A Model for Data Leakage Detection

Panagiotis Papadimitriou [1], Hector Garcia-Molina [2]

Stanford University
353 Serra Street, Stanford, CA 94305, USA
[1]papadimitriou@stanford.edu
[2]hector@cs.stanford.edu

***Abstract*— We study the following problem: 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.**

I. 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 [1]. 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 identification 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.

In this paper we study *unobtrusive* 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 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.

In this paper we develop a model for assessing the "guilt" of agents. We also study algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker. In these algorithms we 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. Our algorithms are presented and evaluated in [2]; here we only briefly describe two of them to illustrate the choices that must be made.

II. Problem Setup and Notation

A. Entities and Agents

A *distributor* owns a set $T = \{t_1, t_2, t_3, \dots\}$ 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 $R_i = \textsc{Sample}(T, m_i)$: Any subset of m_i records from T can be given to U_i.
- Explicit request $R_i = \textsc{Explicit}(T, cond_i)$: Agent U_i receives *all* the T objects that satisfy $cond_i$.

Example. Say T contains customer records for a given company A. Company A hires a marketing agency U_1 to do an on-line survey of customers. Since any customers will do for the survey, U_1 requests a sample of 1000 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."

B. 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 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 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 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 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.

III. 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 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 p_t, the probability that object t can be guessed by the target.

To simplify the formulas that we present in the rest of the paper, we assume that all T objects have the same p_t, which we call p. Our equations can be easily generalized to diverse p_t's 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.

Assumption 1. *For all $t, t' \in S$ such that $t \neq t'$ the provenance of t is independent of the provenance of t'.*

To simplify our formulas, the following assumption states that joint events have a negligible probability. As we argue in the example below, this assumption gives us more conservative estimates for the guilt of agents, which is consistent with our goals.

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

- *A single agent U_i leaked t from his own R_i set; or*
- *The target* guessed *(or obtained through other means) t without the help of any of the n agents.*

In other words, for all $t \in S$, the event that the target guesses t and the events that agent U_i ($i = 1, \ldots, n$) leaks object t are disjoint.

Before we present the general formula for computing $Pr\{G_i|S\}$, we provide a simple example. Assume that sets T, R's and S are as follows:

$$T = \{t_1, t_2, t_3\},\ R_1 = \{t_1, t_2\},\ R_2 = \{t_1, t_3\},\ S = \{t_1, t_2, t_3\}.$$

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 the 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 it is the only agent that has this data object.

Given these values, the probability that agent U_1 is not guilty, namely that U_1 did not leak either object is:

$$Pr\{\bar{G_1}|S\} = (1 - (1-p)/2) \times (1 - (1-p)) \qquad (1)$$

Hence, the probability that U_1 is guilty is:

$$Pr\{G_1|S\} = 1 - Pr\{\bar{G_1}|S\} \qquad (2)$$

In the general case (with our assumptions), to find the probability that an agent U_i is guilty given a set S, first we compute the probability that he leaks a single object t to S. To compute this we define the set of agents $V_t = \{U_i | t \in R_i\}$ that have t in their data sets. Then using Assumption 2 and known probability p, we have:

$$Pr\{\text{some agent leaked } t \text{ to } S\} = 1 - p. \qquad (3)$$

Assuming that all agents that belong to V_t can leak t to S with equal probability and using Assumption 2 we obtain:

$$Pr\{U_i \text{ leaked } t \text{ to } S\} = \begin{cases} \frac{1-p}{|\mathbf{V}_t|}, & \text{if } U_i \in V_t \\ 0, & \text{otherwise} \end{cases} \qquad (4)$$

Given that agent U_i is guilty if he leaks at least one value to S, with Assumption 1 and Equation 4 we can compute the probability $Pr\{G_i|S\}$ that agent U_i is guilty:

$$Pr\{G_i|S\} = 1 - \prod_{t \in S \cap R_i} \left(1 - \frac{1-p}{|V_t|}\right) \qquad (5)$$

IV. Data Allocation Problem

The main focus of our work is the data allocation problem: how can the distributor "intelligently" give data to agents to improve the chances of detecting a guilty agent? As illustrated in Figure 1, there are four instances of this problem we address, depending on the type of data requests made by agents (E for Explicit and S for Sample requests) and whether "fake objects" are allowed (F for the use of fake objects, and $\overline{F}$ for the case where fake objects are not allowed).

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 the T objects, in order to increase the chances of detecting agents that leak data. We discuss fake objects in more detail in [2].

