electronics

MDPI

Article

Homomorphic Encryption Based Privacy Preservation Scheme for DBSCAN Clustering

Mingyang Wang [1], Wenbin Zhao [1,*], Kangda Cheng [2], Zhilu Wu [2] and Jinlong Liu [2]

[1] Southwest Institute of Electronic Technology of China, Chengdu 610036, China; wangmy_papers@163.com
[2] School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China; kangdacheng@stu.hit.edu.cn (K.C.); wuzhilu@hit.edu.cn (Z.W.); yq20@hit.edu.cn (J.L.)
* Correspondence: zhaowenbin19870712@163.com; Tel.: +86-028-87556387

Abstract: In this paper, we propose a homomorphic encryption-based privacy protection scheme for DBSCAN clustering to reduce the risk of privacy leakage during data outsourcing computation. For the purpose of encrypting data in practical applications, we propose a variety of data preprocessing methods for different data accuracies. We also propose data preprocessing strategies based on different data precision and different computational overheads. In addition, we also design a protocol to implement the cipher text comparison function between users and cloud servers. Analysis of experimental results indicates that our proposed scheme has high clustering accuracy and can guarantee the privacy and security of the data.

Keywords: privacy protection; density clustering; homomorphic encryption

Citation: Wang, M.; Zhao, W.; Cheng, K.; Wu, Z.; Liu, J. Homomorphic Encryption Based Privacy Preservation Scheme for DBSCAN Clustering. *Electronics* **2022**, *11*, 1046. https://doi.org/10.3390/electronics11071046

Academic Editor: Pal Varga

Received: 20 February 2022
Accepted: 20 March 2022
Published: 26 March 2022

1. Introduction

The rapid development of cloud services has provided users with more diverse ways of handling data. The user can choose to upload the dataset to the cloud server, which will perform the relevant operations, thus reducing the use of the user's computing resources. Due to the wide variety of users, the datasets they provide cover a wide range of types, such as email information, back-office user database information, personal health status details, and other information that involves personal privacy or corporate trade secrets [1–5]. When users outsource the above information, there is a risk of leakage of sensitive information contained therein. Therefore, how to protect privacy with respect to sensitive data is a hot topic in current research.

There are two common ways of handling privacy protection: one is to secure the data from theft during transmission through a trusted channel provided by hardware or a trusted third party; the other is to encrypt the data and transmit it so that, even if an attacker obtains the ciphertext, it cannot be decrypted without a key. Although the first approach ensures security during transmission, it relies heavily on third parties, and, because it uses plaintext transmission, there is still a risk that the dataset will be stolen when it is transmitted to the cloud server. The second method encrypts the dataset and then transmits it so that the data is not compromised in plaintext, either during transmission or during server-side computation.

When encrypting a dataset, if a traditional encryption algorithm is used, the server cannot process the cryptomorphic data directly and requires the user to provide the server with a key or to perform a decryption operation. Homomorphic encryption is a new cryptographic tool that supports arbitrary functions on encrypted messages and decrypts them with the same result as if the corresponding operation had been performed on the plaintext. Homomorphic encryption algorithms can be broadly classified into the following categories according to the types and number of operations they support: FHE (fully homomorphic encryption) algorithms that support an unlimited number of operations and multiple operations, PHE (partial homomorphic encryption) algorithms that support an

unlimited number of operations and a limited number of types, and SWHE (somewhat homomorphic encryption) algorithms that support a limited number of operations and multiple operations [6–14]. Most applications require only a finite number of homomorphic addition or multiplication operations, and the SWHE algorithm is more efficient than the FHE algorithm, so the SWHE algorithm has a wider range of applications [15]. The BGV (Brakerski–Gentry–Vaikuntanathan) scheme is a commonly used SWHE algorithm that supports encryption of polynomials or integers, and enables parallelization operations using the Chinese residue theorem.

Homomorphic encryption satisfies both confidentiality and ciphertext manipulability, making it widely used in the field of privacy protection. The privacy protection model of outsourced computing based on homomorphic encryption is illustrated in Figure 1. The user encrypts the dataset to be outsourced and uploads it to the server-side (①), where the server performs data processing of higher computational complexity; then the server returns the result to the user (②).

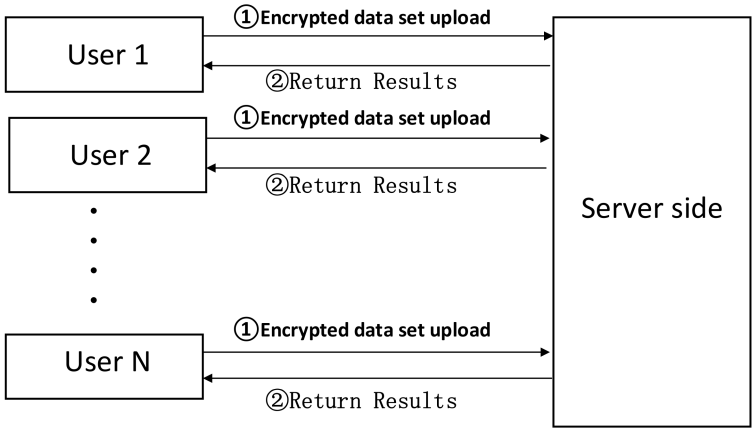

Figure 1. Privacy-preserving model for outsourced computing based on homomorphic encryption.

Machine learning is an important type of data outsourcing computing, and its privacy protection problem is an important class of application scenarios for homomorphic encryption. In the past few years, researchers have carried out many studies on the application of homomorphic encryption to machine learning privacy protection for different application scenarios, different data types, and different machine learning algorithms. The authors of [16] implemented an efficient and secure k-nearest neighbor algorithm. Other studies [17,18] involved implementation of a homomorphic K-means clustering algorithm on encrypted datasets, though this suffered from the problem of a large time overhead. In [19–21], the encrypted objects were pictures. After performing encryption operations on picture-based datasets, the authors extracted features on dense pictures and applied different image matching algorithms to achieve matching of dense pictures. A statistical learning algorithm on a dense state dataset was implemented and described in [22]. Further studies [23,24] reported the implementation of comparison algorithms or protocols on cryptographic datasets by means of bit-by-bit operations; however, their overhead was high. In [25], relevant features on digital-type datasets were pre-extracted, then the features were encrypted and uploaded to the cloud, where the classification operation was completed after performing homomorphic operations on the ciphertext of the relevant features. The scheme proposed in this paper is different in that the encrypted object is the original dataset, while the homomorphic clustering operation is performed in the cloud afterwards.

Common machine learning algorithms are mainly divided into supervised and unsupervised types. Supervised machine learning algorithms extract the features and true labels of the training dataset to build a model, and test the test data to produce the corresponding results, representing algorithms such as Bayesian classifier, hyperplane classifier, decision tree classifier, etc.; unsupervised machine learning algorithms do not need to pre-train the model, representing algorithms such as the K-means clustering algorithm, the DBSCAN

(density-based spatial clustering of application with noise) algorithm, etc. DBSCAN is an unsupervised density-based machine learning clustering algorithm that finds clusters of arbitrary shape in the presence of noisy points and has a wider range of applications than K-means, another common clustering algorithm, for example, for the implementation of advanced systems, such as recommender systems. DBSCAN can also be used for the design of a database structure in conjunction with accelerated range access (e.g., R*-tree). Therefore, the study of DBSCAN privacy protection issues is of great importance.

This paper presents a privacy-preserving scheme for homomorphic encryption-based clustering learning, which implements homomorphic DBSCAN for clustering operations on ciphertext datasets. To encrypt floating-point data in real-world scenarios, we propose three data preprocessing methods for different data precision, propose a data preprocessing strategy based on data characteristics and consider data precision and computation overhead. Since homomorphic encryption does not support ciphertext comparison operation, we design a protocol between the user and the cloud server to implement the ciphertext comparison function. Our work is expected to address privacy protection during data outsourcing computation in scenarios such as 5G communications [26], blockchain technology [27,28], IoT systems [29], wireless sensor networks [30], and big data system security [31].

2. Basic Knowledge

2.1. Introduction to Symbols

For a real number z, $\underline{z}$, $\overline{z}$ and $\langle z \rangle$ denote rounding z up, down and to the nearest integer, respectively; $[z]_m$ denotes z mod m. Use lowercase bold letters for vectors; uppercase letters $X = \{x_1, x_2, \cdots, x_i\}$ for sets and $|X|$ for the number of elements in the set; uppercase bold letters for matrices (e.g., $\boldsymbol{A}$) and $\boldsymbol{A^T}$ denotes the transpose of a matrix. The symbol $\leftarrow$ indicates random selection; R denotes the ring, $R_q = R/qR$. For an element $m \in M$ in the plaintext space, the symbol $[\![\mathrm{m}]\!]$ denotes its ciphertext.

