Comparative evaluation of algorithms for effective data leakage detection

Conference Paper · April 2013

DOI: 10.1109/CICT.2013.6558085

Sowmya Kamath S
National Institute of Technology Karnataka
106 PUBLICATIONS **356** CITATIONS

Preprint of paper published by IEEE
In the proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT)

Comparative Evaluation of Algorithms for Effective Data Leakage Detection

Ajay Kumar
Dept of Information Technology
NITK Surathkal, India
ajaysonkar91@gmail.com

Ankit Goyal
Dept of Information Technology
NITK Surathkal, India
ankit.goyal.it@gmail.com

Ashwani Kumar
Dept of Information Technology
NITK Surathkal, India
ashwanikmr121@gmail.com

Navneet Kumar Chaudhary
Dept of Information Technology
NITK Surathkal, India
navneet604779@gmail.com

Sowmya Kamath S
Dept of Information Technology
NITK Surathkal, India
sowmyakamath@ieee.org

***Abstract* — Researchers have proposed several mechanisms to secure data from unauthorized use but there is very less work in the field of detecting and managing an authorized or trustworthy agent that has caused a data leak to some third party advertently or unknowingly. In this paper, we implement methods aimed at improving the odds of detecting such leakages when a distributer's sensitive data has been leaked by trustworthy agents and also to possibly identify the agent(s) that leaked the data. We also implement some data allocation strategies that can improve the probability of identifying leakages and can also be used to assess the likelihood of a leak at a particular agent assuming the fact that the data was not simply guessed by the third party where the leaked data set has been found. We also propose new allocation strategies that work on the basis of No-Wait model, i.e. agent does not need to wait for other agents' allocation and it is different from already proposed model that makes an agent wait for others. These methods do not rely on the alterations of the distributed data, but rather focus on minimizing the overlapping of the allocated data items to various agents, thus facilitating an exact determination of the guilty agent in a particular data leakage scenario.**

***Index Terms*— Allocation Strategies, Data privacy, Data Leakage, Detection and Prevention, Guilt model.**

1. INTRODUCTION

Data Leakage, put simply, is the unauthorized transmission of private or sensitive data or information from within an organization to a third party, i.e., an unauthorized recipient. Sensitive data in companies and organization include intellectual property (IP), financial information, personal information (like credit card data) and other information depending on the business and the industry. In the real world scenario, a distributer needs to share sensitive data among various stakeholders such as employees, business partners and customers. [1] This increases the risk that confidential information will fall into unauthorized hands, whether caused by force or by error (maliciously intended or an inadvertent mistake by an employee or a customer). Now, what is needed is a way to find the leakage points so that one can identify the leakage points (if inadvertent mistake) or guilty agents (if intended data leakage). Figure 1 depicts such a scenario.

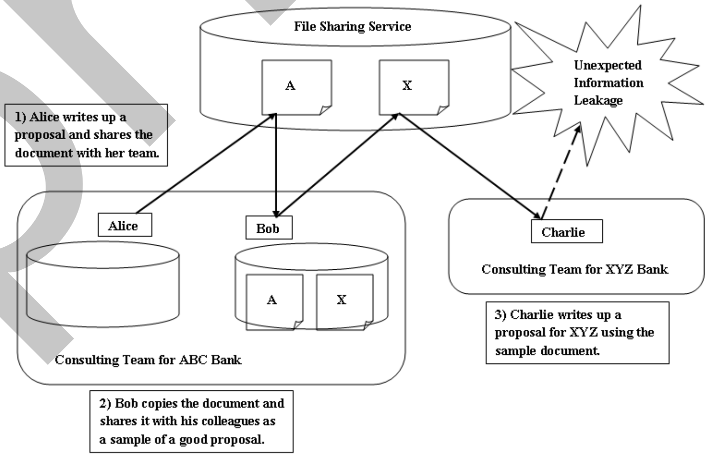

File Sharing Service
A
X
Unexpected Information Leakage
1) Alice writes up a proposal and shares the document with her team.
Alice
Bob
A
X
Consulting Team for ABC Bank
Charlie
Consulting Team for XYZ Bank
3) Charlie writes up a proposal for XYZ using the sample document.
2) Bob copies the document and shares it with his colleagues as a sample of a good proposal.

