http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (2): 347-352, 2006
ISSN 1812-5638

Designing Efficient Techniques for Searching Encrypted Data in Untrusted Infrastructure

[1]Hasan Al-Sakran and [2]Mohamad Adnan Ali
[1]MIS Department, [2]CS Department, IT College, Yarmouk University, Jordan

Abstract: In this study an efficient cryptography system that enables searching and sending encrypted data without violating privacy was proposed. It was assumed, that this system is working in untrusted environment. The untrusted server can not learn anything about the encrypted data or about the encrypted queries. The efficiency of the proposed system achieved through the use of extendable hashing technique, which enable answering queries in constant time regardless of the data size.

Key words: Cryptography system, encrypted data, Public Key Infrastructure (PKI)

INTRODUCTION

Due to economical, technical and convenience reasons, many companies and users store their data on databases on remote servers. For example, the use of mobile devices becomes popular and users want to be always connected, but there are limitations in the battery life, memory and bandwidth available for these devices. To overcome the limitation of memory available for these devices, users can use servers to store data instead of storing it locally. Thus these databases need to be secured.

To maintain the confidentiality of a data stored on an untrusted server, users may need to store their data in encrypted form. However, searching such encrypted data is not an easy task. To search such data client will have to retrieve the entire database to his local machine before querying it. This solution is inefficient one. The user mostly download the data from the server and decrypt it locally, then perform the searching. This solution wastes bandwidth, needed to download data that the user may be not interested in and the battery life, by unnecessary data processing.

A number of researchers tried to tackle the problem of searching encrypted data. Song *et al.*[1] suggested the using of symmetric key encryption algorithm to encrypt the text file word by word i.e. treating each word as a whole and perform sequential search at the server. This solution involves heavy processing at the server because of sequential search and, due to the required padding, redundant size of the encrypted text. Other researchers[2] discuss the Conjunctive Word Search over Encrypted Data. Previous Information Retrieval schemes[3-5] allow queries to a database where the database learns nothing about records being retrieved. This solution will waste the bandwidth because we download such records in which we are not interested.

In this study we design a cryptography system for searching encrypted data that enables the user to store encrypted data at an untrusted server. The proposed system can be used in many applications such as:

1. Searching encrypted databases, stored on untrusted and uncontrolled third party servers (file server and e-mail server), while preserving privacy and security of data. Securing local sensitive data from users with full privilege to access it (for example system administrator have root access).
2. Securing and searching emails stored on an untrusted server.

In such system data processing is transferred from the client side to the server. The server performs searching with time complexity O(1) using extendable hashing[6]. The cryptography system uses public key infrastructure (PKI) to exchange keys. Given below is the discussion about how users can search encrypted documents on the server before deciding to download the ones he needs.

Research conducted in this field suffers from different problems, which include:

- Performance
- Assumption about the existence of a trusted server
- Dealing with two sides (client and server) only
- And weak cryptography system

Corresponding Author: Hasan Al-Sakran, MIS Department, IT College, Yarmouk University, Jordan
Tel: 962-2-7211111 Fax: 962-2-7274725

Searching encrypted data was discussed for the first time by Song *et al.*[1]. They suggest encrypting text word-by-word using symmetric encryption algorithm. This requires fixed word length; to achieve required padding. The output blocks (encrypted word plus padding) then stored in a file. Upon searching the client side encrypts the query word by word using the same encryption parameters and sends it to the server side. On the server side sequential search has to be performed (block by block).

However, their method suffers from:

- The limitation of the cryptography system. Their study did not discuss how to send encrypted data to another client and enable him to search it, so it does not present technique for exchanging encryption parameters. Note that we can't send the parameters using public key because the other side can distribute the parameters to a third party, so any one can search and decrypt the data.
- The searching technique is not practical for large databases; each query requires the server to search through the whole database linearly. That means accessing the entire database needs O(n) operations. And a very large number of disk accesses are required.
- The use of static word length, which require padding. Because the system presented encrypt the text word by word so it needs fixed word length. This not the case in natural languages so padding technique is required. The fixed length must be the longest word length, which leads to waste of space.

