Oblivious Information Retrieval on Outsourced Database Servers

Sunil B Mane, Pradeep K Sinha

Abstract— The internet has become a ubiquitous source of information. But while the amount of information has increased, so has a possibility of invasion of privacy. How can a user query a server for a set of web documents, while maintaining their relevance rankings without revealing his query to the server and without letting the server know which documents have been retrieved? In this paper, we develop a protocol for retrieval of web documents while guaranteeing user privacy as well as server privacy if needed. We also describe a technique for preventing information about the documents from being revealed to the server itself.

Index Terms— Database Outsourcing, Data Privacy, Document Searching, Encrypted Databases, Oblivious Information Retrieval, Pallier Cryptosystem,Vector Space Model.

1 Introduction

Since the last few years, the web has become the primary source of information. The web contains both public information intended for the world as a whole as well as private information intended only for a restricted audience set. Public information includes universally accessible information such as public websites like Wikipedia or Youtube. Private information includes internal documents of companies, health records, etc. While new technology is improving our experience online, it is also increasing the possibilities of invasions of privacy. It would be desirable for users to query a set of web documents from a search engine, while preserving the privacy of their search queries.

In this paper, we introduce a new protocol for information retrieval in the form of web documents while preserving the privacy of the user and if necessary, the privacy of the server as well. This allows a user to query a server for documents without revealing his query or his results set to the server and if necessary, prevent the user from obtaining any information that is not related to his query. To build this protocol, we use the Paillier cryptosystem, which provides additive homomorphic encryptions. In order to identify similar documents, we consider them as bags of words (according to the vector space model) and quantify their similarity using Pearson's correlation coefficient. Similarly, we also demonstrate a technique for building search indexes on web pages without revealing information about those web pages. To hide information about these documents from the server, we use a form of micro aggregation, which can guarantee k-anonymity. Finally, we evaluate our model in terms of the privacy it provides and the computational overhead involved. We also evaluate an implementation of this model on real world data sets and present our results.

- *Sunil B Mane, pursuing PhD from University of Pune and Assistant Professor, Computer Engg. & IT, College of Engineering, Pune E-mail: sunilbmane@gmail.com*
- *Pradeep K Sinha, Senior Director, Corporate Strategy and R&D, C-DAC, Pune E-mail: psinha@cdac.in*

2 Background and Related Work

Information retrieval techniques allow a user to query a collection of information and obtain a set of zero or more information elements which are relevant to his query. The most common information retrieval scenario and the one considered in this work is the retrieval of relevant results from search engines. A search engine in its most basic form crawls the web for web documents and indexes them on the basis of keywords within those documents. When a user queries a search engine, the search engine responds with a ranked set of web documents, which it believes (based on its ranking algorithm) are the most relevant to the user's query. Information retrieval is one of the fundamental areas of computer science and has received a large amount of attention for many years [100,108]. We use the document vector scheme as proposed by Salton et. al. [55] to represent documents. This is a very well established model for information retrieval and has been used to good effect in many systems such as the ontology based information retrieval system by Castells et al [96], and the cross document co-referencing model by Bagga and Baldwin [8]. Based on this model, a number of different methodologies exist for calculating document similarities as well as performing text categorization and clustering such as cosine similarity, Jaccard coefficient, Kullback-Leibler distance and Pearson's correlation coefficient. In this work we use the Pearson correlation coefficient as a similarity function. However, in this form, as described in the previous example, information retrieval leaks a lot of information about the user to the server - his query and his search results. This information, which the user gives to the server implicitly, may be used by the server for malicious purposes or for such purposes for which the user has not given his consent.

A possible solution to this problem is that of privacy preserving information retrieval. This scheme allows the user to maintain his/her privacy while searching for web documents. It has the added advantage of being extended to preserve the privacy of the server and the web documents as well. Privacy preserving information retrieval while not new, has gained

increased interest in recent years due to the ubiquity of the Internet and all the vulnerabilities it holds. A fundamental model for privacy protection is the k-anonymity model as described by Sweeney. Abril et al [29] consider the concept of privacy preservation through semantic micro aggregation in which they build a model for preserving the anonymity of private web pages via the application of the k-anonymity model. They use wordnet based Wu-Palmer distance and a derived web vector distance for identifying similar document vectors.