2.2. Homomorphic Encryption Algorithms

Definition 1. *Homomorphic encryption. The homomorphic encryption algorithm consists of four main elements: KeyGen, Enc, Eval, and Dec.*

① $(pk, sk) \leftarrow KeyGen(\lambda)$*: Enter the security parameter* λ *and output the public key pk and the private key sk.*

② $c \leftarrow Enc(pk, m)$*: Enter the message plaintext* $m \in M$ *and the public key pk , and output the ciphertext c.*

③ $m \leftarrow Dec(sk, c)$*: Enter the ciphertext c and the private key sk and output the plaintext m of the message.*

④ $c_{Eval} \leftarrow Eval\ (\mathcal{C}, \{c_1, \cdots, c_k\},\ pk)$*: Enter a Boolean circuit* $\mathcal{C}$*, a set of ciphertexts* $\{c_1, \cdots, c_k\}$ *and the public key pk and output the resulting ciphertext* c_{Eval}*.*

In this paper, we choose the BGV scheme proposed in the literature [9] and implement it with the help of the homomorphic encryption algorithm library HElib. The BGV program consists of the following algorithms.

$Setup(1^\lambda, 1^\mu)$. The modulus q of μ bit is chosen randomly and $n = n(\lambda)$, $N = N(\lambda)$, $\chi = \chi(\lambda)$, $d = d(\lambda)$ is chosen. Such that $R = [x]/f(x)$, where $f(x)$ is an n order partition polynomial. The above parameters should be chosen in such a way that the difficulty of the scheme can be based on the lattice difficulty problem GLWE (general learning with error) and can resist existing attacks.

KeyGen(λ). Choose $\mathbf{s} \leftarrow \chi^d$ such that the private key is $\mathrm{sk} = (1, \mathbf{s}) \in R_q^{d+1}$. Randomly draw the matrix $\mathbf{A} \leftarrow R_q^{Nd}$, the vector $e \leftarrow \chi^N$, and compute $\boldsymbol{b} \leftarrow \boldsymbol{A}s + t > \boldsymbol{e}$, such that the public key is $\mathbf{B} = [\mathbf{b}, -\mathbf{A}]$. Clearly, $\mathbf{Bs} = te$.

Enc (pk, m). Expand the plaintext $m \in R_t$ into p_{k+1}, pick $\mathbf{r} \leftarrow R_2^N$ at random, and output the ciphertext $\mathbf{c} = \mathbf{m} + \mathbf{B}^{\mathrm{T}}\mathbf{r} \in R_q^{d+1}$.

$\mathrm{Dec}(\mathrm{sk}, c)$. Output $m = \left[\left[\langle \mathbf{c}, \mathbf{s} \rangle\right]_q\right]_t$.

Eval (c_1, c_2). Ciphertext addition outputs $c_1 \oplus c_2$ and ciphertext multiplication outputs $c_1 \otimes c_2$.

In addition to the above algorithms, HElib also includes some processing procedures for implementing the FHE algorithm [32], such as key switching, modulus switching, and bootstrapping. The SWHE algorithm does not support infinite ciphertext operations, but its ciphertext and key size are smaller, and it does not require bootstrapping to refresh the ciphertext, so it operates more efficiently. Since the scenario in this paper requires only a limited number of ciphertext operations, the SWHE algorithm is chosen to obtain high computational efficiency.

2.3. Data Pre-Processing Algorithms

Most data in real applications cannot be directly used as the plaintext of the encryption algorithm and requires some pre-processing algorithm to map the actual data into the plaintext space. The data to be processed by the DBSCAN algorithm studied in this paper is a floating-point dataset and therefore needs to be mapped to the plaintext space of the encryption algorithm using a preprocessing algorithm. In this section, three types of data pre-processing are described.

Mode 1. Integer pair encoding

Let the original data be a floating-point number c. If the integers a and b are chosen so that $b = \lceil a \times c \rfloor$, then use the number pair (a, b) to represent the number c. For example, 1.5 can be represented by the number pair (2,3); 1.2 can be represented by the number pairs (2,2), (5,6), etc. The selection of pairs should take into account both the time cost and the accuracy requirements.

Mode 2. Shift rounding encoding

Let the original data be c.d, where c is the integer part and d is the decimal part, and c and d have approximately the same number of digits. Combine the original data c.d is shifted by v decimal places to obtain the new data cd'.

Input c.d

Output $\mathrm{cd}' \leftarrow \lceil \mathrm{c} \cdot \mathrm{d} \times 10^{\mathrm{v}} \rfloor$

This processing is equivalent to expanding the original data by a factor of 10^v and then rounding the fractional part of the expanded data appropriately according to the actual requirements, thus enabling the accuracy of this type of data to be guaranteed without the need to select larger parameters for the encryption scheme at the time of encryption.

Mode 3. Multi-processing encoding

The original data c.d is as described in Mode 2, and the number of bits in both c and d is less than the maximum number of bits that the encryption algorithm can handle.

Input c.d

Output $(\mathrm{c.d})$

Split the integer part c and the fractional part d of the original data into the binary group $(\mathrm{c.d})$.

This method is the most universal and provides complete assurance that data accuracy will not be lost due to pre-processing operations; however, there are some drawbacks to this method of processing. If the user turns the original data into a binary using method 3, the amount of data becomes twice as large, which increases the overhead of the encryption phase. Moreover, when performing a homomorphic multiplication operation, instead of multiplying one pair of integers by two pairs of integers, the number of multiplications increases, and the complexity of the calculation increases.

In practice, depending on the accuracy of the data to be processed, one of the three pre-processing methods mentioned above can be chosen, or a combination of pre-processing methods can be chosen to maintain a balance between efficiency and accuracy. At the same time, when choosing a specific pre-processing method, the impact of the pre-processing method on the computational overhead of the subsequent homomorphic encryption should

also be taken into consideration, as the type of plaintext space and the fault tolerance of the chosen homomorphic encryption algorithm vary.

2.4. DBSCAN Algorithm

The DBSCAN algorithm is a common clustering algorithm used to construct clusters in a dataset and to discover noisy data [33,34]. Compared to the equally common K-means clustering algorithm, the DBSCAN algorithm does not require a predefined number of clusters and is suitable for the construction of clusters of any shape, even unconnected cyclic clusters. Due to the minimum number of points restriction, the DBSCAN algorithm avoids the single-link effect compared to K-means and, therefore, has better clustering results for any shape of data distribution.

The DBSCAN algorithm itself has many variations, the original DBSCAN algorithm has a complexity of $O(n^2)$, and the ρ-approximate DBSCAN algorithm has a complexity of $O(n)$, but it has some restrictions on the dimensionality of the data. The selection of the DBSCAN algorithm is related to the type of data. The two-dimensional DBSCAN algorithm was chosen to complete the experiments based on the data type of the dataset of the scheme in this paper.

Some concepts of the DBSCAN algorithm are defined as follows: MinPts define the minimum number of data points required for a cluster, and the ε-neighborhood represents the area covered by a circle with a point as its center and ε as its radius. Centroids, i.e., the centers of clusters, contain more data points than MinPts in their ε-neighborhood; edge points, i.e., nodes at the edges of clusters, contain fewer data points than MinPts in their ε-neighborhood and are in the ε-neighborhood of other centroids; and noise points, i.e., other data points in the dataset that are not centroids and edge points. An instantiated depiction of the node definition is shown in Figure 2. In addition, Definition 2 and Definition 3 give the definitions of density reachable and density connected.

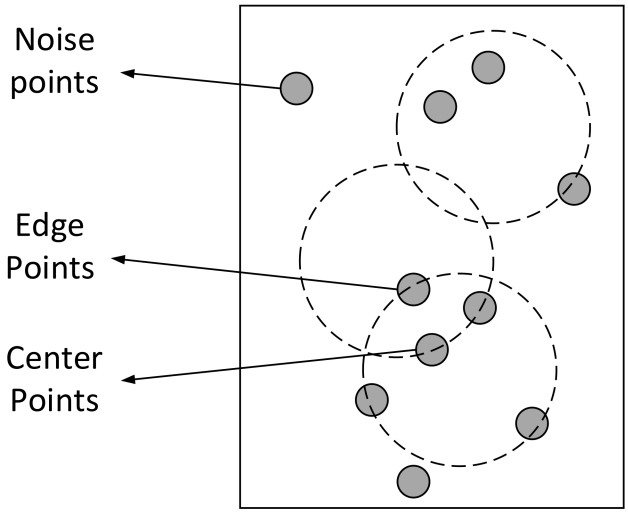

Figure 2. Instantiated description of the node definition.

