Improvised Privacy-Preserving Multi-keyword Top-*k* Similarity Search Over Encrypted Data

[1]Pooja Kuber Patil, [2]Rupali A. Mangrule

[1]ME Student, [2]Assistant Professor
MIT College of Engineering

***Abstract*: Cloud computing enables individuals and organizations to run large, scalable computing to support large data applications in the domain, such as healthcare and scientific research. As a result, data owners are involved in outsourcing their data on cloud servers for data availability. However, data sets such as client files and records in electronic documents often contain sensitive information, which raises concerns about personal information, if the document is published or shared with some unreliable third party in the cloud. The most widely used and practical technique for keeping personal information is encrypting data before outsourcing to a cloud server. This reduces the utility of data and enables traditional data analysis operators. Fetching documents on k using outdated keywords. In this article, we will investigate the top-k search queries for multiple keywords for large data encodings to protect privacy, and attempt to identify effective and secure solutions for this problem. Particularly for privacy concerns about query data, we've created a special tree-like index structure and designed a randomized search algorithm that makes even the same search query create routes. Different views on the index can also be maintained. Under the stronger privacy. For query performance improvement, we have proposed a multi-layered top-k search scheme based on the concept of a partition, which contains a cluster of indexes based on the tree created for all documents. Finally, we combine these methods together as a powerful and secure way of identifying similar top-k searches. Our experimental results in real-life data sets show that the guidelines we offer can be improved. The ability to protect privacy, scalability, and efficiency in query processing is greatly enhanced by modern methods.**

***Keywords*: Cloud computing, privacy preserving, data encryption, multi-keyword top-k search, trapdoor, cosine similarity, lucene indexing**

Introduction

Cloud computing has become a disruptive trend in the IT and community industries involving research. Recent research, such as scalability and pay as you go, has helped consumers. Clouds can buy powerful computing resources based on real needs. Cloud users no longer need to worry about consuming resources on the computer system and complexity in managing the hardware platform. [1] [2] Currently, many companies and individuals from large data applications have outsourced information and deployed their services to cloud servers for easy data management, data mining, and query processing. But when companies and individuals benefit from these advantages in cloud computing, they need to take into account the privacy of the information they are hired. Because many in-app sets often contain sensitive information, such as email, electronic health records, and financial transaction records, when the data owner outsources such important information to a cloud server, it is considered partially reliable. These data are easily accessible and analyzed. Because analyzing these data sets may provide insights into key social issues (such as e-research, health, medical, and government services), data owners need to be effectively secure with the cloud. Data encryption is widely used for storing personal information in a data-sharing scenario, which means mathematical calculations and algorithms that convert plaintext to cipher text, which is unreadable for unauthorized people. [3], [4], and [5] are used to encrypt data before outsourcing to a cloud server.

However, using these approaches to encrypt data often entails a tremendous expense in the data utility, which makes traditional data processing designed for plaintext data not work well with encrypted data. Keyword-based searches are operators. The widespread data in databases and applications, large amounts of data retrieval, and traditional processing methods cannot be directly applied to encrypted data. So how to process such queries in encrypted data and at the same time guarantee data privacy becomes a hot topic of research. For example, [6], [7], [8] deal with single keyword searches and results [9], [10], [11], [12], [13] supports multi-keyword Boolean search. However, single keyword search is not smart enough to support advanced search and bold search is unrealistic as it causes high cost of communication. Recent assignments such as [14], [15], [16] focus on multi-word search, which is the concept of paying more in the cloud-based paradigm. But most of these methods fail to meet high search performance and robust data integrity at the same time, especially when applied to large data encoders, resulting in scalable capabilities. We focus on multi-keyword searches as search terms, which have database operations and that are extremely popular in major applications, where we can return k documents based on highest relevance rating. For multi-word search support, we recommend vectors that represent documents and search terms as vectors. In order to support top-k queries, the relevance scores between documents and queries should be calculated, so TF = IDF (frequency, frequency, words). Inverse) is a weighting rule for calculating relevant scores for order purposes.

