# Data Clustering using Homomorphic Encryption and Secure Chain Distance Matrices

Nawal Almutairi[1,2], Frans Coenen[1] and Keith Dures[1]
[1]*Department of Computer Science, University of Liverpool, U.K.*
[2]*Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia*

Keywords: Privacy Preserving Data Mining, Secure Clustering, Homomorphic Encryption, Order Preserving Encryption, Secure Chain Distance Matrices.

Abstract: Secure data mining has emerged as an essential requirement for exchanging confidential data in terms of third party (outsourced) data analytics. An emerging form of encryption, Homomorphic Encryption, allows a limited amount of data manipulation and, when coupled with additional information, can facilitate secure third party data analytics. However, the resource required is substantial which leads to scalability issues. Moreover, in many cases, data owner participation can still be significant, thus not providing a full realisation of the vision of third party data analytics. The focus of this paper is therefore scalable and secure third party data clustering with only very limited data owner participation. To this end, the concept of Secure Chain Distance Matrices is proposed. The mechanism is fully described and analysed in the context of three different clustering algorithms. Excellent evaluation results were obtained.

## 1 INTRODUCTION

Data mining is a well established research field that has been effectively employed in many disciplines. However, the exponential growth in data availability has lead to the involvement of third parties for the purpose of storing and processing data on behalf of data owners; for example third party data mining or collaborative data mining. The involvement of third parties has clear security risks associated with it, such as unauthorised data access (data leakage) and issues concerning data privacy preservation. One current solution is Privacy Preserving Data Mining (PPDM) (Agrawal and Srikant, 2000; Lindell and Pinkas, 2002). The typical approach is to conceal sensitive data attributes by applying some form of data transformation to generate "sanitised" counterpart attribute values that can be safely disclosed to (untrusted) third parties (Chhinkaniwala and Garg, 2011). Well known transformation techniques include value perturbation and data anonymisation (Chhinkaniwala and Garg, 2011). Such techniques tend to operate by introducing "statistical noise" to either the entire dataset or to selected sensitive attributes. An issue with these techniques is that they cannot guarantee data confidentiality; it might still be possible to "reverse engineer" the original data (Vaidya et al., 2006; Berinato, 2015).

Data confidentiality, and protection against leakage, can be assured using data encryption. However, standard forms of encryption do not support data mining activities, which typically require data manipulation and record comparison. A potential solution is the use of Homomorphic Encryption (HE) schemes that provide *malleability properties* that permit limited calculation over cyphertexts without compromising security. Although HE schemes support primitive operations that go some way to supporting data mining, they do not provide an entire solution. For example they do not support record comparison; a requirement with respect to many data mining algorithms. One mechanism whereby this can be addressed is to involve data owners so that the operations that a given HE scheme does not support can be performed by the data owners (Erkin et al., 2009; Liu et al., 2014). For example, in the context of data clustering, record similarity checking can be conducted in this manner. However, the degree of data owner involvement can be substantial given any kind of sophisticated data analysis task, which in turn detracts from the vision of third party data mining.

There has been some work that seeks to diminish data owner participation with respect to third party data clustering. Of note is the 3-D Updatable Distance Matrix (UDM) introduced in (Almutairi et al., 2017). However, use of the UDM featured two disadvantages: (i) a substantial memory requirement, because the first two dimensions of the matrix were correlated to the number of records in the given dataset thus limiting the scalability and (ii) the potential for reverse engineer given that a UDM is essentially a (very large) set of linear equations.

Given the above, this paper proposes the idea of the Secure Chain Distance Matrix (SCDM) which provides for secure third party data mining using a proposed Order Preserving Encryption (OPE) scheme, which can limit recourse to data owners during the processing of the data (depending on the nature of clustering) and features none of the memory requirement and security disadvantages associated with the UDM concept proposed in (Almutairi et al., 2017). The novel elements of the SCDM concept are firstly the chaining mechanism used, which means that the storage requirement, compared with UDMs, is reduced by a factor equivalent to the number of input data records ($-1$). Secondly, the proposed Order Preserving Encryption (OPE) scheme with which the matrix is encoded, thus allowing for third party record comparison without the risk of potential reverse engineering as in the case of UDM. The SCDM concept is fully described and evaluated. The evaluation is conducted in the context of three different clustering algorithms (Nearest Neighbour, DBSCAN and k-Means), however, the SCDM idea clearly has wider application.

The rest of this paper is structured as follows. Section 2 provides a review of related research. Section 3 presents the data encryption schemes used to provide for proposed secure clustering methods. The proposed Secure Chain Distance Matrix (SCDM) idea is then detailed in Section 4. The utilisation of the SCDM concept, in the context of secure data clustering, is presented in Section 5. Section 6 then reports on the experiments conducted to evaluate the SCDM concept and the results obtained (in the context of secure data clustering). The paper is concluded in Section 7, with a summary of the main findings and suggestion for future work.

## 2 PREVIOUS WORK

This section presents a review of previous work on secure data clustering that uses HE schemes as a data confidentiality preservation method. The main challenge of HE-based privacy preserving data clustering (and other forms of data mining), is that HE schemes support only a limited number of operations. Several solutions have been proposed to address this challenge, mostly in the context of collaborative data clustering whereas the work presented in this paper is directed at third party data clustering, which can be broadly categorised into: (i) involving data owners when unsupported operations are required, and (ii) utilising the concept of "secret sharing" to delegate a key and operations to semi-honest and non-colluding parties that collaboratively perform operations on the data owners' behalf. Both have limitations in term of communication complexity and security threats.

