# An Efficient Guilt Detection Approach for Identifying Data Leakages

Anand Kiran
*Assistant Professor, Department Of Computer Science & Engineering, Government Women's Polytechnic Muzaffarpur, Bihar, INDIA*

***Abstract* —** *In this paper 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. In a sense, the fake objects acts as a type of watermark for the entire set, without modifying any individual members. If it turns out an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.*

*A distributor owns a set T = {t1, t2, . . . , tm} of valuable data objects. The distributor wants to share some of the objects with a set of agents U1, U2,... Un, but does not wish the objects be leaked to other third parties. The objects in T could be of any type and size, e.g., they could be tuples in a relation, or relations in a database. An agent Ui receives a subset of objects Ri ⊆ T, determined either by a sample request or an explicit request:*

- *Sample request Ri = SAMPLE(T,mi): Any subset of mi records from T can be given to Ui.*
- *Explicit request Ri = EXPLICIT(T, condi): Agent Ui receives all the T objects that satisfy condi.*

*A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data is leaked and found in an unauthorized place (e.g., on the web or somebody's laptop). The distributor must assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means.*

**Keywords —** *Fake Object, Guilty Agent, Data Object, Third Party, Watermark, Data Warehousing.*

## I. INTRODUCTION

In this paper 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. In a sense, the fake objects acts as a type of watermark for the entire set, without modifying any individual members. If it turns out an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.

A distributor owns a set T = {t1, t2, . . . , tm} of valuable data objects. The distributor wants to share some of the objects with a set of agents $U_1$, $U_2$,… $U_n$, but does not wish the objects be leaked to other third parties. The objects in T could be of any type and size, e.g., they could be tuples in a relation, or relations in a database.

## II. SYSTEM ANALYSIS

### A. *Existing Systemt*

Traditionally, leakage detection is handled by watermarking, e.g., a unique code is embedded in each distributed copy. If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but gain, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious.

As far as the data allocation strategies are concerned, our work is mostly relevant to watermarking that is used as a means of establishing original ownership of distributed objects. Watermarks were initially used in images, video and audio data whose digital representation includes considerable redundancy. Our approach and watermarking are similar in the sense of providing agents with some kind of receiver-identifying information. However, by its very nature, a watermark modifies the item being watermarked. If the object to be watermarked cannot be modified then a watermark cannot be inserted. In such cases methods that attach watermarks to the distributed data are not applicable.

### B. *Proposed Systemt*

We propose data allocation strategies (across the agents) that improve the probability of identifying leakages. The two types of requests we handle were defined to sample and explicit. Fake objects are objects generated by the distributor that are not in set T. The objects are designed to look like real objects, and are distributed to agents together with the T objects, in order to increase the chances of detecting agents that leak data.

We have shown it is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the

probability that objects can be "guessed" by other means.

## III. LTERATURE SURVEY

we enunciate the need for watermarking database relations to deter their piracy, identify the unique characteristics of relational data which pose new challenges for watermarking, and provide desirable properties of a watermarking system for relational data. A watermark can be applied to any database relation having attributes which are such that changes in a few of their values do not affect the applications. We then present an effective watermarking technique geared for relational data. This technique ensures that some bit positions of some of the attributes of some of the tuples contain specific values. The tuples, attributes within a tuple, bit positions in an attribute, and specific bit values are all algorithmically determined under the control of a private key known only to the owner of the data. This bit pattern constitutes the watermark. Only if one has access to the private key can the watermark be detected with high probability. Detecting the watermark neither requires access to the original data nor the watermark. The watermark can be detected even in a small subset of a watermarked relation as long as the sample contains some of the marks. Our extensive analysis shows that the proposedtechnique is robust against various forms of malicious attacks and updates to the data. Using an implementation running on DB2, we also show that the performance of the algorithms allows for their use in real world applications.[1]

