Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 5634561, 16 pages
https://doi.org/10.1155/2018/5634561

Research Article

Flexible, Secure, and Reliable Data Sharing Service Based on Collaboration in Multicloud Environment

Qiang Wei[1], **Huaibin Shao**[2] **and Gongxuan Zhang**[1]

[1]*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China*
[2]*Jiangxi G.H. Quality & Security Technology Co., Ltd., China*

Correspondence should be addressed to Qiang Wei; weiqiangarticle@163.com

Received 11 January 2018; Accepted 15 March 2018; Published 30 April 2018

Academic Editor: Ao Zhou

Due to the abundant storage resources and high reliability data service of cloud computing, more individuals and enterprises are motivated to outsource their data to public cloud platform and enable legal data users to search and download what they need in the outsourced dataset. However, in "Paid Data Sharing" model, some valuable data should be encrypted before outsourcing for protecting owner's economic benefits, which is an obstacle for flexible application. Specifically, if the owner does not know who (user) will download which data files in advance and even does not know the attributes of user, he/she has to either remain online all the time or import a trusted third party (TTP) to distribute the file decryption key to data user. Obviously, making the owner always remain online is too inflexible, and wholly depending on the security of TTP is a potential risk. In this paper, we propose a flexible, secure, and reliable data sharing scheme based on collaboration in multicloud environment. For securely and instantly providing data sharing service even if the owner is offline and without TTP, we distribute all encrypted split data/key blocks together to multiple cloud service providers (CSPs), respectively. An elaborate cryptographic protocol we designed helps the owner verify the correctness of data exchange bills, which is directly related to the owner's economic benefits. Besides, in order to support reliable data service, the erasure-correcting code technic is exploited for tolerating multiple failures among CSPs, and we offer a secure keyword search mechanism that makes the system more close to reality. Extensive security analyses and experiments on real-world data show that our scheme is secure and efficient.

1. Introduction

With the arrival of data times, both individuals and enterprises are producing huge amount of data every day, which profoundly influences our living style. On one hand, the explosively increasing data creates the limits for data storage and data transmission; on the other hand, data has become an important productivity resource that could help users make data-driven decisions by some computational methods, such as giving optimized scheme by operational research [1, 2], giving accurate prediction by regression analysis [3], or carrying out geometrical computation [4]. Therefore, it is necessary to design an efficient data sharing model to share huge amount of data among different individuals or organizations. Obviously, cloud computing is certainly a prefect data sharing tool for its vast storage space and reliable distributed storage system.

However, even if there are various advantages of cloud computing, security and privacy concerns are the primary obstacles to wide adoption [5]. In terms of data sharing, the data owner would like to outsource his/her data files to the remote public cloud servers and enable other users to find and download what they are interested in from cloud dataset. Although the security issues of data transmission have been deeply researched [6], which guarantees that the adversary is difficult to steal data privacy during transmission process, some potential risks still exist. For example, "curious" cloud service providers may steal valuable data by themselves for some economic benefits. To address such problem, a general approach is that the owner encrypts his/her data files with self-chosen keys before outsourcing [7]. But, in this case, how the owner flexibly and securely shares data with users is still not fully investigated.

Previous works have proposed two kinds of encrypted cloud data sharing models. *The first one* is that the data owner

encrypts their outsourced data with specific encryption key(s) so that only specific authorized data users could access and decrypt encrypted data. For example, the identity-based encryption (IBE) schemes [8, 9] and proxy-based reencryption (PRE) schemes [10, 11] enable the owner to encrypt outsourced data based on the users' identification or their public key information. Moving a step forward, the attribute-based encryption (ABE) schemes [12, 13] could achieve a more flexible encryption model that enables data owner to encrypt data for user groups with certain attributes' combinations instead of one specific user. The ABE has become one of the most popular methods for access control field. *The second one* is that the data owner directly shares the decryption key information with authorized users. This method is widely used in most searchable encryption (SE) schemes [14, 15]; it enables the owner to give the decryption key after knowing who (data user) will download which data files of him/her. Obviously, this method provides a significant advantage in "Paid Data Sharing" model ("Paid Data Sharing" model: users should pay for the downloaded data files according to their amount and prices).

Unfortunately, both methods still have some limits when utilizing them in real applications. *Firstly*, in real data sharing system (like 4shared.com and mymedwall.com), data owner cannot determine which outsourced data files will be downloaded by which data users (even the users' attributes) before outsourcing. So the solutions based on the first method like IBE, PRE, and ABE are not feasible anymore, because they need the owner to possess the specific information of users for each data file. *Secondly*, under the "Paid Data Sharing" model, it is unreasonable to make owner give all the decryption keys to the authorized users in system initialization phase. That is to say, if we follow the second method, the owner has to remain online all the time for users instantly obtaining corresponding decryption key(s). However, in reality, it is common that the data owner is the individual or resource-limited enterprises that find it difficult to always remain online.

Aside from the two limits we mentioned above, in "Paid Data Sharing" scenario, it is significantly important to correctly record the details of each data exchange transaction (called "bill"), which is directly related to the benefits of owner. An alternative solution to address all these problems is to import a trusted third party (TTP) in the system; the TTP is responsible for storing and distributing the encryption keys to authorized users and maintaining the bills simultaneously. Obviously, there is no doubt that data confidentiality and bill correctness are both well guaranteed by TTP, even if the owner is completely offline. But the security of such solution relies wholly on the TTP; once the TTP is broken, all plaintext form of data will be leaked. Besides, it is not easy to find a trusted entity as TTP in practical network environment. The similar views have been early proposed in many previous works, such as the key escrow problem in ABE schemes: the security of these schemes depends majorly on the key authority (like our TTP), which will make the confidentiality of outsourced data at a potential security risk [12, 16]. From the above, it is necessary to design a flexible and secure data sharing scheme in "Paid Data Sharing" scenario, which supports offline data owner without TTP.

In this paper, we discuss a "Paid Data Sharing" scheme that can be simply described as follows: a data owner encrypts his/her multiple data files and outsources them to public cloud platform, so that all authorized users could search what they are interested in and then pay and download them from cloud server. Finally, users decrypt downloaded encrypted data locally. Note that our scheme is more practical and flexible compared with previous works. Specifically, the owner does not know which data users (even their attributes) will download and decrypt which outsourced data files of him/her, while the owner cannot always keep online and there is no trusted third party in the system. For providing immediate data sharing service when the owner is offline, the owner needs to utilize the secret sharing approach to split and encrypt the file decryption key and then outsource the encrypted key blocks to multiple cloud service providers (denoted as CSPs hereafter). Except from decryption key, the owner also encrypts and distributes data blocks encoded with the erasure-correcting code technology in multiple CSPs, so that even if there are data losses occurring in some CSPs, the data redundancies can help to successfully recover the complete outsourced data for supporting reliable data sharing service. Meanwhile, an elaborately designed cryptographic protocol guarantees the correctness of each of data transactions (bills); if there are some entities deceiving the data owner for the bill content, this will be detected by the system as a high probability. Besides, we import a semitrusted proxy to provide a secure and efficient keyword-based query function that enables data user to search what he/she needs in outsourced data files before downloading. Our major contributions are shown as follows:

(i) For the first time, we explore the problems of such a practical data sharing scenario: the data owner does not know which data users (even the attributes) will download his/her encrypted files before data outsourcing, and the owner cannot always remain online simultaneously. Besides, there is no TTP in our system.

(ii) We design an elaborately cryptographic protocol that could guarantee the correctness of each of data exchange transactions (bills), so that the deceptive behavior from some semitrusted entities will be detected as a high probability.

(iii) Thorough security analysis and experiments on the real-world dataset show that our proposed scheme is secure and efficient.

The remainder of this paper is organized as follows. In Section 2, we give some necessary notations of our scheme and describe the system model, threat model, and design goals. Section 3 introduces some background knowledge so that readers could understand our later scheme easily, followed by Section 4, in which we propose the details of our scheme. And then we analyze the security and evaluate the performance of our proposed scheme in Section 5. The related

works are discussed in Section 6. Finally, Section 7 covers the conclusion.

2. Problem Formulation

2.1. Notations. We first introduce some necessary notations that will make our later description convenient, shown as follows:

(i) F: the collection of m data files that would be outsourced to multiple cloud service providers, denoted as $F = \{F_1, F_2, \ldots, F_m\}$, where F_i, $i \in \{1, 2, \ldots, m\}$, is the ith file in the collection

(ii) D_i: the encoded results for data file F_i. Assuming that (n, t) Reed-Solomon code is exploited for encoding data in our scheme, D_i can be denoted as the collection of totally n blocks: $D_i = \{D_{i,1}, \ldots, D_{i,t}, \ldots, D_{i,n}\}$, where $D_{i,j}$, $j \in \{1, 2, \ldots, n\}$, represents the jth data block

(iii) C_i: the ciphertext form of D_i: $C_i = \{C_{i,1}, C_{i,2}, \ldots, D_{i,n}\}$

(iv) K: the encryption key collection for data file collection F: $K = \{K_1, K_2, \ldots, K_m\}$, where K_i, $i \in \{1, 2, \ldots, n\}$, denotes the encryption key for F_i

(v) S_i: the encoded results for K_i. Similar to encoding method of D_i, $S_i = \{S_{i,1}, \ldots, S_{i,t}, \ldots, S_{i,n}\}$, where $S_{i,j}$, $j \in \{1, 2, \ldots, n\}$, represents the jth key block

(vi) $S^*_{i,j}$ and $S^{**}_{i,j}$: the ciphertext form of key block $S_{i,j}$ and the reencrypted form of key block $S_{i,j}$, respectively, where $S^*_{i,j}$ is encrypted by owner and $S^{**}_{i,j}$ is reencrypted by CSP.

