# Secure Third Party Data Clustering Using $\Phi$ Data: Multi-User Order Preserving Encryption and Super Secure Chain Distance Matrices

Nawal Almutairi[1,2], Frans Coenen[1], and Keith Dures[1]

[1] Department of Computer Science, University of Liverpool, Liverpool, UK
{n.m.almutairi,coenen,dures}@liverpool.ac.uk
[2] Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia
nawalmutairi@ksu.edu.sa

**Abstract.** The paper introduces the concept of $\Phi$-data, data that is a proxy for some underlying data that offers advantages of data privacy and security while at the same time allowing particular data mining operations without requiring data owner participation once the proxy has been generated. The nature of the proxy representation is dependent on the nature of the desired data mining to be undertaken. Secure collaborative clustering is considered where the $\Phi$-data is in the form of a Super Secure Chain Distance Matrices (SSCDM) encrypted using a proposed Multi-User Order Preserving Encryption (MUOPE) scheme. SSCDMs can be produced with respect to horizontal and vertical data partitioning. The DBSCAN clustering algorithm is adopted for illustrative and evaluation purposes. The results indicate that the proposed solution is efficient and produces comparable clustering configurations to those produced using an unencrypted, "standard", algorithm; while maintaining data privacy and security.

**Keywords:** Privacy Preserving Data Mining, Order Preserving and Homomorphic Encryption, $\Phi$-data, Super Secure Chain Distance Matrices.

## 1 Introduction

The resources facilitated through cloud computing have allowed for the delivery of a great variety of services to businesses that would not otherwise be available. One example, and that of interest with respect to this paper, is Data Mining as a Services (DMaaS). The emergence of the potential for third party data analysis using DMaaS has changed the way that data mining is traditionally conducted. However, an issue of significant concern is data privacy and security, a legitimate concern that has served to limit the uptake of DMaaS and which has instigated the research domain of Privacy Preserving Data Mining (PPDM) [2].

Early work on PPDM adopted the idea of Secure Multi-Party Computation (SMPC) which resolved data privacy concerns by precluding any form of data sharing [8, 9, 12]. The idea was for the individual data owners to locally process

their data to produce local statistical characteristics describing their data which could then be used as an input to computation protocols that securely computed global characteristics. Although, to a certain extent, SMPC addressed the problem of data confidentiality, the requirement for data owner participation, as the data mining progressed, resulted in a significant computation and communication overhead and a consequent drain on local resources. These, in turn, effected scalability; thus rendering the approach infeasible for any form of large scale collaborative data mining.

A more desirable PPDM solution, that does not feature the limitations of SMPC, is to entirely outsource the data mining to a third party while maintaining data privacy and security. The idea here is to modify that data either by transforming it or encrypting it, in such a way that data mining activities can still be applied effectively. However, many transformation methods have been shown to adversely affect accuracy. Further, in the collaborative data mining context [3, 4, 6], data owners are all required to transfer their data in the same manner, which makes the solution vulnerable to breaches of privacy. A further criticism is that it has been shown that the data distribution may be reconstructed from the modified data [1]. Cryptography, in turn, provides a substantial guarantee for data privacy. A potential solution is the use of Homomorphic Encryption (HE) schemes 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. One mechanism whereby this can be addressed is with recourse to bespoke SMPC protocols, such as "Yao's Millionaires Problem" protocol as used in [12], or by recourse to data owners as in the case of [15]; in either case undesired communication and/or computational overheads are introduced.

This paper presents a solution to the above in the context of distributed/collaborative data clustering using a third party data miner. More specifically the paper proposes the idea of using a proxy for the real data, an idea referred to as the "$\Phi$-data" concept, where $\Phi$ data is a secure transform of the actual data (not a modification of the data), that supports some specific form of secure data mining that does not entail data owner participation once the proxy has been constructed. The concept of $\Phi$ data can be implemented in a variety of ways. In this paper it is illustrated using Super Secure Chain Distance Matrices (SSCDMs), a data proxy designed for secure collaborative data clustering using the DBSCAN algorithm. The exemplar scenario is that of a number of data owners who wish to produce collaboratively a cluster configuration without sharing their data. A CDM is a 2D matrix $M$ where one dimension represents the set of records in a dataset (-1) and the other the set of attributes. Each cell $M_{i,j}$ holds the distance between the $j$th attribute value in $i$the record and the value of the same attribute in $i + 1$th record. A SCDM is then an encrypted CDM. This paper also proposes an order preserving encryption scheme, Multi-User Order Preserving Encryption (MUOPE), suited to encrypting CDMs. A SSCDM is then the union set of two or more SCDMs. The SSCDM construction process is

facilitated by a Semi-honest Third Party (STP). Once complete it can be passed to a third party data miner who can produce a cluster configuration without requiring further data owner participation and without ever having had access to the original datasets held by individual data owners.

