# Privacy-Preserving Clustering with High Accuracy and Low Time Complexity

Yingjie Cui[1] and W. K. Wong[1] and David W. Cheung[1]

Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong
{yjcui, wkwong2, dcheung}@cs.hku.hk

**Abstract.** This paper proposes an efficient solution with high accuracy to the problem of privacy-preserving clustering. This problem has been studied mainly using two approaches: data perturbation and secure multiparty computation. In our research, we focus on the data perturbation approach, and propose an algorithm of linear time complexity based on 1-$d$ clustering to perturb the data. Performance study on real datasets from the UCI machine learning repository shows that our approach reaches better accuracy and hence lowers the distortion of clustering result than previous approaches.

## 1 Introduction

Nowadays, as information accumulates rapidly, data mining - the technology of discovering knowledge out of information, becomes more and more prevalent. Clustering analysis is a commonly used data mining tool. It is a process of grouping a set of physical or abstract objects into classes of similar objects [5]. Given a similarity measure, it tries to maximize the intra-cluster similarity as well as minimize the inter-cluster similarity of data. $K$-means clustering is one of the most famous clustering problems. The objective of $k$-means clustering is to cluster a set of $n$ objects into $k$ partitions such that the average (Euclidean) distance from objects to its assigned cluster center is minimized where $k$ is given by the user. $K$-means clustering has been applied in many applications in real life, such as customer behavior analysis, biology research and pattern recognition, etc.. Many of these applications are done over very large datasets, for example, millions of transaction records. Besides, $k$-means clustering is shown to be an NP-hard problem [4]. So, the heuristic algorithm proposed by Lloyd [13] is usually used in real applications. The algorithm adopts the hill climbing technique and returns a local optimum solution to the user.

The data used in the clustering process may contain sensitive information of an individual. For example, the financial transaction records held by a bank. Privacy issues are concerned on how the bank uses the information collected. The data owner (the bank) should not let others (the general public) observe the data (the clients' transaction records). On the other hand, there are many situations that there are other parties involved in clustering analysis. First, consider the situation that a law enforcement agency needs to investigate people's financial transactions which are divided between different financial institutions, e.g. banks and credit card companies. These institutions cooperate together to compute a global $k$-means clustering. The database in one financial institution should not be revealed to others. Consider another situation: as the development in cloud computing introduces the idea of software as a service (SaaS), a data owner can outsource the clustering task to a service provider in order to gain the benefits such as cost relief and payment on consumed resources only. The data owner sends his data to the service provider and executes an application at service provider. Since the service provider is a third party, it is not trusted. Privacy of individuals should be protected against the service provider. So, there is a need in studying privacy-preserving clustering problem in which the clustering result is found without accessing the original data.

Some algorithms have been proposed to address the privacy-preserving clustering problem. We describe two main approaches as follows:

1. Secure multiparty computation (SMC) approach: this approach addresses the privacy-preserving clustering problem in the multiparty case [18, 8]. A number of data owners, each of them owns a database, cooperate together to compute global clustering result. The result is computed through certain rounds of complex communications among the data owners.
2. Perturbation approach: the data owner generates a perturbed database from the original database by adding noise to it [15, 6, 11]. The perturbed database can be accessed by any other parties. Hence, one can collect the perturbed databases he want to perform clustering on and performs data mining on his own.

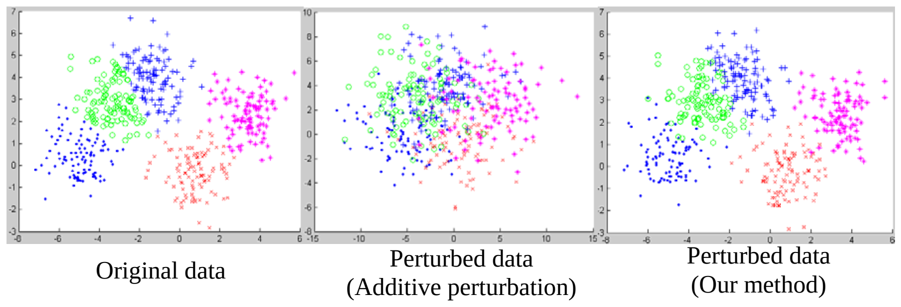

**Fig. 1.** An example illustrating the distortion of clustering result. The leftmost graph represents the clustering result on the original dataset. The clusters are represented in different colors. The graph in the center represents the disturbed points by using additive noise perturbation. The rightmost graph represents the disturbed points using our method.