(vii) B_i: the bill for data file $F_i(C_i)$, which consists of each transaction of F_i. The transaction generally includes the content about who download $F_i(C_i)$ at what price

(viii) W: the dictionary whose elements are extracted distinct keywords from all data files

(ix) Q: the query sent by user, denoted as the collection of multiple query keywords

(x) TD: the trapdoor generated from Q (i.e., encrypted form of Q)

2.2. System Model. There are totally 4 entities involved in our secure keyword search scheme, data owner, multicloud alliance, semitrusted proxy, and data users, as illustrated in Figure 1.

Data owner would like to share his/her valuable data with authorized users through public cloud platform. But he/she does not know who will download which data files before outsourcing and cannot remain online all the time. To ensure that only authorized users could use his/her outsourced data, owners will share a secret value L with all authorized data users in the initialization phase. For achieving the reliability, in the process of data upload, the owner firstly exploits the erasure-correcting code approach (like Reed-Solomon [17]) to split each data file F_i and its corresponding encryption key K_i as multiple block collection (i.e., D_i and S_i); then the owner encrypts D_i and S_i and sends encrypted data/key blocks (i.e., $\{C_{i,j}, S^*_{i,j}\}$) to all cloud service providers, respectively (each CSP will receive one encrypted data block and one encrypted key block). The owner also generates a searchable encrypted index for all files and outsources them to proxy, so that users could search what they are interested in before downloading. After carrying out all above upload tasks, the data owner could be offline.

Proxy is a semitrusted entity that maintains the encrypted index submitted by data owner. For improving the search efficiency, the index is an inverted list that was tailor-made for our scheme. Upon receiving query request from data user, the proxy searches on its local index and sends all identifications of encrypted data/key blocks that were included in searched results (we called these identifications "data request") to the multicloud alliance. Besides, since the proxy knows who download which outsourced data files (and their corresponding decryption key), it is also responsible for updating each bill B_i for each data file $F_i(C_i)$ and sending corresponding bills to the data user.

Multicloud alliance is composed of multiple cloud service providers (CSPs). It is responsible for storing the encrypted data blocks and encrypted key blocks. Upon receiving the data request from proxy, each CSP will firstly update the bills corresponding to encrypted data blocks it stores; then the randomly chosen t CSPs (chosen by user; the details are shown in next paragraph) reencrypt the corresponding encrypted key block with its updated bill and users public key. Finally, they send the encrypted data blocks as well as reencrypted key blocks together to the data user.

Data users would like to send a search request to the semitrusted proxy. The search request includes the following 3 items: the trapdoor TD, search parameter k, and t randomly chosen identifications of CSPs, where parameter k denotes that the user needs the top-k ranked results and CSPs' identifications represent user needs to download data blocks and key blocks from these t CSPs. After receiving corresponding encrypted data blocks and reencrypted key blocks from t CSPs, user could carry out the following three operations to recover the original plaintext form of data files: (1) with the bills sent by proxy and its secret key, user decrypts the reencrypted key blocks as their encrypted form that owner sends to CSPs (i.e., $S^{**}_{i,j} \rightarrow S^*_{i,j}$); (2) with the secret value L given by owner, user decrypts the encrypted key blocks as unencrypted one (i.e., $S^*_{i,j} \rightarrow S_{i,j}$); (3) with all plaintext forms of key blocks, users recover the decryption key K_i (K_i is also regarded as decryption key because we use symmetrical encryption mechanism in the scheme) and then decrypt the encrypted data blocks and recover the plaintext form of data file F_i.

2.3. Threat Model. From the system model we described above, the threat model can be defined in terms of the security settings of cloud service providers and proxy as follows.

Cloud service providers (CSPs) are "mixed-trusted" ones. (1) Most of them are "honest but curious" (i.e., semitrusted), which carry out the designed protocol honestly but are curious about the plaintext form of outsourced data file and

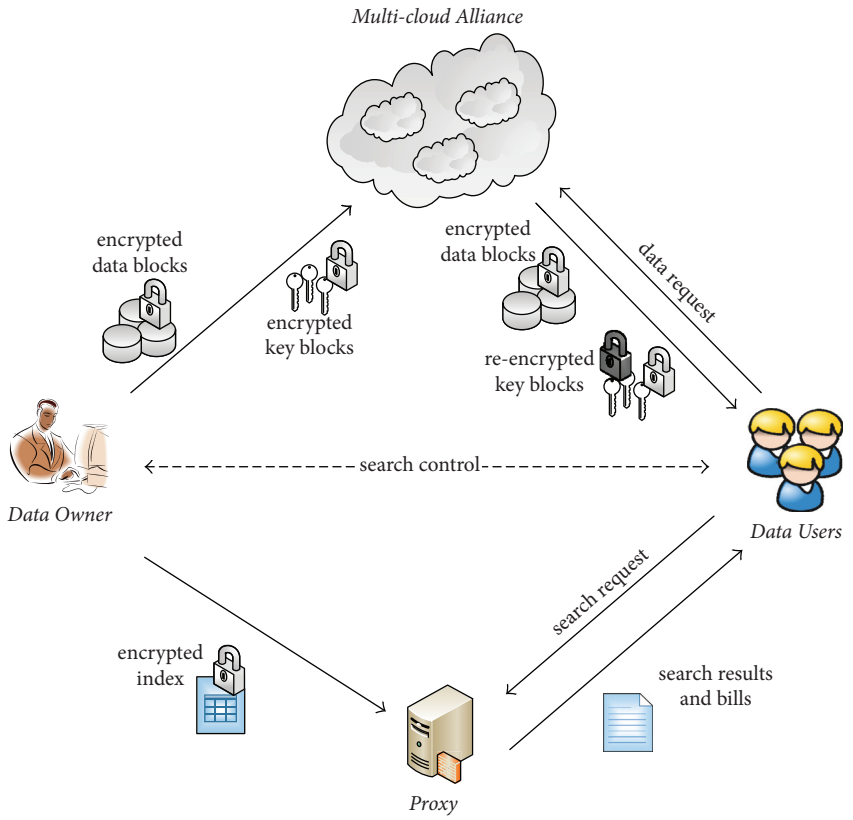


Figure 1: The architecture of our proposed data sharing system.

even collude with the proxy for deceiving the data user in terms of bills for stealing the economic benefits of data owner. (2) A small part of them are completely trusted, which will not collude with other CSPs or the proxy for destroying the data confidentiality or the correctness of bills. Note that the threat model we set for CSPs is based on the reality that there are many companies providing cloud computing service at present, and some of them are big companies that significantly value their reputation.

Proxy is "honest but curious," which executes the designed protocol honestly but is curious about the plaintext form of outsourced index and query keywords sent by user. Besides, the proxy would like to collude with CSPs for forging the bills.

Users are "honest but curious" ones who execute the designed protocol honestly but are curious about the plaintext form of outsourced data that other users downloaded. However, we assume that users do not collude with CSPs for revealing the secret value L or forging the bills, and they also would not reveal L to the proxy.

2.4. Design Goals. Our proposed data sharing scheme should achieve the following 4 design goals.

2.4.1. Security. Under the threat model we defined above, the adversary should not obtain the additional information of data owner and data user except the message it receives in protocol. Besides, the bills that record the data sharing information also should be confirmable ones to protect the economic benefits for data owner. Specifically, there are 3 aspects of security, shown as follows:

(i) *Data confidentiality*: the confidentiality of each of outsourced data blocks should not be revealed by any adversary

(ii) *Index/query privacy*: the plaintext form of outsourced index and encrypted query keyword (trapdoor) should not be revealed by any adversary

(iii) *Confirmable bills*: any adversary cannot deceive data owner successfully in terms of the forging data exchange bills

2.4.2. Reliability. Our scheme can provide reliable data sharing service even if the integrity of data blocks or key blocks of some CSPs is destroyed.

2.4.3. Flexibility. In the premise that the owner does not know who will download which outsourced data files before data outsourcing, the proposed scheme should guarantee achieving a real-time data sharing service even if data owner is offline, that is, data owner in our scheme is flexible enough.

2.4.4. Efficient. The response time of downloading the interested outsourced data files (the details of response time are shown in latter scheme) should be in an acceptable range.

Namely, for the entities who do not possess abundant computational resources (like the owner, user, and proxy), the computational overhead of them should be relatively small.

3. Preliminaries

In this section, we briefly introduce 3 pieces of basic knowledge utilized in our proposed scheme.

3.1. Erasure-Correcting Code. The erasure-correcting code is utilized to tolerate multiple failures in distributed storage systems. Simply, an integrate data file can be encoded as n blocks and only t of them can recover the original data file, where $t \leq n$. In this paper, if we store the encoded results of data file or encryption key (i.e., n data/key blocks) into n different CSPs, respectively, the original data file or encryption key can survive the failure of any $n - t$ of the n CSPs without any data loss. As for encoding method, the widely used Reed-Solomon (RS) erasure-correcting code is utilized in our scheme; taking the (n, t) RS code as an example, we briefly give the process of encoding as follows; more details are shown in literature [17].

