Enforcing Data Privacy and User Privacy over Outsourced Database Service

Yonghong Yu
School of Information engineering of Anhui University of Finance & Economics, Bengbu, China
Email: express_yu@163.com

Wenyang Bai
State key Laboratory for Novel Software Technology of Nanjing University, Nanjing, China
Email: wyb@nju.edu.cn

***Abstract*—Privacy requirements have an increasing impact on the real-world applications. Technical considerations and many significant commercial and legal regulations demand that privacy guarantees be provided whenever sensitive information is stored, processed, or communicated to external parties. It is therefore crucial to design solutions able to respond to this demand with a clear integration strategy for existing applications and a consideration of the performance impact of the protection measures. In this paper, we propose a solution to enforce data privacy and user privacy over outsourced database services. The approach starts from a flexible definition of privacy constraints on a relational schema, applies encryption on information in a parsimonious way and mostly relies on attribute partition to protect sensitive information. Based on the approximation algorithm for the minimal encryption attribute partition with quasi-identifier detection, the approach allow storing the outsourced data on a single database server and minimizing the amount of data represented in encrypted format. Meanwhile，by applying cryptographic technology on the auxiliary random server protocol, the approach can solve the problem of private information retrieval to protect user privacy. The theoretical analysis and experimental results show that our new model can provide efficient data privacy protection and query processing, efficient in computational complexity and dose not increase the cost of communication complexity of user privacy protection.**

***Index Terms*—outsourced database services, data privacy, user privacy, attribute partition, encryption**

I. INTRODUCTION

Privacy requirements have an increasing impact on the real-world applications. Technical considerations and many significant commercial and legal regulations demand that privacy guarantees be provided whenever sensitive information is stored, processed, or communicated to external parties. It is therefore crucial to design solutions able to respond to this demand with a clear integration strategy for existing applications and a consideration of the performance impact of the protection measures. As a recent manifestation of this trend, there has been growing interest in outsourcing database services (ODBS) in both the commercial world and research community.

Consider the following real-life scenario: A hospital M has a patient database containing pattern about various diseases. M stores these disease patterns on an un-trusted external database server DB and allows a client A to access the database to get information with respect to A disease. This scenario poses several security issues as follows:

1. Because DB is an un-trusted external server, M has to protect its data contents from being accessed and analyzed by DB and other intruders. This security issue is referred to as data confidentiality.
2. Whenever a accesses DB, he does not want M or even DB operators to know exactly what he is concerned about, both the query and its result. This security issue is referred to as user privacy.
3. Client A is not allowed to get more information other than what he is querying on DB. This is an important aspect in the real-world scenarios because A may have to pay for what he can get from DB and M does not allow him to get more that what he has paid for even A does not want to get what he does not need from DB and M. This security issue is referred to as data privacy.

In general, protecting outsourced data mainly relates to the three security issues as mentioned above. The need of data confidentiality, data or user privacy depends on particular scenarios in the outsourced database services model and this must be considered carefully.

A significant amount of research has recently been dedicated to the study of the outsourced data paradigm. Most of this research has assumed the data to be entirely encrypted, focusing on the design of techniques for the efficient execution of queries. One of the first proposals towards the solution of this problem is presented in [1,2,3], where data is encrypted on the client side before being stored in the un-trusted external server. In order to answer all database queries, the client need to fetch the entire database from server, decrypt it, and execute the query on this decrypted database. Of course, such an approach is far too expensive to be practical. With trusted computing [4], a tamper-proof secure co-processor could be installed on the server side, which allows executing a function while hiding the function from the server. However, such a scheme could involve significant

doi:10.4304/jsw.6.3.404-412

computational overhead due to repeated encryption and decryption at the row level. Reference [5] proposed secure multi-party computation techniques to ensure database safety, but the excessive communication overhead involved makes this approach even more inefficient than the trivial scheme in which the client fetches the entire database from the server. These approaches can protect the data from outsiders as well as the server, but they introduce difficulties in the querying process over encrypted data while still maintaining an acceptable query processing performance. In fact, a server that is secure under formal crypto-graphic notions can be proved to be hopelessly inefficient for data processing [6]. Reference [7] proposed a distributed architecture for outsourced database services which is different from data encryption techniques mentioned above. The key idea is to allow the client to partition its data across two or more logically independent database systems that cannot communicate with each other. Due to not involving repeated encryption and decryption, the use of such a distributed database for obtaining outsourced database services offers many advantages, such as un-trusted service providers, provable privacy and efficient queries. While presenting an interesting idea, it suffers from the assumption of the complete absence of communication among the servers. This assumption is clearly too strong and difficult to enforce in real environments. However, all of the above described data privacy/confidentiality solutions fail to satisfy the user privacy objectives.

To satisfy the user privacy requirement, private information retrieval protocols (PIR) are employed. In principle, the PIR protocol allows a client to access a database without revealing to the server both the query and the returned result. Protocols for PIR schemes [8,9,10] are all based on the idea of using multiple copies of the database that are not allowed to communicate with each other. This allows the user to ask different questions from different copies of the database and combine the responses to get an answer to his query, without revealing his original query to any single database. Reference [11] uses a single database, but guarantees only computational privacy under the assumption that distinguishing quadratic residues from non-residues modulo composites is intractable. In fact, it can be shown that using a single database makes it impossible to achieve information theoretic privacy with sub-linear communication complexity. Reference [12] still uses a multiple database model, but the multiple databases are not copies of the original principal database. Rather, they are auxiliary database provided by WWW servers for user privacy purpose. It performs the initial setup computation which will be proportional to the size of the principal database, then, the principal database only needs to make *O(1)* on line computation to answer queries of users, while all the extra computation required on-line for privacy are done by the server, but the servers and the database need to engage in a re-setup after every m queries. Nevertheless, all of the above described PIR protocols cannot satisfy the data confidentiality objectives.