While Song *et al.*[1] does not involve the cognitive word search, Golle *et al.*[2] presented a technique that enables users to give the server a capability to identify the documents that contain certain keywords like FROM: Bob, SUBJECT: Urgent, etc.

Golle *et al.*[2] suggest encrypting text using symmetric key encryption and generating a keyword vector. It defines a model for conjunctive keyword search of encrypted text. The server performs sequential search using keyword vector, which is not practical for large databases.

While above mentioned research[1-3,7] deals with unstructured or semi-structured data, Brinkman *et al.*[8] presents an algorithm that is suitable for databases in XML format. It conducts comparison between linear and tree search and shows how to use the structure of XML files to process query efficiently, but the problem of exchange of documents, for example in case of disconnected XML databases, has not been addressed.

An important schema for searching encrypted data has been described[9]. Goh[9] defined efficient secure searching technique over encrypted data. This method determines whether a document contains a word in O(1) time complexity. A secret key can generate the trapdoor and nothing can be learned from the index without the trapdoor. The schema uses secure indexing technique and represents a secure indexing model named semantic security against adaptive chosen keyword attack (IND-CKA). It uses bloom filter to produce the trapdoor to enable the server to answer a query. However, this method does not support multiple keywords search.

Damiani *et al.*[10] discussed the impact of the untrusted database architecture on query translation and optimization and the requirements for implementation of encrypted database in a web based environment, which include compromise between security and performance.

Secret sharing scheme was used[11,12] to make query highly efficient and allows to handle more complex queries. Using of public key encryption has been discussed, where the researchers[13-15] developed Searchable Public Key Encryption. This schema require sequential search at the server. Implementation of the Private Information Retrieval (PIR) schemes[3,5,14-16] can potentially be used to solve searching secure data problem. PIR schemes allow to query a database where the database learns nothing about records being retrieved. This solution will result in wasting the bandwidth because we download records that we are not interested in and therefore increase the communication cost.

EXTENDABLE HASHING

Hashing is the most attractive searching technique because of its time complexity of O(1) , but the problem of static hashing is that as the size of the data grows more collisions occurs because the hashing function maps more than an item to the same bucket . So when the database becomes very large there would be a large number of collisions. Dynamic hashing, introduced in 70's, allowed the hashing function to grow as the data grows with little overhead.

Extendable dynamic hashing was introduced in 1979[13]. In this technique, when bucket becomes large it splits into two buckets. The hashing function used should produce uniform distribution of data in bucket with respect to their binary representations. The number of binary digits used is dynamic and increased when more collisions occurs in contrast to static hashing. An example on Fig. 1 explains how the extendable hashing works.

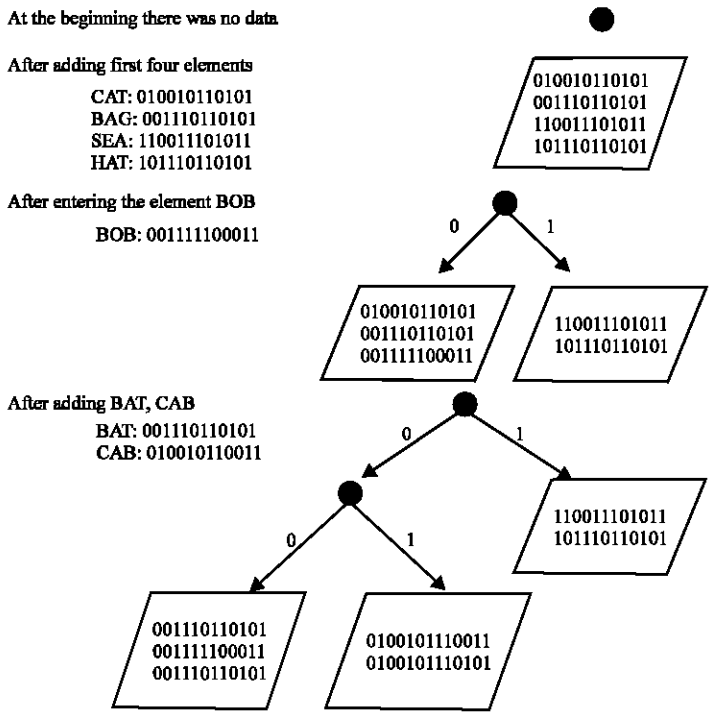

Fig. 1: Example of extendable hashing

Suppose we have an extendable hashing function that depends on the binary representation of the ASCII code for the input. And the page or bucket size is 4 elements. At the beginning we have no data, so we have just null pointer. But after entering data for example: CAT, BAG, SEA, HAT. The page now is full and will not accept a new item, if there is a new input (for example: BOB). The page will split into two pages (buckets). The data will split according to its first bit in its binary representations.

