International Journal of Advanced Computer and Mathematical Sciences
ISSN 2230-9624. Vol 3, Issue 3, 2012, pp 337-342
http://bipublication.com

IDENTIFYING LEAKAGE OF DISTRIBUTOR SENSITIVE INFORMATION BY AGENTS

D. Krishna Madhuri, A. Jagadeswara Rao, K. C. Ravi Kumar

SRI DEVI WOMEN'S ENGINEERING COLLEGE, HYDERABAD, AP, INDIA

[Received-09/09/2012, Accepted-25/09/2012]

ABSTRACT:

A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data is leaked and found in an unauthorized place (e.g., on the web or somebody's laptop). 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. We propose data allocation strategies (across the agents) that improve the probability of identifying leakages. These methods do not rely on alterations of the released data (e.g., watermarks). In some cases we can also inject "realistic but fake" data records to further improve our chances of detecting leakage and identifying the guilty party.

Index Terms—Allocation strategies, data leakage, data privacy, fake records, leakage model.

INTRODUCTION

IN 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 have 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 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 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, 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. We study 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 website, 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 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.
We develop a model for assessing the "guilt" of agents. We also present 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 that an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.

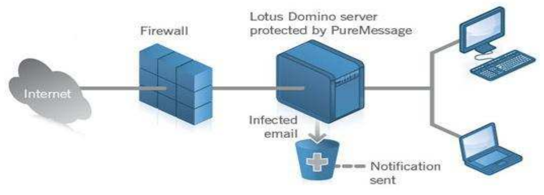

Fig 1:block diagram [19]

PROBLEM SETUP AND NOTATION

Entities and Agents

A distributor owns a set T={t1,…..tm}of valuable data objects. The distributor wants to share some of the objects with a set of agents U1, U2. . . , Un, 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 Ui receives a subset of objects $R_i \subseteq T$, determined either by a sample request or an explicit request:

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

Guilty Agents

Suppose that after giving objects to agents, the distributor discovers that a set $S \subseteq T$ 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 website, or perhaps as part of a legal discovery process, the target turned over S to the distributor.
Since the agents $U_1, \dots, U_n$ 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 were obtained by the target through other means. For example, say that 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. 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 Ui is guilty and if it contributes one or more objects to the target. We denote the event that agent U_i is guilty by G_i and the event that agent U_i is guilty for a given leaked set S by $G_i|S$. Our next step is to estimate $Pr\{G_i|S\}$, i.e., the probability that agent U_i is guilty given evidence S.

RELATED WORK

The guilt detection approach we present 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 the 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. 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 [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. Our 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.

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 that some of the objects in S are e-mails of individuals. We can conduct an experiment and ask a person with approximately the expertise and resources of the target to find the e-mail of, say, 100 individuals. If this person can find, say, 90 e-mails, then we can reasonably guess that the probability of finding one e-mail 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.

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 pts. Similarly, the component probabilities are usually conservative estimates, rather than exact numbers. For example, say that 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 pts that are higher than the true values, we will know that the agents will be guilty with at least the computed probabilities.

To simplify the formulas that we present, we assume that all T objects have the same p_t, which we call p. Our equations can be easily generalized to diverse p_ts though they become cumbersome to display.

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. In [14], we study a scenario where the actions for different objects are related, and we study how our results are impacted by the different independence assumptions.

Assumption 1. For all t, t'€ S such that t ≠ t', the provenance of t is independent of the provenance of t'. 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 itself (guessing).

To simplify our formulas, the following assumption states that joint events have a negligible probability.

Assumption 2. An object $t \in S$ can only be obtained by the target in one of the two ways as follows:

. A single agent U_i leaked t from its own R_i set.

. The target guessed (or obtained through other means) t 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.

Guilt Model Analysis:

Our model parameters interact and to check if the interactions match our intuition, in this section we study two simple scenarios as **Impact of Probability** p and **Impact of Overlap between** $_{Ri}$ **and** S. In each scenario we have a target that has obtained all the distributor's objects, i.e., T = S.