Definition 2. *The density can be reached. Let x_i, x_j be 2 data points. Let there exist a sequence of samples $p_1, p_2, \cdots, p_n$, where $p_1 = x_i$, $p_n = x_j$, if for $1 \leqslant k \leqslant n$, all have p_{k+1} in the ε-neighborhood with p_k as the centroid, then x_j is said to be reachable by x_i density.*

Definition 3. *Density is connected. For a data point x_i, x_j, x_i is said to be density connected to x_j if there exists a point x_k such that x_i and x_j are density reachable after passing through x_k.*

The flow of the DBSCAN algorithm is as follows.

Step 1. Enter a collection of data. Select an unlabeled observation x as the current node and label the first clustering cluster 1.

Step 2. Find all the nodes in the ε-neighborhood centered on x, which are all neighbors of x. Perform the following operation.

① If the number of neighboring nodes found is less than MinPts, then x is a noisy point and step 4 is performed.

② If the number of neighboring nodes found is not less than MinPts, then x is a centroid and step 3 is executed.

Step 3. Separate all neighboring nodes as centroids and repeat step 2 until there are no new neighboring nodes to use as centroids.

Step 4. Select the next unlabeled point from dataset X as the current node, update the number of clusters and add 1.

Step 5. Repeat steps 2 to 4 until all points in dataset X have been marked.

2.5. Evaluation Criteria for the Programme

The program is evaluated according to two criteria: time efficiency and accuracy.

In terms of time, this paper is concerned with the time to execute the homomorphic DBSCAN algorithm in the cloud. In addition to the evaluation of time, another important evaluation criterion is the accuracy of the clustering. According to the characteristics of the DBSCAN algorithm, the evaluation criterion for accuracy contains the number of clusters and the noise point judgment. In this paper, we compare the results obtained by performing the DBSCAN algorithm directly on the plaintext with the results obtained by performing the DBSCAN algorithm homomorphically on the ciphertext and decrypting it. The accuracy of clustering is defined as the percentage of identical results in the two clusters—ideally, the two results should be identical.

3. DBSCAN Privacy Protection Solutions

In this paper, a homomorphic clustering algorithm on encrypted datasets is constructed to enable privacy protection of sensitive datasets during outsourcing computations. The operations in the DBSCAN algorithm can be divided into two categories according to the operations supported by the homomorphic encryption algorithm: operations supported by the homomorphic encryption algorithm and operations not supported by the homomorphic encryption. The homomorphic DBSCAN algorithm is shown in Algorithm 1.

Common homomorphic encryption schemes only support additive (subtractive) and multiplicative operations, so complex operations need to be converted accordingly so that they can be expressed in additive and multiplicative terms. The operations involved in the function findneighbor $\left([\![x_j]\!], [\![\varepsilon]\!]\right)$ in step 3) of Algorithm 1 are not fully supported by the homomorphic encryption algorithm. The following two operations are not supported.

(1) When computing $[\![\varepsilon]\!]$-neighborhoods, the distance between points needs to be calculated, the most common being the Euclidean distance, but the open square operation involved is not supported by homomorphic encryption algorithms.

(2) The homomorphic encryption scheme chosen in this paper does not have the nature of order-preserving encryption, and the size relationship between the encrypted ciphertexts will not be maintained between the original plaintexts, so the problem of ciphertext size comparison needs to be solved.

The solutions to these two problems are described in Sections 3.2 and 3.3, respectively.

Algorithm 1. Homomorphic DBSCAN algorithm

Input Ciphertext set $X = \{[\![x_1]\!], [\![x_2]\!], \ldots, [\![x_m]\!]\}$, ciphertext parameters $[\![\varepsilon]\!]$, MinPts
Output Clustered clusters $C_1, C_2, \ldots, C_k$
(1) Initializing a collection of centroids $\Omega = \varnothing$
(2) for $j = 1, 2, \ldots, m$ do
(3) Obtain the $[\![\varepsilon]\!]$-neighborhood of $[\![x_j]\!]$: $N_{[\varepsilon]}\left([\![x_j]\!]\right) = findneighbor\left([\![x_j]\!], [\![\varepsilon]\!]\right)$)
(4) if $\left|N_{[\varepsilon]}\left([\![x_j]\!]\right)\right| \geq MinPts$ *then*
(5) Put $[\![x_j]\!]$ into the set : $\Omega = \Omega \cup \left\{ [\![x_j]\!] \right\}$
(6) end if
(7) end for
(8) Initialize the set of clusters $k = 0$
(9) Initialize the set of unvisited data points $\Gamma = X$
(10) while $\Omega \neq \varnothing$ do
(11) Record the unvisited dataset $\Gamma_{\text{old}} = \Gamma$
(12) Pick a random center point $o \in \Omega$, initialize the queue $Q = \langle o \rangle$ ($\langle \bullet \rangle$ for queue, a first-in-first-out data structure)
(13) $\Gamma = \Gamma \backslash \{o\}$
(14) while $Q \neq \langle o \rangle$
(15) Pick $[\![q]\!]$ from queue Q
(16) if $\left|N_{[\varepsilon]}([\![, q,]\!])\right| \geq MinPts$ *then*
(17) Let $\Delta = N_{[\varepsilon]}([\![, q,]\!]) \cap \Gamma$
(18) Put the sample from Δ into Q
(19) $\Gamma = \Gamma \backslash \{\Delta\}$
(20) endf if
(21) end while
(22) $k = k + 1$, obtain the clusters $C_k = \Gamma_{\text{old}} \backslash \Gamma$
(23) $\Omega = \Omega \backslash C_k$
(24) end while

3.1. Dataset Pre-Processing

This section first gives a method for selecting a data pre-processing method based on data characteristics, combined with accuracy and computational overhead, etc. The selection process is shown in Figure 3.

Let the dataset to be processed be $X = \{x_1, x_2, \cdots, x_n\}$. The number of digits in the integer part of the data is p and the number of digits in the fractional part is q (insufficient valid digits are filled with zeros). Select the larger number x_{i} and the smaller number x_j from X and calculate the values of $\left|x_i - x_j\right|$. Determine whether this absolute value satisfies $\left|x_i - x_j\right| < \varsigma$, where ς is a user-defined threshold based on the characteristics of the dataset. If it is satisfied (i.e., the difference between the values is small), then the multiprocessing coding method is used directly to ensure data accuracy (in this case, the difference between the values in the set is mainly in the decimal part); if it is not satisfied, then it is further judged whether it satisfies $q - p > \eta$, where η is a threshold value set by the user according to the situation. If $q - p > \eta$, the lower digit of the decimal part will be rounded off (approximately $q - p$ bits), i.e., the digit of the decimal part will be changed from q to q' (so that $q' \approx p$); otherwise, it will directly jump to the next step of judgment. At this point, it is necessary to take into account the retention accuracy and the subsequent calculation overhead to determine whether to choose the integer-pair coding method or the shift-rounding coding method: if the loss of data accuracy at this point has a greater impact on the subsequent calculation results, in order to improve the calculation accuracy, then choose the integer-pair coding method; if the loss of data accuracy has a smaller impact on the subsequent calculation results, then choose the shift-rounding coding method.

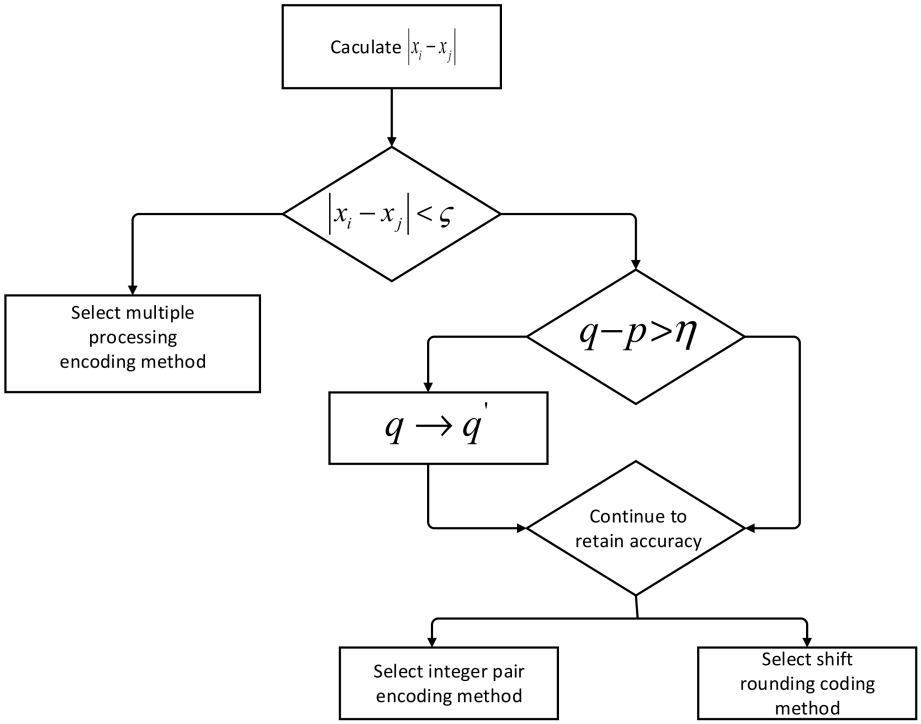

Figure 3. Selection process of pre-processing method.

