ISSN:0975-9646

Assessing the Guilt Probability in Intentional Data Leakage

Yadav Gitanjali B., Bhaskar P. C., Dr. Kamat R. K.
[#]*Department Of Technology, Shivaji University*
Kolhapur, Maharashtra, India

***Abstract*—For most corporations the volume of sensitive data used by outsourcing providers continues to increase. As the number of different entities having access to a database increases, it gets harder to prevent and trace-back data leakage. We address the problems of proving ownership and unauthorized data distribution (leakage) for relational databases. We propose the technique that altogether may be used to detect, deter and trace-back data leaks from relational databases. We use business process outsourcing scenarios as the descriptive use case, but our techniques are equally applicable in other use cases when a relational database is shared among many parties and its confidentiality and authenticity needs to be protected. In early days watermarking and fingerprinting is used to prove ownership and track unauthorized redistributions of numerical relational data respectively. In this work we are inserting "realistic but fake" objects to our database, which is just similar to watermarking.**

Keywords:* *distribution model; guilty agent; fake objects; sensitive information.

I. Introduction

Nowadays, sensitive information has become more significant for financial departments, medical organizations, security sectors, and enterprises worldwide. Once the information is leaked, the owners may suffer a great loss. Here sensitive data encompasses a wide range of information and can include: your ethnic or racial origin; political opinion; religious or other similar beliefs; memberships; physical or mental health details; personal life; or criminal or civil offences. These examples of information are protected by your civil rights.

Sensitive data can also include information that relates to you as a consumer, client, employee, patient or student; and it can be identifying information as well: your contact information, identification cards and numbers, birth date, and parents' names.

Based on publicly disclosed Data Leakage breaches, the type of data leaked is broken down as follows:

Type Of Information Leaked	**Percentage**
Confidential Information	15%
Intellectual Property	4%
Customer Data	73%
Health records	8%

In the course of doing business, sometimes this sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise a new treatment. Similarly, accompany may have partnership 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 consider here two entities:

Distributor: The owner of data

Agents: Supposedly trusted third parties to whom data is given.

Our goal is to detect which agent leaked the data.

We consider applications where 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 [7]. However, in some cases it is important not to alter the original distributor's data. For example, if an outsourcer is doing a 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,[5] 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. Watermark can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermark can sometimes be destroyed if the data recipient is malicious.

Here we introduced unobtrusive techniques for detecting data leakage of set of objects or records. We developed a model for assessing the "guilt" of agents. We also gave the algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker. Finally, we also consider the option of adding "fake" objects to the distributed set. Such objects do not correspond to real entities but appear realistic to the agents. In a sense, the fake objects act as a type of watermark for the entire set, without modifying any individual members. If it turns out an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty. Data allocation strategies featured with fake objects injection are proposed by Ref.[2] , but they consider fixed number of agents. In this paper we are considering the online requests of the agents.

II. Related Work

The data leakage prevention based on the trustworthiness [1] is used to assess the trustiness of the customer. Maintaining the log of all customer's request is related to the data provenance problem [3] i.e. tracing the lineage of objects. The data allocation strategy used is more relevant to the watermarking [4],[5] that is used as a means of establishing original ownership of distributed objects.

There are also different mechanisms to allow only authorized users to access the sensitive information [6] through access control policies, but these are restrictive and may make it impossible to satisfy agent's requests.

III. GUILT DETECTION MODEL

A. Problem Definition:

The distributor owns the sensitive data set T= { t_1, t_2, ………., t_n}. The agent A_i request the data objects from distributor. The objects in T could be of any type and size, e.g. they could be tuples in a relation, or relations in a database. The distributor gives the subset of data to each agent. After giving objects to agents, the distributor discovers that a set L of T has leaked. This means some third party has been caught in possession of L. The agent A_i receives a subset R_i of objects T determined either by implicit request or an explicit request.

- Implicit Request R_i = Implicit(T, m_i) : Any subset of m_i records from T can be given to agent A_i
- Explicit Request R_i = Explicit(T, $Cond_i$) : Agent A_i receives all T objects that satisfy $Cond_i$

B. Guilt Assessment:

Let L denote the leaked data set that may be leaked intentionally or guessed by the target user. Since agent having some of the leaked data of L, may be susceptible for leaking the data. But he may argue that he is innocent and that the L data were obtained by target through some other means.

