Multi-User Revocable Keyword Search Scheme for

Liu Mei TAN, Xiao Lei DONG, Zhen Fu CAO, Jia Chen SHEN
School of Computer Science and Software Engineering, East China Normal University, Shanghai, Shanghai 200062, China
Secure Data Sharing in Cloud
51151500048@stu.ecnu.edu.cn

Abstract. With the rapid development of cloud computing, more and more users store their data in the cloud. Considering the privacy problem of the data sharing in the cloud, searchable encryption makes it possible to efficiently share and retrieve data in multi-user settings. The existing keyword search scheme of sharing the private key with authorized users to generate trapdoors may increase the risk of key exposure, and the scheme of encrypting the data with the secret key of each authorized user and generating a ciphertext for each user may not be suitable for data sharing between a large number of users. In this paper, we combine the secure IBBE system with the keywords search and propose the scheme to achieve safe and efficient multi-user data sharing in the cloud. Authorized users can perform keyword search directly on encrypted data without sharing the secret key, and for a shared file, the data owner only needs to encrypt one ciphertext for all authorized users. Besides, authorized users of shared files can be revoked. User authorization tables and IBBE ensure that revoked users cannot access files.

1. Introduction
In the era of big data, more and more amounts of data are created, making it difficult for users to store, share, analyze and utilize existing data through local limited hardware devices such as intelligence Mobile phones and tablet PCs [1]. In order to solve this problem, cloud computing [2] is proposed and applied. Cloud computing has become a new model with convenient computing resource allocation, flexible and various access, and capital savings.

While cloud computing has improved, users’ concerns about data security are major obstacles to widespread deployment of cloud computing [3]. In order to protect the privacy of the data, the data owner encrypts all data before sending it to the cloud server, and then the encrypted data can be retrieved and decrypted by the user with the decryption key. However, encryption may destroy some of the properties of the original data and make it difficult to use encrypted data, which let many data operations become infeasible. Among them, the keyword search is one of the most important operations for efficient data retrieval and sharing in the cloud [4]. Therefore, developing efficient and secure searchable encryption (SE) [5-6] schemes and allowing users to search over ciphertext through keywords is necessary.

Many searchable encryption schemes have been proposed to solve the problem of multi-user data sharing. One idea of existing keyword search schemes is to allow authorized users to perform encryption/decryption and generate a trapdoor for a specific keyword by sharing the private key, which implies that a group of users perform operations with the same private key like the scheme in [7]. It is efficient but may increase the risk of key exposure. Another idea as the scheme [8] is that the data owners encrypt files by using the key of each authorized user and generate a ciphertext for each user,
users then use their own private key to search. While this technology is secure, it may not be suitable for data sharing between a large number of users.

In our paper, utilizing the identity-based broadcast encryption (IBBE) in [9] and the idea of [4], we build an efficient and simple user keyword search scheme, in which the data owner only needs to encrypt files once when he/she want to share the file with other users. Our scheme achieves multi-user data sharing, keyword search and receiver revocation, and the safety could be ensured at the same time.

1.1. Related Work
The notion of searching on encrypted data was first introduced by Song et al. [5]. In [5], the authors propose an encryption scheme where users store encrypted data in a server then search for encrypted data, and the scheme is provided proof of security for the resulting. Later, Goh [10] introduce a bloom filter to build a keyword search security index that allowed the server to check whether a document contained the keyword without having to decrypt the entire file. The public key encryption with keyword search (PEKS) scheme is created by Boneh et al. [11]. Di Crescenzo and Saraswat [12] present a PEKS scheme based on Jacobi symbol and Khader [13] introduces how to construct PEKS based on K-Resilient IBE. Based on pairings on elliptic curves, Golle et al. [14] first proposed keyword conjunctions. Bao [15-16] consider multi-user query. The scheme makes each user have a private key to generate trapdoors for keywords which they want to search for. In [17], the authors extend the scheme in multi-user setting. It needs to generate a private key for users, and cannot allow the user’s dynamic revocation and authorization. Dong [18] proposes an encryption scheme in which each authorized user in the system has its own key to encrypt and decrypt the data and execute keyword search. And Li et al. [3] proposed a scheme which achieves efficient and secure data sharing in cloud computing with the hybrid cloud for keyword search.

