Abstract

Data is an essential property of almost all organizations, and it is the intellectual property of organizations. Every organization has sensitive data such as customer, financial, patient, personal credit card, and other information based on management, institute, or industry. For the areas like this, information leakage is a significant problem that the organization must face, which poses a high cost if information leakage is done. All the more definitely, information leakage is classified as the intentional exposure of individuals or any information to unapproved outsiders. When the critical information goes to unapproved hands or moves towards an unauthorized destination, this will prompt the direct and indirect loss of a particular industry in terms of cost and time. Information leakage is the outcome of vulnerability or its alteration. So information can be protected by stranger leakages. There must be an efficient and effective system to avoid and protect authorized information to solve this issue. Researchers have proposed several mechanisms to secure data from unauthorized that are analyzed here in this report.

Keywords: Allocation Strategies, Data privacy, Data Leakage, Detection and Prevention, Guilt model

Contents

List of Figures

List of Tables

List of Acronyms

MAC - Media Access Control

AES - Advanced Encryption standard

FCFS - First Come First Serve

LRF - Longest Request First

SRF - Shortest Request First

List of Symbols

$\in$	Belongs to
$\prod$	Product over the series
$\cap$	Intersection of Sets
Δ	Delta
$\sum$	Summation

Chapter 1

Introduction

Data leakage is nothing but getting access to essential data of a person or an organization by unauthorized users. Important data of organization can have information customer, business plans, financial condition, information of patients, credit-card data of employs, Payment details and so on depending on the business or a person or industry. it's 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).

1.1 Application

Many websites publish information and provide access to information on the internet. All those websites have many different web application programs containing potential information or important data that must be protected. In many organizations, business situations, and companies outsource their data to other companies or organizations or agents, all these agents got recognized as trusted third-party agents. The data distributor gives the confidential data to the trusted third-party agents, and there are chances that any authorized agents can leak the potential data. Data leakage is the unintentional or unexpected distribution of private or sensitive data to an unauthorized party. Detection of leaked data and the guilty agents is a significant challenge in many industries. If any agent leaks confidential data, it leads to tremendous financial loss. So, there is a need to provide protection and privacy to the data and identify the faulty agents, which are from the trusted agents, who leaked the data to the untrusted persons.

1.2 Motivation

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 permitted 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. It would help develop techniques that would enable the parent site to identify the application responsible for this Data leakage to protect the other users from identity theft, which is why data leakage detection algorithms are crucial in the present scenario.

1.3 Overview

This seminar report aims to summarize the works that were done till today on behalf of Data leakage detection, mainly focusing on how it works and how it will get implemented. In this paper, I 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 paper's outline is as follows: In section 2, I present a study of related work in this area. Section 3 describes the problem Definition in general and the key terms and their notations, etc., that are used later in the algorithms. I 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 Algorithms analysis, followed by conclusion in section 6, acknowledgement and references

Chapter 2

Literature Survey

Early work in data leakage detection resulted in the idea of using watermarks within sensitive digital information.[4] Here, a uniquely distinguishing text or image is embedded with each copy distributed to authorized agents. When data leakage occurs, this unique code will help identify the party responsible for the leak. The problem with this method was that even though this is a simple solution, it still involves modifying the initial data information set. Also, it was observed that such Digital watermarks could be tampered with to sufficiently change the uniquely identifying code or sometimes completely destroyed if the data recipient is malicious.