For the data file F, we firstly let $F = (F_1, F_2, \ldots, F_t)$, where each data block F_j, $j \in \{1, 2, \ldots, t\}$, can be regarded as an l-length column vector whose elements all belong to $GF(2^p)$. Then we randomly generate a $t \times n$ Vandermonde matrix A as the dispersal matrix; according to a sequence of elementary row transformations, matrix A can be rewritten as the form of $(E \mid P)$, where E is a $t \times t$ identity matrix and P is the secret parity generation matrix with size $t \times (n-t)$, shown as follows:

$$A = (E \mid P) = \begin{pmatrix} 1 & 0 & \cdots & 0 & p_{11} & p_{12} & \cdots & p_{1(n-t)} \\ 0 & 1 & \cdots & 0 & p_{21} & p_{22} & \cdots & p_{2(n-t)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{t1} & p_{t2} & \cdots & p_{t(n-t)} \end{pmatrix}. \quad (1)$$

With the dispersal matrix A, we can encode the original data file F as follows:

$$\begin{aligned} D = F \cdot A &= F_i \cdot (E \mid P) \\ &= (D_1, D_2, \ldots, D_t, D_{t+1}, \ldots, D_n) \\ &= (F_1, F_2, \ldots, F_t, D_{t+1}, \ldots, D_n), \end{aligned} \quad (2)$$

where D are the encoded results composed with n blocks. Since $A = (E \mid P)$ is derived from a Vandermonde matrix, any t out of the n columns form an invertible matrix; that is, a $t \times t$ matrix whose t column vectors are distinct column vectors randomly sampled from D can be used to recover F.

3.2. Discrete Logarithm Assumption. Firstly, the discrete logarithm problem can be described as follows: for $x \in Z_p$, given g and $g^x \in G$, output x, where G is a multiplicative cyclic group in large prime order p and g is the generator of group G.

The discrete logarithm assumption means that it is computationally infeasible to solve the discrete logarithm problem in G.

3.3. Bilinear Maps. Assuming that G_1, G_2, and G_T are three multiplicative cyclic groups of prime order p; g_1 and g_2 are the generators of G_1 and G_2, respectively. Then a bilinear map $\hat{e} : G_1 \times G_2 \to G_T$ satisfies the following properties:

(i) *Bilinear*: for any $u \in G_1$, $v \in G_2$, and two random numbers $a, b \in Z_p$, $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$

(ii) *Nondegenerate*: $\hat{e}(u, v) \neq 1$

(iii) *Computable*: there is a polynomial-time algorithm for computing the map $\hat{e}(u, v)$

4. The Proposed Scheme

In this section, we propose a feasible data sharing scheme that could achieve all the design goals we defined in Section 2.4.

4.1. Setup

4.1.1. Parameters Generation. First of all, the system generates and publishes the following parameters in initialization phase:

(i) G and G_T: two multiplicative cyclic groups of prime order p, where g is the generator of G

(ii) L: one of the secret values shared between owner and users. All users have the same L

(iii) r: the other secret value shared between owner and user. Different users have different r

(iv) $\hat{e}$: a bilinear map $G_1 \times G_1 \to G_T$. Its specific properties are shown in Section 3.3

(v) Z_p^*: a set of nonzero elements in q-order integer group

(vi) $H(\cdot)$: a secure collision-resistant hash function that could map the message with any length into an element of Z_p^*

(vii) $H_b(\cdot)$: a secure collision-resistant hash function that could map the exchange bill with any length into an element of G

(viii) $H_l(\cdot)$: a secure collision-resistant hash function that could map any element in G_T into a fix length (in our scheme, the fix length is 128-bit) binary number

(ix) A: the $t \times n$ dispersal matrix used for encoding data file and encryption key

4.1.2. Symmetric Key Generation. The data owner utilizes the traditional symmetric cryptographic mechanism (like AES algorithm) to encrypt each data block. Obviously, for protecting data confidentiality in "Paid Data Sharing" model, different data files (i.e., data block collection) should be encrypted with different encryption keys. Therefore, the owner generates encryption key K_i for each data file F_i. Besides, the secret values L and r are also generated in this

phase (shown as the "search control" in Figure 1), where L is a isometric binary number as K_i. For example, in terms of AES algorithm, K_i and L are both 128-bit length binary numbers.

4.1.3. Public/Secret Keys Generation. Upon receiving the data request from the proxy, t randomly chosen CSPs will firstly reencrypt the encrypted key blocks and then send the corresponding encrypted data blocks and reencrypted key blocks together to the user. For releasing from the heavy load of maintaining certificates produced by multiple users, we exploit the idea of identity-based encryption (IBE) to encrypt key information for user. Specifically, each authorized user should prepare a pair of public and secret keys, where the hash value of each user's identification (H(uID)) is the public key and the value $r \cdot H$(uID) is each user's secret key (as we mentioned above, different users have different secret values r). Besides, each data user publishes the parameter $P_u = g^r$ in the whole system.

4.2. Build Index. To enable data users to search on encrypted cloud data, the data owner needs to generate a searchable index and submit it to the semitrusted proxy. The proxy maintains the index and obtains the search results for the search request sent by user according to the index. For providing efficient search, we exploit the inverted index structure widely used in many related works [18, 19] to construct index for all outsourced data files, and each term in the inverted index is the encrypted form of corresponding keyword. The specific steps for building such inverted index are shown as follows.

4.2.1. Dictionary Generation. The data owner firstly needs to construct the dictionary that contains all distinct keywords extracted from all outsourcing data files. In this paper, we exploit the method proposed by Cash et al. [20] to extract keyword and then generate the dictionary W, which includes a total of 4 steps: (1) segmenting words, (2) removing stop words, (3) stemming words, and (4) merging keywords. Due to the more detailed steps shown in [20], we omit them here.

4.2.2. Secure Index Construction. According to the dictionary generated above, the plaintext form of inverted index can be built by data owner. Formally, it can be shown as Table 1, where $w_1, w_2, \ldots$ in the table are keywords of dictionary W; that is, the size of index is equal to the length of W. Besides, the document list in the table (i.e., $F_1, F_3, \ldots$) represents the documents containing corresponding keyword, and the identifications of encrypted data block collections and encrypted key block collections (i.e., C_1.ID, C_3.ID, ... and S_1^*.ID, S_3^*.ID, ...) denote the identifications of corresponding encrypted data/key blocks stored in the multicloud alliance, where C_i.ID $= \{C_{i,1}$.ID, $C_{i,2}$.ID, ..., $C_{i,n}$.ID$\}$ (S_i^*.ID can be denoted as the similar form). In practice, the inverted index does not contain the second column "document list" of Table 1; we show it here only for easily understanding the structure of index.

After constructing the plaintext form of inverted index, in order to protect the index privacy, owner should encrypt the keywords in the index before outsourcing them to semitrusted proxy. We design an efficient encryption algorithm based on the assumption that the discrete logarithm problem in large prime order cyclic group is difficult (the detailed explanation is shown in Section 3.2). The specific encryption approach is shown as follows:

$$w^* = \left(g^{H(L\|w)} \bmod p\right)^{-1}, \tag{3}$$

where w^* represents the ciphertext form of keyword w, g and $H(\cdot)$ are public parameters generated in system initialization phase (see Section 4.1.1), L is the secret value shared between owner and all users, and the operator "$\|$" denotes cascade. Note that the encrypted index only replaces the plaintext form of keywords with encrypted keywords, but the block list and the structure of index do not change at all.

Finally, data owner outsources the encrypted inverted index (denoted as I) generated above to the proxy.

4.3. Upload Encrypted Data/Key Blocks. Now we show how the data owner generates and uploads his/her data blocks and key blocks to the multicloud alliance.

4.3.1. Encrypted Data Blocks Generation. For providing the reliable and secure data sharing service, before outsourcing the data file to CSPs, we should encode and encrypt each original data file based on erasure-correcting code approach, so that our system could tolerate multiple failures (i.e., the data integrities of some CSPs in the multicloud alliance are destroyed). In our scheme, we utilized the (n, t) Reed-Solomon erasure-correcting code for data file encoding, where $t < n$ and n is the total number of CSPs in the multicloud alliance (according to the principle of RS code, the parameters t and n should satisfy $t \le n$, but $t = n$ means that user could recover the integrated data file if and only if there is no data loss of all CSPs. Therefore, for tolerating multiple failures in multicloud alliance, we let $t < n$).

Firstly, we generate plaintext form of n data blocks for each data file F_i in the file collection F; the specific calculation steps are the same as (2); we directly give the encoded result D_i for F_i as follows:

$$D_i = F_i \cdot A = \left(D_{i,1}, D_{i,2}, \ldots, D_{i,t}, D_{i,t+1}, \ldots, D_{i,n}\right). \tag{4}$$

Secondly, in order to protect the data confidentiality, data owner should encrypt all data block collections with corresponding encryption key K_i. In this paper, the encryption approach we utilized is the traditional AES algorithm, and the encrypted results of D_i are denoted as C_i:

$$D_i \xrightarrow[K_i]{\text{AES}} C_i = \left\{C_{i,1}, C_{i,2}, \ldots, C_{i,n}\right\}. \tag{5}$$

For each data file that needs to be outsourced, the owner encodes it and encrypts it as (4) and (5) show and then uploads all the m (i.e., the number of data files) encrypted data block collections $\{C_1, C_2, \ldots, C_m\}$ to the multicloud alliance, where each data block collection is composed of n data blocks; they should be randomly dispersed to n CSPs, respectively.

Table 1: The form of inverted index constructed by data owner.

