Adaptive Multi-Keyword Ranked Search over Encrypted Cloud Data

Daudi Mashauri[1], Ruixuan Li[1], Hongmu Han[1], Xiwu Gu[1], Zhiyong Xu[2], Cheng-zhong Xu[3,4]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China
[2]Department of Mathematics and Computer Science, Suffolk University, Boston, MA 02114, USA
[3]Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202, USA
[4]Shenzhen Institute of Advanced Technology, Chinese Academy of Science, Shenzhen, Guangdong 518055, China
daudimasha@live.com, {rxli, hanhongmu, guxiwu}@hust.edu.cn, zxu@mcs.suffolk.edu, czxu@wayne.edu

Abstract. To preserve data privacy and integrity, sensitive data has to be encrypted before outsourcing to the cloud server. However, this makes keyword search based on plaintext queries obsolete. Therefore, supporting efficient keyword based ranked searches over encrypted data became an open challenge. In recent years, several multi-keyword ranked search schemes have been proposed in trying to solve the posed challenge. However, most recently proposed schemes don't address the issues regarding dynamics in the keyword dictionary. In this paper, we propose a novel scheme called A-MRSE that addresses and solves these issues. We introduce new algorithms to be used by data owners each time they make modifications that affects the size of the keyword dictionary. We conduct multiple experiments to demonstrate the effectiveness of our newly proposed scheme, and the results illustrates that the performance of A-MRSE scheme is much better that previously proposed schemes.

Keywords: Cloud computing, searchable encryption, multi-keyword query, ranked search, encrypted data.

1 Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources, such as networks, servers, storage, applications and services, which can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. Cloud computing provides affordable and convenient ways to store and manage huge amounts of data generated by data owners. However, even with all of its advantages, cloud computing still faces great challenges following a serious

threat posed to data owners about security and privacy, especially when it comes to sensitive data. Encryption of data before shipping to the cloud server offers a viable solution to data owners regarding data integrity and confidentiality. However, keyword searches based on plaintext queries became obsolete on encrypted data, and data owners have to download the entire database they own and start decrypting files one by one, looking for those of interest. It goes without saying that this is too much to bear, especially in today's pay-as-you use fashion.

Searchable encryption is the earliest scheme that utilizes keyword search over encrypted data [2] [3]. Solutions proposed by these schemes resolved the issues concerning security and privacy of data. However, they also introduced more obstacles as they encountered huge computation and communication overheads. These overheads resulted from the fact that the proposed solutions do not offer any ranking mechanism after performing a keyword search as they based on disjunctive searches. To resolve this issue, another era of ranked keyword search schemes over encrypted came to the rescue. Single keyword ranked search [4] was among the first published works providing a practical implementation. It fulfilled its designed goals as far as ranked searches are concerned. However, supporting only a single keyword search from thousands of encrypted files was not an efficient solution that data owners anticipated for a while.

Recently, a number of research works, such as [5] [6], have been done in order to facilitate multi-keyword queries over encrypted cloud data and they also support result ranking. MRSE [1] is one of the earlier works crowned in supporting multi-keyword ranked queries over encrypted data. It also provides a viable solution that works under practical implementation. Most of the recent works on multi-keyword ranked searches don’t address the issues regarding any future modifications that will affect the size and content of the keyword dictionary. There is a large computation and communication overhead posed to data owners each time they modify their keyword dictionaries. As a privacy requirement, two query vectors resulting from similar set of keywords cannot be the same. Hence, the cloud server won't be able to determine if they come from the same set of keywords. However, the cloud server can still determine due to the fact that finally they will result into similar sets of ranked files, although they look different upon submission.

In this paper, we propose a new scheme called A-MRSE in order to resolve the resulting issues of modifications on keyword dictionaries. In our newly proposed scheme, we consider both real life scenarios where the data owners can either insert or remove certain keywords from the dictionary. We propose new algorithms that can be used each time the data owner makes these changes. They present minimum communication and computation overhead.

