Journal of Electronics and Information Science (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/jeis.2023.080101
ISSN 2371-9524 Vol. 8 Num. 1

A Public Key Searchable Encryption Method Based on Multiple Keywords

Wenrui Ji[1], Yan Wang[1], Xin Luo[1], Li Li[2,*], Guangwei Xu[1], Wei Li[1]

[1]School of Computer Science and Technology, Donghua University, Shanghai, 201620, China
[2]School of Architecture and Urban Planning, Tongji University, Shanghai, 200082, China
**Corresponding author*

Keywords: Searchable encryption, multi-keyword, random element padding

Abstract: In recent years, the secure search of encrypted cloud data has become a hot research topic and a challenging task. Several secure search schemes have been proposed to address this challenge. However, existing public-key searchable encryption schemes still face many problems. Among these schemes, most of them are based on single-key searchable encryption schemes, and although some schemes are designed for multi-key search, they still disclose the secret information of the encrypted index. Based on this, this paper proposes a multi-key public-key searchable encryption scheme without a secure channel, which uses a random element padding method to construct an encrypted index to ensure the security of the index information, and then improves the efficiency of the search by aggregating the query keyword information to generate query trapdoors. The simulation results of the algorithm show that the algorithm improves the query efficiency and query accuracy under the condition that the index and trapdoor are secure.

1. Introduction

With the development of cloud computing technology0 , more and more individuals and businesses are choosing to store their data on cloud servers for management purposes. As cloud servers are not fully trusted and users' data stored in the cloud may be attacked or at risk of privacy breaches, users encrypt their data before uploading it. While this method protects users' data, when users want to find it, they need to download the encrypted data locally from the cloud, decrypt it and then search for it, which consumes a lot of network bandwidth and is inefficient.

In order to improve the efficiency of ciphertext retrieval, many scholars have conducted extensive research on searchable encryption technique0000. Song et al.0 proposed the first SE scheme, which requires only a small amount of communication, but the computational overhead is linear in the size of the search query. To solve this problem, Boneh et al.00 proposed PEKS (Public key Encryption with Keyword Search), where the data owner encrypts the data with the public key and uploads it to the cloud server, and the data user searches the data with his private key and the query keyword to generate the corresponding trapdoor and then uploads it to the cloud server for keyword search, and the cloud server By matching the cipher text with the trapdoor, the cloud server returns the data needed by the user. However, Boneh et al.00 's public key searchable encryption scheme has the disadvantage that a secure channel needs to be established between the user and the cloud server to

transmit the query trapdoor, and the overhead of establishing a secure channel is often expensive. To address this drawback, Beak et al.0 proposed a public key searchable encryption scheme without a secure channel. In 2007, Gu et al.0 proposed a more efficient public-key searchable encryption scheme based on bilinear pairs. In their scheme, the encryption process has no bilinear pair computation operation, making the efficiency of the scheme improved compared to other schemes. In Beak et al.0 the model of the scheme suffers from the drawback that the attacker does not have access to the test queries, so Rhee et al.00 enhanced the Beak et al.0 The security model in the proposed scheme enables an attacker to obtain a relationship between a ciphertext and a trapdoor beyond the challenge ciphertext. A public key searchable encryption scheme without a secure channel under the enhanced model is also proposed. However Beak et al.0 proposed a scheme that uses a random prediction machine in the security proof process, such that the scheme implementation may lead to insecurity under the standard model, based on this Fang et al.0 proposed a public-key searchable encryption scheme that is resistant to keyword guessing attacks. Although the appealing scheme improves the security of public-key searchable encryption schemes under unsecured channels, the schemes are designed to perform only single-keyword searches, and data users who want to query encrypted data corresponding to multiple keywords need to generate many query traps, and the results obtained are likely to be imprecise due to the inclusion of only a single keyword.

To solve this problem, this paper proposes a multi-keyword-based public key searchable encryption method, with specific contributions summarized below.

(1) The method uses a random element padding method to ensure that the ciphertext size of all encrypted indexes is the same, thus protecting the keyword ciphertext information while preventing the leakage of the number of keywords corresponding to the encrypted data.

(2) The method designs an aggregated keyword information trapdoor generation method to prevent the leakage of query keyword information in the query trapdoor, while only one query trapdoor needs to be generated regardless of how many keywords the data user wants to query each time, reducing the computational overhead of user-generated query trapdoors and improving the efficiency of query matching on the cloud server.

(3) The security of the encrypted index and query trapdoor in the method is demonstrated by security analysis.