1.2. Organization
The rest of the paper is organized as follows. In Section 2, we first introduce the system architecture, next we review the preliminary knowledge. Then there will be the formal definition of our scheme. In Section 3, we describe our scheme in detail. The Security analysis will be presented in Section 4. Finally, we will draw our conclusion in Section 5.

2. System Formulation

2.1. System Architecture
We consider the application scenario of our scheme like that company colleagues can upload their own files to the cloud to share, such as departmental business, work experience and technology sharing. The file owner specifies that the departmental business documents can only be accessed by colleagues in this department, and work experience and technology sharing documents can be obtained by colleagues across the company. Colleagues can search for the files they need. When a departmental colleague is transferred to another department, the data sharer can revoke the colleague's right to access the department's business documents, but the colleague can still search other documents like work experience and technology sharing documents. The system model is shown as Figure 1, which involves the following participants.

**Trusted Authority (TA):** Trusted authority is responsible for initializing system parameters, granting new users to the system, and generating private keys of users.

**Cloud Server:** It is in charge of storing encrypted file message for data owners who want to share file in cloud. In addition, if the user who originally accessed the file was told to revoke, the server executes revocation by updating the ciphertext of the file.

**Data Owner:** Data owners can share files on the cloud server, generate indexes for keywords and upload encrypted files with the indexes. Besides, data owners decide who can access the file.

**Data User:** Valid data users can choose keywords to generate trapdoor, search files with the help of
the cloud server, and decrypting the file. Once the user is revoked, the user cannot search and decrypt the file.

![System model](image)

**Figure 1 System model**

### 2.2. Formal Definition

The multi-user revocable keyword search scheme consists of nine algorithms, which are defined as follows.

- **Setup(\(\lambda\))**: The initialization algorithm is run by the TA. The algorithm takes as input a security parameter \(\lambda\), outputs a master public key \(PK\) and a master secret key \(SK\).

- **GrantUser(\(PK, SK, ID\))**: The grant user algorithm is run by the TA. The algorithm takes as input a master public key \(PK\), a master secret key \(SK\) and an user identity \(ID\) who want to join the system, generates an identity key pair \(<uk_{ID}, uk'_{ID}>\) for \(ID\), \(uk_{ID}\) is the identity key and \(uk'_{ID}\) is the complementary key of identity key, sends \(<ID, uk_{ID}>, <ID, uk'_{ID}>\) to the user and cloud server respectively.

- **KeyGen(\(PK, SK, ID\))**: The key generation algorithm is run by the TA. The algorithm takes as input a master public key \(PK\), a master secret key \(SK\) and user identity \(ID\), generates private key \(pk_{ID}\) for each user \(ID\).

- **BuildIndex(\(W, PK\))**: The build index algorithm interacts by the cloud server and data owner. The algorithm takes as input the keyword set \(W\) and a master public key \(PK\), outputs the index \(I(W)\) of keyword set.

- **Encrypt(\(PK, M, S\))**: The encryption algorithm is run by data owners. The algorithm takes as input the master public key \(PK\), a file message \(M\) and a set of identity \(S\) who can access the file, generates the ciphertext \(C\) with a unique file identifier \(f_{id}\), and sends the related information \((f_{id}, C, I(W), S)\) of shared file to cloud server.

- **Revoke(\(PK, R, CT\))**: The revocation algorithm is run by the cloud server. The algorithm takes as input the master public key \(PK\), and ciphertext information \(CT\), and receiving a revoked user set \(R\) from data owner, The algorithm updates the ciphertext \(C\) and accessible user set of identity \(S\), and save \((f_{id}, C, I(W), S, R)\) for the specific shared file.
**Trapdoor**($w', ID$): The revocation algorithm is run by data users. The algorithm takes as input the keyword $w'$ and the identity $ID$ of the user, generates the trapdoor $Q(w')$, and sends the $ID$ and $Q(w')$ to the server.

**Search**($ID, Q(w'), CT'$): The search algorithm is run by the cloud server. The algorithm takes as input the user identity $ID$, the trapdoor $Q(w')$, and ciphertext information $CT'$, sends ciphertext information $CT'$ to the user under the condition that there is matching keywords in the ciphertext and the user is authorized.

