A Seminar Report on:

“Data Leakage Detection”

Prepared by : Shubham Shekhaliya

Roll. No. : U18CO018

Class : B.Tech –IV (Computer Engineering) 7th Semester

Year : 2021-22

Guided by : TA9 - Shah Sachi

Department of Computer Engineering

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

CERTIFICATE

This is to certify that the seminar report entitled **Data Leakage Detection** is prepared and presented by **Mr. Shubham Shekhaliya** bearing Roll No.: **U18CO018**, 4th Year of **B. Tech (Computer Engineering)** and his/her work is satisfactory.

GUIDE	**JURY**	**HOD**
(TA-9 Sachi Shah)		**COED**

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 a particular industry's direct and indirect loss 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 different 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. Depending on the business, a person, or an industry, an organization's critical data might include client information, business goals, financial situation, patient information, credit-card data of employees, payment details, and so on. It raises the possibility of sensitive information falling into the wrong hands, whether by force or by accident (maliciously intended or an inadvertent mistake by an employee or a user). Now, what is needed is to locate the leaking point so that the leakage site (if accidental) or guilty actors may be identified (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

Because so much of our information is available online through social networking sites and third-party aggregators, the issue of data leakage is much more crucial nowadays. [2] LinkedIn, Facebook, Instagram, Twitter, and other social networking sites, as well as their third-party applications, all use a portion or all of its users' personal information, which they promise to keep private and safe. Consider the case of a user who has given three separate applications permission to access his data, which he believes is only a minor portion.

However, that piece of the data's security was compromised in some way. It would help create processes that would allow the parent site to identify the program responsible for the data leak to protect other users from identity theft, which is why the data leakage detection algorithm is so critical in the current 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 work, I compare and contrast different data allocation techniques that may be used to determine the likelihood or probability that a specific agent was responsible for the dataset leak. 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., used later in the algorithms. I also discuss the agent guilt model, which is a probabilistic model for determining the relative guilt of agents, a specification of the data allocation problem and how it should be handled, and an optimization model for the agent guilt model findings. Section 4 gives an overview of the various data allocation techniques that we are testing in the lab. Further, section 5 presents Algorithms analysis, followed by a conclusion in section 6, acknowledgment, 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 codes or sometimes destroyed if the data recipient is malicious.

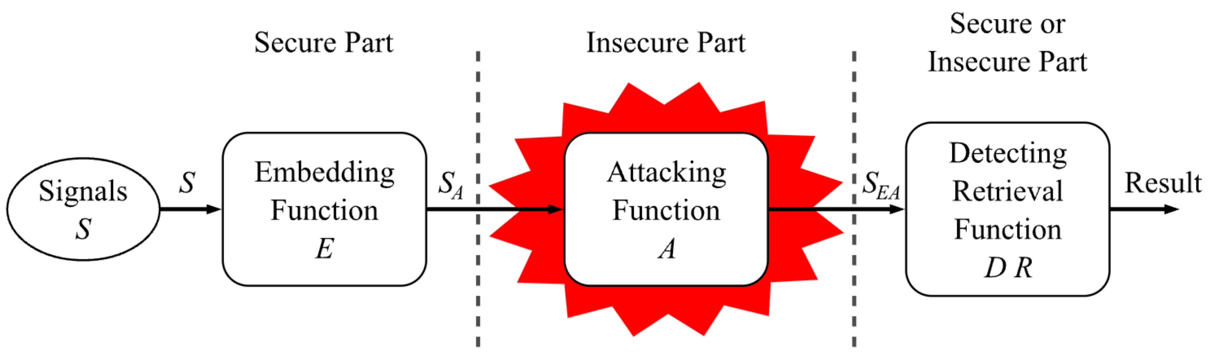


Figure 1: Digital Watermark life cycle [10]

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 to 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.

The author's findings [5], the introduction of essential data is not primarily 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 while exhibiting 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 stated here is how data is distributed efficiently to optimize the chances of identifying a guilty agent accountable for leaking the data to a suspicious 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 -

1. Sample Data Request – In this type of request, an agent needs to specify the number of data records (maybe tuples from a relation) required.
2. Explicit Data Request – This sort of data request is handled based on the agent's ability to expressly request specific data, such as tuples requested based on certain employee uid's in the employee relation.

