International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Special Issue 31(September, 2015), PP. 22-28

AGENT GUILT MODEL AND FAKE OBJECT DISTRIBUTION FOR IDENTIFYING DATA LEAKAGE

Shruti D. Meshram[#1], H. K. Chavan[#2]
[#]Information Technology, Terna Engineering College, Mumbai University
Sector-22, Nerul, Navi Mumbai, Maharashtra, India
[1]shrutimeshram143@yahoo.in
[2]chavan.hari@gmail.com

***Abstract*— Data Leakage is one of the major concerns of the corporate world today. When a company outsources its confidential data to a third party e.g. to a service provider, it may be the case that some of the data are leaked and found in an unauthorized place (e.g., on the web or somebody's laptop). This would cause a large setback to the company. Thus Data Leakage Detection needs to be done in order to find out the guilty party. In the proposed system an Agent Guilt Model is developed wherein the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. In order to increase the chances of detecting the data leakage, the proposed system also deals with the addition of dummy objects to the real objects.**

***Keywords*— leakage detection, guilty party, Agent Guilt.**

I. INTRODUCTION

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the distributor and the supposedly trusted third parties the agents. Our goal is to detect when the distributor's sensitive data has been leaked by agents, and if possible to identify the agent that leaked the data.

We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents. For example, one can add random noise to certain attributes, or one can replace exact values by ranges [12]. However, in some cases it is important not to alter the original distributor's data. For example, if an outsourcer is doing our payroll, he must have the exact salary and customer identification numbers. If medical researchers will be treating patients (as opposed to simply computing statistics), they may need accurate data for the patients.

Traditionally, leakage detection is handled by watermarking, [2], [6], [7], [8], e.g., a unique code is embedded in each distributed copy. If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious.

In this paper we study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a web site, or may be obtained through a legal discovery process.) At this point the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. In this paper we develop a model for assessing the "guilt" of agents.

This paper is organized as follows: The next section presents the related work. In section 3 we introduce problem setup and notation. In section 4 and 5we present the Agent Guilt Model and its analysis. In section 6 we present data allocation problem. Finally, the conclusion is presented.

II. RELATED WORK

Being one of the common problems worldwide, Data Leakage Detection has been addressed by many authors, since 1996. Researches done by different authors on Data Leakage Detection are:

As Watermarking is one of the useful techniques which is used traditionally, rights management of relational data can be done through watermarking [2]. The basic idea is to ensure that some bit positions for some of the attributes of some of the tuples contain specific values. The tuples, attributes within a tuple, bit positions in an attribute, and specific bit values are all algorithmically determined under the control of a private key known only to the owner of the relation. This bit pattern constitutes the watermark. Only if one has access to the

private key, the watermark is detected with high probability. Similarly, marks insertion to relational data is an effective way for right protection of relational data [3].

The proposed guilt detection approach is somewhat related to the data provenance problem. The data provenance problem i.e. tracing the lineage of objects implies essentially finding out the source of the data [4]. A good overview on the research conducted in this field is provided in [5] which include the points such as Importance of data Provenance, Overview of Provenance, Applications of Provenance, Other emerging applications.

Lineage tracing for general data warehouse transformations can be done in an efficient way, which reduces over all tracing cost, storage and runtime overhead [6]. In the proposed system the problem formulation with objects and sets is more general and simplifies lineage tracing, since there is no consideration of any data transformation.

Digital music distribution and audio watermarking can be done by adding a bit stream in digital audio with a secret key [7]. This secret key is known only to the water marker. It is used to hide the bit stream so that no other party can locate the watermark in the digital audio. To recover the watermark, this secret key is used along with the watermarked audio.

Similarly, author Jen-Sheng Tsai, *et al* [8], have proposed a robust digital image watermarking method to achieve the goal of image content authentication and copyright protection.

Likewise, author F. Hartung*etal*[9], have presented watermarking of uncompressed and compressed video by adding some noise signal to each video frame to obtain a watermarked video frame. All three watermarks: audio [7], images [8] and video [9] data includes considerable redundancy in their respective digital representation.

Author Yingjiu Li, *et al* [10] has provided schemes for fingerprinting relational databases. In fingerprinting scheme a buyer-specific mark is embedded into a data copy that is to be provided to a buyer. So Watermarking and Fingerprinting have different goals and hence different schemes.