**Decrypt**($CT', PK, ID_j, pk_{id_j}$): The decryption algorithm is run by the data user. The algorithm takes as input ciphertext information $CT'$, a master public key $PK$, the user identity $ID_j$ and private key $pk_{id_j}$ of the user, outputs the plaintext $M$ for the valid user.

### 2.3. Preliminary

**Bilinear Map.** Let $G$ and $G_T$ be two cyclic groups of prime order $p$, and $g$ be a generator of $G$. Let $e: G \times G \to G_t$ denote the bilinear map. The bilinear map $e$ satisfies the following properties:

- Bilinear: for all $g_1, g_2 \in G$ and $a, b \in Z_p^*$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- Non-degeneracy: exists $g_1, g_2 \in G$ such that $e(g_1, g_2) \neq 1$;
- Computable: $e(g_1, g_2)$ can be efficiently computed for any $g_1, g_2 \in G$.

**Bilinear Diffie-Hellman Problem (BDH).** Let a generator $g$ of $G$. The BDH problem is as follows: given $g, g^a, g^b, g^c \in G$ for $a, b, c \in Z_p^*$, compute $e(g, g)^{abc} \in G_T$. It is intractable if all polynomial time algorithms have a negligible advantage $\epsilon$ in solving BDH.

### 3. The Proposed Scheme

#### 3.1. Construction

**Setup($\lambda$):** Given a security parameter $\lambda$, the setup algorithm chooses bilinear groups $G, G_t$ with prime order $p$, and a bilinear map $e: G \times G \to G_t$. Pick a random $\alpha \in Z_p^*$ and a generator $g$ of $G$, then set $BG = \{G, G_t, e, p\}$, $g_\alpha = \alpha \cdot g$. Select four secure cryptographic hash functions $H: \{0,1\}^\ast \to Z_p^*$, $H_1: \{0,1\}^\ast \to G$, $H_2: G \times \{0,1\}^\ast \to G$, $H_3: G \times \{0,1\}^\ast \to G$, $H_4: G_t \to \{0,1\}^{log \ p}$. Finally, output master public key $PK = \{BG, g, g_\alpha, H, H_1, H_2, H_3, H_4\}$, master private key $SK = \alpha$.

**GrantUser($PK, SK, ID$):** Given master key pair $PK, SK$ and an identity $ID \in \{0,1\}^\ast$. First, it generates an identity key pair $<uk_{id}, Z_p^*, uk_{id}^* = g^{\alpha uk_{id}}>$ for user $ID$. Then, it sends $<ID, uk_{id}>$ to the user, and sends $<ID, uk_{id}^*>$ to cloud server to save the system user list $L$.

**KeyGen($PK, SK, ID$):** The algorithm generates decryption private key $pk_{id}$ for user $ID$ as $pk_{id} = \alpha H(ID)$.

**BuildIndex($W, PK$):** Given the keyword set $W = \{w_1, w_2, ..., w_m\}$ of the shared file message $M$, the index is built as follow. First, the data owner randomly chooses $r \in Z_p^*$, and uploads his/her identity $ID$ and $\{H_1(w_1), H_1(w_2), ..., H_1(w_m)\}$ to the cloud server. After the cloud server receives the request, it searches the corresponding $uk_{id}^*$ of $ID$ in $L$. Then, the cloud server calculates $C_{p_n} = e(H_1(w_i)^r, uk_{id}^*)$ for each $w$ and sends them to the data owner. Finally, the data owner computes $k_{W_i} = H_4(C_{p_n}^{uk_{id}})$, and chooses $D \in Z_p^*$, then builds the index $I(W) = \{D, HMAC_{\alpha}(D), i \in [1, m]\}$ for keyword set $W$. 
Encrypt(\(PK, M, S\)) : Suppose a data owner wants to share a file \(M\) with a set identity of user \(S = (ID_1, ID_2, \ldots, ID_n)\). First, select a unique file identifier \(f_{id} \in \{0,1\}^*\). Second, randomly choose \(r_i, r_j \in Z_p^*\) and \(t_i \in G\), compute \(x_i = H(ID_i)\) for \(i = 1, 2, \ldots, n\), and construct
\[
f_i(x) = \prod_{j=1}^{n} x_i - \sum_{j=0}^{n} d_{i,j} x_j \mod p, \quad A_i = H_2(e(H(ID_i), g_{x_i}^y), ID_i), \quad B_i = t_i \cdot H_3(e(H(ID_i), g_{x_i}^y), ID_i).
\]
We can know
\[
f_i(x) = 1 \text{ and } f_j(x) = 0 \text{ for } i \neq j.
\]
Third, generate the ciphertext \(CT = (f_{id}, C_1, I(W), S)\) to cloud server.

