社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－CSEC－34 (48)
2006／7／21

プライバシー保護した分散的ドキュメントクラスタリング

蘇 春華† 周 建英†† 鮑 豊†† 櫻井 幸一†††

† Department of Computer Science and Communication Engineering, Kyushu University.
†† Institute for Infocomm Research (I^2R), Singapore
††† Department of Computer Science and Communication Engineering, Kyushu University.
E-mail: †su@itslab.csce.kyushu-u.ac.jp, ††{jyzhou,baofeng}@i2r.a-star.edu.sg,
†††sakurai@csce.kyushu-u.ac.jp

Distributed Privacy-preserving Document Clustering

Chunhua SU†, Jianying ZHOU††, Feng BAO††, and Kouichi SAKURAI†††

† 九州大学大学院システム情報科学府
†† シンガポール国立情報技術研究院
††† 九州大学大学院システム情報科学研究院
E-mail: †su@itslab.csce.kyushu-u.ac.jp, ††{jyzhou,baofeng}@i2r.a-star.edu.sg,
†††sakurai@csce.kyushu-u.ac.jp

Abstract Many government organizations and companies want to share their documents in a similar theme to get the joint benefits. Textual document clustering is a powerful data mining technique to analyze the large amount of documents and structure large sets of text or hypertext documents. While doing the document clustering in the distributed environment, it may involve the users' privacy of their own document. In this paper, we propose a framework to do the privacy-preserving text mining among the users under the distributed environment: multiple parties, each having their private documents, want to collaboratively execute agglomerative document clustering without disclosing their private contents to any other parties.

Key words Documents Clustering, Privacy-preserving Data Mining, Secure Multi-party Computation

1. Introduction

In today's information age, data collection is ubiquitous, and every transaction is recorded somewhere. Data mining is becoming increasingly common in many areas such as banking, insurance, medicine, scientific research, and even in homeland security area to detect the terrorism. The resulting data sets can consist of terabytes or even petabytes of data. In our paper, we focus on the textual document clustering problem which is a kind of textual data mining techniques. The difference between regular data mining and text mining is that in text mining the patterns are extracted from natural language text rather than from structured databases of facts. Document clustering is the act of collecting similar documents into bins, where similarity is computed by some functions on the documents. Text is considered to be composed of two fundamental units, namely the documents and the term. A document can be a traditional document such as a book or journal paper and an term can be a word, word-pair, or phrase with a document. There are many different ways we can show how a set of documents are related to one another. One solution we study is to show the user how the terms in the query are related to the words in the document. Standard partitioned or agglomerative clustering methods efficiently compute results to this end. The main idea is to find which documents have many words in common, and place the documents with the most words in common into the same groups. Agglomerative clustering is to build the hierarchy bottom-up by iteratively computing the similarity between all pairs of clusters and then merging the most similar pair.

1.1 Privacy Preserving Data Mining

The rapid development of Internet provides us with tremendous opportunities for cooperative computations. Many organizations and companies want to do joint work to get mutual benefit of their individual data. At the same time, with the powerful data mining techniques, it also can bring some privacy violation problems. In the real world, many organizations and individuals may have concern on their own privacy, and may be reluctant to share their own data for data mining unless their privacy will not be violated or misused by other parties. The privacy-preserving data mining research aims at solving such problems. In 2000, R. Agrawal et al. proposed a reconstruction procedure which is possible to accurately estimate the distribution of original data values from the perturbed data [1]. After this paper, many researchers became interested in this topic. The first paper to take the classic cryptographic approach to privacy preserving data mining was presented by Lindell and Pinkas [2]. They present an efficient protocol for the problem of distributed decision tree learning.

1.2 Relative Works

In the past few years, a lot of different document clustering algorithms have been proposed in the literature, including Scatter/Gather [3], SuffixTree Clustering [4]. Bisecting k-means is proposed by M. Steinbach et al [5] based on an analysis of the specifics of the clustering algorithms and the nature of document data. The above methods of text clustering algorithms do not really address the special challenges of text clustering. They do not provide an understandable description of the clusters. This has motivated the development of new special text clustering methods which are not based on the vector space model and some frequent-term based methods have been proposed such as Florian Beil et al [6].