## 2 Related Works

This section presents a review of previous work directed at collaborative secure data clustering. The focus is on DBSCAN clustering; the clustering mechanism used to illustrate the solution proposed in this paper. Generally, the main challenge in collaborative data clustering is that of maintaining data confidentiality during processing without adversely affecting the calculation accuracy; thus in the case of DBSCAN when calculating distances between records and when comparing such distances against a threshold $\epsilon$. As noted in the introduction, the proposed solutions can be categorised as being founded on either: (i) SMPC or (ii) secure outsourcing to a third party who has the permissions to carry out the required calculations. Both approaches are considered in further detail below.

The fundamental idea of SMPC is for the collaborating parties to jointly compute functions concerning their data while maintaining the privacy of their data. The nature of these functions, and the protocols used to calculate them, depend on the nature of the application. There are a number of examples where multiparty DBSCAN has been implemented using SMPC [8, 9, 12] where: (i) distance between data records is calculated using either a "secure multi-party scalar product" protocol or the homomorphic properties of a HE scheme, and (ii) secure comparison was achieved using either "Yao's Millionaires' Problem" protocol (YMPP) [18] or Cachin's scheme [5]. However, the solutions presented in [8, 9, 12] all entailed a significant computational overhead and consequently they were only suited to two party collaborative data clustering. In terms of security, involving data owners in the calculation of distances, and the comparison of distances, gives rise to the potential for "overlapping attacks" where a non-honest participant uses knowledge of their local data, and the computation results, to identify intersections with the data held by other parties and then uses this information to estimate records held by other parties. In the specific context of SMPC-based DBSCAN clustering a further security risk is that the total number of points within the $\epsilon$-radius is revealed to all participants. Theses limitations render SMPC-based solution inadequate for many instances of DMaaS.

The alternative solution is to outsource the data and data analysis to a third party data miner. In this case privacy is preserved by modifying the data in some way. Two well documented modification techniques are data perturbation and encryption. The basic idea of data perturbation is to distort individual values by adding additive or multiplicative noise, or by applying some form of randomisation, so that the statistical characteristics of the data are retained. From the literature a variety of perturbation methods have been proposed, both two party [2] and multiparty [3, 4, 6]. However, it has been demonstrated that the higher the level of security provided by the perturbation the worse the final

data mining result, this is especially the case where each party applies a local perturbation method. This is why in [3] it was proposed that all parties use the same perturbation method so that an acceptable accuracy is achieved. Whatever the case, perturbation has two major disadvantages. The first is that the final results are adversely affected. The second is that the original data distribution can be reconstructed from the perturbed data [1]. Homomorphic Encryption provides an effective alternative that does not feature these disadvantages. In [15] a HE scheme was used to encrypt multiple source data before outsourcing to a third party data miner who then utilised the HE properties to calculate the required distances. However, 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 $\epsilon$ value.

In [17] a mechanism is presented for applying DBSCAN in a secure distributed manner that combines the SMPC idea with the usage of a third party data miner. The basic idea is for each party to first apply DBSCAN to their local data and then to share the resulting boundary points and cluster labels using a third party data miner. The data miner's role is then to determine global boundary points which are then used by the individual data owner to update their local clusters. However, the local boundary points are sent to the data miner in plaintext form, which presents a security threat.

## 3 Paillier Homomorphic Encryption

Before considering the proposed secure DBSCAN clustering algorithm, and the SSCDM concept, in detail, Paillier Homomorphic Encryption, utilised in the context of the proposed MUOPE scheme, is briefly described in this section. The Paillier encryption scheme [14] is an additive, probabilistic and asymmetric HE scheme that encodes a plaintext value $m$ to a cyphertext value $c$ using the equation $c = g^m r^N \pmod{N^2}$ where $N$ is the Rivest-Shamir-Adleman (RSA) modulus, $g$ is a non-zero integer of order divisible by $N$; and $r$ is a random number, $r \in \mathbb{Z}_N$, used to ensure the probabilistic feature of the scheme. The scheme has an additive homomorphic feature that maps plaintext addition ($+$) to cypher multiplication ($\otimes$) as given in Eq. 1, where $a$, $b \in \mathbb{Z}_N$.

$$E(a + b) = E(a) \otimes E(b) \pmod{N^2} \tag{1}$$

The decryption function decodes $c$ to the original plaintext value $m$ using Algorithm 1 where: $LCM$ is a *Least Common Multiple* function, $L$ is a function defined as $L(x) = \frac{x-1}{N}$, $(N, g)$ is the public key and $(\lambda, \mu)$ is the secret key.

