Privacy Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data

M. Gowsika[1] Dr. A. Saradha[2]
[1]M.E Student, [2]Professor,
Department of Computer Science and Engineering,
Institute of Road and Transport Technology,
Erode.

***Abstract* - With the advancement of cloud computing, the secure data which is outsourcing, unauthorized accesses is at risk. To protect data privacy, the secure data should be encrypted before outsourcing done by the data owner. It is an important thing to explore secure encrypted cloud data search service. A practically efficient and flexible searchable encrypted scheme which supports multi-keyword ranked search. 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. These methods do not rely on alterations of the released data (e.g., watermarks). In some cases we can also inject "realistic but fake" data records to further improve our chances of detecting leakage and identifying the guilty party.**

Keywords:* *Multi-keyword search, ranked search, encrypted cloud data, security, Secure data.

I INTRODUCTION TO CLOUD COMPUTING

Cloud Computing is a new but increasingly mature model of enterprise IT infrastructure that provides on-demand high quality applications and services from a shared pool of configuration computing resources.The cloud customers, individuals or enterprises, can outsource their local complex data system into the cloud to avoid the costs of building and maintaining a private storage infrastructure. However, some problems may be caused in this circumstance since the Cloud Service Provider (CSP) possesses full control of the outsourced data. Unauthorized operation on the outsourced data may exist on account of curiosity or profit. To protect the privacy of sensitive information, sensitive data (e.g., emails, photo albums, personal health records, financial records, etc.) should be encrypted by the data owner before outsourcing, which makes the traditional and efficient plaintext keyword search technique useless. The simple and awkward method of downloading all the data and decrypting locally is obviously impractical. So, two aspects should be concentrated on to explore privacy-preserving effective search service. Firstly, ranked Search, which can enable data users to find the most relevant information quickly, is a very important issue.

The number of documents outsourced to the cloud is so large that the cloud should have the ability to perform search result ranking to meet the demand for effective data retrieval. Secondly, multi-keyword search is also very important to improve search result accuracy as single keyword search often return coarse search results. A practically efficient and flexible searchable encrypted scheme. To address multi-keyword search and result ranking, we use Vector Space Model (VSM) to build document index. To improve search efficiency, we use a tree-based index structure which is a balanced binary tree. We construct the searchable index tree based on the document index vectors.

Our contributions are summarized as follows:

(1) We study the problem of multi-keyword ranked search over encrypted cloud data while supporting strict privacy requirements.

(2) With the index tree designed in this paper, the search time complexity close to O (r log m) (r is the number of documents including the search keywords and m is the whole number of documents in the dataset).

The owner of the data is called distributor and receiver of the data is called as agent. The data leaked by agents to other parties called as target. The distributor distributes data to set of agents. Any of the agents may leak data to target. Distributor must assess the likelihood that who leaks the data. Distributor develops the guilt agent model to find out exact leaker in which the probability is drawn for leaked data.

II BACKGROUND AND RELATED WORK

Consider there are three different entities in the system model: the data owner, the user, and the cloud server respectively. The data owner encrypts document collection in the form of before outsourcing it to the cloud in order to protect the sensitive data from unauthorized entities. And for the purpose of searching interested data, the data owner will also generate an encrypted searchable index based on a set of distinct keywords extracted from. In the search stage, the system will generate an encrypted search trapdoor based on the keywords entered by the user (has been authorized by data owner). Given the trapdoor, the cloud server will search the index and then return the ranked search results to the user. As the search results are well ranked by the cloud server, the user can send a parameter together with search

query to get top-most relevant documents. As the issue of key distribution is out of the scope of this paper, we assume that data users have been authorized by data owner.

In some cases it is important not to alter the original distributor's data. 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 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 web site, 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. 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.

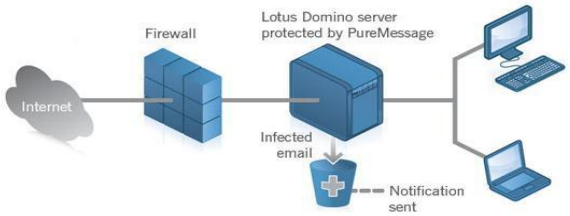
III EXISTING SCHEME

The need of effective data retrieval is, the large amount of documents demand the cloud server to perform result relevance ranking, instead of returning undifferentiated results. Such ranked search system enables data users to find the most relevant information quickly, rather than burdensomely sorting through every match in the content collection. Ranked search can also elegantly eliminate unnecessary network traffic by sending back only the most relevant data, which is highly desirable in the "pay-as-you-use" cloud paradigm. For privacy protection, such ranking operation, however, should not leak any keyword related information. On the other hand, to improve the search result accuracy as well as to enhance the user searching experience, it is also necessary for such ranking system to support multiple keywords search, as single keyword search often yields far too coarse results.

