IJERST

International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 2, No. 2
May 2013

www.ijerst.com

ISSN 2319-5991 www.ijerst.com
Vol. 2, No. 2, May 2013
© 2013 IJERST. All Rights Reserved

Research Paper

DATA LEAKAGE DETECTION FOR HOSPITAL MANAGEMENT SYSTEM

E A Vimal[1], A Amritha Begam[1]* and P Abinaya[1]

Corresponding Author:* ***A Amritha Begam,* *✉ amrithabegam@yahoo.com*

A data distributor gives the sensitive data to a set of supposedly trusted agents (third parties). And the distributor found that the some of the data are leaked and found in an unauthorized place (e.g., on the web or somebody's laptop). The distributor must identify the leaked data came from which agents. Here we the propose data allocation strategies (across the agents) to improve the probability of identifying the leakages. By means of these methods the released data are not altered (e.g., watermarks). In some cases, we can also insert "realistic but fake" data records to improve the chances of detecting leakage and identifying the guilty party.

Keywords: Distributor, Agent, Target, Allocation strategies, Data leakage, Data privacy, Fake records, Leakage model

INTRODUCTION

In the course of any business, the distributor might give the sensitive data must to the supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Here the owner of the data is considered as the distributor and the supposedly trusted third parties as the agents. Our objective is to identify the agent who leaked the data and also to identify when it was leaked. This application does not support for perturbation. Perturbation is a technique of modifying the original data before handovering it to the third parties. However, in certain cases, the original data is supposedly not to be modified. For example, in case of banking the details like customer account number and name should not be modified. And in case of Hospital, the medical researchers need the exact patient records for analyzing. Earlier, the data leakage detection is done with watermarking, i.e, a unique code is associated with each distributed copy. And if the distributor found that the copy is in any unauthorized place, he may detect that the data is leaked. Watermarks is a useful technique in certain cases but it has the drawback that it may results in the modification of the original data that might leads to future issues. And also with another

[1] Department of IT, Kumaraguru College of Technology, Coimbatore.

drawback is that, watermarks can be easily removed or destroyed if the data recipient is malicious. Here we study the following scenario: After the distributor gives the set of records to the agents. He may found that the record is leaked and found in an unauthorized place. The distributor finds the guilty agent who leaked the data. And he stop doing business from him further. Here, we construct a model for identifying the guilty agents. We also represents certain algorithms for distributing data or records to the agents, in such a way to improve our chances of finding the data leaker. Atlast, we also include the option of inserting “fake” records to the distributed set. These records do not corresponds to real objects but appear to be realistic to the agents. Here, the fake objects act as a type of watermark for the entire set of records, without altering any individual members. If it is found out that an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty and he has leaked the record. We start in Section 2 by introducing our problem setup and the notation we use. In Sections 4 and 5, we present a model for calculating “guilt” probabilities in cases of data leakage. Then, in Sections 6 and 7, we present strategies for data allocation to agents. Finally, in Section 8, we evaluate the strategies in different data leakage scenarios, and check whether they indeed help us to identify a leaker.

In case of hospital, each patient record is kept sensitive. Such that the records should not to be leaked to the third party (target). Therefore we make use of algorithms like Explicit, Sample algorithms to minimize the probability of data getting leaked.

Incase if the distributor found that any of the patient record is leaked, he may use guilt model analysis to identify the guilty agents. Previously we use watermarks technique, i.e., a unique code is embedded in each record to identify the leaked data. Due to some drawbacks in this technique we make use of algorithms.

PROBLEM SETUP AND NOTATION

Entities and Agents

A distributor has a set of records T = {t1, t2, .., tm} of valuable data objects. The distributor needs to share some of these records with a set of agents U1, U2, ..., Un but he does not wants these records to be leaked to other third parties called the target. The objects in T maybe of any type and size, e.g., they might be tuples in a relation, or relations in a database. The distributor distributes any subset of objects Ri belongs to T to any agents. The agent may request for the records to the distributor in any of the following ways:

Sample request Ri = SAMPLE(T,mi): Any subset of mi records from T can be given to Ui.

Explicit request Ri = EXPLICIT (T, condi): Agent Ui receives all T objects that satisfy condition.

Guilty Agents

Suppose after the distributor gives the records to agents, and later he found that any of the records from set S of T has leaked. That is, some of the third party, called the target, has been received any records from the Set S. For example, this target may publish S on its website by means of a legal discovery process, the target turned over S to the distributor.

Since these agents U1, . . . , Un holds certain records from the distributor, it is reasonable to suspect these agents may leaked the records to the target. However, the suspected agents can try to convince the distributor that they are innocent, and that the target obtained the records by through other ways. For example, the hospital may distributes the patient records to certain agents for analyzing. The agent may leaked the data to the third parties. The target may wantedly modify any of the details from the patient records which leads to the severe issue in future.