The contributions of this paper can be summarized as follows:

- We design a novel scheme that is adaptive and supports any modification made on the keyword dictionary, either inserting or removing keywords with minimum overhead.
- We improve security of the ranked results by sealing the cloud server from any form of statistical attacks.

The rest of this paper is organized as follows. Section 2 introduces our A-MRSE scheme, which is followed by results and a discussion in Section 3. Section 4 describes related works, and we conclude with future works in Section 5.

2 Adaptive Multi-Keyword Ranked Search over Encrypted Cloud Data

In this section, we will present our adaptive multi-keyword ranked search over encrypted cloud data (shorted as A-MRSE) scheme. In order to quantitatively evaluate the coordinate matching like in MRSE, we adopt "inner product similarity" in our work as well. Also like in MRSE [1], we define an index vector for each file based on the keywords it contains from the dictionary; two invertible matrices and a bit vector are also used for index vector encryption and trapdoor generation. However, our work solves the issue with MRSE in a sense that it allows more keywords to be added in the dictionary as well as some of them to be removed from it. The detailed design of A-MRSE scheme includes the following six aspects.

(1) *InitialSetup*: The data owner selects a set of $\boldsymbol{n}$ keywords from the sensitive plaintext dataset F, and $\boldsymbol{u}$ dummy keywords to be inserted in the indexing vector in order to strengthen security and maintain privacy. The index vector mandates any future modifications that can be made on the keyword dictionary. In MRSE, the index vector has three parts which are first $\boldsymbol{n}$ locations used to indicate presence or absence of real keywords, followed by $\boldsymbol{u}$ locations for dummy keywords, and terminated by the constant 1 at the last position as shown in Fig. 1.

MRSE vector structure makes any modification on the keyword dictionary unworkable since positions of the keywords are fixed. However, in A-MRSE, we mirror the existing structure and derive a new vector structure having security locations (last constant dimension and dummy keywords locations) at the beginning, followed by $\boldsymbol{n}$ locations of real keywords as shown in Fig. 2.

With this vector structure, A-MRSE supports any future modifications of the keyword dictionary size. For any file in the dataset, if it has real keyword W_j, then in the corresponding index vector $p[1+u+j] = 1$, otherwise 0.

(2) *KeyReduce*: since A-MRSE supports keyword dynamics in the dictionary as compared to MRSE, the data owner calls this algorithm with number of keywords to be reduced as an input parameter to generate a new secret key **SK** $\boldsymbol{k_2}$ from the previously generated **SK** $\boldsymbol{k_1}$**.** Previously generated matrices M_1 and M_2 will then be resized into new matrices M'_1 and M'_2, each having $\boldsymbol{(d-r)\times(d-r)}$ dimension. That is accomplished by removing the last $\boldsymbol{r}$-rows and $\boldsymbol{r}$-columns, which finally yields a $\boldsymbol{d-r}$ square matrix.

Modification of the splitting vector S demands special attention due the role played of each bit position in it. Basically, the dictionary size will change from $\boldsymbol{n}$ to $\boldsymbol{(n-r)}$ after removing $\boldsymbol{r}$ keywords. This algorithm inspects the new dictionary. If a keyword

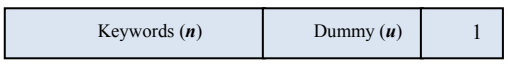

Fig. 1. MRSE index vector structure.

Fig. 2. A-MRSE vector structure.

still exists in both dictionaries (the old and new sized), then that particular bit is copied into a new vector S' and omitted otherwise. The process continues for all $\boldsymbol{n}$ locations and finally gives a $\boldsymbol{(d-r)}$ S' bit vector. Algorithm 1 shows how KeyReduce works.

Algorithm 1. KeyReduce(k_1, r)

Input: number of reduced keywords and original secret key.
Output: new secret key **SK $\boldsymbol{k_2}$.**
Method: the key reduce algorithm works as follows.