Keyword	Document list	Identifications of encrypted data block collections	Identifications of encrypted key block collections
w_1	$F_1, F_3, \ldots$	$C_1.\mathrm{ID}, C_3.\mathrm{ID}, \ldots$	$S_1^*.\mathrm{ID}, S_3^*.\mathrm{ID}, \ldots$
w_2	$F_2, F_5, \ldots$	$C_2.\mathrm{ID}, C_5.\mathrm{ID}, \ldots$	$S_2^*.\mathrm{ID}, S_5^*.\mathrm{ID}, \ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$

Input: the outsourced data file collection F; the encryption key collection K; the secret binary number L.
Output: the encrypted data/key blocks that should be outsourced to multi-cloud alliance.
(1) **for** each data file F_i in F **do**
(2) the owner encode it with (n, t) - RS code:
$$F_i \xrightarrow{\text{encode}} D_i = \{D_{i,1}, D_{i,2}, \ldots, D_{i,n}\}$$
(3) **for** each data block $D_{i,j}$ in D_i **do**
(4) the owner encrypts it with K_i: $D_{i,j} \xrightarrow[K_i]{\text{encrypt}} C_{i,j}$
(5) **end for**
(6) **end for**
(7) **for** each encryption (decryption) key K_i in K **do**
(8) the owner splits K_i as t isometric binary number:
$$K_i \xrightarrow{\text{split}} \{K_{i,1}, K_{i,2}, \ldots, K_{i,n}\}$$
(9) the owner encode it with (n, t) - RS code:
$$\{K_{i,1}, \ldots, K_{i,n}\} \xrightarrow{\text{encode}} S_i = \{S_{i,1}, \ldots, S_{i,n}\}$$
(10) **for** each key block $S_{i,j}$ in S_i **do**
(11) the owner encrypts it with L:
$$S_{i,j} \xrightarrow[L]{\text{encrypt}} S_{i,j}^* = S_{i,j} \oplus L$$
(12) **end for**
(13) **end for**
(14) **return** $\{C_i, S_i^*\}$, $i \in \{1, 2, \ldots, m\}$

Algorithm 1: *Upload*(F, K, L).

4.3.2. Encrypted Key Blocks Generation. In a similar way, we can also encode the encryption key K_i as n key blocks. However, since we use the AES algorithm for encrypting, K_i is only a 128-bit binary number that is not easily encoded. We will first split all K_i, $i \in \{1, 2, \ldots, m\}$, as t isometric binary number based on the XOR approach. Specifically, we randomly choose $t - 1$ 128-bit length binary vectors, denoted as $\{K_{i,1}, K_{i,2}, \ldots, K_{i,t-1}\}$; then the tth key blocks can be calculated as follows:

$$K_{i,t} = K_{i,1} \oplus K_{i,2} \oplus \cdots \oplus K_{i,t-1} \oplus K_i. \tag{6}$$

Obviously, we can recover K_i if and only if we possess all these t blocks.

Then we utilize the RS code to add $(n - t)$ redundant isometric binary number, shown as follows:

$$\begin{aligned} S_i &= \{K_{i,1}, K_{i,2}, \ldots, K_{i,t-1}, K_{i,t}\} \cdot A \\ &= (S_{i,1}, S_{i,2}, \ldots, S_{i,t}, S_{i,t+1}, \ldots, S_{i,n}), \end{aligned} \tag{7}$$

where S_i is a collection that contains n key blocks and any t blocks of them could be used to recover the original t key blocks we generated in (6).

Finally, the owner encrypts all n key blocks of each S_i as (8) and randomly disperses them to n CSPs, respectively, as the upload process of data blocks:

$$S_{i,t}^* = S_{i,t} \oplus L. \tag{8}$$

The whole process of uploading encrypted data/key blocks is shown as Algorithm 1.

4.4. Search. When the data user would like to search what he/she needs in the outsourced dataset, he/she will first generate trapdoor TD and randomly choose t CSPs to download data. Then users send TD, k, and the t identifications of CSPs together (called search request) to the proxy. Upon receiving the search request, the proxy performs search algorithm on the outsourced inverted index I and obtains the search results. If we call the data files that possess the query keywords "target files," then, according to the structure of index I, the search results are the identifications of encrypted data/key blocks corresponding to the target files. Finally, the proxy informs multicloud alliance to send corresponding data to users based on the search results (called data request).

4.4.1. Trapdoor Generation. Firstly, the data user needs to generate trapdoor TD based on several query keywords. In fact, the trapdoor consists of all encrypted query keywords. The algorithm for encrypting one query keyword w_q can be shown as follows:

$$w_q^* = \left\{ g^{(1+r)\cdot H(L\|w_q)} \bmod p, g^{r\cdot H(L\|w_q)} \bmod p \right\}, \quad (9)$$

where w_q^* is the ciphertext form of query keyword w_q; from the equation above, we see it is a two-tuple, $g, H(\cdot)$ are public parameters the system generates in the initialization phase, L is the secret value shared between owner and all users, and r is the secret value possessed by different owners.

4.4.2. Search Process. With trapdoor TD and index I, the proxy could find which encrypted index keywords in I match the encrypted query keywords in TD; then it could obtain the corresponding encrypted data/key blocks. For ranking the search results, we follow the basic principle that "more match, higher ranking"; that is to say, for one data file (i.e., encrypted data blocks), the more keywords occur in the data file and query simultaneously, the higher score the data file will obtain. For example, the data file $F_i(C_i)$ has the keywords w_1, w_2, and the data file $F_i'(C_i')$ has the keywords w_2, w_3; if the trapdoor is generated from w_1 and w_2, then the search score of $F_i(C_i)$ is 2 but the search score of $F_i'(C_i')$ is 1. According to the search score of each data file (encrypted data blocks collection), the proxy obtains identifications of the top-k results. The algorithm for determining whether a query keyword matches the index keyword is shown as follows.

We assume that the proxy could recognize identification of data user who sends the search request. Taking one query keyword w_q and one index keyword w as an example, if (10) holds, we say that w_q matches w.

$$\left(g^{(1+r)\cdot H(L\|w_q)} \bmod p \right) \cdot w^* = g^{r\cdot H(L\|w_q)} \bmod p, \quad (10)$$

where w_q^* is the encrypted query keyword in the trapdoor TD and w^* is the encrypted index keyword in I.

Correctness Proof. We can prove the correctness of (10) according to (9) and (3) as follows. If $w_q = w$,

$$\begin{aligned} &\left(g^{(1+r)\cdot H(L\|w_q)} \bmod p \right) \cdot w^* \\ &= \left(g^{(1+r)\cdot H(L\|w_q)} \bmod p \right) \cdot \left(g^{H(L\|w_q)} \bmod p \right)^{-1} \\ &= g^{r\cdot H(L\|w_q)} \bmod p. \end{aligned} \quad (11)$$

After obtaining search results, the proxy sends data request to t CSPs that are randomly chosen by user. And it also sends the data exchange transaction to all n CSPs, so that all CSPs in multicloud alliance could update their self-maintained bills. Finally, the proxy updates the bills it maintained (i.e., adding the exchange transaction to corresponding bills) and sends each bill B_i that has been updated to the user. Note that these bills will be used for decrypting encryption key in latter download process (see Section 4.5).

4.5. Download Encrypted Data/Key Blocks. For the t randomly chosen CSPs, they will firstly update the bills corresponding to data files (i.e., encrypted data blocks) that need to be downloaded and then reencrypt corresponding encrypted key blocks with updated bills and the public key of users. Finally, they send the encrypted data blocks with reencrypted key blocks together to the user. For example, assuming that the encrypted data block collection C_i is what user needs, for one of the t CSPs that stores one of the encrypted data blocks $C_{i,j}$ and encrypted key blocks $S_{i,j}$, it firstly encrypts $K_{i,j}$ with the bill B_i' (For distinguishing the bills proxy sends to user and the bills CSPs maintain, we denote the bill maintained by one CSP that stored $C_{i,j}$ as B_i' and denote the bill proxy sends to user as B_i) and sends the two-tuples $\{C_{i,j}, S_{i,j}^{**}\}$ to the user, where $S_{i,j}^{**}$ is the encrypted form of key block $S_{i,j}$. We follow the idea of identity-based encryption (IBE) to design the encryption/decryption algorithm for encrypted key blocks, shown as follows.

4.5.1. Key Blocks Reencryption. For one of the t CSPs that store $C_{i,j}$ and $S_{i,j}^*$, it randomly chooses $q \in Z_p^*$; then the CSP calculates 3 variates $\alpha_{i,j}$, $\mu_{i,j}$, and $\upsilon_{i,j}$ as shown in the following equations:

$$\begin{aligned} \alpha_{i,j} &= \widehat{e}\left(H_b \left(B_i' \right)^{H(\mathrm{uID})}, P_u \right), \\ \mu_{i,j} &= g^q, \\ \upsilon_{i,j} &= S_{i,j}^* \oplus H_l \left(\alpha_{i,j}^q \right), \end{aligned} \quad (12)$$

where the six parameters $\{\widehat{e}, H_b(\cdot), H_l(\cdot), H(\cdot), g, P_u\}$ are all predefined in the *Setup* phase of our scheme.

Note that the reencrypted form of key block $S_{i,j}$ (i.e., $S_{i,j}^{**}$) is denoted as the two-tuples $\{\mu_{i,j}, \upsilon_{i,j}\}$ and each of t CSPs will send it together with encrypted data block $C_{i,j}$ to the user. Due to different identifications, different users would receive different form of $S_{i,j}^{**}$.

