The ISC Int'l Journal of Information Security

ISeCure

January 2012, Volume 4, Number 1 (pp. 41–50)

http://www.isecure-journal.org

Private Key Based Query on Encrypted Data☆

Hammad Afzali [1,*], Hamed Nemati [2], and Reza Azmi [3]

[1] *Operating System Security Lab, Alzahra University, Tehran, Iran*
[2] *School of Computer Science and Communication, KTH University, Stockholm, Sweden*
[3] *Engineering & Technology Department, Alzahra University, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 26 November 2011
Revised: 8 March 2012
Accepted: 19 March 2012
Published Online: 25 June 2012

Keywords:
Encryption, Query on Encrypted Data, Private Key Search, Privacy Preserving.

ABSTRACT

Nowadays, users of information systems have inclination to use a central server to decrease data transferring and maintenance costs. Since such a system is not so trustworthy, users' data usually upkeeps encrypted. However, encryption is not a nostrum for security problems and cannot guarantee the data security. In other words, there are some techniques that can endanger security of encrypted data. Majority of existing methods for encrypted data management have some critical defects such as cryptanalysis attacks, encryption/decryption overhead, and inefficient data storing and retrieval. In this paper, at first we propose a prototype model of private key based search on encrypted data. Then we try to improve it significantly to meet security requirements. Our main goal is to offer a practical method of querying arbitrary words on encrypted data using a minimal trust model. Moreover, we present a model for balancing between performance and security based on user's requirements. In comparison with other methods, query response time is improved and the probability of statistical deductions is reduced.

1 Introduction

Nowadays, wages of transferring and maintenance of information are increasing. In order to reduce the mentioned wages, users prefer to store their data in a central and secure place, for example a central server or a Database As a Service provider (DAS), and access it remotely. In this model, users submit their requests to clients. Then after, clients send the proportionate query to the server [1–3]. Prevalence of central service providers encourages users to store their data which may consist of sensitive materials such as emails, financial data, and personal health records in centralized data warehouses. As a result, reliability, scalability, cost effectiveness, and easy maintenance are enjoyable [4–6]. While centralized storages are preferred protection of users' confidential data (relational, text, XML, etc.), security is very important and usually satisfied by encryption before outsourcing. After storing encrypted data, its owners may want to retrieve information such as an email consisting of especial keywords [7, 8]. Although data encryption has security advantages, managing encrypted data for constructing a proper response to user's query remains so challenging. Its intricacy comes from offering a secure method for information storing and retrieval, a suitable encryption algorithm, the architecture of key management and distribution, integrity control, retrieving all files containing the desired keyword, and eliminating unnecessary traffic [3, 9, 10]. Query on encrypted data, based on probing one or several specific words are classified in the following four groups:

(1) private key based techniques based on collating a word with several words in a particular

☆ This article is an extended/revised version of an ISCISC'11 paper.

* Corresponding author.
Email addresses: afzali@ce.sharif.edu (H. Afzali), hamednemati@srtu.edu (H. Nemati), azmi@alzahra.ac.ir (R. Azmi).

ISSN: 2008-2045

encrypted text [7, 11, 12].
(2) Conjunctive keyword search used for collation of several words [2, 3, 13].
(3) Other techniques for passive data querying, similar to previous category, use public key to bundle keywords. These mechanisms aim at preserving the privacy of data receiver [9, 14, 15].
(4) Search techniques based on secure indexing [7, 16].

It is Noticeable that most of these approaches are stemmed from secure indexing. Computation overhead and insecurity against cryptanalysis attacks are two main problems of these techniques [4, 5, 7, 9–11, 13, 17]. A proper query method should hinder disclosure of information to unauthorized entities [18]. On the other hand, query on encrypted data should be resistant against attacks that compromise data protection mechanisms [17].

In the following sections, fundamental of query on encrypted data is discussed. These methods usually try to introduce an encryption model, an encrypted data storage scheme, a method of secure querying on encrypted data, and the quality of server's response.

In this paper we propose a basic scheme for obviating overhead of computation process by reducing encryption and decryption costs. By analyzing our primary scheme, we try to improve the previously mentioned security requirements. Specially, our improved model provides no obligation for user to trust client side agents, as previous approaches [1, 3, 4, 6, 8–10, 14, 15, 18–21] did. In addition, we propose the possibility of balancing between performance and security.

The remainder parts of this paper are organized as follows. Section 2 outlines the background of querying on encrypted data. Section 3 describes our model for encrypted data probing based on private key. An evaluation of this model is presented in Section 4. Section 5 briefly reviews related work. Finally, we conclude in Section 6 with some thoughts as future work.

2 Query on Encrypted Data

In this section, we try to clarify challenges of search on encrypted data. The prior private key based techniques (one or multi keywords) are also evaluated.