2. System model and programme brief

2.1 System model

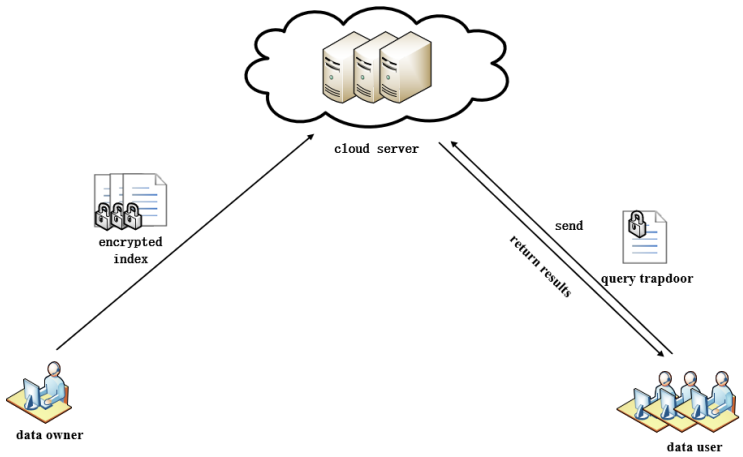

Figure 1. System model

Searchable cryptography provides ciphertext retrieval as well as ciphertext data security. A typical public key searchable encryption system model consists of three parties, namely Data Owner (DO), Cloud Server (CS), and Data User (DU), as shown in 0. First, the Data Owner extracts the keyword set from the shared file and creates an encrypted query index and uploads the generated encrypted index and encrypted file to the Cloud Server. Next, the data user generates the corresponding trapdoor based on the keywords to be retrieved and submits the trapdoor to the cloud server. Finally, the cloud server uses the query-matching algorithm of the query trapdoor and the encrypted index to pass all the encrypted data matching the keywords to the user, and then the user can decrypt the data locally to obtain the required data.

2.2 Formal definition of the programme

The method in this paper consists of five polynomial algorithms: initialization, key generation, multi-keyword index construction, multi-keyword trapdoor generation, and multi-keyword query matching, as defined below.

$Setup(1^{\lambda}) \rightarrow (param)$. Input safety parameter, output public parameter ram .

$KeyGen(param) \rightarrow (\{pk_s, sk_s\}, \{pk_u, sk_u\})$. Enter the public parameter $param$ to generate the cloud server key, respectively $\{pk_s, sk_s\}$ and the data user's key $\{pk_u, sk_u\}$.

$BuildIndex(param, pk_s, pk_u, W) \rightarrow I$. The data owner extracts a collection of keywords W, uses the cloud server's public key pk_s and the user's public key pk_u to create the encrypted index I.

$Trapdoor(param, pk_u, sk_u, Q) \rightarrow Tr_Q$. The data user uses the set of query keywords Q and his own key to generate a query trapdoor Tr_Q.

$Test(param, pk_u, sk_s, Tr_Q, I) \rightarrow true\backslash false$. The cloud server uses the query trapdoor Tr_Q and the encrypted index I to perform a bilinear pair operation, and if the result matches, output $true$, otherwise output $false$.

3. Related concepts

3.1 Bilinear mapping

Set G_1, G_2 are the cyclic group and G_T is the multiplicative cyclic group and all the order of the group is prime q. Z_q is a finite field of order q of a finite field. Bilinear mappings $e: G_1 \times G_2 \rightarrow G_T$ has the following properties.

Bilinear. For any $u \in G_1$, the $v \in G_2$, the $a, b \in Z_q$. When G_1 and G_2 are multiplicative cyclic groups, the $e(u^a, v^b) = e(u, v)^{ab}$; when G_1 and G_2 are additive cyclic groups, the $e(au, bv) = e(u, v)^{ab}$.

Non-degenerative. $e(g_1, g_2) \neq 1$, where g_1 is the generating element of the G_1, g_2 is the generating element of the G_2.

Computability. For any $u \in G_1$, the $v \in G_2$, all can be efficiently calculated $e(u, v)$.