Revoke(\(PK, CT, R\)) : After receiving a revocation identity set \(R\) with file identifier \(f_{id}\) from data owner where \(|R| = t (t < n)\). If \(R = \emptyset\), the algorithm sets \(C' = C\). Otherwise, it searches for \(CT\) according to \(f_{id}\), and updates \(CT\) as follow. First, it randomly selects \(t_z \in G\) and computes \(x_i = H(ID_i)\) for each \(ID_i \in R\). \(C_0 = t_z \cdot C_0\) constructs \(g(x) = \prod_{i=1}^{t} (x - x_i) = \sum_{i=0}^{t} b_i x^i \mod p\). Then for each \(i = 1, 2, \ldots, n\), it computes \(P_i = P_i - t_z \cdot x_i\) and sets \(b_i = 0\) for \(i = t+1, t+2, \ldots, n-1\). Finally, it updates the ciphertext \(C' = (C_0, C_1, C_2, P_i, U_i, i \in [1, n] )\), accessible user set of identity \(S' = S - R\), and saves \(CT' = (f_{id}, C', I(W), S', R)\) for the specific shared file.

Trapdoor(\(w, ID\)) : The algorithm generates the trapdoor \(Q(w) = H_1(w^w)\) for the keyword \(w\). Then, send the user identity ID and \(Q(w)\) to the server.

Search(\(ID, Q(w)\), \(CT\)) : After receiving the trapdoor \(Q(w)\) from an identity \(ID\), the server performs the search as follows. First, check and get the complementary key \(uk_{ID}'\) of the user ID in user list \(L\). Second, compute \(kw' = H_4(e(Q(w), uk_{ID}'))\) and check each index of the data \(CT'\) as \(HMAC_{kw'}(ID) = HMAC_{kw}(ID)\)? . If existing equations, then determine whether the user ID belongs to \(S'\) of \(CT'\) which satisfies keyword search. If all stages are passed, the server sends \(CT'\) to user.

Decrypt(\(CT', PK, ID, pk_{ID}\)) : After receiving \(CT'\), the algorithm is run by the user \(ID_j \in S'\) with the private key \(pk_{ID}\) as follow. First, compute \(x_j = H(ID_j)\), \(U = U_1 \cdot U_2 \cdot U_3 \cdot \cdots U_n\), \(P = P_1 \cdot P_2 \cdot P_3 \cdot \cdots P_n\). Second, compute \(g(x) = \prod_{i=1}^{t} (x - x_i) = \sum_{i=0}^{t} b_i x^i \mod p\). Third, compute \(t_1, t_2\) with private key \(pk_{ID}\) as
\[
t_1 = H_2(e(C_2, pk_{ID}), ID_j)^{-1} U, \quad t_2 = (P \cdot H_2(e(C_1, pk_{ID}), ID_j)^{x_j-1})^{\frac{1}{t_2}}.
\]
Finally, the user can recover \(M = C_0 \cdot (t_1, t_2)^{-1}\).

3.2. Correctness
The construction meets the correctness requirement as the following description.

1) \(kw' = H_4(e(Q(w'), uk_{ID}')) = H_4(e(H_1(w')^{uk_{ID}'}, uk_{ID}'))\),
\(kw = H_4(C_{w'}^{uk_{ID}'}) = H_4(e(H_1(w')^{uk_{ID}'}, uk_{ID}'))\)

It is easy to know the keyword satisfies \(w' = w\) if \(kw' = kw\).

