Imperial Journal of Interdisciplinary Research (IJIR)
Vol-2, Issue-9, 2016
ISSN: 2454-1362, http://www.onlinejournal.in

Efficient Encrypted Data Search Using Trapdoor for Mobile Cloud Service

Shruti S Konkanagaon[1] & Doddegowda B G[2]
[1]Department Of Computer Science And Engineering ,VTU University, India
[2]Assoc Prof Of Department Of Computer Science And Engineering, VTU University, India

Abstract—*Document stockpiling in the cloud framework is quickly picking up fame all through the world. Be that as it may, it postures dangers to shoppers unless the information is encoded for security. Scrambled information ought to be viably searchable and retrievable with no security spills, especially for the versatile customer. Albeit late research has fathomed numerous security issues, the engineering can't be connected on cell phones specifically under the portable cloud environment. This is because of the difficulties forced by remote systems, for example, inertness affectability, poor availability, and low transmission rates. This prompts a long inquiry time and additional system activity costs when utilizing customary pursuit plans. This study addresses these issues by proposing an effective Encrypted Data Search plan as a versatile cloud administration.*

1. INTRODUCTION

Since cloud computing can support elastic services and provide an economical use of storage and computation resources, it is rapidly gaining popularity. With powerful cloud services, many data providers can populate their data in clouds instead of directly serving users. The cloud also allows providers to delegate important tasks such as document searches. To protect data security users need to query certain documents, they first send keywords to the original data provider. The provider then generates encrypted keywords (also called trapdoors) and returns the trapdoors to the user. The user then sends these trapdoors to the cloud. Upon receiving the trapdoors, the Cloud uses a special search algorithm, the documents and their indexes are usually encrypted before outsourcing to the cloud for searches. When to select a set of desired documents (encrypted) based on the encrypted indexes and given trapdoors. Finally, the user receives these encrypted search results and uses the private key from the provider to decrypt documents. This architecture, as depicted in Figure 1, protects data security while entitling the providers to use both the computation and storage power of the Cloud for document searches. Due to these advantages, this architecture has already been well-adopted in privacy-preserving search systems.

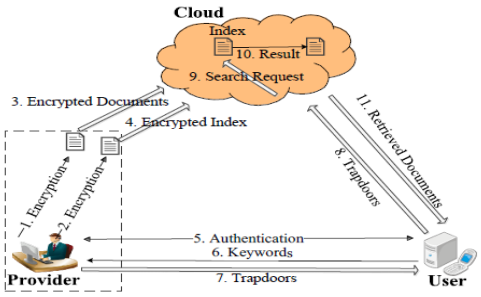

Figure 1. Traditional Encrypted Search System over Cloud

Cell phones (e.g. cell phones and tablets) were evaluated to surpass two billion development (0.3 billions for PCs) in the year 2014, which commands the general shipment of shopper hardware gadgets. Now a days, clients intensely use cell phones to demand archive look administrations.

By and large, cell phones interface with the Internet principally by means of remote systems (WiFi/3G/4G/LTE), which brings about some difficulties when contrasted with conventional wired systems.

These challenges include:

1) **Latency affectability**: these remote systems cause longer system inertness, which can back off a solitary inquiry demand if the hunt demand re-quires numerous system round outings. For instance, in the conventional configuration appeared in Figure 1, a solitary inquiry requires three round excursions and results in remarkable latency for remote correspondence.

2)**Poor availability**: Mobile gadgets are typically unequipped for keeping up a long-running association with the Cloud, generally for vitality sparing purposes. Different inquiry solicitations could bring about various re-association operations and additional authentication costs.