There are also lots of other works on mechanisms that allow only authorized users to access sensitive data through access control policies specified by author Sushil Jajodia, *et al* [11]. 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 agent's requests.

III. PROBLEM SETUP AND NOTATION

A. Entities and Agents

Set $S=\{s_1,s_2,s_3\ldots\ldots\}$ of valuable data object is owned by the distributor. There are set of agents say $A_1,A_2,\ldots A_n$ with whom the distributor wants to share some objects i.e. an agent A_i receives a subset of objects O_i that belong to S. The objects in S could be of any type and size, e.g., they could be tuples in a relation, or relations in a database. The distributor hopes that the agents do not leak that objects to any other third party.

B. Guilty Agents

Suppose after the distribution of objects to agents, the distributor find outs that some subset of S, say L has been leaked. This means that the distributor caught, some third party owns S let us call this third party the target. For example, this target may be displaying Lon its website, or perhaps as part of a legal discovery process, the target turned over L to the distributor or the distributor found L on targets laptop. Now since the distributor has share some objects with different agents $A_1\ldots\ldots A_n$ so it is likely to suspect them, leaking the data. Here the problems of the distributor do not stop as the agents can forbid the suspicion telling that they are blameless n not guilty, and that the L data was obtained by the target through other means. For example, say one of the objects in L represents a customer C. Perhaps C is also a customer of some other company, and that company provided the data to the target. Or it might be possible that can be reconstructed from various publicly available sources on the web.

Our Moto is to find out the likelihood that the target has obtained the leaked data from the agents instead, from other sources. Intuitively, the more data in L, the more difficult it is for the agents to deny that they leak anything. Similarly, the "rarer" the objects, the harder it is to argue that the target obtained them through other means. Finding out the likelihood that the agent leaked data is not only the mo to, but we would also like to find out if one of them in particular was more likely to be the leaker. For instance, if one of the L objects was only given to agentA_1, while the other objects were given to all agents, we may suspect A_1 more. The model we present next captures this intuition.

We say an agent A_i is guilty if it contributes one or more objects to the target. We denote the event that agent A_i is guilty for a given leaked set L by Gi|L. Our next step is to estimate $P_b\{Gi|L\}$, i.e., the probability that agent A_i is guilty given evidence L.

IV. AGENT GUILT MODEL

Agent Guilt Model, as specified in [1], is used to assess the likelihood that leaked data came from one or more agents, by finding out the probability of guiltiness of an agent.

As mentioned earlier, while detecting the guilty agents we will also consider that the target may obtained the leaked data through other sources and that none of the agents have leaked the data. So, we need an estimate for the Probability that values in L can be "guessed" by the target, while computing

the $P_b\{Gi|L\}$. We call these estimate e_s, the probability that object scan be guessed by the target.

In the rest of the paper, we assume that all S objects have the same e_s, which we finally call e, this is to simplify the formulas that we present. Our equations can be easily generalized to diverse e_s's though they become cumbersome to display.

Assumption: An object sЄL can only be obtained by the target in one of two ways:
A single agent A_i leaked s from his own O_i set; or The target guessed (or obtained through other means) s without the help of any of the n agents. In other words, for all sЄL, the event that the target guesses and the events that agent A_i (i = 1…… n) leaks object s are disjoint.

Next, we provide a simple example, before we present the general formula for computing $P_b\{Gi|L\}$. Assume that sets S, O's and L are as follows:
$S= \{s_1, s_2, s_3\}$, $O1= \{s_1, s_2\}$, $O2 = \{s_1, s_3\}$, $L= \{s_1, s_2, s_3\}$
In the above example, as L contains all three of the distributor's objects that means it is clear that all three of the distributor's objects have been leaked. Also it is clear that the object s_1 is given to both the agents. Let us first consider how the target may have obtained objects$_1$. From Assumption, the target either guessed s_1 or one of A_1 or A_2 leaked it. We know that the probability of the former event is e, so assuming that the probability that each of the two agents leaked s_1 is the same we have the following cases:

- the leaker guessed s_1 with probability e;
- agent A_1 leaked s_1 to L with probability (1-e)/2
- agent A_2 leaked s_1 to L with probability (1-e)/2

