Prometheus: Privacy-Aware Data Retrieval on Hybrid Cloud

Zhigang Zhou[1], Hongli Zhang[1], Xiaojiang Du[2], Panpan Li[1] and Xiangzhan Yu[1]
[1]School of Computer Science and Engineering, Harbin Institute of Technology，Harbin, 150001, China
Email: {zhouzhigang, zhanghongli, lipan, yxz}@ pact518.hit.edu.cn
[2]Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
Email: dux@temple.edu

***Abstract*—With the advent of cloud computing, data owner is motivated to outsource their data to the cloud platform for great flexibility and economic savings. However, the development is hampered by data privacy concerns: Data owner may have privacy data and the data cannot be outsourced to cloud directly. Previous solutions mainly use encryption. However, encryption causes a lot of inconveniences and large overheads for other data operations, such as search and query. To address the challenge, we adopt hybrid cloud. In this paper, we present a suit of novel techniques for efficient privacy-aware data retrieval. The basic idea is to split data, keeping sensitive data in trusted private cloud while moving insensitive data to public cloud. However, privacy-aware data retrieval on hybrid cloud is not supported by current frameworks. Data owners have to split data manually. Our system, called Prometheus, adopts the popular MapReduce framework, and uses data partition strategy independent to specific applications. Prometheus can automatically separate sensitive information from public data. We formally prove the privacy-preserving feature of Prometheus. We also show that our scheme can defend against the malicious cloud model, in addition to the semi-honest cloud model. We implement Prometheus on Hadoop and evaluate its performance using real data set on a large-scale cloud test-bed. Our extensive experiments demonstrate the validity and practicality of the proposed scheme.**

***Index Terms*—data retrieval, hybrid cloud, MapReduce, privacy-aware, data partition.**

I. Introduction

Cloud computing services enable organizations outsource data with ease and low cost. In recent years, several large IT companies have provided their own cloud platform, such as Amazon's EC2 and S3, Google AppEngine, IBM's Blue Cloud, and Microsoft Azure platform. Cloud offers a number of advantages: location independent data storage, ubiquitous data access and on-demand high quality services [2]. On the other hand, data owners lose direct control of systems that store and maintain their data, which inevitably causes potential security concerns. Data owner may have private data, e.g., health records, credit card records, etc. Such security concerns have been aggravated by the recent incidents of Gmail data breach and Amazon outages [3].

A natural solution is to encrypt data before outsourcing them to cloud [4-6]. However, data processing based on the encryption (or other cryptographic operations) is not efficient. For example, existing techniques, such as homomorphic encryption [7], secret-sharing [8], are still not up to the challenge posed by large data retrieval.

Another solution to this problem is data partition technique, which acts on plaintext dataset. It separates the attribute set (with semantic info.) into several subsets without semantic information. This technique eliminates functional dependencies among data attributes, and hence it's a good approach for protecting data privacy. However, there are two challenges: 1) data is outsourced to cloud in the form of plaintext, and cloud may be able to reconstruct the original data; 2) how to partition the data. Previous studies (e.g., [9, 10]) are associated with specific applications, and need to construct constraint rule set by application domain experts. This kind of approaches has limited scopes.

Secure hybrid cloud [1] is a new type of framework for cloud computing, and it is based on the following facts: *data analysis task (such as intrusion detection, credit analysis of bank users, and medical data analysis) involves both public and sensitive data, and we need to separate sensitive data from public data.* The main idea of secure hybrid cloud is to partition data into two parts, where sensitive part is stored in the private cloud within the organization, and sanitized data (without sensitive info.) is handled by public cloud. Secure hybrid cloud could solve the data privacy problem. However, current cloud frameworks, such as MapReduce [12], do not support privacy-aware data operations (such as data search and retrieval) in hybrid cloud. On the other hand, existing secure hybrid cloud solutions do not provide a data partition method, and they require users to manually split data. Hence, it is crucial to develop efficient privacy-aware data retrieval techniques for hybrid cloud. The information gain approach [11] may be used for data separation. However, it does not provide data privacy because it cannot eliminate quasi-identifier (QID) in data set, where the QID is a set of attributes that can identify an object.

Our goal is to achieve both system security and good usability. Our scheme exploits a data fragment technique to automatically separate core data from a given data set. We use attribute hypergraph to extract core data, and the rest of the data is stored in public cloud. Our scheme ensures that data in public cloud do not include QID. In order to minimize data load in private cloud, we provide an interactive interface by which a data owner can further reduce core data in private cloud. We build a prototype system, named Prometheus, which is a privacy-aware data retrieval system. Prometheus includes mapping operation and reducing operation, where mapping operation divides user data retrieval job into many sub-tasks and distributes them to different clouds based on the data fragment; and reducing operation is responsible for receiving and restructuring results returned from hybrid cloud. We also show that our scheme can defend against the malicious cloud model, in addition to the semi-honest cloud model. Formal security analyses and extensive experiments show that the proposed scheme is provably secure and highly practical. Our contributions can be summarized as follows:

1) To the best of our knowledge, this is the first study that formalizes the problem of secure data retrieval on hybrid cloud and provides a complete system design. Our design is based on the popular MapReduce, and it achieves the privacy-preserving data retrieval functionality.

2) We give formal security proofs to rigorously show the privacy-preserving guarantee of our scheme and that an honest-but-curious cloud service provider (CSP) can gain zero-knowledge. Furthermore, we show that our scheme can defend against malicious CSP.

3) We provide thorough analysis of privacy guarantee and efficiency of the proposed scheme. We implement our scheme on Hadoop and evaluate it in a large-scale cloud testbed, using real data set. Our extensive experiments further demonstrate the validity and practicality of our scheme.

The rest of the paper is organized as follows. Section II introduces the system and threat model, and our design goal. Section III provides the detailed description of our scheme, followed by security analysis in Section IV. Section V describes our strategy against malicious CSP. Section VI presents experiment results. Section VII overviews related work, and Section VIII concludes this paper.

II. System and Threat Model

System Model. A high-level architecture for data retrieval over hybrid cloud is illustrated in Figure 1, which involves four different entities: data owner, authorized user, private cloud and public cloud. Here, the data owner may represent either an individual or an organization, who has a collection of relation schema $S(X,A,F,d)$ to be stored in the hybrid cloud, where X denotes the object set $\{x_1,x_2,\ldots,x_n\}$; A is the attribute set $\{a_1,a_2,\ldots,a_m\}$; d is the decision attribute, $d \in A$; F is the relation set between X and A, and $F=\{f_k: X \rightarrow V_k\}$, where V_k is the value range of a_k $(a_k \in A)$. Two tasks need to be done before using our system. First, to prevent unauthorized users access data, before data owner uploads data to private cloud, data access control strategy has to be applied. We assume that the authorization (access control) to users has been appropriately done. Second, to move as much data to public cloud as possible, private cloud only stores core/sensitive data. An authorized user generates a search job and submits it to private cloud. Prometheus automatically maps the job into a suit of ordered tasks according to data distribution (between private and public clouds). Then these tasks are orderly distributed into interrelated clouds based on dependency of the requested data. In [1], public cloud sends computing result to private cloud and data reduction is completed in private cloud. We use a different approach. To minimize the inter-cloud communication, Prometheus employs a data integration tool to reduce data at user end. Specifically, when returning search results, besides sending results from hybrid cloud, private cloud sends mapping table records related with this search to user end as well. Then the user end obtains results according to the mapping table. Note that, in this paper data are organized in a relational table, while our proposal can be adapted to generic resource scenes.

