Implementation of Guilt Model and Allocation Strategy for Data Leakage Detection

Divya Chaube[1], Sonali Gandhi[2], Priyanka Gupta[3], Rajesh Kolte[4]

[1,2,3] Usha Mittal Institute of Technology, Mumbai, India
[4]Asst. Professor, IT department, Usha Mittal Institute of Technology, Mumbai, India

Abstract- A data distributor of an organization has hand over sensitive and confidential data to supposedly trusted third parties (agents) .The data is leaked and is discovered in an unauthorized place. The paper presents a model which includes detecting the guilty agent by the means of data allocation strategy and agent guilt model for guilt probability calculation. . Algorithms proposed for distributing data objects to agents, improves our chances of identifying a leaker. In certain scenario the idea of injecting realistic but fake data in real data is considered for further enhancing guilt probability. The major contribution in this system is to develop a guilt model using fake elimination concept.

Index Terms- Allocation strategies, data leakage, fake objects, guilty agent

I. INTRODUCTION

In enterprise, owner must hand over sensitive data to supposedly trusted agents. For example, financial data give to the financial employee for making balance sheet or for making financial transaction this may result in data leakage. Similarly, a company may have partnerships with other companies that require sharing customer data. On the other hand, limiting access to the information in the safety of preserving secrecy might damage their ability to implement the actions that can best serve the organization. While there are several advantages to ubiquitous access to data, there is also the potential for breaching the privacy of individuals.
Finding the guilty party becomes nontrivial task to the distributor. Traditionally leakage was handled by watermarking which included modification of real confidential data. The model is developed for assessing the "guilt" of agents [1][2].

II. LITERATURE SURVEY

The problem of data leakage has been an issue of major concern for enterprises since quite a long time. Various techniques for prevention of data leakage as well as detection of leaked data has been drafted and used. Traditionally leakage was handled using watermarking [3],[4], where in a unique code or a signal is securely, imperceptibly, and robustly embedded into each copy distributed to third party. Watermarking was initially used in images [5]. Technique of data provenance is also employed for the purpose of detection. Watermarking technique includes modification in original data which is not acceptable in certain cases. Recently, [6], [7], and other works have also studied marks insertion to relational data. The guild detection model proposed in this paper is similar to data provenance [8]: tracing the lineage of S objects implies essentially the detection of the guilty agents. Suggested solutions are domain specific, such as lineage tracing for data warehouses [9], and assume some prior knowledge on the way a data view is created out of data sources. The model presented here is much general and simplifies lineage tracing**.**

III. PROPOSED METHOD

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. In the proposed approach [1][2], 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. If it turns out that an agent was given one or more fake objects that were present in leaked data, then the distributor can be more confident that agent was guilty. In the Proposed System the hackers can be traced with good amount of evidence.

A. Problem Setup

A distributor [1][2][10]is the one who is responsible for distributing the data objects to all agents (agents being denoted by $U_1, U_2, \ldots.., U_n$). Let's say that the data objects owned by distributor is denoted as a set
T= $\{t_1, t_2, \ldots\ldots t_n\}$. When a distributor wants to send some of the data objects from the set T, we can say that a agent U_i can receive a set of objects $R_i \subseteq T$, determined by a sample request or explicit request

- Sample request R_i = SAMPLE (T, m_i): Any subset of mi records from T can be given to U_i.

- Explicit request R_i = EXPLICIT (T, $cond_i$): Agent U_i receives all T objects that satisfy $cond_i$.