2.1 Private Key Based Approach

In private key based approach, data encryption can be performed using a single key for each word or for each text. Although the security of the former method, i.e., using a single key for each word, is better, it suffers from excessive overhead. While exerting an access control mechanism with the granularity of a file to share the encrypted data depends on the existence of a key for each encrypted file, we should keep in mind that management of excessive number of keys is intricate [5, 20].

Due to some disadvantages of the previous method, in the following we introduce fundamentals of encrypted data retrieval methods based on the second access model, i.e., using a single key for each text, originating from [7]. Using a single key for each text, these methods retrieve the encrypted file on the server without using a public key or revealing the context of the file.

2.2 Storing and Restoring Methods

Before explaining the basic storage method, we clarify that this algorithm considers the following materials as its assumptions: texts with l words of length n, owner private key k_p, text bundle key k_c, and a pseudo random function $F_{k_c}(s)$. Each text is encrypted in the user's client and the encrypted hyper-text will be stored on the server for replying further user queries.

As it is perceivable from Algorithm 1, in the first step a random bit string s_i with the length of $n-m$ bits is generated. Then a bit string with the length of m, corresponding to s_i, is generated (step 4). In the next step, s_i and F_{k_c} (s_i) are merged together and produce a bit string with the length of n, called t_i. Each word of encrypted text is encrypted by k_p. Finally,its index, $x_i \bigoplus t_i$, is added to the index set in the server to facilitate responding queries.

After storing encrypted data, for searching a word, user's client sends the index of desired word to the server. Algorithm 2 shows the pseudocode of search procedure on encrypted data. Server will obtain t by calculating $x_i \bigoplus c_i$ for all of the text's indices. If the value of F_{k_c} for $n-m$ initial bits of t is equal to next m bits, the probed word has been found with time complexity of $O(l)$.

2.3 Evaluation Criteria

In this section, we discuss some security measures of previous private key based methods. Looking into several existing methods such as [7, 11, 12], we can list the main characteristics of these methods as follow:

- **Brute-force attack prevention**: encrypted data is blinded using a pseudorandom function.
- **The possibility of search for every files or texts (such as multimedia)**: encryption algorithm is independent from context.
- **The possibility of cryptanalysis**: the lack of

Algorithm 1 Private Key Based Encryption

Cipher$(D(w_1, w_2, \ldots, w_l), k_p, k_c)$

1: Generate Pseudorandom $s_1, s_2, \ldots, s_l$ using stream cipher // (s_i has n-m bits.)
2: $I_D = \emptyset$
3: **for** all $w_i \in D$ **do**
4: Compute $F_{k_c}(s_i)$ seeded on k_c // F's output has m bits. k_c is the words collection key.
5: Compute $t_i = < s_i \parallel F_{k_c}(s_i) >$ //t_i has n bits.
6: Compute $x_i = T(k_p, w_i) = E_{k_p}(w_i)$ // k_p is the user private key.
7: Compute $c_i = x_i \bigoplus t_i$
8: $I_D = I_D \cup c_i$
9: **end for**
10: **return** I_D

Algorithm 2 Private Key Based Search

Search(I_D, x_i)

1: **for** all $c_i \in I_D$ **do**
2: **If** $c_i \bigoplus x_i$ is of the form $< s \parallel F_{k_c}(s) >$
3: **return** D
4: **end for**
5: **return** $\emptyset$

keywords encryption in communication channel and using unsafe algorithms result in the possibility of cryptanalysis.

- **Calculation overhead**: encryption operation is intricate.
- **Requiring high bandwidth and storage at server side**: storing high volume of meta data along with main cipher text increases the memory and bandwidth usages.
- **Restriction in words' length**: there is no facility to encrypt words with different sizes.
- **Information disclosure**: analysis of words which are not meaningless could reveal some information [4, 18].

3 Proposed Method

In this section we present our method for querying encrypted data based on private key. Our goal is to offer a secure method to efficiently search an arbitrary word on encrypted data by reducing the trust level of previous methods [1, 7, 9, 11, 12, 14, 18]. At first, we present a prototype to describe the most important security and performance challenges related to this model. Then by evaluation of this primary scheme, we identify security and performance defects, and eliminate them in a final model. In each section, we describe storing and querying procedures in addition to their evaluation. Hereafter in this paper, we use following notations to denote our architecture elements:

- D: owner's data
- D_c: blinded data by owner
- D_s: blinded data by client
- l: number of owner's plain-text words.
- w_i: a word from owner's plain-text data
- H(w_i): output of hash function for wi
- H(w_i, k_p): output of a hash function for $w_i \bigoplus k_p$
- k_p: user's master key
- Q: modulo of a congruous function
- w_{ic}: encrypted value of w_i in the client
- w_{is}: stored value in the server corresponding to w_{ic}