MATERIALS AND METHODS

Nowadays there is an emergent need to encrypt data exchanged between people for privacy reasons. Data could be emails or documents. Although there have been a lot of advances in network technologies some of them can not be accessed by all users (e.g. mobile devices, hand held, etc.). So users can not download all files or emails to find out their content. Previous researches, or at least the accessible ones, ignore multi clients' environments, or are built on assumption of using a trusted server which is not the case all of the time.

This study was built on assumption untrusted environment and describes a cryptography system that allows a user with limited resources searching and exchanging encrypted data without violating privacy.

The major objective of the proposed system was to perform certain computation over the encrypted data in order to find whether a document (or email) contains a specific word or words without learning anything else about the original text. This system supports hidden queries (does not reveal the actual search word), controlled searching (can not generate the original words without the secret keys) and query isolation (the server learns nothing about the documents than the search results).

To speed up the searching functionality a pre-computed index should be build. This index should list all documents (emails), which contain key words, within the encrypted database. The query could be simple or complex and will use the index generated by the sender which based on extendable hashing.

In this study two schemas are presented. First one is suitable when there is a need of independent indexes (e.g., databases), the second schema is suitable for applications that require one index (e.g., e-mail) which means having one index for all e-mails of each user.

First schema: The proposed cryptographic system consists of two main parts: client and server. The client side performs the encryption and builds the index that is sent to server which will be assumed untrusted side.

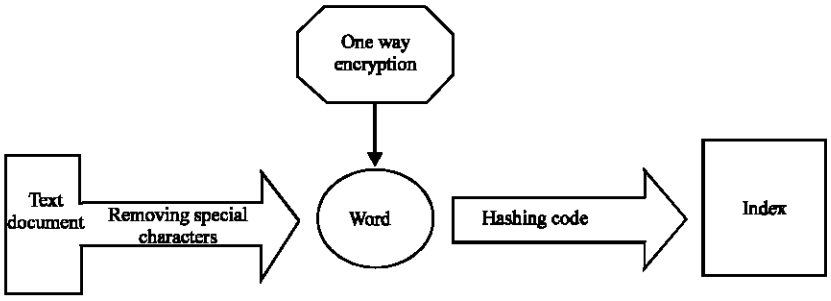

Fig. 2: Algorithm for encryption of words of the document

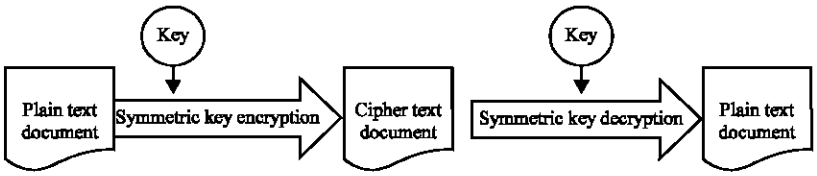

Fig. 3: Encryption of a document using symmetric key

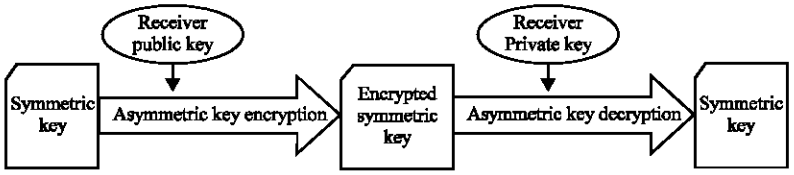

Fig. 4: Encrypting the symmetric key

The server will not be able to get any information from the encrypted text or from the index.

If the client wants to search for files that contain a specific word(s) at first it encrypt the query and send it to the server. The server can't understand anything from the query because it is in encrypted form. It just searches the index for the query and sends the result back to the client who will decide to download the file for local processing or not. In the two following sections all events happening on client and server sides will be discussed.