Our objective of this paper is to discover the leaked data and the agent or guilt who leaked that records. If the distributor set has larger data or records then it is difficult for the agents to assure that they haven't leaked any of the records. Similarly, If the distributor holds only small set of data then it is difficult to argue that the target may received any of these records from other means. Here we not only wanted to find the leaked data, it is also necessary to find out the guilty agents. This may help the distributor not to have further partnership with those guilty agents. Consider, if the distributor distributes any of the S objects to agent U1, and the other objects were given to all agents, the distributor suspect U1 more.

We assume that an agent Ui is guilty and he leaked one or more records to the target. We assure the event that agent Ui is guilty by Gi and the event that agent Ui is guilty for a given leaked set S by Gi|S. Our next step is to estimate Pr{Gi|S}, i.e., the probability that agent Ui is the leaker with given evidence S.

RELATED WORK

The technique of detecting the guilty agent is related to the data provenance problem (Sandip A Kale and Kulkarni, 2012): the lineage of S objects indicates necessary of detecting the guilty agents. Tutorial (Buneman and Tan, 2007) gives fine overview on the research applied in this field. Suggested solutions are domain specific, such as lineage tracing for datawarehouses (Cui and Widom, 2003), and assume some prior knowledge on the way a data view is created out of data sources. Our problem formulation with these data and sets is more general and simplifies lineage tracing, since we do not consider any data transformation from Ri sets to S. As far as the data allocation strategies are concerned, our work is mostly relevant to watermarking that is used as a means of establishing original ownership of distributed objects. Watermarks were initially used in images (Panagiotis Papadimitriou, 2011), video (Hartung and Girod, 1998), and audio data (Czerwinski *et al*., 2007) whose digital representation includes considerable redundancy. Recently, (Agrawal and Kiernan, 2002; Li *et al*., 2005; Guo *et al*., 2006; Shabtai *et al*., 2010), and other works have also studied marks insertion to relational data. Our approach and watermarking are similar in the sense of providing agents with some kind of receiver identifying information. However, by its very nature, a watermark modifies the item being watermarked. If the object to be watermarked cannot be modified, then a watermark cannot be inserted. In such cases, methods that attach watermarks to the distributed data are not applicable. Finally, there are also lots of other works on mechanisms that allow only authorized users to access sensitive data through access control policies (Jajodia *et al*., 2001; Bonatti *et al*., 2002). Such approaches prevent in some sense data leakage by sharing information only with trusted parties. However, these policies are restrictive and may make it impossible to satisfy agents' requests.

AGENT GUILT MODEL

To calculate this Pr{Gi,S} we need to estimate for the probability that records in S can be "guessed" by the target. Consider, that some of the records in S are the e-mails of the individuals.

We can conduct an experiment and ask a person to identify the e-mail of the individuals and if the person can find 90 e-mails, then the probability of identifying one e-mail is 0.9. And the other example is that, if the data set of the distributor has the customer details like bank account numbers. And if the person finds the 20 account numbers, then the probability of identifying the account numbers is 0.2. We call this estimate pt, the probability that object t can be guessed by the target.

Probability pt is similar to the probabilities used in designing fault-tolerant systems. That is, to analyze how likely it is that a system will be operational throughout a given period, we need the probabilities that individual components will or will not fail. A component failure in our problem is the event of identifying the records of S by the target itself. The component failure computes the overall system reliability, while the distributor finds the probability of the data getting leaked. The component failure probabilities are analyzed based on experiments, just as we propose to estimate the pts. Similarly, the component probabilities are usually conservative estimates, rather than exact numbers. Consider, if actual probability is lesser, when comparing with the component failure probability, and the system is to be designed with high level of reliability. Then it is come to know that the actual system possesses the least level of reliability, but possibly higher. In the same way, if pts are used which are larger than the true values, is to be come to know that the agents is "guilt" with the least computed probabilities.

We make the following two assumptions regarding the relationship among the various leakage events. The first assumption defines that an agent's decision of leaking an object is not related with other objects. In [14], we study a scenario where the actions for different objects are related, and we study how our results are impacted by the different independence assumptions.

Assumption 1: For all t belongs to S such that t ! = t', the provenance of t'. The term "provenance" in this assumption statement is independent refers to the source of a value t that founds to be present in the leaked set. The source may be any of the agents who have t in their sets or the target itself (guessing). To simplify these calculation, the following assumption states that joint events have a negligible probability. As we argue in the example below, this assumption gives us more conservative estimates for the guilt of agents, which is consistent with our objective.

Assumption 2: An object t belongs to S can only be obtained by the target in one of the two ways as follows:

- Any agent Ui leaked the data from Ri set.
- The target himself guessed (or obtained through other means) without the help of any of the n agents.