2) As \(x_i = H(ID_i)\), if \(ID_i \in R\), we have \(g(x) = 0\). The revoked user with identity \(ID\) cannot get the
decryption key $t_2$. If $ID \in S'$, we have $g(x_i) \neq 0$. And
\[
P = P_1^2 \cdot P_2^x \cdots P_n^x = (P_1 \cdot (P_2)^x \cdots (P_n)^x) \cdot (t_2^x + b_{x_i}z + b_{x_i}z^2 + h_{x_i}x^{-1})
\]
\[
= (A_1^{x_1} \cdot A_2^{x_2} \cdots A_n^{x_n}) \cdots (A_1^{x_n} \cdot A_2^{x_2} \cdots A_n^{x_n}) \cdot (t_2^x + b_{x_i}z + b_{x_i}z^2 + h_{x_i}x^{-1})
\]
\[
= (A_1^{x_1} + a_{x_1}z + a_{x_1}z^2 + h_{x_1}x^{-1}) \cdots (A_1^{x_n} + a_{x_n}z + a_{x_n}z^2 + h_{x_n}x^{-1}) \cdot (t_2^x)
\]
\[
= A_1^{x_1} \cdot A_2^{x_2} \cdots A_n^{x_n} \cdot t_2^x = A_1^{x_1} \cdot A_2^{x_2} \cdots A_n^{x_n}
\]
\[
t_2 = (P \cdot H_2(e(C_1, pk_{ID}), ID))^{-1} = (A_1 \cdot A_2 \cdots A_n)^{-1}
\]
\[
= (A_1 \cdot t_2^x \cdot H_2(e(C_1, pk_{ID}), ID))^{-1} = (t_2^x)^{-1} = t_2
\]
\[
U = U \cdot U_2^x \cdots U_n^x = (B_1^{x_1} \cdots B_2^{x_2} \cdots B_n^{x_n}) \cdots (B_1^{x_n} \cdots B_2^{x_2} \cdots B_n^{x_n})
\]
\[
= (B_1^{x_1} + a_{x_1}z + a_{x_1}z^2 + h_{x_1}x^{-1}) \cdots (B_2^{x_2} + a_{x_2}z + a_{x_2}z^2 + h_{x_2}x^{-1}) \cdots (B_n^{x_n} + a_{x_n}z + a_{x_n}z^2 + h_{x_n}x^{-1})
\]
\[
= B_1^{x_1} \cdots B_2^{x_2} \cdots B_n^{x_n}
\]
\[
t_1 = H_1(e(C_1, pk_{ID}), ID)^{-1} = H_1(e(C_1, pk_{ID}), ID)^{-1} = t_1
\]

So authorized users can recover the file message after revocation as
\[
C_0 \cdot (t_1 \cdot t_2)^{-1} = t_2 \cdot C_0 \cdot (t_1 \cdot t_2)^{-1} = t_2 \cdot t_1 \cdot t_1 \cdot t_2 = M \cdot (t_1 \cdot t_2)^{-1} = M
\]

If no one is revoked, we have $P = A_1$, $U = B$ and know $t_2 = 1$, $t_1 = t_1$. So users can recover the file message as $C_0 \cdot (t_1 \cdot t_2)^{-1} = C_0 \cdot (t_1 \cdot t_2)^{-1} = t_1 \cdot t_1 \cdot t_1 = M \cdot (t_1 \cdot t_2)^{-1} = M$

4. Security Analysis

4.1. Message Privacy

In our scheme, the message in the ciphertext $C$ cannot be distinguished without a valid private key associated with an identity $ID \in S$. The message in $CT'$ cannot be distinguished without a valid private key associated with an identity $ID' \in S$ and $ID' \notin R$. The scheme satisfies IND-ID-CPA security under BDH problem as proved in [9]. Besides, in the scheme, only the authorized users can decrypt the ciphertext. After revocation, even if all the revoked users collude, they cannot recover the one of decryption keys $t_2$. In order to obtain the decryption keys, the user must belong to $S$ and not belong to $R$. Thus our scheme ensures that even if all the revoked users collude, they still cannot access the file. So, no plaintext information is leaked to the cloud server or unauthorized users. However, in the scheme, the cloud server can know which identity of users can access the ciphertext.

4.2. Keyword Privacy

In our scheme, while the data owner is generating the index, the cloud server and attackers can obtain $h(w)'$ in the process. However, because of the one-way hash function used, both the cloud server and attackers cannot deduce any information about the keyword set. As random number $r$ is used, replay attack can also be prevented. Besides, each user creates the trapdoor for keywords by user’s identity privacy key instead of the queried keyword being submitted to server, so the server cannot know which keyword is searched. Unless a user’s secret query key is compromised, neither the cloud server nor unauthorized user can forge a query. And for compromised keys, our scheme is allowed to dynamically and efficiently revoke users.
5. Conclusion
In this paper, we present the scheme in which users can share files with others in cloud. For a shared file, the data owner only needs to encrypt one ciphertext for all authorized users, and authorized users can perform keyword search directly on ciphertext and decrypt files without sharing the secret key. Besides, authorized users of shared files can be revoked, and user authorization tables and IBBE ensure that revoked users cannot access file. After the construction we analyze the message privacy and keyword privacy of our scheme.