Although SMC approach provides accurate result and is provably secure, it cannot be applied to the general case of privacy- preserving clustering, for example the outsourcing scenario. In addition, there are considerable communication overheads in the computation. So, we adopt the perturbation approach that can be applied in general case of privacy-preserving clustering and we can use existing algorithms for $k$-means clustering to compute the clusters. However, current perturbation approaches on privacy-preserving clustering are either insecure against a practical attacker [10] or destructive to the clustering result. Most of current studies focus on protecting the privacy of the data and do not consider the distortion to the mining result in the generation of noises. Figure 1 shows the distortion introduced to the clustering result by an existing perturbation method [6] and that by our method. If nosies are not added carefully, the noises can easily disturb the clustering result. In addition, a secure additive perturbation takes $O(nd^2)$ time, where $n$ is the number of objects in the database, $d$ is the number of dimensions of each object. It is a considerable high cost especially when the number of dimensions is high.

In this paper, we study the problem of privacy-preserving clustering and give a *secure* and *efficient* perturbation method in which the clustering result is *accurate*.

**Contributions of this paper**: we study the problem of privacy-preserving clustering using the data perturbation approach. Our contributions include: (1) we propose a new attack model to the distance-preserving perturbation and hence show that it is not secure even the third party obtains the distances between points from a black box oracle; (2) we propose a novel perturbation method using 1-$d$ clustering to perturb the data in database which is secure, efficient and preserves accuracy of clustering result; (3) we give a theoretical study on the cost and security of the proposed technique; (4) we evaluate the proposed scheme with experiments on real datasets.

The rest of this paper is organized as follows. Section 2 mentions some related work. Section 3 defines the problem of privacy-preserving clustering and states the requirements on the solution. Then, we propose a new attack to distance-preserving perturbations in section 4 and hence show that distances between points cannot be revealed to attackers although distances alone do not reveal the location of points. Section 5 introduces our solution to this problem, and we perform experiments in Section 6 to compare the accuracy and efficiency of our solution with previous approaches. Section 7 concludes the paper and gives directions for future work.

## 2 Related Work

[15] first addressed the problem of privacy-preserving clustering. They proposed geometric transformations, which includes shift, scaling and rotation, to disguise the original data. The geometric transformations used in [15] except scaling can be summarized as distance-preserving transformations. By distance-preserving transformations, we mean that $|x - y| = |T(x) - T(y)|$ for all $x$ and $y$ in the dataset, where $T$ denotes the transformation.

Since all the pairwise distances are preserved, this kind of transformation always preserves the accuracy of the clustering result. The privacy of distance-preserving transformation has been studied by [10]. In algebra [2] this kind of transformation is called rigid motion, and can be represented by $T(x) = Mx+v$, where $M$ is an orthogonal matrix and $v$ is a vector. The *known input-output* attack assumes that some of the original data records are leaked and thus the attacker knows them. Besides, the attacker also has the released data, and knows the correspondence of the leaked original data and the transformed data. So the attacker's task is to solve $M$ and $v$ using some linearly independent *known input-output* points to set up linear regression equations. If $v = 0$, knowing $d$ linearly independent points is enough for the attacker to solve $M$ thus recover the whole original dataset. If $v$ is not equal to 0, one more point is needed. This infers the safety bound of distance-preserving transformation against regression attack is not high.

[15] also proposed an additive random perturbation to improve the privacy by adding normally or uniformly distributed noise to the sensitive numerical attributes, which can be represented as $y_i = x_i + r_i$. Using this additive perturbation approach can raise the difficulty for the attacker to recover the original data. However, it can largely distort the clustering result. Besides, it is susceptible to data reconstruction methods such as PCA and Bayesian estimation [6], which can filter out much of the noise if the data is highly correlated. When the correlation of the data is high, the information of data concentrates in several dimensions that have higher variances than others. If the additive noise is evenly distributed in all dimensions, it can be largely filtered out if the attacker applies PCA to reduce dimensions that has small variances, while the information of the data is still approximately preserved. So [6] proposed a method to add noise for better resistance against this eigen-analysis attack, using the same covariance for the noise as that of the original data. However, since [6] mainly focused on the privacy problem of data publishing, its method of perturbation did not consider the need of accuracy in clustering. In other words, the clustering result can be largely distorted. And we have conducted experiments to show this. Besides, the calculation of a covariance matrix of the original data requires the time complexity of $O(nd^2)$, where $n$ is the cardinality of the dataset and $d$ is the number of dimensions. This is a considerable cost for high dimensional data.