Similar to block ciphers, our approach encrypts each word independent of previous words. Encryption algorithm assumes that plain-text words are categorized in some moulds, for example 8. Each word consists of at least 1 character, and at most 16 characters. Since the most frequent words length is 4 and 5, categorizing the text into 8-character words will be useful for expediting the search procedure. These moulds are:

- 32 bits for words with 1 or 2 character
- 48 bits for words with 3 characters
- 64 bits for words with 4 characters
- 80 bits for words with 5 characters
- 96 bits for words with 6 characters
- 128 bits for words with 7 or 8 characters
- 192 bits for words with 9 to 12 characters
- 256 bits for words with 13 to 16 characters

3.1 Threat Model

We assume the server is not honest and is able to learn and infer the data contents or searched words. The server could also collude with users to derive additional information about users' queries and the encrypted data. Although having trust to the client has been widely accepted, we assume that the data owner has no trust to clients. Therefore, only the data owner, besides authorized end users, can access plain texts.

3.2 Primary Scheme

3.2.1 Data Storing

In the primary scheme we use a blinding function instead of an encryption algorithm for data storing. At first, owner generates 8 private keys k_p commensurate with 8 moulds. Then using SHA2 algorithm (256 to 128 bit), the digest of each word is generated. Referring to Algorithm 3, owner mixes plain word, its digest, and his private key k_p based on *OwnerCipher* in Algorithm 3. As a consequence, each word is hidden as a meaningless form w_{ic}. Then using *ClientCipher* in Algorithm 3 another steganography operation is exerted at client side for producing w_{is}.

Algorithm 3 Storage Procedure in the Primary Scheme

OwnerCipher$(\mathrm{D}(w_1, w_2, \ldots, w_l), k_p)$

1: **for** all $w_i \in D$ **do**
2: $w_{ic} = w_i + H(w_i) \times k_p$
$//\mathrm{H}(w_i) = \mathrm{SHA1}(w_i)$:256 bits

ClientCipher$(D_c(w_{1c}, w_{2c}, \ldots, w_{lc}), k_c)$

1: **for** all $w_{ic} \in D_c$ **do**
2: $w_{is} = w_{ic} + H(w_{ic}) \times k_c$
$//\mathrm{H}(w_{ic}) = \mathrm{MD5}(w_i)$: 32bit

Algorithm 4 Search Procedure in the Primary Scheme

UserSearch$(H(w_{ic}))$

1: **for** all $w_{is} \in D_s$ **do**
2: $T = w_{is} \bmod H(w_{ic})$
3: **if** $\mathrm{H(T)} = \mathrm{H}(w_{ic})$ **then**
Return I_{D_s}
4: **end for**

3.2.2 Data Querying

As shown in Algorithm 4, when users (the owner or other authorized persons) submit a query, the server machine compares digest values with the obtained value from the client (based on *UserSearch* in Algorithm 4). In case of equality of these values, the server sends ID's of data as well as encrypted data (if requested by the user). After that, the client machine calculates w_{ic} and sends it to the user. The user can get w_i value by calculating (w_{ic} mod k_p). Albeit in this scheme, the goal of w_{is} formula is the bandwidth reduction. Therefore, for situations that there is no requirement for this feature it is possible to store w_{ic} and withhold the digest operation.

Based on the data storing and querying procedures, we can illustrate our architecture in Figure 1. This method consists of three steps of encryption. By going through these steps and without using ordinary complex encryption algorithms, we reach our goals to provide the confidentiality of information with untrusted clients.

It must be noted that in the data storing step, only the owner can send an encrypted text for storing; while the data querying step can be done by the owner or another privileged user who has a key corresponding to a special kind of data.

3.2.3 Evaluation

In order to determine weaknesses and strengths of our primary scheme, in this section we evaluate its structure. As clients cannot access the users' key and

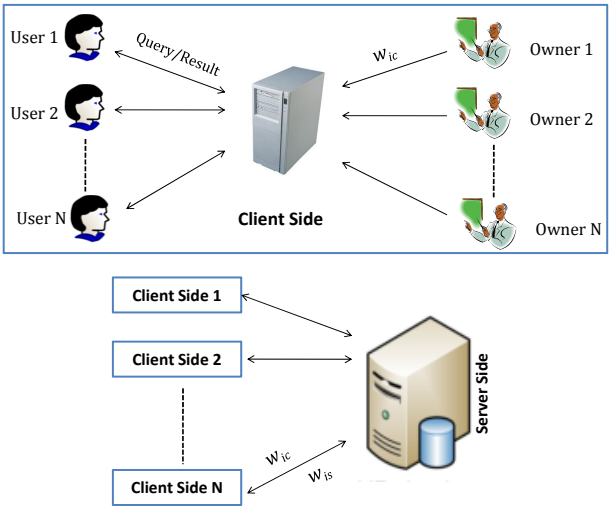


Figure 1. System architecture for private key based query on encrypted data

words' digest, they would not be able to discover the plain-text data. This can help us to omit the obligation of holding trust to clients. To put it in other words, by eliminating the necessity of trust to client devices, our model is able to obviate the requirement of some applications for the precise identification of modifier agent.