3 Oblivious Transfer

Oblivious Transfer (OT) is a protocol by which a sender sends one of multiple pieces of information to the receiver, but does not know which piece of information has been transferred. Oblivious transfer was first introduced by Rabin in 1981 [84] and later developed by Naor and Pinkas [83]. In the context of information retrieval, an oblivious transfer protocol is one in which a user queries a data store and receives an information element from the store. In this scenario, the data store does not know which element was queried or sent back and the user does not receive any information other than that which he/she queried. OT requires user privacy as well as server privacy. In this work, we use a simple XOR based oblivious transfer scheme for transferring web documents (or their corresponding URLs) from the server to the client. This scheme has also been used by Sabbu et al [98] for developing an oblivious retrieval protocol for images using SIFT.

Assume an honest but curious server. Suppose the client wishes to receive a set of document URLs from the server but desires that the server should not know what he/she is querying. The server maintains a set of N document URLs $D = \{d_1,d_2,...,d_N\}$. Let $(Pub, Priv)$ be the public and private keys of the server obtained using a standard public key encryption scheme. We denote encryption of a using the public key Pub by $[[a]]$ and decryption using the private key $Priv$ by $]]a[[$ so that $]](([[a]])[[= a$.

1. The client wishes to obtain a set of those documents given by the indices $X = \{x_1,x_2,...,x_m\}$ where $x_i \in \mathbb{Z}$. Now, in order to mask his/her query, the client chooses a set of N random numbers $R = \{r_1,r_2,...,r_N\}$ and encrypts the numbers at the positions $\{x_1,x_2,...,x_m\}$. Thus, the final encrypted vector sent to the server is of the following form: $= \{v_1,v_2,...,v_N\}$ where:

$$V_i = \begin{cases} r_i & \text{if } i \notin X \\ [[r_i]] & \text{if } i \in X \end{cases} \quad (5)$$

Because the plaintext is indistinguishable from cipher text, the server has no way to know which of the numbers are encrypted and which are not.

2. The server applies the decryption function to each element in the vector which it receives from the client. By doing this, all non-encrypted random numbers are converted to new meaningless random numbers (unknown to the client) and encrypted random numbers are converted to the random numbers whose position corresponds to the index which the client wishes to receive a document from as follows:

$$]]v_i[[= \begin{cases}]]r_i[[& \text{if } i \notin X \\ r_i & \text{if } i \in X \end{cases} \quad (6)$$

The server then XOR's the decrypted vector with the document URLs in its index term-by-term to produce the final vector $Y = \{y_1,y_2,...,y_N\}$ where each y_i is given by:

$$y_i = d_i \oplus]]V_i[[\quad (7)$$

This vector is sent back to the client.

3. The client, on obtaining Y, XOR's each term with the corresponding random number at the same index in R. If the position corresponds to an index in X, the result will be the original URL but if not, the result will be meaningless. The client does not have the private key and hence cannot use the decryption function.

For each y_i in Y,

if $i \in X$,

$$y_i \oplus r_i = (d_i \oplus]]v_i[[) \oplus r_i$$
$$= (d_i \oplus r_i) \oplus r_i$$
$$= d_i$$

Which gives the original document URL

if $i \notin X$,

$$y_i \oplus r_i = (d_i \oplus]]v_i[[) \oplus r_i$$
$$= (d_i \oplus]]v_i[[) \oplus r_i$$

This gives a meaningless result

Thus, at the end of this protocol, the client obtains every document URL which he/she requested. The server does not know what the query was or what the results were. Thus, user privacy is preserved. At the same time, the client cannot obtain any information which does not pertain to his/her query, thus ensuring that server privacy is also preserved.

4 Proposed Algorithm

In this model, we assume the following scenario: The server is honest but curious. The client wishes to query the server

which maintains an index of web documents based on certain keywords. The web documents are public domain and known to the server. The client however, wishes to maintain privacy (user privacy) by not allowing the server to know what his/her query is or what results he/she receives. In other words, we are specifically interested in client privacy. The server is expected to return relevance ranked results without knowledge of the client query or which results it is returning to the client.

We assume that the set of keywords is constant and denoted by $T = \{t_1, t_2, \ldots, t_m\}$. Assume that the server has indexed a set of documents $D = \{d_1, d_2, \ldots, d_n\}$ and each document has been assigned a *tfidf* vector of the form described in the vector space model section namely:

$$\vec{d} = \{w_{t1\ d}, w_{t2\ d}, \ldots, w_{tn\ d}\}$$

We denote the Paillier encryption of a plaintext a by $[[a]]$. Suppose the client makes a query consisting of keywords $K = \{t_{k1}, t_{k2}, \ldots, t_{k|Q|}\}$. For clarity, we have assumed here that the query will contain only keywords from the term set T. This is a reasonable assumption because even if the client query contains a keyword not in T, it will not contribute to the *tfidf* ranking because it is known that no document contains that keyword.

