K-Means Clustering Using Homomorphic Encryption and an Updatable Distance Matrix: Secure Third Party Data Clustering with Limited Data Owner Interaction

Nawal Almutairi, Frans Coenen, and Keith Dures

Department of Computer Science, University of Liverpool, UK
{n.m.almutairi,coenen,dures}@liverpool.ac.uk

Abstract. Third party data analysis raises data privacy preservation concerns, therefore raising questions as to whether such outsourcing is viable. Cryptography allows a level of data confidentiality. Although some cryptography algorithms, such as Homomorphic Encryption (HE), allow a limited amount of data manipulation, the disadvantage is that encryption precludes any form of sophisticated analysis. For this to be achieved the encrypted data needs to coupled with additional information to facilitate third party analysis. This paper proposes a mechanism for secure k-means clustering that uses HE and the concept of an Updatable Distance Matrix (UDM). The mechanism is fully described and analysed. The reported evaluation shows that the proposed mechanism produces identical clustering results as when "standard" k-means is applied, but in a secure manner. The proposed mechanism thus allows the application of clustering algorithms to encrypted data while preserving both correctness and data privacy.

Keywords: Homomorphic Encryption, k-means clustering, Privacy preserving data mining, Secure k-means clustering

1 Introduction

Data mining techniques have been exploited to improve quality of service and guide decision makers with respect to many application domains. However, the exponential growth in data availability often makes in-house data analysis impractical and expensive. This has fuelled the idea of Data Mining as a Service (DMaaS); third parity analysis using cloud computing facilities [3]. This also opens the door to collaborative data mining where a number of data owners pool their data so as to gain some (for example commercial) mutual advantage. Although DMaaS reduces the cost of managing and analysing data, and facilitates collaborative data mining, it introduces several challenges, the most significant of which is data privacy preservation. Hence the research domain of Privacy Preserving Data Mining (PPDM) [1, 9, 16]. The typical approach to PPDM is to use some form of data transformation, applied either to the entire data set (for example data perturbation) or selective sensitive data attributes (for example value anonymisation) [4]. In perturbation, individual attributes are distorted or randomised by adding noise, whereas data anonymity relies on modifying the data by replacing sensitive values with proxy values or removing the values all together. However, although perturbed data may look quite different from the original data, an adversary may take advantage of properties such as correlations and patterns in the original data to identify the real values [16]. It has also been observed that data cannot be 100% anonymised [2]. In addition the anonymisation/perturbation of data may adversely effect the quality of the analysis [1, 16].

Data privacy and security can be substantially guaranteed when data is encrypted. However, standard forms of encryption do not support data mining activities, which typically require the comparison of records to determine their similarity (or otherwise). Several encryption schemes have been developed to enforce privacy and confidentiality while at the same time supporting some manipulation of encrypted data. Searchable Encryption (SE) [17] is an example of an encryption scheme that preserve data privacy while supporting different kinds of search operations over encrypted data. However, such search capabilities still do not readily support data mining activities. One potential solution is Homomorphic Encryption (HE), a family of emerging encryption mechanism that supports a limited number of simple mathematical computations. The precise nature of the mathematical capabilities supported by HE schemes is dependent on the nature of the adopted scheme. However, there is no HE scheme that provides all the mathematical operations that we might want so as to perform data mining activities; for example the similarity calculation operations required in the case of data clustering [12, 15]. The solutions that have been proposed to date all entail a significant element of data owner participation so that operations not supported by the selected HE mechanism can be conducted using unencrypted data by the data owner. The degree of data owner participation is such that the only reason for using a third party is to support cooperative data mining. For example in the context of k-means cooperative clustering, as described in [7] and [13], local cluster centroids are shared on each iteration so that global centroids can be calculated on each k-means iteration, the majority of the work is conducted by the data owners. In the case of a single data owner there seems little point in conducting third party k-means clustering if most of the work needs to be conducted by the data owner. The challenge is therefore to limit the amount of data owner participation.