Our primary scheme suffers from the weakness of cryptanalysis attacks, which depends only on nonlinear blinding function. In this model the lack of requirement for using encryption functions such as AES, accelerates the search operation. Moreover, operations used to calculate w_{is} reduce the bandwidth and accelerate the query response. It is worth mentioning that the blinded value w_{ic} seems to be inane and does not add any security to *ClientCipher* output in Algorithm 3. Therefore, there is no point to use strong functions such as SHA2 in this step and digesting can be MD5 when $H(w_{ic})$ is going to be sent to the server. In the other word, for situations that each word is searched, this function could be useful at most once.

3.3 Final Scheme

Evaluation of the primary scheme demonstrated main reasons of inefficiencies. To alleviate such problems, a final scheme is presented such that it can frustrate cryptanalysis attacks which try to breakdown the blinding function of *ClientCipher* in Algorithm 3. Also, it could persist against attacks which try to discover value of w_{ic} from the server.

In this scheme, we try to rule out cryptanalysis attacks and increase the security of the model by using the residue calculation theorem as an NP hard problem. Moreover, we speed up data querying by

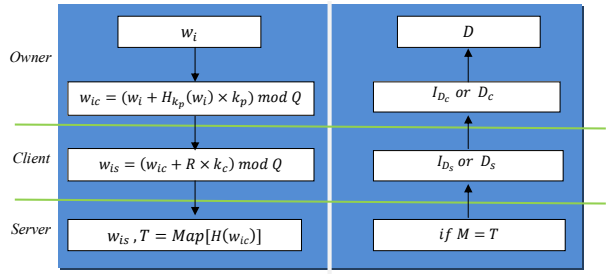


Figure 2. Encryption and search operations at three levels in the final scheme

tolerating an overhead related to copious data storage.

3.3.1 Data Storing

As shown in Algorithm 5, in data encryption step owner selects 8 private keys commensurate with each format of data. He also produces 8 modulo (Q) for each of them. Before residue calculations, the value of $w_i \bigoplus k_p$, and then the value of w_{ic} (according to *OwnerCipher* in Algorithm 5) are calculated. In Algorithm 5, Q_2 is a constant 256-bit value, different from k_p. So, the number of r_1 bits, based on the number of k_p bits, are selected in a way that Q has 256 bits. After blinding data by the owner, client produces w_{is} using a private key k_c and a random value. This value along with the result of $\mathrm{H}(w_{ic})$ are sent to the server. In this model we suffer from some overheads related to maintenance of words' digest and a hash table.

To reduce the overhead of generating random numbers and residue calculations, we can compute the value of w_{is} using Equation 1. In order to expedite search operation we use a server side hash table. Therefore, the values of $\mathrm{H}(w_{ic})$ are mapped to a smaller domain and I_D of all corresponding data are stored in a conforming field.

$$\begin{aligned} w_{is} &= (w_{ic} \times k_c) \bmod Q \quad or \\ w_{is} &= (w_{ic} + k_c) \bmod Q \quad or \\ w_{is} &= w_{ic} \bmod Q \end{aligned} \tag{1}$$

3.3.2 Data Querying

In order to search the encrypted data upon the database server the digest value of data is sent to the server. Then server would be able to locate the desired data and send it back to the requestor based on *UserSearch* in Algorithm 6. If the value is not found in the hash table, a negative response will be sent to the requestor. It is worth mentioning that positive responses are not unique. The main operations of the system, including data storing and data querying, can be summarized in Figure 2.

Algorithm 5 Encryption Procedures in the Final Scheme

OwnerCipher$(\mathrm{D}(w_1, w_2, \ldots, w_l), k_p)$

1: **for** all $w_i \in D$ **do**
2: $w_{ic} = (w_i + H(w_i, k_p) \times k_p) \bmod Q$
// $H_{k_p} = \mathrm{SHA1}(w_i \bigoplus k_p)$: 256 bits
// $Q = r_1 \times k_p \bmod Q_2$
3: **endfor**

ClientCipher$(D_c(w_{1c}, w_{2c}, \ldots, w_{lc}), k_c, R)$

1: **for** all $w_{ic} \in D_c$ **do**
2: $w_{is} = (w_{ic} + R \times k_c) \bmod Q$
// R is a random value
3: $M_{ic} = \mathrm{Map}(\mathrm{H}(w_{ic}))$
4: **endfor**

Algorithm 6 Search Procedure in the Final Scheme

UserSearch(M)
// $\mathrm{M} = \mathrm{Map}(\mathrm{H}(w_{ic}))$

1: **for** all $(T, w_{is}) \in D_s$ **do**
2: $T = w_{is} \bmod H(w_{ic})$
3: **if** T = M **then**
4: **return** I_{D_s}
5: **end for**

4 Evaluation

Having assessed the effectiveness of the final scheme, in this section we evaluate the proposed encryption and search algorithms. At first, we discuss the probability of a successful attack against the final scheme and the performance affection. Next, we present a specific model of balancing between security and performance regarding user's requirements.

