IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727
PP 21-26
www.iosrjournals.org

Efficient Semantic Search over Encrypted Data in cloud

[1]Anju T R, [2]Sangeeta Srinivas
[1]*Final Year M.Tech Department of CSE SNGCE, Kerala, India*
anjutraj@gmail.com
[2]*Assoc. Professor, Department of CSE SNGCE, Kerala, India*
sangeetahserus@gmail.com

Abstract: *Semantic search enable to improve search efficiency by understanding searcher intent and the contextual meaning as they appear in searchable space, whether on the web or within a closed system. semantic search generate synonyms set to provide relevant search results. Cloud storage has become more and more popular as it provides many benefits over traditional storage solutions. Despite the many benefits provided by cloud storage, many security problems have also arisen in cloud storage, which prevents companies from migrate their data to cloud storage. As a result, the owners encrypt their sensitive data before storing it in cloud storage. While encryption increases the security of the data, it also reduces the search ability of the data and thus, the Efficiency of the search. Recently, research has been done on several schemes which enable keyword searching on encrypted data in cloud computing. In this paper, we developed a system to support semantic search on encrypted data in cloud computing with scheme "Synonym Keyword Search". Our results demonstrated that our schemes are more efficient in terms of performance and storage requirements than the former proposed schemes.*

Keywords : cloud storage; encrypted cloud data; keyword search

I. Introduction

Historically, information retrieval (IR) has followed two principally different paths that we call syntactic IR and semantic IR. In syntactic IR, terms are represented as arbitrary sequences of characters and IR is performed through the computation of string similarity. In semantic IR, instead, terms are represented as concepts and IR is performed through the computation of semantic relatedness between concepts. Semantic search enable to improve search efficiency by understanding searcher intent and the contextual meaning as they appear in searchable space, whether on the web or within a closed system. Semantic search generate synonyms set to provide relevant search results.

Cloud storage has become a preferred method of storage as it provides many benefits over traditional storage solutions. With cloud storage, corporations can purchase only the needed amount of storage from the cloud storage provider (CSP) to fulfill their storage needs instead of maintaining their own data storage infrastructures. They can rely on CSP to handle all data maintenance tasks such as backup and recovery. It also allows all data to be accessed remotely in order to streamline their operations among different locations. With all these benefits, companies can significantly reduce their operation costs by simply outsourcing their business data to cloud storage. However, besides these benefits provided by cloud storage, many security problems arise in cloud storage that prevents companies from migrating their data to cloud storage [4]. Due to the fact that cloud storage is usually hosted by a third party provider and the cloud storage infrastructure is usually shared among different users, data stored in cloud storage can easily be targeted by the masquerade attack [6, 5] and the insider data theft attack [7,8]. These attacks threaten the security and the privacy of the stored data. As a result, the data owners cannot rely on CSP to secure their confidential data. These attacks also induce the data owners to encrypt all their sensitive data, such as social security numbers (SSN), credit card information, and personal tax information before they can be saved in cloud storage. The encryption approach may have strengthened the security of cloud data, but it has also degraded the efficiency of the data because the encryption will reduce the search ability of the data. Especially in the cloud computing environment, it is impractical for the user to download and decrypt all of the encrypted data from the remote cloud server before a search can occur. Therefore, an efficient scheme that supports search on encrypted data in cloud computing becomes very significant. In this paper, we developed a system to support semantic search over encrypted data in cloud computing with a different schemes "Synonym Keyword Search (SKS)",

II. Review Of Related Literature

A. Wildcard-based Fuzzy Set Construction (WFSC)

Li, Wang, et al in proposed another scheme called "Wildcard-based Fuzzy Set Construction (WFSC)" to enable fuzzy keyword search over encrypted cloud data. The key concept behind WFSC is maintaining an index file that covers all possible variations of a keyword within a predefined edit distance. Instead of simply encrypting the keywords extracted from the data file, WFSC expands each extracted keyword into a set of modified keywords by inserting wildcard characters into the keyword. For example "student" using the wildcard character '*' when the predefine edit distance value is equal to 1. Each modified keyword in the set will be hashed with a secured hash function to create a trapdoor. The trapdoor will be appended to the encrypted information that forms the uploaded data file to form an index entry. The resulting index entries form an index file which will be uploaded to the cloud storage together with the encrypted data files to support fuzzy keyword search. There are several weaknesses in WFSC if the user tries to expand the search coverage by increasing the predefined edit distance value. The increase in the edit distance value will cause a huge increase in the size of the modified keyword set.

