International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.1, Jan-Feb 2012 pp-479-483 ISSN: 2249-6645

An Improved Collude Attack Prevention for Data Leakage

[1]Keerthana.P, [2]Narmadha.R.P

[1](Final ME (CSE), Sri Shakthi Institute Of Engineering and Technology, Coimbatore) India.
[2](Assistant Professor, Sri Shakthi Institute Of Engineering and Technology, Coimbatore) India

ABSTRACT:

A data distributer needs to secure the sensitive data at the time of distributing to the different agents. The sensitive data should be transferred in a secured way and it should not be leaked to some other persons. The existing system uses the technique called Watermarking to prevent the leakage. In the other techniques, the fake objects are attached with the real objects to detect the leakage and the guilty agent who leaking the data. The disadvantages of this system are the originality of the data is lost and some data cannot be transformed. In the proposed system, it concentrates on preventing two agents to compare and extract the fake objects. Symmetric Inference Model (SIM) is introduced for cases where agents can collude and identify fake tuples. In this technique it uses the symmetric inference graph approach for the symmetric inference model that represents the possible colluding attacks from any agents to the different data allocation strategies. SIM represents dependent and semantic relationships among attributes of all the entities in the information system. This prevents the agents from comparing their data with one another to identify fake objects.

Index terms- Allocation strategies, data leakage, data privacy, fake records, leakage model, Symmetric inference model.

I.INTRODUCTION

While doing business, sometimes we have to transfer the sensitive data to trusted third parties. For example, a company may have partnerships with many companies. They need to share their sensitive data. We call the customer as agents and the owner of data as distributor. Our main aim is to detect when the data is leaked by agents and also to find who had leaked out the data.

In some cases the original sensitive data is modified and handed over to the agents. The technique used to modify the data and make "less sensitive" is perturbation. But there are some cases like medical researches and payroll in which the ranges of data should not be modified as they need accurate data for treating patients in case of medical and correct account number for salary calculation in payroll case.

At first the leakage is controlled by the method called watermarking. In that a unique code will be embedded with the distributed copy. If the data is found in unauthorized parties we can identify the leaker. The disadvantage in this is the original copy need to be modified. And also in some cases the watermarks can be destroyed if the recipient is malicious.

After giving certain set of objects to agents, the distributor may find some data in unauthorized place. For example if it s found in some web site, we can identify the leaker with the help of cookies. But with a single cookie we can't proof his leakage. So if we have strong information like four or five cookies we can decide what to do with that agent.

In this paper we develop a model to protect the data if the agents decide to meet and find the fake objects by comparing with their records. Here the algorithms are designed to distribute the data. Then the fake objects are attached with the original data and distributed. The fake objects will look like the original to the agents. They will act as watermark to find the leaker of data. It will turn up and intimate the distributor, and then the distributor can be confident that the agent is guilty agent.

Entities and Agents

The distributor wants to share some of the objects with a set of agents U1, U2, . . . , Un, but does not wish the objects to 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 Ri in T, determined either by a sample request or an explicit request:

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

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. Although we do not deal with the implementation of CREATE FAKE OBJECT(), we note that there are two main design options. The function can either produce a fake object on demand every time it is called or it can return an appropriate object from a pool of objects created in advance.

In section 2 we start by describing the relative work whereas in section 3 discuss briefly about the analysis of agent guilt model and the various data allocation strategies respectively. The experimental results and the technique to calculate the probability of agent colluding attacks is explained in section 4.

II.AN OVERVIEW OF RELATED WORK

The guilt detection approach we present is related to the data provenance problem: tracing the lineage of S objects implies essentially the detection of the guilty agents. Tutorial provides a good overview on the research conducted in this field. Suggested solutions are domain specific, such as lineage tracing for data warehouses and assume some prior knowledge on the way a data view is created out of data sources.

In the watermarking relational databases paper, the watermark is inserted in the range of bits. The bits get replaced to act as the watermark. It helps to prevent the attack from the malicious attack and bit flipping attack and so on. The disadvantage is the watermarking is not developed for the non-numeric attributes.