Fig 1: Possible data leakage scenarios in companies

The problem of data leakage is much more relevant and crucial nowadays as much of our information is available online through social networking sites and third party aggregators. [2] Social networking sites like LinkedIn, Facebook, Twitter etc along with their third party applications all use a part or whole of their users' personal information which, they promise to keep undisclosed and secure. Consider a situation where a user has given permission to three different apps to use his personal data, which he thinks is only a small part. However, somehow the safety of that part of the personal data was compromised, then it would be much help to develop techniques that would enable the parent site to identify the application responsible for this leakage, so as to protect the other users from identity theft, which is why data leakage detection algorithms are crucial in the present scenario.

978-1-4673-5758-6/13/$31.00 © 2013 IEEE
DOI 10.1109/CICT.2013.6558085

In this paper, we present a comparative evaluation of various data allocation algorithms that can be used to assess the likelihood or probability that a particular agent was responsible for leaking a defined dataset. The outline of the paper is as follows: In section 2, we present a study of related work in this area. In section 3, we describe the problem setup in general, the key terms and their notations etc that are used later in the algorithms. We also present the agent guilt model i.e. the probabilistic model that can be used to find the relative guilt of the agents and a formal definition of the data allocation problem and how it is to be handled along with an optimization model for the agent guilt model results. Section 4 presents an overview of the various data allocation strategies that we are experimentally evaluating. Further, section 5 presents the observed results and their analysis, followed by conclusion & future work in section 6, acknowledgement and references.

2. Related Work

Early work in the area of data leakage detection resulted in the idea of using watermarks within sensitive digital information.[4] Here, a uniquely identifying text or image is embedded within each copy that is distributed to authorized agents. When leakage occurred, then this unique code would help identify the party that was responsible for the leak. The problem with this approach was that even though this is an easy solution, it still involves a certain modification of the original data information set. Also, it was observed that such watermarks could be tampered with to sufficiently distort the uniquely identifying code or sometimes completely destroyed if the data recipient is malicious.

Papadimitriou and Garcia-Molina [5], two Stanford researchers proposed a non-obtrusive leakage detection system which can detect the guilty leaker without changing the integrity of the original data. In their paper, they proposed the main premises on which much work in this area has been based. They propose several data allocation strategies (across the agents) that improve the probability of identifying leakages. The chief contribution here was that the proposed techniques were not based on altering the distributed data in any way, but allocating the data intelligently so as to identify the guilty party.

In their work, Agarwal and Gaikwad [7] dealt with data leakage issues that arise from popular applications like email, IM and other Internet channels. E- Mail filtering was dealt on the basis of the fingerprints of message bodies, the white and black lists of email addresses and the words specific to spam. Also, in the case of data leakage from trusted agents, the distributor must evaluate the odds that the leaked records came from one or more agents. For this purpose, they used data allocation strategies or injecting "realistic but fake" data records to improve identification of leakage. Jagtap et al. **[**6] implemented a system called the Data Watcher and Leakage Detector to detect and prevent data leakage. The authors developed two models - first, if data leakage occurs when an employee of an enterprise accesses confidential data without the consent of owner, the Data Watcher model is used to identify the data leaker. Second, if data leakage occurs when an employee has given data outside the enterprise, then a second model called the Leakage Detector is used for assessing the "guilt" of the involved parties. Their Guilt model uses fake objects as a watermarking tool to improve the probability of identifying guilty third parties.

3. Problem Setup

The problem to be defined here focuses on how data is to be allocated in an efficient manner so as to maximize the chances of identifying a guilty agent that is responsible for leaking the data to the some distrusted or unknown source. Consider that agents A_1, A_2, …. A_n request the data from the distributor possessing a set of objects, say D = { d_1, d_2,…., d_m }. An agent can specify its request in two ways namely -

- *Sample Data Request* – In this type of request, an agent just needs to specify the number of data records (may be tuples from a relation) required.
- *Explicit Data Request* – This type the data request is handled on the basis of a condition that the agent can explicitly ask for special data, for example, tuples requested on the basis of specific uid's of employees in the employee relation.

Consider that the requested objects from D were allocated on the basis of requests from several agents A_1,A_2,…,$A_{n,}$ now, suppose some part of the data set S has been found leaked to a untrusted source or any unknown identity then there are two possibilities, either the leaked data objects has been guessed by the untrusted source or the data object has been leaked by some trusted agent. In order to access the likelihood of the data objects that has been leaked, we need to apply some data allocation strategies along with some probabilistic agent guilt model.

