Privacy Preserving Data Distribution in Outsourced Environments

T.J.V.R.K.M.K. Sayi*[1], R.K.N. Sai Krishna*[2], R. Mukkamala[3], P.K. Baruah[4]

Sri Sathya Sai Institute of Higher Learning, Prasanthi Nilayam, India
{[1]sai.jyothir,[2]rkn.sai}@gmail.com, [4]pkbaruah@sssihl.edu.in

[3]Old Dominion University, Norfolk, VA, USA
[3]mukka@cs.odu.edu

***Abstract*—With the increasing cost of maintaining IT centers, there is a perceived trend among organizations to turn to cloud servers for their storage and computational needs. However, such outsourcing has also raised the more serious issue of data privacy. In this paper, we summarize our on-going work in privacy preserving data outsourcing. In particular, in this paper, we discuss the issue of employing vertical fragmentation to a relation so that the fragment that is assigned to the cloud server contains maximum data without violating privacy. We represent the confidentiality constraints as a graph and apply 2-coloring algorithms for the acyclic portion of the graph. We use some heuristic to eliminate the cycles, and complete the coloring of all nodes. We are currently extending the work to multiple relations and constraints with multiple attributes.**

Keywords: Fragmentation, 2-graph coloring, outsourcing, confidentiality constraints.

I. Introduction

As the amount of information held by organizations is increasing rapidly, there is a need for third-party providers that can offer storage and computational facilities with significant economies of scale. This enables the organizations to outsource their data management functions to third-party providers. The need is currently met by cloud providers such as Amazon, Google, and Akamai. But, this has also given rise to the issues of data privacy and confidentiality. The topic of privacy in outsourced environments has been the focus of several researchers. Samarati and Vimercati discuss different solutions to address the issue of data confidentiality violations in outsourcing environments [1]. In particular, the authors discuss the issues in data protection, query execution, data privacy, data integrity and correctness, access control enforcement, and collaborative computing. Wang et al propose fine-grained access control (encrypting each data block with a different key) and adopting over encrypting for data isolation among users as the primary means to achieve security and efficiency in accessing outsourced data [2]. Aggarwal et al. propose a solution in which only sensitive attributes are encrypted and the attributes in a sensitive association are split among different non-communicating servers [3]. But, this scheme has two main limitations: (i) It assumes that servers are non-communicating; in general, data owners cannot enforce this rule on cloud operators; (ii) high communication cost during query execution as the data owner needs to interact with several servers.

Ciriani et al. [4] propose a solution using both fragmentation and encryption in which all fragments are unlinkable. Further, in [5][6][7], the authors propose different heuristics that relies only on fragmentation but involves data owner in data storage and management. In these solutions, the data owner holds all the sensitive attributes and one of the attributes in each sensitive association.

To avoid the high cost of encryption and decryption, we propose a solution based only on fragmentation. For simplicity, we consider a single data owner and a single server scenario. However, the proposed algorithms are easily extendable to other environments. Our method fragments the data using graph-coloring techniques. The solution aims to minimize the amount of data stored and/or the workload at the data owner such that there is no violation of data confidentiality. The confidentiality constraints are expressed as singleton attributes or attribute pairs. For example, while *Name* and *Disease* by themselves may not be confidential, the <*Name, Disease*> pair may be confidential. Work on extending

*Student Author

this scheme to more complex confidentiality constraints is currently in progress.

The paper is organized as follows. In section II, we describe the system model that we have adopted. It also introduces the needed notation and terminology. Section III summarizes the proposed scheme, especially the algorithm for constructing a confidentiality graph and the fragmentation algorithm. In section IV, we illustrate our scheme using an example. Finally, section V summarizes the contributions of this paper and describes future work.

II. System Model

We consider a relation r over the relation schema $R(a_1, a_2, ..., a_m)$ which consists sensitive data i.e., the data which cannot be revealed to an external party. We represent the sensitive data in the form of a set of confidentiality constraints $C(c_0, c_1, ..., c_n)$ which is a subset of the relation schema *R* as given in [1]. For simplicity, we consider confidentiality constraints with only one or two attributes.