4.1 Algorithm *OwnerCipher*

As we mentioned in Section 3.2.1, *OwnerCipher* in Algorithm 5 uses reminder theory as an NP hard problem, 256-bit module Q and a private key commensurate with word's length. After multiplying the word's digest (128 bit) and the key value, a key with length of 170 to 380 bits will be produced. As values of w_{ic} are mapped in a space with 256 bits, some words are repetitious and there is false-positive quale in results of queries. In order to increase the security of plain-text retrieval from the blinded one, in the final scheme we used $\mathrm{H}(w_i)$ value instead of $\mathrm{H}(w_i \bigoplus k_p)$. According to *OwnerCipher* in Algorithm 5, data owner can retrieve w_i if and only if Q is a multiple of k_p:

$$Q = r_1 \times k_p \tag{2}$$

$$w_{ic} = (w_i + H \cdot k_p) \bmod Q$$
$$H = H(w_i, k_p) \quad (3)$$

If r is extracted from Equation 2, according to Equation 3, the expiration for Equation 4 can be inferred as follows:

$$w_{ic} - w_i = H \cdot k_p - m \cdot Q = k_p(H - m \cdot r_1) \quad (4)$$

Equation 5 and Equation 6 are deducible from Equation 4:

$$m = \frac{w_{ic} - w_i}{k_p \cdot r_1} + \frac{H}{r_1} = E + H/r_1 \quad ; \quad 0 < E < 1 \quad (5)$$

$$w_{ic} - w_i = k_p(H - H - E) = k_p E r_1 \quad (6)$$

If Q is supposed to be public, according to Equation 2, security of k_p depends on the security of r_1 and its state-space (for example 216). Considering Equation 5 and Equation 6, we can say that the security of private key (k_p) against cryptanalysis will be reached to 2^{64+r}. If r_1 is consisted of few bits, the possibility of k_p disclosure increases. Since, calculation of Q is based on a direct and first-order relation, increasing the number of r_1 bits can not prevent the crypt analysis attacks. To address this problem, we can use an NP-complete problem as presented in Equation 7.

$$|Q| = |Q_2| = 256\ ,\ (|Q| : \textit{the number of } Q\textit{'s bits})$$
$$32 < |w_i|\ ,\ |k_p| < 256\ ,\ |H| = 128 \quad (7)$$

Referring to Equation 7, we can say that revealing Q_2 needs polynomial time in S_1 space. Also, an attacker can expose the value of $w_i c$ in S_2 space with polynomial time Equation 8. Referring to Equation 7, we can say revealing Q_2 needs polynomial time in S_1 space. Also, an attacker can expose the value of w_{ic} in S_2 space with polynomial time.

$$S_1 = 2^{|Q_2|} = 2^{256}$$
$$S_2 = max(2^{|w_i|}, 2^{|k_p| \cdot |H|}) \rightarrow 2^{160} < S_2 < 2^{384} \quad (8)$$

As mentioned in the previous section, our approach reduces trusted parties and the users only provide security of k_p and Q_2. Clients should also protect blind text. Security of hash function depends on solving an NP-complete problem in S_1 and S_2 space. Moreover, the use of NP-complete problem and digest of blinded data has improved the statistical disclosure control. This algorithm could be expedited by increasing the number of moulds. Also, increasing the number of moulds could obviate the restriction of words with equal length and plethora of meaningless characters.

4.2 Algorithm *ClientCipher*

In Algorithm 5 (*ClientCipher*), in order to cope with security problems, we use an NP hard problem. If the server cannot obtain the client's key or the random value R, it can utilize cryptanalysis attacks to discover blinded data by user's client (D_s). In case H(w_{ic}) is used instead of the random value (R), w_{ic} value can be calculated by a simple reminder computation. It is important to note that in order to prevent cryptanalysis attacks, we do not send w_{ic} to the server and do not use Equation 7. Using the mappings of H(w_{ic}), we can reduce the amount of reserved memory for each data and accelerate search operations. Moreover, the false-positive feature will be observed. In this model we need the following computations for storage of each word on the server:

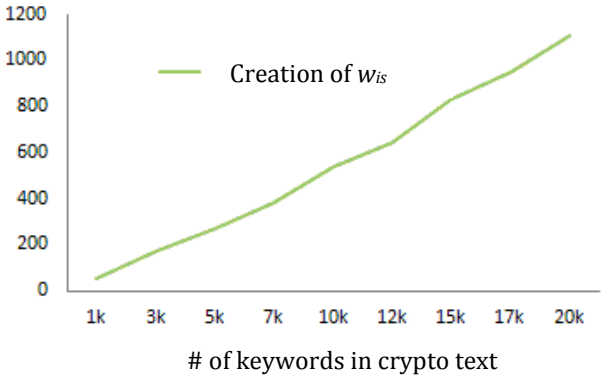
- the calculation of word length
- an XOR operation
- two summation operation
- two multiplication operation
- two digest calculation
- two residue calculation
- two random number generation
- one digest word mapping
- the transmission of blinded word w_{is} and table of mapped values

On the other hand, we need the following steps for search operations:

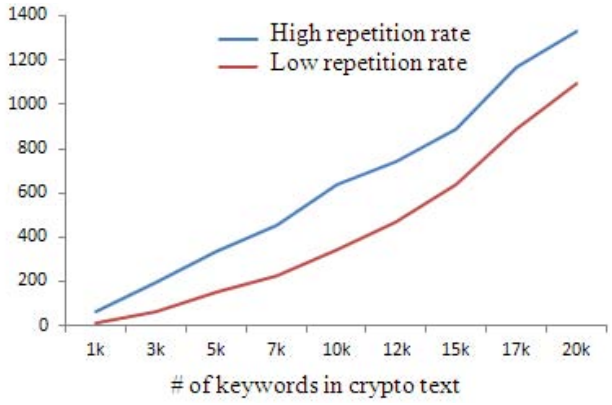
- locating the value of M in the hash table
- transmission of blinded data or its corresponding I_D

The proposed model has a positive effect on the search operation speed as well as bandwidth usage by words' digest transmission. Moreover, by using a hash table, a query is replied in the form of false-positive in a time order of $O(1)$. In comparison with the initial model, the volume of required operations for blinded data analysis at the server side is reduced.

In the case of reluctancy for usage of digested values map, search operation for l words with N characters is performed with a time order of $O(Nl)$. This search operation is faster than other similar methods, in that digest producer algorithms such as SHA2 are faster than encryption algorithms such as AES. As it is perceivable form Figure 3, all queries on encrypted data almost have linear time. Also, words with high frequency are searched faster. In comparison with other similar methods [4, 7, 11, 16–18], the computational overhead of random number generation and the storage overhead related to the hash table, decrease the performance of the proposed method. However, expurgating this overhead has not significant effect on the security of this method.



(a) Encryption time for w_{is} in the final scheme



(b) Server side query execution time without using hash table

Figure 3. Simulation results

Based on our review, establishing a balance between security and performance could be provided by selection of suitable blinding and storing algorithms. According to Table 1, users are able to choose one of the main data blinding functions (w_{ic}) at the client side and their storage methods (w_{is}) at the server side.

We have listed appropriate algorithms, proportionate to some applications, in Table 2. We can say that proper selection of b-3 and a-6 algorithms guarantee the system security and performance. Considering the proposed method evaluation for different applications, achievable security and performance features of the final model are as follows:

- **Possibility of balancing between security and performance based on user requirements**: choosing arbitrary algorithms according to Table 2.
- **Prevention of statistical deduction**: using digest producer equation and non-linear equation in blinded word with the normalization of statistical frequencies.
- **Impossibility of a word existence by querying in a dataset and cryptanalysis attacks**: impossibility of blinded word calculation without knowing k_p and Q_2.
- **Lack of obligation of trust to client side**: blinding function utilization by data owner
- **Eligible Query Time**: usage of hash table in order to reduce expected delay
- **Dictionary attack prevention**: Combination of words and its digest
- **Lack of restricting on words' length**: information is divided into some moulds that can be extended
- **Reducing the memory, bandwidth, and computation overhead**: achieved by utilizing message digest and reducing metadata volume, in comparison with other complex algorithms

Table 1. Blinding plain text (w_{ic}) and its server side storage (w_{is})

w_{ic}	w_{is}
1: $w_i + H(w_i) \times k_p$	4: $w_{ic} + H(w_{ic}) \times k_c$
2: $w_i \bmod Q$	5: w_{ic}
3: $(w_i + H_{k_p}(w_i) \times k_p) \bmod Q$	6: $(w_{ic} + R \times k_c) \bmod Q \& M$
$a: Q = r_1 \times k_p$ $b: Q = r \times k_p mod Q_2$	$a: M = H(w_{ic})$ $b: M = Map(H(w_{ic}))$

Table 2. Appropriate algorithms in different applications (NI: Not Important)

Application	w_{ic}	w_{is}
Cryptanalysis Prevention at Client Side	3_b	NI
Cryptanalysis Prevention at Server Side	3	6_b
Reducing Response Time at Server Side	NI	6
Bandwidth and Storage Reduction	1 or 2	5

5 Related Work

This paper mainly resides in the area of secure query on encrypted data using private key. In this section, we introduce some previous work in the area of private key based search.