1: Receive the integer input parameter r;
2: Retrieve the original secret key SK k_1 components;
3: Resize matrices M_1, M_2 to M'_1, M'_2 by applying dimension reduction;
4: Read the old dictionary as file f_1 and new dictionary as file f_2;
5: **for** each line in file f_1 and file f_2
6: **if** (keyword in f_1 exists in f_2)
7: copy the value of bit position for this keyword from S into S';
8: **else**
9: skip bit position;
10: **end** if
11: **end** for
12: Give SK k_2 with 3-tuple as $\{S', M'_1, M'_2\}$;

---

(3) *KeyExtend*: after adding more files on the cloud server, the data owner will definitely need to include new keywords in the dictionary. In this case, MSRE cannot work any longer as it suffers huge computation overhead as well as bandwidth inefficiency. This is where A-MRSE comes into account as it allows easy expansion of the secret keys relative to the increase of keywords in the dictionary.

If $\boldsymbol{z}$ keywords are added, this algorithm generates two new $\boldsymbol{z \times z}$ invertible matrices, M_{z1}, M_{z2}, and a new $\boldsymbol{z}$ bit vector S_z. These newly created matrices will be added to original matrices M_1 and M_2 and finally gives two modified matrices M'_1 and M'_2 having $\boldsymbol{(d+z)\times(d+z)}$ dimension according to block diagonal matrix theorem [7].

On the other hand, splitting vectors S_z and S will be joined and make a new vector S' by copying all elements of vector S into S' then followed by appending elements of S_z. Algorithm 2 shows how KeyExtend works.

Algorithm 2. KeyExtend(k_1, z)

Input: original secret key and number of newly added keywords.
Output: new secret key **SK $\boldsymbol{k_3}$.**
Method: the key extend algorithm works as follows.

1: Receive the integer input z;
2: Retrieve original key SK k_1 components;

3: Generates two invertible matrices Mz_1, Mz_2 and a bit vector S_z;
4: Add Mz_1 to M_1, Mz_2 to M_2 by using diagonal block matrix operation and produces M'_1 and M'_2 having $(d+z)\times(d+z)$ dimensions;
5: **for** each bit position in vector S
6: copy it into a new vector S';
7: **end** for
8: **for** each bit position in vector S_z
9: copy and append it into new vector S';
10: **end for**
11: Give SK k_3 with 3-tuple as $\{S', M'_1, M'_2\}$;

---

(4) *BuildIndex*: this algorithm builds an encrypted searchable index for plaintext files in the original set F. Initially, the data owner applies similar procedures as in MRSE [1] before addition or reduction of keywords from the dictionary.

For each file, a bit vector pi is set. Then starting with security positions, $p[1]$ is set to 1, and values in dummy keyword positions between $p[1]$ and $p[2+\boldsymbol{u}]$ are set to a random number ε. The remaining positions will be filled, indicating whether the file contains keywords from the dictionary. Therefore, $p[2+\boldsymbol{u}]$ to $p[1+\boldsymbol{u}+\boldsymbol{n}]$ will be set to 1 if the file contains a dictionary keyword and 0 otherwise. After setting all bit positions in vector pi, splitting procedures will then follow as in secure kNN computation [8] except that the index structure is reversed. This implies in A-MRSE, we start with security locations then followed by real keyword locations. The BuildIndex algorithm is shown in Algorithm 3.

Algorithm 3. BuildIndex(F, SK)

Input: the secret key SK, and the file set F.
Output: the encrypted searchable index.
Method: the build index algorithm works as follows.