In other words, for all t belongs to S, the event that the target himself guesses any data t without the help of any of the n agents. Assume that the distributor set T, the agent sets Rn, and the target set S are: T = $\{t_1, t_2, t_3\}$, R1 = $\{t_1, t_2\}$, R2 = $\{t_1, t_3\}$, S = $\{t_1, t_2, t_3\}$.

In this case, all the records (objects) of the distributor's set has been leaked and found in any unauthorized place. Initially consider how the target may have received records t1 from the distributor, which was given to both agents. From Assumption 2, the target either guessed t1 or one of U1 or U2 leaked it. We know that the probability of the former event is p, so assuming that probability that each of the two agents leaked t1 is the same, we have the following cases:

- The target guessed t1 with probability p,
- Agent U1 leaked t1 to S with probability (1–p)/2
- Agent U2 leaked t1 to S with probability (1–p)/2

Similarly, if the record t2 is found in S, the agent U1 is purely responsible for that leakage, since he is the only agent holds the record t2. And the probability that U1 leaked t2 is 1 – p. Known all these values, the probability that agent U1 is not guilty, such that U1 did not leak either object, is

$$Pr\{G1|S\} =(1 -(1 -p)/2 *(1-(1-p)) \quad ...(1)$$

and the probability that U1 is guilty is

$$Pr\{G_1,S\}=1-P_r\{G_1\} \quad ...(2)$$

Note that if Assumption 2 did not satisfy, our determination would be more difficult because we would need to consider joint events, e.g., And consider the case if the target identifies the records t1, t2 at the same time, one or two agents who holds the corresponding data may leak the records. In our simplified analysis, we say that an agent is not guilty when the object can be guessed, regardless of whether the agent leaked the value. Since we are "not counting" instances when an agent leaks information, the simplified analysis yields conservative values (smaller probabilities).

With the general assumptions, to find the probability that an agent Ui is guilty given a set S, first, we calculate the probability that he leaks a single object t to S. To calculate this, we define the set of agents Vt = $\{Ui \mid t \in R_i\}$ that have t in their data sets. Then, using Assumption 2 and known probability p, we have the following:

$$Pr \{\text{some agent leaked t to S}\}= 1 - p \quad ...(3)$$

Assuming that all agents who receives the records from the distributor can leak the records t to S with equal probability and using Assumption 2, we gets

$$Pr\{\text{Ui leaked t to S}\}=\{(1-p)/v_{t,} \text{ if } U_i \in V_t \quad ...(4)$$

Assume that agent Ui is guilty and if he leaks at least one of the value to S, with Assumption 1 and Equation (4), we can compute the probability Pr{Gi,S} that agent Ui is guilty:

$$Pr\{Gi,S\}=1-(1-(1-p)/V_t) \quad ...(5)$$

GUILT MODEL ANALYSIS

This model is used to analyze the guilty agents who leaked the records to the target. Here we have two scenarios. And in each scenario we have a target who have received all the records from the distributor, i.e., T = S.

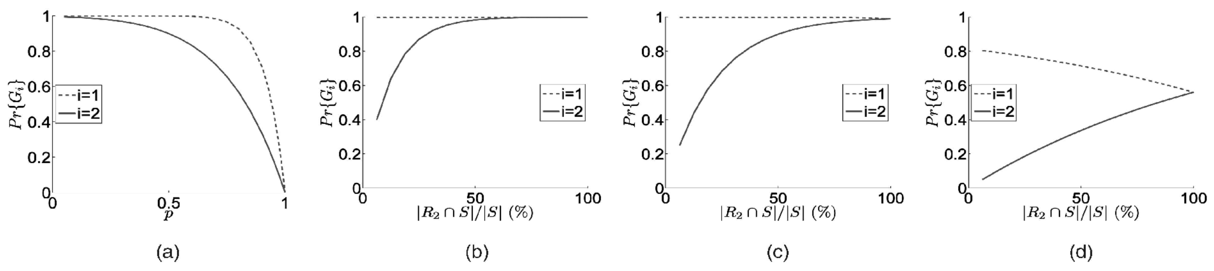
Impact of Probability p

In the first scenario, consider that the distributor set T contains 16 Objects and all of them is given to the agent U_1 and only eight of the objects is given to the agent $U_{2.}$ We calculate the probabilities Pr{G1|S} and Pr{G2,S} for p in the range [0, 1] and we present the results in Figure 1a. The dashed line represents Pr{G1,S} and the solid line represents Pr{G2,S}.

As p approaches to 0, it is assumed that the target is guessed for all 16 values. Each agent has enough of the leaked data that its individual guilt approaches 1. However, as p increases in value, the probability that U2 is guilty decreases significantly: all of U2's eight objects were also given to U1, so it gets harder to blame U2 for the leaks.