Some privacy-preserving $k-$means clustering schemes are also proposed. The work by Vaidya and Clifton [7] introduces a solution based on secure multi-part computation techniques. Specifically, the authors proposed a method for $k-$means clustering when different sites contain different attributes for a common set of entities. In the solution, each site learns the cluster of each entity, but learns nothing about the attributes at other sites. This work ensures reasonable privacy when the communication cost is low. Geetha Jagannathan and Rebecca Wright proposed the privacy-preserving k-means clustering scheme [8] and introduced the concept of arbitrarily partitioned data which is generalization of both horizontally and vertically partitioned data and provides a privacy-preserving protocol for k-means clustering in the setting of arbitrarily partitioned data.

1.3 Our Contributions

Our work is based the relative works and make a extension to preserve the privacy document clustering over the distributed network environment. There are important differences between previous works and our work. In the traditional document clustering techniques, there is no consideration toward the privacy. Also, the existing privacy-preserving data mining research, there is no proposal to deal with the unstructured textual data. Our work is going to solve the two problems.

- **Extension to Distributed Documents Clustering.** Our work is based on the relative work. We make an extension to preserve the privacy document clustering over the distributed network environment. And we assume that there are n parties distributed in the network, holding their private documents dataset respectively, having their documents clustered. And finally, the server can output the cluster description, and the parties can check to which cluster their documents belong to.
- **Privacy-preserving Framework.** We propose a framework to preserve the each party's privacy in distributed document clustering. In our proposal, every party do the documents clustering computation via the interactive protocol and output the clusters for each document without revealing the document's content. During the execution of the protocol, all the privacy of the parties is preserved.
- **Dealing with Unstructured Data.** The existing privacy-preserving data mining techniques [1] [7] [8] focus databases which are clearly defined and structured, it is easy to run queries and formulas which extract meaningful information. Different from the existing scheme, we deal with documents data which are unstructured. We use privacy keyword searching techniques to extract frequent terms from large unstructured document data sets, discover relationships and summarize the information.

The paper is organized as follows: We give out problem setting and security definition in next section and then present our proposed privacy-preserving protocol in Section 3 and the implementation of the privacy-preserving protocol is given out in section 4. The performance and security is discussed in Section 5. We give our conclusions in Section 6.

2. Problem Definitions and Cryptographic Primitives

2.1 Problem Setting

Our purpose in this paper is to automatically group related documents based on their content with respect to the document privacy. We consider the scenario where multiple n parties in a distributed network, each having a private document database denoted by $D_1, D_2, ..., D_n$ respectively.

They want to collaboratively build a agglomerative documents clustering on the concatenation of their databases, and get the common benefit for doing clustering analysis in the joint databases $D = D_1 \cup D_2 \cup ... \cup D_n$. Further more, they also want discovered clusters to provide an understandable description of the clusters. It is very helpful when a party who holds some documents which belong to a discovered cluster with the description "security, privacy, database" want to do a co-research with other parties who hold the similar documents. In the same cluster must have same description, it is very easy to make it. Moreover, the agglomerative clustering can enforce this function by building a hierarchical tree to show the relationship between two similar clusters.

Because they are concerned about their data privacy, neither party is willing to disclose his raw document data set to others. For this reason, they need a private preserving system to execute the joint document clustering analysis. The concern is solely that values associated with an individual entity not be released (e.g., personal or sensitive information), the techniques must focus on protecting such information.

2.2 Computation Model and Its Security Definition

Our protocol construction is based on secure multi-party computation techniques. Secure multi-party computation protocols, which is extensive since it was introduced by Yao [12] and extended by Goldreich, Micali, and Wigderson [13]. It allow a set of n players to securely compute any agreed function on their private inputs and the corrupted players do not learn any information about the other players' inputs. In Secure Multi-party Computation, we always assume that *semi-honest model* exists. A semi-honest party follows the rules of the protocol giving its correct input, but it is very curious and it only tries to deduce information on the inputs of the honest parties by inspecting all the information available to the corrupted parties. This is somewhat realistic in the real world because parties who want to mine data for their mutual benefit will follow the protocol to get correct results. Also, a protocol that is buried in large, complex software can not be easily altered.