Majority of approaches for query on encrypted data use secure indexing idea. Goh [16] used the idea of bloom filter [22] to introduce a secure index. After that, several researches have been conducted based on secure indexing [2, 8, 11, 12, 14, 15, 21]. However, most of them suffer from the bloom filters weakness [1, 3].

The earliest practical approach for providing secure search on encrypted data was proposed by Song *et al.* [7]. They used trapdoors for enabling server to match a specific cipher text. However this approach suffers from the search complexity and performance overhead.

Privacy preserving in management of encrypted data was introduced by Chang *et al.* [12]. They proposed a search engine for matching an index with several documents on a server. This approach cannot prevent information disclosure in encrypted data retrieving procedure. Nergiz *et al.* [23] create equivalence classes to separate sensitive values from corresponding encrypted data and enhance user anonymity.

The other group of privacy preserving mechanisms established based on public key encryption to protect the privacy of the data receiver. The first searchable algorithm in this category was proposed by Boneh *et al.* [15]. Since server could encrypt any keyword with public key, these approaches suffer from keyword privacy.

Tang [14] considered a keywords randomizer including a trapdoor to prevent any unauthorized capability creation. While this approach improved performance of the previous public key schemes [9, 15], it failed for unstructured text. Implementation based on secret sharing and excessive interaction could be considered as a vulnerability point for the work presented in [14]. Decreasing the overhead of acquiring search capabilities, a similar method in [19] was introduced to deal with a common problem in a multi-user scheme.

Benaloh *et al.* [17] proposed a hierarchical based encryption model that generates and delegates a specific private sub-key for accessing each section of records. Key distribution cost and inherent limitation of hierarchical-based encryption decrease performance and flexibility of such models. Pang *et al.* [18] proposed a method for protecting user privacy by adding meta-data datasets. In comparison with our paper, data redundancy and salient false positive in query results is avoidable.

In [3] Wang *et al.* provide a query method based on secure indexing [16]. While their method preserves the data confidentiality, it does not guarantee the access control. However, they improved secure rank search using a searchable symmetric encryption. Later, an authorized private keyword search is proposed in [1, 15] that improved information access with the trustworthy assumption at server side. The other conjunctive keyword search proposed by Cao *et al.* [2]. While their approach facilitates the usage of user's capability list for querying, they use a set of strict privacy requirements. However, in this paper we focused on the single key search on encrypted data. Therefore, comparison between our approach and multi-keyword public key search would be meaningless.

6 Conclusion

In this paper, we proposed a cryptographic data querying system based on a private key approach. Our scheme, introduces a balancing technique between security enhancement and performance overhead in spite of existing approaches. We investigated benefits and limitations of our system and we believe that the advantages that it provides outweigh the potential shortages. In particular, by increasing the text blinding it becomes harder to perform cryptanalysis and statistical attacks. In comparison with similar works, our system enjoys the advantages of bandwidth reduction, eliminating trust to clients side, and non-limited to a specific size of words. In future work we will extend our system to allow more complex queries. Ideally, the extension can be exposed in these directions:

- Enhancing user anonymity: Guaranteeing privacy of user query results and preventing user profiling (anonymity), are important issues without an efficient solution. However, majority of existing approaches do not prevent an attacker from linking some sensitive value to a person [23, 24]. While [2, 11] cannot prevent attackers to associate a specific record to a specific person, [4, 6] conquest this problem with a substantial overhead.
- Developing conjunctive keyword search: Single keyword queries are inadequate for real world search in encrypted applications. Similar multi ranked methods are effective to improve data retrieval. Therefore, an important issue which is not addressed by our system is allowing the service provider to execute practical multi ranked query.

Acknowledgements

The authors would like to thank Fatemeh Sedaghatnia for her interest and support of this work and Behnam Nikbakht for his help to complete the experiments.

References

[1] Ming Li, Shucheng Yu, Ning Cao, and Wenjing Lou. Authorized Private Keyword Search over Encrypted Data in Cloud Computing. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 383 –392. IEEE, June 2011.

[2] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. In *INFOCOM, 2011 Proceedings IEEE*, pages 829 –837, april 2011.

[3] Cong Wang, Ning Cao, Jin Li, Kui Ren, and Wenjing Lou. Secure Ranked Keyword Search over Encrypted Cloud Data. In *ICDCS*, pages 253–262. IEEE Computer Society, 2010.

[4] Qiang Tang. Privacy Preserving Mapping Schemes Supporting Comparison. In *CCSW '10: Proceedings of the 2010 ACM Workshop on Cloud Computing Security*, pages 53–58, New York, October 2010. ACM.

[5] Hakan Hacigümüs, Balakrishna R. Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD Conference*, pages 216–227, 2002.

