[Downloaded from www.aece.ro on Saturday, October 13, 2018 at 08:42:51 (UTC) by 159.226.100.198. Redistribution subject to AECE license or copyright.]

Secure Multi-Keyword Search with User/Owner-side Efficiency in the Cloud

Younho LEE[1], Pyung KIM[2], Yongsu PARK[3]
[1]*Department of Industrial and Systems Engineering, Seoul National University of Science and Technology, Seoul, 139743, Republic of Korea*
[2]*Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, 305338, Republic of Korea*
[3]*Division of Computer Science and Engineering, Hanyang University, Seoul, 133791, Republic of Korea*
**Corresponding author Yongsu PARK: yongsu@hanyang.ac.kr*

***Abstract*—As the amount of data in the cloud grows, ranked search system, the similarity of a query to data is ranked, are of significant importance. on the other hand, to protect privacy, searchable encryption system are being actively studied. In this paper, we present a new similarity-based multi-keyword search scheme for encrypted data. This scheme provides high flexibility in the pre- and post-processing of encrypted data, including splitting stem/suffix and computing from the encrypted index-term matrix, demonstrated to support Latent Semantic Indexing(LSI). On the client side, the computation and communication costs are one to two orders of magnitude lower than those of previous methods, as demonstrated in the experimental results. we also provide a security analysis of the proposed scheme.**

***Index Terms*—keyword search, encryption, data security, information security, security.**

I. Introduction

Cloud computing is a computing model in which programs or applications are run on Internet-connected servers rather than on local computing devices, i.e., a user does not need to have vast computing resources, but can outsource (some part of) the computing tasks. In cloud computing, the user rents IT resources (e.g., software, storage, or computing resources) as required. Service capacity can be flexibly extended for heavy loads, but the user can pays for the exact amount they have used [1–3].

In this paper we focus on the Data as a Service (DAS) model [4]. In DAS, there are three entities: data owner, cloud server, and data client. The data owner has authority to the cloud server. The cloud server receives service queries (e.g., keyword searching, deleting, or updating) from the data client/owner and responds with the results.

The data owner may be reluctant to upload his/her private data such as e-mail, medical records, or photos. Hence, privacy protection is one of the most crucial requirements to make the DAS model more popular.

To provide privacy protection, data confidentiality, where only permitted users can access the data, can be used. To provide data confidentiality in DAS, in the naïve approach, the data owner first encrypts all his/her data and uploads it. If conventional encryption methods are used, the cloud server cannot perform diverse service transactions, including searches or modifications, since the server does not know the secret key. In this regards, to provide data confidentiality together with flexible transaction-service is considered to be difficult.

As in our previous work [4], we assume that the cloud server is a passive adversary/curious intruder, i.e., it collects and analyzes the service requests and results but cannot conduct active attacks, e.g., modify data/queries/results, insert additional queries, etc. In addition, we assume the known ciphertext model [5] where the server (adversary) can read the encrypted data and the encrypted keywords but does not know statistical information about the related plaintext data/keywords (note that the statistical distribution of plaintext data/keywords is different from that of conventional data/keywords). Our paper does not consider the known background model [5] where the adversary compares queries/results with the statistical information of text data.

In this environment, there are two approaches to providing data confidentiality together with flexible transaction-services: enabling keyword searching over encrypted data or enabling query processing over the encrypted database [4]. The latter provides high flexibility because it supports various SQL commands, but it relies on non-conventional cryptography, e.g., order preserving encryption [18], which is considered as experimental and less secure [10].

The former approach includes single keyword searching [6–10], multi-keyword searching [5,11–13], and the scheme [14] based on Latent Semantic Indexing (LSI) [15–16]. Because data should be encrypted by the data owner, he/she should conduct pre-processing before encryption, e.g., splitting stems and suffixes to extract keywords or statistical processing for LSI. When the data is updated, he/she must repeat this job. To support the known background model, the communication overhead is very high and additional work must be processed on the data client.

This paper takes the former approach, where we do not use a known background model, but assume a known ciphertext model. At the expense of slightly increasing the adversaries' power, the cloud server can support flexible services including splitting stems and suffixes, constructing index-term matrice, statistical processing (for LSI), and data updates. To demonstrate this, we present a new scheme to support LSI for encrypted data. Unlike previous methods, it

This study was supported by the Research Program Funded by the Seoul National University of Science and Technology

Digital Object Identifier 10.4316/AECE.2016.02002

1582-7445 © 2016 AECE

[Downloaded from www.aece.ro on Saturday, October 13, 2018 at 08:42:51 (UTC) by 159.226.100.198. Redistribution subject to AECE license or copyright.]

supports data updates (and the corresponding post-processing) on the cloud server. We implemented our scheme as well as the method in [14] that supports LSI, and experimental results show that our scheme has much lower communication and computation overheads on the data client side.

This paper is organized as follows. Section 2 briefly describes related work. Section 3 presents the proposed model, called "encrypted DAS." As an example, we also present a lightweight LSI-based privacy-aware search scheme that supports data updates and LSI processing on the cloud server. Section 4 describes a performance analysis of the proposed scheme and an experimental comparison with previous schemes. In Section 5, security analysis is provided, and Section 6 concludes the paper.

II. Related Work

The first subsection deals with background and definitions. The next subsection describes related work with respect to (searchable encryption based) privacy-preserving Boolean keyword matching. In the final subsection, we explain related work with respect to similarity based privacy-preserving retrieval.