Our goal is to assess the likelihood that the leaked data came from the agents as opposed to other resources. e.g. if one of the object of L was given to only agent A_1, we may suspect A_1 more. So probability that agent A_1 is guilty for leaking data set L is denoted as Pr {G_1 | L}.

C. Guilt Probability Computation:

For the sake of simplicity our model relies on two assumptions:

Assumption 1: For all t_1, t_2, ………..,t_n Є L and $t_1 \neq t_2$, the provenance of t_1 is independent of t_2 .

Assumption 2: Tuple t Є L can only be obtained by third user in one of the two ways:

1. Single user A_1 leaked t or
2. Third user guessed t with the help of other resources.

Now to compute the guilt probability that he leaks a single object t to L, we define a set of users. To find the probability that an agent A_i is guilty for the given set L, consider the target guessed t_1 with probability p and that agent leaks t_1 to L with probability 1-p. First compute the probability that he leaks a single object to L. To compute this, define the set of agents U_t = {A_i | t Є R_i } that have t in their data sets. Then using Assumption 2 and known probability p, we have,

$$\textbf{Pr \{Some agent leaked t to L\} = 1-p} \qquad \text{------(1)}$$

Assuming that all agents that belongs to U_t can leak t to L with equal probability and using Assumption 2 we get,

$$Pr(\mathbf{A_i\ leaked\ t\ to\ L}) = \begin{cases} \frac{1-p}{U_t}, & if\ A_i \in U_t \\ 0 & Otherwise \end{cases} \qquad \text{----(2)}$$

Given that user A_i is guilty if he leaks at least one value to L, with assumption 1 and equation 2, we can compute the probability Pr {G_i | L} that user A_i is guilty :

$$\mathbf{Pr\ \{Gi \mid L\} = 1 - \prod_{t \in L \cap Ri} \left(\frac{1-(1-p)}{Ut}\right)} \qquad \text{-----(3)}$$

D. Data Allocation Strategies:

The distributor gives the data to agents such that he can easily detect the guilty agent in case of leakage of data. To improve the chances of detecting guilty agent, he injects fake objects into the distributed dataset. These fake objects are created in such a manner that, agent cannot distinguish it from original objects. One can maintain the separate dataset of fake objects or can create it on demand. In this paper we have used the dataset of fake tuples.

Depending upon the addition of fake tuples into the agent's request, data allocation problem is divided into four cases as:

i. Explicit request with fake tuples
ii. Explicit request without fake tuples
iii. Implicit request with fake tuples
iv. Implicit request without fake tuples.

For example, distributor sends the tuples to agents A_1 and A_2 as

R_1= {t_1, t_2} and R_2= { t_1}. If the leaked dataset is L={ t_1}, then agent A_2 appears more guilty than A_1. So to minimize the overlap, we insert the fake objects into one of the agent's dataset.

E. Overlap Minimization:

The distributor's data allocation to agent has one constraint and one objective. The distributor's constraint is to satisfy agent's request, 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 his constraint as strict. The distributor may not deny serving an agent request and may not provide agents different perturbed versions of the same object.

The objective is to maximize the chances of detecting guilty agent that leaks all his data objects.

The Pr {G_i | L= R_i} is the probability that agent A_i is guilty if distributor discovers a leaked table L that contains all R_i objects.

The difference function Δ (i, j) is defined as

$$\Delta(i,j) = \Pr\{G_i|R_i\} - \Pr\{G_j|R_i\} \qquad i,j = 1, \ldots\ldots n$$

The value of Δ is positive if for any agent A_j, whose set R_j does not contain all data of L. It is negative in case of $R_i \subseteq R_j$. The larger the Δ value is, the easier it is to identify A_i is guilty.

