Relational Data Leakage Detection using Fake Object and Allocation Strategies

Jaymala Chavan
Thakur College of Engg. & Technology

Priyanka Desai
Thakur College of Engg. & Technology

ABSTRACT
In today's world, there is need of many companies to outsource their sure business processes (e.g. marketing ,human resources) and related activities to a third party like their service suppliers. In many cases the service supplier desires access to the company's confidential information like customer data, bank details to hold out their services. And for most corporations the amount of sensitive data used by outsourcing providers continues to increase**.** So in today's condition data Leakage is a Worldwide Common Risks and Mistakes and preventing data leakage is a business-wide challenge**.** Thus we necessitate powerful technique that can detect such a dishonest. Traditionally, leakage detection is handled by watermarking, Watermarks can be very useful in some cases, but again, involve some modification of the original data. So in this paper, unobtrusive techniques are studied for detecting leakage of a set of objects or records. The model is developed for assessing the "guilt" of agents. The algorithms are present for distributing objects to agents, in a way that improves our chances of identifying a leaker. Finally, consider the option of adding "fake" objects to the distributed set. The major contribution in this system is to develop a guilt model using fake elimination concept

General
Data Privacy, Data Leakage, Algorithm

Keywords
Allocation Strategies, Fake Records, Guilt Model

1. INTRODUCTION
In the business, sometimes it is necessary to send confidential data to trusted third parties. For example, a company may have partnerships with other companies that require sharing customer data. Similarly, a hospital may give patient records to researchers who will devise new treatments. Another enterprise may outsource its data processing, so data must be given to various other companies. So in this system owner of the data is called as distributor and the supposedly trusted third parties is called as agents.

The system goal is to detect Which distributor's sensitive data has been leaked by agents, and if possible to identify the agent that leaked the data.

Traditionaly, Leakage detection is handled by watermarking e.g unique code embedded in each distributed copy. But this watermarking involve some modification of original data. Furthermore watermarks sometimes can be destroyed if data recipient is malicious. But in some cases it is important not to alter the original distributor's data

It 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 which is achieved through k-anonimity privacy protection algorithm[5]. However, in some cases it is important not to alter the original distributor's data. In paper[1][10], there is an 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.

So, this paper proposed a model for assessing the "guilt" of agents on basis of fake elimination method which is proposed in this paper. An algorithms for distributing objects to agents is proposed[1][10], in a way that improves our chances of identifying a leaker. Finally, considering 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.

2. RELATED WORK
The data allocation strategies[1][10] used is more relevant to the watermarking [6],[9][11] that is used as a means of establishing original ownership of distributed objects.
The data leakage prevention based on the trustworthiness [3] is used to assess the trustiness of the agent. Maintaining the log of all agent's requests is related to the data provenance problem [7] i.e. tracing the lineage of objects. There are also different mechanisms to allow only authorized users, to access the sensitive information [4] through access control policies, but these are restrictive and may make it impossible to satisfy agent's requests.

3.PROPOSED WORK

In this paper, a model is develop for assessing the "guilt" of agents on the basis of fake object. The algorithms are presents for distributing objects to agents, in a way that improves our chances of identifying a leaker. Finally, it 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. Today the advancement in technology made the watermarking system a simple technique for data authorization. There are various software which can remove the watermark from the data and makes the data as original.

So the advantages of this system using allocation strategies and fake object is as follows:-
This system includes the data hiding along with the provisional application with which only the data can be accessed. This system gives privileged access to the administrator (data distributor) as well as the agents registered by the distributors. Only registered agents can access the system. The Agent accounts can be activated as well as Edited. The exported file will be accessed only by the system. The agent has given only the permission to access the requested data and view the data. The data can be copied by this application. If the data is leaked by the agent system and if distributor found that leaked data on websites or some other sources then distributor give leaked input set to the system. The system identity guilty Agent with their guiltiness probability value for that object.

4.PROBLEM SETUP AND NOTATION

A distributor owns a set T = {t1 . . . t_m} of valuable data objects. The distributor wants to share some of the objects with a set of agents U1; U2; . . . ; Un, but does not wish the objects be leaked to other third parties. The objects in T could be of any type and size. An agent Ui receives a subset of objects in T, i.e. Ri ⊆T, determined either by a sample request or an explicit request:

4.1 Sample request Ri = **SAMPLE** (T, mi): Any subset of mi records from T can be given to Ui.

4.2 Explicit request Ri=**EXPLICIT** (T, condi):

Agent Ui receives all T objects that satisfy condi.