A. Background and definitions

As described in Section I, this paper focuses on the DAS in cloud computing. DAS consists of three components: data owner, cloud server, and data client. The data owner has authority to access the data. Initially, the owner sends his/her data to the cloud server. The cloud server takes service queries (e.g., keyword searches, deletion requests, and updates) from the data client/owner and returns the results.

To support privacy protection, one major challenging problem in DAS is how to provide data confidentiality together with flexible transaction-services. The naïve approach, where the data owner first encrypts all his/her data and uploads it to the cloud server, cannot provide diverse service transactions (e.g., searches or updates) on the cloud server if the conventional encryption methods are used because the server does not know the secret key.

As in our previous work [4], we assume that the cloud server is a passive adversary/curious intruder, i.e., it collects and analyzes the service requests and results but cannot conduct active attacks, e.g., modify data/queries/results, insert additional queries, etc. This model can be divided into two types: the known ciphertext model and known background model [12]. The known ciphertext model [12] is where the server(adversary) can read the encrypted data and encrypted keywords but does not know statistical information about the related plaintext data/keywords. The known background model [12] implies stronger adversaries who can compare queries/results with statistical information about the text data.

To defend against adversaries under the known ciphertext model, keywords in the query and data are encrypted using (searchable) encryption to thwart analysis. In the known background model, the query is modified or screened to increase false positive rates and thwart analysis that compares queries/results with the statistical information of text data.

In plaintext retrieval, three classical approaches, boolean, vector space, and probabilistic methods, are suggested. Boolean methods return the Boolean vectors that represent matching results for keywords in the queries. In these methods, searchable encryption can be used to protect privacy. Vector space methods find and return the most similar data to the keywords in the query. For probabilistic methods, to the best of our knowledge, there is not yet a means to preserve privacy. Sections 2.B and 2.C describe Boolean keyword matching methods and similarity methods, respectively.

B. Privacy-preserving boolean keyword matching

Searchable encryption can be grouped into two approaches: symmetric key based [6–7] and public key based [8–9]. The former methods assume that the data owner and data client share the same secret key information. The data owner encrypts his/her data using the key and builds information for searching encrypted data, which is called the trapdoor. Specifically, each trapdoor corresponds to each keyword in the data. The data client use trapdoors in the queries to hide access patterns. The method in [7] builds a pre-computed index to reduce search time. The method in [8] builds index information using a public key while trapdoors are generated by the data owner. In this scheme, the data owner should always be online, and the search overhead is large because of public key algorithms and redundancy in the keyword set needed to build index information. The proposal in [9] can be considered less secure than other schemes because its trapdoors are deterministically generated.

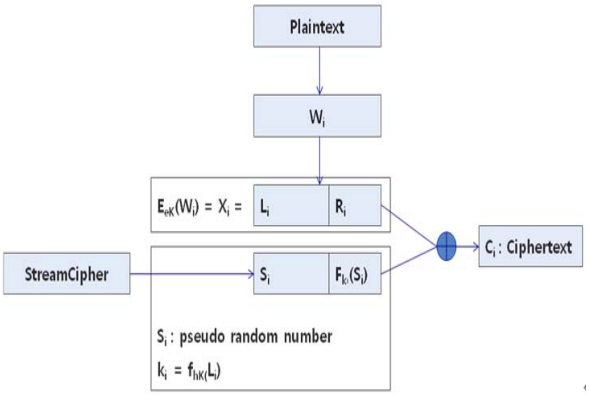

Figure 1. Overview of searchable encryption

As described in Section I, this paper focuses on the DAS in cloud computing. DAS consists of three components: data owner, cloud server, and data client. The data owner has authority to access the data. Initially, the owner sends his/her data to the cloud server. The cloud server takes service queries (e.g., keyword searches, deletion requests, and updates) from the data client/owner and returns the results.

We now briefly describe the proposed approach in [6]. Assume that the data owner and data user share the block ciphering algorithm key eK and the hashing key hK. The data owner encrypts his/her data, where encryption is done for each word W_i. First, the data owner computes $E_{eK}(W_i) = X_i$, where X_i is a concatenation of two strings L_i and R_i. Using L_i, a pseudo random function f, and hK, k_i can be computed. A pseudo random number S_i is then generated. From S_i, another pseudorandom function F, and <k_i, V_i> is

[Downloaded from www.aece.ro on Saturday, October 13, 2018 at 08:42:51 (UTC) by 159.226.100.198. Redistribution subject to AECE license or copyright.]

computed. The ciphertext for W_i is $C_i = \langle L_i, R_i \rangle \oplus \langle S_i, V_i \rangle$, which is stored in the cloud server. The trapdoor for W_i is $\langle X_i, k_i \rangle$.

If the data client wants to search W_i, he/she sends the trapdoor $\langle X_i, k_i \rangle$ to the cloud server. The server computes $Xi \oplus Ci = \langle S_i, V_i \rangle$. Then, it computes k_i (the procedure of which is similar to that of encryption) and by using S_i and k_i, V_i' is made. If $V_i = V_i$', the server regards C_i contains the keyword for W_i.

Because the approaches in [6–9] return a Boolean vector containing the matching results, they do not support ranked searching. The approach in [10] increases search efficiency using fuzzy techniques as follows. In the query, additional search keywords are added using fuzzy techniques that increase the possibility of a successful match. However, this method does not support ranked searching.

C. *Similarity-based privacy-preserving retrieval*

The methods described in Section 2.1 do not support ranked searching, which measures the similarity between keywords in the query and data entries. In this section we describe three similarity-based schemes: single keywords, multiple keywords, and semantic schemes.

