Privacy Preserving Distributed DBSCAN Clustering*

Jinfei Liu[1], Li Xiong[1], Jun Luo[2], Joshua Zhexue Huang[2]

[1] Department of Mathematics & Computer Science, Emory University, 30322, USA

[2] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, 518055, China

E-mail: `jinfei.liu@emory.edu`, `lxiong@mathcs.emory.edu`, `jun.luo@siat.ac.cn` and `zx.huang@siat.ac.cn`

Abstract. DBSCAN is a well-known density-based clustering algorithm which offers advantages for finding clusters of arbitrary shapes compared to partitioning and hierarchical clustering methods. However, there are few papers studying the DBSCAN algorithm under the privacy preserving distributed data mining model, in which the data is distributed between two or more parties, and the parties cooperate to obtain the clustering results without revealing the data at the individual parties. In this paper, we address the problem of two-party privacy preserving DBSCAN clustering. We first propose two protocols for privacy preserving DBSCAN clustering over horizontally and vertically partitioned data respectively and then extend them to arbitrarily partitioned data. We also provide performance analysis and privacy proof of our solution.

Keywords: Privacy Preserving; DBSCAN; Clustering; Secure Multi-Party Computation; Distributed Data

1 Introduction

Consider a scenario in which each hospital has its own database of medical records. If the data from different hospitals can be shared, we can mine the data and extract more meaningful results than just using partial data independently. However, since medical records are subject to privacy and confidentiality constraints, how to extract the knowledge from the integrated data set without revealing one party's data to other parties is the theme of privacy preserving distributed data mining. Privacy preserving data mining, first introduced by Agrawal et al. [2] (using perturbation techniques) and Lindell et al. [16] (using Secure Multiparty Computation (SMC)) techniques), allows different parties to cooperate in the extraction of knowledge without any of the cooperating parties having to reveal their individual data items. In this paper, for the convenience of analysis, we only focus on the two-party privacy preserving data mining (PPDM) algorithm. However, the two-party algorithm can be extended to multi-party cases.

*A preliminary version of this paper appeared in the Proceedings of the 2012 Joint EDBT/ICDT Workshops [17].

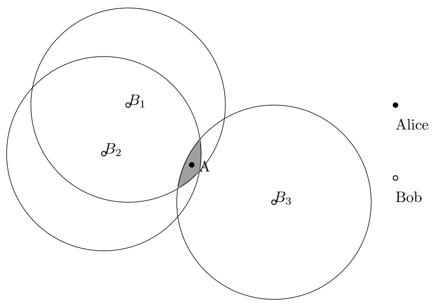


Figure 1: Bob could decide that A is in the small gray region.

DBSCAN (Density Based Spatial Clustering of Applications with Noise) [8] is a well-known density-based clustering algorithm which offers several advantages compared to partitioning and hierarchical clustering methods. First, DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means clustering algorithm. Second, DBSCAN is better at finding arbitrarily shaped clusters and can even find a cluster completely surrounded by a different cluster. Finally, DBSCAN has a notion of noise or outliers and requires just two parameters (Eps and $Minpts$ in this paper) and is mostly insensitive to the ordering of the points in the database. Conclusively, it is of practical importance to design distributed protocols DBSCAN based clustering in a distributed environment with sensitive data.

Kumar et al. [14] proposed protocols for the privacy preserving distributed DBSCAN clustering. However, the privacy proofs of their protocols do not strictly follow the definition of privacy in the semi-honest model [10]. More importantly, it poses significant privacy risks of identifying individual records from other party. Considering a scenario as shown in Figure 1. Bob has three records B_1, B_2, B_3 and knows one of Alice's record A is in their neighborhood. So Bob knows that A is in the intersection of his three records' neighborhood (the gray area in Figure 1). It could happen that the intersection area is so small that Bob could determine the location of the point A that is not far away from its true location.

In this paper, we present privacy preserving algorithms for DBSCAN clustering for horizontally, vertically and arbitrarily partitioned data distributed between two parties. The main contributions of our paper are:

1. We design a Multiplication Protocol (Subsection 4.1) based on Pailler's Additive Homomorphic cryptosystem [17].

2. We present a set of privacy preserving distributed DBSCAN clustering protocols utilizing the above multiplication protocol over horizontally (Subsection 4.2), vertically (Subsection 4.3), and arbitrarily (Subsection 4.4) partitioned data [17]. Compared to the existing work [8], the only information disclosed in our protocols is that Bob only knows there is a record owned by Alice in B_1, B_2, B_3's neighborhood respectively. However, Bob does not know whether those three records are the same or not. Therefore, Bob can not determine whether there is a record owned by Alice in the small gray region.

3. We further present a new protocol with enhanced privacy than the above protocols (Section 5). Compared to the protocols in the preliminary version [17] which reveal

the number of points from the other party in the neighborhood of a point when deciding whether the point is a core point, we address this disclosure in this new protocol.

4. We present formal performance and privacy analysis and demonstrate that our protocols are efficient in terms of communication and achieve adequate privacy.

The rest of the paper is organized as follows. We review some related work in Section 2. Section 3 introduces some background that will be used in this paper. The preliminary algorithms for computing privacy preserving distributed DBSCAN clustering are given in Section 4. Section 5 presents the new enhanced protocols addressing the disclosure problem in the previous protocols. Section 6 concludes the paper with directions for future work.

2 Related Work

Privacy preserving data mining, first introduced by Agrawal et al. [2] and Lindell et al. [16], could be roughly classified into two major categories: data mining on modified or perturbed data [2] [9] [1] [15] and cooperative data mining on distributed private data [16] [22] [13]. The initial idea of the former is to modify or perturb the data to mask private information but in such a way that the perturbed data can be still mined. The key is how to modify the data and how to recover the data mining result from the modified ("unprivate") data.

The privacy preserving distributed data mining problem in the latter category is typically formulated as a secure multi-party computation problem [10]. Yao's general protocol for secure circuit evaluation [26] can be used to solve any two-party privacy preserving distributed data mining problem in theory. However, since data mining problems usually involve millions or billions of data items, the communication cost of this protocol renders it impractical for these purposes. This is a good motivation to the search for problem specific protocols that are efficient in terms of communication complexity. Therefore, as Goldreich points out in [10], we do not recommend using the solutions derived by these general results in practice; special solutions should be developed for special cases or data mining algorithms for efficiency reasons. In many cases, including our solution described in this paper, the more efficient solutions still make use of a general solution such as Yao's, but only to carry out a much smaller portion of the computation. The rest of the computation uses other methods to ensure the privacy of the data. One such complementary approach uses randomization techniques [7]. Although such solutions tend to be more efficient than cryptographic solutions, they are generally less private or less accurate.