Problem definition: Let the distributor have data request from n agents. The distributor wants to give tables R_1 ,R_2…….. R_n to agents A_1 ,A_2…………. A_n respectively, so that

- Distribution satisfies agent's request; and
- Maximizes the guilt probability differences Δ (i, j) for all i, j= 1, 2, ……n and i≠j.
-

Optimization Problem:

Maximizing the difference among distributed dataset increases the minimization of overlap.

i.e $\mathbf{max}_{(overR_1,\ldots\ldots\ldots,R_n)}(\ldots\ldots, \Delta(\boldsymbol{i},\boldsymbol{j}), \ldots.) \qquad \boldsymbol{i \neq j}$

Then

$$\min_{(over\ R_1,\ldots\ldots,R_n)} \left(\ldots\ldots, \left|\frac{R_i \cap R_j}{|R_i|}\right|\right) \qquad \boldsymbol{i \neq j}$$

IV. EXPERIMENTAL SETUP

In this paper, we presented the algorithm and the corresponding results for the explicit data allocation with the addition of fake tuples. Whenever any user request for the tuples, it follows the following steps:

1. The request is sent by the user to the distributor.
2. The request may be implicit or explicit.
3. If it is implicit a subset m_i of the data set T is given.
4. If request is implicit, it is checked with the log, if any previous request is same.
5. If request is same then system gives the data objects that are not given to previous agent.
6. If request is explicit, 10% tuples inserted in it are fake.
7. Leaked data set L, obtained by distributor is given as an input.
8. Calculate the guilt probability G_i of user using II.
9.

In the case where we get similar guilt probabilities of the agents, we consider the trust value of agent. These trust values are calculated from the historical behavior of agents. The calculation of trust value is not given here, we just assumed it. The agent having low trust value is considered as guilty agent.

The algorithm for allocation of dataset on agent's explicit request is given below.

A. Explicit data Allocation with fake tuples:

To improve the chances of finding guilty agent we can add the fake tuples to their data sets. Here we maintained the table for fake tuples and add randomly these tuples to the agent's dataset.

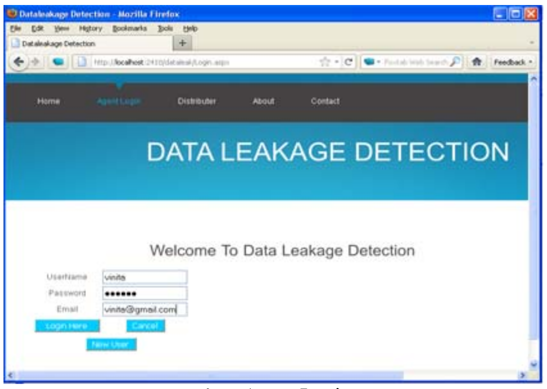
Similarly, we can distribute the dataset for implicit request of agent. For implicit request the subset of distributor's dataset ($m_i \subseteq T$) is selected randomly. Thus with the implicit data request we get $\binom{|T|}{m_i}$ different subsets. Hence there are $\prod_{i=1}^{n}\binom{|T|}{m_i}$ different data allocations. An object allocation that satisfies requests and ignores the distributor's objective is to give each agent A_i unique subset of T of size m. The following algorithm allocates to an agent the data record that yields the minimum increase of the maximum relative overlap among any pair of agents.

Here the agent's requests are not of the same size (i.e. for every agent m_i is different). Thus our algorithm works well for the case $|T| \leq M \leq 4|T|$, where $M = \sum_{i=1}^{n} m_i$. The algorithm presented implements a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. It is shown that distributing objects judiciously can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive. Again addition of fake tuples to the implicit request maximizes the chances of detecting guilty agent.

IV. EXPERIMENTAL RESULTS

In our scenarios we have taken a set of 600 objects and requests from every agent are accepted. The system works fine for 20 agents where total sent tuples are nearly 5|T|. The flow of our system is given as follows

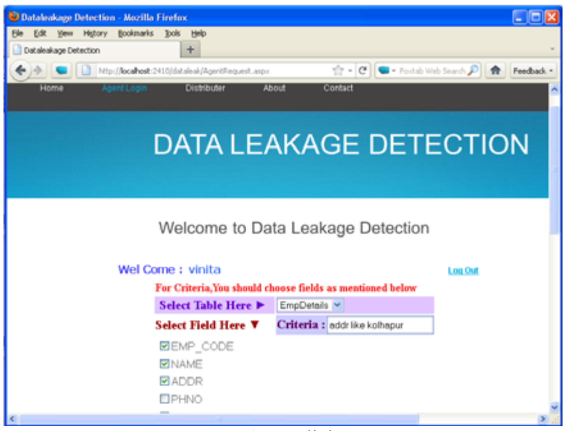
1. Agent send explicit or implicit request to the distributor