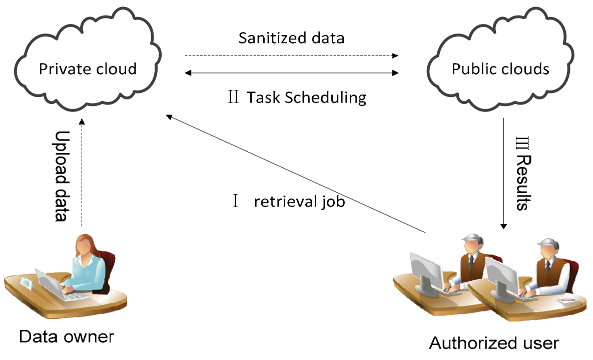


Figure 1. Data retrieval over hybrid cloud

Threat Model. We consider public CSP as "honest-but-curious", meaning that the CSP will follow the protocol but may attempt to derive additional (private) information from data. Specifically, public CSP acts in an "honest" way and correctly follow the designated protocols and business rules. However, it is "curious" to deduce and analyze data in its storage and temporary result sets to learn sensitive information. On the other hand, private cloud is built by the organization and assumed to be trustworthy.

Definition 1 (Confidentiality Privacy). Given a relation schema $S(X,A,F,d)$, and $B \subseteq A$. Confidentiality privacy over S is denoted as $f_B(X) \rightarrow V_d$, where V_d is the range of d.

Definition 2 Quasi-Identifier (QID). On the relation schema $S(X,A,F,d)$, for attribute set A, B, where $B \subseteq A$. $iff\ R_B = R_A$, and for $\forall Z \subset B$, $R_Z \neq R_A$, we call B as an quasi-identifier over S.

TABLE I. Raw Data for Fitting Contact Lenses

X	a_1	a_2	a_3	a_4	d
x_1	young	myope	normal	yes	hard
x_2	old	myope	reduce	no	hard
x_3	young	myope	normal	yes	hard
x_4	young	hyperope	reduce	no	none
x_5	young	hyperope	reduce	no	none
x_6	young	hyperope	more	no	soft
x_7	young	hyperope	reduce	no	none
x_8	young	hyperope	more	no	soft

a_1: age	{young, old}
a_2: spectacle-prescript	{myope, hyperope}
a_3: tear-prod-rate	{reduced, normal, more}
a_4: astigmatism	{no, yes}
d: contact-lenses	{soft, hard, none}

The semantics of confidentiality privacy is as follows. Definition 1 states that some knowledge contained in the data attributes plays an important role in decision-making (determining V_d). A QID is sensitive and it involves multiple attributes, therefore these attributes should not be published in combination. The following example illustrates this point. Example: Table I is a data set for fitting contact lenses, which has eliminated single attribute identifier.

It's easy to see that both attribute sets {a1, a3} and {a2, a3} are QID, each of which contains enough information to determine the tag attribute d. In addition, a public CSP can deduce some decision information using statistics, e.g., let minimum_confidence=75%, a public CSP could learn the following knowledge:

tear-prod-rate := reduced ==> contact-lens := none,
tear-prod-rate := normal ==> contact-lens := hard,
tear-prod-rate := more ==> contact-lens := soft .

Design Goals. We want to enable privacy-preserving and practical data retrieval over hybrid cloud under the aforementioned model. Specifically, our system has been designed to achieve the following goals: 1) Strong privacy guarantee: To prevent a CSP from learning either private data or statistical information of the data; 2) High efficiency: moving as much workload to public cloud as possible, given that the data privacy is preserved; 3) Usability: The system should be transparent to users. The system should have low communication and computation overheads.

III. THE PRIVACY-PRESERVING DATA RETRIEVAL FRAMEWORK

First, we give some details about the data partition technique. Then, we present a two-step privacy-preserving data retrieval framework based on MapReduce.

Preliminaries:

Equivalence Partitioning. On relation schema $S(X, A, F, d)$, $[x_i]_A$ is an equivalence partitioning on X, iff: $\forall x_i, x_j \in [x_i]_A$, $f_k(x_i) = f_k(x_j)$ $(\forall a_k \in A)$. We use R_A to denote the equivalence partitioning set over attribute set A, and the element of R_A satisfy: for $\forall [x_i]_A \neq [x_j]_A$, $[x_i]_A \cap [x_j]_A = \phi$ and $\bigcup_{k=1}^{n} [x_i]_A = X$.

Discernibility Matrix. On relation schema $\mathrm{S(X, A, F, d)}$, $\mathrm{D}\left([\mathrm{x_i}]_\mathrm{A}, [\mathrm{x_j}]_\mathrm{A}\right) =$

$$\begin{cases} a_k \in A | f_k(x_i) \neq f_k(x_j), & [x_i]_d \cap [x_j]_d = \emptyset \\ \phi, [x_i]_d \cap [x_j]_d \neq \emptyset \end{cases}$$

, where $\mathrm{D}\left([\mathrm{x_i}]_\mathrm{A}, [\mathrm{x_j}]_\mathrm{A}\right)$ is the attribute discernibility set of $[\mathrm{x_i}]_\mathrm{A}$ and $[\mathrm{x_j}]_\mathrm{A}$. $\mathrm{D} = (\mathrm{D}\left([\mathrm{x_i}]_\mathrm{A}, [\mathrm{x_j}]_\mathrm{A}\right) \big| [\mathrm{x_i}]_\mathrm{A}, [\mathrm{x_j}]_\mathrm{A} \in \mathrm{R_A})$ is the discernibility matrix based on X.

Attribute Hypergraph. Attribute hypergraph is a tuple (*V*, *HE*), where *V* denotes vertex set consisting of all the attributes, *HE* denotes hyperedge. One hyperedge represents a cell in the discernibility matrix.

A. Data Partition

Partitioning data without jeopardizing privacy is our first task. Not every attribute has the same importance for making decision, that is, some attributes provide more information than others with respect to decision making. Therefore, attributes are not equally contributing to a decision process. Safe data partition must satisfy the following condition: there is no confidentiality privacy in each data fragment. That is, an attribute subset containing QID needs to be partitioned into different fragments.

