Toward Secure Multikeyword Top-k Retrieval over Encrypted Cloud Data

Jiadi Yu, *Member*, *IEEE*, Peng Lu, Yanmin Zhu, *Member*, *IEEE*,
Guangtao Xue, *Member*, *IEEE Computer Society*, and Minglu Li

Abstract—Cloud computing has emerging as a promising pattern for data outsourcing and high-quality data services. However, concerns of sensitive information on cloud potentially causes privacy problems. Data encryption protects data security to some extent, but at the cost of compromised efficiency. Searchable symmetric encryption (SSE) allows retrieval of encrypted data over cloud. In this paper, we focus on addressing data privacy issues using SSE. For the first time, we formulate the privacy issue from the aspect of similarity relevance and scheme robustness. We observe that server-side ranking based on order-preserving encryption (OPE) inevitably leaks data privacy. To eliminate the leakage, we propose a two-round searchable encryption (TRSE) scheme that supports top-k multikeyword retrieval. In TRSE, we employ a vector space model and homomorphic encryption. The vector space model helps to provide sufficient search accuracy, and the homomorphic encryption enables users to involve in the ranking while the majority of computing work is done on the server side by operations only on ciphertext. As a result, information leakage can be eliminated and data security is ensured. Thorough security and performance analysis show that the proposed scheme guarantees high security and practical efficiency.

Index Terms—Cloud, data privacy, ranking, similarity relevance, homomorphic encryption, vector space model

1 Introduction

Cloud computing [1], a critical pattern for advanced data service, has became a necessary feasibility for data users to outsource data. Controversies on privacy, however, have been incessantly presented as outsourcing of sensitive information including e-mails, health history and personal photos is explosively expanding. Reports of data loss and privacy breaches in cloud computing systems appear from time to time [2], [3].

The main threat on data privacy roots in the cloud itself [6]. When users outsource their private data onto the cloud, the cloud service providers are able to control and monitor the data and the communication between users and the cloud at will, lawfully or unlawfully. Instances such as the secret NSA program, working with AT&T and Verizon, which recorded over 10 million phone calls between American citizens, cause uncertainty among privacy advocates, and the greater powers it gives to telecommunication companies to monitor user activity [7]. To ensure privacy, users usually encrypt the data before outsourcing it onto cloud, which brings great challenges to effective data utilization. However, even if the encrypted data utilization is possible, users still need to communicate with the cloud and allow the cloud operate on the encrypted data, which potentially causes leakage of sensitive information.

• *The authors are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 DongChuan Road, Shanghai 200240, P.R. China.*
E-mail: {jiadiyu, perlony, yzhu, gt_xue, mlli}@sjtu.edu.cn.

Manuscript received 15 July 2012; revised 11 Nov. 2012; accepted 23 Jan. 2013; published online 30 Jan. 2013.
Recommended for acceptance by S. Distefano, A. Puliafito, and K.S. Trivedi.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSCSI-2012-07-0159.
Digital Object Identifier no. 10.1109/TDSC.2013.9.

Furthermore, in cloud computing, data owners may share their outsourced data with a number of users, who might want to only retrieve the data files they are interested in. One of the most popular ways to do so is through keyword-based retrieval. Keyword-based retrieval is a typical data service and widely applied in plaintext scenarios, in which users retrieve relevant files in a file set based on keywords. However, it turns out to be a difficult task in ciphertext scenario due to limited operations on encrypted data. Besides, to improve feasibility and save on the expense in the cloud paradigm, it is preferred to get the retrieval result with the most relevant files that match users' interest instead of all the files, which indicates that the files should be ranked in the order of relevance by users' interest and only the files with the highest relevances are sent back to users.

A series of searchable symmetric encryption (SSE) schemes have been proposed to enable search on ciphertext. Traditional SSE schemes [22], [23] enable users to securely retrieve the ciphertext, but these schemes support only Boolean keyword search, i.e., whether a keyword exists in a file or not, without considering the difference of relevance with the queried keyword of these files in the result. To improve security without sacrificing efficiency, schemes presented in [9], [10], [24] show that they support top-k single keyword retrieval under various scenarios. The authors of [25], [26] made attempts to solve the problem of top-k multikeyword over encrypted cloud data. These schemes, however, suffer from two problems—Boolean representation and how to strike a balance between security and efficiency. In the former, files are ranked only by the number of retrieved keywords, which impairs search accuracy. In the latter, security is implicitly compromised to tradeoff for efficiency, which is particularly undesirable in security-oriented applications.

Preventing the cloud from involving in ranking and entrusting all the work to the user is a natural way to avoid information leakage. However, the limited computational power on the user side and the high computational overhead precludes information security. The issue of secure multikeyword top-k retrieval over encrypted cloud data, thus, is: How to make the cloud do more work during the process of retrieval without information leakage.

In this paper, we introduce the concepts of similarity relevance and scheme robustness to formulate the privacy issue in searchable encryption schemes, and then solve the insecurity problem by proposing a two-round searchable encryption (TRSE) scheme. Novel technologies in the cryptography community and information retrieval (IR) community are employed, including homomorphic encryption and vector space model. In the proposed scheme, the majority of computing work is done on the cloud while the user takes part in ranking, which guarantees top-k multikeyword retrieval over encrypted cloud data with high security and practical efficiency. Our contributions can be summarized as follows:

1. We propose the concepts of similarity relevance and scheme robustness. We, thus, perform the first attempt to formulate the privacy issue in searchable encryption, and we show server-side ranking based on order-preserving encryption (OPE) inevitably violates data privacy.
2. We propose a TRSE scheme, which fulfills the secure multikeyword top-k retrieval over encrypted cloud data. Specifically, for the first time, we employ relevance score to support multikeyword top-k retrieval.
3. Thorough analysis on security demonstrates the proposed scheme guarantees high data privacy. Furthermore, performance analysis and experimental results show that our scheme is efficient for practical utilization.

The rest of this paper is organized as follows: We provide scenario and related background in Section 2, and then we give the security definitions and problems with existing schemes in Section 3. In Section 4, we present the detailed description of the proposed searchable encryption scheme. In Section 5, we discuss two main issues of our scheme. Sections 6 and 7 give the security analysis and performance analysis, respectively. Related works are reviewed in Section 8. Section 9 concludes this paper.

2 Preliminaries

2.1 Scenario

We consider a cloud computing system hosting data service, as illustrated in Fig. 1, in which three different entities are involved: *cloud server*, *data owner*, and *data user*.

The cloud server hosts third-party data storage and retrieve services. Since data may contain sensitive information, the cloud servers cannot be fully entrusted in protecting data. For this reason, outsourced files must be encrypted. Any kind of information leakage that would affect data privacy are regarded as unacceptable.

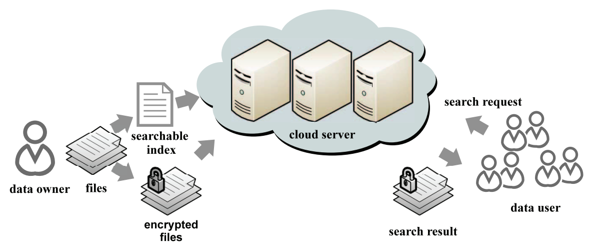


Fig. 1. Scenario of retrieval of encrypted cloud data.

The data owner has a collection of n files $C = \{f_1, f_2, \ldots, f_n\}$ to outsource onto the cloud server in encrypted form and expects the cloud server to provide keyword retrieval service to data owner himself or other authorized users. To achieve this, the data owner needs to build a searchable index I from a collection of l keywords $W = \{w_1, w_2, \ldots, w_l\}$ extracted out of C, and then outsources both the encrypted index I' and encrypted files onto the cloud server.

The data user is authorized to process multikeyword retrieval over the outsourced data. The computing power on the user side is limited, which means that operations on the user side should be simplified. The authorized data user at first generates a query $REQ = \{(w'_1, w'_2, \ldots, w'_s) | w'_i \in W, 1 \leq i \leq s \leq l\}$. For privacy consideration, which keywords the data user has searched must be concealed. Thus, the data user encrypts the query and sends it to the cloud server that returns the relevant files to the data user. Afterward, the data user can decrypt and make use of the files.