Note that the exposure of a set of attribute values corresponding to a row may result in a privacy violation, while the exposure of only some subset of it may be harmless. For example, revealing an individual's name and his salary may be a serious privacy violation. However, exposing the name alone, or exposing the salary alone, may not be a big deal. Hence, there is no need to encrypt both name and salary if there are alternative ways to protect the association between them.

In this paper, we propose a novel approach to enabling data privacy and data confidentiality protection in an outsourced database service. The key idea is to combine attribute partition and encryption that allows storing data as only one principal database on a single database server. Attribute partition and encryption provide protection of data in storage, or when disseminated, ensuring no sensitive information is disclosed neither directly nor indirectly. The original relation schema R is split into two attribute partition. The first partition consists of attributes needed be encrypted to achieve data privacy/confidentiality, all these attributes are encrypted into a single encrypted attribute. The second partition consists of attributes needed not be encrypted and can be stored in plaintext. Then, we need to detect whether the combinations of distinct values of attributes in the second partition could identify individual row with sufficiently high probability. If there is such a set of attributes (quasi-identifier), we need to adjust the two attribute partition to ensure data privacy. Last, combing the single encrypted attribute and others attributes belong to the second partition, we create a new relation schema R', and provide an approximation algorithm to minimize the amount of data represented only in encrypted format, therefore allowing for efficient query execution. With this design, the data can be outsourced and stored on an un-trusted server, typically obtaining lower costs, greater availability and more efficient distributed access. The consequence of this design choice is that to evaluate a query, it is sufficient to access a single relation, thus avoiding join operations, which are quite expensive. Last, the advantage of having only part of data encrypted is that all the queries that do not access the confidential information will be managed more efficiently and securely.

Meanwhile, based on the combination of auxiliary random servers and crypt-graphic technology, this paper introduces a new paradigm for private information retrieval, which allows for information theoretic privacy without replication of database. Since it is not possible to use a single database and achieve sub-linear communication complexity information theoretic results, this new model utilize auxiliary random servers to simulate multiple database model. Instead of replicating the principal database as in the underlying scheme, every copy is replaced by t+1 random servers whose contents are n pairs of keys $<U_i,V_i>$, where U_i is the result of encrypting sensitive attributes of each row, and V_i is the result of encrypting encryption function. An auxiliary server can not obtain information about the data.

The rest of this paper is organized as follows. Section 2 presents the general architecture of outsourced database

services, describing the space of techniques available for partitioning data and the trade-offs involved. Section 3 describes how to define privacy constraints in outsourced database services. In section 4, we describe how to obtain data privacy/confidentiality based on attribute partition and encryption, introduce an approximation algorithm for minimal encrypted attribute partition. Section 5 describes how to obtain user privacy based on cryptographic technology on the auxiliary random server protocol. Finally, section 6 draws some conclusions.

II. General Architecture

The general architecture of an outsourced database service, as illustrated in Fig. 1, consists of a trusted client front-end as well as many auxiliary random servers that provide a database service.

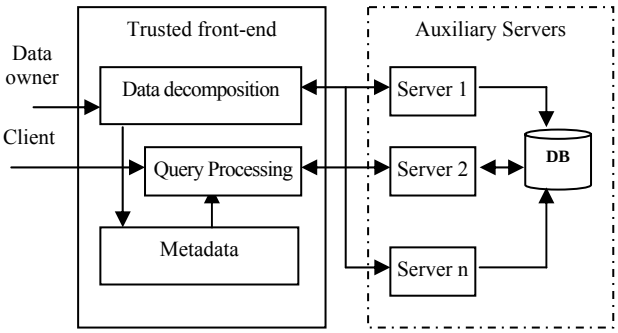

Figure 1. The System Architecture of Outsourced Database services

These auxiliary servers provide reliable content storage, data management and user privacy protection but are not trusted by the trusted client front-end to preserve data privacy. The principal database after engaging the services of some servers for the purpose of offering private and secure access to users, performs an initial setup computation with auxiliary servers. The servers are then ready to assist users in retrieving information from the principle database efficiently and privately during the online stage.

The trusted front-end provides three pieces of functionality:

1. Data Decomposition. By running approximation algorithm for attribute partition and encryption, the trusted front-end can first partition original relation in encrypted attributes and non-encrypted attributes, then it create a new relation and populate the new relation, and stores only the encrypted data on $t+1$auxiliary random servers, meanwhile, it stores schema metadata of relation decomposition at the trusted front-end.

2. Query Processing. The queries received by the trusted front-end need to be translated into appropriate SQL sub-queries to be sent to the servers, and the results are gathered and post-processed before being returned in a suitable form to the user application.

3. Metadata Repository. The schema metadata of relation decomposition, the generalization involved in encryption processing, and the statistics for query optimization are stored in metadata repository of the trusted front-end. All data in metadata repository can be used to achieve efficient database queries.

Our work assumes that access to data is realized by an application that includes trusted auxiliary random servers, which are invoked every time there is need to access sensitive information. By contrast, the DBMS needs not be trusted, since accessing encrypted information in a single relation does not expose to any privacy breach. The front-end module is also assumed to be completely trusted and secure. Preventing the front-end breaches is a traditional security problem unrelated to privacy-preserving data storage, and we do not concern ourselves with this problem here. Thus, we assume that every server be honest, it does not act maliciously by providing erroneous service to the front-end or by altering the stored data. We also assume that the front-end maintains separate, permanent channels of communication to every server.

III. Defining The Privacy Constraints

We consider a scenario where the data to be protected are represented with a single relation r over a relation schema $R(a_1, a_2, \cdots\cdots, a_n)$, and we use R to denote either the relation schema R or the set of attributes in R. Our privacy requirements are specified as a set of privacy constraints C, expressed on the schema of relation R. Each privacy constraint c is represented by a subset of the attributes of R.