The main feature of the first category is the maintenance of data confidentiality by allowing a third party to only manipulate cyphertexts using HE properties (no access to any secret key). In this case, in the context of data clustering, data owner participation becomes a necessity. In some cases, the majority of the work is done by data owners. For example, a number of authors have proposed mechanisms for k-means clustering using Secure Multi-Party Computation (SMPC), where data owners repeatedly cluster their own data and only share encrypted data centroids so that an eventual global clustering can be arrived at (Jha et al., 2005; Mittal et al., 2014). A similar idea is used in (Tong et al., 2018) to implement DBSCAN where data owners independently apply DBSCAN on their local data. The resulting boundary records and their labels are then shared (in plaintext) with the third party who then determines global boundary records which are returned to the individual data owner so that they can update their local clusters. However, sharing boundary data records in plaintext form presents a security threat. Secure nearest neighbour clustering is presented in (Shaneck et al., 2009) using SMPC primitives; secure product for distance calculation and Yao's millionaires' protocol for data comparison. A significant drawback of these proposed solutions is that they introduce a computation/communication overhead because of the amount of data owner participation required.

In (Erkin et al., 2009; Liu et al., 2014; Almutairi et al., 2017; Rahman et al., 2017) the basic idea was for the third party to do as much of the clustering as possible (centroid calculation, data aggregation and so), using the properties of a selected HE scheme, and involve data owners only when the properties of the particular HE scheme used do not support the desired analysis. For example, in the case of (Erkin et al., 2009), in the context of collaborative clustering, the adopted HE scheme does not support record similarity checking, thus this is done by a randomly selected data owner. The number of data owner participation instances is given by $n \times |C| \times i$, where $n$ is the number of records, $|C|$ the number of centroids and $i$ the number of clustering iterations; thus the amount of data owner participation is considerable. The concept of “trapdoors” are used to minimise the number of data owner participation in (Liu et al., 2014). “Static” and “dynamic” trapdoors were therefore calculated by the data owners so as to convert cyphertexts to order cyphertexts; consequently off-line comparison (without data owner participation) was supported. However, the main issue with this approach was that it was very inefficient, particularly when considering large datasets; in addition data owner participation could still be high (depending on the nature of the clustering) because of the need to recalculate the dynamic trapdoors on each iteration. An alternative is the UDM concept presented in (Almutairi et al., 2017) that dramatically reduces the data owner participation overhead to $|C| \times i$, however this also has limitations; firstly in terms of security in that a UDM represents a set of linear equations that might be reverse engineered (although the set of linear equations is very large), and secondly in terms of memory requirement and communication complexity cost in that the size of a UDM increases exponentially with the number of records in the input dataset. The third party collaborative DBSCAN mechanism described in (Rahman et al., 2017) uses HE properties to calculate the required distances. However, as noted previously, the generated cyphers do not preserve the data ordering, thus data owner participation was still required to determine whether the distances were below or above the DBSCAN threshold $\varepsilon$ value.

The second category comprises more recent work that uses “secret sharing” to eliminate data owner participation. The basic idea is to use a scheme, as in (Hazay et al., 2012), that mathematically splits a secret key among multiple semi-honest and non-colluding parties that collaboratively manipulate data on behalf of data owners. In (Rao et al., 2015; Samanthula et al., 2015), the concept of secret sharing was used to design secure computation protocols that securely execute mathematical operations by third parties without involving data owners in the data mining process. The limitation of this approach is that it tends to be inefficient and not practical for large datasets. In addition, the requirement for at least two semi-honest and none-colluding parties is of concern, and for many data owners a security risk. Moreover, the secret key that has been generated to encrypt a given dataset cannot be revoked by data owners, thus a version of the data needs to be stored locally by each individual data owners.

## 3 DATA ENCRYPTION

The proposed Secure Chain Distance Matrix (SCDM) based clustering approaches utilised two encryption schemes: (i) Liu’s HE scheme (Liu, 2013) and (ii) a proposed Order Preserving Encryption (OPE). The first is used to encrypt the data to be outsourced, the second to encrypt the CDM. Both are discussed in further detail in the following two sub-sections, Subsections 3.1 and 3.2 respectively.

### 3.1 Liu’s Homomorphic Encryption Scheme

In Liu’s scheme each data attribute $v$ is encrypted into $m$ sub-cyphers, $C = \{c_1, c_2, \ldots, c_m\}$ where $m \geq 3$. Algorithm 1 shows the pseudo code for the $Encrypt(v, K(m))$ where $K(m)$ is a list of secret keys. $K(m) = [(k_1, s_1, t_1), \ldots, (k_m, s_m, t_m)]$ and $k_i$, $s_i$ and $t_i$ are real numbers. Given a set of sub-cyphers $C = c_1, \ldots, c_m$ and the key $K(m)$ Algorithm 2 gives the pseudo code for the $Decrypt(C, K(m))$ decryption function to return the value $v$.

Algorithm 1: Encrypt(v, K(m)).

```
1: procedure ENCRYPT($v, K(m)$)
2:     generate $m$ arbitrarily real random numbers $r_1, \ldots\ldots, r_m$
3:     Declare $C$ as a real value array of $m$ elements
4:     $c_1 = k_1 * t_1 * v + s_1 * r_m + k_1 * (r_1 - r_{m-1})$
5:     for $i = 2$ to $m - 1$ do
6:         $c_i = k_i * t_i * v + s_i * r_m + k_i * (r_i - r_{i-1})$
7:     $c_m = (k_m + s_m + t_m) * r_m$
8:     Exit with $C$
```

Algorithm 2: Decrypt(C, K(m)).

```
1: procedure DECRYPT($C, K(m)$)
2:     $T = \sum_{i=1}^{m-1} t_i$
3:     $S = c_m / (k_m + s_m + t_m)$
4:     $v = (\sum_{i=1}^{m-1} (c_i - S * s_i) / ki) / T$
5:     Exit with $v$
```