2.2 Relevance Scoring

Some of the multikeyword SSE schemes support only Boolean queries, i.e., a file either matches or does not match a query. Considering the large number of data users and documents in the cloud, it is necessary to allow multikeyword in the search query and return documents in the order of their relevancy with the queried keywords.

Scoring is a natural way to weight the relevance. Based on the relevance score, files can then be ranked in either ascendingly or descendingly. Several models have been proposed to score and rank files in IR community. Among these schemes, we adopt the most widely used one *tf-idf* weighting, which involves two attributes-term frequency and inverse document frequency. The *tf-idf* weighting involves two attributes: *Term frequency* and *inverse document frequency*. *Term frequency* $(tf_{t,f})$ denotes the number of occurrences of term t in file f. *Document frequency* (df_t) refers to the number of files that contains term t, and the *inverse document frequency* (idf_t) is defined as: $idf_t = \log \frac{N}{df_t}$, where N denotes the total number of files. Then, the *tf-idf* weighting scheme assigns to term t a weight in file f given by $tf\text{-}idf_{t,f} = tf_{t,f} \times idf_t$. By introducing the IDF factor, the weights of terms that occur very frequently in the collection are diminished and the weights of terms that occur rarely are increased.

2.3 Vector Space Model

While $tf\text{-}idf$ depicts the weight of a single keyword on a file, we employ the *vector space model* to score a file on multikeyword. The vector space model [19] is an algebraic model for representing a file as a vector. Each dimension of the vector corresponds to a separate term, i.e., if a term occurs in the file, its value in the vector is nonzero, otherwise is zero. The vector space model supports multi-term and nonbinary presentation. Moreover, it allows computing a continuous degree of similarity between queries and files, and then ranking files according to their relevance. It meets our needs of top-k retrieval. A query is also represented as a vector $\vec{q}$, while each dimension of the vector is assigned with 0 or 1 according to whether this term is queried. The score of file f on query $q(score_{f,q})$ is deduced by the inner product of the two vectors: $score_{f,q} = \vec{v}_f \cdot \vec{q}$. Given the scores, files can be ranked in order and, therefore, the most relevant files can be found.

3 Problem Statement

The cloud server in our work is considered as "honest-but-curious" [9], a model extensively used in SSE and characterized by that the cloud server will honestly follow the designed protocol but is curious to analyze the hosted data and the received queries to learn extra information.

3.1 Statistic Leakage

Although all data files, indices, and requests are in encrypted form before being outsourced onto cloud, the cloud server can still obtain additional information through statistical analysis. We denote the possible information leakage with *statistic leakage*. There are two possible statistic leakages, including *term distribution* and *interdistribution*. The term distribution of term t is t's frequency distribution of scores on each file $i(i \in C)$. The interdistribution of file f is file f's frequency distribution of scores of each term $j(j \in f)$. Term distribution and interdistribution are specific [10]. They can be deduced either directly from ciphertext or indirectly via statistical analysis over access and search pattern [8]. Here, access pattern refers to which keywords and the corresponding files have been retrieved during each search request, and search pattern refers to whether the keywords retrieved between two request are the same.

Based on our observation, distribution information implies a similarity relationship among terms or files. On the one hand, terms with similar term distribution always have simultaneous occurrence. For instance, obviously, the term "states" are very likely to co-occur with "united" in an official paperwork from the White House, and their term distribution, not surprisingly, are very same in a series of such a kind of paperwork. Given that this paperwork is encrypted but term distribution is not concealed, once an adversary somehow cracks out the plaintext of "united," he can reasonably guess the term that shares a similar term distribution with "united" may be "states." On the other hand, files with similar interdistribution are always the same category, e.g., two medical records from a dental office surely are the same category, and they are very likely to share a similar interdistribution (such as the titles of each entries are the same). Therefore, this specificity should be hidden from an untrusted cloud server.

3.2 κ-Similarity Relevance

To avoid information leakage in server-side ranking schemes, a series of techniques [9], [10] have been employed to flatten or transfer the distribution of relevance scores. These approaches, however, only cover the distribution of individual term or file, ignoring the relevance between them and the violation of data privacy that arouses thereafter. To formulate this problem, we propose the concept of *κ-similarity relevance.*

Definition 3.1. *The file sequence (FS) of term $i(i \in W)$, denoted by $\vec{ts}_i = \{d_1^{'} d_2^{'} \cdots d_k^{'}\}$, is a sequence of files induced by sorting the term vector $\vec{tv}_i = \{d_1 d_2 \cdots d_k\}$ with scores in nondecreasing order.*

Definition 3.2. *The term sequence (TS) of file $j(j \in C)$, denoted by $\vec{fs}_j = \{t_1^{'} t_2^{'} \cdots t_l^{'}\}$, is a sequence of terms induced by sorting the file vector $\vec{fv}_j = \{t_1 t_2 ... t_l\}$ with scores in nondecreasing order.*

Definition 3.3. *Given two sequences (FS or TS) $\vec{v}_1$ and $\vec{v}_2$, their longest common subsequence (LCS) $\vec{lcs}_{v_1 v_2}$, we call $\vec{v}_1$ and $\vec{v}_2$ are relevant by similarity relevance of $\kappa_{v_1 v_2}$ if $\kappa_{v_1 v_2} \geq \kappa_0$, where*

$$\kappa_{v_1 v_2} = \frac{2\|\vec{lcs}_{v_1 v_2}\|}{\|\vec{v}_1\| + \|\vec{v}_2\|}$$

and $\|\vec{v}\|$ denotes the dimensionality of vector $\vec{v}$.

Since $LCS \subseteq FS$, i.e., $\|LCS\| \ll \|FS\|$, thus $\kappa \leq 1$. The similarity relevance denotes how often two terms co-occur with each other in files, e.g., $\kappa_{ij} = 0.5$ means term i occurs in half number of the files that term j occurs. The threshold $\kappa_0(\kappa_0 \in (0,1))$ is set to narrow down the scope. Two terms are regarded as *relevant* if $\kappa \geq \kappa_0$ or *irrelevant* otherwise. The divisor is introduced to avoid terms with longer FSs to get higher κ value. Due to the similarity between TS and FS, we only discuss FS. Note that the IDF value is constant for one term in one file set, so it will not affect the order of files in FS if we omit it here for simplicity.

We have researched in a files set of 45,800 files from the National Science Foundation (NSF) Research Awards Abstracts 1990-2003 [15]. According to the statistic data, in which terms are sorted in nondecreasing order by their term frequencies (the same order as by tf-idf), e.g., the 160th term is "resources," whose FS length is 8703, i.e., "resources" appears in 8,703 files.

Let len_t and $len_t^{'}$ denote the length of LCS of term i with term "resources" before and after one-to-many order-preserving mapping (OPM), respectively, and $ratio_t = \frac{len_t^{'}}{len_t}$. Since different TF values are mapped to nonoverlapping intervals after OPM, the order of files in FS is almost undisturbed. Therefore, the longest common sequence is barely affected. For example, the term "human" is the most relevant term with "resources" in the five terms by $\kappa = 0.885$ before one-to-many OPM, as shown in Table 1. After OPM, however, len_t of the five terms remain almost the same, i.e., $len_t \approx len_t^{'}$, and the corresponding similarity relevance almost maintained. The term "human" still is the most relevant term with "resources."

In a larger range of 218 terms, which are randomly chosen from the top 1,000 terms with the highest term frequencies, as shown in Fig. 2, 98 percent of their $ratio$ are greater than 0.9, i.e., the lengths of their LCSs remain at least 90 percent after OPM. Fig. 3a illustrates the similarity relevance of term "resources" with the 218 terms, from which we can see the distribution of similarity almost changeless. There still are

TABLE 1
Similarity Relevance with "Resources" before and after OPM

term	len_t	κ	len'_t	κ'
directorate	264	0.023	264	0.023
education	4826	0.544	3573	0.403
human	7648	0.885	7647	0.885
provide	1014	0.098	1014	0.098
sciences	2480	0.226	2480	0.226

two terms that can be considered relevant with "resources" after OPM even set κ_0 as high as 0.8. Additionally, we also studied the 20 Newsgroups data set [16], which consists of 20,000 messages taken from 20 Usenet newsgroups. As shown in Fig. 3b, the distribution of similarity relevance of 142 terms with "data" remains almost constant before and after OPM, which agrees with the observation on the NSF file set. More essentially, the order of terms is changeless, i.e., which term is more relevant with a term than other terms has not been concealed.