3)**Low system transmission rate**: Cell phones are regularly furnished with low-control transmission segments, bringing slower transmission rates. For instance, the customary framework appeared in Figure 1 requires two system round treks between

the client and the supplier (for verification and trapdoor generation) and one between the client and the cloud (for document recovery). Three round outings essentially force prominent hunt delay and intemperate system movement, which could be immoderate for a cell phone. As indicated by our estimation, a hunt demand in the customary framework could create trapdoors with a size up to 1.2MB. At the point when performing seek asks for, the trapdoor must be sent twice (step 7 and 8). In such case, security protecting ventures could prompt longer inquiry deferral and more data transmission utilization, which couldn't be moderate to versatile clients. This study concentrates on movement and pursuit time inefficiency issues over the portable cloud. We introduce a proficient Encrypted data Search (EnDAS) plan as a versatile cloud administration to handle these issues. Our sys-tem bolsters multi-catchphrase security safeguarding seek and enormously lessens system movement and pursuit delays. For system activity, EnDAS pre-figures trapdoors for basic hunt watchwords and along these lines stays away from one system round outing for re-processing trapdoor per demand.

We promote propose a few instruments to pack trap-entryways and exhibit that our pre-processed trapdoor table has a size of 0.31MB and could be adequately put away and stacked in cell phone memory. Regarding seek time, EnDAS retrofits the inquiry calculation in the cloud. In view of the parallel tree rule, we display Ranked Serial Binary Search (RSBS) calculation, which could lessen inquiry time in the cloud. Our commitments can be compressed as takes after:

1)We inspected the customary scrambled inquiry architecture as far as system activity and hunt time. Results demonstrate that the customary methodology is not appropriate in versatile cloud situations.

2)We created EnDAS to address these difficulties. Our engineering incorporates a trapdoor pressure strategy to diminish activity costs, and a Trapdoor Mapping Table (TMT) module and RSBS calculation to decrease look time.

3)We assessed the effectiveness of EnDAS in system activity and hunt time. We exhibited that with EnDAS design, we can decrease system movement by 17% to 41% and look time by 34% to 47%. The rest of this article is sorted out as takes after: Section 2 portrays the conventional encoded look sys-tem design and issues. Area 3 portrays the point by point outline of the EnDAS framework, and also dissect its system activity and hunt time proficiency. Segment 4 assesses the frameworks, and related work is secured in Section 5. Segment 6 gives conclusions and implications.

2. PROBLEM STATEMENT

In this segment, we quickly present existing security saving pursuit models and blueprint their weaknesses, both as far as hunt postpone and system activity.

2.1 Traditional Encrypted Search System

As appeared in Figure 1, the customary scrambled hunt framework over the cloud is made out of three diverse standard participants, Provider, Cloud and User, which are characterized beneath. The Provider has an arrangement of archives and their files. It means to outsource these to the cloud and let clients contact the cloud for the inquiry administration. The Cloud is a business association that gives calculation and capacity assets as virtual machines, normally known as "cloud" administrations. The User is somebody who submits catchphrases to hunt archives that contain these watchwords. In our situation, clients would utilize cell phone, for example, cell phones and tablets to submit look demands. Figure 1 subtle elements the execution stream of a conventional encoded seek over the cloud, including three primary streams: records and files transferring process (steps 1 to 4), trapdoor era process (steps 5 to 8) and report recovery process (steps 9 to 11). The heaviness of lines demonstrates the measure of information being exchanged.Reports and records transferring process: First, the supplier accountable for this stream stems all words in these archives to be put away in the cloud and holds these terms. The encryption calculation can employ the exemplary symmetric-key cryptography calculation, for example, the Advanced Encryption Standard . The recurrence of every term in the archive set is numbered and after that composed into the comparing section of the record file. At long last, the supplier encodes this list and outsources it to the cloud with the scrambled records. Generally, this file is a word recurrence table scrambled by the calculable encryption calculation.

Some studies have used the Fast Accumulated Hash (FAH) calculation to accomplish these reasons. Trapdoor era procedure: To perform a pursuit demand, the client first verifies with the supplier. Amid confirmation, the give would send its mystery key to the client to decode the records put away in cloud. Once confirmed, the client would send the inquiry catchphrases to the supplier. The supplier then processes trapdoors, regularly with FAH calculations and answers back. In such case, two round outings are required (authentication and trapdoor era) for a client to get the trapdoor for the inquiry catchphrases. Report recovery process: In this procedure, the client sends the noised trapdoor to the cloud. The cloud then evacuates commotion in the trapdoor and hunts the

records with a pursuit calculation. At the point when reports are found, the cloud positions them as per every archive's score. At that point the top-k important archives are picked and sent to the client. At long last, they are decoded and recuperated by the client. By and large, the Ranked Serial Search (RSS) calculation is picked as the pursuit calculation.