The sensitive information that needs to be protected in this solution is the coordinate value of each piece of data, which is floating-point data, so a suitable pre-processing algorithm needs to be chosen according to the characteristics of the dataset and the encryption scheme chosen in this solution. In this paper, two floating-point datasets A and B are selected. The number of decimal places in the data in A is large, but the difference in the size of the values is large, and the number of decimal places is much larger than the number of integer places, so, according to the selection process shown in Figure 3, the integer-pair coding method or the shift-rounding coding method can be considered, and the decimal places are rounded off. The differences between the data in B are large and the number of fractional digits is not significantly different from the number of integer digits, so integer-pair coding or shift-rounding can also be considered, based on the selection methods described above. The encryption process, after processing the data using the integer-pair encoding method, is shown in Figure 4.

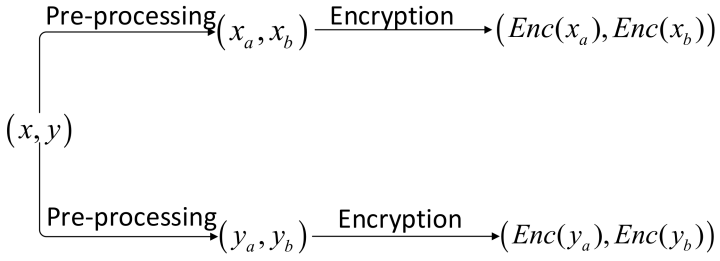

Figure 4. Encryption process after processing data using the integer-pair encoding method.

As can be seen in Figure 4, data is converted into a binary for encryption after processing in Mode 1, and subsequent ciphertext operations are increased, resulting in higher computational overhead, but the integer-pair encoding method preserves the accuracy of the current data as much as possible. The shift-rounding encoding method results in some loss of data precision, but the computational overhead is lower compared to the integer-pair encoding method. Therefore, it is possible to experimentally observe the effect of the loss of data precision on the clustering results for the two datasets A and B, This determines which

preprocessing method is more appropriate to use (see Section 4.1 for experimental results). The plaintext space of the encryption algorithm chosen for the scheme in this paper is a finite set of bit-length integers so that the preprocessed data can be encrypted directly.

3.2. Distance Measure Selection

In machine learning algorithms, commonly used distance measures include Euclidean distance, Manhattan distance, deformed Euclidean distance, etc. Compared to the traditional Euclidean distance, the deformed Euclidean distance can go further to preserve the accuracy of the data and reduce the impact of errors caused by open-square rounding.

Definition 4. *Deformed Euclidean distance. There are 2 points* $\boldsymbol{a}(x_1, y_1)$ $\boldsymbol{b}(x_2, y_2)$*, and the deformed Euclidean distance can be expressed as*

$$\mathrm{dis}(\mathbf{a}, \mathbf{b}) = (x_1 - x_2)^2 + (y_1 - y_2)^2 \tag{1}$$

When performing distance comparisons, the Euclidean and deformed Euclidean distances have high accuracy and the Manhattan distance has some error. However, the open-square operation in the Euclidean distance is not supported by the homomorphic encryption algorithm, and further processing of the open-square operation is required. A Newton iteration method for homomorphic open-square operations is used in the literature [35]*, and the procedure is as follows.*

Theorem 1. *Newton's iterative solution. If the equation* $x^2 - t = 0$*, where* t *is a real number, has a root near* x_0*, use the iterative formula*

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2}$$

Calculating $\mathrm{x}_1, \mathrm{x}_2, \ldots$ *in sequence..., the sequence will infinitely approximate the roots of the equation.*

As shown in Theorem 1, Newton's iterative method can convert the open-square operation into a base operation, thus satisfying the requirements of the homomorphic encryption scheme. Although this solution can achieve open-square operations, all constants involved in Equation (2) need to be encrypted in advance, and several ciphertext multiplication operations are involved in the operation process, resulting in high computational complexity. In addition, the result obtained by the Newton iteration method is an approximation, which will have a certain error, and this error may have a greater impact on the clustering results.

Theorem 2. *Let a and b be 2 integers and let the bit length of a be* l_1 *and the bit length of b be* l_2*, then we have ab with a bit length no greater than* $l_1 + l_2$ *and* $\sqrt{ab}$ *with a bit length no greater than* $(l_1 + l_2)/2$*.*

As shown in Theorem 2, the bit length of the result using the deformed Euclidean distance is twice as long as the Euclidean distance result. Although this results in a slightly higher computation time due to the larger size of the data during the subsequent computation, the additional overhead of the increased bit length is relatively low compared to the Newtonian iteration method (which requires multiple iterations and thus multiple multiplications to ensure the accuracy of the open-square result), where only two multiplications are required.

At the same time, this paper conducts a test of the effect of distance measures on the accuracy of clustering. In general, the distance measure used in clustering algorithms is the Euclidean distance, so the main way to test this is to replace the Euclidean distance with the Manhattan distance and the deformed Euclidean distance, respectively, and see if the clustering results are similar to those when the Euclidean distance is used. The results obtained by selecting the same input parameters and clustering on the plaintext using these

three distance measures are shown in Figure 5. The horizontal and vertical coordinates are the values of the data points, where the same color is the same clustering cluster, different colors are different clusters, and the darker data points at the edges are noisy. The three plots in Figure 5 show the clustering results for the selected Euclidean distance, Manhattan distance, and deformed Euclidean distance, respectively. As can be seen from Figure 5, the clustering results of Euclidean distance and deformed Euclidean distance are similar, while the clustering results for Manhattan distance have some errors compared to those of Euclidean distance. Combining the above analysis with the experimental results and considering the accuracy of the dataset and the overhead of the subsequent homomorphic operations, the deformed Euclidean distance is used in this paper.

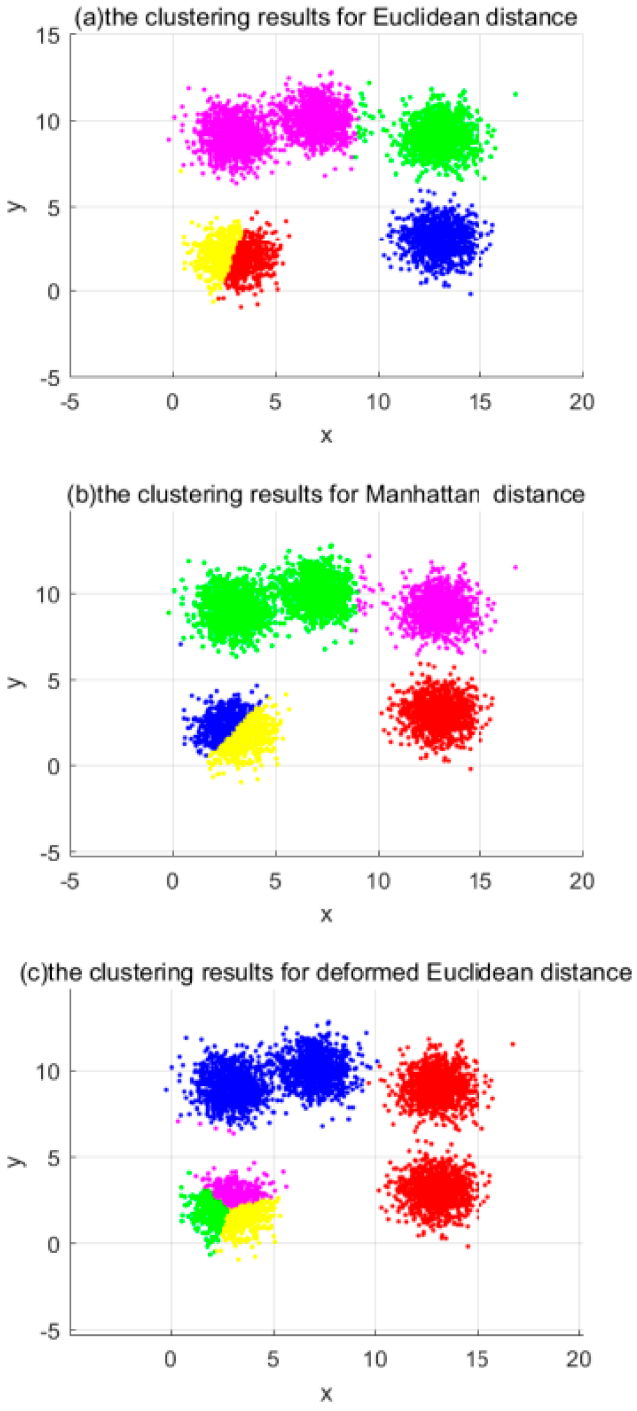

Figure 5. Dataset A: Clustering results with different distance measures.