On the other hand, U2's probability of guilt

Figure 1: Guilt probability as a function of the guessing probability p (a) and the overlap between S and R2 (b)-(d), in all scenarios, it holds that R1 ∩ S = S and |S|=16, (a) ($|R_2 \cap S|/|S|$) = 0:5, (b) p = 0.2, (c) p =0.5, and (d) p =0.9



remains close to 1 as p increases, since U1 has eight objects not seen by the other agent. At the extreme, as p approaches 1, it is very possible that the target guessed all 16 values, so the agent's probability of guilt goes to 0.

Impact of Overlap Between Ri and S

In this case, we have two agents, one receives all the records from the distributor, i.e., the agent receives all the patient records from the hospital administrator and the second agent receives the varying fraction of data. Figure 1b shows the probability of guilt for both agents, as a function of the fraction of the objects owned by U2, i.e., as a function of |R2)H S|/|S|. In this case, p has a low value of 0.2, and U1 continues to have all 16S objects. Note that in our previous scenario, U2 has 50% of the S objects. We see that when objects are rare (p = 0.2), it does not take many leaked objects before we can say that U2 is guilty with high confidence. This result matches our intuition: an agent that owns even a small number of incriminating objects is clearly suspicious.

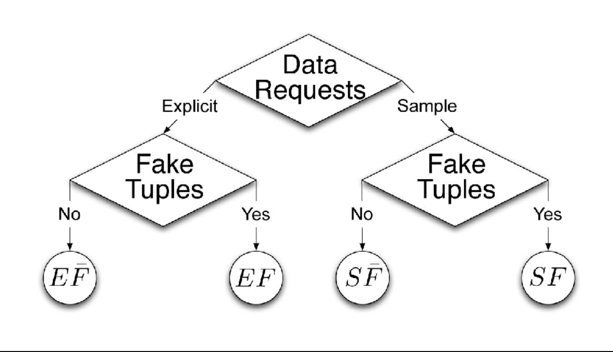
Figures 1c and 1d show the same scenario, except for values of p equal to 0.5 and 0.9. We see clearly that the rate of increase of the guilt probability decreases as p increases. This observation again matches our intuition: As the objects become easier to guess, it takes more and more evidence of leakage (more leaked objects owned by U2) before we can have high confidence that U2 is guilty. In [14], we study an additional scenario that shows how the sharing of S objects by agents affects the probabilities they are guilty. The scenario conclusion matches our intuition: with more agents holding the replicated leaked data, it is harder to lay the blame on any one agent.

DATA ALLOCATION PROBLEM

The main focus of this project is the Data allocation problem. Such that the distributor intelligently distributes the data to the agents. As described in Figure 2, there are four instances of this problem we address, depending on the type of data requests made by agents and whether "fake objects" are allowed. The two types of requests: sample and explicit. Fake objects are objects determined by the distributor that are not in set T. The objects are designed to look like real objects, and are distributed to agents together with T objects, in order to increase the chances of detecting agents that leak data.

As shown in Figure 2, we represent our four problem instances with the names EF, EF', SF, and SF', where E stands for explicit requests, S for sample requests, F for the use of fake objects,

Figure 2: Leakage Problem Instances



and F' for the case where fake objects are not allowed.

Note that, for simplicity, we are assuming that in the E problem instances, all agents make explicit requests, while in the S instances, all agents make sample requests. Our results can be extended to handle mixed cases, with some explicit and some sample requests. We provide here a small example to illustrate how mixed requests can be handled, but then do not elaborate further. Assume that we have two agents with requests R1 = EXPLICIT(T, cond_1) and R2 = SAMPLE (T',1), where T' = EXPLICIT(T, cond2). Further, say that cond1 is "state = CA" (objects have a state field). If agent U2 has the same condition $\text{cond}_2 = \text{cond}_1$, we can create an equivalent problem with sample data requests on set T'. That is, our problem will be how to distribute the CA objects to two agents, with R1 = SAMPLE (T',|T|) and R2 = SAMPLE (T', 1). If instead U2 uses condition "state =NY," we can solve two different problems for sets T0 and T – T'. In each problem, we will have only one agent. Finally, if the conditions partially overlap, R1 $\cap$ T' $\neq \Phi$, but R1 $\neq$ T', we can solve three different problems for sets R1 – T', R1 $\cap$ T', and T'- R1.

Fake Objects

The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in finding the guilty agents. However, fake objects may impact the correctness of what agents do, so they may not always be allowable. In our case, we are altering the set of distributor records by adding fake records. In some applications, fake objects may results in fewer issues that altering real records. For example, say that the distributed data objects are medical records and the agents are hospitals. In this case, even small modifications to the records of actual patients may be undesirable. However, the addition of some fake medical records may be acceptable, since no patient matches these records, and hence, no one will ever be treated based on fake records.

