Achieving Authorized and Ranked Multi-keyword Search over Encrypted Cloud Data

Hongwei Li[†] [§], Dongxiao Liu[†], Kun Jia[†], and Xiaodong Lin[SS]
[†]School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
[§]State Key Laboratory of Information Security (Institute of Information Engineering,
Chinese Academy of Sciences, Beijing 100093)
[SS]Faculty of Business and Information Technology, University of Ontario Institute of Technology, Canada

***Abstract*—In cloud computing, it is important to protect user data. Thus, data owners usually encrypt their data before outsourcing them to the cloud server for security and privacy concerns. At the same time, very often users need to find data for specific keywords of interest to them. This motivates the research on the searchable encryption technique, which allows the search user to search over the encrypted data. Many mechanisms have been proposed, and are mainly focusing on the symmetric searchable encryption (SSE) technique. However, they do not consider the search authorization problem that requires the cloud server only to return the search results to authorized users. In this paper, we propose an authorized and ranked multi-keyword search scheme (ARMS) over encrypted cloud data by leveraging the ciphertext policy attribute-based encryption (CP-ABE) and SSE techniques. Security analysis demonstrates that the proposed ARMS scheme can achieve confidentiality of documents, trapdoor unlinkability and collusion resistance. Extensive experiments show that the ARMS is more superior and efficient than existing approaches in terms of functionalities and computational overhead.**

***Index Terms*—Searchable Encryption, Multi-keyword Ranked Search, Search Authorization**

I. INTRODUCTION

Cloud computing [1] is a new pattern for data service, which can provide the storage and computing resources to the public over the Internet. In the cloud computing, data owners are moving their data to the cloud server. Despite the benefits of this kind of data outsourcing, such as low-cost and flexible data access, it can also cause some privacy problems since the outsourced data would contain some sensitive information. Thus, it is necessary to encrypt the sensitive data before outsourcing them to the cloud server. Moreover, data owners would like to allow many data users to search over the encrypted data while enforcing access control policies.

Searchable encryption [2], [3] has been recently developed as a fundamental approach to enable searching over encrypted cloud data. In the searchable encryption, the documents and the associated keywords are encrypted by data owners and outsourced to the cloud server. Search users would generate the encrypted trapdoor containing some keywords of interest to search over the dataset on the cloud server. Finally, the cloud server returns the matched results to the search user. To improve the search accuracy, the searchable encryption schemes should support multi-keyword search instead of single-keyword search [4]. And search users would like the cloud server to return results in a specific order, so that they can obtain the more relevant results quickly. Moreover, to make the searchable encryption schemes suitable for more practical scenarios, such as the scenario that the data is contributed from many data owners and can be searched by many search users, the schemes should support search authorization, which means the cloud server would only return the authorized results to the search users.

Recently, by adopting the secure k-nearest neighbors (kNN) technique [5], Cao et.al. [6] propose a symmetric searchable encryption scheme, which supports multi-keyword search and can return the results in a relevance-based order. However, the scheme does not consider search authorization problem. Sun et.al. [2] propose an attribute-based keyword search scheme to achieve owner-enforced search authorization, which only returns the authorized documents to the search user. However, the proposal cannot rank the search results. Moreover, their scheme requires high computational cost.

In this paper, on addressing the above issues, we propose an authorized and ranked multi-keyword search scheme (ARMS) over encrypted cloud data. Specifically, the contributions of this paper can be summarized as follows:

- By leveraging the secure kNN [6], [5] and ciphertext policy attribute-based encryption (CP-ABE) techniques [7], [8], we design a secure multi-keyword search scheme supporting result ranking and search authorization. The security analysis demonstrates that the ARMS can achieve confidentiality of documents, trapdoor unlinkability and collusion resistance.
- By conducting the real experiments, we show that the ARMS can achieve the above all functionalities and better efficiency in terms of computational overhead compared with the scheme in [2].

The remainder of this paper is organized as follows. In Section II, the system model and security requirements are formalized. In Section III, we propose the ARMS. The security analysis and performance evaluation of the ARMS are presented in Section IV and Section V, respectively. In Section VI, we present related works. Finally, we conclude this paper in Section VII.

978-1-4673-6432-4/15/$31.00 ©2015 IEEE

II. SYSTEM MODEL AND SECURITY REQUIREMENTS

A. System Model

As shown in Fig. 1, the system model in the ARMS consists of five entities: data owners, search users, cloud server, certificate authority (CA) and third-party auditor (TPA).

