Privacy Preserving Distributed DBSCAN Clustering*

Jinfei Liu
University of Science and Technology of China
Shenzhen Institutes of Advanced Technology
jf.liu@siat.ac.cn

Jun Luo
Shenzhen Institutes of Advanced Technology
Chinese Academy of Sciences, China
jun.luo@siat.ac.cn

Joshua Zhexue Huang
Shenzhen Institutes of Advanced Technology
zx.huang@siat.ac.cn

Li Xiong
Emory University
Atlanta, USA
lxiong@mathcs.emory.edu

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 analysis of the performance and proof of privacy of our solution.

Keywords

Privacy Preserving, DBSCAN, Secure Multi-Party Computation, Distributed Data

1. INTRODUCTION

Consider a scenario in which each hospital has its own database of medical records about patients medical records. If the data from different hospitals can be shared, we can extract more meaningful results than just considering partial data independently. However, since medical records are subject to privacy and confidentially constraints, how to extract the knowledge from the whole 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 (Privacy Preserving Data Publishing (PPDP)) et al. [1] and Lindell (Secure Multiparty Computation (SMC)) et al. [10], allows different parties to cooperate in the extraction of knowledge without any of the cooperating parties having to reveal their individual data items to each other or any other parties. 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 PPDM algorithm can be easily extended to the multi-party PPDM algorithm.

Techniques from secure multi-party computation [6] form one approach to privacy preserving data mining. Yao's general protocol for secure circuit evaluation [16] can be used, in theory, to solve any two-party privacy preserving distributed data mining problem. However, since data mining usually involves millions or billions of data items, the communication cost of this protocol renders it impractical for these purposes. This has led to the search for problem specific protocols that are efficient in terms of communication complexity. 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. A complementary approach to privacy preserving data mining uses randomization techniques [4]. Although such solutions tend to be more efficient than cryptographic solutions, they are generally less private or less accurate.

K-means clustering is a simple technique to group items to K clusters. There are many privacy preserving K-means algorithms [8] [14]. DBSCAN [5] is also a popular 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 [12] and CLARANS [11] algorithm as well as hierarchical methods. Similar clustering algorithms based on density are OPTICS [2], DENCLUE [7]. However, there is little literature considering the problem of privacy preserving distributed distributed density-based clustering. So far [9] is the only work we are aware of but it did not provide adequate privacy as

*This research has been partially funded by the International Science & Technology Cooperation Program of China (2010DFA92720) and Shenzhen Fundamental Research Project (grant no. JC201005270342A).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PAIS 2012, March 30, 2012, Berlin, Germany.
Copyright 2012 ACM 978-1-4503-1143-4/12/03 ...$10.00.

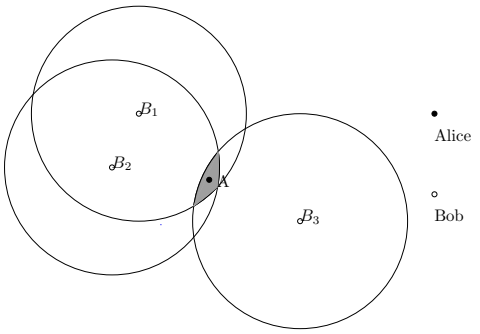

Figure 1: Bob could decide that A is in the small gray region.

we discuss below.

In [9], Kumar et al. proposed protocols for the privacy preserving distributed DBSCAN clustering. However, the proofs of privacy of their protocols do not strictly follow the definition of privacy in the semi-honest model [6]. More importantly, it poses significant privacy risks of identifying individual records from other party, a simple example shows that their algorithms do not guarantee the privacy. Considering a scenario like this, Bob has three records B_1, B_2, B_3 and knows one of Alice's record A is in their neighborhood. So Bob could know A is in the intersection of three records' neighborhood (the gray area in Figure 1). It could happen that the intersection area is so small that Bob could decide the location of the point A that is not far away from its true location. In this paper, we propose algorithms such 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 decide whether there is a record owned by Alice in the small gray region.

In this paper, we present privacy preserving algorithms for DBSCAN clustering for the horizontally, vertically and arbitrarily partitioned data distributed between two parties. We demonstrate that our protocols are efficient in terms of communication, and prove the privacy properties of the protocol.