Another kind of perturbation is the multiplicative perturbation, which is based on the Johnson-Lindenstrauss lemma [9]: if data points in a high dimensional space are projected onto a space of lower dimension, the distances between the data points can be approximately preserved. Also, after the projection, high dimensional data can not be recovered from low dimensional data because of the loss of information. Thus projections can be applied to distance-based operations such as privacy-preserving clustering. The best (in terms of accuracy) and most commonly used dimension reduction technique is PCA. It can be used to select the dimensions that have greater variances than others, and then the data can be projected onto these dimensions. Its costs which include calculating and decomposing the covariance matrix and matrix multiplication require a time complexity of $O(nd^2) + O(d^3)$, which is not suitable for high dimensional data. So a computationally simpler but less accurate dimension reduction method, the random projection, was proposed [3]. This projection can be represented as $Y = XM$, where $X$ is the $n * d$ original dataset, $M$ is a $d * e$ randomized matrix, $e$ is the dimension after reduction. The elements of $M$ are i.i.d. samples of a certain distribution, and $Y$ is the $n * e$ projected dataset. This projection method reduced the time complexity to be $O(nde)$. [11] used this random projection to perturb the original data for distributed privacy-preserving data mining.

[18] proposed a solution to the privacy-preserving clustering problem using a different approach. Data are partitioned according to different attributes and distributed to several parties. These parties together find the clustering result, but do not share their original data with each other. The privacy of this approach is preserved using Yao's framework of secure multiparty computation [19]. Since all the computations are done through encryption and decryption, no distortion of data happens, so the accuracy is preserved. There has been other work [8] which partitions the data in a different way and uses the same secure framework with [18]. This approach assumes several non-colluding parties to do the clustering task, which may be difficult to find in realistic situation. Besides, the bit communication cost among different parties in each iteration of $k$-Means algorithm is $O(nrk)$, where $n$ is the number of data points, $r$ is the number of parties and $k$ is the number of clusters, which is not low.

## 3 Problem definition

A data owner owns a database $DB$, which consists of $n$ objects. Each object is represented by a $d$-dimensional point. The distance between two points $x = (x_1,\ x_2,\ \dots\ x_d)$ and $y = (y_1,\ y_2,\ \dots,\ y_d)$ is measured by Euclidean distance[1], i.e., $|x - y| = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$. The problem of k-means clustering is, given $k$ as input, to partition the $n$ into $k$ partitions: $P_1$, $P_2$, ..., $P_k$, such that the intra-cluster variance is minimized. The intra-cluster variance is measured by $\sum_{i=1}^{k}\sum_{x \in P_i}(x_i - \mu_i)^2$, where $\mu_i$ is the cluster center of $P_i$ and $\mu_i = \frac{\sum_{x \in P_i} x_i}{|P_i|}$.

In privacy-preserving clustering, a third party data miner requires computing $k$-means clustering on $DB$. Due to privacy of information, the data owner releases a perturbed database $DB'$ to the data miner. $DB'$ is computed by perturbing the original objects in $DB$. We model the perturbation as $DB' = \{y \mid \forall x \in DB, y = T(x)\}$, where $T$ represents the perturbation process[2]. Our objective in this paper is to develop a transformation process $T$ such that

1. $T$ is efficient. The cost at the data owner side is low and should be lower than the cost of performing clustering on his own. Otherwise, our technique is not suitable in privacy preservation of outsourcing scenarios.

[1] There are other distance measures used in the literature, e.g., Manhattan distance. We focus to Euclidean distance in this paper because it is more popular in practice.
[2] The perturbation function $T$ can be irreversible and non-deterministic.

2. $T$ is secure. An attacker who obtains $DB'$ and some background information on $DB$ cannot recover $DB$.
3. $T$ preserves the accuracy of the clustering result. The clustering result on $DB'$ should be similar to that on $DB$.

## 4 Distance Based Attack

An ideal perturbation method is that we can preserve clustering results for any datasets. The major part of clustering algorithms is to compute the distances between points and compare the distances to partition the points. If the data miner can compute the distances accurately, he can compute the correct mining result. Otherwise, if the distances are disturbed by random noises, the correctness of the mining result can not be ensured. So, it is an intuitive idea to explore the feasibility of a perturbation scheme which the data miner can accurately compute the distance between any two points[3]. However, in this section, we will show a negative result that such perturbation scheme cannot be secure by proposing an attack to it.