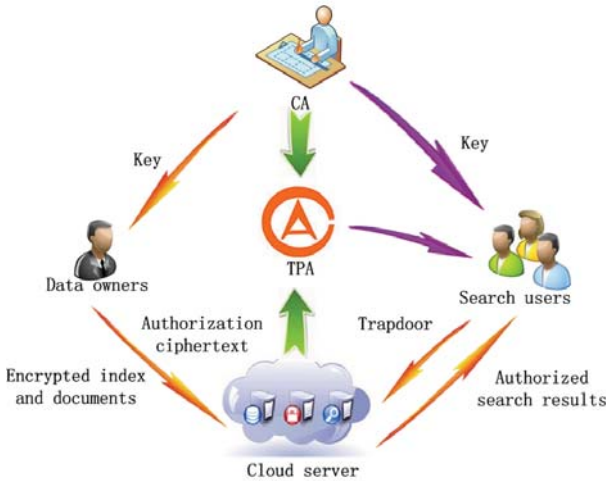


Fig. 1: System Model

CA is a global trusted authority and in charge of the system initialization, including search user authentication and key generation. In the system, data owners would set a keyword dictionary W, which contains d keywords. For each document, the data owner selects some relevant keywords to construct the search index. Both the documents and the index are encrypted before being outsourced to the cloud server. To enable search authorization over the dataset, data owners would define the access policy and compute the authorization ciphertexet for each document. Finally, the data owner sends the authorization ciphertext, the encrypted documents and index to the cloud server.

To search over the encrypted dataset, a search user would generate a trapdoor, including an encrypted query vector Q and a search token tk. Then, the search user sends the trapdoor to the cloud server to request the search results. Upon receiving the query vector Q, the cloud server uses it to search over the encrypted index. Then, the cloud server asks TPA to check the search token tk of the search user and only returns the authorized relevant documents to the search user. The search user could use her secret key to further decrypt the received documents.

B. Security Requirements

In the ARMS, we assume CA and TPA are global trusted. It is reasonable since they would be audited by the government office. Cloud server is honest but curious, which means it would execute the assigned task correctly but it is still curious about the encrypted dataset. Search users are curious about the encrypted documents and may collude with each other to get access to the unauthorized documents. Specifically, the ARMS aims to cover the following security requirements:

- *Confidentiality of Documents:* Since data owners would not like the unauthorized entities to obtain the content of the documents, the documents should be encrypted before being outsourced to the cloud server. And the decryption key should be well kept and distributed.
- *Trapdoor Unlinkability:* Trapdoor unlinkability means that the two trapdoors should be totally different even if they contain the same keywords. Specifically, the trapdoor generation function should be randomized rather than determined. And the cloud server cannot deduce any associations between the documents and the index.
- *Collusion Resistance:* The ARMS should be collusion resistant, which means any two search users cannot combine their search tokens to get access to the documents that they cannot access individually.

III. PROPOSED SCHEME

In this section, we propose the ARMS by modifying the kNN [5] and attribute-based keyword search [7], [8], [9] techniques. The ARMS consists of the following phases: System Setup, Key Generation, Encrypted Database Setup, Trapdoor Generation, Search and Auditing, Retrieve Results, and Attribute Revocation.

A. System Setup

CA takes into a security parameter l and outputs two cyclic groups G, G_T of a l-bit prime order p, a generator g of G, and a map $e : G \times G \rightarrow G_T$. Then, CA selects a hash function $H : \{0,1\}^* \rightarrow G$, which is modeled as random oracle. Finally, CA randomly chooses $\alpha, \beta, a, b, c \in Z_p$ as the secret key of the system and publishes the public parameters as

$$Pm = \{e, g, p, H, G, G_T, g^a, g^b, g^c, g^\beta, e(g,g)^\alpha\} \quad (1)$$

B. Key Generation

If a search user is legal in the system, CA would assign a set of attributes Ats to the search user according to her role. To generate the associated attribute keys for the search user, CA randomly chooses $r \in Z_p$ and computes $A = g^{(ac-r)/b}$, $B = g^{(\alpha+r)/\beta}$. For each attribute $a_j \in Ats$ of the search user, CA randomly selects $r_j \in Z_p$, a version number $v_j \in Z_p$ and computes $S_j = (g^r H(a_j)^{r_j})^{v_j}$, $K_j = (g^{r_j})^{v_j}$. Then, CA sends (j, v_j) to TPA through a secure channel, where j is the id of the search user.

