ISSN XXXX XXXX © 2018 IJESC

Research Article Volume 8 Issue No.12

Stamp Enabled Energy Efficient Search Scheme in Data Mining Over Encrypted Data on Mobile Users into Cloud

M.Subha[1], M.Umasankar[2]
Assistant Professor[1], M. Phil Student[2]
Department of Computer Science
Kaamadhenu Arts & Science College, Sathyamangalam, India

Abstract:
Cloud storage provides a convenient, massive, and scalable storage at low cost, but data privacy is a major concern that prevents users from storing files on the cloud trustingly. One way of enhancing privacy from data owner point of view is to encrypt the files before outsourcing them onto the cloud and decrypt the files after downloading them. However, data encryption is a heavy overhead for the mobile devices, and data retrieval process incurs a complicated communication between the data user and cloud. Normally with limited bandwidth capacity and limited battery life, these issues introduce heavy overhead to computing and communication as well as a higher power consumption for mobile device users, which makes the encrypted search over mobile cloud very challenging. In this work, we propose TEES (Traffic and Energy saving Encrypted Search), a bandwidth and energy efficient encrypted search architecture over mobile cloud. The proposed architecture offloads the computation from mobile devices to the cloud, and we further optimize the communication between the mobile clients and the cloud. It is demonstrated that the data privacy does not degrade when the performance enhancement methods are applied. Our experiments show that TEES reduces the computation time by 23% to 46% and save the energy consumption by 35% to 55% per file retrieval, meanwhile the network traffics during the file retrievals are also significantly reduced.

1. INTRODUCTION

1.1 STATEMENT OF THE PROBLEM

The data privacy issue is paramount in cloud storage system, so the sensitive data is encrypted by the owner before outsourcing onto the cloud, and data users retrieve the interested data by encrypted search scheme. In MCS, the modern mobile devices are confronted with many of the same security threats as PCs, and various traditional data encryption methods are imported in MCS. However, mobile cloud storage system incurs new challenges over the traditional encrypted search schemes, in consideration of the limited computing and battery capacities of mobile device, as well as data sharing and accessing approaches through wireless communication. Therefore, a suitable and efficient encrypted search scheme is necessary for MCS. Generally speaking, the mobile cloud storage is in great need of the bandwidth and energy efficiency for data encrypted search scheme, due to the limited battery life and payable traffic fee. Therefore, focused on the design of a mobile cloud scheme that is efficient in terms of both energy consumption and the network traffic, while keep meeting the data security requirements through wireless communication channels.

1.2 OBJECTIVES

1) TEES reduces the energy consumption by35%55% by offloading the computation of the relevance scores to the cloud server. This reduces the computing workload on the mobile device side while at the same time significantly speeding up the mobile file access speed (e.g. it doubles the speed for accessing a 100KB file).
2) With a simplified search and retrieval process, TEES reduces the network traffic for the communication of the selected index, and reduces the file retrieval time by 23%46% in our experiments.
3) In implementing the redesigned encrypted search procedure, TEES redistributes the encrypted index to avoid statistics information leak, and wraps key words adding noise in order to render them indistinguishable to the attackers. Security analysis shows that the security level of TEES is guaranteed and enhanced for MCS wireless communication channels.

1.3 MOTIVATION

Introducing TEES (Traffic and Energy saving Encrypted Search) architecture for mobile cloud storage applications. TEES achieves the efficiencies through employing and modifying the ranked keyword search as the encrypted search platform basis, which has been widely employed in cloud storage systems. Traditionally, two categories of encrypted search methods exit that can enable the cloud server to perform the search over the encrypted data: ranked keyword search and Boolean keyword search. The ranked keyword search adopts the relevance scores to represent the relevance of a file to the searched keyword and sends the top-k relevant files to the client. It is more suitable for cloud storage than the boolean keyword search approaches since Boolean keyword search approaches need to send all the matching files to the clients, and therefore incur a larger amount of network traffic and a heavier post-processing overhead for the mobile devices.

