Data Sharing with Fine-Grained Access Control for Multi-tenancy Cloud Storage System

Zhen Li[1,2(✉)], Minghao Zhao[1], Han Jiang[1], and Qiuliang Xu[1]

[1] School of Computer Science and Technology, Shandong University, Jinan, China
sdufelizhen@l26.com, zhaominghao@hrbeu.edu.cn, {jianghan, xql}@sdu.edu.cn
[2] School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan, China

Abstract. Data sharing is one of the most significant applications of cloud computing. For security and privacy concerns, clients generally encrypt their data before upload them to the cloud. The existing data sharing schemes either entirely rely on the cloud to enforce access control or inevitably involve a trusted third party (TTP) to perform secret key distribution. This thesis proposes a secure data sharing scheme without TTP involved. Our scheme allows users to classify their data and achieves a fine-gained access authorization. The key-distribution is integrated with the user authorization and data sharing procedure. In terms of security, except for semi-honest cloud service provider and external adversary, we also take internal adversary into consideration and analysis security in this strong model.

Keywords: Cloud security · Data sharing · Fine-gained access control · Data reliability and privacy

1 Introduction

Cloud computing provides a practical method to offer service to the tenants at a low price, high performance and high flexibility. Cloud customers are free to care about service development, while cloud providers can concentrate on management activities providing an infrastructure that gives to customers the illusion of the availability of infinite resources [1]. One of the most promising benefits in cloud storage system is to provide ubiquitous, convenient and on-demand access to data shared among the Internet, and data sharing has become the most frequently used service of cloud storage system.

However, although constructed with highly reliable software-hardware architecture, cloud storage system still has some certain security and trust concerns and anxieties. Specifically, when the user outsource their data to the cloud, they totally loss the control of them, and the cloud acquires the chance to dispose the data. It might utilize the data beyond the user's expect, such as to get additional profit, or just simply sell the user's data to other organizations [2]. Thus, client tends to not fully trust the cloud

J. Wan et al. (Eds.): CloudComp 2016, SPNCE 2016, LNICST 197, pp. 123–132, 2018.
https://doi.org/10.1007/978-3-319-69605-8_12

service provider, and for privacy concerns, they encrypt their data before upload them to the cloud.

Secure data sharing among multiple users in untrusted cloud environment is a tough problem as we cannot rely on the cloud to perform access control and user revocation. One of the naive solutions is to encrypt all the data with same symmetric key and send the key to the authorized user. But sharing keys may increase the risk of key exposure, especially when a user is no longer qualified to access the file, the entire file will be re-encrypted with a new key. Another traditional solution is to adopt public-key encryption or identity based encryption (IBE), but it needs to encrypt a document multiple times using different users' public-keys or identity-keys, which lays a heavy burden for the data owner. Broadcast encryption (BE) can alleviate this issue but cannot achieve fine-gained access control.

Attribute based encryption (ABE), especially ciphertext-policy ABE (CP-ABE) provides another seemingly feasible solution for cloud data sharing, which allows the data owner to fully control the access policy of his data. However, although achieving fine-gained authorization, the CP-ABE based schemes require a fully-trusted third party (e.g. Key authority), which is not commonly exist in real-world. In addition, these schemes do not take internal adversaries into consideration.

Thus, it is desirable to construct flexible and scalable fine-gained access control for data sharing, in which data owner can share his data to users with different priority without a TTP involved. In this paper, we propose a scheme achieving this goal. In addition, we also consider the scenario that the data owner want to split the operation of updating data and changing the authority. This is because his some files may be need to update such as changing their grade or re-encryption with a new key for some security reason. He is not willing to affect the existing authority information. Similarly, when he has changed some data users' privileges, he isn't willing to update his data. And our scheme achieves high security that we do not only consider the CSP as a semi-honest adversary, but also consider the misbehaviors of authorized users.