In the secure computation setting, there is the real and ideal model paradigm: *Ideal model*: parties send inputs to a trusted party, who computes the function and sends the outputs. *Real model*: parties run a real protocol with no trusted help. We say that a real protocol that is run by the parties (in a world where no trusted party exists) is secure, if no adversary can do more harm in a real execution than in an execution that takes place in the ideal world. Stated differently, for any adversary carrying out a successful attack on a real protocol, there exists an adversary that successfully carries out the same attack in the ideal world.

Client party's privacy:(indistinguishability). For any PPT executing the server's part and for any inputs X, w, w' the views that sees on input X, in the case that the client uses the searchword w and the case that it uses w' are computationally indistinguishable.

Server party's privacy: (comparison with the ideal model). For every PPT machine C substituting the client in the real protocol, there exists a PPT machine C' that plays the client's role in the ideal implementation, such that on any inputs (X, w), the view of C is computationally indistinguishable from the output of C'. (In the semi-honest model $C' = C$). For constructing a secure documents clustering protocol, we apply the following cryptographic primitives.

2.3 Cryptographic Primitives

Homomorphic Encryption: In our protocol, we require a homomorphic encryption scheme satisfying: $E(a) * E(b) = E(a+b)$, where E is a cryptosystem, $*$ and $+$ denote modular multiplication and addition, respectively. It also follows that $E(a)^c = E(a * c)$ for $c \in N$. The Paillier cryptosystem [16] is a proper scheme which has this property and is the cryptosystem of our choice to construct a secure protocol.

Oblivious Transfer (OT): The 1-out-of-N oblivious transfer protocol is used to do the circuit evaluation privately. In the case of semi-honest adversaries, there exist simple and efficient protocols for oblivious transfer. An 1-out-of-N Oblivious Transfer protocol refers to a protocol where at the beginning of the protocol one party, party B has N inputs $x_1, ..., x_N$ and at the end of the protocol the other party, party A learns one of the inputs x_i for some $1 \leqq i \leq N$ of her choice, without learning anything about the other inputs and without allowing B to learn anything about x_i.

Oblivious Polynomial Evaluation (OPE): Oblivious Polynomial Evaluation(OPE) is one of fundamental cryptographic techniques. It involves a sender and a receiver. The sender's input is a polynomial $Q(x)$ of degree k over some field F and the receiver's input is an element $z \in F$. The receiver learns $Q(z)$. It is quite useful to construct some protocols which enable keyword queries while providing privacy for both parties: namely, (1) hiding the queries from the database (client privacy) and (2) preventing the clients from learning anything but the results of the queries (server privacy).

3. Privacy-Preserving Documents Clustering Protocol

Our protocol is based a novel approach proposed by Florian Beil et al [6], and modify it into a multi-party divide-and-merge frequent term-based clustering protocol. A frequent item-based approach of clustering is promising because it provides a natural way of reducing the large dimensionality

of the document vector space. There is two phases in our protocol, the first one is divide phase and the other is merge phase. It works by breaking down the distributed documents clustering problem into two sub-problems and the solutions to the sub-problems are then combined to give a solution to the original problem. In our protocol, divide-and-merge algorithms are implemented in a non-recursive way and the computation is interactive, so every party plays roles of both client and server, called client party and server party, respectively.

3.1 Local Pre-computation for Text Data

All methods of document clustering require several steps of as preprocessing of the data. First, any non-textual information and punctuation is removed from the documents as well as stopwords such as "I", "am", "and". Note that a term may either be a single word or consist of several words. After that, every party should do the pre-computation on their own text data in every individual document. every party form his database which contains N frequent terms as $X = (x_i, p_i)$ with $1 \leq i \leq N$. In every document which are held by the participant party, x_i is a keyword and p_i is frequency that the keyword occurs in one document. Every party numbers all his own document for 1 to m, where m is the number of total documents which are held by a party. For example, the i_{th} party can number his document like $Doc_i^1, ..., Doc_i^m$.

