Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 6, June 2014, pg.873 – 877

RESEARCH ARTICLE

Data Leakage Detection Using GSM

Chandni Bhatt[1], Prof. Deepak Kapgate[2]

[1]PG Student (Dept. of CSE) & Nagpur University, India
[2]Assistant Professor (Dept. of CSE) & Nagpur University, India
[1] chandnibhatt3@gmail.com, [2] deepak.kapgate@raisoni.net

ABSTRACT: A data distributor has given sensitive data to a set of supposedly trusted agents. Sometimes data is leaked and found in unauthorized place e.g., on the web or on somebody's laptop. 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 might be given to various other companies. The owners of the data are called as distributors and the trusted third parties are called as agents. Data leakage happens every day when confidential business information such as customer or patient data, company secrets, budget information etc., is leaked out. When this information is leaked out, then the companies are at serious risk. Most probably data are being leaked from agent's side. So, company has to very careful while distributing such a data to agents. The Goal of Our project is to analyze "how the distributer can allocate the confidential data to the Agents so that the leakage of data would be minimized to a Greater Extent by finding a guilty agent".

KEYWORDS: distributor, fake object, data leakage, watermarking, guilt agent

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. 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) 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 l 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. In this paper, we will be detecting the fake agent using GSM modem which can help to protect the data from getting being leaked.

II. SYSTEM MODEL AND ASSUMPTIONS

AGENT GUILT MODEL: To compute this P rfGijSg, 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 pt 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 in the rest of the paper, we assume that all T objects have the same pt, which we call p. Our equations can be easily generalized to diverse pts 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, $t' \epsilon$ S such that $t \neq t'$, 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. 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 ϵ S can only be obtained by the target in one of the two ways as follows:

1)A single agent Ui leaked t from its own Ri set.

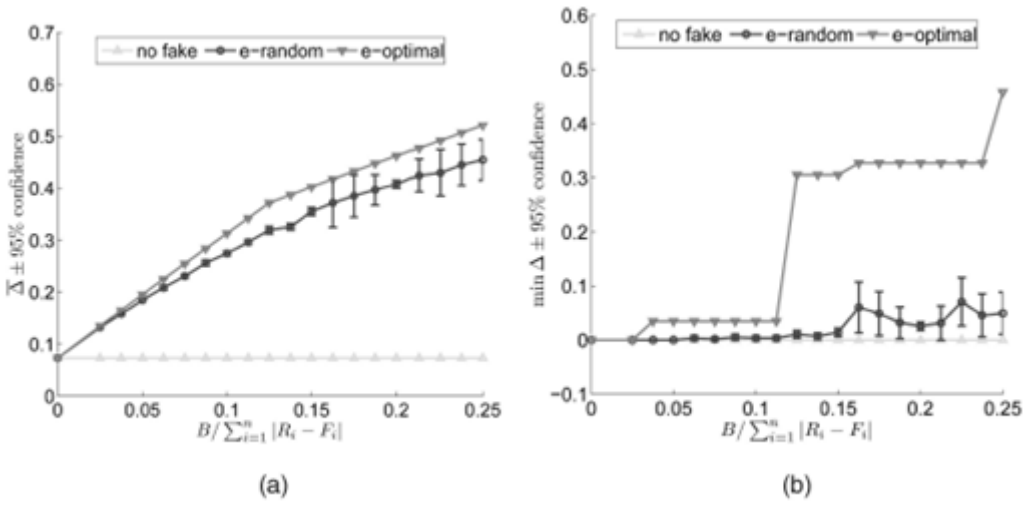
2) The target guessed (or obtained through other means) t without the help of any of the n agents.

III. 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. In some applications, fake objects may cause fewer problems that perturbing real objects. For example, say that 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 Fi Ri be the subset of fake objects that agent Ui receives. As discussed below, fake objects must be created carefully so that agents cannot distinguish them from real objects. In many cases, the distributor may be limited in how many fake objects he can create. For example, objects may contain e-mail addresses, and each fake e-mail address may require the creation of an actual inbox (otherwise, the agent may discover that the object is fake). The inboxes can actually be monitored by the distributor: if e-mail is received from someone other than the agent who was given the address, it is evident that the address was leaked. Since creating and monitoring e-mail accounts consumes resources, the distributor may have a limit of fake objects. If there is a limit, we denote it by B fake objects. Similarly, the distributor may want to limit the number of fake objects received by each agent so as to not arouse suspicions and to not adversely impact the agents' activities. Thus, we say that the distributor can send up to bi fake objects to agent Ui.

