IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Efficient Data Allocation Model for Data Leakage Detection System

Anju Sebastian[1], Malliha A.[2] and Sarah Prithvika P.C.[3]

[1,2,3] **Department of Computer Science and Engineering**
Agni college of Technology. Anna University
Chennai, Tamil Nadu, India

Abstract

In the course of doing business, a distributor sometimes needs to hand over sensitive data to supposedly trusted agents. After giving a set of data to agents, the distributor discovers some of those same objects in an unauthorized place. The distributor must assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Some data allocation strategies (across the agents) can be used to improve the probability of identifying leakages. In some cases, "realistic but fake" data records can be injected to further improve the chances of detecting leakage and identifying the guilty party.

Keywords: *Allocation strategies, data leakage, data privacy, fake records, leakage model.*

1. Introduction

In today's business environment, there is a need to communicate sensitive data. Corporates need to hand over this sensitive information to a trusted third party. After giving a set of objects to the third party, the objects may be discovered in an unauthorized place. The owner of the data is called the distributor and the supposedly trusted third parties are called the agents. The objective is to find out the source of data leakage. 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. Traditionally, leakage detection is handled by steganography, perturbation and watermarking.

Steganography is the art and science of communicating in a way which hides the existence of the communication. The goal of Steganography is to hide messages inside other harmless messages in a way that does not allow any enemy to even detect that there is a second message present. There are some drawbacks in this technique. With encryption, the receiver can be reasonably sure that he has received a secret message when a seemingly meaningless file arrives. It has either been corrupted or is encrypted. But, with hidden data it is not so clear; the receiver simply receives an image, and needs to know that there is a hidden message and how to locate it. Otherwise the receiver may not get the message because he is unaware that the message is hidden. Another limitation is due to the size of the medium being used to hide the data. In order for steganography to be useful, the message should be hidden without any major changes to the object it is being embedded in. This leaves limited room to embed a message without noticeably changing the original object. Robustness attacks, presentation attacks, interpretation attacks and implementation attacks may occur.

Perturbation is a very useful technique where the data are 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 [18]. 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 are to treat patients (as opposed to simply computing statistics), they may need accurate data for the patients.

In **watermarking** 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. There are many schemes used for implementing watermarking. *Public watermarking and blind watermarking* mean the same; the original cover signal is not needed during the detection process to detect the mark. *Private watermarking and non-blind-watermarking* mean the same; the original cover signal is required during the detection process. Watermarking is not suitable in all cases because it obscures the image, it involves modification of the original data, it is time consuming, it can be easily removed and it offers limited protection. So there is a need for alternate effective methods to overcome these drawbacks. Some of the possible attacks in watermarking are Scrambling Attack Sensitivity Analysis Attack and Gradient Descendent Attack and Collusion Attack.

A model to assess the "guilt" of agents must be developed. Several algorithms are used for distributing objects to agents, in a way that improves the chances of identifying a leaker. Finally, the option of adding "fake" objects to the distributed set is also considered. 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 that an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty. In Section 2 the problem setup and the notation that is used is introduced and in Sections 4 the Efficient Data Allocation Model for Data Leakage Detection System.

2. Problem Setup and Notation

2.1 Entities and Agents

A distributor owns a set $T = \{t_1. . . t_m\}$ of valuable data objects. The distributor wants to share some of the objects with a set of agents $U_1, U_2, . . ., U_n$, but does not wish the objects 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 $R_i \subseteq T$ determined either by a sample request or an explicit request:

- Sample request R_i =SAMPLE(T,m_i): Any subset of m_i records from T can be given to U_i
- Explicit request R_i = EXPLICIT(T,$cond_i$): Agent U_i receives all T objects that satisfy $cond_i$
.

Example. Say that T contains customer records for a given company *A*. Company A hires a marketing agency U_1 to do an online survey of customers. Since any customers will do for the survey, U_1 requests a sample of 1,000 customer records. At the same time, company A subcontracts with agent U_2 to handle billing for all California customers. Thus, U_2 receives all T records that satisfy the condition "state is California".

Although it is not discussed here, this model can be easily extended to requests for a sample of objects that satisfy a condition (e.g., an agent wants any 100 California customer records). Also note that the randomness of a sample is not considered. (The assumption is that, if a random sample is required, there are enough T records so that the to-be-presented object selection schemes can pick random records from T.)

2.2 Guilty Agents

After giving objects to agents, suppose the distributor discovers that a set $S \subseteq T$ has leaked. This means that some third party, known as the target, has been caught having S. For instance, the target may be displaying S on their website, or as part of a legal discovery process, might have handed over S to the distributor.

The agents $U_1. . .U_n$ are suspected to have leaked the data, as they were in possession of some of the data. But, the agents can deny the allegations, and argue that the target obtained the data through some other means. For instance, if one of the objects in S represents a customer X. If X is also a customer of some other company, that company might have provided the data to the target. Or it is possible that X could be reconstructed from several publicly available sources on the web.