The method in [11] encrypts each keyword in a document using searchable encryption and the frequency of each keyword using order preserving encryption. To process the query, first it searches all the documents containing the keyword, sorts them with respect to the encrypted frequency value, and then returns the k documents with the highest frequency. The order preserving encryption enables the method to compare encrypted frequency values.

The method in [12] is the first proposal, to the best of our knowledge that addresses Multi-keyword Ranked Search over Encrypted cloud data (MRSE). MRSE uses a pre-defined similarity function that measures similarity between two data (i.e., keyword and document). A similarity function measures the similarity as follows.

- Frequently used keywords in the documents have low values when computing similarity.
- If a specific keyword is frequently used in a single document but is not frequently used in others, it has a high similarity value for that document.
- A document that has a greater number of different keywords has a lower similarity value.

The term-document matrix (TDM) is a widely used similarity function that is constructed as follows. Assume that $f_{d,t}$ is the frequency of appearance of keyword d in document t, f_t is the frequency of appearance of the keyword t in the all documents, and N is the number of documents. We can then calculate two weight values:

$$w_{d,t} = f_{d,t} \times \log \frac{N}{f_t} \qquad (1)$$

$$W_d = \sqrt{\sum\nolimits_t w_{d,t}^2} \qquad (2)$$

Let $P=\{p_1, p_2, \ldots, p_m\}$ denote the set of all keywords. for each document d, weight vector v_d is defined as follows;

$$v_d = (w_{d,p_1} / W_d, w_{d,p_2} / W_d, \ldots, w_{d,p_m} / W_d) \qquad (3)$$

The TDM is a matrix where the row vectors are the weight vectors for all documents.

Figure 2 presents an overview of a similarity-based retrieval system that uses TDM. First, the cloud server extracts keywords from the documents to construct set P. It then builds the TDM such that each row represents the (frequency) weights of keywords for each document. The cloud server is then ready for service.

If the data client enters a query containing keywords for searching, the client software builds a Query Matrix (QM) using the keyword. QM is a vector of size |P| and each each entry represents the frequency of the keyword in the query. After the QM is sent to the cloud server, the server computes matrix operations to calculate the similarity value of each document. Finally, the server returns the most or *k* most similar document(s) to the client.

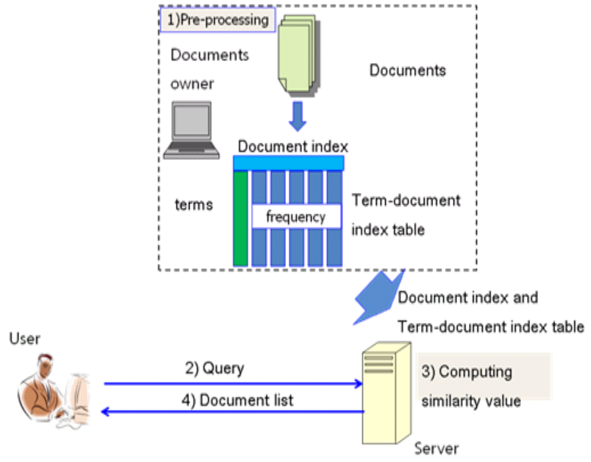

Figure 2. Overview of a similarity-based retrieval system

Here, we briefly explain similarity-based privacy-preserving retrieval schemes. In [12], the data owner builds a binary vector that represents the appearance of keywords in the query and then encrypts it. This system uses a special encryption called asymmetric scalar product preserving encryption, denoted by $E_{ASPP}()$. If vectors A, B, C have the following property $A \cdot B = C$ (here, $\cdot$ denotes the inner product), then $E_{ASPP}(A) \cdot E_{ASPP}(B) = E_{ASPP}(C)$. This is a probabilistic encryption scheme because it adds random values to the encryption process to make the statistical analysis of encrypted queries difficult. However, the size of the ciphertext is quite large, which incurs high communication overhead, and the data client must perform additional computation for decryption. The system in [13] uses a binary vector that is obtained from statistical processing using term frequency and inverse document frequency. This scheme could be considered more correct than the previous schemes, but it does not support semantic similarity, e.g., LSI. To defend against a known background model, [12] adds additional keywords in the query, but this incurs additional communication and computation costs as well as false positives.

LSI: In similarity based schemes, similarities of keywords are compared or the frequency distribution of keywords is analyzed in documents. In comparison, we should consider semantic similarity to improve search accuracy, e.g., fruit, apple, and car are different keywords, but fruit and apple are semantically closer than fruit and car. Semantic similarity is an important technique for measuring similarity in text documents, and one widely used method is LSI [15–16]. LSI uses TDM for data structure and the technique is called Singular Value Decomposition (SVD) to analyze the relevance of keywords, as follows.

[Downloaded from www.aece.ro on Saturday, October 13, 2018 at 08:42:51 (UTC) by 159.226.100.198. Redistribution subject to AECE license or copyright.]

When the TDM X consists of m keywords and n documents, the SVD for X is $X = U\sum V^T$, where $U = [u_{il}]$ is an orthogonal left singular matrix, $\sum$ is a diagonal matrix, diag($\partial 1$, $\partial 2$, ..., ∂n), and $V = [v_{jl}]$ is an orthogonal right singular matrix. In the matrix $TDM^T TDM = V\sum^2 V^T = [z_{ij}]$, z_{ij} is the inner product of the i-th document and j-th document. Because $\sum$ is a diagonal matrix, it affects scalability only, and V contains the key information for document relevance, i.e., V is the vector space for semantic similarity-based search. Because $\sum$= diag($\partial 1$, $\partial 2$, ..., ∂n), where $\partial 1 > \partial 2 > ...> \partial r > ..., > \partial n = 0$, we can truncate some entries in $\sum$ for efficient computation if ∂r is negligible.