Similarly, we find that agent A_1 leaked s_2 to L with probability (1-e) since it is the only agent that has this data object. Given these values, the probability that agent A_1 is not guilty, namely that A_1 did not leak either object is:

$$P_b\{\bar{G}_1|L\}= (1 -(1 -e)/2) *(1 -(1 -e)) \quad (1)$$

Hence, the probability that A_1 is guilty is:

$$P_b\{G_1|L\}= 1 –P_b\{\bar{G}_1|L\} \quad (2)$$

In the general case (with our assumption), to find the probability that an agent A_i is guilty given a set L, first we compute the probability that he leaks a single object s to L. To compute this we define the set of agents $W_s= \{A_i|sЄO_i\}$ that have sin their data sets. Then using Assumption and known probability e, we have:

$$P_b\{\text{some agent leaked sto L}\}= 1 –e. \quad (3)$$

Assuming that all agents that belong to W_s can leak s to L with equal probability and using Assumption we obtain:

$$P_b\{A_i \text{ leaked s to L}\}=\begin{cases}(1-e)/|Ws|, & \text{if Ai ЄWs}\\ 0, & otherwise\end{cases} \quad (4)$$

Given that agent A_i is guilty if he leaks at least one value to L, with Assumption and Equation 4 we can compute the probability $P_b\{Gi|L\}$that agent Ai is guilty:

$$P_b\{Gi|L\}= 1 –\prod_{s\in L\cap Oi} (1 - (1 - e)/|Ws|) \quad (5)$$

V. GUILT MODEL ANALYSIS

The Agent Guilt Model helps the distributor to analyse the probability that an agent G_i is guilty given some evidence L i.e. nothing but $P_b\{Gi|L\}$.Next we present the analysis of the Agent Guilt Model by considering one of the parameter $W_s$and its impact on P_b.

A. Impact of W_s on $P_b\{Gi\}$

Figure 1, presents the impact of distribution of single object s which belongs to L (set of leaked data).The x-axes represents Ws i.e. number of agents that have s in there data set. Here we have taken max value 50 and the y-axes represent the guilt probability, 1 is the highest probability. Looking at the graph, we can say, more the number of agents that have s in there data set (i.e. O_i) less will be the probability of finding the guilty agent. The lesser the distribution of s the more will be probability of finding the guilty agent. So in order to keep the probability of finding guilty agent high, the distributor must consider some way of distributing the data objects to the different agents so that there are less overlaps which leads to increase the chances of detecting the guilty agent.

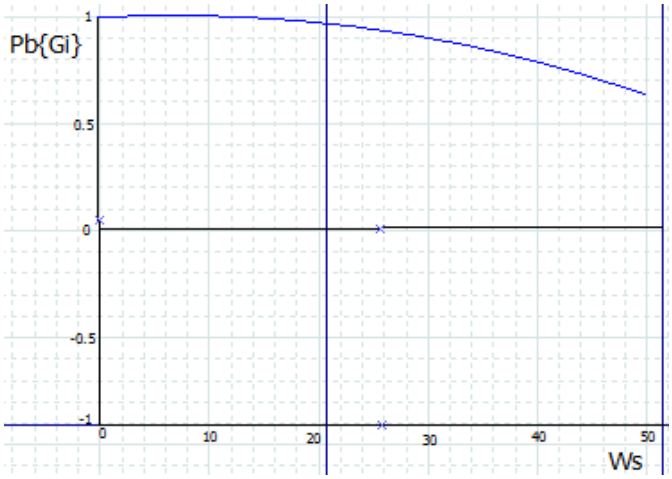


Fig. 1

VI. DATA ALLOCATION PROBLEM

This paper also focuses on the data allocation problem: In order to improve the chances of detecting a guilty agent, how can the distributor "intelligently" give data to agents. As illustrated in Figure. 2, we address four instances of this

problem, depending on the type of data requests made by agents and whether "fake objects" are allowed [1].

The two types of requests we handle are: sample and explicit.

1) Sample Request: Oi= SAMPLE(S,mi): Any subset of mi records from S can be given to Ai.

2) Explicit Request: Oi= Explicit(S,cond): Agent Ai receives all S objects that satisfy condition specified by Ai himself.