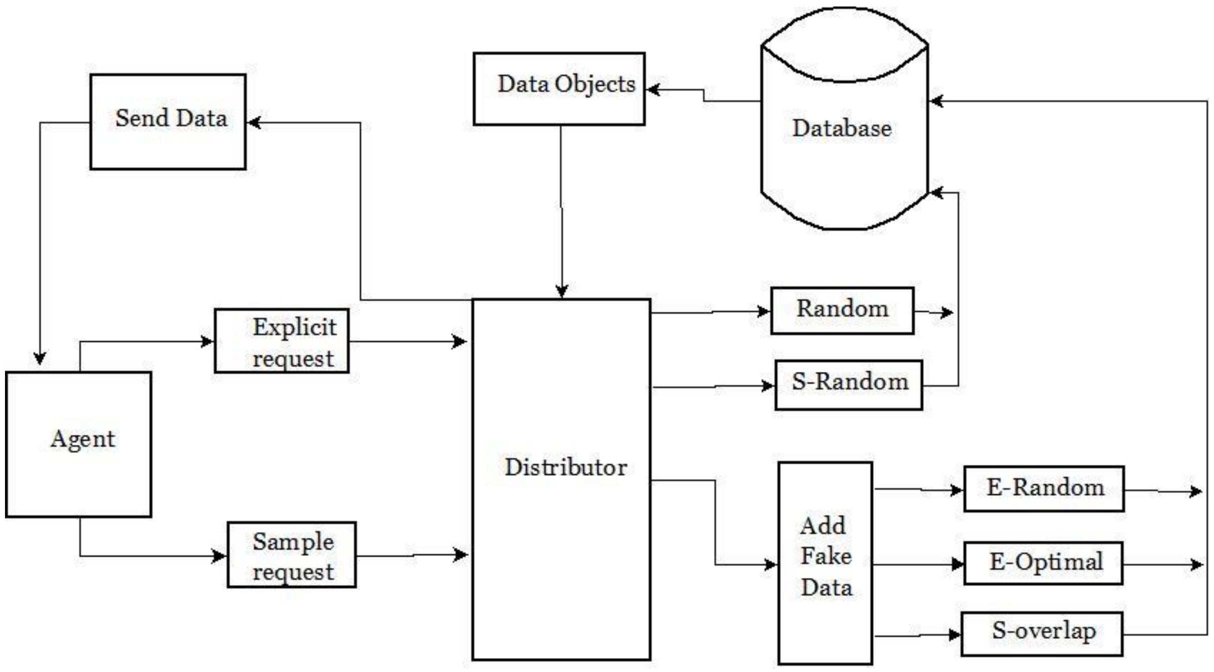


Figure 1: System architecture design

IV. AGENT GUILTY MODEL

We say that an agent is guilty when we find that data found at an unauthorized place was given to that agent. Probability [2] of an agent (*U_i*) is guilty given evidence (leaked data set 'S') is denoted by *$Pr\{G_i|S\}$*. An agent may argue that he did not leak the data since it might also be guessed or collected from various other organizations. So we need to consider the probability that data can be guessed or collected from other places, this probability we called as guessing probability, denoted by p. p depends on type of data and it is an estimate rather than exact value. P is an important criterion for fault tolerant system, value of *p* makes the system reliable. Higher the value of p we can say that an agent is guilty with at least the calculated probability. In the general case, 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 need to know which agents were given *t* , that set is *Vt*.

$$P_r\{\textit{some agent leaked } t \textit{ to } S\} = 1-p \quad (1)$$

All the agents possessing t have equal chances of leaking, thus

$$P_r\{U_i \textit{ leaks } t \textit{ to } S\} = \frac{(1-p)}{|V_t|} \quad (2)$$

Probability that an agent leaked an object will be equally divided by number of agents who were given that object. If no agent has that object, i.e. V_t=0, then *$Pr (U_i$ leaks t to S)=0.*
To compute probability that an agent *U_i* leaked is calculated by multiplying probability of each
Object leak that are in both S as well as *R_i*, i.e.

$$Pr\{G1 \mid S\} = 1 - \prod_{t \in S \cap R_i} \left(1 - \frac{(1-p)}{|V_t|}\right) \quad (3)$$

V. GUILT MODEL ANALYSIS

We see how varying the parameters in (3) impacts on guilt probability.

5.1 *Impact Of Probability p*

Consider a scenario in which an agent 1 was given all the data T and agent 2 was given only half the data. By plotting the graph against guilt probability of an agent versus p, we see that as value of p increases, so guilt probability decreases. As shown in figure 2(a)

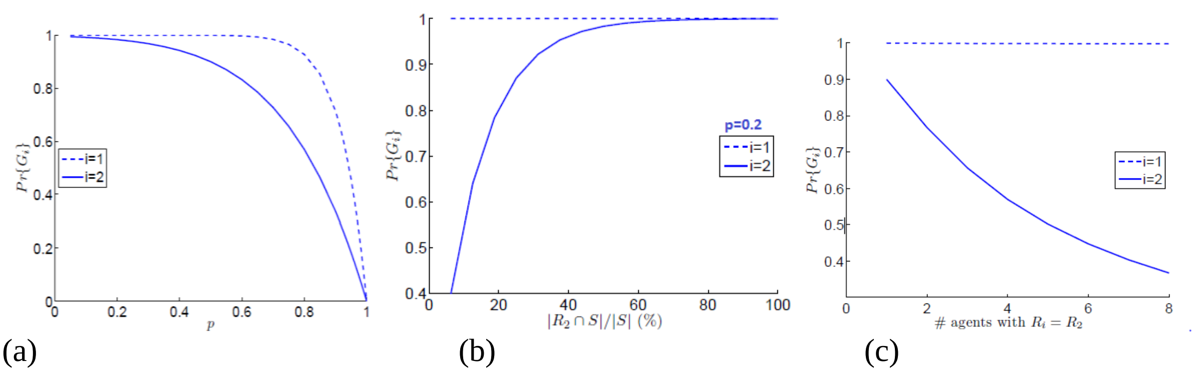


Figure 2: Dependency of Probability on various parameter

B. Impact Of Overlap Between R_i and S

As the number of objects in R_i and S increases the guilt probability increases. The agent who has more data common to leak data will appear guiltier. As more agents holding the replicated leaked data, it is harder to lay the blame on any one agent.