B. Counter filtering

Gravano proposed Counter filter [1] is another important filter, which uses all the grams of a string as the prefix. Hence counter filter can also be regarded as a special case of general prefix filtering. The basic idea of count filtering is to take advantage of the information conveyed by the sets $G\sigma 1$ and $G\sigma 2$ of q-grams of the strings $\sigma 1$ and $\sigma 2$, ignoring positional information, in determining whether $\sigma 1$ and $\sigma 2$ are within edit distance k. The intuition here is that strings that are within a small edit distance of each other share a large number of q-grams in common.

Given the distance threshold τ , if $d(s1, s2) \leq \tau$, then they share at least the following number of common grams $\max(|Gq(s1)|, |Gq(s2)|)-\tau \cdot q$ in $Gq(s1)$ and $Gq(s2)$, denoted by *LB*, where *q* represents the gram length and $Gq(s1)$ represents the *q*-gram set of string *s*1. If only the query string *s*1 is considered, the lower bound of the number of common grams *LB* is $|Gq(s1)|-\tau \cdot q$, where $|Gq(s1)|$ is the number of *q*-grams. If we partition the query string into non-overlap segments with fixed length, called *q*-chunk, the common chunks lower bound[8] becomes $|Cq(s1)| - \tau$, where $|Cq(s1)|$ is the number of *q*-chunks. Consider a string s1, and let *s*2 be obtained by a substitution of a single character in *s*1. Then, the sets of *q*-grams*Gs*1 and*Gs*2 differ by at most *q* (the length of the *q*-gram). This is because *q*-grams that do not overlap with the substituted character must be common to the two sets, and there are only *q* *q*-grams that can overlap with the substituted character.

C. Prefix filtering

Prefix filter can be also generalized by extending the prefix. In recent study, Wang et al. [2] demonstrate that adding extra grams can decrease the candidate size by sacrificing a little estimation and filtering time. They select a prefix of each object and prune object pairs whose prefixes have no overlap It deals with ,Given a global order ∂ for all the appeared grams2, and the distance threshold τ , if $d(s1, s2) \leq \tau$, then the $(\tau \cdot q + 1)$-prefix of *s*1 and *s*2 must share at least one gram. *L-prefix* represents the first *l* grams of *q*-gram set sorted by ∂. For *q*-chunk, we just need to consider the $(\tau+1)$-prefix for query *s*1. The common grams lower bound *LB* is always 1. A global order can guarantee correctness of removing some grams to be indexed. However, this is based on an assumption knowing maximum τ. The assumption can't be usually Holden in query situation, since τ is a part of query which can't be known in advance. prefix-filter can be implemented using a combination of standard relational operators such as group by, order by, and join, and the notion of group wise processing where we iteratively process groups of tuples (defined as in group-by, i.e., where every distinct value in a grouping column constitutes a group) and apply a sub query on each group. In this case, group the tuples of *R* on *R:A* and the sub query would compute the prefix of each group it processes.

D.Gfilter

Haoji Hu a general gram filter is proposed [3]. We formulate the instantiation of general gram filter consists of two steps: (a) choosing high quality grams as the base query-gram set (b) extending the prefix by choosing more grams until no benefit can be gained from the new gram. a general gram filter. This generalization provides a chance to select optimized combination of grams from *q*-gram set of a query. Theoretical analysis is conducted on the complexity of the problem, which is NP-hard.