Definition 1 (Privacy constraint): Given a relational schema $R(a_1, a_2, \cdots\cdots, a_n)$, a privacy constrain c over R is a singleton set {a}, stating that the values of the attributes are sensitive, or a subset of attributes in R, stating that the association between values of the given attributes is sensitive.

We illustrate this definition by an example. Consider a company desiring to store relation R consisting of the following attributes of employees: SSN, Name, Date of Birth(DoB), Zip, Job, Salary, Des. Fig. 2 illustrates an example of relation employees.

SSN	Name	DoB	Zip	Job	Salary	Des
340-34-01	Zhangsan	18/02/78	233011	Worker	123.4	Aaa
342-45-23	Lisi	25/10/76	233041	Salesman	234.5	Bbb
340-34-12	Wangwu	15/08/77	233051	Manager	456.7	Ccc
340-34-07	Zhaoliu	12/06/78	233061	Worker	321.4	Ddd
……	……	……	……	……	……	……

Figure 2. An example of plaintext relation

The company may have the following considerations about privacy:

1. *SSN* is sensitive information subject to misuse, therefore the *SSN* attribute form a singleton privacy constraint and cannot be stored in the clear under any circumstances.

2. *Salary* and *Job* are considered private details of individuals, and so cannot be stored together with an individual's name in the clear. Therefore, the sets {Name, Salary} and {Name, Job} are all privacy constraints.

3. The set of attributes {DoB, Zip, Salary} can help identify a person in conjunction with other publicly available data. Therefore, the set {DoB, Zip, Salary} is also a privacy constraint.

4. In order to prevent an adversary from learning sensitive association, for example, between job and salary. Therefore, we need to add privacy constraint {Job, Salary}

In general, we are interested in enforcing a set of well defined privacy constraints, formally defined as follows.

Definition 2 (Well defined privacy constraints): Given a relation schema $R(a_1, a_2, \cdots\cdots, a_n)$, a set of privacy constraints $C = \{c_1, \cdots, c_m\}$ is said well defined over *R*, iff $\forall c_i \in C(1 \le i \le m)$, $\forall c_i, c_j \in C, i \ne j, c_i \not\subset c_j \wedge c_j \not\subset c_i$.

According to the definition, a set of privacy constraints *C* over *R* cannot contain a constraint that is a subset of another constraint. The Rationale behind this property is that, whenever there are two constraints c_i, c_j and c_i is a subset of c_j, the satisfaction of constraint c_i implies the satisfaction of constraint c_j, and therefore c_j is redundant.

The Company may have the following privacy constraints defined:

c_1= {SSN}
c_2= {Name, Job}
c_3= {Name, Salary}
c_4= {DoB, Zip, Salary}
c_5= {Job, Salary}

Thus, the set of well defined privacy constraints is {{SSN}, {Name, Job}, {Name, Salary}, {DoB, Zip, Salary}, {Job, Salary}}

Note that the association of employee's SSN and name is sensitive and should be protected. However, such a privacy constraint is not specified since it is redundant, given that SSN as an individual attribute sensitive.

IV. Obtaining Data Privacy/ Confidentiality via Attribute Partition and Encryption

The approach to satisfy privacy constraints is based on the use of two techniques: encryption and partition. Consistently with how the privacy constraints are specified, encryption applies at the attribute level, that is, encrypting an attribute means encrypting all its values. Partition, like encryption, applies at the attribute level, that is, partition means splitting sets of attributes so that they are not visible together without access to the encryption key. It is straightforward to see that singleton constraints can be solved only by encryption. By contrast, an association constraint can be solved by encrypting any of the attributes involved in the constraint or partition the attributes involved in the constraint so that they are not visible together.

A. Attribute Partition and Encryption

Given a set of privacy constraints over relation *R*, our goal is to split *R* into two party F_1 and F_2, in such a way that all sensitive data and associations are protected. F_1 includes all encrypted attributes and F_2 includes all non-encrypted attributes, and the partition should satisfy two important requirements:1)all attributes in R should appear in at least one party to avoid loss of information; 2)the privacy constraints should be properly protected, meaning that from the partition stored at the external server should not be possible to reconstruct the content of the original relation *R*. these two requirements are formally captured by the following definition of correct attribute partition.

Definition 3 (Correct partition): Let $R(a_1, a_2, \cdots; a_n)$ be a relation schema, $C=\{c_1, \cdots; c_m\}$ be a set of well defined privacy constraints over *R*, and $F = \{F_1, F_2\}$ be a attribute partition for *R*, where F_1 includes all encrypted attributes and F_2 includes all non-encrypted attributes and $F_1 \bigcap F_2 = \phi$. *F* is a correct partition for *R*, with respect to *C*, iff: 1) $F_1 \bigcup F_2 = R$ and 2) $\forall c \in C, c \not\subseteq F_2$.

The first condition requires every attribute in the schema of the original relation *R* to be represented in at least a party. The second condition requires the party stored at the external server in clear to not be a superset of any privacy constraint. Note that since F_1 stores all encrypted attributes, it can contain sensitive data and association constraints. For instance, given the set of well defined privacy constraints {{SSN}, {Name, Job}, {Name, Salary}, {DoB, Zip, Salary}, {Job, Salary}}, partition F={{SSN, Name, Salary}, {DoB, Job, Zip, Des}} is a correct partition for employees.