2.2 Network Traffic Inefficiency Problem

As saw in the trapdoor era handle, the trap-entryway is customarily created by the supplier to genius vide information security. In any case, in such case, the trapdoors would should be transmitted twice per demand (between the supplier and the client in addition to between the client and cloud). Figure 1 delineates the hunt stream with two net-work correspondence round

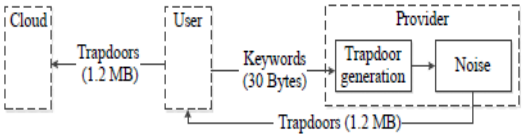

Figure 2. Trapdoor Generation in the Traditional System

outings for customary frameworks, including trapdoor era process and record recovery process. Here we couldn't care less for the authentication process and in addition transmitting target archives from the cloud to the client. So the aggregate system movement of the customary framework relies on upon system activity cost while creating trapdoors. At that point we dissect the system movement expense of the traditional framework with two system round outings, which is appeared in Figure 2.high correspondence inertness, poor availability and low system transmission rate. As indicated by Figure 2, we could ascertain the trapdoor era time by Equation The customary framework obliges clients to transmit the trapdoor twice (up to 1.2MB a period), which could without much of a stretch achieve 300ms.

Ttrr = 2 Tnet + Tgen + Tnoi (2)

where Ttrr speaks to the aggregate time delay, Tnet is as the time postponement of one round outing, Tgen is as the time deferral of trapdoor era, and Tnoi is as the time postponement of including commotion in the trapdoor. Additionally as per our estimation, the trapdoor era (steps 5 to 8 in Figure 1) time represents 59.9% of the aggregate pursuit delay. Then again, record recovery time relies on upon the pursuit calculation in the cloud. The RSS calculation is regularly used to recover reports in the cloud (steps 9 and 10 in Figure 1), which positions the records as per significance scores. In any case, it must experience a 3-level cycle to acquire related reports, and its time many-sided quality is about O(n3).

This hunt procedure is essentially wasteful, prompting a long recovery time in the versatile cloud that is not achievable for the client. This places of business these difficulties with a creative, productive encoded look plot that can be utilized over versatile cloud, as portrayed in Sections

3..ENDAS DESIGN

This area presents the outline of the EnDAS framework and retrofitted trapdoor era process in EnDAS. Contrasted the EnDAS framework and traditional system , the fundamental distinction is that network traffic is diminished by a solitary round excursion information exchange and the trapdoor pressure strategy; and the pursuit time is decreased by the RSBS calculation and the TMT module; and the processing trouble for producing trapdoors is likewise offloaded by the TMT module.

3.1 Architecture Of the Endas System

For trapdoor generation, EnDAS stores a pre computed Trapdoor Mapping Table (TMT) in mobile devices, which maps common English words to corresponding trapdoors. When the mobile device initiates A search request, the trapdoor is looked up from the table instead of being requested from the provider. This optimization saves one network round trip for the trapdoor generation. Furthermore, EnDAS also provides new algorithms to optimize and compress trapdoors to reduce network traffic to transmit trapdoors. For the search algorithm, EnDAS proposes to leverage a binary tree structure to reduce the lookup costs and thus improve the search responsiveness.

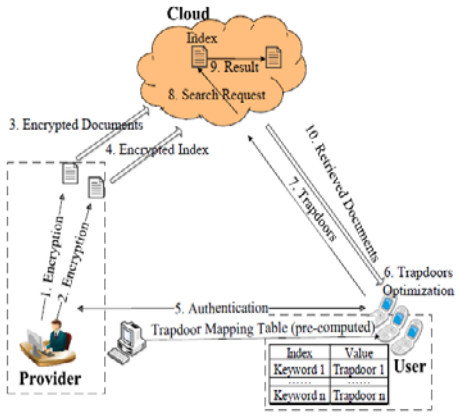

Figure 3. EnDAS system over mobile cloud

new algorithms to optimize and compress trapdoors to reduce network traffic to transmit trapdoors. Section 3.2 will elaborate the details of the EnDAS trapdoor generation process. For the search algorithm, EnDAS proposes to lever-age a binary tree structure to reduce the lookup costs and thus

improve the search responsiveness. Section 3.3 would further explain the details.