Fig.2 shows system architecture with these two types of requests.

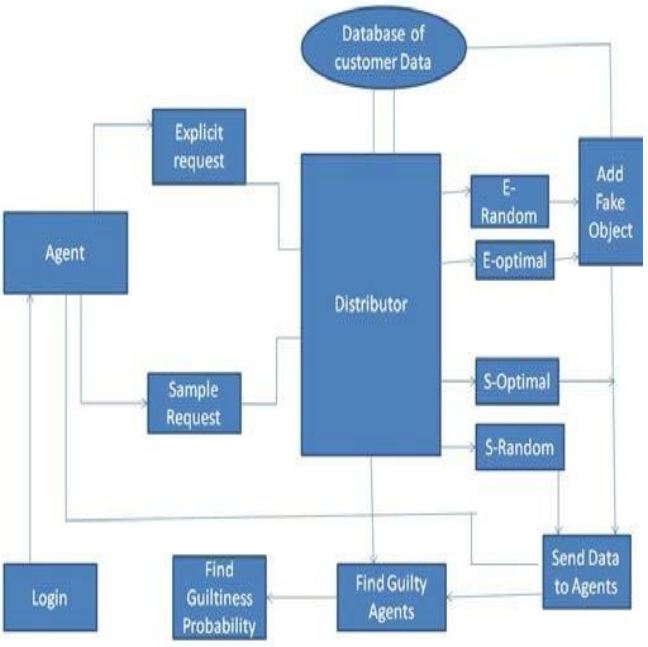

Fig 1: System Architecture

5. DATA ALLOCATION ALGORITHM

5.1 Algorithm for Explicit Data Request –

In this allocation strategy, agent request distributor data objects on a constraint i.e. distributor had to distribute data objects to agent satisfying the specified condition. For e.g. Agent request distributor for customers records with constraint "customer of state Maharashtra".

Algorithm 1 :- Allocation for Explicit Data Requests (*EF*):

Input: *R*1… *Rn*, *cond*1…*condn*, b1…b*n*, *B* // *B* – fake objects created by distributor, bi – fake objects agent Ui can receive
Output: *R*1… *Rn*, *F*1… *Fn* // *Fi* – fake object received by selected agent U*i*

1: R ← ∅ Agents that can receive fake objects
2: for i = 1, . . . , n do
3: if bi > 0 then
4: R ← R ∪ {i} // *i* – Agent that was selected to add fake objects
5: Fi ← ∅
6: while B > 0 do
7: i ← SELECTAGENT(R,R1, . . . , Rn) // *i* –selected agent either by random selection or by optimal selection
8: f ← CREATEFAKEOBJECT(Ri, Fi, condi) // black box function for fake object creation
9: Ri ← Ri∪ {f} // *f* – Fake object that was created for agent *Ui* is inserted to *f*
10: Fi ← Fi ∪ {f}
11: bi ← bi − 1
12: if bi = 0 then
13: R ← R\{Ri}
14: B ← B – 1

Algorithm 1 is a general "driver" that will be used by other strategies i) e-random allocation strategies. ii)e-optimal allocation strategies.

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

Algorithm 2 with algorithm 1 is a strategy for randomly allocating fake objects. In lines **1-5**, Algorithm 1 finds agents that are eligible to receiving objects in O (*n*) time. Then in main loop in lines **6- 14**, 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 3 Agent Selection for e-optimal

1: **function** SELECTAGENT(R, *R*1…

*R*n)
2: $i \leftarrow$ argmax $(1/|Ri^l| - 1/(|Ri^l| + 1)) \Sigma j\ |Ri^l \cap Rj|\ i^l{:}Ri^l \in$
R 3: return *i*

Algorithm 3 makes a greedy choice by selecting the agent that will yield the greatest improvement in the sum-objective. The overall running time of e-optimal is O (*n* + *n*2*B*) = O (*n*2*B*).

5.2 Algorithm for Sample Request

In this allocation strategy agent doesn't demand data objects, distributor itself distribute sample of data objects to agent. For e.g., agent request 50 customer records, distributor will distribute customer records with any condition by picking samples randomly.