C. Dependence on the Number of Agents with R_i Data

We see as more agents holding the replicated leaked data S, it is harder to lay the blame on any one agent. Consider agent U1 with 16 T = S objects, and a second agent U2 with 8 objects, and p = 0.5. We calculate the guilt probabilities as we increase the number of agents that have the same 8 objects, and plot it in figure 2(c). We observe that Pr(G_1) (dotted line) remains almost unaffected by the increase in the number of agents. This insensitivity is because U_1 owns 8 of the leaked objects that do not have copies elsewhere. On the other hand, the probability of guilt of the remaining agents (solid line) drops rapidly as more agents are introduced. We see as more agents holding the replicated leaked data S, it is harder to lay the blame on any one agent.

VI. FAKE OBJECT

Fake objects are data objects which are not a part of original data objects but appear realistic to agents. Fake objects are created by distributor in accordance to certain conditions and up to a limit say B and set of fake objects is denoted by F_i. If data contains email then creation of actual email account becomes mandatory so that agent does not discover that the object is fake. Hence number of fake object created is limited. Also not many fake objects should be distributed to agent as it may raise suspicion.

VII. ALLOCATION STRATEGIES

Allocation must be intelligently done to find guilty agent. There should be minimum overlapping between two agents and maximum difference between guilt probabilities.

7.1 Explicit

Explicit data allocation is driven by agent's data request specifying a condition. In $E\bar{F}$ addition of fake object is not an option, so allocation of data depends solely upon agent's request.

Algorithm 1: Allocation for Explicit Random ($E\bar{F}$)
Input: $U_1, \ldots, U_n$, $cond_1, \ldots, cond_n$, $m_1, \ldots, m_n$
Output: $R_1, \ldots, R_n$

```
1.    for i = 1 to n do
2.    R ← ∅
3.    if $m_i > 0$ then
4.    R ← R ∪ select an object randomly
5.    end if
6.    end for
```

In EF case insertion of fake objects is allowed, hence overall allocation strategy is improved. But care must be taken that the fake objects are inserted in such a way that it maximizes difference between guilt probabilities; which can be achieved by minimizing the sum-objective.

If the distributor is able to create more fake objects, this will further improve the objective. Algorithm 2 is selects an agent who is eligible to receive fake object. For every agent the maximum allowable fake object is added and checks if unused fake objects are available in line 6. Line 7 can be performed by using Algorithm 3 (E-Random) or Algorithm 4(E-Optimal).

Algorithm 2: Allocation for Explicit Data Requests (EF)

```
Input: $R_1, . . . .,R_n, cond_1, . . . , cond_n, b_1, . . . , b_n, B$
Output: $R_1, . . . .,R_n, F_1, . . . , F_n$

1.  $R \leftarrow \emptyset$                        ➔ Agents that can receive fake objects
2.  for $i = 1, . . . , n$ do
3.  if $bi > 0$ then                   ➔ number of fake objects that can be allocated
4.  $R \leftarrow R \cup \{i\}$
5.  $F_i \leftarrow \emptyset$
6.  while $B > 0$ do
7.  $i \leftarrow$ SELECTAGENT($R,R_1, . . .,R_n$)
8.  $f \leftarrow$ CREATEFAKEOBJECT ($R_i, F_i, cond_i$)
9.  $Ri \leftarrow R_i \cup \{f\}$
10. $Fi \leftarrow Fi \cup \{f\}$
11. $bi \leftarrow bi -1$
12. if $bi = 0$ then
13. $R \leftarrow Rn / \{Ri\}$
14. $B \leftarrow B - 1$
```

This Algorithm selects an agent to allocate a fake object; this algorithm is called till there are unused fake objects available. Therefore Algorithm 2 and 3 together selects a random agent and allocate a fake object.

Algorithm 3: Agent Selection for e-random

```
1. function SELECTAGENT($R, R_1, . . ., R_n$)
2. $i \leftarrow$ select at random an agent from R
3. return $i$
```

This algorithm selects the agent to allocate the fake object. The Agent having the maximum overlap is preferred first; it makes a greedy choice by selecting the agent that will yield the greatest improvement in the sum-objective. Line 2 decides to which agent the fake object should be allocated, the agent for whom the value will be least is selected.

Algorithm 4: Agent Selection for e-optimal

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

7.2 Sample

With sample data requests, each agent Ui may receive any mi objects out T, i.e.$\binom{|T|}{m_i}$. Hence, there are $\prod_{i=1}^{n}\binom{|T|}{m_i}$different object allocations. The distributor's should select one out so that he optimizes his objective. The distributor can increase the number of possible allocations by adding fake objects.