The scheme has both security and homomorphic properties. The scheme is semantically secure in that it produces different cyphertexts for the same plaintext on each occasion, even when the same secret key is used. Further detail regarding the security of Liu’s scheme is given in Section 6. In terms of its homomorphic properties the scheme supports: the addition of cyphertexts $\oplus$ and multiplication of a cyphertext with a real value $\otimes$ as shown in Equation 1. Hence the subtraction $\ominus$ of cyphertexts and the division $\oslash$ of a cyphertexts by a real value are implemented as given in Equation 2.

$$\begin{array}{rll} C \oplus C' = & \{c_1 \oplus c'_1, \ldots, c_m \oplus c'_m\} & = v + v' \\ r \otimes C = & \{r \otimes c_1, \ldots, r \otimes c_m\} & = r \times v \end{array} \tag{1}$$

$$\begin{aligned} C \ominus C' &= C \oplus (-1 \otimes C') \\ C \oslash r &= \frac{1}{r} \otimes C \end{aligned} \tag{2}$$

## 3.2 Order Preserving Encryption

A Chain Distance Matrix (CDM) holds distances between every attribute value in each consecutive data record according to whatever ordering is featured in the data; further detail regarding the generation of CDMs is given in Section 4. As in the case of the UDM, the content of a CDM can be used to define a set of linear equations that might allow for re-engineering. To prevent such re-engineering the idea is to encode the CDM, to give a Secure CDM (SCDM), by using an Order Preserving Encryption (OPE) scheme, a form of encryption where the ordering of the values is maintained so as to allow (secure) comparison.

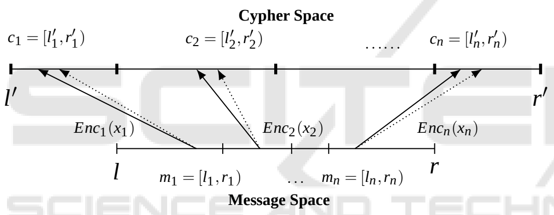

Figure 1: Message and extended cypher space splitting.

The proposed scheme is an amalgamation of two existing OPE schemes, that of (Liu et al., 2016) and (Liu and Wang, 2013). Using the proposed scheme the expected message space $M$ and the expanded cypher space $C$ is known in advance as in the case of most OPE schemes. The expanded cypher space should be much larger than the message space, $|C| \gg |M|$. Using the proposed scheme $M = [l, r)$ and $C = [l', r')$ where $l$ and $l'$ are the minimum boundary values and $r$ and $r'$ are the maximum boundary values (as demonstrated in Figure 1). The key feature of the proposed OPE scheme is that it obscures any data distribution that might be included in the generated cyphertexts using the concept of message space splitting and non-linear cypher space expansion. To generate the desired cypher space the first step is to randomly split the message space $M$ into $n$ successive intervals $M = \{m_1, m_2, \ldots, m_n\}$ where $m_i = [l_i, r_i)\ \forall (i = 1, 2, \ldots, n)$. The process of splitting satisfies the following: $M = \cup_{i=1}^{i=n} m_i = \cup_{i=1}^{i=n} [l_i, r_i) = [l, r)$ and $[l_i, r_i) \cap [l_j, r_j) = \phi\ \forall\ i \neq j$. Next, the cypher space $C$ was also split into $n$ successive intervals $C = \{c_1, c_2, \ldots, c_n\}$. The length of the cypher space intervals are determined by the density of the corresponding message space interval, so that for a dense interval, containing high frequency data, its corresponding cypher space interval will result in a longer cypher interval range.

Each interval has associated with it an encryption function that maps data from the message space $m_i$ to cyphertext in the corresponding extended cypher space $c_i$. In context of the work presented in this paper, the encryption function of the $i$th interval is shown in Algorithm 3; the algorithm encrypts a plaintext value $x \in m_i$ to an encrypted value $x' \in c_i$. Range and Range$'$ (lines 2 and 3) return the maximal and minimal value for the message space interval $m_i$ and the corresponding cypher space $c_i$ respectively. In line 4 the $Scale_i$ value is calculated as a division of the cypher space size over the corresponding message space size. The values of the minimal cypher space, minimal message space and the scale are used to generate a cypher $x'$ (line 5). To obfuscate the occurrence frequency of a data value a random value $\delta_i$ is applied to $x'$ (lines 6 and 7). The value of $\delta_i$ is sampled from the range $[0, Sens * Scale_i)$ where $Sens$ represents the data sensitivity as proposed in (Liu and Wang, 2013); the minimum distance between plaintext values.

Algorithm 3: Order Preserving Encryption algorithm.

```
1: procedure ENC_i($x, Sens$)
2:     $l_i, r_i \leftarrow$ Range($i$)
3:     $l'_i, r'_i \leftarrow$ Range$'$($i$)
4:     $Scale_i = \frac{(l'_i - r'_i)}{(l_i - r_i)}$
5:     $x' = l'_i + Scale_i \times (x - l_i)$
6:     $\delta_i$ = Random(0, $Sens \times Scale_i$)
7:     $x' = x' + \delta_i$
8:     Exit with $x'$
```

## 4 THE SECURE CHAIN DISTANCE MATRIX (SCDM)

Regardless of whether standard or HE encryption is used, data encryption randomly transfers plaintexts values in a dataset $D$ to cyphertexts in such a way that any ordering is not preserved. Therefore, data comparisons cannot be directly applied to the cyphertexts, and hence clustering algorithms cannot be directly applied. The proposed idea is to support third party secure clustering over encrypted data using the concept of a Secure Chain Distance Matrix (SCDM) that holds the distance between every attribute in every consecutive data records in $D$. An SCDM is a 2D matrix whose first dimension is $n - 1$ where $n$ is the number of records in $D$ and whose second dimension is $|A|$ (the size of the attribute set $A$). An SCDM is generated in two steps: (i) CDM calculation and (ii) CDM encryption. Algorithm 4 gives the CDM calculation process. Instead of calculating distances between attribute values in records with the corresponding attribute values in every other record, as in the case of UDMs, a CDM holds only distances between $n-1$ records. This small set of distances allows a third party to calculate the "order" of similarity between any two data records $r_x$ and $r_y$ (where $x < y$) in $D$ as per equation 3. In the case of $x = y$ the distance will clearly be 0.