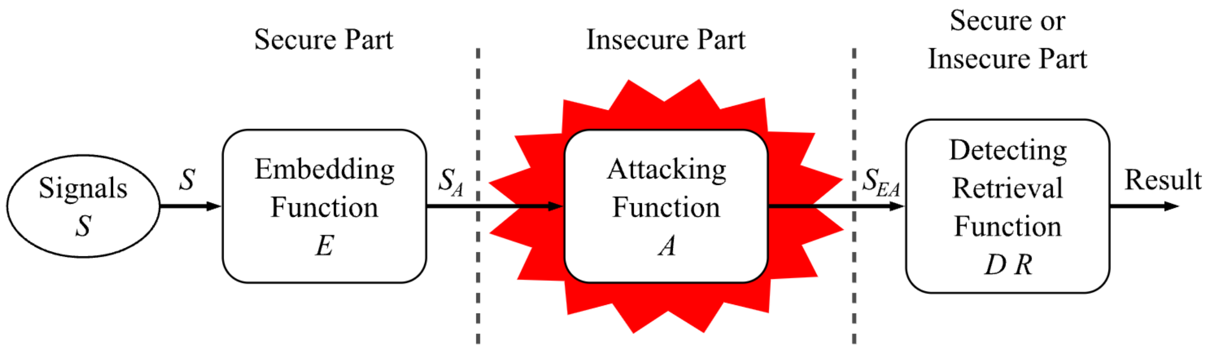


Figure 1: Digital Watermark life cycle [10]

Panagiotis Papadimitriou and Hector Garcia-Molina designed "DATA LEAKAGE DETECTION" [3] In their model, it is explained that there is a chance to judge the likelihood that a third party agent is responsible for a leak, based on the overlapping of his data with the leaked data and the data from the other agents, and based on the probability that objects can be "guessed" by other techniques.

The presented algorithms implement various data distribution techniques that can improve the distributer‟s possibilities of detecting the leaker. It is also observed that distributing objects thoughtfully can make an essential difference in detecting guilty agents, especially in scenarios where there is a considerable similarity in the information agents must get.

Maulik Bhagat and Prof. G.B. Jethava proposed “Detection of the guilty agent using an encryption algorithm and MAC(Media Access Control) address,” in which desired objective of the distributor is to recognize the guilty agent who leaked the data. This system identifies the guilty agent using the MAC address and AES Encryption algorithm without adding fake objects or calculating guilt probability. AES algorithm encrypts the potential content, so an unauthorized agent is unable to use the confidential information. MAC address is used to identify the guilty agent, so this system identifies the guilty agent by providing the confidentiality of data.

Findings of the author [5], the introduction of essential data are not basic because of information change in the content. Transformations (for example, insertion and deletion) results in significantly unpredictable leakage patterns. Present automata-based string coordinating algorithms are illogical for finding transformed data leakage due to its formidable complexity nature while exhibiting the required consistent expressions. They create two novel algorithms for recognizing long and also wrong data leakage. Their framework achieves high detection precision in perceiving changed breaks contrasted and the best in class inspection techniques. They parallelized our design on the graphics processing unit and demonstrated the solid scalability of data leakage detection arrangement examining enormous information.

Chapter 3

Data leakage Detection Technique

3.1 Problem Definition

The problem to be defined here focuses on how data is allocated efficiently to maximize the possibilities of recognizing a guilty agent responsible for leaking the data to some distrusted or unknown source. Consider that agents A1, A2, …. An request the data from the distributor possessing a set of objects, say D = { d1, d2,…., dm }. An agent can designate 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 (maybe tuples from a relation) required.
• **Explicit Data Request** – This type of data request is handled based on a condition that the agent can explicitly ask for particular data, such as tuples requested based on specific uid's of employees in the employee relation.

Consider that the requested objects from D were allocated based on requests from several agents A1, A2,…, An, now, suppose some part of the data set S has been found leaked to an untrusted source or any unknown identity, then there are two probabilities, either the leaked data objects have been guessed by the untrusted source or the data object has been leaked by some trusted agent. In order to reach the likelihood of the data objects that have been leaked, we need to apply some data allocation strategies along with some probabilistic agent guilt models.

3.2 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 Ai is guilty if it contributes one or more objects to the leaked data set. Let the event that agent Ai is guilty be G_i and the event that agent A_i is guilty of 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 -

$$Pr\{G_i \mid S\} = 1 - \Pi_d \in S \cap A\,(1 - (1 - p) / \mid V_d \mid) \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Where V_d is the count for the number 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.