In addition to improving search performance for a better user experience, we also offer group-based bulk segmentation (GMTS) search schemes that use the partition and support for similar searches. Encrypt In this project, the owner of the information will divide the keywords in the

dictionary. (Assuming the dictionary contains all the keywords that can be extracted from the whole document) into multiple groups, and the indexes can be searched for each group on the other side. In order to better control the size of the index, , [18] into our chart, where the index of the keyword group holds only the topmost documents of the matched keyword (top-k, the keyword document shows the c).

We also offer a random exploration algorithm (RTRA) to increase data security, whereby the data owner creates a binary tree that is searchable and determinable. Provide a random switch to each node, so the data user can assign a random key to each query. As a result, data users can change their search results and access the search query path using a different key, which will store high-precision search queries. Finally, we combine GMTS and RTRA into a powerful and secure solution for the problems we offer.

Proposed Methodology

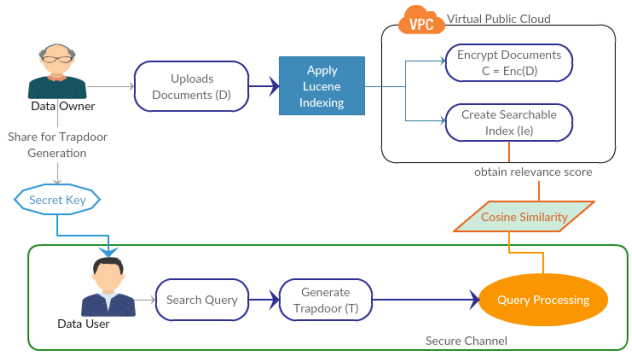

Figure 1 Proposed Architecture

As shown in Figure 1, the system layout we consider following components: data owner, data user, and cloud server. The data owner uploads the D collection to the Cloud server, but this collection may contain sensitive information. To protect the privacy of information, the owner of the data must encrypt D before hiring it to the cloud server. In addition, for cloud servers to efficiently process queries through an encrypted document collection, the owner of the data generates an encrypted search index, such as on the machine.

Finally, the data owner stores both an encrypted C-encrypted collection of files and searchable indexes, such as Ie, to the cloud, and shares the secret key to create and capture confidential documents with the user. Information authorized by the channel. Safe When a user wants to search with a query, he / she creates a trapdoor T for this query first by encrypting the query and then sends a shortcut to the cloud server for query processing. After receiving the T, the cloud server will calculate the relative score between the trapdoor T and the document in the Ie index, and return k the highest-scoring document to the user.
Let D be a set of plaintext documents that the data owner will outsource to the cloud server and Di will display the document in D.

Key Generation: - The data owner generates a key using a random generator. This secret key is used for encryption of the document index.

Encrypted Index Generation: - The document index is prepared using Lucene indexer and applies the cosine similarity score calculator to obtain scores for relevant documents.

TF IDF Score Calculation

$$\text{Score}(Q, Dj) = \sum_{k=0}^{n} \text{TF} * \text{IDF} \quad \text{..... (1)}$$

Cosine Score Calculation

$$\text{Cosine} = \frac{A.B}{|A| + |B| - A * B} \quad \text{..... (2)}$$

Lucent Score Index : Every document has its own score in index. These indices are encrypted using matrix multiplication with the secret key. These encrypted indices are stored with the cloud server.

Document Upload: - The documents are encrypted using Blowfish algorithm. These encrypted documents are uploaded to the cloud server.

Trapdoor Generation: - The data owner shares a secret key sk with the data user to generate a trap door via a secure channel.

The owner of the data has a collection of F data documents to be outsourced to the server in the encrypted C form. To enable search capability on C for effective data utilization, the data owner will first build a search index I using F's Lucene Indexer before outsourcing, and then outsource both the index I and the collection of encrypted documents C to the cloud server.

