$\leftarrow$SM(E(V$_j$), E(sample_data$_{i,l}$))
27. E(new_center$_j$.sum$_l$) $\leftarrow$ E(new_center$_j$.sum$_l$) $\times$ E(V'^{j,l})
//E(new_center) is the same structure with E(initial_center)
28. E(new_center$_j$.cnt) $\leftarrow$ E(new_center$_j$.cnt) $\times$ E(V$_j$)
29. α = SETC(E(initial_center), E(new_center), E(threshold))
30. E(initial_center) $\leftarrow$ E(new_center)
31. if α = 1
32. return E(initial_center)
33. else
34. go to line 10 in Algorithm 3

Figure 5 shows the example of the preprocessing phase with k = 2. The top-left corner of Figure 5 assumes a sampling rate of 25% with 16 data points. First, C_A samples data and sets two of the sampled data points as centroids. In Figure 5, the sampled data are represented as E([sample]) = [E(d_1), E(d_5), E(d_9), E(d_{13})], and the centroids are E([c]) = [E(c_1) = E(d_1), E(c_2) = E(d_5)].

Secondly, C_A calculates the distance between the initial centroids E([c]) and E([sample]) using the SSED protocol. In Figure 5, the distances between $sample_1$ and the two centroids are calculated as $E(dist_{1,1})$ = SSED($E(sample_1)$, $E(c_1)$) = E(0), and $E(dist_{1,2})$ = SSED[($E(sample_1)$, $E(c_2)$) = E(40). Thus, applying the SSED protocol to the sample data yields E([dist]) = [[E(0), E(40)], [E(40), E(0)], [E(16), E(40)], [E(72), E(16)]].

Thirdly, C_A performs the $ASMIN_n$ protocol on all computed distances E[dist] to find the smallest value, subtracts it from E([dist]), and sends the result to C_B. In Figure 5, between $E(dist_{1,1})$ and $E(dist_{1,2})$, the smallest distance is 0, so the $ASMIN_n$ protocol results in E(0), and E([dist-min]) = [E(0), E(40)]. Applying the same process to all sampled data, E([dist-min]) = [[E(0), E(40)], [E(40), E(0)], [E(0), E(24)], [E(56), E(0)]].

Fourthly, C_B decrypts each element of E([dist-min]), saves $E(U_i)$ = E(1) if the value is 0, saves $E(U_i)$ = E(0) otherwise, and sends E(U) to C_A. Therefore, E([U]) = [[E(1), E(0)], [E(0), E(1)], [E(0), E(0)], [E(0), E(0)]].

Fifthly, C_A executes the SM protocol on E([U]) and E([sample]), combines the results for each of the sampled data points, and adds the data point that minimizes the distance to the new centroid E([NC]). In Figure 5, SM(E([U]), E([sample])) = [[$E(sample_1)$, E(0)], [E(0), $E(sample_2)$], [$E(sample_3)$, E(0)], [E(0), $E(sample_4)$]], resulting in E([NC.cnt]) = [E(2), E(2)], and E([NC.sum]) = [[$E(sample_1)$, $E(sample_3)$], [$E(sample_2)$, $E(sample_4)$]].

Finally, using the SETC protocol, C_A calculates the difference between the initial centroids and the new centroid E([NC]). If the difference is less than the threshold, C_A stores E([NC]) in E([IC]) and returns E([IC]); otherwise, steps 2 to 5 are repeated to recompute the clustering. The final centroids computed through this process are shown in the bottom-left corner of Figure 5.

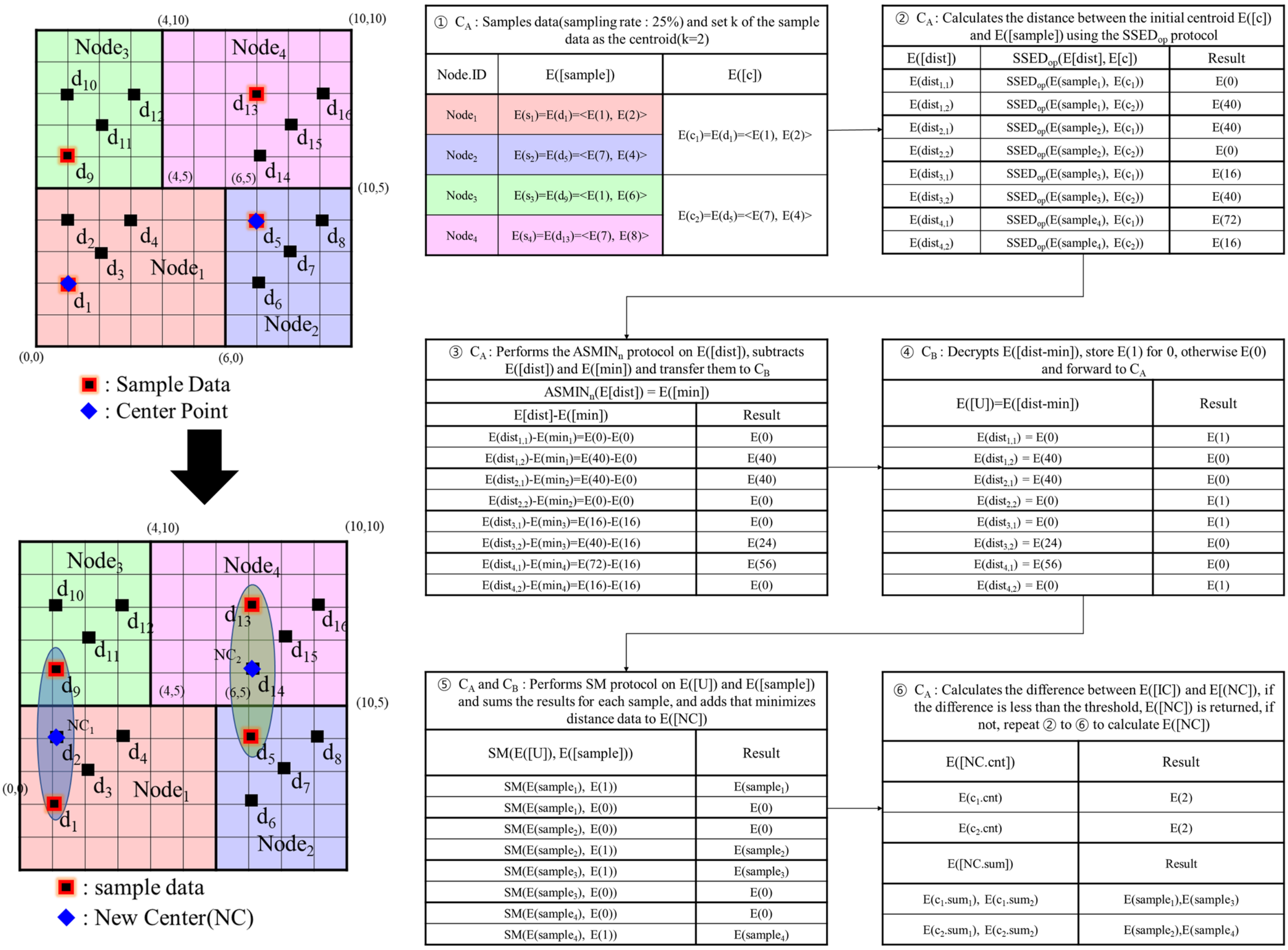

Figure 5. Example of the preprocessing phase (k = 2).

4.2. k-Means Clustering Phase

The k-means clustering phase involves exploring the centroids of the entire dataset using the initial centroids calculated in the preprocessing phase. The execution steps of this phase are described in Algorithm 4.

1. Calculate the distances between E(data) and E(initial_center) using the $SSED_{op}$ protocol and determine the smallest distance (i.e., E(dist_min)) through the proposed $ASMIN_n$ protocol (lines 1–4).
2. C_A calculates the difference between E(dist_min) and each encrypted distance, performs random insertion and permutation, and transmits the result to C_B (lines 5–9).
3. C_B decrypts each element of the received E(β), sets $E(U_i) = E(1)$ if β = 0, and sets $E(U_i) = E(0)$ otherwise. C_B then sends E(U) to C_A (lines 10–13).
4. C_A reverses E(U) and stores it in E(V) (line 14).
5. Perform the SM protocol for $E(V_j)$ and $E(data_{i,l})$, summing the results for each dimension, and add the data with a distance of E(dist_min) to the new centroid E(new_center) (lines 15–19).

Use the SETC protocol to calculate the termination condition (i.e., α) between the initial and new centroids. Then, store new_center in initial_center, and if α is 1, return initial_center to the user; otherwise, repeat the clustering algorithm from line 1 (lines 20–36).