1.4 A secured and searchable encryption algorithm for cloud storage

Cloud computing is a new generation technology which efficiently support the client oriented services. Now in these days there are a number of applications which consumes the cloud storage service for storing and retrieving information. In such conditions the data owner management and privacy preservation cryptographic techniques are utilized frequently. But due to cryptographic technique of security implementation the data leave their own format and converted into other unreadable format. Due to this retrieval of required information becomes complex. Therefore in this work a proposed solution incorporate the hash table management and indexing techniques to keep track the actual data contents in terms of document features which may help for encrypting user data and identifying the user data and privacy**.** The Internet access becomes available in the recent years, Cloud computing is an internet based technology; it is using hardware and software as computing resources to provide service through internet. Cloud computing is being widely used now-a-days enabling the end user to create and use software from anywhere at any time without worrying about the execution of the technical information. Cloud computing technology provides unlimited resources and services like data storage service which helps to manage the user data. Nowadays, with the help of dynamic data operation with computation user can store their data in cloud which makes the copy of data for further updating and verification of the data loss. Here with the help of cryptographic technique data can be secured from unauthorized user or access. The benefits of the cloud storage are flexible with reduced cost and you also manage the data loss risk and so on. To provide an information retrieval function while addressing the security and privacy issues, the concept of searchable encryption was introduced in an earlier study. With searchable encryption, the entity performing the retrieval service is not allowed to learn the content of the queries and responses. Instead, some additional encrypted index terms (serving as keywords or hints) are used for the data search process. The entity uses a cryptographic algorithm similar to decryption for finding the correspondence between the encrypted query and the encrypted data content. They summarize the essential requirements specified in for searchable encryption in terms of privacy and security issues from the view point of unqualified users as follows:

Privacy of data: No one can uncover information about data content from the query and response as well as the cipher text itself.

Privacy of the data owner: No one can learn about the identity of the data owner from the encrypted content you suggested that one's identity can be viewed as a combination of several attributes expressing the characteristics of the user in the form of access policy by using Boolean expressions such as AND, OR, or NOT. On the other hand, CP-ABE is complementary to KP-ABE by enabling encrypt or to specify access policy combined with the cipher text. Both schemes allow secure one-to-many communications such as targeted broadcasts for a specific group and individual user according to their attributes, some studies suggested modification of ABE schemes by hiding the access policy. These schemes operate on the assumption that the data owner directly delivers the cipher text to the receiver without an intermediate third party. In other words, when adopting those approaches directly in cloud storage, decryption keys can be exposed to an unauthorized third party. Hence, they are not feasible for data retrieval services in the cloud storage systems because the test procedure allows the CSP to learn which attributes the user has to motivate and solve the problem of supporting effective ranked keyword search for providing efficient use of remotely stored encrypted data in Cloud. You firstly give a fundamental scheme and show that by same existing searchable encryption method, it is not efficient to achieve ranked search. So you appropriately weaken the security guarantee, to solve this security problem you developed cryptography first OPSE, and derive an efficient one-to-many order-preserving mapping function, which allows the effective RSSE to be designed. Through thorough security analysis, you show that their proposed solution is secure and maintain the privacy, while correctly realizing the aim of ranked keyword search. And also shows that their solution enjoys "as-strong-as possible" security guarantee compared to conventional SSE schemes; Note that in their design, you focus on single keyword search. This section includes involved work and identified issues in system in addition to that an optimum solution is also provided. The cloud environment provides support for efficient computing and enables to provide the storage solutions at the remote end. The main aim is to address the following issues in the existing cloud storage:

1. **Data security:** The data is placed on the cloud which is not much secured due to third party access and treads therefore the data security in cloud storage is required.
2. **Data owner and client privacy management:** The data owner and client in not distinguishable using the data additionally the privacy on such data is access is required.
3. **Searchable data space:** The cryptographic manner of data security converts the formats and not a bit of data recovered during the information retrieval. In this work we discussed various method of searchable encryption to secure the data in cloud storage. Also we discussed about cryptography methods which helps to convert the data from readable to unreadable form so our data could be saved in cloud storage from adversary. By studying all these work we can conclude that the essence of security for cloud storage is very necessary so that client could feel secure while accessing the cloud storage services. In this work we proposed a scheme for secure data accessing with maintaining its privacy by using strong cryptographic algorithm. Our future work will attempt to enhance the feasible solution.