Given the above, this paper presents a k-means clustering mechanism that limits interaction with data owners by using the concept of an *Updatable Distance Matrix* (UDM). More specifically the idea is that data is stored, using third party storage, in an encrypted form, together with an associated UDM. The data owner can then request the data to be clustered, specifying the number of clusters k, as required. For the clustering to operate correctly the UDM matrix needs to be updated on each iteration of the k-means algorithm. To do this the differences between the centroids of iteration i and $i-1$ are calculated in encrypted form and returned to the data owner for decryption, after which the decrypted differences (as real numbers) are returned to the third party in the form of a *Shift Matrix* (SM) which is then used to update the UDM ready for iteration i; this is the only data owner participation that is required. Using this mechanism, as will be made clear in the paper, the amount of data owner participation is significantly reduced (by a factor of n, where n is the number of records in the dataset). Furthermore, although the idea is presented using the well known k-means clustering algorithm [12], it can, with some adjustment, be extend to encompass alternative clustering mechanisms. It should also be noted that the nature of the encryption used is Liu's scheme [10], which supports addition and subtraction of cypher texts, and division and multiplication of cypher texts by real numbers.

The rest of this paper is organised as follows, in Section 2 a review of related work is presented. Section 3 provides the fundamental background concerning standard k-means clustering and Liu's Homomorphic Encryption scheme. Section 4 explains the UDM concept as proposed in this paper. The next section, Section 5, presents the proposed Secure k-means cluster over the homomorphically encrypted data founded on the UDM concept introduced in the previous section. Method and results of experimenting the proposed algorithm have been discussed in section 6. Finally, section 7 concludes the paper with some ideas concerning future work to improve proposed mechanism.

2 Related Work

The main challenge of third party (k-means) data privacy preserving clustering using encryption is that, although the HE schemes used support a range of arithmetic operations that can be applied to cypher-data, there are some operations that are not supported. Mechanisms have been developed to address this challenge. These can be categorised as featuring either Multi-Parity Computation (MPC) or Partial Third-party Computation (PTC). Both feature significant data owner participation.

MPC is only applicable where the desired clustering is to be undertaken with respect to data belonging to two or more data owners. The basic idea is that the majority of the processing is conducted in-house by the data owners, only the centroids are shared on each k-means iteration [6]. Examples can be found in [13] and [7]. In [13] the data owners, on each k-means iterations, generate local clusters using their individual datasets. Each party then calculates the "centroid" of their local cluster, encrypts this and shares the encrypted centroid with the other owners. In [7] a similar mechanism is proposed except that only one of the users calculates the global centroids and then shares this. Using MPC data confidentiality is preserved since data is stored locally and is not shared with other parities. However, data owners are expected to do much of the work and are thus required to have the expertise and resource to carry out the desired clustering.

Using PTC most of the processing is conducted by a third party, but with recourse to the data owner (or owners) for the similarity calculations required as the k-means clustering proceeds. A disadvantage is thus that typically a great many of such calculations will be required. Using PTC data is encoded using some form of HE and sent to a third party where the homomorphic properties of the encryption are used to manipulate the data so far as possible (centroid calculation, data aggregation, and so). For example in [5] a PTC setting is used to cluster data belonging to social network users into k clusters. The data is passed to a "Semi Honest" third party where the clustering is performed, whilst similarity is determined with reference to randomly selected users. The mechanism proposed in this paper falls into the second category, but seeks to significantly minimise the interaction with the data owner(s).

There has also been work directed at incorporating the idea of order preservation into homomorphic encryption schemes to support secure k-means clustering. One example can be found in [11] where the authors describe a mechanism whereby partial order preservation can be included in Liu's encryption scheme [10]. Using this mechanism the third party uses *dynamic trap-doors* that are calculated on each iteration to convert cypher-text to order preserving text[1]. However, data owner participation is still significant because the trapdoors need to be recalculated on each iteration. Liu's scheme as used in [11], is also the encryption scheme adopted with respect to the work presented in this paper, because of its particular homomorphic properties; it is therefore discussed in further detail in Sub-section 3.2.

3 Preliminaries

Before considering the proposed secure k-means third party clustering mechanism and the proposed UDM concept, some preliminaries are presented in this section. Firstly, although the k-means algorithm is well understood [12], for completeness it is briefly presented in Sub-section 3.1. Liu's homomorphic encryption scheme [10] is then described in Section 3.2.

3.1 K-Means clustering

K-means is an iterative clustering algorithm where n data items are grouped into k clusters [12]; k is specified by the user. Each cluster is represented by its *centroid*. On the first iteration the first k records are usually used as the centroids. The remaining records are then assigned to clusters according to the shortest "distance" from each record to each centroid. At the end of the first iteration the centroids are recalculated; this is typically, but not necessarily, done using the mean values of the attribute values for the records present in each cluster. The algorithm then again assigns the records to the clusters and continues in this iterative manner until the centroids become fixed.

