Keyword Search on Public Key Ciphertexts with Reduced overhead using a Query based Ranking System

CP Ramya
M. Tech. (Cyber Security)
Department of Computer Science & Engineering
Nehru College of Engineering and Research Centre Thrissur, Kerala

Vipin KM
Assistant Professor
Department of Computer Science & Engineering
Nehru College of Engineering and Research Centre Thrissur, Kerala

Mary Mareena
Assistant Professor
Department of Computer Science & Engineering
Nehru College of Engineering and Research Centre Thrissur, Kerala

Abstract

Since it is necessary to store data on different storage devices so to have privacy and security. It is desirable to encrypt the data before storing them. As the data is stored in a third party storage device in encrypted form it is very difficult to search data in its encrypted form, thus making the retrieval process difficult in large scale databases. There are many mechanisms to search and retrieve the encrypted data. The most trending searching method is based on keyword search on public key cipher texts. Existing systems takes enormous time to search and retrieve the encrypted set of keywords. This paper proposes a semantically secure searching mechanism by structuring the searchable cipher text with hidden relations with trapdoor corresponding to each keyword providing minimum information about the relation to the search algorithm. Here we add a ranking system for faster retrieval of frequently used files. This feature is useful when there is huge number of matched files, but the user only needs a small subset from those files hence by reducing the overall searching overhead.

Keywords: Key Word Search, Identity based Encryption, Ciphertext Security, Weil Pairing

I. INTRODUCTION

Today's data storage services must be totally trusted since they have full access to the data and thus they should not relieve it to others. Thus to provide a security to the data we use different techniques to encrypt the data before storing it on a third party server. Now the main issue arises when the user need to retrieve file based on certain keywords. So the fundamental problem is searching over the encrypted data.

Suppose a user Alice wishes to access her data from a number of devices like laptop, pager, desktop etc. Now these devices will be having large amount of data in it in an encrypted form. Alice only wishes to access data that contains the word "urgent" in it. Now the Bob sends a message with "urgent" in it. Now when Alice wishes to access all messages which are containing the keyword "urgent", in such a case the storage server need to search all the files containing the keyword "urgent". The biggest challenge for the server is that the files will be encrypted. So searching over these files is difficult. Thus to overcome such a situation many methods have been developed over the past few years.

In this paper we discuss a new enhanced method of searching over encrypted data using a hidden structure combined with a ranking system to achieve a fast and secure mechanism. Existing Generating Searchable Public-Key Ciphertexts with Hidden Structures for Keyword Search (SPCHS) is by far the fastest scheme available but when we added a ranking system to this mechanism and we noted that retrieval of files became faster.

II. RELATED WORK

In the recent years searching on encrypted data has been extensively investigated. Cryptographic searching techniques can all into two categories i.e, symmetric searchable encryption and public key searchable encryption.

Searchable Symmetric Encryption (SSE) introduced by Reza Cutmola et al. in [1] allows a user to outsource the storage of its data to another server in a secure private manner at the same time allowing to selectively searching over the data. The basic primitives for SSE contains three polynomial time algorithm ($\hat{G}$, ε, Đ) such that :

- $\hat{G}$: Takes the parameter K and returns a secret key.
- ε: Takes input as key K along with n bit message m and returns a cipher text c.

– Ð: Takes a key K and cipher text c as input and returns m. if the key K matches the key under which c was produced.

Initially this scheme took search time linear with the size of the databases. A number of efforts were to improve this scheme. Later on the search time was reduced to logarithmic time, but at the same time making the length of keyword search trapdoor linear with the size of databases.

This scheme does not provide any security about the history of the search. Thus patterns of a sequence of searches can be easily known to the adversary.