Creation: Here, we model the creation of a fake object for agent Ui as a black box function CREATEFAKEOBJECT(Ri, Fi, condi) which takes the following as the input ,the set of all records Ri, the subset of fake records Fi that Ui has received so far, and condi of the explicit request sent by the agent, and returns a new fake object. This function needs condi to produce a valid object which satisfies Ui's condition. Set Ri is needed as input so that the created fake object is not only valid but also indistinguishable from other real objects. CREATEFAKEOBJECT() has to be aware of the fake objects Fi added so far, again to ensure proper statistics. The distributor can also use function CREATEFAKEOBJECT() when the distributor needs to send the same fake records to a set of agents. In this case, the function arguments are the union of the Ri and Fi tables, respectively, and the intersection of the conditions.

Optimization Problem

When the distributor allocates data to agents he should satisfy a constraint and an objective. The constraint of the distributor is to satisfy agent's requests, such that the distributor should provide the number of objects that the agents request or providing the objects which satisfies agent's constraint. And the objective of the distributor is to detect the guilty agent who has leaked his data to the third parties. The constraint is strictly considered. The distributor may not deny serving an agent request as in [13] and may not provide agents with different perturbed versions of the same objects as in [1]. We consider fake record distribution as the only possible constraint relaxation. Our detection objective is ideal and intractable. Detection would be assured only if the distributor gave no data object to any agent (Mungamuru and Garcia-Molina [11] discuss that to attain "perfect" privacy and security, we have to sacrifice utility). We use instead the following objective: maximize the chances of detecting a guilty agent that leaks all his data objects. We now introduce some notation to state formally the distributor's objective $Pr\{G_i|S=R_i\}$ or simply $Pr\{G_{j,}R_j\}$ is the probability that agent Uj is guilty if the distributor discovers a leaked table S that contains all Ri objects. We define the difference functions $\Delta(i,j)$ as

$$\Delta(i,j)=P_r\{G_i|R_j\}-P_r\{G_j|R_i\} \quad i,j=1,\ldots,n \qquad ...(6)$$

Note that differences have nonnegative values: given that set Ri contains all the leaked objects, agent Ui is at least as likely to be guilty as any other agent. Difference $\Delta(i,j)$ is positive for any agent Uj, whose set Rj does not contain all data of S. It is zero if Ri not subset of Rj. In this case, the distributor will assume that both the agents Ui and Uj are guilty with equal probability. Since they both have received all the leaked objects. The larger a $\Delta(i,j)$ value is, the easier it is to identify Ui as the leaking agent. Thus, we want to distribute data so that Δ values are large.

Problem Definition: Let the distributor (Hospital admin) have data requests from n agents. The distributor wants to give the patient records R1, ..., Rn to agents U1, ..., Un, respectively, so that

a. He satisfies agents' requests, and

b. He maximizes the guilt probability differences $\Delta(i,j)$ for all i, j = 1, ..., n and $i \neq j$.

Assuming that the Ri sets satisfy the agents' requests, we can express the problem as a multicriterion optimization problem:

$$\text{Maximize}(..., \Delta(i,j), \ldots) \quad i \neq j \qquad ...(7)$$

If the optimization problem has an optimal solution, it means that there exists an allocation $D^* = \{R^*_1, \ldots, R^*_n\}$ such that any other feasible allocation D = {R1, ..., Rn} yields $\Delta(i, j) \geq \Delta^*(i, j)$ for all i, j. This means that allocation D^* allows the distributor to discern any guilty agent with higher confidence than any other allocation, since it maximizes the probability Pr{Gi,Ri} with respect to any other probability Pr{Gi,Rj} with $j \neq i$. Even if there is no optimal allocation D^*, a multicriterion problem has Pareto optimal allocations. If $D^{po} = \{R_1^{po}, ..., R_n^{po}\}$ is a Pareto optimal allocation, it means that there is no other allocation that yields $\Delta(i, j) \geq \Delta^{po}(i, j)$ for all i, j. In other words, if an allocation yields $\Delta(i,j) \geq \Delta^{po}(i, j)$ for some i, j, then there is some i', j'such that $\Delta(i', j') \geq \Delta^{po}(i', j')$. The choice among all the Pareto optimal allocations implicitly selects the agent(s) we want to identify.

Objective Approximation

We can approximate the objective of Equation (7) with (8) that does not depend on agents' guilt probabilities, and therefore, on p.

$$\text{Maximize}\left(\ldots,\frac{|R_i \cap R_j|}{|R_i|},\ldots\right) \qquad i \neq j \qquad \ldots(8)$$

This approximation is valid if minimizing the relative overlap $\frac{|R_i \cap R_j|}{|R_i|}$ maximizes Δ(i, j). The intuitive argument for this approximation is that the fewer leaked objects set Rj contains, the less guilty agent Uj will appear compared to Ui (since S = Ri). The example of Section 5.2 supports our approximation. In Figure 1, we see that if S =R1, the difference Pr{G_1|S}-P_r{G_2|S} decreases as the relative overlap $\frac{|R_2 \cap s|}{|s|}$ increases.