3.3. Ciphertext Comparison Protocol

In Step 3 of Algorithm 1, determining whether a point is in a neighborhood of $[\![x_j]\!]$ requires a compare-size operation. However, the BGV homomorphic encryption algorithm used in this paper does not have the feature of order-preserving encryption, so the data encrypted using it cannot maintain the original size order and cannot be compared directly. To solve this problem, the solution in this paper is to design a protocol to compare the size of the ciphertext, as shown in Protocol 1. Let $x = (x_{l-1}, x_{l-2}, \cdots, x_0)_2$ and $y = (y_{l-1}, y_{l-2}, \cdots, y_0)_2$ be the binary representations of x and y, respectively. When comparing the size of the ciphertext on x and y, $\mathrm{Enc}(x) - \mathrm{Enc}(y) + \mathrm{Enc}(K)$ is first calculated, where K denotes an integer with more bits than l, and then the resulting ciphertext is returned to the user. The user decrypts the cipher text to obtain $\mathrm{tmp} = x - y + K$ and extracts its highest bit, $x > y$ if the bit is 1, or $x < y$ if the bit is 0. Since no homomorphic multiplication operation is used in the protocol, there is no need to introduce complex noise reduction techniques, which results in low time complexity. In a round protocol, the server sends to the user the ciphertext intermediate result $Enc(tmp)$ with a bit length of $O(dn \log q)$ (see Section 2.2 for the definition of the parameters n, d, q). After receiving $Enc(tmp)$, the user returns a 1-bit result $(0/1)$ to the server side. Therefore, the theoretical value of the communication volume of the round comparison protocol is $O(dn \log q)$.

Protocol 1. Ciphertext Comparison Protocol

Input $\mathrm{Enc}(\mathrm{dis}_1)$, $\mathrm{Enc}(\mathrm{dis}_2)$
Output **0/1**
(1) Server calculation $r = \mathrm{Enc}(\mathrm{dis}_1) - \mathrm{Enc}(\mathrm{dis}_2) + \mathrm{Enc}(K)$
(2) The server returns *r* to the client
(3) The user side decrypts *r* to get *tmp* and extracts the highest bit tmp_l
(4) if $tmp_l = 1$ **then**
(5) **return** **1**
(6) else
(7) **return** **0**
(8) end if

The interaction scenario between the user and the server is shown in Figure 6. The server sends the cipher text that needs to be decrypted to the user, and the user decrypts the cipher text and sends the highest bit of the decryption result back to the server. In this process, the user and the server can negotiate a time period t. Every t time, the user initiates a query to the server, and the server returns the intermediate result to be decrypted or the homomorphic clustering result according to the computing process. In this case, the user does not need to wait online all the time, but only needs to initiate a query at intervals. This process involves the direct transmission of plaintext, and an attacker may be able to obtain both plaintext and ciphertext through eavesdropping. The security aspects of this process are analyzed in detail in Section 5.2.

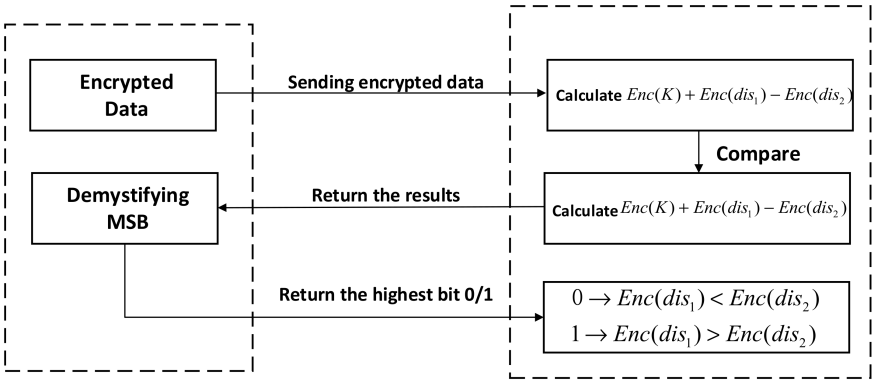

Figure 6. User-server interaction scenarios.

4. Program Realization

The configuration used in this paper is an Intel(R) Core (TM) i7-6700 CPU @ 3.40 GHz 3.41 GHz, 16 GB of RAM, and the Helib homomorphic encryption library for encryption and homomorphic clustering of datasets. Two common clustering datasets were selected for the solution. Dataset A is a common shape dataset with 5 clusters and 2000 data items, each containing two coordinates with 10 decimal places. Dataset B is the aggregation dataset, a common special shape clustering dataset with 750 data items, each containing two coordinates with two decimal places after each coordinate, essentially the same number of places as the integer part.

In this paper, we verify the accuracy of the scheme by comparing the clustering results on plaintext data with the homomorphic clustering results on ciphertext data. In the experiments, the parameters ε and MinPts used in the execution of the DBSCAN algorithm are the same for both plaintext and ciphertext datasets, with ε denoted as *eps* and MinPts denoted as min_*Pts* in the experiments.

4.1. Selection of Data Pre-Processing Methods

Dataset A and dataset B were selected from common clustering algorithm datasets, representing two cases with large and small differences between integer and fractional parts, respectively, to illustrate the universality of this paper's solution. The experimental results in this section are used to verify the validity of the method selected in Section 3.1 and to derive a more suitable preprocessing method for dataset A and dataset B. The three preprocessing methods were used to encode the two datasets, and then homomorphic clustering was performed to obtain the time overhead and accuracy of the homomorphic clustering algorithm, as shown in Table 1. As can be seen from Table 1, the accuracy of the multiprocessing coding is 100%, but the time overhead is higher; the accuracy of the integer-pair coding and the shift-rounding coding, although not 100%, is also very accurate, which indicates that the selection process in Section 3.1 first excluded the multiprocessing coding and considered dataset A and dataset B to be more suitable for the integer-pair coding or and that the fractional part of the data in dataset A was correctly rounded to some extent. In addition, the accuracy of the integer-pair coding method is slightly higher than that of the shift-rounding coding method, but the accuracy of the shift-rounding coding method is also very high, and the computational efficiency of the shift-rounding coding method is much higher than that of the integer-pair coding method. The experimental results show that the dataset selected for this paper is more suitable for the shift-rounding coding method, so the specific conclusions given in this paper are based on the choice of the shift-rounding coding method.

Table 1. Time overhead and accuracy of different coding methods.

Datasets	Integer Pair Encoding		Shift Rounding Encoding		Multiprocessing Codes	
	Time/s	Accuracy	Time/s	Accuracy	Time/s	Accuracy
A	298.45	99.81%	115.48	99.75%	302.45	100%
B	115.45	100%	45.28	99.74%	155.65	100%

4.2. Dataset Explicit Clustering Results

The results of the clustering process are shown in Figures 7 and 8. The parameters chosen for dataset A were $eps = 0.547$ (5470 after coding) and min_$Pts = 9$; dataset B was chosen with $eps = 1.8$ (180 after coding) and min_$Pts = 11$.

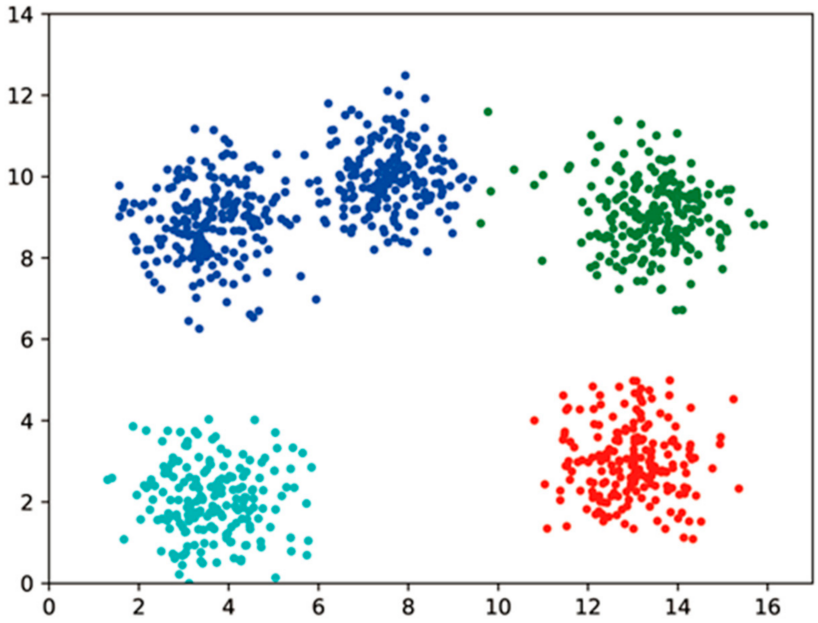

Figure 7. A plaintext clustering results.