To address this problem, we utilize an automatic data partition technique. For completeness, we provide a fully formalized design in Algorithm 1 and briefly describe the technique below, using the same example in subsection II-A. The idea is to find QIDs that can identify a unique record. First, we establish the equivalence class of the attribute set A and d on object set:$R_A = \{\{x_1, x_3\}, \{x_2\}, \{x_4, x_5, x_7\}, \{x_6, x_8\}\}$, and $R_d = \{\{x_1, x_2, x_3\}, \{x_4, x_5, x_7\}, \{x_6, x_8\}\}$. Then we find the discernibility matrix **D** that can identify various attributes of different equivalence classes.

D =

$$\begin{bmatrix} \phi & \phi & \{a_2, a_3, a_4\} & \{a_2, a_3, a_4\} \\ \phi & \phi & \{a_1, a_2\} & \{a_1, a_2, a_3\} \\ \{a_2, a_3, a_4\} & \{a_1, a_2\} & \phi & \{a_3\} \\ \{a_2, a_3, a_4\} & \{a_1, a_2, a_3\} & \{a_3\} & \phi \end{bmatrix} \quad (1)$$

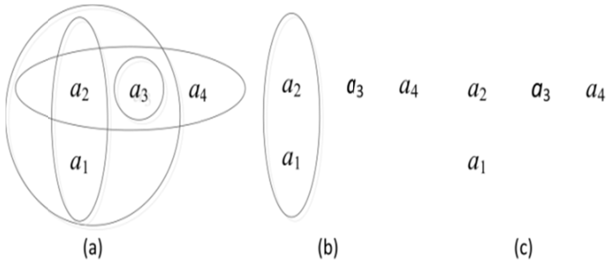


Figure 2. Attribute hypergraph

$$\begin{cases} C = B_1 \cap B_2 = \{a_3\} \\ R = (B_1 \cup B_2) - C = \{a_1, a_2\} \\ L = A - (B_1 \cup B_2) = \{a_4\} \end{cases} \quad (2)$$

Algorithm 1 Finding QID using hypergraph

```
INPUT:
HE{ $\sum_{i=1}^{k} \sum_{j=i+1}^{k} D[i][j]$ } // each cell $D[i][j]$ denotes one edge
OUTPUT:
$B$
MAIN
1: c=0; count =0; e=0; $B$=NULL;
2:  while (HE ≠ NULL) do
3:       for each HE$_k$ do
4:            for $i \leftarrow 0$ to |D| do
5:                 if (HE$_k$⊂HE$_i$ ) then
6:                      count++;
7:                 Temp.add(HE$_i$)
8:            if (c<count)
9:                 c =count; e=k; Edge=Temp;
10:  HE=HE – Edge;
11:  $B$=$B$ × HE$_e$;
12:  return $B$;
```

According to definition 2, QID is not unique. We propose an attribute hypergraph digestion method to obtain the QID, as illustrated in Fig. 2(a). The rules of the hypergraph digestion are given below: (1) finding a hyperedge that is contained by

3

the most hyperedges; (2) deleting all hyperedges containing the chosen one; (3) repeating step (1) until there is no hyperedge. For the example in Table I, according to the hypergraph digestion rules, first, $\mathrm{HE}(a_3)$ is chosen, deleting all hyperedges containing attribute a_3, and we have Fig. 2(b). Second, choosing $HE(a_1, a_2)$, and deleting it, and we have Fig 2(c). Now, we can get $B = a_3 \times (a_1, a_2)$ (details given in the proof of Theorem 1, Section IV). That is, $B_1 = \{a_1, a_3\}$, $B_2 = \{a_2, a_3\}$. Here, attributes can be divided into three different classes: core attribute set, relative necessary attribute set, low information attribute set. The core attribute set C has the most capable for decision analysis, which is selected from the QID set, making the rest of attributes cannot form the QID, where $C = \bigcap_{k \le |B|} B_k$; the relative necessary attribute set $R = \bigcup_{k \le |B|} B_k - C$; while the low information attribute set $L = A - \bigcup_{k \le |B|} B_k$. Formula (2) expresses the result of data partition.

```
Algorithm 2  Attribute partition
INPUT:
$B\{B_1, \ldots, B_m\}$
$S\{a_1, \ldots, a_r\}$
OUTPUT:
$P$
MAIN
1: for $i \leftarrow 1$ to $r$
2:    $M[i]=0;$ /*initial mark */
3: while $B \neq NULL$ do
4:    $j{+}{+}$; $k=|B|$; $T=False$; $S_j = S$;
5:    for $q \leftarrow 1$ to $j$ do
6:       for $i \leftarrow 1$ to $k$ do
7:          if $B_i \subseteq S_j$ then
8:             $T=True$;
9:             Search correspond M mark of each attribute
               belonging to $B_i$;
10:            if $\nexists a_t \in B_i$ and $M[t]==1$ then
11:               Choose attribute $a_e \in B_i$, let $M[e]==1$;
12:         Move all attribute where the corresponding Mark
            is 0 belonging $B_i$ from $S_j$ to $P_{j+1}$;
13:      $B = B - B_i$;
14:   if $T==False$ then
15:      break;
16:return $P$;
```

Discussions. We discuss two cases, focusing on the QID set B: *1)* the core attribute set exists; *2)* the core attribute set doesn't exist, that is, $C = \bigcap_{k \le |B|} B_k = \phi$. For case *1)*, $\forall B_k \in B$, there exists $a_i \in B_k$ and $a_i \in C$. According to Definition 2, the set $B_k - \{a_i\}$ cannot form the QID. For case *2)*, we need to split B into n classes such that each class contains no QID and the number of classes is minimized (Min-Partition). The Min-Partition problem can be converted to the minimum vertex coloring problem [13], which is formulated as follows: Given a graph *G(V,E)*, determine a minimum coloring of *G*, that is, assign to each vertex in V a color such that adjacent vertices have different colors, and the number of colors is minimized. The correspondence between the Min-Partition problem and the minimum vertex coloring problem is given below. The entire data set is defined as a schema *S(B)*. A vertex v in G corresponds to an attribute $a \in B$. An edge (v_i, v_j) in G corresponds to a conflict between a_i and a_j with respect to B, that is, a_i and a_j constitute a QID. A solution to the minimum vertex coloring problem corresponds to a minimum partition of B. Specifically, the color assigned to a vertex v in G corresponds to the class including attribute a represented by vertex v. Therefore, any algorithm that solves the minimum vertex coloring problem can be used to solve the Min-Partition problem. We solve the Min-Partition problem by converting it into the minimum vertex coloring problem. We utilize heuristics and approximation algorithms, which efficiently compute a near optimum solution. We define a set of conflict rules to model conflicts between attributes in B, where each conflict rule is one cell of B. Algorithm 2 is the attribute partition algorithm, which ensures no QID exist in any partition. For completeness, we briefly introduce the technique below. The idea is to use bitmap to mark attributes that is the part of the conflict rules$B_i (i = 1, \ldots, |B|)$, dividing each B_i into different partitions. Note that, for any conflict rule B_k, dividing unmarked attributes into the exist partition does not affect the previous conflict rules $B_j (j < k)$. If the attributes contained in B_k are all marked, then generating a new partition and randomly selecting one attribute in B_k to join it.