The objective is to estimate the chances that the agent leaked data as opposed to other sources. The more data in S, the more difficult it is for the agents to deny they did not leak anything. The “rarer” the objects, it is more difficult to argue that the target obtained them through other means. The likelihood that the agents leaked the data must be found and if one of them in particular was more likely to be the leaker. For instance, an agent may be suspected more if one of the S objects was only given to him, while the other objects were given to all agents. The below model describes this intuition.

An agent U_i is said to be guilty if it contributes one or more objects to the target. The event that agent U_i is the guilty agent is symbolized by G_i and the event that agent U_i is the guilty agent for a given leaked set S by $G_i|S$. The next step is to find the probability that the agent U_i is the data leaker given the evidence S, this is denoted by $Pr\{G_i|S\}$.

3. Related Work

The guilt detection approach presented is related to the data provenance problem [3]: tracing the lineage of *S* objects implies essentially the detection of the guilty agents. Tutorial [4] provides a good overview on then research conducted in this field. Suggested solutions are domain specific, such as lineage tracing for data warehouses [5], and assume some prior knowledge on the way a data view is created out of data sources. The problem formulation with objects and sets is more general and simplifies lineage tracing, since data transformation from R_i sets to *S* is not considered.

As far as the data allocation strategies are concerned, the 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 [16], video [8], and audio data [6] whose digital representation includes considerable redundancy. Recently, [1], [17], [10], [7], and other works have also studied marks insertion to relational data. This 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 [9], [2]. 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.

4. Data Allocation Problem

The main focus is on the data allocation problem. The goal is to find how distributers can allocate data "intelligently" to agents in order to improve the chances of detecting a guilty agent? As illustrated in fig 1, four different instances of this problem are addressed, depending on the type of data requests made by agents and whether "fake objects" are allowed.

The two types of request that are being used were mentioned in section 2.1: simple and explicit. Fake objects are objects that are generated by distributors which are not in set *T*. These objects are designed to look like real objects and are given to agents along with the objects in set *T*, in order to improve the chances of detecting the guilty agent.

As mentioned earlier, there are four problem instances and represented with the names *EF*, $E\dot{F}$, *SF*, and $S\dot{F}$, where *E* stands for explicit requests, *S* for sample requests, *F* for the use of fake objects, and $\dot{F}$ for the case where fake objects are not allowed.

Note that, for simplicity, it is assumed that in the *S* problem instances, all agents make sample requests, while in the *E* instances, all agents make explicit requests. The results can be extended to handle mixed cases, with some explicit and some sample requests. For that, a small example is provided to illustrate how these mixed requests can be handled, but then do not elaborate further.

Assume that there are two agents with requests R_1 = EXPLICIT(T,$cond_1$) and R_2 = SAMPLE(T',1), where T' = EXPLICIT(T,$cond_2$). Further say that $cond_1$ is "state=CA" (objects have a state field). If agent U_2 has the same condition $cond_2$=$cond_1$, it is possible to create an equivalent problem with sample data requests on set T'. That is, the problem will be how to distribute the CA objects to two agents, with R_1 =SAMPLE(T',$|T'|$) and R_2=SAMPLE(T',1). If instead U_2 uses condition "state=NY," two different problems for sets T' and $T- T'$ can be solved. In each problem, there will be only one agent. Finally, if the conditions partially overlap, $R_1 \cap T' \neq 0$, but $R_1 \neq T'$, three different problems for sets $R_1- T'$, $R_1 \cap T'$, and $T'-R_1$ can be solved.

4.1 Fake Objects

Sometimes, the distributors may be able to add fake objects to the distributed data, in order to improve the chances of detecting agents that leak data. This may not always be available, because fake objects may impact the correctness of what agents do.
The idea of perturbing data to detect leakage is not new, e.g., [1]. However, in most cases, individual objects are perturbed, e.g., by adding random noise to sensitive records, or adding a watermark to an image. The set of distributor objects are perturbed by adding fake elements.

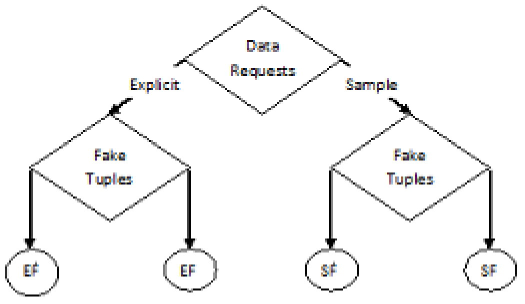

Fig.1 Leakage problem instances.

In some cases, perturbing real objects by adding fake objects may cause fewer problems. But in some applications, even a small modification can cause large problems. For example, let us consider medical records as distributed data objects and agents as hospitals. , even small modifications to the records of actual patients may be undesirable.

The use of fake objects is inspired by the use of "trace" records in phone number lists. In this case, company A sells to company B a phone number list to be used once (e.g., to send advertisements). Company A adds trace records that contain phone numbers owned by company A. Thus, each time company B makes a call to the phone numbers on the purchased phone number list, A receives a call. If company A receives more than one call, company B can be accused of improper use of data. The distributor creates and adds fake objects to the data that he distributes to agents. $F_i \subseteq R_i$ is the subset of fake objects that agent U_i receives. As discussed below, fake objects must be created carefully so that agents cannot distinguish them from real objects.