Figure 6 shows the example of the k-means clustering phase (k = 2). Firstly, C_A calculates the distance between the initial centroids E([c]) and E([sample]) using the SSED protocol. In Figure 6, the distances between d_1 and the two centroids are computed as $E(dist_{1,1}) = SSED(E(d_1),E(c_1)) = E(4)$, and $E(dist_{1,2}) = SSED[(E(d_1),E(c_2)) = E(50)$. Applying the SSED protocol to all 16 data points yields E([dist]) = [[E(4),E(50)], [E(0),E(40)], [E(2),E(34)], [E(4),E(20)], [E(36),E(4)], [E(40),E(16)], [E(50),E(10)], [E(64),E(8)], [E(4),E(36)], [E(16),E(40)], [E(10),E(26)], [E(20),E(20)], [E(4),E(50)], [E(40),E(0)], [E(58),E(2)], [E(80),E(8)]]. In Figure 6 ①, due to space limitations, $E(dist_{3,1})$, $E(dist_{3,2})$ to $E(dist_{15,1})$ $E(dist_{15,2})$ are omitted.

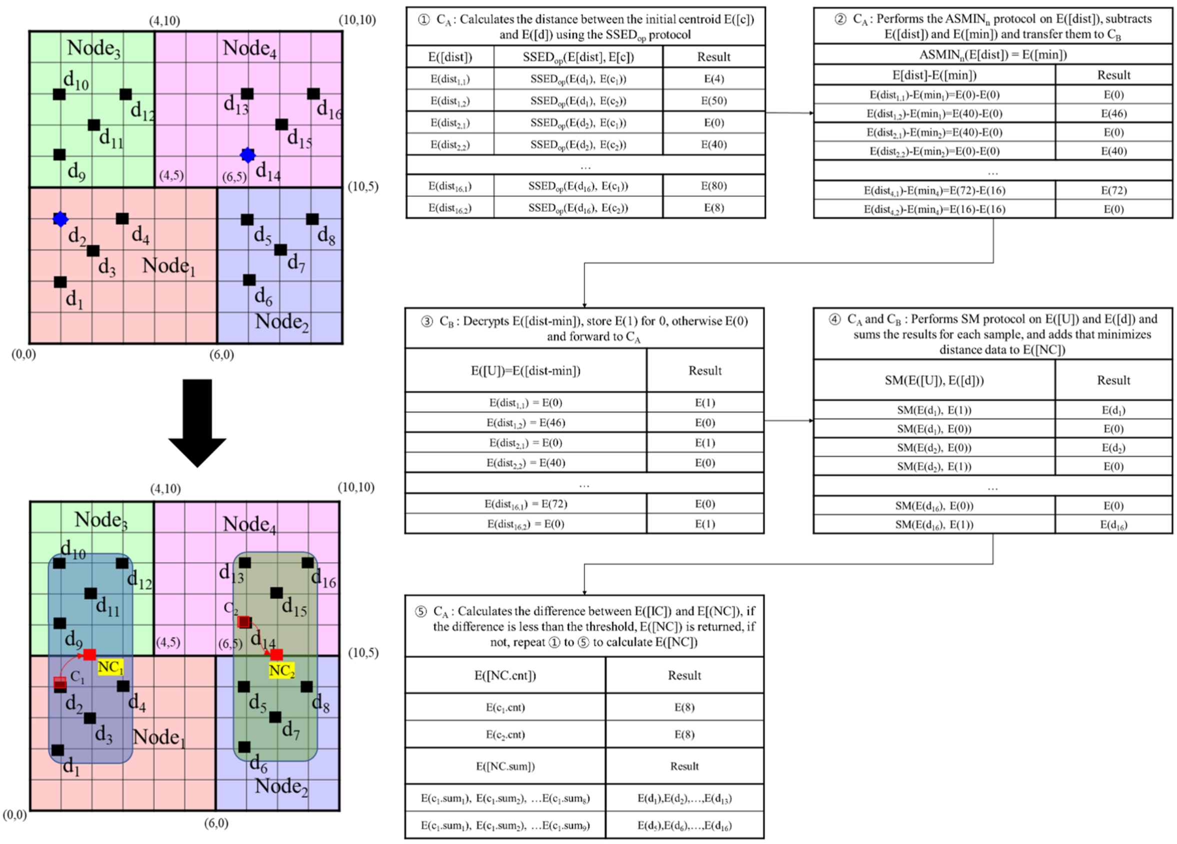

Figure 6. Example of the k-means clustering phase (k = 2).

Algorithm 4 Encrypted center search phase for k-means clustering

Input: E(data), E(node), E(threshold), E(initial_center)
Output: cluster_center
C_A:
for $1 \leq i \leq n$
  for $1 \leq j \leq k$
    $E(dist_j) = SSED_{OP}(E(data_i), E(initial\_center))$
  $E(dist\_min) \leftarrow ASMIN_n(E(dist_1), \ldots, E(dist_k))$
  for $1 \leq i \leq k$
    $E(\delta_i) \leftarrow E(dist\_min) \times E(dist_i)N - 1$
    $E(\delta'_i) \leftarrow E(\delta_i)r_i$
  $E(\beta) \leftarrow \pi(E(\delta'_i))$
  send $E(\beta)$ to C_B
C_B:
  for $1 \leq i \leq cnt$
    if $D(E(\beta_i)) = 0$ then $E(U_i) \leftarrow E(1)$
    else $E(U_i) \leftarrow E(0)$
  send E(U) to CA
C_A:
  $E(V) \leftarrow \pi - 1(E(U))$
  for $1 \leq j \leq k$
    for $1 \leq l \leq m$
      $E(V'_{j,l}) \leftarrow SM(E(V_j), E(data_{i,l}))$
      $E(new\_center_j.sum_l) \leftarrow E(new\_center_j.sum_l) \times E(V'_{j,l})$
    $E(new\_center_j.cnt) \leftarrow E(new\_center_j.cnt) \times E(V_j)$
$\alpha = SETC(E(initial\_center), E(new\_center), E(threshold))$
if $\alpha = 1$
  for $1 \leq i \leq k$
    for $1 \leq j \leq m$
      $E(new\_center_i.sum_j + r) \leftarrow E(new\_center_i.sum_j) \times E(r)$
  $E(new\_center_i.cnt + r) \leftarrow E(new\_center_i.cnt) \times E(r)$
else
  $E(initial\_center) \leftarrow E(new\_center)$
  go to line 1 in Algorithm 4
send r to AU and send E(new_center + r) to C_B
C_B:
  for $1 \leq i \leq k$
    for $1 \leq j \leq m$
      $new\_center_i.sum_j + r \leftarrow D(E(new\_centeri.sumj + r))$
    $new\_centeri.cnt + r \leftarrow D(E(new\_centeri.cnt + r))$
send new_center + r to AU
AU:
receive new_center + r from CB and r from CA
cluster_center = new_center.sum/new_center.cnt

Secondly, C_A performs the $ASMIN_n$ protocol on all distances E[dist] to find the smallest value, subtracts it from E([dist]), and sends the result to C_B. In Figure 6, for d_1, the smallest distances to c_1 and c_2 are 4 and 50, so the $ASMIN_n$ protocol computes E(4) and E(50), respectively. So, E([dist-min]) = [E(0),E(46)]. Applying the same process to all 16 data points yields E([dist-min]) = [[E(0),E(46)], [E(0),E(40)], [E(0),E(34)], [E(0),E(16)], [E(32),E(0)], [E(24),E(0)], [E(40),E(0)], [E(56),E(0)], [E(32),E(0)], [E(0),E(24)], [E(0),E(16)], [E(0),E(0)], [E(0),E(46)], [E(40),E(0)], [E(56),E(0)], [E(72),E(0)]].

Thirdly, C_B decrypts each element of E([dist-min]), saves $E(U_i) = E(1)$ if the value is 0, saves $E(U_i) = E(0)$ otherwise, and sends E(U) to C_A. Therefore, E([U]) = [[E(1),E(0)], [E(1),E(0)], [E(1),E(0)], [E(1),E(0)], [E(0),E(1)], [E(0),E(1)], [E(0),E(1)], [E(0),E(1)], [E(1),E(0)], [E(1),E(0)], [E(1),E(0)], [E(1),E(1)], [E(1),E(0)], [E(0),E(1)], [E(0),E(1)], [E(0),E(1)]].