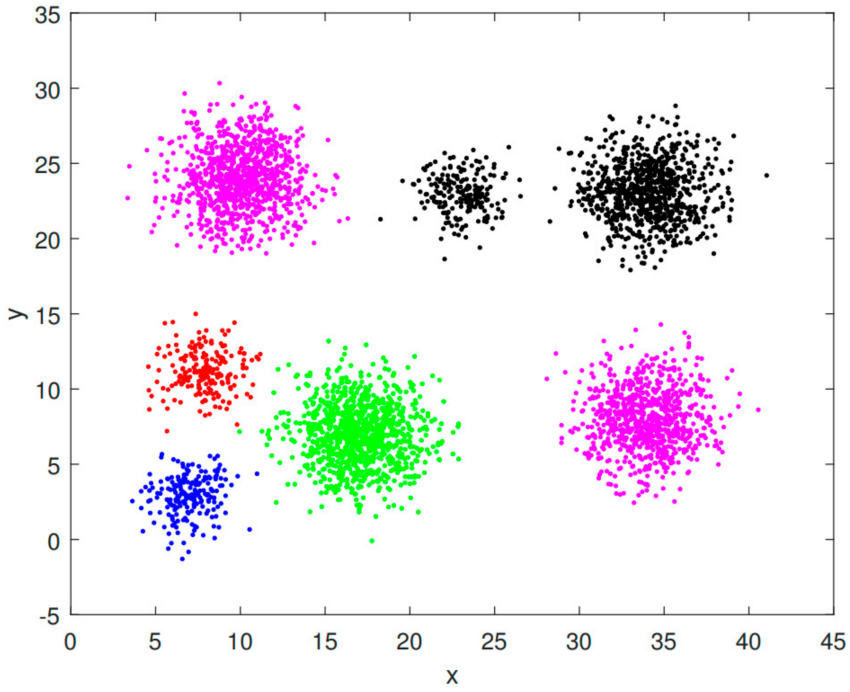

Figure 8. Dataset B plaintext clustering results.

4.3. Dataset Ciphertext Clustering Results

Dataset A and dataset B are encrypted by shifting and rounding, and then homomorphic clustering is performed on the encrypted data. The data in dataset A was shifted and rounded, with 3, 4, and 5 decimal places retained, and then the homomorphic clustering algorithm was performed on the cipher text. The clustering results after decryption are shown in Figures 9–11. The dataset B is directly encoded by shift rounding, encrypted and homomorphic clustering is performed; the clustering results obtained by decryption after the calculation are shown in Figure 12.

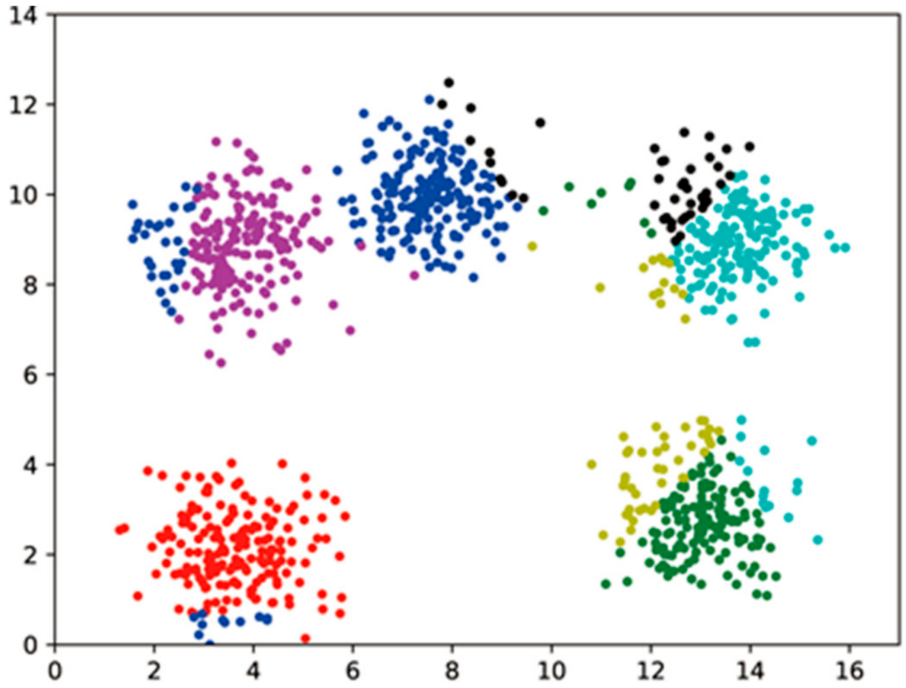

Figure 9. Dataset A Ciphertext clustering results with 3 decimal places.

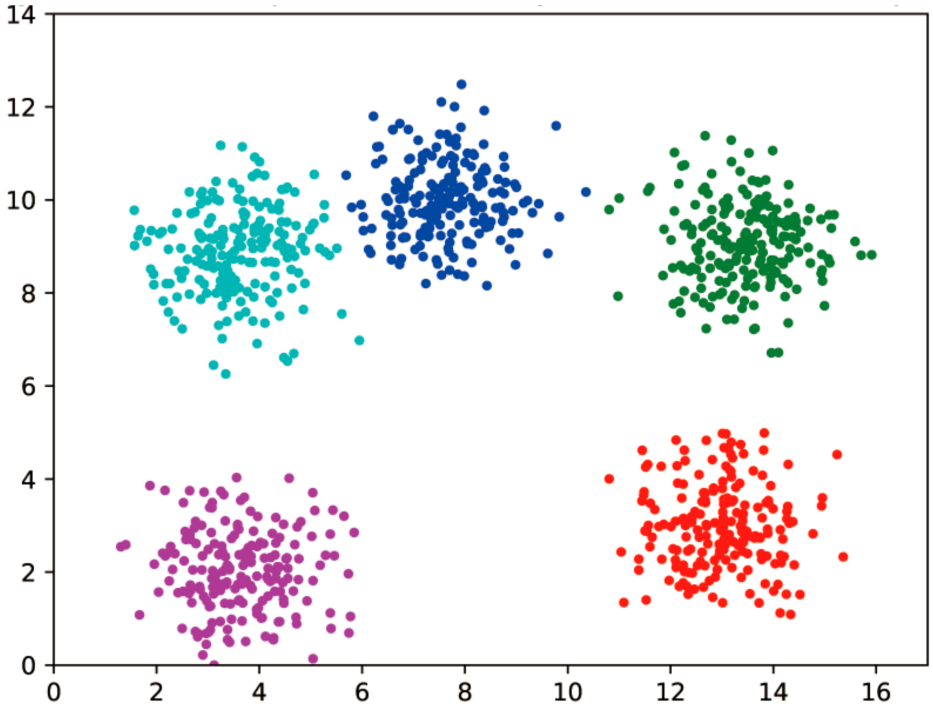

Figure 10. Dataset A Ciphertext clustering results with 4 decimal places.

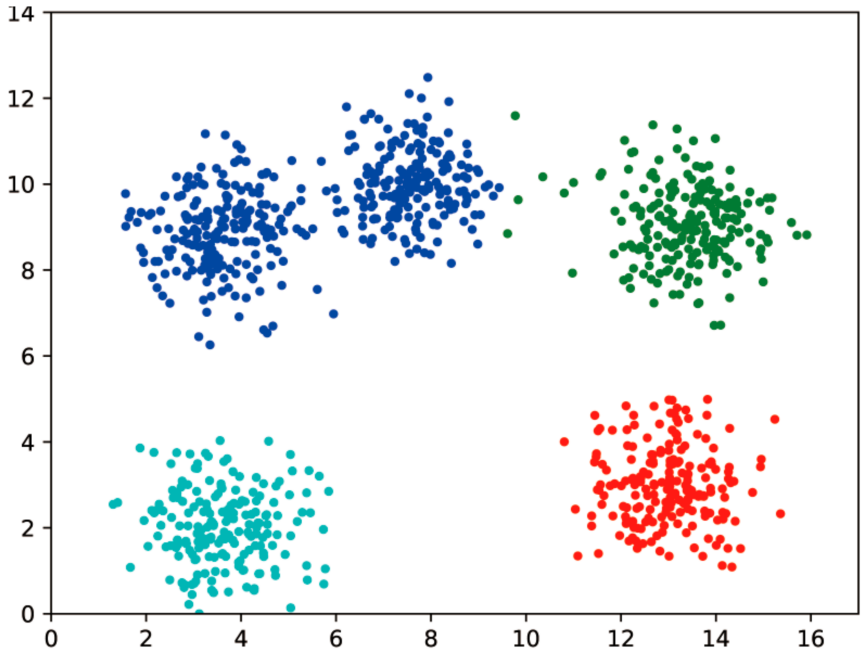

Figure 11. Dataset A retains 5 decimal ciphertext clustering results.