Welcome To Data Leakage Detection

Username	vinita
Password	●●●●●●
Email	vinita@gmail.com

Fig1. Agent Login



Welcome to Data Leakage Detection

Wel Come : vinita

For Criteria,You should choose fields as mentioned below

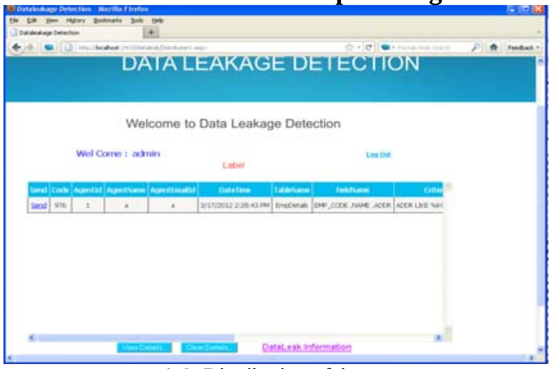
Select Table Here ▶ EmpDetails

Select Field Here ▼ Criteria : addr like kolhapur

☑EMP_CODE
☑NAME
☑ADDR
☐PHNO

Fig2. Agent's Explicit Request

2. Distributor sends tuples to agents



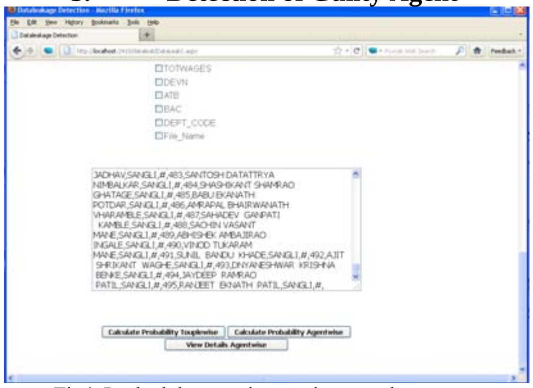
Welcome to Data Leakage Detection

Wel Come : admin

Send	Code	AgentId	AgentName	AgentEmailId	DateTime	TableName	FieldName	Criter
Send	976	1	a	a	3/17/2012 2:28:43 PM	EmpDetails	EMP_CODE ,NAME ,ADDR	ADDR LIKE %K

Fig3. Distribution of dataset

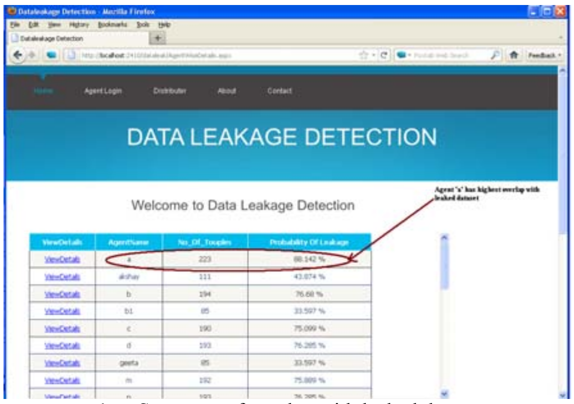
3. Detection of Guilty Agent


☐TOTWAGES
☐DEVN
☐ATB
☐BAC
☐DEPT_CODE
☐File_Name

```
JADHAV,SANGLI,#,483,SANTOSH DATATTRYA
NIMBALKAR,SANGLI,#,484,SHASHIKANT SHAMRAO
GHATAGE,SANGLI,#,485,BABU EKANATH
POTDAR,SANGLI,#,486,AMRAPAL BHAIRWANATH
VHARAMBLE,SANGLI,#,487,SAHADEV  GANPATI
 KAMBLE,SANGLI,#,488,SACHIN VASANT
MANE,SANGLI,#,489,ABHISHEK AMBAJIRAO
INGALE,SANGLI,#,490,VINOD TUKARAM
MANE,SANGLI,#,491,SUNIL  BANDU  KHADE,SANGLI,#,492,AJIT
 SHRIKANT  WAGHE,SANGLI,#,493,DNYANESHWAR  KRISHNA
 BENKE,SANGLI,#,494,JAYDEEP  RAMRAO
 PATIL,SANGLI,#,495,RANJEET  EKNATH  PATIL,SANGLI,#,
```

