Data Leakage Detection

Sachin A. Bansode, Computer Department, Sinhgad Institute of Technology, Lonavala, India
Uday M. Jadhav, Computer Department, Sinhgad Institute of Technology, Lonavala, India
Prof. N. K. Patil, Computer Department, Sinhgad Institute of Technology, Lonavala, India

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 are 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 dependently 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.

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

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 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. 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. 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 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. In this paper, 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. We start by introducing our problem setup and the notation we use. We present a model for calculating “guilt” probabilities in cases of data leakage. Then, we present strategies for data allocation to agents. Finally, we evaluate the strategies in different data leakage scenarios, and check whether they indeed help us to identify a leaker.
2. Existing System
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. E.g. 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.

2.1. Digital Watermark
Digital watermarking is a technology for embedding various types of information in digital content. In general, information for protecting copyrights and proving the validity of data is embedded as a watermark. A digital watermark is a digital signal or pattern inserted into digital content. The digital content could be a still image, an audio clip, a video clip, a text document, or some form of digital data that the creator or owner would like to protect. The main purpose of the watermark is to identify who the owner of the digital data is, but it can also identify the intended recipient. Why do we need to embed such information in digital content using digital watermark technology? The Internet boom is one of the reasons. It has become easy to connect to the Internet from home computers and obtain or provide various information using the World Wide Web (WWW).

All the information handled on the Internet is provided as digital content. Such digital content can be easily copied in a way that makes the new file indistinguishable from the original. Then the content can be reproduced in large quantities. For example, if paper bank notes or stock certificates could be easily copied and used, trust in their authenticity would greatly be reduced, resulting in a big loss. To prevent this, currencies and stock certificates contain watermarks. These watermarks are one of the methods for preventing counterfeit and illegal use. Digital watermarks apply a similar method to digital content. Watermarked content can prove its origin, thereby protecting copyright. A watermark also discourages piracy by silently and psychologically deterring criminals from making illegal copies.

3. Proposed System
This paper we are going to implement by using fake object addition technique and 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. 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.

3.1 Allocation Strategies
In this section, we describe allocation strategies that solve exactly or approximately the scalar versions for the different instances presented in different forms. We resort to approximate solutions in cases where it is inefficient to solve accurately the optimization problem. 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 e-optimal algorithm relative to a random allocation.

Algorithm 1: Allocation for Explicit Data Requests (EF)

Step 1: Calculate total fake records as sum of fake records allowed.
Step 2: While total fake objects > 0
Step 3: Select agent that will yield the greatest improvement in the sum objective i.e.
\( i = \arg \max (1) \)…………………………
Step 4: Create fake record
Step 5: Add this fake record to the agent and also to fake record set.
Step 6: Decrement fake record from total fake record set. Algorithm makes a greedy choice by selecting the agent that will yield the greatest improvement in the sum-objective.
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 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.

**Algorithm 2. Allocation for Sample Data Requests (SF)**

Step 1: Initialize $\text{Min\_overlap} \leftarrow 1$, the minimum out of the maximum relative overlaps that the allocations of different objects to

Step 2: for $k \in \{k \mid \}$ do

Initialize $\text{max\_rel\_ov} \leftarrow 0$, the maximum relative overlap between and any set that the allocation of to

Step 3: for all $j = 1, \ldots, n : j = i$ and do

Calculate absolute overlap as

Calculate relative overlap as

Step 4: Find maximum relative as

If $\text{max\_rel\_ov} \leq \text{min\_overlap}$ then

Return $\text{ret\_k}$

It can be shown that algorithm s-max is optimal for the sum-objective and the max-objective in problems where $M \leq |T|$ and $n < |T|$. It is also optimal for the max-objective if $|T| \leq M \leq 2 |T|$ or all agents request data of the same size.

With sample data requests, each agent $Ui$ may receive any $T$ from a subset out of different ones. Hence, there are different 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. An object allocation that satisfies requests and ignores the distributor’s objective is to give each agent a unique subset of $T$ of size $m$. The s-max algorithm allocates to an agent the data record that yields the minimum increase of the maximum relative overlap among any pair of agents. The s-max algorithm is as follows.