The goal of watermarking is to insert the mark in the object without destroy the value of the object and it is difficult for the adversary to remove or alter the mark beyond detection without destroying the original value. Database semantics and structured data are the challenges faced during this process.

The protection is also achieved through the process of k-anonymity. In this k-anonymity provides privacy protection by guaranteeing that each record relates to at least k individuals even if the released records are directly linked to external information. It provides a formal presentation of achieving k-anonymity using generalization and suppression. Generalization involves replacing a value with a less specific but semantically consistent value. Suppression involves not releasing a value at all.

Finally, there are also lots of other works on mechanisms that allow only authorized users to access sensitive data through access control policies.

III. AGENT GUILT MODEL ANALYSIS

For finding the guilty agent, the probability is to be calculated. For example if we are giving data to two agents then the probability of the agent to be guilty is 0.5. 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. Similarly, the component probabilities are usually conservative estimates, rather than exact numbers.

Data Allocation Strategies

The main problem is to allocate the data to the agents with the high probability of finding the guilty agent. For that four possibilities are found. As illustrated in Fig. 1, there are four instances of this problem we address, depending on the type of data requests made by agents whether it is explicit data request or sample data request and whether the "fake objects" are allowed or not.

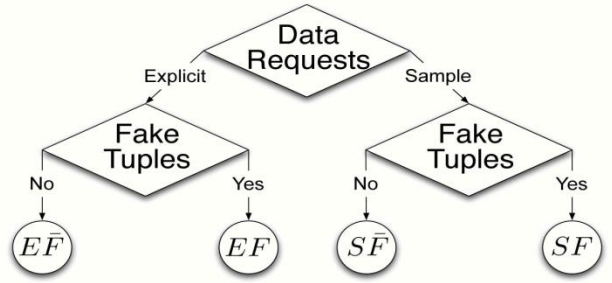

Fig. 1. Leakage problem instances.

While allocating data to the agents, the constraint is to satisfy the agent's request by providing the entire request that are available and the objective is to detect the agent if he leaks the data.

A. Explicit data request

Explicit Data Request with e-random

In this model we present an approach of explicit data request based on e-random. Here we combine the allocation of the explicit data request with the agent selection of e-random. We use e-random as our baseline in our comparisons with other algorithms for explicit data requests. Initially we the finds agents that are eligible to receiving fake objects in O(n) time. Then, the algorithm creates one fake object in every iteration and allocates it to random agent. The main loop takes O(B) time. Hence, the running time of the algorithm is O(n + B).

Algorithm 1. Allocation for Explicit Data Requests (EF)

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: R ←ϕ . Agents that can receive fake objects
2: for i = 1,. . . ,n do
3: if $b_i > 0$ then
4: R ←RU{i}
5: Fi ←ϕ
6: while B > 0 do
7: i ←SELECTAGENT(R,$R_1, \ldots, R_n$)
8: f ←CREATEFAKEOBJECT(R_i, F_i, $cond_i$)
9: R_i ←R_i U {f}
10: F_i ←F_i U {f}
11: b_i ←b_i - 1
12: if b_i = 0then
13: R ←R\{R_i}
14: B←B – 1

Algorithm 2. Agent Selection for e-random

1: function SELECTAGENT (R,R1, . . .,Rn)
2: i ←select at random an agent from R
3: return i

Explicit Data Request with e-optimal

Still to improve the algorithm for allocation explicit data request we are combining this algorithm with the agent selection for e-optimal method. This algorithm based on e-optimal makes a greedy choice by selecting the agent that will yield the greatest improvement in the sum-objective. The cost of this greedy choice is $O(n^{2)}$ in every iteration. The overall running time of e-optimal is $O(n + n^2B) = O(n^2B)$. Here the optimal value is calculated between the different agents and that value is used to attach the fake records in the agent. For calculating the optimal value the co-occurrences between the agents is calculated and the agent with high value is attached with the fake record.