Data warehousing systems integrate information from operational data sources into a central repository to enable analysis and mining of the integrated information. During the integration process, source data typically undergoes a series of transformations, which may vary from simple algebraic operations or aggregations to complex "data cleansing" procedures.In a warehousing environment, the data lineage problem is thatof tracing warehouse data items back to the original source items from which they were derived. We formally define the lineage tracing problem in the presence of general data warehouse transformations, and we present algorithms for lineage tracing in this environment. Our tracing procedures take advantage of known structure or properties of transformations when present, but also work in the absence of such information. Our results can be used as the basis for a lineage tracing tool in a general warehousing setting, and also can guide the design of data warehouses that enable efficient lineage tracing.[2]

Privacy, preservation and performance ("3 P's") are central design objectives for secure distributed data management systems. However, these objectives tend to compete with one another. This paper introduces a model for describing distributed data management systems, along with a framework for measuring privacy, preservation and performance. The framework enables a system designer to quantitatively explore the tradeoff between the 3 P's

## IV. SYSTEM DESIGN

### A. *Module*

1) Data distribution
2) Fake objects
3) Sample request
4) Explicit request
5) Identifying the Guilty Agent

### B. *Module Description*

1. ***Data distribution:*** Distributor "intelligently" gives data to agents in order to improve the chances of detecting a guilty agent. The two types of requests we handle to distribute the data are sample and explicit. Fake objects are objects generated by the distributor that are not in set T. The algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker.

2. ***Fake objects:*** The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. Here, we model the creation of a fake object for agent $U_i$ as a black-box function CREATEFAKEOBJECT($R_i$, $F_i$, condi) that takes as input the set of all objects $R_i$ the subset of fake objects $F_i$ that $U_i$ has received so far and condi, and returns a new fake object. This function needs condi to produce a valid object that satisfies $U_i$'s condition. Set $R_i$ is needed as input so that the created fake object is not only valid but also indistinguishable from other real objects.

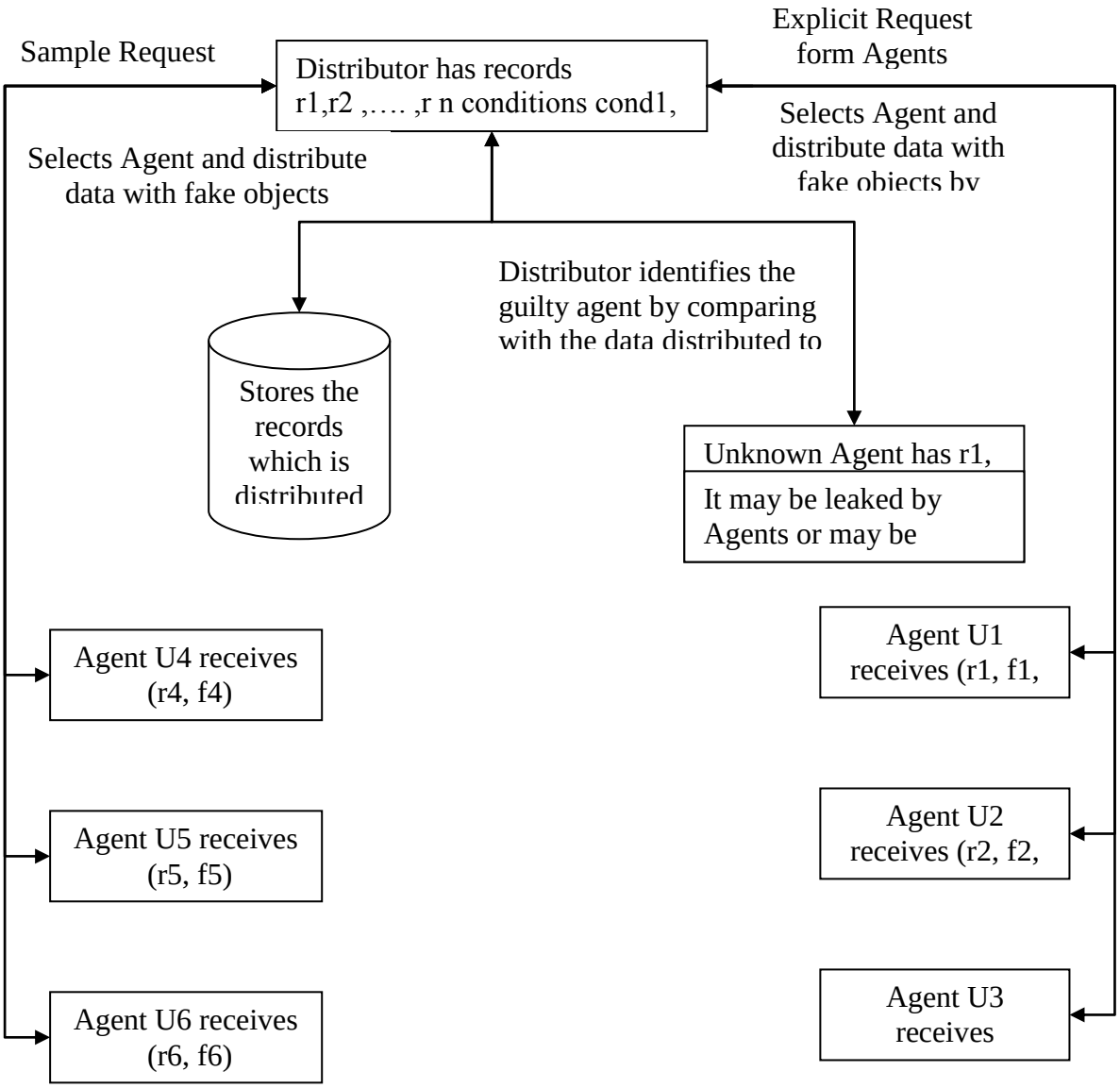
3. ***Sample request:*** Sample request Ri = SAMPLE(T,mi): Any subset of mi records from T can be given to $U_i$ With sample data requests, each agent Ui may receive any T subset out of different ones. Hence, there are different object allocations. In every allocation, the distributor can permute T objects and keep the same chances of guilty