4. Programme description

4.1 Initialization

$Setup(1^{\lambda}) \rightarrow (param)$. Input safety parameter λ, given the parameters of the bilinear pair $\gamma = (p, G_1, G_2, e, g)$, where G_1, G_2 are the multiplicative cyclic group, and g is the generating element of the groupG_1, the bilinear map is $e: G_1 \times G_1 \rightarrow G_2$. The two hash functions are $H_1: G_2 \rightarrow Z_p^*$, and

$H_2: \{0,1\}^* \rightarrow Z_p^*$. The output common parameters are

$$param = (p, G_1, G_2, e, g, H_1, H_2, Z_p^*).$$

4.2 Key generation

$KeyGen(param) \rightarrow (\{pk_s, sk_s\}, \{pk_u, sk_u\})$. It is executed for the system participants and is divided into two parts, i.e. key generation for the cloud server and key generation for the data user. The exact process is as follows.

(1) Cloud server key generation

$KeyGen_{Server}(param) \rightarrow (pk_s, sk_s)$. The cloud server picks a random value $x \in Z_p^*$ to calculate the first part of the public key $pk_{s,1} = g^x$, selects a random value $\xi \in G_1^*$ as the second part of the public key $pk_{s,2} = \xi$, the cloud server makes the public key $pk_s = \{pk_{s,1}, pk_{s,2}\}$ and keeps its own private key $sk_s = x$.

(2) Data user key generation

$KeyGen_{user}(param) \rightarrow (pk_u, sk_u)$. The data user picks a random number $y \in Z_p^*$, calculates the first part of the public key $pk_{u,1} = g^y$, selects a random value $\theta \in G_1^*$ as the second part of the public key $pk_{u,2} = \theta$, the data user discloses the public key $pk_u = \{pk_{u,1}, pk_{u,2}\}$ and retains their private key $sk_u = y$. This key pair is $\{pk_u, sk_u\}$ which is used to generate the query trapdoor.

4.3 Multi-keyword index construction

$BuildIndex(param, pk_s, pk_u, W) \rightarrow I$. First, the data owner extracts a collection of keywords $W = \{w_1, \dots, w_n\}$ based on the shared file, then the data owner selects a random value $s, r \in Z_p^*$, computes $I_1 = g^s$. Then the data owner selects a random value and uses the public key of the cloud server to calculate the secret value t,

$$t = H(e(pk_{s,1}, pk_{s,2})^s), \tag{1}$$

Then the data owner calculates

$I_2 = \left\{R_1, \left(pk_{u,1} \cdot g^{-H_2(w_2)}\right)^{\frac{r}{t}}, \dots, \left(pk_{u,1} \cdot g^{-H_2(w_i)}\right)^{\frac{r}{t}}, R_{i+1}, \dots, R_n\right\} (i \in [1,n])$, Where R is a random element. If the keyword at the corresponding position of the current keyword set is not included in the file, select a random element to fill. And $I_3 = e(g,g)^r$, $I_4 = e(g, pk_{u,2})^r$.

The final computed encryption index is $I = \{I_1, I_2, I_3, I_4\}$

$$\begin{cases} I_1 = g^s; \\ I_2 = \{R_1, \left(pk_{u,1} \cdot g^{-H_2(w_2)}\right)^{\frac{r}{t}}, \dots, \left(pk_{u,1} \cdot g^{-H_2(w_i)}\right)^{\frac{r}{t}}, R_{i+1}, \dots, R_n\} (i \in [1,n]); \\ I_3 = e(g,g)^r; \\ I_4 = e(g, pk_{u,2})^r。 \end{cases} \tag{2}$$

Encrypted indexes created by the data owner are uploaded to the cloud server.

4.4 Multi-keyword trapdoor generation

$Trapdoor(param, pk_u, sk_u, Q) \rightarrow Tr_Q$. The data user randomly selects $s_Q \in Z_p^*$ as $Tr_1 = s_Q$ and then calculates the aggregated keyword information based on the keywords they want to query $KW = \sum_{i=1}^{|Q|} H_2(w_i)$ and then calculates $Tr_2 = (pk_{u,1} \cdot g^{-s_Q})^{\frac{1}{sk_s - KW}}$, and finally the query keyword is given in the keyword set W the location information in the set of $Tr_3 = \{a_1, \dots, a_{|Q|}\}$ generates

a trapdoor $Tr_Q = \{Tr_1, Tr_2, Tr_3\}$.

$$\begin{cases} Tr_1 = s_Q; \\ Tr_2 = \left(pk_{u,2} \cdot g^{-s_Q}\right)^{\frac{1}{|Q| \cdot sk_u - KW}}; \\ Tr_3 = \{a_1, \ldots, a_{|Q|}\}。 \end{cases} \tag{3}$$