DBSCAN [8] is a well-known density-based clustering algorithm for discovering clusters in large spatial databases with noise. It is significantly more effective in discovering clusters of arbitrary shape than the partitioning methods such as the well known k-means [19] and CLARANS [18] algorithm as well as hierarchical methods. Similar clustering algorithms based on density are OPTICS [4] and DENCLUE [11]. While there are many privacy preserving K-means algorithms [12] [23] [5], there is little literature considering the problem of privacy preserving distributed density-based clustering. A. Amirbekyan et al. [3] and W. Xu et al. [24] only discussed the vertically partitioned data case. So far [14] is the only both horizontally and vertically partitioned data work we are aware of but it did not provide adequate privacy as we discussed above.

3 Preliminaries

In this section, we briefly review the DBSCAN clustering algorithm, and describe the concept of horizontally, vertically and arbitrarily partitioned data. Some definitions borrowed from cryptology, and two protocols that will be used in this paper are also given.

3.1 DBSCAN Algorithm

We briefly review the DBSCAN algorithm. Details are described in [8]. DBSCAN is a density-based algorithm which can detect arbitrary shaped clusters. The key idea is that for each point of a cluster, the neighborhood of which within a given radius (*Eps*) has to contain at least a minimum number (*MinPts*) of points, i.e. the density in the neighborhood has to exceed some threshold. Therefore, the critical step in this algorithm is to decide whether the distance between two points is less than or equal to *Eps*. It is an easy task if two points are owned by one party. Otherwise, we need to develop a private protocol to compute the distance between two points that are owned by two parties.

We illustrate several definitions in DBSCAN algorithm [8] that will be used in the next Section.

Definition 1. (density-reachable) A point p is density-reachable from a point q w.r.t. Eps and $MinPts$ if there is a chain of points $p_1, ..., p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

Definition 2. (density-connected) A point p is density-connected to a point q w.r.t. Eps and $MinPts$ if there is a point o such that both p and q are density-reachable from o w.r.t. Eps and $MinPts$.

Definition 3. (cluster) Let D be a database of points. A *cluster* C w.r.t. Eps and $MinPts$ is a non-empty subset of D satisfying the following conditions:

1. $\forall$ point p, q: if $p \in C$ and q is density-reachable from p w.r.t. Eps and MinPts, then $q \in C$. (Maximality)
2. $\forall$ point p,q $\in C$: p is density-connected to q w.r.t. Eps and MinPts. (Connectivity)

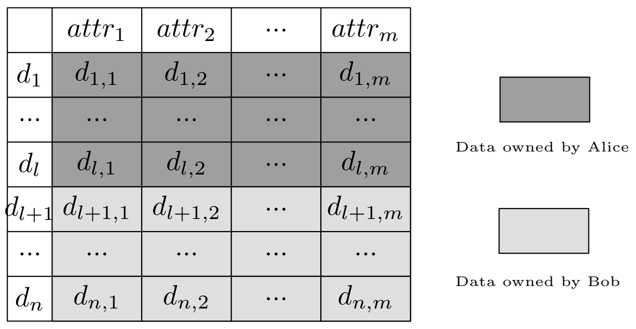
Definition 4. (noise) Let $C_1, ..., C_k$ be the clusters of the database D w.r.t. parameters Eps and $MinPts$. Then we define the *noise* as the set of points in the database D not belonging to any cluster C_i , i.e. noise $=p \in D \mid \forall i : p \notin C_i$.

3.2 Partitioned Data

In the two-party distributed data setting, two parties (call them Alice and Bob) hold data forming a (virtual) database consisting of their joint data. More specifically, the virtual database $D = \{d_1, d_2, ..., d_n\}$ consists of n records. Each record d_i has m values for m attributes $(d_{i,1}, d_{i,2}, ..., d_{i,m})$.

There are three formats for partitioned data:

- **Horizontally Partitioned Data**: each party owns a subset of records with full attributes (see Figure 2).
- **Vertically Partitioned Data**: each party owns all records with partial attributes (see Figure 3).

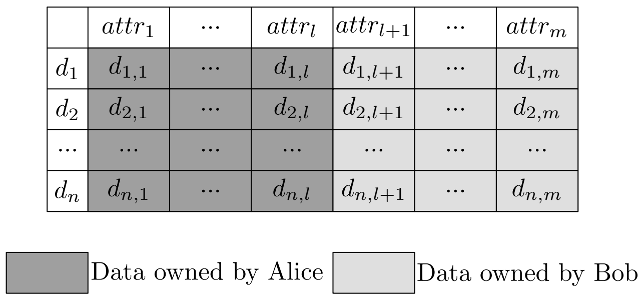


	$attr_1$	$attr_2$	$\cdots$	$attr_m$
d_1	$d_{1,1}$	$d_{1,2}$	$\cdots$	$d_{1,m}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
d_l	$d_{l,1}$	$d_{l,2}$	$\cdots$	$d_{l,m}$
d_{l+1}	$d_{l+1,1}$	$d_{l+1,2}$	$\cdots$	$d_{l+1,m}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
d_n	$d_{n,1}$	$d_{n,2}$	$\cdots$	$d_{n,m}$

▨ Data owned by Alice

□ Data owned by Bob

Figure 2: Horizontally partitioned data.



	$attr_1$	$\cdots$	$attr_l$	$attr_{l+1}$	$\cdots$	$attr_m$
d_1	$d_{1,1}$	$\cdots$	$d_{1,l}$	$d_{1,l+1}$	$\cdots$	$d_{1,m}$
d_2	$d_{2,1}$	$\cdots$	$d_{2,l}$	$d_{2,l+1}$	$\cdots$	$d_{2,m}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
d_n	$d_{n,1}$	$\cdots$	$d_{n,l}$	$d_{n,l+1}$	$\cdots$	$d_{n,m}$

▨ Data owned by Alice □ Data owned by Bob

Figure 3: Vertically partitioned data.