The data owner takes a security parameter λ, and randomly selects two invertible matrixes $M_1, M_2 \in R^{(d+2)*(d+2)}$ and a (d+2)-dimension binary vector S as the secret index key, where d represents the size of the keyword dictionary.

Finally, the search user obtains the secret key including the attribute keys and the index key through a secure channel as

$$SK = \{Ats, A, B, \{(S_j, K_j) | a_j \in Ats\}, M_1, M_2, S\} \quad (2)$$

C. Encrypted Database Setup

The data owner builds the encrypted database as follows.

Step 1: For each document, the data owner computes the d-dimension relevance vector p using the $TF\text{-}IDF$ weighting technique [6], where the j-th item of p represents the relevance score of keyword ω_j in the document. Then, the data owner extends the p to a (d+2)-dimension vector $p^* = (p, \varepsilon, 1)$. We would let ε follow a normal distribution $N(\mu, \sigma^2)$ [6]. The data owner splits p^* into two (d+2)-dimension vectors $p^{'}$ and $p^{''}$ using the index key S as

$$\begin{aligned} &p_j^{'} = p_j^{''} = p_j^*, \ if \ S_j = 1 \\ &p_j^{'} = \frac{1}{2}p_j^* + r, \ p_j^{''} = \frac{1}{2}p_j^* - r, \ otherwise \end{aligned} \tag{3}$$

where r is a random number. Finally, the data owner computes $P = \{M_1^T \cdot p^{'}, M_2^T \cdot p^{''}\}$ as the encrypted relevance vector.

Step 2: The data owner chooses a symmetric cryptography $Enc()$, e.g., AES. For each document d_i, the data owner chooses a content key κ for the method $Enc()$ and encrypts the document as $Enc_\kappa(d_i)$. Then, the data owner selects two random numbers $r_1, r_2 \in Z_p$ to compute $W = g^{cr_1}$, $W_0 = g^{a(r_1+r_2)}g^{br_1}$, $W^{'} = g^{br_2}$, $C = \kappa e(g,g)^{\alpha r_2}$ and $C_1 = g^{\beta r_2}$. The data owner defines the access policy of this document as an access tree [8]. The access policy contains a set of attributes (Ts) and each leaf node of the access tree is associated with one of these attributes. And we let the $at(n)$ represent the associated attribute of the leaf node n. By using the access tree, the data owner computes the secret share of the r_2 as specified in [8], so that each leaf node n is associated with a secret share value $q_n(0)$. Then, for each leaf node n of the access tree, the data owner computes $W_n = g^{q_n(0)}$ and $D_n = H(at(n))^{q_n(0)}$. The authorization ciphertext Cph of the document is as

$$Cph = \{W, W_0, W^{'}, C, C_1, \{(W_n, D_n) | at(n) \in Ts\}\} \tag{4}$$

Finally, the data owner sends $Enc_\kappa(d_i)$, the encrypted relevance vector P and the Cph to the cloud server.

D. Trapdoor Generation

The search user generates the trapdoor including the encrypted query vector and the search token as follows.

Step 1: The search user takes a keyword conjunction $\varpi = (\omega_1, \omega_2, \cdots \omega_l)$ with l keywords of interest in W and generates a d-dimension binary query vector q, where the j-th item of q represents whether the keyword ω_j is in the conjunction or not. Then, the search user selects two random numbers r, t and extends the q to a (d+2)-dimension vector $q^* = (rq, r, t)$. The search user splits the vector q^* into two (d+2)-dimension vector $q^{'}$ and $q^{''}$ using the index key S as

$$\begin{aligned} &q_j^{'} = q_j^{''} = q_j^*, \ if \ S_j = 0 \\ &q_j^{'} = \frac{1}{2}q_j^* + r^{'}, \ q_j^{''} = \frac{1}{2}q_j^* - r^{'}, \ otherwise \end{aligned} \tag{5}$$

where $r^{'}$ is a random number. The search user computes the $Q = \{M_1^{-1} \cdot q^{'}, M_2^{-1} \cdot q^{''}\}$ as the encrypted query vector.