1: Receive the file set F;
2: **for** each $F_i \in F$
3: Generate a bit vector p_i;
4. Set p[1] = 1, and $p[2] - p[1+u] = \varepsilon_i$;
5. **for** $j = (u + 2)$ to $(1 + u + n)$
6. **if** $F_{idj} \in W$
7. Set $p[j] = 1$;
8. **else** set $p[j] = 0$;
9. **end** if
10. **for** $j = 1$ to $(1 + u + n)$
11. **if** $S[j] = 1$
12. $p_1[j] + p_2[j] := p[j]$;
13: **else** $p_1[j] := p_2[j] := p[j]$;
14: Run $\tilde{p_1} = M_1^T p_1$, $\tilde{p_2} = M_2^T p_2$, and set $I_i = \{\tilde{p_1}, \tilde{p_2}\}$;
15: Upload encrypted files $\{F_i\} \in C$ and $I = \{I_i\}$ to the cloud server;

---

(5) *TrapdoorGen*: Having a set of interested keywords, an authorized data consumer calls this algorithm to generate a secure trapdoor in order to search and retrieve a number of encrypted files from the cloud server. For a multi-keyword query q, a query vector is generated using the same strategy as in MRSE with $\boldsymbol{v}$ number of dummy locations set to 1 and all remaining locations set to 0.

A score is used to determine the location of the file in the matching result set. In MRSE, this score was calculated by using Equation 1 for index file p_i.

$$p_i \cdot q = r\left(x_i + \sum \varepsilon^{(v)}\right) + t_i \tag{1}$$

Xu et al [5] discovered the impact of values of the dummy keywords inserted in the final score, which causes "out-of-order" problem. This happens when a file with popular keywords obtaining lower score and finally will not be included in the returned list to the data consumer.

To ameliorate the in-order ranking result while maintaining privacy-preserving property, all locations containing real keywords are multiplied by random number $\boldsymbol{r}$, and all locations containing dummy keywords are multiplied by another random number $\boldsymbol{r_2}$ which is obtained by using Equation 2. Finally, the score is calculated by using Equation 3.

$$r2 = \frac{Random(0,r)}{\left(v * MAX\left((u-c),(u+c)\right)\right)} \tag{2}$$

$$p_i \cdot q = I_i \cdot T = r \times x_i + r2 \times \sum \varepsilon^{(v)} + t_i \tag{3}$$

Finally, the trapdoor T will be generated as $\{M_1^{-1}q', M_2^{-1}q''\}$.

(6) *Query*: After receiving T from the data consumer, the cloud server calls this algorithm to calculate the score for each file in the encrypted index I. The data consumer also includes parameter K so that the cloud server will return a list of only top-K files after a ranked search over the encrypted index. The trapdoor generated from a similar set of keywords will be different each time. This prevents the cloud server from performing statistical attacks; however the cloud server can still determine the trapdoors came from the same keyword set since finally they all retrieve identical top-K files though with different scores.

To resolve this issue, in A-MRSE we designed a new way to obfuscate the cloud server from performing statistical attacks. We modify the total number of retrieved files by adding K' files randomly such that $K' < K$. The value of K' is obtained by using Equation 4.

$$K' = \varepsilon \times K \tag{4}$$

The value of ε is used as a security parameter, and it grows from 0% depending on the number of files to be retrieved from the cloud server. If it set to 0%, the implication is that the data owner prefers efficiency over security. For instance, when K is less than 10, the value of ε can be set to 25%, when K lies between 10-20 it can be set to 20%, and when K reaches 50% it can be 15%. By doing so, the cloud server cannot determine whether two queries originated from the same keyword set.

3 Performance Evaluation

In this section we present the results obtained after performing multiple experiments with different settings. We selected a real life dataset, Enron Email Dataset [9], various numbers of emails were randomly selected from the dataset for each test. The workbench was a Dell Latitude E-5520 machine with an Intel CoreTM i7 CPU @ 2.20GHz × 8 with 8GB of RAM. The operating system is Linux Mint 15 (Olivia x64), simulation codes were implemented by using Java programming language and elements in invertible matrices were double, generated randomly by using a Jama-1.0.3.jar package [10]. For each test taken, we observed results for both A-MRSE and MRSE scheme and then we compared their performances.