B. Attribute Labeling and Privacy Table

Attribute Labeling. To perform a privacy-aware user job splitting, Prometheus employs an attribute-marking tool. The tool marks out the location of attributes, which are distributed into different cloud server according to the count of hiring cloud servers and attribute partition condition conducted by Algorithm 2. Specifically, it maintains an attribute location table: creating one record (tablename, attribute, location) for each attribute, where the value range of "location" is{private_cloud, public_cloud}. For simplicity, we use one public cloud as an example for attribute labeling. If there exist the core attributes, then tags their location as private_cloud; otherwise, taging one partition containing maximum attributes as public_cloud, and the rest are tagged as private_cloud. Note that, our mechanism can be easily extended to multiple public clouds, where the same rule should be followed: Given Min-Partition set $P\{P_1, \ldots, P_n\}$, for $\forall a_i, a_j, a_i \in P_i$, $a_j \in P_j$, and $P_i \neq P_j$, then $location(a_i) \neq location(a_j)$.

Privacy Table. To further improve the practicality and flexibility of our system, Prometheus provides an interface to data owners, where data owners can keep some non-core data in the private cloud. It is possible that some data are not core data but the data owner wants to protect them. Similar to Sedic [1], Prometheus uses a string scan tool that searches keywords in non-core data. If the keywords are found, then privacy tables are used to store them, where each piece of sensitive information is a tuple *(tablename, attribute, recordID, value)*.

C. Data Balance

In the above data partition mechanism, the mapping relationship among data partitions is removed. However, this still cannot prevent a public CSP from inferring statistic correlations of attributes among partitions. To address this kind of attack, we propose a (β, k)-group balance strategy.

Definition 3 Data Distribution Function. $S(X, A, F, d)$ is a relation schema, $B \subseteq A$ and $\forall x_i, f_{a_k}(x_i) = f_{a_k}(x_j)$ $(a_k \in B)$, $R_B = \{[x_i]_B \mid x_i \in X\}$, $R_d = \{[x_i]_d \mid x_i \in X\}$. Denote $R_d = \{d_1, d_2, \ldots, d_{|R_d|}\}$ as the equivalence class of attribute d in X. For $[x_i]_B \subseteq R_B$, $D\left(\frac{d_k}{[x_i]_B}\right) = \frac{|d_k \cap [x_i]_B|}{[x_i]_B}$ $(k \leq |R_d|)$, the Probability Distribution Function is defined as $\mu_B(x_i) = (D(d_1/[x_i]_B), \ldots, D(d_{|R_d|}/[x_i]_B))$.

$\mu_B(x_i)$ lists data distribution of each attribute. To prevent a public CSP from obtaining statistical information, the first stage is to balance value distribution across class attributes.

Definition 4 (β, k)-Group Balance. Given a fragment instance P and a set of group identifiers GID_i. $|GID_i|$ denotes the count of the dataset. A k-group over S satisfies the minimal anonymity requirement: $count(v(d)) \geq k$, where $v(d)$ is the range of attribute d and k is the anonymity threshold. β is a distribution probability, which is the minimum of $\mu_{a_i}(x_j)$.

A k-group function for a fragment record is a group identifier that each group has at least k records mapping to. A k-group is minimal when the size of the groups is minimized. Grouping is based on the range of class attribute a_d, that is, $|GID_i| = |v(d)|$. β is the distribution balance threshold of the values of an attribute, which determines how to distribute the values of an attribute in all groups. Algorithm 3 describes the (β, k)-group balance strategy.

Algorithm 3 (β, k)-group balance strategy

```
INPUT:
$R_{a_i}$ ($i$=1,...,|A|), $R_d$, $\beta$, $k$;
OUTPUT:
$X_{new}$
MAIN
1: for each $a_i \in A$ do
2:     j = 1;
3:     while (j ≤ $|R_d|$) do
4:         m = 1;
5:         while (m ≤ $|R_{a_i}|$) do
6:             if (D($d_j$/($R_{a_i}$ [m]))< $\alpha$ )
7:                 Insert ($R_{a_i}$ [m]) → $d_j$;
8:         while (|$d_j$|<k) do
9:             Insert (Min ($R_{a_i}$ [1],..., $R_{a_i}$ [m]))→ $d_j$;
10:return $X_{new}$;
```

D. Map and Reduce

Mapping Task Scheduling. After outsourcing data to hybrid cloud, authorized user can submit search jobs to Prometheus. As data partition is transparent to user, user can use normal data retrieval statements. When Prometheus receives user's retrieval job, it first calls JobTracker that breaks the job into tasks. Each task is a TaskInProgress object. Then it creates a task execution scheduling that assembles TaskProgress objects into one JobInProgress queue according to the relationship of the tasks. The current Hadoop does not offer privacy-preserving data search on hybrid cloud. Our system provides such capability. The details are given below. Prometheus modifies InputSplit and adds an instance attribute, called "attr_loca_tag". This attribute sets the location of the data containing searched attribute. If the "attr_loca_tag =private cloud", then the task containing it is always scheduled to be executed in private cloud. Therefore the tag plays an important role in data privacy-preserving. Consider a general search statement: *" Select A.attr$_1$, A.attr$_2$,..., A.attr$_n$, B.attr$_1$ from Table$_1$ as A, Table$_2$ as B where A.attr$_{n+1}$=B.attr$_2$ and A.attr$_1$=value$_1$ and ... and A.attr$_n$=value$_n$ and B.attr$_1$=value$_{n+1}$;"* , here we assume *A.attr$_1$* is the core data and the part value of B.attr$_1$ is labeled as sensitive data in the private table; *A.attr$_2$* to *A.attr$_k$* are distributed into public cloud$_1$ while other data are distributed into public cloud$_2$. JobTracker breaks this search job into *(n+4)* search tasks where each search task can be decomposed into one tuple *(tablename, attrname, value, attr_loca_tag)* and two match tasks: 1) comparing values between *A.attr$_{n+1}$ and B.attr$_2$*; 2) merging the values of *B.attr$_1$*. There are three rules: 1) match task can be performed just after the corresponding search tasks are done; 2) any operation involving private data must be conducted in private cloud; 3) search results from multiple public clouds are combined together to reduce overhead.