The work deals with efficient algorithms to assign identifiers (ID) to users in the cloud in such a way that the FILE identifiers are anonymous using a distributed calculation without central authority as the data is encrypted.

Since there are N nodes, this assignment is essentially a permutation of the integers {1 ... .N} with each FILE that is known only by the node to which it is assigned. Our main algorithm is based on a method of anonymously sharing simple data and results in methods for the efficient exchange of complex data.

To search the collection of documents for certain keywords, an authorized user who has an identification and a specific designation acquires a corresponding K through our search control mechanisms.

Upon receiving T from a data user, the server in the cloud is responsible for searching the index I and then returns the corresponding set of encrypted documents. To improve the accuracy of document retrieval, the cloud server must classify the search result according to some classification criteria (for example, coordinate match) and assign anonymous FILE ID [6] to the user in the cloud to Make the

data cloud more secure. In addition, to reduce the cost of communication, the user of the data can send an optional k number together with the trap door T, so that the server in the cloud only sends the top-k documents that are most relevant to the query of search.

Finally, the access control mechanism is used to manage the decryption capabilities provided to users and the data collection can be updated in terms of inserting new documents, updating existing ones and deleting existing documents.

Encryption Algorithm

Blowfish is a popular security algorithm that was developed by Bruce Schneier in the advent of the year 1994. The algorithm works on the same line as DES and consumes block blocks with blocks of a size of 64 bits. Blowfish became quite popular after its arrival, just because Bruce Schneier [1] himself is one of the most famous experts in cryptology and, above all, the algorithm is not patented, open source is free and available for its use and modifications. Blowfish is a 64-bit block cipher with a variable length key. Define 2 different boxes: S boxes, one box P and four boxes S [3].

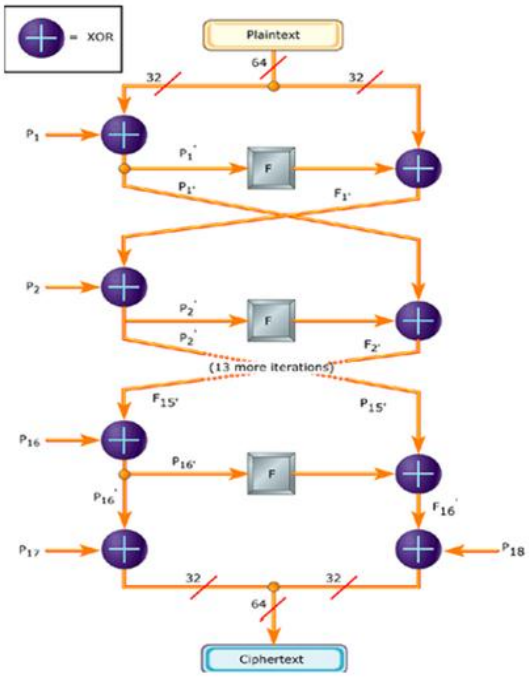

Figure 2 Fiestal Network

Taking into account that P box P is a one-dimensional field with 18 values of 32 bits. The tables contain variable values; those can be implemented in the code or generated during each initialization. The frames S S1, S2, S3 and S4 each contain 256 32-bit values. Blowfish is a symmetric encryption algorithm, which means that it uses the same secret key to encode and decrypt messages. Blowfish is also a block cipher [5], which means that it divides the message into blocks of fixed length during encryption and decryption. The block length for Blowfish is 64 bits; Messages that do not have a size of multiples of eight bytes must be filled.