(1) *Key Generation and Editing*: the total time in key generation includes the time to generate bit vector S, as well as the two invertible matrices which then followed by adding the time to transport and compute the inverse of these matrices. Fig. 3 shows how A-MRSE outperforms MRSE during key generation starting from 1000 keywords dictionary size and keeps growing up to 8000. In this phase, the previously generated 1000 key size was used to create new expanding keys. For instance, for a 5000 key, A-MRSE uses 32.94% of total time, and for a 6000 A-MRSE uses 37.85% of total time as compared with MRSE.

A remarkable gain is observed when the dictionary size is reduced. It took much less time to edit the key with A-MRSE than to regenerate with MRSE. As shown in Fig. 4, A-MRSE remains almost flat during key regeneration while MRSE raises as the keywords grow in the dictionary. For example, it took 0.33% of the total time for A-MRSE to generate a 5000 key from an existing 6000 in case 1000 keywords are dropped from the dictionary where by MRSE took 298 times more than A-MRSE.

(2) *Index Building*: the time taken to build a searchable index *I* for all documents is the sum of the individual time taken to build indexes *I* for each document. This includes mapping of keywords extracted from file *F*i to a data vector pi, followed by encrypting all the data vectors and finally builds a searchable index that will be uploaded to the cloud server. The cost of mapping or encrypting primarily depends on the dimension of the data vector which is tied up to the dictionary size. Also, the cost of building the whole searchable index *I* depends on the number of sub indexes which implies the total number of documents in the dataset.

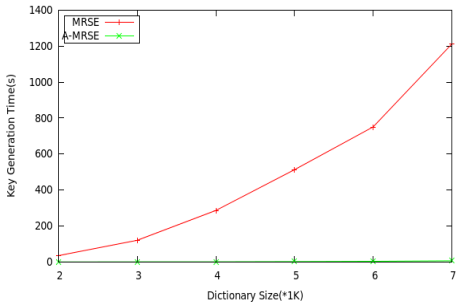

Fig. 3. Key reduction and generation.

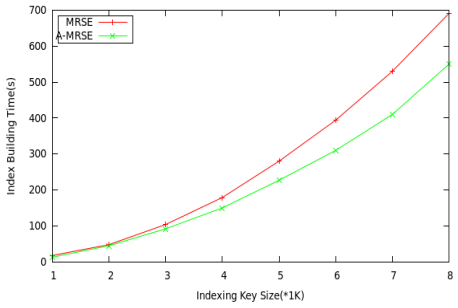

Fig. 4. Index building.

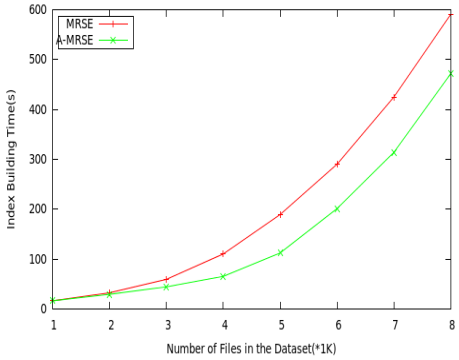

Fig. 5. Index building.

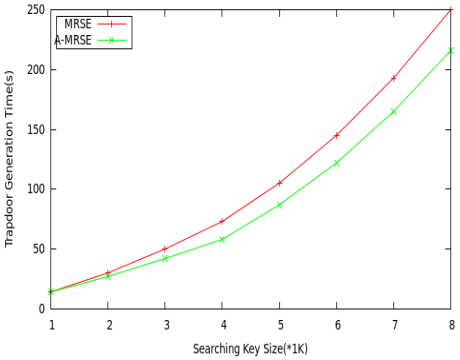

Fig. 6. Trapdoor generation.

Fig. 5 shows a comparison of the time taken to build the searchable index for both MRSE and A-MRSE with different numbers of keywords in the dictionary where the documents in the dataset were fixed to 3000. A-MRSE outperforms MRSE, for instance building index with a 6000 key size, it took 44% of the total time as compared with MRSE. Fig. 6 shows comparison of the total time taken to build the searchable index between A-MRSE and MRSE for different numbers of files in the dataset ranging from 1K to 8K inclusive with the key size fixed to 2K. Again, A-MRS saw off MRSE in terms of efficiency, for example it took 37.14% of total time to build index for a 4000-files dataset.