Algorithm 3. Agent Selection for e-optimal

1: function SELECTAGENT ($R,R_1, . . .,R_n$)

$$i \leftarrow \underset{i':R_{i'} \in \mathbf{R}}{\operatorname{argmax}} \left(\frac{1}{|R_{i'}|} - \frac{1}{|R_{i'}|+1} \right) \sum_j |R_{i'} \cap R_j|$$

2: return i

B. Sample data request

Sample Data Request with s-random

Here in this module we present the sample data request with s-random. Here in this method we present the object selection for s-random. In s-random, we introduce vector $a \in N^{|T|}$ that shows the object sharing distribution. In particular, element a[k] shows the number of agents who receive object t_k. Algorithm s-random allocates objects to agents in a round-robin fashion. After the initialization of vectors d and a, the main loop is executed while there are still data objects (remaining > 0) to be allocated to agents. In each iteration of this loop, the algorithm uses function SELECTOBJECT()to find a randomobject 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 $O(\tau \sum_{i=1}^{n} m_i)$ and depends on the running time τ of the object selection function SELECTOBJECT(). In case of random selection, we can have τ = O(1) by keeping in memory a set $\{k' \mid t_{k'} \notin R_i\}$ for each agent U_i.

Algorithm 4. Allocation for Sample Data Requests (SF)

Input: $m_1, . . .,m_n, |T|$. Assuming $m_i \leq |T|$
Output: $R_1,. . .,R_n$
1: a ← $O_{|T|}$. a[k]:number of agents who have received object t_k
2: $R_1 \leftarrow \phi , . . .,R_n \leftarrow \phi$,
3: remaining ← $\sum_{i=1} m_i$
4: while remaining > 0 do
5: for all i = 1,. . ., n : $|R_i| < m_i$ do
6: k ←SELECTOBJECT(i,R_i) . May also use additional parameters
7: $R_i \leftarrow R_i \cup \{t_k\}$
8: a[k]← a[k]+ 1
9: remaining← remaining -1

Algorithm 5. Object Selection for s-random

1: function SELECTOBJECT(I,R_i)
2: k ←select at random an element from set $\{k^1 | t_k^1 ¢ R_i\}$
3: return k

Sample Data Request with s-overlap

In the previous section the distributor can minimize both objectives by allocating distinct sets to all three agents. Such an optimal allocation is possible, since agents request in total fewer objects than the distributor has. This is overcome by presenting an object selection approach for s-overlap. Here in each iteration of allocating sample data request algorithm, we provide agent U*i* with an object that has been given to the smallest number of agents. So, if agents ask for fewer objects than jTj, agent selection for s-optimal algorithm will return in every iterations an object that no agent has received so far. Thus, every agent will receive a data set with objects that no other agent has. The running time of this algorithm is O(1).

Algorithm 6. Object Selection for s-overlap

1: function SELECTOBJECT (i,R_i, a)
2: K← {k | k = argmin $a[k^1]$)
3: k ←select at random an element from set $\{k^1 | k^1 € K ∧ t_k ¢ R_i\}$
4: return k

Sample Data Request with s-max

In this module we present an improved algorithm than s-overlap and s-random which we used in allocation algorithm. This algorithm we present here is termed as object selection for s-max. If we apply s-max to the example above, after the first five main loop iterations in algorithm of allocating data request, the R_i sets are:

$R_1 = \{t_1, t_2\}$; $R_2 = \{t_2\}$; $R_3 = \{t_3\}$; and $R_4 = \{t_4\}$:

In the next iteration, function SELECTOBJECT() must decide which object to allocate to agent U_2. We see that only objects t_3 and t_4 are good candidates, since allocating t_1 to U_2 will yield a full overlap of R_1 and R_2. Function SELECTOBJECT() of s-max returns indeed t_3 or t_4. The running time of SELECTOBJECT() is O(|T|n).

Algorithm 7. Object Selection for s-max

1: function SELECTOBJECT (i,$R_1, . . .,R_n$;$m_1, . . .,m_n$)
2: min_overlap ←1 . the minimum out of the maximum relative overlaps that the allocations of

different objects to U_i yield
3: for k € $\{k^1 \mid t_k^1$ ¢ $R_i\}$ do
4: max_rel_ov← 0 . the maximum relative overlap between R_i and any set R_j that the allocation of t_k to U_i yields
5: for all j =1; . . . ; n : j ≠ i and t_k € R_j do
6: abs_ov ←|Ri ∩R_j |+ 1
7: rel_ov← abs_ov/min (m_i,m_j)
8: max_rel_ov ←Max (max_rel_ov, rel_ov)
9: if max_rel_ov ≤min_overlap then
10: min_overlap←max_rel_ov
11: ret_k← k
12: return ret _k

In this three overlap values are calculated. The values are absolute overlap value, relative overlap value and minimum overlap value. The minimum overlap value is nothing but the columns allocated to them. The absolute overlap value is the unique column between the agents and the relative overlap value is calculated by dividing the absolute value with the minimum value. According to this value the fake objects will be attached. The agents having the minimum overlap value is attached with the fake records.