Task Reducer. The task reducer provides data integration for hybrid cloud. First, we discuss two simple but inappropriate approaches. *1)* Public cloud completes data integration: In this case, private cloud sends reducer results from task executions on privacy data. This may cause leaking of privacy data and therefore it is inappropriate. *2)* Private cloud completes data integration: Public clouds need to send results to private cloud, which cause large inter-cloud communication overhead. And private cloud could become bottleneck.

To address the above problems, Prometheus places the reducer at user end. To reduce communication overhead between hybrid cloud and user end, public cloud can merge multiple tasks before sending them to user end. Besides returning search results, private cloud also sends mapping table records to user end. Then the reducer can construct the final search results at user end.

IV. Security Analysis

In this Section, we prove the security guarantee stated in subsection II-B. Similar to the Min-CF scheme [93], our data partition mechanism also needs to satisfy non-QID. That is, the security guarantee should ensure that there is no QID in each data partition.

Theorem *1*. Our data fragment design satisfies the requirement of data privacy protection.

Proof: The proof consists of three steps. First, we prove that the QID definitely exists. For $\forall a \in A, if\ R_{A-\{a\}} \neq R_A$, set A is a QID. If $\exists a \in$ A, $R_{A-\{a\}} = R_A$, then let $B = A - \{a\}$, by a finite number of iterations, we can get $\forall b \in B, R_{B-\{b\}} \neq R_A$, then B is a QID. Second, we prove that the set B is a QID. According to Algorithm 1, $B = \{a_i \mid \forall D\left([x_i]_A, [x_j]_A\right)$ $\in$ **D**, $\exists a_i \in D\left([x_i]_A, [x_j]_A\right)\}$ that is, $B \cap D\left([x_i]_A, [x_j]_A\right) \neq \phi$, where $D\left([x_i]_A, [x_j]_A\right)$ denotes all attributes that are different between $[x_i]_A$ and $[x_j]_A$. As the attribute selection in hypergraph, each round we select the hyperedge that is contained by most hyperedges. Then, B is the minimal attribute set that forms a QID. Third, we prove that no data partition contains a QID. There are two cases that should be discussed: with and without core attribute set. For the case with core attribute set, first, according to Algorithm 1 we have: "$if\ |B| \geq 2, \bigcap_{k \leq |B_k|} B_k \neq \phi$".

Second, we prove "if a is the core attribute, then $\forall x_i, x_j \in X$, $D([x_i]_A, [x_j]_A) = \{a\}$". This is proved by contradiction. Suppose that $\nexists x_i, x_j \in X$, $D([x_i]_A, [x_j]_A) = \{a\}$. That is, for $a \in D([x_i]_A, [x_j]_A)$, and $|D([x_i]_A, [x_j]_A)| \geq 2$, there is a set $B = \cup \{D([x_i]_A, [x_j]_A) - \{a\} \mid [x_i]_A, [x_j]_A = \phi\}$. Hence for $[x_i]_A, [x_j]_A = \phi$, there is a set $B \cap D([x_i]_A, [x_j]_A) \neq \phi$, then $R_B = R_A$. Therefore, $\exists C \subseteq B$ that makes C the minimal attribute set. However, $a \notin C$, it forms the contradiction with the hypothesis that a is the core attribute. Third, we prove this "if a is the core attribute, then $R_{B-\{a\}} \neq R_B$". According to step (2), $\exists x_i, x_j \in X$ that make $f_a(x_i) \neq f_a(x_j)$ and $f_b(x_i) = f_b(x_j)$. Then, $\{x_i, x_j\} \in R_{A-\{a\}}$, $\{x_i, x_j\} \notin R_A$, since $R_{A-\{a\}} \neq R_A$. Also, $R_B = R_A, B \subseteq A$, since $R_{B-\{a\}} \neq R_B$. We have the following result: $B - \{a\}$ is not a QID. Because B is the minimal attribute set and $a \in B$, attribute a can't form a QID; For the second case, we proved it in subsection Ⅲ-A. □

Next, we show that our scheme does not let a public CSP gain any knowledge of private data.

Theorem *2*. Our data fragment design provides Zero-Knowledge to a public CSP.

Proof: Data fragmentations are based on plaintext database, which divide the association relationship among data attributes. The real dataset is contained in the Cartesian product formed by the data fragmentations. A data fragment design is Zero-Knowledge, iff $Q < \varepsilon$, where Q denotes the probability of reconstructing the original dataset, and ε is a very small number. According to Algorithm 3, we use (β, k)-group balance strategy in fragment design, where the range of β's value is $[0, 1/|R_d|]$, with $\beta \to 1/|R_d|$, the attribute value distributes more uniform cross d's range. Similar to k-anonymity, for each value within d's range, the corresponding attribute has k equal values. Based on *Theorem 1*, the number of data partitions is related with the specific data set but it is at least 2. Thus, the probability that a public CSP can reconstruct the exact data set from query results is $1/k^n$ $(n \geq 2)$.

V. Defense Against malicious CSP

In this Section, we discuss a threat model that is stronger than the semi-honest model. We assume the CSP is malicious or un-trusted, that is, beyond "curious". The CSP could tamper data stored in public cloud; the CSP could only search a subset of the data. These could happen due to various reasons, e.g., the CSP wants to save its computation, software bugs, hardware failures, or even outsider attacks.

The Idea. Data are uploaded to cloud in plaintext format, and the CSP has full control of the cloud. Hence, there is no way to stop the CSP from modifying the data. To verify the integrity of outsourced data, the private cloud should compute the hash value of every minimal data query unit (MDQU) before outsourcing the data to public cloud (similar to the approaches in [14], [15]). Besides storing the logical view of the original dataset and the mapping relationship between metadata and the fragment structure, the private cloud should also store the new index entries and the hash values of each record in every fragment.

Definition 5 (MDQU). MDQU is a description for the query granularity. Consider $S(X, A, F, d)$ and $S'(X, B, F, d)$, where $B = \{b_1, \dots, b_k\}$ and $B \subseteq A$. That is to say, S' is a subset of S. For $\forall x_i \in X$, $V_{b_j}(x_i)$ denotes x_i's value on attribute b_j. $\{V_{b_1}(x_i), \dots, V_{b_m}(x_i)\}$ is the MDQU for x_i in S'.

Note that, according to our definition, MDQU is the minimal verifiable search execution unit in terms of cloud data. However, users don't know the result of data partition, and in order to support flexible operations, MDQU should be transparent to authorized users.

To verify the integrity of the outsourced data, we use a data structure called Taxonomy Index Tree (TIT).

Definition 6 (TIT). TIT is a tree structure. The root node represents the metadata of the dataset. Each internal node represents a specialization of its parent node on a set of attributes, which is a subset of the records contained by its parent nodes. Each leaf node represents a set of records that meet the constraints in the path from the root to the leaf. If the same value of an attribute is divided into different subtrees, to satisfy the range query, a link is maintained.