Blowfish consists of two parts: key expansion and data encryption. During the expansion stage of the key, the key entered becomes several matrices of sub-keys in a total of 4168 bytes. There is the matrix P, which is eighteen boxes of 32 bits, and the boxes S, which are four matrices of 32 bits with 256 entries each. After initialization of the string, the first 32 bits of the key are XORed with P1 (the first 32-bit box in the matrix P). The second 32 bits of the key are XORed with P2, and so on, until all 448 or fewer key bits have been XORed. Cycle through the key bits returning to the beginning of the key, until the entire set P has been processed. XORed with the key. Encrypt the zero string with the Blowfish algorithm, using the modified P matrix above, to getablock 64 bits. Replace P1 with the first 32 output bits, and P2 with the second 32 output bits (from the 64-bit block). Use the 64-bit output as input again in the Blowfish encryption, to get a new block of 64 bits. Replace the following values in the matrix P with the block. Repeat for all the values in the matrix P and all the squares S in order.

Encrypt the whole zero chain using the Blowfish algorithm [12], using the modified P matrix above, to obtain a block of 64 bits. Replace P1 with the first 32 output bits, and P2 with the second 32 output bits (from the 64-bit block). Use the 64-bit output as input again in the Blowfish encryption, to get a new block of 64 bits. Replace the following values in the matrix P with the block. Repeat for all the values in the matrix P and all the squares S in order.

Conclusion

In this document, we focus on improving the performance and security of multiple top-k similarities finding on encrypted data. In order to improve search performance, we will store top-ck documents of each word segment when indexing. Previous work [1] focused on providing personal information to the data in the cloud, using multi-word search in encrypted cloud data using a similar measure of efficiency. Coincidence of coordinates Previous work [4] also presents the basic concepts of using internal product safety calculations.

References

[1] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya, "Ensuring security and privacy preservation for cloud data services," *ACM Computing Surveys*, 2016.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM, 2006, pp. 79–88.

[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in

Advances in Cryptology- Eurocrypt 2004. Springer, 2004, pp. 506–522.

[5] Z. Ying, H. Li, J. Ma, J. Zhang, and J. Cui, "Adaptively secure ciphertext-policy attribute-based encryption with dynamic policy updating," *Sci China Inf Sci*, vol. 59, no. 4, pp. 042 701:1–16, 2016.

[6] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. SP 2000. Proceedings. 2000 IEEE Symposium on*, 2000, pp. 44–55.

[7] E.-J. Goh *et al.*, "Secure indexes." *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[8] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Applied Cryptography and Network Security*. Springer, 2005, pp. 442–455.

[9] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," Proc. IEEE INFOCOM, 2010.

[10] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," Proc. IEEE INFOCOM, 2010.

[11] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacy preserving Query over Encrypted Graph-Structured Data in Cloud Computing," Proc. Distributed Computing Systems (ICDCS), pp. 393-402, June, 2011.

[12] Bruce Schneier, "Applied Cryptography", John Wiley & Sons, Inc. 1996

[13] The homepage of description of a new variable-length key, 64-bit block cipher http://www.counterpane.com/bfsverlag.html

[14] Patterson and Hennessy, "Computer Organization & Design: The Hardware/ Software Interface", Morgan Kaufmann, Inc. 1994

[15] B. Schneier, "Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)," Fast Software Encryption: Second International Workshop, Leuven, Belgium, December 1994, Proceedings, Springer-Verlag,1994, pp.191-204.

[16] S. Vaudenay, "On the Weak Keys in Blowfish," Fast Software Encryption, Third International Workshop Proceedings, SpringerVerlag, 1996, pp. 27-32.

[17] P. Karthigai Kumar and K. Baskaran. 2010. An ASIC implementation of low power and high throughput blowfish crypto algorithm. Microelectron. J. 41, 6 (June 2010), 347-355.

[18] TingyuanNie; Chuanwang Song; XulongZhi; , "Performance Evaluation of DES and Blowfish Algorithms," Biomedical Engineering and Computer Science (ICBECS), 2010 International Conference on , vol., no., pp.1-4, 23- 25 April 2010. [8] TingyuanNie; Teng Zhang; , "A study of DES