***Theorem 1** shows that a solution to (7) yields the solution to (8) while every record is allocated to same number of agents but does not depend on whether the agent leaked the data or not . The proof of the theorem is in [14].*

Theorem 1 If a distribution D = {R1, . . .,Rn} that satisfies agents' requests minimizes and |V_t|= |Vt^1| for all t, t' ∈ T, then D maximizes Δ(i, j).

These optimization problem still have multiple criteria and it can results in either an optimal or multiple Pareto optimal solutions. Pareto optimal solutions let us detect a guilty agent Ui with high confidence, at the expense of an inability to detect some other guilty agent or agents. Since the distributor has no priority information for the agents' intention to leak their data, he has no reason to bias the object allocation against a particular agent. Therefore, we can scalarize the problem objective by assigning the same weights to all vector objectives. We present two different scalar versions of our problem in (9a) and (9b). In the rest of the paper, we refer to objective (9a) as the sum-objective and to objective (9b) as the max-objective:

$$\text{Maximize } \sum_{i=1}^{n}\left(\left(\frac{1}{R_i}\right)\sum_{j=1}^{n}\left(|R_i \cap R_j|\right)\right) \qquad \ldots(9a)$$

$$\text{Maximize } \max(i, j)\frac{|R_i \cap R_j|}{|R_i|} \qquad \ldots(9b)$$

If such a solution exists, then the scalar optimization problems yield the optimal solution of the problem of (8). The sum-objective solution yields pareto optimal solution, If there is no global optimal solution exists, which allows the distributor to detect the guilty agent, on average (over all different agents), with higher confidence than any other distribution. The distributor will detect the guilty agent with a certain confidence in the worst case by means of this max-objective, which may adversely impact the average performance of the distribution.

ALLOCATION STRATEGIES

We describe allocation strategies that solve exactly or approximately the scalar versions of (8) for the different instances presented in Figure 2. We resort to approximate solutions in cases where it is inefficient to solve accurately the optimization problem. In Section 7.1, we deal with problems with explicit data requests, and in Section 7.2, with problems with sample data requests. The proofs of the theorems that are stated in the following sections are available in (Sion *et al*., 2003).

Explicit Data Requests

In problems of class EF', the fake record is not added by the distributor while allocating records to the agents. Hence, the allocation of data is fully defined by the agent's data requests. Therefore, there is nothing to optimize. In EF problems, the fake records are added while allocating data to agents and the objective values are initialized by agent's data requests. Say, for example, that T = {t_1, t_2} and there are two agents with explicit data requests such that R1 = {t1, t2} and R2 = {t1}. The value of the sum objective is in this case

$$\sum_{i=1}^{2} 1/R1 \sum_{i=1}^{2} |Ri \cap Rj| = \left(\frac{1}{2}+\frac{1}{2}\right) = 1.5$$

The Records of R1 and R2 cannot be removed or altered to minimize the overlap R1 $\cap$ R2. Somehow, consider that the distributor can create a fake record (B = 1) and both agents can receive one fake object (b1 = b2 = 1). In this case, a single fake record can be added to either R1 or R2 by the distributor to maximize the corresponding denominator of the summation term. Assume that the distributor creates a fake object f and he gives it to agent Agent U1 has now R1 = {t1,t2, f} and F_1 = {f} and the value of the sum objective decreases to (1/3) + (1/1) = 1.33 < 1.5.

If the distributor is able to create more fake objects, he could further improve the objective. We present in Algorithms 1 and 2a strategy for randomly allocating fake objects. Algorithm 1 is a general "driver" that will be used by other strategies, while Algorithm 2 actually performs the random selection. We denote the combination of Algorithm 1 with 2 as e-random. We use e-random as our baseline in our comparisons with other algorithms for explicit data requests.

Algorithm 1: Allocation for Explicit Data Requests (EF)

Input: R1,...,Rn, cond1, . . . , condn, b1,. . . , bn, B

Output: R1, ..., Rn, F1, ..., Fn

```
1: R←□Agents that can receive fake objects
2: for i = 1, ..., n do
3: if bi > 0 then
4: R←R U {i}
5: Fi←□
6: while B > 0 do
7: i SELECTAGENT(R,R1, . . .,Rn)
8: f ← CREATEFAKEOBJECT(Ri, Fi, condi)
9: Ri ← Ri U {f}
10: Fi ← Fi U {f}
11: bi ← bi -1
12: if bi =0then
13: R ← R\{Ri}
14: B ← B – 1
```

Algorithm 2: Agent Selection for e-random

```
1: function SELECTAGENT (R, R1, ..., Rn)
2: i select at random an agent from R
3: return i.
```

In lines 1-5, Algorithm 1 finds agents that are eligible to receive fake objects in O(n) time. Then, in the main loop in lines 6-14, the algorithm creates one fake object in every iteration and allocates it to random agent. The algorithm minimizes every term of the objective summation by adding the maximum number bi of fake objects to every set Ri, yielding the optimal solution. The algorithm just selects at random the agents that are provided with fake objects. We return back to our example and see how the objective would change if the distributor adds fake object f to R2 instead of R1. In this case, the sum-objective would be

(1/3) + (1/1) = 1.33 < 1.5

The reason why we got a greater improvement is that the addition of a fake object to R2 has greater impact on the corresponding summation terms, since(1/R1)–(1/|R1|+1)=1/6<(1/R2)–(1/|R2|+1)=1/2.