3.2 Divide Phase of Document Clustering

In this phase, every party execute the privacy-preserving frequent term query scheme interactively and compute the It uses frequent item (term) sets for text clustering. Such frequent sets can be efficiently discovered using algorithms for association rule mining. To cluster based on frequent term sets, every party have to pre-determine the threshold minimum support *minsupp* and measure the mutual overlap of frequent sets with respect to the sets of supporting documents.

We use frequent term-based clustering determines a flat clustering an unstructured set of clusters covering the whole databases of all parties. Let $D_i = \{Doc_i^1, ..., Doc_i^m\}$ be a database of m text documents held by the i_{th} party. Each document Doc_i^j is represented by the set of terms occurring in D_i. Let *minsupp* be a real number, $0 \leq minsupp \leq 1$. Let $w_i = \{w_i^1, ..., w_i^k\}$ be the set of all frequent term sets in D_i with respect to *minsupp*, the set of all term sets contained in at least *minsupp* of the D_i documents, let $cov(w_i)$ denote the *cover* of w_i, the set of all documents containing all terms of w_i, more precisely, $cov(w_i) = \{Doc_i^j \in D_i \mid w_i \subseteq Doc_i^j\}$ A frequent term set of cardinality k is called a frequent k-term set. The cover of each element w_i is a cluster candidate. A cluster can be any subset of the set of all subsets of all parties' database $D = D_1 \cup D_2 \cup ... \cup D_n$ such that each document of D is contained in at least one of the sets (clusters). We define a clustering description CD as a subset of $w_1, ..., w_n$.

We determine a cluster with a minimum overlap of the cluster candidates. Let f_i^j denote the number of all frequent term sets supported by document Doc_i^j in remaining frequent term sets. The overlap of a cluster of C_i^j with the other clusters is the smaller, the smaller the f_i^j values of its documents are. We define the entropy overlap $EO(C_i^j)$ as the distribution of the documents of cluster C_i^j over all the remaining cluster candidates, $EO(C_i^j) = \sum_{Doc^j \in C_i^j} -\frac{1}{f_i^j} \cdot ln(\frac{1}{f_i^j})$.

In the divide phase computation, every party starts with an empty set, it continues selecting one more element (one cluster description) from the set of remaining frequent term sets until the database of selected documents is contained in the cover of the set of all chosen frequent term sets (the clustering). Setting the database formed by selected documents to D_{sel}. In each step, party selects the remaining frequent term set with a cover having the minimum overlap with the other cluster candidates locally. Note that the documents covered by the selected frequent term set are removed from the database D_{sel} and in the next iteration, the overlap for all remaining cluster candidates is recalculated with respect to the reduced database.

Clustering Algorithm of Divide Phase

Input: Every party i's frequent keyword w_i of databases D_i, where $1 \leq i \leq n$ and the threshold support *minsup*

Output: The clusters and their descriptions based on frequent terms in w_i.

(1) Do the private keyword queries with local frequent term set w_i and select all parties' documents whose keyword frequency $p_i \geq minsup$. the *SelectedTermSets* := {}, $k := |D_{sel}|$

(2) **While** $cov(SelectedTermSets) \neq k$ do

(3) **for each** set in *RemainingTermSets* **do**

(4) Calculate overlap entropy for set;

(5) *BestCandidate*:= element of *RemainingTermSets* with minimum entropy overlap;

(6) *SelectedTermSets* := *SelectedTermSets* ∪ {*BestCandidate*}

(7) *RemainingTermSets* := *RemainingTermSets*− {*BestCandidate*}

(8) Remove all documents in $cov(BestCandidate)$ from D_{sel} and from the coverage of all of the *RemainingTermSets*;

(9) Return *SelectedTermSets* as cluster descriptions and the cover of *SelectedTermSets* as clusters.