**Theorem 1.** *A perturbation scheme $T$ is **insecure** against a known input-output attack given $T$ allows an attacker to observe the distances between points in the original space.*

*Proof.* Suppose an attacker has retrieved the perturbed database $DB'$ and a black box oracle $G$. $G$ lets the attacker compute the distance between two points $x$ and $y$ by $|x - y| = G(x', y')$, where $x'$ ($y'$ resp.) is the perturbed point of $x$ ($y$ resp.). In a *known input-output* attack, the attacker obtains a number of $m$ original points $z_i$ and the corresponding perturbed points $z_i'$ in the database. In order to recover a victim point $v'$ in $DB'$ to $v$ in $DB$, the attacker first computes the distances between $v$ and every known point $z_i$. So, we can set up $m$ quadratic equations[4]: $|v - z_i| = G(v', z_i')$. Each of the equation forms a d-hypersphere in the original space and $v$ lies on the intersection of the hyperspheres. In general, if $m \geq d + 1$, the intersection is in fact a point. So, the attacker can conclude the original value of $v$. $\square$

We use an example to illustrate the proposed attack in 2-$D$ space as shown in figure 2. The attacker knows a set of three points $A$, $B$ and $C$ in the $DB$. He also

[3] This scheme covers the distance-preserving transformation, but is not restricted to it. For example, this perturbation scheme can also include encryption, with the help of a trusted secure device[1]. The data owner can encrypt the data and then send it to the miner along with the secure device, and the data miner then uses the secure device to decrypt the data and calculate the pairwise distances in the process of clustering.

[4] Some previous work has also proposed scaling in perturbation which the distances are not exactly preserved. In such case, we may replace $G(v', z_i')$ by $sG(v', z_i')$ in the equation where $s$ is the unknown scaling factor. In general, the attack can still recover the points given the RHS of the equation is a polynomial of $G(v', z_i')$, though the attacker may need more known points for solving the increased number of unknowns.

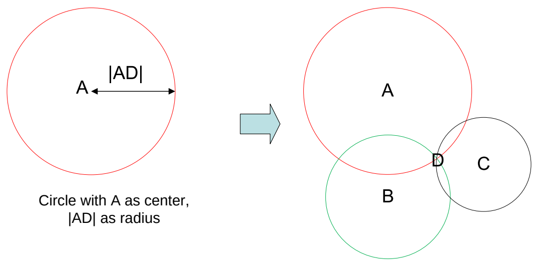

**Fig. 2.** An example illustrating the proposed attack. The attacker observes the original value of the point $D$ by knowing $A$, $B$, $C$ and the distances $|AD|$, $|BD|$, $|CD|$.

knows the corresponding perturbed points $A'$, $B'$, and $C'$ in $DB'$ respectively. Suppose now he is trying to infer the original point $D$ of a perturbed point $D'$ in $DB'$. He calculates the distance of $D$ to every known point using $G$. So, he gets $|AD|$, $|BD|$, $|CD|$. The attacker then draws three circles with $A$, $B$, $C$ as the centers and $|AD|$, $|BD|$, $|CD|$ as the radii in the 2-$D$ space. $D$ must be on the perimeter of each circle. Now there is only one intersection of the circles. So, the attacker can conclude that the intersection in the example is $D$. Notice that even the attacker knows two points (say, $A$, $B$) only in the example, he can infer that $D$ is one of the two intersections of the two circles. So, there is $\frac{1}{2}$ chance that the attacker obtains $D$ by a random guess. This infers the safety bound of such perturbation scheme that preserves the pairwise distances is low, and leads us to develop a heuristic perturbation scheme in which distances between points are distorted but the clustering result is almost preserved.

## 5 Perturbation Based on 1-d Clustering

### 5.1 The Measure of Distortion

Previous perturbation methods use additive noise and multiplicative noise to improve the privacy. However, the additive noise can largely distort the clustering result, since it does not take into account the pairwise distances of the original data. The distortion of the clustering result can be measured by the difference between the clustering results of the original data and the perturbed data, and there are several metrics that have been proposed, such as Variation of Information (VI), Mirkin Metric and Van Dongen Metric. [14] studied these metrics, and proved that VI is a sensible metric for comparing clusters. As a metric of comparing the difference between two clustering results, VI is defined as the formula[5] below:

$$VI = -\sum_{i=1}^{k} p_i \log p_i - \sum_{j=1}^{k} q_j \log q_j - 2\sum_{i=1}^{k}\sum_{j=1}^{k} t_{ij} \log \frac{t_{ij}}{p_i q_j}$$

Here $p_i$ $(i \in [1,k])$ is the proportion of the cluster i in one cluster result, i.e. $p_i$ is the number of the elements in cluster $i$ divided by the total number of data points. And $q_j$ $(j \in [1,k])$ is the proportion of the cluster j in the other

[5] The base of the logarithm in this formula is 2.

cluster result. $t_{ij}$ is the proportion of the same data points shared by $p_i$ and $q_j$. This metric has a range [0, 2log$k$], and higher value means greater difference. Besides, since the $k$-means problem is NP-hard [4], much of the previous work has adopted the iterative approach proposed by Lloyd [13]. The result of this approach heavily depends on the initial choice of means, so the comparison of clustering results between original data and perturbed data should be restricted, for example, they should be with the same initial means set.

