Advances in Computer Science and its Applications (ISSN 2166-2924)
Vol. 1, No. 2, June 2012


www.worldsciencepublisher.org

# A MODEL FOR IMPROVING GUILT PROBABILITIES TO IDENTIFY DATA LEAKAGES

**[1]A. Mounika, [2]M. Satwik, [3]G. Rajesh Chandra**

[123]Dept of ECM, KLCE, A.P, INDIA

Email: [1]b.mounikaaluri@gmail.com, [2]motamarrisatwik@gmail.com, [3]grajeshchandra@gmail.com

**ABSTRACT:** Sometimes sensitive data may be distributes by a data distributor to a set of trusted agents (third parties), but some of the distributed data we may find in an unauthorized website or systems. So the distributor must assess the probability of data leakage from its trusted agents. So in this project we proposed a model for assessing the guilt probabilities of agents, in a way that improves the chances of identifying a data leakage. We also propose algorithms for distributing objects to agents. Finally we also consider the option of adding fake data objects to the distributed data set. Such objects do not correspond to the real entries but appear realistic to the trusted agents. These fake objects used like watermarking for the entire dataset. These schemes do not make any alternations of the released data.

**Keywords:** Perturbation; data leakage; fake records; leakage model.

## 1. INTRODUCTION

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the distributor and the supposedly trusted third parties the agents. Our goal is to detect when the distributor's sensitive data have been leaked by agents, and if possible to identify the agent that leaked the data [1].

We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data are modified and made "less sensitive" before being handed to agents. For example, one can add random noise to certain attributes, or one can replace exact values by ranges [2]. However, in some cases, it is important not to alter the original distributor's data. For example, if an outsourcer is doing our payroll, he must have the exact salary and customer bank account numbers. If medical researchers will be treating patients (as opposed to simply computing statistics), they may need accurate data for the patients.

Traditionally, leakage detection is handled by water-marking, 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 again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. In this paper, we study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a website, or may be obtained through a legal discovery process.) At this point, the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with five cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings.

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

The rest of the paper is organized as follows. Section2 describes about various software used for the implementation of the project. Proposed project work is depicted in Section3. Section 4 gives conclusion to the paper. Future Scope is pointed out in Section 5.

## 2. SOFTWARES USED

### *2.1 Front-End (Java)*

Java is a programming language originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented

language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java is currently one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Class path. One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java byte code, instead of directly to platform-specific machine code. Java byte code instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets. Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using byte code is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executable would. Just-in-Time (JIT) compilers were introduced from an early stage that compiles byte codes to machine code during runtime.

### *2.2 MS Access:*

Microsoft Office Access, previously known as Microsoft Access, is a database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately. On May 12 2010, the current version of Microsoft Access 2010 was released by Microsoft in Office 2010; Microsoft Office Access 2007 was the prior version.MS Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other applications and databases.

Software developers and data architects can use Microsoft Access to develop application software, and "power users" can use it to build software applications. Like other Office applications, Access is supported by Visual Basic for Applications, an object-oriented programming language that can reference a variety of objects including DAO (Data Access Objects), ActiveX Data Objects, and many other ActiveX components. Visual objects used in forms and reports expose their methods and properties in the VBA programming environment, and VBA code modules may declare and call Windows operating-system functions.

### *2.3 SQLite:*

SQLite implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it. SQLite read operations can be multitasked, though writes can only be performed sequentially.

SQLite is a popular choice for local/client storage on web browsers. It has many bindings to programming languages. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others. SQLite implements most of the SQL-92 standard for SQL but it lacks some features. For example it has partial support for triggers, and it can't write to views (however it supports INSTEAD OF triggers that provide this functionality). While it supports complex queries, it still has limited ALTER TABLE support, as it can't modify or delete columns. SQLite has automated regression testing prior to each release. Over 2 million tests are run as part of a release's verification. Starting with the August 10, 2009 release of SQLite 3.6.17, SQLite releases have 100% branch test coverage, one of the components of code coverage.