LSI-based privacy-preserving retrieval: To the best of our knowledge, the only privacy-preserving retrieval scheme so far to support LSI is the system proposed in [14]. To protect privacy, some part of V^T is encrypted using the encryption algorithm and then sent to the server. When the query is constructed in the data client, this part is masked to defend against adversaries in the known background model, which results in false positives in the results and incurs high communication overheads. To obtain the correct result, the data client should do the post-processing, which requires a considerable amount of computational cost. Moreover, masking queries does not safely prevent information leakage.

III. Proposed Scheme

A. Objectives

To provide data confidentiality and flexible transaction-services together, we have four objectives, which are described as follows.

1) To provide provable security to protect privacy: We design the proposed scheme to be mathematically proven to be secure under the known ciphertext model [12]. Our scheme is not secure under the known background model [12]. To defend against the known background model, additional techniques, e.g., query masking, should be used, which is outside the scope of this paper.

2) To support diverse post-processing of data on the cloud server: To use semantic searching, e.g., LSI, the scheme should support diverse post-processing on data, including matrix operations. As shown in Section 4, in some cases this processing takes a considerable amount of time and is better done on the cloud server, not on the owner/client side. Our scheme supports diverse post-processing (statistical analysis on keywords in the encrypted documents), and we demonstrate this by showing that our scheme supports LSI.

Moreover, in our scheme, the data owner does not need to split stems and suffixes from the documents to build the index-term matrix. He/she simply encrypts the documents and the stemming work is done on the cloud server.

3) To add/delete/update documents and related keyword information efficiently: In previous work, index generation work (building index-term matrices) is done by the data owner to protect privacy. If the size of the data is large, the processing time is long, which is less efficient in cloud computing environments. If this work is done on the cloud server, data owners can freely generate/update/delete documents without any burden.

4) To minimize communication and computation overheads on the data client: In similarity-based privacy-preserving retrieval, the cloud server returns the *k* most relevant documents with respect to the query. Some privacy protection methods, including [14], add a large amount of false-positive results to defend against adversaries in the known background model, which incurs high communication overheads. Moreover, to filter out the false-positive results, the data client must do post-processing that incurs high computational overheads. In this section, we present a scheme that does not include false-positive results and as a result, minimizes communication and computation overheads.

B. Overview of the proposed scheme

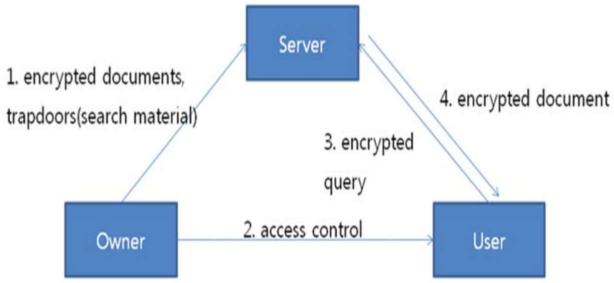

Figure 3. Overview of the proposed scheme

Figure 3 shows an overview of the proposed scheme, which works as follows.

1) The data owner extracts keywords from the documents, encrypts the documents, and computes trapdoors for the keywords. Trapdoors are sent to the cloud sever.

2) The data owner shares the secret key that is used for encryption with the data client.

3) The cloud server receives the trapdoors and the encrypted documents. From these, it generates the TDM and performs LSI.

4) The data client encrypts the query and sends it to the cloud server.

5) The cloud server processes the transaction and returns the *k*-most relevant documents.

C. Base algorithm

Before describing the proposed scheme, we first describe two algorithms that our scheme is based on: prefix searchable encryption [17] and searchable encryption [6], as follows.

Prefix searchable encryption: Unlike the previous approach, where the data owner splits stems and suffixes in the documents before encryption, in this method, the cloud server does this work using encrypted documents. To do this, we use a prefix searchable encryption [17], called Hash-CBC (HCBC).

HCBC consists of three procedures, as follows:

1) KeyGen(1^e): this procedure takes security parameter 1e and returns secret key K_E.

2) $Enc_H(m, K_E)$: given plaintext message m and K_E, this procedure returns ciphertext c.

3) $Dec_H(c, K_E)$: given ciphertext c and K_E, if $c = Enc_H(m, K_E)$, this procedure returns m. Otherwise, it returns $\perp$,

[Downloaded from www.aece.ro on Saturday, October 13, 2018 at 08:42:51 (UTC) by 159.226.100.198. Redistribution subject to AECE license or copyright.]

implying failure.

4) Trapdoor(m, K_S): given plaintext m and K_S, this procedure returns trapdoor T.

5) Search(c, T): given ciphertext c and trapdoor T. If T = Trapdoor(Dec(c, K_S), K_S), this procedure returns true, else it returns false.

D. Proposed scheme

In the proposed scheme, the encryption procedure E is as follows: E_H denotes the encryption algorithm in HCBC. For each word in the documents, the data owner finds the root of m and split the root and the suffix such that m = <root, tail>. The root is regarded as the prefix, and HCBC is applied as $E_H(m) = <c_1, c_2>$. (If the tail is empty, HCBC works just like a conventional block ciphering algorithm that produces c_1 only.) Next, [1] is applied to c_1 only, and the final ciphertext message for m is $<E_S(c_1), c_2>$.