With the TIT, we can verify the integrity of query results from public cloud. The root node of the TIT represents the generalized record and all data records. The first layer is the result set of classifying the root node based on low information attributes. The next iteration classifies data records sourced from previous iteration by ordered using relative necessary attributes and core attributes in order. Therefore, TIT can be as a classification from coarse to fine. The leaf node consists of outsourced data's IDs, which satisfy the path constraints from root to leaf. So before public clouds return the search results, the private cloud could exactly check the number of records, and even the recordIDs.

As illustrated in Fig 3, TIT has a root node representing the generalized record <a1, a2, a3, a4, count (8)>. The last field is the number of outsourced records by this node. {a4} is the low information attribute set, {a1, a2} is the relative necessary attribute set, and {a3} is the core attribute set. First, the root node is divided into two parts by a4→{no, yes}. Second, two subtree are formed where the first layer are roots. Third, according to {a1,a2} → {{old, myope},{young, hypermetrope}, {young, myope}}, several new subtrees are formed at the second layer. Similarly, the third layer are formed based on the value of a3→{more, reduced, normal}. At the second layer, the records (whose attribute a1:= young and a2:= myope) are divided into different subtrees. Thus, these nodes are linked by Link_{a1} and Link_{a2}, respectively.

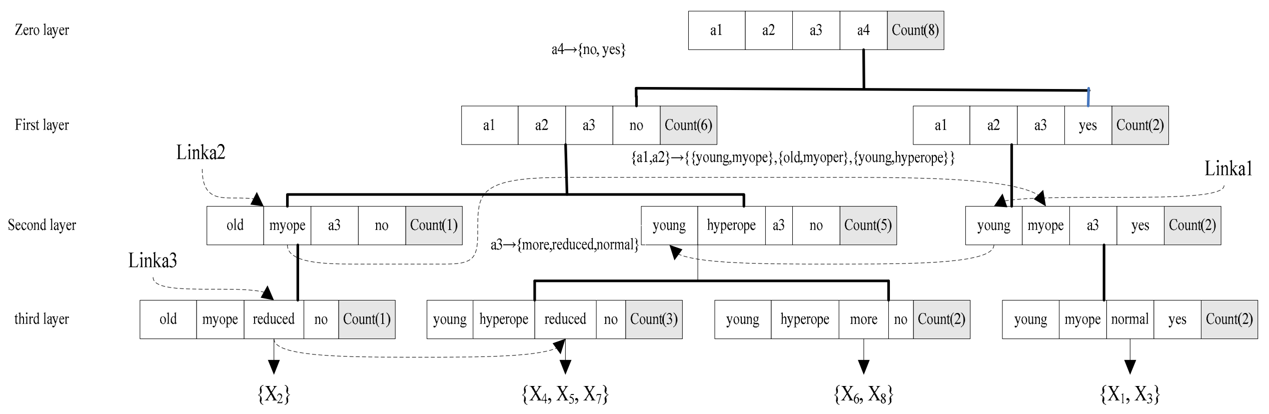

Zero layer
First layer
Second layer
third layer
a1 a2 a3 a4 Count(8)
a4→{no, yes}
a1 a2 a3 no Count(6)
a1 a2 a3 yes Count(2)
{a1,a2}→{{young,myope},{old,myoper},{young,hyperope}}
Linka2
Linka1
Linka3
old myope a3 no Count(1)
young hyperope a3 no Count(5)
young myope a3 yes Count(2)
a3→{more,reduced,normal}
old myope reduced no Count(1)
young hyperope reduced no Count(3)
young hyperope more no Count(2)
young myope normal yes Count(2)
{X2}
{X4, X5, X7}
{X6, X8}
{X1, X3}

Figure 3. Taxonomy index tree architecture

In the Verification phase:

1) Hybrid cloud sends search results to the authorized user. In addition, private cloud sends the recordIDs (obtained from the TIT), hash values of data partitions in the corresponding recordIDs, and the mapping table records involved to the user.

2) The authorized user checks the number of results, and then verify the hash values of some randomly selected data (no more than a certain threshhold σ). If the results pass the verification, user will rebuild the records according to the mapping table. Otherwise, it means that the CSP has cheating behaviours. Then the user sends a four-tuple (user, research request, {false recordIDs}, error_no) to the private cloud, where the value range of error_no is {1, 2, 3}: "1" - data tampering; "2" - lazy operation (only returns partial data); "3" - both.

3) Private cloud has penalty for malicious public cloud. According to the Nash equilibrium strategy [16] of Game Theorey, if the penalty is more than the payoff of defection, no public cloud has incentive to cheat. Table II is the payoff matrix of the game for hybrid cloud. Here, to simplify the model, hybrid cloud is divided into two different classes: the malicious CSP (Pub_CSP_k) is class 1, while other CSP is class 2. In each round, suppose the total utility is w, and $\overline{Pub\_CSP_k}$ is rational. In the issue of (Trust, Cheat), if Pub_CSP_k is cheating but not detected (due to the random selection of checking), this negatively affects authorized users' credibility on all public clouds in the next round, therefore, $\overline{Pub\_CSP_k}$ gains utility 0. There does not exist the action pair (Punish, Cooperate), where Punish could be move data away from Pub_CSP_k. For the action pair (Trust, Cooperate), let p be the portion that Pub_CSP_k gains utility w and $1-p$ be the portion for $\overline{Pub\_CSP_k}$. For the action pair (Punish, Defect), we move all the utility from Pub_CSP_k to $\overline{Pub\_CSP_k}$, thus, $\overline{Pub\_CSP_k}$ gets utility w, and Pub_CSP_k gains zero utility. Based on the above discussion, we have Formula (3), where u denotes the utility of Pub_CSP_k, $\lambda = 1-\sigma$ (λ is the probability that we check Pub_CSP_k in each round), and M denotes the penalty that we give to Pub_CSP_k. When M is larger than u, there is no incentive for a public CSP to cheat.

$$u = \lambda w + \lambda^2 w + \cdots + \lambda^n w + \cdots = \frac{\lambda w(1-\lambda^n)}{1-\lambda} \leq M \quad (3)$$

TABLE II. PAYOFF MATRIX OF DATA AUDIT GAME

Pub_CSP_k \ $\overline{Pub\_CSP_k}$	Cooperate	Cheat
Trust	*((1-p)w, pw)*	*(0,w)*
Punish	*Do not exist*	*(w,0)*