Moreover, although the expected value of *ratio* can be reduced by properly choosing mapping function, the relative order of them still remains as a result of the order-preserving property. Therefore, the fact that some terms are more relevant than other terms is still exposed after order-preserving one-to-many mapping.

3.3 Scheme Robustness

Given the similarity relevance, which implies terms' co-occurrence, data privacy may be potentially threatened. According to [17], co-occurrence of words, which means how often a word co-occur with another word in a text, is one of the most basic corpus linguistics statistics, and it is measurable through various means including but not limited to pointwise mutual information and the t-score.

This kind of plaintext statistic may violate the privacy of ciphertext if it is not properly handled in encryption scheme design. Consider two terms t_1 and t_2, given they co-occur with each other most of the time in C, and then it can be easily deduced that $\kappa_{t_1t_2} \approx 1$. Conversely, given $\kappa_{t_1t_2} > \kappa_0$, t_1 and t_2 are likely to appear simultaneously by probability of $\kappa_{t_1t_2}$. According to this simultaneous occurrence, if t_1 is known while t_2 is unknown (this is possible due to

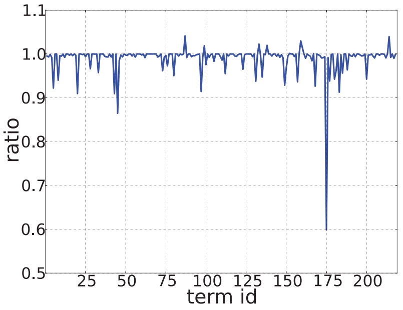


Fig. 2. Ratio of 218 terms with "resources."

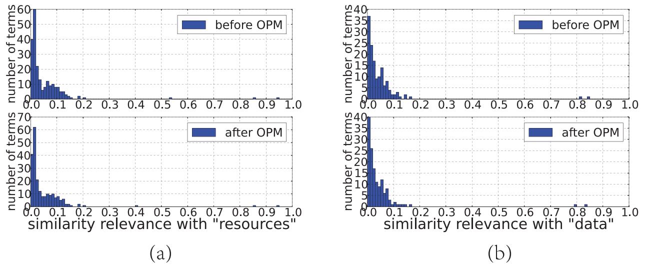


Fig. 3. Distribution of similarity relevance of (a) 218 terms with "resources" before and after OPM in the NSF file set. (b) 142 terms with "data" before and after OPM in the 20 Newsgroups data set.

background information leakage may occur in practical situations, typical examples are available in [4], [5]), then t_2 can be speculated with probability of $p_{t_1t_2}$ by applying bigram frequency attack. For example, according to [12], bigram "of the" occurs much more frequent than any other bigrams based on millions of books from the year 1520 to 2008, i.e., once "of" is known, the word that next to "of" most likely is "the." The total probability to crack t_2 is $p_{total} = \kappa_{t_1t_2} \cdot p_{t_1t_2}$, e.g., assume $\kappa_{t_1t_2} = 0.9$ and $p_{t_1t_2} = 0.6$, then $p_{total} = 0.9 \times 0.6 = 0.54$, which means that once a part of plaintext is known, the rest of ciphertext may be cracked at a probability much greater than that of brute force (typically exponential in ρ, where ρ is the bit length of ciphertext [11]). To formulate this problem, we introduce the concept of *scheme robustness*.

Definition 3.4. *Let Γ denotes the output collection of a searchable encryption scheme, $\forall \zeta \subseteq \Gamma$, $\forall \tau \subseteq \Gamma$, and $\zeta \cap \tau = \emptyset$. Scheme robustness is denoted by*

$$\varrho = \min\left\{\frac{p(\zeta)}{p(\zeta|\tau)}\right\},$$

where $p(\zeta|\tau)$ denotes the crack probability of ζ on condition that τ is known.

It is obvious that $\varrho \leq 1$, and the higher ϱ implies the higher scheme robustness. Variants of OPM have been employed to help shelter the real score distribution from the cloud in existing SSE schemes. It seems that the transferred distribution may be distinct from what it used to be. But it is actually not, due to that, by our definition, the similarity relevance is still a result of the order-preserving property in the presence of one-to-many mapping.

Without losing any generality, suppose that the overall attack complexity of brute force a passage of ciphertext with bit-length of ρ is T_ρ, e.g., 2^ρ, the crack probability is $p_{total} = \frac{1}{T_\rho}$. Assume that each bit is independent, the crack probability of each bit is $p_{bit} = \sqrt[\rho]{p_{total}} = \sqrt[\rho]{\frac{1}{T_\rho}}$. Once item i is brute-forced out, suppose that a k bit item j is similarity relevant with i by $\kappa_{ij} > \kappa_0$ and predictable by probability of p_{ij}, and then item j's crack probability raises to $\kappa_{ij} \cdot p_{ij}$, and

$$\varrho_{ij} = \frac{T_\rho^{-\frac{k}{\rho}}}{\kappa_{ij} \cdot p_{ij}} \ll 1,$$

which means the scheme robustness is low and, thus, more vulnerable to attack.

On the basis of the above discussion, the similarity relevance is specific for terms and, thus, should be properly hidden from the cloud server. However, order-preserving symmetric encryption cannot conceal the similarity relevance, so the scheme robustness of order-preserving symmetric encryption scheme is low. Furthermore, it requires the ciphertext to be order-preserving to support server-side ranking, so server-side ranking is insecure for inevitably leaking sensitive information. For this reason, ranking cannot be entirely left to the cloud server.

4 TRSE Design

Existing SSE schemes employ server-side ranking based on OPE to improve the efficiency of retrieval over encrypted cloud data. However, server-side ranking based on OPE violates the privacy of sensitive information, which is considered uncompromisable in the security-oriented third-party cloud computing scenario, i.e., security cannot be tradeoff for efficiency. To achieve data privacy, ranking has to be left to the user side. Traditional user-side schemes, however, load heavy computational burden and high communication overhead on the user side, due to the interaction between the server and the user including searchable index return and ranking score calculation. Thus, the user-side ranking schemes are challenged by practical use. A more server-siding scheme might be a better solution to privacy issues.

We propose a new searchable encryption scheme, in which novel technologies in cryptography community and IR community are employed, including homomorphic encryption and the vector space model. In the proposed scheme, the data owner encrypts the searchable index with homomorphic encryption. When the cloud server receives a query consisting of multikeywords, it computes the scores from the encrypted index stored on cloud and then returns the encrypted scores of files to the data user. Next, the data user decrypts the scores and picks out the top-k highest-scoring files' identifers to request to the cloud server. The retrieval takes a two-round communication between the cloud server and the data user. We, thus, name the scheme the TRSE scheme, in which ranking is done at the user side while scoring calculation is done at the server side.

4.1 Practical Homomorphic Encryption Scheme

To alleviate the computational burden on the user side, computing work should be at the server side, so we need an encryption scheme to guarantee the operability and security at the same time on server side. Homomorphic encryption allows specific types of computations to be carried out on the corresponding ciphertext. The result is the ciphertext of the result of the same operations performed on the plaintext. That is, homomorphic encryption allows computation of ciphertext without knowing anything about the plaintext to get the correct encrypted result. Although it has such a fine property, the original fully homomorphic encryption scheme, which employs ideal lattices over a polynomial ring [18], is too complicated and inefficient for practical utilization. Fortunately, as a result of employing the vector space model to top-k retrieval, only addition and multiplication operations over integers are needed to compute the relevance scores from the encrypted searchable index. Therefore, we can reduce the original homomorphism in a full form to a simplified form that only supports integer operations, which allows more efficiency than the full form does.

In the fully homomorphic encryption over the integers (FHEI) scheme [11], the approximate integer greatest common divisor (GCD) is used to provide sufficient security, i.e., given a list of integers $\ell = \{i_1, i_2, \ldots, i_n\}$ that are approximate multiples of a hidden integer j, to find the hidden integer j. The approximate GCD problem has been proven hard by Howgrave-Graham [14]. Let m and c denote the plaintext and ciphertext of the integer, respectively. Our encryption scheme can be expressed as the following formulation: $c = pq + 2r + m$, where p denotes the secret key, q denotes the multiple parameter, and r denotes the noise to achieve proximity against brute-force attacks. The public key is $pq + r$.