Algorithm 4: Chain Distance Matrix Calculation.

```
1: procedure CDMCALCULATION($D$)
2:     $CDM = \emptyset$ array of $n-1$ rows and $|A|$ column
3:     for $i = 1$ to $i = n-1$ do
4:         for $j = 1$ to $j = |A|$ do
5:             $CDM_{[i,j]} = D_{[i,j]} - D_{[i+1,j]}$
6:     Exit with $CDM$
```

$$Sim(CDM, r_x, r_y) = \sum_{j=1}^{j=|A|} \left| \sum_{i=x}^{i=(y-1)} CDM_{[i,j]} \right| \tag{3}$$

Although the CDM reduces the memory requirement, compared to UDM, it still essentially comprises a set of linear equations that may support re-engineering. Therefore, the second step is to encrypt the matrix so that the distance ordering is preserved to give an SCDM. To this end, the OPE described in Sub-section 3.2 above was used. The key feature of the encrypted CDM, the SCDM, is that a third party now has access to the distance value ordering, not the original distance values, between the data records. The order of similarity is determined as per Equation 3 but with the CDM replaced by the SCDM.

## 5 SECURE DATA CLUSTERING

This section presents a number of examples secure clustering algorithms that operate over HE data and utilise the SCDM concept, two of these achieve the ideal solution (require no data owner participation whilst the third party processing is taking place), the third only requires minimal data owner participation. Whatever the case the clustering process has two parts, data preparation (Sub-section 5.1) and the data clustering (Sub-sections 5.2, 5.3 and 5.4), conducted by a data owner and the third party data miner respectively.

### 5.1 Data Preparation

The initial step in the preparation process is to translate a given dataset $D$ into a suitable format that allows distance calculation and data comparison. In the context of the work presented in this paper, casting is used to transfer categorical values to discrete integer equivalents. The next step is to encrypt the data attribute values to produce an encrypted dataset $D'$ using Liu's HE scheme and $m = 4$ (see Sub-section 3.1). The CDM is then calculated using Algorithm 4 and encoded using the proposed OPE scheme (Sub-Section 3.2). The output from the preparation process is the encrypted dataset $D'$ and the *SCDM* ready to be sent to the third party data miner.

### 5.2 Secure Nearest Neighbour Clustering (SNNC)

The proposed SCDM concept was combined with three popular clustering schemes: (i) Nearest Neighbour Clustering, (ii) DBSCAN and (iii) k-Means. The first is discussed in this section, and the remaining two in the following two sections. The pseudo code for the proposed SNNC approach is presented in Algorithm 5, it operates in a similar manner to the standard NNC algorithm (Cover and Hart, 1967). The main differences are that the data and threshold values $\sigma'$ is encrypted and that the similarity between data records is determined using the SCDM. The algorithm commences by adding the first encrypted record $r'_1$ to the first cluster (lines 2 and 3) and then iteratively clustering the remaining records (lines 5 to 11). As in case of standard NNC, a record $r'_i$ will be assigned to a cluster if there exists some record $r'_m$ whose distance from $r'_i$ is less than or equal to $\sigma'$ (lines 5 to 8). If there is no such record, $r'_i$ is assigned to a new cluster (lines 10 and 11). The similarity between records $r'_i$ and $r'_m$ is determined using Equation 3 (and the SCDM). The algorithm will exit with a cluster configuration $C$.

### 5.3 Secure DBSCAN (SDBSCAN)

The SDBSCAN algorithm is presented in Algorithm 6. The inputs are the encrypted dataset $D'$ and SCDM previously provided by the data owner and the desired density parameters $(MinPts, \varepsilon')$. Similar to the standard DBSCAN (Ester et al., 1996), density is defined as the minimum number of points, *MinPts*, within a certain distance $\varepsilon$. Note that, in case of SDBSCAN the $\varepsilon$ value is encrypted using the OPE scheme to give $\varepsilon'$ that allows secure comparison and hides the correlation (distance) between data records when the third party data miner executes SDBSCAN.

Algorithm 5: Secure NN clustering algorithm.

```
1:  procedure SNNC($D', SCDM, \sigma'$)
2:      $C_1 = \{r'_1\}$
3:      $C = \{C_1\}$
4:      $k = 1$
5:      for $i = 2$ to $i = |D'|$ do
6:          Find $r'_m$ in some cluster in $C$ where $Sim(SCDM, r'_i, r'_m)$ is minimised
7:          if $Sim(SCDM, r'_i, r'_m) \leq \sigma'$ then        ▷ (Eq. 3)
8:              $C_m = C_m \cup r'_i$
9:          else
10:             $k++$
11:             $C_k = \{r'_i\}$
12:     Exit with $C$
```

