International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

A Survey on Encrypted String Search Using Hash Chain Technique

Syeda Deema Quadri[1], Dr. Asma Parveen[2]

[1]PG Scholar, Department of Computer Science and Engineering, KBN College of Engineering, Kalaburagi, India

[2]Professor & HOD Department of Computer Science and Engineering, KBN College of Engineering, Kalaburagi, India

Abstract: ***A growing cloud services has made it possible to consider a third-party server as private storage. When cloud storage is used as a document archive, it is useful to provide functionality in which users can find the documents which contains a specified string as a substring. This can be obtained by using an efficient and easier-to-implement Symmetric Searchable Encryption Scheme (SSE) for string search, which takes one round of communication, O(n) times of computations over n documents. Usage of hash-chain technique in place of chain of encryption operations for index generation is what makes it suitable for lightweight applications. In this scheme, server is deprived of the frequency and the positions of the words being searched except that information which it can gain from the history. A new measure for privacy of search pattern is also to be introduced, which gives a measure of security against the leakage from trapdoor.***

Keywords: Cloud storage, Searchable encryption, hash-chain, multi keyword search.

1. Introduction

The cloud is designed to hold a large number of encrypted documents. With the advent of cloud computing, growing number of clients and leading organizations have started adapting to the private storage outsourcing. This allows resource constrained clients to privately store large amounts of encrypted data in cloud at low cost. However, this prevents one from searching. This inability gives rise to a newly emerging field of research, called Searchable Encryption (SE). Searchable Encryptions can be categorized into Symmetric Searchable Encryptions (SSE) and Asymmetric Searchable Encryptions (ASE), in this present case the Symmetric Searchable Encryptions (SSE). In this scheme, the client encrypts the data to store it on the cloud. It may be noted that client can organize the data in an arbitrary manner and can maintain additional data structures to achieve desired data efficiently. In this procedure, the initial client side computation is as large as the data, but subsequent computations to access data is less for both client and the cloud server.

Since huge volumes of documents are stored in a cloud server, searching against a keyword may result into large number of documents, most of which are not intended, causing unnecessary network traffic. This motivates the idea of searching against a string, which allows the search to be more specific. Searching for string is a multiple-keyword search where the ordering of keywords is preserved. So in addition to the presence of all the keywords in a document, their ordering and adjacency are to be taken care off while searching. So the index table needs to be prepared in such a way that the adjacency information of the words can be preserved. In order to protect the data security, the users generally encrypt their data before uploading them to the cloud. Searchable Encryption is one of the basic foundation for the data utilization in the cloud computing. The present method provides the functionality while keeping contents of the documents confidential.

2. Related Works

In Public Key Encryption with Keyword Search [1] authors analyzed and obtained the solution for the problem of searching on data that is encrypted using a public key system. Suppose user E sends email to user F encrypted under F's public key. An email gateway wants to detect whether the email contains the user F's keyword so that it could route the email accordingly. User F, on the other hand does not want the gateway to decrypt all her messages. Authors proposed a mechanism which enables the user F to provide a key to the gateway that enables the gateway to test whether the word "ARC" is a keyword in the email without learning anything else about the contents of the email and referred to this mechanism as Public Key Encryption with keyword Search. In [2] authors presented an approach in the field of encryption that allows both encrypted phrase searches and proximity ranked multi-keyword searches to encrypted datasets on untrusted cloud. However, this scheme still has areas that could be attacked. There are many-to-one relationship between keywords and their encrypted counterparts but not all encrypted values are equally distributed. In [3] authors designed and constructed a phrase search technique which is based on Bloom filters that is comparatively faster than many existing solutions, with better storage and communication cost. This technique uses a series of n-gram filters to support the functionality. The scheme exhibits a trade-off between storage and false positive rate, and is adaptable to defend against inclusion-relation attacks. In [4] authors proposed the first Dynamic Searchable Symmetric Encryption (DSSE) scheme that achieves the best of both worlds, i.e, both small leakage and efficiency. [5] is an extension of [4], authors showed that SSE is feasible on very large databases. Their basic construction was for single-keyword searches. In [6] authors presented as-strong-as-possible definitions of privacy, and constructions achieving them, for public-key encryption schemes where the encryption algorithm is deterministic.

One of their constructs, called RSA-DOAEP, has the added feature of being length preserving, so that it is the first example of a public-key cipher. They also generalized this to obtain a notion of efficiently-searchable encryption schemes which permit more flexible privacy to search-time trade-offs via a technique called bucketization. This scheme only provides privacy for plaintexts that have high min-entropy. They do not claim database fields being encrypted.

3. Algorithms

The scheme is a collection of four polynomial time algorithms (KeyGen, BuildIndex, Trapdoor, Search) such that:

1) **KeyGen** (1^{λ}) : KeyGen is a probabilistic key generation algorithm that is run by the client to setup the scheme.

Algorithm 1. Keygen
Input security parameter λ.
Output k_m, k^1, k_s and p.
km, k0 , ks ← Gen(1^{λ});
p ← P P NG(1^{λ});

2) **BuildIndex** (k_m, k^1, k_s, p) : BuildIndex is a probabilistic algorithm run by the client to generate sequence. An index is referred to a data structure that stores document collections.