## 5.2 Our Solution

We observe that for multidimensional data, the clustering result of the whole dataset will not change too much if we perturb the data in such a way that the clustering of each single dimension is preserved and the range of each dimension is also preserved. Having this observation, we propose a randomized perturbation algorithm that is linear in time complexity and based on 1-$d$ clustering.[6]

The details of the proposed algorithm is shown in Algorithm 1. We perform this algorithm on each dimension of the dataset, firstly calculate the 1-$d$ clustering result and then perturb the data based on that clustering result.

The distance between two neighboring clusters $C_1$ and $C_2$ are calculated using the formula in [16]: $C_1.count * C_2.count * (C_1.mean - C_2.mean)^2/(C_1.count + C_2.count)$.

## 5.3 The Analysis of Time Complexity

Step 1 requires 1 scan of the data. Step 2 and step 3 require 1 scan of the data. After step 3, we get $m$ clusters, and $m$ is less than or equal to $\theta$, the input number of initial intervals which is proportional to the cardinality of elements.

As for step 4 and step 5, we first look at the distance calculation and merging part. Since the number of clusters diminishes by a half after each round, the total step of these operations is the sum of a geometric progression with the proportion 1/2. So the total step of distance calculation is less than $2m - 1$, and the total step of merging is the half of that of distance calculation. Then we look at the finding of median of the distances. This can be done using a divide-and-conquer approach, and has been implemented in the STL of C++ as the function nth_element(), and its time complexity has been shown to be linear [7].

Step 6 is designed to control the number of final clusters, and its time complexity is $O(\mu\log\mu)$, where $\mu$ is the specified number of final clusters. The time complexity of step 7 is equal to one scan of the elements.

In conclusion, the time complexity of the above algorithm is linear with respect to the cardinality of data, i.e. $O(n)$. When applied to the whole dataset, the time complexity of this algorithm is $O(nd)$, where $n$ is the count of the data

[6] We use a heuristic to compute the clusters on one dimension but not the classic k-means clustering because we just need a approximate partitioning result and it is too expensive to execute classic k-means algorithms.

**Algorithm 1** Perturbation Based on 1-$d$ Clustering

Input: the data array of each dimension, the number of initial intervals $\theta$, the number of final clusters $\mu$.
Output: the perturbed data array.

PHASE 1, 1-$d$ Clustering: partition the data on one dimension

1. Scan the data to find the maximum and minimum element of the data array.

2. Divide the range of data into $\theta$ intervals, with the length of each interval to be $(max - min)/\theta$, where $max$ and $min$ are the maximum and minimum elements of the array, and $\theta$ is an input number given by the user. Generally, $\theta$ is proportional to the data cardinality.

3. Project each data element to the intervals. Since elements are generally not evenly distributed, some intervals will have elements in them and some will not. Elements that belong to the same interval are treated as the initial clusters. The lower bound and upper bound of each interval that contains elements are used as the lower bound and upper bound of that cluster.

4. Calculate the distances between neighboring clusters.

5. Find the median of these distances, and merge the neighboring clusters that have distances smaller than the median. After merging, use the lower bound of the lower cluster as the new lower bound, and the upper bound of the upper cluster as the new upper bound.

6. Repeat step 4 and step 5 until the number of clusters is smaller than $2\mu$. Then execute step 4, sort the distances, merge the neighboring clusters that have the smallest distance and continue merging until the number of clusters is $\mu$.

PHASE 2, Random Perturbation: Generate a random point in the partition as the perturbed point

7. For each cluster, calculate the distance between the lower bound and the mean and the distance between the upper bound and the mean. Use the smaller one of these two distances as the radius and the mean as the center to calculate a range. For each element in that cluster, generate a uniformly distributed random number in that range, and replace the original element with the random number.

points and $d$ is the dimension. So its scalability to large dataset with high dimension is good, compared with other approaches that require the time complexity $O(nd^2)$, $O(nd^2) + O(d^3)$ or $O(nde)$.

### 5.4 The Analysis of This Perturbation against Existing Attack Models

[6] proposed two attack models of the additive random perturbation. The first one is the eigen-analysis attack based on PCA, which tries to filter out the random noise in case that the original data is highly correlated. And the second one is the Bayesian Attack which tries to maximize the probability $P(X|Y)$, where $Y$ is the disguised data and $X$ is the reconstructed data.