The algorithm commences by initialising the global variables *MinPts*, $\varepsilon'$, *SCDM* and $D'$ by the values received from the data owner (line 5). In line 6, an empty set of clusters $C$ is created and the number of clusters so far is set to 0. For each record $r'_i$ in $D'$ that has not been previously assigned to a cluster, “unclustered”, the set $S$ is determined. The set $S$ is the $\varepsilon$-neighbourhood of $r'_i$ and comprises the set of records in $D'$ whose distance from $r'_i$ is less than or equals to $\varepsilon'$. The set is determined by calling the *regionQuery* procedure (line 9) where the SCDM is used to determine the overall distances between records (see Equation 3). If the number of records in $S$ is greater than or equals to *MinPts* the density requirement is satisfied thus $r'_i$ is marked as “clustered” and considered to represent a new cluster $C_k$ (lines 11 to 13). This cluster is then expanded by considering the records in $S$ using the *expandCluster* procedure called in line 14. The input to the *expandCluster* procedure is: the cluster $C_k$ so far and the set $S$. The *expandCluster* procedure is a recursive procedure. For each record in $S$ which has not been previously clustered we add the record to $C_k$ and then determine the $\varepsilon$-neighbourhood $S_2$ for this record (line 22). If the size of $S_2$ is greater than or equals to *MinPts* we call the *expandCluster* procedure again and so on until all the records in $D'$ are processed at which point the algorithm will exist with the cluster configuration $C$.

## 5.4 Secure k-Means (Sk-Means)

The secure k-Means process is again very similar to the standard k-Means algorithm (MacQueen et al., 1967). However, the mathematical operations are replaced with equivalent secure operations using the HE properties as presented in Sub-section 3.1. The pseudo code is given in Algorithm 7. The inputs are the encrypted dataset $D'$, the SCDM and number of desired clusters $k$. The algorithm commences by: initialising the global variables (line 5), dimensioning

Algorithm 6: Secure DBSCAN clustering algorithm.

```
1:  global variables
2:      $MinPts$, $\varepsilon'$, $SCDM$, $D'$
3:  end global variables
4:  procedure SDBSCAN($D'$, $SCDM$, $MinPts$, $\varepsilon'$)
5:      Initialized global variables with received values
6:      $C = \emptyset$, $k = 0$
7:      for $i = 1$ to $i = |D'|$ do
8:          if $r'_i$ is $Unclustered$ then
9:              $S = regionQuery(r_i)$
10:             if $|S| \geqslant MinPts$ then
11:                 mark $r'_i$ as $clustered$
12:                 $k = k + 1$
13:                 $C_k = r'_i$
14:                 $C_k = expandCluster(C_k, S)$
15:                 $C = C \cup C_k$
16:     Exit with $C$
17: procedure EXPANDCLUSTER($C$,$S$)
18:     for $\forall\, r'_i \in S$ do
19:         if $r'_i$ is $Unclustered$ then
20:             mark $r'_i$ as $clustered$
21:             $C = C \cup r'_i$
22:             $S_2 = regionQuery(r_i)$
23:             if $|S_2| \geqslant MinPts$ then
24:                 $C = expandCluster(C, S_2)$
25:     Exit with $C$
26: procedure REGIONQUERY($r'_{Index}$)
27:     $N_\varepsilon =$ empty set
28:     for $\forall\, r'_j \in D$ do
29:         $distance = Sim(SCDM, r'_{Index}, r'_j)$        ▷ (Eq. 3)
30:         if $distance \leq \varepsilon'$ then
31:             $N_\varepsilon.add(r'_j)$
32:     Exit with $N_\varepsilon$
```

the cluster array $C = \{C_1, C_2, \ldots, C_k\}$ and assigning the first $k$ encrypted records to it (lines 6 and 7). A centroid set $Cent = \{cent_1, cent_2, \ldots, cent_k\}$ is thus defined to hold the current centroids (lines 8 and 9). The remaining encrypted data records are then assigned to a cluster according to their similarity with respect to the cluster centroids using the *populateClusters* procedure (called from line 10) and given at the end of algorithm. In *populateClusters* the order of similarity is calculated using the SCDM as shown in Equation 3. A set of new centroids ($Cent'$) are then calculated (line 11) using the HE properties of Liu’s scheme. An iterative loop is then entered (lines 12 to 19) that repeats until stable centroids are arrived at. The first step of each k-Means iteration is to calculate the Shift matrix $S$ (line 13) that represents the distances between the previous iteration centroids ($Cent$) and the newly calculated centroids ($Cent'$). Note that $S$, is calculated using the HE properties, over the HE data ,therefore, the next step requires recourse to the data owner (lines 14) to decrypt matrix ($S$) and re-encrypt it using OPE to give $S'$ so that it can be used to update the SCDM (line 15). The algorithm will use

Algorithm 7: Secure k-Means clustering algorithm.

```
1:  global variables
2:      $D'$, $SCDM$
3:  end global variables
4:  procedure SK-MEANS($D'$, $SCDM$, $k$)
5:      Initialized global variables with received values
6:      $C$ = Set of $k$ empty clusters
7:      Assign first $k$ records in $D'$ to $C$ (one per cluster)
8:      $Cent$ = Set of $k$ cluster centroids
9:      Assign first $K$ records in $D'$ to $Cent$
10:     $C$ = PopulateClusters($k+1, C, Cent$)
11:     $Cent'$ = CalculateCentroids($C$)
12:     while $Cent \neq Cent'$ do
13:         $S = Cent \ominus Cent'$
14:         $S' = S$ decrypted and encrypt result using OPE
15:         $SCDM = SCDM + S'$
16:         $C$ = Set of $k$ empty clusters
17:         $C$ = PopulateClusters($1, C, Cent'$)
18:         $Cent = Cent'$
19:         $Cent'$ = CalculateCentroids($C$)
20:     Exit with $C$
21: procedure POPULATECLUSTERS($x, C, Cent$)
22:     $id = null$
23:     for $x = x$ to $x = |D'|$ do
24:         for $y = 1$ to $y = |C|$ do
25:             $sim = Sim(SCDM, r_x, c_y)$                ▷ (Eq. 3)
26:             $id$ = cluster identifier with lowest $sim$
                        value so far
27:         $C_{id} = C_{id} \cup r_x$ ($C_{id} \in C$)
28:     Exit with $C$
```

$S'$ to update the SCDM by concatenating the $k$ elements in $S'$ to SCDM (line 15). In the following iteration $S'$ is used to update the first $k$ elements in the SCDM. Using the newly calculated centroids all records are again assigned to each cluster, using the *populateClusters* procedure, in the same manner as before; and so on until a fixed configuration is reached.