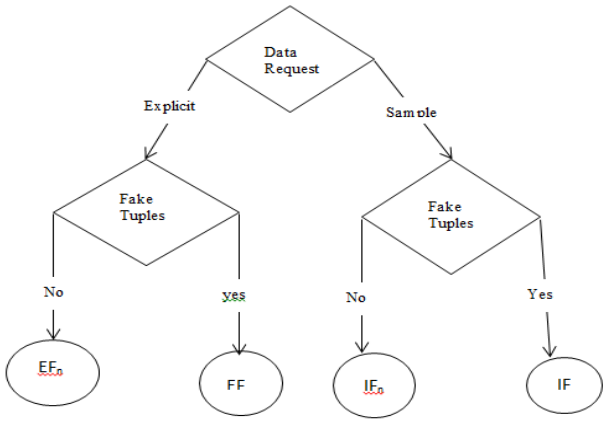


Fig. 2 Leakage problem instances [1].

The objects created by the distributor, which look like the real objects, that do not really belong to S but are distributed to the agents together with the objects in S are addressed as "Fake objects" in this paper. In order to increase the chances of detecting agents that leak data, the distributor injects fake object along with the real objects. We discuss fake objects in more detail in Section 6.1.

As shown in Figure. 2, we represent our four problem instances with the names EF, EF_n, IF, and IF_n, where E stands for explicit requests, I for sample requests, F for the use of fake objects, and F_n 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 I instances, all agents make sample requests. If the agents makes sample request specifying only the number of records then it is totally on the distributor whom to give which record intelligently so that there will be less overlap. If the request is explicit, the agents will provide the records that satisfy the particular condition, say for example: assume that there are two agents A_1 and A_2 .A_1 make requestO_1=EXPLICIT(S, cond1) and request made by A_2 is O_2=EXPLICIT(S, cond2).Say cond1 is "city=Mumbai" and cond2 is "city=Pune" so in this case cond1≠cond2 we can solve the problem by providing the respected data to both the agents from the set S. The problem occurs if cond1=cond2, that is our problem will be how to distribute the same objects to two agents.

A. Fake Objects

In order to improve the effectiveness in detecting the guilty agents the distributor may add fake objects to the distributed data. However, due to fake objects there may be impact on the accuracy of what agents do, so it may not be always applicable.

Data leakage detection by perturbing data is a well-known approach. However, in most cases, individual objects are perturbed, e.g., by adding random noise to sensitive salaries, or adding a watermark to an image. In our case, we are perturbing the set of distributor objects by adding fake elements. In some applications, fewer problems may be caused by using fake objects than perturbing real objects. For example, say the agents are hospitals so it is understood that the data objects that the distributor needs to distribute will be medical records In this case, the traditional perturbation techniques (eg. watermarking or addition of noise) are not applicable since even minute changes in the records of actual patients are undesirable and will have major impact on the patient's treatment. 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.

The distributor creates and adds fake objects to the data that he distributes to agents. Let agent A_i receives O_i along with the set of fake objects say F_i which is a subset of O_i. The fake objects creation must be such, that it is indistinguishable from the real objects to the agent; this is discussed below in details.

In many cases, the distributor may have constrain on the number of fake objects that he can create. For example, objects to be distributed may hold e-mail addresses, simply creating and adding a fake object which is an e-mail address is very likely for the agent to discover that it is a fake one. An e-mail address has its inbox therefore each fake e-mail address may require the creation of an actual inbox. To detect whether there is leakage, the distributor has to actually monitor the inboxes. If the distributer gets an e-mail from someone else (other than the agent who was given the address), it is obvious that leakage of address took place. Thus distributor may have a limit of fake objects as they may consume resources, as in the case of e-mail accounts. We denote limit by B fake objects. Similarly, we can say an agent A_i can receive up to b_i fake objects. This means the distributor may also have constrain on the distribution of fake objects to individual agent. This is to reduce the impact of fake objects on the agent's task and the suspicion of agent on fake objects.