Step 2: The search user chooses $s \in Z_p$ to compute $tk_1 = (g^a g^b)^s$, $tk_2 = g^{cs}$ and $tk_3 = A^s$. Then, for each $a_j \in Ats$, the search user computes $S_j^{'} = S_j^s$ and $K_j^{'} = K_j^s$. The search token tk of the search user is as

$$tk = (Ats, tk_1, tk_2, tk_3, \{(S_j^{'}, K_j^{'}) | a_j \in Ats\}) \tag{6}$$

Finally, the search user sends the Q, tk, and an optional number k to the cloud server to ask the most k relevant results.

E. Search and Auditing

Upon receiving the encrypted query vector Q, the cloud server computes the relevance score with the encrypted relevance vector as follows.

$$\begin{aligned} Score &= P \cdot Q \\ &= \{M_1^T \cdot p^{'}, M_2^T \cdot p^{''}\} \cdot \{M_1^{-1} \cdot q^{'}, M_2^{-1} \cdot q^{''}\} \\ &= p^{'} \cdot q^{'} + p^{''} \cdot q^{''} \\ &= p^* \cdot q^* \\ &= (p, \epsilon, 1) \cdot (rq, r, t) \\ &= r(pq + \varepsilon) + t \end{aligned} \tag{7}$$

By sorting the relevance scores, the cloud server obtains the documents that are more relevant to the search request. Then, the cloud server sends the Cph of these documents and the search token tk to the TPA to ask the TPA to check whether the search user can access these documents or not. Upon receiving the Cph and tk, TPA checks if there is a subset of the search user's attributes that satisfies the access policy defined in the Cph of these retrieved documents. If there is such a subset S_a, for each leaf node n, where the associated attribute a_j belongs to S_a, TPA computes

$$E_n = e(S_j^{'\,1/v_j}, W_n) / e(K_j^{'\,1/v_j}, D_n) = e(g,g)^{rsq_n(0)} \tag{8}$$

Then, TPA combines these values (E_n) using the access tree to recover $E_r = e(g,g)^{rsr_2}$ as specified in [8]. TPA checks if

$$\begin{aligned} &e(W, tk_1) E_r e(tk_3, W^{'}) \\ &= e(g^{cr_1}, (g^a g^b)^s) e(g,g)^{rsr_2} e(g^{(acs-rs)/b}, g^{br_2}) \\ &= e(g,g)^{acs(r_1+r_2)+bcsr_1} \\ &= e(g^{a(r_1+r_2)} g^{br_1}, g^{cs}) \\ &= e(W_0, tk_2) \end{aligned} \tag{9}$$

If the above equation establishes, the search user can access the associated document. Finally, the cloud server sends the most relevant and authorized documents, the associated ciphertext C, C_1 and E_r of these documents to the search user.

F. Retrieve Results

Upon receiving the encrypted documents and the associated C, C_1, E_r, the search user only needs to compute

$$\begin{aligned} &\frac{C}{e(C_1, B)/(E_r)^{1/s}} \\ &= \frac{\kappa e(g,g)^{\alpha r_2}}{e(g^{\beta r_2}, g^{(\alpha+r)/\beta}) e(g,g)^{-rr_2}} \\ &= \kappa \end{aligned} \tag{10}$$

Then, the search user can use the content key κ to further decrypt the encrypted documents.

G. Attribute Revocation

When a search user's role has been changed, the set of her attributes may need to be updated. CA needs to regenerate a new version number $v_j^{'}$ for the search user and send the $(j, v_j^{'})$ to TPA. Then, CA assigns a new set of attributes for the search user and re-computes the attribute keys using the new version number $v_j^{'}$ as discussed in Section III-B. Thus, the search user's attributes have been changed and the previous attribute keys are invalid since they are generated by the old version number v_j. In the $Auditing$ phase, TPA would use the new version number $v_j^{'}$ to check the validity of the search user's attribute keys. Thus, the equation 9 would not establish if the search user uses the outdated attribute keys.

IV. SECURITY ANALYSIS

Under the assumption presented in Section II, we analyze the security properties of the ARMS. We give analysis of the ARMS on confidentiality of documents, trapdoor unlinkability, and collusion resistance.

A. Confidentiality of Documents

The documents are encrypted by traditional symmetric method using different keys before being outsourced to the cloud server and the content key κ is encrypted by the ABE technique. Only the search user with attributes that satisfy the access policy defined by data owners can decrypt the authorization ciphertext Cph to get the content key and further decrypt the encrypted document. Moreover, the search token contains a random number s which is kept secret by the search user. The cloud server and TPA cannot decrypt the Cph though they do most of the decryption work for the search user. Thus, the confidentiality of documents is well protected in the ARMS.