However, as the scores of items in file vector of searchable index I_p is multibit, the total size of I_c and the computed results will be very large due to the FHEI scheme encrypts 1 bit to $\|p\| + \|q\|$ bit (here $\|p\|$ refers to bit length of p, i.e., $\|p\| = \lceil \log p \rceil$). To downsize the ciphertext and, thus, mitigate the communication overhead, we modify the original FHEI scheme more flexible to meet our needs: $c = pq + xr + m$, where $x = 2^{2\|m\|}$, $p \gg r$, and $r \gg x$ to ensure the correctness of the decryption. Since the size of the result will be doubled after multiplication, the noise parameter x is, thus, required to be at least $2^{2\|m\|}$. Therefore, multibit is considered as a unit for encryption, and the size of ciphertext is significantly reduced, i.e., the size of ciphertext can be reduced down to $\frac{1}{\|m\|}$ of that in original FHEI scheme. For example, assume the value of scores is up to 2^{10}, then the size of ciphertext will be $10(\|p\| + \|q\|)$ for encryption of each bit of m if applying the original FHEI scheme, while only $(\|p\| + \|q\|)$ in the modified FHEI scheme. The modified FHEI scheme guarantees homomorphism property according to the following theorem:

Theorem 4.1. *The modified FHEI scheme is homomorphic for addition and multiplication.*

Proof. Given two plaintext m_1, m_2 and their corresponding ciphertext c_1, c_2 by employing the modified FHEI scheme, where $c_i = pq_i + xr_i + m_i (i = 1, 2)$. Then, we have

$$\begin{aligned} c_1 + c_2 &= (pq_1 + xr_1 + m_1) + (pq_2 + xr_2 + m_2) \\ &= p(q_1 + q2) + x(r_1 + r_2) + (m_1 + m_2)) \end{aligned} \tag{1}$$

$$\begin{aligned} c_1 \cdot c_2 &= (pq_1 + xr_1 + m_1) \cdot (pq_2 + xr_2 + m_2) \\ &= p^2 q_1 q_2 + px(q_1 r_2 + q_2 r_1) + p(q_1 m_2 + q_2 m_1) \\ &\quad + x^2 r_1 r_2 + x(r_1 m_2 + r_2 m_1) + m_1 m_2. \end{aligned} \tag{2}$$

Note that $p \gg r$, $r \gg x$. Thus, from (1) and (2), we can deduce that

$$\begin{aligned} ((c_1 + c_2) \quad \bmod p) \quad \bmod x &= m_1 + m_2 \\ ((c_1 \cdot c_2) \quad \bmod p) \quad \bmod x &= m_1 \cdot m_2. \end{aligned}$$

Hence, Theorem 4.1 is true. □

On the basis of homomorphism property, the encryption scheme can be described as four stages: $KeyGen$, $Encrypt$, $Evaluate$, and $Decrypt$.

- *KeyGen*(λ). The secret key SK is an odd η-bit number randomly selected from the interval $[2^{\eta-1}, 2^{\eta})$. The set of public keys $PK = \{k_0, k_1, \ldots, k_\tau\} \subseteq \{pq + r | q \in [0, 2^\gamma/p), r \in 2Z \cap (-2^\rho, 2^\rho)\}$ and ρ denotes the bit length of r. The noise factor x is randomly selected from the interval $(2^{2\mu}, 2^{2(\mu+1)}]$, where μ denotes the bit length of atomic plaintext. Note that the secret key is used for encryption and the public keys are used for decryption, which are different from the concepts of keys in public-key cryptography.
- *Encrypt*(PK,m). Randomly choose a subset $R \subseteq \{1, 2, \ldots, \tau\}$ and an integer $r' \in (-2^{2\rho}, 2^{2\rho})$, and then return ciphertext $c = m + xr' + \sum_{i \in R} k_i$.
- *Evaluate*($c_1, c_2, \ldots, c_t$). Apply the binary addition and multiplication gates to the t ciphertext c_i, perform all necessary operations, and then return the resulting integer χ.
- *Decrypt*(p, χ). Output $m' = (\chi \bmod p) \bmod x$.

Here, $\rho = \lambda$, $\eta = O(\lambda^2)$, and $\gamma = O(\lambda^5)$. The modified FHEI scheme is relatively time-consuming, so we only employ it to encrypt the searchable index I, while the file set C can be encrypted with other symmetric encryption scheme. Note that the Evaluate stage sets no limit to how many addition or multiplication operations can be executed without recryption. In fact, the ciphertext of an integer, which is another integer, can be applied to as many evaluations as needed.

4.2 Framework of TRSE

The framework of TRSE includes four algorithms: $Setup$, $IndexBuild$, $TrapdoorGen$, $ScoreCalculate$, and $Rank$.

- *Setup*(λ). The data owner generates the secret key and public keys for the homomorphic encryption scheme. The security parameter λ is taken as the input, the output is a secret key SK, and a public key set PK.
- *IndexBuild*(C,PK). The data owner builds the secure searchable index from the file collection C. Technologies from IR community like stemming are employed to build searchable index I from C, and then I is encrypted into I' with PK, output the secure searchable index I'.
- *TrapdoorGen*(REQ, PK). The data user generates secure trapdoor from his request REQ. Vector T_ω is built from user's multikeyword request REQ and then encrypted into secure trapdoor T_ϖ with public key from PK, output the secure trapdoor T_ϖ.
- *ScoreCalculate*(T_ϖ, I'). When receives secure trapdoor T_ϖ, the cloud server computes the scores of each files in I' with T_ϖ and returns the encrypted result vector $\aleph$ back to the data user.
- *Rank*($\aleph$,SK,k). The data user decrypts the vector $\aleph$ with secret key SK and then requests and gets the files with top-k scores.

Note that λ is only involved in the Setup algorithm, and the Setup algorithm needs to be processed only once by the data owner, λ, thus, is a constant integer for one individual application instance. The whole framework can be divided into two phases: $Initialization$ and $Retrieval$. The Initialization phase includes *Setup* and *IndexBuild*. The Setup stage involves the secure initialization, while the IndexBuild stage involves operations on plaintext. For security concerns, the vast majority of work should only be done by the data owner. Moreover, for convenience of retrieve, we modify the original vector space model by adding each vector v_i a head node id_i at the first dimension of v_i to store the identifier of f_i. In this way, the correspondence between scores and files is established. The details of the Initialization phase are as follows:

Initialization Phase:

1. The data owner calls KeyGen(λ) to generate the secret key SK and public key set PK for the homomorphic encryption scheme. Then the data owner assigns SK to the authorized data users.
2. The data owner extracts the collection of l keywords, $W = \{w_1, w_2, \ldots, w_l\}$, and their TF and IDF values out of the collection of n files, $C = \{f_1, f_2, \ldots, f_n\}$. For each file $f_i \in C$, the data owner builds a $(l+1)$-dimensional vector $v_i = \{id_i, t_{i,1}, t_{i,2}, \ldots, t_{i,l}\}$, where $t_{i,j} = tf\text{-}idf_{w_j,f_i} (1 \le j \le l)$. The searchable index $I = \{v_i | 1 \le i \le n\}$.
3. The data owner encrypts the searchable index I to secure searchable index $I' = \{v'_i | 1 \le i \le n\}$, where $v'_i = \{id'_i, t'_{i,1}, t'_{i,2}, \ldots, t'_{i,l}\}$, $id'_i = Encrypt(R_{i,0}, id_i)$ and $t'_{i,j} = Encrypt(R_{i,j}, t_{i,j})(R_{i,0} \subseteq PK, R_{i,j} \subseteq PK, 1 \le j \le l)$.
4. The data owner encrypts $C = \{f_1, f_2, \ldots, f_n\}$ into $C' = \{f'_1, f'_2, \ldots, f'_n\}$ with other cryptology schemes, and then outsources C' and I' to the cloud server.

