ISSN 2229-5518

Data Leakage and Detection of Guilty Agent

Rupesh Mishra, D.K. Chitre

Abstract— Organizations apply data or information security only in terms of protecting their network from intruders (e.g. hackers) but with growing amount of sensitive data and rapid growth in the sizes of organizations (e.g. due to globalization), rise in number of data points (machines and servers) and easier modes of communication, sometime accidental or even deliberate leakage of data from within the organization has become a painful reality. This has lead to growing awareness about sensitive data security in general and about outbound content management in particular.

Index Terms— Allocation strategies, Data distribution, Data leakage, Data privacy, Fake records, Guilty agent, Leakage Model.

1 Introduction

DATA leakage is the unauthorized transmission of data or information from within an organization to an external destination or recipient. This may be electronic, or may be via a physical method [1]. Data Leakage is synonymous with the term Information Leakage. Here unauthorized does not actually mean intentional or malicious. Even unintentional or inadvertent data leakage is also unauthorized. The protection measure must be taken to secure these valuable data because if data is leaked from an organization and cybercriminals "cash out" or sell this data for profit it costs organizations money, damages the brand, image and destroys customer trust.

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties for some enhancement or operations, 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 of customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. Owner of the data is termed as the distributor and the supposedly trusted third parties are called as the agents. Our goal is to identify the guilty agent when distributor's sensitive data have been leaked by some agents. Perturbation and watermarking are techniques which can be helpful in such situations. 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 [2]. 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 bank account numbers. If medical researchers treating the patients (as opposed to simply computing statistics), they may need accurate data for the patients.

In this paper such applications are considered where the original sensitive data cannot be perturbed. Traditionally, leakage detection is handled by watermarking, e.g., a unique code is embedded in each distributed copy [3]. 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, they involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious, for leakage detection unobtrusive technique is required.

In the business process after giving 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 website, or obtained through a legal discovery process). At this point, the distributor can assess the likelihood that the leaked data has come from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with five cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings [4].

In this paper, section 2 includes the problem definition in detail, Section 3 covers work done in this area, Section 4 represents problem setup and notation related with the data leakage detection system, Section 5 explains the agent guilt model which calculate or estimate the guilt of agent by counting appropriate number of evidences, Section 6 represents distribution and detection algorithm along with description fake record and optimization problem in the system, Section 7 includes some experimental results and outcome of the algorithms, Section 8 includes conclusion of and list of references used to prepare this paper.

2 Problem Definition

In the business process, sometime owner of data gives set

- *Rupesh Mishra is currently pursuing masters degree program in computer engineering in University, India,. E-mail: rrupesh.mishra@gmail.com*
- *Prof. D. K. Chitre has masters degree in computer engineering and working As assistant Professor in Terna Engineering College, India, E-mail: dkchitre@rediffmail.com*

of sensitive data to trusted agents for performing some operation on it.

This type of data is very sensitive and leakage of this type of data happens when confidential business data is leaked out, if that data leaked and found in some unauthorized place, it leaves the company unprotected and destroys the image and customers trust and goes outside the jurisdiction of the corporation. This uncontrolled data leakage puts business in a vulnerable position. If this data is no longer within the domain, the company is at serious risk hence distributor must find out the guilty agent if the leaked from one or more agents, as opposed to having been independently gathered by other means.

Here the data allocation strategies (across the agents) that improve the probability of identifying guilty agent are proposed. This method works if leaked data is obtained as it was distributed or if fake records are deleted.

3 Related Work

The guilt detection approach is related with the data provenance problem, which means tracing the source of an object which leads to the detection of the guilty agents [5]. It provides a good overview on the research conducted in this field, and some domain specific solutions, such as lineage tracing for data Warehouses. The problem formulation with objects and sets is more general and simplifies lineage tracing, since we do not consider any data transformation. As far as the data allocation strategies are concerned, the work is based on watermarking. This technique is used as a means of establishing original ownership of distributed objects [6]. Watermarks were initially used in images, video and audio data whose digital representation includes considerable redundancy and also there is danger of damage by intruders [3].

Finally, there are lots of other works on mechanisms that allow only authorized users to access sensitive data through access control policies [7], [8]. Such approaches prevent data leakage in some sense by sharing information only with trusted parties. However, these policies are restrictive and may make it impossible to satisfy request of an agent every time, therefore some techniques are required for efficient distribution of data among agents in such a way that tracing them from the leaked set improves.