For example, in Table I(b), c_0 represents a singleton confidentiality constraint which indicates that the attribute *SSN* should not be revealed. c_1 indicates that the attributes *Name* and *Illness* in association are sensitive. Hence, both of the attributes in association should not be revealed.

To preserve the sensitivity of an attribute or attributes in association we divide the relation into two vertical fragments so that one fragment, F_o is stored at the owner and the other, F_s at an external server. To reconstruct the *r* from F_o and F_s, they must have a common tuple-id *(tid)*. The *tid* can either be a primary key, if it is not sensitive or an attribute which is not part of *R* that will be included after the fragmentation process.

For example, in Table. I(a), the attribute *SSN* is a primary key of the Patient relation but it cannot be the *tid* because it is sensitive. The Table also indicates the size of each attribute in bytes.

TABLE I

SSN(9)	Name(15)	DoB(8)	ZIP(5)	Job(10)	Illness(7)	Treatment(100)
322-42-4224	Mary	02/11/76	42422	Manager	laryngitis	antibiotic
424-22-2522	Pete	23/02/79	63244	Secretary	diabetes	insulin
522-93-5221	Lisa	15/06/82	74224	lawyer	flu	aspirin
422-52-5225	Deo	29/03/88	42214	Student	laryngitis	antibiotic
942-25-8242	Bobby	31/10/88	53252	banker	diabetes	insulin

(a) PATIENT relation

$c_o = \{SSN\}$
$c_1 = \{Name, Illness\}$
$c_2 = \{Name, Treatment\}$
$c_3 = \{ZIP, Illness\}$
$c_4 = \{ZIP, Treatment\}$
$c_5 = \{Job, Illness\}$
$c_6 = \{Job, Treatment\}$

(b) Set of Confidentiality constraints

A fragment *F* violates a confidentiality constraint *c* if all the attributes in *c* are present in *F*. For example, if the Patient relation in Table. I(a) is fragmented as F = <F_o, F_s> where F_o = *{SSN, Job, ZIP}* and F_s = *{Name, Illness, Treatment, DoB}*, then F_s violates both c_1 and c_2 since *Name, Illness* and *Treatment* are present in F_s.

Now, our aim is to fragment *r* so that the data storage and/or the workload at the owner is minimized and none of the confidentiality constraints are violated by the server fragment. In [5][7], the authors give four different metrics to determine which of the attributes are stored at the owner and which of them at the server through a weight function and summarize them as given in Table. II. For example, given F = <F_o, F_s> where F_o = *{Name, SSN, Job, ZIP}*, F_s = *{Illness, Treatment, DoB}* and the sizes of attributes as in Table. III(a) and the complete expected query workload profile as in Table. III(b) then the weight of the fragmentation with respect to different metrics are as follows.

W_a(F) = 4; W_s(F) = 15 + 9 + 10 + 5 = 39; W_q(F) = 1 + 4 + 10 = 15; W_c(F) = 1 + 1 + 4 + 10 = 16

The query workload profile for the metric Min-Query is a set of triplets of the form $\{(q_1$,*freq*(q_1),*Attr*$(q_1))$, ..., $(q_k$,*freq*(q_k),*Attr*$(q_k))\}$ and for the metric Min-Cond it is a set of triplets of the form $\{(q_1$,*freq*(q_1),*cond*$(q_1))$, ..., $(q_k$,*freq*(q_k),*cond*$(q_k))\}$.
Attr(q) and *cond(q)* are the set of attributes and the set of conditions that appear in the *where* clause of the query respectively. *freq(q)* is the expected execution frequency of q.