4. agent detection. The reason is that the guilt probability depends only on which agents have received the leaked objects and not on the identity of the leaked objects.

5. ***Explicit request:*** Explicit request Ri = EXPLICIT (T, condi): Agent Ui receives all the T objects that satisfy condi With explicit data requests, each agent Ui may receive T subset which satisfies the condition condi. Hence, there are different object allocations. In every allocation, the distributor can permute T objects and keep the same chances of guilty agent detection. The reason is that the guilt probability depends only on which agents have received the leaked objects and not on the identity of the leaked objects.

6. ***Identifying the Guilty Agent:*** Suppose that after giving objects to agents, the distributor discovers that a set $S \subseteq T$ has leaked. This means that some third party called the *target* has been caught in possession of *S*. Since the agents $U1, \ldots, Un$ have some of the data, it is reasonable to suspect them leaking the data.

## V. ARCHITECTURAL DIAGRAM



## VI. SYSTEM IMPLEMENTATION

Implementation is the most crucial stage in achieving a successful system and giving the user's confidence that the new system is workable and effective. Implementation of a modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user. And so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions clearly and quickly.

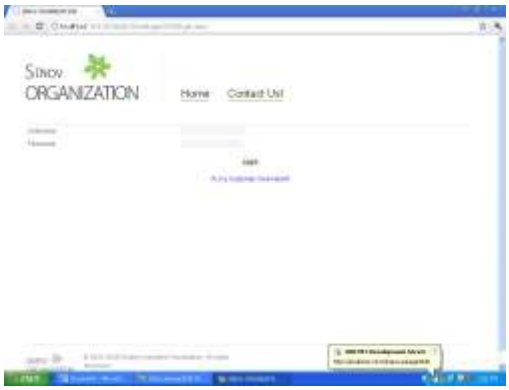
Initially as a first step the executable form of the application is to be created and loaded in the common server machine which is accessible to the entire user and the server is to be connected to a network. The final stage is to document the entire system which provides components and the operating procedures of the system.