Finally with the help of these algorithms the probability is calculated and the guilty agent can be identified. The agent having the highest probability to leak the data is considered as the guilty agent.

Symmetric Inference Model

To represent the possible colluding attacks from any agents to the different data allocation strategies, Here we use the semantic inference model. SIM represents dependent and semantic relationships among attributes of all the entities in the information system. The related attributes (nodes) are connected by three types of relation links: dependency link, schema link, and semantic link.. To evaluate the inference introduced by semantic links, we need to compute the CPT for nodes connected by semantic links.

Symmetric Inference Graph

In order to perform inference at the instance level, we instantiate the SIM with specific entity instances and generate a SIG. Each node in the SIG represents an attribute for a specific instance. The attribute nodes in the SIG have the same CPT as in the SIM because they are just instantiated versions of the attributes in entities. As a result, the SIG represents all the instance-level inference channels.

Instance level dependency link

When a SIM is instantiated, the dependency within- entity is transformed into dependency-within-instance in the SIG. Similarly, the dependency-between-related-entities in the SIM is transformed into a dependency between two attributes in the related instances. This type of dependency is preserved only if two instances are related by the instantiated schema link. That is, if attribute B in instance *e2* depends on attribute A in instance *e1*, and instances *e1* and *e2* are related by R.

Instance level schema link

The schema links between entities in the SIM represent "key, foreign-key" pairs. At instance level, if the value of the primary key of an instance *e1* is equal to the value of the corresponding foreign key in the other instance *e2* which can be represented as R(*e1*, *e2*), then connecting these two attributes will represent the schema link at the instance level. Otherwise, these two attributes are not connected.

Instance level semantic link

At the instance level, assigning the value of the source node to "unknown" disconnects the semantic link between the attributes of two instances. On the other hand, if two instances have a specific semantic relation, then the inference probability of the target node will be computed based on its CPT and the value of the source node.

IV. EXPERIMENTAL RESULTS EXPLICIT REQUESTS

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. Next, we wanted to evaluate our e-optimal algorithm realative to a random allocation.

It focuses on the 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.

In our scenarios we have a set of |T|= 10 objects for which there are requests by n= 10 different agents. We assume that each agent request 8 particular objects out these 10. Such scenarios yield very similar agent guilt probabilities and it is important to add fake objects. We generated a random scenario that yielded Δ = 0.073 and min Δ = 0.35 and we applied the algorithms e-random and e-optimal to distribute fake objects to the agents. 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.

In Figure 2(a) it shows how fake object allocation can affect negotiation of Δ. Three curves are plotted in the figure. The solid curve is constant and it shows the negotiation of Δ value for an allocation without fake objects.The other two curves look at algorithms e-optimal and e-random. The y-axis shows negotiation of Δ 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% fake objects yields negotiation of Δ > 0.3. The use of e-optimal improves Δ further, since the e-optimal curve is consistently over the 95% 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 asymmetirc scenarios.

