Practical and Privacy-assured Data Indexes for Outsourced Cloud Data

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

***Abstract*—Cloud computing allows individuals and organizations outsource their data to cloud server due to the flexibility and cost savings. However, data privacy is a major concern that hampers the wide adoption of cloud services. Data encryption ensures data content confidentiality and fine-grained data access control prevents unauthorized user from accessing data. An unauthorized user may still be able to infer privacy information from encrypted data by using indexing techniques. In this paper, we investigate the problem of sensitive information leakage caused by orthogonal use of these two kinds of techniques. Based on that, we propose "core attribute"-aware techniques that can ensure privacy of outsourced data. The techniques focus on confidential attribute set of outsourced data. We adopt k-anonymity technique for the attribute indexes to prevent user from inferring privacy from unauthorized data. We formally prove the privacy-preserving guarantee of the proposed mechanism. Our extensive experiments demonstrate the practicality of the proposed mechanism, which has low computation and communication overhead.**

***Keywords*—privacy-assured; data indexes; cloud computing; data outsourcing; k-anonymity**

I. INTRODUCTION

With a number of advantages: location independent data storage, ubiquitous data access and on-demand high quality services, cloud computing has become a popular computing service for both individuals and enterprises, who outsource their data to the cloud for the flexibility and cost savings [1]. On the other hand, data owner outsourcing data to the cloud relinquishes direct control of the system devices, which inevitably brings in potential security problems: Data may contain sensitive information (e.g., health records [2], credit card records [3], etc.), while commercial clouds do not provide data security guarantee. The security concern has been aggravated by recent incidents of Gmail data breach and Amazon outages.

A natural solution for ensuring privacy-preserving to outsourced cloud data uses data encryption. Encryption provides both data integrity as well as content confidentiality, since only the authorized users know the decryption key. The assurance that data confidentiality is protected is desired by the data owner. Meanwhile, in order to make cloud server offer data retrieval service for authorized users, the outsourced encrypted data are usually associated with indexing information which can be exploited by the cloud server that executes data retrieval requests. Several indexing approaches for encrypted data have been proposed for supporting efficient retrieval services (e. g., [4-6]).

While data confidentiality is essential for outsourced data, fine-grained data access control as an orthogonal aspect becomes the focus of concern of several recent papers [7, 10, and 11]. With key derivation technique and user hierarchy grouping, fine-grained data access control can provide different data views to different users. At the user side, one user only needs store little information to derive the access key.

The combined use of the two techniques, however, could cause privacy breach, because the current commercial clouds (e.g., Amazon's EC2 and S3) only offer storage services but don't guarantee strict data access control that separates user information according to user's privilege. As the matter of fact, a user's retrieval pattern is privacy information and should be protected, and hence data owners do not outsource the access matrix to the cloud server. It potentially opens a door for users to infer encrypted data by the visibility of data indexes even if the users can not access the data. The authors of [9] solve this problem by building conflict graph according to the relationship among tuples, and then combine users' access privilege to construct the index system. However, it causes the "multilingual queries" problem, in which a user has to send n time retrieval requests using different keys for one original query, and this is not acceptable.

Similar to [9], in this paper, data are organized in a relational table and indexes are defined as attributes. Meanwhile our proposal can be adapted to generic resource scenes. In this paper, we investigate the problem of privacy-assured data index and we propose a suit of novel techniques to solve this issue. First, we formally prove that the impact of different attributes varies (details given in the Section III). Some attributes play an important role in making a decision, and they are referred to as the core attribute set; while other attributes (referred to as unrelated attributes) do not affect the decision. The disclosure of unrelated attributes does not provide any additional knowledge to an attacker. For the core attributes, we apply k-anonymity to the corresponding indexes. Different from the traditional k-anonymity [12], our scheme does not reduce the description precision of the attribute value. We just generalize the pointing domain of indexes. In addition, to protect outsourced data from statistic attacks, we introduce extra dummy entries in indexes using random tuples such that

978-1-4799-1353-4/13/$31.00 ©2013 IEEE

the retrieval results for each search have the same length. We formally prove the privacy-preserving guarantee of our proposed scheme, and our extensive experiments demonstrate the efficiency of the proposed solution.