A. *Agent Guilt Model Analysis*

A model for formally defining the concept of guilt for the Data Leakage problem is discussed here. We can say that an agent A_i is guilty if it contributes one or more objects to the leaked data set. Let the event that agent A_i is guilty be G_i and the event that agent A_i is guilty for a given leaked set S be G_i|S. To compute the Pr{G_i|S}, we need an estimate of the probability that values in S can be "guessed" by the target. [3]. Pr{G_i|S} can be calculated as follows -

$$Pr\{G_i \mid S\} = 1 - \prod_{d\ \epsilon\ S \cap A} (1 - (1 - p)/ |V_d|) \tag{1}$$

where V_d is the count for no. of agents that are requesting every element that is in the leaked set S. Pr(G_j| S = A_i) or simply Pr(G_j|A_i) is the probability that agent A_j is guilty if the distributor discovers a leaked table S that contains all R_i objects.

B. *Data Allocation Problem*

Our proposed work presented here is divided into two parts: the first being implementing and verifying the allocation

978-1-4673-5758-6/13/$31.00 © 2013 IEEE
DOI 10.1109/CICT.2013.6558085

strategies for sample data requests in a round-robin fashion (proposed by Papadimitriou and Garcia-Molina [3]); and the second part is developing three new techniques for data allocation and comparing the results with the already proposed technique. Our proposed techniques have an additional advantage that an agent is completely satisfied (i.e. it receives all the objects requested) before allocating any object to the next agent. Also, in this case, the data allocation to an agent is independent of agent's requests that come afterwards. Hence, the data allocation of this kind can be possibly extended to handle requests in case when the number of agents is known in advance, as the data allocation to an agent is independent of the agents that are yet to be allocated.

C. *Optimization Problem*

After satisfying the agent request, it is crucial to identify the possibility of any sort of leakage in which some of the agents were involved for a particular leaked set S. For the same purpose, we use a notation to state formally the distributor's objective. As discussed earlier, $Pr(G_j|A_i)$ is the probability that agent A_j is guilty if the distributor discovers a leaked table S that contains all A_i objects [3]. We define the difference functions $\Delta(i , j)$ as

$$\Delta(i , j) = Pr(G_i |A_i) - Pr(G_j |A_i) \text{ where } i , j = 1,2,\ldots,n \qquad (2)$$

Here $\Delta(i , j)$ is a metric for assessing the guilt of an agent, assuming that leaked set contains all A_i objects and Agent A_i is at least as likely to be guilty as any other agent. Thus, for every such i,j pair we find $\Delta(i , j)$ values. Difference $\Delta(i , j)$ is positive for any agent A_j whose set does not contain all data of leaked set. If the agent that leaked the data is to be identified then the minimum value of $\Delta(i , j)$ should be maximized.

4. Data Allocation Strategies

In this section, we discuss the several ways of allocating data objects to the agents, such that in the case of any data leakage, the agent which has leaked the data can be traced back from the distributed dataset.

First, the allocation strategies for sample data requests using the three algorithms s-random, s-overlap and s-max based on round robin approach proposed by Papadimitriou and Garcia-Molina [3] were implemented. Their general algorithm for data allocation of data set D is shown in figure 2.

```
Input : Requests of each agent req1,req2,req3……
    1.  SUM ← sum of all requests.
    2.  while SUM > 0 do        ►for each data request
    3.      i ← SelectAgent( )
    4.          k ← SelectObject( )
    5.          Add k to the allocated set of i.
    6.      SUM ← SUM – 1.
```

Fig 2: General Algorithm for Data Allocation [3]

The running time of algorithm is O (η λ SUM) since the loop run SUM times and η denotes the running time of SelectAgent() and λ denotes the running time of SelectObject().