- **Arbitrarily Partitioned Data** [12]: mixture of horizontally and vertically partitioned data (see Figure 4).

3.3 Privacy Properties

Our desired outcome is that clusters are computed on Alice and Bob's joint data. Alice should learn the cluster number (or the NOISE record) for each data record that she owns completely, and Bob should learn the cluster number (or the NOISE record) for each data record that he owns completely. For records that are split between Alice and Bob, they should both learn the cluster number.

In an ideal world, Alice and Bob would have access to a trusted third party who could perform the necessary calculations. Alice and Bob would securely send their data to this trusted party, which would then compute the cluster to which each object is assigned using the appropriate clustering algorithm, and send the desired information to the party that owns the object. However, trusted third parties are hard to find in the real world, and even then, such trust may be misplaced. In this work, we do not rely on a third party. Instead, we provide protocols by which Alice and Bob can carry out the functionality that would be provided by the trusted third party without actually requiring such third party or requiring the parties to send their data to each other.

	$attr_1$	$attr_2$	$attr_3$	$attr_4$
d_1	▨ $d_{1,1}$	▢ $d_{1,2}$	▨ $d_{1,3}$	▨ $d_{1,4}$
d_2	▨ $d_{2,1}$	▢ $d_{2,2}$	▢ $d_{2,3}$	▢ $d_{2,4}$

$=$

	$attr_1$	$attr_2$
d_1	▨ $d_{1,1}$	▢ $d_{1,2}$
d_2	▨ $d_{2,1}$	▢ $d_{2,2}$

$+$

	$attr_3$	$attr_4$
d_1	▨ $d_{1,3}$	▨ $d_{1,4}$
d_2	▢ $d_{2,3}$	▢ $d_{2,4}$

▨ Data owned by Alice ▢ Data owned by Bob

Figure 4: Arbitrarily partitioned data =vertically partitioned data + horizontally partitioned data.

3.4 Two-party Computation

A two-party protocol problem [10] is casted by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a *functionality* and denote it as $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$. That is, for every pair of inputs (x, y), the output pair is a random variable $(f_1(x, y), f_2(x, y))$ ranging over pairs of strings. The first party (with input x) wishes to obtain $f_1(x, y)$ and the second party (with input y) wishes to obtain $f_2(x, y)$. We often denote such a functionality by $(x, y) \mapsto (f_1(x, y), f_2(x, y))$.

3.5 Semi-honest Model

In any multi-party computation setting, a *malicious* adversary can always alter its input. In the data mining setting, this fact can be very damaging since the adversary can define its input to be the empty database. Then the output obtained is the result of the algorithm on the other party's database alone. For this work we assume that the adversary is *semi-honest* [10] (also known as *passive*). That is, it correctly follows the protocol specification, yet attempts to learn additional information by analyzing the transcript of messages received during the execution. We remark that although the semi-honest adversarial model is weaker than the malicious model (where a party may arbitrarily deviate from the protocol specification), it is often a realistic one. This is because deviating from a specified program which may be buried in a complex application is a non-trivial task. Further, the parties cooperate with each other because they wish to mine the integrated data for their mutual benefit.

3.6 Definition of Privacy in the Semi-honest Model

Intuitively, a protocol is private if whatever can be computed by a party participating in the protocol can be computed based on its input and output only. This is formalized according to the simulation paradigm. Loosely speaking, we require that a party's view in a protocol execution be simulative given only its input and output. This then implies that the parties learn nothing from the protocol execution itself, as desired.

We begin with the following notations:

• Let $f = (f_1, f_2)$ be a probabilistic, polynomial-time functionality and let $\prod$ be a two-party protocol for computing f.

• The view of the first (resp., second) party during an execution of $\prod$ on (x, y), denoted by $view_1^{\prod}(x, y)$(resp., $view_2^{\prod}(x,\ y)$), is $(x, r^1, m_1^1, ..., m_t^1)$ (resp., $(y, r^2, m_1^2, ..., m_t^2)$) where r^1 (resp., r^2) represents the outcome of the first (resp., second) party's internal coin tosses, and m_i^1 (resp., m_i^2) represents the ith message it has received.

• The output of the first (resp., second) party during an execution of $\prod$ on (x, y) is denoted by $output_1^{\prod}(x, y)$ (resp., $output_2^{\prod}(x, y)$), and is implicit in the party's view of the execution.

Definition 5. Privacy with Respect to Semi-Honest Behavior
For a functionality f, we say that $\prod$ privately computes f if there exist probabilistic polynomial time algorithms, denoted by S_1 and S_2, such that

$$\{(S_1(x, f_1(x, y)), f_2(x, y))\}_{x,y\in\{0,1\}^*} \overset{c}{\equiv} \{(view_1^{\prod}(x, y), output_2^{\prod}(x, y))\}_{x,y\in\{0,1\}^*}, \quad (1)$$

$$\{(f_1(x, y), S_2(y, f_2(x, y)))\}_{x,y\in\{0,1\}^*} \overset{c}{\equiv} \{(output_1^{\prod}(x, y), view_2^{\prod}(x, y))\}_{x,y\in\{0,1\}^*}, \quad (2)$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability.

Equations 1 and 2 state that the view of a party can be simulated by a probabilistic polynomial-time algorithm given access to the party's input and output only. We emphasize that the adversary here is semi-honest and therefore the view is exactly according to the protocol definition. See [10] for a full discussion.

Theorem 6. *(Composition Theorem for the semi-honest model, two parties) [6]: Suppose that g is privately reducible to f and that there exists a protocol for privately computing f. Then there exists a protocol for privately computing g.*

3.7 Paillier's Additive Homomorphic Properties

We briefly describe the Paillier's additive homomorphic cryptosystem [20] and its homomorphic properties which will be used in our protocols.