The remainder of the paper is organized as follows. Section II describes the system and threat model. In Section III, we present our secure data indexing mechanism and the security analysis. In Section IV, we give the experimental results. Section V overviews related work, followed by the concluding remarks in Section VI.

II. Problem Formulation

A. System Model

Fig. 1 illustrates a high-level architecture for cloud storage/retrieval services, which involves three different entities: the data owner, authorized users, and the cloud server. Here, the data owner may represent an individual or enterprise, which has a relation table r of relation schema $S(X, A, F, d, I)$ to be outsourced to the cloud server, where X denotes object set $\{x_1, x_2, \dots, x_n\}$; A is attribute set $\{a_1, a_2, \dots, a_m\}$, d is the decision attribute; F is the relation set between X and A, $F = \{f_k: X \to V_k\}$, where V_k is the value range of $a_k (a_k \in A)$; and I is the index set $\{I_1, \dots, I_m, I_d\}$, where I_i is the corresponding index for attribute a_i. To keep data content confidential from unauthorized entities, the data owner should encrypt the relation table r before outsourcing it. The encrypted relation is defined as follows.

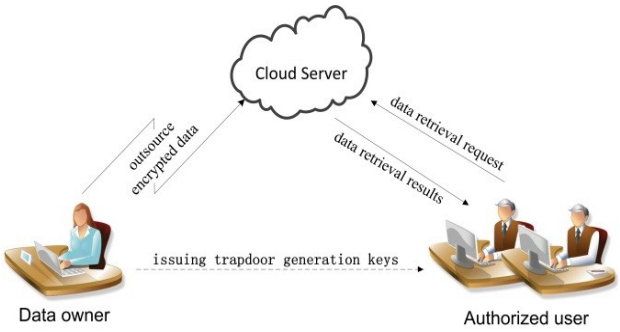

Fig. 1. Architecture of data retrieval over outsourced data

Definition 1 (Encrypted Relation). Let r be a relation of schema $S(X, A, F, d, I)$. r^e is the encrypted relation of r. The corresponding encrypted relation schema is $S^e(Xid, X^e, I^e)$, where Xid is the identifier set for object set X; for $\forall x_i \in r$, $\exists x_i^e \in r^e$; and I^e is the encrypted version of I.

According to Definition 1, we notice that the data encryption granularity is one tuple (record). To initialize the service, we assume that the data owner has assigned the search request keys to authorized users. An authorized user uses his key to generate a data retrieval request, and submits it to the cloud server. The cloud server implements the request over outsourced data and sends back the results.

B. Threat Model

The security threats faced by our system model primarily come from unauthorized parties (e.g., the cloud server, and unauthorized users). We consider them as "honest-but-curious", i.e. the cloud server will follow the protocol but may attempt to derive additional information from the received retrieval requests. On the user side, a user with limited data access retrieves data followed by the mechanism that combines encryption and fine-grained data access control. There are financial incentives for unauthorized users to inferring additional information from data stored in the cloud.

For instance, Table I is a data set for fitting contact lenses, which has eliminated single attribute identifier, while Fig. 3 illustrates the knowledge of user Adam for example, providing his view over the encrypted relation. For simplify expression, we use upper-latin letters to denote encrypted records. Here, grey cells in the relation denote the records that Adam is authorized to access. It's easy to see that Adam can infer the content of records he is not authorized to access. When direct indexes are used, the authorized information exposes all the cells having the same plaintext values as the ones that the user knows, since each plaintext value is always represented by the same index value. Here, light red cells represent the information in Table I infered by Adam exploiting such index values.