Client side: Most of the major activities are taking place on the client side of the proposed system. This section presents the discussion about how the client side encrypts the documents, how to generate the index, how to connect to the server, what to send and how and how search take place.

On the client side, when a user wants to send an encrypted message to the server the system first examine the document word by word and perform an encryption using the algorithm shown in Fig. 2. The cipher text generated by the encryption algorithm is used to build the index. After generating the index the whole file is encrypted using symmetric encryption algorithm. The key (used in the symmetric key encryption) is encrypted using asymmetric encryption algorithm.

The processing of the document on the client side proceeds as follows. The document is processed word by word, punctuation marks and other invisible characters are removed because they will affect the search. Stop words do not need to be removed because their presence will make the search more exact while searching for a particular pattern. And more important, the program must be more general, abstract from specifics of certain language because it is undesirable or may be impossible to generate a list of all stop words for each language. Another important reason not to remove the stop words is that the document may be written in Para language.

After removing the punctuation marks, the document is encrypted word by word using one way encryption algorithm (keyed hashing algorithm). In one way encryption no one can regenerate the plain text from the cipher text again. After encrypting all the words we build the index using extendable hashing. Now the file can be encrypted using symmetric key encryption algorithm. The receiver can decrypt the document using the same key, if he wants to download and read the document (Fig. 3).

The key used in the symmetric key algorithm and the key used in keyed hashing algorithm are encrypted using asymmetric key encryption algorithm. This will prevent the possibility of someone accessing the key, decrypting document cipher text and generating the plain text document; or searching the data using hashing key. The key is encrypted using the receiver's public key (Fig. 4).

After generating the index using extendable hashing the encrypted document and the encrypted index the client sends these objects to the untrusted server using a TCP connection over a network or internet work.

Server side: When the server is connected to a client using a TCP connection it can receive files, answer hidden quires and send requested files. Let's start from the end of previous section. The server now receives the three objects which are:

- The document encrypted using symmetric key encryption algorithm.
- The symmetric key and the hashing key encrypted using asymmetric key encryption algorithm.
- The index which was generated by the client and encrypted using one way encryption algorithm.

The server stores these files and allows only the receiver to access them by default, but as was mentioned above the server assumed to be untrusted and can give the access to any one. This is not a problem because no one can decrypt the files without the receiver's private key and it is the responsibility of the receiver to protect it.

To conduct search the receiver should:

- Establish a connection with the server.
- Download the required encrypted hashing key and decrypt it.
- Encrypt the word using one way encryption algorithm (using the hashing key),
- Send the encrypted query.

Upon receiving the encrypted query the server conducts search using extendable hashing and returns the encrypted result to the client side. The result will contain the file name and the positions of the word in that file. If the user is interested in a file he can download the encrypted file and the encrypted symmetric key. Upon receiving the encrypted symmetric key (which was encrypted using asymmetric key encryption algorithm) the client decrypts the encrypted key using his own private key locally.

After the receiver has obtained the key he decrypts the document using the received key to generate the plain text again.

Second schema: Some applications require building a single index. For example on e-mail server it is not practical to build an index for every e-mail when a user receives emails form different sources and wants to search these e-mails. The previous scheme requires a user to download the key for every index and send the query encrypted with the key. Instead we need a technique to build one index for all e-mails.

The proposed technique works as follows. Suppose Bob wants to send a message to Sam. First Bob examines the document and produces a hash code for every word. Note that a hash algorithm with standard parameters, i.e. without a key, is used. This step is similar to the step in the first schema when the hash code splits into two parts. The first part will be used for indexing in the extendable hashing. It is sufficient to let the first part to be a size of 40 bits; this enables the extendable hashing method to build tera-page. For the second part a mask known Sam is applied. Bob also encrypts each word using symmetric key and associates it with the hash word for each word.

The whole document is encrypted using symmetric key and the key is encrypted using Sam's public key like the first schema described above. When the server receives the encrypted document and the sequence of hash codes, it adds the hash codes block to Sam's index using the first 40 bits. Then server saves the hash code and encrypted word in the page.