Algorithm 2 Build Index
Input k_m, k^1, k_s, p, $D = (D_1, \ldots, D_n)$.
Output $SI = (I, I_f, I_c)$.
Form a collection $W = \{w_1, \ldots, w_d\}$ of all distinct words occurring in D;
$j \leftarrow 1$;
while $j \leq d$ **do**
$c_{i_j} = MAC_{km}(w_j)$;
$I_f[j] = c^{i-1}{}_j$;
$j \leftarrow j + 1$;
end while
$i \leftarrow 1$;
while $i \leq n$ **do**
$I_c[i] \leftarrow Enc_{km}(id(D_i))$;
For each sentence $s = (w_{s1}, \ldots\ldots, w_{sl})$ in D_i, chose $r \in Z_p$ randomly and form $(r_1, \ldots, r_l)$ such that $r_1 = r$ and for $2 \leq j \leq l$, $r_j = MAC_{ks}(r_j - 1)$. Associate r_j with the word W_{sj}.
$j \leftarrow 1$;
while $j \leq d$ **do**
set I[i][j] as all integers in Z_p that are associated with the word w_j and add the mask $m_j = MAC_{k0}(w_j)$ with all of them in modulo p, i.e., in Z_n;
if ($|I[i][j]| < f$) **then**
Inject $(f - |I[i][j]|)$ number of random elements from $Z^*{}_p$ in I[i][j];
end if
$j \leftarrow j + 1$;
end while
$i \leftarrow i + 1$;
end while

3) Trapdoor (k_m, k_s, p, s) :Trapdoor is a probabilistic algorithm run by the client to generate a trapdoor for a given string of words $s = (w_1, w_2, \ldots, w_l)$. The server uses the trapdoor to perform the search operation and recover pointers to appropriate document.

Algorithm 3 Trapdoor
Input $w = (w_{c1}, w_{c2}, \ldots, w_{cl})$, k_m, k^1, k_s, p, $MAC_k(.)$.
Output $t = (t_1, \ldots, t_l)$.
$j \leftarrow 1$;
while $j \leq l$ **do**
$e = Enc_{km}(w_{cj})$;
$msk = MAC_{k^1}(w_{cj})$;
$ci = MAC_{km}(w_{cj})$;
$t_{j1} = MAC_{ks}(e \oplus msk \oplus ci)$;
$t_{j2} = e \otimes ci$;
$t_{j3} = e \otimes msk$;
$t_j = (t_{j1}, t_{j2}, t_{j3})$;
$j \leftarrow j + 1$;
end while

4. Search (SI, t): Search is run by the server in order to search for the documents in D that contain the string s.

Algorithm 4 Search
Input $t = (t_1, \ldots, t_l)$, SI, k_s, $MAC_k(.)$.
Output encrypted_file_pointers, a list of encrypted document pointers;
the list column and column_msk are set empty;
$i \leftarrow 1$;
while $i \leq l$ **do**
$j \leftarrow 1$;
while $j \leq d$ **do**
$e = (t_{i2} \otimes I_f[j])$;
$m = t_{i3} \otimes e^{-1}$;
if ($MACk_s\, e \oplus m \oplus I_f[j]^{-1} == t_{i1}$) **then** set mask of j^{th} column as msk = m, add j to column and msk to column msk;
end if
$j \leftarrow j + 1$;
end while
$i \leftarrow i + 1$;
end while
the list encrypted_f ile_pointers is set empty;
$i \leftarrow 1$;
while $i \leq n$ **do**
if (there exists l integers $p_1, \ldots, p_l$ such that $(p_j \oplus$ column_msk[j]) $\in$ I[i][column[j]], $1 \leq j \leq l$ and $MAC_{ks}(pj) == p_{j-1}$ for $1 \leq j \leq l - 1$) **then**
add $I_c[i]$ to encrypted_f ile_pointers.
end if
$i \leftarrow i + 1$;
end while

5. Motivation

The cloud is designed to hold a large number of encrypted documents. This allows clients to privately store large amounts of encrypted data in cloud at low cost. However, this prevents one from searching. This gives rise to a newly emerging field of research, called searchable encryption (SE). As huge volumes of documents are stored in a cloud server, searching against a keyword may result into large number of documents, causing network traffic. This motivates the idea of searching against a string, which allows the search to be more specific. Searching for string is a multi-keyword search where the ordering and adjacency of keywords is preserved. Creation of index tables using linked lists results in inefficiency and increase in performance time

and therefore motivates the idea of creating index tables using hash chain technique which makes the scheme suitable for lightweight applications and is faster. Existing encryption schemes lacks formal security and also leaks lots of information beyond what is leaked from history. This drawback motivates the solution that Guarantees minimal leakages where the server is unaware of the frequency and search patterns thus providing search pattern privacy.

6. Conclusion

Due to the increase in the number of documents stored on the cloud, searching for a particular document might be difficult and time consuming which is undesirable. One solution is using Symmetric Searchable Encryption Scheme (SSE) which allows the client to encrypt the document before uploading it to the cloud thus providing security against privacy breaches. This scheme uses hash chain technique for index generation which makes it light weight and provides modification by providing security against active adversaries and trapdoors.

References

[1] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption With Keyword Search. In International Conference on the Theory and Applications of Cryptographic Techniques, pages 506–522. Springer, 2004.
[2] Steven Zittrower and Cliff C Zou. Encrypted Phrase Searching In The Cloud. In Global Communications Conference (GLOBECOM), 2012 IEEE, pages 764–770. IEEE, 2012.
[3] Hoi Ting Poon and Ali Miri. Fast phrase search for encrypted cloud storage. IEEE Transactions on Cloud Computing, 2017.
[4] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical Dynamic Searchable Encryption With Small Leakage. In NDSS, volume 14, pages 23–26, 2014.
[5] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. volume 2014, page 853. Citeseer, 2014.
[6] Mihir Bellare, Alexandra Boldyreva, and Adam ONeill. Deterministic and Efficiently Searchable Encryption. In Annual International Cryptology Conference, pages 535–552. Springer, 2007.