**Algorithm 1** Paillier decrypt function

1: **procedure** DECRYPT($c$)
2: $\lambda = LCM(p-1, q-1)$ ▷ $p$ and $q$ are two prime numbers
3: $\mu = (L(g^\lambda \pmod{N^2}))^{-1} \pmod{N}$
4: $m = L(c^\lambda \pmod{N^2})\mu \pmod{N}$
5: **Exit** with $m$

## 4 The Multi-User Order Preserving Encryption (MUOPE) Scheme

The CDM, generated by individual data owners and described further in Section 5 below, is essentially a set of linear equations that might support the undesirable re-engineering of the original data distribution. Therefore, to prevent such re-engineering, while still permitting comparison of distances, in this paper it is proposed that CDMs are encrypted to give Secure CDMs (SCDMs) using a bespoke encryption scheme, the MUOPE scheme.

The idea of the proposed MUOPE scheme is founded on the scheme presented in [13] which was directed at encrypting data in such a way that the order of data items was preserved; however the scheme was not applicable to data from multiple sources. The main objective of the proposed MUOPE scheme is to encrypt two or more SCDMs, that are to be combined into a single SSCDM, in such a way that any data distribution that might exist in the generated cyphertexts is entirely obscured. To this end, the concepts of *message space splitting* and *non-linear cypher space expansion* were adopted so that the third party data miner could have access to the ordering of distances between records and not the original CDM distance values themselves.

In the proposed MUOPE scheme a Semi-honest Third Party (STP) is used to act as a mediator between $u$ participating parties (data owners), $P = \{p_1, \ldots, p_u\}$. The STPs role is to: (i) derive MUOPE encryption parameters and (ii) manage the SSCDM generation process. The STP starts by determining the required "interval" of message space $M = [l, h)$ and the associated expanded "interval" of cypher space $C = [l', h')$ where $h$ is the maximum interval boundary and $l$ is the minimum interval boundary in such a way that $|C| \gg |M|$. The STP then randomly splits the message space into $t$ consecutive intervals, where $t$ is a random number, to give $M = \{m_1, \ldots, m_t\}$, where $m_i = [l_i, h_i)$; as demonstrated in Figure 1. The message space interval boundaries are then sent to the data owners.

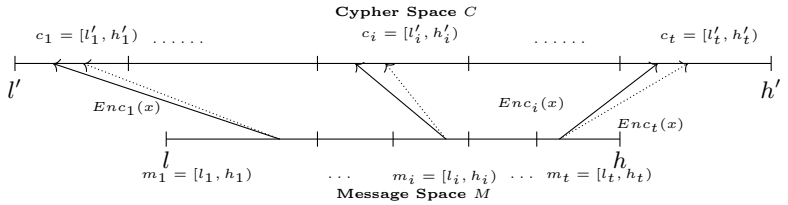



Fig. 1: Message and expanded cypher space splitting

To generate the cypher space intervals, the STP needs to know how many distances fall into each interval. The STP does this by creating a list $V$ comprised of $t$ items $\{v_1, \ldots, v_t\}$ where each item in a list will eventually hold a count of the number of distances that fall in each interval. The STP populates $V$ with a random set of values and encrypts it using the Paillier encryption from Section 3. Thus the STP is also responsible for generating Paillier public-private key pairs. The Paillier encrypted list $V'$ is then sent, together with the Paillier public key,

to the first data owner $p_1$ who then updates $V'$ with their data density, for each interval, using the additive feature of the Paillier scheme. The updated list $V'$ is then sent in turn to the remaining data owners. The last party $p_u$ will return $V'$ to the STP who decrypts it and subtracts the original values used to populate $V$. The results (the data density for each interval) is then used to dimension the cypher space $C$ to give $C = \{c_1, \ldots, c_t\}$, in such a way that the length of each interval $c_i$ is determined according to the density of the data in the corresponding message space interval $m_i$. The aim is to ensure that message space intervals with a high "density" correspond to larger (expanded) cypher space intervals, and vice versa. The cypher space boundaries are then sent to the data owners. The intervals boundaries represent MUOPE encryption keys.

On receiving the MUOPE encryption keys, from the STP, the data owners encrypt their individual CDMs to give SCDMs, the process for this is discussed in Section 5 below. The encryption is conducted as indicated by Eq. 2, where: $i$ represents the ID number of an "interval" within which a distance $dist$ is contained; $l_i$ and $h_i$ are the boundaries for the $i$th message space interval; and $l'_i$ and $h'_i$ are the boundaries for the $i$th cypher space interval. The variable $\delta_i$ is a random number sampled from the range 0 to $Sens \times Scale$ where $Sens$ is defined, as in [11], as the data sensitivity value that represents the minimum distance between plaintext values.

$$Scale = \frac{(l'_i - h'_i)}{(l_i - h_i)}, \quad Enc(dist) = l'_i + (Scale \times (dist - l_i)) + \delta_i \tag{2}$$

The STP then commences the SSCDM generation process. How this is done depends on whether we have horizontally or vertically partitioned data and is described in Section 5. Once the SSCDM has been calculated, the STP passes this on to the third party data miner. The STPs role is now complete.