The overall protocol is described as follows. We assume that the communication is secured and authenticated to simplify the explanation.

1) Setup: Three participants (data owner, cloud server, and data client) encrypt documents and initialize data, as follows.

a. Three entities share the algorithms in [6] and [17]. The data owner and the data client share the keys K_S and K_E for the encryption algorithms in [6] and [17], respectively.

b. The data owner extracts all keywords W = {w_1, w_2, … w_N} from documents F = {d_1, d_2, …, d_N}. Assume that there is no suffix in the extracted keywords, only a prefix (root). He/she encrypts W using [17], re-encrypts them using [6], and builds trapdoors for W. The set of all generated trapdoors is denoted by T.

c. The data owner encrypts the documents in set F = {d_1, d_2, … d_N}. Encryption is done on units of words. For each word, a pair (prefix, suffix) is encrypted using [17], and then the encrypted prefix is re-encrypted using [6].

d. The data owner constructs the encrypted document using encrypted words. The set of encrypted documents is denoted by C = {c_1, c_2, … c_N}.

e. The data owner sends C and T to the cloud server.

2) BuildIndex: The cloud server receives C and T. It then performs post-processing on the encrypted documents. In this paper, we describe how to compute the TDM and V^T used in LSI.

a. In T, each trapdoor corresponds to each keyword in documents in C. Hence, the cloud server can compute all statistical information using the Search() algorithm in [6]. In this way, the server computes the TDM.

b. The server uses the LSI algorithm to compute U, ∑, and V. The i-th column of V^T corresponds to the i-th document, and the j-th row V^T corresponds to the j-th keyword. Therefore, each column vector of V^T holds the semantic distribution of keywords in the corresponding document.

3) Query: the data client sends the query, and the cloud server returns the result, as follows.

a. The data client uses K_S and K_E to construct query Q and encrypts Q. Encryption is done in units of words. For each word, a pair (prefix, suffix) is encrypted using [17], and then the encrypted prefix is re-encrypted using [6]. He/she sends the encrypted query to the cloud server.

b. The cloud server receives the encrypted query and computes v_q using T. It then computes $\Sigma^{-1}\cdot U^T\cdot v_q$ to measure similarities in the vector space V^T, the semantic distribution of keywords in the query.

c. The cloud server returns *k*-ranked results. The measure of similarity is the inner product between each column vector of V^T and $\Sigma^{-1}\cdot U^T\cdot v_q$. Higher similarity value indicates more semantic correlation between the query and corresponding document.

4) Update: After building the index on the cloud server, the data owner can later modify/update some documents. To do this, he/she first analyzes added or deleted keywords in the modified documents, computes trapdoors for them, and encrypts the modified documents, all of which is described in Step 1. The data owner sends C′ and T′, where C′ and T′ contain only added/deleted keywords and documents, respectively. After receiving these, the cloud server computes *BuildIndex* to compute the index information and V^T.

IV. Experimental Results

A. Comparison of overheads in terms of computation and communication

To evaluate the proposed scheme, we chose the system in [14] for comparison, since [14] is the only work to support LSI to the best of our knowledge. We implemented both schemes using the Java language in Windows XP, where the hardware specifications are as follows: Intel® Core™ i3 CPU with 2.93 GHz and 3 GB main memory. For the data set, we randomly chose 5,000, 10,000, 15,000, and 20,000 documents from the Reuters Corpus Volume No. 1 (RCV1).

1) Computational overhead for *BuildIndex*: Figures 4 and 5 show the overall computation overhead and the computation overhead for the data owner, respectively. The y-axis of both graphs is log-scaled. Recall that BuildIndex encrypts documents, extracts keywords, and builds TDM and LSI. Figure 4 shows that the overall computational cost of the proposed scheme is slightly larger than that of [14] because we use double encryption and an additional encryption for extracted keywords. However, unlike [14], building TDM and processing LSI, which require a significant computational overhead, are performed on the cloud server. Hence, if we compare the computational overhead for the data owner, the proposed scheme is up to hundreds of times faster, as shown in Figure 5. We believe that this is a strong point of our scheme, because in cloud computing environments, the server's computational power is sufficient with respect to the clients.

2) Communication overhead: Figure 6 compares the results for communication overheads between the proposed scheme and [14]. Experimental results show that, compared with [14], the communication overhead of our scheme is up to hundreds of times lower. This is because [14] uses masking in the query to defend against adversaries in the known background model. This masking incurs a large number of false-positives in the results. As *k* becomes smaller and the degree of masking becomes higher, the false-positive rate increases. This leads to larger communication overheads. Furthermore, in [14], after receiving the results, the data client must perform post-processing to filter out the false-positive results, incurring

[Downloaded from www.aece.ro on Saturday, October 13, 2018 at 08:42:51 (UTC) by 159.226.100.198. Redistribution subject to AECE license or copyright.]