The key point of the first attack model is that the attacker can calculate the covariance matrix of the original data from the disguised data, which can be represented as the formula below:

$$Cov(X_i + R_i, X_j + R_j) = \begin{cases} Cov(X_i, X_j) + \delta^2, \text{ for } i = j \\ Cov(X_i, X_j), \text{ for } i \neq j \end{cases}$$

The assumption of applying this formula is that the attacker knows the distribution of the additive noise, i.e. he knows the variance or noise $\delta^2$. However, in our approach, the original data is not additively perturbed, and the variance of the noise can not be separately learnt by the attacker.

The Bayesian attack is based on the assumption that both the original data and the randomized noise are subjected to multivariate normal distributions and the attacker knows this. This assumption is quite strong [12]. In our approach where the noise does not have such kind of distribution, this assumption can not be satisfied.

As for the *known input-output* attack and the distance based attack, since our approach changes the pairwise distances among data points, it can not be regarded as the orthogonal transformation, so the attacker can not use linear regression to recover the orthogonal transformation matrix, or use the pairwise distances to infer the original data.

In conclusion, our perturbation approach has good resistance against the above previous attack methods and our new attack model.

## 6 Experiments

In this section, we evaluate our proposed perturbation scheme in three aspects: privacy preserved, accuracy of clustering result, and execution cost. We compare the performance of our scheme against two existing approaches that are robust to all existing attacks: the additive random noise using the same covariance matrix as the original data [6], and the multiplicative perturbation, which includes the projection based on PCA dimension reduction and the random projection [11]. We denote the approaches as “Additive perturbation” and “Multiplicative perturbation” respectively.

There are different techniques in the multiplicative perturbation approach. The dimension reduction based on PCA is the most accurate dimension reduction technique, since the principle components captures the maximum possible variance [12]. In other words, PCA-based projection reaches the highest accuracy in the multiplicative perturbation approach. So we choose the PCA-based projection as the representative multiplicative perturbation to compare the distortion of clustering result with other approaches. Besides, as we have described in Section 2, the random projection outperforms PCA-based projection in time complexity. So we choose random projection to be the representative multiplicative perturbation when comparing the time complexity of different approaches.

## 6.1 Implementation details

For the additive perturbation approach, we first calculate the covariance matrix of the original data, and then use the *mvnrnd* function of MATLAB to generate the multivariate random noise which has the same covariance matrix as the original data. There are no input parameters to this approach.

For the PCA-based projection for multiplicative perturbation, more noise will be introduced if more dimensions are reduced, but it also causes a higher distortion to the clustering result. We need to give the number of dimensions to preserve as the input to the algorithm. We pick the input such that the proportion of preserved variance is higher than 0.99.

For the random projection for multiplicative perturbation, the elements of the randomized projection matrix is chosen from a standard normal distribution $N(0, 1)$. We use MATLAB to generate projection matrices of different numbers of dimensions, and then project the original dataset using these matrices. The algorithm requires the number of dimensions to preserve as the input. We will try different numbers of dimensions in the experiment for comparisons.

For the perturbation based on 1-$d$ clustering, the number of initial intervals $\theta$ and the number of final clusters $\mu$ are the input to the algorithm. The number of final clusters is a trade off between the accuracy and privacy: the more final clusters in each dimension, the lower distortion can be reached, and the less noise can be introduced into the original data. Here we choose $\mu$ for different datasets in a way to make the accuracy of clustering result comparable with other approaches. The number of initial intervals $\theta$ is also specified by the user, and here we set the numbers to be half of the data cardinality for all the datasets.

## 6.2 Experiment Settings

We have implemented the algorithms using C++ and MATLAB. The version of C++ compiler is gcc 3.4.2. All the experiments are performed on a PC with Intel Core 2 Duo CPU E6750 2.66G and 2G RAM. The operating system is Windows XP Professional Version 2002 SP2.

### 6.3 Datasets

We use 4 real datasets from the UCI machine learning repository [17]: Wine, Breast Cancer Wisconsin (Original), Statlog (Shuttle) and Musk (Version 2). The details of the datasets are shown in the table.

| Dataset Name | Number of Records | Dimensions | Classes |
|---|---|---|---|
| Wine | 178 | 13 | 3 |
| Shuttle | 58000 | 9 | 7 |
| BCW | 683 | 10 | 2 |
| Musk | 6598 | 166 | 2 |

**Table 1.** Datasets used in the experiment

We will evaluate the privacy preserved, and accuracy of clustering result of all approaches on the Wine, Shuttle and BCW datasets. The Musk dataset has a relatively high number of dimensions, so it is used to compare the execution time in perturbing the data.