We now proceed to describe the Steps in our proposed Algorithm:

Algorithm 6.1: OBLIVIOUSRETRIEVAL

1. The client first generates the public and private keys for the Paillier encryption and informs the server of the public key to allow the server to encrypt plaintext via some secret sharing protocol. The client does not reveal the private key.
2. This query is converted to a vector in m dimensional space similar to the document vector representations. However, a problem in converting the query to a *tfidf* score is that, while the *tf* portion is easy enough to calculate, there is no way to calculate the *idf* score without the help of the server. Generally, in normal search engine systems, a query would be sent to the server in raw form and converted into *tfidf* by the server for whom *idf* knowledge is also available. However, this is not possible in this case as it is not desired that the server gain knowledge about the query. A possible alternative is to store the *idf* scores of all terms at the client side itself. However, this does not scale or update well because it forces the client to maintain possibly very long *idf* score lists of terms. Therefore, we compute only the *tf* score of the keywords while converting the query to m space. We base this on the intuition that the user is most likely to include terms which he/she believes are relevant to his/her search, and not terms which would be dampened by their *idf* scores such as stop words. Therefore, we believe that the *idf* portion is not as relevant and hence we ignore it. Thus, $K \rightarrow \vec{Q} = \{w_{t_i}, Q\} \in R^m$ where:

$$w_{t_i} Q = \begin{cases} 1 + \log \mathrm{tf}(t_i) & \text{if } t_i \in K \\ 0 & \text{otherwise} \end{cases}$$

3. Once the client generates the query vector $\vec{Q} = \{w_{ti,Q}\}$, he/she proceeds to encrypt each term separately using the Paillier cryptosystem. Thus, he/she arrives at the encrypted vector $[[\vec{Q}]]$ given by:
$$[[\vec{Q}]] = \{[[w_{t_i}, Q]], [[w_{t_i}, Q]], \ldots, [[w_{t_m}, Q]]\}$$
4. The client sends the encrypted query vector $[[\vec{Q}]]$ to the server. The server does not have the private key and hence cannot decrypt the query vector. The server then proceeds to calculate the correlation coefficient of the query vector with every web document it has indexed (the server index possibly consists of a mapping of web document vectors to their corresponding URLs). In order to achieve this we analyze Pearson's Correlation Coefficient equation and separate out the various terms. For an indexed document d, the equation to calculate $r(\vec{Q}, \vec{d})$ (note we are considering an unencrypted query vector at the moment) can be divided into the following five elements:
   a) $m\sum_{t \in T} w_{t,Q} w_{t,d}$
   b) $\sum_{t \in T} w_{t,Q}$
   c) $\sum_{t \in T} w_{t,d}$
   d) $m\sum_{t \in T} w_{t,Q}^2$
   e) $m\sum_{t \in T} w_{t,d}^2$

Elements (b) and (d) can be calculated by the client alone while elements (c) and (e) can be calculated by the server alone. Note, elements (c) and (e) are independent of the query vector and thus can be calculated before hand for each document to increase efficiency. The only element that remains to be calculated is (a). This can be easily calculated using the adapted homomorphic property of the Paillier system. For a particular document d, the server then performs the following calculation:

$$\prod_{i=1}^{m} [[w_{t_i Q}]]^{w_{t_i}, d} = \sum_{i=1}^{m} [[w_{t_i Q} w_{t_i}, d]] = [[\sum_{i=1}^{m} w_{t_i Q} w_{t_i}, d]]$$

Multiplying the result by m yields the encrypted version of element (a) (Multiplication by a constant is possible in the Paillier cryptosystem). The server achieves this by encrypting each element of the document separately using its public key to yield

$$[[\vec{d}]]=\{[[w_{t_1,d}]],\left[\left[w_{t_2,d}\right]\right],\dots,[[w_{t_n,d}]]$$

By multiplying all the elements in the encrypted query vector, the server obtains the encrypted value of element (b) (due to the additive homomorphism) as:

$$\prod_{i=0}^{m}[[w_{t_i,Q}]] = [[\sum_{i=0}^{m} w_{t_i,Q}]] \quad (9)$$

Thus, at the end of the process, for each document d, the server returns the following elements back to the client:

1. $[[\alpha_d\ (m\sum_{t\in T}\ w_{t,Q}\ w_{t,d} - \sum_{t\in T}\ w_{t,Q} \sum_{t\in T} w_{t,d})]]$
2. $\propto d\sqrt{(m\sum_{t\in T} w_{t,d}^2 - (\sum_{t\in T} w_{t,d})^2)}$

along with an identifier to allow the client to request that document if he/she wishes to. α_d is a random integer unique to each document which is used to reduce the amount of information the client obtains. This is explained further in the analysis section. The identifier can be the actual identifier of the document (i.e the plaintext URL) or an encrypted one (using a server owned private key for decryption).