1.1 Related Works

Fine-gained access control for encrypted data sharing in untrusted storage system has been extensively studied. Traditionally, researchers tend to use access control lists (ACLs) to manage users' authorization [3, 4]. Specifically, in Kallahalla et al.'s [3] scheme, the data owner firstly classifies his data and generates success control list for each file-group and then encrypts each file-group with a symmetric key. This symmetric key is latterly distributed to the data user according to the ACL, and thus only the authorized user in ACL can have access to this group of files. However it involves a heavily burden in key management for the data owner, as the key size will growth linearly with the number of file-groups. Gol et al. [4] combine symmetric and public encryption and propose a scheme that enable the users in a specific ACL to use their secret key to recover the symmetric key which is used for decrypting the document. Similarly, it also lays a huge burden for the data owner, as the encryption operation also grows linearly with the number of users in the ACL.

Another approach for cloud data sharing is to utilize the attribute-based encryption. The primary idea of ABE is proposed by Sahai and Waters [5]. After that, according to

either the policy is associated with a ciphertext or a key, two variants of ABE – known as ciphertext-policy ABE (CP-ABE) [6] and key-policy ABE (KP-ABE) [7] – are proposed. Based of ABE, Li et al. [8] propose an accountable CP-ABE scheme, which allows tracing the identities of misbehaving users who leaked the description key to others. Li et al. [9] also propose an outsourced ABE construction which provides checkability of the outsourced computation results in an efficient way. Xie et al. [10] propose a novel access control scheme for cloud data sharing system with efficient attribute and user revocation, which largely eliminates the overhead computation (i.e. from O($2n$) to O(n), where n is the number of attributes). Liang et al. [11] propose a scheme that enable service provider to implement practical and fine-grained encrypted data sharing (i.e. the data owner is allowed to share a ciphertext of data among others under some specified conditions), meanwhile achieves the anonymity of data users who has gotten access to the data. Liang et al. [12] propose another scheme that achieves secure search functionality and fine-gained access control, which enables a data owner to efficiently share his data to a specified group of users matching a sharing policy. In addition, their scheme supports keywords updating after the data has been uploaded to the cloud. Recently, Wang et al. [13] take a fully investigation of key escrow problem in attribute-based encryption and propose a scheme to solve this issue. They propose an improved two-party key issuing protocol that can guarantee that neither key authority nor cloud service provider can compromise the whole secret key of a user individually.

A recent guideline for secure cloud data sharing is to deploy key aggregate method. Chu et al. [14] propose a scheme in which the data owner first categorizes his files into different classes, and encrypts files by the class. After that, he chooses some encrypted files and computes the aggregated key. The key is used for decrypting a group of documents and will be latterly sent to the authorized users. Cui et al. [15] also construct a scheme with high flexibility that can share any group of selected documents with any group of users. And they also propose a key-aggregate searchable encryption scheme. These data sharing schemes are efficiently and flexibly constructed, but they demand the data owner to transfer the aggregated key to the data user, so the data owner would be online all time.

1.2 Contributions

This paper proposes a data sharing scheme with fine-gained access control for multi-tenancy cloud storage architecture. The contribution of this work is listed as follows:

- *Autonomous Authorization.* Our scheme enables the data owner to autonomously manage his data. The data owner is supported to classify his data and share to data users with different priorities. Also, the data user can choose to retrieval data belong to specified group of data owners.
- *Getting rid of TTP.* We propose a secure data sharing scheme without TTP involved. It only needs a trusted key distribution center (KDC) for key generation, which is easy to obtain through existing public key infrastructures (PKI).
- *High efficiency and scalability.* In this scheme, each document is encrypted by symmetric encryption with an independent key, and it is not required for the data

owners to distribute the keys. Instead of that, after received request from the data user, the CSP can generate a partial key for each document and send it along with the document to the user. The data user can recover the deception key by using his secret key.