The Retrieval phase involves *TrapdoorGen*, *ScoreCalculate*, and *Rank*, in which the data user and the cloud server are involved. As a result of the limited computing power on the user side, the computing work should be left to server side as much as possible. Meanwhile, the confidentiality privacy of sensitive information cannot be violated. According to the discussion in Section 3, the ranking should be left to the user side while the cloud server still does most of the work without learning any sensitive information. Note that the file vector v'_j in I' is $(l+1)$-dimensional while the request vector is l-dimensional, and the score is the inner product of $v'_j[1:l]$, the later l-dimensional sub vector of v'_j, with the secure trapdoor T_ϖ. The details of the Retrieval phase are as follows:

Retrieval Phase:

1. The data user generates a set of keywords $REQ = \{w'_1, w'_2, \ldots, w'_s\}$ to search, and then the query vector $T_\omega = \{m_1, m_2, \ldots, m_l\}$ is generated in which $m_i = 1 (1 \le i \le l)$ if $t_i \in REQ$ or $m_i = 0$ otherwise. After that, T_ω is encrypted into trapdoor $T_\varpi = \{c_1, c_2, \ldots, c_l\}$, where $c_i = Encrypt(R, m)$ and $S \subseteq PK$, and then the user sends T_ϖ to the cloud server.
2. For each file vector $v'_j (0 \le j \le n)$ in I', the cloud server computes the inner product $p'_j = v'_j[1:l] \cdot T_\varpi$ with

```
Algorithm 1 TOPKSELECT($source, k$)
Input:
    list $source$ to be selected
    number $k$
Initialization:
    Set $topk \leftarrow \emptyset; topkid \leftarrow \emptyset$;
Iteration:
 1: for all $item \in source$ do
 2:     INSERT($topk, (item, itemindex)$)
 3: end for
 4: for all $tuple \in topk$ do
 5:     $topkid.append(tuple[1])$
 6: end for
Output:
    $topkid$
```

(a)

```
Algorithm 2 INSERT($topk, (item, itemindex)$)
Input:
    list $topk$ to store the top-$k$ scoring item
    tuple $(item, itemindex)$
Iteration:
 1: if $len(topk) < k$ then
 2:     insert $(item, itemindex)$ into $topk$ in nondecreasing
        order of $item$
 3: else
 4:     for all $element \in topk$ do
 5:         if $item < element[0]$ then
 6:             continue
 7:         else
 8:             discard $topk[0]$, insert $(item, itemindex)$ into
                $topk$ in nondecreasing order of $item$
 9:         end if
10:     end for
11: end if
```

(b)

Fig. 4. (a) Algorithm TOPKSELECT. (b) Algorithm INSERT.

modular reduction and then compresses and returns the result vector $\aleph' = \{(id'_1, p'_1), (id'_2, p'_2), \ldots, (id'_n, p'_n)\}$ to the data user.

3. The data user decrypts $\aleph'$ into $\aleph = \{(id'_1, p_1), (id'_2, p_2), \ldots, (id'_n, p_n)\}$, where $p_j = Decrypt(SK, p'_j)$. Then TOPKSELECT($\aleph$,k) is invoked to get the top-k highest-scoring files' identifiers $\{i_1, i_2, \ldots, i_k\}$ and sends it to the cloud server.
4. The cloud server returns the encrypted k files $\{f_{i_1}, f_{i_2}, \ldots, f_{i_k}\}$ to the data user.

As a result of the limited computing power on the user side, we are mostly concerned about the complexity of ranking. Since the decryption of $\aleph$ can be accomplished in $O(n)$ time, the only function that could influence the time complexity of ranking is the top-k select algorithm, i.e., TOPKSELECT algorithm. The details of TOPKSELECT algorithm are shown in Fig. 4a. Since the complexity of the INSERT algorithm is $O(k)$, as illustrated in Fig. 4b, the overall complexity of TOPKSELECT algorithm is $O(nk)$. Note that k, which denotes the number of files that are most relevant to the user's interest, is generally very small compared to the total number of files. In case of large value of k, the complexity of the TOPKSELECT algorithm can be easily reduced to $O(n \log k)$ by introducing a fixed-size min-heap.

5 Discussion

Based on the current research, two issues remain to be addressed in secure multikeyword top-k retrieval over encrypted cloud data.

5.1 Efficiency Improvement

The main appeal of the modified FHEI that we employ in the TRSE scheme is its conceptual simplicity compared to Gentry's [18]. This simplicity is achieved at the cost of a large key size. Although optimizations like modular reduction and compression can be employed to reduce the size of ciphertext, the key size is still too large for the practical system.

As discussed in Section 4, the user encrypts his trapdoor and sends the ciphertext to the cloud server. Therefore, the communication overhead will be very high if the encrypted trapdoor size is too large. To solve this problem and, thus, improve efficiency, a tradeoff of the security of search pattern may be needed unless a new encryption scheme that provides more reasonable ciphertext size becomes available. Researchers from cryptography community [29], [30] have made several attempts to move toward practical fully homomorphic encryption over integers. These progresses indicate that the efficiency of the TRSE scheme can be further improved.

5.2 Enable Update

In a practical cloud computing system, data updates like adding or deleting files lead to a new challenge to the searchable encryption scheme. Since data updates may be frequent, e.g., doctors update patients' medical records every day in a medical system and users update their photo albums weekly or even daily, it is necessary to consider the efficiency of update in searchable encryption design.

In the presence of an update, both the file itself and the searchable index require update operation. The vector space model employed in the TRSE scheme relies on the tf-idf weight, in which the inverse document frequency (idf) factor depends on the number of files that contain a keyword. When a file is added or deleted, the idf factor may change for a keyword. To avoid updating all the searchable index when updates occur, the file vectors should be independent to each other. Since the searchable index is built for each file, a possible solution is to only store tf values in file vectors and add another auxiliary vector to store idf values for each keyword. In this way, update is limited to the auxiliary vector, rather than all searchable index. The expense is that the tf-idf weights need to be calculated to get the relevance scores during each search request. Since the calculation is on the server side and the computing power on the server side is high, the overall efficiency is almost immune to the update.

6 Security Analysis

We evaluate the security of the proposed scheme by analyzing its fulfillment of the security guarantees of traditional SSE and the privacy requirements discussed in Section 3. First, the cloud server should not learn either the plaintext of the data files, index, and the searched keywords or their statistic information including access pattern, search pattern, and distribution. Second, the cloud server should not learn the similarity relevance of terms or files so that the scheme has high robustness. We start with the security analysis of the modified FHEI encryption scheme. Then, we analyze the security of TRSE scheme.

6.1 Security Analysis for the Modified FHEI Scheme

The security of the modified FHEI encryption is equivalent to the hardness of solving the approximate-gcd problem in Number Theory [22]. Namely, given a set of integers, $X = \{x_0, x_1, \ldots, x_t\}$, where $x_i = pq_i + r_i$, all randomly chosen close to multiples of an η-bit large integer p, find this "common near divisor" p. The known attacks on the approximate-gcd problem includes *brute-force attack*, *the continued fractions attack*, [20] and *Howgrave-Grahams approximate-gcd attack* [14]. We evaluate the security of the TRSE scheme under the three attacks, respectively, as follows:

The brute-force attack is a natural way to solve the normal approximate-gcd problem. The basic idea is to speculate r_i and r_j, then check whether the speculation is right with a gcd calculation. Specifically, when $t = 2$, for $r_1', r_2' \in (-2^\rho, 2^\rho)$, set $x_1' = x_1 - r_1'$, $x_2' = x_2 - r_2'$, $p' = gcd(x_1', x_2')$, if p' is an η-bit integer, and then p' is a possible solution. By the brute-force attack, the solution will certainly be found. The complexity of the brute-force attack is $O(2^{2\rho})$. For arbitrary $t > 2$, the complexity grows to $O(t^3 2^{2\rho})$ for checking every pair in X, which is too time-consuming to implement.