At the physical level, a decomposition of relation *R* translates to a combination of attribute partition and encryption. Each partition $F = \{F_1, F_2\}$ is mapped into a physical partition containing all the attributes of *R*. All attributes in F_1 are encrypted and all attributes in F_2 appear in the clear. The reason for keeping all the original attributes in physical partition is to guarantee that any query can be executed by querying a single relation and can avoid the expensive join operation. For the sake of simplicity and efficiency, we assume that all attributes in F_1 are encrypted all together. Physical partition is defined as follow.

Definition 4 (Physical partition): Let *R* be a relation schema, and $F = \{F_1, F_2\}$ a decomposition of *R*. The physical partition of *R* is a relation schema *R'(enc, $a_1, \ldots, a_n$)*, where *enc* represents the encryption of all the attributes of *R* that belong to F_1, attributes $a_1, \ldots, a_n$ belong to F_2.

At the level of instance, given a partition $F = \{F_1, F_2\}$ and a relation r over schema *R*, the physical partition *R'* is such that each plaintext tuple $t \in r$ is mapped into a tuple $t^e \in r'$, where r' is tuple of relation *R'*.

$t^e[enc] = E_k(t[a_{i1}, \cdots, a_{ip}])$, $\{a_{i1}, \cdots, a_{ip}\} = F_1$

$t^e[a_{j1}, \cdots, a_{jk}] = t[a_{j1}, \cdots, a_{jk}])$, $\{a_{j1}, \cdots, a_{jk}\} = F_2$

It is clear that it is always possible to obtain a partition of attributes that obeys all the privacy constraints. In the worst case, we can encrypt all the attributes to obey all possible privacy constraints. A key question that remains is: What is the best partition to use, where "best" refers to the partition that minimizes the communication cost of queries the being executed against the database? Let size(*a*) be the physical size of attribute *a* and size(*F*) be the size of partition, computed as the physical size of the attribute composing F_1, that is $Size(F_1) = \sum_{a \in F_1} size(a)$.

Definition 5 (Minimal encrypted attribute partition): Given a relation $R(a_1, a_2, \cdots; a_n)$ and a set of well defined

privacy constraints $C=\{c_1,\cdots,c_m\}$ over R, $F=\{F_1,F_2\}$ is a minimal encrypted attribute partition, if and only if F is a correct partition and $\nexists F'$ such that $size(F_1')<size(F_1)$.

The minimal encrypted attribute partition directly corresponds to the classical *NP*-hard Weighted Minimum Hitting Set problem [13], which can be formulated as follows;

Definition 6 (Weighted minimum hitting set): Given a finite set *S*, a collection *C* of subsets of *S*, a weight function: $w: S \to R^+$, find a hitting set *S'*, that is, a subset of *S* containing at least one element for each subset in *C*, such that $w(S)=\sum_{a\in S'} w(s)$ is minimum.

The minimal encrypted attribute partition can be formulated as the problem of finding the set of attributes with lowest size for breaking each constraint. It is then immediate to see that there is a correspondence between the two problems that can be determined by taking $R=\{a_1,a_2,\cdots\cdots,a_n\}$ as the finite set *S*, $C=\{c_1,\cdots,c_m\}$ as the collection *C*, and by taking as a weight function *w* the attribute size (*w(a)=size(a)*). Since the two problems are equivalent, the minimal encrypted attribute partition is *NP*-hard.

The classical approximation algorithm for the weighted minimum hitting set problem [14] follows a greedy strategy. Fig. 3 represents this approximation algorithm working on an instance of our minimal encrypted attribute partition.

Algorithm1. Approximation algorithm for the physical partition

INPUT
$R=\{a_1,a_2,\cdots\cdots,a_n\}$
$C=\{c_1,\cdots,c_m\}$
$size(a_i)(1\le i\le n)$
OUTPUT
A physical partition R'

MAIN
$F_1=\phi$
While $C\neq\phi$ do
  Let $a\in(R-F_1)$ be the attribute maximizing
    $|\{c\in C: a\in c\}|/size(a)$
  For each $c\in C$ do
    If ($a\in c$) $C=C-c$
  $F_1=F_1\cup\{a\}$
$F_2=R-F_1$
$enc=E_k(F_1)$ // *enc*rypting all attributes in F_1
$R'=\{enc\}\cup F_2$
For each $t\in r$ do // populate new schema R'
  $t^e[enc]=E_k(t[a_{i1},\cdots,a_{ip}])$ // $\{a_{i1},\cdots,a_{ip}\}=F_1$
  $t^e[a_{j1},\cdots,a_{jk}]=t[a_{j1},\cdots,a_{jk}])$ // $\{a_{j1},\cdots,a_{jk}\}=F_2$
  Insert t^e in R'
return R'

Figure 3. approximation algorithm for the minimal encrypted partition

Initially, the F_1 is initialized to the empty set. At each iteration of the while loop, the algorithm choose the attribute a that does not belong to F_1 and that maximizes the ratio between the number of constraints in *C* in which it is involved according to the size of attribute a. Hence, a is insert into F_1 and set *C* of non solved constraints is updated removing the constraints in which the chosen attribute a is involved. The loop terminates when all constraints are solved. Then, F_2 is obtained as the complement of F_1 with respect to *R*.