1.5 OVERVIEW

TEES is implemented with security enhancement based on popular TF-IDF, but the essential security defects of this encryption approach cannot be completely resolved. To the best of our knowledge, there is no unbreakable security scheme, but TEES architecture is general enough to host and enhance various encrypted search schemes as we will discuss in Section 4.3. Moreover, we suggest that a cloud storage service provider is semi-honest and will not collude with attacker in TEES, as most of the related works. TEES employs the architecture redesign over traditional encrypted search procedure, and our comprehensive experiments prove the TEES has following advantages in comparison with the traditional complex encrypted.

1.6 PERFORMANCE ANALYSIS

In this section, we analyze the security of TEES based on important security threats mentioned. Concretely, the most important principle of the design is to prevent the attacker from obtaining any plaintext information regarding our data file set or the searched keyword. Then we should let the trusted but curious mobile cloud server learn as little information as possible. Last but not least, an unauthenticated data user should not be able to perform any file retrieval. We ran experiments to test the security of TEES.

- Statistics Information Leak Control
- Keywords-files Association Leak Control
- Server Information Acquisition Control

2 .Experimental Setup [or] Study of the work [or] Proposed Methodology

2.1 ABOUT THE THESIS

To effectively support an encrypted search scheme with high security level over cloud data, we introduce a new architecture that is named as TEES. Our aim is to design a practical solution for secure encrypted search over mobile cloud storage. We first introduce the design idea and then introduce development of our own protocol with the change of the traditional process of file search and retrieval for the cloud data our scheme achieves the security and efficiency goals mentioned above. Thereafter, we discuss the reasons why TEES can achieve performance efficiency.

2.2 THE IDEA OF TEES

The basic idea behind TEES is to offload the calculation and the ranking load of the relevance scores to the cloud. It has been highlighted that offloading some computation intensive applications onto the cloud can be an efficient low power design philosophy. Cloud providers can provide computing cycles, and users can use these cycles to reduce the amounts of computation on mobile systems and save energy. However, at the same time, offloaded applications intend to increase the transmission amount and thus increase the energy consumption from another aspect. This double effect motivates us to carefully redesign the traditional file encrypted search and retrieval process. You first take an overview of major processes for all file encrypted search and retrieval schemes. There are normally three main processes:

- The process of authentication is used by the data owner to authenticate the data users.
- The file set and its index are stored in the cloud after being encrypted by the data owner during the preprocessing and indexing stages.
- The data user searches the files corresponding to a keyword by sending a request to the cloud server in the search and retrieval processes.

2.3 TEES IMPLEMENTATION FOR SECURITY ENHANCEMENT FOR MOBILE CLOUD

In order to achieve security enhancement with energy and traffic efficiency, you implement the modules in TEES using modified routines and new algorithms. Our system will be introduced in three parts. As previously mentioned, the data owner should build a TF table as index and encrypt it using OPE in order to offload the calculation and ranking load of the relevance scores to the cloud. So as to control the statistics information leak, you implement our one-to-many OPE in the data owner module. You also wrap the keywords to be searched by adding some noise in the data user module to help controlling the keywords-files association leak. In order to get top-k relevant files, you implement ranking function to calculate the relevant score on the cloud. Given a keyword in ORS, the cloud server is in charge of calculating the relevance scores or the data user to get the corresponding top-k relevant files. Therefore, you implement both the unwrapandrank functions in the cloud server module.

In STAMP Enabled TEES, the cloud server calculates the relevance score and finds the most relevant files corresponding to a given keyword. In order to find the TF value of a queried keyword, the cloud server should also know the unwrap function of the user-supplied wrapped tuple.Therefore, the cloud server gets more information than any potential attacker. As most of the previous schemes, we assumed that our test cloud server semi-trusted, and therefore we only need to minimize the amount of information it acquires. Moreover in terms of performance improvements, information leakage does not seem to be a very serious problem, and updating theTF table periodically also protects the index from being inferred by the server.

Note that TEES is established with widely used TFIDFencryption approach, and all nature defects of this encrypted search scheme cannot be completely resolved even TEES In addition, when a data user performs search, the keyword to be queried will be encrypted by the data owner's key. The user receives a hash table after the first time it is authorized by the data owner.