TABLE I. Payoff Matrix of Data Audit Game

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-prescrip {myope, hyperope}
a_3: tear-prod-rate {reduced, normal, more}
a_4: astigmatism {no, yes}
d: contact-lenses {soft, hard, none}

	X^e	I_{a1}	I_{a2}	I_{a3}	I_{a4}	I_d
x_1	A	i(young)	i(myope)	i(normal)	i(yes)	i(hard)
x_2	B	i(old)	i(myope)	i(reduced)	i(no)	i(hard)
x_3	A	i(young)	i(myope)	i(normal)	i(yes)	i(hard)
x_4	C	i(young)	i(hyperope)	i(reduced)	i(no)	i(none)
x_5	C	i(young)	i(hyperope)	i(reduced)	i(no)	i(none)
x_6	D	i(young)	i(hyperope)	i(more)	i(no)	i(soft)
x_7	C	i(young)	i(hyperope)	i(reduced)	i(no)	i(none)
x_8	D	i(young)	i(hyperope)	i(more)	i(no)	i(soft)

Fig. 2. The infered knowledge of user Adam according to the authorized information

III. The Secure and Efficient Indexing Mechanism

In this section, we first discuss how to find core attributes, which play an important role in making decision over attributes. Using the core attributes could minimize the security overhead. Based on the core attributes, we apply the k-anonymity and

data balance techniques over the attribute index values according to the user access matrix.

A. The Core Attributes

The process of seeking core attributes is to find attributes that have the most important factors for decision making, while other attributes contain nothing for decision making. A quasi-identifier (QID) is an attribute set that can identify one (or a class of) object. First, we present a few definitions and theorems related with core attributes.

Definition 2 (Equivalence Partition). On the schema $S(X, A, F, d)$, $[\mathrm{x_i}]_\mathrm{A}$ is an equivalence partition on X, $iff: \forall x_i, x_j \in [x_i]_A,\ f_k(x_i) = f_k(x_j)$ $(\forall a_k \in A)$. Here, we use $\mathrm{R_A}$ to express the equivalence partition set over attribute set A, and the element of $\mathrm{R_A}$ satisfy: for $\forall[\mathrm{x_i}]_\mathrm{A} \neq [\mathrm{x_j}]_\mathrm{A}$, $[\mathrm{x_i}]_\mathrm{A} \cap [\mathrm{x_j}]_\mathrm{A} = \phi$ and $\cup_{\mathrm{k=1}}^{\mathrm{n}}[\mathrm{x_i}]_\mathrm{A} = \mathrm{X}$.

Definition 3 (Core Attributes). On the schema $S(X, A, F, d)$, for attribute set A, C, where $C \subseteq A$. $iff\ R_C = R_A$, and for $\forall Z \subset C$,$R_Z \neq R_A$, we call attribute set C as a core attribute over S.

According to Definition 2 and 3, we have Theorem 1.

Theorem 1. On the schema $S(X, A, F, d)$, the core attributes C is the quasi-identifier, while $\overline{C}$ contain nothing from the decision attribute d.

Proof: First let's prove core attributes C is the quasi-identifier. For any $a \in A$, $R_{A-\{a\}} \neq R_A$, there does not exist proper subset B of A that makes $R_B = R_A$, i.e. $C = A$; there exist $a \in A$, making $R_{A-\{a\}} = R_A$. Let $C_1 = A - \{a\}$, if for any $c_1 \in C$, making $R_{C_1-\{c_1\}} \neq R_A$, it means C_1 is the core attribute set. Otherwise, there exists $c_1 \in C$, making $R_{C_1-\{c_1\}} = R_A$, let $C_2 = C_1 - \{c_1\}$, repeat the process above. A is limit set, it can find $C \subseteq A$ in limit steps, making $R_C = R_A$. Thus, the core attribute set exists surely. Combing the Definition 2 and $R_C = R_A$, the core attributes C is QID.