TABLE II

Problem	Metrics	Weight Function
Min-Attr	Number of attributes	$card(F_o)$
Min-Size	Size of attributes	$\sum_{a \in F_o} size(a)$
Min-Query	Number of queries	$\sum_{q \in Q} freq(q) s.t. Attr(q) \cap F_o \neq \emptyset$
Min-Cond	Number of conditions	$\sum_{cond \in Cond(Q)} freq(cond) s.t. cond \cap F_o \neq \emptyset$

(a) Classification of the weight metrics

Problem	Target T	$w(t)\ \forall t \in T$
Min-Attr	$T = \{\{a\} \mid a \in R\}$	$w(t) = 1$
Min-Size	$T = \{\{a\} \mid a \in R\}$	$w(t) = size(a) \ni \{a\} = t$
Min-Query	$T = \{attr \mid \exists q \in Q, Attr(q) = attr\}$	$w(t) = \sum_{q \in Q} freq(q) s.t. Attr(q) = t$
Min-Cond	$T = \{cond \mid \exists q \in Q, cond \in Cond(q)\}$	$w(t) = freq(cond) s.t. cond = t$

(b) Target sets for different metrics

III. Proposed Scheme

Since arriving at a minimal fragmentation is an NP-Hard problem as discussed in [5], we propose a heuristic algorithm to find the owner and server fragments.

TABLE III

Query q	**freq**(q)	**Attr**(q)	**Cond**(q)
q1	5	DoB, Illness	$< DoB >, < Illness >$
q2	4	ZIP, Illness	$< ZIP >, < Illness >$
q3	10	Job, Illness	$< Job >, < Illness >$
q4	1	Illness, Treatment	$< Illness >, < Treatment >$
q5	7	Illness	$< Illness >$
q6	7	DoB, Treatment	$< DoB >, < Treatment >$
q7	1	SSN, Name	$< SSN >, < Name >$

Complete query workload

The singleton confidentiality constraints represent sensitive attributes. Hence we store them at the owner and remove from *C*. In addition, we remove all the constraints which include attributes in the singleton constraints from *C*. Also, we store the attributes that are not present in any of the constraints in *C* at the server. For example, given *r* and *C* as in Table. I, we remove {*SSN*} from *C* and add it to F_o and add *DoB* to F_s.

Our algorithm *FRAGMENTATION* takes as input, a relation Schema *R*, a set *C* of confidentiality constraints, a set *T* of targets and a weight function *W* defined on *T* and returns a correct fragmentation <F_o, F_s> using the *2-graph coloring algorithm*.

The algorithm constructs an acyclic graph *G = (V,E)* using the function *Construct-graph*. In addition to G, the function also returns a set of constraints To_solve that are not considered so that an acyclic graph can be constructed. Then the algorithm runs the 2-graph coloring algorithm on G to find two fragments F_1 and F_2. Subsequently, it labels the fragment with minimum weight as F_o and the other as F_s. If *G* contains unconnected components, then 2-graph coloring algorithm returns a fragment pair for each unconnected component. Then the algorithm finds the fragment with minimum weight in each fragment pair and merges them to a single fragment F_o. Similarly, it merges the other parts into F_s.

Now, the algorithm removes all the confidentiality constraints in the set To_solve which contains at least one attribute in F_o. If To_solve is nonempty, then it contains the confidentiality constraints where all the attributes in the constraints are in F_s. Hence, F_s violates all the constraints in To_solve. So, to satisfy the constraints, algorithm recurses with To_solve as the new set of confidentiality constraints.

The function *Construct-graph*, constructs an acyclic graph *G* as follows.

For each confidentiality constraint c = (a_i, a_j) in *C*, the function adds a_i and a_j to *V* if they do not belong to *V* and an edge (a_i, a_j) to *E*. If the function finds a cycle with even number of edges by the inclusion (a_i, a_j), then it removes (a_i, a_j) from *E* and adds it to the set To_solve to avoid multiple cycles in future. If the function detects a cycle with odd number of edges, say *N*, then 2-graph coloring algorithm results in a graph which contains two adjacent vertices with same label. That is, 2-graph coloring algorithm returns a fragment pair $< F^1, F^2 >$ such that there is a constraint (a_m, a_n) whose attributes are present in the same fragment. Thus, the fragment violates the constraint (a_m, a_n). So, an edge must be ignored to apply the 2-graph coloring algorithm. But, for a cycle with odd number of edges, to satisfy all the constraints involved, $\lceil N/2 \rceil$ attributes must be stored at the owner and $\lfloor N/2 \rfloor$ attributes at the server. To find the set of $\lceil N/2 \rceil$ attributes which gives a correct and minimal owner fragment with respect to the weight function, the algorithm iteratively finds the pair of fragments for all the sub-graphs of the cycle with $N-1$ edges using the 2-graph coloring algorithm. Then, it chooses a fragment pair in which the weight of the fragment with $\lceil N/2 \rceil$ attributes is minimum compared to the remaining fragment pairs.