3.3 Data Allocation Problem

The proposed work presented here is divided into two parts: the first being implementing and verifying the allocation strategies for sample data requests in a round-robin fashion (proposed by Papadimitriou and Garcia-Molina [1]); and the second part is developing three new techniques for data allocation and comparing the results with the already proposed technique.

The proposed techniques have an additional advantage: an agent is delighted (i.e., receiving 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 the agent's requests that come afterwards.

Hence, the data allocation of this kind can be possibly extended to handle requests in the 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.

3.4 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 containing all A_i objects [1]. We define the different functions Δ(i, j) as

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

Here Δ(i, j) is a metric for assessing the guilt of an agent, assuming that the leaked set contains all Ai 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 Δ(i, j) values. Difference Δ(i, j) is positive for any agent A_j whose set does not contain all data of the leaked set. If the agent that leaked the data is to be identified, then the minimum value of Δ(i, j) should be maximized.

Chapter 4

Data Allocation Strategies

In this section, I discuss 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 the round-robin approach proposed by Papadimitriou and Garcia Molina [1] 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.
```

Figure 2: General Algorithm for Data Allocation [1]

The running time of the 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().

4.1 Object Allocation

The SelectObject( ) function selects a data object to be allocated to the agent i [1]. Object Selection can be made 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 an s-random The algorithm may yield a poor data allocation. For example, if the distributor set D has three objects and three agents request one object each, then s-random may provide all three agents with the same object.

ii) s-overlap: Each agent can receive a distinct data object using the s-overlap selection in the above example. In this Algorithm, the distributor first creates a list of data objects that have been allocated to the 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 the 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 Ri and any set Rj that
allocation of k yields.
  4.     For each agent j such that j ≠ i and k € $R_j$
  5.         abs_overlap ← $|R_i \wedge R_j| + 1$
  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 [1]

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

4.2 Agent Selection

The agent selection can also be made in several ways.