The left-hand side of the inequality corresponds to the objective improvement after the addition of a fake object to R1 and the right-hand side to R2. We can use this observation to improve Algorithm 1 with the use of procedure SELECTAGENT() of Algorithm 3. We denote the combination of Algorithms 1 and 3 by e-optimal.

Algorithm 3: Agent Selection for e-optimal

```
1: function SELECTAGENT (R,R1, . . .,Rn)
2: i ← argmax $\left(\frac{1}{Ri'} - \frac{1}{|Ri'|+1}\right)^{n} \sum_{j}^{i} (Ri \cap Rj)$
3: return i
```

Algorithm 3 makes a greedy choice by selecting the agent that will yield the greatest improvement in the sum objective.

Sample Data Requests

In this type of request the agents does not mention any constraints (Cond), Instead they mention about the number of patient records they need (m) with the size the agent may receive any number of records. Here the guilty probability depends on which agent has received the leaked object. In this method we have two algorithms namely random and s-random.

Note that the distributor can increase the number of possible allocations by adding fake objects (and increasing |T| but the problem is essentially the same. So, in the rest of this section, we will only deal with problems of class SF', but our algorithms are applicable to SF problems as well.

Random

An object allocation that satisfies requests and ignores the distributor's objective is to give each agent Ui a randomly selected subset of T of size mi. We denote this algorithm by s-random and we use it as our baseline. We present s-random in two parts: Algorithm 4 is a general allocation algorithm that is used by other algorithms in this section. In line 6 of Algorithm 4, there is a call to function SELECTOBJECT() whose implementation differentiates algorithms that rely on Algorithm 4. Algorithm 5 shows function SELECTOBJECT() for s-random.

Algorithm 4: Allocation for Sample Data Requests

Input: m1,…m2….,mn,|T|

Output: R1,R2,…,Rn

```
1: a←$0_{|T|}$,a[k]-number of agents who have received the object $t_k$
2: R1←□,….,Rn←□
3: remaining←”$_{i=1}m_i$
4: while remaining > 0 do
5: for all i = 1,….,n:|Ri|<$m_i$ do
6: k←SelectObject(i,$R_i$)
7: $R_i$←$R_i$ U {$t_k$}
8: a[k]←a[k]+1
9: remaining←remaining-1
```

Algorithm 5: Object Selection for s-random

```
1: function SELECTOBJECT(i,$R_i$)
2: k ← select at random an element from set
    {k'|$t_{k'}$ not belongs to Ri}
3: Return k
```

In s-random, we introduce vector a ∈ $N^{|T|}$ that shows the object sharing distribution. In particular, element a[k] shows the number of agents who receive object tk. Algorithm s-random allocates objects to agents in a round-robin fashion. After the initialization of vectors d and a in lines 1 and 2 of Algorithm 4, the main loop in lines 4-9 is executed while there are still data objects (remaining > 0) to be allocated to agents. In each iteration of this loop (lines 5-9), the algorithm uses function SELECTOBJECT() to find a random object to allocate to agent Ui. This loop iterates over all agents who have not received the number of data objects they have requested.

The s-random algorithm allocates the objects to agents in a round-robin fashion. In each iteration the algorithm uses selectobject(). Here the algorithm yields poor data allocation, since it provides all the agent with the same object.

The running time of the algorithm is O(T) and depends on the running time of the object selection function SELECTOBJECT(). In case of random selection, we can have T=O(1) by keeping in memory a set $\{k' \mid t_k \text{ F } R_i\}$ for each agent Ui. Algorithm s-random may yield a poor data allocation. Consider, for example, initially the distributor has three sets of records and each record is requested by the three agents. Using s-random, allocates same record with the all three agents. Therefore such an allocation increases both objectives (9a) and (9b) instead of minimizing them.

Overlap Minimization

From the final example, the distributor can minimize the sum objective and Maximize objective by providing unique sets to all three agents. Thus the allocation is possible, because the agent's request of the records is lesser than the records holded by the distributor, such an allocation is achieved by using Algorithm 4 and SELECTOBJECT() of Algorithm 6. The resulting algorithm is denoted by s-overlap. We use Algorithm 6, in every iteration of Algorithm 4, we allocate agent Ui with an record that has been given to the smallest number of agents. Therefore, if agents request for lesser records than |T|, Algorithm 6 will return that no agent has received the record so far. Therefore, that record will be allocated to the agent.