### 6.4 Measurements of performance

For each of the dataset, we perturb it using the three approaches: additive perturbation and multiplicative perturbation and our approach using 1-$d$ clustering perturbation. So, we obtain the a perturbed datasets for each of the approach. Then, we perform $k$-means clustering algorithm by Lloyd [13] over the original dataset and different perturbed datasets. Since the initial cluster centers may affect the output of $k$-means clustering algorithm, we use the same initial centers for all cases. The measurement of the approaches are defined as follows:

**Privacy preserved** Privacy is measured as the amount of difference between the original dataset and the perturbed dataset [6]. We use the mean square error (MSE) as the measure of difference between original and perturbed dataset, which is the same as [6]. MSE can be calculated as MSE $= \sum_{x \in DB} |T(x) - x|^2$, where $T(x)$ represents the perturbed point of $x$. If an approach has a smaller MSE, the perturbed data is very similar to the original data. So, an attacker can approximately acquire the sensitive information. More privacy is preserved in the scheme that has a higher MSE.

**Accuracy of clustering result** We use Variation of Information (VI) as described in section 5.1. A smaller VI represents less difference between the clustering results on original dataset and that on the perturbed dataset. An ideal situation is VI= 0 which means the clustering results are the same.

**Execution cost** We measure the total execution time of each of perturbation approach in generating the perturbed dataset from the original dataset.

### 6.5 Experiment Results on Privacy Preserved and Accuracy of Clustering Result

The experiment results on the 3 datasets: Wine, BCW, Shuttle are shown in table 2, table 3, table 4 respectively.

| | Parameters | VI | MSE |
|---|---|---|---|
| Multiplicative perturbation | 1 dimension | 0 | 14.5115 |
| Addictive perturbation | | 0.0839841 | 16.5334 |
| 1-$d$ Clustering Perturbation | $\mu$ = 8*3, $\theta$ = 0.5*178 | 0 | 53.0283 |

**Table 2.** Experiment Results on Dataset: Wine

| | Parameters | VI | MSE |
|---|---|---|---|
| Multiplicative perturbation | 1 dimension | 0 | 7.06364 |
| Addictive perturbation | | 1.708 | 4.00013e+010 |
| 1-$d$ Clustering Perturbation | $\mu$ = 3*2, $\theta$ = 0.5*683 | 0 | 2.06604e+009 |

**Table 3.** Experiment Results on Dataset: BCW

| | Parameters | VI | MSE |
|---|---|---|---|
| Multiplicative perturbation | 4 dimension | 1.36994 | 53.8315 |
| Addictive perturbation | | 3.7063 | 6317.08 |
| 1-$d$ Clustering Perturbation | $\mu$ = 4*7, $\theta$ = 0.5*58000 | 0.990317 | 113.62 |

**Table 4.** Experiment Results on Dataset: Shuttle

In the experiments, our proposed algorithm outperforms both of the existing approach except that MSE of additive perturbation is larger (preserves more privacy) than our approach in BCW and Shuttle datasets. However, VI of addictive perturbation approach on these two datasets are extremely high for these two datasets. Note that VI represents the accuracy of clustering result and is bounded by $2logk$. In BCW dataset, VI is bounded by $2log2 = 2$. So, additive perturbation gives nearly the worst mining result of clustering (1.708). In Shuttle dataset, VI is bounded by $2log7 = 5.61$. So, addictive perturbation gives a very bad mining result too (3.7063). Similarly, VI of addictive perturbation is poor in the Wine dataset. So, although addictive perturbation can preserve more privacy, the noises introduced are too large that it heavily damages the clustering result. We remark addictive perturbation is not able to preserve the clustering result and hence is not suitable in the problem of privacy-preserving clustering.

On the other hand, multiplicative perturbation has shown a comparable (a little bit higher in average) VI compared to our approach. It is because the principle components with large variants are preserved in the multiplicative perturbation approach. Only the principle components with a little variance are filtered. So, the perturbed dataset by multiplicative perturbation is in fact similar to the original dataset. So, it has the smallest MSE among the three approaches in all datasets. The privacy preserved by multiplicative perturbation is not as high as the other two approaches.