Algorithm 5 is baseline for all Sample data request. Line 6 calls SELECTOBJECT(), this is implemented using algorithm 6,7 or 8.

Algorithm 5: Allocation for Sample Data Requests (SF)

```
Input: $m_1, . . . , m_n, |T|$            ➔ Assuming $m_i <= |T|$
Output: $R_1, . . . , R_n$

1. $a \leftarrow 0_{|T|}$                     ➔ a[k]:number of agents who have received object $t_k$
2. $R_1 \leftarrow \emptyset$ , . . . , $Rn \leftarrow \emptyset$
3. remaining $\leftarrow \sum_{i=1}^{n} m_i$
4. while $remaining > 0$ do
5. for all $i = 1, . . . , n$ : $|R_i| < m_i$ do
6. $k \leftarrow$ SELECTOBJECT ($i, R_i$)        ➔May also use additional parameters
7. $R_i \leftarrow R_i\ U \{t_k\}$
8. $a[k] \leftarrow a[k] + 1$
9. $remaining \leftarrow remaining - 1$
```

This algorithm is called in Line 6 of algorithm 5. An object allocation that satisfies requests and ignores the distributor's objective is to give each agent Ui a randomly selected subset of T of size mi. Line 2 select an object randomly that has yet not been allocated to the agent Ui.

Algorithm 6: Object Selection for s-random

```
1. function SELECTOBJECT ($i, R_i$)
```

```
2.    $k \leftarrow$ select at random an element from set $\{k' | t_{k'} \notin R_i\}$
3.    return $k$
```

Object Selection for s-overlap we provide agent Ui with an object that has been given to the smallest number of agents. So, if agents ask for fewer objects than |T|, Algorithm 7 will return in every iteration an object that no agent has received so far. Thus, every agent will receive a data set with objects that no other agent has.

Algorithm 7: Object Selection for s-overlap

```
1.    function SELECTOBJECT($i, R_i, a$)
2.    $K \leftarrow \{k \mid k = \underset{k'}{\operatorname{argmin}}\, a[k']\}$
3.    $k \leftarrow$ select at random an element from set $\{k' | k' \in K \wedge t_{k'} \notin R_i\}$
4.    return $k$
```

In this algorithm, we allocate to an agent the object that yields the minimum increase of the maximum relative overlap among any pair of agents. Any two agents should have minimum possible overlap.

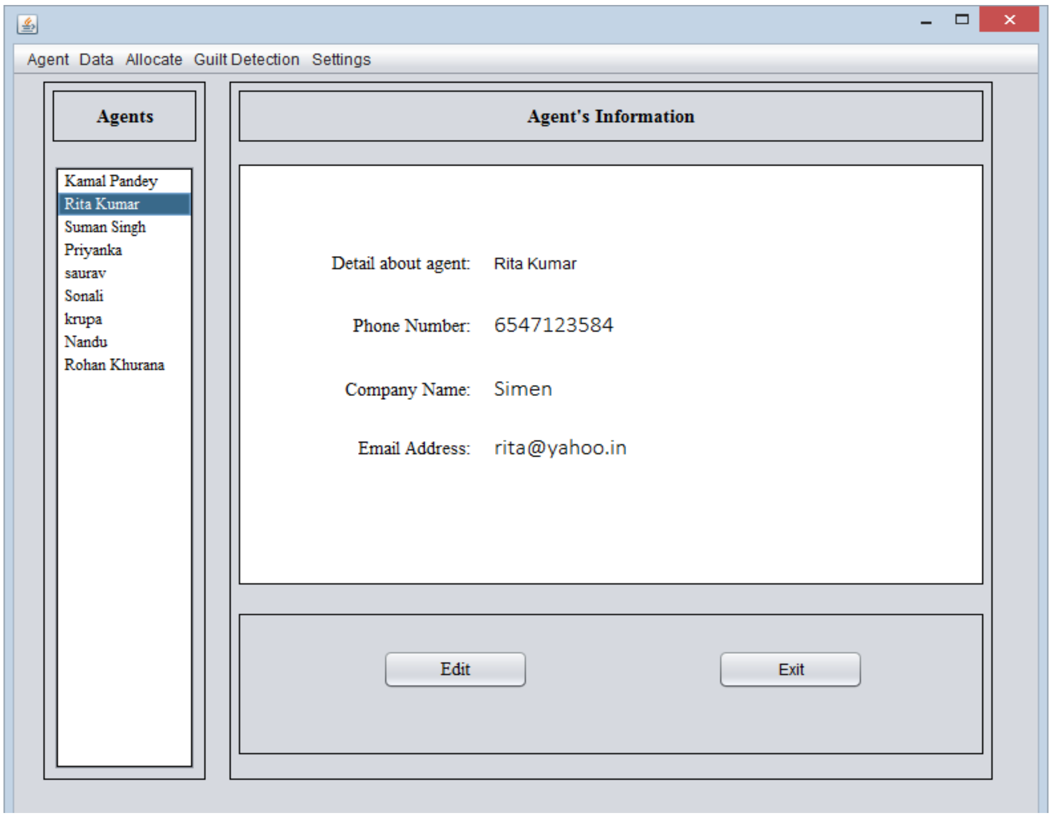
Algorithm 8: Object Selection for s-max

```
1.    function SELECTOBJECT ($i, R_1,..., R_n, m_1,..., m_n$)
2.    Initialize $min\_overlap \leftarrow 1$    ➔ the minimum out of the maximum relative overlaps that the allocations of different objects to $U_i$
3.    for $k \in \{k \mid t_k \in R_i\}$ do
4.    Initialize $max\_rel\_ov \leftarrow 0$,    ➔ the maximum relative overlap between Ri and any set Rj that the allocation of tk to Ui
5.    for all $j = 1,..., n : j = i$ and $t_k \in R_j$ do
6.    Calculate absolute overlap as $abs\_ov \leftarrow |Ri \cap Rj| + 1$
7.    Calculate relative overlap as $rel\_ov \leftarrow abs\_ov / min(m_i, m_j)$
8.    $max\_rel\_ov \leftarrow max(max\_rel\_ov, rel\_ov)$
9.    If $max\_rel\_ov \leq min\_overlap$ then
10.   $min\_overlap \leftarrow max\_rel\_ov$
11.   $ret\_k \leftarrow k$
12.   Return $ret\_k$
13.
```