Sample request Ri = SAMPLE (T, mi): Any subset of mi records from T objects that satisfy condi.
Ri – Requested subset of objects by agent Ui in T.
T – Objects with the distributor i.e., tuples in a relation or relations in a database.
mi – sample of objects requested by agent Ui.

Algorithm 4 Allocation for Sample Data Requests (SF^l)

Input: m1… mn, |T| // T- data set having objects with distributor
// assuming mi ≤ |T| //mi – sample data objects distributed to agent
Output: R1… Rn // Ri – Data set allocated to agent i which have mi sample data objects

1: a ← 0|T| a[k]:number of agents who have received object tk
2: R1 ← ∅, . . . ,Rn←∅
3: remaining ← $\Sigma i = 1^{\ n}\ mi$ // No of sample sets that we have to distribute to agents
4: while remaining > 0 do
5: for all i = 1, . . . , n : |Ri| < mi 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 4 is a general allocation algorithm that is used by other algorithms-i)s-random ii)s-optimal

Algorithm 5 shows function SELECTOBJECT for **s-random**

1: **function** SELECTOBJECT (*i*, *Ri*)
2: *k* ←select at random an element from set $\{k^l \mid tk^l \notin Ri\}$
// $tk^l \notin Ri$ – to avoid repetition of same element in the set of *R*i 3: return *k*

In line **6** of Algorithm 4 there is a call to function SELECTOBJECT () whose implementation differentiates algorithms that rely on Algorithm 4. The running time of the s-random algorithm is O ($T \Sigma i=1n$ mi).

Algorithm 6 shows function SELECTOBJECT for **s-optimal**

1: **function** SELECTOBJECT (*i*, *R*i,
a,*Fi*) 2:*K* ←{ *k* | *k* = argmin *a*[k^l] }
3: *k* ←select at random an element from set { $k^l \mid k^l \in K \cap tk^l \notin Ri$ } // element that was in K and tk' was not allocated to *Ri*
3: Fi- Add fake object in original data which was not allocated to previous agent
4: $R \leftarrow RU\{t_k\}\ U\ F_i$
5: **return** *k*

Algorithm s-random may yield a poor data allocation. Using Algorithm 6 of this paper, in each iteration of Algorithm 4 we provide agent *U*i with an object that has been given to smallest number of agents i.e. *t*k object will be given less number of agents. If agents asks for fewer objects than |*T*|, Algorithm 6 will return in every iteration an object that no agent has received so far with addition of fake object which is not received by previous agent also i.e., each agent receive distinct objects. The total running time of the algorithm i.e. 4 and 6 is O(Σ*i*=1*nmi*).

6. FAKE OBJECT

Fake objects must be created carefully so that agents cannot distinguish them from real objects. 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, it is say that the distributor can send up to bi fake objects to agent Ui Creation. The creation of fake but real-looking objects is a non-trivial problem whose thorough investigation is beyond the scope of this paper. The creation of a fake object for agent Ui as a black-box function CREATEFAKEOBJECT(R_i; F_i; $cond_i$). R_i is the set of all objects, Fi is the subset of fake objects that Ui has received so far , the function returns a new fake object. This function needs condi to produce a valid object that satisfies Ui's condition. Set Ri is needed as input so that the created fake object is not only valid but also indistinguishable from other real objects. The function CREATEFAKEOBJECT() has to be aware of the fake objects Fi added so far, again to ensure proper statistics. Although system do not deal with the implementation of CREATEFAKEOBJECT(), 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 this, system create fake objects in advance.

7. GUILT MODEL ANALYSIS

The model parameters interact and check if the interaction matches the intuition. In this section we study two simple scenarios, Impact of Probability p and Impact of Overlap between Ri and S. In each scenario a target, that has obtained all the distributor's objects, i.e., T = S.

7.1 Guilty Agents

Suppose that after giving objects to agents, the distributor discovers that a set S (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 website, or perhaps as part of a legal discovery process, the target turned over S to the distributor. Since the agents U1.. . . .Un has 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 were obtained by the target through other means. For example, say that one of the objects in S represents a customer A. Perhaps A is also a customer of some other company, and that company provided the data to the target or perhaps A can be reconstructed from various publicly available sources on the web. The 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 were only given to agent U1, while the other objects were given to all agents, we may suspect U1 more.