Now the algorithm adds the edge, whose removal from the cycle resulted in a sub-graph with the chosen fragment pair, to To_solve and removes it from *E*. Then the function *Construct-graph* returns *G*.

Algorithm 1: FRAGMENTATION

input: R , C , T and W
output: F = $< F_o, F_s >$
`/* A correct fragmentation */`
$G := Construct - graph(C, R)$
`/* G is a graph representation of the confidentiality constraints */`
$< F_1, F_2 >:=$ 2-graph coloring(G)
if $weight(F_1) < weight(F_2)$ **then**
  $F_o := F_1$ and $F_s := F_2$
else
  $F_o := F_2$ and $F_s := F_1$
end
foreach $c = (a_i, a_j) \in To\_solve$ **do**
if a_i *or* $a_j \in F_o$ **then**
  $To\_solve := To\_solve \setminus c$
end
if $To\_solve \neq \emptyset$ **then**
  $C := To\_solve$
  $< F_1, F_2 >:= FRAGMENTATION$
  $F_o := F_o \cup F_1$ and $F_s := F_s \setminus F_1$
end
return $(< F_o, F_s >)$

Function Construct-graph(C,R)

$G := (V, E)$; $V := \emptyset$; $E := \emptyset$
$To\_solve := \emptyset$
foreach $c \in C$ **do**
  /* c = (a_i, a_j) */
  if $a_i, a_j \notin V$ **then**
    Add $a_i, a_j toV$
  end
  Add $(a_i, a_j) toE$
  $G1 := find\_cycle(G)$
    /* $G_1 := (V_1, E_1)$ is a sub graph where $V_1 = \{a_{l_0}, a_{l_1}, a_{l_2}, ..., a_{l_k}\}$ */
  if *Number of edges in the path are even* **then**
    Remove (a_i, a_j) from E_1 and add it to To_solve
  else
    for $iter := 0$ **to** k **do**
      $< F^1, F^2 >:=$ 2-graph coloring$(V_1, E_1 \setminus (a_{iter \ mod \ k+1}, a_{iter+1 \ mod \ k+1}))$
      $weight[iter] := W(F^i)$ where F^i is the fragment with more number of vertices
    end
    $Min\_weight\_index :=$ $min(weight[0], ..., weight[k])$
    $Remove(a_{l_{min\_weight\_index \ mod \ k+1}}, a_{l_{min\_weight\_index+1 \ mod \ k+1}})$ from E and add it to To_solve.
  end
end
return G.

The query evaluation process follows either client-server strategy or server-client strategy as proposed in [5][6][7].

IV. Illustration

In this section, we illustrate the proposed scheme using an example scenario. We consider the PATIENT relation (R) and a set of seven confidentiality constraints (C). These are shown in Table I. Six of the constraints are pairs (c_1-c_6) and one is a singleton (c_0). With this input information, we run the proposed scheme to determine the owner and server fragments. For simplicity, *we represent each attribute in the relation with its initial.* For example, SSN is referred to as S and ZIP by Z.

First, we start building the constraint graph using *Construct-graph* algorithm. c_0, being a singleton constraint, need not be considered here. c_1-c_3 result in a graph shown in Fig. 1(a). When c_4 is considered, the function adds an edge between *Z* and *T* which results in a cycle *ZINTZ*. Since the number of edges in the cycle is even, the function adds *(Z,T)* to the set To_solve and removes it from *G*. Subsequently, considering c_5, it adds the node *J* and an edge between *J* and *I*. The resulting graph *G* is shown in Fig. 1(b).