### 6.6 Experiment Results on Execution Time

| | Parameters | Time(s) |
|---|---|---|
| Multiplicative perturbation | 120 dimensions | 2.828 |
| | 80 dimensions | 2.344 |
| | 40 dimensions | 1.875 |
| | 20 dimensions | 1.626 |
| | 10 dimensions | 1.497 |
| Additive perturbation | | 5.031 |
| 1-$d$ Clustering Perturbation | $\mu$ = 3*2, $\theta$ = 0.5*6598 | 1.341 |

**Table 5.** Experiment Results on Dataset: Musk

The experiment results of perturbation time complexity on the Musk dataset is shown in table 5. It shows that our proposed approach is the fastest among the three approaches. It is because we have the lowest time complexity which is linear to the number of objects in the dataset and the number of dimensions. If the number of dimension is increased, the difference in execution time will be further widened. Multiplicative perturbation has a comparable execution time when the number of dimensions preserved is reduced to a smaller value like 10. It is because the time complexity of random projection technique is $O(nde)$, where $e$ is the number of dimensions preserved. So, when $e$ is 10, the performance will be comparable to our proposed algorithm. However, as more dimensions are reduced, more information is lost in the original dataset, which will result in poorer clustering result. Note that the random projection itself is a less accurate projection method than the projection based on PCA, and in the previous section we have done experiments to show the accuracy of the PCA-based projection.

## 7 Conclusions and Future Work

In this paper we proposed a solution with high accuracy and low time complexity to the problem of privacy-preserving clustering. Besides, we proposed a new distance-based attack model to the distance-preserving perturbation, which strengthened our motivation to find solutions using perturbations that do not preserve pairwise distances. Previous approaches such as perturbation using additive random noises can largely distort the clustering result. In order to improve the accuracy, our approach takes into account the distribution of the original data by doing 1-$d$ clustering on each dimension and then perturbs it using random noise. Another drawback of previous approaches is the high time complexity, especially when dealing with high dimensional data. Our approach is linear with respect to both the cardinality and dimensionality, i.e. $O(nd)$, thus its scalability to large and high dimensional dataset is good. The performance study on real datasets shows our approach reaches good accuracy and causes low time overhead compared with previous approaches.

As future work, we plan to study the applicability of our approach to general problems of data publishing, such as range queries and aggregate queries.

## References

1. Agrawal, R., Asonov, D., Kantarcioglu, M. and Li Y.: Sovereign Joins. In: IEEE ICDE (2006)
2. Artin, A.: Algebra. Prentice Hall, New Jersey (1991)
3. Bingham, E. and Mannila, H.: Random Projection in Dimensionality Reduction: Applications to Image and Text Data. In: ACM SIGKDD (2001)
4. Drineas, P., Frieze A., Kannan R., Vempala S., and Vinay V.: Clustering Large Graphs via the Singular Value Decomposition. In: Machine Learning., 56, 9-33, (2004)
5. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kauffmann, San Francisco (2001)
6. Huang, Z., Du, W., and Chen, B.: Deriving Private Information from Randomized Data. In: ACM SIGMOD (2005)
7. ISO/IEC 14882:2003(E), Page 562, (2003)
8. Jagannathan, G., and Wright, R.: Privacy-Preserving Distributed $k$-Means Clustering over Arbitrarily Partitioned Data. In: ACM KDD (2005)
9. Johnson, W. and Lindenstrauss, J.: Extensions of Lipshitz Mapping Into Hilbert Space. In: Proc.of the Conference in Modern Analysis and Probability, pages 189-206, volume 26 of Contemporary Mathematics (1984)
10. Liu, K., Giannella C. and Kargupta H.: An Attacker's View of Distance Preserving Maps for Privacy Preserving Data Mining. In: PKDD (2006)
11. Liu, K. and Kargupta H.: Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining. In: IEEE TKDE Vol. 18 No. 1 (2006)
12. Liu, K., Giannella C. and Kargupta H.: A survey of Attack Techniques on Privacy-Preserving Data Perturbation Methods. In: Privacy-Preserving Data Mining: Models and Algorithms (2007)
13. Lloyd, S.: Least Squares Quantization in Pcm. In: IEEE Transactions on Information Theory, VOL. IT-28, NO.2 :129-136 (1982)
14. Meila, M.: Comparing Clusterings - An Axiomatic View. In: ICML (2005).
15. Oliveira, S. and Zaiane, O.: Privacy Preserving Clustering By Data Transformation. In: Proceedings of the 18th Brazilian Symposium on Databases, pages 304-318 (2003).
16. Ward, J., Jr.: Hierarchical Grouping to Optimize an Objective Function. In: Journal of the American Statistical Association, Vol. 58, No. 301 (1963).
17. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/.
18. Vaidya, J. and Clifton, C.: Privacy Preserving K-Means Clustering over Vertically Partitioned Data. In: ACM SIGKDD (2003).
19. Yao, A.: How to Generate and Exchange Secrets. In: Proceedings of. the 27th IEEE FOCS, pages 162-167 (1986).