In the continued fractions attack, a sequence of integer pairs is obtained (y_i, z_i) such that $|\frac{x_1}{x_2} - \frac{y_i}{z_i}| < \frac{1}{z_i^2}$. Since $\frac{q_1}{q_2}$ is a good approximation of $\frac{x_1}{x_2}$, i.e.,

$$\left|\frac{x_1}{x_2} - \frac{q_1}{q_2}\right| \approx 0,$$

(q_1, q_2) probably occurs in the sequence. If so, p can be recovered by $p = [\frac{x_1}{q_1}]$. The $|\frac{x_1}{x_2} - \frac{q_1}{q_2}|$ in our scheme, however, is not small enough to be recovered by this attack. Specifically,

$$\left|\frac{x_1}{x_2} - \frac{q_1}{q_2}\right| = \left|\frac{q_2 r_1 - q_1 r_2}{q_2(pq_2 + r_2)}\right| \approx \left|\frac{q_2 r_1 - q_1 r_2}{p}\right| \cdot \frac{1}{q_2^2},$$

since $|\frac{q_2 r_1 - q_1 r_2}{p}| \gg 1$ according to the parameter selection in our scheme, the pair (q_1, q_2) cannot be obtained in the sequence. Therefore, the continued fractions attack does not impair our scheme.

Howgrave-Graham gives a lattice attack on the multi-element approximate-gcd problem. In this attack, when $t = 2$, the relevant lattice may contain exponential vectors unrelated to the approximate-gcd solution, so that lattice reduction turns out to be in vain. For arbitrary $t > 2$, the time needed to guarantee a 2^η approximation is roughly $2^{\frac{\gamma}{\eta^2}}$, resulting the overall computing complexity is $\Omega(2^\lambda)$, which is difficult to crack. In conclusion, the modified FHEI scheme guarantees sufficient security.

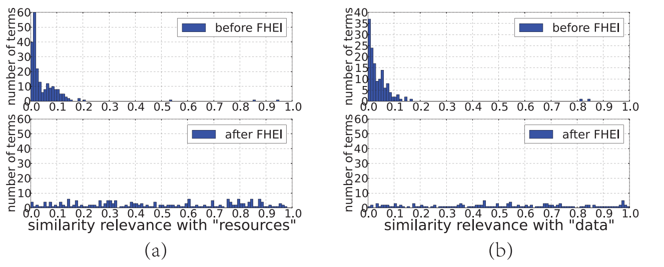


Fig. 5. Distribution of similarity relevance of (a) 218 terms with "resources" before and after FHEI in the NSF file set, and. (b) 142 terms with "data" before and after FHEI in the 20 Newsgroups data set.

6.2 Security Analysis for the TRSE Scheme

Compared to the traditional SSE, our TRSE scheme reduces the information leakage asymptotically equal to zero. First, for access pattern and search pattern, e.g., if the same keyword t_i is requested in two queries REQ_1 and REQ_2, then $m_{1i} = m_{2i} = 1$ in the corresponding query vector $T_{\omega 1}$ and $T_{\omega 2}$. After that, m_{1i} and m_{2i} are encrypted into different ciphertext by employing $Encrypt(R_{1i}, m_{1i})$ and $Encrypt(R_{2i},\ m_{2i})$. Namely, as well as same keywords in different queries, the encryptions of different keywords in same queries are independent, i.e., which keywords have been retrieved are conceale; thus, the access pattern and search pattern are secure.

Second, since the modified FHEI encryption requires no order-preserving property, the scores in the secure searchable index I' are encrypted into random intervals according to the randomly selected subset of PK. For a keyword t_i, a term vector, $v_i = \{f_{i1}, f_{i2}, \ldots, f_{in}\}$, can be deduced from I', where the f_{ij} denotes t_i's encrypted tf-idf weighting on file f_i. As stated in Section 2, the tf-idf weighting represents the TF and IDF values directly, which are specific not only in value but also in distribution. After FHEI encryption, v_i changes into $v_i' = \{f_{i1}', f_{i2}', \ldots, f_{in}'\}$, and the original order are totally disrupted. Since the interdistribution is similar to the issue of term distribution, both the term distribution and the interdistribution are secure.

Third, the random mapping disrupts the original order of the files in FS; thus, the common subsequence of two terms is randomly disrupted. Thus, the resulting similarity relevance cannot be retained after FHEI encryption. As shown in Fig. 5a, the distribution of similarity relevance of "resources" with other 218 terms is flattened after FHEI encryption, e.g., only 2 terms are relevant with "resources" before FHEI, while 42 terms can be considered relevant after FHEI (set $\kappa_0 = 0.8$). The comparative experiment on the 20 Newsgroups data set also demonstrates the similar conclusion, e.g., only 1 term is relevant with "data" before FHEI, while 27 terms can be considered relevant after FHEI, which is shown in Fig. 5b. Generally speaking, as Table 2 shows, the κ value randomly changes, e.g., the κ value of "human" changes from 0.885 to 0.085, while the κ value of "directorate" changes from 0.023 to 0.283. In other words, which term is more relevant to "resources" than other terms is concealed. As a result of the flattening of similarity relevance, our scheme robustness reaches the theoretical upper bound:

TABLE 2
Similarity Relevance with "Resources" before and after FHEI

term	len	κ	len$'$	κ'
directorate	264	0.023	3248	0.283
education	4826	0.544	6273	0.707
human	7648	0.885	711	0.082
provide	1014	0.098	1132	0.109
sciences	2480	0.226	45	0.004

$$\varrho = \min\left\{\frac{p(\zeta)}{p(\zeta|\tau)}\right\} = \min\left\{\frac{p(\zeta)}{p(\zeta)}\right\} = 1.$$

In general, the TRSE scheme we proposed is adequate to overcome the inevitable compromise of security caused by the OPE-based traditional server-side ranking SSE schemes. Specifically, TRSE conceals the similarity relevance and retains scheme robustness. Therefore, the TRSE scheme guarantees high data privacy.

7 Performance Analysis

We conducted a thorough experimental evaluation of the proposed TRSE scheme on the file set of National Science Foundation Research Awards Abstracts 1990-2003 [15]. Our experiment environment includes a user and a server. The user uses C language on a Windows 7 machine with Core 2 Duo CPU running at 2.0 GHz, and the server uses C language on a linux machine with Xeon E5620 CPU running at 2.4 GHz. The user acts as a data owner and a data user, and the server acts as a cloud server.

7.1 Performance of the Initialization Phase

The Initialization phase includes Setup and IndexBuild, and needs to be processed only once by the data owner. According to the parameter selection in the modified FHEI scheme, the complexity of the Setup stage is $O(\lambda^{10})$. Note that λ is a fixed integer for a realistic scheme, e.g., $\lambda = 128$ in our experiment, so the Setup stage costs a fixed time.

The IndexBuild stage includes building searchable index I and then encrypt I into I'. To build I, several technologies from IR community, e.g., stemming for reducing inflected words to their root words, can be employed to improve efficiency, which is not in the scope of this paper. To improve the computing efficiency, the tf-idf values are rounded to integers when building I, which does not affect the retrieve accuracy. Note that encryption needs only the addition operation, so the complexity of encrypting I is $O(nl)$, where n denotes the number of files and l denotes the number of keywords.

7.2 Performance of the Retrieval Phase

The Retrieval phase includes TrapdoorGen, ScoreCalculate, and Rank. The Rank stage can be subdivided into *ResultDecrypt* and *TopK*. Since the Initialization phase needs to be processed only once and the Retrieval phase can be processed many times, the overall efficiency is, thus, dominated by the Retrieval phase, and we compared the efficiency of this phase between our approach with a server-side ranking SSE approach adopted from [25]. As our approach employed two-round communication, which is different from any server-side ranking SSE schemes, there are only two shared stages including TrapdoorGen and ScoreCalculate that we can take for comparison.

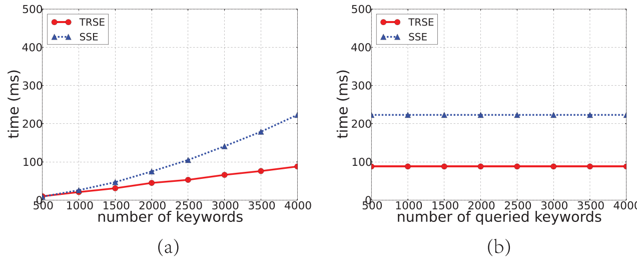


Fig. 6. (a) The time to generate trapdoor on different scale of keyword sets. (b) The time to generate trapdoor for different number of queried keywords, the number of keywords in the keyword set is $l = 4{,}000$.