Every party executes the clustering algorithm of divide phase, and get the clusters. Because from the step 2, all the party can compute locally, so the problem is how to execute the step 1 for select the documents by keyword queries privately.

3.3 Merge Phase of Document Clustering

In the divide phase, every party get the local some non-overlap clusters based on the frequent terms of their own documents. The agglomeration algorithm creates hierarchical clusters. At each level in the hierarchy, clusters are formed from the union of two clusters at the next level down. In the merge step, every party starts with his own cluster and gradually merge clusters until all clusters have been gathered together in one big cluster.

There is two step for clusters merging computation: **1. Cluster Inclusion Merging Step:** A smaller cluster which is included by the larger one will be merged into the larger one. **2. Agglomerative step:** The two similar cluster will be merge as new cluster according to the similarity computation. Note that the description of clusters will be merged into a intersection at the same time.

Algorithm of Merge Phase

(1) Initially, every party uses his clusters to do a private inclusion test with other party's clusters. Merge the two cluster if one cluster is included in the other cluster. Stop until every included subset of clusters is merged.

(2) Among all remaining clusters, pick the two clusters to do the private similarity computation.

(3) Replace these two clusters with a new cluster, formed by merging the two original ones with the most similarity.

(4) Repeat the above step 2 and step 3 until there is only one remaining cluster which covers all parties' databases.

Note that in the algorithm, we can preserve the privacy by only outputting the cluster description. The merged cluster description CD is a intersection of two original cluster descriptions, not the union of the two. Because the coverage $cov(CD)$ can cover all the documents whose frequent terms are included in CD, with the output of the protocol, the clients match their documents with the cluster descriptions CD and assign them to the proper cluster with a subset relationship between each cluster and its predecessors in the hierarchy privately. This produces a binary tree or dendrogram, of which final agglomerative cluster is the root and each cluster is a leaf, the height of the bars indicates how close the clusters and their descriptions are.

4. Implementing the Privacy-preserving Protocol

Here in this section, we show how to implement the privacy-preserving protocol using the cryptographic techniques which we have mentioned in section 3.

4.1 Private Document Selecting

When the a party gets the local frequent keywords, he have to construct some queries to select the documents which contain the same frequent term with respect to the privacy. In this section, we construct a protocol using oblivious polynomial evaluation (OPE) based on [15]'s scheme and apply the zero knowledge proof to avoid the malicious inputs of the client party. The basic idea of the construction is to encode the database $D's$ entries in $\{X = (x_1, num_1), ..., (x_n, num_n)\}$ as values of a polynomial, i.e., to define a polynomial Q such that $Q(x_i) = (num_i)$, where x_i denotes the keyword and num_i denotes the document number for clustering. Note that this design is different than previous applications of OPE, where a polynomial (of degree k) was used only as a source for $(k+1)$-wise independent values.

Document Selecting with Private Keyword Search

Input:1. Client party input his local frequent keyword w; Server party input $\{x_i, num_i\}_{i\in[n]}$, all x_i's are distinct

Output: Client party get document number $number_i$ if $w = x_i$, nothing otherwise; Server party: nothing

(1) The server party defines L bins and maps the n items into the L bins using a random, publicly-known hash function H with a range of size L. H is applied to the database's frequent keywords, i.e., x_i ($p_i \geqq minsupp$) is mapped to bin $H(x_i)$. Let m be a bound such that, with high probability, at most m items are mapped to any single bin.

(2) For every bin j, the server party defines two polynomials P_j and Q_j of degree $(m-1)$. The polynomials are defined such that for every pair (x_i, num_i) mapped to bin j, it holds that $P_j(x_i) = 0$ and $Q_j(x_i) = (num_i|0^l)$, where l is a statistical security parameter.

(3) For each bin j, the server party picks a new random value r_j and defines the polynomial $Z_j(w) = r_j \cdot P_j(w) + Q_j(w)$.

(4) The two parties run an OPE protocol in which the sever party evaluates all L polynomials at the searchword w.