3.2 Retrofitted Trapdoor Generation Process

The retrofitted trapdoor generation process is described in this subsection, as shown in Figure 5 and Algorithm 1.This process includes the trapdoor mapping table and the trapdoor compression algorithm. To address this issue, as shown in line 8 in Algorithm 1, a lightweight trapdoor compression method is used to extract each trapdoors characteristic bits, record as well as accumulate location of each characteristic bit in order, and transmit the compressed trapdoor to the cloud.

Since these characteristic bits only occupy a small proportion in this trapdoor, the compressed trapdoor will lead to additional reduced traffic cost for transmitting the trapdoors to the cloud. We will elaborate the two aforementioned components in the following

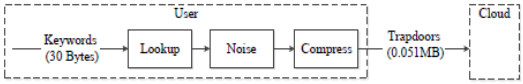

Figure 5. Trapdoor Generation in EnDAS System

3.2.1 Overview

With retrofitted trapdoor generation process, it is not necessary for an authenticated user calculate pure trap-doors (which will incur heavy computation). After a keyword is stemmed, an user can just query the trapdoor mapping table for the trapdoors, as shown in Algo-rithm 1. Since the trapdoor mapping table stores the information needed for mapping and search, the heavy computation for generating trapdoors is not needed to be conducted online. This not only avoids the recalculation if the term is found, but also reduces the number of necessary round trips from two to one. However, it is inevitable that the trapdoors of some keywords have not been stored in the trapdoor mapping table in advance. In this case, the keyword is encrypted by the user(instead of the provider (line 1 to 6 in Algorithm 1).

Then the newly retrieved or generated pure trapdoor is added with some noises from a noise set , to prevent the cloud from examining the same trapdoors, shown in line 7 of Algorithm 1.

Note that the size of the trapdoor and the noise can be too large to bring too much burden for the transmission.

3.2.2 Trapdoor Mapping Table Module

We found that there was a long calculation time from building the trapdoor on the provider side. In a traditional system, the calculation of generating a trapdoor of a given keyword is constituted by term stemming, encryption and adding noise by the provider. Among these three steps, it is shown in Figure 4 that the time of encryption stands for a significant proportion of the the total trapdoor calculation time. Figure 4 displays three columns, denoting the total calculation time for generating trapdoors for one key-word, two keywords and three keywords respectively. As shown in Figure 4, the encryption time occupies nearly 85% of the total calculation time. This is because that the encryption operation requires more computing resources than others, as it accumulates all terms together to generate a hash code. To reduce trapdoor construction time, our method ships the encryption process from the online approach to offline. Furthermore, the trapdoor generation process utilizes a Trapdoor Mapping Table (TMT), which stores a large amount of frequently-used trapdoors (since an English vocabulary of just 3,000 words provides coverage for around 95% of common texts [16], here we assume a proper size of keywords is about 3,000 words) calculated offline.

The key for this trapdoor mapping table is a term from stemmed keywords, while its value corresponds to encrypted terms (a pure trapdoor without any noise).Next we analyzed the availability of the TMT module. According to our measurement, we found that in 20,000 trapdoors, the size of more than 80% of trapdoors ranges from 20 to 60 bytes.

That is, encrypted keywords have a small size. So we selected 5,893 different words (in-cluding 3,000 common words and 2,893 rare/uncommon words as keywords to be encrypted, and then stored them in TMT module. We achieve that the actual size of TMT was about 0.31 MB, according to our measurement. Although some rare words are not in the TMT module, users rarely search documents with them, and therefore we can fully ignore these words. Figure 3 indicates that this TMT requires only one transfer to the user, while its size is smaller than the size of one noised trapdoor (0.4 MB) from the provider to the user (step 7 in Figure 1). That is, the TMT module saves not only traffic but also search time. TMT module does not require the provider to compute trapdoors through expensive communication between the provider and the user, while it only requires the user to look up trapdoors, avoiding re-computing trapdoors. It reduces network round trips for trapdoor generation from two to one, which is shown in Figure3.We analyze the performance of EnDAS in search time when generating trapdoors. Utilizing TMT module,