- *Meliorative security model.* We compose an improved security model to capture a wide range of adversarial behavior. We not only consider the CSP and unauthorized user (i.e. include unregistered users and users that have been revoked) as potential adversary, but also consider the misbehaviors of authorized user.

The rest of this paper is structured as follows. In Sect. 2 we present some preliminaries, including basic computational complexity assumptions and security specifications. We present detailed description of our scheme in Sect. 3. The analysis of our scheme is presented at Sect. 4. And we conclude this paper in Sect. 5.

2 Preliminaries

2.1 Complexity Assumption

Definition 1 (Bilinear Map). Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of some large prime order q. A bilinear map is a map $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

1. Bilinearity. For all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$, we have $e\left(u^a, v^b\right) = e(u, v)^{ab}$.
2. Non-degeneracy. $e(g, g) \neq 1$.
3. Computability. There is an efficient algorithm to compute $e(u, v)$ for any $u, v \in \mathbb{G}_1$.

Definition 2 (Discrete Logarithm). Let g be a primitive root for $\mathbb{F}_q$ and let h be a nonzero element of $\mathbb{F}_q$. The *Discrete Logarithm Problem* (*DLP*) is the problem of finding an exponent x such that

$$g^x \equiv h \mod q$$

The number x is called the discrete logarithm of h to the base.

Definition 3 (Hardness of DLP). The DLP assumption is that there exists no probabilistic polynomial algorithm that can solve the DLP problem.

Definition 4 (Computation Diffie-Hellman Problem). The challenger chooses $a, b \in \mathbb{Z}_p$ at random and outputs $(g, A = g^a, B = g^b)$. The adversary then attempts to output $g^{ab} \in \mathbb{G}$. An adversary $\mathfrak{B}$ has at least an ε advantage if

$$\Pr[\mathfrak{B}\left(g, g^a, g^b\right) = g^{ab}] \geq \varepsilon$$

where the probability is over the randomly chosen a, b and the random bits consumed by $\mathfrak{B}$.

Definition 5 (CDH Hardness Assumption). The computational (t, ε)-CDH assumption holds if there exists no probabilistic polynomial time adversary has at least ε advantage in solving the above game.

2.2 System Architecture and Security Goals

We design a TTP-free multi-user data sharing scheme, which have three entities: cloud server provider (CSP), data owner and data user. The main responsibility of the CSP is to store and process the encrypted data according to authorized users' request. Note that there are many data owners and many data users, so this is a Multi-tenancy Cloud Storage System.

We consider three types of adversaries include external adversary and *honest-but-curious* CSP and legal user as internal adversary. Specifically, CSP are assumed to be semi-honest, which mean the server is honest to save the data owners' files and perform data operations requested for authorized data users, but tries to deduce or guess the extra information from the interactive data. Internal user adversary refers to the legal user participated in the protocol. We assume that he can get all the interactive information. He will deduce the extra data by using his private key and the information he has received. External user adversary means the illegal users including the revoked users and this is the basic demand for any access control and authority management system. Note that we assume that the user-server collusion is not included in our adversarial model.

3 Concrete Constructions

3.1 Proposed Scheme

The proposed scheme consists of the following phases:

(1) *Initialization Phase.*
This is the first phase in our data sharing protocol. In this phase, the system generates public parameters and the public/private key pair of users and CS to initialize the algorithm.
Step 1: $param \leftarrow \text{setup}(\lambda)$
Input security parameter λ and get the public parameter *param*, Let $\psi = (q, \mathbb{G}_1, \mathbb{G}_2, e, g)$, $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclical groups of prime order q, a generator $g \in \mathbb{G}_1$, a Bilinear Map $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and $H: \mathbb{G}_2 \rightarrow \{0,1\}^{\tau}$ is a collision resistant hash function, and τ is key-length of AES encryption, then $param = (\psi, H)$.
Step 2: $PK_i, SK_i \leftarrow \text{KeyGen}(param)$
The key generation algorithm takes security parameter *param* as input and generates the keys for the clients and the CSP. The client $u_i (1 \leq i \leq n)$ gets its secret key and public key as $SK_i = x_i, PK_i = g^{\frac{1}{x_i}}$, $x_i \in_R \mathbb{Z}_q$. The CSP, as a special participant, also gets its public key PK_c and secret key SK_c as $SK_c = \alpha, PK_c = g^{\alpha}$, $\alpha \in_R \mathbb{Z}_q$.