A. *Object Allocation*

The SelectObject( ) function selects a data object to be allocated to the agent i [3]. This can be done in three ways:

i) s-random : This allocation satisfies agents' requests but ignores the distributor's objective. It randomly selects an object from the distributor's data set and so s-random algorithm may yield a poor data allocation. For example, if the distributor set D has three objects and there are three agents who request one object each, then it is possible that s-random provides all three agents with the same object.

ii) s-overlap: In the above example, each agent can receive a distinct data object using the s-overlap selection. In this, the distributer first creates a list of data objects that has been allocated to a least number of agents and then, an object is randomly selected from this subset. This algorithm yields a disjoint set when the sum of requests from all the agents is less than the size of data set, but as *SUM* increases, it yields an object sharing distribution.

iii) s-max: 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. Figure 3 shows the s-max algorithm.

```
Object Selection for S-max
    1.  min_overlap ← 1       ►denotes min of max overlaps
            that allocation of different objects to i yields.
    2.  For each data object k € D
    3.  max_rel_overlap ← 0   ► max relative overlap
            between R_i and any set R_j that allocation of k yields.
    4.      For each agent j such that j ≠ i and k € R_j
    5.        abs_overlap ← |R_i ∧ R_j| + 1
    6.        rel_overlap ← abs_overlap / min (req_i , req_j)
                  max_rel_overlap ← max_of
                  (max_rel_overlap ,  rel_overlap)
    7.      if max_rel_overlap ≤ min_overlap
    8.            min_overlap ← max_rel_overlap
    9.            ret_k ←  k
    10.   return ret_k
```

Figure 3: s-max algorithm

In the case of s-random, $\lambda = O(1)$ and if we keep in memory the set of k objects which is allocated to least number of agents, then $\lambda = O(1)$ for s-overlap. The running time of s-max algorithm is O(|D| n) since the external loop runs for all the agents and internal loop runs for all agents.

B. *Agent Selection*

The agent selection can also be done in several ways.

978-1-4673-5758-6/13/$31.00 © 2013 IEEE
DOI 10.1109/CICT.2013.6558085

i) Round Robin: The round robin technique is proposed in [3], where the data allocation to agents is done by allocating one data object to each agent. For example, suppose there are three agents who request three objects each. Then, first one data object is allocated to first agent, then to the second agent and then the third agent. Now, again one data object is given to the first agent, and so on till all the agents are satisfied. The biggest disadvantage of this case is that, if the first agent requests only two objects and there are 100 agents, the first one has to wait till all 100 agents gets one object and after that, the second object will be allocated to first agent.

ii) First Come First Serve (FCFS): In this type, we completely satisfy an agent before allocating any data object to the next agent, i.e. in the above example, the first agent gets the three objects first and then the second agent gets the three objects and then the third agent gets the three objects. Here, the agent requesting earlier does not need to wait for other agents.

iii) Longest Request First (LRF) : In this , we first order the agents in decreasing order of their requests and then the requests are served as in FCFS, i.e. first the agent with highest request are satisfied and then the one having smaller request. In this case, the agent with highest request does not need to wait for agents having smaller requests. However, LRF has a disadvantage. For example, consider a data set D = { d1, d2, d3, d4, d5 } and three agents requesting 2, 3 and 1 objects. A data allocation for FCFS or LRF using s-overlap or s-max may yield: R_1 = { d_1, d_2 }, R_2 = { d_3, d_4, d_5 }, and R_3 = { d_1 }. Now, if d_1 is leaked, then the distributor will equally suspect agents A_1 and A_3. However, if third agent requests two objects, then s-max yields R_3 = { d_1, d_3 } and then if both d_1 and d_3 are leaked, the agent A_3 can be found guilty.

iv) Shortest Request First (SRF): As we have seen from the above example, that if the agent with smaller request is satisfied after the agents with higher requests, it may give poor allocation. Hence, we also propose a case where the agents with smaller size of requests are satisfied first completely before moving on to the ones with larger requests. For the above example, SRF using s-max yields R3 = { d1 }, R1 = { d2, d3 } and R2 = { d4, d5, d2 }.

The SRF allocation is obviously better than the LRF allocation as in the LRF allocation, R1 completely overlaps R3 but in SRF, there is no complete overlapping of an agent by any other agents (R1 and R2 have d2 in common but both contain different data objects too).

The time complexity of Round-Robin and FCFS is O(1). Also, if we keep a set of agents in memory that are sorted on the basis of their request size, then η = O(1). However, the sorting of agents will take either O(N logN) time or it can be implemented using a priority queue.

5. Results And Analysis

We have implemented the above presented algorithms in .Net framework using C# and have conducted several experiments with different type of data requests and presented the results.