Calculate Probability Touplewise
Calculate Probability Agentwise
View Details Agentwise

Fig4. Leaked dataset given as input to the system


Datalekage Detection - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Home
Agent Login
Distributer
About
Contact
DATA LEAKAGE DETECTION
Welcome to Data Leakage Detection
Agent 'a' has highest overlap with leaked dataset

ViewDetails	AgentName	No_Of_Touples	Probability Of Leakage
ViewDetails	a	223	88.142 %
ViewDetails	akshay	111	43.874 %
ViewDetails	b	194	76.68 %
ViewDetails	b1	85	33.597 %
ViewDetails	c	190	75.099 %
ViewDetails	d	193	76.285 %
ViewDetails	geeta	85	33.597 %
ViewDetails	m	192	75.889 %
ViewDetails	n	193	76.285 %

Fig5. Summary of overlap with leaked dataset

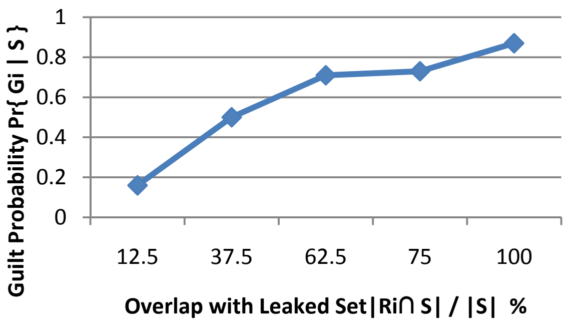


Fig 6. Sample request from 10 agents

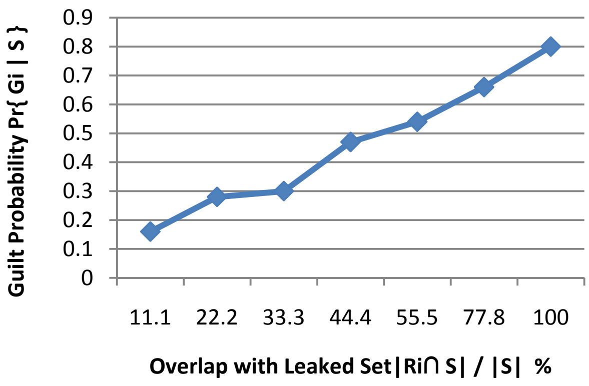


Fig.7 Sample request from 14 agents

V. CONCLUSION

Data leakage happens every day when sensitive business information such as customer or patient data, source code or design specifications, intellectual property and trade secrets are leaked out. When these are leaked out it leaves the company unprotected and goes outside the jurisdiction. This uncontrollable data leakage put business in a vulnerable position. Once this data is no longer within the domain, then company is at serious risk. When cybercriminals "cash out" or sell this data for profit it costs our organization money, damages the competitive advantage, brand, and reputation and destroys customer trust. Our presented model assesses the "guilt" of agents. The main focus of this project is the data allocation problem. It specifies how the distributor can "intelligently" give data to agents in order to improve the chances of detecting a guilty agent. By adding fake objects to distributed set, the distributor can find the guilt agent easily.

REFERENCES

1. YIN Fan, WANG Yu, WANG Lina, Yu Rongwei. A Trustworthiness-Based Distribution Model for Data Leakage Detection: Wuhan University Journal Of Natural Sciences.
2. P. Papadimitriou and H. Garcia-Molina. Data leakage detection.Technical report, Stanford University, 2008.
3. P. Buneman, S. Khanna and W.C. Tan. Why and where: A characterization of data provenance. ICDT 2001, 8th International Conference, London, UK, January4-6, 2001, Proceedings, volume 1973 of Lecture Notes in Computer Science, Springer, 2001.
4. S. Czerwinski, R. Fromm, and T. Hodes. Digital music distribution and audio watermarking.
5. Rakesh Agrawal, Jerry Kiernan. Watermarking Relational Databases// IBM Almaden Research Center
6. S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. ACM Trans. Dataset Systems, 26(2):214-260,2001.
7. L. Sweeney. Achieving k- anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzzyness and Knowledge-based Systems-2002