Consider that D's requested items were distributed in response to requests from various agents A1, A2,... If a portion of the data set S was discovered to have been leaked to an untrusted source or an unknown identity, there are two possibilities: the leaked data objects were either guessed by the untrusted source, or the data object was released by a trustworthy agent. We need to use some data allocation techniques and probabilistic agent guilt models to determine the likelihood of the data items that have been leaked.

3.2 Agent Guilt Model Analysis

This paper presents a methodology for explicitly defining the idea of guilt in the context of the Data Leakage problem. If an agent Ai adds one or more items to the leaked data collection, we can claim it is guilty. Let Gi be the event in which agent Ai is found guilty, and Gi|S be the event in which agent Ai is found guilty of a specific leaked set S. We require an estimate of the likelihood that values in S can be "guessed" by the target to compute the $Pr\{G_i|S\}$. [3].

The following formula is used to compute Pr{G_i|S}:

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

Where V_d is the number of agents seeking each element in the leaked set S, and S is the leaked set. If the distributor finds a leaked table S containing all R_i items, Pr(G_j| S = A_i) or simply Pr(G_j|A_i) is the chance that agent A_j is responsible.

3.3 Data Allocation Problem

The proposed work is divided into two parts: the first is implementing and verifying round-robin allocation strategies for sample data requests (proposed by Papadimitriou and Garcia-Molina [1]), and the second is developing three new data allocation techniques and comparing the results with the previously proposed technique.

The suggested approaches have an additional benefit: before assigning any item to the following agent, an agent is happy (i.e., receives all of the objects requested). Furthermore, the data allocation to an agent in this scenario is independent of the agent's subsequent demands.

As a result, this type of data allocation may be expanded to handle requests when the number of agents is known ahead of time because data distribution to one agent is independent of the agents that are still to be assigned.

3.4 Optimization Problem

After meeting the agent's request, it is critical to rule out any chance of leaking involving some of the agents for a specific leaked set S. We utilize a notation to describe the distributor's goal for the same reason officially. Pr(G_j|A_i) is possible that agent A_j is guilty if the distributor finds a leaked table S containing all A_i objects [1], as previously described. We define the different functions Δ(i, j) as

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

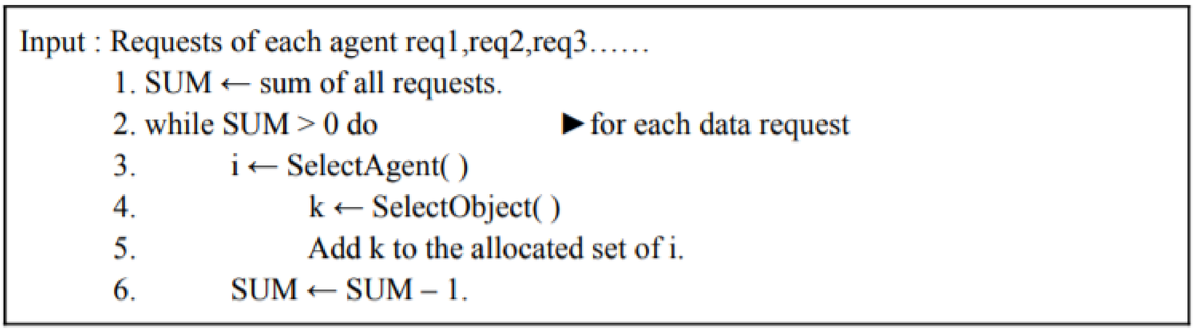
If the leaked set comprises all A_i objects and Agent A_i is at least as likely to be guilty as any other agent, then Δ(i, j) is a metric for measuring an agent's guilt. As a result, we may discover Δ(i, j) values for each such Δ(i, j) pair. For every agent A_j whose set does not contain all of the data from the leaked set, Difference(i, j) is positive. The minimal value of Δ(i, j) should be minimized if the agent who leaked the data is identified.

Chapter 4

Data Allocation Strategies

In this part, I explore different methods for distributing data objects to agents such that, in the event of a data breach, the agent responsible may be identified from the distributed dataset.