additional computational overheads for the client

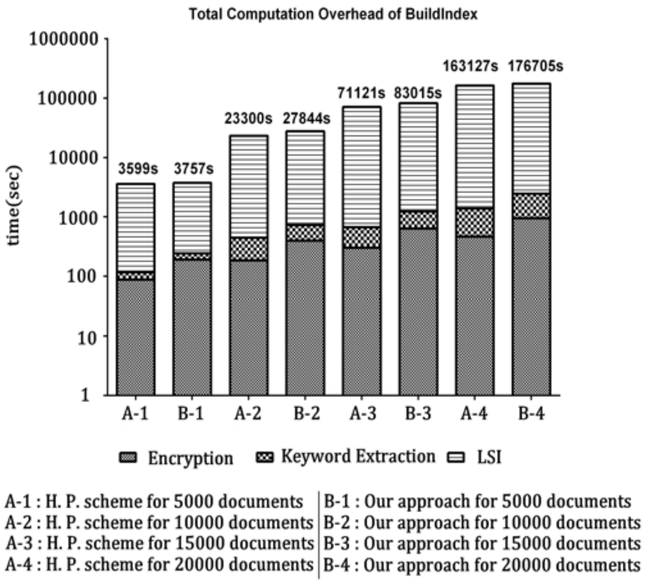

Figure 4. Overall computational overhead

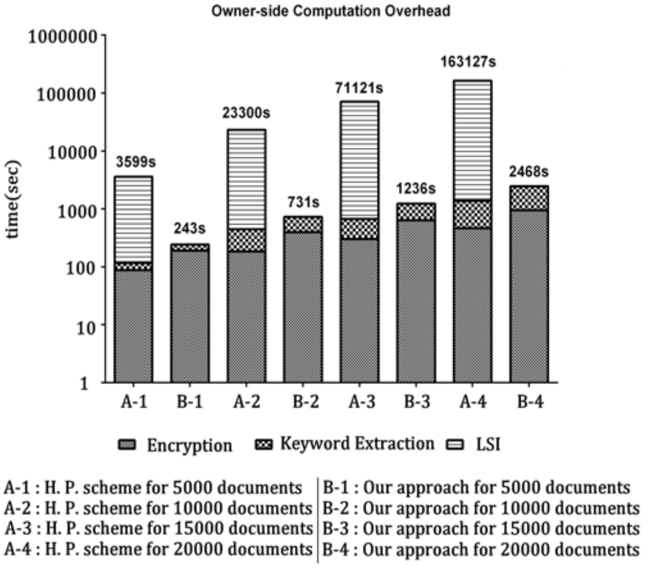

Figure 5. Computational overhead for the data owner

3) Precision: The scheme precision measures whether the data client obtains the most k relevant documents with respect to the query. In this regard, the proposed scheme has exactly the same precision as [14]. This is because our scheme uses a stemming algorithm to split roots and suffixes to extract the keyword that is identical to the one in [14]. Hence, the proposed scheme and the scheme in [14] have the exactly same keywords (and corresponding trapdoors) for the same document set. This implies that both schemes have the same TDM and return the most *k* similar documents for the query. The difference is that our scheme returns exactly *k* documents, whereas [14] returns a considerable number of documents that are false-positives.

B. Functionality comparison

Table 1 shows the functionality comparison results between our scheme and the previous schemes [12–14]. For semantic searching, [14] and our scheme are the only ones to support it. For most criteria, our scheme can be considered better than the previous schemes, except with respect to security under the known background model. We intentionally designed it like this because we believe that our approach has high flexibility, no false-positives, and efficiency at the expense of sacrificing security in this case.

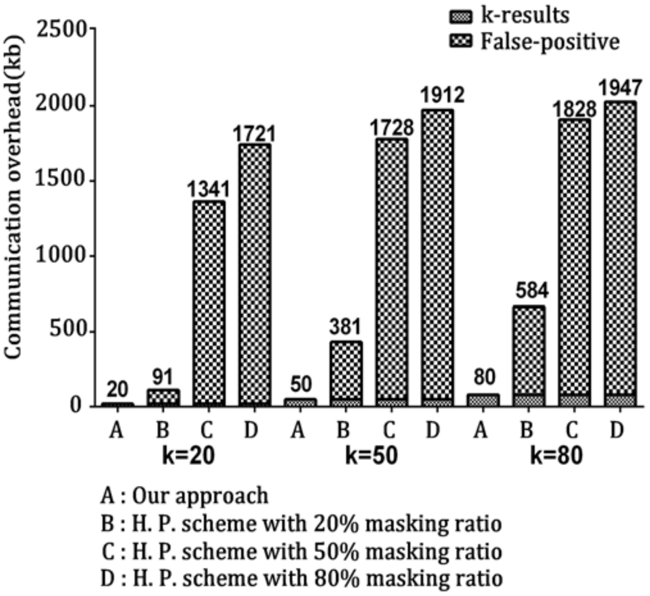

Figure 6. Comparison results for communication overhead

TABLE I. FUNCTIONALITY COMPARISON
(O:SUPPORTED, Δ:PARTIALLY SUPPORTED, X:NOT SUPPORTED)

Functionality	[14]	[12]	[13]	Proposed scheme
MRSE	O	O	O	O
LSI	O	X	X	O
Statistical processing on encrypted documents on the server	X	X	X	O
Security for the known ciphertext model [11]	Δ	Δ	Δ	O
Security for the known background model [11]	Δ	Δ	Δ	X
No additional work for the data client	X	X	X	O
Efficient communication overhead	X	O	O	O
Efficient update/deletion of documents	X	X	X	O

V. SECURITY ANALYSIS

In this section, we analyze the security of our scheme. Section 5.1 describes the standard definition of provable security for secret key cryptography, which is used in Section 5.2 to analyze the security of the proposed scheme.