Key generation
• Choose two large prime numbers p and q randomly and independently of each other such that $gcd(pq, (p-1)(q-1)) = 1$.
• Compute $n = pq$ and $\lambda = lcm(p-1, q-1)$.
• Select random integer g where $g \in \mathbb{Z}_{n^2}^*$.
• Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: $\mu = (L(g^\lambda \, mod \, n^2))^{-1} \, mod \, n$, where function L is defined as $L(u) = \frac{u-1}{n}$.
• The public encryption key is (n, g) and private decryption key is (λ, u).
Encryption
• Let m be a plaintext to be encrypted where $m \in \mathbb{Z}_n$.
• Select random r where $r \in \mathbb{Z}_n^*$.
• Compute ciphertext as: $c = g^m r^n \, mod \, n^2$.
Decryption
• Ciphertext $c \in \mathbb{Z}_{n^2}^*$.
• Compute plaintext: $m = L(c^\lambda \, mod \, n^2)\mu \, mod \, n$.
Homomorphic properties
• Homomorphic addition of plaintexts:

$$D(E(m_1, r_1)E(m_2, r_2) \, mod \, n^2) = (m_1 + m_2) \, mod \, n$$

• Homomorphic multiplication of plaintexts:

$$D(E(m_1, r_1)^{m_2} \, mod \, n^2) = m_1 m_2 \, mod \, n$$

3.8 Yao's Millionaires' Problem Protocol (YMPP)

We also briefly describe the Yao's Millionaires' Problem Protocol (YMPP) which will be used in our protocol. Alice has i millions and Bob has j millions, where $1 < i, j < n'$ and n' is the limit value of i, j. Yao's millionaires' problem [25] decides whether $i < j$, such that this is also the only thing they know in the end. The protocol is described in Algorithm 1.

Algorithm 1 Yao's Millionaires' Problem Protocol

```
Input:Alice inputs $i$, Bob inputs $j$.
Output:Alice and Bob know whether $i < j$ and cannot get any more information about the wealth of the other party.
1: Bob picks a random $N$-bits integer, and computes privately the value of $E_a(x)$; call the result $k$.
2: Bob sends Alice the number $k - j + 1$;
3: Alice computes privately the values of $y_u = D_a(k - j + u)$ for $u = 1, 2, ..., n'$.
4: Alice generates a random prime $p$ of $N/2$ bits, and computes the values $z_u = y_u$(mod $p$) for all u; if all $z_u$ differ by at least 2 in the mod $p$ sense, stop; otherwise generates another random prime and repeat the process until all $z_u$ differ by at least 2; let $p, z_u$ denote this final set of numbers;
5: Alice sends the prime $p$ and the following $n'$ numbers to $B : z_1, z_2, ..., z_i$ followed by $z_i + 1, z_{i+1} + 1, ..., z_{n'} + 1$; the above numbers should be interpreted in the mod $p$ sense.
6: Bob looks at the $j$-th number (not counting $p$) sent from Alice, and decides that $i \geq j$ if it is equal to $x$ mod $p$, and $i < j$ otherwise.
7: Bob tells Alice what the conclusion is.
```

4 Privacy Preserving Distributed DBSCAN Clustering

In this section, we first introduce our Multiplication Protocol, then extend the single party DBSCAN algorithm of [8] to privacy preserving distributed DBSCAN clustering on horizontally, vertically and arbitrarily partitioned data, respectively. We also provide analysis of the communication complexity and privacy proofs of our protocols.

4.1 Multiplication Protocol

Multiplication Problem can be described as: Alice inputs private number x and Bob inputs private number y, Alice (but not Bob) is to get the result $u = xy + v$, where v is a random number known only to Bob. Alice should not be able to derive the value of y from u and the execution of the protocol. Similarly Bob should not be able to get the result of xy or the value of x.

Multiplication Problem is a special case of scalar products problems, but the known scalar products protocols are not fit for the Multiplication Problem. In our method, Paillier's additive homomorphic encryption schemes [20] are used to solve the Multiplication Problem. First, Alice generates a key pair. Then, both Alice and Bob transform their inputs to positive integers. After that, Alice sends the disguised data and the public key to Bob, Bob repeats multiplying the disguised data and adds a random number to disguise it, then returns it to Alice. Finally, Alice can use the private key to decode the value and get the desired result (details are given in Algorithm 2).

```
Algorithm 2 Multiplication protocol
Input:Alice inputs $x$, Bob inputs $y$.
Output:Alice gets $u = xy + v$ where $v$ is a random number known to Bob only.
1: Alice generates a pair keys $(E_A, D_A)$ in homomorphic encryption schemes.
2: Alice and Bob collaborate to select a random $r \in \mathbb{Z}_n^*$.
3: Alice sends $E_A(x, r)$, $E_A$ and $r$ to Bob.
4: Bob generates a random $v$.
5: Bob computes $u' = (E_A(x, r))^y \times E_A(v, r)$.
6: Bob sends $u'$ to Alice.
7: Alice computes $u = D_A(u')$.
```

4.1.1 Correctness Proof

$$u' = (E_A(x, r))^y \times E_A(v, r)$$

Based on the Pallier's homomorphic properties,

$$u = D_A(u') = D_A((E_A(x, r)^y \times E_A(v, r)) = xy + v$$

Lemma 7. Multiplication Protocol is secure.

Proof. We start by presenting a simulator for the Alice's view. In the protocol, the only information Alice received from Bob is the u' in Step 6. Alice can simulate the y and v by y' and v' generated from a single random number with uniform random distribution. That is, on Alice's view, the simulated view $(E_A(x, r))^{y'} \times E_A(v', r)$ is computationally indistinguishable from the u' that Alice had received in Step 6.

We now turn to the Bob's view. Bob receives an encryption key E_A and a encrypted value $E_A(x)$. Bob can simulate the encryption key by generating a single random number from a uniform random distribution and the encrypted value can also be simulated simply by generating a random from an uniform distribution.

The simulator for both runs in linear time, which meets the requirement of a polynomial time simulation. □

4.2 Privacy Preserving Distributed DBSCAN Clustering over Horizontally Partitioned Data

Distance protocol for horizontally partitioned data. We first present a distance protocol (HDP) for evaluating the distance between two horizontally split records. Consider the horizontally partitioned data as shown in Figure 2. Let d_y $(l + 1 \leq y \leq n)$ denote one record of Bob and d_x $(1 \leq x \leq l)$ denote one record of Alice. The purpose of this protocol is to let Alice know whether $dist(d_x, d_y)$ is smaller than Eps, without obtaining any knowledge about Bob's $d_{y,1}, d_{y,2}, ..., d_{y,m}$ and Bob could not obtain any knowledge about Alice's $d_{x,1}, d_{x,2}, ..., d_{x,m}$. We can evaluate whether $dist(d_x, d_y)^2 < Eps^2$, so that $dist(d_x, d_y) < Eps$ can be obtained.