[1] A "trapdoor" in this context is a function that can be simply computed on one direction but is difficult to compute in the reverse direction without additional knowledge; the concept is widely used in cryptography.

Algorithm 1 Encrypt(v, K(m))

```
1: procedure ENCRYPT($v, K(m)$)
2:     Uniformly generate $m$ arbitrarily real random numbers $r_1, ....., r_m$
3:     Declare $E$ as a real value array of $m$ elements
4:     $e_1 = k_1 * t_1 * v + s_1 * r_m + k_1 * (r_1 - r_{m-1})$
5:     for $i = 2$ to $m - 1$ do
6:         $e_i = k_i * t_i * v + s_i * r_m + k_i * (r_i - r_{i-1})$
7:     end for
8:     $e_m = (k_m + s_m + t_m) * r_m$
9:     Exit with $E$
10: end procedure
```

Algorithm 2 Decrypt(E, K(m))

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

3.2 Liu's Homomorphic Encryption Scheme

As noted above, the adopted encryption scheme with respect to the work presented in this paper was Liu's homomorphic scheme [10]. Using Liu's scheme each data attribute value is encrypted to m sub-cyphers $E = \{e_1, e_2, \dots, e_m\}$ where m is the *public key* ($m \geq 3$) and $K(m)$ is a list of secret keys. $K(m) = [(k_1, s_1, t_1), \dots, (k_m, s_m, t_m)]$ where k_i, s_i and t_i are real numbers. The secret values satisfy the following conditions: (i) $k_i \neq 0$ $(1 \leq i \leq m-1)$, (ii) $k_m + s_m + t_m \neq 0$ and (iii) there exists only one i $(1 \leq i \leq m)$ such that $t_i \neq 0$ (this last so as to preserve a partial ordering of the data). Algorithm 1 shows the pseudo code for the *Encrypt*$(v, k(m))$ encryption function that converts a real value v to a set of sub-cyphers $E = \{e_1, \dots, e_m\}$. Given a set of sub-cyphers $E = \{e_1, \dots, e_m\}$ and the key $k(m)$ Algorithm 2 gives the pseudo code for the *Decrypt*$(E, k(m))$ decryption function to return the value v.

Liu's scheme has both security and homomorphic properties. In terms of security, the scheme is probabilistic, therefore it produces different cypher-texts for the same plain-text each time it is applied even when using the same secret key. This feature makes the scheme semantically secure and guards against "Chosen Plain-text Attacks" (CPAs) because if $E = \{e_1, \dots, e_m\}$ is a cypher-text which encrypts v_1 or v_2, the adversary will not know which one E encrypts. The homomorphic properties of the scheme support addition and subtraction of cypher texts, and multiplication and division of cypher texts with a real number c.

$$\begin{aligned}
E + E' &= \{e_1 + e'_1, \dots, e_m + e'_m\} &= v + v' \\
E - E' = E + (E' \times -1) &= \{e_1 + (e'_1 \times -1), \dots, e_m + (e'_m \times -1)\} &= v - v' \\
c \times E &= \{c \times e_1, \dots, c \times e_m\} &= c \times v \\
c \div E &= \{c \div e_1, \dots, c \div e_m\} &= c \div v
\end{aligned}$$

In the context of k-means clustering, as will become clear, addition is required for summing the attribute values of records in a cluster and division to consequently arrive at a centroid for the cluster. Subtraction is required to determine the difference between the centroids arrived at on iteration i with those from iteration $i - 1$.