The main contributions of our paper are:

1. We design a Multiplication Protocol (Subsection 3.1) based on Pailler's Additive Homomorphic crtptosystem.
2. We present privacy preserving distributed DBSCAN clustering protocol utilizing the above multiplication protocol over horizontally (Subsection 3.2), vertically (Subsection 3.3), and arbitrarily (Subsection 3.4) partitioned data that provide adequate privacy.

The rest of the paper is organized as follows. Section 2 introduces some background knowledge that will be used in this paper. The algorithms for computing privacy preserving distributed DBSCAN clustering are given in Section 3. Section 4 concludes the paper with directions for future work.

2. 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.

2.1 DBSCAN Algorithm

We briefly review the DBSCAN (Density Based Spatial Clustering of Applications with Noise) algorithm. Details are described in [5]. 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 [5] that will be used in the next Section.

Definition 1. **(cluster)** Let D be a database of points. A *cluster* C wrt. 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 wrt. Eps and MinPts, then $q \in C$. (Maximality)
2. $\forall$ point p,q $\in C$: p is density-connected to q wrt. Eps and MinPts. (Connectivity)

Definition 2. **(noise)** Let $C_1, ..., C_k$ be the clusters of the database D wrt. parameters Eps_i and $MinPts_i, i = 1, ..., k$. Then we define the *noise* as the set of points in the database D not belonging to any cluster C_i , i.e. noise $= p \in D \mid \forall i : p \notin C_i$.

2.2 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 different records with full attributes (see Figure 2).
- **Vertically Partitioned Data**: each party owns all records with partial attributes (see Figure 3).
- **Arbitrarily Partitioned Data** [8]: mixture of horizontally and vertically partitioned data (see Figure 4).

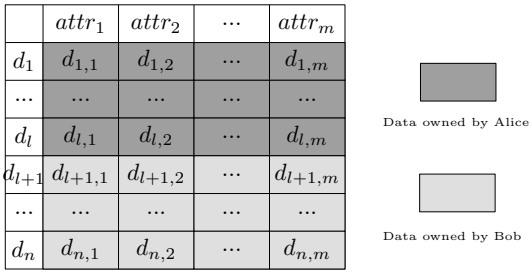

Figure 2: Horizontally partitioned data.

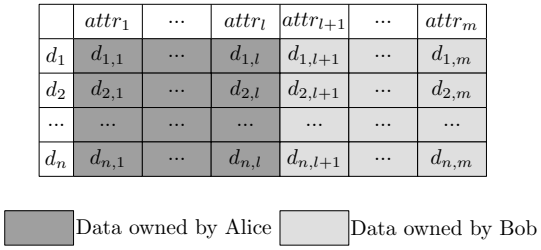

Figure 3: Vertically partitioned data.

2.3 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 data record, that some records do not belong to any clusters) for each data record that she owns completely, and Bob should learn the cluster number (or the NOISE data record) for each data object 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.

2.4 Two-party Computation

A two-party protocol problem [6] 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 $(x, y) \mapsto (f_1(x, y), f_2(x, y))$.

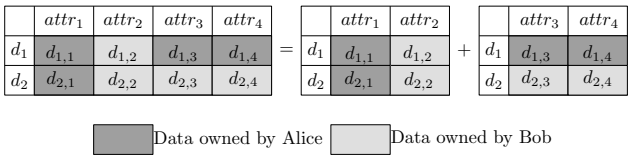

Figure 4: Arbitrarily partitioned data =vertically partitioned data + horizontally partitioned data.

2.5 Semi-honest

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 partyąŕs database alone. Although this attack cannot be prevented, we would like to prevent a malicious party from executing any other attack. However, for this initial work we assume that the adversary is *semi-honest* [6] (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 far 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. Semi-honest adversarial behavior also models a scenario in which both parties that participate in the protocol are honest. However, following the protocol execution, an adversary may obtain a transcript of the protocol execution by breaking into one of the parties' machines.

2.6 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) partyąŕs 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 partyąŕs view of the execution.