The model captures this intuition. We say an agent Ui is guilty and if it contributes one or more objects to the target. It denote the event that agent Ui is guilty by Gi and the event that agent Ui is guilty for a given leaked set S by G_i |
S. Our next step is to estimate $P_r\{ G_i| S \}$, i.e., the probability that agent Ui is guilty given evidence S.

7.2 Guilty Agents on Basis of Fake Object Eliminaton Method

This paper extend guilt model analysis by addition of fake object elimination method. In this system can find particular guilty agent on basis of fake object only. For instance, if one of the S objects were given to multiple agents. Then it may suspect multiple agents because they contributes their one or more objects to the leaked set that object may be original object or fake object. But using this fake elimination method it only matches fake object in leaked set with fake object of agent's received set. So it can say perticular agent Ui is guilty if he contributes real objects with his unique fake object to the leaked set. Thus system can find particular one guilty agent among multiple agents using this method.

8. EXPECTED RESULTS

8.1 Distributor Module:

All the privileges of the system are only available with the distributor.

As shown in fig.2 Distributor can add, Edit and update agents, contact data and fake data easily using this form. Also Distributor can view agent request for data using this form

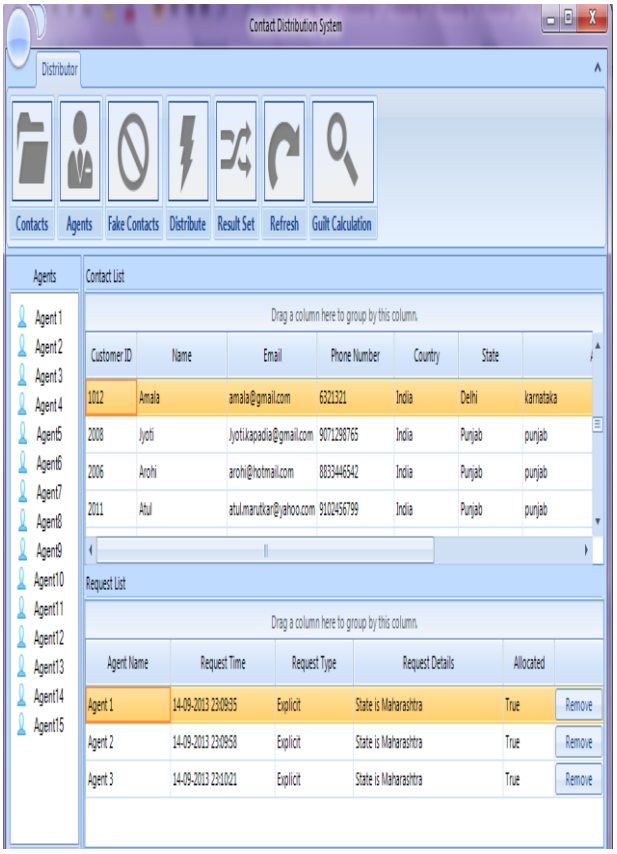

Fig.2 Distributor GUI

As shown in fig 3. Distributor can add ,edit and update customer contact data easily into the database using this form.

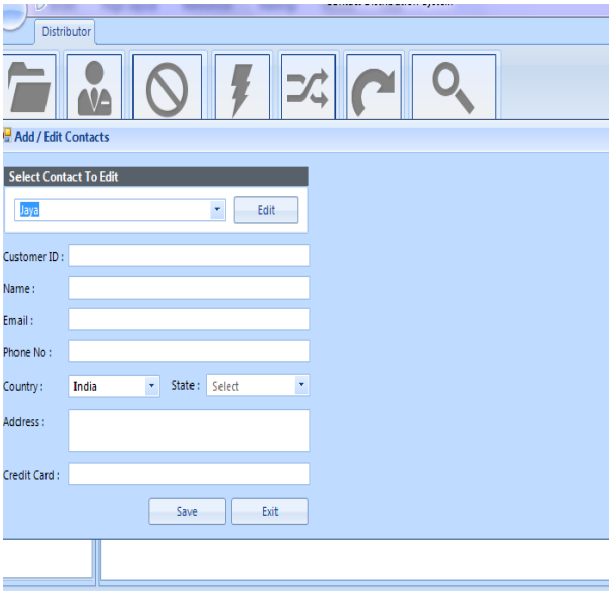

Fig.3 Distributor adding contact data into Database[2]

As shown in fig.4 Distributor Add & Manage number of Agents using this form. In this, distributor set username, password for each added agent. Also he decide whether fake contact allowed for that agent if yes then he decides maximum number of fake contact for each agent.