En-DAS has only one network round trip used to search target documents, which is shown in Figure 3. The total search delay of EnDAS when generating can be calculated by Equation 3

$T_{ssr=1} T_{net}+T_{look}+T_{noi}+T_{comp}$ (3)

where T_{srr} represents the total time delay, T_{net} is as the time delay of one round trip, T_{gen} is as the time delay of trapdoor lookup, T_{noi} is as the time delay of adding noise in the trapdoor, and T_{comp} is as the time delay of compressing noised trapdoor. Comparing Equation (3) with Equation (2), we find that $T_{srr} < T_{trr}$. This is because EnDAS is only required to look up, noise and compress trapdoors, rather than encrypting trapdoors. In essence, the noise method is that accumulating noises chosen from the noise set after each trapdoor in order would consume a few time. On the other hand, the trapdoor compression method also causes a few time, and the reason will be elaborated in Section 3.2.3. Looking up trapdoors in the TMT module spends so few computing resources that we can fully ignore it. According to our measurement, encrypting trapdoors in the traditional system costs much longer calculation time (85% of total time for trapdoor generation) than other operations on trapdoors (e.g. noise).

4.Trapdoor Compression

The key idea behind this trapdoor compression method is that we utilize the location of each trapdoor's characteristic bit to represent this trapdoor, since characteristic bit 0 can show all the features of the trapdoor and also occupy a much smaller proportion compared with non-characteristic bit 1.

The numbers of 0 and 1 is almost

$$\frac{\frac{(p+1)\times r}{2^d}}{r-\frac{(p+1)\times r}{2^d}} = \frac{(p+1)}{2^d-p-1}.$$

From this result, we can deduce that the number of 0 will be much less than the number of 1. This is a very important attribute.

Algorithm 1 Trapdoor Generation Process

Input:
Keyword: K
Hash function in FAH algorithm: $H()$
Mapping function in FAH algorithm: $G()$
Noise set: $\Theta = \{\varepsilon_1, \varepsilon_2, \ldots \varepsilon_p\}$
Output:
Index: Compressed trapdoor $\tau_t{}'$
1: Extract the term t from K.
2: **if** the term t hits in the TMT module **then**
3: Obtain its pure trapdoor without any noise.
4: **else**
5: Hash it by $H()$ and get its l-bit hash code $\tau_t = H(t)$;
Map τ_t to $\bar{\tau}_t = \{0,1\}^r$ by $G()$, which contains r bits
6: **end if**
7: Choose q noises from the noise set Θ to build a subset $\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots \varepsilon_q\}$, and accumulate it with $\bar{\tau}_t$ to get $\bar{\tau}_t \bigwedge \varepsilon$.
8: Calculate the location of each characteristic bit 0 in $\bar{\tau}_t \bigwedge \varepsilon$ by utilizing an m-bit $\{0,1\}$ codes to record this location ($r = 2^m$), accumulate values of locations in order $\underbrace{\{0,1\}^m \bigwedge \{0,1\}^m \bigwedge \ldots \{0,1\}^m}_{f}$, get a compressed trapdoor $\bar{\tau}_t{}' = \{0,1\}^{f\times m}$ (f as the number of characteristic bits).
9: **return** $\tau_t{}'$.

This analysis demonstrates that adding noise into the trapdoor still ensures a small proportion for characteristic bit 0. ignored. We easily get each characteristic bit's location value and then accumulate them in order. This new accumulated code created by the locations of characteristic bits is considered as the compressed trapdoor, which is finally sent to the cloud. In fact, the trapdoor compression method keyword. The process of encrypting the keyword is that all the characteristic bits 0 and non-characteristic bits 1 are required to be calculated twice, including dividing and mapping. And this keyword encryption process is the same as the process of term encryption when encrypting index in the provider side.

Efficient Search Algorithm.