Fourthly, C_A performs the SM protocol on E([U]) and E([d]), combines the results for each data point, and adds the data point that minimizes the distance to the new centroid E([NC]). In Figure 6, SM(E([U]), E([d])) = [[E(d_1),E(0)], [E(d_2),E(0)], [E(d_3),E(0)], [E(d_4),E(0)], [E(0),E(d_5)], [E(0),E(d_6)], [E(0),E(d_7)], [E(0),E(d_8)], [E(d_9),E(0)], [E(d_{10}),E(0)], [E(d_{11}),E(0)], [E(d_{12}),E(d_{12})], [E(d_{13}),E(0)], [E(0),E(d_{14})], [E(0),E(d_{15})], [E(0),E(d_{16})]] resulting in E([NC.cnt]) = [E(8),E(9)], and E([NC.sum]) = [[E(d_1),E(d_2),E(d_3),E(d_4),E(d_{10}),E(d_{11}),E(d_{12}), E(d_{13})], [E(d_5),E(d_6),E(d_7),E(d_8), E(d_9), E(d_{12}), E(d_{14}), E(d_{15}), E(d_{16})]].

Finally, using the SETC protocol, C_A calculates the difference between the initial centroids and the new centroid E([NC]). If the difference is less than the threshold, C_A stores E([NC]) in E([IC]) and returns E([IC]); otherwise, steps 1 to 5 are repeated to recompute the clustering. The final k-means clusters computed through this process are depicted in the bottom-left corner of Figure 6.

5. Privacy-Preserving Parallel k-Means Clustering Algorithm

In this section, we propose a privacy-preserving parallel k-means clustering algorithm. To perform the parallel processing of the k-means clustering algorithm, we utilize a thread pool to prevent data bottlenecks and parallelize encryption operation protocols for efficient support of k-means clustering. However, when parallelizing homomorphic encryption techniques, there is the issue of the increased operational overhead per core, leading to bottlenecks and performance degradation. To mitigate this bottleneck in parallel processing, we reduce the computational cost during k-means clustering by preprocessing operations that generate random values for hiding data and encrypting them in the encryption operation protocol. This random value pool generates random numbers (i.e., integers) in the form of plaintext and ciphertext pairs <r, E(r)> before starting k-means clustering, storing them in a queue-like memory space. When needed, the data are extracted and used in a first-in-first-out (FIFO) manner.

The Paillier encryption system used in this paper consists of computationally expensive operations such as the exponentiation function (i.e., exp) and modular function (i.e., mod). These operations impose a significant computational overhead on the CPU, leading to CPU bottlenecks and performance degradation. This problem is exacerbated when memory access is frequent in parallel processing algorithms. In [17], the SM protocol generates and encrypts random numbers before performing homomorphic addition operations. Encrypting random numbers incurs a cost of two encryption operations, utilizing the CPU's computational resources. Additionally, the SM protocol, which performs multiplication as an encryption operation, is used within the ASMIN encryption operation protocol. Generating and encrypting random numbers during k-means clustering leads to high computational costs. Therefore, this paper addresses this issue by storing the generated random numbers (i.e., r) and their encrypted counterparts (i.e., E(r)) in the random value pool's memory space. When needed, the cryptographic authority (CA) can select <r, E(r)> pairs from the random value pool, enabling their use at a lower memory load cost. The reduced computational cost for each protocol using the random value pool is summarized in Table 2.

Table 2. Computational cost without/with a random value pool.

Secure Protocol	Computational Cost without a Random Value Pool	Computational Cost with a Random Value Pool
SM protocol	$3 \times E$	$1 \times E$
ASMIN protocol	$10 \times E$	$4 \times E$

E = encryption.

The proposed parallel k-means clustering algorithm is executed in two phases: parallel preprocessing and parallel k-means clustering.

5.1. Parallel Preprocessing Phase

In the preprocessing phase of the privacy-preserving k-means clustering algorithm that supports information protection, a parallel processing technique utilizing a thread pool is proposed. The execution process of the parallel preprocessing phase is outlined in Algorithm 5.

1. C_A calculates the number of data points to be selected from each node based on the sampling ratio (*cnt*) (line 1).
2. C_A initializes the initial center of the preprocessing phase (lines 2~5). In this case, as the Paillier encryption system cannot encrypt real numbers, the center point (initial_center) is represented by the sum of center points (initial_center.sum) and the count of center points (initial_center.cnt).
3. Each node selects and stores *cnt* samples of data (i.e., E(sample_data)) (lines 7~9).
4. The thread pool performs parallel processing of the calculate_new_center procedure. The calculate_new_center procedure involves the following steps. First, each thread on C_A calculates the distance between E(sample_data) and E(initial_center) using the $SSED_{op}$ protocol. Then, it determines the smallest distance using the $ASMIN_n$ protocol. Second, each thread on C_A calculates the difference between E(dist_min) and each encrypted distance. After inserting random numbers and changing the order, this information is sent to C_B. Third, each thread on C_B decrypts the received E(β), setting $E(U_i)$ to E(1) if β = 0; otherwise, $E(U_i)$ is set to E(0). C_B then sends E(U) back to C_A. Fourth, each thread on C_A reverses E(U) and stores it in E(V). Fifth, each thread on C_A performs the SM protocol on $E(V_j)$ and $E(sample\_data_{i,l})$. By summing the results for each dimension, it adds the sample data with a distance of E(dist_min) to the new center point E(new_center) = <E(new_center.sum), E(new_center.cnt)>. Sixth, the thread pool allows for the parallel computation of the new center point E(new_center) through the five steps mentioned above.
5. The SETC protocol calculates the termination condition (α) between the initial center point and the new center point. Subsequently, new_center is stored in initial_center. If α is 1, initial_center is returned; otherwise, clustering is re-executed from line 10 (lines 12~17).

Algorithm 5 Parallel encrypted initial center selection for k-means clustering

Input: E(data), E(node), E(threshold)
Output: E(initial_center) = <E($initial\_center_1$), . . ., E($initial\_center_k$)>
//E($initial\_center_i$) = <E(initial_centeri.sum), E($initial\_center_i$.cnt)>, E($initial\_center_i$.sum) = <E($initial\_center_i.sum_1$), . . ., E($initial\_center_i.sum_m$)>
CA:
01. cnt = node.cnt/sampling rate
02. for $1 \le i \le k$
03. E($initial\_center_i$.cnt) ← E(1)
04. for $1 \le j \le m$
05. select random number r from the random value pool
06. E($initial\_center_i.sum_j$) ← E(r)
07. for $1 \le i \le$ NumNode
08. for $1 \le j \le$ cnt
09. E(sample_data) ← E($node_i.data_j$)
10. for $1 \le i \le$ cnt
11. thread_pool_push(**calculate_new_center**(k, m, cnt, E(sample_data), E(initial_center), E(new_center)))
12. α = SETC(E(initial_center), E(new_center), E(threshold))
13. E(initial_center) ← E(new_center)
14. if α = 1
15. return E(initial_center)
16. else
17. go to line 10 in Algorithm 5

Algorithm 5 *Cont.*

Procedure 1. calculate_new_center($E(data)$, $E(node_i)$, $E(dist_k)$, $E(\delta_i)$)
Begin Procedure
C_A:
  for $1 \le j \le k$
    $E(dist_j) = SSED_{op}(E(sample\_data_i), E(initial\_center_i))$
  $E(dist\_min) \leftarrow ASMIN_n(E(dist_1), \ldots, E(dist_k))$
  for $1 \le i \le k$
    $E(\delta_i) \leftarrow E(dist\_min) \times E(dist_i)N - 1$
    $E(\delta'_i) \leftarrow E(\delta_i)r_i$
  $E(\beta) \leftarrow \pi(E(\delta'_i))$
  send $E(\beta)$ to C_B
C_B:
  for $1 \le i \le cnt$
    if $D(E(\beta_i)) = 0$ then $E(U_i) \leftarrow E(1)$
    else $E(U_i) \leftarrow E(0)$
  send $E(U)$ to C_A
C_A:
  $E(V) \leftarrow \pi - 1(E(U))$
  for $1 \le j \le k$
    for $1 \le l \le m$
      $E(V'_{j,l}) \leftarrow SM(E(V_j), E(sample\_data_{i,l}))$
      $E(new\_center_j.sum_l) \leftarrow E(new\_center_j.sum_l) \times E(V'_{j,l})$
    $E(new\_center_j.cnt) \leftarrow E(new\_center_j.cnt) \times E(V_j)$
  return $E(\delta_i)$
End Procedure

5.2. Parallel k-Means Clustering Phase

In the parallel k-means clustering phase in the parallel k-means clustering algorithm, a parallel processing technique using a thread pool is proposed. The execution process of the parallel k-means clustering phase is outlined in Algorithm 6.