Definition 3. **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

$$\{(S_1(x, f_1(x,y)), f_2(x,y))\}_{x,y\in\{0,1\}^*} \stackrel{c}{\equiv}$$

$$\{(view_1^{\prod}(x,y), output_2^{\prod}(x,y))\}_{x,y\in\{0,1\}^*}, \quad (1)$$

$$\{(f_1(x,y), S_2(y, f_2(x,y)))\}_{x,y\in\{0,1\}^*} \stackrel{c}{\equiv}$$

$$\{(output_1^{\prod}(x,y), view_2^{\prod}(x,y))\}_{x,y\in\{0,1\}^*}, \quad (2)$$

where $\stackrel{c}{\equiv}$ 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 [6] for a full discussion.

THEOREM 1. *(Composition Theorem for the semi-honest model, two parties) [3]: 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.*

2.7 Paillier's Additive Homomorphic Properties

For some notations will be used in Subsection 3.1, we briefly describe the Paillier's additive homomorphic cryptosystem [13] 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:

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

2.8 Yao's Millionaires' Problem Protocol (YMPP)

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 [15] decide 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 can not 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 $z_u = y_u (\text{mod } p)$ 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.

3. PRIVACY PRESERVING DISTRIBUTED DBSCAN CLUSTERING

In this section, we first introduce our multiplication protocol, then extend the single party DBSCAN algorithm of [5] to privacy preserving distributed DBSCAN clustering on horizontally vertically and arbitrary partitioned data, respectively. We also provide analysis of the communication complexity and proofs of privacy of our protocols.

3.1 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 problem, but the known scalar products protocols are not

fit for the multiplication problem. In our method, Paillier's additive homomorphic encryption schemes [13] 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 $u = D_A(u')$.

3.1.1 Correctness proof

Since,

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

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 1. *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. □

3.2 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 or not, 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 $d_{x,1}, d_{x,2}, ..., d_{x,m}$. In order to evaluate whether $dist(d_x, d_y) < Eps$, we can evaluate whether $dist(d_x, d_y)^2 < Eps^2$.

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

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

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

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

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 number $r_1, r_2, ..., r_m$ 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 $dist(d_x, d_y)^2 \leq Eps^2$ or not.

LEMMA 2. *Protocol HDP is secure.*

PROOF. Besides the protocol YMPP, the only communication is Alice sends $r_1, r_2, ..., r_m$ to Bob. On 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 time linear, which meets the requirement for a polynomial time simulation. □

3.2.1 DBSCAN Algorithm for Horizontally Partitioned Data

The details of the DBSCAN algorithm for horizontally partitioned data are given in Algorithm 3. SetOfPointsOfAlice (SetOfPointsOfBob) is either the complete database of Alice's (Bob's) or a discovered cluster from a previous run of Alice (Bob). Eps and MinPts are the global density parameters. A call of SetOfPointsOfAlice.regionQuery(PointOfAlice, Eps) (SetOfPointsOfBob.regionQuery(PointOfBob,Eps)) returns the Eps-Neighborhood of PointOfAlice (PointOfBob) in SetOfPointsOfAlice (SetOfPointsOfBob) as a list of points. The function SetOfPointsOfAlice.get(i) (SetOfPointsOfBob.