B. Trapdoor Unlinkability

Since the search user would like to conceal what she is truly searching for, the trapdoors should be encrypted before being sent to the cloud server. However, if the trapdoor generation function is determined, the trapdoors would be the same if they contain the same keywords, which reveals the search information to the cloud server. Under the $Knowing\ Backgroud$ model, the cloud server would know the statistic knowledge of the trapdoors and further determine the keywords contained in them. Thus, the trapdoor generation function should be randomized. In the ARMS, the trapdoor consists of the encrypted query vector and the search token. When generating the encrypted query vector, the search user would choose two random numbers and scale the query vector q to (rq, r, t). Then, the search user encrypts the extended query vector using the secret key (M_1, M_2, S). Thus, the encrypted query vectors would be totally different even if they contain the same keywords. And the search user would randomly choose a number s when computing the search token. That is, the cloud server cannot deduce any association between the trapdoors.

C. Collusion Resistance

In the key generation phase, for each search user, CA would choose a random number v, which is only known to CA and TPA, to compute the attribute keys. Thus, the attribute keys of different search users would be totally different even if two search users have the same attribute set. In the $Auditing$ phase, TPA would check the validity of the search token using the version number v. If two or more search users combine their search tokens to access the documents that they cannot access individually, the verification equation 9 would not establish since the search tokens of different search users contain different version numbers. Thus, the cloud server would not send the search results to the collusion search users.

TABLE I: Comparison of Security Level

	Sun's [2]	ARMS
Confidentiality of Documents	√	√
Trapdoor Unlinkability	√	√
Collusion Resistance	√	√

The comparison of security level between Sun's scheme [2] and ARMS is shown in TABLE I. As we can see, the ARMS can achieve the same security level as that of Sun's.

V. PERFORMANCE EVALUATION

In this section, we would compare the ARMS with Sun's scheme [2] in terms of functionalities and computational overhead.

A. Functionalities

To increase the search accuracy and enable the owner-enforced access policy, the searchable encryption schemes should support multi-keyword ranked search and search authorization. Since the role of the search user may dynamically change, the scheme should also support attribute revocation.

TABLE II: Comparison of Functionalities

	Sun's [2]	ARMS
Search Authorization	√	√
Attribute Revocation	√	√
Multi-keyword Ranked Search		√

As shown in TABLE II, Sun's scheme [2] can achieve search authorization and attribute revocation but cannot support multi-keyword ranked search, while the ARMS can achieve the above all functionalities.

B. Computational Overhead

In this subsection, we would evaluate the computational overhead of the ARMS in terms of index construction, trapdoor generation, search efficiency and attribute revocation. Let T_g be the time for an exponentiation operation in G, T_{gt} be the time for an exponentiation operation in G_T, T_h be the time that maps a string to an element of G and T_p be the time for a paring operation. We ignore multiplication and other hash operations [8]. Let n represent the number of attributes that are involved in the owner-defined access policy of the document and N be the number of attributes in the system. Let s be the number of attributes that a search user holds.

1) Index Construction: Index construction in the ARMS consists of the encrypted relevance vector and the associated ciphertext Cph. For each document, to compute the encrypted relevance vector, the data owner needs multiplication operations of a (d+2)-dimension vector and a (d+2)*(d+2) matrix with complexity $O(d^2)$, where d represents the size of the keyword dictionary. Compared with exponentiation operation in G, exponentiation operation in G_T and pairing operation, time cost for computing the encrypted relevance vector is negligible. As for computing the ciphertext Cph, the data owner needs $(2n+5)T_g + T_{gt} + nT_h$ time for one document in the ARMS. While in Sun's scheme, for each document, the data owner needs $(N+1)T_g + T_{gt}$ time to compute the subindex, which is linearly increasing with the number of the attributes in the whole system.

2) Trapdoor Generation: Trapdoor generation in the ARMS consists of the encrypted query vector and the search token. For the computing of the encrypted query vector, the search user needs two multiplications of a (d+2)*(d+2) matrix and a (d+2)-dimension vector, which is negligible compared with time-cost operations in group G and G_T. In Sun's scheme, time cost for a search user to compute the search token is $(2N+1)T_g$, which is lineally increasing with the number of the attributes in the whole system. While in the ARMS, the search user needs $(2s+4)T_g$ time to compute the search token, which is increasing with the number of the attributes that a search user holds.