VIII. EXPERIMENTAL RESULTS

8.1 Distributor Module

Distributor is the one who will have all the privilege. Figure 3 show the distributor GUI. A distributor can add new agent, edit agent's information, add fake data and send data to agents.



Kamal Pandey
Rita Kumar
Suman Singh
Priyanka
saurav
Sonali
krupa
Nandu
Rohan Khurana

Detail about agent: Rita Kumar

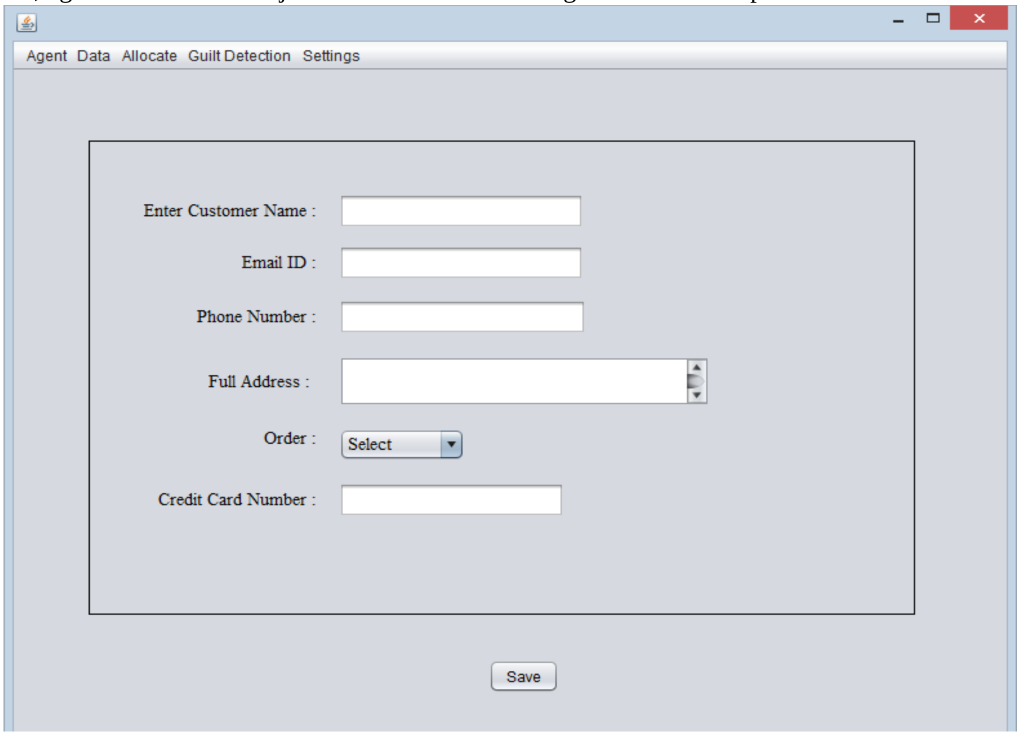
Phone Number: 6547123584

Company Name: Simen

Email Address: rita@yahoo.in

Figure 3: Distributor GUI

As shown in figure 4, agent can add fake objects which can used during allocation to improve our result.