(5) The client party learns the result of $Z_{H(w)}(w)$, i.e.,

of the polynomial associated with the bin $H(w)$. If this value if of the form $number_i|0^l$ the client party outputs get the $number_i$,

---

Our construction uses an OPE method based on homomorphic encryption such as Paillier's system [16] in the following way.

- The sever party's input is a polynomial of degree m, where the polynomial $P(x) = \sum_{i=0}^{m} a_i x^i$, The client party's input is a keyword value w.
- The client party sends to the server party homomorphic encryptions of the powers of w up to the m_{th} power, i.e., $Enc(w), Enc(w^2), ..., Enc(w^m)$.
- The server party uses the homomorphic properties to compute the following:

$\prod_{i=0}^{m} Enc(a_i w^i) = \sum_{i=0}^{m} Enc(a_i w^i) = Enc(P(w))$. The client party sends this result back to the server party.

For preventing client from cheating in OPE, the server party can ask client party do zero knowledge proof of $Enc(w_i)$ before the construction in terms of a single database bin. We can use the Damgård and Jurik's scheme proposed in [17] to prove that the input is the encryption of w_i without disclosing the keyword w.

4.2 Private Testing Inclusion of Cluster

Every cluster can be represented as a binary string according to documents' order from party 1 to party n, such as $Doc_1^1, Doc_1^2 ..., Doc_n^m$. Each bit of the string corresponds to a document, there is 1 in the entry i if and only if the cluster contains the party 1's document Doc_1^i, if the document doesn't exist, there is 0. Client party i has a set $C_i \subseteqq D$, Server party j has a set $C_j \subseteqq D$, and the two party must establish whether $C_i \subseteqq C_j$ or not without neither of the parties obtaining any additional information. More precisely, the protocols must satisfy client-privacy and server-privacy. we assume that the client has n words in his database. We use a the idea if $C_i \in C_j$ then $|C_i \cap C_j| = |C_j|$.

Here, we modify the matrix-based private inclusion scheme proposed [18] into binary string-based scheme to construct our private cluster merging protocol. We implement this with the homomorphic cryptosystem which is proved to be secure in the sense of IND-CPA under reasonable complexity assumptions. A public-key cryptosystem is a triple $\phi = (G, E, D)$, where G is the key generation algorithm that returns (sk, pk) consisting of a secret key k and a public key pk, E is the encryption algorithm and D is the decryption algorithm.

Private Cluster Inclusion Testing Protocol

PRIVATE INPUT: Sever party j: cluster C_i and client party i:set C_j.

PRIVATE OUTPUT: Client party knows whether $C_i \subseteqq C_j$, output $CD_i \cap CD_j$.

(1) Client party generates a new key pair $(sk, pk) \longleftarrow G$. Send pk to sever party. For any $i \in [n]$, generate a new nonce $r_i \longleftarrow_r R$. Send $e_i \longleftarrow E_{pk}(C_i; r_i)$ to server party.

(2) Server party draws $s \longleftarrow_r P$, $r \longleftarrow_r R$ uniformly at random. Set $e \longleftarrow (\prod_{t=1}^{l} C_i[t]/C_j[t])^s \cdot E_{pk}(0; r)$, where l is the last l_{th} bit of 1. Send e to server party.

(3) Client party sets $d \longleftarrow D_{sk}(e)$. Accept that $C_i \subseteqq C_j$ iff $d = 0$ and send the result to sever party.

(4) Sever party returns cluster C_j as a merged cluster and outputs the $CD_j = CD_i \cap CD_j$.

After this process, the flat clusters for the agglomerative document clustering are generated and only the cluster descriptions are output. And all the parties can use those clusters descriptions to group their documents. By using zero-knowledge proofs, client party can prove the correctness of (a) pk is a valid public key and that (b) every bit of c_i encrypts either 0 or 1.