Creation: It is difficult to create fake but real looking objects. The creation of fake but real-looking objects is beyond the scope of this paper. We model the creation of a fake object for agent Ai as a black box function CREATEFAKEOBJECT(O_i,F_i,cond$_i$) that takes as input, the set of all objects O_i, the subset of fake objects F_i that A_i has

received so far, and $cond_i$, and returns a new fake object [1]. This function needs $cond_i$ to produce a valid object that satisfies A_i's condition. Set O_i is needed as input so that the created fake object is not only valid but also indistinguishable from other real objects.

For example, the creation function of a fake payroll record that includes an employee rank and a salary attribute may take into account the distribution of employee ranks, the distribution of salaries, as well as the correlation between the two attributes. Ensuring that key statistics do not change by the introduction of fake objects is important if the agents will be using such statistics in their work. Finally, function 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 it wants to send the same fake object to a set of agents. In this case, the function arguments are the union of the O_i and F_i tables, respectively, and the intersection of the conditions $cond_i$s.

Although we do not deal with the implementation of CREATEFAKEOBJECT(), we note that there are two main design options[1]. The function can either produce a fake object on demand every time it is called or it can return an appropriate object from a pool of objects created in advance.

B. Optimization Problem

The distributor's data allocation to agents has one constraint and one objective. The distributor's constraint is to satisfy agents' requests, by providing them with the number of objects they request or with all available objects that satisfy their conditions. His objective is to be able to detect an agent who leaks any portion of his data. We consider the constraint as strict. The distributor may not deny serving an agent request and may not provide agents with different perturbed versions of the same objects as in. We consider fake object 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. 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. Recall that $P_b\{G_j|L=O_i\}$ or simply $P_b\{G_j|O_i\}$is the probability that agent $A_j$is guilty if the distributor discovers a leaked table L that contains all O_i objects. We define the difference functions $\Delta(i,j)$ as

$$\Delta(i,j)= P_b\{G_i|O_i\} - P_b\{G_j|O_i\} \quad i,j=1\ldots\ldots\ldots\ldots n \qquad (6)$$

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

Problem Definition: Let the distributor have data requests from n agents. The distributor wants to give tables $O_1,\ldots..,O_n$to agents $A_1\ldots..A_n$, respectively, so that he satisfies agents' requests, and he maximizes the guilt probability differences$\Delta(i,j)$for all i, j = 1,…, n and $i \neq j$. Assuming that the O_i sets satisfy the agents' requests, we can express the problem as a optimization problem:

$$\text{maximize}_{(\text{over } O1\ldots On)} (\ldots.., \Delta(i,j) ,\ldots....) \qquad (7)$$

If the optimization problem has an optimal solution, it means that there exists an allocation $D^*=\{O_1^*\ldots\ldots O_n^*\}$such that any other feasible allocation $D=\{O_1\ldots\ldots O_n\}$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 $P_b\{G_i|O_i\}$ with respect to any other probability $P_b\{G_i|O_j\}$ with $j\neq i$.

Let us get more clear idea by conducting a small experiment. Figure3. Shows Students record which the distributor holds. It consist of 12 students records along with their id, name, course and marks. This data is nothing but according to this paper set S.

ID	NAME	COURSE	MARKS
1	Jack	Software Engg.	60
2	Billy	Requirment Engg.	90
3	Mcfaden	Computer Networks	34
4	Steven	Software Arch.	96
5	Ruby	DBMS	70
6	Mark	PHP Development	36
7	Philip	Dot Net	78
8	Erik	HTML & Scripting	87
9	Ricky	Data communication	78
10	miecky	Computer Networks	89
11	Monty	Dot Net	45
12	Richa	DBMS	92

Fig. 3 Student Record (set S).

Figure 4 contains 5 records which are leaked and they belong to S i.e. this is set L. Figure 5 and 6 show the records that agent 1 and agent 2 have i.e. those are the O_1 and O_2 respectively.

ID	NAME	COURSE	MARKS
1	Jack	Software Engg.	60
2	Billy	Requirment Engg.	90
3	Mcfaden	Computer Networks	34
4	Steven	Software Arch.	96
5	Ruby	DBMS	70

Fig. 4 Leaked Records (set L).