## 3. **RESULT**

The project illustrated in this paper is entirely based on the idea of detecting the agent who leaked the data. Here, the main objective of this project is to add fake objects to the originally distributed data in order to find out the guilt agent. It includes 4 modules: Data Allocation, Fake object, Optimization, Data distributor.

### *3.1 Data Allocation*

The main focus of my project is the data allocation problem as how can the distributor “intelligently” give data to agents in order to improve the chances of detecting a guilty agent. In this module, administrator has to login with his id and password. Administrator has all the agent information, user data inside his database. Administrator is now able to view the database consisting of the original data as well as the fake data. Administrator can also list the agents here. He will be able to add additional information to the database. Agent’s information can be added here [3].

### *3.2. Fake Object:*

Fake objects are objects generated by the distributor in order to increase the chances of detecting agents that leak data. The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness

in detecting guilty agents. Our use of fake objects is inspired by the use of "trace" records in mailing lists.

| Name | ID | Branch | eMail |
|---|---|---|---|
| harold | 215 | CSE | harold@gmail.com |
| stephen | 220 | ECE | stephen@yahoo.com |
| carl | 232 | CSE | carl21232@in.com |
| john | 235 | EEE | john123@gmail.com |
| jennifer | 240 | ECE | jeniffer@yahoo.com |
| mark | 243 | IST | markmark@in.com |
| james | 261 | EEE | jameseee@yahoo.com |
| marcus | 273 | IST | mightymarcus@gmail.com |

*Fig-3.1: Original Database*

| Name | ID | Branch | e-Mail |
|---|---|---|---|
| henry | 312 | ecm | henryhen@gmail.com |
| sam | 321 | mec | samsam@gmail.com |
| jill | 322 | cse | jill12jill@gmail.com |
| robin | 341 | ist | robinhood@yahoo.com |
| sid | 342 | ipe | sid123@yahoo.com |
| mona | 355 | ist | mona123@gmail.com |
| jackson | 356 | ecm | peterjackson@yahoo.com |
| peter | 364 | mec | petereng@gmail.com |
| madison | 365 | ecm | madisonman@yahoo.com |
| jack | 375 | ece | jaackhero@yahoo.com |

*Fig-3.2: Fake Objects*

### *3.3 Optimization*

The Optimization Module is the distributor's data allocation to agents has one constraint and one objective. The distributor's constraint is to satisfy agents' requests, by providing them with the number of objects they request or with all available objects that satisfy their conditions. His objective is to be able to detect an agent who leaks any portion of his data.

### *3.4 Data Distribution*

A data distributor has given sensitive data to a set of supposedly trusted agents. Some of the data is leaked and found in an unauthorized place. 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 [4],[5].