i) Round Robin: The round-robin technique is proposed in [1], where the data allocation to agents is done by allocating one data object to each agent. For example, suppose three agents request three objects each. Then, the first one data object is allocated to the first agent, then to the second agent, and then the third agent. Again, one data object is given to the first agent and so on until all the agents are satisfied. The most significant 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 get one object, and after that, the second object will be allocated to the first agent.

ii) First Come First Serve (FCFS): In this type, we completely satisfy an agent before allocating any data object to the following 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 the highest requests are satisfied, and then the one having smaller request. In this case, the agent with the 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: R1 = { d1, d2 }, R2 = { d3, d4, d5 }, andR3 = { d1 }. Now, if d1 is leaked, then the distributor will equally suspect agents A1 and A3. However, if the third agent requests two objects, then s-max yields R3 = { d1, d3 } and then if both d1 and d3 are leaked, agent A3 can be found guilty.

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

The SRF allocation is 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 based on their request size, then $\eta = O(1)$. However, the sorting of agents will take either O(N logN) time, and it can be implemented using a priority queue.

Chapter 5

Algorithms Analysis

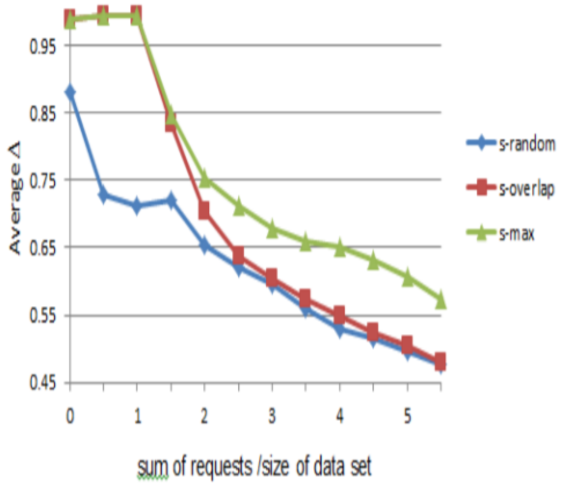
We will measure not only the algorithm performance but also will implicitly evaluate how effective the approximation is. Δ is the difference functions are used to evaluate a given allocation with objective scalarization in (3) and (4) as metrics.

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

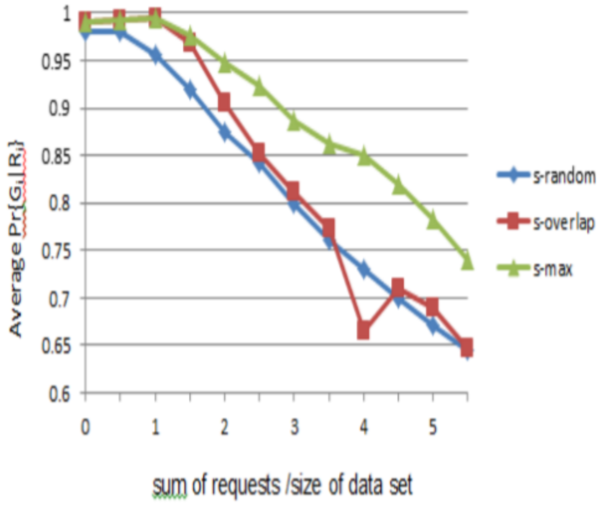
$$min\,\Delta = min_{i \neq j}\ {}^{i,j=1,,,n\Delta(i,j)} \ldots\ldots\ldots\ldots\ldots (4)$$

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.

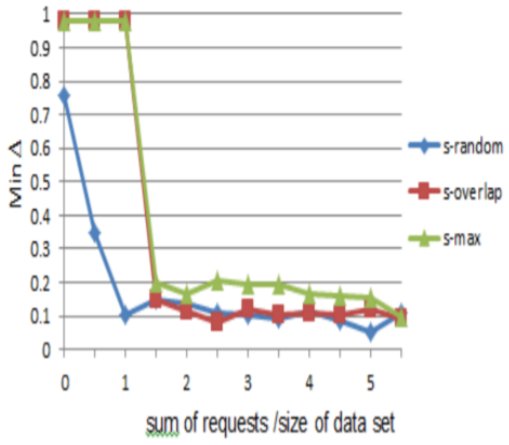
In the case of sample data requests, agents are not requesting for particular objects. Hence, object allotment is not overtly defined by the requests. The distributor allocates certain objects to multiple agents only if the sum of requests SUM exceeds the number of objects in set D. The parameter that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



(a)

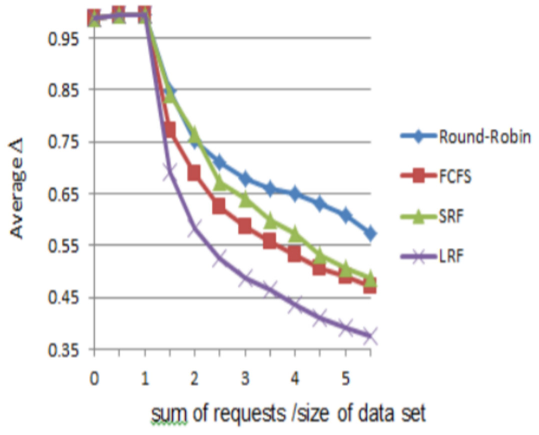


(b)

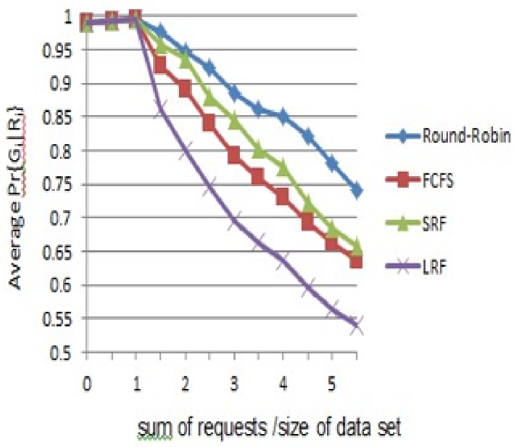


(c)

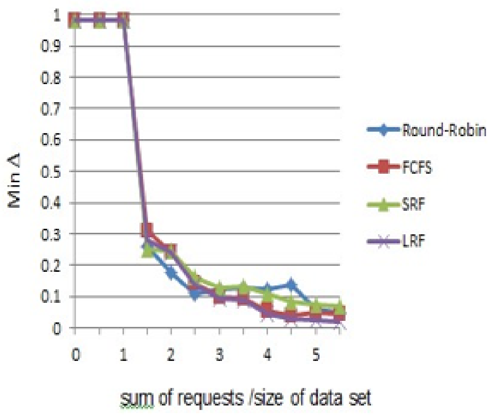
Figure 4. Evaluation of Sample Data Requests with Round-Robin technique (a) Average Δ (b) Average Pr{Gi|Si} (c) Average minΔ [1]



(a)



(b)



(c)

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

In our experimental setup, we take a dataset 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 the same (say 5)

b) when the agent's 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 three agents request three objects each. Then, the first one data object is allocated to the first agent, then to the second agent, and then the third agent. Again, one data object is given to the first agent and goes on until all the agents are satisfied.