## 6 EVALUATION

The evaluation of the proposed clustering approaches is presented in this section. For the purpose of the evaluation fifteen datasets from the UCI data repository (Lichman, 2013) were selected in a manner so that datasets of a variety of sizes and different numbers of classes could be considered (these are listed in columns 2 and 3 of Table 1). The number of classes in each case was used as the value for $k$ in the case of k-Means clustering. The proposed approaches were implemented using the Java programming language. The overall objective was to evaluate the proposed algorithms in term of: (i) data owner participation, (ii) scalability, (iii) clustering efficiency, (iv) clustering accuracy and (v) security.

Data owner participation was measured in terms of the runtime required for data preparation and SCDM generation, and the amount of data owner involvement during the clustering process. Preparation time results are presented in Table 1 where columns 4, 7 and 8 give the preparation times for: data encryption, CDM calculation and CDM encryption respectively. From the table, it can be seen that negligible time was required for the data preparation; even with respect to the largest dataset, Arrhythmia. For SDBSCAN and SNNC no further data owner participation was required, whereas in the case of Sk-Means the participation was limited to the decryption and re-encryption of the shift matrix, $S$, on each iteration; thus data owner participation was limited to $O(|C| \times i)$, where $|C|$ is the number of centroids and $i$ is the number of k-Means iterations, the same as in the case of the UDM approach from (Almutairi et al., 2017).

The chain feature in SCDM reduces the required memory resources compared to the UDM concept in (Almutairi et al., 2017). The number of elements in a UDM grows exponentially with the data volume; more formally it equals to ($\frac{n(n+1)}{2} \times |A|$)) (Column 6 in Table 1 gives the number of UDM elements for each experimental dataset). The SCDM is more compact and hence requires significant lower resource (Column 9 in Table 1) which makes it more appropriate for big data. This small number of elements means that the time required to calculate an SCDM is less than the UDM (as shown in Columns 5 and 7 in Table 1).

In terms of clustering efficiency, the runtime to cluster the data using the proposed secure clustering mechanisms was compared with the standard equivalent processes. The runtime results are presented in Figure 2. From the figure, it can be seen that the overall runtimes required for the secure clustering approaches, as expected, were longer than in the case of standard approaches, however, inspection of the recorded results indicates that this did not present a significant overhead. Of course, the bigger the dataset the larger the SCDM, and consequently the greater the time required to interact with the SCDM to cluster data.

In terms of accuracy, cluster configuration “correctness” was measured by comparing the results obtained with those obtained using standard (unencrypted) clustering algorithm equivalents. The parameters used (in practice selected by the data owner) are given in columns 10 to 13 of Table 1. The accuracy metric used was the Silhouette Coefficient (Sil. Coef.) (Rousseeuw, 1987), a real number value between $-1$ and $+1$, the closer the value is to 1 the bet-

Table 1: Run times for data owner data preparation and algorithm operating statistics.

| No. Dataset | R × C | Num Class Labels | Data Encrypt. (MSec) | UDM Cal. (MSec) | UDM Size | CDM Cal. (MSec) | CDM Encrypt. (MSec) | CDM Size | σ | *k* | *Min Pts* | ε |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. Arrhythmia | 452×279 | 16 | 11.08 | 218.5 | 28563462 | 3.33 | 136.09 | 125829 | 1 | 16 | 2 | 600 |
| 2. Banknote Authent. | 1372×4 | 2 | 1.24 | 81.5 | 3767512 | 0.2 | 32.08 | 5484 | 5 | 2 | 2 | 3 |
| 3. Blood transfusion | 748×4 | 2 | 0.73 | 28.2 | 1120504 | 0.13 | 20.17 | 2988 | 68 | 2 | 2 | 10 |
| 4. Brest Cancer | 699×9 | 2 | 1.64 | 31.6 | 2201850 | 0.19 | 27.36 | 6282 | 10 | 2 | 2 | 5 |
| 5. Breast Tissue | 106×9 | 6 | 0.36 | 2.1 | 51039 | 0.03 | 23.73 | 945 | 1 | 6 | 2 | 100 |
| 6. Chronic Kidney Dis. | 400×24 | 2 | 1.56 | 20.2 | 1924800 | 0.28 | 37.38 | 9576 | 100 | 2 | 2 | 70 |
| 7. Dermatology | 366×34 | 6 | 1.88 | 22.9 | 2283474 | 0.43 | 31.77 | 12410 | 18 | 6 | 2 | 10 |
| 8. Ecoli | 336×7 | 8 | 0.98 | 7.8 | 396312 | 0.09 | 31.54 | 2345 | 1 | 8 | 3 | 60 |
| 9. Ind. Liver Patient | 583×10 | 2 | 0.99 | 23.3 | 1702360 | 0.15 | 39.39 | 5820 | 99 | 2 | 3 | 40 |
| 10. Iris | 150× 4 | 3 | 0.24 | 2.9 | 45300 | 0.04 | 17.87 | 596 | 1 | 3 | 5 | 2 |
| 11. Libras Movement | 360×90 | 15 | 4.01 | 50.3 | 5848200 | 1.26 | 92.07 | 32310 | 4 | 15 | 5 | 5 |
| 12. Lung cancer | 32×56 | 3 | 0.61 | 1 | 29568 | 0.05 | 13.34 | 1736 | 1 | 3 | 2 | 20 |
| 13. Parkinsons | 195×22 | 2 | 1.01 | 6.1 | 420420 | 0.13 | 36 | 4268 | 73 | 2 | 3 | 10 |
| 14. Pima Ind. Diabetes | 768×8 | 2 | 1.18 | 36.6 | 2362368 | 0.18 | 37.18 | 6136 | 100 | 2 | 5 | 20 |
| 15. Seeds | 210×7 | 3 | 0.51 | 4.8 | 155085 | 0.06 | 26.77 | 1463 | 1 | 3 | 5 | 1 |