4.5 Multi-keyword query matching

$Test(param, pk_u, sk_s, Tr_Q, I) \rightarrow true \backslash false$. The cloud server first calculates the secret value t for

$$t = H_1(e(I_1, pk_{s,2})^{sk_s}) = H_1(e({g_1}^s, \xi)^x), \tag{4}$$

Then according to $Tr_3 = \{a_1, \ldots, a_{|Q|}\}$. The aggregated keyword information is then calculated based on the position given in IW for

$$IW = \prod_{i=1}^{|Q|} \left(pk_{u,1} \cdot g^{-H_2(w_i)}\right)^{\frac{r}{t}}, \tag{5}$$

Final verification of the equation

$$e(IW^t, Tr_2) \cdot {I_3}^{s_Q} = I_4. \tag{6}$$

The server verifies that equation (6) holds. If the above formula holds, then return $true$, otherwise return $false$. Only if the secret values t calculated by formula (1) and formula (4) are equal, and when the encrypted index is the same as all the keywords contained in the query trapdoor, the correct result is obtained.

5. Programme analysis

To ensure the security of the algorithm in this paper, this section first analyses the correctness of equation (6), and then analyses the security of the scheme.

5.1 Correctness analysis

The correctness of equation (6) is analysed as follows.

$$left = e(IW^t, Tr_2) \cdot {I_3}^{s_Q}$$

$$= e\left(\prod_{i=1}^{|Q|} \left(pk_{u,1} \cdot g^{-H_2(w_i)}\right)^{\frac{r}{t} \cdot t}, \left(pk_{u,2} \cdot g^{-s_Q}\right)^{\frac{1}{|Q| \cdot sk_u - KW}}\right) \cdot e(g, g)^{s_Q r}$$

$$= e\left(g^{|Q| y} \cdot g^{-\sum_{i=1}^{|Q|} H_2(w_i)}, \left(\xi \cdot g^{-s_Q}\right)^{\frac{1}{|Q| \cdot y - \sum_{i=1}^{|Q|} H_2(w_i)}}\right)^r \cdot e(g, g)^{s_Q r}$$

$$= e\left(g^{|Q| y - \sum_{i=1}^{|Q|} H_2(w_i)}, \left(\xi \cdot g^{-s_Q}\right)^{\frac{1}{|Q| \cdot y - \sum_{i=1}^{|Q|} H_2(w_i)}}\right)^r \cdot e(g, g)^{s_Q r}$$

$$= e\left(g^{|Q| y - \sum_{i=1}^{|Q|} H_2(w_i)}, g^{\frac{1}{|Q| \cdot y - \sum_{i=1}^{|Q|} H_2(w_i)}}\right)^{-s_Q r} \cdot e\left(g^{|Q| y - \sum_{i=1}^{|Q|} H_2(w_i)}, \xi^{\frac{1}{|Q| \cdot y - \sum_{i=1}^{|Q|} H_2(w_i)}}\right)^r \cdot e(g, g)^{s_Q r}$$

$$= e(g, \xi)^r = right.$$

5.2 Safety analysis

The approach in this paper focuses on two aspects of security, the security of encrypted indexes and the security of query trapdoors.

(1) Security of encrypted indexes: Firstly, the same keywords in different encrypted indexes have completely different ciphertexts. Assume that given two encrypted indexes I_1 and I_2, assume that $I_1 = I_2$, then there exists $I_{1,1} = I_{2,1}$, and $I_{1,2} = I_{2,2}$, and $I_{1,3} = I_{2,3}$, and $I_{1,4} = I_{2,4}$, and since all parts of the encrypted index are blinded using random elements, the probability of two encrypted indexes being identical is much less than $\frac{1}{p}$, and in addition, the

$I_{1,2} = \{R_{1,1}, \left(pk_{u,1} \cdot g^{-H_2(w_2)}\right)^{\frac{r_1}{t_1}}, \ldots, \left(pk_{u,1} \cdot g^{-H_2(w_i)}\right)^{\frac{r_1}{t_1}}, R_{1,i+1}, \ldots, R_{1,n}\}(\, i \in [1,n])$, and

$I_{2,2} = \{R_{2,1}, \left(pk_{u,1} \cdot g^{-H_2(w_2)}\right)^{\frac{r_2}{t_2}}, \ldots, \left(pk_{u,1} \cdot g^{-H_2(w_i)}\right)^{\frac{r_2}{t_2}}, R_{2,i+1}, \ldots, R_{2,n}\}(\, i \in [1,n])$.