4.5.2. Key Blocks Decryption. Upon user receiving all t encrypted key blocks (i.e., $\{\mu_{i,j}, \upsilon_{i,j}\}$) from t CSPs (here, we assume that user could identify the identifications of all t CSPs who send data), he/she could decrypt them with his/her secret key, $r \cdot H(\mathrm{uID})$, shown as follows:

$$S_{i,j}^* = \upsilon_{i,j} \oplus H_l \left(\widehat{e}\left(H_b \left(B_i \right)^{r\cdot H(\mathrm{uID})}, \mu_{i,j} \right) \right), \quad (13)$$

where $\upsilon_{i,j}$, $\mu_{i,j}$ are included in the encrypted form of $S_{i,j}$, r is the secret value shared between owner and the user, B_i is the bill for C_i sent by the proxy, and $H_l(\cdot)$, $H_b(\cdot)$ are public parameters we prepared in our *Setup* phase. Finally, user could decrypt $S_{i,j}^*$ with L.

$$S_{i,j} = S_{i,j}^* \oplus L. \quad (14)$$

4.5.3. File Decryption. For obtaining plaintext form of original data file F_i, after decrypting all t encrypted key blocks, the data user could obtain the original unencrypted encryption

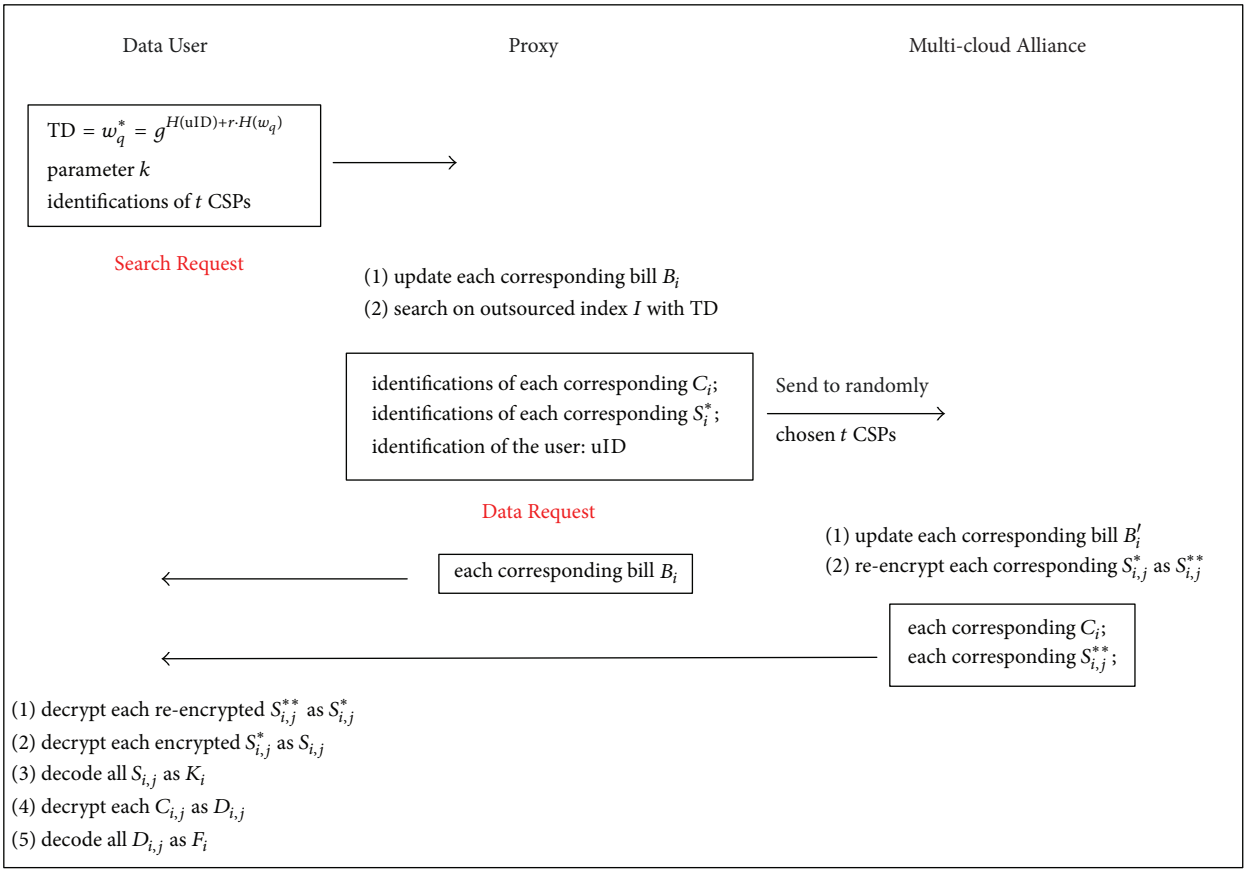


Figure 2: The processes of search and download encrypted data/key blocks.

key K_i by decoding t unencrypted key blocks with dispersal matrix A. Then user decrypts each of encrypted data blocks in C_i and decode them in the same way. Now user could obtain the plaintext form of original data file F_i. The detailed processes of searching and downloading encrypted data/key blocks are summarized in Figure 2.

4.6. Correctness Analysis. In this section, we will show why data user could correctly decrypt the encrypted key blocks from each CSP if and only if the two bills of one corresponding data file (i.e., the bill from proxy and the bill form CSP) are exactly equal to each other. Without loss of generality, taking the key block $S_{i,j}$ as an example, according to the encryption given by (12) and the decryption process shown in (13) and (14), we can prove that $S_{i,j}$ could be correctly decrypted when $B_i = B_i'$, as shown as follows.

Firstly, the reencrypted form of $S_{i,j}(S_{i,j}^{**})$ is shown as follows:

$$\begin{aligned} S_{i,j}^{**} &= \left\{\mu_{i,j}, \upsilon_{i,j}\right\} \\ &= \left\{g^q, S_{i,j}^* \oplus H_l\left(\widehat{e}\left(H_b\left(B_i'\right)^{H(\text{uID})}, P_u\right)^r\right)\right\} \\ &= \left\{g^q, S_{i,j} \oplus L \oplus H_l\left(\widehat{e}\left(H_b\left(B_i'\right)^{H(\text{uID})}, P_u\right)^r\right)\right\}. \end{aligned} \tag{15}$$

Secondly, if and only if $B_i = B_i'$, we can correctly decrypt $S_{i,j}^*$, as shown as follows:

$$\begin{aligned} &\upsilon_{i,j} \oplus H_l\left(\widehat{e}\left(H_b\left(B_i\right)^{r\cdot H(\text{uID})}, \mu_{i,j}\right)\right) \\ &\quad = S_{i,j}^* \oplus H_l\left(\alpha_{i,j}^q\right) \oplus H_l\left(\widehat{e}\left(H_b\left(B_i\right)^{r\cdot H(\text{uID})}, \mu_i\right)\right) \\ &\quad = S_{i,j}^* \oplus H_l\left(\widehat{e}\left(H_b\left(B_i'\right)^{H(\text{uID})}, P_u\right)^q\right) \\ &\qquad \oplus H_l\left(\widehat{e}\left(H_b\left(B_i\right)^{r\cdot H(\text{uID})}, g^q\right)\right) \\ &\quad = S_{i,j}^* \oplus H_l\left(\widehat{e}\left(H_b\left(B_i'\right)^{H(\text{uID})}, P_u\right)^q\right) \\ &\qquad \oplus H_l\left(\widehat{e}\left(H_b\left(B_i\right)^{H(\text{uID})}, P_u\right)^q\right) = S_{i,j}^*. \end{aligned} \tag{16}$$

Obviously, if we successfully decrypt $S_{i,j}^*$, we can correctly decrypt $S_{i,j}^*$ as secret binary number L; we omit the correctness proof here.

4.7. Summary of Scheme. Now we summarize our proposed scheme in terms of reliability and flexibility. The detailed analyses about security and efficiency are shown in Section 5.

4.7.1. Reliability. The (n, t) Reed-Solomon (RS) erasure-correcting code guarantees the reliability of our system. If

there are some errors, such as data loss (key blocks or data blocks) occurring in some CSPs, the properties of erasure-correcting code could help the system to recover the complete data in time. In practice, when the owner deploys cloud services, some mature technology like virtual machines (VM) replication also could be used to enhance the data reliability, and the efficient issues like huge network resource consumption in failure recovery process have been deeply researched [21] to improve the quality of data service. That is to say, many technological means could guarantee the reliability of our system.

4.7.2. Flexibility. The owner outsources the (1) encrypted data blocks, (2) encrypted key blocks, and (3) encrypted index to the multicloud alliance and proxy, respectively. Therefore, data user could search what he/she needs in outsourced cloud data and download them as well as corresponding encrypted key blocks and finally decrypt the key and data. The whole data sharing process does not need the owner to remain online after finishing the upload process. It obviously brings significant flexibility to data owner.

5. Security and Performance Analysis

5.1. Security Analysis. We analyze the security of our system about the 3 aspects we mentioned in Section 2.4.1: data confidentiality, index and query privacy, and confirmable bills.

5.1.1. Data Confidentiality. The AES encryption algorithm is utilized to encrypt the outsourced data in our scheme. In general, we consider that the AES algorithm with random 128-bit key is secure; that is, the only way to destroy the data confidentiality is to deduce the 128-bit encryption (decryption) key. Except from completely guessing the key, there are two other kinds of attacks that adversary can carry out based on the threat model we defined in Section 2.3. Assuming that the adversary wants to deduce one of the decryption keys K_i for the encrypted data block collection C_i, the specific attacks can be shown as follows:

(i) *Attack-1*: after receiving all encrypted data blocks and encrypted key blocks, there are more than t CSPs colluding together for deducing one of the encryption keys K_i; that is, the adversary can totally obtain more than t encrypted data/key blocks

(ii) *Attack-2*: when the t CSPs send t encrypted data blocks $(\{\ldots, C_{i,j}, \ldots\})$ and t reencrypted key blocks $(\{\ldots, S^{**}_{i,j}, \ldots\})$ to one user, other users would like to eavesdrop the sending information to deduce K_i and further obtain the plaintext form of data file F_i

As for *Attack-1*, the secret binary number L is used to protect each encrypted key block $S^{*}_{i,j}$. According to the encryption process shown in (8), since the users in our system would not collude with any CSP, the adversary can only deduce the 128-bit L based on completely guessing way. Such difficulty is equal to guessing AES key; that is, our system is secure in this case.