Table 2: Cluster configuration comparison using standard and secure algorithms (differing results highlighted in bold font).

| No. | Standard DBSCAN | | SDBSCAN | | Standard NNC | | SNNC | | Standard k-Means | | Sk-Means | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Num. Clus. | Sil. Coef | Num. Clus. | Sil. Coef | Num. Clus. | Sil. Coef | Num. Clus. | Sil. Coef | Iter. | Sil. Coef | Iter. | Sil. Coef |
| 1. | 6 | 0.472 | 6 | 0.472 | 452 | 1.00 | 452 | 1 .00 | 10 | 0.536 | 10 | 0.536 |
| 2. | 7 | 0.922 | 7 | 0.922 | 21 | 0.895 | 21 | 0.895 | 16 | 0.407 | 16 | 0.407 |
| 3. | **27** | **0.971** | **33** | **0.976** | **34** | 0.999 | **35** | 0.999 | 12 | 0.595 | 12 | 0.595 |
| 4. | **4** | **0.678** | **1** | **0.485** | **108** | **0.903** | **135** | **0.926** | 3 | 0.515 | 3 | 0.515 |
| 5. | 3 | 0.628 | 3 | 0.628 | 105 | 1.00 | 105 | 1.00 | 18 | 0.984 | 18 | 0.984 |
| 6. | 19 | 0.970 | 19 | 0.970 | 243 | 0.999 | 243 | 0.999 | 8 | 0.723 | 8 | 0.723 |
| 7. | **16** | **0.853** | **15** | **0.881** | **32** | **0.919** | **37** | **0.915** | **15** | **0.744** | **9** | **0.713** |
| 8. | 1 | -1 .000 | 1 | -1 .000 | 2 | 0.353 | 2 | 0.353 | **23** | **0.628** | **14** | **0.631** |
| 9. | 7 | 0.789 | 7 | 0.789 | 100 | 0.997 | 100 | 0.997 | 13 | 0.569 | 13 | 0.569 |
| 10. | 2 | 0.722 | 2 | 0.722 | **15** | **0.922** | **16** | **0.927** | 14 | 0.789 | 14 | 0.789 |
| 11. | 11 | 0.715 | 11 | 0.715 | 224 | 0.969 | 224 | 0.969 | 18 | 0.557 | 18 | 0.557 |
| 12. | 1 | 0.053 | 1 | 0.053 | 32 | 1.00 | 32 | 1.00 | **8** | **0.146** | **3** | **0.076** |
| 13. | 5 | 0.829 | 5 | 0.829 | 11 | 0.953 | 11 | 0.953 | 7 | 0.406 | 7 | 0.406 |
| 14. | 4 | 0.691 | 4 | 0.691 | 22 | 0.956 | 22 | 0.956 | 8 | 0.485 | 8 | 0.485 |
| 15. | 7 | 0.852 | 7 | 0.852 | 103 | 0.979 | 103 | 0.979 | 6 | 0.681 | 6 | 0.681 |

ter the clustering. The results obtained are presented in Table 2. From the table, it can be seen that the cluster configurations produced using the proposed secure algorithms were the same in 35 of the 45 cases (same number of clusters). Where the configurations were different, in one case the Sil. Coef. value was the same, in the remaining nine cases the Sil. Coef. value using the secure clustering was better in five of the nine occasions. The reason for the different configurations sometimes obtained was the nature of the proposed OPE scheme; although ensuring that the CDM was secure against Statistical Attacks and Cyphertext Only Attacks by producing different cyphertext for the same plaintext, usage of the OPE scheme did sometimes affect the nature of the clustering because equality is not preserved. Consequently different Sil. Coef. values were sometimes produced in these cases, although in most cases these differences were not significant.

The security of the proposed clustering relies on the security of: (i) Liu's scheme used to encrypt the raw data and (ii) the proposed OPE scheme used to encrypt the CDM. Liu's scheme has been shown to be semantically secure (Liu, 2013); given any cyphertext $C$ within a message $m$ the ability of an adversary to determine any partial information concerning the message will be negligible in terms of the input, hence the scheme is "probably secure". In other words, it will be computationally expensive to derive information concerning the encrypted plaintext given only the cyphertexts and the corresponding encryption public key. This feature makes the proposed method secure against Chosen-Plaintext Attack (CPA) and consequently secure against Knowing Plaintext Attack (KPA) and Cyphertext Only Attack (COA). Moreover, once the data is encrypted and outsourced to a third party data miner, using the proposed approaches, no decryption takes place at the third party side which implies even more security. In terms of the proposed OPE scheme, preserving the order of generated cyphertexts raises a threat of Cyphertext Only Attacks (COAs) that use statistical features, assuming that the data distribution is known. Therefore, the adopted OPE mechanism utilises the concept of "message space splitting" and "non-linear cypher space expansion" to obscure the data distribution in the generated cyphertexts, thus protecting against COAs. Furthermore, the encryption function has a one-to-many mapping feature that produces different cyphertexts for the same plaintext value even when the same keys