Enter Customer Name :

Email ID :

Phone Number :

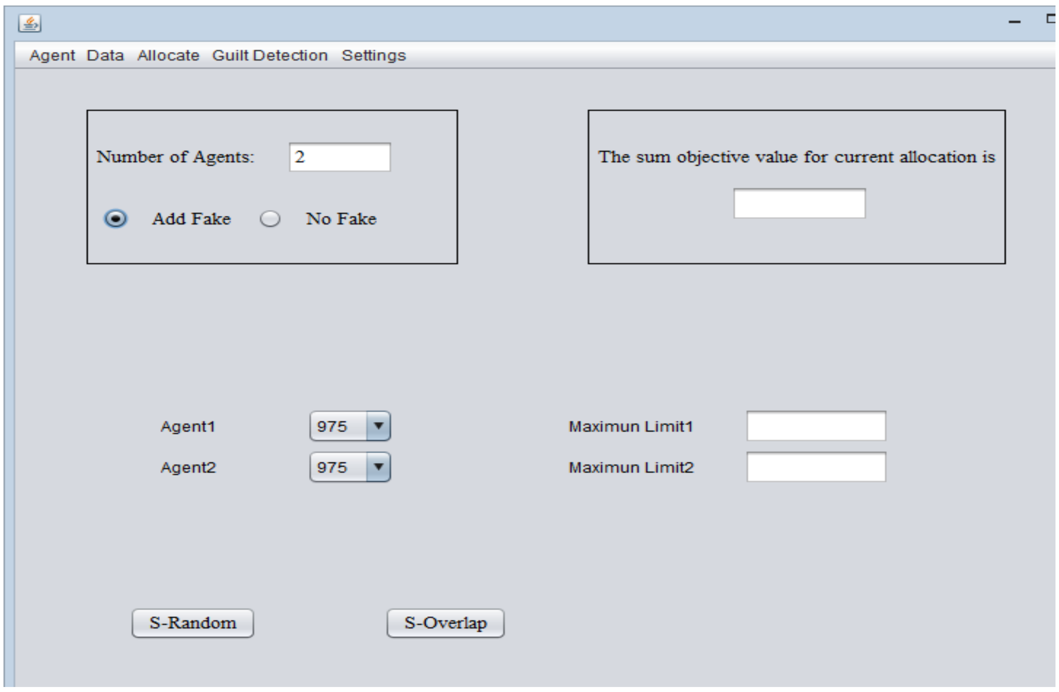
Full Address :

Order : Select

Credit Card Number :

Figure 4: Add Fake data

The modules shown in figure 5 and figure 6 are explicit and sample data allocation modules which is used by distributor to allocate data using S-random, S- Overlap, Random, E-Random and E-optimal algorithm. One of the outputs of the allocation is shown in the figure 7. The allocated data is saved in an excel sheet which is sent to intended agents.