As a result of being different random elements r, the t blinds to the keyword information, the cloud server cannot compute the keyword information according to the discrete logarithmic difficulty assumption and therefore does not disclose any keyword information. Furthermore, it can be noted that the second part of each encrypted index is a collection of encrypted keywords, and the rest of the positions are filled with random elements except for the corresponding keywords, which nicely hides the number of keywords contained in each encrypted index.

(2) Security of query trapdoors, given a query trapdoor Tr_Q that, since the trapdoor uses the authorized user's key sk_u and the trapdoor contains the aggregated information of all the keywords that the authorized user wants to query $\sum_{i=1}^{|Q|} H_2(w_i)$. Therefore, the server cannot decrypt the plaintext keyword information, satisfying the discrete logarithmic difficulty assumption. In addition, the trapdoor satisfies the unlinkability between trapdoors, assuming that the query keyword set $Q = \{w_1, \ldots, w_{|Q|}\}$ query trapdoor is generated, the data user runs the $Trapdoor$ algorithm to generate $Tr_Q^1 = \{Tr_{1,Q}^1 = s_Q, Tr_{2,Q}^1 = \left(pk_{u,2} \cdot g^{-s_Q}\right)^{\frac{1}{|Q| \cdot sk_u - KW}}, Tr_{3,Q}^1 = \{a_1, \ldots, a_{|Q|}\}\}$, and $Tr_Q^2 = \{Tr_{1,Q}^2 = s_Q', Tr_{2,Q}^2 = \left(pk_{u,2} \cdot g^{-s_Q'}\right)^{\frac{1}{|Q| \cdot sk_u - KW}}, Tr_{3,Q}^2 = \{a_1', \ldots, a_{|Q|}'\}\}$. Assuming $Tr_Q^1 = Tr_Q^2$, then there must exist $Tr_{1,Q}^1 = Tr_{1,Q}^2$ that $Tr_{2,Q}^1 = Tr_{2,Q}^2$, and $Tr_{3,Q}^1 = Tr_{3,Q}^2$, i.e. $s_Q = s_Q'$, however $s_Q = s_Q'$ the probability that is less than or equal to $\frac{1}{p}$, is negligible, so it is not possible to distinguish whether two query traps contain the same keyword, i.e. the query traps corresponding to the same keyword have unlinkability.

6. Analysis of experimental results

Experiments were conducted using the JPBC library on two windows 11 systems with 1.80Ghz AMD Ryzen 74800U and 16GB RAM, one simulating a data owner and one simulating a data user, in addition to a cloud server with a dual-core CPU and 8GB RAM selected from the Aliyun ECS for storing the encrypted index and for query matching. In this paper, Type A symmetric bilinear pairing is chosen to complete the specific algorithm. The dataset is selected from the real dataset provided by Google.

(1) The computational overhead of creating an encrypted index. The calculation cost required by the data owner to create an encrypted index is shown in 0. As the number of keywords increases, the calculation cost of index construction increases slowly. This is because the second part of the encrypted index needs to be encrypted according to the size of the keyword set, thus causing a part of the computational overhead, and secondly, the computational overhead of the index is also mainly affected by the change of the number of files, the more The higher the number of files, the higher the

number of encrypted indexes.

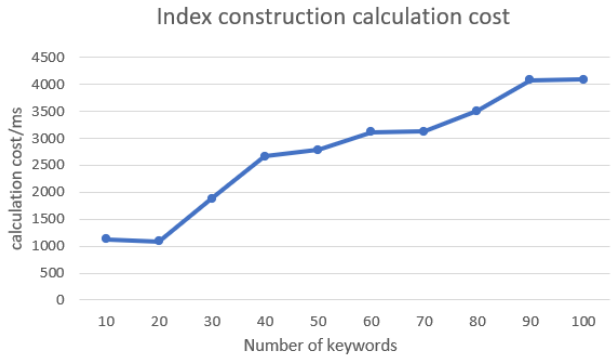

Figure 2. Index construction calculation cost

(2) Computational overhead for trapdoor generation. 0 shows the computational overhead for generating query traps when a data user wants to query. This is because when a data user wants to query, he only needs to calculate the aggregated query information and generate a valid query trap door based on all the keywords he wants to query, instead of generating the corresponding query trap door for all the keywords separately, which greatly reduces the computation overhead of generating the query trap door for the data user. This greatly reduces the computational overhead of query trapdoor generation by data users.