Similar to c_4, when c_6 is considered, it results in a cycle with even number of edges. To avoid the cycle, *JINTJ*, the function adds *(J,T)* to the set To_solve removing it from *G*. Hence, To_solve consists of *(J,T)* and *(Z,T)*. Then the algorithm performs 2-graph coloring to *G* as shown in Fig. 1(c). This results in two fragments F_1 = {*J,Z,N*} and F_2 = {*I,T*}.

Now, the algorithm determines the owner and server fragments using the weight functions (Table II).

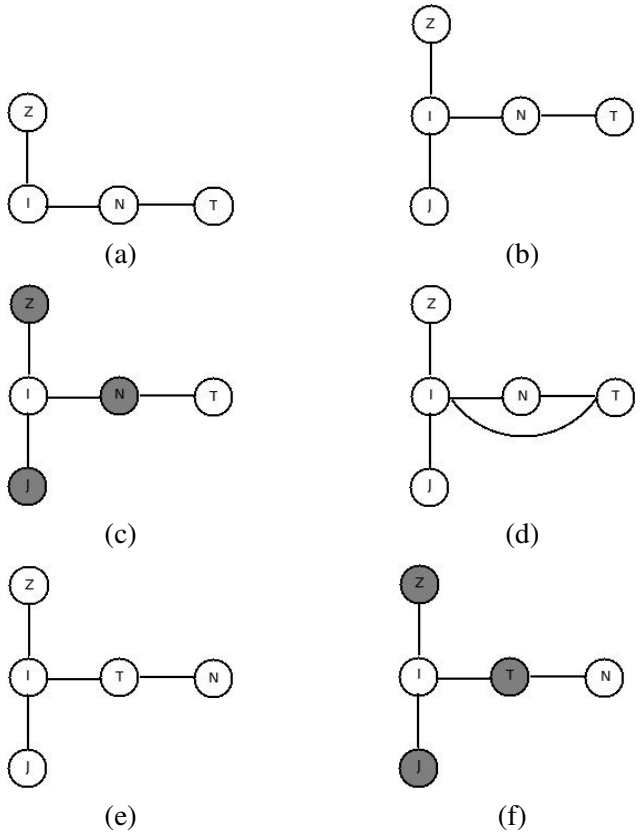

Fig. 1. Different instances of the confidentiality graph

- If the weight function is that of *Min-Attr*, then $W_a(F_1)$ = 3 and $W_a(F_2)$ = 2. Since $W_a(F_1) > W_a(F_2)$, $F_o = F_2$ and $F_s = F_1$. Since, the attribute 'Treatment' is present in F_o, F_s does not violate any of the two constraints in To_solve. Hence, the algorithm removes them from To_solve. Since,

To_solve is empty, the algorithm does not recurse.

- If the weight function is that of *Min-Size*, then $W_s(F_1)$ = 10 + 5 + 15 = 30 and $W_s(F_2)$ = 7 + 100 = 107. Since $W_s(F_1) < W_s(F_2)$, F_o = F_1 and F_s = F_2.Since the attributes 'ZIP' and 'Job' are present in F_o, similar to the previous case, the algorithm does not recurse.
- If the weight function is that of *Min-Query*, then $W_q(F_1)$ = 10 + 4 + 1 = 15 and $W_q(F_2)$ = 10 + 4 + 5 + 1 + 7 + 7 = 34. Hence, F_o = F_1 and F_s = F_2.
- If the weight function is that of *Min-Cond*, then $W_c(F_1)$ = 10 + 4 + 1 = 15 and $W_c(F_2)$ = 5 + 4 + 10 + 1 + 7 + 7 + 1 = 35. Hence, F_o = F_1 and F_s = F_2.

After adding the singleton constraints to F_o and the attributes that are not involved in *C* to F_s, the fragments F_o and F_s corresponding to the four metrics are as shown below.

- ***Min-Attr:*** F_o = {SSN, Illness, Treatment}; F_s = {DoB, Name, Job, ZIP}.
- ***Min-Size, Min-Query, Min-Cond:*** F_o = {SSN, Job, ZIP, Name}; F_s = {DoB, Illness, Treatment}.