A. Metrics Used:

We will measure not only the algorithm performance, but also will implicitly evaluate how effective the approximation is. The Δ difference functions are used to evaluate a given allocation with objective scalarization in (3) and (4) as metrics.

$$\overline{\Delta} = \frac{\sum_{i,j=1,,,,,n \atop i \neq j} \Delta(i,j)}{n(n-1)} \tag{3}$$

$$\min \Delta = \min_{i,j=1,,,,,n \atop i \neq j} \Delta(i,j) \tag{4}$$

Metric Δ is the average of Δ (i, j) values for a given allocation and it shows how successful the guilt detection is, on average, for this allocation. Metric min Δ is the minimum Δ (i, j) value and it corresponds to the case where agent i has leaked his data and both i and another agent j have very similar guilt probabilities. If min Δ is small, then we will be unable to identify i as the leaker, versus j.

In case of sample data requests, agents are not requesting for particular objects. Hence, object allotment is not overtly defined by the requests. The distributor allocates certain objects to multiple agents only if the sum of requests SUM exceeds the number of objects in set D. The parameter that primarily defines the difficulty of a problem with sample data requests is the ratio of SUM to the size of data set, which is denoted as load. Note also that the absolute values of requests and size of data set play a less important role than the relative values req_i / |D|.

In our experimental setup, we take a data set of 50 objects fixed and vary the load by increasing the number of agents. We calculate the metrics defined for two types of requests: a) the requests of all agents are same (say 5); and b) when the agents requests are different and provided in a random fashion (unordered). In this, the data allocation to agents is done by allocating one data object to each agent, i.e. suppose there are three agents who requests three objects each. Then, first one data object is allocated to first agent, then to second agent and then the third agent. Now, again one data object is given to first agent and goes on till all the agents are satisfied.

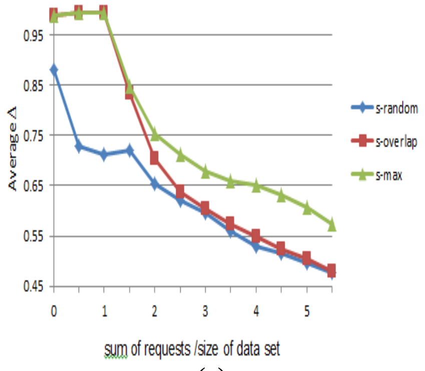
B. Verification

We have first implemented the algorithms [3] in a round robin fashion. We calculate the results for both type of requests as mentioned earlier and have taken the average values and plotted the graphs. and size of data set play a less important role than the relative values req_i / |D|.

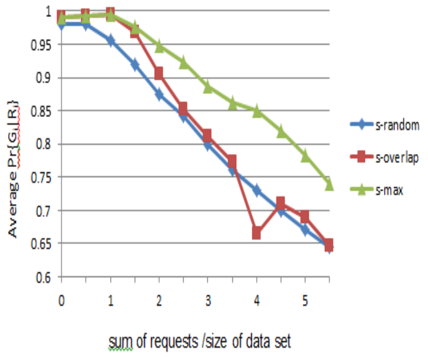
In our experimental setup, we take a data set of 50 objects fixed and vary the load by increasing the number of agents. We calculate the metrics defined for two types of requests –

978-1-4673-5758-6/13/$31.00 © 2013 IEEE
DOI 10.1109/CICT.2013.6558085

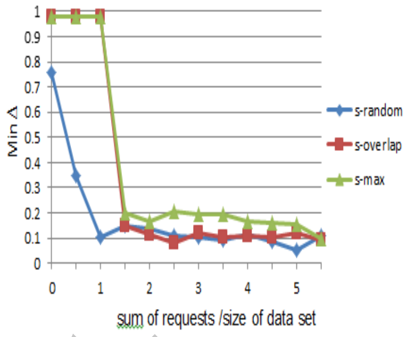
Preprint of paper published by IEEE
In the proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT)



(a)

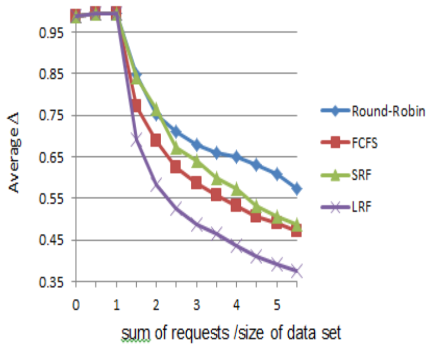


(b)