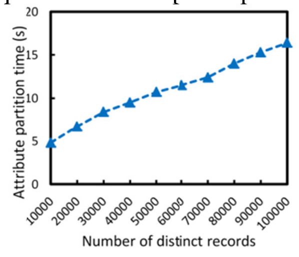
VI. PERFORMANCE EVALUATION

We conduct a thorough experimental evaluation of the proposed techniques on real data set: Intrusion Detection Evaluation (IDE) [18]. We built a hybrid cloud in our lab. The public cloud includes three nodes: each with cores of Xeon E5506, 2.13GHz CPUs, 16G dual-channel 1333GHz memory, 2T 7200 RPM disk and also Ubuntu 12.04 LTS OS with Xen 4.0.3. The private cloud contains the same two nodes. The authorized user side runs in PC with two 2.1GHz Intel Core2 Duo T6570 CPU, 4GB RAM, 7200RPM disk drive and Linux CentOS 4.8 operating system. Note that in our experiment, we do not include the data transmission time in the cloud testbed, which is an inherent overhead of using the cloud platform. The core components of Prometheus are data partition, tasks scheduling, and results reconstruction subsystems. We choose different record collections from the IDE data set, where the number of a collection ranges from 10,000 to 100,000, we conduct experiments over each collection multiple times, and report the average in the paper.

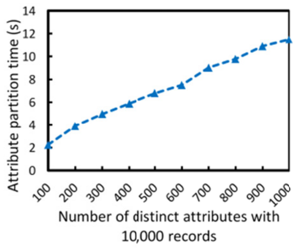
A. Evaluation of Preprocessing

1) Cost For Data Partition: We first evaluate the performance of the data partition technique, which was discussed in subsection III-A. Fig. 4(a) shows data partition time for collections IDE data set ranging from 10,000 to 100,000. For completeness, we also conduct additional experiments where we generate the relation schema following the TPC-H [17] benchmark specifications. The relation schema considered in the experiments consists of 10,000 records with 1,000 attributes. Our experiments consider configurations with various numbers of attributes, from 100 to 1,000. Fig. 4(b) shows the time cost of attribute partition, with attribute counts varying from 100 to 1,000, in steps of 100.

Experimental results show that the time is a linear function of the attributes. Though the data partition time of both tests is in the order of seconds, we want to point out that the entire data partition operation is just a one-time cost and can be conducted off-line. Both the experiments show that our data partition technique is quite efficient.



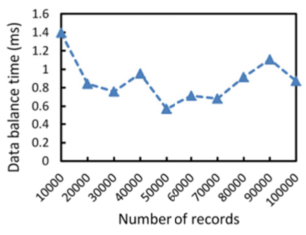
(a) Attribute partition time with 42 attributes in IDE.



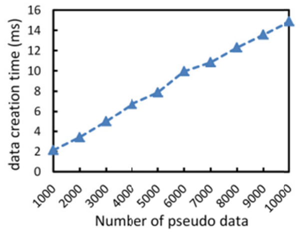
(b) Attribute partition time with 10,000 records.

Figure 4. Evaluation for attribute partition.

2) Cost For Data Balance: In our experiment, the data balance operation is conducted on the private cloud. To test the data balance capability, we conduct two sets of experiments. The first set of experiments uses the IDE data set, and the results are shown in Fig. 5(a), where the data balance time for different number of records is plotted. To further measure the pseudo-data construction performance of our scheme, we run the second set of experiments, and the results are plotted in Fig. 5(b), which shows the linear relationship between the number of generated pseudo-data and construction time.



(a) Data balance time.



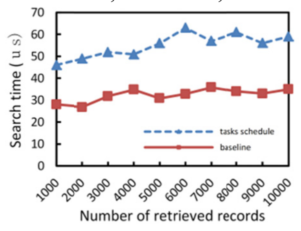
(b) Pseudo data construction time.

Figure 5. Evaluation for data balance cost and pseudo data construction.

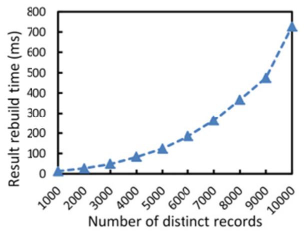
B. *Evaluation of Data Retrieval*

1) *Cost For Data Retrieval:* The data retrieval operation is conducted on the hybrid cloud testbed, where the private cloud first generates tasks for a user's retrieval job, according to the data partition condition; then the public cloud implements the corresponding tasks. Fig. 6(a) shows the cost of generating tasks, executing tasks, and data retrieval. For comparison, we include the hadoop-based data retrieval time as a baseline. Fig. 6(a) confirms the near-constant time property of the proposed scheme. As the baseline scheme does not need task schedule for user's retrieval job, it always performs faster than our scheme. However, our scheme provides secure search without privacy breaches. The search time of our scheme is only a little bit longer than the baseline, so it is still quite efficient.

2) *Cost For Reconstruction:* The search returns partitioned result set, which has to be reconstructed by the user according to mapping table returned by the private cloud. This operation requires few processing capabilities. It is determined by the number of returned results. Fig. 6(b) is the time to reconstruct the results, with record counts varying between 1,000 to 10,000.



(a) Cloud side search cost.



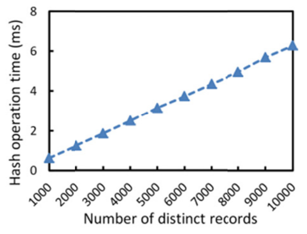
(b) User side reconstruction cost.

Figure 6. Evaluation for search cost and data reconstruction.

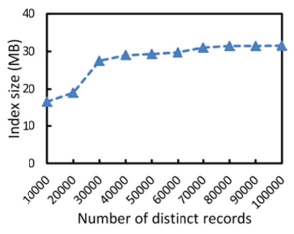
C. *Evaluation of Verification Cost*

1) Cost For Hash Operation: To verify the integrity of search results returned by public cloud, Prometheus stores the hash value of each record before outsourcing data to public cloud. Fig. 7(a) shows the time of computing hash values for all the test records. Experiment results confirm the linear dependency on the number of test records.

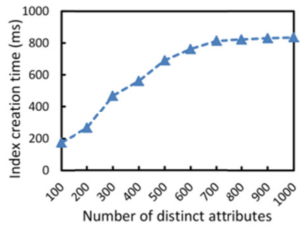
2) Cost For Building TIT Index: Before our test, we conduct interval dividing preprocessing operation for numerical attribute data. Fig. 7(b) shows the TIT index construction time for the IDE data set. Unlike the pseudo-data generation, the TIT index storage size does not increase linearly with the record number. We conduct two groups of experiments on our constructed data set. The data set is the same as that used in testing data partition. The first set of experiments measure the influence of the number of attributes on the efficiency of building TIT index. As shown in Fig. 7(c), the time cost does not linearly increase with the number of attribute, which is consistent with the discussion in subsection III-A: only core attributes and relative necessary attributes may affect the data decision. The second set of experiments (Fig. 7(d)) show that the index storage cost is not a linear function of the number of attributes.