1. The thread pool performs parallel processing of the calculate_new_center procedure. The calculate_new_center procedure is the same as Procedure 1 in Algorithm 5.
2. The SETC protocol calculates the termination condition (α) between the initial center point and the new center point (line 3).
3. If the result of the SETC protocol (i.e., α) is 0, re-execute from line 1.
4. If the result of the SETC protocol (i.e., α) is 1, C_B adds a random number (i.e., r) to E(new_center) and sends it to the AU. Simultaneously, the random number r is sent to C_B by the AU.
5. C_B decrypts E(new_center + r) and forwards it to the AU. Authenticated users perform the subtraction of r from new_center + r received from C_B, obtaining the final k-means clustering result (lines 4~19).

Algorithm 6 Parallel encrypted center search phase for k-means clustering

Input: E(data), E(node), E(threshold), E(initial_center)
Output: cluster_center
C_A:
01. for $1 \leq i \leq n$
02. thread_pool_push(**calculate_new_center**(k, m, cnt, E(data), E(initial_center), E(new_center)))//calculate_new_center() is the Procedure 1 in Algorithm 5
03. α = SETC(E(initial_center), E(new_center), E(threshold))
04. if α = 1
05. for $1 \leq i \leq k$
06. for $1 \leq j \leq m$
07. E(new_center$_i$.sum$_j$ + r)←E(new_center$_i$.sum$_j$) × E(r)
08. E(new_center$_i$.cnt + r)←E(new_center$_i$.cnt)×E(r)
09. else
10. E(initial_center)←E(new_center)
11. go to line 1 in Algorithm 6
12. send r to the AU and send E(new_center + r) to C_B
C_B:
13. for $1 \leq i \leq k$
14. for $1 \leq j \leq m$
15. new_centeri.sum$_j$ + r ← D(E(new_center$_i$.sum$_j$+r))
16. new_center$_i$.cnt + r ← D(E(new_center$_i$.cnt+r))
17. send new_center + r to the AU
AU:
18. receive new_center + r from CB and r from CA
19. cluster_center = new_center.sum/new_center.cnt

6. Security Analysis

6.1. Security Analysis of Security Protocols

(1) ASMIN Protocol

In this section, the proposed ASMIN protocol is proven to be secure in the semi-honest attack model. To conduct a security analysis of the protocol, execution images based on input data were generated. The information obtainable by C_A and C_B during the execution of the ASMIN protocol is summarized in Table 3. Here, the information about r_1 and r_2 is safe from exposure since they are random variables.

Table 3. Information obtainable in the ASMIN protocol.

	Execution Image of the ASMIN Protocol $= \prod_C(P(E(\alpha)), E(\beta))$	**Simulation Image** $\prod_C(P(E(u)), E(v))$
C_A	Input Data: $E(\alpha)$, $E(\beta)$ Generated Variables: r_1, r_2, Output Data: $E(min)$	Input Data: $E(u)$, $E(v)$ Generated Variables: r_1, r_2, Output Data: $E(min')$
C_B	Input Data: X, Y Output Data: $E(r_1)$, $E(r_2)$	Input Data: x, y Output Data: $E(w_1)$, $E(w_2)$

First, the simulation image of the ASMIN protocol on the C_A side is given by Equation (4). The indistinguishability of the simulation images of $E(u)$, $E(v)$, $E(min')$ and the execution images of $E(\alpha)$, $E(\beta)$, $E(min)$ can be proven through Equations (5) and (6). Since the Paillier encryption system generates random numbers during ciphertext creation, attackers cannot identify the execution image from the simulation image.

$$\prod_{C_A^s} (ASC(E(u),\ E(v))) = \{E(u),\ E(v),\ r_1, R_2, (E(\gamma)\} \tag{4}$$

$$E(u) = g^u r^N \ mod\ N^2 \ \neq E(\alpha) = g^\alpha r^N \ mod\ N^2,\ where\ 0 < r < N \tag{5}$$

$$E(min') = g^{min'} r^N \ mod \ N^2 \neq E(\min) = g^{min} r^N \ mod \ N^2, \ where \ 0 < r < N \quad (6)$$

On the C_B side, the simulation image of the ASMIN protocol is given by Equation (7). The indistinguishability of the simulation images of x, y, $E(w_1)$, $E(w_2)$ and the execution images of X, Y, $E(\gamma_1)$, $E(\gamma_2)$ can be proven through Equations (8) and (9). The variable x in the simulation image involves the addition and multiplication of two random numbers, making the probability of identifying the execution image of X be $1/(x-1)$. Since the minimum value of the random numbers is 3, the probability $1/(x-1)$ is less than $1/2$, making identification impossible. $E(w_1)$, $E(w_2)$, $E(\gamma_1)$, and $E(\gamma_2)$ involve random numbers generated during ciphertext creation in the Paillier encryption system, making it impossible for the attacker to distinguish between the execution and simulation images.

$$\prod\nolimits_{C_A^s} (ASMIN(x, \ y)) = \{x, \ y, \ (E(w_1), \ E(w_2)\} \quad (7)$$

$$\begin{cases} x = ur_1 + r_2 \neq X = \alpha r_1' + r_2'\prime \ where \ 3 < r_1, r_2, r_1'\prime \ r_2' < N \\ P(The \ probability \ of \ identifying \ X \ through \ x) = \frac{1}{x-1} \leq \frac{1}{2}, \ where \ x > 3 \end{cases} \quad (8)$$

$$\begin{cases} E(w_1) = g^{w_1} r^N mod \ N^2 \neq E(\gamma) = g^{\gamma_1} r^N \ mod \ N^2, \ where \ 0 < r < N \\ E(w_2) = g^{w_2} r^N mod \ N^2 \neq E(\gamma) = g^{\gamma_2} r^N \ mod \ N^2, \ where \ 0 < r < N \end{cases} \quad (9)$$

In summary, the analysis demonstrates that no information is exposed during the execution of the ASMIN protocol on the C_A and C_B sides. Therefore, the ASMIN protocol is secure under the semi-honest attack model.

(2) ASMIN_n Protocol

The part where C_A and C_B exchange data in the ASMIN_n protocol is the execution part of the ASMIN protocol. Therefore, if the ASMIN protocol is secure under the semi-honest attack model, by the composition theory [33], we can conclude that the ASMIN_n protocol is also secure under the semi-honest attack model. Since the security of the ASMIN protocol has been demonstrated earlier, it follows that the ASMIN_n protocol is secure under the semi-honest attack model.

6.2. Security Analysis of the k-Means Clustering Algorithm

To demonstrate the safety of the proposed k-means clustering algorithm in the semi-honest attack model, security analyses were performed for both C_A and C_B. First, the information accessible to C_A is described. The information accessible to C_A includes encrypted data and encrypted queries (Table 4).

Table 4. Information accessible to C_A in the k-means clustering algorithm.

	C_A's Accessible Execution Image	**Simulation Image**
Encrypted Data	E(data)	E(sim_data)
Encrypted Query	E(threshold)	E(sim_threshold)

The encrypted data are the encryption of the original data by the Paillier encryption system. The encryption threshold is the encryption of the user query by the Paillier encryption system. Therefore, through Equation (10), the impossibility of computation between the execution image and the simulation image in the k-means clustering algorithm can be demonstrated.

$$\begin{aligned} &E(threshold) = g^{threshold} r^n \ mod \ N^2 \\ &\neq E(sim\_threshold) = g^{threshold} r^N \ mod \ N^2, \ where \ r < N \end{aligned} \quad (10)$$

Secondly, let us describe the information that C_B can obtain. The information obtainable by C_B includes the encrypted data received from C_A (i.e., E(data)) and the decrypted data (i.e., data) (Table 5).

Table 5. Information obtainable by C_B in the k-means clustering algorithm.

	Information Obtainable by C_B	Simulation Image
Encrypted Data Received from C_A	E(data)	E(sim_data)
Decrypted Data	data	sim_data

The encrypted data received from C_A are identical to the encrypted data in Table 4, making it impossible to perform calculations between the execution image and the simulation image. The decryption data in the simulation image, sim_data, consist of random number additions. Therefore, the probability of identifying the data in the execution image is $\frac{1}{sim\_data}$. Since the minimum value of the random number is 3, the probability $\frac{1}{sim\_data}$ is lower than $\frac{1}{2}$, making identification impossible (Equation (11)).

$$\begin{cases} data = data + r_1 \neq sim\_data + r_2,\ where\ 3 < r_1,\ r_2 < N \\ P(\text{The probability of identifying data through sim\_data}) = \frac{1}{simdata} \leq \frac{1}{2},\ where\ sim\_data > 3 \end{cases} \tag{11}$$

Finally, as demonstrated earlier, each step of the k-means clustering algorithm is secure under the semi-honest attack model. Therefore, the proposed privacy-preserving k-means clustering algorithm is secure under the semi-honest attack model according to the composition theory [33].