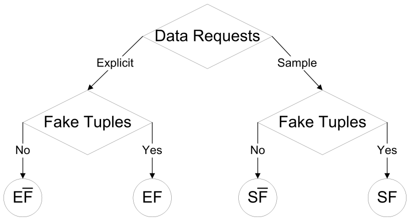

Fig. 1. Leakage Problem Instances

A. 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 as in [3] and may not provide agents with different perturbed versions of the same objects as in [4]. 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. We use instead 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. Recall that $Pr\{G_j|S=R_i\}$ or simply $Pr\{G_j|R_i\}$ 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 $\Delta(i,j)$ as:

$$\Delta(i,j) = Pr\{G_i|R_i\} - Pr\{G_j|R_i\} \qquad i,j=1,\dots,n \quad (6)$$

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 $\Delta(i,j)$ 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 $\Delta(i,j)$ value is, the easier it is to identify U_i as the leaking agent. Thus, we want to distribute data so that Δ values are large:

Problem Definition. *Let the distributor have data requests from n agents. The distributor wants to give tables $R_1,\dots,R_n$ to agents $U_1,\dots,U_n$ respectively, so that:*

- *he satisfies agents' requests; and*
- *he maximizes the guilt probability differences $\Delta(i,j)$ for all $i,j=1,\dots,n$ and $i \neq j$.*

Assuming that the R_i sets satisfy the agents' requests, we can express the problem as a multi-criterion optimization problem:

$$\underset{(\text{over } R_1,\dots,R_n)}{\text{maximize}} \quad (\dots,\Delta(i,j),\dots) \quad i \neq j \qquad (7)$$

If the optimization problem has an optimal solution, that means that there exists an allocation $\mathcal{D}^* = \{R_1^*,\dots,R_n^*\}$ such that any other feasible allocation $\mathcal{D} = \{R_1,\dots,R_n\}$ yields $\Delta^*(i,j) \geq \Delta(i,j)$ for all i,j. If there is no optimal allocation $\mathcal{D}^*$, a multi-criterion problem has Pareto optimal allocations. If $\mathcal{D}^{po} = \{R_1^{po},\dots,R_n^{po}\}$ is a Pareto optimal allocation, that means that there is no other allocation that yields $\Delta(i,j) \geq \Delta^{po}(i,j)$ for all i,j.

B. Objective Approximation

We can approximate the objective of Equation 7 with Equation 8 that does not depend on agents' guilt probabilities and therefore on p.

$$\underset{(\text{over } R_1,\dots,R_n)}{\text{minimize}} \quad \left(\dots,\frac{|R_i \cap R_j|}{|R_i|},\dots\right) \quad i \neq j \qquad (8)$$

This approximation is valid if minimizing the relative overlap $\frac{|R_i \cap R_j|}{|R_i|}$ maximizes $\Delta(i,j)$. The intuitive argument for this approximation is that the fewer leaked objects set R_j contains, the less guilty agent U_j will appear compared to U_i (since $S=R_i$). In [2] we prove that problems 7 and 8 are equivalent if each T object is allocated to the same number of agents, regardless of who these agents are.

We present two different scalar versions of our problem in Equations 9a and 9b. We will refer to objective 9a as the *sum-objective* and to objective 9b as the *max-objective*.

$$\underset{(\text{over } R_1,\dots,R_n)}{\text{minimize}} \quad \sum_{i=1}^{n} \frac{1}{|R_i|} \sum_{\substack{j=1 \\ j\neq i}}^{n} |R_i \cap R_j| \qquad (9a)$$

$$\underset{(\text{over } R_1,\dots,R_n)}{\text{minimize}} \quad \max_{\substack{i,j=1,\dots,n \\ j\neq i}} \frac{|R_i \cap R_j|}{|R_i|} \qquad (9b)$$

Both scalar optimization problems yield the optimal solution of the problem of Equation 8, if such solution exists. If there is no global optimal solution, the sum-objective yields the Pareto optimal solution that allows the distributor to detect the guilty agent on average (over all different agents) with higher confidence than any other distribution. The max-objective yields the solution that guarantees that the distributor will detect the guilty agent with a certain confidence in the worst case. Such a guarantee may adversely impact the average performance of the distribution.

```
Input: $R_1, \ldots, R_n, cond_1, \ldots, cond_n, b_1, \ldots, b_n, B$
Output: $R_1, \ldots, R_n, F_1, \ldots, F_n$
 1: $\mathbf{R} \leftarrow \emptyset$                    ▷ Agents that can receive fake objects
 2: for $i = 1, \ldots, n$ do
 3:     if $b_i > 0$ then
 4:         $\mathbf{R} \leftarrow \mathbf{R} \cup \{i\}$
 5:     $F_i \leftarrow \emptyset$                       ▷ Set of fake objects given to agent $U_i$
 6: while B $> 0$ do
 7:     $i \leftarrow$ SELECTAGENTATRANDOM($\mathbf{R}, R_1, \ldots, R_n$)
 8:     $f \leftarrow$ CREATEFAKEOBJECT($R_i, F_i, cond_i$)
 9:     $R_i \leftarrow R_i \cup \{f\}$
10:     $F_i \leftarrow F_i \cup \{f\}$
11:     $b_i \leftarrow b_i - 1$
12:     if $b_i = 0$ then
13:         $\mathbf{R} \leftarrow \mathbf{R} \backslash \{R_i\}$
14:     $B \leftarrow B - 1$
```