3) Search Efficiency: In Sun's scheme [2], to search over one single document, the cloud server needs $(N+1)T_p + T_{gt}$ time. Thus, the time cost for search the whole dataset is linearly increasing with N and the number of the documents in the system. This can cost much time when there are a large number of documents in the dataset.

In the ARMS, upon receiving the encrypted query vector and the search token, the cloud server can first use the encrypted query vector to search over the index. For each document, the cloud server needs to compute the inner product of two (d+2)-dimension vectors. Thus, the complexity for searching over the dataset is $O(md)$, where m represents the number of documents and d is the size of the keyword dictionary. Then for the search results that are more relevant to the search request, TPA would check whether the search user can access these documents, requiring $(2n+3)T_p + nT_{gt} + 2nT_g$ time for the verification of each document. Compared with Sun's scheme [2], which asks the cloud server to perform the time-cost paring operations over the whole dataset, the ARMS can achieve better search efficiency.

4) Attribute Revocation: As discussed in Sun's scheme [2], we denote that $1 \leqslant \alpha, \beta \leqslant N$, which represent the number of attributes that a revoked search user holds. When one search user is revoked from the system, the cloud server needs to update all the associated subindexes that involve the attributes of the revoked search user and the number of these subindexes is denoted as N_c, resulting in $(\alpha N_c T_g)$ execution time. Then, the cloud server needs to re-compute the attribute keys for the non-revoked search users who hold the revoked attributes and the number of these search users is denoted as N_s, resulting in $\beta N_s T_g$ execution time. This leads to much unnecessary computational cost.

In the ARMS, when some attributes of one search user need to be changed, the ARMS only requires CA to re-compute the attribute keys for this search user, resulting in $(3s+2)T_g + sT_h$ execution time. Since the ARMS does not require the cloud server to update all the subindexes that contain the revoked attribute and re-compute the attribute keys for the non-revoked search users, it is much more efficient than that of Sun's scheme. The comparison of the computational overhead is shown in TABLE III.

TABLE III: Comparison of Computational Overhead

	Sun's [2]	ARMS
Index	$(N+1)T_g + T_{gt}$	$(2n+5)T_g + T_{gt} + nT_h$
Trapdoor	$(2N+1)T_g$	$(2s+4)T_g$
Search	$(N+1)T_p + T_{gt}$	$(2n+3)T_p + nT_{gt} + 2nT_g$
Revocation	$(\alpha N_c T_g)$+$\beta N_s T_g$	$(3s+2)T_g + sT_h$

5) Experiments Results: We conduct real experiments on a 2.13Hz-processor, 4GB memory computing machine with Java Pairing-Based Cryptography Library (JPBC) [10] to study the true execution time. We mainly focus on the execution time of the search and attribute revocation phase.

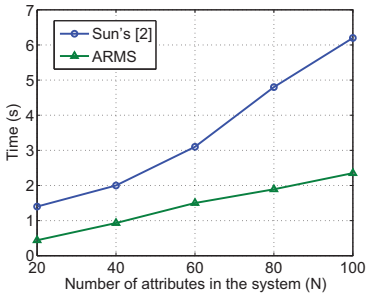
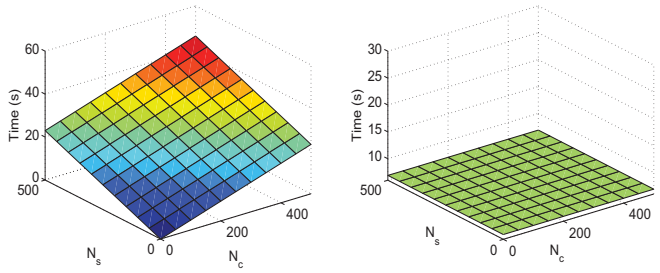


Fig. 2: Time for verifying one single document

As shown in Fig. 2, we compare the execution time of the search operation of one single subindex. The computational cost of search phase is mainly affected by the number of attributes that are involved in one document (n) in the ARMS and the number of documents of the system (N) in Sun's scheme. It is reasonable to assume that n is about one fifth of



(a) Sun's [2] (b) ARMS

Fig. 3: Comparison of computational overhead when revocation occurs

N. As we can see, although the time costs in both schemes are linearly increasing, the increase rate of the ARMS is less than that in Sun's scheme. As shown in Fig. 3, we compare the execution time of the revocation operations. In the real experiments, we let $\alpha = \beta = 1$, which means only one attribute of the search user is revoked. And we assume that a search user could hold at most 50 attributes, where $s = 50$. As we can see, while the execution time is linearly increasing with the number of search users (N_s) and subindexes (N_c) that involves the revoked attribute in Sun's scheme, time cost for attribute revocation is constant in the ARMS.