Fig. 4 represents the execution of attribute partition, step by step, of the algorithm in Fig.3 on relation employees, considering the privacy constraints {{SSN}, {Name, Job}, {Name, Salary}, {DoB, Zip, Salary}, {Job, Salary}} and suppose size(SSN)=18, size (Name)=15, size(DoB)=8, size(Zip) =6, size(Job) =10, size(Salary) =20, size(des)=100.

privacy constraints C	$\lvert\{c\in C: a\in c\}\rvert/size(a)$							F_1
	SSN	Name	DoB	Zip	Job	Salary	Des	
c_1,c_2,c_3,c_4,c_5	1/18	2/15	1/8	1/6	2/10	3/20	3/100	Job
c_1,c_2,c_4	1/9	1/15	1/8	1/6	_	2/20	2/100	Job,Zip
c_1,c_2	1/9	1/15	0	_	-	1/20	1/100	Job,Zip,SSN
c_2	_	1/15	0	_	_	1/20	0	Job,Zip,SSN,Name
Φ	_	_	0	_	_	0	0	Job,Zip,SSN,Name

Figure 4. An example of execution of the attribute partition algorithm

The first column in the table represents the set of privacy constraints that are still unsolved; the subsequent seven columns represent, for each attribute in *R*, the number of unsolved constraints in which they are involved, divided by the attribute's size; the last column represents the attributes composing F_1. Symbol – associated with an attribute means that the attribute already belongs to F_1 and therefore it does not need to consider anymore. The solution computed by the algorithm is *F*={{SSN, Name, Zip, Job}, {DoB, Salary, Des}}, which is minimal partition for the considered example.

The algorithm in Fig. 3 also shows the construction and population of physical partition. Fig. 5 illustrates an example of physical partition for original relation *R* that correctly enforces the well defined constraints in *C*.

Enc	DoB	Salary	Des
α	18/02/78	123.4	Aaaaaa
β	25/10/76	234.5	Bbbbbb
γ	15/08/77	456.7	Ccccc
δ	12/06/78	321.4	Dddddd
……	……	……	……

Figure 5. An example of physical partition for original relation R

Enforcing selective access with the explicit definition of authorizations requires the trusted front-end to intercept and process each query request and each reply to filter out data the client is not authorized to access to increase the processing and communication load at the trusted front-end site. To enforce selective access, this model defines and maintains additional information at the level of metadata at the trusted front-end site.

Fig. 6 illustrates an example of metadata stored at the trusted front-end site during the processing of attribute

partition and encryption. The first four columns represent the original schema and new schema; the flag column represents whether the attribute is encrypted; the last column stores the keys used to encrypt tuples.

Ori Table	Ori Attr	New Table	New Attr	Flag	Keys
Emploree	SSN	Emploree'	Enc	1	Abcd
Emploree	Name	Emploree'	Enc	1	Abcd
Emploree	DoB	Emploree'	DoB	0	Null
Emploree	Zip	Emploree'	Enc	1	Abcd
Emploree	Job	Emploree'	Enc	1	Abcd
Emploree	Salary	Emploree'	Salary	0	Null
Emploree	Des	Emploree'	Des	0	Null

Figure 6. An example of metadata associated with the trusted front-end

Algorithm running time complexity analysis: Given relation schema $R=\{a_1,a_2,\cdots,a_n\}$ and a set of privacy constraints $C=\{c_1,\cdots,c_m\}$, to chose attribute a from R, in the worst case the algorithm scans Constraints C, and adjusts C for each attribute involved in at least one constraint with a. This operation cost $O(mn)$ for each attribute chosen.

B. Physical Partition with Quasi-Identifier Detection

The sensitive attributes can be defined explicitly when user want to outsource data to the service provider. However, it is difficult for user to define those privacy constraints implied by the combinations of distinct seemingly innocuous attribute values. For example, in a statistic database, attributes that seem innocuous, such as gender, date of birth, zipcode, can be joined with other data to re-identify individuals. It is necessary to find automatically whether the combinations of distinct seemingly innocuous attribute values could identify individual row with sufficiently high probability. If there is such a set of attributes, we need to adjust the result relation R'. We can choose randomly an attribute belong to the quasi-identifier and put the attribute into F_1 to adjust the relation R'. In order to automatically detect quasi-identifier [15], following basic concepts are defined:

Definition 7 (Quasi-identifier set): A quasi-identifier set is a minimal set of attributes in Relation R that can be joined to re-identify individual records with sufficiently high probability.

Definition 8 (α-quasi-identifier): An α-quasi-identifier is a set of attributes along which an α fraction of records in the universe can be identified by values along the combination of these attribute columns uniquely.

Definition 9 (Attribute multiple domain): Let $n_1,n_2,..,n_k$ be the number of distinct values along columns $a_1,a_2,..,a_k$ respectively. The Total number of distinct values taken by the $a_1,a_2,..,a_k$ columns is $D = d_1 \times d_2 \times \ldots \times d_k$.

Suppose that a set of columns take D different values with probabilities $p_1, p_2,\ldots, p_n$, Where $\sum_{i=1}^{D} p_i = 1$. Let us calculate the probability that i^{th} element is a singleton in the universal table R. It means first selecting one of the entries in the table (there are n choices), setting it to be this i^{th} element with probability p_i, and setting all other entries in the table to something else(which happens with probability $(1-p_i)^{n-1}$). The probability of i^{th} element being a singleton in the universal table R is $np_i(1-p_i)^{n-1}$.

Let X_i be the indicator variable representing whether i^{th} element is a singleton, then its expectation:

$$E[X_i] = P[X_i = 1] = np_i(1-p_i)^{n-1} \approx np_i e^{-np_i}.$$

Let $X = \sum_{i=1}^{D} X_i$ be the counter for number of singleton, its expectation is given by $E[X] = \sum_{i=1}^{D} E[X_i] = \sum_{i=1}^{D} np_i e^{-np_i}$.

Let us analyze which distribution maximizes expected number of singletons. We aim to maximize $\sum_{i=1}^{D} x_i e^{-x_i}$, subject to $\sum_{i=1}^{D} x_i = n$ and $0 \le x_i, \forall 1 \le i \le D$.

Theorem 1 If $D \le n$, then the expected number of singletons is bounded above by D/e.