4 The Updatable Distance Matrix Concept

Liu's scheme, described in Section 3.2, transfers plain-texts values randomly to cypher-texts, hence any ordering that might feature in the plain-text data is not transferred to the cypher-texts. Therefore, cypher-texts cannot be directly compared and hence k-means clustering cannot be directly applied. The propose idea is to guide the k-means clustering using what we refer to as an *Updatable Distance Matrix* (UDM) that holds the distances between attribute values in records with the corresponding attribute values in every other record. A UDM is thus a 3D matrix with the first two dimensions corresponding to the records in the data set and the third to the set of attributes that feature in the dataset. Thus, more formerly, given a data set $D = \{r_1, r_2, \dots r_n\}$ where each record r_x is a feature vector comprised a set of values $\{v_{x_1}, v_{x_2}, \dots v_{x_m}\}$ (each value corresponding to an attribute in the set of attributes $A = \{a_1, a_2, \dots, a_m\}$ featured in the data set), the associated UDM $\mathbf{U}$ will be comprises of a set of elements $u_{[x,y,z]}$ where x and y indicates the record numbers ($x < n$ and $y < n$), and z the attribute number ($z < m$). Given an element $u_{[x,y,z]} \in \mathbf{U}$ the value held will then be calculated using:

$$v_{x_z} \sim v_{y_z} \tag{1}$$

Note that where $x \equiv y$ the UDM values will equate to 0. Note also that UDMs are symmetric about there leading diagonal, and hence only the 3D leading triangle needs to be considered.

The UDM for a particular dataset is generated by the data owner and sent to the third party data miner together with the encrypted data set. The owner can then request a clustering specifying k. On start up the first k records are used as the centroids; then, as the k-means algorithm progresses, the centroids are updated. However, it is not necessary for the UDM to be sent back to the data owner as in the case of other PTC approaches, such as that described in [5]. It is only necessary for differences between the newly identified centroids C' on iteration i and the centroids C from iteration $i-1$ to be returned to the data owner so that offsets, referred to as *shift values*, can be calculated and stored in a Shift value Matrix (SM) $\mathbf{S}$, to be returned to the third party data miner who can then update the UDM $\mathbf{U}$ to generate an updated version of the UDM $\mathbf{U}'$ ($\mathbf{U}' = \mathbf{U} + \mathbf{S}$). The process whereby the proposed UDM concept, together with the use of SMs, is utilised in the context of secure k-means clustering is discussed in further detail in the following section.

5 Secure k-means Clustering Using the UDM Concept

This section presents the proposed *UDM* based on secure k-means clustering process. The process has two parts, a data preparation part and a clustering part. The first is conducted by the data owner and is detailed in Sub-section 5.1, while the second is conducted by the third party and is detailed in Sub-section 5.2. Note that the proposed process delegates some of the processing to the data owners, but this is limited to the generation of SMs and is significantly less than in the case of more conventional PTC approaches such as that presented in [5].

5.1 Data Owner Process

Data sets are prepared for outsourcing by generating a UDM and translating the raw data into a suitable encrypted format. The pseudo code for the process is given in Algorithm 3. The input to the algorithm is a dataset D, a set of attributes A that are featured in D and the number of required sub-cyphers m, the later required for Liu's encryption scheme adopted with respect to the work presented in this paper. Note that k-means clustering works only on numerical (or pseudo numerical) data, because similarity measurements are central to the operation of the mechanism. Therefore categorical (or labelled) data needs to be replaced with discrete integers values before any further processing can be conducted, this is done in line 2 of the algorithm. A list of secret keys $K(m)$ is then defined (line 3). The processed data set is then encrypted to form a cypher data set D' using Algorithm 1 (lines 5 to 7). Next the desired UDM, $\mathbf{U}$, is dimensioned (line 8) and populated (lines 9 to 15); recall from Section 4 that v_{x_z} is the value of the zth attribute in the feature vector describing the xth record in the dataset, while v_{y_z} is the value of the zth attribute in the feature vector describing the yth record in the dataset. On completion (line 16) the process returns the encrypted data set D' and the generated UDM $\mathbf{U}$ ready to be sent to the third party data miner.

5.2 Third Party Process

The secure k-means clustering is conducted by the third party data miner following a processes very similar to the traditional k-means algorithm as described in Sub-section 3.1. The pseudo code presented in Algorithm 4 describes this process. The input is the encrypted data set D', the UDM $\mathbf{U}$ (previously submitted to the third party) and the number of desired clusters k. We commence by dimensioning the set of clusters $C = \{C_1, C_2, \ldots, C_k\}$ and assigning the first