Let C be the core attribute set, suppose $\exists a \in \overline{C}$, i.e., a is non-core attribute, making $R_{A-\{a\}} \neq R_A$, which is a contradiction with Definition 3. That means a contains nothing from the decision attribute d. □

In Algorithm 1, we provide a formal description of finding the core attribute set. The algorithm uses two concepts: discernibility matrix and attribute hypergraph, which are defined below.

Definition 4 (Discernibility Matrix). On the schema $S(X, A, F, d)$, $R_A = \{[x_i]_A | x_i \in X\}$, $M\left([x_i]_A, [x_j]_A\right) = \begin{cases} a_k \in A \mid f_k(x_i) \neq f_k(x_j), [x_i]_A \neq [x_j]_A \ and\ [x_i]_d \neq [x_j]_d \\ \emptyset, others \end{cases}$,

where $M\left([x_i]_A, [x_j]_A\right)$ is the attribute discernibility set about $[x_i]_A$and$[x_j]_A$. $\mathbf{M} = (M\left([x_i]_A, [x_j]_A\right) \mid [x_i]_A, [x_j]_A \in R_A)$ is the discernibility matrix over X.

Definition 5 (Attribute Hypergraph). Attribute hypergraph $G_A = (V, HE)$, where $V = \{a_1, \ldots, a_m\}$ denotes vertex set which is consisted by all the attributes in A, $HE = \{M_k | M_k \in \mathbf{M}\}$ denotes hyperedge. One hyperedge represents one item in the discernibility matrix.

Now we briefly explain Algorithm 1. First we construct the equivalence class of the attribute set A and d, respectively. Then we use the attribute hypergraph reducing method to identify the core attributes of different equivalence classes based on identification matrix. Here, the steps of hypergraph reducing are: 1) choosing one hyperedge that is contained by the most hyperedges; 2) deleting all hyperedges containing the chosen one; 3) repeating steps 1) and 2) until there is no hyperedge. For the example in Table I, 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 **M** that can identify various attributes of different equivalence classes.

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

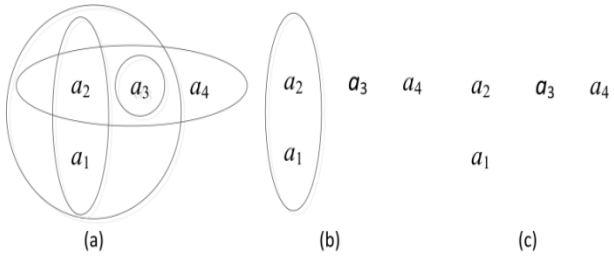

Fig. 3. Attribute hypergraph of Table I

We propose an attribute hypergraph digestion method to obtain the QID, as illustrated in Fig. 3(a). The rules of the hypergraph digestion are given below: (1) finding a hyperedge that is contained by the most hyperedges; (2) deleting all hyperedges containing the chosen one; (3) repeating step (1) until there is no hyperedge. According to the hypergraph digestion rules, first, HE(a_3) is chosen, deleting all hyperedges containing attribute a_3, and we have Fig. 3(b). Second, choosing HE(a_1, a_2), and deleting it, and we have Fig. 3(c). Now, we can get $C = a_3 \times (a_1, a_2)$. That is, $c_1 = \{a_1, a_3\}, c_2 = \{a_2, a_3\}$.

Algorithm 1 Finding Core Attributes

INPUT:
$HE\{\sum_{i=1}^{k}\sum_{j=i+1}^{k} D[i][j]\}$
OUTPUT:
C
MAIN
1: $c \leftarrow 0$, $count \leftarrow 0$, $e \leftarrow 0$, $C \leftarrow \emptyset$, $Edge \leftarrow \emptyset$
2: **while** $HE \neq \emptyset$ **do**
3: **for each** HE_k **do**
4: $Temp \leftarrow \emptyset$
5: **for** $i \leftarrow 0$ **to** $|HE|$ **do**
6: **if** $HE_k \subset HE_i$ **then**
7: $count \leftarrow count + 1$