Public Key Encryption with Keyword Search was introduced by Boneh et al. in [2] which is based on problems on searching on data which is encrypted using a public key system. Suppose Bob sends data to user Alice which is encrypted using Alice's public key. Now Alice wish to check whether the data contains the word "urgent" in it, the gateway server need to search the file sent by Bob for the word "urgent", but Alice does not want to give the privilege of searching the document to the server in such a case the server has to search over the encrypted data. Using this scheme Alice can sent the server a key that will enable the server to identify all the files containing the specific keyword and learn nothing else.

Thus each time Alice wants to search a keyword she would construct a trapdoor for the corresponding word and will give it to the server. The server on receiving this trapdoor will run the test function to get the corresponding documents containing the specific keyword. Because of the trapdoor function the server learns nothing about the files. PEKS is semantically secure against an adaptive chosen keyword attack if for any polynomial time attacker Â we have the advantage in breaking function as negligible.

PEKS has the advantage that anyone can upload keyword searchable ciphertext to a server who knows receiver's public key similarly each sender needs to encrypt a file and its extracted keyword and send the resulting ciphertext to the server but this scheme takes search time linear with total number of ciphertext. Therefore in case of large database this scheme becomes inefficient.

From all the schemes mentioned so far Public key Encryption with Keyword Search is the most semantically secure scheme but it takes search time linear with the total number of ciphertexts. This problem is alleviated in Searchable public-Key Ciphertext with Hidden Structures (SPCHS) by Peng et al. in [3]. It provides keyword search as fast as possible without sacrificing semantic security of the encrypted keywords. Thus schemes use the concept of hidden structures that is all the keyword-searchable ciphertexts are structured by a hidden structure as shown in figure 1. Minimum information is disclosed to a search algorithm by using trapdoor corresponding to a keyword

The hidden structure is formed by ciphertexts as (ℂ, Pri, Pub), where ℂ denotes the set of all ciphertexts, Pri denotes the hidden relations among ℂ, and Pub denotes the public parts. In case there is more than one hidden structure formed by ciphertexts, then the multiple hidden structures formed by ciphertexts can be denoted by (ℂ, (Pri_1, Pub_1),…., (Pri_N, Pub_N)), where N ϵ Ǹ. Given (ℂ, Pub_1,…., Pub_N) and (Pri_1,…., Pri_N) except (Pri_i ,…, Pri_j) (where $i \neq j$), one can neither learn anything about (Pri_i, Pri_j) nor decide whether a ciphertext is associated with Pub_i or Pub_j .

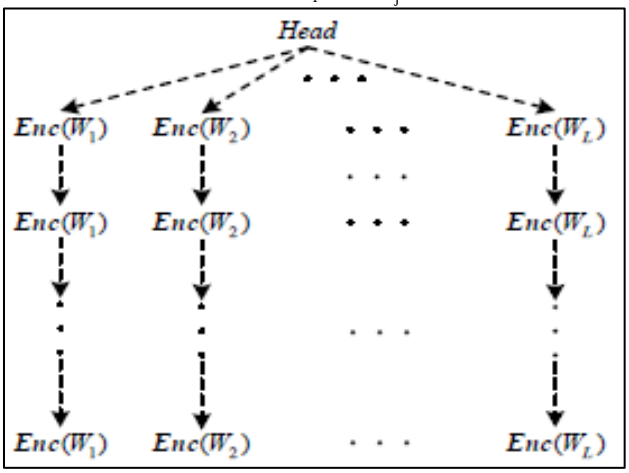

Fig. 1: Hidden Structure formed by Searchable Keyword Ciphertext. The dashed arrows represent the hidden relations. *Enc(W_i)* denotes the searchable ciphertext of keyword W_i.

In SPCHS, the encryption algorithm has two functionalities. One is to encrypt a keyword, and the other is to generate a hidden relation, which can associate the generated ciphertext to the hidden structure. Let (Pri, Pub) be the hidden structure. The encryption algorithm must take Pri as input; otherwise the hidden relation cannot be generated since Pub does not contain anything about the hidden relations. At the end of the encryption procedure, the Pri should be updated since a hidden relation is newly generated (but the specific method to update Pri relies on the specific instance of SPCHS). In addition, SPCHS needs an algorithm to initialize (Pri, Pub) by taking the master public key as input, and this algorithm will be run before the first time to generate a ciphertext. With a keyword search trapdoor, the search algorithm of SPCHS can disclose partial relations to guide the discovery of the ciphertexts containing the queried keyword with the hidden structure.