Document Index Construction

$$A = (I_1, I_2, \ldots, I_N) = \begin{pmatrix} RS_{1,1} & RS_{1,2} & \ldots & RS_{1,N} \\ RS_{2,1} & RS_{2,2} & \ldots & RS_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ RS_{T,1} & RS_{T,2} & \ldots & RS_{T,N} \end{pmatrix} \quad (4)$$

We compute the score for a particular document c as the sum of all elements within its index I_c, which is

$$Score_c = \sum_{t=1}^{T} RS_{t,c} \quad (5)$$

5.Ranked Serial Binary Search Algorithm

Algorithm 2 Ranked Serial Binary Search (RSBS) algorithm

Input:
Noised trapdoors (one per search keyword): $\bar{\tau}_1{}', \ldots, \bar{\tau}_e{}'$
Encrypted document indexes: $A = \bar{I}_1' \cdots \bar{I}_N'$
The number of documents to return: k
Output:
Top-k documents that best match the search request: $D = \{D_1, D_2, \cdots, D_k\}$
1: $Scores = zeros(0, N)$ // create an array of N zeros
2: **for** $i := 1$ **to** N **do**
3: **for** $n := 1$ **to** e **do**
4: Score[i] ← Score[i] + bsearch($\bar{\tau}_n{}'$, $\bar{I}_i'$, 1, s_i) // search if the keyword appears in any of the s slices of the document
5: **end for**
6: **end for**
7: sorted, indices = sort(*Scores*) // sort the score array and get the indices or old element in the sorted array.
8: $D \leftarrow indices[0 : k-1]$ // get the top-k documents
9: **return** D

To evaluate the EnDAS system, we implemented our system on the private cloud with Open stack Essex [19] from our lab. We rented a virtual machine with 8G memory for the cloud. We also implemented the RSBS algorithm, written as a python program, to search and return the retrieved documents to the user. Here, the user utilized a mobile device utilized an Android tablet with a Cortex- A9 Quad 1.4GHz CPU, and 2GB memory. The tablet is connected to a mobile network with 72Mbps rate. The trapdoor mapping table is pre-

computed on a PC and uploaded to the mobile device before experiments, which consumes 0.31MB of device storage.

Search Time Evolution.

To reduce the search time and improve the calculation efficiency, we utilized the TMT module and the RSBS algorithm in the EnDAS system. In this part, we first evaluate the overall search time and its breakdown.

Network Traffic Evolution.

EnDAS system, which benefits from the trapdoor compression method and the TMT module, we reduced network traffic significantly. Next we evaluate and analyze the overall system network traffic reduction and the performance of the trapdoor compression method.

6.CONCLUSION

In this work, we proposed a novel encrypted search system EnDAS over the mobile cloud, which improves network traffic and search time efficiency compared with the traditional system.

We started with a thorough analysis of the traditional encrypted search system and analyzed its bottlenecks in the mobile cloud: network traffic and search time inefficiency. Then we developed an efficient architecture of EnDAS which is suitable for the mobile cloud to address these issues, where we utilized the TMT module and the RSBS algorithm to cope with the inefficient search time issue, while a trapdoor compression method was employed to reduce network traffic costs. Finally our evaluation study experimentally demonstrates the performance advantages of EnDAS.

7. REFERENCES

1] K. Nyberg, "Fast accumulated hashing," in Proc. Int. Workshop Fast Softw. Encryption (FSE), Feb. 1996, pp. 83–87.

2] Nyberg and Kaisa, "Commutativity in cryptography," in Proc. Int. Workshop Funct. Anal., 1995.

3] J. Benaloh and M. De Mare, "One-way accumulators: A decentral-ized alternative to digital signatures," in Advances in Cryptology-EUROCRYPT 1993, 1994, pp. 274–285.

4] C. Orencik and E. Savas¸, "An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking," Distrib. Parallel Databases, vol. 32, no. 1, pp. 119–160, Mar. 2014.

5] P. Wang, H. Wang, and J. Pieprzyk, "An efficient scheme of common secure indices for conjunctive keyword-based retrieval on encrypted data," pp. 145–159, 2009.

6] S. Gendreau, "How many words do i need to know? the 95/5 rule in language learning, part 2/2," http://www.lingholic.com/ how-many-words-do-i-need-to-know-the-955-rule-in-language/ -learning-part-2.