5. The client receives the above two elements (and a document identifier) for each document indexed. The client decrypts them using his/her private key. The random constant α is eliminated via division. Thus, based on the values he/she obtains, he/she can compute the final correlation coefficient for each document as:

$$r(\vec{Q},\vec{d})=\frac{m\sum_{t\in T} w_{tQ} w_{td} - W_Q W_d}{\sqrt{[m\sum_{t\in T} w_{t,Q}^2 - W_Q^2][m\sum_{t\in T} w_{t,d}^2 - W_d^2]}}$$

Where $W_Q = \sum_{t\in T}\ w_{t,Q}$ and $W_d = \sum_{t\in T}\ w_{t,d}$.

6. Therefore, the client obtains a set of scores $\{r(\vec{Q},\vec{d}) \mid d \in D\}$. Based on these scores and their corresponding document identifiers, the client can choose the top n scores and request the corresponding documents using the oblivious transfer protocol described in the previous section.

5 Privacy Preserving Indexing

Our model again uses the algorithm described in the previous section, but this time the indexes are built after statistical micro-aggregation of the confidential web documents has taken place. Micro-aggregation is a statistical disclosure control technique described by Domingo-Ferrer et al in [70]. By clustering the web documents into groups of at least k and taking the centroid of their document vectors as the representative of the set, we can guarantee at least k *–anonymity* to the owner of those documents. This clustering takes place before any query is made on the documents, therefore, we can define a distance function and aggregation function independent of Pearson's correlation coefficient if required. For example, in Abril et al [29], a novel web vector distance based on semantics is defined along with a corresponding aggregation function.

6.5.1. Document Vector Distance

For this work, we define the document vector distance as the similarity function as obtained from the Pearson correlation coefficient. We have explained this distance in the previous sections.

6.5.2. Document Aggregation

Aggregation involves 'aggregating' all the documents in the cluster and creating a representative vector of that cluster. We adopt a simplistic approach by taking the mean of the corresponding terms of the document vectors in the cluster. Therefore, for a cluster C containing a set of documents, the aggregate vector is given by $M_c = \{\mu_1, \mu_2, \dots, \mu_m\}$ where

$$\mu_i = \frac{\sum_{d\in C} w_{t_i,d}}{|C|}$$

6.6. Analysis of the Proposed Algorithm

We analyze both the client and server privacy obtained using our algorithm:

1. **Client Privacy:** The client sends the server his/her query in an encrypted form. Because the server does not have the private key, it cannot decrypt the query. Similarly, the server also cannot infer the scores of each document and thereby infer the requested documents. The portion of the score equation sent back is encrypted, thus preventing the server from obtaining it. Thus, client privacy is at least as secure as the Paillier system.

2. **Server Privacy:** Server privacy requires that the client should not learn anything about the documents indexed by the server apart from those that the client

has queried for. This implies that the client should not be able to reconstruct the document vectors from the information it receives from the server. Suppose:

$$m\sum_{t\in T} w_{t,Q}\, w_{t,d} - \sum_{t\in T} w_{t,Q} \sum_{t\in T} w_{t,d} = A$$

$$\sqrt{\sum_{t\in T} w_{t,d}^2 - (\sum_{t\in T} w_{t,d})^2} = B$$

Where $A, B, w_{t,Q}, w_{t,d} \in Z$. Based on this notation, the values obtained from the server by the client are: $A\alpha$ and $B\alpha$.

We consider the following argument: Breaking server privacy is equivalent to solving the following non linear Diophantine equation for m variables obtained by eliminating the random factor α via division from the following two elements:

$$m\sum_{i=0}^{m} c_i x_i - \sum_{i=0}^{m} c_i \sum_{i=0}^{m} x_i = A\alpha \qquad (10)$$

$$\sqrt{m\sum_{i=0}^{m} x_i^2 - (\sum_{i=0}^{m} x_i)^2} = B\alpha \qquad (11)$$

To yield:

$$m\sum_{i=0}^{m} c_i x_i - \sum_{i=0}^{m} c_i \sum_{i=0}^{m} x_i = G\sqrt{m\sum_{i=0}^{m} x_i^2 - (\sum_{i=0}^{m} x_i)^2} \qquad (12)$$

We note that A and B are not known to the client due to the randomizing factor α. The client can only calculate $G = A/B$. We can obtain one infinite set of solutions by setting all x_is to be equal to say y. Substituting this in equation 12 yields the identically true expression 0 = 0. Thus, there are infinite number of solutions to the values of $\{x_i\}$.