A. Definition of SPCHS

SPCHS consists of five algorithms:

- System Setup(1^k;Ẅ): It takes a security parameter 1^k and a keyword space Ẅ as input and probabilistically gives a pair of master public-and-secret keys (PK,SK) as output, where PK includes the keyword space Ẅ and the ciphertext space C.
- Structure Initialization (PK): It takes PK as input, and probabilistically initializes a hidden structure by giving private and public parts (Pri , Pub) as output.
- Structured Encryption (PK, W, Pri): It takes PK, a keyword W ϵ Ẅ and a hidden structure's private part Pri as input, and probabilistically gives a keyword-searchable ciphertext C of keyword W with the hidden structure as output, and updates the Pri.
- Trapdoor (SK,W): It takes SK and a keyword W ϵ W as input, and calculates the corresponding keyword search trapdoor T_W of W.
- Structured Search(PK, Pub, ℂ, T_W): It takes PK, a hidden structure's public part Pub, all keyword-searchable ciphertexts ℂ and a keyword search trapdoor T_W of keyword W as input, disclose partial relations to guide finding out the ciphertexts containing keyword *W* with the hidden structure.

This scheme is consistent since given any keyword search trapdoor T_W and any hidden structure's public part Pub, algorithm Structured Search(PK, Pub, ℂ, T_W) will find out all ciphertexts of keyword *W* containing the hidden structure Pub.

A receiver runs algorithm System Setup which sets up the SPCHS. Each sender uploads the public part of his hidden structure and keyword-searchable ciphertexts to the server, respectively using the algorithms Structure Initialization and Structured Encryption. Algorithm Trapdoor allows the receiver to delegate a keyword search trapdoor to the server. Then the server runs algorithm Structured Search for all sender's structures to find out the ciphertexts of the queried keywords.

SPCHS provides semantic security it resist adaptively chosen keyword and the structure attacks. In case the sender loses his local privacy Pri the SPCHS provides forward security that is existing hidden structure of ciphertexts still stays confidential. Using Structure Initialization sender can initialize new structure for new ciphertexts.

Each keyword-searchable ciphertexts are indexed by their first parts' binary bits. If there are in total n ciphertexts from n_S hidden structures, and the i-th hidden structure contains $n_{w,i}$ ciphertexts of keyword W ϵẄ. For the i-th hidden structure, the search complexity will be O(n_w, i log n). For all hidden structures, the sum search complexity is O((n_s + n_w) log n), where $n_w = \sum_{i=1}^{n_s} n_w$, i=1. As n = $\sum_{W\epsilon\ddot{W}} n_w$ and $n_w = \sum_{i=1}^{n_s} n_{w,i}$ and $n_s \ll n_w \ll n$. Thus the SPCHS instance is a more efficient search as compared to PEKS schemes, which has *O(n)* search complexity.

III. KEYWORD SEARCH ON PUBLIC KEY CIPHERTEXTS WITH REDUCED OVERHEAD USING A QUERY BASED ON RANKING SYSTEM

To improve the efficiency of SPCHS we are adding a ranking system to it. The keywords will be ranked on basis of its search frequency. Each time a keyword is searched its count will increase and so its rank.