$(dist(d_x, d_y))^2 = (d_{y,1} - d_{x,1})^2 + (d_{y,2} - d_{x,2})^2 + ... + (d_{y,m} - d_{x,m})^2$

$$= \overbrace{d_{x,1}^2 + d_{x,2}^2 + ... + d_{x,m}^2}^{Owned\ by\ Alice} +$$

$$\overbrace{d_{y,1}^2 + d_{y,2}^2 + ... + d_{y,m}^2 - 2(d_{x,1}d_{y,1} + ... + d_{x,m}d_{y,m})}^{Owned\ by\ Alice\ and\ Bob}$$

For the part owned by Alice and Bob,

$$d_{y,1}^2 + d_{y,2}^2 + ... + d_{y,m}^2 - 2(d_{x,1}d_{y,1} + ... + d_{x,m}d_{y,m})$$

Alice generates m random numbers $r_1, r_2, ..., r_m$ such that

$$r_1 + r_2 + ... + r_m = 0$$

then

$$(d_{y,1}^2 + d_{y,2}^2 + ... + d_{y,m}^2) - 2\{d_{x,1}d_{y,1} + ... + d_{x,m}d_{y,m}\}$$
$$= (d_{y,1}^2 + d_{y,2}^2 + ... + d_{y,m}^2) - 2\{d_{x,1}d_{y,1} + r_1 + ... + d_{x,m}d_{y,m} + r_m\}$$

Alice and Bob use the Multiplication Protocol to let Bob get $d_{x,1}d_{y,1}+r_1+...+d_{x,m}d_{y,m}+r_m$. Then Alice and Bob use the Protocol YMPP to decide whether $dist(d_x, d_y)^2 \leq Eps^2$.

Lemma 8. Protocol HDP is secure.

Proof. Besides the Protocol YMPP, the only communication is that Alice sends $r_1, r_2, ..., r_m$ to Bob. In Bob's view, he could simulate the $r_1', r_2', ..., r_m'$ by generating m random numbers from a uniform random distribution. The simulator runs in linear time, which meets the requirement for a polynomial time simulation. □

4.2.1 DBSCAN Algorithm for Horizontally Partitioned Data

Algorithm 3 DBSCAN(SetOfPointsOfAlice, SetOfPointsOfBob, Eps, MinPts)

```
1: Party Alice DOES:
2: //SetOfPointsOfAlice is UNCLASSIFIED
3: ClusterId:=nextId(NOISE);
4: For i FROM 1 TO SetOfPointsOfAlice.size DO
5:   Point:=SetOfPointsOfAlice.get(i);
6:   IF Point.ClusterId=UNCLASSIFIED THEN
7:       IF ExpandCluster(SetOfPointsOfAlice, SetOfPointsOfBob, PointOfAlice, PointOfBob, ClusterId, Eps,
     MinPts) THEN
8:         ClusterId:=nextId(ClsuterId);
9:       END IF;
10:   END IF;
11: END FOR;
12: END;
13: Party B DOES: repeats step 1 to 12 by replacing Alice for Bob and Bob for Alice.
```

The details of the DBSCAN algorithm for horizontally partitioned data are given in Algorithm 3. $SetOfPointsOfAlice$ is either the completed database of Alice's or a discovered cluster from a previous run of Alice. Eps and $MinPts$ are the global density parameters. A call of $SetOfPointsOfAlice.\ regionQuery$ ($PointOfAlice$, Eps) returns the Eps-Neighborhood of $PointOfAlice$ in $SetOfPointsOfAlice$ as a list of points. The function $SetOfPointsOfAlice.\ get(i)$ returns the i-th element of $SetOfPointsOfAlice$. The most important function used by DBSCAN is ExpandCluster which is presented in Algorithm 4. $SetOfPointsOfBobPermutation$ is $SetOfPointsOfBob$. But in each iteration, the points' order is permutated randomly. Therefore, Alice only knows there is a point of Bob (but does not know which point of Bob) in neighborhood of one of her points. In this way, the situation in Figure 1 can be avoided. What should be noted is that in Steps 3 and 13, only Alice knows whether the point in $SetOfPointsOfBobPermutation$ is seed. Otherwise, Bob could decide Alice in a small intersection region as we argued in Figure 1. Explain above again by replacing Alice for Bob and Bob for Alice.

Algorithm 4 ExpandCluster(SetOfPointsOfAlice, SetOfPointsOfBob, PointOfAlice, PointOfBob, ClusterId, Eps, MinPts) : Boolean

```
1: Party Alice DOES:
2: seeds$_A$:=SetOfPointsOfAlice.regionQuery(PointOfAlice,Eps);
3: seeds$_B$:=SetOfPointsOfBobPermutation.regionQuery(PointOfAlice,Eps);//Alice and Bob cooperatively use Protocol HDP
4: IF seeds$_A$.size+seeds$_B$.size<MinPts THEN
5:   SetOfPointsofAlice.changeClusterId(PointOfAlice,NOISE);
6:   RETURN False;
7: ELSE
8:   SetOfPointsOfAlice.changeClusterIds(seeds$_A$,ClusterId);
9:   seeds$_A$.delete(PointOfAlice);
10:   WHILE seeds$_A$ $\neq$ Empty DO
11:     currentP:=seeds$_A$.first();
12:     result$_A$:=SetOfPointsOfAlice.regionQuery(currentP,Eps);
13:     result$_B$:=SetOfPointsOfBobPermutation.regionQuery(currentP,Eps);//Alice and Bob cooperatively use Protocol HDP
14:     IF result$_A$.size+result$_B$.size$\geq$MinPts THEN
15:       FOR i FROM 1 TO result$_A$.size DO
16:         result$_A$P:=result$_A$.get(i);
17:         IF result$_A$P.ClusterId IN {UNCLASSIFIED,NOISE} THEN
18:           IF result$_A$P.ClusterId=UNCLASSIFIED THEN
19:             seeds$_A$.append(result$_A$P);
20:           END IF;
21:           SetOfPointsOfAlice.changeClusterId(result$_A$P,ClusterId);
22:         END IF; //UNCLASSIFIED or NOISE
23:       END FOR;
24:     END IF; //result.size$\geq$MinPts
25:     seeds$_A$.delete(currentP);
26:   END WHILE; //seeds$_A$ $\neq$ Empty
27:   RETURN True;
28: END IF;
29: END; //ExpandCluster
30: Party B DOES: repeats step 1 to 29 by replacing Alice for Bob and Bob for Alice.
```

4.2.2 Communication Complexity and Privacy

For each record d_i ($1 \leq i \leq n$), it has m components. Assume that each component is represented by c_1 bits. Communication is involved when the two parties are engaged in the Multiplication Protocol to judge the core point. The communication complexity of each Multiplication Protocol is $O(c_1)$. Assume that each number in Protocol YMPP is represented by c_2 bits. Hence, it requires a communication of $O(c_1ml(n-l)+c_2n'l(n-l))$ bits to execute the algorithm, where $n' = |u|$ and l is the number of records owned by one party.