First, the allocation techniques for sample data requests were developed using the three algorithms s-random, s-overlap, and s-max based on Papadimitriou and Garcia Molina's round-robin methodology [1]. Figure 2 depicts their generic data allocation technique for data set D.



```
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 algorithm's running time is O (η λ SUM) since the loop runs SUM times. η and λ represent the running time of SelectAgent() and SelectObject(), respectively.

4.1 Object Allocation

The SelectObject() method chooses a data object for the agent I [1]. There are three methods for selecting objects:

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 method may result in poor data distribution. For example, if the distributor set D contains three items and three agents for each request, s-random may assign the same object to all three agents.

ii) s-overlap: Each agent can receive a unique data object using the s-overlap selection in the previous example. The distributor prepares a list of data items that have been assigned to the fewest number of agents in this Algorithm, and then a random object is chosen from this subset. When the sum of requests from all agents is less than the size of the data set, this algorithm produces a disjoint set; nevertheless, as SUM increases, it produces an object-sharing distribution.

iii) s-max: In this Algorithm, we assign an object to an agent that produces the smallest increase in the maximum relative overlap between any two agents. The s-max Algorithm is depicted in Figure 3.

```
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 $j \neq i$ 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 s-random, $\lambda = O(1)$, and for s-overlap, $\lambda = O(1)$ if we preserve the set of k objects allocated to the fewest number of agents in memory. The s-max algorithm takes $O(|D| n)$ time to run since the external loop runs for all 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: [1] proposes the round-robin strategy, in which data is allocated to agents by allocating one data item to each agent. Assume that each of the three agents requests three objects. The first data object is then assigned to the first agent, followed by the second agent, and finally the third agent. The first agent is provided one data object and so on until all of the agents are satisfied. The most notable drawback of this scenario is that if the first agent requests just two objects and there are 1000 agents, the first agent must wait until all 1000 agents have received one object before receiving the second object.

ii) First Come First Serve (FCFS): In this type, we entirely fulfill an agent before allocating any data object to the following agent; for example, in the preceding example, the first agent gets the three objects first, followed by the second agent, and finally the third agent. The agent who made the request first does not have to wait for other agents.

iii) Longest Request First (LRF): We first sort the agents in decreasing order of their demands, and then the requests are provided as in FCFS, i.e., the highest-requested agent is satisfied first, followed by the agent with the smallest request. In this instance, the agent with the greatest request does not need to wait for agents with lower demands. LRF, on the other hand, has a drawback. For example, Consider a dataset D = {d1, d2, d3, d4, d5} with three agents seeking 2, 3, and 1 objects, respectively. R1 = {d1, d2}, R3 = {d1}, and R2 = {d3, d4, d5} are examples of data allocations for FCFS or LRF using s-overlap or s-max. If d1 is leaked, the distributor will suspect agents A1 and A3 as well. If the third agent requests two objects, however, s-max returns R3 = {d1, d3}, and if both d1 and d3 are leaked, agent A3 is deemed guilty.

iv) Shortest Request First (SRF): As we can see from the example above, if the agent with the smallest requests is served after the largest requests, the allocation may be inadequate. As a result, we propose a scenario in which agents with smaller requests are satisfied first before moving on to those with greater needs. R3 = {d1}, R1 = {d2, d3}, and R2 = {d2, d4, d5} are the results of SRF using The s-max for the given example.

The SRF allocation is superior to the LRF allocation because, in the LRF allocation, R1 entirely overlaps R3, whereas, in the SRF allocation, no agent is completely overlapping by any other agent (R1 & R2 have d2 in common, but both have distinct data objects too).

Round-Robin and FCFS have the same time complexity O(1). In addition, if we retain a list of agents in memory that is sorted by request size, then η = O (1). On the other hand, the sorting of agents will take O(N logN) time and can be done with a priority queue.