8: $Temp \leftarrow Temp \cup HE_i$
9: **if** c<count **then**
10: $c \leftarrow count$, $e \leftarrow k$, $Edge \leftarrow Temp$
11: $HE \leftarrow HE - Edge$
12: $C \leftarrow C \times HE_e$ /* "×" denotes Cartesian product */
13: **return** C

B. Attribute Partition

Note that, according to Definition 3, the core attribute set is not unique. Next, our work is to partition the core attributes into two parts, making the attributes E satisfy: for $\forall C, C \cap E \neq C$. The maximal attribute fragmentation problem (Max-AF) is NP-hard. We present a heuristic Algorithm 2 for solving it.

Theorem 2. The Max-AF problem is NP-hard.

Proof: The proof is a reduction from the NP-hard problem of vertex covering, formulated as follows: given a graph $G = (V, E)$, determing a minimum vertex covering of G which is a subset $V' \subseteq V$, such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$ or both. That is, each vertex covers its incident edges, and a vertex cover for G is a set of vertices that covers all the edges in E.

Given a schema $S'(X, C, F, d)$, where C denotes a set of core attribute sets, the correspondence between the Max-AF problem and the vertex covering problem can be defined as follows. For $\forall C_i \in C$, $C_i = \{a_{i1}, \dots, a_{in}\}$, and C_i is a complete subgraph. $C' \subseteq C$, such that if $(a_{ij}, a_{ik}) \in C_i$, then $a_{ij} \in C'$ or $a_{ik} \in C'$ or both. The Max-AF $C'' = C - C'$. Hence, the Max-AF problem is NP-hard. □

Algorithm 2 Max-AF
INPUT:
S'
OUTPUT:
C''
1: $C' \leftarrow \emptyset$
2: $E' \leftarrow E[S']$
3: **while** $E' \neq \emptyset$ **do**
4: let (a_{ij}, a_{ik}) be an arbitrary edge of E'
5: $C' \leftarrow C' \cup (a_{ij}, a_{ik})$
6: remove edges incident on either a_{ij} or a_{ik}
7: $C'' \leftarrow C - C'$
8: **return** C''

Discussions. Algorithm 2 is a heuristic algorithm for attribute partition, which ensures no QID exists in the fragmentation. According to Theorem 2, for covering every edge in S', any vertex covering must contain at least one vertex of every edge, while line 5 of Algorithm 2 add one edge (two vertexes) to the essential attributes. Therefore, the maximal deviation between C'' and C^* is $\frac{C'}{2}$.

C. The k-Anonymity Mechanism for Index Pointers

By the essential attributes location process above, we minimize the processed attribute set C'. After that, we implement k-anonymity and search balance operation for index pointers corresponding to C'. Our scheme is different from the traditional k-anonymity technique, and it has several features: *1) we do not implement generalization operation on data, thus the accuracy of data is not reduced; 2) the value of the threshold "k" does not depend on expert experiences, the value is dynamically set based on user resource access distribution and the attribute index balance tree.*

Definition 6 (User Access Matrix). User access matrix **D** is a table that describes the relationship between authorized users $U = \{u_1, \dots, u_n\}$ and the corresponding resources allowed to access - $X_U = \{X_{u_1}, \dots, X_{u_n}\}$, where $\mathbf{D} = \left(D(u_i, X_{u_i}) | u_i \in U, X_{u_i} \in X_U\right)$.

Definition 7 (User *Resource* Access *Distribution*). $S(X', a_k, F, d)$ is a relation schema, where $X' = \bigcup_{u_1}^{u_n} X_{u_i}$, $a_k \in C'$. R_{a_k} is user resource access distribution over attribute a_k, where $R_{a_k} = \{[x_i]_{a_k} \mid x_i \in X'\}$.