VI. RELATED WORKS

Searchable encryption [11] can be classified into two types: Searchable Symmetric Encryption (SSE) and Searchable Public-key Encryption (SPE).

Song et.al. [12] propose the first SSE scheme, which supports single-keyword search over the encrypted index. Later, Curtmola et al. [13] propose an efficient SSE scheme. Recently, Cao et.al. [6] propose a multi-keyword ranked search scheme, which can return results in a relevance-based order but cannot achieve search authorization. Li et.al. [14] propose a ranked search scheme over the encrypted cloud data through blind storage to conceal the access pattern of the search user from the cloud server. Yang et.al. [3] propose a secure dynamic searchable encryption scheme with constant document update cost.

The first SPE scheme is introduced by Boneh et al. [15], which can support more flexible search requests. Recently, Sun et.al. [2] propose an attribute-based keyword search scheme, which supports owner-enforced search authorization. However, their scheme cannot return the more relevant results. Moreover, their scheme is lack of efficiency. Zheng et.al. [8] propose a verifiable attribute-based keyword search scheme by leveraging the attribute-based encryption [7]. However, their scheme does not consider the attribute revocation problem.

VII. CONCLUSION

In this paper, we propose an authorized and ranked multi-keyword search scheme over encrypted cloud data. Security analysis demonstrates that the proposal can achieve confidentiality of documents, trapdoor unlinkability and collusion resistance. Extensive experiments show that the proposal can achieve better efficiency in terms of functionalities and computational overhead compared with the existing ones. In our future work, we would explore the dynamic searchable encryption in cloud computing.

VIII. ACKNOWLEDGEMENT

This work is supported by the Natural Science Foundation of China under Grants 61472065, 61350110238, 61103207, U1233108, U1333127, and 61272525, the International Science and Technology Cooperation and Exchange Program of Sichuan Province, China under Grant 2014HH0029, China Postdoctoral Science Foundation funded project under Grant 2014M552336, and State Key Laboratory of Information Security Open Foundation under Grant 2015-MS-02.

REFERENCES

[1] H. Liang, L. X. Cai, D. Huang, X. Shen, and D. Peng, "An smdp-based service model for interdomain resource allocation in mobile cloud networks," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 5, pp. 2222–2232, 2012.

[2] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proceedings of INFOCOM*. IEEE, 2014.

[3] Y. Yang, H. Li, W. Liu, H. Yang, and M. Wen, "Secure dynamic searchable symmetric encryption with constant document update cost," in *Proceedings of GLOBECOM, USA, 2014, to appear*.

[4] Y. Yang, H. Li, M. Wen, H. Luo, and R. Lu, "Achieving ranked range query in smart grid auction market," in *Proceedings of ICC*, 2014, pp. 951–956.

[5] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 139–152.

[6] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

[7] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy*, 2007, pp. 321–334.

[8] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proceedings of INFOCOM*. IEEE, 2014.

[9] D. Liu, H. Li, Y. Yang, and H. Yang, "Achieving multi-authority access control with efficient attribute revocation in smart grid," in *Proceedings of ICC*, 2014, pp. 634–639.

[10] "Java pairing-based cryptography library (jpbc). http://gas.dia.unisa.it/projects/jpbc."

[11] H. Li, Y. Yang, M. Wen, H. Luo, and R. Lu, "Emrq: An efficient multi-keyword range query scheme in smart grid auction market." *KSII Transactions on Internet and Information Systems*, vol. 8, no. 11, pp. 3937–3954, 2014.

[12] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE Symposium on Security and Privacy*, 2000, pp. 44–55.

[13] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *the 13th ACM conference on Computer and communications security*, 2006, pp. 79–88.

[14] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted cloud data through blind storage," *IEEE Transactions on Emerging Topics in Computing*, 2014, DOI: 10.1109/TETC.2014.2371239.

[15] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proceedings of EUROCRYPT*. Springer, 2004, pp. 506–522.