The TrapdoorGen stage needs $O(l)$ time to build the l-dimension query vector T_ω from the multikeyword request. To encrypt T_ω to T_ϖ, each dimension needs to be encrypted. Since the encryption requires only addition operations, the complexity of this stage is $O(l)$. Fig. 6a shows the time cost to generate a trapdoor of different lengths. For example, it costs 88 ms to generate a trapdoor over a file set containing 4,000 different keywords with TRSE, while the SSE scheme needs 223 ms to do the same work. The comparative experiment data on the SSE scheme show that our scheme is more efficient in this stage. Specifically, TRSE reduces the time cost from a exponential growth down to a linear growth against the increment of keyword set size. Besides, the length of query vector is fixed to l, so the time to generate trapdoor is changeless when the number of queried keywords increases. Specifically, TRSE costs about half of the time of the SSE scheme in this stage when the number of keywords in the keyword set is $l = 4{,}000$, as illustrated in Fig. 6b.

In the ScoreCalculate stage, the cloud server calculates the inner product of T_ϖ with each row in I'. To calculate the inner product, each row needs l multiplications and l-1 additions. Therefore, the complexity of ScoreCalculate is $O(nl)$. Fig. 7a shows the time cost to calculate scores on different scale of file set. For example, it costs 4.5 s to calculate scores on a file

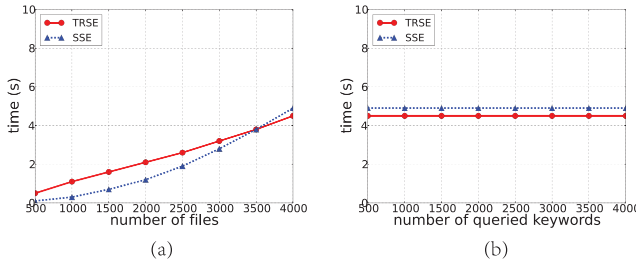


Fig. 7. (a) The time to calculate scores on different scale of file sets, the number of keywords in the keyword set is 1,000. (b) The time to calculate scores for different number of queried keywords. Here, the number of files in the file set is $n = 4{,}000$.

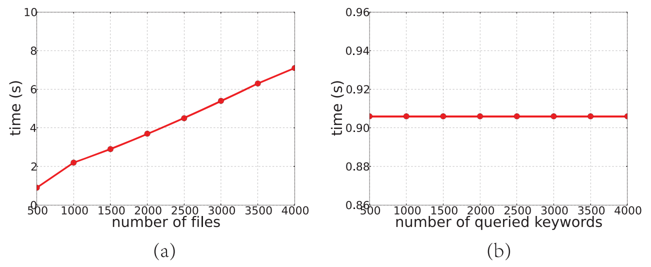


Fig. 8. (a) The time to decrypt the result vector on different scale of file sets. (b) The time to decrypt the result vector for different numbers of queried keywords; the number of files in the file set is $n = 500$.

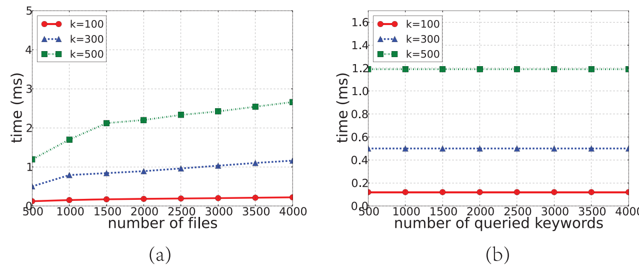


Fig. 9. (a) The time to select the top-k files' identifiers on different scales of file sets. (b) The time to select the top-k files' identifiers for different numbers of queried keywords; the number of files in the file set is $n = 500$.

set of 4,000 files and 1,000 keywords, while the SSE scheme needs 4.9 s to do the same work. In fact, the comparative experiment data on the SSE scheme show that our scheme reduces the time cost from a exponential growth down to a linear growth against the increment of file set size. Since the scale of the calculation is fixed to the scale of the file set, the time cost is changeless when the number of queried keywords increases. Specifically, TRSE performs better than the SSE scheme after the size of file set grows beyond 3,500, which is shown in Fig. 7b. Moreover, the difference of computing power between the server side and the user side can be much greater than that in our experimental environment in general, so the time to calculate scores can probably be further reduced in practice.

In the ResultDecrypt stage, the data user decrypts the n-dimension result vector to get the plaintext of the scores. Since the size of the result vector depends only on the number of files in the file set and the decryption of each dimension costs constant number of modular computations, the overall complexity of decryption is $O(n)$. Fig. 8a shows the time cost to decrypt the result vector on different scales of file set when $k = 50$. For example, it costs 0.905 s to decrypt the result vector on a file set of 500 files, whereas it costs 2.106 s for 1,000 files. Similar to the pervious two stages, the number of query keywords does not influence the time cost either, as shown in Fig. 8b.

In the TopK stage, the data user goes over the decrypted result to get the top-k highest scoring files' identifers. Fig. 9a shows the time cost to select the top-k files' identifiers on different scale of file set by the TOPKSELECT algorithm. For example, it costs 0.108 ms to select the top-100 files' identifiers from a file set of 500 files, whereas it costs 2.188 ms for the top-500 from 2,000 files. Although the time cost is low, there is still room for reduction in case of large k. As discussed in Section 4.2, the complexity of top-k selection algorithm can be easily modified to $O(n \log k)$ by introducing a fixed-size min-heap. Fig. 9b demonstrates that the time cost of this stage is independent to the number of queried keywords. From the experimental data, we can see that the decryption is more time-consuming than the time cost of top-k selection. Since the increment of k affects only the time cost of the TopK stage, which accounts for only a small fraction of the overall time cost, its impact on the overall time cost of the Retrieval phase is negligible.

Although the two-round communication subdivides the Retrieval phase with two additional stages and thus introduces extra overhead, our approach still guarantees practical efficiency while scheme robustness and security are significantly improved. Specifically, the scale of computing on the user side is smaller than that on the server side, i.e., the majority of computing is done by the cloud server. Moreover, as previously discussed, the increased number of query keywords does not degrade performance of the Retrieval phase, which introduces the TRSE scheme good scalability.

7.3 Communication Overhead

According to Section 4.1, binary addition and multiplication operations are involved in the TRSE scheme. The size of ciphertext doubles after multiplication. To further downsize the ciphertext and reduce the communication overhead, we apply a couple of optimizations in the TRSE scheme. During the Evaluate stage, modular reduction [11] can help to keep the size of evaluated ciphertexts at the same length as the original ciphertexts by executing a sequence of modular reductions when the size of ciphertexts grows beyond 2^{λ}. Even though modular reduction is employed, however, the size of ciphertexts is still very large, e.g., $\Theta(\lambda^5)$ bits under the suggested parameters. It can be further shrunk to the size of a RSA modulus [21] by ciphertext compression, e.g., 1,024 bits for one dimension, reducing the communication complexity of our scheme dramatically.

The $tf - idf$ values are less than 1,000 in our experimented file set, so 10 bits is enough for each dimension of file vector in I. The size of ciphertext grows to $1{,}024 \div 10 = 102.4$ times of the original size. For example, if considering a file set of 500 files and 1,000 distinct keywords, then the size of one encrypted result that needs to be sent back to the user is $500 \times 1{,}024$ bit $= 62.5$ KB. Taking into account the data transfer rate of the widely used Internet, e.g., 800 KB/s, the communication can be done within 78.125 ms. In traditional user-side ranking SSE approaches in which the cloud server needs to return the entire searchable index to the user, an index size of about $500 \times 1,000 \times 1{,}024$ bit ≈ 61 MB needs to be transferred and then all the scores are calculated on the user side. Compared with that, TRSE vastly reduces the communication overhead and the computing burden on the user side.

8 Related Work

Traditional searchable encryption are investigated in [8], [22], [23] focusing on security definitions and encryption efficiency, and these works support only Boolean keyword retrieval without ranking. Swaminathan et al. [24] explored secure rank-ordered retrieval with improved searchable encryption in the scenario of the data center. They built a framework for privacy-preserving top-k retrieval, including secure indexing and ranking with OPE. Zerr et al. [10] proposed a ranking model to guarantee privacy-preserving document exchange among collaboration groups, which allows for privacy-preserving top-k retrieval from an outsourced inverted index. They proposed a relevance score transformation function to make relevance scores of different terms indistinguishable and such that improves the security of the indexed data. Wang et al. [9] explored top-k retrieval over encrypted data in cloud computing. On the basis of SSE, they proposed the one-to-many OPM to further improve the efficiency while security guarantee and retrieval accuracy are slightly weakened. However, these schemes support only single keyword retrieval.