Theorem 9. *Algorithm 3 privately computes the privacy preserving distributed DBSCAN clustering over horizontally partitioned data assuming semi-honest, revealing the number of points from the other party in the neighborhood of this point.*

Proof. Step 3 and Step 13 are the only two steps of Algorithm 4 requiring communication. As Lemma 8 has been proved that protocol HDP is private, by applying the composition theorem (Theorem 6), we can conclude that the privacy preserving distributed DBSCAN clustering over horizontally partitioned data is private. □

4.3 Privacy Preserving Distributed DBSCAN Clustering over Vertically Partitioned Data

Distance protocol for vertically partitioned data. We first present a distance protocol (VDP) for evaluating the distance between two vertically split records. Consider the vertically partitioned data as shown in Figure 3. Let d_y denote one record and $d_{y,1}, d_{y,2}, ..., d_{y,l}$ are owned by Alice while $d_{y,l+1}, d_{y,l+2}, ..., d_{y,m}$ are owned by Bob. Let d_x $(x \neq y)$ denote another record and $d_{x,1}, d_{x,2}, ..., d_{x,l}$ owned by Alice while $d_{x,l+1}, d_{x,l+2}, ..., d_{x,m}$ are owned by Bob. The purpose of this protocol is to let Alice and Bob know whether $dist(d_x, d_y)$ smaller than Eps. At the same time, Alice (Bob) knows nothing about Bob's (Alice's) data, respectively. In order to determine

$$(dist(d_x, d_y))^2 = \overbrace{(d_{x,1} - d_{y,1})^2 + ... + (d_{x,l} - d_{y,l})^2}^{Owned\ by\ Alice} +$$
$$\overbrace{(d_{x,l+1} - d_{y,l+1})^2 + ... + (d_{x,m} - d_{y,m})^2}^{Owned\ by\ Bob} \leq Eps^2$$

Bob obtained

$$Eps^2 - \{(d_{x,l+1} - d_{y,l+1})^2 + ... + (d_{x,m} - d_{y,m})^2\}$$

then use Protocol YMPP, Alice and Bob could decide whether $dist(d_x, d_y) \leq Eps$.

4.3.1 DBSCAN Algorithm for Vertically Partitioned Data

Algorithm 5 DBSCAN(SetOfPoints, Eps, MinPts)

```
1: //SetOfPoints is UNCLASSIFIED
2: ClusterId:=nextId(NOISE);
3: For i FROM 1 TO SetOfPoints.size DO
4:   Point:=SetOfPoints.get(i);
5:   IF Point.ClusterId=UNCLASSIFIED THEN
6:     IF ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) THEN
7:       ClusterId:=nextId(ClsuterId);
8:     END IF
9:   END IF
10: END FOR
11: END
```

The details of the DBSCAN algorithm for vertically partitioned data are given in Algorithm 5. $SetOfPoints$ is either the completed database or a discovered cluster from a previous run. Eps and $MinPts$ are the global density parameters. A call of $SetOfPoints.regionQuery(Point, Eps)$ returns the Eps-Neighborhood of Point in $SetOfPoints$ as a list of points. The function $SetOfPoints.get(i)$ returns the i-th element of $SetOfPoints$. The most important function used by DBSCAN is ExpandCluster which is presented in Algorithm 6.

4.3.2 Communication Complexity and Privacy

For each record d_i, it has m components. Communication is involved when the two parties are engaged in the Protocol YMPP to judge the core point. Assume that each number in Protocol YMPP is represented by c_2 bits. The communication complexity of each YMPP is $O(c_2 n')$ where $n' = |u|$ and there are $O(n^2)$ times of executing YMPP if we implement DBSCAN without spatial index. Hence, it requires a communication of $O(c_2 n' n^2)$ bits totally.

Algorithm 6 ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) : Boolean

```
seeds:=SetOfPoint.regionQuery(Point,Eps); //Alice and Bob cooperatively use Protocol VDP.
IF seeds.size<MinPts THEN //no core points
  SetOfPoints.changeClusterId(Point,NOISE);
  RETURN False;
ELSE
  SetOfPoints.changeClusterIds(seeds,ClusterId);
  seeds.delete(Point);
  WHILE seeds $\neq$ Empty DO
    currentP:=seeds.first();
    result:=SetOfPoints.regionQuery(currentP,Eps); //Alice and Bob cooperatively use Protocol VDP.
    IF result.size$\geq$MinPts THEN
      FOR i FROM 1 TO result.size DO
        resultP:=result.get(i);
        IF resultP.ClusterId IN {UNCLASSIFIED, NOISE} THEN
          IF resultP.ClusterId=UNCLASSIFIED THEN
            seeds.append(resultP);
          END IF;
          SetOfPoints.changeClusterId(resultP, ClusterId);
        END IF; //UNCLASSIFIED or NOISE
      END FOR;
    END IF; //result.size$\geq$MinPts
    seeds.delete(currentP);
  END WHILE; //seeds $\neq$ Empty
  RETURN True;
END IF;
END; //ExpandCluster
```

Theorem 10. *Algorithm 5 privately computes the privacy preserving distributed DBSCAN clustering over horizontally partitioned data assuming semi-honest, revealing the number of points in the neighborhood of this point.*

Proof. The key privacy of the algorithm 6 is the comparison of $dist(d_x, d_y)$ with Eps. This is guaranteed by Protocol YMPP. By applying the composition theorem (Theorem 6), we can conclude that the privacy preserving distributed DBSCAN clustering over vertically partitioned data is secure. The only information revealed is the output which must be known by Alice and Bob. □

4.4 Privacy Preserving Distributed DBSCAN Clustering Algorithm over Arbitrarily Partitioned Data

In arbitrarily partitioned data, there is not necessarily a simple pattern of how data are shared between the parties. For each record d_i $(1 \leq i \leq n)$, Alice knows the values for a subset of the attributes, and Bob knows the values for the remaining attributes. That is, each d_i is partitioned into disjointed subsets $Alice_{d_i}$ and Bob_{d_i} so that Alice knows $Alice_{d_i}$ and Bob knows Bob_{d_i}. Although extremely patchworked data is infrequent in practice, the generality of this model can make it better suited to practical settings in which data may be mostly, but not completely, vertically or horizontally partitioned.