6.3. Security Analysis of the Parallel k-Means Clustering Algorithm

The proposed privacy-preserving parallel k-means clustering algorithm consists of two phases: the parallel preprocessing phase (Algorithm 5) and the parallel k-means clustering phase (Algorithm 6). To demonstrate the security of the parallel k-means clustering algorithm in the semi-honest attack model, security analyses were performed for each phase.

Firstly, Algorithm 5 is secure, as it was proven to be a secure version of Algorithm 3, and the execution images on the C_A and C_B sides are identical. The difference between Algorithm 5 and Algorithm 3 lies in the use of a thread pool to parallelize the SM, $SSED_{op}$, and $ASMIN_n$ protocols. Since the SM, $SSED_{op}$, and $ASMIN_n$ protocols were individually proven to be secure, Algorithm 5 is secure in the semi-honest attack model according to the composition theory [33].

Secondly, Algorithm 6 is secure, having been proven to be a secure version of Algorithm 4, and the execution images on the C_A and C_B sides are identical. The difference between Algorithm 6 and Algorithm 4 lies in the use of a thread pool to parallelize the SM, $SSED_{op}$, and $ASMIN_n$ protocols. Given the individual security proofs for the SM, $SSED_{op}$, and $ASMIN_n$ protocols, Algorithm 6 is secure in the semi-honest attack model according to the composition theory [33].

In conclusion, each phase of the privacy-preserving parallel k-means clustering algorithm is secure in the semi-honest attack model. Therefore, the entire parallel k-means clustering algorithm is secure in the semi-honest attack model according to the composition theory [33].

7. Performance Analysis

This section evaluates the performances of both the proposed privacy-preserving k-means clustering algorithm and the proposed parallel k-means clustering algorithm. The evaluations were conducted in the Linux Ubuntu 18.04.2 environment with an Intel(R) Xeon(R) CPU E5-2630 2.20 GHz 10-Core 3.10 GHz processor and 64 GB (16 GB $\times$ 4 AE) DDR3 UDIMM 1600 MHz RAM. The algorithms were implemented in C++ and, to represent the range 0 to 2 key_size in the Paillier encryption system, the GMP library's mpz_t data type was used instead of basic data types. The GMP library supports operations on larger numbers compared with typical numeric data types (e.g., short, int, long).

The performance evaluation focused on the execution time, measured as the difference between the timestamp just before the algorithm's execution and the timestamp just after its completion. The dataset used for the performance evaluation was the synthetic uniform dataset [34] as shown in Table 6.

Table 6. The data used for the performance evaluation.

Synthetic Uniform Data [34] Description: Synthetic Data with a Uniform Distribution	
Number of data points	100,000
Number of columns	6
Data domain	0~512

In this section, we evaluate the performance of both the Secure k-Means Clustering algorithm with Advanced Secure Protocol ($SkMC_A$) proposed in Section 4 and the Parallel Secure k-Means Clustering algorithm with Advanced Secure Protocol ($PSkMC_A$) proposed in Section 5. To measure the performance improvement of $SkMC_A$, we compare it with [25] (i.e., the Secure k-Means Clustering algorithm with Index filtering ($SkMC_I$)), which provides a similar level of information protection (data protection, query protection, and access pattern protection). Additionally, as no parallel k-means clustering algorithm that supports information protection currently exists, we created the parallel processing version of $SkMC_I$ (i.e., $PSkMC_I$) in order to compare it with $PSkMC_A$. The query accuracy of both the existing k-means clustering algorithm and the proposed k-means clustering algorithm is 100%. Table 7 summarizes the comparison targets for the proposed k-means clustering algorithms.

Table 7. Targets for the performance comparison of the proposed k-means clustering algorithm.

Proposed k-Means Clustering Algorithm ($SkMC_A$)	**$SkMC_I$ [25]**
Proposed Parallel k-Means Clustering Algorithm ($PSkMC_A$)	$PSkMC_I$ (Parallel processing version of $SkMC_I$ [25])

The performance evaluation of the k-means clustering algorithm was performed by selecting random data points as queries from the generated data. The performance evaluation compared the query processing times of the proposed algorithms and the existing algorithms with changes in the number of data points (n), k, and threads. The threshold for the k-means clustering algorithm was set to 10. Table 8 lists the parameters considered in the performance evaluation.

Table 8. Parameters considered in the performance evaluation of the k-means clustering algorithms.

Parameter	**Values**	**Default**
# of data points (n)	2k, 4k, 6k, 8k, 10k	10k
k	5, 10, 15, 20	10
# of data dimensions (m)	2	2
Encryption key size (K)	512	512
Bit length	22	22
Threshold	10	-
# of Threads	2, 4, 6, 8, 10	10

Figure 7 illustrates the performance of $SkMC_A$ and $SkMC_I$ with varying total data sizes when k = 10, m = 2, and K = 512. When n = 2k, $SkMC_A$ takes approximately 1807 s, while $SkMC_I$ requires around 14,456 s. The significant performance improvement is attributed to the faster execution time of the proposed ASMIN and $ASMIN_n$ protocols compared with the SMIN and $SMIN_n$ protocols used in $SkMC_I$. Through the enhanced efficiency of the encryption operation protocols, $SkMC_A$ demonstrates approximately 10.8 times better performance than $SkMC_I$.

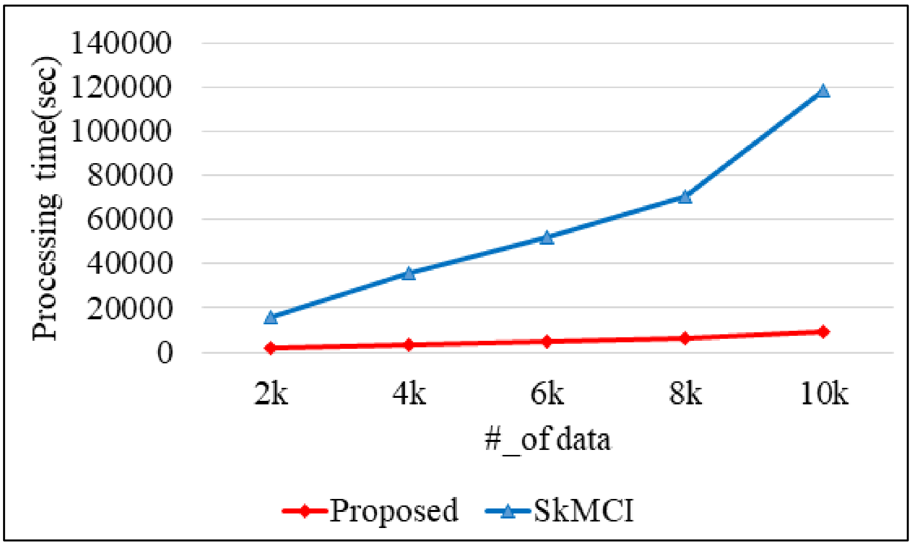

Figure 7. Performance evaluation of k-means clustering algorithms with a changing data size (n).

Figure 8 depicts the performance of $SkMC_A$ and $SkMC_I$ with varying values of k when n = 10k, m = 2, and K = 512. For the case where k = 20, $SkMC_A$ takes approximately 18,791 s, while $SkMC_I$ requires around 236,502 s. The significant performance improvement is attributed to the proposed k-means clustering algorithm, which rapidly performs encryption protocols through decimal-based arithmetic operations. In contrast, $SkMC_I$ uses iterative encryption protocols based on binary arithmetic. $SkMC_A$ demonstrates approximately 12.3 times better performance than $SkMC_I$.

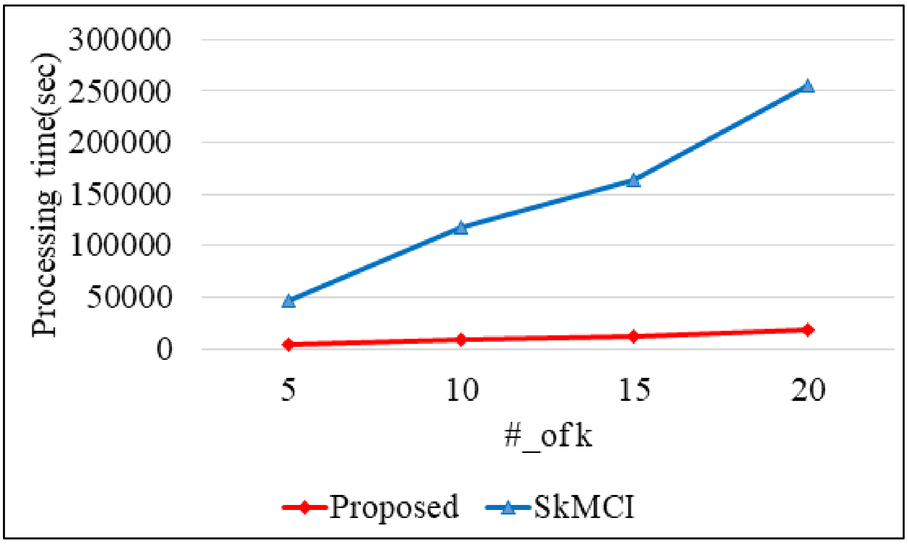

Figure 8. Performance evaluation of k-means clustering algorithms with a changing k value.