4.3 Private Measuring the Similarity

We use Hamming distance to measure that similarity of two clusters. The Hamming distance is the number of positions in two strings of equal length for which the corresponding elements are different. Every cluster can be represented as a binary string as in the private inclusion cluster merging protocol. To compute the Hamming distance privately, we use the Private-Sample-XOR Protocol proposed by J. Feigenbaum [19] as following:

Notions: In this protocol, We let $d_h(a, b)$ denote the Hamming distance between (a, b), for any $x \in \{0,1\}^n$ $r \in [n]$ and $m\{0,1\}^n$, we denote by $x << r$ a cyclic shift of x by r bits to the left, and by $x \bigoplus m$ the string whose i-th it is $x_i \bigoplus m$

Private Approximation of Hamming Distance

(1) Party A generates a random mask $m_A \stackrel{R}{\leftarrow} 0,1$ and a random shift amount $r_A \stackrel{R}{\leftarrow} [n]$. And he computes the n-bit string $a' \stackrel{def}{=} (a << r_A) \bigoplus m_A$

Symmetrically, Party B generates $m_B \stackrel{R}{\leftarrow} 0,1$ and $r_B \stackrel{R}{\leftarrow} [n]$, and computes $b' \stackrel{def}{=} (a << r_B) \bigoplus m_B$

(2) A and B invoke in parallel two $\binom{n}{1}$-OT protocols:

- A retrieves $z_A \stackrel{def}{=} b'_{r_A}$ from B;
- B retrieves $z_B \stackrel{def}{=} a'_{r_B}$ from A;

(3) A sends $z'_A \stackrel{def}{=} z_A \bigoplus m_A$ to B. B sends $z'_B \stackrel{def}{=} z_B \bigoplus m_B$ to A. Both parties locally output $z'_A \bigoplus z'_B$.

After executing the protocol we can get the approximate result of similarity of the two clusters. The smaller the hamming distance, the more similar of two clusters, and the most similar two clusters' cluster description's will be joined into a intersection, i.e, $CD_A \cap CD_B$.

4.4 Performance Evaluation

During the private keyword search, we assume that client party assign the n items to L bins arbitrarily and evenly, ensuring that L items are assigned to every bin; thus, $L = \sqrt{n}$. The server party's message during the OPE consists of $L = O(\sqrt{n})$ homomorphic encryptions, he evaluates L polynomials by performing n homomorphic multiplications, and replies with the $L = \sqrt{n}$ results. This protocol has a communication overhead of $O(\sqrt{n})$, $O(n)$ computation overhead at the client party's side, and $O(\sqrt{n})$ computation overhead at the server party's side. In private inclusion cluster testing protocol, server party does not perform any precomputation, when server party gets client party's query as an encrypted binary string, the communication of this protocol is $len(|d|)$ bits. For computation of similarity of clusters, we use a $\binom{n}{1}$-OT protocol (in the semi-honest model) as a sub-protocol. Then, the protocol for approximating $d_h(a,b)$, and whose round complexity is $OT+1$. Hamming distance function can be privately ϵ-approximated with communication complexity $O(n^{1/2}/\epsilon)$ and three rounds of interaction.

5. Security Analysis

The document selecting protocol preserves the client party's privacy because the server party cannot distinguish between any two of clients' inputs w, w', the protocol also protects the server party's privacy if a polynomial Z with fresh randomness is prepared for every query on every bin, then the result of the client party's query w is random if w is not a root of P, and the malicious input of clients party can be prevented by using the zero knowledge proof of the frequent keyworkd w. In the private cluster inclusion testing protocol, computational client-privacy follows directly from the IND-CPA security. So an adversary can learn nothing about the plaintext corresponding to a given ciphertext, even when the adversary is allowed to obtain the plaintext corresponding to ciphertexts of its choice. As server party sees only ciphertexts of encrypted clusters, his privacy is guaranteed as the second step depends only on whether $C_i \in C_j$ or not, the malicious In similarity measuring computation, when computing the Hamming distance between the inputs a and b, the view of each party in these invocations can be simulated from its input and $d_h(a,b)$. Summarizing, we have it has a simulator S such that $S(d_h(a,b))$ and the output $d'_h(a,b)$ are identically distributed according to [19]'s security proof, so that no PPT adversary can distinguish $S(d_h(a,b))$ and $d'_h(a,b)$.