## 5 The Super Secure Chain Distance Matrix (SSCDM)

A SCDM is a mechanism for realising the envisioned $\Phi$-data concept in the context of collaborative data clustering, specifically DBSCAN clustering. A SCDM allows for secure data comparison in the absence of the original data. A Super SCDM (SSCDM) is then a combination of a number of SCDMs generated by individual data owners. In the following subsections the SSCDM generation process is given in further detail. The generation of SCDM is presented in Subsection 5.1. This is followed, Subsections 5.2 and 5.3, with discussion of the "binding" process to give SSCDM given either horizontally or vertically partitioned data.

### 5.1 The Secure Chain Distance Matrix (SCDM)

A CDM is a 2D matrix that holds the distances (differences) between each attribute value within a record $i$ and the corresponding attribute value in the following record $i + 1$ according to whatever ordering is featured in the dataset $D$. The matrix thus measures $(|R| - 1) \times |A|$, where $|R|$ is the number of records

in $D$ and $|A|$ is the size of the attribute set $A$. A SCDM is then an encrypted CDM. The SCDM is generated in two steps: (i) CDM calculation and (ii) CDM encryption to arrive at a SCDM. Algorithm 2 gives the CDM calculation process. The algorithm commences by dimensioning the desired CDM (line 2) which is then populated (lines 3 to 5) by calculating the distances between the values for attributes in the $i$th and $i+1$th data records (line 5).

**Algorithm 2** Chain Distance Matrix Calculation

```
1: procedure CDMCalculation(D)
2:     CDM = ∅ array of |R| − 1 rows and |A| column
3:     for i = 1 to i = |R| − 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
```

The next step is to encrypt the calculated CDM to give a SCDM. To this end the MUOPE scheme presented in Section 4 was used. The key feature of the resulting SCDM is that a third party has access to the distance value ordering, but not the actual distance values. In addition, the chain feature of SCDMs allows a number of SCDMs to be "bound" to form a SSCDM that then permits similarity calculations between data records, possibly owned by different parties, without involving the data owners. The similarity between a record $r_x$ and a record $r_y$ (where $x < y$), is calculated according to Eq. 3. In the case of $x = y$ the distance will clearly be 0.

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

### 5.2 SSCDM for Horizontal Data Partitioning

Horizontally distributed data is where each partition conforms to the same set of attributes $A$, but features different records; in other words the global dataset $D$ has been partitioned by dividing it up "horizontally". To "bind" two SCDMs, $SCDM_i$ and $SCDM_{i+1}$, representing horizontally partitioned data, belonging to two data owners $p_i$ and $p_{i+1}$ respectively, an additional "pivot" record, with $|A|$ attributes, needs to be inserted between the two SCDMs, recording the differences between attribute values in the last record in $D_i$ owned by $P_i$, and the first record in $D_{i+1}$ owned by $P_{i+1}$. The process is as shown in Algorithm 3. The inputs to the process are the two SCDMs ($SCDM_i$ and $SCDM_{i+1}$) and the global SSCDM accumulated so far. The algorithm commences with the STP randomly generating a record, $R = \{r_1, r_2, \ldots, r_{|A|}\}$ and encrypting this using the MUOPE scheme (line 2); this is then sent to $p_i$ and $p_{i+1}$. Data owner $p_i$ calculates the distances between the MUOPE cypher of the last record in its dataset and the content of $R$ to give a record $C_1$ (line 3); whilst data owner $p_{i+1}$ calculates the distances between the content of $R$ and the MUOPE cypher of the first record in its dataset to give a record $C_2$ (line 4). Both $C_1$ and $C_2$

are returned to the STP who calculates the pivot record, $pivot = C_1 + C_2$. The pivot record is then used to bind $SCDM_i$ and $SCDM_{i+1}$ (line 6) and append this to the SSCDM so far. The process repeats with $SCDM_{i+1}$ and $SCDM_{i+2}$ and continues until the entire SSCDM has been generated.

**Algorithm 3** Horizontal binding process

1: **procedure** HorizontalBinding($SCDM_i, SCDM_{i+1}, SSCDM$)
2: $R = \{r_1, \ldots, r_{|A|}\}$ ▷ Encrypted using MUOPE
3: $C1$ = Distances between last record in $D_i$ and $R$
4: $C2$ = Distances between $R$ and first record in $D_{i+1}$
5: $Pivot = C_1 + C_2$
6: $SSCDM = concatenate(SSCDM, SCDM_i, Pivot, SCDM_{i+1})$
7: **Exit** with **SSCDM**

### 5.3 SSCDM for Vertical Data Partitioning

Vertically distributed data is where each partition features the same set of records but a specific sub-set of attributes from a global set of attributes $A$; the global dataset has been partitioned by being divided up "vertically". The binding process for vertically partitioned data is as shown in Algorithm 4. The inputs are: a SCDM, $SCDM_i$ belonging to data owner $p_i$, and the SSCDM so far. On start up the SSCDM so far will simply be $SCDM_1$, belonging to data owner $p_1$. The algorithm operates by simply appending records to one another, (line 2), there is no need for a pivot record. As before, the process will continue until the entire SSCDM has been generated.

