REVIEW ON DATA LEAKAGE DETECTION.

Naresh Bollam
M.tech(C.S.E)
Jits,karimnagar

Mr.V.Malsoru
M.tech(S.E),Assosciate Professor
Dept Of C.S.E,Jits Karimnagar

Abstract — In this paper, we present a mechanism for proof of ownership based on the secure embedding of a robust imperceptible watermark in relational data. We formulate the watermarking of relational database as a constraint optimization problem and discuss efficient techniques to solve the optimization problem and handle the constraint. We study the following problem: A data distributor has given sensitive data to a set of supposedly trusted agents. Some of the data is leaked and found in an un authorized place. The distributor must assess the likelihood that the leaked came from one or more agents, as opposed to having been independently gathered by other means. We propose data allocation strategies that improve the probability of identifying leakages. In some cases we can also inject "realistic but fake" data record to further improve our changes of detecting leakage and identifying the guilty party.

1 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.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 will be treating patients (as opposed to simply computing statis-tics), they may need accurate data for the patients.

Traditionally, leakage detection is handled by *water-marking*, e.g., a unique code is embedded in each dis-tributed copy. If that copy is later discovered in the hands of an unauthorized party,

the leaker can be iden-tified. 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 de-tecting 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. 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 5 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.

In this paper we develop a model for assessing the "guilt" of agents. We also present algorithms for dis-tributing 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 acts 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.

PROBLEM SETUP AND NOTATION

2.1 Entities and Agents

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 □ 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 same time, company A subcon-tracts 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."

2.2 Guilty Agents

Suppose that after giving objects to agents, the distrib-utor discovers that a set S □ T has 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, the target turned over S to the distributor.

Since the agents $U_1, \ldots, U_n$ have some of the data, it is reasonable to suspect them leaking the data. However, the agents can argue that they are innocent, and that the S data was obtained by the target through 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 theleaked 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. Similarly, the "rarer" the objects, the harder it is to argue that the target obtained them through other means. Not only do we want to estimate the likelihood 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* and if it contributes one or more objects to the target. We denote the event that agent U_i is guilty as G_i and the event that agent U_i is guilty for a given leaked set S as $G_i|S$. Our next step is to estimate P $r\{G_i|S\}$, i.e., the probability that agent U_i is guilty given evidence S.

3 RELATED WORK

The guilt detection approach we present is related to the data provenance problem tracing the lineage of S objects implies essentially the detection of the guilty agents. Tutorial provides a good overview on the research conducted in this field. Suggested solutions are domain specific, such as lineage tracing for data warehouses and assume some prior knowledge on the way a data view is created out of data sources. Our problem formulation with objects and sets is more general and simplifies lineage tracing, since we do not consider any data transformation from R_i sets to S.

As far as the data allocation strategies are concerned, our work is mostly relevant to watermarking that is used as a means of establishing original ownership of distributed objects. Watermarks were initially used in images video and audio data whose digital representation includes considerable redundancy. Recently,and other works have also studied marks insertion to relational data. Our approach and watermarking are similar in the sense of providing agents with some kind of receiver-identifying informa-tion. However, by its very nature, a watermark modifies the item being watermarked. If the object to be water-marked 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 mecha-nisms that allow only authorized users to access sensi-tive data through access control policies. Such ap-proaches 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 AGENT GUILT MODEL

To compute this $Pr\{G_i|S\}$, 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 S are emails of individuals. We can conduct an experiment and ask a person with
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 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 p_t, the probability that object t can be guessed by the target.

Probability p_t is analogous to the probabilities used in designing fault-tolerant systems. That is, to estimate how likely it is that a system will be operational throughout a given period, we need the probabilities that individual components will or will not fail. A component failure in our case is the event that the target guesses an object of S. The component failure is used to compute the overall system reliability, while we use the probability of guessing to identify agents that have leaked information. The component failure probabilities are estimated based on experiments, just as we propose to estimate the p_t's. Similarly, the component probabilities are usually conservative estimates, rather than exact numbers. For example, say we use a component failure probability that is higher than the actual probability, and we design our system to provide a desired high level of reliability. Then we will know that the actual system will have at least that level of reliability, but possibly higher. In the same way, if we use p_t's that are higher than the true values, we will know that the agents will be guilty with at least the computed probabilities.

5 GUILT MODEL ANALYSIS

In order to see how our model parameters interact and to check if the interactions match our intuition, in this section we study two simple scenarios. In each scenario we have a target that has obtained all the distributor's objects, i.e., $T = S$.

6 DATA ALLOCATION PROBLEM

The main focus of our paper is the data allocation prob-lem: how can the distributor "intelligently" give data to agents in order to improve the chances of detecting a guilty agent? As illustrated in Figure 2, there are four instances of this problem we address, depending on the type of data requests made by agents and whether "fake objects" are allowed.

The idea of perturbing data to detect leakage is not new,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, fake objects may cause fewer problems that perturbing real objects. For example, say 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.

Our use of fake objects is inspired by the use of "trace" records in mailing lists. In this case, company A sells to company B a mailing list to be used once (e.g., to send advertisements). Company A adds trace records that contain addresses owned by company A. Thus, each time company B uses the purchased mailing list, A receives copies of the mailing. These records are a type of fake objects that help identify improper use of data.

The distributor creates and adds fake objects to the data that he distributes to agents. We let F_i □ R_i be 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.

7 CONCLUSIONS

In a perfect world there would be no need to hand over sensitive data to agents that may unknowingly or maliciously leak it. And even if we had to hand over sen-sitive 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 must indeed work with agents that may not be 100% 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 difficulties, we have shown it is pos-sible 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. Our model is relatively simple, but we believe it captures the essential trade-offs. The algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. We have 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.

Our future work includes the investigation of agent guilt models that capture leakage scenarios that are not studied in this paper. For example, what is the appro-priate model for cases where agents can collude and identify fake tuples? A preliminary discussion of such a model is available in.Another open problem is the extension of our allocation strategies so that they can handle agent requests in an online fashion (the presented strategies assume that there is a fixed set of agents with requests known in advance).

ACKNOWLEDGMENTS

This work was supported by NSF Grant CFF-0424422 and an Onassis Foundation Scholarship. We would like thank Paul Heymann for his help with running the non-polynomial guilt model detection algorithm that we present in the Appendix of on a Hadoop cluster. We also thank Ioannis Antonellis for fruitful discussions and his comments on earlier versions of this paper.

REFERENCES

[1] R. Agrawal and J. Kiernan. Watermarking relational databases. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 155–166. VLDB Endowment, 2002.

[2] P. Bonatti, S. D. C. di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Trans. Inf. Syst. Secur.*, 5(1):1–35, 2002.

[3] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In J. V. den Bussche and
V. Vianu, editors, *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer, 2001.

[4] P. Buneman and W.-C. Tan. Provenance in databases. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1171–1173, New York, NY, USA, 2007. ACM.

[5] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. In *The VLDB Journal*, pages 471–480, 2001.

[6] S. Czerwinski, R. Fromm, and T. Hodes. Digital music distribution and audio watermarking.

F. Guo, J. Wang, Z. Zhang, X. Ye, and D. Li. *Information Security Applications*, pages 138–149. Springer, Berlin / Heidelberg, 2006. An Improve