Impact of Probability p

In our first scenario, T contains 16 objects: all of them are given to agent U_1 and only eight are given to a second agent U_2. We calculate the probabilities $Pr\{G_1|S\}$ and $Pr\{G_2|S\}$ for p in the range [0, 1] and we present the results in Fig. 1a. The dashed line shows $Pr\{G_1|S\}$ and the solid line shows $Pr\{G_2|S\}$.

As p approaches 0, it becomes more and more unlikely that the target guessed all 16 values. Each agent has enough of the leaked data that its individual guilt approaches 1.

However, as p increases in value, the probability that U_2 is guilty decreases significantly: all of U_2's eight objects were also given to U_1, so it gets harder to blame U_2 for the leaks.

On the other hand, U_2's probability of guilt remains close to 1 as p increases, since U_1 has eight objects not seen by the other agent. At the extreme, as p approaches 1, it is very possible that the target guessed all 16 values, so the agent's probability of guilt goes to 0.

Impact of Overlap between Ri and S

We study two agents, one receiving all the T = S data and the second one receiving a varying fraction of the data. Fig. 1b shows the probability of guilt for both agents, as a function of the fraction of the objects owned by U_2, i.e., as a function of $|R_2 \cap S|/|S|$. In this case, p has a low value of 0.2, and U_1 continues to have all 16S objects. Note that in our previous scenario, U_2 has 50 percent of the S objects.

We see that when objects are rare (p=0:2), it does not take many leaked objects before we can say that U_2 is guilty with high confidence. This result matches our intuition: an agent that owns even a small number of incriminating objects is clearly suspicious.

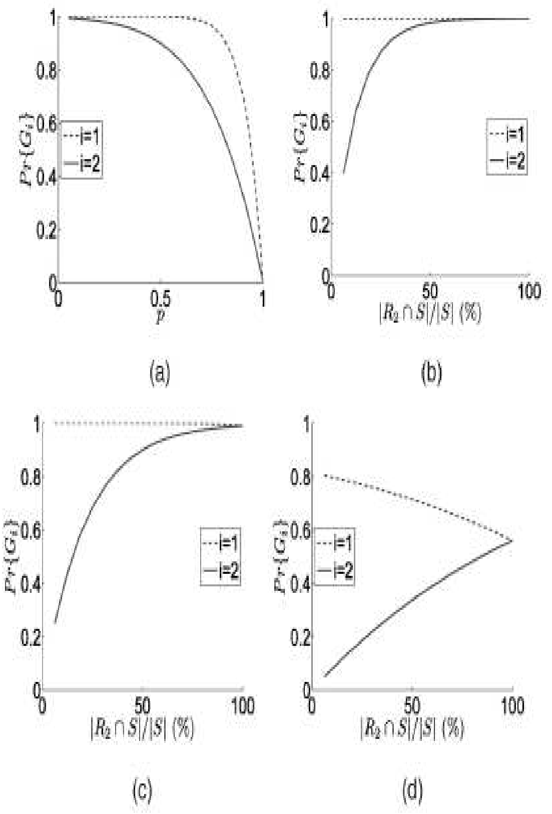

Fig1: Guilt probability as a function of the guessing probability p (a) and the overlap between S and $_{R2}$ (b)-(d) In all scenarios, it holds that R1∩S=S and |S|=16.

We see clearly that the rate of increase of the guilt probability decreases as p increases. This observation again matches our intuition: As the objects become easier to guess, it takes more and more evidence of leakage (more leaked objects owned by U_2) before we can have high confidence that U_2 is guilty.

In [14], we study an additional scenario that shows how the sharing of S objects by agents affects the probabilities that they are guilty. The scenario conclusion matches our intuition: with more agents holding the replicated leaked data, it is harder to lay the blame on any one agent.

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 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 must indeed work with agents that may not be 100 percent 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 that 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. Our model is relatively simple, but we believe that 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 appropriate model for cases where agents can collude and identify fake tuples? A preliminary discussion of such a model is available. 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).

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.

[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.

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