As for *Attack-2*, all users have the same secret value L which can successfully decrypt the encrypted key blocks (e.g., $S^{*}_{i,j} \rightarrow S_{i,j}$), but the "curious" ones have to decrypt the reencrypted key blocks firstly (e.g., $S^{**}_{i,j} \rightarrow S^{*}_{i,j}$). In our scheme, the secure identity-based encryption algorithm proposed by Boneh and Franklin [22] is used to protect the confidentiality of reencrypted key blocks (i.e., $S^{**}_{i,j}$). Recalling the reencryption process we proposed in (12), the only difference compared with classical IBE algorithm in [22] is that we import the hash value of bill B_i into encryption, but it does not influence the security of encryption algorithm. The detailed security analysis of IBE is shown in literature [22], so we omit it here.

The above two kinds of attacks are the two strongest attacks under the threat model we defined; that is, if our system can successfully resist the above two attacks, other weaker attacks also cannot work well. In summary, the data confidentiality can be protected well in our scheme.

5.1.2. Index/Query Privacy. As for index privacy, since we only encrypt the keywords in the index, so if we can prove the security of index keywords encryption algorithm, we say that the index privacy is well protected. Recall the form of encrypted keywords in index:

$$w^{*} = \left(g^{H(L\|w)} \bmod p\right)^{-1}. \tag{17}$$

Firstly, the discrete logarithm problem in Z_p with large prime p prevents the proxy from knowing the hash value $H(L \| w)$, even if the proxy could guess the encrypted keyword by trying all keywords in dictionary, but under the security assumption, in our threat model, users would not collude with proxy or any other adversary, which means that the proxy cannot obtain the secret value L. Therefore the proxy cannot deduce any plaintext information about encrypted keywords in index.

As for query privacy, before we prove the security of query keywords encryption, we first define a traditional challenger-adversary game. The challenger firstly gives a public key pk to the adversary and then the adversary randomly chooses two query keywords w_0 and w_1 and sends them to the challenger. Upon receiving w_0 and w_1, the challenger randomly chooses w_s ($s = 0$ or $s = 1$) with equal probability 1/2 and then sends $w^{*}_{s} = \mathrm{Enc}(w_s, pk)$ to the adversary. The adversary tries to guess the value of s and outputs its guess s'. The adversary's advantage of this game is the probability $\Pr[s = s'] - 1/2$. Generally, the encryption algorithm is regarded as semantically secure against a chosen plaintext attack (CPA) if the adversary's advantage is negligible.

Theorem 1. *If DDH (Decisional Diffie-Hellman) problem is hard, our keywords encryption is semantically secure.*

Proof. We assume that there is a polynomial time algorithm $\mathscr{A}$ that has a nonnegligible advantage ε as the adversary in the game described above. Then we define a DDH adversary $\mathscr{B}$ that could access $\mathscr{A}$ and achieves a nonnegligible advantage. $\mathscr{B}$ is given the tuple $(g, g^{r}, g^{H(L\|w)}, T)$ as input. Now, $\mathscr{B}$ tries

to guess whether T is equal to $g^{r \cdot H(L\|w)}$ or T is a random value; we denote $\gamma = 0$ as $T = g^{r \cdot H(w)}$ and $\gamma = 1$ as T is random.

Firstly, $\mathcal{B}$ gives the tuple (g, g^r) to $\mathcal{A}$ as input and then $\mathcal{A}$ randomly chooses keywords w_0 and w_1 and sends them to $\mathcal{B}$.

Secondly, $\mathcal{B}$ sets a bit s randomly and sends ciphertext $C_1 = g^r$, $C_2 = g^{H(L\|w_s) \cdot T}$ to $\mathcal{A}$.

Thirdly, $\mathcal{A}$ sends $\mathcal{B}$ a bit s', which is its guess for s. $\mathcal{B}$ guesses that $\gamma = 0$ if and only if $s' = s$. □

Now we analyze the two situations of γ:

(i) If $\gamma = 0$ (i.e., $T = g^{r \cdot H(L\|w_s)}$), then $C_2 = g^{H(L\|w_s)} \cdot T$ is a valid encryption for keywords encryption. In this case, $\mathcal{A}$ will guess correctly with probability $1/2 + \varepsilon$.

(ii) If $\gamma = 1$ (i.e., T is random), the keywords encryption adversary $\mathcal{A}$ receives the value $g^{H(L\|w_s)} \cdot T^{H(L\|w_s)}$, where T is random value. In this case, s is hidden to $\mathcal{A}$, so the probability that $\mathcal{A}$ will guess it is simply 1/2.

Therefore, the probability that $\mathcal{B}$ could successfully solve the DDH problem is shown as follows:

$$\Pr = \frac{1}{2}\left(\frac{1}{2} + \varepsilon\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon}{2}. \tag{18}$$

Since ε is nonnegligible, this demonstrates that $\mathcal{B}$ violates the assumption that DDH is hard.

5.1.3. Confirmable Bills. The method that regards the exchange bill as a part of encryption key for reencrypting key blocks protects the bills from being forged. From the analysis we mentioned in Section 4.6, we find that the user could not decrypt the reencryption key blocks successfully with a high probability under the threat model we defined in Section 2.3. Specifically, in our scheme, user randomly chooses t CSPs from the whole multicloud alliance for retrieving encryption key K_i. According to the security assumption we defined above, a small part of CSPs can be completely trusted; that is to say, if we choose proper parameters for n and t, it is highly possible that there is at least one trusted CSP in the randomly chosen t CSPs. As we say in Section 4.5.2, the user could identify the identifications of all t CSPs that send data, and user would not collude with proxy; therefore these t CSPs are randomly chosen by user; there does not exist the situation where proxy chooses t compromised CSPs that are not wholly chosen by user. Hence, even if the proxy colludes with some CSPs to forge bill B_i for C_i, there is at least one reencrypted key block that cannot be correctly decrypted by data user (see (16)), not to mention to obtain the correct unencrypted data file F_i. Of course, once the user fails to decrypt the data file, he/she will report an error to the system, which could help the owner detect whether the proxy or some CSPs forge the bills immediately.

From the above analysis, the probability that all randomly chosen CSPs are semitrusted (i.e., adversary could successfully deceive the owner in terms of forging the one of the bills) can be calculated as follows:

$$\Pr = \frac{C_{n \cdot (1-P_T)}^t}{C_n^t}, \tag{19}$$

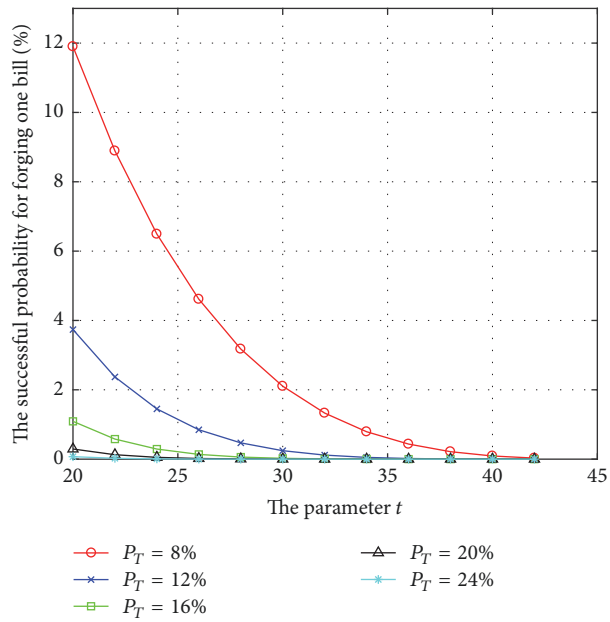


FIGURE 3: The successful probability for forging one bill.

where P_T denotes the proportion of trusted CSPs in whole multicloud alliance, n is the total number of CSPs in the whole multicloud alliance, and t is the encoding parameter that owner chooses. We assume that there are a total of 50 CSPs in the multicloud alliance and then analyze the successful probability for forging one bill based on different P_T and parameter t as Figure 3 shows.

From Figure 3, we can see that larger t and P_T will lead to lower probability that adversary can successfully deceive the owner for one forged bill. When $t \geq 28$ and $P_T \geq 16\%$, the probability that owner suffers such collusive attack is negligible. However, we have to say that even if the owner chooses proper t and the multicloud alliance is relatively secure (i.e., higher P_T), there still exists the situation where the randomly chosen t CSPs are all semitrusted. In this case, all the key blocks would be correctly decrypted. In fact, there is a tradeoff between the high reliability and high security (the "security" here particularly means the correctness of exchange bills) of our data sharing scheme. On one hand, an extreme situation is to choose $n = t$ as the encoding parameter; under our threat model that a small part of CSPs is wholly trusted, the attack above will never be carried out successfully. But if there are some errors like data losses in some CSPs, the system could not give the complete original data file to the user. On the other hand, if t is far less than n, the probability that proxy successfully colludes with the semitrusted CSPs will be high. In a word, the contradiction between reliability and security is still not fully solved in this paper. We will continue to research this problem in the future.

5.2. Performance Analysis. We implemented the proposed scheme using by C language in Windows 10 operation system and tested its efficiency in real-world data files: the Request for Comments (RFC) [23]. Since the upload process executed

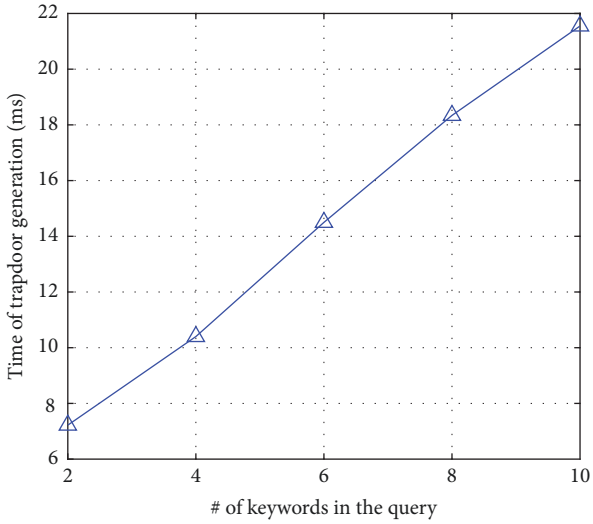


Figure 4: The time of trapdoor generation.