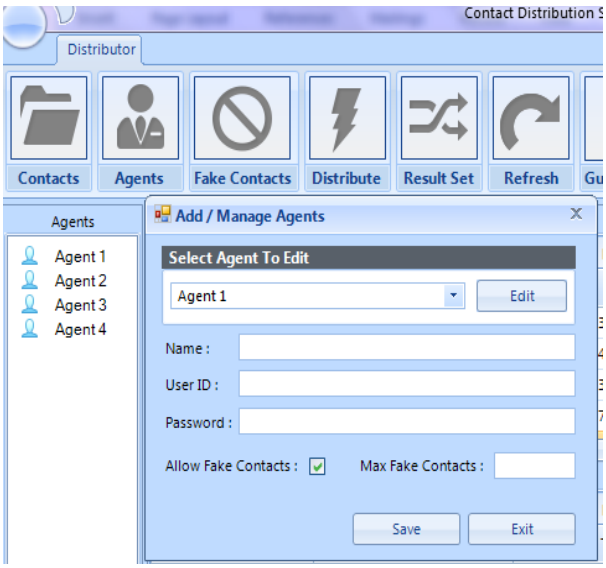

Fig4.Distributor add/manage agents[2]

As shown in fig.5 Distributor can add fake contact data into the database using this form. And this fake data is look like real data. It contain same field to that of original contact data.

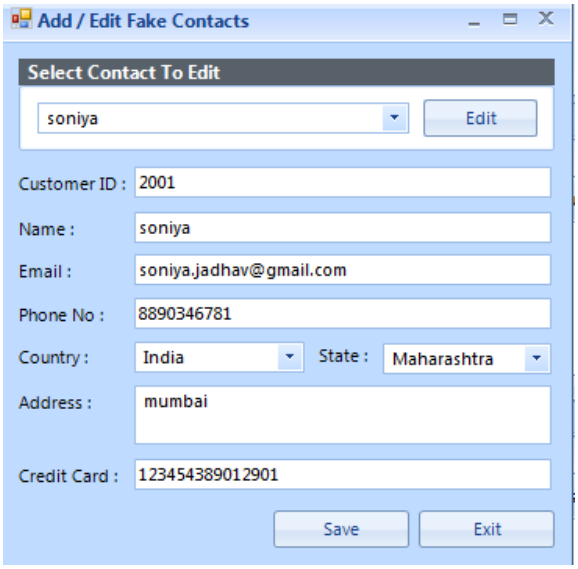

Fig.5 Add/Edit Fake contacts[2]

8.2 Agent Module:

Some of the privileges are restricted to the agents by the Distributor. Only few permissions are available with the Agents. Agents have the following task/responsibilities. Agents have Read-only access to the content.

-Agent can make request for data
-Agents can read that data

As shown in fig.6 Agent can request for data using two type of request.

8.2.1 Sample request

Here agent can enter only number of samples of data required by him. And that number of sample data should be less than upper limit.

8.2.2 Explicit request

Here Agent can specify their own condition for data. Agent can demand for specific data according to customer name, city, state etc

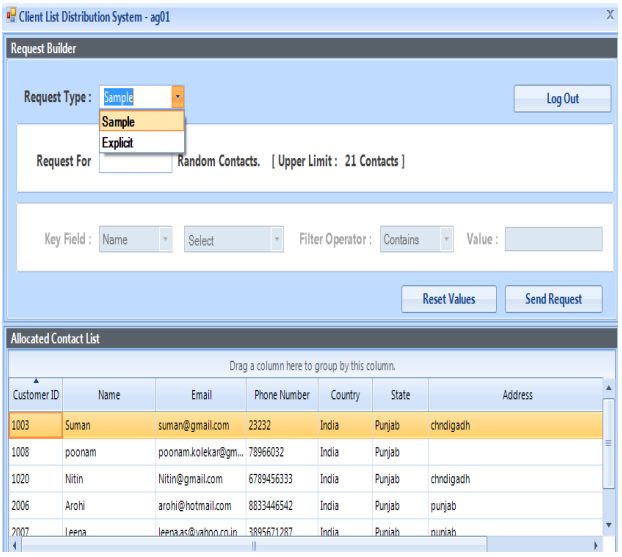

Fig.6 Agent GUI[2]

8.3 Data Leakage Detection

The main scope of this module is provide complete information about the data/content that is accessed by the users within the system.