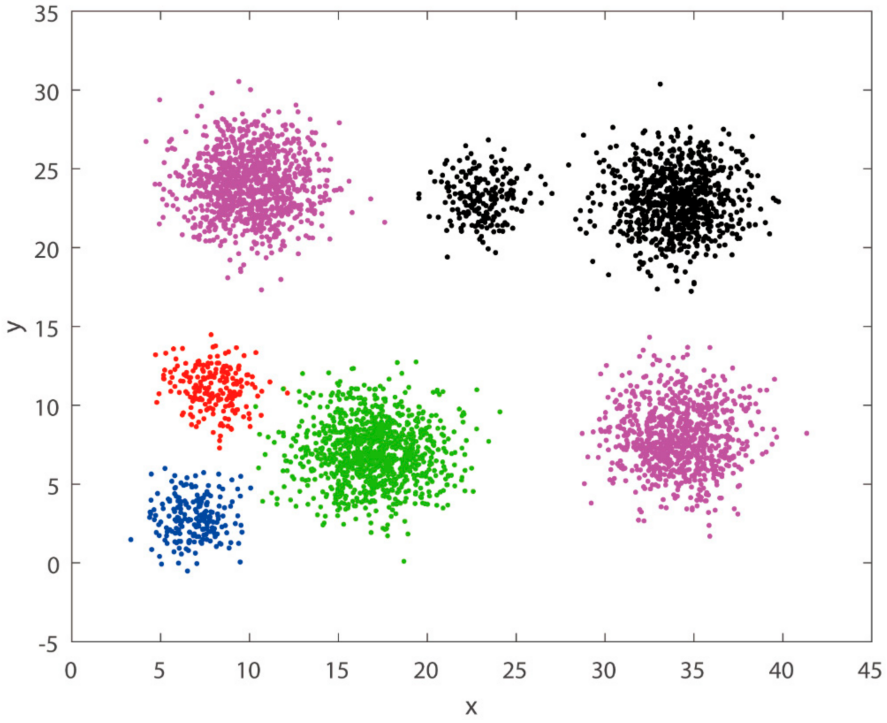

Figure 12. Dataset B ciphertext clustering results.

5. Program Evaluation

5.1. Analysis of Experimental Results

This section compares the plaintext and ciphertext clustering results for dataset A and dataset B, respectively, resulting in the scheme accuracies shown in Tables 2 and 3.

Table 2. Comparison of plaintext and ciphertext clustering results for dataset A.

Dataset A	Number of Clusters	Number of Noise Points	Accuracy
Plaintext datasets	5	93	100%
Encrypted datasets (3 decimal places)	4	90	80%
Encrypted datasets (4 decimal places)	5	90	99.75%
Encrypted datasets (5 decimal places)	5	90	99.75%

Table 3. Comparison of plaintext and ciphertext clustering results for dataset B.

Dataset B	Number of Clusters	Number of Noise Points	Accuracy
Plaintext datasets	7	8	100%
Encrypted datasets	7	6	99.74%

As can be seen from Table 2, the number of clusters for the encrypted dataset with 3 decimal places preserved at pre-processing differs significantly from the plaintext with an accuracy of only 80%; the encrypted dataset with further precision preserved (4 and 5 decimal places) has better clustering accuracy, with an accuracy of about 99.8% in these two cases. The results in Table 3 show that in the clustering results of dataset B, the number of clusters is the same for plaintext and ciphertext, the number of noise points is close, and the accuracy is above 99.7%. The accuracy of other existing machine learning privacy-preserving schemes using homomorphic encryption is reported in [36], with an average accuracy of 95% and a maximum accuracy of 99.8%, so the scheme in this paper achieves higher accuracy compared to previous schemes.

In terms of time overhead, scheme [18] encrypts the data bit-by-bit, resulting in a long computing time of 15 h for 400 dense 2D data; this scheme has a lower time overhead, taking only 45.28 s to perform the homomorphic clustering algorithm on 750 dense 2D data when processing dataset B.

The main reasons for the error in the clustering results include two aspects: on the one hand, the rounding error of the actual data, which was discarded to a certain extent during the data pre-processing stage in consideration of the subsequent encryption time overhead; on the other hand, as the ciphertext comparison protocol in this paper can only yield a greater-than or less-than relationship between two ciphertexts, the equal relationship is directly classified as greater than, which will cause some error. On the other hand, since the ciphertext comparison protocol in this paper only yields a greater-than or less-than relationship between two ciphertexts, the equals relationship is directly classified as greater than. The second problem can be improved by using a numerical comparator; HElib already has a function for this, but it requires several ciphertext multiplication operations, which greatly affects the computational efficiency.

5.2. Security Analysis

This section evaluates the security of the solutions in this paper. In this paper's scheme, the cloud server is set up as an honest but curious semi-honest model and the chosen BGV algorithm is a homomorphic encryption algorithm that satisfies semantic security.

The desired security goal of this solution is that the cloud server and an eavesdropping adversary cannot access the user's data information and cannot reverse the original data through the homomorphic DBSCAN algorithm. With these assumptions in mind, several theorems are given and proven below to illustrate the security of this solution.

Theorem 3. *The probability of the cloud server, or an adversary with eavesdropping capabilities, successfully recovering the user's original data based on the information they have obtained is negligible.*

Proof. That the data available to the cloud server, or an adversary with eavesdropping capabilities, is $X = \{[\![x_1]\!], [\![x_2]\!], \ldots, [\![x_m]\!]\}$ and the parameters $[\![\varepsilon]\!]$, MinPts. In terms of the composition of the data, with the exception of MinPts, all are ciphertexts obtained through BGV homomorphic encryption. The probability of the cloud server and the adversary successfully obtaining the user's original data without knowing the private key is equivalent to the probability of breaking the BGV algorithm. From the semantic security of the homomorphic encryption algorithm, it is clear that the probability of breaking the BGV algorithm is negligible and therefore the probability of the cloud server and adversary successfully obtaining the user's data is also negligible. The only plaintext in the data available to the cloud server or adversary is MinPts, which only indicates the minimum number of data contained in the clusters and does not reveal any of the user's data.

In addition, the scenario herein involves a protocol between the cloud server and the user. Essentially, in this protocol, the cloud server sends a BGV cipher to the user and the user returns to the cloud server a 0 or 1. The cloud server or adversary does not have access to the exact contents of the cipher without knowing the private key, and the plaintext (0 or 1) returned by the user at this point, and its corresponding cipher sent by the cloud server, do not constitute a "plaintext pair" as defined in cryptography. The user's returned plaintext (0 or 1) and its corresponding ciphertext sent by the cloud server do not constitute a "plaintext-ciphertext pair" as defined in cryptography, and thus the key cannot be deduced with enough 0 or 1 plaintexts and their corresponding ciphertexts. The proof is over. □

Theorem 4. *The probability that a cloud server, or an adversary with eavesdropping capabilities, can successfully obtain the original user data after acquiring the homomorphic DBSCAN algorithm is negligible.*

Proof. A cloud server or an adversary with eavesdropping capabilities may obtain the details of the homomorphic DBSCAN algorithm executed in the cloud by eavesdropping, etc. DBSCAN is an unsupervised machine learning algorithm, i.e., instead of pre-constructing a model by setting up a training set and then using the model to process other data, the results are obtained by clustering directly on the data. The solution in this paper retains this feature of the DBSCAN algorithm and works directly on the ciphertext data to obtain homomorphic clustering results, without a model. Unlike privacy-preserving schemes for classifiers, attacks that use the model parameters to invert the user's original data cannot be applied to this scheme. In addition, the algorithm in this paper essentially computes distances based on the homomorphism of the ciphertext data and then labels each data point with the cluster it belongs to, or marks it as a noisy point, so even if an attacker had access to the algorithm process, he would not be able to, Therefore, the data security of this paper is based on the encryption algorithm.

In summary, the probability that an attacker obtaining the homomorphic DBSCAN algorithm can successfully obtain the original user data is equivalent to the probability of breaking the BGV algorithm, which is negligible by Theorem 4 Proof. □

6. Conclusions

In this paper, we propose a scheme to perform a homomorphic DBSCAN clustering algorithm on encrypted datasets for solving the privacy protection problem during data outsourcing computation. For the comparison operation, which is not supported by the homomorphic encryption algorithm, an interaction protocol is designed to implement this function. The scheme proposes multiple coding preprocessing methods for different datasets and gives a strategy for selecting data preprocessing methods according to the characteristics of the datasets, taking into account the accuracy of the data and the computational overhead. Our proposed scheme has reliable data security, a good clustering effect, and computational performance. The drawback is that this paper only discusses the privacy leakage risk in the clustering operation process, and we will discuss a more

comprehensive computation process in subsequent work. This research is expected to help solve the privacy protection problem in data-outsourcing computation, in scenarios such as 5G communication, blockchain technology, IoT systems, wireless sensor networks, and big data system security.