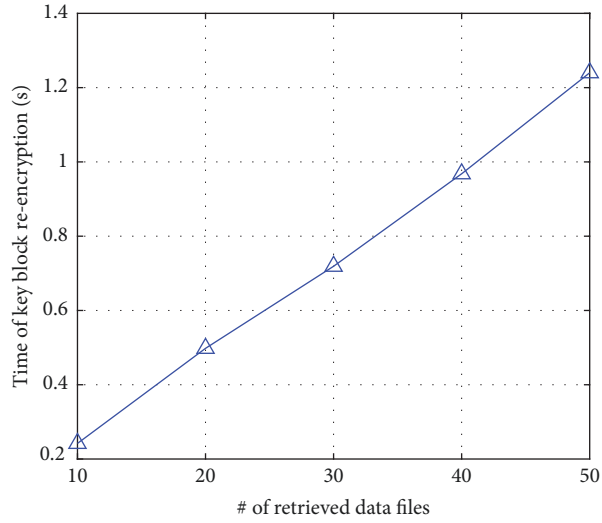


Figure 5: The time of key block reencryption.

by owner can be regarded as the precomputation that will be executed only once, we only focus on the time cost of users and the communication cost of the proxy. Specifically, it includes (1) the time of trapdoor generation, (2) the time of search, (3) the time of key blocks reencryption, (4) the time of reencrypted key blocks decryption, and (5) the traffic analysis of proxy. Of course, the energy consumption is also an important indicator to evaluate the performance of such data-intensive cloud service [24], but due to the restricted experimental conditions, we would not discuss it in this paper. Our experiments are conducted on a system with an Intel(R) Core (TM) i5-3470 processor running at 3.74 GHz, 16.00 GB of RAM, and a 7200 RPM Western Digital 1 TB Serial ATA drive. All results represented the mean of 20 trials.

5.2.1. Time of Trapdoor Generation. According to (9), the encryption algorithm for generating trapdoor for each query keyword totally includes two modular exponentiation operations; that is, the time complexity of trapdoor generation in our schemes is $O(N^2)$, where N denotes the size of dictionary. Figure 4 shows that the time of trapdoor generation increases linearly with N increasing.

5.2.2. Time of Search. The proxy should traverse the inverted index I and rank the data files according to the number of query keywords in the file. Figure 6(a) shows that, with the same number of query keywords and data files, the search time increases almost linearly with the size of dictionary increasing. Figure 6(b) shows that, with the same number of data files and the same size of dictionary, the search time increases almost linearly with the number of query keywords increasing. But the search time cost is not significantly affected by the number of data files due to the structure of our inverted index; specifically, after locating query keywords, all candidates are filtered out for ranking, while the rank operation will not cost too much time, as shown in Figure 6(c).

5.2.3. Time of Key Blocks Reencryption. Each of the t randomly chosen CSPs will reencrypt one encrypted key block for one data file as given in (12). Since the t CSPs reencrypt encrypted key blocks simultaneously, the waiting time for user is only related to the number of data files that will be downloaded by data user (top-k ranked results). That is, the time complexity of key blocks reencryption is $O(k)$. As shown in Figure 5, with k increasing, the time of reencryption increases almost linearly.

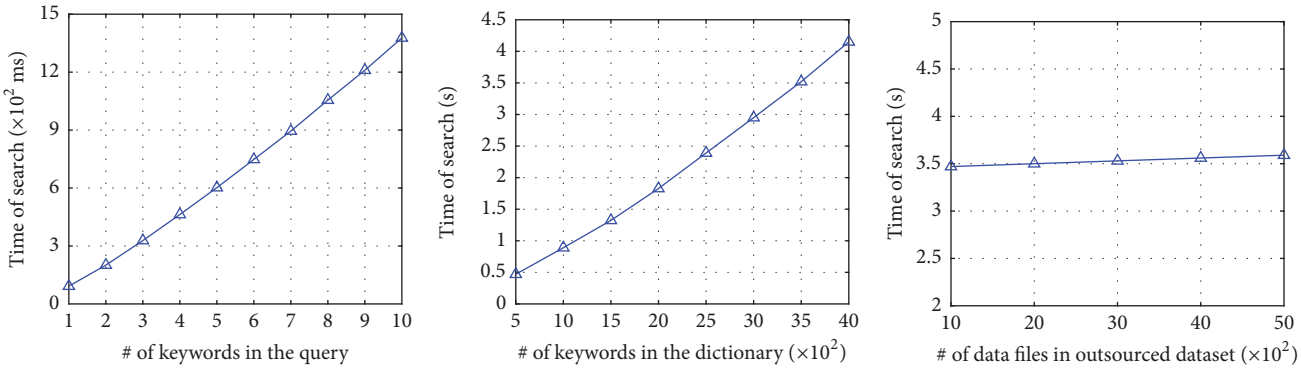
5.2.4. Time of Reencrypted Key Blocks Decryption. Upon receiving all reencrypted key blocks corresponding to retrieved k data files (i.e., encrypted data block collections), the user needs to decrypt them as given in (13). Obviously, the user totally decrypts $t \cdot k$ reencrypted key blocks. Figure 7 shows that, with the increasing of parameter t or k, the time of decryption increases almost linearly.

5.2.5. Traffic Analysis of Proxy. Since proxy plays as "Interaction Center" in our system, it is necessary to analyze its traffic. The whole communication process of proxy can be divided into two phases. Specifically, the first one is "upload"; the proxy only receives the encrypted index from data owner; the second one is "download"; the proxy receives search request from data user, sends data request to multicloud alliance, and sends all corresponding bills to data user.

According to the above two phases, the complexity of communication of proxy can be shown in Table 2. We assume that the encrypted keyword is 128 bytes, the identification of encrypted data/key block or CSP is about 128 bytes, and the size of a bill is about 512 bytes. Then we set the number of documents $m = 5000$, the number of CSPs in whole multicloud alliance (i.e., the encode parameter n) is 50, the encode parameter $t = 35$, the size of dictionary $|W| = 12000$, the number of keywords in query $|Q| = 5$, and the search parameter $k = 50$. Based on such parameter settings, we can obtain that the whole communication cost of proxy in upload

Table 2: The traffic analysis of proxy.

	Data owner	Data user	Multicloud
Upload	$O(\|W\| + m + 2mn)$	0	0
Download	0	$O(t + \|Q\| + k)$	$O(2kt)$

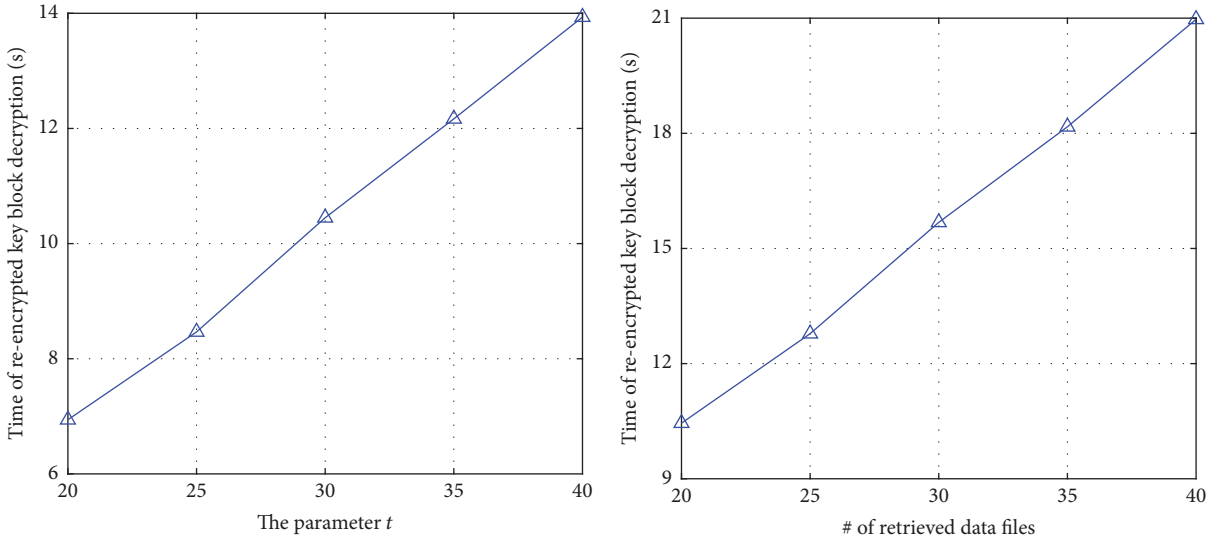


(a) For different number of query keywords with the same size of dictionary, $|W| = 4000$, and for the same number of query data files, $n = 2000$

(b) For different size of dictionary with the same number of query keywords, $|Q| = 5$, and for the same number of query data files, $n = 2000$

(c) For different number of data files with the same size of dictionary, $|W| = 4000$, and for the same number of query keywords, $|Q| = 5$

Figure 6: The time of search.



(a) For different parameter t with the same number of retrieved data files, $k = 20$

(b) For different number of retrieved data files with the same parameter t, $t = 30$

Figure 7: The time of reencrypted key blocks decryption.

phase is about 63 M, and in download phase it is about 0.45 M. Although the communication cost in upload is a little big, it is one-time process for uploading index. In a word, even the communication resource-constrained device could play as proxy.