IV. RESULT



Evaluation of explicit data request algorithms. (a) Average (b) Average min

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. 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 non 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. In our scenarios, we have a set of jTj ¼ 10 objects for which there are requests by n ¼ 10 different agents. We assume that each agent requests eight particular objects out of these 10. Hence, each object is shared, on average, among agents.

$$\frac{\sum_{i=1}^{n}|R_i|}{|T|} = 8$$

Such scenarios yield very similar agent guilt probabilities and it is important to add fake objects. We generated a random scenario that yielded Average = 0:073 and min Average = 0:35 and we applied the algorithms e-random and e-optimal to distribute fake objects to the agents (see [14] for other randomly generated scenarios with the same parameters). We varied the number B of distributed fake objects from 2 to 20, and for each value of B, we ran both algorithms to allocate the fake objects to agents. We ran e-optimal once for each value of B, since it is a deterministic algorithm. Algorithm e-random is randomized and we ran it 10 times for each value of B. The results we present are the average over the 10 runs. Fig. a shows how fake object allocation can affect Average. There are three curves in the plot. The solid curve is constant and shows the Average value for an allocation without fake objects (totally defined by agents' requests). The other two curves look at algorithms e-optimal and e-random. The y-axis shows Average and the x-axis shows the ratio of the number of distributed fake objects to the total number

of objects that the agents explicitly request. We observe that distributing fake objects can significantly improve, on average, the chances of detecting a guilty agent. Even the random allocation of approximately 10 to 15 percent fake objects yields Average > 0:3. The use of e-optimal improves Average further, since the e-optimal curve is consistently over the 95 percent confidence intervals of e-random. The performance difference between the two algorithms would be greater if the agents did not request the same number of objects, since this symmetry allows non smart fake object allocations to be more effective than in asymmetric scenarios. However, we do not study more this issue here, since the advantages of e-optimal become obvious when we look at our second metric. Fig. b shows the value of min Average, as a function of the fraction of fake objects. The plot shows that random allocation will yield an insignificant improvement in our chances of detecting a guilty agent in the worst-case scenario. This was expected, since e-random does not take into consideration which agents "must" receive a fake object to differentiate their requests from other agents. On the contrary, algorithm e-optimal can yield min Average > 0:3 with the allocation of approximately 10 percent fake objects. This improvement is very important taking into account that without fake objects, values min Average and Average are close to 0. This means that by allocating 10 percent of fake objects, the distributor can detect a guilty agent even in the worst-case leakage scenario, while without fake objects, he will be unsuccessful not only in the worst case but also in average case. 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. We implemented algorithm which helps us to find the guilt agent (fake agent) who is trying to access the data unauthorisedly. Hence, we will be able to find the culprit using GSM Modem in which we will get the fake agent detail. Above graph is showing the probability of fake agent. Hence we can try to prevent data from getting leaked

V. CONCLUSION

In a perfect world there would be no need to handover 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, since certain data cannot admit watermarks. 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 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.

REFERENCES

[1] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," IEEE Transactions on Knowledge and Data Engineering, pages 51-63, volume 23, 2011.

[2] 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

[3]Hartung and kutter,‖Watermarking technique for multimedia data‖2003.

[4] Chun-Shien Lu, Member, IEEE, and Hong-Yuan Mark Liao, Member, IEEEMultipurpose Watermarking for Image Authentication and Protection

[5] Mr.V.Malsoru, NareshBollam/ REVIEW ON DATA LEAKAGE DETECTION, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 1, Issue 3, pp.1088-1091 1088 | P a g e.

[6] YIN Fan, WANG Yu, WANG Lina, Yu Rongwei. A Trustworthiness-Based Distribution Model for Data Leakage Detection: Wuhan University Journal Of Natural Sciences.

[7] L. Sweeney. Achieving k- anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzzyness and Knowledge-based Systems-2002

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

[9] P. Buneman, S. Khanna and W.C. Tan. Why and where: A characterization of data provenance. ICDT 2001, 8th International Conference, London, UK, January4-6, 2001, Proceedings, volume 1973 of Lecture Notes in Computer Science, Springer, 2001

[10] L. Sweeney. Achieving k- anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzzyness and Knowledge-based Systems-2002
[11]Edward P. Holden, Jai W. Kang, Geoffrey R. Anderson, Dianne P. Bills,Databases in the Cloud: A Work in Progress,2012
[12]Michael Miller, ‖Cloud Computing‖ Web-Based Applications that change the way you work and Collaborate Online,Pearson Education,2012
[13]Rudragouda G Patil, "Development of Data leakage Detection Using Data Allocation Strategies International Journal of Computer Applications in Engineering Sciences[VOL I, ISSUE II, JUNE 2011
[14]Detection of Guilty Agents, S.Umamaheswari #1H.Arthi Geetha #2 #1.2M.E II Year, Department Of Computer Science, Coimbatore Institute of Engineering and Technology; Coimbatore, Tamilnadu, India
[15]N.Sandhya , K. Bhima , G. Haricharan Sharma," Exerting Modern Techniques for Data Leakage Problems Detect ".International Journal of Electronics Communication and Computer Engineering-2012, Volume 3, Issue 1.
[16]ArchanaVaidya, PrakashLahange, Kiran More, ShefaliKachroo&NiveditaPandey. "DATA LEAKAGE DETECTION ". IJAET, 2012
[17] Data Leakage Prevention: A news letter for IT Professionals Issue 5 P.P