Number of Agents: 2

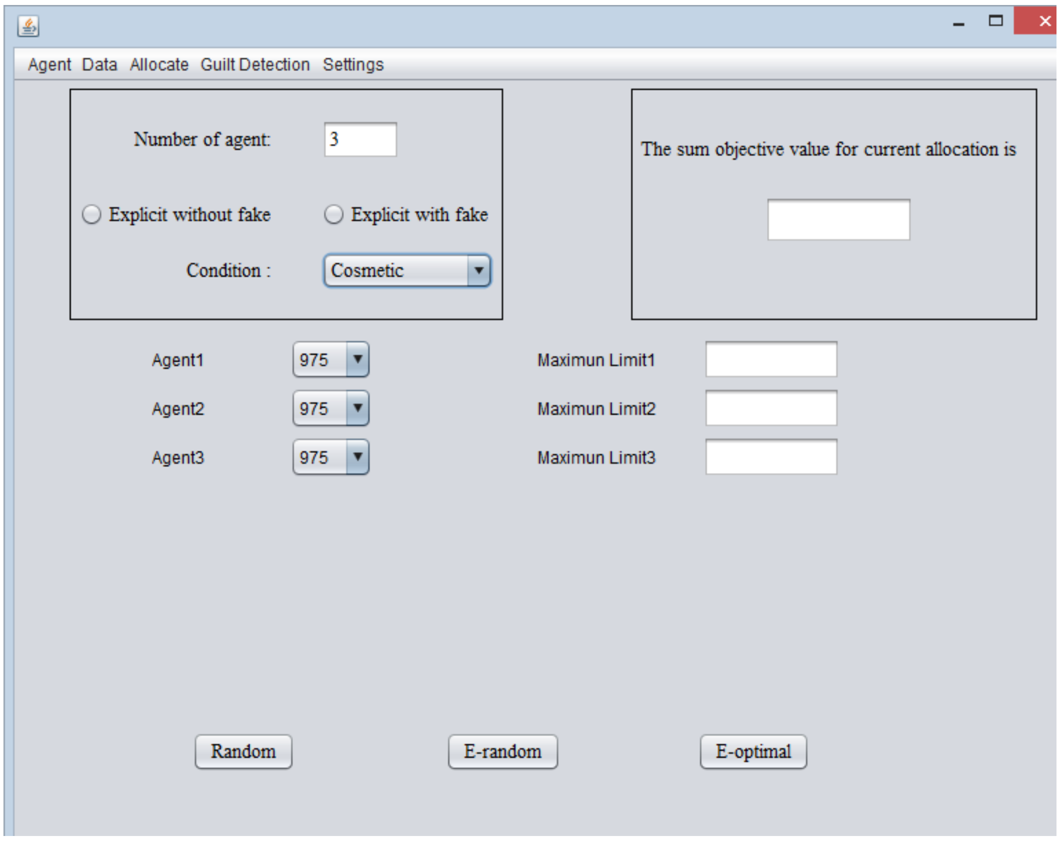
[✓ Add Fake] No Fake

The sum objective value for current allocation is

Agent1 975 Maximun Limit1

Agent2 975 Maximun Limit2

Figure 5: Sample data allocation module



Number of agent: 3

Explicit without fake Explicit with fake

Condition : Cosmetic

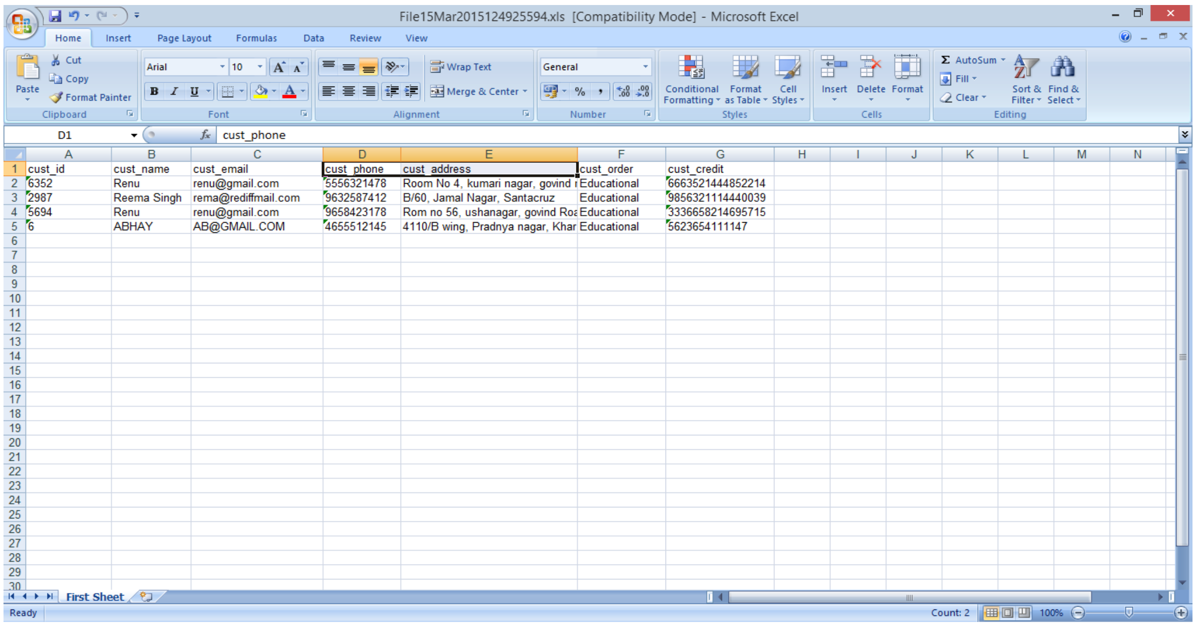
The sum objective value for current allocation is

Agent1 975 Maximun Limit1

Agent2 975 Maximun Limit2

Agent3 975 Maximun Limit3

Figure 6: Explicit data allocation module



cust_id	cust_name	cust_email	cust_phone	cust_address	cust_order	cust_credit
6352	Renu	renu@gmail.com	5556321478	Room No 4, kumari nagar, govind r	Educational	6663521444852214
2987	Reema Singh	rema@rediffmail.com	9632587412	B/60, Jamal Nagar, Santacruz	Educational	9856321114440039
5694	Renu	renu@gmail.com	9658423178	Rom no 56, ushanagar, govind Roa	Educational	3336658214695715
6	ABHAY	AB@GMAIL.COM	4655512145	4110/B wing, Pradnya nagar, Khar	Educational	5623654111147

Figure 7: Output in excel file

Figure 8 shows guiltiness probability of agents. A file have leaked set of objects is taken as input to calculate guilt probability and find the guilty agent.

