ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

EVALUATE FAKE OBJECTS TO PREDICTS UNAUTHENTICATED DEVELOPERS

A. Santhi Lakshmi
Roll no: 10G21D2502
M.Tech (SE),ASCET,
Gudur,Andhra Pradesh, India.
Mail: santhilakshmi14@gmail.com
Ph: 09493357739.

Prof. C. Rajendra,
Head, Dept. Of CSE,
ASCET,Gudur,
Andhra Pradesh, India,
Mail:hod.cse@audisankara.com
Ph: 09248748404.

Abstract: Modern business activities rely on extensive email exchange. Email leakages have become widespread, and the severe damage caused by such leakages constitutes a disturbing problem for organizations. In this paper we studied the following problem: In the course of doing business, sometimes the data distributor will give sensitive data to trusted third parties. Some of the data is leaked and found in an unauthorized place. The distributor cannot blame the agent without any evidence. Current approaches can detect the hackers but the total number of evidence will be less and the organization may not be able to proceed legally for further proceedings. In this work, we implement and analyze a guilt model that detects the agents using allocation strategies without modifying the original data. Our work identifies the agent who leaked the data with enough evidence. The main focus of this paper is to distribute the sensitive data "intelligently" to agents in order to improve the chances of detecting a guilty agent with strong evidence. The objective of our work is to improve the probability of identifying leakages using Data allocation strategies across the agents and also to identify the guilty party who leaked the data by injecting "realistic but fake" data records.

Keywords: Allocation strategies, sensitive data, data leakage, fake records, third parties.

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. There always remains a risk of data getting leaked from the agent. 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. But this technique requires modification of data. 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. But again it requires code modification. Watermarks can sometimes be destroyed if the data recipient is malicious. 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. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents. We develop *unobtrusive* techniques for detecting leakage of a set of objects or records. In this section 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 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

2 PROBLEM SETUP AND NOTATION

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 $R_i = Sample\ (T, m_i)$. :

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

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 we do not discuss it here, our 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 we do not concern ourselves with the randomness of a sample. (We assume 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).

3 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 ware-houses [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.

4. DATA ALLOCATION PROBLEM

The main focus of this paper is the data allocation problem: how can the distributor “intelligently” give data to agents in order to improve the chances of detecting a guilty agent? As illustrated in Fig. 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 two types of requests we handle were defined in Section 2: sample and explicit. Fake objects are objects generated by the distributor that are not in set T . The objects are designed to look like real objects, and are distributed to agents together with T objects, in order to increase the chances of detecting agents that leak data. We discuss fake objects in more detail in Section5.1.
As shown in Fig. 2, we represent our four problem instances with the names $E\bar{F}, EF, S\bar{F}$ and SF where E stands for explicit requests, S for sample requests, F for the use of fake objects, and $\bar{F}$ for the case where fake objects are not allowed.

4.1 Fake Objects

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, e.g., [1]. 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

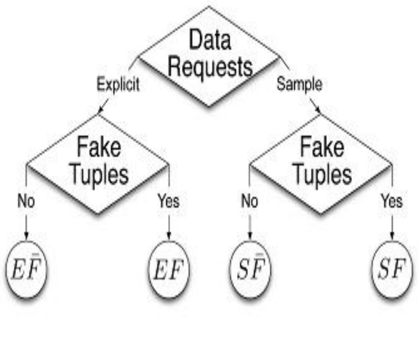

Fig. 2. Leakage problem instances.

4.2 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 portion of his data.

We consider the constraint 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]. We consider fake object distribution
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 (Mungamuru and Garcia-Molina [11] discuss that to attain "perfect" privacy and security, we have to sacrifice utility). We use instead the following objective: maximize the chances of detecting a guilty agent that leaks all his data objects.

5. Allocation Strategies:

In this section, we describe allocation strategies that solve exactly or approximately the scalar versions of (8) for the different instances presented in Fig. 2. We resort to approximate solutions in cases where it is inefficient to solve accurately the optimization problem.

We deal with problems with explicit data request, with problems with sample data requests.

5.1 Explicit Data Requests

In problems of class EF , the distributor is not allowed to add fake objects to the distributed data. So, the data allocation is fully defined by the agents' data requests. Therefore, there is nothing to optimize.