4 Problem Setup and Notation

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.

The constraint is considered as strict. The distributor may not deny serving an agent request and may not provide agents with different perturbed versions of the same objects. For this fake object distribution is the only possible constraint relaxation. The detection objective is ideal and tractable. The main objective is to maximize the chances of detecting the guilty agent who leaks his data objects.

4.1 Entities and agents

A distributor owns a set $T = \{t_1 \ldots t_m\}$ of valuable data objects. The distributor wants to share some of the objects with a set of agents $U_1, U_2 \ldots U_n$, but does not wish the objects to be leaked to other third parties. 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. An agent U_i receives a subset of objects $Ri \subseteq T$, determined either by a sample request or an explicit request:

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

Explicit request R_i = EXPLICIT (T, condition): Agent U_i receives all T objects that satisfy condition.

4.2 Guilty agents

Suppose, after giving objects to agents, the distributor discovers that a set $S \subseteq T$ has been leaked. This means that some third party called the target has been caught in possession of S. For example, this target may be displaying S on its web site, or perhaps as part of a legal discovery process data obtained in someone's laptop, then the target turned over S to the distributor, Since the agent $U_1 \ldots\ldots U_n$ has some of the data, it is reasonable to suspect them on leaking the data. However, the agents can argue that they are innocent, and that the S data was obtained by the target by other means. For example, say one of the objects in S represents a customer X. Perhaps X is also a customer of some other company, and that company provided the data to the target. Or perhaps X can be reconstructed from various publicly available sources on the web. Our goal is to estimate the likelihood that the leaked data came from the agents as opposed to other sources. Intuitively, the more data in S, the harder it is for the agents to argue they did not leak anything and the target obtained them through other means and estimate the likelihood that the agents leaked data, 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 S objects was only given to agent U_1, while the other objects were given to all agents, we may suspect U_1 more. The model we present next captures this intuition. We say an agent U_i is guilty if it contributes one or more objects to the target. We denote the event that agent U_i is guilty for a given leaked S by $\{G_i \mid S\}$. Our next step is to estimate Pr $\{G_i \mid S\}$, i.e., the probability that agent U_i is guilty given evidence S.

5 Agent Guilt Model

To compute this Pr $\{G_i \mid S\}$, we need an estimate for the probability that values in S can be guessed by the target. For instance, say some of the objects in T are emails of individuals. We can conduct an experiment and ask a person with approximately the expertise and resources of the target to find the email of say 100 individuals. If this person can find say 90 emails, then we can reasonably guess that the probability of finding one email is 0.9. On the other hand, if the objects in question are bank account numbers, the person may only discover say 20, leading to an estimate of 0.2. We call this estimate pt, the probability that object t can be guessed by the target [4].

For simplicity we assume that all T objects have the same p_t, which we call p. Next, we make two assumptions regarding the relationship among the various leakage events. The first assumption simply states that an agent's decision to leak an object is not related to other objects.

We are setting some constraint under which the problem is evaluated

1. For all t, t′ ∈ S such that t ≠ t′ the provenance of t is independent of the provenance of t′ [4]. The term "provenance" in this assumption statement refers to the source of a value t that appears in the leaked set. The source can be any of the agents who have t in their sets or the target itsel. The following assumption states that joint events have a negligible probability

2. An object t ∈S can only be obtained by the target in one of the two ways as follows [4].
A single agent U_i leaked set t from its own R_i set.
The target t guessed or obtained through other means without the help of any of the n agents.

In other words, for all t ∈ S, the event that the target guesses t and the events that agent U_i (i = 1 . . . n) leaks object t are disjoint, let the distributor set T, the agent sets R_s, and the target set S are
T= {t_1, t_2, t_3}, R_1 = {t_1, t_2}, R_2 = {t_1, t_3}, S = {t_2, t_1, t_3}.
In this case, all three of the distributor's objects have been leaked and appear in S. Let us first consider how the target may have obtained object t_1, which was given to both agents. From assumption 2, the target either guessed t_1 or one of U_1 or U_2 leaked it. We know that the probability of the former event is p, so assuming that probability that each of the two agents leaked t_1 is the same, we have the following cases:
The target guessed t_1 with probability p,
Agent U_1 leaked t_1 to S with probability (1-p)/2,
Agent U_2 leaked t_1 to S with probability (1-p)/2.
Similarly, we find that agent U_1 leaked t_2 to S with probability (1-p) since he is the only agent that has t_2. Given these values, the probability that agent U_1 is not guilty, namely that U_1 did not leak either object, is