III. System Architecture

The system developed includes a user application, a cloud computing server, and a cloud storage server. The user application is hosted and maintained by the user party and is considered a trusted component while the cloud computing server and the cloud storage server are hosted and maintained by a third party CSP and are considered non-trusted components. Any communication channel between the user application and the cloud servers is also considered non-trusted because it can be targeted by attackers.

A. User Application

The user application in our system acts as the interface to handle all communications between the user and the cloud computing servers. The user application contains a small amount of local storage to hold different dictionaries that are needed for index creation and trapdoor creation. The user application is responsible for creating the index and encrypting the data file when the data owner wants to upload the files to cloud storage. When the user needs to do a search with a specific keyword, the user application is responsible for modifying the keyword into a trapdoor and submitting the search to the cloud computing server. The user application is also responsible for decrypting the returned index entries in order to retrieve the file information for the user when the cloud computing server returns the search results.

B. Cloud Computing Server

The cloud computing server acts as a controller which decides how the uploaded data file should be stored in the cloud storage server and retrieves the data file from the cloud storage server. The cloud computing server also performs the actual search operation using the trapdoor and the index file and returns the index entries that fulfill the search requirements.

C. Cloud Storage Server

The cloud storage server is simply a storage server that is used to store the uploaded data. There can be more than one cloud storage server in the system and the cloud computing server can be tuned to decide how the uploaded data files should be stored across different cloud storage servers.

IV. Semantic Search Schemes

Data owner wants to upload the data files to the cloud storage. The data owner will first submit the data files and a shared key to the user application. The user application will create an index file with different formats based on each developed scheme. After having created the index file, the user application will encrypt the data files using a symmetric-key algorithm with the shared key from the data owner. Both the index file and the encrypted data files will be uploaded together by the user application to the cloud computing server. The cloud computing server will store the index file and the encrypted data files in cloud storage server. When the data users want to conduct a search with a specific keyword, they will submit the search keyword and the shared key to the user application. The user application will modify the search keyword to form the trapdoor with different formats based on each developed scheme. The user application will then submit the trapdoor to the cloud computing server. The cloud computing server will search the index with the trapdoor based on each developed scheme. After searching through the whole index, the cloud computing server will return all index entries that matched the trapdoor back to the user application. The user application will decrypt the returned index entries and return the decrypted index entries back to the data users. The data users can use the information from the decrypted index entries to decide which data files should be retrieved from the cloud storage. Fig. 1 shows an overview of our schemes for semantic search over encrypted data in cloud

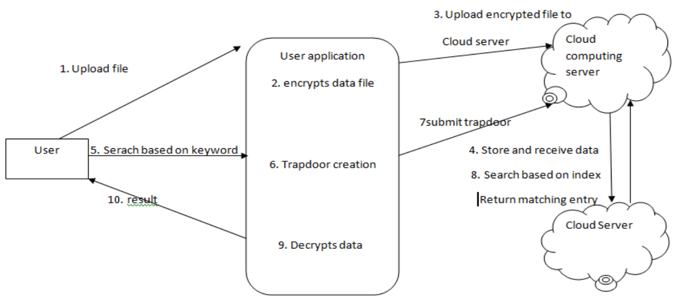


Figure 1. An overview of semantic search over encrypted data

D. Encryption/Decryption

This shows you how to basically encrypt and decrypt files using the *Advanced Encryption Standard (AES)* algorithm. AES is a symmetric-key algorithm that uses the same key for both encryption and decryption of data.
Basic Steps
Here are the general steps to encrypt/decrypt a file

- Create a Key from a given byte array for a given algorithm.
- Get an instance of Cipher class for a given algorithm transformation. See document of the Cipher class for more information regarding supported algorithms and transformations.
- Initialize the Cipher with an appropriate mode (encrypt or decrypt) and the given key.
- Invoke doFinal(input_bytes) method of the Ciphe**r** class to perform encryption or decryption on the input_bytes, which returns an encrypted or decrypted byte array.
- Read an input file to a byte array and write the encrypted/decrypted byte array to an output file accordingly.