Algorithm 3 Data encryption and UD matrix generation

```
 1: procedure OutsourceData($D$,$A$,$m$)
 2:     ProcessedData = Dataset $D$ converted to numeric data set where necessary
 3:     $K(m) = LiuScheme(m)$
 4:     $D' = \emptyset$
 5:     for all $r \in ProcessedData$ do
 6:         $D' = D' \cup Encrypt(r, K(m))$ (Algorithm 1)
 7:     end for
 8:     $\mathbf{U}$ = Empty UDM dimensioned according to $|D|$, $|D|$ and $|A|$
 9:     for $x = 1$ to $x = |ProcessedData|$ do
10:         for $y = x$ to $y = |ProcessedData|$ do
11:             for $z = 1$ to $z = |A|$ do
12:                 $u_{[x,y,z]} = v_{x_z} \sim v_{y_z}$ $(u_{[x,y,z]} \in \mathbf{U})$
13:             end for
14:         end for
15:     end for
16:     return $D'$ and $\mathbf{U}$
17: end procedure
```

k encrypted records from D' to it (lines 2 and 3). We then define a second set $Cent = \{c_1, c_2, \ldots, c_k\}$ to hold the current centroids (line 4). Next the remainder of D' is processed and assigned to clusters using the *populateClusters* sub-process (line 5) given at the end of the algorithm. Records are assigned to clusters according to the similarity between the record and centroids as calculated using the UDM $\mathbf{U}$ as follows (where r_x is record $x \in D$, and r_y is record $y \in D$ representing cluster centroid y $(1 \leq y \leq k)$:

$$sim(\mathbf{U}, r_x, r_y) = \sum_{z=1}^{z=|A|} \mathbf{U}_{[x,y,z]} \tag{2}$$

We then calculate the new centroids (line 6). Next we enter into a loop (lines 7 to 14) which repeats until $Cent$ and $Cent'$ are the same. At the start of each iteration we create an encrypted SM $\mathbf{S}'$ (line 8). However, to update $\mathbf{U}$ we need real values, the content therefore needs to be decrypted. This can only be done by the owner (line 9). Note that (not shown in the algorithm) when updating $\mathbf{U}$ we only need to update the records in the second dimension representing cluster centroids. With the new centroids we again assign all records to C using the *populateClusters* sub-process (line 12) in the same manner as before. We continue in this manner till the process terminates.

6 Evaluation

The evaluation of the proposed method is presented in this section using ten datasets from the UCI data repository [8]. Table 1 gives some statistical information concerning these data sets. The Data sets have integer, real and categorical attribute types. The number of classes in each case was used as the value

Algorithm 4 Secure k-means clustering algorithm

```
 1: procedure SECURE K-MEANS($D'$, $\mathbf{U}$, $k$)
 2:     $C$ = Set of $k$ empty clusters
 3:     Select first $k$ records in $D'$ and assign to $C$ (one per cluster)
 4:     $Cent$ = Set of first $K$ records in $D'$ (the $k$ cluster centroids)
 5:     $C = populateClusters(k+1, \mathbf{U}, C, D', Cent)$
 6:     $Cent'$ = CalculateCentroids($C$)
 7:     while $Cent \neq Cent'$ do
 8:         $\mathbf{S}'$ = SM obtained from comparing $Cent$ and $Cent'$
 9:         $\mathbf{S} = \mathbf{S}'$ decrypted by data owner
10:         $\mathbf{U} = \mathbf{U} + \mathbf{S}$
11:         $C$ = Set of $k$ empty clusters
12:         $C = populateClusters(1, \mathbf{U}, C, D', Cent')$
13:         $Cent = Cent'$
14:         $Cent'$ = CalculateCentroids($C$)
15:     end while
16:     Exit with $C$
17: end procedure
18: procedure POPULATECLUSTERS($x$,$\mathbf{U}$,$C$,$D'$,$Cent$)
19:     $id = null$
20:     for $x = x$ to $x = |D'|$ do
21:         for $y = 1$ to $y = |C|$ do
22:             $sim = sim(\mathbf{U}, r_x, c_y,)$ where $r_x \in D'$ and $c_y \in Cent$ (Equation 2)
23:             $id$ = cluster identifier with lowest $sim$ value so far
24:         end for
25:         $C_{id} = C_{id} \cup r_x$ ($C_{id} \in C$)
26:     end for
27:     return $C$
28: end procedure
```

for k and the class column omitted from the data set. The proposed process was implemented using the Java object oriented programming language. Cluster configuration correctness was measures by comparing the results obtained with results obtained using standard (unencrypted) k-means clustering. The metric used was the Silhouette Coefficient (Sil. Coef.) [14] calculated using Equation 3 where $a(x_j)$ describes the cohesion between each record x_j and the other records in the same cluster, whilst $b(x_j)$ describes the separation between each record x_j and records in other clusters. The Silhouette coefficient is a real number between -1 and 1 where the closer the coefficient is to 1 the better the clustering. We were also interested in overall runtime, the time to complete the clustering.