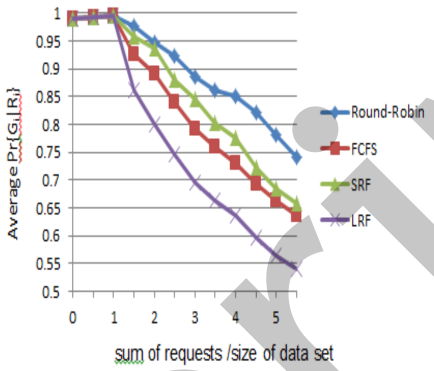


(c)

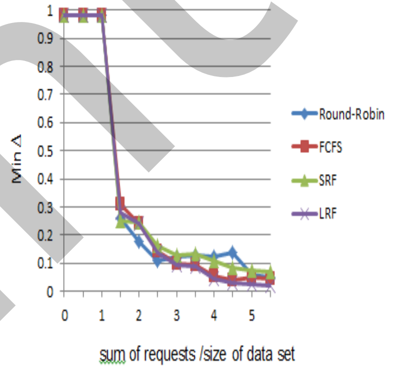
Fig. 4. Evaluation of Sample Data Requests with Round-Robin technique (a) Average Δ (b) Average Pr{G_i|S_i} (c) Average minΔ.



(a)



(b)



(c)

Fig. 5. Comparison of Round-Robin, FCFS, SRF and LRF techniques (a) Average Δ (b) Average Pr{G_i|S_i} (c) Average minΔ.

(a) Requests of all agents are the same. (Say 5 data objects).
(b) The agents have overlapping requests.

Figure 4 shows the evaluation of Sample Data Requests with Round-Robin technique for the case of average Δ, average Pr{G_i|S_i} and average min Δ.

In figure 4(a), average Δ are plotted against the load for all the three algorithms. It may be noted that algorithms s-max and s-overlap yields value close to 1 if sum of agents' requests is less than |D|. This was expected since in such scenarios, each of the three algorithms will result in an allocation that is disjoint, which is actually the optimal solution. For the higher load values, all algorithms yield value close 0.5, which we believe is an acceptable value.

Figure 4(b) shows the average guilt probability in each case for actual guilty agent. As we can see the s-max algorithm outperforms both s-random and s-overlap algorithms for load value greater than 1. However, s-overlap and s-max yields same value for load below 1.

In Figure 4(c), we compare the three algorithms with respect to min Δ metric. Random allocation shows poor performance, as the probability that at least two agents receive many common objects keeps rising as the number of agents rise. Algorithm s-overlap limits the random allocation selection to allocations that display the minimum absolute overlap. The result is that the min Δ values reduce further, since the smaller absolute overlap reduces object sharing. Therefore, the chances that any two agents receive sets with many common objects also reduce. Algorithm s-max, which uses a greedy strategy to allocate objects optimally, shows the best performance among these algorithms.

C. Comparison of Round-Robin, FCFS, LRF and SRF techniques

We compared the results obtained by each of these techniques with s-overlap and s-max algorithms. As mentioned above, the results are calculated for two scenarios, i.e. one in which each agents' request are same and the other which requests are different. For the first scenario, SRF, LRF and FCFS perform in the same way as when there is no ordering. The results are calculated for both the scenarios several times and the average of all the values are taken to plot the graphs.

All the four techniques behave in a very similar way for s-overlap algorithm. But the values are slightly better for Round-Robin than the other three for the case when the agents' requests are different and unordered.

Figure 5(a) shows the average Δ values as plotted for all the four techniques using s-max algorithm. It can be observed that values for the four techniques are same for load value less than or equal to 1, but as the load increases Round-Robin outperforms all the three techniques but we can see that the SRF and FCFS

978-1-4673-5758-6/13/$31.00 © 2013 IEEE
DOI 10.1109/CICT.2013.6558085

values are somewhat nearer to those of Round-Robin. The LRF technique yields low average Δ values for higher load. This is due to that agents with much smaller requests receive the objects which overlaps with the objects allocated to previous agents.