As the distributor has distributed the data from S to both agent 1 and agent 2 so it is likely to suspect that the data might have been leaked by any one of them. We know the leaked data, now we have to find out, amongst agent 1 and agent2 who has leaked the data. We will compare both the data n find out the difference between the records that each of the agents has i.e.O_1 and O_2.From Figure7 we get to know that out of 7 records there is a difference of 5 records so we have a better chance of identifying the correct leaker

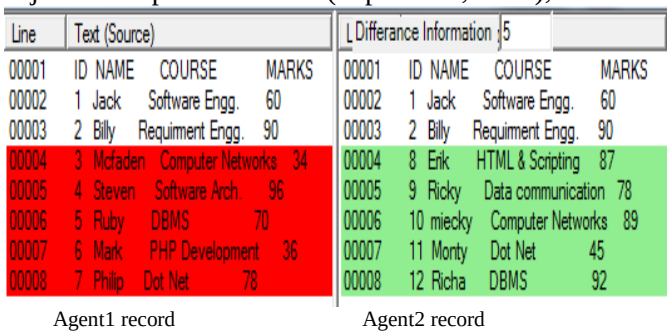
ID	NAME	COURSE	MARKS
1	Jack	Software Engg.	60
2	Billy	Requirment Engg.	90
3	Mcfaden	Computer Networks	34
4	Steven	Software Arch.	96
5	Ruby	DBMS	70
6	Mark	PHP Development	36
7	Philip	Dot Net	78

Fig. 5 Records with agent1 (set O_1).

ID	NAME	COURSE	MARKS
1	Jack	Software Engg.	60
2	Billy	Requirment Engg.	90
8	Erik	HTML & Scripting	87
9	Ricky	Data communication	78
10	miecky	Computer Networks	89
11	Monty	Dot Net	45
12	Richa	DBMS	92

Fig. 6 Records with agent2 (set O_2).

Next we compare L with O_1, as shown in figure 8.It is seen that 5 of the records matched and the difference is of 2. Similarly, we compare L with O_2 in figure 9. We get to know that the similarity is only 2 and the difference is 5. So based on this experiment it is clear that agent 1 is guilty and he is the leaker as more number of records that he has in his data set matches with the leaked set as compared to Agent 2.Hence it is seen that if the difference between the distributed sets is more, then finding out the leaker is easy.

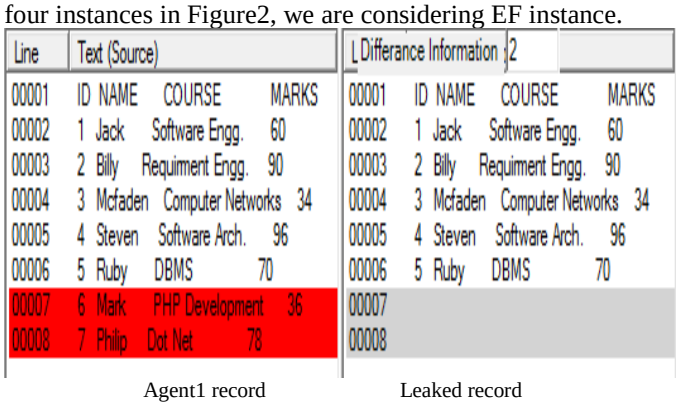


Agent1 record Agent2 record

Fig. 7 Difference between Agent1 and Agent2 records.

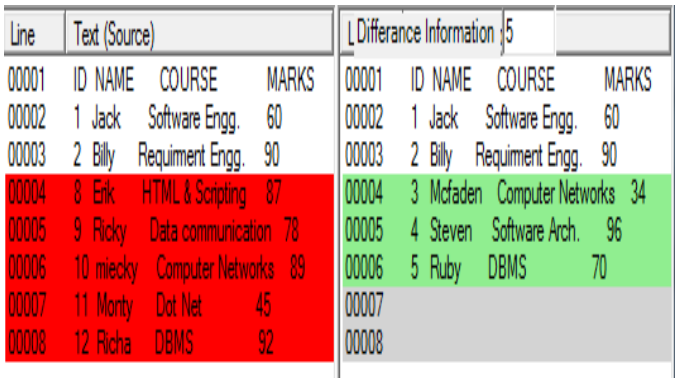
C. *Allocation Strategy*

Algorithm that we present next is a baseline algorithm for distributing data to agents with explicit request. Out of the four instances in Figure2, we are considering EF instance.



Agent1 record Leaked record

Fig. 8 Difference between Agent1 and Leaked records.