Figure 9 illustrates the performance of $PSkMC_A$ and $PSkMC_I$ with varying numbers of threads when n = 10k, k = 10, m = 2, and K = 512. When the number of threads is 2, $PSkMC_A$ takes approximately 4918 s, while $PSkMC_I$ requires around 40,345 s. When the number of threads is 10, $PSkMC_A$ takes about 1287 s, and $PSkMC_I$ takes around 16,446 s. Overall, $PSkMC_A$ demonstrates approximately 13.5 times better performance than $PSkMC_I$.

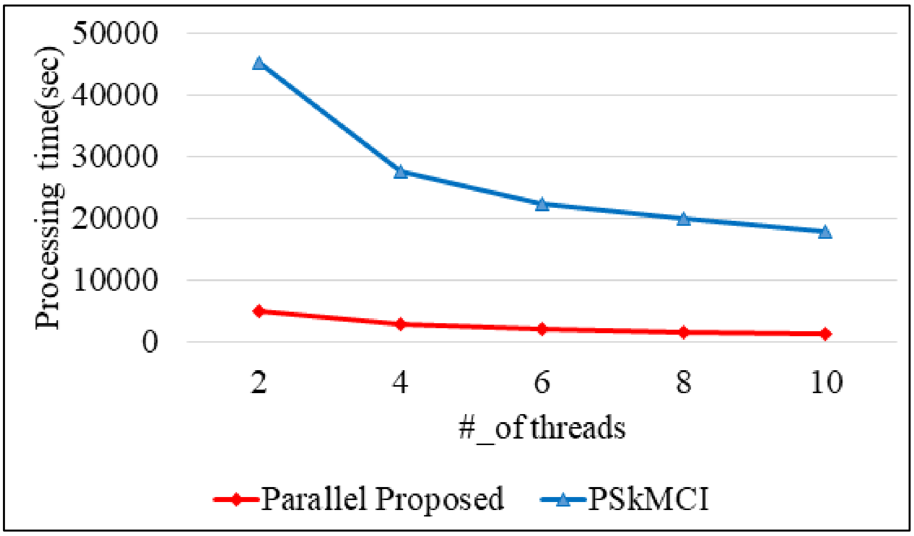

Figure 9. Performance evaluation of k-means clustering algorithms with a changing number of threads.

As shown in Figure 9, the proposed parallel k-means clustering algorithm demonstrates a linear decrease in processing time as the number of threads increases from 2 to 10. In other words, there is no performance degradation as the number of threads increases. This indicates that the proposed parallel k-means clustering algorithm exhibits scalability, where the performance can scale effectively with the number of threads.

Figure 10 illustrates the memory usage of the proposed k-means clustering algorithm and the existing algorithm. When n = 10,000, k = 10, and K = 512, the existing k-means clustering algorithm uses approximately 1000 kilobytes, while the proposed algorithm uses around 1150 kilobytes. Figure 10 shows that the proposed algorithm requires approximately an additional 15% memory usage compared with the existing algorithm. The reason for this is that the proposed algorithm consumes more memory due to the preprocessing step involved in sampling the data. However, the 150 kilobytes in total of memory overhead compared with the existing algorithm is quite small in terms of memory utilization.

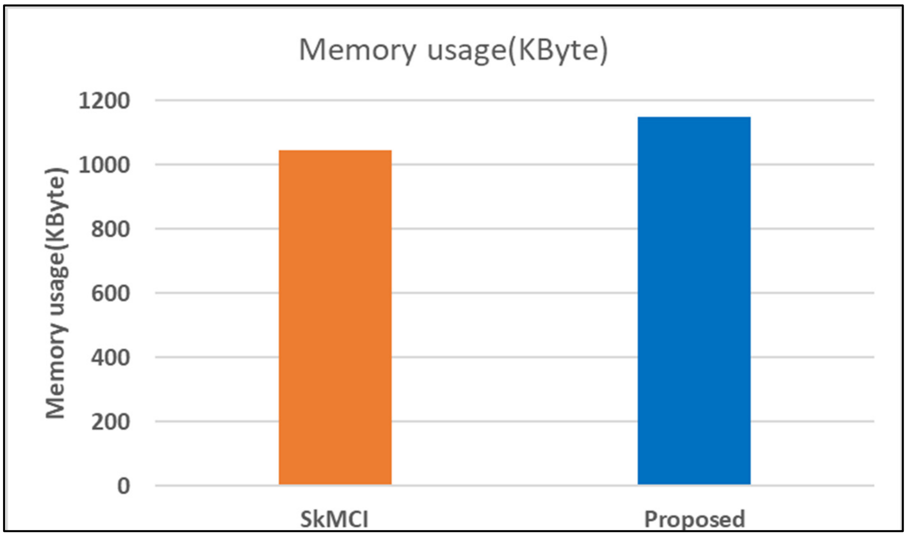

Figure 10. Memory usage of the proposed k-means clustering algorithm and the existing algorithm.

8. Discussion

8.1. The Time Complexity of the Proposed Secure Protocols

To measure the time complexity of the proposed secure protocols, we analyzed the number of homomorphic operations performed in each protocol. For this, let n represent the number of input data points, m the number of dimensions, and l the number of bits. The time complexities of each secure protocol are as shown in Table 9. For the ASMIN protocol, a total of eight homomorphic operations were performed to obtain the result of the comparison. Therefore, the time complexity of the ASMIN protocol is constant time, and can be simplified to O(1). In the case of the $\mathrm{ASMIN_n}$ protocol, it performs ASMIN protocols $\log_2{}^n$ times, so the time complexity of the $\mathrm{ASMIN_n}$ protocol is $O(\log_2{}^n)$. The proposed secure protocols have constant time complexity regardless of the number of bits, while the existing secure protocols exhibit a time complexity that is proportional to the number of bits.

Table 9. Time complexity of each secure protocol.

Encryption Operation Protocol	Function	Time Complexity
ASMIN protocol	MIN operation	$O(8) \approx O(1)$
$\mathrm{ASMIN_n}$ protocol	MIN operation among n inputs	$O(8 \times \log_2{}^n) \approx O(\log_2{}^n)$

8.2. The Time Complexity of the Proposed k-Means Clustering Algorithm

The time complexity of the proposed k-means clustering algorithm ($\mathrm{SkMC_A}$) and that of $\mathrm{SkMC_I}$ are shown in Table 10. The time complexity of $\mathrm{SkMC_A}$ is proportional to the number of data points × k × (dimension + log2k), while that of $\mathrm{SkMC_I}$ is proportional to k × (dimension × number of data + log2k × bit length).

Table 10. Time complexity of the k-means clustering algorithms.

k-Means Clustering Algorithm	Time Complexity
$\mathrm{SkMC_A}$ (proposed)	$O(n \times k \times (\log_2 k + m))$
$\mathrm{SkMC_I}$ [23]	$O(n \times k \times (l \times \log_2 k + m))$

When comparing the time complexity of the proposed k-means clustering algorithm with that of the existing algorithm, the proposed k-means clustering algorithm exhibits a time complexity that is independent of the bit length. In contrast, the existing algorithm shows a time complexity that is proportional to the bit length.

8.3. Theoretical Analysis of the Proposed Algorithm in Terms of Privacy

Assuming that an attacker does not have any information on original data items, an adversary needs a tremendous amount of time to obtain the original plaintext from a Paillier cryptosystem while using a brute force attack. This means that it is impossible to do an experiment to prove that data, queries, and access patterns are all protected. Therefore, instead of an experimental analysis, we conducted a theoretical analysis of data privacy, query privacy, and access pattern privacy to support the security analysis of the proposed algorithm. For this, we estimated the time it takes for the original data to be exposed and calculated the probability of access pattern leakage.