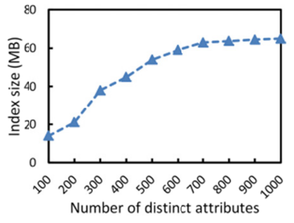
(a) Generating hash value cost.



(b) TIT index storage size.



(c) Index construction cost.



(d) Index storage size.

Figure 7. Evaluation of verification cost.

VII. Related Work

Much work has been done on privacy-preserving data outsourcing. Previous work mainly uses data encryption, e.g., [4-6]. However, data encryption causes large overhead for search, retrieval, and other data operations. Some recent studies focus on restriction-based techniques and perturbation-based techniques. Restriction-based techniques (e.g., [19, 20]) protect data privacy by adding restriction on query. V.Ciriani et al. [25] propose using fragmentation lattice to split the connection among data attributes, which builds self-contained attributes lattice sets. S. De Capitani di Vimercati et al. [9] propose a safe fragmentation scheme based on constraints to split sensitive associations while ensuring visibility. All these approaches, however, need the application field experts to build constraint rules. Swift [26] uses a secure information flow analysis to partition a web application into client and server components. In perturbation-based approaches, the system computes the correct result and outputs a perturbed version of the result by adding noise [24]. Traditional k-anonymity [22], ℓ-diversity [23], and confidence bounding [9, 10] are based on a predefined set of QID attributes. None of these techniques are designed for privacy-preserving data outsourcing. In this paper, we design an automatic data partition scheme without setting constraint rules, and we adopt the TIT structure to verify the integrity of data retrieval results. Our approach and t-closeness [21] share the idea that the distribution of a sensitive attribute in any group should be close to the distribution of the attribute in the overall data set. Our work achieves this goal by balancing the data distribution of attributes in decision domain.

VIII. Conclusion

In this paper, we studied the important problem of privacy-preserving and efficient data retrieval over hybrid cloud. We want to efficiently search outsourced data without jeopardizing data privacy. In our design, first we exploited an automatic secure data partition technique that keeps core data within the private cloud, while moving non-sensitive data to public cloud. Second, we leverage the feature of MapReduce to support privacy-aware job-tracker with low inter-cloud communication overhead. We formally proved the privacy-preserving feature and the correctness of the proposed scheme. In addition, we demonstrated that our scheme can defend the malicious cloud model. We conducted real experiments in our hybrid cloud testbed, and our results showed that the proposed scheme indeed has low overhead.

Acknowledgment

This work is supported in part by the project of National Natural Science Foundation of China (60903166,61100188, 61173144), National Basic Research Program (973 Program) of China (2011CB302605) and National High Technology Research and Development Program (863 Program) of China (2010AA012504), as well as by the US National Science Foundation under grants CNS-0963578, CNS-1022552 and CNS-1065444.

References

[1] K. Zhang, X. Zhou, and Y. Ruan. Sedic: Privacy-Aware Data Intensive Computing on Hybrid Clouds. *In Proc. of CCS'11*, pages 515-525, Chicago, Illinois, USA,2011.ACM.

[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," ACM SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 50–55, 2009.

[3] Summary of the amazon ec2 and amazon rds service disruption in the us east region. http://aws.amazon.com/message/65648/, 2011.

[4] M. Blanton and M. Aliasgari. Secure outsourcing of dna searching via finite automata. Volume 6166 of Lecture Notes in Computer Science, pages 49–64. Springer, 2010.

[5] M. Atallah, F. Kerschbaum, and W. Du. Secure and private sequence comparisons. *In Proc. of WPES '03*, pages 39–44, 2003.

[6] M. Atallah and J. Li. Secure outsourcing of sequence comparisons. Int. J. Inf. Secur., 4:277–287, October 2005.

[7] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. Advances in Cryptology - EUROCRYPT 2010, volume 6110 of Lecture Notes in Computer Science, pages 24–43. Springer Berlin / Heidelberg, 2010.

[8] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Overencryption: Management of Access Control Evolution on Outsourced Data.VLDB: 123-134. Vienna, Austria, September 2007.

[9] S. De Capitani di Vimercati *et al.*. Fragments and Loose Associations: Respecting Privacy in Data Publishing. *In Proc. of VLDB*:1370-1381. 2010.

[10] V. Ciriani *et al.*, Fragmentation Design for Efficient Query Execution over Sensitive Distributed Databases. *In Proc. of ICDCS'09*, 2009.

[11] Noman Mohammed, Benjamin C. M. Fung, Mourad Debbabi. Anonymity meets game theory: secure data integration with malicious participants. VLDB Journal(2011) 20:567-588.

[12] J. Tan, X. Meng *et al.*. Design and Performance Analysis of Coupling Scheduler for MapReduce/Hadoop. *In Proc. of INFOCOM*, 2012.

[13] M. Garey and D. Johnson. Computers and Intractability. Freeman and Company, 1979.

[14] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. *In Proc. of ACM CCS*, 2006, pp. 79–88.

[15] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in Proc. of CRYPTO, 2007, pp. 535–552.

[16] John Nash. Non-cooperative games. Ann. Math. 54(2), 286–295(1951).

[17] The transaction processing performance council (TPC) benchmark H. http://www.tpc.org/tpch/.

[18] IDE, "DARPA Intrusion Detection Evaluation" http://www. ll.mit. edu/mission/communications/ist/corpora/ideval/data/1999data.html.

[19] A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Modeling and assessing inference exposure in encrypted databases. ACM TISSEC, 8(1):119–152, February 2005.

[20] E. Damiani *et al.*,. Balancing confidentiality and efficiency in untrusted relational DBMSs. *In Proc. of CCS '03*, October 2003.

[21] N.Li, T.Li, S.Venkatasubramanian. t-closeness: privacy beyond k-anonymity and ℓ-diversity. In: ICDE 2007.

[22] L. Sweeney : k-anonymity: a model for protecting privacy. In: International Journal on Uncertainty, Fuzziness and Knowledge-based Systems .2002b.

[23] A. Machanavajjhala, J. Gehrke, and D. Kifer. ℓ-diversity: Privacy beyond k-anonymity. *In Proc. of ICDE*, USA, 2006.

[24] C. Dwork, F. McSherry, K. Nissim, A. Smith. Calibrating noise to sensitivity in private data analysis. *In Proc. of TCC*, 2006.

[25] V.Ciriani,S. De Capitani di Vimercati et al. Fragmentation Design for Efficient Query Execution over Sensitive Distributed Databases. *In Proc. of ICDCS*, Canada, June 22-26, 2009.

[26] S. De Capitani di Vimercati *et al.*, Integrating Trust Management and Access Control in Data-Intensive Web Applications, in ACM Transactions on the Web (TWEB), May, 2012.