A. Algorithm

Let $\gamma \xleftarrow{\$} \Re$ denote an element γ randomly sampled from ℜ. Let $\mathbb{G}$ and $\mathbb{G}_1$ denote two multiplicative groups of prime order q. Let g be a generator of G. A bilinear map $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ [4], [5] is an efficiently computable and non-degenerate function, with the bilinearity property $\hat{e}(g^a, g^b) = \hat{e}(g,g)^{ab}$, where $(a,b) \xleftarrow{\$} \mathbb{Z}_q^*$ and $\hat{e}(g, g)$ is a generator of $\mathbb{G}_1$. Let BGen(1^k) be an efficient bilinear map generator that takes as input a security parameter 1^k and probabilistically outputs (q, $\mathbb{G}$, $\mathbb{G}_1$, g,ê). Let keyword space Ẅ = $\{0,1\}^*$.

- System Setup(1^k, $\ddot{W}$): Take as input 1^k and the keyword space $\ddot{W}$, compute (q, $\mathbb{G}$, $\mathbb{G}_1$, g,ê) = BGen(1^k), picks $s \xleftarrow{\$} \mathbb{Z}_q^*$, set $P = g^u$, choose a cryptographic hash function H: Ẅ → $\mathbb{G}$, set the ciphertext space $C \subseteq \mathbb{G}_1 \times \mathbb{G} \times \mathbb{G}_1$, and finally output the master public key PK= (q, $\mathbb{G}$, $\mathbb{G}_1$, g, ê ,P, H, $\ddot{W}$, C), and the master secret key SK = s.
- Structure Initialization (PK): Take as input PK, pick $\xleftarrow{\$} \mathbb{Z}_q^*$, and initialize a hidden structure by outputting a pair of private-and-public parts (Pri = (u), Pub = g^u). Note that Pri here is a variable list formed as (u, {(W, $Pt[u,W]$)|$W \in \ddot{W}$}), which is initialized as (u).
- Structured Encryption(PK, *W*, Pri): Take as inputs PK, a keyword $W \in \ddot{W}$, a hidden structure's private parts Pri, picks $s \xleftarrow{\$} \mathbb{Z}_q^*$ and do the following steps:

1) Search (W, Pt[u, W]) for W in Pri;
2) If it is not found, insert (W, Pt[u, W] $\xleftarrow{\$}$ $\mathbb{G}_1$) to Pri, and output the keyword-searchable C = (ê, (P, H(W)^u, g^r, ê (P, H(W)^u · Pt[u, W]);
3) Otherwise, pick $R \xleftarrow{\$} \mathbb{G}_1$, set $C = (Pt[u,W], g^r, \hat{e}(P, H(W))^r \cdot R)$, update $Pt[u,W] = R$, and output the keyword searchable ciphertext *C*;

- Trapdoor (SK, T_W): Take as inputs SK and a keyword $W \in \ddot{W}$, and output a keyword search trapdoor $T_W = H(W)^s$ of keyword W.
- Structured Search(PK, Pub, $\mathbb{C}$, T_W): Take as inputs PK, a hidden structure's public part Pub, all keyword-searchable ciphertexts $\mathbb{C}$ (let $\mathbb{C}[i]$ denote one ciphertext of $\mathbb{C}$, and this ciphertext can be parsed as ($\mathbb{C}[i,1]$, $\mathbb{C}[i,2]$, $\mathbb{C}[i,3]$) $\in \mathbb{G}_1 \times \mathbb{G} \times \mathbb{G}_1$) and a keyword trapdoor T_W of keyword *W*, set $\mathbb{C}' = \emptyset$ and do the following steps:
  1) Compute $Pt' = \hat{e}(Pub, T_W)$;
  2) Seek a ciphertext $\mathbb{C}[i]$ having $\mathbb{C}[i, 1] = Pt'$; if it exists, add $\mathbb{C}[i]$ into $\mathbb{C}'$;
  3) If no matching ciphertext is found, output $\mathbb{C}'$;
  4) Compute $Pt' = \hat{e}(\mathbb{C}[i, 2], T_W)^{-1} \cdot \mathbb{C}[i,3])$ and go to Step 2.
- Ranking system: each time a word *W* is searched assign it with a priority number. As its count increases increase its rank.