Proof: if $f(x)=xe^{-x}, f'(x)=(1-x)e^{-x}$ and $f''(x)=(x-2)e^{-x}$. Thus, the function f has a global maximum at x=1, since $f'(1) = 0$ and $f''(1) < 0$. Now the expected number of singletons is

$$\sum_{i=1}^{D} x_i e^{-x_i} \le \sum_{i=1}^{D} e^{-1} = \frac{D}{e}.$$

Theorem 2 If $D \ge n$, the expected number of singleton is bounded above by $ne^{-n/D}$.

Proof: f $f(x)=xe^{-x}, f'(x)=(1-x)e^{-x}$ and $f''(x)=(x-2)e^{-x}$. The function f has a point of inflection at x=2, since $f''(x) < 0$ for x<2 implying the function is concave here, and $f''(x) > 0$ for x>2 implying the function is convex here. It is sure that no $x_i \geqslant 2$ while maximizing $\sum_{i=1}^{D} x_i e^{-x_i}$. Otherwise: after maximizing $\sum_{i=1}^{D} x_i e^{-x_i}$, some $x_a \geqslant 2$. As $D \ge n$ *and* $\sum_{i=1}^{D} x_i = n$, some $x_b < 1$. For some small δ, replacing x_a by $x_a - \delta$ and x_b by $x_b + \delta$ while retaining $\sum_{i=1}^{D} x_i = n$. As $f(x) = xe^{-x}$ increases towards x=1, $f(x_a-\delta) > f(x_a)$ and $f(x_b+\delta) > f(x_b)$. Thus $\sum_{i=1}^{D} x_i e^{-x_i}$ is increased, contradicting the fact that it was maximized. Thus, $\forall 1 \le i \le D, x_i \le 2$. Now $f''(x) < 0$ for $0 \le x \le 2$. Since f is concave, we can get

$$\sum_{i=1}^{D} x_i e^{-x_i} = D\sum_{i=1}^{D} \frac{1}{D} x_i e^{-x_i} \le D \bullet (\sum_{i=1}^{D} \frac{x_i}{D}) e^{-(\sum_{i=1}^{D} \frac{x_i}{D})} = ne^{\frac{-n}{D}}.$$

Thus, if $D \ge n$, the expected number of singletons is bounded above by $ne^{-n/D}$.

Theorem 3 Given a table of size n, a set of attributes can form an α-quasi-identifier if the number of distinct values along the columns, $D > n/\ln(1/\alpha)$.

Proof: Note that $D > n$. If not, then, by Theorem 1, the maximum expected fraction of rows taking unique values is $D/en \le 1/e < a$. From Theorem 2, the maximum expected fraction of rows taking unique values along the columns with D distinct values is $e^{-n/D}$. For the set of documents to form a α-quasi-identifier, this fraction must be larger then α. Thus $e^{-n/D} > \alpha$, which implies that $D > n/\ln(1/\alpha)$.

In order to detect whether quasi-identifier exists in non-encrypted attributes automatically, we need to add

the quasi-identifier detection in algorithm 1. Fig. 7 illustrates the approximation algorithm for physical partition with automatically detection of quasi-identifier.

Algorithm2. Approximation algorithm for the physical partition with automatically detection of quasi-identifier

INPUT
$R = \{a_1, a_2, \cdots\cdots, a_n\}$
$C = \{c_1, \cdots, c_m\}$
$size(a_i)(1 \le i \le n)$
α
OUTPUT
A physical partition R'

MAIN
$F_1 = \phi$
While $C \neq \phi$ do
Let $a \in (R - F_1)$ be the attribute maximizing
$| \{c \in C : a \in c\} | / size(a)$
For each $c \in C$ do
If ($a \in c$) C=C-c
$F_1 = F_1 \bigcup \{a\}$
End while
$F_2 = R - F_1$
// detecting quasi-identifier in F_2
QI=Quasi-identifier(F_2, α)
While $QI \neq \phi$ do
Let b be any attribute in QI
$F_1 = F_1 \bigcup \{b\}$
$F_2 = F_2 - \{b\}$
QI=Quasi-identifier(F_2, α)
End while
// populate new schema R'
$enc = E_k(F_1)$ // *enc*rypting all attributes in F_1
$R' = \{enc\} \bigcup F_2$
For each $t \in r$ do
$t^e[enc] = E_k(t[a_{i1}, \cdots, a_{ip}])$ // $\{a_{i1}, \cdots, a_{ip}\} = F_1$
$t^e[a_{j1}, \cdots, a_{jk}] = t[a_{j1}, \cdots, a_{jk}])$ // $\{a_{j1}, \cdots, a_{jk}\} = F_2$
Insert t^e in R'
return R'

Figure 7. Approximation algorithm for the minimal encrypted partition with automatically detection of quasi-identifier

We test the automatically detect quasi-identifier on the cell-phone card data provided by a mobile company. This dataset has total n =3*107 rows with 31 columns. We choose 25860 rows from the dataset and consider 8 non-sensitive attributes of them. The 8 non-sensitive attributes are open date, null flag, use state, resource type, manufacture code, business type, node level and card type. The distinct values of the 8 non-sensitive attributes are 60, 2, 8, 14, 7, 40, 5 and 20. The experiment was run on a machine with 2.31GHz processor and 2GB of RAM running windows XP. Fig. 8 illustrates an example of detecting quasi-identifier automatically.

The column *ROW* is the number of test; the column *CN* is the number of columns that make the quasi-identifier; the column *N* is the number of rows uniquely identified in the projection; the column *G1* is the fraction of rows uniquely identified, given by *N*/25860; the column *D* is the product of the domain sizes of the attributes; and the column *G2* is the fraction of singletons, given by $f(D/n)=D/en$ for $D<n$ and $e^{-n/D}$ for $D>n$.

Given a=0.2, if we only consider the value of *G1*, then rows numbered 5,6,7 and 8 are all quasi-identifier and can not be publish. In fact, *G1* > 0.2 fraction of the rows should not get uniquely identified, then, only row 8 qualifies as a possible 0.2-quasi-identifier as only its *G2* value exceeds 0.2.