![Block diagram]

**3. Agent Guilt Model**

Before we present the general formula for computing the probability $\Pr\{Gi|S\}$ that an agent $Ui$ is guilty, we provide a simple example with set.

$T = \{t_1, t_2, t_3\}$ // distributor set $T$

$R_1 = \{t_1, t_2\}$ // agent set $R_1$

$R_2 = \{t_1, t_3\}$ // agent set $R_2$

$S = \{t_1, t_2, t_3\}$ // Target leaked set $S$

Here all three objects $t_1, t_2$ and $t_3$ leaked by $S$. Distributor gives objects $t_1$ to both $U_1$ and $U_2$.

Means probability of leaking object $t_1$ is get divided in $U_1$ and $U_2$. Here we have following cases:

1. The target guessed $t_1$ with probability $p$.
2. Agent $U_1$ leaked $t_1$ to $S$ with probability $(1-p)/2$.
3. Agent $U_2$ leaked $t_1$ to $S$ with probability $(1-p)/2$.
4. Similarly, agent $U_1$ leaked $t_2$ to $S$ with probability $(1-p)$.
5. And agent $U_2$ leaked $t_3$ to $S$ with probability $(1-p)$.
Given this values, the probability that agent U1 is not guilty, namely that U1 and U2 did not leak either object is:
\[
\Pr\{\neg G_1 | S \} = (1-(1-p)/2) * (1-(1-p)) \quad \text{//for agent U1}
\]
\[
\Pr\{\neg G_2 | S \} = (1-(1-p)/2) * (1-(1-p)) \quad \text{//for agent U2}
\]
And, the probability that agent U1 and U2 is guilty is:
\[
\Pr\{G_1 | S \} = 1-\Pr\{\neg G_1 \} \quad \text{//for agent U1}
\]
\[
\Pr\{G_2 | S \} = 1-\Pr\{\neg G_2 \} \quad \text{//for agent U2}
\]

In general case, first we compute probability that he leaks a single object t to S.
To compute this, we define set of agents V_t = \{U_i|t in R_i\} that have t in their data sets so that we have following:
1. Probability that some agents leaked object t to S is:
\[
\Pr\{U_i \text{ leaked } t \text{ to } S \} = 1-p
\]
2. Probability that all agents leaked objects t to S is:
\[
\Pr\{U_i \text{ leaked } t \text{ to } S \} = (1-p)/|V_t|, \text{if } U_i \text{ is in } V_t
\]

4. Experimental Results
We implemented the presented allocation algorithms in Python and we conducted experiments with simulated data leakage problems to evaluate their performance.

4.1 Explicit Requests
We focus on scenarios with a few objects that are shared among multiple agents. These are the most interesting scenarios, since object sharing makes it difficult to distinguish a guilty from no guilty agents. Scenarios with more objects to distribute or scenarios with objects shared among fewer agents are obviously easier to handle. As far as scenarios with many objects to distribute and many overlapping agent requests are concerned, they are similar to the scenarios we study, since we can map them to the distribution of many small subsets. Incidentally, the two jumps in the e-optimal curve are due to the symmetry of our scenario. Algorithm e-optimal allocates almost one fake object per agent before allocating a second fake object to one of them. The presented experiments confirmed that fake objects can have a significant impact on our chances of detecting a guilty agent. Note also that the algorithm evaluation was on the original objective. Hence, the superior performance of e-optimal (which is optimal for the approximate objective) indicates that our approximation is effective.

4.1 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 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.

5. 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. 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? 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 are due to the symmetry of our scenario. Algorithm e-optimal allocates almost one fake object per agent before allocating a second fake object to one of them. The presented experiments confirmed that fake objects can have a significant impact on our chances of detecting a guilty agent. Note also that the algorithm evaluation was on the original objective. Hence, the superior performance of e-optimal (which is optimal for the approximate objective) indicates that our approximation is effective.

6. References