(1) Theoretical analysis of data privacy

In C_A, an attacker only obtains the ciphertext of data. Because the data are protected by the Paillier cryptosystem, the security performance is measured through the time complexity of the brute force attack used to break down the Paillier cryptosystem. Our Paillier cryptosystem uses a 512-bit encryption key size. Assuming that the CPU cycle is 4 GHz, the time required to decrypt the ciphertext by changing the key is as shown in Equation (12).

$$BFAtime(sec) = \frac{2^{512}}{4GHz} \approx \frac{1.3 \times 10^{154}}{4GHz} \quad (12)$$

It is impossible to break down a Paillier cryptosystem because it takes about 4.2×10^{146} years with a 512-bit key size. This means that the proposed privacy-preserving k-means clustering algorithm is secure in terms of data privacy even if the ciphertext is exposed. Figure 11 shows the time taken for a brute force attack in C_A as the key size is changed. In C_B, an attacker only obtains plaintext data that add a random number to the original data. In the Paillier cryptosystem, because the range of the plaintext data is $0 \leq m \leq 2^{512}$, the brute force attack time in C_B is the same as that in C_A.

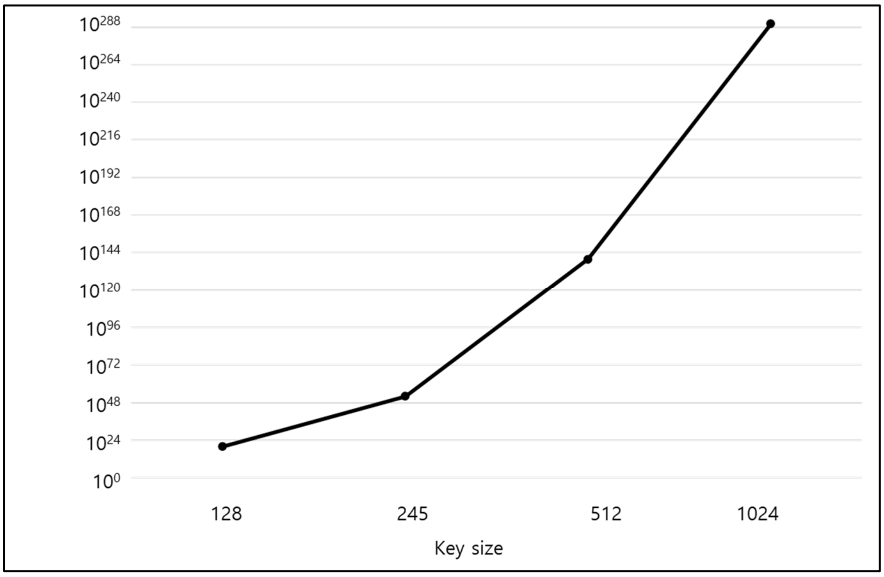

Figure 11. Time taken in a brute force attack on the Paillier encryption system.

(2) Theoretical analysis of query privacy

In C_A, an attacker only obtains the ciphertext of a query. Because the query is protected by the Paillier cryptosystem, the security performance is measured through the time complexity of the brute force attack used to break down the Paillier cryptosystem. Since our Paillier cryptosystem uses a 512-bit encryption key size, the time required to decrypt the ciphertext by changing the key is as shown in Equation (10), where the CPU cycle is 4GHz. It is impossible to break down a Paillier cryptosystem because it takes about 4.2×10^{146} years with a 512-bit key size. This means that the proposed privacy-preserving k-means clustering algorithm is secure in terms of query privacy even if the ciphertext is exposed. The time taken in a brute force attack on C_A is the same as that for data privacy in C_A (Figure 11). In C_B, query privacy is preserved because C_B does not receive the query.

(3) Theoretical analysis of access pattern privacy

An access pattern describes the access sequence for a data item. In the proposed algorithm, the sequence for accessing a data item consists of accessing the leaf node of the kd-tree and accessing the data in the leaf node. In C_A, an attacker only obtains the ciphertext of the leaf node. Because all the leaf nodes have the same number of data items, an attacker cannot distinguish the leaf node by using the density of data items. If the kd-tree level is h, the number of leaf nodes is 2^{h-1}. The probability that an attacker can distinguish a node ($node_i$) from the others, i.e., $P(node_i)$, is $\frac{1}{2^{h-1}}$. Because $node_i$ includes the same number of data items as fanout, the probability that an attacker can distinguish a data item from the others in $\mathrm{node_i}$, i.e., $P(\mathrm{node_i.data_j})$, is $\frac{1}{fanout} = \frac{1}{\frac{the\ number\ of\ data}{2^{h-1}}} = \frac{2^{h-1}}{n}$. Therefore, the probability of data access pattern leakage (P_{APL}) is as shown in Equation (13).

$$P_{APL} = P(node_i) \times P\left(node_i, data_j\right) = \frac{1}{2^{h-1}} \times \frac{2^{h-1}}{n} = \frac{1}{n} \quad (13)$$

P_{APL} is equal to the probability that an attacker distinguishes a specific data item from the others in the entire set of data items. Therefore, the proposed algorithm can preserve the access pattern privacy in C_A. In C_B, the access pattern privacy is preserved because C_B does not have any data items.

8.4. Practical Example of the Proposed k-Means Clustering Algorithm

The proposed secure k-means clustering algorithm can be used in various fields. For example, the proposed algorithm can be applied in healthcare to diagnose diseases using big data by clustering patients' symptoms. This allows for the assessment of the adaptability of the proposed algorithm in practical applications [31,35]. Because the existing disease diagnosis system depends only on the doctor's knowledge and experience, it may cause harm to patients due to misdiagnoses. Therefore, k-means clustering algorithms can help doctors classify the pattern of the patient's symptoms so as to diagnose what kind of disease it is. However, because patient information contains sensitive data, such as past medical history, family history, and allergies, the proposed privacy-preserving k-means clustering algorithm can be used to protect the sensitive data of patients. In addition, the proposed privacy-preserving k-means clustering algorithm can be used to solve the problem of insurance coverage recommendations where insurance companies provide the most suitable coverage to customers [36]. The insurance coverage recommendation clusters customers based on various types of customer information, such as movement patterns and lifestyles. To perform the grouping of customers, the proposed privacy-preserving k-means clustering algorithm can be used to protect the personal information of customers.

9. Conclusions and Future Work

In this paper, we proposed novel arithmetic-based encryption protocols (ASMIN, $\mathrm{ASMIN_n}$) designed for encrypted databases in cloud computing. These protocols address the challenges of the existing secure protocols, where the performance degradation is proportional to the data size. Our proposed protocols overcome these limitations, providing excellent processing performance that is independent of the data size. Additionally, we proposed a k-means clustering algorithm that supports privacy preservation using the proposed secure protocols. The algorithm utilizes enhanced secure protocols to perform encrypted index searches and queries, ensuring the protection of data, queries, and access patterns while providing superior processing performance. Additionally, the proposed k-means clustering algorithm requires moderate computational overhead because it uses decimal-based encryption operations, while the existing k-means clustering algorithms, i.e., Liu et al.'s work [24] and F. Rao et al.'s work [25], have high computational overhead due to the use of binary-based encryption operations.

Furthermore, we proposed a parallel k-means clustering algorithm for multi-core environments that leverages thread pools and random value pools. To the best of our knowledge, this is the first work to study a privacy-preserving parallel k-means clustering algorithm. For efficient support of k-means clustering, we utilized a thread pool to prevent data bottlenecks and parallelized encryption operation protocols. However, when parallelizing homomorphic encryption techniques, there is the issue of the increased operational overhead per core, leading to bottlenecks and performance degradation. To mitigate this bottleneck, the proposed parallel algorithm uses a random value pool to reduce the computational cost by preprocessing operations that generate random values. This algorithm optimizes its performance by eliminating the overhead associated with random number generation and encryption during k-means clustering.

Through performance evaluations, our proposed k-means clustering algorithm demonstrated approximately 10–12 times better performance than the existing algorithm. Moreover, the proposed parallel k-means clustering algorithm exhibited around 13 times better performance compared with the parallel processing versions of the existing algorithm.

Our future work will involve the actual implementation of a privacy-preserving healthcare system using the proposed k-means clustering algorithm in a cloud computing environment. For instance, within the healthcare system, the proposed algorithm can be applied to diagnose diseases using big data by clustering patients' symptoms. This allows for the assessment of the adaptability of the proposed algorithm in practical applications. In addition, our future work will focus on leveraging the advanced secure protocols proposed in this paper to support various cloud-based data analysis algorithms, including association rules, deep learning, and federated learning.

Author Contributions: Conceptualization, Y.S.; methodology, Y.S.; software, Y.S. and H.-J.K.; validation H.-J.K. and H.-J.L.; resources, Y.S. and H.-J.K.; data curation, H.-J.K.; writing—original draft preparation, Y.S.; writing—review and editing, Y.S., H.-J.L. and J.-W.C.; supervision, J.-W.C.; project administration, J.-W.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not relevant.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset used for the performance evaluation in our paper is the synthetic uniform dataset reference [34].