As we can see, in all settings whether the key is fixed with increasing documents in the dataset or the other way round, A-MRSE still outperforms MRSE. This is because the keys in A-MRSE contain many zero-valued elements as a result of key expansion. This makes multiplications during index building to be much faster in A-MRSE than in MRSE.

(3) *Trapdoor Generation*: trapdoor generation involves two multiplications of a matrix and a splitting query vector. Fig. 8 shows the trapdoor generation cost for both A-MRSE and MRSE. Again, as seen from the graph, A-MRSE saw off MRSE as it takes less time to complete the whole process. For example, with 5000 keywords dictionary size, A-MRSE took 45.31% of the total time as compared with MRSE scheme. As more keywords are added, more performance gain can be achieved with our A-MRSE. We are certain that this gain is highly contributed with many zero-values elements in A-MRSE matrices compared with the ones in MRSE.

(4) *Query:* the cloud server is responsible for performing query execution by using querying algorithm with which it computes and ranks the similarity scores for all documents in the dataset. We conducted several experiments for both A-MRSE and MRSE with fixed number of documents in the dataset while varying the dictionary size. The performances were almost comparable between A-MRSE and MRSE. The key point of our scheme is that, it enables data owners to make use of existing keys and generate new searching and indexing keys without the need of running key generation again from scratch. This is for both scenarios of increasing and decreasing number of keywords in the dictionary.

4 Related Works

Cloud computing is delivered as a product from both software and hardware evolutions as well as the Internet. It offers actual realization of the long waited utility computing service. However, with all its benefits, it still comes with a number of security challenges posed to both individual data owners as well as organizations which use cloud technology [11]. Enabling keyword search over encrypted data in cloud computing environment became an open question.

Song et al [2] was among the earliest researchers to present a practical solution of keyword search over encrypted data. They offered a solution in which each word in the plaintext is encrypted with a two-layer encryption scheme that uses stream ciphers. Boneh et al [12] introduced a public keyword encryption scheme with keyword search. Similar works were also presented in [13] and [14] that put forward searchable encryption schemes. However, solutions based on public key encryption are usually very computation expensive. Furthermore, keyword privacy is not protected in public key setting since the server could encrypt any keyword with the known public key and then use the received trapdoor to evaluate its ciphertext.

Wang et al [4] presented the earlier work that explores user's capability of ranked search over encrypted cloud data. Ranked search improves systems efficiency and usability by returning all matching files in a ranked order depending on predefined user rules, and for this case is its file length and document frequency. Other solutions are presented in [3] and [15], but all these works presented solutions based on single keyword search only. Singe keyword ranked search is computational inefficiency when the number of documents is quite large as the final result can include almost all documents in the set C as long as they contain a single searched keyword. Multi-keyword ranked searches came to puzzle out this issue, as initially presented by Cao et al [1]. The scheme is semantically secure. However, it lacks actual implementation.

None of the above schemes addressed the challenges of varying keyword dictionary size following addition or reduction of files on cloud server. A-MRSE presents a novel scheme with new algorithms that address this issue and can support any dynamic in keyword dictionary size with minimum communication and computation overhead. A-MRSE achieves its desired goals and leaving no drawback as in [1].

5 Conclusion and Future Works

In this paper, we present a novel scheme that is adaptive and it supports multi-keyword ranked search over encrypted cloud data. We present a novel scheme, called A-MRSE, which uses new algorithms to solve the existing challenges on multi-keyword keyword searches over encrypted data. We also strengthen security and privacy by preventing the cloud server from performing statistical attacks based on the results to be returned to data consumers after performing a ranked search. A-MRSE can be easily deployed in many cloud scenarios, such as UniDrive, a synergizing multiple consumer cloud storage service [16]. We conducted multiple experiments under different settings and the results illustrates that our new A-MRSE scheme is much better than MRSE.