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, 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 steps 1 to 12 by replacing Alice by Bob and Bob by Alice.
```

get(i)) returns the i-th element of SetOfPointsOfAlice (SetOfPointsOfBob). The most important function used by DBSCAN is ExpandCluster which is presented in Algorithm 4. SetOfPointsOfBobPermutation (SetOfPointsOfAlicePermutation) is SetOfPointsOfBob (SetOfPointsOfAlice). But in each iteration, the points' order is permutated randomly. Therefore, Alice (Bob) only knows there is a point of Bob (Alice) (but does not know which point of Bob (Alice)) in neighborhood of one of her (his) points. In this way, the situation in Figure 1 can be avoided. What should be noted is that in Step 3 and 13, only Alice (Bob) knows whether the point in SetOfPointsOfBobPermutation (SetOfPointsOfAlicePermutation) is seed or not. Otherwise, Bob (Alice) could decide Alice (Bob) in a small intersection region as we argued in Figure 1.

3.2.2 Communication Complexity and Privacy

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

THEOREM 2. *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 steps of Algorithm 4 requiring communication. As Lemma 2 has proved, protocol HDP is private. Applying the composition theorem (Theorem 1), we can conclude that the privacy preserving distributed DBSCAN clustering over horizontally partitioned data is private. □

While not truly zero-knowledge for the number of points from the other party in the neighborhood of this point is revealed, it reduces the confidence of the leaked knowledge of the exact points. In practical terms, revealing the number is likely to be of little concern.

3.3 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 or not. 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 $dist(d_x, d_y) \leq Eps$ or not.

3.3.1 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 complete 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.

3.3.2 Communication Complexity and Privacy

For each record d_i, it has m components. Communication is involved when the two parties engage in the protocol YMPP to judge the core point. Let us 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.

THEOREM 3. *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. Applying the composition theorem (Theorem 1), 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. □

While not truly zero-knowledge for the number of points in the neighborhood of this point is revealed, it reduces the confidence of the leaked knowledge of the exact points. In practical terms, revealing the number is likely to be of little concern.

3.4 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 disjoint subsets $Alice_{d_i}$ and Bob_{d_i} such that Alice knows $Alice_{d_i}$ and Bob knows Bob_{d_i}. Although extremely patchworked data is unlikely 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 2.1, 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

$$\overbrace{(A_{2,1} - A_{1,1})^2}^{Vertically\ partitioned} + (B_{2,2} - B_{1,2})^2 +$$

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

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 divide to data owned by Alice ($Alice_h$) and data owned by Bob (Bob_h). Then Alice and Bob use protocol HDP to let Bob get $(B_{2,3} - A_{1,3})^2 + (B_{2,4} - A_{1,4})^2 = Bob_h$. Together 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.

Due to limited space, we do not present the detailed algorithm for arbitrarily partitioned data. However, it should be easy to figure out the algorithm for the arbitrarily partitioned data based on the algorithms for horizontally and vertically partitioned data.

4. 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. However, in order to decide whether one point is core point, our method reveals the number of points from the other party in the neighborhood of this point. In the future, we want to develop the algorithms of deciding whether one point is core point without knowing the number of points from the other party in the neighborhood of this point. Also, it will be interesting to see whether we can extend our method to other privacy preserving distributed data mining algorithms.

5. REFERENCES

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

[2] 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.

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

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

[5] 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.

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

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

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

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

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

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

[12] 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.

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

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

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

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

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 ≠ 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≥MinPts THEN
15:         FOR i FROM 1 TO result_A.size DO
16:            result_AP:=result_A.get(i);
17:            IF result_AP.ClusterId IN {UNCLASSIFIED,NOISE} THEN
18:               IF result_AP.ClusterId=UNCLASSIFIED THEN
19:                  seeds_A.append(result_AP);
20:               END IF;
21:               SetOfPointsOfAlice.changeClusterId(result_AP,ClusterId);
22:            END IF; //UNCLASSIFIED or NOISE
23:         END FOR;
24:      END IF; //result.size≥MinPts
25:      seeds_A.delete(currentP);
26:   END WHILE; //seeds_A ≠ Empty
27:   RETURN True;
28: END IF;
29: END; //ExpandCluster
30: Party B DOES: repeats steps 1 to 29 by replacing Alice by Bob and Bob by Alice.
```

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

```
1: seeds:=SetOfPoint.regionQuery(Point,Eps); //Alice and Bob cooperatively use protocol VDP.
2: IF seeds.size<MinPts THEN //no core points
3:    SetOfPoints.changeClusterId(Point,NOISE);
4:    RETURN False;
5: ELSE
6:    SetOfPoints.changeClusterIds(seeds,ClusterId);
7:    seeds.delete(Point);
8:    WHILE seeds ≠ Empty DO
9:       currentP:=seeds.first();
10:      result:=SetOfPoints.regionQuery(currentP,Eps); //Alice and Bob cooperatively use protocol VDP.
11:      IF result.size≥MinPts THEN
12:         FOR i FROM 1 TO result.size DO
13:            resultP:=result.get(i);
14:            IF resultP.ClusterId IN {UNCLASSIFIED, NOISE} THEN
15:               IF resultP.ClusterId=UNCLASSIFIED THEN
16:                  seeds.append(resultP);
17:               END IF;
18:               SetOfPoints.changeClusterId(resultP, ClusterId);
19:            END IF; //UNCLASSIFIED or NOISE
20:         END FOR;
21:      END IF; //result.size≥MinPts
22:      seeds.delete(currentP);
23:   END WHILE; //seeds ≠ Empty
24:   RETURN True;
25: END IF;
26: END; //ExpandCluster
```