In many cases, the distributor may be limited in how many fake objects he can create. For example, objects may contain phone numbers, and each fake phone number requires a valid phone connection and someone to answer calls (otherwise, the agent may discover that the object is fake). If a call is received from someone other than the agent who was given the phone number, it is evident that the phone number was leaked. Since we have to get a real connection and have someone answer the call, resources are consumed; the distributor may have a limit of fake objects. If there is a limit, it is denoted by B fake objects.

Similarly, the distributor may want to limit the number of fake objects received by each agent so as to not arouse suspicions and to not adversely impact the agents activities. Thus, the distributor can send up to b_i fake objects to agent U_i. The creation of fake object is modeled for agent U_i as a black box function CREATEFAKEOBJECT(R_i,F_i, $cond_i$), that takes the set of all objects R_i, the subset of fake objects F_i that U_i has received so far, and $cond_i$ as input and returns a new fake object. This function needs $cond_i$ to produce a valid object that satisfies U_i condition. Set R_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. The function CREATEFAKEOBJECT() has to be aware of the fake objects F_i 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 R_i and F_i tables, respectively, and the intersection of the conditions $cond_i$. Although the implementation of CREATEFAKEOBJECT() is not dealt with, it is noted that there are two main design options. 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.

4.2 Optimization Problem

While allocating data to the agents, the distributor has one constraint and one objective. The constraint is to satisfy the agent’s requests, by providing them with all available objects that satisfy the given condition or with the number of objects they request. His objective is to find out the agent who leaked his data.
The constraint is considered as strict. 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]. The fake object distribution is considered as the only possible constraint relaxation.

The 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 [d11] discuss that to attain "perfect" privacy and security, utility must be sacrificed). Instead, the following objective is used: maximize the chances of detecting a guilty agent that leaks all his data objects.

5. Conclusions

Ideally, there would be no need to hand over sensitive data to agents that may unknowingly or maliciously leak it. Even if we had to hand over sensitive data, we could watermark each object so that we could trace its origins with absolute certainty. However, in many cases, we must indeed work with agents that may not be fully trusted, and we may not be certain if a leaked object came from an agent or from some other source, since certain data cannot admit watermarks.

In spite of these problems, 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. This model is relatively simple, but it is efficient in finding the guilty agents. Distributing objects intelligently can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive.

Our future work includes the investigation of agent guilt models in real time applications in insurance and banking sectors, so agents can be proved to be guilty and data leakage can avoided.

References

[1] 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.
[2] P. Bonatti, S.D.C. di Vimercati, and P. Samarati, “An Algebra for Composing Access Control Policies,” ACM Trans. Information and System Security, vol. 5, no. 1, pp. 1-35, 2002.
[3] 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.
[4] P. Buneman and W.-C.Tan, “Provenance in Databases,” Proc. ACM SIGMOD, pp. 1171-1173, 2007.
[5] Y. Cui and J. Widom, “Lineage Tracing for General Data Warehouse Transformations,” The VLDB J., vol. 12, pp. 41-58,2003.
[6] S. Czerwinski, R. Fromm, and T. Hodes, “Digital Music Distribution and Audio Watermarking,” http://www.scientificcommons.org/43025658, 2007.
[7] F. Guo, J. Wang, Z. Zhang, X. Ye, and D. Li, “An Improved Algorithm to Watermark Numeric Relational Data,” Information Security Applications, pp. 138-149, Springer, 2006.
[8] F. Hartung and B. Girod, “Watermarking of Uncompressed and Compressed Video,” Signal Processing, vol. 66, no. 3, pp. 283-301,1998.
[9] 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.
s[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] B. Mungamuru and H. Garcia-Molina, “Privacy, Preservation and Performance: The 3 P’s of Distributed Data Management,” technical report, Stanford Univ., 2008.
[12] V.N. Murty, “Counting the Integer Solutions of a Linear Equation with Unit Coefficients,” Math. Magazine, vol. 54, no. 2, pp. 79-81, 1981.
[13] S.U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani,“Towards Robustness in Query Auditing,” Proc. 32nd Int’l Conf. Very Large Data Bases (VLDB ’06), VLDB Endowment, pp. 151-162,2006.
[14] P. Papadimitriou and H. Garcia-Molina, “Data Leakage Detection,” technical report, Stanford Univ., 2008.
[15] P.M. Pardalos and S.A. Vavasis, “Quadratic Programming with One Negative Eigenvalue Is NP-Hard,” J. Global Optimization,vol. 1, no. 1, pp. 15-22, 1991.
[16] J.J.K.O. Ruanaidh, W.J. Dowling, and F.M. Boland, “Watermarking Digital Images for Copyright Protection,” IEE Proc. Vision,Signal and Image Processing, vol. 143, no. 4, pp. 250-256, 1996.
[17] R. Sion, M. Atallah, and S. Prabhakar, “Rights Protection for Relational Data,” Proc. ACM SIGMOD, pp. 98-109, 2003.
[18] L. Sweeney, “Achieving K-Anonymity Privacy Protection Using Generalization and Suppression,” http://en.scientificcommons.org/43196131, 2002