Author Contributions: Conceptualization, Z.W.; Data curation, K.C.; Software, J.L.; Writing—original draft, M.W.; Writing—review & editing, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the 'Fund project of innovation theory and technology group' of CETC Tianao Co., Ltd., Chengdu, Sichuan, China. No. 2020JSQ020501.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declares that there are no conflict of interest regarding the publication of this article.

References

1. Alabdulkarim, A.; Al-Rodhaan, M.; Al-Dhelaan, T.A. A Privacy-Preserving Algorithm for Clinical Decision-Support Systems Using Random Forest. *Comput. Mater. Contin.* **2019**, *58*, 585–601. [CrossRef]
2. Centonze, P. Security and Privacy Frameworks for Access Control Big Data Systems. *Comput. Mater. Contin.* **2019**, *59*, 361–374. [CrossRef]
3. Patel, D.; Srinivasan, K.; Chang, C.Y.; Gupta, T.; Kataria, A. Network Anomaly Detection inside Consumer Networks—A Hybrid Approach. *Electronics* **2020**, *9*, 923. [CrossRef]
4. Salim, M.M.; Kim, I.; Doniyor, U.; Lee, C.; Park, J.H. Homomorphic Encryption Based Privacy-Preservation for IoMT. *Appl. Sci.* **2021**, *11*, 8757. [CrossRef]
5. Ghani, N.B.; Ahmad, M.; Mahmoud, Z.; Mehmood, R.M. A Pursuit of Sustainable Privacy Protection in Big Data Environment by an Optimized Clustered-Purpose Based Algorithm. *Intell. Autom. Soft Comput.* **2020**, *26*, 1217–1231. [CrossRef]
6. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *arXiv* **2017**, arXiv:1704.03578. [CrossRef]
7. Liu, X.; Zhang, X. NOMA-based Resource Allocation for Cluster-based Cognitive Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5379–5388. [CrossRef]
8. Liu, X.; Zhai, X.B.; Lu, W.; Wu, C. QoS-guarantee Resource Allocation for Multibeam Satellite Industrial Internet of Things with NOMA. *IEEE Trans. Ind. Inform.* **2021**, *17*, 2052–2061. [CrossRef]
9. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science*; ACM Press: New York, NY, USA, 2012; pp. 309–325.
10. Xin, L.; Zhang, X. Rate and Energy Efficiency Improvements for 5G-Based IoT with Simultaneous Transfer. *IEEE Internet Things J.* **2018**, *6*, 5971–5980.
11. Liu, X.; Zhang, X.; Jia, M.; Fan, L.; Lu, W.; Zhai, X. 5G-based green broadband communication system design with simultaneous wireless information and power transfer. *Phys. Commun.* **2018**, *28*, 130–137. [CrossRef]
12. Brakerski, Z.; Vaikuntanathan, V. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Annual Conference on Advances in Cryptology*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 505–524.
13. Lu, W.; Gong, Y.; Liu, X.; Wu, J.; Peng, H. Collaborative Energy and Information Transfer in Green Wireless Sensor Networks for Smart Cities. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1585–1593. [CrossRef]
14. Kim, J.; Kim, S.; Seo, J.H. A new scale-invariant homomorphic encryption scheme. *Inf. Sci.* **2018**, *422*, 177–187. [CrossRef]
15. Jiang, L.Z.; Xu, C.X.; Wang, X.F.; Chen, K.F.; Wang, B.C. Application of homomorphic encryption for encrypted computing models. *J. Cryptologic Res.* **2017**, *4*, 596–610.
16. Yang, H.; He, W.; Li, J.; Li, H. Efficient and Secure kNN Classification over Encrypted Data Using Vector Homomorphic Encryption. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–7.
17. Cheon, J.H.; Jeong, J.; Ki, D.; Kim, J.; Lee, J.; Lee, S.W. Cryptology ePrint Archive: Report 2019/466—Privacy Protection K-Means Clustering with Multiple Data Owners. 2019. Available online: https://eprint.iacr.org/2019/466 (accessed on 19 February 2022).
18. Angela, J.; Armknecht, F. Unsupervised machine learning on encrypted data. In *International Conference on Selected Areas in Cryptography*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 453–478.
19. Hu, S.; Wang, Q.; Wang, J.; Qin, Z.; Ren, K. Securing SIFT: Privacy protection outsourcing computation of feature extractions over encrypted image data. *IEEE Trans. Image Process.* **2016**, *25*, 3411–3425. [CrossRef] [PubMed]
20. Chen, G.; Chen, Q.; Zhu, X.; Chen, Y. Encrypted image feature extraction by privacy protection MFS. In Proceedings of the 2018 International Conference on Digital Home, Guilin, China, 30 November–1 December 2018; p. 42.

21. Jiang, L.; Xu, C.; Wang, X.; Luo, B.; Wang, H. Secure outsourcing SIFT: Efficient and Privacy-preserving Image Feature Extraction in the Encrypted Domain. *IEEE Trans. Dependable Secur. Comput.* **2017**, *17*, 179–193. [CrossRef]
22. Jiang, L.; Xu, C.; Wang, X.; Lin, C. Statistical learning based fully homomorphic encryption on encrypted data. *Soft Comput.* **2017**, *21*, 7473–7483. [CrossRef]
23. Bacon, D.F.; Bent, G.A.; Bergamaschi, F.A.; Zhang, W. Performing Efficient Comparison Operations on Encrypted Data. U.S. Patent 10,015,007, 3 July 2018.
24. Jiang, L.; Cao, Y.; Yuan, C.; Sun, X.; Zhu, X. An effective comparison protocol over encrypt-ed data in cloud computing. *J. Inf. Secur. Appl.* **2019**, *48*, 102367.
25. Jia, C.F.; Wang, Y.F.; Chen, Y.; Sun, M.F.; GE, F.Y. Machine learning algorithms on homomorphic encrypted data set. *J. Tsinghua Univ. (Sci. Technol.)* **2020**, *60*, 456–463.
26. Wang, J.; Han, H.; Li, H.; He, S.; Sharma, P.K.; Chen, L. Multi Strategies Differential Privacy on Sparse Tensor Factorization for Network Traffic Analysis in 5G. *IEEE Trans. Ind. Inform.* **2021**, *18*, 1939–1948. [CrossRef]
27. Zhang, J.; Zhong, S.; Wang, J.; Yu, X.; Alfarraj, O. A Storage Optimization Scheme for Blockchain Transaction Databases. *Comput. Syst. Sci. Eng.* **2021**, *36*, 521–535. [CrossRef]
28. Zhang, J.; Zhong, S.; Wang, J.; Wang, L.; Yang, Y.; Wei, B.; Zhou, G. A Review on Blockchain-Based Systems and Applications. In *International Conference on Internet of Vehicles*; Springer: Berlin/Heidelberg, Germany, 2020.
29. Le Nguyen, B.; Lydia, E.L.; Elhoseny, M.; Pustokhina, I.; Pustokhin, D.A.; Selim, M.M.; Nguyen, G.N.; Shankar, K. Privacy Preserving Blockchain Technique to Achieve Secure and Reliable Sharing of IoT Data. *Comput. Mater. Contin.* **2020**, *65*, 87–107. [CrossRef]
30. Li, X.; Chen, D.; Li, C.; Wang, L. Secure Data Aggregation with Fully Homomorphic Encryption in Large-Scale Wireless Sensor Networks. *Sensors* **2015**, *15*, 15952–15973. [CrossRef] [PubMed]
31. Wang, J.; Yang, Y.; Wang, T.; Sherratt, J.; Zhang, J. Big data service architecture: A survey. *J. Internet Technol.* **2020**, *21*, 393–405.
32. Halevi, S.; Shoup, V. Bootstrapping for HElib. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 641–670.
33. Kryszkiewicz, M.; Skonieczny, L. Faster clustering with DBSCAN. In *Intelligent Information Processing and Web Mining*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 605–661.
34. Chen, Y.; Tang, S.; Bouguila, N.; Wang, C.; Du, J.; Li, H. A fast clustering algorithm based on pruning unnecessary distance computations in DBSCAN for high-dimensional data. *Pattern Recognit.* **2018**, *83*, 375–387. [CrossRef]
35. Cheon, J.H.; Kim, D.; Kim, D.; Lee, H.H.; Lee, K. Numerical method for comparison on homomor-phically encrypted numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Cham, Switzerland, 2019; pp. 415–445.
36. Tan, Z.W.; Zhang, L.F. Survey on privacy preserving techniques for machine learning. *J. Softw.* **2020**, *31*, 2127–2156.