2.4. MODULES DESCRIPTION

CLOUD SERVER MODULE

In this module the functions that unwraps the keywords and rank the relevance scores for the cloud server module. These functions are used to get the top-k relevant files according to a given search keyword.

Unwrap Function: Note that the cloud server is semi-trusted, and the unwrap function can be processed by the server. Upon receiving the tuple Wrap$(\mathcal{w}) = (h_1, h_2)$, the server calls

Unwrap$((h_1, h_2))$, to get $\psi(\pi_\alpha(\tilde{\varpi})) = h$, searches into the TF table, and then sends back the corresponding files. The equation is:

$$\begin{cases} h^2 + h - \dfrac{\sqrt{h_1} + h_2}{\lambda} = 0 & \text{if } h_1 < h_2 \\ h^2 + h - \dfrac{h_1 - \sqrt{h_2}}{\mu} = 0 & \text{if } h_1 \geq h_2 \quad (5) \end{cases}$$

Assuming that the random number (noise) created by the wrap function is positive, the unwrap function behave as expected. Since h is a positive integer, we could recover h using Unwrap():

$$h = \begin{cases} \dfrac{\sqrt{1 + \frac{4\left(\sqrt{h_1} + h_2\right)}{\lambda}} - 1}{2} & (h_1 < h_2) \\ \dfrac{\sqrt{1 + \frac{4\left(h_1 + \sqrt{h_2}\right)}{\mu}} - 1}{2} & (h_1 < h_2)(6) \end{cases}$$

Ranking Function:

Cloud server calculates the relevance scores and returns top-k relevant files according to the searching query from data user. The calculation scheme in is used in our scheme. Note that due to the order preserving index, any other relevance scores calculation method can also be employed. TEES calculates the relevance score as Equation (7):

$$\text{Score}\ (W_s\ , Fc) = \sum_{\omega\epsilon W_s} \frac{1}{|F_c|} \times \left(1 + \text{In}\ f_{c,w}\right) \times \times\ \text{In}(1 + \frac{D}{f_\omega})(7)$$

Here, Ws is the keyword set to be searched; Fc is a certain file in the file set; $f_{c,w}$ denotes the TF of the keyword ω and D is the total number of files.

When performing a single keyword search, the IDF factor in Equation (7) is constant. Thus, we simplify the equation as follows:

$$\text{Score}(\omega, Fc) = \frac{1}{|F_c|} \times (1 + \text{In}\ f_{c,w})\ (8)$$

The cloud server sends back the top-k relevant files after ranking the scores using this relevance score calculation algorithm, as depicted in Algorithm 5.

Algorithm 5 Top-k Ranking Function

```
Input:  ω, k
Output:  topFiles
1:ifthis request is sent by a "legal" user then
2:for each file Fc ϵ F do
3:Calculate Score (ω, Fc) but with a warning.
4:end for
5: else
6: Return "No Permission"
7:end if
8: Rank the scores to get top-k files topFiles = {$topF_1$,$topF_2$, ... .. $topF_k$}.
9:returntopFiles.
```

TEES can also support other document modelling method such as Latent Dirichlet allocation. We can also find the probability for a term to appear in a file using a middle tier "topic", and then store its encrypted value in the index. Note that TEES employs single-keyword search, but the basic ideas, such as design efficiency by offloading and security enhancement by adding noise, can be extended to all other encrypted search schemes. Moreover, it is known that multiple-keywords search can provide more accurate the search results, but makes the search more complicated at the same time. In mobile cloud, the single-keyword is enough to distinguish the documents that users need since our documents are classified clearly. Moreover, if we search encrypted data with multi-keywords, we should sacrifice the search accuracy because most popular OPE (Order Preserving Encryption) does not support multi-keyword well. Most OPE could guarantee the order between two values, but when these two values multiplied by several factors and then get the arithmetic sum (as multi-keyword search process), we cannot insure the order between the arithmetic sums. Another important respect is that, decreasing the number of keywords could decrease the consumption of energy for mobile device. Overall, developing a single keyword search scheme is a proper solution of encrypted search data sharing for mobile cloud storage. Moreover, TEES is a general architecture, where the OPE method proposed here can be substituted by other novel schemes.