Algorithm 6: Object Selection for s-overlap

```
1: function SELECTOBJECT (i,Ri,a)
2: K ← {k| k=argmin a[k']}
3: K ← select at random an element from set
    {k' | k ∈ k□ t_k' F R_i}
4: return k.
```

CONCLUSION

To avoid the leakage, the distributor need not want to handover sensitive data to agents who might leak the data to the third parties. Even if the distributor needs to send the data to the agent. Traditionally we could watermark each record inorder to find the data getting leaked. Somehow, in certain situation, we are supposed to work with agents who are not be 100% trusted. And also in some cases use of watermark results in altering the original data. In case of hospital management system, the patient record is very sensitive and should not be leaked to third parties.

In this paper we proposed the data allocation strategies to allocate the objects to the desired agents. And also we use various algorithms to improve the techniques of identifying a leaker. The various algorithm to identify the leaked objects and the guilty person is identified. Comparing to Explicit request algorithm, the sample request algorithm provides poor data allocation techniques. The use of these algorithm minimizes the probability of data getting leaked to the third party.

REFERENCES

1. Agrawal R and Kiernan J (2002), "Watermarking Relational Databases", Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02), VLDB Endowment, pp. 155-166.
2. Bonatti P, di Vimercati S D C and Samarati P (2002), "An Algebra for Composing Access Control Policies," *ACM Trans. Information and System Security*, Vol. 5, No. 1, pp. 1-35.
3. Sandip A Kale and Kulkarni S V (2012), Department of Computer Sci. & Engg, MIT College of Engg, Dr.B.A.M.University, Aurangabad (M.S), India, Data Leakage Detection: A Survey (IOSR *Journal of Computer Engineering* (IOSRJCE), ISSN : 2278-0661, Vol. 1, No. 6, pp. 32-35, available at www.iosrjournals.org
4. Buneman P and Tan W C (2007), "Provenance in Databases", Proc. ACM SIGMOD, pp. 1171-1173.
5. Cui Y and Widom J (2003), "Lineage Tracing for General Data Warehouse Transformations", *The VLDB J.*, Vol. 12, pp. 41-58.
6. Czerwinski S, Fromm R and Hodes T (2007), "Digital Music Distribution and Audio Watermarking", http://www.scientific commons.org/43025658.
7. Guo F, Wang J , Zhang Z, Ye X and Li D (2006), "An Improved Algorithm to Watermark Numeric Relational Data," *Information Security Application*s, pp. 138-149, Springer.
8. Hartung F and Girod B (1998), "Watermarking of Uncompressed and Compressed Video," *Signal Processing*, Vol. 66, No. 3, pp. 283-301.
9. Jajodia S, Samarati P, Sapino M L and Subrahmanian V S (2001), "Flexible Support for Multiple Access Control Policies", *ACM Trans. Database Systems*, Vol. 26, No. 2, pp. 214-260.
10. Li Y, Swarup V and Jajodia S (2005), "Fingerprinting Relational Databases: Schemes and Specialties," *IEEE Trans. Dependable and Secure Computing,* Vol. 2, No. 1, pp. 34-45.
11. Mungamuru B and Garcia-Molina H (2008), "Privacy, Preservation and Performance: The 3 P's of Distributed Data Management," Technical Report, Stanford Univ.
12. Murty VN (1981), "Counting the Integer Solutions of a Linear Equation with Unit Coefficients," *Math. Magazine*, Vol. 54, No. 2, pp. 79-81.
13. Nabar S U, Marthi B, Kenthapadi K, Mishra N and Motwani R (2008), "Towards Robustness in Query Auditing," *Proc. 32nd Int'l Conf. Very Large Data Bases* (VLDB '06), VLDB Endowment, pp. 151-162.
14. Panagiotis Papadimitriou (2011), "Student Member, IEEE, and HectorGarcia-Molina, Member, Data Leakage Detection IEEE", pp. 2-6, IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No. 1.
15. Rudragouda G Patil (2011), Dept of CSE, The Oxford College of Engg, Bangalore, *International Journal of Computer Applications in Engineering Sciences*, Vol.

I, No. II, ISSN: 2231- 4946, pp. 1, 4, Development of Data Leakage Detection Using Data Allocation Strategies.

16. Sion R, Atallah M and Prabhakar S (2003), "Rights Protection for Relational Data," *Proc. ACM SIGMOD*, pp. 98-109.

17. Sweeney L (2002), "Achieving K-Anonymity Privacy Protection Using Generalization and Suppression," http://en.scientificcommons.

18. Shabtai A, Gershman A, Kopeetsky M, Elovici Y (2010), "DeutscheTelekom Laboratories at Ben-Gurion University, Israel", Technical Report TR-BGU-2409-2010 24 Sept. 2010 1 A Survey of Data Leakage Detection and Prevention Solutions, pp. 1-5, 24-25.

International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com

ISSN 2319-5991
9 772319 599001 02