A. Security definition

Generally, security of secret key encryption algorithms relies on the indistinguishing probability of cryptographic primitive output to random numbers. Assume that random

[Downloaded from www.aece.ro on Saturday, October 13, 2018 at 08:42:51 (UTC) by 159.226.100.198. Redistribution subject to AECE license or copyright.]

variables X and Y denote the uniformly distributed values that were drawn from $\{0,1\}^n$. The distinguishing probability of an algorithm $A:\{0,1\}^n \rightarrow \{0,1\}$, called the advantage over X and Y, is as follows:

$$Adv\,(A) = |\Pr[\,A(\mathrm{X}) = 1] - \Pr[\,A(\mathrm{Y}) = 1]\,| \qquad (4)$$

We can define the advantages of A over various cryptographic objects as follows:

- A pseudorandom generator G (G:K$_G$→S)

$$Adv\,(A_{\mathrm{G}}) = |\Pr[\,A_{\mathrm{G}}^{\mathrm{G(U_{KG})}} = 1] - \Pr[\,A_G^{\mathrm{U_S}} = 1]\,| \qquad (5)$$

where U$_{KG}$, U$_S$ are random variables distributed uniformly on K$_G$, S, where K$_G$ is the key space of the pseudorandom generator and S is the output space of the pseudorandom generator.

- A pseudorandom function F (F:K$_F$×X→Y)

$$Adv\,(A_{\mathrm{F}}) = |\Pr[\,A_{\mathrm{F}}^{\mathrm{F_{K_F}}} = 1] - \Pr[\,A_F^{\mathrm{R}} = 1]\,| \qquad (6)$$

where R represents a random function selected uniformly from the set of all maps from X to Y, K$_F$ refers to the key space of the pseudorandom function.

- A pseudorandom permutation function E (E:K$_E$×Z→Z)

$$Adv(A_{\mathrm{E}}) = |\Pr[A_{\mathrm{E}}^{\mathrm{E_{K_E}}, E_{K_E}^{-1}} = 1] - \Pr[A_{\mathrm{E}}^{\pi,\pi^{-1}} = 1]\,| \qquad (7)$$

where π represents a random permutation selected uniformly from the set of all bijections on Z, which is the output space of the pseudorandom permutation.

B. Security analysis

To summarize the security of the proposed scheme: 1) as in [6], it is secure against the known plaintext model and 2) the trapdoor distinguishes the encrypted documents containing the corresponding keyword from those that do not. For additional information, it is infeasible to compute. We explain this in detail, as follows.

First, note that [6] proved that the security of the scheme in [6] is that of secure pseudo-random generator. That is, if the employed pseudo random number generator is secure, no adversaries can practically distinguish the ciphertext from the output of a pseudo random number generator with a random seed. Moreover, [6] showed that even if the adversary has a single trapdoor, for other keywords, he/she can extract negligible information at best.

We assume that the adversary is the cloud server and it is “Honest-but-Curious” [12], meaning that it does not conduct active attacks but may conduct passive attacks, i.e., following protocol, collecting traffic information, and performing analysis. Under this assumption, the information that the server can collect comprises encrypted documents, encrypted queries, trapdoors, and TDM.

We analyze the security of attacks on encrypted documents, queries, and trapdoors as follows.

The reference [14] defines the degree of risk as fidelity, or the degree of inference information in TDM. More specifically, the Frobenius normal form:

$$1 - \frac{\left\| \mathrm{X} - \overline{\mathrm{X}} \right\|_{\mathrm{F}}}{\left\| \mathrm{X} \right\|_{\mathrm{F}}} \qquad (8)$$

where $\overline{\mathrm{X}}$ is the masked TDM matrix and X is the original TDM matrix. The fidelity of our scheme is 0 since all the entries in the TDM are encrypted using secure encryption algorithms.

We define the security of our scheme as advantage, in other words, the distinguishing probability of l ciphertexts and l random numbers. When the advantages of pseudorandom function F, pseudorandom function f, pseudorandom generator G, and pseudorandom permutation E are *Adv(A*$_F$*), Adv(A*$_f$*), Adv(A*$_G$*),* and *Adv(A*$_E$*)*, respectively, [6] shows that the advantage of the proposed scheme is

$$Adv(A) = l \cdot Adv(A_{\mathrm{F}}) + Adv(A_{\mathrm{G}}) + Adv(A_f) + \frac{l(l-1)}{2\,|\,X\,|} \qquad (9)$$

The reference [14] defines additional notation to represent the degree of the exposed queries and documents. In [14], query results include false-positives as to protect correlation between exposed queries and documents. It defines the number of false-positives as anonymity or compromised fidelity. In the proposed scheme, the security for compromised queries and documents is supported by [1]. If a keyword is exposed, [6] shows that the advantage is

$$Adv\,(A') = l \cdot Adv\,(A) + Adv\,(A_E) + \frac{l}{|\,X\,|} \qquad (10)$$

Unlike [14], the proposed scheme has additional strong points for security: 1) When building the TDM, the information of every keyword is not leaked. In the proposed scheme, to acquire this information, the attack would have to do a large amount of computation that is a non-polynomial time process. 2) Because every keyword is encrypted in the query, in query processing, the adversary does not know the keyword information in the query.

Even though false keywords can be injected into the exposed query result to increase the false positive rate when guessing the queried keywords, because of the restriction of the number of the false keywords that can be inserted for performance reasons, the adversary is able to extract query keywords with a very high probability in [14]. In the proposed scheme, however, it is difficult to determine the queried keywords because they are encrypted.