IV PROPOSED SCHEME

Our goal is to detect when the distributor's sensitive data has been leaked by agents, and if possible to identify the agent that leaked the data. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents. we develop *unobtrusive* techniques for detecting leakage of a set of objects or records.

In this section 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. *Problem Setup and Notation:*

A distributor owns a set $T=\{t_{1,\ldots},t_m\}$of valuable data objects. The distributor wants to share some of the objects with a set of agents $U_1,U_2,\ldots U_n$, 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. An agent Ui receives a subset of objects, determined either by a sample request or an explicit request:

1. *Sample request*
2. *Explicit request*

B. *Module Description*

1. ***Fake objects:*** Fake objects are objects generated by the distributor in order to increase the chances of
detecting agents that leak data. The distributor is capable to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. The use of fake objects by the system is inspired by the use of "trace" records in mailing lists. Fake objects are the objects which look exactly like the real objects. Fake objects are created by the distributor before sending data to agents. In every data which distributor will send to his agents, the position and number of fake objects will differs. Depending on the number of records the number of fake objects will differ so that it will be easy for system to detect the guilty agent.

2. Data allocation strategies: The data allocation problem as how can the distributors "intelligently" gives data to agents in order to improve the chances of detecting a guilty agent. Data allocation depends on the request done by the agent and whether system can add fake object to it. The request done by the agent can be of two types:

Sample- In sample data request agent receives a subset of distributers data which required by agent.

Explicit- In explicit data request data satisfying a special condition is given to agent.

3. Optimization Module: 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.

4. Data Distributor: 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.

C. Algorithms:

Guilt Model Analysis:

Our model parameters interact and to check if the interactions match our intuition, in this section we study two simple scenarios as **Impact of Probability** p and **Impact of Overlap between** Ri **and** S. In each scenario we have a target that has obtained all the distributor's objects, i.e., T = S.

T is a set of data objects.

T= {*t1*, *t2*... *tn*};

Set of agents= {*U1*, *U2*... *Un*}

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 objects *Ri*, where *Ri* is a subset of *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*.

We say an agent *Ui* is *guilty* and if it contributes one or more objects to the target. We denote the event that agent *Ui* is guilty as *Gi* and the event that agent *Ui* is guilty for a given leaked set *S* as *Gi*|*S*.

1. Algorithm for Find Guilt Agent:

a. Distributor select agent to send data. Distributor selects the agents to send the data according to agent request.

b. Distributor creates fake data and allocates it to the agent. The distributor can create fake data and distribute with agent data or without fake data. Distributor is able to create more fake data; he could further improve the chance of finding guilt agent.

c. Check number of agents, who have already received data. Distributor checks the number of agents, who have already received data.

d. Check for remaining agents. Distributor chooses the remaining agents to send the data. Distributor can increase the number of possible allocations by adding fake data.

e. Estimate the probability value for guilt agent. To compute this probability, we need an estimate for the probability that values can be "guessed" by the target.

2. Evaluation of Explicit Data Request Algorithms

In the first place, the goal of these experiments was to see whether fake objects in the distributed data sets yield significant improvement in our chances of detecting a guilty agent. In the second place, we wanted to evaluate our e-optimal algorithm relative to a random allocation.

3. Evaluation of Sample Data Request Algorithms

With sample data requests agents are not interested in particular objects. Hence, object sharing is not explicitly defined by their requests. The distributor is "forced" to allocate certain objects to multiple agents only if the number of requested objects exceeds the number of objects in set T. The more data objects the agents request in total, the more recipients on average an object has; and the more objects are shared among different agents, the more difficult it is to detect a guilty agent.