Definition 8 (Index Balance Tree). The index balance tree is a statistic structure. The root node represents the metadata of the essential attributes. Each internal node represents a specialized value (or a generalized value interval) of its parent node. Each leaf node can be expressed as a tuple (Xid, *count*), where Xid means that index I_{value} can retrieve the data ID, *count* is a statistic that records the number of parent nodes having the value in the dataset. For any two leaf nodes, $|\Delta count| = 1$.

According to Definition 7, the data owner can build indexes over every attribute before outsourcing. Note that, attribute index is not related with X, but depending one user resource access distribution. To prevent unauthorized parties from inferring new knowledge, we propose a data retrieval balance strategy based on the index balance tree. Algorithm 3 describes the index-creating strategy for an arbitrary attribute a_t.

Algorithm 3 Index_Creating
INPUT:
Index_Balance_Tree root$[a_t]$
Object R_{a_t}
Object X'
security_factor δ
OUTPUT:
$I_{a_t}^e$
MAIN
1: Scanning R_{a_t} to build Index_Blacnce_Tree root$[a_t]$
2: Setting k
1) $w = Max(count)$, w denotes the maximum *count* in all leaf nodes
2) $k = \lceil w \times (1 + \delta) \rceil$
3: Balancing root$[a_t]$
For any leaf node l_i that its $count < w$, randomly select $x_j \in X'$ and $x_j \notin l_i[X_{id}]$, insert x_j into l_i, until all leaf node's $count = w$.
4: Creating index for every leaf node l_i and encrypting it
5: return $I_{a_t}^e$

Theorem 3. Algorithm 3 guarantees that the rate of referring new knowledge of unauthorized users is not higher than L.

Proof: Based on the threat model, the proof consists of two parts. For cloud server, all data and the corresponding indexes

are encrypted. Suppose the encryption is secure, then the cloud server doesn't know the data and the semantic information. Due to the data retrieval balance process that makes all records as k-anonymity, the cloud server cannot infer statistic information from user's queries. Specifically, let w be the maximum number of records in all class records, δ be the confusion factor, w_ρ is the non-balanced data set in user's one time data retrieval, and w_u is the authorized data corresponding with w_ρ. Therefore, for any specific retrieval, the sensitive information leak probability $L = \frac{w_\rho - w_u}{w(1+\delta) - w_u}$. For unauthorized user, the condition is similar with the former. That is omitted here. In addition, the joining of a new user or permission change of an existing user likely triggers the execution of Algorithm 3, since the trigger condition of Algorithm 3 is: $|\Delta count|>1$, where $|\Delta count|$ is the difference value of the contained data records among any two leaf nodes. Thus, for data requests with the same query condition, the result to different users may be different because the requests may be processed at different trigger cycles and may be added with different noises. This can further reduce the leakage of privacy information. □

IV. Experimental Results

To verify the effectiveness and practicality of the proposed mechanism, we conduct a series of experiments on our cloud testbed. To simulate a real database, we generate a test dataset following the TPC-H benchmark specifications [8]. We build our testbed as follows: there are three nodes, and each one has 8-core 2.93 GHz Intel Xeon CPU, 24 GB memory, 500 GB local disk and installed with Linux 2.6.18. Our proposed algorithms are implemented in Python. For each configuration, we run experiments multiple times and report the average.

First we evaluate the performance of Algorithm 1 - finding core attributes. We choose different record collections from the IDE data set [13], where the number of a collection ranges from 10,000 to 100,000. Fig. 4 shows that the time of finding core attributes increases linearly with the number of data. Though the time is in the order of seconds, we should note that the core attributes search process is just a one-time cost and can be conducted off-line. Fig. 3 plots the time of attribute partition, which is a linear function, and the attribute count varies from 100 to 1,000, in steps of 100. Fig. 5 matches Algorithm 2's running time: $O(V + E)$.