Chapter 5

Algorithms Analysis

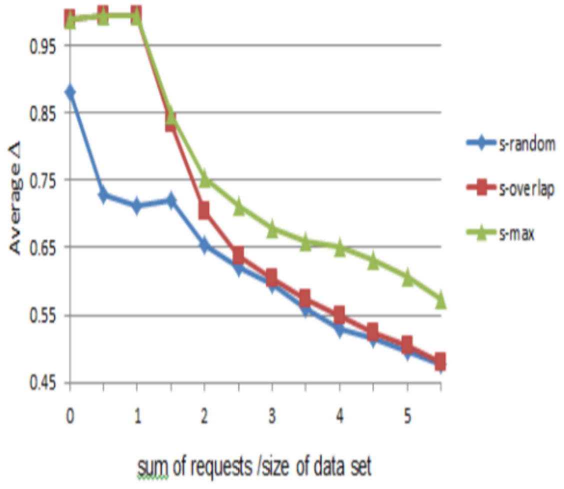
We will evaluate not only the algorithm's performance but also the approximation's effectiveness implicitly. In (3) and (4), the difference functions are employed as metrics to evaluate a given allocation using objective scalarization.

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

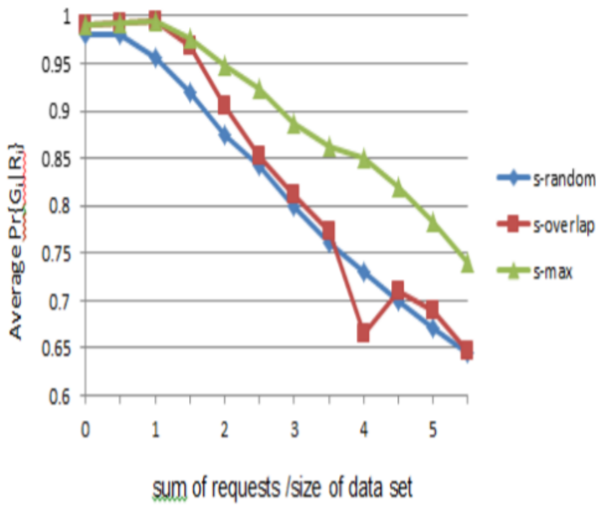
$$min\,\Delta = min_{i \neq j}\ i,j=1,,,n\Delta(i,j) \quad \ldots\ldots\ldots\ldots\ldots (4)$$

The average of Δ (i, j) values for a specific allocation Δ is the metric, and it indicates how successful the guilt detection is on average for this allocation. Agent i has spilled his data, and both he and another agent j have very similar guilt probabilities. Metric min Δ is the minimal Δ (i, j) value, and it corresponds to the circumstance where agent i has leaked his data, and both he and another agent j have very similar guilt probabilities. If min Δ is tiny, we will not distinguish between i and j as the leaker.

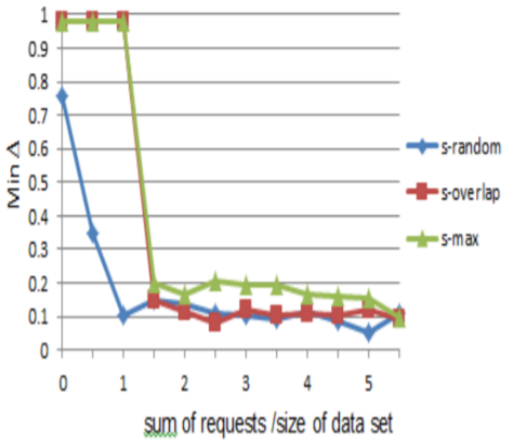
Agents are not requesting specific objects when making sample data requests. As a result, object allocation is not explicitly determined by the requests. Only if the total number of requests SUM exceeds the number of objects in set D does the distributor distribute some items to multiple agents. The ratio of SUM to the amount of the data collection, which is designated as load, is the metric that defines the severity of a problem with sample data requests. It is also worth noting that the absolute values of requests and the size of the data set matter less than the relative values req_i / |D|.