ROW	*CN*	*N*	*G1*	*D*	*G2*
1	1	2	$7.0*10^{-5}$	60	$7.4*10^{-7}$
2	2	835	0.029	1200	$1.48*10^{-5}$
3	3	52	$1.8*10^{-3}$	600	$7.4*10^{-6}$
4	4	2756	0.096	268800	$3.29*10^{-3}$
5	4	7081	0.247	672000	$8.24*10^{-3}$
6	5	11567	0.405	940800	$1.15*10^{-2}$
7	5	10034	0.351	$5.37*10^{6}$	$6.58*10^{-2}$
8	8	21802	0.763	$3.76*10^{8}$	0.92

Figure 8. An example of detecting quasi-identifier automatically

V. Obtaining User Privacy via Auxiliary Random Servers Based on Encryption

We still use the protocol like RSM-PIR [12] to achieve user privacy protection in our new model. Although the computational complexity of the RSM-PIR protocol reaches *O(1)*, its communication complexity reaches $O(C_s*logn)$, *logn* times of the information retrieval scheme *S*. To ensure safety, auxiliary servers needs to set up safety coefficient to reset the random string r and permutation π, which increase the computation cost of setup stage. To overcome the drawback of the RSM-PIR protocol, we combine auxiliary servers and crypt-graphic technology to achieve the same functions of the RSM-PIR protocol by replacing the random string r and permutation π with n pairs of private keys $<U_i, V_i>$. Due to encryption in the new protocol, the computation complexity increases, and the communication complexity remains the same $O(C_s)$ as protocol *S*. Because the new protocol needs no safe coefficient, the computation cost during setup stage decrease. Before we detail our new model, we introduce parameters setting as follows:

$t'_1, t'_2, \cdots, t'_n$: be tuples of the relation R'.

E: be efficient symmetrical encryption algorithms, such as AES and DES, where $U = E_k(T)$ and $T = E_k^{-1}(U)$.

F: be permutable encryption function, which means using different keys to encrypt plaintext in different order may obtain the same result, that is, $F_{k_1}(F_{k_2}(T)) = F_{k_2}(F_{k_1}(T))$

To achieve total independence, all auxiliary servers must be independent of the data. On the other hand we must use a number of multiple servers in order to achieve information theoretic results that are efficient. To accommodate these two conflicting requirements we use the following steps during setup stage:

1. $\forall i, 1 \le i \le n$, the database server chooses at random n keys $k_1, k_2, \cdots, k_n$ for symmetrical encryption function E, and chooses key e for permutable encryption function F to encrypt $t'_1, t'_2, \cdots, t'_n$, the results represent as follows:

$U_1 = E_{k_1}(t'_1), V_1 = F_e(k_1)$

$U_2 = E_{k_2}(t'_2), V_2 = F_e(k_2)$

……

$U_n = E_{k_n}(t'_n), V_n = F_e(k_n)$

2. The database server sends *n* pair of cipher $<U_1,V_1>$, $<U_2,V_2>$, ……, $<U_n, V_n>$ to *k* auxiliary servers, and each auxiliary servers contains the *n* pair of cipher $<U_i, V_i>$.

In the on line stage the client interacts with the servers and the principal database in order to obtain his query. The steps can be described as follows:

1. Utilizing information retrieval scheme *S*, the client accesses the *k* auxiliary servers to get the pair of cipher $<U_i, V_i>$ as the security label which be only known to the client.

2. The client chooses a private key *r* to compute $G = F_r(V_i)$ and sends *G* to the database server.

3. The database server decrypts *G* using the permutable encryption function *F* to compute *H* and sends *H* to the client:

$$H = F_e^{-1}(G) = F_e^{-1}(F_r(V_i)) = F_e^{-1}(F_r(F_e(k_i))) = F_e^{-1}(F_e(F_r(k_i))) = F_r(k_i)$$

4. The client first decrypts *H* using the private key *r* to compute k_i : $F_r^{-1}(H) = F_r^{-1}(F_r(k_i)) = k_i$, then , it decrypts U_i using k_i to compute t'_i: $E_{k_i}^{-1}(U_i) = E_{k_i}^{-1}(E_{k_i}(t'_i)) = t'_i$.

Here, we choose Pohlig-Hellman encryption scheme [16] as the permutable encryption function *F, where encryption processing is* $U = T^e \bmod p$ and decryption processing is $T = U^d \bmod p$. We assume that *p* is a big prime (1024 bit), the client's pair of private keys be (*r,s*), and the database server's pair of private keys be (*e,d*). If an adversary has the *e* and *p*, he can know *d*, otherwise, he has to computer $e = \log_U T \bmod p$ to get *d*.

It is clear that our scheme is correct. Utilizing protocol *S*, the client accesses the *k* auxiliary servers to get $<U_i,V_i>$, and can get what he want to get t'_i using the permutable encryption function *F*.

Our scheme is safety and discloses no client's privacy information. The security of getting $<U_i,V_i>$ depends on the information retrieval scheme *S*, and the *k* auxiliary servers have not known anything about *i*. The Client sends *G* to the database server, but the database server does not know the private *r* key, it can not decrypt *G* to obtain any information about t'_i. If in multiple executions two clients are interested in the same query, the database will receive the same query and will know that the two queries are the same. This problem can be fixed by changing the client's pair of private keys $<r,s>$.

Our scheme also ensures that the client be not allowed to get more information other than what he is querying on the database server. If the underlying *S* is database private, the client can get nothing except for the pair of $<U_i,V_i>$. If the underlying *S* is not database private, the client can get $<U_j,V_j>$ besides the pair of $<U_i,V_i>$, hence, the client can get more information then he should get. This problem can be fixed by replacing $U_i = k_i^e \bmod p$ with $U_i = (P(k_i)^e \bmod p$ in Pohlig-Hellman encryption scheme, where p is a padding format.