$$\begin{aligned} OverallSil &= \frac{\sum_{i=1}^{k} \frac{\sum_{j=1}^{j=|c_i} Sil(x_j)}{|C_i|}}{k} \\ Sil(x_j) &= \frac{b(x_j) - a(x_j)}{max(a(x_j), b(x_j))} \end{aligned} \tag{3}$$

Datasets	Num. Records	Num. Attributes	Num. Clusters	Attributes Description
Banknote Authent.	1372	4	2	Real
Breast Cancer	198	33	2	Real
Breast Tissue	106	9	6	Integer and real
Lung Cancer	32	56	3	Integer
Blood Transfusion	748	4	2	Integer
Cardiotocography	2126	36	3	Integer and real
Chronic-kidney	400	24	2	Categorical, int. and real
Iris	150	4	3	Categorical, int. and real
Pima Disease	768	8	2	Integer and real
Seeds	210	7	3	Integer and real

Table 1. Statistical information for the data sets used in the evaluation

Datasets	Secure k-means			Standard k-means		
	Runtime (Mili. Secs.)	Num. Iterations	Sil. Coef.	Runtime (Mili. Secs.)	Num. Iterations	Sil. Coef.
Banknote Authent.	87	17	0.207	58	17	0.207
Breast Cancer	31	9	0.020	11	9	0.020
Breast Tissue	26	19	0.787	19	19	0.787
Lung Cancer	17	9	0.645	3	9	0.645
Blood Transfusion	27	13	0.370	19	13	0.370
Cardiotocography	333	25	0.065	216	25	0.065
Chronic-kidney	29	9	0.009	14	9	0.009
Iris	14	15	0.836	9	15	0.836
Pima Disease	30	9	0.000	19	9	0.000
Seeds	10	7	0.706	9	7	0.706

Table 2. Execution time for secure k-means clustering

The results are presented in Table 2, the runtime is the time to complete the clustering excluding encryption and UDM generation. From the table it can be seen firstly that the cluster configurations produced using the proposed secure k-means clustering were identical to those produced using standard k-means clustering as evidenced by the Silhouette Coefficients produced (columns 4 and 7 in the table) which were identical in all cases. Thus it can be concluded that the homomorphic properties used to calculate the centroids and distances between centroids do not effect the accuracy of the clustering. In terms of run time the same number of iterations were required with respect to each dataset, although from the table it can be seen that the bigger the dataset the more iterations are required. The overall run time required for the secure k-means approach to produce the desired cluster configurations, as expected, was longer than in the case of standard k-means clustering. Inspection of the table indicates that this was not significant. The runtime required by the data owner to calculate a UDM and encrypt the data did not add a significant overhead. Even for the largest data set, Cardiotocography, the time to create the UDM was 912ms. and to encrypt the data was 25ms. Note that the time complexity for generating an initial UDM will be in the order of $O(|D| \times |C| \times k)$. The time to decrypt a SM, on each iteration, will be negligible. However, it should be noted that with respect to the experiments reported here the data owner and third party were both

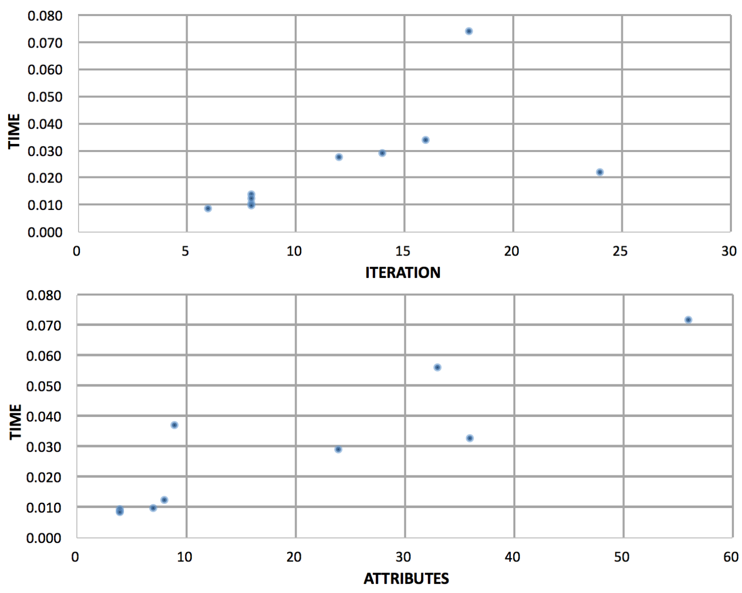

Fig. 1. Time required by data owner to decrypt shift (distance) between two centroids