8.3.1 Forms Authentication technique is used to provide security to the system in order to prevent the leakage of the data.

8.3.2 Continuous observation is made automatically and information is send to the Distributor so that he can identify whenever the data is leaked.

8.3.3 Above all the important aspect providing proof against the Guilty Objects. The following techniques are used.

- Fake Object Generation.
- Data Allocation strategies

As shown in fig.7 Distributor can send data using four algorithms.
Two algorithms for Sample request and two for Explicit request

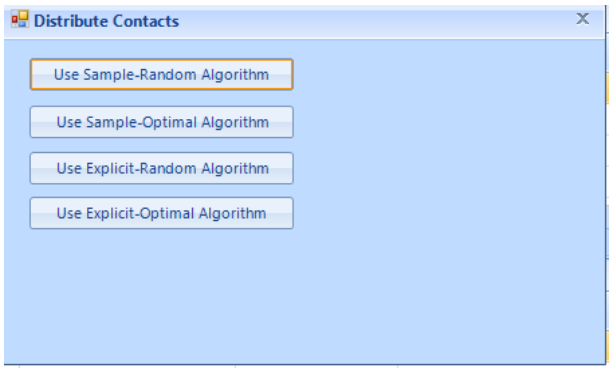

Fig.7 four allocation algorithms

As shown in fig.8 Distributor can view sent data of each agent. Also he can de-allocate data easily using this form.

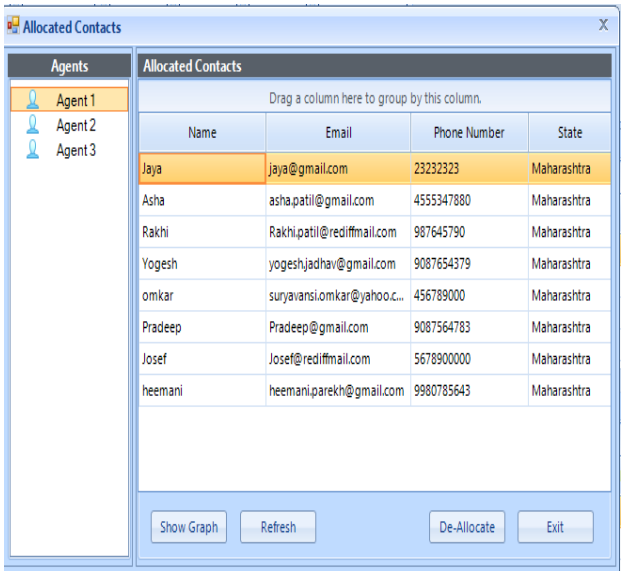

Fig.8 Allocated data of agents

As shown in fig.9 Distributor can give leaked data set as an input to this form. And on the basis of that leaked data set he can detect guilty agents and calculate guiltiness probability of that agents using this form. This form calculate guiltiness probability per object.

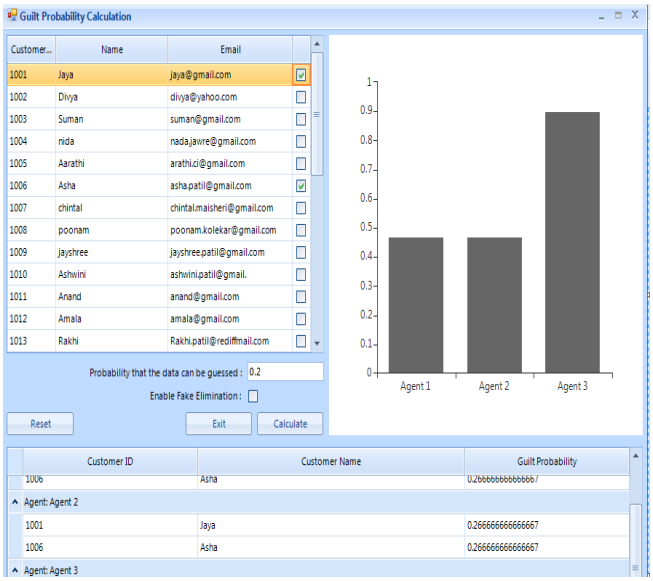

Fig.9 Calculation of guilt probability

As shown in fig.10 Distributor find particular guilty agents using proposed fake elimination technique. This form shows guiltiness probability value of guilty agent and guiltiness value per object.