**Algorithm 4** Vertical binding process

1: **procedure** VerticalBinding($SCDM_i, SSCDM$)
2: $SSCDM = concatenate(SSCDM, SCDM_i)$
3: **Exit** with **SSCDM**

## 6 Secure DBSCAN (SDBSCAN)

The SDBSCAN clustering is conducted by the third party data miner following a processes very similar to the standard DBSCAN [7]. The pseudo code is given in Algorithm 5. The inputs are the SSCDM received from the STP and the desired density parameters, $MinPts$ and $\epsilon'$, that are agreed by the participating parties. The $\epsilon$ value is encrypted using the proposed MUOPE scheme to give $\epsilon'$ so that the third party data miner does not have the real radius value. The algorithm uses a "virtual" dataset $VR$ where the indexes refer to the data held by data owners, thus $VR = \{vr_1, vr_2, \ldots, vr_{|SSCDM|+1}\}$. The order of the data indexes matches the order used to bind the SCDMs. For example, indexes 0 to $|SCDM_1 + 1|$ represent the $p_1$ virtual dataset. The algorithm commences by creating the ordered set $VR$, creating an empty set of clusters $C$ and setting the number of clusters so far to 1 (line 2). The set $VR$ is then processed. For each "virtual" record

$vr_i \in VR$ that has not been previously assigned to a cluster, is "unclustered", the set $S$ is determined. The set $S$ is the $\epsilon$-neighbourhood of $vr_i$ and comprises the set of record IDs in $VR$ whose distance from $vr_i$ is less than or equals to $\epsilon'$. The set is determined by calling the *RegionQuery* procedure (line 5) where the SSCDM is used to determine the overall distances between records (see Eq. 3). If the number of records in $S$ is greater than $MinPts$ the density requirement is satisfied thus $vr_i$ is marked as "clustered" and considered to represent a new cluster $C_k$ (lines 6 to 8). This cluster is then expanded by considering the points in $S$ using the *ExpandCluster* procedure called in line 9. The inputs to the *ExpandCluster* procedure are: the cluster $C_k$ so far, the set $S$, SSCDM and the density parameters $MinPts$ and $\epsilon'$. 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 $\epsilon$-neighbourhood $S_2$ for the record. If the size of $S_2$ is greater than $MinPts$ we call the *ExpandCluster* procedure again and so on until all the "virtual" records in $VR$ are processed at which point the algorithm will exit with the cluster configuration $C$. For the purpose of data privacy each participating party will receive their own data clustering results.

## 7 Experimental Evaluation

This section reports on the evaluation of the $\Phi$-data concept in the context of MUOPE and SSCDM as implemented with respect to SDBSCAN. The objectives of the evaluation were to consider the proposed approach in terms: (i) data owners participation, (ii) clustering efficiency, (iii) clustering accuracy, (iv) security and (v) scalability. Two different types of data were used, synthetic data and data from the UCI machine learning repository [10].

### 7.1 Data Owner Participation

Individual data owner participation was measured in terms of the runtimes (ms) required to: (i) generate CDMs (CDM Gen.), (ii) encrypt CDMs (CDM Enc) and (iii) calculate the data density required to dimension the MUOPE cypher space (Dens Cal). Experiments were conducted using a sequence of ten synthetic datasets increasing in size from 1000 to $10,000$ records, in steps of $1,000$; the number of attributes ($|A|$) was kept constant at 125. The results are presented in Figure 2. Inspection of the figure indicates that, as was expected, time complexity increases in a linear manner as the number of records ($|R|$) increases. For example, in the case of the $|R| = 1$K dataset, the recorded runtimes for *CDM Gen* and *Dens Cal* are both 163ms, while for the 10K dataset the recorded run times were 445ms and 493ms respectively. The time complexity for *CDM Gen* is O($|R| - 1 \times |A|$). The *CDM Enc* is slightly higher; the 1K required 0.5sec which increased to 2.4sec for 10K. What is noteworthy is that, regardless of the number of records considered, the run times are not significantly high; hence the amount of data owner participation can be argued to be minimal. Recall