Algorithm 1: Random Fake Object Allocation

V. Allocation Strategies for Explicit Requests

In this paper we discuss only allocation strategies that are applicable to problem instances with explicit data requests. Strategies for the rest of the instances and an extensive experimental evaluation are available in [2].

In problems of class $E\overline{F}$ 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:

$$\sum_{i=1}^{2} \frac{1}{|R_i|} \sum_{\substack{j=1 \\ j \neq i}}^{n} |R_i \cap R_j| = \frac{1}{2} + \frac{1}{1} = 1.5.$$

The distributor cannot remove or alter the R_1 or R_2 data to decrease the overlap $R_1 \cap R_2$. However, say the distributor can create one fake object ($B = 1$) and both agents can receive one fake object ($b_1 = b_2 = 1$). In this case, the distributor can add one fake object to either R_1 or R_2 to increase the corresponding denominator of the summation term. Assume that the distributor creates a fake object f and he gives it to R_1. Agent U_1 has now $R_1 = \{t_1, t_2, f\}$ with $F_1 = \{f\}$ and the value of the sum-objective decreases to $\frac{1}{3} + \frac{1}{1} = 1.33 < 1.5$.

If the distributor is able to create more fake objects, he could further improve the objective. We present in Algorithm 1 a strategy for randomly allocating fake objects.

In lines 1-5, Algorithm 1 finds agents that are eligible to receiving fake objects in O(n) time. Then, in the main loop, the algorithm creates one fake object in every iteration and allocates it to a randomly selected agent from set $\mathbf{R}$ in line 7. The main loop takes O(B) time. Hence, the running time of the algorithm is O$(n + B)$.

If $B \geq \sum_{i=1}^{n} b_i$, the algorithm minimizes every term of the objective summation by adding the maximum number b_i of fake objects to every set R_i, yielding the optimal solution. Otherwise, if $B < \sum_{i=1}^{n} b_i$ (as in our example where $B = 1 < b_1 + b_2 = 2$) the algorithm just selects at random the agents to provide with fake objects. We return back to our example and see how the objective would change if the distributor adds fake object f to R_2 instead of R_1. In this case the sum-objective would be $\frac{1}{2} + \frac{1}{2} = 1 < 1.33$.

```
1: function SELECTAGENT($\mathbf{R}, R_1, \ldots, R_n$)
2:     $i \leftarrow \underset{i':R_{i'} \in \mathbf{R}}{\text{argmax}} \left(\frac{1}{|R_{i'}|} - \frac{1}{|R_{i'}|+1}\right) \sum_j |R_{i'} \cap R_j|$
3:     return $i$
```

Algorithm 2: Optimal Agent Selection

The reason why we got a greater improvement is that the addition of a fake object to R_2 has greater impact on the corresponding summation terms, since

$$\frac{1}{|R_1|} - \frac{1}{|R_1| + 1} = \frac{1}{6} < \frac{1}{|R_2|} - \frac{1}{|R_2| + 1} = \frac{1}{2}.$$

The left side of the inequality corresponds to the objective improvement after the addition of a fake object to R_1 and right part to R_2. We can use this observation to improve Algorithm 1 with the use of procedure SELECTAGENT() of Algorithm 2 in place of procedure SELECTAGENTATRANDOM() in line 7.

Algorithm 2 makes a greedy choice by selecting the agent that will yield the greatest improvement in the sum-objective. The cost of this greedy choice in O(n^2) in every iteration. The overall running time of the resulting algorithm is O$(n + n^2 B) = $ O$(n^2 B)$. This greedy approach finds an optimal distribution with respect to both optimization objectives defined in Equation 9 (proof in [2]).

VI. 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% trusted, and we may not be certain if a leaked object came from an agent or from some other source. In spite of these difficulties, we have shown it is possible to assess the likelihood 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. Our model is relatively simple, but we believe it captures the essential trade-offs.

References

[1] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," River Edge, NJ, USA, pp. 571–588, 2002.

[2] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," Stanford University, Tech. Rep., 2008. [Online]. Available: http://dbpubs.stanford.edu/pub/2008-23

[3] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani, "Towards robustness in query auditing," in *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, pp. 151–162.

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