D. Performance Analysis

In this section, we estimate the overall performance of our proposed scheme by implementing the secure search system using C# language on a Windows7 server with Intel(R) Core(TM)2 Quad CPU 2.83GHz. The document set is built from the real-world data set: Request for comments database (RFC), which includes about 6500 publications.

The effectiveness of a system is most commonly described with its "Record wise leak report" and "Probability of agent guilty".

$$\text{Record wise leak report} = \frac{|T \cap S|}{|S|}$$

This formula calculates records wise leak data without considering agents overlapping. According to that the distributor knows which agents consider for calculating guilt probability.

$$\text{Probability of agent guilty} = \frac{|Ri \cap S|}{|S|}$$

This formula calculates records wise leak data with considering agents overlapping mean that particular record share by how many agents. According to that the distributor knows which agents consider for calculating guilt probability. It will enhance the Agent Probabilities.

V CONCLUSION AND FUTURE WORK

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 hand over 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. 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. Our future work 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 .Another open problem is the extension of our allocation strategies so that they can handle agent requests in an online fashion (the presented strategies assume that there is a fixed set of agents with requests known in advance).

REFERENCES

[1] Data Leakage Detection And E-Mail Filtering Mr.Zarif Shaukat Ansari 1, Ms. Anagha Mahadeo Jagtap 2, Ms. Shilpa Suresh Raut Student, Dept. Of Computer Engineering, Trinity College Of Engineering, Pune, Maharashtra, India.

[2] An Efficient And Robust Model For Data Leakage Detection System Janga Ajay Kumar 1 And K. Rajani Devi 2 1Student, M.Tech (IT), Siddharth Nagar, Nalanda Institute Of Engineering And Technology, A.P, India.

[3] Allocation Strategies For Detecting And Identifying The Leakage And Guilty Agents Lata Dudam[1], Dr.Prof.Mrs.S.S.Apte[2] Walchand Institute Of Technology,Solapur, India.

[4] R. Agrawal And J. Kiernan. Watermarking Relational Databases. In VLDB '02: Proceedings Of The 28th International Conference On Very Large Data Bases, Pages 155–166. VLDB Endowment, 2002.

[5] F. Guo, J. Wang, Z. Zhang, X. Ye, And D. Li. Information Security Applications, Pages 138–149. Springer, Berlin / Heidelberg, 2006. An Improved Algorithm To Watermark Numeric Relational Data.

[6] F. Hartung And B. Girod. Watermarking Of Uncompressed And Compressed Video. Signal Processing, 66(3):283–301, 1998.

[7] An Image Watermarking Method Based On Mean-Removed Vector Quantization For Multiple Purposes Zhe-Ming Lu, Zhen Sun, Department Of Automatic Test And Control, Harbin Institute Of Technology, Harbin 150001,China.

[8] P. Bonatti, S. D. C. Di Vimercati, And P. Samarati. An Algebra For Composing Access Control Policies. ACM Trans. Inf. Syst. Secur., 5(1):1–35, 2002.

[9] Provenance. In J. V. Den Bussche And V. Vianu, Editors, Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings, Volume 1973 Of Lecture Notes In Computer Science, Pages 316–330. Springer, 2001.

[10] P. Buneman And W.-C. Tan. Provenance In Databases. In SIGMOD '07: Proceedings Of The 2007 ACM SIGMOD International Conference On Management Of Data, Pages 1171–1173, New York, NY, USA, 2007. ACM.

[11] Y. Cui And J. Widom. Lineage Tracing For General Data Warehouse Transformations. In The VLDB Journal, Pages 471–480, 2001.

[12] F. Guo, J. Wang, Z. Zhang, X. Ye, And D. Li. Information Security Applications, Pages 138–149. Springer, Berlin / Heidelberg, 2006. An Improved Algorithm To Watermark Numeric Relational Data.

[13] S. Jajodia, P. Samarati, M. L. Sapino, And V. S. Subrahmanian. Flexible Support For Multiple Access Control Policies. ACM Trans. Database Syst., 26(2):214–260, 2001.

[14] Y. Li, V. Swarup, And S. Jajodia. Fingerprinting Relational Databases: Schemes And Specialties. IEEE Transactions On Dependable And Secure Computing, 02(1):34–45, 2005.

[15] B. Mungamuru And H. Garcia-Molina. Privacy, Preservation And Performance: The 3 P's Of Distributed Data Management. Technical Report, Stanford University, 2008.