6. Related Work

6.1. Secure Data Sharing with IBE and PRE. The concept of identity-based encryption (IBE) was first introduced by Shamir [25]; then Boneh and Franklin [22] proposed the first secure and practical IBE scheme with the computable bilinear

map. Then Canetti et al. [26] designed the first construction for IBE which was provably secure without the random oracle model. Moving a step forward, in 2004, Boneh and Boyen [27] proposed two efficiency improved IBE schemes under the "selective-ID" model proposed in [26]. The IBE approach provides an efficient cryptographic system where the public key can be any arbitrary string and the secret key is extracted from a trusted party called a private key generator (PKG). Motivated by its significant advantage that owner could release from heavy load of managing certificates, Kackeri [9] utilized IBE in the secure data distribution system that only users with authorized identifications could access the encrypted data.

Another popular secure data sharing method is proxy-based reencryption (PRE), which was first proposed by Blaze et al. [28]. With a proxy server, the PRE primitive can transfer a ciphertext designated for one user to another ciphertext designated for another user without knowing the plaintext content of data. After that, many PRE schemes have been proposed for the improved security or efficiency [29–31]. In 2006, Ateniese et al. [32] then improved the concept of PRE and employed it for remote data storage. In their scheme, owner encrypts his/her files and outsources them to proxy. The proxy can transfer the owner's ciphertext to the ciphertext of the requester if and only if he/she has obtained a reencryption key from the owner. Then, combining technologies of IBE and PRE, Green and Ateniese [33] introduced the concept of identity-based proxy reencryption (IBPRE) scheme and then the IBPRE scheme is further developed in terms of security and functionality enhancement [34, 35]. The proxy in IBPRE schemes also plays a similar role to traditional PRE, but the transfer key is related to the identities of owners and users.

However, when we utilize the method of IBE or PRE for data sharing system, it is necessary for data owner to remain online all the time (such as generating reencryption key for user in PRE schemes) or know the specific information about the data user (such as encrypting data according to the user's identification in IBE schemes).

6.2. Secure Data Sharing with ABE. To address the mentioned problems of IBE and PRE, Goyal et al. [36] firstly proposed Key-Policy Attribute-Based Encryption, where the ciphertexts are labeled with sets of attributes and private keys are associated with access structures that control which ciphertexts a user is able to decrypt. Then, a number of works used ABE to realize fine-grained access control for outsourced data [12, 13, 37]. There is no doubt that the ABE schemes improve the scalability of data sharing system, so that the data owner could encrypt outsourced data for multiple users with one encryption key. Besides, ABE schemes also provide great flexibilities since only attributes of the user set instead of specific identity (or public key) need to be known by owner. But there are still some limits for directly using ABE scheme in our "Paid Data Sharing" model: one is that may be the owner even does not know the user's attributes for one data file; another is that the authority in ABE system suffers from the key escrow problem [12, 16], since the authority can access all the encrypted files.

Besides, Hiremath and Annapurna [38] discussed the framework and security issues at collaboration in multicloud computing environments; inspired by the "proxy as service" model they proposed, we design a distributed data/key storage system architecture in this paper.

7. Conclusion

In this paper, for the first time, we explore the problems in a practical secure data sharing scenario. Specifically, the data owner does not know which data users (even their attributes) will download and decrypt which outsourced data files of him/her, while the owner cannot always remain online and there is no trusted third party in the system. We utilize the secret sharing approach to split and encrypt the decryption key and then outsource it to multiple cloud service providers (CSPs). Such distributed key management not only protects the data confidentiality but also guarantees that if there are not enough CSPs collude with each other, the adversaries cannot successfully forge the bills for data transaction. In order to improve the reliability of such data service, we exploit the erasure-correcting code for distributing file to multiple CSPs. Even if there are data losses occurring in some CSPs, the data redundancies that erasure-correcting code produced can help to recover the complete outsourced data immediately. A semitrusted proxy that offered a secure and efficient keyword-based query function enables the data user to search what he/she needs in outsourced data files before downloading. Experiments on real-world dataset demonstrate the time efficiency of our proposed scheme.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] S. Lian and Y. Duan, "Smoothing of the lower-order exact penalty function for inequality constrained optimization," *Journal of Inequalities and Applications*, Paper No. 185, 12 pages, 2016.

[2] J. Zhang, B. Qu, and N. Xiu, "Some projection-like methods for the generalized Nash equilibria," *Computational optimization and applications*, vol. 45, no. 1, pp. 89–109, 2010.

[3] M. Wang, L. Song, and G.-l. Tian, "SCAD-penalized least absolute deviation regression in high-dimensional models," *Communications in Statistics—Theory and Methods*, vol. 44, no. 12, pp. 2452–2472, 2015.

[4] P. Wang and L. Zhao, "Some geometrical properties of convex level sets of minimal graph on 2-dimensional Riemannian manifolds," *Nonlinear Analysis*, vol. 130, pp. 1–17, 2016.

[5] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[6] Z. Kong, S. Yang, F. Wu, S. Peng, L. Zhong, and L. Hanzo, "Iterative Distributed Minimum Total MSE Approach for Secure Communications in MIMO Interference Channels," *IEEE*

Transactions on Information Forensics and Security, vol. 11, no. 3, pp. 594–608, 2016.

[7] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security*, vol. 6054 of *Lecture Notes in Computer Science*, pp. 136–149, Springer, Berlin, Germany, 2010.

[8] J. Han, W. Susilo, and Y. Mu, "Identity-based data storage in cloud computing," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 673–681, 2013.

[9] R. R. Kackeri, *Identity-based encryption system for secure data distribution*, U.S. Patent No. 7,003,117, 2006.

[10] L. Xu, X. Wu, and X. Zhang, "CL-PRE: A certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, (ASIACCS '12)*, pp. 87-88, Republic of Korea, May 2012.

[11] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, vol. 258, pp. 355–370, 2014.

[12] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communication Security, (ASIACCS '10)*, pp. 261–270, April 2010.

[13] M. Yang, F. Liu, J.-L. Han, and Z.-L. Wang, "An efficient attribute based encryption scheme with revocation for outsourced data sharing control," in *Proceedings of the 1st International Conference on Instrumentation and Measurement, Computer, Communication and Control, IMCCC '11*, pp. 516–520, China, October 2011.

[14] 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.

[15] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[16] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131–143, 2013.

[17] J. S. Plank and Y. Ding, "Note: Correction to the 1997 tutorial on Reed-Solomon coding," Tech. Rep. CS-03-504, University of Tennessee, 2003.

[18] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '10)*, pp. 441–445, IEEE, San Diego, Calif, USA, March 2010.

[19] W. Zhang, S. Xiao, Y. Lin, T. Zhou, and S. Zhou, "Secure ranked multi-keyword search for multiple data owners in cloud computing," in *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN '14*, pp. 276–286, USA, June 2014.

[20] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15*, pp. 668–679, USA, October 2015.

[21] A. Zhou, S. Wang, B. Cheng et al., "Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization," *IEEE Transactions on Services Computing*, vol. 10, no. 6, pp. 902–913, 2017.

[22] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in cryptology—CRYPTO*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229, Springer, Berlin, 2001.

[23] (2016). Request for comments. [Online]. Available: http://www.rfc-editor.org/index.html.

[24] S. Wang, A. Zhou, C.-H. Hsu, X. Xiao, and F. Yang, "Provision of Data-Intensive Services Through Energy-and QoS-Aware Virtual Machine Placement in National Cloud Data Centers," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 290–300, 2016.

[25] A. Shamir, "Identity-based cryptosystems and signature schemes," *Workshop on the Theory and Application of Cryptographic Techniques—CRYPTO*, vol. 84, 1984.

[26] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," in *Advances in cryptology—EUROCRYPT*, vol. 2656 of *Lecture Notes in Computer Science*, pp. 255–271, Springer, Berlin, 2003.

[27] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Advances in cryptology—EUROCRYPT*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 223–238, Springer, Berlin, 2004.

[28] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in cryptology—EUROCRYPT'98 (Espoo)*, vol. 1403 of *Lecture Notes in Computer Science*, pp. 127–144, Springer, Berlin, 1998.

[29] T. Matsuda, R. Nishimaki, and K. Tanaka, "CCA proxy re-encryption without bilinear maps in the standard model," in *Public Key Cryptography—PKC*, vol. 6056 of *Lecture Notes in Comput. Sci.*, pp. 261–278, Springer, Berlin, 2010.

[30] S. S. Chow, J. Weng, Y. Yang, and R. H. Deng, "Efficient unidirectional proxy re-encryption," in *International Conference on Cryptology in Africa*, vol. 6055 of *Lecture Notes in Comput. Sci.*, pp. 316–332, Springer, Berlin, 2010.

[31] J. Weng, Y. Zhao, and G. Hanaoka, "On the security of a bidirectional proxy re-encryption scheme from PKC 2010," in *Public Key Cryptography—PKC*, vol. 6571 of *Lecture Notes in Comput. Sci.*, pp. 284–295, Springer, Heidelberg, 2011.

[32] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pp. 1–30, 2006.

[33] M. Green and G. Ateniese, "Identity-Based Proxy Re-encryption," in *International Conference on Applied Cryptography and Network Security*, Lecture Notes in Computer Science, pp. 288–306, Springer-Verlag, 2007.

[34] C. K. Chu and W. G. Tzeng, "Identity-based proxy re-encryption without random oracles," in *International Conference on Information Security*, pp. 189–202, Springer-Verlag, 2007.

[35] L. Wang, L. Wang, M. Mambo, and E. Okamoto, "Identity-Based Proxy Cryptosystems with Revocability and Hierarchical Confidentialities," in *International Conference on Information and Communications Security*, vol. 6476 of *Lecture Notes in*

Computer Science, pp. 383–400, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[36] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 89–98, November 2006.

[37] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, May 2007.

[38] M. M. Hiremath and P. P. Annapurna, "Collaboration in multi-cloud computing environments: Framework and security issues," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 3, pp. 2859–2862, 2015.