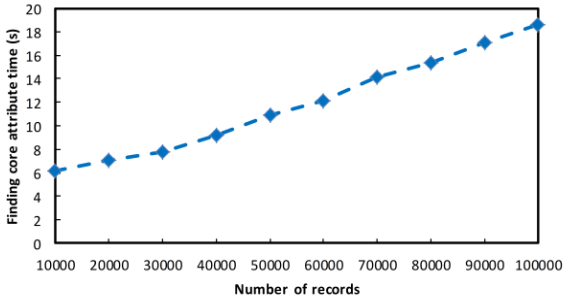

Fig. 4. Time of finding core attributes

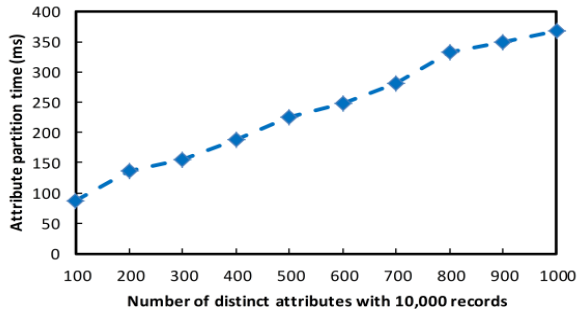

Fig. 5. Time for attribute partition

Fig. 6 and Fig. 7 show the impact of the number of attributes on the efficiency of creating index and the auxiliary storage space, respectively. As the number of attributes grows, the time overhead increases, but not at the same rate. This is consistent with Algorithm 3. The time of building attribute index depends on the specific attribute value distribution in the entire spatial range due to the data balance operation. There is some similarity between the time overhead and storage overhead when number of attributes increases. We use security index provided by literature [9] as the baseline. The result shows that our index creating time is faster than the literature [9]'s while the space overhead is smaller.

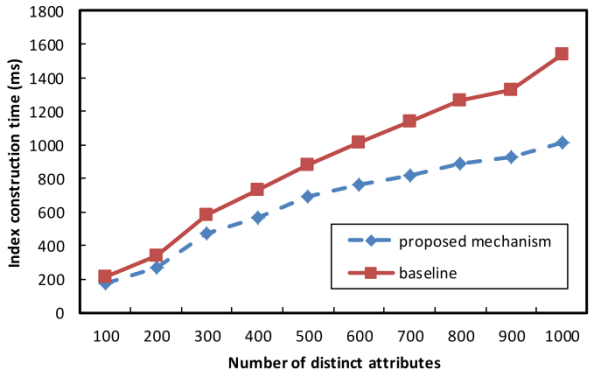

Fig. 6. Time of creating index

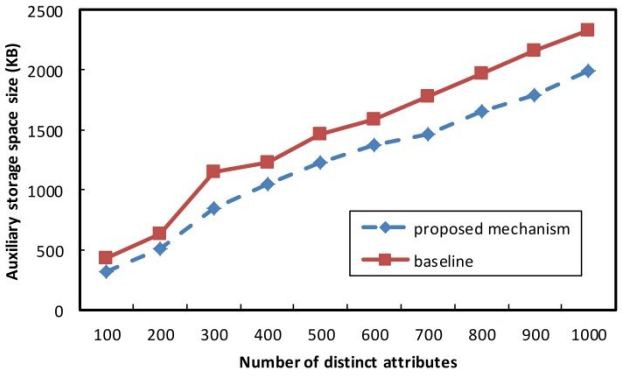

Fig. 7. Auxiliary storage overhead

Fig. 8 plots the size of retrieval set returned by the cloud for different amount of data. We include a general data retrieval method as the baseline. Since the baseline scheme does not

need to return any noise data, its result-set is smaller than that of our mechanism. We notice that the result-set size of our mechanism is very close to that of the baseline scheme. Our mechanism supports secure search without privacy breaches, hence our mechanism is still quite efficient.