DATA OWNER MODULE

We modify the way of building the index to support the ORS scheme by our one-to-many OPE and implement it to control the statistics information leak. The authentication between the data owner and the data user is also redesigned in order to ensure the security of TEES. We now elaborate the implementation of the index construction, the encryption functions and detail the authentication process.

TF-IDF:

TF-IDF is the product of two statistics, term frequency and inverse document frequency. Various ways for determining the exact values of both statistics exist. In the case of the **term frequency (TF)** t f (t , d), the simplest choice is to use the raw frequency of a term in a document, i.e. the number of times that term t occurs in document d. If we denote the raw frequency of t by f(t, d), then the simple TF scheme is t f (t, d) = f(t, d).

The inverse document frequency (IDF) is a measure of whether the term t is common or rare across all documents. It is obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient. Then TF-IDF is calculated as:

$$\text{tf} - \text{idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D), \qquad (4)$$

Where D denotes the total number of documents in the data set. A high weight in TF-IDF is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights thus tend to filter out common terms. Since the ratio inside the IDF's log function is always greater than or equal to 1, the value of IDF (and TF-IDF) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the IDF and TF-IDF closer to 0.

KEYWORD INDEX BUILDING MODULE

In TEES, the data owner starts by collecting the files he wants to store into the cloud. Consider a file set $\mathcal{F} = (F_1, F_2, \dots, F_n)$ containing the number of $|\mathcal{F}|$ files, in which a term set $\mathcal{T} = (t_1, t_2, \dots, t_m)$ and the number of $|\mathcal{T}|$ terms appear. We create a table of size$|\mathcal{F}| \times |\mathcal{T}|$ for all the files and all the terms, where the value at the i_{th} row and j_{th} column denotes the number of occurrences of the i_{th} term in the j_{th} file. This numerical value is the TF value.Then a constant S is chosen as a cofactor to standardize these occurrences with the size of the files. In TEES, we use this TF table as our index and the cloud server calculates the relevance scores using the encrypted TF values. Let GenKey() be the function that generates the keys. Let π() be a hash function that encrypts the terms. In TEES, π() is instantiated by a hash function such as MD5. Let ψ() be the hash of the encrypted terms,$\psi(t_i)$ be the entry of the term t_i in the index. Let ε() be an encryption algorithm for the TF values.

When building an index, it executes following two steps:

1) First, the data owner starts by calling GenKey(), to generate a key α to encrypt the terms, a key β to encrypt the index, and the noise $\lambda > 0$; $\mu > 0$ to wrap the keywords. He then outputs $K = \{\alpha, \beta\}$ and $N = \{\lambda, \mu\}$.

2) Second, the data owner builds a secure index by calling BuildIndex($\mathcal{K}, \mathcal{F}$)(encrypted $\mathcal{N}$ is sent to cloud server) as described in Algorithm 1:

Algorithm 1 BuildIndex

```
Input: $\mathcal{K}, \mathcal{F}$
Output: $\mathcal{I}$
1: Extract the terms $T = (t_1, t_2, \dots t_m)$ from the file set $\mathcal{F}$.
2: For $t_i \in T$ do
3:    Get the encrypted term $\pi_\alpha(t_i)$ and hash it to get its entry $\psi(\pi_\alpha(t_i))$ in the TF
          table.
4: end for
5: For $t_i \in T$ and $1 \le j \le |\mathcal{F}|$ do
6:    Calculate the term frequency $tf_{ij}$ and get $\widetilde{tf_{ij}} = |(\frac{S}{|F_i|}) \times tf_{ij}|$.
7: end for
8: Compute $\varepsilon_\beta(\widetilde{tf_{ij}})$, and store it in the index I.
9: return I;
```