In the future, we are looking forward to building schemes that can work under stronger security threats especially when the cloud server is capable of colluding. We are working on building adaptive schemes that can work for fuzzy keyword searches.

Acknowledgments. This work is supported by National Natural Science Foundation of China under grants 61173170, 61300222, 61433006 and U1401258, Innovation Fund of Huazhong University of Science and Technology under grants 2015TS069 and 2015TS071, and Science and Technology Support Program of Hubei Province under grant 2014BCH270.

References

1. Cao N., Wang C., Li M., K. Ren, and Lou W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of INFOCOM, IEEE, pp. 829--837 (2011)
2. Song D. X., Wanger D., and Perrig A.: Practical Techniques for Searches on Encrypted Data. In: 2000 IEEE Symposium on Security and Privacy, IEEE, pp. 44-55 (2000)
3. Michel A., Mihir B., Catalano D., Kiltz E., Kohno T., Lange T., Malone-Lee J., Neven G., Paillier P., and Shi H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Advances in Cryptology – CRYPTO, pp. 205--222, Springer Berlin / Heidelberg (2005)
4. Wang C., Cao N., Li J., Ren K., and Lou W.: Secure Ranked Keyword Search over Encrypted Cloud Data. In: Proceedings of IEEE 30th International Conference in Distributed Computing Systems (ICDCS), pp. 253--262, IEEE (2010)
5. Xu Z., Kang W., Li R., Yow K., and Xu C.: Efficient Multi-keyword Ranked Query on Encrypted Data in the Cloud. In: The 18th Intentional Conference on Parallel and Distributed Systems (ICPADS), pp. 244--251, IEEE (2012)
6. Jiad Y., Lu P., Zhu Y., Xue G., and Li M.: Towards Secure Multi-keyword Top-k Retrieval over Encrypted Data. IEEE Transactions on Dependable and Secured Computing, 10(4), 239--250 (2013)
7. Feingold D. G., and Richard S. V.: Block Diagonally Dominant Matrices and Generalizations of the Gerchgorin Circle Theorem. Pacific J. Math, 12(4), 1241--1250 (1962)
8. Wong W. K., Cheung D. W., Kao B., and Mamoulis N.: Secure kNN Computation on Encrypted Databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 139--152, ACM (2009)
9. Cohen W. W.: Enron Email Dataset. http://www.cs.cmu.edu/~enron
10. Hicklin J., Moler C., Webb P., Boisvert R. F., Miller B., Pozo R., and Remington K.: JAMA: Java Matrix Package. http://www.math.nist.gov/javanumerics/jama/
11. Mather T., Kumaraswamy S., and Latif S.: Cloud Security and Privacy: an Enterprise Perspective on Risks and Compliance, O'Reilly Media Inc. (2009)
12. Boneh D., Di Crescenzo G., and Persiano G.: Public Key Encryption with Keyword Search. In: Advances in Cryptlogy – EUROCRYPT 2004, Springer Berlin / Heidelberg, pp. 506--522 (2004)
13. Yan-Cheng C., and Mitzenmacher M.: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: Applied Cryptography and Network Security, Springer Berlin / Heidelberg, pp. 442-455 (2005)
14. Mihir B., Boldyvera A., and O'Neal A.: Deterministic and Efficiently Searchable Encryption, In: Advances in Cryptology – CRYPTO, Springer Berlin / Heidelberg, pp. 535--552 (2007)
15. Li J., Wang Q., Wang C, Kao N., Ren K., and Lou W.: Fuzzy Keyword Search over Encrypted Data in Cloud Computing. In: Proceedings of INFOCOM, IEEE, pp. 1--5 (2010)
16. Tang H., Liu F.M., Shen G., Jin Y., Guo C.: UniDrive: Synergize Multiple Consumer Cloud Storage Services. In: ACM/USENIX/IFIP Middleware, Vancouver, Canada (2015)