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

References

1. Agrawal, D.; Das, S.; Abbadi, A.E. Data management in the cloud: Challenges and opportunities. In *Synthesis Lectures on Data Management*; Springer Nature: Cham, Switzerland, 2012; Volume 4, p. 138. [CrossRef]
2. Grolinger, K.; Higashino, W.A.; Tiwari, A.; Capretz, M.A. Data management in cloud environments: NoSQL and NewSQL data stores. *J. Cloud Comput. Adv. Syst. Appl.* **2013**, *2*, 22. [CrossRef]
3. Zhao, L.; Sakr, S.; Liu, A.; Bouguettaya, A. Conclusions. In *Cloud Data Management*; Springer: Berlin/Heidelberg, Germany, 2014; p. 189. [CrossRef]
4. Bisht, J.; Vampugani, V.S. Load and cost-aware min-min workflow scheduling algorithm for heterogeneous resources in fog, cloud, and edge scenarios. *Int. J. Cloud Appl. Comput. (IJCAC)* **2022**, *12*, 1–20. [CrossRef]
5. Kumbhojkar, N.R.; Menon, A.B. Integrated predictive experience management framework (IPEMF) for improving customer experience: In the era of digital transformation. *Int. J. Cloud Appl. Comput. (IJCAC)* **2022**, *12*, 1–13. [CrossRef]
6. Sun, Y.; Zhang, J.; Zhu, G. Data security and privacy in cloud computing. *Int. J. Distrib. Sens. Netw.* **2014**, *10*, 190903. [CrossRef]
7. Sharma, Y.; Gupta, H.; Khatri, S.K. A security model for the enhancement of data privacy in cloud computing. In Proceedings of the 2019 Amity International Conference on Artificial Intelligence (AICAI), Dubai, United Arab Emirates, 4–6 February 2019; pp. 898–902. [CrossRef]
8. Garigipati, N.; Krishna, R.V. A study on data security and query privacy in cloud. In Proceedings of the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 23–25 April 2019; pp. 337–341. [CrossRef]
9. Sen, J. Security and Privacy Issues in Cloud Computing. *Cloud Technol. Concepts Methodol. Tools Appl.* **2015**, *2015*, 1585–1630. [CrossRef]
10. Jansen, W.A. Cloud hooks: Security and privacy issues in cloud computing. In Proceedings of the 2011 44th Hawaii International Conference on System Sciences, Kauai, HI, USA, 4–7 January 2011; pp. 1–10. [CrossRef]
11. Yiu, M.L.; Ghinita, G.; Jensen, C.S.; Kalnis, P. Enabling search services on outsourced private spatial data. *VLDB J.* **2010**, *19*, 363–384. [CrossRef]
12. Boldyreva, A.; Chenette, N.; Lee, Y.; O'Neill, A. Order-preserving symmetric encryption. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, 26–30 April 2009; pp. 224–241. [CrossRef]
13. Boldyreva, A.; Chenette, N.; O'Neill, A. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2011; pp. 578–595. [CrossRef]
14. Qi, Y.; Atallah, M.J. Efficient privacy-preserving k-nearest neighbor search. In Proceedings of the 28th International Conference on Distributed Computing Systems, Beijing, China, 17–20 June 2008; pp. 311–319. [CrossRef]

15. Shaneck, M.; Kim, Y.; Kumar, V. Privacy preserving nearest neighbor search. In *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*; Yu, P.S., Tsai, J.J.P., Eds.; Springer: Boston, MA, USA, 2009; pp. 247–276. [CrossRef]
16. Vaidya, J.; Clifton, C. Privacy-preserving top-k queries. In Proceedings of the 21st International Conference on Data Engineering (ICDE'05), Tokyo, Japan, 5–8 April 2005; pp. 545–546. [CrossRef]
17. Elmehdwi, Y.; Samanthula, B.K.; Jiang, W. Secure k-nearest neighbor query over encrypted data in outsourced environments. In Proceedings of the 2014 IEEE 30th International Conference on Data Engineering, Chicago, IL, USA, 31 March–4 April 2014; pp. 664–675. [CrossRef]
18. Kim, H.J.; Kim, H.I.; Chang, J.W. A privacy-preserving kNN classification algorithm using Yao's garbled circuit on cloud computing. In Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, 25–30 June 2017; pp. 766–769. [CrossRef]
19. Zhou, L.; Zhu, Y.; Castiglione, A. Efficient k-NN query over encrypted data in cloud with limited key-disclosure and offline data owner. *Comput. Secur.* **2017**, *69*, 84–96. [CrossRef]
20. Kim, H.I.; Kim, H.J.; Chang, J.W. A secure kNN query processing algorithm using homomorphic encryption on outsourced database. *Data Knowledge. Eng.* **2019**, *123*, 101602. [CrossRef]
21. Sun, X.; Wang, X.; Xia, Z.; Fu, Z.; Li, T. Dynamic multi-keyword top-k ranked search over encrypted cloud data. *Int. J. Secur. Its Appl.* **2014**, *8*, 319–332. [CrossRef]
22. Zhang, W.; Liu, S.; Xia, Z. A distributed privacy-preserving data aggregation scheme for smart grid with fine-grained access control. *J. Inf. Secur. Appl.* **2022**, *66*, 103118. [CrossRef]
23. Hozhabr, M.; Asghari, P.; Javadi, H.H.S. Dynamic secure multi-keyword ranked search over encrypted cloud data. *J. Inf. Secur. Appl.* **2021**, *61*, 102902. [CrossRef]
24. Liu, D.; Bertino, E.; Yi, X. Privacy of outsourced k-means clustering. In Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, Kyoto, Japan, 4–6 June 2014; pp. 123–134. [CrossRef]
25. Rao, F.Y.; Samanthula, B.K.; Bertino, E.; Yi, X.; Liu, D. Privacy-preserving and outsourced multi-user k-means clustering. In Proceedings of the 2015 IEEE Conference on Collaboration and Internet Computing (CIC), Hangzhou, China, 27–30 October 2015; pp. 80–89. [CrossRef]
26. 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.
27. Hazay, C.; Lindell, Y. Efficient Secure Two-Party Protocols: Techniques and Constructions. In *Information Security and Cryptography*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2010. [CrossRef]
28. Zhu, H.; Meng, X.; Kollios, G. Privacy Preserving Similarity Evaluation of Time Series Data. In Proceedings of the 17th International Conference on Extending Database Technology, Athens, Greece, 24–28 March 2014; pp. 499–510. [CrossRef]
29. Hu, H.; Xu, J.; Xu, X.; Pei, K.; Choi, B.; Zhou, S. Private search on key-value stores with hierarchical indexes. In Proceedings of the 2014 IEEE 30th International Conference on Data Engineering, Chicago, IL, USA, 31 March–4 April 2014; pp. 628–639. [CrossRef]
30. Liu, A.; Zhengy, K.; Liz, L.; Liu, G.; Zhao, L.; Zhou, X. Efficient secure similarity computation on encrypted trajectory data. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; pp. 66–77. [CrossRef]
31. Yang, Y.; Zheng, X.; Guo, W.; Liu, X.; Chang, V. Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system. *Inf. Sci.* **2019**, *479*, 567–592. [CrossRef]
32. Bugiel, S.; Nurnberger, S.; Sadeghi, A.R.; Schneider, T. Twin clouds: An architecture for secure cloud computing. In Proceedings of the Workshop on Cryptography and Security in Clouds, Zurich, Switzerland, 15–16 March 2011.
33. Goldreich, O. *Foundations of Cryptography: Volume 2, Basic Applications*; Cambridge University Press: Cambridge, UK, 2009; Volume 2.
34. Gray, J.; Sundaresan, P.; Englert, S.; Baclawski, K.; Weinberger, P.J. Quickly generating billion-record synthetic databases. In Proceedings of the 1994 ACM SIGMOD international conference on Management of data, Minneapolis, MN, USA, 24–27 May 1994; pp. 243–252. [CrossRef]
35. Hashi, E.K.; Zaman, M.S.U.; Hasan, M.R. An expert clinical decision support system to predict disease using classification techniques. In Proceedings of the 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox's Bazar, Bangladesh, 16–18 February 2017; pp. 396–400. [CrossRef]
36. Khalili-Damghani, K.; Abdi, F.; Abolmakarem, S. Solving customer insurance coverage recommendation problem using a two-stage clustering-classification model. *Int. J. Manag. Sci. Eng. Manag.* **2019**, *14*, 9–19. [CrossRef]