DATA ENCRYPT FUNCTION AND AUTHENTICATION MODULE

An aforementioned OPE approach employing a one-to-one mapping can be used to $\varepsilon_\beta()$.However such an OPE strategy cannot be directly employed to secure the TF values in consideration of security issues in MCS. In order to flatten the distribution, we use a one-to-many OPE, i.e., we map every TF value to a random number in a certain range. For example, a TF value t f will be mapped to a range $[tf^x, tf^y]$, where $0 \le tf^x \le tf^y < 2^B$ are the lower and upper bound of the random mapping range (B is set to 16 in our performance evaluation and to 8 in our security evaluation). For two adjacent TF values tf_1 and tf_2 the random mapping ranges $[tf_2^x, tf_2^y]$ are chosen to be non-overlapping but close to one another, that is if $tf_1 < tf_2$then $tf_1^y < tf_2^x$.
Therefore, the Order Preserving Encryption (OPE) is maintained. In order to increase the entropy of the ciphertext, we adaptively generate the mapping range according to the distribution of the raw TF values and encrypt them as described in Algorithms 2 and 3. Here, $G(TF)=\{G(tf_1) \dots \dots G(tf_i)\}$ is a lower bound set for all the TF values in a certain TF table, while H (TF) is the upper bound set. Our algorithm is order-preserving since a given TF value$tf_1$is encrypted to an integer $E(tf_1)$ which is less than $tf_1^y$and $tf_2^x = tf_1 + 1$ is mapped into a new range whose lower boundary is larger than tf_1^y.

Our one-to-many OPE flattens the TF values histogram over a file set so as to ensure the system security, which will be discussed and evaluated. OPE is a secure algorithm for encryption.

Our OPE algorithm is secure enough for daily use. Meanwhile, our algorithm is a simple implementation of order-preserving encryption which will consume less energy than other complicated ones. And if we desire to update our files or index with our mobile device, this energy efficiency algorithm will become a good choice for data owners. The data owner can also use AES (Advanced Encryption Standard) to encrypt the files.

Algorithm 2 Key Generation

```
Input: TF Table
Output: $\tilde{G}(TF), \tilde{H}(TF)$
1: Get the distribution histogram $\psi$ of the TF table and get $TF_x$ as all TF values occur in $\psi$
  2: for $tf_i \in TF_x$ do
  3: Get the occurrence $c_i$.
  4: end for
5: Get $C = \sum_{i=1}^{|TF_x|} c_i$.
6: for $tf_i \in TF_x$ do
7: Calculate $p_i = c_i / C$
8: end for
9: for $tf_i \in TF_x$ do
 10: if i==1 then
11: Get $G(tf_i) = 1$ and $H(tf_i) = floor\ (2^B \times p_i)$.
12: else
13: Get $G(tf_i) = H(tf_{i-1}) + 1$ and $H(tf_i) = H(tf_{i-1}) + floor\ (2^B \times p_i)$.
14: endif
15: end for
16: return $\tilde{G}(TF), \tilde{H}(TF)$.
```

Algorithm 3 Order Preserving Encryption

```
Input: t f
Output: $E(tf)$
  1: for $t_i \in T$ and $1 \le j \le |\mathcal{F}|$ do
2: Get $E(tf_{ij}), E(tf_{ij}) \overset{R}{\leftarrow} \{G(tf_{ij}), G(tf_{ij}) + 1, \ldots, H(tf_{ij})\}$.
  3: end for
  4: return $E(tf)$.
```

Authentication:
In TEES, the data owner maintains a set of legal users ("legal set") and a set of users that will become invalid in after a defined delay ("overdue set"). When a user intends to access the file, he first sends his information to be authenticated by the data owner. In our design, we use our unified school authentication in TEES and transfer it through https for safety concern. Then the data owner records the "International Mobile Equipment Identity" of the user's mobile device and stores its encrypted version into the cloud. Note that the data owner should regularly update the hash table and the keys such that only users in the "legal set" will be notified. At the same time, this authentication process needs the data owner be online, but mature notification methods can be involved to push the authentication requests to the offline data owner.

REDESIGN OF THE DATA USER MODULE: The data user module is executed on the mobile client's side. The wrap function of the keywords is implemented to solve the keywords-files association leak. In the wrap function, the stem, the encryption and the hash operation are exactly the same as in the index building algorithm. The function decrypting the files corresponds to the encryption done by the data owner. The authentication function is used for authentication. We now detail the wrap function of this module.