B Synonym Keyword Search (SKS)

The first scheme we developed is called "Synonym Keyword Search (SKS)". SKS is an improved version of WFSC proposed in [3]. Instead of expanding the keyword with a wildcard character as in WFSC, SBKS uses the "Synonym Set Construction (SSC)" process to expand upon each keyword with synonyms of the keyword. Using the expanded keyword set, SBKS is able to improve the search quality by extending the search coverage to cover keywords that share similar meanings and reducing the number of unrelated keywords in the search result. The index construction of SBKS begins when the user application extracts distinct keywords from each data file. The user application will use the Synonym Set Construction (SSC) process to expand each extracted keyword into the synonym keyword set. The SSC process will first add the keyword to the keyword set and then check the keyword against the dictionary to determine if the keyword is misspelled. If the keyword is misspelled, spell check will be performed on the keyword to generate a list of keyword suggestions with correct spellings. Each keyword in the list of keyword suggestions will be added to the keyword set. After the spell check, the keyword set will go through the synonym dictionary to retrieve all the synonyms of each keyword in the keyword set. The SSC process will return all the distinct synonyms retrieved and all the distinct keywords in the keyword set to form the synonym keyword set. Table I shows examples of the synonym keyword set returned by the SSC process. Each keyword in the synonym keyword set will be hashed with a secure hash function to create a trapdoor. An index entry will be created for each trapdoor with the following format:
H(Synonym_Keyword)|Enc(List_of_File_ID|Shared_Key| Original_Keyword)
H(Synonym_Keyword) is the trapdoor created with the secure hash function.
Enc(List_of_File_ID|Shared_Key|Original_Keyword) contains the original keyword that was used to generate the synonym keyword set, the shared key submitted by the data owner, and the list of data files ID that contains the original keyword. All this information will be encrypted by the symmetric-key encryption with the shared key. For example, the following 4 index entries will be created for the keyword "holllday" that extracted from FILE1 and
FILE2:
• *H(holladay)|Enc(FILE1:FILE2|K|holllday)*
• *H(holiday)|Enc(FILE1:FILE2|K|holllday)*
• *H(holllday)|Enc(FILE1:FILE2|K|holllday)*
• *H(vacation)|Enc(FILE1:FILE2|K|holllday)*

The index construction will end when all the keywords extracted from the data files have been processed with index entries created. All the index entries will form an index file that will be updated to the cloud storage. When the user wants to search with a specific keyword, the user application will use the SSC process to generate the synonym keyword set for the search keyword. Each of the keyword in the synonym keyword set will be hashed by the secure hash function to generate the list of trapdoors. For example, the keyword "student" will generate the following trapdoors:
H(bookman)|H(educate)|H(pupil)|H(scholar)|
H(scholarly person)|H(student)

The list of trapdoors will be submitted to the cloud computing server. The cloud computing server will search the index by comparing the trapdoor in each index entry to each trapdoor in the list. The cloud computing server will return all matched index entries back to the user application. The user application will decrypt the information portion of the index entry and return the information to the data users.

TABLE I. **TABLE TYPE STYLES**

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

a. Sample of a Table footnote. (Table footnote)

V. Test Implementation

In our tests, we uploaded different numbers of files in our
data file collection using the user application and measured the changes in the size of the index file and the number of index entries in the index when the number of files uploaded
increased. Fig. 2 shows the comparison of the sizes of the indexes that were created by each supported schemes. According to the data in Fig. 2, the index files created by our developed schemes are smaller than both the index files created by WFSC-ED1 and WFSC-ED2. The data shows the index file of SBKS has an average of 23% size reduction when compared to WFSC-ED1 and an average of 92% size reduction when compared to the WFSC-ED2. The data shows the index files of WBKS and WBSKS have an average of 94% size reduction when compared to WFSC-ED1 and an average of 99% size reduction when compared to WFSC-ED2. These significant size reductions and index entry reductions proved our developed schemes are more efficient than WFSC in terms of storage requirements.