As discussed in subsection 3.1, the key of DBSCAN is to judge $dist\{d_i, d_j\} \leq Eps$ for two records d_x, d_y owned by Alice and Bob, respectively. For example, in Figure 4, the problem is to let Alice and Bob cooperate to decide

$$\overbrace{(A_{2,1} - A_{1,1})^2 + (B_{2,2} - B_{1,2})^2}^{Vertically\ partitioned} + \overbrace{(B_{2,3} - A_{1,3})^2 + (B_{2,4} - A_{1,4})^2}^{Horizontally\ partitioned} \leq Eps^2$$

For the vertically partitioned data, $(A_{2,1} - A_{1,1})^2 = Alice_v$ is owned by Alice and $(B_{2,2} - B_{1,2})^2 = Bob_v$ is owned by Bob. For the horizontally partitioned data, we could process them using the Protocol HDP. That is, the horizontally partitioned data could be divided into data owned by Alice ($Alice_h$) and data owned by Bob (Bob_h). Then Alice and Bob use Protocol HDP to let Bob get $(B_{2,3} - A_{1,3})^2 + (B_{2,4} - A_{1,4})^2 = Bob_h$. Along with Bob_v, Bob knows $Bob_h + Bob_v$. Since Alice knows $Alice_v$, Alice and Bob could decide $dist\{d_i, d_j\} \leq Eps$ by using Protocol YMPP.

Since the arbitrarily partitioned data could be decomposed into horizontally and vertically partitioned data, with Theorem 6, the algorithm for the arbitrarily partitioned data is the combination of algorithms for horizontally and vertically partitioned data. Hence, we do not repeat the algorithm here.

5 Enhanced Protocol of Privacy Preserving DBSCAN Clustering over Horizontally Partitioned Data

In this section, we present an enhanced protocol with improved privacy for the protocol in Subsection 4.2. As it is stated in Theorem 9, the privacy preserving distributed DBSCAN clustering algorithm over horizontally partitioned data reveals the number of points in the neighborhood of a point. This enhanced protocol only reveals whether the number of the other party's points in his/her point's neighborhood is greater than $Minpts$ minus ||his/her points in this point's neighborhood||.

If Alice wants to decide whether the point is a core point, she only needs to know the distance between k^{th} smallest $Dist^2\{A, B_i\}$'s B_i and A is less than or equal to Eps, where k equals to $MinPts$ minus ||her points in this point's neighborhood||. Hence, this problem is decomposed into two sub-problems. One is how to compare the distances securely. The other one is how to find the k^{th} order statistic.

For how to compare the distances securely, assume Alice has one point A and Bob has n points $B_1, B_2, ..., B_n$. The distance between Alice A and Bob $B_i, 1 \leq i \leq n$ is:

$$Dist^2\{A, B_i\} = (A_1 - B_{i1})^2 + (A_2 - B_{i2})^2 + ... + (A_m - B_{im})^2$$

$$= (A_1^2 + A_2^2 + ... + A_m^2) - (2A_1B_{i1} + 2A_2B_{i2} + ... + 2A_mB_{im}) + (B_{i1}^2 + B_{i2}^2 + ... + B_{im}^2)$$

$$= (A_1^2 + A_2^2 + ... + A_m^2, -2A_1, -2A_2, ..., -2A_m, 1) \cdot (1, B_{i1}, B_{i2}, ..., B_{im}, B_{i1}^2 + B_{i2}^2 + ... + B_{im}^2)$$

Then use the Multiplication Protocol,

$$u_1 = Dist^2\{A, B_1\} + v_1$$

$$u_2 = Dist^2\{A, B_2\} + v_2$$

$$\cdots\cdots$$

$$u_n = Dist^2\{A, B_n\} + v_n$$

Alice knows $u_1, u_2, ..., u_n$ and Bob knows $v_1, v_2, ..., v_n$. Alice and Bob could use the Protocol YMPP to decide whether $Dist\{A, B_i\}$ is greater than $Dist\{A, B_j\}$ as follows, where $1 \leq i \neq j \leq n$.

$$Dist\{A, B_i\} - Dist\{A, B_j\} = (v_1 - u_1) - (v_2 - u_2) = (v_1 - v_2) - (u_1 - u_2)$$

Alice and Bob could use the Protocol YMPP to decide whether $(v_1 - v_2) - (u_1 - u_2)$ is greater than 0, i.e., whether $Dist\{A, B_i\}$ is greater than $Dist\{A, B_j\}$.

Then this problem turns to how to find the k^{th} smallest $Dist\{A, B_i\}, 1 \leq i \leq n$ in an array. We illustrate two different algorithms. The first algorithm is appropriate when the k is small. The algorithm scans the $Dist\{A, B_i\}, 1 \leq i \leq n$, the first iteration finds the smallest $Dist\{A, B_i\}, 1 \leq i \leq n$, then deletes the $Dist\{A, B_{smallest}\}$ and iteratively to find the smallest in the rest of $Dist\{A, B_i\}, i \neq smallest$. So the iteration continues until finding the k^{th} smallest number. The time complexity is $O(kn)$, which is a good time complexity for a small k. Though k is often very small, we here also illustrate an algorithm appropriate when the k is greater, quick sorted based algorithm [21]. The worst time complexity is just like that of quick sort $\Theta(n^2)$. The balanced time complexity is $O(n + n/2 + n/4 + ... + 1) = O(n)$.