**Algorithm 5** Secure DBSCAN clustering algorithm

**procedure** SDBSCAN( $SSCDM$, $MinPts$, $\epsilon'$)
  $C = \emptyset$, $VR$ =list of record IDs, $k = 1$
  **for** $\forall\ vr_i \in VR$ **do**
    **if** $vr_i$ *is Unclustered* **then**
      $S$ = RegionQuery($vr_i, \epsilon', SSCDM$)
      **if** $|S| > MinPts$ **then**
        mark $vr_i$ as *clustered*
        $C_k = vr_i$ (new cluster)
        $C_k$ = ExpandCluster($C_k$,$S$, $SSCDM$,$\epsilon'$,$MinPts$)
        $C = C \cup C_k$
        $k = k + 1$
  **Exit** with **C**
**procedure** EXPANDCLUSTER($C$,$S$, $SSCDM$,$\epsilon'$,$MinPts$)
  **for** $\forall\ vr_i \in S$ **do**
    **if** $vr_i$ *is Unclustered* **then**
      mark $vr_i$ as *clustered*
      $C = C \cup vr_i$
      $S_2$ = RegionQuery($vr_i, \epsilon', SSCDM$)
      **if** $|S_2| > MinPts$ **then**
        $C$ = ExpandCluster($C$,$S_2$, $SSCDM$,$\epsilon'$,$MinPts$)
  **Exit** with **C**
**procedure** REGIONQUERY($Index$, $\epsilon'$, $SSCDM$)
  $N_\epsilon = \emptyset$
  **for** $\forall\ vr_j \in VR$ **do**
    $distance = Sim(SSCDM, Index, j)$ ▷ (Eq. 3)
    **if** $distance \leq \epsilon'$ **then**
      $N_\epsilon.add(j)$
  **Exit** with $N_\epsilon$

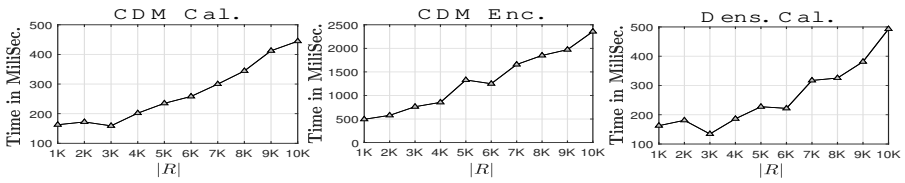



Fig. 2: Time required (ms) for data owner participation in term of number of records in a data owner's local dataset

that once the SSCDM has been generated no further data owner participation is required other than instructing the third party data miner to undertake specific clustering exercises.

### 7.2 Clustering Efficiency

A comparison of the runtimes required to cluster data using standard (unencrypted) DBSCAN and the proposed SDBSCAN is given in columns 6 and 9 of Table 1 which gives clustering outcomes using fifteen UCI datasets [10]. Note

that runtimes for standard DBSCAN are reported in milliseconds (ms), while runtimes for SDBSCAN are reported in seconds (sec). The $MinPts$ and $\epsilon$ values reported in the table are randomly selected; in practice these are prescribed by the data owners. From the table, it can be seen that reported runtimes were larger for SDBSCAN than in the case of the standard approach. The difference is due to the utilisation of SSCDMs. Note that the bigger the dataset the larger the SSCDM, hence the greater the time required to process the SSCDM to determine record similarity. However, inspection of the recorded results indicates that usage of SSCDMs did not introduce an unreasonable overhead.

| DataSet | $MinPts$ | $\epsilon$ | Standard DBSCAN | | | Secure DBSCAN | | |
|---|---|---|---|---|---|---|---|---|
| | | | Num. Clus. | Sil. Coef. | Exec. Time **(ms)** | Num. Clus. | Sil. Coef. | Exec. Time **(sec)** |
| 1. Arrhythmia | 2 | 600 | 6 | 0.472 | 187.16 | 6 | 0.472 | 367.24 |
| 2. Banknote Auth. | 2 | 3 | 7 | 0.922 | 686.37 | 7 | 0.922 | 254.25 |
| 3. Blood Trans. | 2 | 10 | **27** | **0.971** | 54.20 | **33** | **0.976** | 4.73 |
| 4. Breast Cancer | 2 | 5 | **4** | **0.678** | 61.64 | **1** | **0.485** | 9.60 |
| 5. Breast Tissue | 2 | 100 | 3 | 0.628 | 3.93 | 3 | 0.628 | 0.27 |
| 6. Chronic kidney | 2 | 70 | 19 | 0.970 | 57.36 | 19 | 0.970 | 12.64 |
| 7. Dermatology | 2 | 10 | **16** | **0.853** | 19.46 | **15** | **0.881** | 5.86 |
| 8. Ecoli | 2 | 60 | 1 | -1 .000 | 45.81 | 1 | -1 .000 | 4.58 |
| 9. Ind. Liver Patient | 3 | 40 | 7 | 0.789 | 120.48 | 7 | 0.789 | 25.53 |
| 10. Iris | 5 | 2 | 2 | 0.722 | 11.89 | 2 | 0.722 | 0.33 |
| 11. Libras Move. | 5 | 5 | 11 | 0.715 | 61.75 | 11 | 0.715 | 120.62 |
| 12. Lung Cancer | 2 | 20 | 1 | 0.053 | 0.32 | 1 | 0.053 | 0.01 |
| 13. Parkinsons | 3 | 10 | 5 | 0.829 | 14.84 | 5 | 0.829 | 4.06 |
| 14. Pima Disease | 5 | 20 | 4 | 0.691 | 221.87 | 4 | 0.691 | 30.15 |
| 15. Seeds | 5 | 1 | 7 | 0.852 | 16.90 | 7 | 0.852 | 1.43 |