In our future work, on the basis of this program function, we intend to achieve anonymous searchable function so that receivers’ identity is unknown to cloud server and only the sender knows the receivers’ identities.

References
[1] Lu R., Zhu H., Liu, X Liu, J.K., Shao J. Toward efficient and privacy-preserving computing in Big Data era[J].IEEE Network, 2014, 28(4):46-50.
[2] Chen K, Zheng WM. Cloud computing: system instances and current research[J]. Journal of software, 2009, 20(5):1337-1348.
[3] J Li, J Li, Z Liu, C Jia. Enabling efficient and secure data sharing in cloud computing[J]. Concurrency and Computation: Practice & Experience, 2014, 26 (5) :1052-1066.
[4] F Zhao, T Nishide, K Sakurai.Multi-user keyword search scheme for secure data sharing with fine-grained access control[C].Proceedings of the 14th international conference on Information Security and Cryptology. ICISC, 2011, 7259:406-418.
[5] Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data[C].Proceedings of the 21th IEEE Symposium on Security and Privacy. IEEE, 2000: 44–55.
[6] Waters BR, Balfanz D, Durfee G, Smetters DK. Building an encrypted and searchable audit log[C]. Annual Network & Distributed System Security Symposium .San Diego,California, 2004:5–6.
[7] Q Xia, J Ni, JC Gee. Searchable public-key encryption with data sharing in dynamic groups for mobile cloud storage[J]. Journal of Universal Computer, 2015, 21 (3):440-453.
[8] Y Miao, J Ma, Z Liu. Revocable and anonymous searchable encryption in multi-user setting[J]. Concurrency and Computation: Practice & Experience, 2016, 28 (4):1204-1218.
[9] J Lai, Y Mu, F Guo, W Susilo, R Chen. Anonymous Identity-Based Broadcast Encryption with Revocation for File Sharing[C].Proceedings of Australia Conference on Information Security and Privacy. ACISP, 2016, 223–239.
[10] Goh EJ. Secure indexes[C]. IACR Cryptology ePrint Archive 2003, 2003:1–18.
[11] Boneh, D., Di Crescenzo, G., Ostrovsky, R. and Persiano, G. Public Key Encryption with Keyword Search[C]. Advances in Cryptology—EUROCRYPT 2004. Computer Science, 3027(16):506-522.
[12] Di Crescenzo, G.,Saraswat,V. Public Key Encryption with Searchable Keywords Based on Jacobi Symbols[C]. Progress in Cryptology—INDOCRYPT 2007, 8th international conference. Computer Science, 2007, 4859:282-296.
[13] Khader, D.Public Key Encryption with Keyword Search Based on k-Resilient IBE[C].Computational Science and Its Applications ICCSA 2006 international conference.Computational Science ,2006,4707:298-308.
[14] Golle, P., Staddon, J. and Waters, B. Secure Conjunctive Keyword Search Over Encrypted Data[C]. Applied Cryptography and Network Security, 2nd Int. Conf. ACNS, 2004, 3089:31–45.
[15] Bao, F., Deng, R.H., Ding, X., Yang, Y. Private Query on Encrypted Data in Multi-user Settings[C].Proceedings of the 4th international conference on Information security practice and experience. ISPEC, 2008, 4991:71-85.
[16]Yang, Y., Ding, X., Deng, R.H., Bao, F. Multi-User Private Queries over Encrypted Databases[J]. International Journal of Applied Cryptography, 2009, 1(4):309-319.
[17] Hwang YH, Lee PJ. Public key encryption with conjunctive keyword search and its extension to a
multi-user system[C]. Proceedings of the First international conference on Pairing-Based Cryptography. In Pairing, 2007, 4575:2-22.

[18] Dong C, Russello G, Dulay N. Shared and searchable encrypted data for untrusted servers[J]. Journal of Computer Security, 2011, 19(3):367–397.