IV. ANALYSIS

We coded SPCHS and analyzed its time cost of algorithm Structured Search and plotted a number of ciphertext verses time graph which can be seen in figure 2.

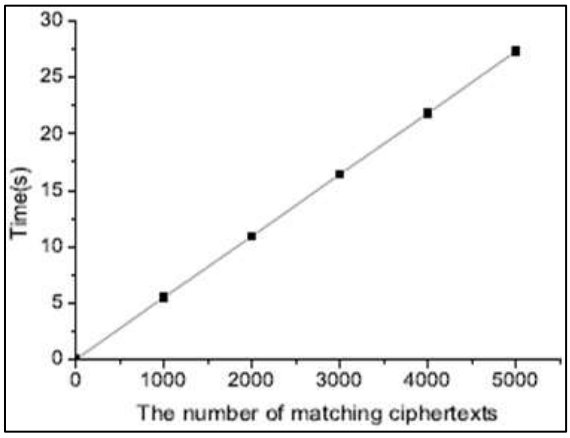

Fig. 2: Time cost of SPCHS

After implementing SPCHS scheme we added the ranking system module to the algorithm making it Keyword Search on Public Key Ciphertexts with Reduced Overhead using a Query based Ranking System We searched the same words in both SPCHS and advanced keyword search using ranking system and found that each time different users search of the same keyword the search time decreases, thus giving us a more efficient searching mechanism. The figure 3 shows the graphical representation of search time of a particular keyword searched in SPCHS and advanced keyword search.

We can clearly see that as the word search count increases the time of retrieving the corresponding files decreases in the advanced search whereas it remains constant in SPCHS.

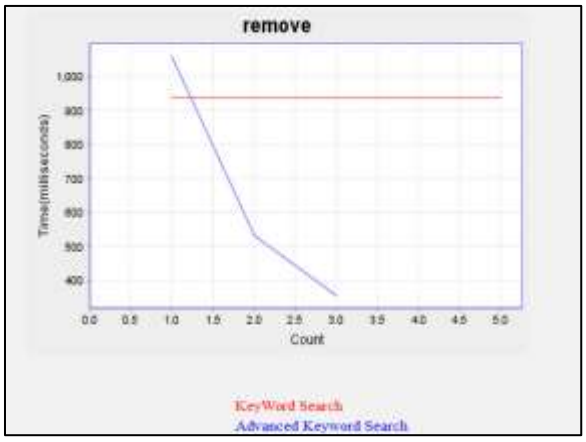

Fig. 3: Comparison of retrieval time of SPCHS and Advanced search with ranking system.

V. CONCLUSION AND FUTURE WORK

SPCHS makes the search secure at the same time makes the search faster than all the existing schemes which are available for keyword searching over encrypted data. But when added the ranking system to it retrieval from a large database becomes faster.

A segregation and distribution mechanism can be added to this scheme which will be helpful in implementing this mechanism in a cloud infrastructure to search and retrieve files from a cloud.

REFERENCES

[1] Curtmola R., Garay J., Kamara S., Ostrovsky R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: ACM CCS 2006, pp. 79-88. ACM (2006)
[2] Boneh D., Crescenzo G. D., Ostrovsky R., Persiano G.: Public Key Encryption with Keyword Search. In: Cachin C., Camenisch J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506-522. Springer, Heidelberg (2004)
[3] Peng Xu., Qianhong Wu., Wei Wang., Willy Susilo., Josep Domingo-Ferrer., Hai Jin.: Generating Searchable Public-Key Ciphertexts with Hidden Structures for Fast Keyword Search. In: IEEE S&P 2015, pp. 1992-2006, IEEE(2015)
[4] Menezes A. J., Okamoto T., Vanstone S. A.: Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. IEEE Transactions on Information Theory, 39(5), pp. 1639-1646 (1993)
[5] Frey G., Muller M., Ruck H.-G.: The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. IEEE Transactions on Information Theory, vol. 45, no. 5, pp. 1717-1719 (1999)