## VII. FUTURE DEVELOPMENT

Every application has its own merits and demerits. The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Changing the existing modules or adding new modules can append improvements. Further enhancements can be made to the application, so that the web site functions very attractive and useful manner than the present one.

## VIII. METHODOLOGY

### *A. LOGIN FOR CUSTOMER, AGENT, ADMIN*




### *B. CUSTOMER PRODUCT DETAILS*


### *C. PRODUCT DETAILS OF CUSTOMER AND ADMIN –AGENT SITE*


### *D. ADMIN PAGE- REGISTER FORM TO ADD NEW AGENT*

### E. AGENT SITE TO ADD NEW CUSTOMER


### F. GENT SITE INFORMATION ABOUT THEIR RESPECTIVE CUSTOMERS


### G. ADMIN PADE FOR VEIWING THE ENTIRE CLIENT DETAILS AND PR ODUCT TRANSACTIONS

### H. ADMIN PAGE TO FIND GUILT AGENT

## IX. CONCLUSION

We have shown it is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the probability that objects can be "guessed" by other means.

Fake objects are objects generated by the distributor that are *not* in set *T*. The objects are designed to look like real objects, and are distributed to agents together with the *T* objects, in order to increase the chances of detecting agents that leak data.

Our model is relatively simple; the algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker.

## X. REFERENCES

[1] R. Agrawal and J. Kiernan,"Watermarking Relational Databases"Proc 28th Int'l Conf. Very Large Data Bases (VLDB '02), VLDB Endowment, pp. 155-166, 2002.

[2] P. Bonatti, S.D.C. di Vimercati, and P. Samarati, "An Algebra for Composing Access Control Policies," ACM Trans. Information and System Security, vol. 5, no. 1, pp. 1-35, 2002.

[3] P. Buneman, S. Khanna, and W.C. Tan, "Why and Where: A Characterization of Data Provenance," Proc. Eighth Int'l Conf.Database Theory (ICDT '01), J.V. den Bussche and V. Vianu, eds.,pp. 316-330, Jan. 2001.

[4] P. Buneman and W.-C. Tan, "Provenance in Databases," Proc.ACM SIGMOD, pp. 1171-1173, 2007.

[5] Y. Cui and J. Widom, "Lineage Tracing for General Data Warehouse Transformations," The VLDB J., vol. 12, pp. 41-58,2003.

[6] S. Czerwinski, R. Fromm, and T. Hodes, "Digital Music Distribution and Audio Watermarking,"http://www.scientificcommons.org/43025658, 2007.

[7] F. Guo, J. Wang, Z. Zhang, X. Ye, and D. Li, "An Improved Algorithm to Watermark Numeric Relational Data," Information Security Applications, pp. 138-149, Springer, 2006.

[8] F. Hartung and B. Girod, "Watermarking of Uncompressed and Compressed Video," Signal Processing, vol. 66, no. 3, pp. 283-301,1998.

[9] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian, "Flexible Support for Multiple Access Control Policies,"

ACM Trans. Database Systems, vol. 26, no. 2, pp. 214-260, 2001.

[10] 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.

[11] B. Mungamuru and H. Garcia-Molina, "Privacy, Preservation and Performance: The 3 P's of Distributed Data Management,"technical report, Stanford Univ., 2008.

[12] V.N. Murty, "Counting the Integer Solutions of a Linear Equation with Unit Coefficients," Math. Magazine, vol. 54, no. 2, pp. 79-81,1981.

[13] S.U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani,"Towards Robustness in Query Auditing," Proc. 32nd Int'l Conf.Very Large Data Bases (VLDB '06), VLDB Endowment, pp. 151-162,2006.

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

[15] P.M. Pardalos and S.A. Vavasis, "Quadratic Programming with One Negative Eigenvalue Is NP-Hard," J. Global Optimization,vol. 1, no. 1, pp. 15-22, 1991.