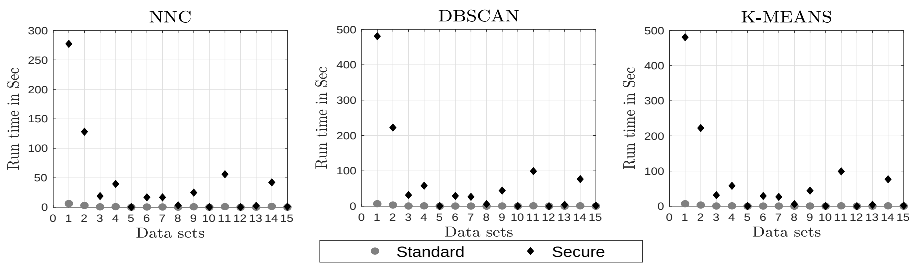

Figure 2: Comparison of run times using standard and secure clustering algorithms (NNC, DBSCAN and k-means).

are used; this makes it harder to derive any information from inspecting the cyphertexts.

## 7 CONCLUSION

In this paper a secure clustering mechanisms has been proposed using the idea of an SCDM. The usage of the SCDM was illustrated in the context of three clustering approaches: (i) Secure DBSCAN, (ii) Secure Nearest Neighbour and (iii) Secure k-Means. The advantages offered by the SCDM are firstly that it is compact and thus appropriate for large datasets; the SCDM requires significantly lower resource in terms of memory and user interaction overhead, compared to (say) the UDM concept presented in (Almutairi et al., 2017). Secondly, the proposed OPE encryption provides an adequate level of security against COA. Thirdly, compared to other proposed secure clustering approaches, data owner participation during the clustering process is zero with respect to SDBSCAN and SNNC, and limited with respect to Sk-Means. Evaluation was conducted, using fifteen UCI datasets, by comparing the operation of the secure clustering algorithms with their standard counterparts. The evaluation demonstrated that the quality of the clustering was similar although not always identical (sometimes better). The reason for the differences was the random parameter δ added to OPE cyphertext which made equality comparison impossible. The runtime, as was to be expected, was greater with respect to secure clustering, but not significantly so. For future work, the authors intend to develop a “Super CDM” where the represented data belongs to two or more data owners who do not wish to share their data in an unencrypted form.

## REFERENCES

Agrawal, R. and Srikant, R. (2000). Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM.

Almutairi, N., Coenen, F., and Dures, K. (2017). K-means clustering using homomorphic encryption and an updatable distance matrix: Secure third party data clustering with limited data owner interaction. In *19th International Conference on Big Data Analytics and Knowledge Discovery*.

Berinato, S. (2015). ThereÕs no such thing as anonymous data. *Harvard Business Review*, February.

Chhinkaniwala, H. and Garg, S. (2011). Privacy preserving data mining techniques: Challenges and issues. In *Proceedings of International Conference on Computer Science & Information Technology, CSIT*, page 609.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Erkin, Z., Veugen, T., Toft, T., and Lagendijk, R. L. (2009). Privacy-preserving user clustering in a social network. In *2009 First IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 96–100. IEEE.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Hazay, C., Mikkelsen, G. L., Rabin, T., and Toft, T. (2012). Efficient RSA key generation and Threshold Paillier in the two-party setting. In *CT-RSA*, pages 313–331. Springer.

Jha, S., Kruger, L., and McDaniel, P. (2005). Privacy preserving clustering. In *European Symposium on Research in Computer Security*, pages 397–417. Springer.

Lichman, M. (2013). UCI machine learning repository.

Lindell, Y. and Pinkas, B. (2002). Privacy preserving data mining. *Journal of cryptology*, 15(3):177–206.

Liu, D. (2013). Homomorphic encrypton for database querying.

Liu, D., Bertino, E., and Yi, X. (2014). Privacy of outsourced k-means clustering. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 123–134. ACM.

Liu, D. and Wang, S. (2013). Nonlinear order preserving index for encrypted database query in service cloud environments. *Concurrency and Computation: Practice and Experience*, 25(13):1967–1984.

Liu, Z., Chen, X., Yang, J., Jia, C., and You, I. (2016). New order preserving encryption model for outsourced databases in cloud environments. *Journal of Network and Computer Applications*, 59:198–207.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Mittal, D., Kaur, D., and Aggarwal, A. (2014). Secure data mining in cloud using homomorphic encryption. In *Cloud Computing in Emerging Markets (CCEM), 2014 IEEE International Conference on*, pages 1–7. IEEE.

Rahman, M. S., Basu, A., and Kiyomoto, S. (2017). Towards outsourced privacy-preserving multiparty DBSCAN. In *Dependable Computing (PRDC), 2017 IEEE 22nd Pacific Rim International Symposium on*, pages 225–226. IEEE.

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

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Samanthula, B. K., Elmehdwi, Y., and Jiang, W. (2015). K-nearest neighbor classification over semantically secure encrypted relational data. *IEEE transactions on Knowledge and data engineering*, 27(5):1261–1273.

Shaneck, M., Kim, Y., and Kumar, V. (2009). Privacy preserving nearest neighbor search. In *Machine Learning in Cyber Trust*, pages 247–276. Springer.

Tong, Q., Li, X., and Yuan, B. (2018). Efficient distributed clustering using boundary information. *Neurocomputing*, 275:2355 – 2366.

Vaidya, J., Clifton, C. W., and Zhu, Y. M. (2006). *Privacy preserving data mining*, volume 19. Springer Science & Business Media.