Table 1: Cluster Configuration for Standard and Secure DBSCAN (differing results highlighted in bold font)

### 7.3 Clustering Accuracy

Clustering accuracy was measured by comparing the clustering configurations obtained using SDBSCAN with those obtained using standard DBSCAN. The intuition was that the secure algorithm should produce comparable configurations to those produced using the standard algorithm; if so the secure algorithm could be said to be operating correctly. The measure used was the established Silhouette Coefficient (Sil. Coef.) [16]; a value between $-1$ and 1, the closer the values is to 1 the better the clustering. The Sil. Coef. values obtained are presented in columns 5 and 8 of Table 1, and the number of generated clusters in Columns 4 and 7. From the table, it can be seen that the cluster configurations produced using the proposed SDBSCAN were the same in 12 out of 15 cases, and slightly different in three cases (Blood Trans., Breast Cancer and Dermatology). It is interesting to note that in two of these three cases (Blood Trans. and Dermatology) SDBSCAN produced better Sil. Coef. values. The reason for the differences

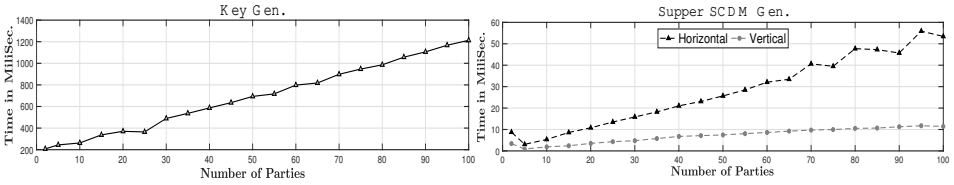
was because the proposed MUOPE scheme produced different cyphertexts for the same plaintext value, which meant that "equality" was not supported; thus if a dataset had many identical values these would result in different cyphertexts which in turn would effect the nature of the clustering (sometimes in a positive manner). This feature of the MUOPE scheme was introduced to hide data value frequency so as to prevent statistical attacks that can be instigated when attackers have knowledge of the data distribution (frequency).

### 7.4 Security Analysis

Security was evaluated by identifying the potential attacks that may threaten the proposed secure data clustering. In the proposed solution, data preservation relies on the $\Phi$ data concept and the security of the MUOPE scheme used to encrypt the SSCDMs. The concept of $\Phi$ data, prevents the data from being confided (in any form) to a third party data miner or shared with any other participants. Therefore, the $\Phi$ data concept precludes any form of attack directed at the actual data, including the overlapping attack possible with respect to other solutions (see Section 2). The only data proxy received by the third party data miner is the SSCDM; there is no further data owner involvement. Hence, the only potential form of attack is Cyphertexts Only Attacks (COAs) that may occur if an adversary somehow has access to a SSCDM. As a countermeasure to COAs the proposed MUOPE was designed to reduce information leakage in cyphertexts by avoiding the deterministic feature that is usually used in COAs. More specifically the MUOPE scheme uses an encryption function that generates different cyphertexts for the same plaintext values on each occasion that the encryption function is applied; this feature makes inferences using COAs harder. COAs are more likely to succeed when attackers have a background knowledge of the data distribution, or frequency, of the original data values. Knowledge associated with the ordering features of some order preserving encryption schemes might allow an adversary to infer the ranges containing dense data. Alternatively, frequency analysis could allow attackers to highlight cyphertexts with the same frequency as plaintexts (if such plaintexts were available) and then identify cyphertexts that have the same frequency. However, this will not be possible in the case of the MUOPE scheme, which incorporates *message space splitting*, *non-linear cypher space expansion* and a *one-to-many encryption function*, that serves to obscure the statistical features of the generated cyphertexts.