VI. Conclusion

In this paper, we presented a new similarity-based multiple keyword search scheme over encrypted data. This scheme provides high flexibility in pre-/post-processing encrypted data including splitting stems/suffixes and computing from an encrypted index-term matrix, which is demonstrated by its support for LSI. To the best of our knowledge, this is the first scheme to do so. On the client’s side, the computational and communication costs are one or two orders of magnitude lower than those of previous methods, as shown in the experimental results. In addition, a security analysis of the proposed scheme is provided. Based on the security of the underlying cryptographic algorithms, the proposed system’s security is guaranteed.

As future work, we are considering using fully

[Downloaded from www.aece.ro on Saturday, October 13, 2018 at 08:42:51 (UTC) by 159.226.100.198. Redistribution subject to AECE license or copyright.]

homomorphic encryption (FHE) or order preserving encryption (OPE) [18] to implement secure keyword search. Because any type of computation is possible on encrypted data in FHE, complex document matching operations are possible without the decryption of both keywords and documents, which would greatly enhance the security of the system. However, it is difficult to make FHE practical in terms of the size of the ciphertext and public key. The required computational time to perform even simple operation search such as bit-wise matching or simple exponentiation is also impractical. Therefore, we need to determine a solution to handle this problem, using not only theoretical methods but also systemic or technical methods. With OPE, we can very efficiently compare two ciphertexts without decrypting them, so it is possible that a range query could be performed without decryption when using OPE. This approach would greatly enhance the performance for similarity computation and finding the best-matching document. We hope they will replace the encryption algorithms that were shown to not be provably secure [19], such as [20].

REFERENCES

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: towards a Cloud Definition", ACM SIGCOMM Comput. Commun. Rev., vol. 39, no.1, pp. 50-55, 2009. doi:10.1145/1496091.1496100

[2] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing", Proc. IEEE INFOCOM, pp. 1-9, 2010. doi:10.1145/1496091.1496100

[3] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving Public auditing for Data Storage Security in Cloud Computing", Proc. IEEE INFOCOM, pp. 1-9, 2010. doi:10.1109/INFCOM.2010.5462173

[4] T. Yu, S. Jajodia, "Secure Data Management in Decentralized Systems." Advances in Information Security, vol. 33, pp. 355-380, Springer Press, 2007.

[5] H. Pang, X. Ding, and X. Xiao. "Embellishing Text Search Queries to Protect User Privacy." Proc. VLDB Endowment vol. 3.1/2, pp. 598-607, 2007. doi:10.14778/1920841.1920918

[6] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data", Proc. IEEE Security and Privacy, pp. 44-55, 2000. doi:10.1109/SECPRI.2000.848445

[7] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions", Proc. ACM Computer and Communication Security, pp. 79-88, 2006. doi:10.1145/1180405.1180417

[8] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and Efficiently Searchable Encryption", Proc. Advances in Cryptology – CRYPTO, pp. 535-552, 2007. doi:10.1007/978-3-540-74143-5_30

[9] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable Encryption Revisited: Consistency properties, relation to Anonymous IBE, and Extensions", Journal of Cryptology, vol. 21, no. 3, pp. 350–391, 2008. doi:10.1007/s00145-007-9006-6

[10] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing", Proc. IEEE INFOCOM'10 Mini-Conference, pp. 1-5, San Diego, CA, USA, March 2010. doi:10.1109/INFCOM.2010.5462196

[11] C. Wang, N. Cao, J. Li, K. Ren and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data", in Proc. ICDCS'10, pp.253-362, 2010. doi:10.1109/ICDCS.2010.34

[12] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. "Privacy-preserving Multi-keyword Ranked Search over Encrypted Cloud Data", in Proc IEEE INFOCOM'11, pp.222-233, 2011. doi:10.1109/INFCOM.2011.5935306

[13] Sun, W. et al. "Verifiable Privacy-Preserving Multi-keyword Text Search in the Cloud Supporting Similarity-based Ranking." IEEE Trans. on Parallel and Distributed Systems, in Press, 2013. doi:10.1109/TPDS.2013.282

[14] H. Pang, J. Shen, and R. Krishnan, "Privacy-Preserving Similarity-Based Text Retrieval", ACM Transactions on Internet Technology, vol. 10, no. 1, article #4, Feb. 2010. doi:10.1145/1667067.1667071

[15] S. T. Dumais, "Latent Semantic Indexing (LSI) and TREC-2", In Second Text REtrieval Conference (TREC2), D. Harman, ed., pp. 105-115, March 1994.

[16] T. K. Landauer, D. S. McNamara, S. Dennis, and W. Kintsch, "Handbook of Latent Semantic Analysis," 1st edition, ISBN-13: 978-0805854183, pp. 293-322, Psychology Press, 2007.

[17] M. Bellare, A. Boldyreva, L. Knudsen, and C. Namprempre, "Online ciphers and the hash-CBC construction", Proc. Advances in Cryptology – CRYPTO, 2001, pp.292-309, 2001. doi:10.1007/3-540-44647-8_18

[18] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order Preserving Symmetric Encryption", Proc. Advances in Cryptology – EUROCRYPT'09, pp. 224-241, 2009. doi:10.1007/978-3-642-01001-9_13

[19] M. Bellare, "Practice-oriented provable-security", in Proc. Information Security. Springer Berlin Heidelberg, pp. 221-231, 1998. doi:10.1007/BFb0030423

[20] W. Wong, D. W. Cheung, B. Kai, and N. Mamouolis, "Secure kNN computation on encrypted databases", Proc. ACM SIGMOD'09, pp. 139-152, 2009. doi:10.1145/1559845.1559862