When Alice knows the k^{th} smallest $Dist^2\{A, B_i\}$'s B_i. Alice and Bob could use the Protocol YMPP to decide whether $Dist\{A, B_i\}$ is smaller than or equal to Eps. If the $Dist\{A, B_i\} \leq Eps$, then the point A is a core point. The entire protocols are as follows (Algorithm 7):

Algorithm 7 DBSCAN(SetOfPointsOfAlice, SetOfPointsOfBob, Eps, MinPts)

```
1: Party Alice DOES:
2: //SetOfPointsOfAlice is UNCLASSIFIED
3: ClusterId:=nextId(NOISE);
4: For i FROM 1 TO SetOfPointsOfAlice.size DO
5:   Point:=SetOfPointsOfAlice.get(i);
6:   IF Point.ClusterId=UNCLASSIFIED THEN
7:     IF EnhancedExpandCluster(SetOfPointsOfAlice, PointOfAlice, ClusterId, Eps, MinPts) THEN
8:       ClusterId:=nextId(ClsuterId);
9:     END IF;
10:  END IF;
11: END FOR;
12: END;
13: Party B DOES: repeats step 1 to 12 by replacing Alice for Bob and Bob for Alice.
```

Algorithm 8 EnhancedExpandCluster(SetOfPointsOfAlice, SetOfPointsOfBob, PointOfAlice, PointOfBob, ClusterId, Eps, MinPts) : Boolean

```
1: Party Alice DOES:
2: IF PointOfAlice is not a core point THEN//Use Updated Protocol.
3:   SetOfPointsofAlice.changeClusterId(PointOfAlice,NOISE);
4:   RETURN False;
5: ELSE
6:   SetOfPointsOfAlice.changeClusterIds(seeds$_A$,ClusterId);
7:   seeds$_A$.delete(PointOfAlice);
8:   WHILE seeds$_A$ $\neq$ Empty DO
9:     currentP:=seeds$_A$.first();
10:    IF currentP is a core point THEN //Use Updated Protocol.
11:      FOR i FROM 1 TO result$_A$.size DO
12:        result$_A$P:=result$_A$.get(i);
13:        IF result$_A$P.ClusterId IN {UNCLASSIFIED,NOISE} THEN
14:          IF result$_A$P.ClusterId=UNCLASSIFIED THEN
15:            seeds$_A$.append(result$_A$P);
16:          END IF;
17:          SetOfPointsOfAlice.changeClusterId(result$_A$P,ClusterId);
18:        END IF; //UNCLASSIFIED or NOISE
19:      END FOR;
20:    END IF; //result.size$\geq$MinPts
21:    seeds$_A$.delete(currentP);
22:  END WHILE; //seeds$_A$ $\neq$ Empty
23:  RETURN True;
24: END IF;
25: END; //ExpandCluster
26: Party B DOES: repeats step 1 to 29 by replacing Alice for Bob and Bob for Alice.
```

5.1 Communication Complexity and Privacy

For each record $d_i (1 \leq i \leq n)$, it has m components. Assume that each component is represented by c_1 bits. Communication is involved when the two parties engage in the Multiplication Protocol to finish Alice and Bob's secret sharing $Dist\{A, B_i\}$. The communication complexity of each Multiplication Protocol is $O(c_1)$. Assume that each number in Protocol YMPP is represented by c_2 bits. In order to find the k^{th} smallest distance and compare whether the distance is greater than Eps, we need $O(l(n-l)c_2n')$ bits to execute, where $n' = |u|$ and l is the number of records owned by one party. Hence, the entire protocol requires $O(c_1ml(n-l) + c_2n'l(n-l))$ to execute.

Theorem 11. *Algorithm 7 privately computes the privacy preserving distributed DBSCAN clustering over horizontally partitioned data assuming semi-honest revealing whether the number of other party's points in his/her point's neighborhood is greater than $Minpts$ minus $||$his/her points in this point's neighborhood$||$.*

Proof. Step 2 and Step 10 are the only two steps of Algorithm 8 requiring communication. As Protocol YMPP and Multiplication Protocol have been proved private, by applying the composition theorem (Theorem 6), we can conclude that the updated privacy preserving distributed DBSCAN clustering over horizontally partitioned data is private. $\square$

6 Conclusion and Future Work

In this paper, we provide efficient privacy preserving algorithms for DBSCAN clustering over the setting of horizontally, vertically and arbitrarily partitioned data, respectively. In the future, it will be interesting to see whether we can extend our method to other privacy preserving distributed data mining algorithms.

7 Acknowledgments

This research has been partially supported by the AFOSR under grant FA9550-12-1-0240 and NSF of China under project 11271351.

References

[1] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining. In *EDBT*, pages 183–199, 2004.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD Conference*, pages 439–450, 2000.

[3] A. Amirbekyan and V. Estivill-Castro. Privacy preserving *dbscan* for vertically partitioned data. In *ISI*, pages 141–153, 2006.

[4] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *SIGMOD Conference*, pages 49–60, 1999.

[5] P. Bunn and R. Ostrovsky. Secure two-party k-means clustering. In *ACM Conference on Computer and Communications Security*, pages 486–497, 2007.

[6] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996.

[7] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *KDD*, pages 505–510, 2003.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[9] A. V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, pages 217–228, 2002.

[10] O. Goldreich. *Foundations of cryptography: Basic applications*, volume 2. Cambridge Univ Pr, 2004.

[11] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, pages 58–65, 1998.

[12] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *KDD*, pages 593–599, 2005.

[13] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16(9):1026–1037, 2004.

[14] K. A. Kumar and C. P. Rangan. Privacy preserving dbscan algorithm for clustering. In *ADMA*, pages 57–68, 2007.

[15] K.-P. Lin and M.-S. Chen. Privacy-preserving outsourcing support vector machines with random transformation. In *KDD*, pages 363–372, 2010.

[16] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.

[17] J. Liu, J. Luo, J. Z. Huang, and L. Xiong. Privacy preserving distributed dbscan clustering. In *EDBT/ICDT Workshops*, pages 177–185, 2012.

[18] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, pages 144–155, 1994.

[19] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *FOCS*, pages 165–176, 2006.

[20] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[21] H. Thomas, E. Charles, and L. Ronald. *Introduce to Algorithms.* MIT press, 2001.

[22] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644, 2002.

[23] J. Vaidya and C. Clifton. Privacy-preserving *k*-means clustering over vertically partitioned data. In *KDD*, pages 206–215, 2003.

[24] W. Xu, L. Huang, Y. Luo, Y. Yao, and W. Jing. Protocols for privacy-preserving dbscan clustering. *Int. J. Secur., Appl*, 1(1):45–56, 2007.

[25] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

[26] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.