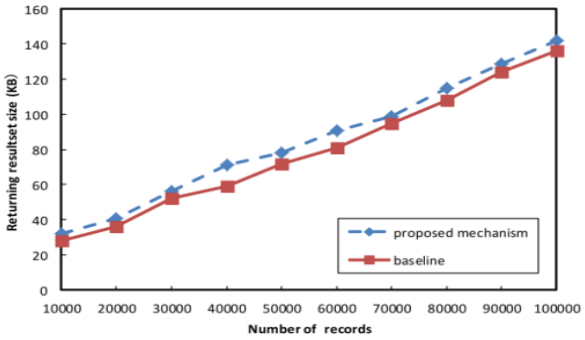

Fig. 8. Evaluation of returning result-set size

V. Related Work

Several research efforts have been performed in the context of data outsourcing [14]. In [5] the authors propose bucket-based index techniques which map multiple plaintext values to the same index value, thus generating collisions that prevent frequency-based attacks. However， the direct and bucket-based indexing techniques do not support range queries. In [6] the authors propose a flattened indexing method that exploits B-trees for supporting both equality and range queries. All these solutions provide support for the evaluation of queries over encrypted data, but do not take access control restrictions into account. In [15], the authors introduce the definition of B-tree indexes, one for each different access control list (acl) in the system. To guarantee confidentiality, the B-tree index of each acl is encrypted with a key that only the users in the acl know. This solution suffers from a high client side overhead in query evaluation, since the user submitting the query must visit all the B-trees associated with the acls to which she belongs. The authors of [9] solve this problem by building conflict graph according to the relationship among tuples, and then combine users' access privilege to construct the index system. However, it causes the "multilingual queries" problem. Our work is complementary to these proposals since it addresses the problem of ensuring that users cannot withdraw any inferences on the data they are not authorized to access by observing indexes associated with the data.

VI. Concluding Remarks

In this paper, we studied the problem of secure and efficient data retrieval over outsourced cloud data. For effective searching outsourced data without jeopardizing data privacy, we first exploited "core attribute"-aware technique, and then adopted k-anonymity technique for indexes of these attributes to prevent any entity from inferring unauthorized data. We formally proved the privacy-preserving guarantee and the correctness of the proposed mechanism under rigorous security treatment. The experiment results further demonstrated the validity and practicality of the proposed mechanism.

Acknowledgment

This work is supported 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), and by the US National Science Foundation under grants CNS-0963578, CNS-1022552 and CNS-1065444.

References

[1] M. Armbrust et al., “A view of cloud computing,” Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[2] Health Insurance Portability and Accountability Act. http://www. hhs. gov/ocr/privacy/.

[3] Payment card industry (PCI) data security standard. https://www. pcisecuritystandards.org/documents/pci dss v2.pdf.

[4] R. Agrawal et al.. Order preserving encryption for numeric data. In Proc. of SIGMOD 2004, Paris, France, June 2004.

[5] E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In Proc. of CCS 2003, Washington, DC, USA, October 2003.

[6] H. Wang and L. V. S. Lakshmanan. Efficient secure query evaluation over encrypted XML databases. In Proc. of VLDB 2006, Seoul, Korea, September 2006.

[7] S. De Capitani di Vimercati et al.. Overencryption: Management of Access Control Evolution on Outsourced Data.VLDB: 123-134. Vienna, Austria, September 2007.

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

[9] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Private Data Indexes for Selective Access to Outsourced Data. 10th Workshop on Privacy in the Electronic Society (WPES):69-80.October 2011.

[10] 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.

[11] E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In Proc. of CCS 2003, Washington, DC, USA, October 2003.

[12] Kisilevich Slava et al.. Efficient multidimensional suppression for k-anonymity. IEEE Transactions on Knowledge and Data Engineering. 334-347, March 2010.

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

[14] P. Samarati and S. De Capitani di Vimercati. Data protection in outsourcing scenarios: Issues and directions. In Proc. of ASIACCS 2010, Beijing, China, April 2010.

[15] E. Shmueli, R. Waisenberg, Y. Elovici, and E. Gudes. Designing secure indexes for encrypted databases. In Proc. of DBSec 2005, Storrs, CT, USA, August 2005.