Wrap Function with Noise: When an authorized data user wants to retrieve files, he needs to encrypt the corresponding query keyword w, and get the hash value h from the hash table. This hash value is then sent to the cloud server and used to compute the relevance scores. In order to render this hash value indistinguishable for an attacker, the cloud client should wrap it, adding some noise before sending it to the cloud server. The wrap function Wrap () will, first of all create a random number r, and then build a tuple (h1, h2) based on Algorithm 4 ($\lambda > 0 \mu > 0$):

Algorithm 4 Wrap Function with Noise

```
Input: $w, \lambda, \mu$
Output: Wrap($w$)
1:  Stem $w$ and get $\tilde{w}$.
2:  Get encrypted term $\pi_\alpha(\tilde{w})$ and hash it to get an entry $h = \psi(\pi_\alpha(\tilde{w}))$.
3:  Create a random number r.
4:  if $r < h$ then
5:      Get Wrap($w$) $= ((\lambda h - \mu r)^2, \lambda h^2 + \mu r)$.
6:  else
7:      Wrap($w$) $= \mu h^2 + \lambda r, (\lambda r - \mu h)^2)$.
8:  end if
9:  return Wrap($w$).
```

3.RESULTS

In our experiments, we use a data set of 1000 files withdifferent sizes and a VM in the cloud with Dual CPUs at2.27GHz. An android smart phone with a CPU at 1GHz sends the queries as the mobile client of TEES throughan about 8M wireless network.An Android program receives the user's input andencrypts it before getting the hash value and then wraps it into a tuple which is sent to the mobile cloud server.Another feature of this program is to retrieve the files back from the mobile cloud server and decrypt them.In addition, we implemented both TRS and PTS for acomparative purpose.As energy consumption is critical for mobile devices, we evaluate TEES energy efficiency in this subsection. Weuse a phone power monitor to accuratelymeasure the system energy consumption.Although slight changes depending upon the environment might occur, the comparison isquite accurate as controlled trials were performed. Observe that the energy consumption is reduced from0.08mAh to 0.036mAh when searching and retrieving files of size 100KB, which means that ORS saves 55%energy compared to TRS. When searching and retrievingfiles of 1MB size, the energy consumption is reduced from 0.164mAh to 0.106mAh, that means a 35% energysaving. So, TEES provides very efficient power consumption. For example, to exhaust our 1650mAh battery,ORS (of TEES) can perform 22000 retrievals while TRScould only retrieve 13000 files of size 600KB.

File Search and Retrieval Time

We compare the File Searching and Retrieval Time (FSRT) for the three schemes in this subsection as illustrated in Figure 10. We test the FSRT for different files with size ranging from 100KB to 1MB. We observe that the FSRT of PTS is the shortest since it does not have toperform any security computation. The FSRT of ORS iseffectively reduced when compared to the one of TRS.This difference is due to the advantages of the TEES design in terms of relevance score calculation offloading,and thus leads to reduction of file search and retrievalprocess. The FSRT value of ORS is very near to the one ofPTS, implying a very low cost to security on the mobiledevice. For example, TEES saves FSRT by 46% compared to TRS for files of size 100KB, and by 23% for 1MB files.The file retrieval time only depends on the file size andnetwork bandwidth. When offered a greater bandwidth, TEES becomes more efficient since downloading timeof files becomes a bottleneck of other schemes. Thedecryption time of the files is equal in all schemes andit is therefore pointless to measure it.

Table. 1. ***FSRT analyze of PTS, TRS and ORS***

	PTS	TRS	ORS
Request/Response	190ms	370ms	190ms
Stemming and Encryption	0	10ms	10ms
Hash and Wrap	0	145ms	150ms
Server file search	80ms	70ms	75ms
Client file search	0	260ms	0
Sum	270ms	855ms	425ms

The efficient FSRT of TEES is achieved by improvingthe process efficiency, since only a single round ofcommunication and relevance score calculation offload are used. The searching process is analyzed in Table 1.Without any security service, PTS (Plain Text search)does not spend any time on stemming and encryption;neither does it on hash and wrap. On the other hand,ORS and TRS provide encrypted search schemes with related overhead. As shown in Table 1, ORS can improvethe “request/response” time significantly than TRS from 370ms to 190ms (saving 180ms), and eliminate the “clientfile search” time by offloading it onto the server (saving260ms). Notice that the “sever file search” calculationworkload of ORS is 75ms, which is 5ms longer than thatof TRS. This is explained by the fact that the server takes the offloaded search calculation of the mobile user. In theother words, TEES eliminates the “client file search” timeat the cost of a little heavier “server file search” time.This proves that the offloading ishighly efficient (5ms vs.260ms). Moreover, ORS spends more 5ms on wrapping the hash value than TRS for enhancing the security. Notethat the “server file search” time of PTS is higher than the other two schemes, since the server should executestem and hash function for plaintext file search, while thehash functions are executed by the mobile data user in both TRS and ORS. Overall, ORS is secure and effective.