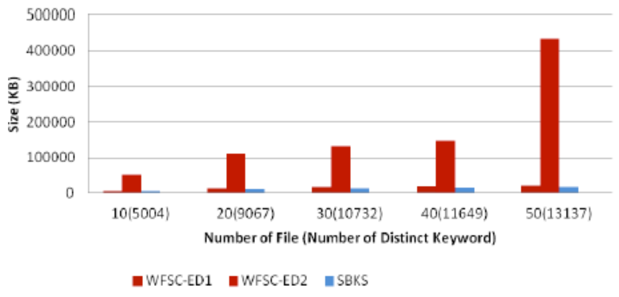
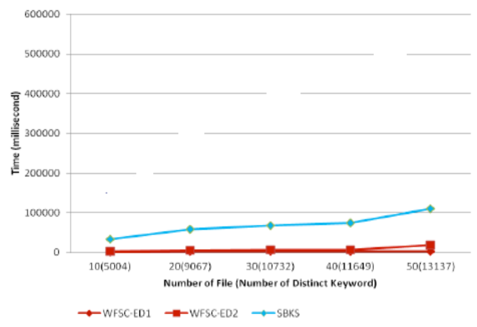


Figure2 Comparison of the sizes of the index files.



VI. Conclusion

In this paper, we developed a system to support semantic search over encrypted data in cloud computing with schemes "Synonym Keyword Search (SBKS)", the analysis shows that our developed schemes performed better than WFSC in terms of storage requirements, performance, and search quality while preserving the data security and the privacy of the uploaded data. Thus, we conclude that our developed schemes are more efficient and more practical than schemes that were proposed previously. Currently, the time of index construction is high so its optimizations could be considered in future work.

References

[1] L. Gravano, P. G. Ipeirotis, H. V. Jagadish N. Koudas, S. Muthukrishnan and D. Srivastava, “Approximate String Joins in a Database (Almost) for Free,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 491-500, 2001.

[2] J. Wang, G. Li and J. Feng, “Can We Beat the Prefix Filtering? An Adaptive Framework for Similarity Join and Search,” Proc. ACM SIGMOD Int’l Conf. Management of Data (SIGMOD), pp.85-96, 2012.

[3] Haoji Hu, Kai Zheng, Xiaoling Wang, and Aoying Zhou “GFilter: A General Gram Filter for String Similarity Search”21pp 1041-4347 2013

[4] F. Rocha, M. Correia, “Lucy in the Sky without Diamonds: Stealing Confidential Data in the Cloud,” Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on, 2011, pp 129-134.

[5] M. Salem, S. Stolfo, “Modeling user search-behavior for masquerade detection,” Recent Advances in Intrusion Detection, in Proceedings of the 14th International Symposium on, Heidelberg: Springer, 2011, pp1–20.

[6] M.T. Khorshed, A.S. Ali, S.A. Wasimi, “Monitoring Insiders Activities in Cloud Computing using Rule Based Learning,” Security and Privacy in Computing Communications (TrustCom), 2011 IEEE 10th International Conference on, 2011, pp. 757-764.

[7] S. Stolfo, M. Salem, A. Keromytis, “Fog Computing: Mitigating Insider Data Theft Attacks in the Cloud,” Security and Privacy Workshops (SPW-2012), 2012 IEEE Symposium on, 2012, pp 125-128.

[8] B. Ribeiro-Neto, M. Cristo, P.B. Golgher, E.S. Moura, “Impedance Coupling in Content-Targeted Advertising,” Proc. SIGIR 05, Salvador, Brazil, August 15–19, pp 496–503., ACM Press, NewYork.

[9] F. Rocha, M. Correia, “Lucy in the Sky without Diamonds: Stealing Confidential Data in the Cloud,” Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on, 2011, pp 129-134.

[10] J. Voris, N. Boggs, S. Stolfo, “Lost in Translation: Improving Decoy Documents via Automated Translation”, Security and Privacy Workshops (SPW-2012), 2012 IEEE Symposium on, pp 129-133.