The communication complexity of the scheme during the on line stage is $O(C_s)$, where C_s is the complexity of the information retrieval scheme *S*. We do not consider the communication complexity of setup stage, because the communication work can be done off line. During the on line stage, the client needs to access the *k* auxiliary servers based on *S*. Let the size of tuple of relation *R'* be *m* bits, the total size of database is $N=m*n$ bits, each auxiliary server contains n pair of $<U_i,V_i>$, the total size of U_i *and* V_i are *N* bits and *1024*n* bits. The communication costs of retrieving U_i *and* V_i are C_s and $(1024/m)* C_s$. The communication cost between the client and database server is *logG+logH*, so the communication cost of this scheme is $(1+1024/m)*C_s+logG+logH$, *and holds the same communication complexity* $O(C_s)$ *as* scheme *S*.

The computation complexity of the scheme during the on line stage is $O(\log^3 p)$. Due to the large amount of encryption computation having been done in setup stage, in fact, the computation cost for the principal database is only an encryption operation during the on line stage, that is, $H = F_e^{-1}(G) = U^e \bmod p$, and the computation complexity of this encryption operation is $O(\log^3 p)$. The computation complexity of this scheme is independent on the size of database and is only dependent on prime *p*.

VI. CONCLUSIONS

We have described new techniques for enforcing data privacy/confidentiality and user privacy over outsourced database services using attribute partition and encryption, and applying cryptographic technology on the auxiliary random server protocol. The main contribution of this paper is threefold. First, we provide a simple and powerful way to capture privacy requirements. Second, we provide a model to obtain data privacy/confidentiality based on attribute partition and encryption, and introduce an approximation algorithm for minimal encrypted attribute partition with automatic detection of quasi-identifier. Third, we propose a new model for PIR, utilizing cryptographic technology on auxiliary random servers providing privacy services for database access. We conclude that this provides a powerful new building block for the construction of outsourced database services in the un-trusted infrastructure.

ACKNOWLEDGMENT

This research was partially supported by Chinese National Programs for High Technology Research and Development under grant No.2007AA01Z448; The Key Research Project of Anhui Higher Institutes Nature Science under grant No.KJ2010A003; The Opening Project of State Key Lab for Novel Software Technology of Nanjing University under grant No.KFKT2010B01.

REFERENCES

[1] H. Hacigumus, B. Iyer, and S. Mehrotra, "Providing Database as a Service," *Proceedings of the International Conference on Data Engineering (ICDE 2002)*, pp. 29–38, February 2002.

[2] S. Mehrotra, H. Hacigumus, and B. Iyer, "Efficient Execution of Aggregation Queries over Encrypted Relational Databases," *Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, pp. 125–136, March 2004.

[3] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu, "Fine Order-Preserving Encryption for Numeric data," *Proceedings of the ACM SIGMOD International Conference on Management of Data (ICMD 2004)*, pp. 563–574, June 2004.

[4] TRUSTED COMPUTING GROUP, "TPM Specification Version 1.2. Part 1 Design Principles," *https://www.trustedcomputinggroup.org/specs/TPM/Main_Part1_rev94.zip*, 2007.

[5] G. Arrarwal, N. Mishra, and B. Pinks, "Secure Computation of the k th-ranked Element," *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2004)*, pp. 40–55, May 2004.

[6] Murat Kantarcioglu and Chris Clifton, "Security Issues in Querying Encrypted Data,' *Purdue University*, 2004.

[7] G. Aggrawal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R.Motwani, *et al*, "Two Can Keep a Secret: A Distributed Architecture for Secure Database Services," *Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, pp. 186–199, January 2005.

[8] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private Information Retrieval," *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1995)*, pp. 41–50, October 1995.

[9] R. Ostrovksy, V. Shoup, "Private Information Storage," *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC 1997)*, pp. 294–303, May 1997.

[10] A. Beimel, Y. Ishai, E. Kushilevitz, J.F Raymond, "Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval," *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pp. 261–270, Nov 2002.

[11] E. Kushilevitz, R. Ostrovsky, 'Replication is Not Needed: Single Database, Computationally-Privacy Information Retrieval,' *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1997)*, pp. 364–373, October 1997.

[12] Y. Gertner, S. Goldwasser, T. Malkin, "A Random Server Model for Private Information Retrieval or How to Achieve Information Theoretic PIR Avoiding Database Replication,' *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM 1998)*, pp. 200–217, October 1998.

[13] M. Garey and D. Johnson, "Computers and intractability: a Guide to the Theory of NP-completeness," W.H. Freeman, 1979.

[14] D. Johnson, "Approximation Algorithm for Combinatorial Problems," *Proceedings of the ACM Symposium on Theory of Computing (STOC 1973)*, pp. 38–49, May 1973.

[15] S. Lodha and D. Thomas, "Probabilistic anonymity," *Proceedings of the First International Workshop on Privacy, Security, and Trust in KDD,* pp. 56–79, May 2007.

[16] S. Pohlig and M. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance,". *IEEE Transactions on Information Theory*, vol. 24, pp. 106–110, 1978.

Yong-Hong Yu received his Ph.D degree in computer science from Nanjing University, Nanjing, China, in 2001. Currently, he is an associate professor in school of information engineering at Anhui University of Finance & Economics. His research interests include information security, database system.

Wen-Yang Bai received his Master degree in computer science from Nanjing University, Nanjing, China, in 1992. Currently, he is working on doctoral degree at Nanjing University. He is currently an associate professor in department of computer science at Nanjing University. His research interests include secure database, data minning.