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., each agent's request is the 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 order. The results are calculated for both the scenarios several times, and the average of all the values is taken to plot the graphs.

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

TABLE 1: COMPARISON OF EXISTING AND THE DIFFERENT PROPOSED TECHNIQUES

Features/Techniques	Round-Robin	FCFS	SRF	LRF
Handling requests	Difficult to implement	Easier	Difficult	Difficult
Probability of finding the Guilty Agent	Highest	Comparatively High	Comparatively High	Lowest

Chapter 6

Conclusion

This work 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 the 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 significant 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 the s-max algorithm performs better than the s-overlap and the s-random algorithm for most cases. Also, they implemented these techniques and observed that in the 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 the case of the s-max algorithm, we observed that Round-Robin and SRF perform 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, the SRF technique must be used to improve the chances of identifying the leaker.

References

[1] 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

[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, Hector Garcia-Molina, Data Leakage Detection IEEE Transaction 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] X. Shu, et al., “Rapid and parallel content screening for detecting transformed data exposure,” in Proc. 3rd Int. Workshop Secur. Privacy Big Data (BigSecurity), pp. 191-196, 2015.

[6] B. Sruthi Patil, Mrs M. L. Prasanthi, Modern Approaches for Detecting Data Leakage Problem, International Journal Of Engineering And Computer Science, Vol-2, Issue-2, Feb 2013

[7] J. J. K. O. Ruanaidh, W. J. Dowling, and F. M. Boland. Watermarking digital images for copyright protection. I.E.E. Proceedings on Vision, Signal and Image Processing,

[8] Jaymala Chavan, Priyanka Desai, Relational Data Leakage Detection using Fake Object and Allocation Strategies, International Journal of Computer Applications, Vol-80, No.16, October 2013

[9] 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, Aug 2012.

[10] https://en.wikipedia.org/wiki/Digital_watermarking

Acknowledgement

I would like to express my deep gratitude to my project guide, Sachi Shah, Teaching Assistant, Computer Engineering Department, SVNIT Surat, for their valuable guidance, helpful feedback, and co-operation with a kind and encouraging at the initial stage. I would also like to thank Dr Udai Pratap Rao, Assistant Professor, Computer Engineering Department. I am also thankful to SVNIT Surat and its staff for providing this opportunity which helps me to gain sufficient knowledge to make my work successful.