Now when Sam wants to search his mail box, he uses a hashing algorithm for the query and split the hash code into two parts. The first part is 40 bits long and the second consist of the remaining bits. A master mask is applied to the second part and then Sam sends the encrypted query to the server.

After receiving the encrypted query the server uses the first 40 bits (note that 40 bits is the maximum, the server usually uses less than this number) to traverse the tree and allocate the required page. The server then uses the remaining of the 40 bits and the second part of the query to search within the page. If a possible match is found, the server sends the encrypted word and the encrypted symmetric key to the client, so that the client can decrypt the word.

CONCLUSIONS

In this study an efficient cryptography system that enables users to search encrypted data in untrusted infrastructure by using PKI has been developed. Two schemas have been presented. The first schema was suitable for applications that require a separate index for each portion of data like document-databases; the second schema may suitable for applications that require one index for different documents like e-mail messages. An

efficient secure index based on extendible hashing approach also has been developed. The proposed system has other advantages such as minimizing communication overhead, minimizing computation on both server and client sides.

REFERENCES

1. Song, D., D. Wagner and A. Perrig, 2000. Practical techniques for searches on encrypted data. In Proceeding of the 2000 IEEE Symposium on Security and Privacy, pp: 44-55.
2. Golle, P., J. Staddon and B. WatersSecure, 2004. Conjunctive Keyword Search over Encrypted Data. Lecture Notes in Computer Science, Springer-Verlag GmbH, pp: 31-45.
3. Chor, B., E. Kushilevitz, O. Goldreich and M. Sudan, 1998. Private information retrieval. J. ACM., 45: 965-981.
4. Gertner, Y., Y. Ishai and E. Kushilevitz, 1998. Protecting data privacy in private information retrieval schemes. In Proceeding of the 30th Annual ACM Symposium on Theory of Computing. ACM., pp: 151-160.
5. Kushilevitz, E. and R. Ostrovsky, 1997. Replication is not needed: Single database, private information retrieval. In Proceeding of the 38th Annual IEEE\Symposium on the Foundations of Computer Science, pp: 364-373.
6. Fagin, R., 1979. Normal forms and relational database operators. In ACM SIGMOD Int'l Conf. on the Management of Data.
7. Chor, B., N. Gilboa and M. Naor, 1998. Private information retrieval by keywords. Technical Report TR CS0917. Department of Computer Science, Technion.
8. Brinkman, L., D.J. Feng, P.H. Hartel and W. Jonker, 2004. Efficient tree search in R. Encrypted Data. 2nd Int'l Workshop on Security in Information Systems, pp: 126-135.
9. Goh, E., 2003. Building secure indexes for searching efficiently on encrypted compressed data. http://eprint.iacr.org/2003/216/.
10. Damiani, E., S. De Capitani di Vimercati and M. Finetti, 2003. Implementation of a storage mechanism for untrusted DBMSs. 2nd Int'l IEEE Security in Storage Workshop, pp: 38-46.
11. Waters, B., D. Balfanz, G. Durfee and D. Smetters, 2004. Building an Encrypted and Searchable Audit Log. Lecture Notes in Computer Science, Springer-Verlag GmbH., pp: 31-45.
12. Brinkman, R., J.M. Doumen, P. Hartel and W. Jonker, 2004. Using secret sharing for searching in encrypted data. In Proceeding of the Secure Data Management Workshop, Toronto, Canada.
13. Boneh, D., G.D. Crescenzo, R. Ostrovsky and G. Persiano, 2004. Public-key Encryption with Keyword Search. In Cachin, C., (Ed.), Proceeding of Eurocrypt 2004, LNCS. Springer-Verlag, pp: 506-522.
14. Iliev, A. and S. Smith, 2003. Privacy-enhanced credential services. 2nd Annual PKI Workshop. http://www.cs.dartmouth.edu/ sws/papers/ilsm03.pdf.
15. Chor, B., O. Goldreich, E. Kushilevitz and M. Sudan, 1998. Private information retrieval. J. ACM., 45: 965-981.
16. Di Crescenzo, G., Y. Ishai and R. Ostrovsky, 1998. Universal service-providers for database private information retrieval. In Proceeding of the 17th Annual. ACM Symposium on Principles of Distributed Computing, pp: 91-100.