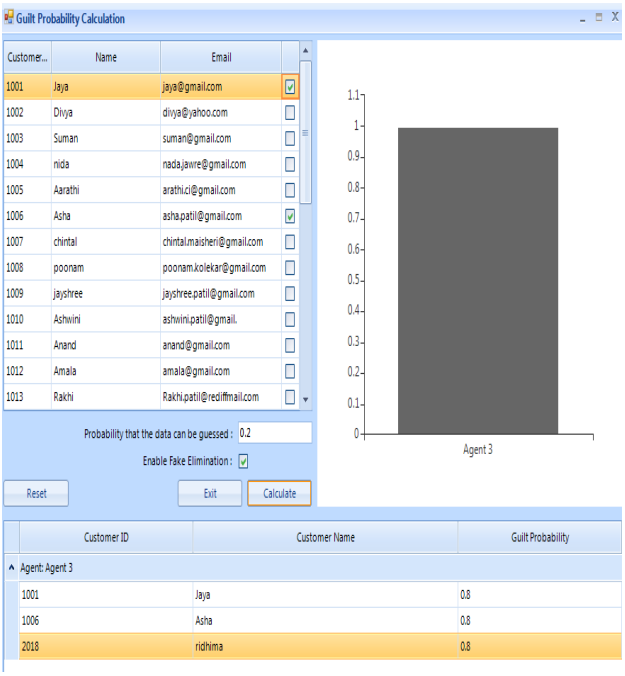

Fig.10 Fake elimination technique

9. CONCLUSION

Data leakage happens every day when confidential business information such as customer data, bank details, source code or design specifications, intellectual property and trade secrets are leaked out. When these are leaked out it leaves the company insecure state and it goes outside the jurisdiction. Because it may not be certain if a leaked object came from an agent or from some other source, since certain data cannot admit watermarks. So this uncontrollable data leakage put business in a susceptible position. In spite of these difficulties, this system 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. The presented model assesses the "guilt" of agents on basis of general method as well as fake elimination method. So in this system we can find particular one guilty agent. The main focus of this project is the data allocation problem. It specifies how the distributor can "intelligently" give data to agents in order to improve the chances of detecting a guilty agent. Finally, by adding fake objects to distributed set, the distributor can find the guilt agent easily.

10. FUTURE SCOPE

The future work is the extension of presented allocation strategies so that they can handle agent requests in an online fashion. Any application does not end with a single version. It can be improved by addition of new features. So this application is no different from this. The future enhancements that can be made to Data Leakage Detection are:

- Providing support for other type of data like file type or audio type data.
- Creation of a web based G UI for application so system can handle requests in online fashion.
- Provision of excellence or precision variance parameter for the user to set.

11. ACKNOWLEDGEMENT

I would like to express my gratitude to Prof.Ms. Priyanka Desai, Dr. Sedamkar sir and Prof. Mr. Kiran Bhandari for invaluable guidance, encouragement and critics throughout the course of my project.

12. REFERENCES

[1] Papadimitriou and H. Garcia-Molina "Data leakage detection " *IEEE Transaction on knowledge and data engineering*, pages 51-63 volume 23,January 2011.

[2] Jaymala Chavan and Priyanka Desai "Data Leakage Detection Using Data Allocation Strategies" International Journal of Advance in Engineering and Technology(IJAET), Volume 6 issue 6,Nov 2013.

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

[4] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Dataset Systems, 26(2):214-260,2001.*

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

[6] Y. Li, V. Swarup, and S. Jajodia. Fingerprinting relational databases: Schemes and specialties. *IEEE Transactions on Dependable and Secure Computing, 02(1):34–45, 2005.*

[7] 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.

[8] Shabtai, Asaf, Elovici, Yuval, Rokach, Lior 2012, VIII, 92 p. 9 illus. A Survey of Data Leakage Detection and Prevention Solutions.

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

[10] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection", *Technical report, Stanford University*, 2008.

[11] Rakesh Agrawal, Jerry Kiernan. Watermarking Relational Databases// *IBM Almaden Research Center*

[12] ORLDatabasehttp://blogs.csoonline.com/1187/DataLeakage