If we run the heuristics proposed in [5][6][7] for the same example, the resulting fragmentation for each metric is as follows.

- ***Min-Attr:*** F_o = {SSN, Illness, Treatment}; F_s = {DoB, Name, Job, ZIP}.
- ***Min-Size:*** F_o = {SSN, Job, ZIP, Name, Illness}; F_s = {DoB, Treatment}.
- ***Min-Query, Min-Cond:*** F_o = {SSN, Job, ZIP, Name}; F_s = {DoB, Illness, Treatment}.

Thus, the proposed scheme has a definite improvement over the existing solutions. It performs at least as good as the current ones and often better.

Further, if we add an additional constraint c_7 = *(I,T)* to the set *C*, then *Construct-graph* adds an edge *(I,T)* to *G* which results in a cycle as shown in Fig. 1(d). Since the number of edges *N* is 3, $\lceil N/2 \rceil$ i.e., two attributes must be stored at the owner. On removing the edges *(I,T)*, *(I,N)* and *(T,N)* 2-graph coloring algorithm returns the fragment pairs $\{\{I,T\},\{N\}\}$, $\{\{I,N\},\{T\}\}$ and $\{T,N\},\{I\}\}$ respectively. Now, considering the weight function corresponding to *Min-Size*, $min\{W_s(I,T), W_s(I,N), W_s(T,N)\}$ = $min\{107,22,115\}$ = 22.

Hence, the algorithm removes *(I,N)* from *G* and adds it to To_solve. The resulting graph is as shown in Fig. 1(e). Then, the algorithm applies 2-graph coloring as shown in Fig. 1(f) which results in the fragments $F_1 = \{Z,T,J\}$ and $F_2 = \{I,N\}$. As we continue with this set further, we obtain the fragments F_o and F_s corresponding to *Min-Size* are $\{$*SSN, ZIP, Job, Illness, Name*$\}$ and $\{$*Treatment, DoB*$\}$ respectively. Clearly, the proposed scheme is efficient as well as effective.

V. Summary and Future Work

Increasingly, organizations are outsourcing their data and computations. However, in order to abide by the privacy laws, they must be careful in what they outsource. While prior research suggested encryption and fragmentation as the tools to achieve privacy, we have only used fragmentation as a mechanism. This is primarily due to the high cost of encryption and decryption as well as the cost of managing the keys. We have extended the work of Ciriani et al. We have employed graph-coloring algorithms to determine which parts of a relation can be outsourced and which need to be kept at the owner.

We are currently extending this work to deal with confidentiality constraints that overlap several relations and constraints with more than 2 attributes. We are also looking into ways of optimally splitting a user query into sub-queries so that the task of query evaluation can be shared suitably between the server and the owner.

Acknowledgment

We dedicate this work to the Founder Chancellor of Sri Sathya Sai Institute of Higher Learning, Bhagawan Sri Sathya Sai Baba.

References

[1] P. Samarati and S. De Capitani di Vimercati. *Data Protection in Out-sourcing Scenarios: Issues and Directions,* in Proc. of ASIACCS 2010, Beijing, China, April 2010.

[2] W. Wang, Z. Li, R. Owens, and B. Bhargava. *Secure and Efficient Access to Outsourced Data,* in Proc. of CCSW 2009, the 2009 ACM Workshop on Cloud Computing Security .

[3] G. Aggarwal, et al. *Two can keep a secret: a distributed architecture for secure database services,* in: Proc. CIDR 2005, Asilomar, CA, USA, January 2005.

[4] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, *Fragmentation design for efficient query execution over sensitive distributed databases,* in: Proc. of ICDCS 2009, Montreal, QC, Canada, June 2009.

[5] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. *Keep a few: Outsourcing data while maintaining confidentiality,* in: Proc. of ESORICS 2009, Saint Malo, France, September 2009.

[6] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. *Enforcing Confidentiality Constraints on Sensitive Databases with Lightweight Trusted Clients,* in: Proc. of DBSec 2009, Montreal, QC, Canada, July 2009.

[7] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. *Selective data outsourcing for enforcing privacy,* in: Journal of Computer Security (2011), 531-566.