Considering the large number of data users and documents in the cloud, it is necessary to allow multikeyword in the search request and return the most relevant documents in the order of their relevancy with these keywords. Some existing works [27], [28] proposed several schemes supporting Boolean multikeyword retrieval. Cao et al. [25] made the first attempt to define and solve the problem of top-k multikeyword retrieval over encrypted cloud data. They employed coordinate matching and inner product similarity to measure and evaluate the relevance scoring. Hu et al. [26] employed homomorphism to preserve the data privacy. They devised a secure protocol for processing k-nearest-neighbor (kNN) index query, thus preserving both the data privacy of the owner and the query privacy of the client. These two schemes employed Boolean representation in their searchable index, i.e., 1 denotes the corresponding term exists in the file and 0 otherwise. Thus, files that share queried keywords have the same score, a situation that is far from precise, thus, weakens the effectiveness of data utilization. Since all these server-side schemes employ server-side ranking based on OPE, the security is compromised. We, therefore, focus on the security, an issue the above schemes fail to address.

9 Conclusion

In this paper, we motivate and solve the problem of secure multikeyword top-k retrieval over encrypted cloud data. We define similarity relevance and scheme robustness. Based on OPE invisibly leaking sensitive information, we devise a server-side ranking SSE scheme. We then propose a TRSE scheme employing the fully homomorphic encryption, which fulfills the security requirements of multikeyword top-k retrieval over the encrypted cloud data. By security analysis, we show that the proposed scheme guarantees data privacy. According to the efficiency evaluation of the proposed scheme over a real data set, extensive experimental results demonstrate that our scheme ensures practical efficiency.

Acknowledgments

This work was supported in part by the Shanghai Pu Jiang Talents Program (10PJ1405800) and the National Natural Science Foundation of China (No. 61170238, 60903190, 61027009).

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and M. Zaharia, "A View of Cloud Computing," *Comm. ACM*, vol. 53, no. 4, pp. 50-58, 2010.

[2] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," http://www.techcrunch.com/2006/12/28/gmail-disasterreports-of-mass-email-deletions/, Dec. 2006.

[3] Amazon.com, "Amazon s3 Availability Event: July 20, 2008," http://status.aws.amazon.com/s3-20080720.html, 2008.

[4] RAWA News, "Massive Information Leak Shakes Washington over Afghan War," http://www.rawa.org/temp/runews/2010/08/20/massive-information-leak-shakes-washington-over-afghan-war.html, 2010.

[5] AHN, "Romney Hits Obama for Security Information Leakage," http://gantdaily.com/2012/07/25/romney-hits-obama-for-security-information-leakage/, 2012.

[6] Cloud Security Alliance, "Top Threats to Cloud Computing," http://www.cloudsecurity alliance.org, 2010.

[7] C. Leslie, "NSA Has Massive Database of Americans' Phone Calls," http://usatoday30.usatoday.com/news/washington/2006-05-10/, 2013.

[8] R. Curtmola, J.A. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *Proc. ACM 13th Conf. Computer and Comm. Security (CCS)*, 2006.

[9] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," *Proc. IEEE 30th Int'l Conf. Distributed Computing Systems (ICDCS)*, 2010.

[10] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k Retrieval from a Confidential Index," *Proc. 12th Int'l Conf. Extending Database Technology: Advances in Database Technology (EDBT)*, 2009.

[11] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," *Proc. 29th Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques*, H. Gilbert, pp. 24-43, 2010.

[12] M. Perc, "Evolution of the Most Common English Words and Phrases over the Centuries," *J. Royal Soc. Interface*, 2012.

[13] O. Regev, "New Lattice-Based Cryptographic Constructions," *J. ACM*, vol. 51, no. 6, pp. 899-942, 2004.

[14] N. Howgrave-Graham, "Approximate Integer Common Divisors," *Proc. Revised Papers from Int'l Conf. Cryptography and Lattices (CaLC' 01)*, pp. 51-66, 2001.

[15] "NSF Research Awards Abstracts 1990-2003," http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html, 2013.

[16] "20 Newsgroups," http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html, 2013.

[17] S. Gries, "Useful Statistics for Corpus Linguistics," *A Mosaic of Corpus Linguistics: Selected Approaches*, Aquilino Sanchez Moises Almela, eds., pp. 269-291, Peter Lang, 2010.

[18] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," *Proc. 41st Ann. ACM Symp. Theory of computing (STOC)*, pp. 169-178, 2009.

[19] D. Dubin, "The Most Influential Paper Gerard Salton Never Wrote," *Library Trends*, vol. 52, no. 4, pp. 748-764, 2004.

[20] A. Cuyt, V. Brevik Petersen, B. Verdonk, H. Waadeland, and W.B. Jones, *Handbook of Continued Fractions for Special Functions.* Springer Verlag, 2008.

[21] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, pp. 856-887. MIT Press and McGraw-Hill, 2001.

[22] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," *Proc. IEEE Symp. Security and Privacy*, 2000.

[23] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, "Public-Key Encryption with Keyword Search," *Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (Eurocrypt)*, 2004.

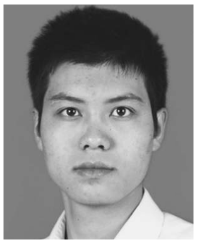
[24] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A.L. Varna, S. He, M. Wu, and D.W. Oard, "Confidentiality-Preserving Rank-Ordered Search," *Proc. Workshop Storage Security and Survivability*, 2007.

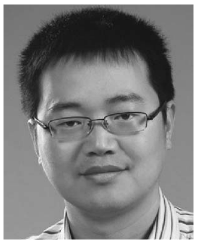
[25] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multikeyword Ranked Search over Encrypted Cloud Data," *Proc. IEEE INFOCOM*, 2011.

[26] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing Private Queries over Untrusted Data Cloud through Privacy Homomorphism," *Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE)*, 2011.

[27] P. Golle, J. Staddon, and B. Waters, "Secure Conjunctive Keyword Search over Encrypted Data," *Proc. Second Int'l Conf. Applied Cryptography and Network Security (ACNS),* pp. 31-45, 2004.
[28] L. Ballard, S. Kamara, and F. Monrose, "Achieving Efficient Conjunctive Keyword Searches over Encrypted Data," *Proc. Seventh Int'l Conf. Information and Communications Security (ICICS),* 2005.
[29] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully Homomorphic Encryption over the Integers with Shorter Public Keys," *CRYPTO '11: Proc. 31st Ann. Conf. Advances in Cryptology,* 2011.
[30] N. Smart and F. Vercauteren, "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes," *Proc. 13th Int'l Conf. Practice and Theory in Public Key Cryptography (PKC),* 2010.

Jiadi Yu received the PhD degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2007. He is an assistant professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. From 2009 to 2011, he worked as a postdoc at Stevens Institute of Technology. His research interests include networking, mobile computing, cloud computing, and wireless sensor networks. He is a member of the IEEE and the IEEE Computer Society.


Peng Lu received the bachelor's degree in software engineering from Huzhong University of Science and Technology, Wuhan, China, in 2011, and the master's degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include cloud computing and mobile computing.


Yanmin Zhu received the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology in 2007. He is an associate professor in the Department of Computer Science and Engineering at Shanghai Jiao Tong University. His research interests include wireless sensor networks and mobile computing. Before that, he was a research associate with the Department of Computing at Imperial College London. He is a member of the IEEE and the IEEE Communication Society.

Guangtao Xue received the PhD degree in computer science from Shanghai Jiao Tong University in 2004. He is an associate professor in the Department of Computer Science and Engineering at the Shanghai Jiao Tong University. His research interests include mobile networks, social networks, sensor networks, vehicular networks, and distributed computing. He is a member of the IEEE Computer Society and the Communication Society.

Minglu Li received the graduate degree from the School of Electronic Technology, University of Information Engineering, in 1985 and the PhD degree in computer software from Shanghai Jiao Tong University (SJTU) in 1996. He is a full professor and the vice chair of the Department of Computer Science and Engineering and the director of the Grid Computing Center at SJTU. Currently, his research interests include grid computing, services computing, and sensor networks.