Agent2 record Leaked record

Fig. 9 Difference between Agent2 and Leaked records.

Algorithm: Allocation of real object along with fake object [1]:
Input:*O*1… *On*, *cond*1…*condn*, b1…b*n*,
B // *B* – fake objects created by distributor,
bi – fake objects agent Ai can receive
Output:
*O*1… *On*,

*F*1… *Fn*// *Fi* – fake object received by selected agent A*i*

1: R ←∅ . Agents that can receive fake objects
2: for i = 1….. n do
3: if bi > 0 then
4: R ← R ∪ {i} // *i* – Agent that was selected to add fake objects
5: Fi ←∅
6: while B > 0 do
7: i ← SELECTAGENT(R,O1, . . . , On) // *i* –selected agent
8: f ←CREATEFAKEOBJECT(Oi, Fi, condi) // black box function for fake object creation
9: Oi ← Oi ∪ {f} // *f* – Fake object that was created for agent Ai is inserted to *f*
10: Fi ← Fi ∪ {f}
11: bi ← bi - 1
12: if bi = 0 then
13: R ← R\{Oi}
14: B ← B – 1

Above algorithm is a general "driver" that will be used for allocation of the requested data to the agents along with fake objects.

VI. CONCLUSION

Data leakage detection was traditionally handled by watermarking. Watermarking includes alteration of original sensitive data. However, in many cases, there is a need, not to alter the original sensitive data. In such cases, determining whether a leaked object came from an agent or from some other source is very uncertain as it does not include any identification mark. In the proposed system, using the Agent Guilt Model, it is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the probability that objects can be "guessed" by other means. Addition of fake or dummy objects while distributing the real object to the agents can further improves the chances of detecting the guilty agent.

Our future work includes the study of different allocation strategies for different instances (except EF) presented in Figure 2. So that we can consider more scenarios that are not included in this paper and also analysis of Agent Guilt Model using different parameters e.g. e, can be done.

REFERENCES

[1] Panagiotis Papadimitriou, and Hector Garcia-Molina, "Data Leakage Detection" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 1, JANUARY 2011.

[2] R. Agrawal and J. Kiernan, "Watermarking Relational Databases, "Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02), VLDB Endowment, pp. 155-166, 2002.

[3] R. Sion, M. Atallah, and S. Prabhakar, "Rights Protection for Relational Data," IEEE Trans. Knowledge And Data Engineering , vol. 16, no. 12, Dec. 2004.

[4] P. Buneman, S. Khanna, and W.C. Tan, "Why and Where: A Characterization of Data Provenance," Proc. Eighth Int'l Conf. Database Theory (ICDT '01), J.V. den Bussche and V. Vianu, eds.,pp. 316-330, Jan. 2001.

[5] P.Buneman and W.-C. Tan "Provenance in Databases," Proc. ACM SIGMOD, pp. 1171-1173, 2007.

[6] Y.Cui and J. Widom, "Lineage Tracing for General Data Warehouse Transformations," The VLDB J., vol. 12, pp. 41-58, 2003.

[7] S.Czerwinski, R. Fromm, and T. Hodes, "Digital Music Distribution and Audio Watermarking," http://www.scientificcommons. org/43025658, 2007.

[8] Jen-Sheng, Win-Bin Huang,Chao-Lieh Chen, Yau-Hwang Kuo, "A Feature-Based Digital Image Watermarking For Copyright Protection and Content Authentication," 1-4244-1437-7/07/$20.00 ,2007 IEEE ,v-469,ICIP 2007.

[9] F. Hartung and B. Girod, "Watermarking of Uncompressed and Compressed Video," Signal Processing, vol. 66, no. 3, pp. 283-301,1998.

[10] Y. Li, V. Swarup, and S. Jajodia, "Fingerprinting Relational Databases: Schemes and Specialties," IEEE Trans. Dependable and Secure Computing, vol. 2, no. 1, pp. 34-45, Jan.-Mar. 2005.

[11] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian, "Flexible Support for Multiple Access Control Policies," ACM Trans. Database Systems, vol. 26, no. 2, pp. 214-260, 2001.

[12] L. Sweeney, "Achieving K-Anonymity Privacy Protection Using Generalization and Suppression," http://en.scientificcommons.org/43196131, 2002.