hosted on the same machine, thus there was no "message passing" overhead; in "real life" the data owner and third party will be hosted on separate machines, time for message passing would add to the run time although not significantly so. Figure 1 shows two plots. The first plots each data set against overall runtime for the secure k-means clustering and number of iterations, the second against overall runtime and $|A|$. The plots confirm, that although not necessarily linear, the run time increases with the number of iterations and attributes.

7 Conclusion

In this paper a secure k-means clustering mechanism has been described that uses the concept of an Updatable Distance Matrix (UDM). The advantage offered is that, unlike comparable secure k-means clustering mechanisms from the literature, data owner participation is negligible, restricted to decrypting a Shift Matrix (SM) on each iteration. The presented evaluation demonstrates that the encryption and processing does not adversely effect the quality of the clusters produced, they are the same as when k-means is applied in a standard manner. The processing time, as was to be expected, is greater but not significantly so. In an ideal situation the data owners should be able to package their data so that it is secure, send it to a third party for storage and analysis, and receive analysis results when required without the need for any further communication whilst the analysis is taking place. However, given that at present, there is no suitable encryption scheme that will support all the necessary mathematical operations for this to happen, the proposed mechanism significantly reduces the amount of data owner participation required. For future work the authors intend to investigate the utility of the UDM concept with respect to alternative clustering algorithms, and other data mining techniques (such as classification and pattern mining techniques).

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. *SIGMOD Rec.*, 29(2):439–450, May 2000.
2. Scott Berinato. Theres no such thing as anonymous data. *Harvard Business Review*, February, 2015.
3. T. Chen, J. Chen, and B. Zhou. A system for parallel data mining service on cloud. In *2012 Second International Conference on Cloud and Green Computing*, pages 329–330, November 2012.
4. Hitesh Chhinkaniwala and Sanjay Garg. Privacy preserving data mining techniques: Challenges and issues. In *Proceedings of International Conference on Computer Science & Information Technology, CSIT*, page 609, July 2011.
5. Zekeriya Erkin, Thijs Veugen, Tomas Toft, and Reginald L Lagendijk. Privacy-preserving user clustering in a social network. In *2009 First IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 96–100. IEEE, December 2009.
6. Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
7. Somesh Jha, Luis Kruger, and Patrick McDaniel. Privacy preserving clustering. In *European Symposium on Research in Computer Security*, pages 397–417. Springer, September 2005.
8. M. Lichman. UCI machine learning repository, 2013.
9. Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *Journal of cryptology*, 15(3):177–206, June 2002.
10. Dongxi Liu. Homomorphic encrypton for database querying, December 2013.
11. Dongxi Liu, Elisa Bertino, and Xun Yi. Privacy of outsourced k-means clustering. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 123–134. ACM, June 2014.
12. James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297. Oakland, CA, USA., June 1967.
13. Deepti Mittal, Damandeep Kaur, and Ashish Aggarwal. Secure data mining in cloud using homomorphic encryption. In *Cloud Computing in Emerging Markets (CCEM), 2014 IEEE International Conference on*, pages 1–7. IEEE, October 2014.
14. Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, November 1987.
15. Meena Dilip Singh, P Radha Krishna, and Ashutosh Saxena. A privacy preserving jaccard similarity function for mining encrypted data. In *TENCON 2009-2009 IEEE Region 10 Conference*, pages 1–4. IEEE, January 2009.
16. Jaideep Vaidya, Christopher W Clifton, and Yu Michael Zhu. *Privacy preserving data mining*, volume 19. Springer Science & Business Media, 2006.
17. Yang Yang and MA Maode. Semantic searchable encryption scheme based on lattice in quantum-era. *Journal of Information Science & Engineering*, 32(2):425–438, March 2016.