[6] Manolis Terrovitis, John Liagouris, Nikos Mamoulis, and Spiros Skiadopoulos. Privacy Preservation by Disassociation. Technical Report TR-IMIS-2011-1, Institute for the Management of Information Systems, "Athena" RC, Greece, 2011.

[7] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[8] Hyun-A Park, Jae Hyun Park, and Dong Hoon Lee. PKIS: Practical Keyword Index Search on Cloud Datacenter. *EURASIP J. Wireless Comm. and Networking*, 2011:64, 2011.

[9] Yong Hwang and Pil Lee. Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. In *Pairing-Based Cryptography Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 2–22. Springer Berlin / Heidelberg, 2007.

[10] Xuhua Zhou, Xuhua Ding, and Kefei Chen. Lightweight Delegated Subset Test with Privacy Protection. In *ISPEC*, volume 6672 of *Lecture Notes in Computer Science*, pages 138–151. Springer, 2011.

[11] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-Preserving Encryption for Numeric Data. In *SIGMOD Conference*, pages 563–574. ACM, 2004.

[12] Yan-Cheng Chang and Michael Mitzenmacher. *Privacy Preserving Keyword Searches on Remote Encrypted Data*, pages 442–455. Springer, 2005.

[13] Philippe Golle, Jessica Staddon, and Brent R. Waters. Secure Conjunctive Keyword Search over Encrypted Data. In *ACNS*, pages 31–45, 2004.

[14] Q. Tang. Revisit the Concept of PEKS: Problems and a Possible Solution. Technical Report TR-CTI, University of Twente, Enschede, The Netherlands, Enschede, 2008. URL `http://doc.utwente.nl/64938/`.

[15] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In *EUROCRYPT*, pages 506–522, 2004.

[16] Eu-Jin Goh. Secure Indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.

[17] Josh Benaloh, Melissa Chase, Eric Horvitz, and Kristin Lauter. Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records. In *CCSW*, pages 103–114. ACM, 2009.

[18] HweeHwa Pang, Xuhua Ding, and Xiaokui Xiao. Embellishing Text Search Queries To Protect User Privacy. *PVLDB*, 3(1):598–607, 2010.

[19] Feng Bao, Robert H. Deng, Xuhua Ding, and Yanjiang Yang. Private Query on Encrypted Data in Multi-user Settings. In *ISPEC*, pages 71–85, 2008.

[20] Erez Shmueli, Ronen Vaisenberg, Yuval Elovici, and Chanan Glezer. Database Encryption: An Overview of Contemporary Challenges and Design Considerations. *SIGMOD Record*, 38(3):29–34, 2009.

[21] Ahmed A. L. Faresi and Duminda Wijesekera. Preemptive Mechanism to Prevent Health Data Privacy Leakage. In *MEDES*, pages 17–24, 2011.

[22] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, 1970.

[23] Mehmet Ercan Nergiz and Chris Clifton. Thoughts on k-Anonymization. *Data and Knowledge Engineering*, 63(3):622–645, 2007.

[24] Gabriel Ghinita, Yufei Tao, and Panos Kalnis. On the Anonymization of Sparse High-Dimensional Data. In *ICDE*, pages 715–724. IEEE, 2008.

Hammad Afzali received his B.Sc. degree in Information Technology Engineering from Sharif University of Technology, Tehran, Iran in 2009, and his M.Sc. degree in Information Security from Malek Ashtar University of Technology, Tehran, Iran in 2012. He is currently a researcher in Operating System Security Laboratory (OSSL) at Alzahra University with the major of VMM-based Kernel Rootkit Detection. His research interests include virtualization based system security, malware capture and analysis, encrypted data management, and digital image forensics.

Hamed Nemati received his BSc degree in Information Technology Engineering from Shahid Rajaee University, Tehran, Iran in 2008. Receiving his MSc degree, he was graduated from Malek Ashtar University of Technology in Information Security in 2010. He is currently a researcher in the School of Computer Science and Communication at KTH University with the major of formal proof and the verification of small hypervisors. His research interests include virtualization technology and security, compression algorithms, formal proof and ad-hoc networks.

Reza Azmi received his BSc degree in Electrical Engineering from Amirkabir University of Technology, Tehran, Iran in 1990 and his MSc and PhD degrees in Electrical Engineering from Tarbiat Modares University, Tehran, Iran in 1993 and 1999, respectively. He is currently an Assistant Professor of Computer Engineering at Alzahra University. He also is the founder of Operating System Security Lab (OSSL), Web-based Anomaly Detection Lab (WADL), Medical Image Processing Lab (MIPL), Face and Facial Expression Recognition Lab (FFERL), and Optical Character Recognition Lab (OCRL) in Alzahra University. His research interests include artificial intelligence, artificial immune systems, host-based anomaly detection, secure kernels, pattern recognition and image processing.