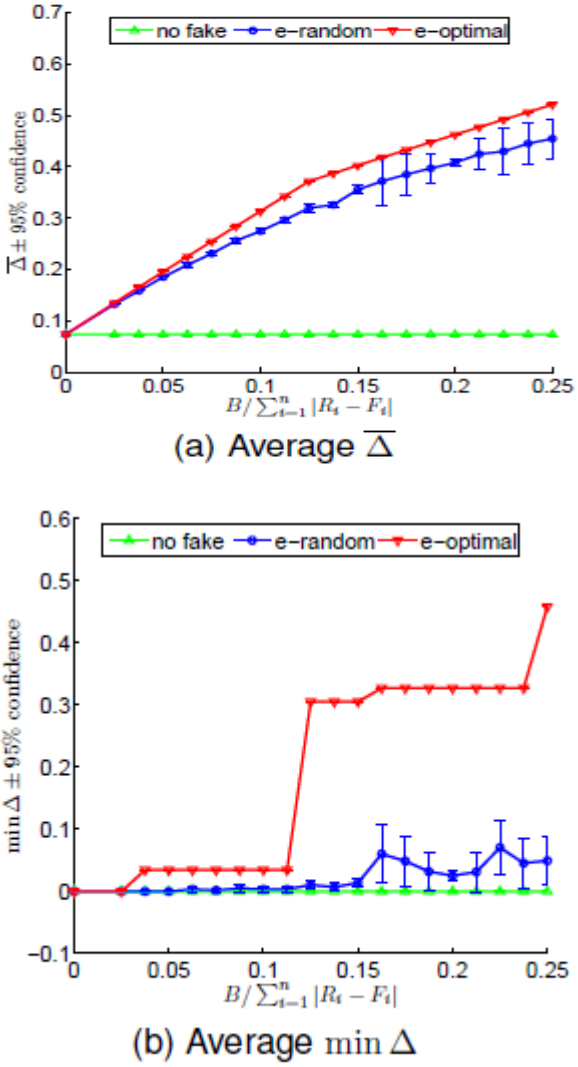

Fig 2 . Evaluation of Explicit Data Request Algorithm

Figure 2(b) shows the value of min Δ, as a function of e-optimal 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 scenarios. 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.

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 before allocating a second fake object to one of them.

The result confirms that fake objects can have a significant impact on our chances of detecting a guilty agent. Hence, the performance of e-optimal indicates that our apporximation is effective.

V. CONCLUSION

While transforming the data to the distributors the senders apply perturbation and watermarking techniques to secure them. The approaches used now in the data transformation are not able to detect the data leakage in an efficient way and also they were all restricted or made impossible to satisfy agent's request. They also not help much in finding the guilty agent who leaked the data. The originality of the data and the quality is mainly concentrated here. The techniques cannot be used to some of the sensitive data too as it is impossible to transfer or modify some of the data. So, to find the data leakage and to protect the data from the guilty agents who trying to find the original data, a Semantic Inference Model is going to be introduced. This model is proposed in such a way that it prevents the problem of leakage of data and helps in finding the guilty agent.

REFERENCES

[1] R. Agrawal and J. Kiernan. Watermarking relational databases. In VLDB *'02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 155–166. VLDB Endowment, 2002.
[2] P. Bonatti, S. D. C. di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Trans. Inf. Syst. Secur.*, 5(1):1–35, 2002.
[3] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In J. V. den Bussche and V. Vianu, editors, *Database Theory - ICDT 2001, 8th International*
[4] *Conference*, London, UK, January 4-6, 2001, Proceedings, volume 1973 of Lecture Notes in Computer Science, pages 316–330. Springer, 2001.
[5] P. Buneman and W.-C. Tan. Provenance in databases. In SIGMOD *'07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1171–1173, New York, NY, USA, 2007. ACM.
[6] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. In The *VLDB Journal*, pages 471–480, 2001.
[7] S. Czerwinski, R. Fromm, and T. Hodes. Digital music distribution and audio watermarking.
[8] F. Guo, J. Wang, Z. Zhang, X. Ye, and D. Li. *Information Security Applications*, pages 138–149. Springer, Berlin / Heidelberg, 2006. An Improved Algorithm to Watermark Numeric Relational Data.
[9] F. Hartung and B. Girod. Watermarking of uncompressed and compressed video. *Signal Processing*, 66(3):283–301, 1998.
[10] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.
[11] Y. Li, V. Swarup, and S. Jajodia. Fingerprinting relational databases: Schemes and specialties. *IEEE Transactions on Dependableand Secure Computing*, 02(1):34–45, 2005.
[12] B. Mungamuru and H. Garcia-Molina. Privacy, preservation and performance: The 3 p's of distributed data management. Technicalreport, Stanford University, 2008.