6 Implementation and Results

We implemented the algorithm described above in java using the step3 java package. We used an Intel i3 processor with a clock frequency of 2.4GHz. We created a sample set of documents and generated their *tfidf* vectors. We varied both the number of documents and the size of the term set. We have considered only the server side logic in our time calculations. Network time is ignored as it will be dependent on the efficiency of the network and not on our protocol.

The time taken to score a single document via our protocol depends, as expected on the size of the term set (depicted in Graph 6.1). We ran our protocol 5 times and averaged out the time taken. A single document took on an average 3.67ms to score with a term set of size 10. For a term set of size 200, the time taken to score increased significantly to 12.17ms. Varying the number of documents in the document set led to approximately a linear relationship between the time taken and the number of documents in the set.

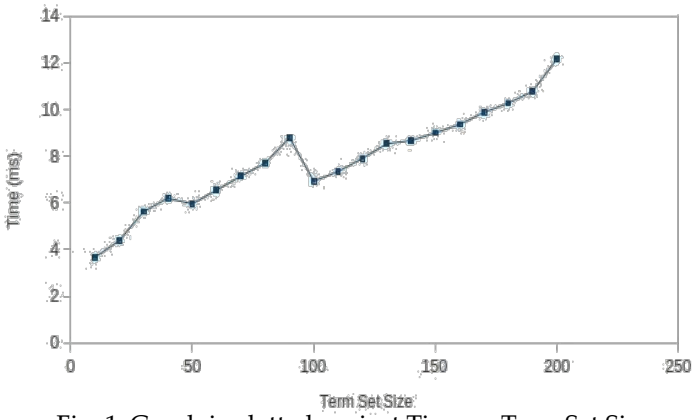

Fig. 1. Graph is plotted against Time vs Term Set Size

7 Conclusions

In this paper, we have developed a privacy preserving protocol for retrieving information based on Pearson's correlation coefficient of similarity. We have represented web documents according to the vector space model as tfidfvectors and used them to ensure privacy of the client while querying the server. The server is kept unaware of the client's query contents as well as the result set which is returned. We have also briefly discussed a technique for preserving the privacy of web documents being indexed by the server (if such a need arises) via aggregation. This protocol was implemented and its time efficiency discussed.

For future work, we wish to explore other techniques of privacy preserving document retrieval based on other similarity measures. Pearson's correlation coefficient used here, or the standard cosine similarity measure are effective but we also wish to explore other graph based algorithms such as pagerank[11] or HITS[9] for developing privacy preserving protocols.

References

[1] D. Abril, G. Navarro-Arribas, and V. Torra. Towards privacy preserving information retrieval through semantic microaggregation. 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2010.

[2] R. Baeza-Yates, B. Ribeiro-Neto, et al. Modern information retrieval, volume 463. ACM press New York, 1999.

[3] Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In Proceedings of the 17th international conference on Computational linguistics-Volume 1, pages 79{85. Association for Computational Linguistics, 1998.

[4] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In Theory of cryptography, pages 325{341. Springer, 2005.

[5] S. Buttcher, C. Clarke, and G. V. Cormack Information retrieval: Implementing and evaluating search engines. The MIT Press, 2010.

[6] P. Castells, M. Fernandez, and D. Vallet. An adaptation of the vector-space model for ontology-based information retrieval. Knowledge and Data Engineering, IEEE Transactions on, 19(2):261{272, 2007.

[7] J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. Knowledge and Data Engineering, IEEE Transactions on, 14(1):189{201, 2002.

[8] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. Information Theory, IEEE Transactions on, 31(4):469{472, 1985.

[9] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM), 46(5):604{632, 1999.

[10] M. Naor and B. Pinkas. E cient oblivious transfer protocols. In Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, pages 448{457. Society for Industrial and Applied Mathematics, 2001.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[12] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Advances in cryptology - EUROCRYPT'99, pages 223{238. Springer, 1999.

[13] M. O. Rabin. How to exchange secrets with oblivious transfer. IACR Cryptology ePrint Archive, 2005:187, 2005.

[14] P. R. Sabbu, U. Ganugula, S. Kannan, and B. Bezawada. An oblivious image retrieval protocol. 2011 Workshops of International Conference on Advanced Information Networking and Applications, 2011.

[15] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5):513{523, 1988.

[16] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. Communications of the ACM, 18(11):613{620, 1975.