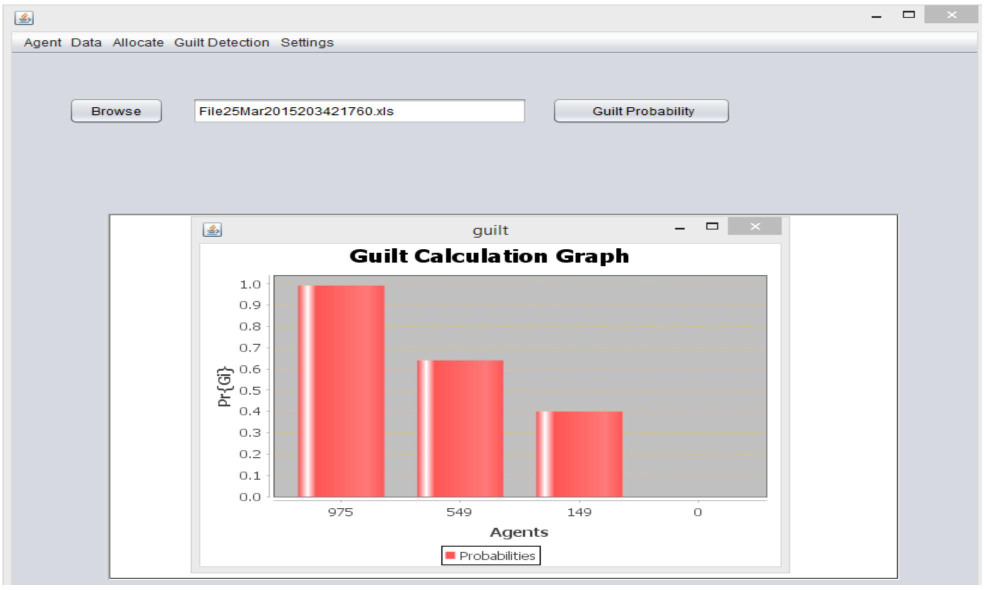


Figure 8: Calculation of Guilt Probability

IX. CONCLUSION

From this project we conclude that the data leakage detection system model is very useful as compare to the existing watermarking model. We can provide security to our data during its distribution or transmission and even we can detect if that gets leaked. Thus, using this model security as well as tracking system is developed. Data leakage is a silent type of threat. Your employee as an insider can intentionally or accidentally leak sensitive information. To assess the risk of distributing data two things are important, first is data allocation strategy that helps to distribute the tuples among customers with minimum overlap and second one is calculating guilt probability which is based on overlapping of his data set with the leaked data set.

REFERENCES

[1] Panagiotis Papadimitriou,"Data Leakage Detection", *IEEE Transactions On knowledge And Data Engineering*, Vol. 23
[2] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection", Technical report, *Stanford University*, 2008.
[3] Rakesh Agrawal, Jerry Kiernan, "Watermarking Relational Databases*", IBM Almaden Research Center.*
[4] S. Czerwinski, R. Fromm, and T. Hodes, "Digital music distribution and audio watermarking".
[5] J.J.K.O. Ruanaidh, W.J. Dowling, and F.M. Boland, "Watermarking Digital Images for Copyright Protection," *IEE Proc. Vision, Signal and Image Processing*, vol. 143, no. 4, pp. 250-256, 1996.
[6] R. Sion, M. Atallah, and S. Prabhakar, "Rights Protection for Relational Data," *Proc. ACM SIGMOD*, pp. 98-109, 2003.
[7] Y. Li, V. Swarup, and S. Jajodia, "Fingerprinting Relational Databases: Schemes and Specialties," *IEEE Trans. Dependable and Secure Computing*, vol. 2, no. 1, pp. 34-45, Jan.-Mar. 2005.
[8] 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.
[9] Y.Cui and J.Widom, "Lineage Tracing for General DataWarehouse Transformations," *The VLDB J.*, vol. 12, pp. 41-58, 2003.
[10] 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.
[11] Shabtai, Asaf, Elovici, Yuval, Rokach, Lior 2012, VIII, 92 p. 9 illus. A Survey of Data Leakage Detection and Prevention Solutions.
[12] ORLDatabasehttp://blogs.csoonline.com/1187/DataLeakage

AUTHORS

Divya Chaube, is a student of Usha Mittal Institute of Technology, Mumbai. She is currently pursuing B.Tech in Information Technology.
Email id: dvchaube@gmail.com

Sonali Gandhi, is a student of Usha Mittal Institute of Technology, Mumbai. She is currently pursuing B.Tech in Information Technology.
Email id: jyt.gandhi@gmail.com

Priyanka Gupta, is a student of Usha Mittal Institute of Technology, Mumbai. She is currently pursuing B.Tech in Information Technology.
Email id: pggupta059@gmail.com

Rajesh Kolte is working as Assistant Professor in Usha Mittal Institue of Technology, Mumbai. He holds B.E. & M.Tech. Degree in I.T.. He has a teaching experience of 15 years.
Email id: raj_kolte@rediffmail.com