$$\Pr\{\overline{G_1} \mid S\} = (1-(1-p)/2)\times(1-(1-p)) \quad (1)$$

And the probability that U_1 is guilty is:

$$\Pr(G_1 \mid S) = 1-\Pr\{\overline{G_1}\} \quad (2)$$

If assumption 2 did not hold, our analysis would be more complex because we would need to consider joint events, e.g., the target guesses t_1, and at the same time, one or two agents leak the value, an agent is not guilty when the object can be guessed, regardless of whether the agent leaked the value. Thus allocation of data set is done in such a way that this situation can be avoided and guilty of agent can be proved.

6 DATA ALLOCATION STRATEGIES

In this paper, allocation strategies are given prime importance as it is the most important step in the data leakage detection system. If the data is distributed properly, the agent can be easily traced from the distribution set. We deal with problems with explicit data requests, and problems with sample data requests.

In problems for both sample and explicit data request, if the distributor is not allowed to add fake objects to the distributed data then the data allocation is fully defined by the agents' data requests, for example T= {t_1, t_2} and there are two agents with explicit data requests such that R_1 = {t_1, t_2} and R_2 = {t_1}. The value of the sum objective is in this case is 1.5. The distributor cannot remove or alter the R_1 or R_2 data to decrease the overlap $R_1 \cap R_2$. If the distributor is able to create more fake objects, he could further improve the objective.

6.1 Distribution logic

In case fake records are allowed to add in the distribution set the chances of identifying agent increases. Data distribution algorithm is proposed which starts whenever an agent requests for data and fake records are possible to add in the set. Algorithm proceeds with the assignment process in which distribution identification number (DID), if objects are given DID at the time of storage then this stage is optional. DID must be unique for each type of record. The process proceeds with hashing in which agent identification number (AID) is hashed in such a way that it can be mapped with DID, this stage gives the record number which is to be replaced by some fake record generated in third step with fake identification number (FID). This helps to detect record at the time of detection, generation of fake record can be done prior with the process but in order to generate realistic fake record it is recommended that it should be generated after getting the record at hashed location, once the fake record is mixed in the distribution set the mapping information of FID, DID with primary key along with other information is stored in the system, before distribution corresponding DID is removed and data set is given to corresponding agent.

Algorithm 1: Distribution Algorithm

1. **Assignment:** Assigning DID to data objects.
2. **Hashing:** H(AID) ⟶ DID
3. **Fake record generation:** FID and Fake Record
4. **Mapping:** FID ⟶ DID with P.K. of record
5. **Backup & Removal:** Store information and remove DID.

6.2 Agent detection

Once the data set is distributed among agents and obtained from some unauthorized place from that data agent is traced and the process of detection starts which involves three stages. At first stage the object of leaked data set S is assigned with corresponding DID which is obtained by mapping primary key of record. Once DID is attached with the data object, tracing of agent begins which can be done in three ways.

First it starts by searching agent with S. After detection of agent in this stage agents are filtered with Fake record. If agent identifies the fake record and deletes the same from the distribution list and is then given to target, then this stage fails and now agent is traced from the missing original record which can be done by adding hashing the AID obtained from first stage and matching them in the set S.
Guessing true record is much difficult than the identification of fake record from agent's side. Therefore this algorithm works even when fake records are absent in S. After the detection of agent, guilt is estimated from the guilt detection model.

Algorithm 2: Agent Detection
Set **S** $\subseteq$ **T** is obtained at unauthorized place

1. **Mapping DID**
2. **Agent Tracing**
    1. Distribution ID set
    2. Fake record
    3. Missing record
3. **Estimate guilt**

If no agent is traced from above algorithm it is assumed that data is obtained by target from some other means.

6.3 Fake object

The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. However, fake objects may impact the correctness of what agents do, so they may not always be allowable. The idea of perturbing data to detect leakage is not new. However, in most cases, individual objects are perturbed. [3] Adding random noise to sensitive salaries or adding a watermark to an image is example of perturbation. In this case, perturbing the set of distributor objects by adding fake elements is done. In some applications, fake objects may cause fewer problems than perturbing real objects. For example, consider 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. A trace file is maintained to identify the guilty agent. Trace file are a type of fake objects that help to identify improper use of data.