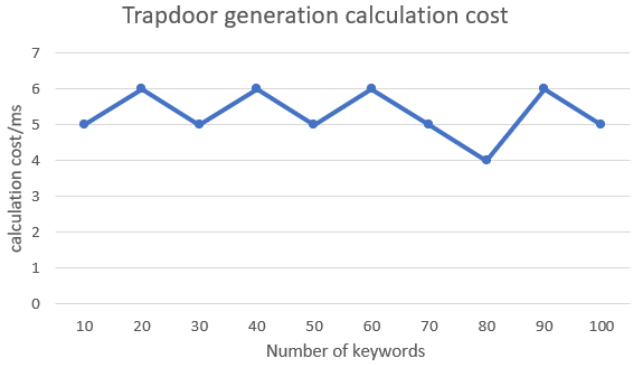

Figure 3. Trapdoor generation calculation cost

(3) Computational overhead for query matching. 0 shows the computational overhead of the cloud server when performing query matching. As can be seen from the dashed line in the figure, the query matching overhead increases approximately linearly as the number of keywords increases and as the number of indexes included increases.

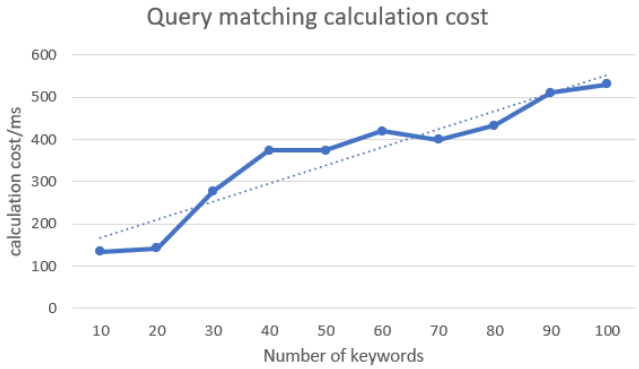

Figure 4. Query matching calculation cost

7. Conclusion

This paper provides a multi-key public key searchable encryption method that can protect the number of keywords in encrypted data, with certain application value, in response to the problem that most traditional public key searchable schemes with keywords without secure channel only have single keyword search.

Acknowledgments

This work was supported by Shanghai Natural Science Foundation (19ZR1402000 and 21ZR1400400), National Natural Science Foundation of China (62172088 and 61772018).

References

[1] Ali M, Khan S U, Vasilakos A V. Security in cloud computing: opportunities and challenges [J]. Information sciences, 2015, 305: 357-383.

[2] D. X. Song, D. Wagner, A. Perrig. Practical Techniques for Searches on Encrypted Data [C]// IEEE Symposium on Security & Privacy. IEEE, 2002.

[3] Dan B, Crescenzo G D, Ostrovsky R, et al. Public Key Encryption with Keyword Search [J]. EUROCRYPT 2004, 2004.

[4] J. Baek, R. Safavi-Naini and W. Susilo. Public Key Encryption with Keyword Search Revisited[C]. In Proc. of Applied Cryptography and Information Security 06 (ACIS 2006), LNCS 5072, Springer-Verlag, 2008:1249-1259.

[5] C. Gu, Y Zhu, and H. Pan. Efficient Public Key Encryption with Keyword Search Schemes from Pairings[C]. In Proc. of Information Security and Cryptology: Third SKLOIS Conference, Inscrypt 2007, LNCS 4990, Springer-Verlag, 2007:372-383.

[6] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. Improved searchable public key encryption with designated tester. Proc. of the 4th international Symposium on information, Computer, and Communications Security, ASIACCS 2009, ACM, New York, NY, 2009:376-379.

[7] Fang L, Susilo W, Ge C, et al. Public key encryption with keyword search secure against keyword guessing attacks without random oracle [J]. Information Sciences, 2013, 238: 221-241.

[8] Z. Fu, K. Ren, J. Shu, X. Sun, F. Huang, Enabling personalized search over encrypted outsourced data with efficiency improvement, IEEE Trans. Parallel Distrib. Syst. 27 (9) (2016) 2546-2559.

[9] Huang Qiong, Li Hong-bo. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks [J]. Information Sciences, 2017, s 403-404:1-14.

[10] Curtmola R, Garay J, Kamara S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[C]//Proceedings of the 13th ACM conference on Computer and communications security. 2006: 79-88.

[11] Yin H, Li Y, Deng H, et al. An Attribute-Based Keyword Search Scheme for Multiple Data Owners in Cloud-Assisted Industrial Internet of Things [J]. IEEE Transactions on Industrial Informatics, 2022.