Figure 5(b) shows the average guilt probability for the actual guilty agent, which is much lower for LRF as compared to other three techniques for higher load values. Here SRF values are somewhat near to the Round-Robin values.

Figure 5(c) shows the minΔ values and we can see that the values are almost same for all the techniques. The minimum difference between the guilt probabilities of any two agents is independent of the technique used for agent selection, but does depend on the algorithm used for object selection

6. Conclusion And Future Work

In this work, we analyzed the likelihood that an agent may be responsible for any data leakage using several techniques. The algorithms were implemented using the four techniques, and in a real world, the distributor can use one of these depending on its needs. We observed that distributing data prudently may improve the chances of detecting the agents effectively especially when there is a large overlap in the data that agents must receive.

Our first objective was to verify the results of algorithms that have already been proposed [3]. We observed that for most cases the s-max algorithm performs better than the s-overlap and s-random algorithm. Also, we implemented these techniques and observed that in case of s-overlap, all the techniques show the same performance and any of these can be used based on the distributor's application requirements. However, in case of s-max algorithm, we observed that Round-Robin and SRF performs somewhat better than the other two. Hence, we can conclude that if the distributor wants to completely satisfy an agent before allocating any object to other agents, SRF technique must be used in order to improve the chances of identifying the leaker.

Our future work includes the implementation of data allocation strategies for explicit data requests. We will also extend our work to handle agents' requests in an online fashion, i.e. when the number of agents and agent requests are not known in advance. The FCFS technique presented above can be extended to handle agents' requests in an online fashion as we know that the data allocation to an agent, in this case, does not depend on the agents that present their request after the first. Hence, the agents requesting after the allocation is already done can be satisfied using FCFS technique.

References

[1] "Data Leakage: The High Cost of Insider Threats." Worldwide White Paper.,http://www.cisco.com/en/US/solutions/collateral/ns170/ns896/ns895/white_paper_c11-506224.html., accessed January 2013

[2] "Facebook Data Leak: Should You Worry?" By Jeanine Skowronski Available:http://www.mainstreet.com/article/moneyinvesting/news/facebook-data-leak-should-you-worry, accessed January 2013

[3] Panagiotis Papadimitriou, *Member, Ieee, Hector Garcia-Molina, Member, IEEE.*, Data Leakage Detection, IEEE Transactions On Knowledge And Data Engineering, Vol. 23, No. 1, January 2011

[4] R. Agrawal and J. Kiernan, "Watermarking Relational Databases," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02),

[5] P. Papadimitriou and H. Garcia-Molina, "Data Leakage Detection," technical report, Stanford Univ., 2008.

[6] N. P. Jagtap, S. J. Patil, A. K. Bhavsar, "Implementation of data watcher in data leakage detection system", International Journal of Computer & Technology Volume 3,No. 1, Aug, 2012.

[7] Ankit Agarwal, Mayur Gaikwad, Kapil Garg, Vahid Inamdar, "Robust Data leakage and Email Filtering System", International Conference on Computing, Electronics and Electrical Technologies, 2012.

[8] Y. Cui and J. Widom (2001) 'Lineage Tracing For General Data Warehouse Transformations' -In the VLDB Journal,pp. 471– 480.

[9] Preeti Patil, Nitin Chavan, Srikantha Rao, S B Patil, "Building of a Secure Data Warehouse by Enhancing the ETL Processes for Data Leakage", Intl Conf & Workshop on Recent Trends in Technology,2012

[10] Dangwal et al. International Journal of Advanced Research in Computer Engineering & Technology, Volume 1, Issue 4, June 2012

[11] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. "Flexible support for multiple access control policies".

[12] P. Buneman, S. Khanna, and W. C. Tan. "Why and where: A characterization of data provenance."

Table 1: Comparison of Existing and the Different Proposed Techniques

Features / **Techniques**	**Round-Robin**	**FCFS**	**SRF**	**LRF**
Waiting	Agent needs to wait for other agents.	Agents receive objects as soon as they request.	All wait till agents with smaller requests are satisfied.	All wait till agents with larger requests are satisfied.
Handling requests in Online Fashion	Difficult to implement	Easier	Difficult	Difficult
Probability of finding the Guilty Agent	Highest	Comparatively High	Comparatively High	Lowest

978-1-4673-5758-6/13/$31.00 © 2013 IEEE
DOI 10.1109/CICT.2013.6558085