6.4 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 of his data objects. 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. We consider fake object allocation 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. Instead we use the following objective: maximize the chances of detecting a guilty agent that leaks all his objects.We now introduce some notation to state formally the distributor's objective.
Pr $\{G_j | S = R = i\}$ or simply Pr $\{G_j | S = R_i\}$, is the probability that agent U_j is guilty if the distributor discovers a leaked table S that contains all R_i objects.
We define the difference functions Δ (i, j) as:

$$\Delta (i, j) = Pr \{G_j \mid S = R_i\} - Pr \{G_i \mid S = R_i\}$$
$$(i, j\ 1,\ldots\ldots..n) \quad (3)$$

Note that differences Δ have non-negative values: given that set R_i contains all the leaked objects, agent U_i is at least as likely to be guilty as any other agent. Difference Δ (i, j) is positive for any agent U_j, whose set R_j does not contain all data of S. It is zero, if $R_i \subseteq R_j$. In this case the distributor will consider both agents U_i and U_j equally guilty since they have both received all the leaked objects. The larger a Δ (i, j) value is, the easier it is to identify Ui as the leaking agent. Thus, we want to distribute data so that Δvalues are large.

7 Experimental Results

Type your main text in 10-point Times, single- In simulated data leakage detection environment random number data objects are distributed among agent with random probability of guilt detection probability. It is observed that the relative performance of our algorithms and our main conclusions do not change. If p approaches to 0, it becomes easier to find guilty agents and algorithm performance converges. On the other hand, if p approaches 1, the relative differences among algorithms grow since more evidence is needed to find an agent guilty. The presented experiments confirmed that fake objects can have a significant impact on our chances of detecting a guilty agent. This means that by allocating 10 percent of fake objects, the distributor can detect a guilty agent even in the worst-case leakage scenario, while without fake objects, successfully agent can be identified.

7.1 Explicit request

With explicit data request few objects that are shared among multiple agents. These are the most interesting scenarios, since object sharing makes it difficult to distinguish a guilty from a non guilty agent. Scenarios with more objects to distribute or scenarios with objects shared among fewer agents are obviously easier to handle. As far as scenarios with many objects to distribute and many overlapping agent requests are concerned, they are similar to the scenarios we study, since we can map them to the distribution of many small subsets.

7.2 Sample requests

With sample data requests, agents are not interested in particular objects. Hence, object sharing is not explicitly defined by their requests. The distributor is "forced" to allocate certain objects to multiple agents only if the number of requested objects exceeds the number of objects in set T.

The more data objects the agents request in total, the more recipients, on average, an object has; and the more objects are shared among different agents, the more difficult it is to detect a guilty agent.

8 Conclusion

In a perfect world there would be no need to hand over sensitive data to agents who may unknowingly or maliciously leak it. And even if we had to hand over sensitive data, in a perfect world we could watermark each object so that we could trace its origins with absolute certainty.

However, in many cases we may have to work with agents who may not be trustworthy, and we may not be certain if the leaked object has come from an agent or from some other source. In spite of these difficulties, we have shown it is possible to detect that an agent is responsible for a leak, based on the overlap of its data with the leaked data and the data of other agents, and based on the probability that objects can be "guessed" by other means.

ACKNOWLEDGMENTS

We would like to thank Prof. V. B. Gayakwad, Assistant Professor, Computer Engineering Department, TCE, Nerul, New Mumbai for his encouragement and motivation to write this paper.

References

[1] Peter Gordon "Data Leakage – Threats and Mitigation" SANS Institute Reading Room October 15, 2007

[2] J. Clerk Ma P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," Stanford University.

[3] 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.

[4] Panagiotis Papadimitriou, Student Member, IEEE, and Hector Garcia-Molina, Member, IEEE "Data Leakage Detection" IEEE Transactions on knowledge and data engineering, Vol. 23, NO. 1, January 2011

[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. 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.

[8] P. Bonatti, S.D.C. di Vimercati, and P. Samarati, "An Algebra for Composing Access Control Policies," ACM Trans. Information and SystemSecurity, vol.5, no.1, pp.1-35, 2002.