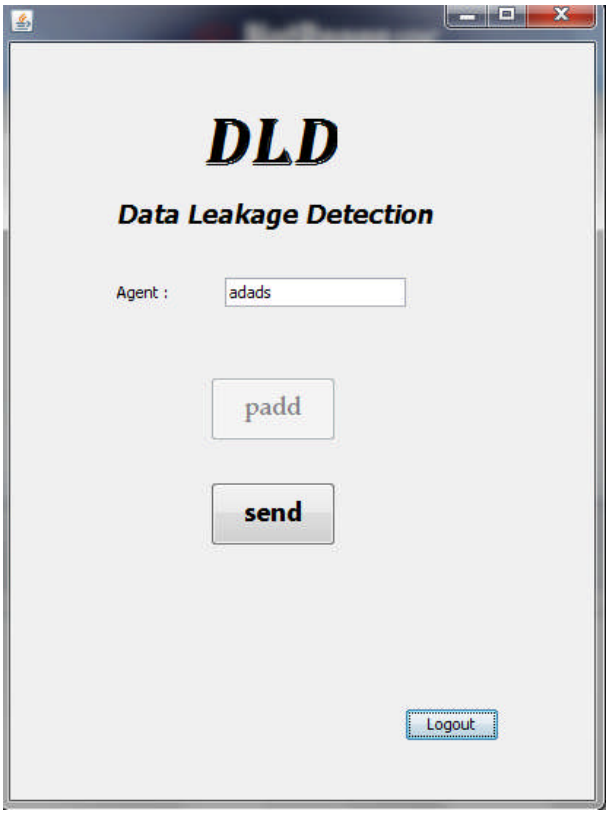


*Fig-3.3: Distribution Page*

### *3.5 Screenshots*

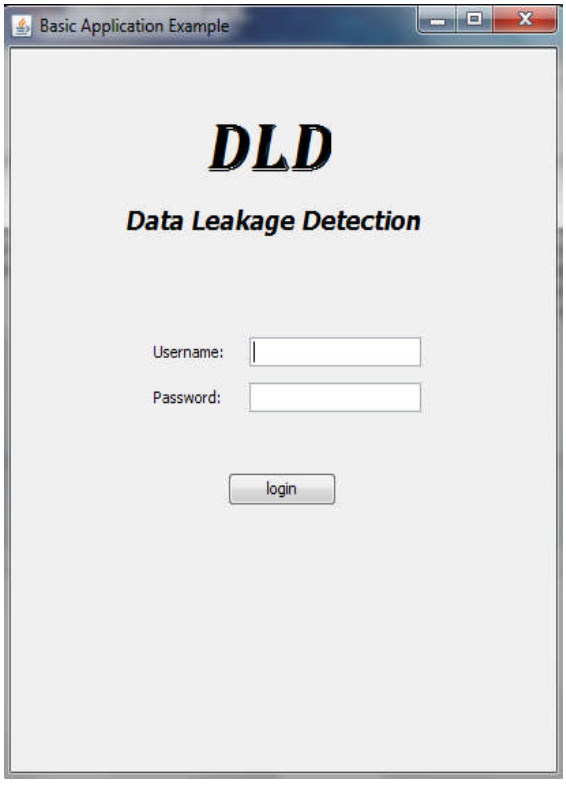


*Fig-3.4: Login Page*

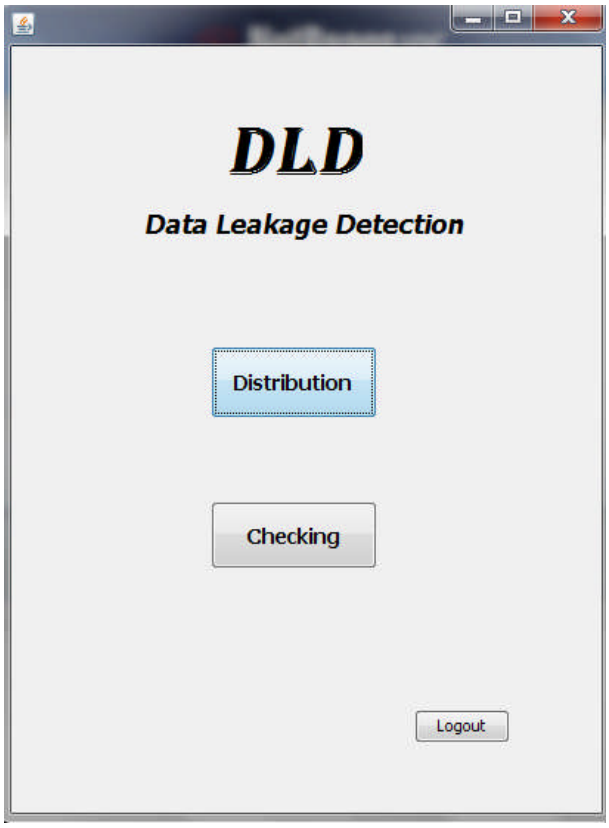


*Fig-3.5: Welcome Page*


*Fig-3.7: Found Guilt Agent*

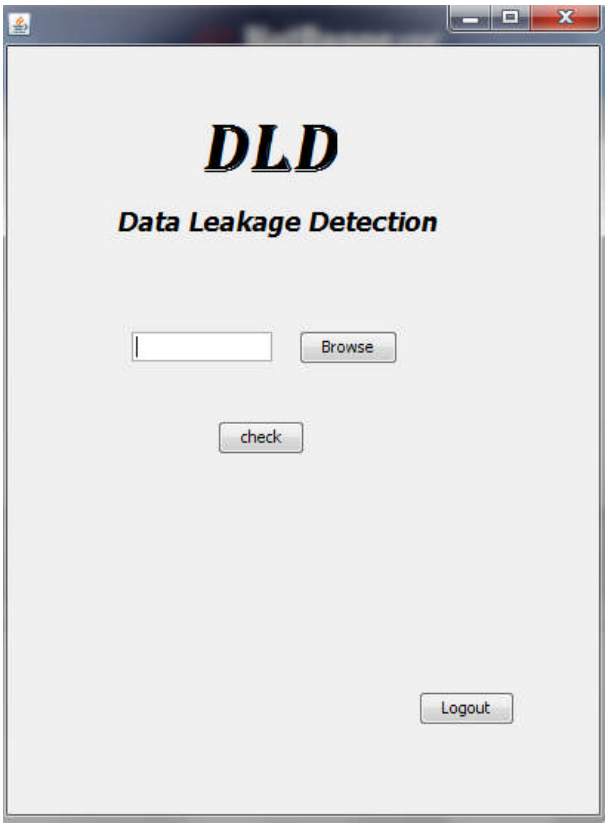


*Fig-3.6: Detection or Checking*

## 4. Conclusion

In a perfect world there would be no need to hand over sensitive data to agents that may unknowingly or maliciously leak it. And even if we had to handover sensitive data, in a perfect world we could watermark each object so that we could trace its origins with absolute certainty. However, in many cases we must indeed work with agents that may not be 100% trusted, and we may not be certain if a leaked object came from an agent or from some other source, since certain data cannot admit watermarks. In spite of these difficulties, 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. Our model is relatively simple, but we believe it captures the essential trade-offs. The algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. We have shown that distributing objects judiciously can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive. It includes the investigation of agent guilt models that capture leakage scenarios that are not studied in this paper. For example, what is the appropriate model for cases where agents can collude and identify fake tuples? A preliminary discussion of such a model is available in.