(a)

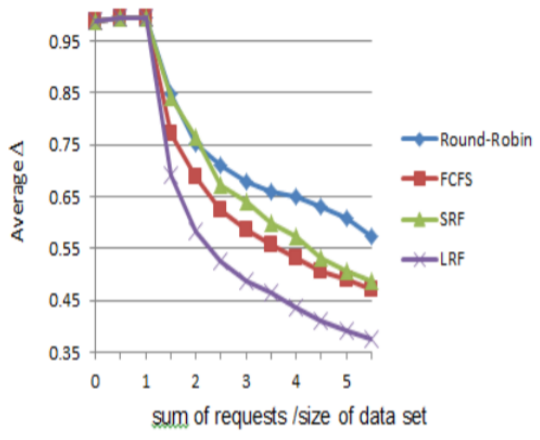


(b)

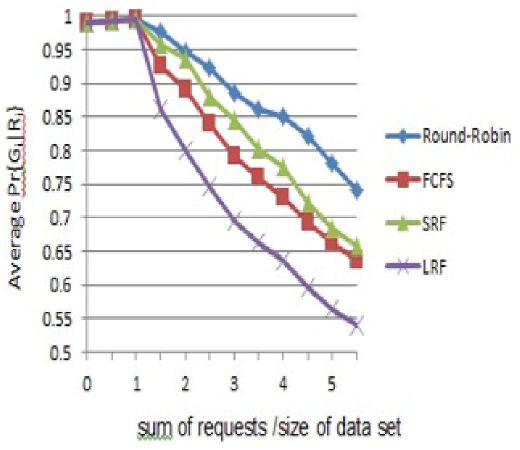


(c)

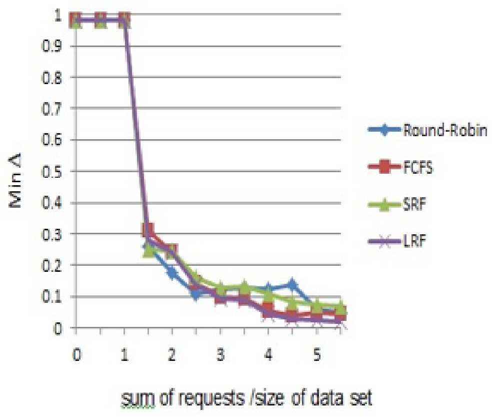
Figure 4. Evaluation of Sample Data Requests using Round Robin technique (a) Average Δ (b) Average Pr{Gi|Si} (c) Average minΔ [1]



(a)



(b)



(c)

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

We use a fixed dataset of 50 objects in our experiment and alter the load by increasing the number of agents. For two sorts of queries, we calculate the defined metrics:

a) All agents' requests are the same (say 5)

b) when the agent's requirements differ and are fulfilled in an ad hoc manner (unordered).

In this case, data is distributed to agents by allocating one data object to each agent, i.e., three agents each requesting three items. The first data object is then assigned to the first agent, then the second agent, and finally the third agent. The first agent is provided one data object and so on until all of the agents are satisfied.

The outcomes of each of these strategies were compared to the s-overlap and s-max algorithms. As previously stated, the results are calculated for two scenarios: each agent's request is the same and one in which each agent's request is different. SRF, LRF, and FCFS behave when there is no order in the first situation. The findings are calculated numerous times for both scenarios, and the graphs are plotted using the average of all the numbers.

For the s-overlap method, all four strategies act in a relatively similar way. When the agents' requests are varied and unordered, the Round-Robin values are slightly better than the other three situations.

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

Several methodologies were used in this study to determine the chance that an agent was responsible for any data leakage. The four strategies were used to implement the algorithms, and the distributor can employ one of them in the actual world depending on its needs. We discovered that distributing data wisely can boost the probability of effectively recognizing agents, especially when the data that agents must obtain overlap significantly. Our first goal was to double-check the results of previously presented methods [3]. In most circumstances, we found that the s-max method outperforms the s-overlap and s-random algorithms. They also put these strategies to the test and discovered that in the situation of s-overlap, all of them function equally well, and any of them can be employed depending on the distributor's application requirements. However, in the case of the s-max method, we discovered that Round-Robin and SRF outperform the other two. As a result, if the distributor intends to satisfy an agent before assigning any object to other agents, the SRF technique should increase the chances of discovering the leaker.

References

[1] Panagiotis Papadimitriou, Member, IEEE, Hector Garcia-Molina, Member, IEEE., Data Leakage Detection, IEEE Transactions On Knowledge & 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

Acknowledgment

I want 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.