Except for the three interactive protocols above, other computation process in our protocol are done locally by the corresponding party, so under the semi-honest model, party only get partial information of other parties' from his own frequent terms. During the divide-and-merged protocol, the clusters are encrypted only the cluster description is known. So any PPT adversary can not distinguish the responding output in real model from the one in ideal model the with any party's input. By using the zero knowledge proof, our protocol are also secure against the malicious party, but the computational and communication complexity will increase.

6. Conclusions and Future Works

In this paper, we propose a divide-and-merge method in distributed document clustering and give out a framework to preserve the privacy of participants. We use the cryptography techniques to construct a secure multi-party computation model for getting final agglomerative clusters without knowing the contend of the documents. Developing the efficient privacy-preserving data mining algorithms is also a challenging task. Our future work is to develop some new techniques with low communication and computation complexity to do other privacy-preserving data mining tasks.

文　献

[1] Rakesh Agrawal, Ramakrishnan Srikant. *Privacy-Preserving Data Mining.* ACM SIGMOD Int'l Conf. on Management of Data, Dallas, May 2000.

[2] Y. Lindell and B. Pinkas. *Privacy preserving data mining.* In Advances in Cryptology CRYPTO '00, volume 1880 of Lecture Notes in Computer Science, pp. 36-54. Springer-Verlag, 2000.

[3] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey, *Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections.* pp.318-329, Proc. ACM SIGIR 92, 1992.

[4] Oren Zamir, Oren Etzioni, *Web Document Clustering: A Feasibility Demonstration.* pp.46-54, Proc. of 21st ACM SIGIR on Research and Development in Information Retrieval, Melbourne, Australia, 1998

[5] Michael Steinbach, George Karypis and Vipin Kumar, *A comparison of document clustering techniques.* In KDD Workshop on Text Mining, 2000.

[6] Florian Beil, Martin Ester, Xiaowei Xu. *Frequent Term-Based Text Clustering,* Proceedings of the 8th Int. Conf. on Knowledge Discovery and Data Mining (KDD)'2002, 2002

[7] J. Vaidya and C. Clifton. *Privacy-Preserving K-Means Clustering Over Vertically Partitioned Data.* In Proc. of the 9th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, Washington, USA,2003.

[8] Geetha Jagannathan and Rebecca Wright *Privacy-Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data.* Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2005.

[9] D. Song, D. Wagner and A. Perrig. *Practical Techniques*

for Searches on Encrypted Data. In Proc. of the 2000 IEEE Security and Privacy Symposium, May 2000.

[10] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. *Public key encryption with keyword search.* In EUROCRYPT, Interlaken, Switzerland, 2004.

[11] Oded Goldreich. *Foundations of Cryptography Volume 2, Chapt.7,* Cambridge Univ. Press, 2004.

[12] A.C Yao, *Protocols for Secure Computation,* In 23rd FOCS, 1982

[13] O. Goldreich, S. Micali and A. Wigderson. emphHow to play any mental game. In Proceedings of the 19th annual ACM symposium on Theory of computing, 1987

[14] Wakaha Ogata and Kaoru Kurosawa. *Oblivious Keyword Search.* Journal of Complexity, 20(2.3)pp.356-371, 2004.

[15] Michael J. Freedman, Yuval Ishai, Benny Pinkas and Omer Reingold. *Keyword Search and Oblivious Pseudorandom Functions.* Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, 2005.

[16] Pascal Paillier. *Public-key cryptosystems based on composite degree residuosity classes.* In EUROCRYPT, Prague, Czech Republic,1999.

[17] Ivan Damgård and Mads Jurik. *Client/server tradeoffs for online elections.* volume 2274 of Lecture Notes in Computer Science, pp.125-140 ,PKC2002, 2002.

[18] Ven Laur, Helger Lipmaa and Taneli Mielikainen. *Private Itemset Support Counting.* volume 3783 of Lecture Notes in Computer Science, pp. 97-111, Beijing, China, 2005.

[19] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright. *Secure multiparty computation of approximations.*