In EF problems, objective values are initialized by agents' data requests. Say, for example, that $T = \{t_1, t_2\}$ 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

$$\sum_{i=1}^{2} \frac{1}{R1} \sum_{j=1}^{2} |R_i \cap R_j| = \frac{1}{2} + \frac{1}{1} = 1.5$$

5.2 Sample Data Requests

With sample data requests, each agent Ui may receive any T subset out of $\binom{|T|}{m}$ different ones. Hence, there are $\prod_{i=1}^{n} \binom{|T|}{m}$ different object allocations. In every allocation, the distributor can permute T objects and keep the same chances of guilty agent detection. The reason is that the guilt probability depends only on which agents have received the leaked objects and not on the identity of the leaked objects.

Therefore, from the distributor's perspective, there are different allocations.

$$\frac{\prod_{i=1}^{n} \binom{|T|}{m}}{|T|}$$

5.2.1 Random

An object allocation that satisfies requests and ignores the distributor's objective is to give each agent Ui a randomly selected subset of T of size mi. We denote this algorithm by s-random and we use it as our baseline. We present s-random in two parts: Algorithm 4 is a general allocation algorithm that is used by other algorithms in this section. In line 6 of Algorithm 4, there is a call to function SELECTOBJECT() whose implementation differentiates algorithms that rely on Algorithm1 . Algorithm 2 shows function SELECTOBJECT()
for s-random.

Algorithm 1: Allocation for Sample Data Requests ($\overline{SF}$)

```
Input: $m_1, . . ., m_n$, $|T|$.
        ➢ Assuming mi ≤ $|T|$
Output: R1, . . .,Rn
1: a ← $0_{|T|}$
        ➢ a$[k]$:number of agents who have
          received object tk
2: R1 ← ϕ, . . .,Rn ← ϕ
3: *remaining* ← $\sum_{i=1}^{n} m_i$
4: **while** *remaining* > 0 **do**
 5: **for all** $i$=1, . . . , n : $|Ri|$ <m$i$ **do**
6: k ←SELECTOBJECT($i, Ri$) .
          ➢ May also use additional parameters
7: Ri ← Ri ∪ $\{tk\}$
8: a$[k]$ ←a$[k]$ + 1
9: *remaining* ←*remaining* − 1
```

Algorithm 2: Object Selection for s-random

```
1: **function** SELECTOBJECT($i; Ri$)
2: k ← select at random an element from
 set $\{k'/t_{k'} \notin R_i\}$
3: **return** k
```

In s-random, we introduce vector a ∈ $N^{|T|}$ that shows the *object sharing distribution*. In particular, element a$[k]$ shows the number of agents who receive object tk.

Algorithm s-random allocates objects to agents in a round-robin fashion. After the initialization of vectors ***d*** and a in lines 1 and 2 of Algorithm 4, the main loop in lines 4-9 is executed while there are still data objects (*remaining* > 0) to be allocated to agents. In each iteration of this loop (lines 5-9), the algorithm uses function SELECTOBJECT() to find a random object to allocate to agent Ui. This loop iterates over all agents who have not received the number of data objects they have requested.
The running time of the algorithm is $0(\tau \sum_{i=1}^{n} m_i)$ and depends on the running time τ of the object selection function SELECTOBJECT(). In case of random selection, we can have $\tau = 0(1)$ by keeping in memory a set $\{k'/t_{k'} \notin R_i\}$ for each agent Ui .

Algorithm s-random may yield a poor data allocation.Say, for example, that the distributor set T has three objects and there are three agents who request one object each. It is possible that s-random provides all three agents with the same object.

6. Experimental Results:

We implemented the presented allocation algorithms in Python and we conducted experiments with simulated data leakage problems to evaluate their performance. We present the evaluation for sample requests and explicit data requests, respectively.

6.1. Explicit Requests:

In the first place, the goal of these experiments was to see whether fake objects in the distributed data sets yield significant improvement in our chances of detecting a guilty agent. In the second place, we wanted to evaluate our optimal algorithm relative to a random allocation.

6.2 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 $\sum_{i=1}^{n} m_i$ 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. Consequently, the parameter that primarily defines the difficulty of a problem with sample data requests is the ratio.

$$\frac{\sum_{i=1}^{n} m_i}{|T|}$$

7. Conclusion:

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.

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 in [14]. 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," fact improves, on average, the min values, since the 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

[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