## 5. Future Scope

The developer of an application can never be carried out to the fullest extend in a stipulated time, the main reason why revisions of the application are always introduced in course of time. This application being restricted to one time development will have no revision done, hence certain areas that can be enhanced is pointed. The enhanced version of this system can detect the guilty agent even through the internet.

## ACKNOWLEDGEMENT

We thank our H.O.D Dr. Balaji and Principal K. Rajasekhar Rao for giving us the eminent facilities to perform our research. We are obliged to whole Electronics and Computers department, KLCE for their timely help and support.

## References

1. Panagiotis Papadimitriou, and Hector Garcia-Molina, "Data Leakage Detection", IEEE transactions on knowledge and data engineering, vol. 23, no. 1, January 2011.
2. L. Sweeney, "Achieving K-Anonymity Privacy Protection Using Generalization and Suppression," http://en.scientificcommons. org/43196131, 2002.
3. 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.
4. 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.
5. P. Buneman and W.-C. Tan, "Provenance in Databases," Proc. ACM SIGMOD, pp. 1171-1173, 2007.
6. Y. Zhang, L. Wu, "Artificial Bee Colony for Two Dimensional Protein Folding", Advances in Electrical Engineering Systems, vol.1, no.1, pp.19-23, 2012
7. Maziar Rezaei Rad, Mani Rezaei Rad, Shahabeddin Akbari, "A Maximum Power Point Tracker for Photovoltaic Arrays Using Genetic Algorithm Plus Fuzzy Cognitive Networks", Advances in Electrical Engineering Systems, vol.1, no.1, pp.35-40, 2012
8. Sajad Farhangi, Saeed Golmohammadi, "A Comparative Study of IS-IS and IGRP Protocols for Real-Time Application Based on OPNET", Advances in Electrical Engineering Systems, vol.1, no.1, pp.65-70, 2012