Throughput

The calculation offload from the mobile device to thecloud data center reduces the execution time of the relevance score calculation due to the higher server capacity.Therefore, this also greatly increases the system throughputbesides FSRT improvement; Weobserve that the file access acceleration is very effectivewhen dealing with small files as the relevance score calculation is executed more frequently. For example,on a 100KB file, the access speed is increased from 104KB/s to 194KB/s, almost doubling the throughput.The acceleration is still effective when accessing fileswith size 1MB (29.6% acceleration). The throughput ofORS is not much less than that of PTS. Implementation to achieve an encrypted search in a mobile cloud.The security study of TEES showed that it is secureenough for mobile cloud computing, while a series of experiments highlighted its efficiency. TEES is slightlymore time and energy consuming than keyword searchover plain-text, but at the same time it saves significant energy compared to traditional strategies featuring asimilar security level. Based on TEES, this work can be extended to more other novel implementations.We have proposed a single keyword search scheme to make encrypted data search efficient. However, thereare still some possible extensions of our current workremaining. We would like to propose a multi-keyword search scheme to perform encrypted data search over mobile cloud in future. As our OPE algorithm is a simple one, another extension is to find a powerful algorithmwhich will not harm the efficiency.

4.CONCLUSION

In this work, we developed a new architecture, TEES asan initial attempt to create a traffic and energy efficientencrypted keyword search tool over mobile cloud storages. We started with the introduction of a basic schemethat we compared to previous encrypted search tools for cloud computing and showed their inefficiency in amobile cloud context. Then we developed an efficientimplementation to achieve an encrypted search in amobile cloud. The security study of TEES showed that it is secureenough for mobile cloud computing, while a series of experiments highlighted its efficiency. TEES is slightly more time and energy consuming than keyword searchover plain-text, but at the same time it saves significantenergy compared to traditional strategies featuring asimilar security level. Based on TEES, this work can be extended to more other novel implementations.

SCOPE FOR THE FUTURE WORK: We have proposed a single keyword search scheme to make encrypted data search efficient. However, thereare still some possible extensions of our current workremaining. We would like to propose a multi-keywordsearch scheme to perform encrypted data search overmobile cloud in future. As our OPE algorithm is a simple one, another extension is to find a powerful algorithmwhich will not harm the efficiency.

5. BIBILIOGRAPHY

[1]. L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, 2008.

[2]. D. Huang, “Mobile cloud computing,” IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter, 2011.

[3]. J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, “Virtualized in-cloud security services for mobile devices,” in Proceedings of the First Workshop on Virtualization in Mobile Computing. ACM, 2008.

[4]. J. Oberheide and F. Jahanian, “When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments,” in Proceedings of the Eleventh Workshop on Mobile Computing Systems ACM, 2010.

[5]. A.Moffat, T. C. Bell, Managing gigabytes: compressing and indexing documents and images. Morgan Kaufmann Pub, 1999.

[6]. D. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on IEEE, 2000.

[7]. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in Advances in Cryptology- Eurocrypt 2004. Springer, 2004.

[8]. C. Wang, N. Cao, K. Ren, and W. Lou, “Enabling secure and efficient ranked keyword search over outsourced cloud data,” Parallel and Distributed Systems, IEEE Transactions on, vol. 23, no. 8, 2012.

[9]. N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy-preserving multi-keyword ranked search over encrypted cloud data,” Parallel and Distributed Systems, IEEE Transactions on, vol. 25, no. 1, 2014.

[10]. S. Kamara and K. Lauter, “Cryptographic cloud storage,” in Financial Cryptography and Data Security. Springer, 2010.

[11]. M. Li, S. Yu, K. Ren, W. Lou, and Y. T. Hou, “Toward privacy assured and searchable cloud data storage services,” Network, IEEE, vol. 27, no. 4, 2013.