### 7.5 Scalability

The scalability of the proposed SDBSCAN approach, founded on the concept of $\Phi$ data realised using SSCDMs, encrypted using the proposed MUOPE scheme, was evaluated by considering the effect on time complexity as the number of data owners (participants) increased. In the proposed approach data owner collaboration occurs when generating: (i) MUOPE encryption keys (Key Gen.) and (ii) SSCDMs (Super SCDM Gen.). For the evaluation a sequence of experiments was conducted where the number of participants was increased from 10 to 100




(a) (b)

Fig. 3: Runtime to generate OPE keys and construct SSCDMs as the number of participants (data owners) increases

in steps of 5 (for completeness experiments using two and four participants were also conducted). A synthetic dataset, comprised of 7000 records and 125 attributes, was equally distributed across the parties in each case. The recorded total runtime results are presented in Figure 3. From the figure it can be seen that the overall time required to generate the encryption keys was negligible; even in the 100 participants case the recorded runtime was $1,213$ms. The scalability, as demonstrated by the reported results, indicates that the MUOPE scheme has potential benefits for many other forms of DMaaS and collaborative PPDM.

With respect to the overall time required to generate a SSCDM the results reported in Figure 3(b) show that, as expected, the time required will increase linearly with the number of participants. Recall that the usage of SSCDMs allows collaborative data clustering to be implemented without requiring extensive communication between participants when calculating distances between data points as in the case of [8, 9, 12]. From the figure it can also be seen that vertical partitioning produced the best performance because we are simply "bolting" one SCDM to another.

## 8 Conclusion and Future Work

This paper has proposed a novel solution for third party privacy preserving collaborative data clustering using the concept of $\Phi$-data and SSCDMs encrypted using MUOPE. The $\Phi$-data concept obviates the need for any form of data sharing between data owners and/or a third party data miner. The proposed approach offers three main advantages. Firstly, the SSCDM proxy representation allows multiple data sources to be compared without data owner involvement or any communication overhead. Secondly, the MUOPE scheme encrypts SSCDMs, in such a way that protection against Cyphertexts Only Attacks (COAs) is provided (other forms of attack are precluded). Thirdly, the secure data clustering is entirely delegated to a third party data miner (over encrypted data), no data owner participation is required. The accuracy of the clustering produced using the SDBSCAN approach was shown to be compatible with those produced using standard DBSCAN, whilst the time complexity was not significantly greater. It was also shown that the proposed approach was readily scalable. For future work, the authors intend to investigate the utility of SSCDMs with respect to alternative clustering algorithms and other data mining techniques.

## References


1. Aggarwal, C.C., Philip, S.Y.: A general survey of privacy-preserving data mining models and algorithms. In: Privacy-preserving data mining, pp. 11–52. Springer (2008)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. SIGMOD Rec. **29**(2), 439–450 (May 2000)
3. Anikin, I.V., Gazimov, R.M.: Privacy preserving DBSCAN clustering algorithm for vertically partitioned data in distributed systems. In: IEEE International Siberian Conference on Control and Communications. pp. 1–4. IEEE (2017)
4. Bhaduri, K., Stefanski, M.D., Srivastava, A.N.: Privacy-preserving outlier detection through random nonlinear data distortion. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **41**(1), 260–272 (2011)
5. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: Proceedings of the 6th ACM conference on Computer and communications security. pp. 120–127. ACM (1999)
6. Chen, K., Liu, L.: Privacy-preserving multiparty collaborative mining with geometric data perturbation. IEEE Transactions on Parallel and Distributed Systems **20**(12), 1764–1776 (2009)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd. vol. 96, pp. 226–231 (1996)
8. Jiang, D., Xue, A., Ju, S., Chen, W., Ma, H.: Privacy-preserving DBSCAN on horizontally partitioned data. In: IEEE International Symposium on Medicine and Education. pp. 1067–1072. IEEE (2008)
9. Kumar, K.A., Rangan, C.P.: Privacy preserving DBSCAN algorithm for clustering. In: International Conference on Advanced Data Mining and Applications. pp. 57–68. Springer (2007)
10. Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml
11. Liu, D., Wang, S.: Nonlinear order preserving index for encrypted database query in service cloud environments. Concurrency and Computation: Practice and Experience **25**(13), 1967–1984 (2013)
12. Liu, J., Xiong, L., Luo, J., Huang, J.Z.: Privacy preserving distributed DBSCAN clustering. Trans. Data Privacy **6**(1), 69–85 (Apr 2013)
13. Liu, Z., Chen, X., Yang, J., Jia, C., You, I.: New order preserving encryption model for outsourced databases in cloud environments. Journal of Network and Computer Applications **59**, 198–207 (2016)
14. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 223–238. Springer (1999)
15. Rahman, M.S., Basu, A., Kiyomoto, S.: Towards outsourced privacy-preserving multiparty DBSCAN. In: 22nd IEEE Pacific Rim International Symposium on Dependable Computing. pp. 225–226. IEEE (2017)
16. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20**, 53–65 (November 1987)
17. Tong, Q., Li, X., Yuan, B.: Efficient distributed clustering using boundary information. Neurocomputing **275**, 2355 – 2366 (2018)
18. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science. pp. 160–164. IEEE (1982)