Step 3:
The CSP firstly initializes an authority distribution matrix Λ with the size of n × n. Each element in this matrix $\pi_{ji} = \{A_{jil} | l \in \{1, 2, \ldots, \rho\}\}$ is a set which represents the data owner u_i to authorize data user u_j the privilege to access the document with grade l. ρ is the number of levels that the documents expected to be classified and it varies for different data owners. The initial value of each element π_{ji} is assigned as Φ, which denote that data owner u_i have not given permission to u_j to access to his any document. In the matrix, each row (e.g. the j-th row) represents the user's permission that has been authorized by other data owners, while each column represents the authorizations he has given to other data user (i.e. which users can have access to his documents).

(2) *Encrypting Files Phase.*
In this phase the data owner carrys out the algorithm to encrypt his files.
Step 1: $CK_{il} \leftarrow CKGen(param)$
The client u_i randomly chooses a secret key for each class of document. He chooses secret key CK_{il} for l-th class, where $1 \le i \le n, 1 \le l \le \rho$ and $CK_{il} \in_R \mathbb{Z}_q$.
Step 2: $D_{ilk}, r_{ilk} \leftarrow \mathrm{DK}Gen(SK_i, CK_{il})$
Data owner u_i generates an encryption key D_{ijk} for each of his document D_{ijk} as:

$$DK_{ilk} = H\left(e\left(g^{\frac{r_{ilk} \cdot CK_{il}}{x_i}}, g^{\frac{\alpha \cdot CK_{il}}{x_i}}\right)\right), r_{ilk} \in_R \mathbb{Z}_q, 1 \le i \le n, 1 \le l \le \rho, 1 \le k \le \sigma,$$

in which σ indicates that there are σ documents classified as a specific level.
Step 3: $D'_{ilk} \leftarrow \mathrm{EncDoc}(D_{ilk}, DK_{ilk})$
Data owner u_i encrypts each of his document D_{ilk} and gets its ciphertext D'_{ilk}. We adopt AES for documents encryption. Then he will send the encrypted document and its classification information to CSP.

(3) *Authentication and Revocation Phase.*
This algorithm is run by the data owner to grant a user the privilege of reading (some) files. In this phase, the data owner authorizes each category of documents to its homologous user group, and sends the authorization information to the CSP. After received the authorization information, the CSP stores it in the authority distribution matrix. Note that both the Grant algorithm and Revoke algorithm satisfy dynamic property, which means the data owner can change the author information of his document at any time. Now we will give a detail description of Grant and Revoke algorithm.

- $A_{jil} \leftarrow \mathrm{Grant}(SK_i, PK_j CK_{il})$ User authorization algorithm. This algorithm is used for data owner u_i to authorize his documents of grade l to u_j, and this algorithm outputs the authorization value as $A_{jil} = g^{\frac{CK_{il}}{x_i x_j}}$. By repetitively invoking this algorithm, u_i can authorize u_j with document of different grade. We use $A_{ji} = \{A_{jil} | l \in \{1, \ldots, \rho\}\}$ to denote the authority that u_j has been granted from u_i. User u_i sent Grant (u_j, u_i, A_{ji}) to the CSP and thus the CSP can update π_{ji} into $\pi_{ji} \bigcup A_{ji}$.

- $A_{jil} \leftarrow \text{Revoke}(SK_i, PK_jCK_{il})$. Authority revoke algorithm. This algorithm is run by the data owner such as to revoke a user's privilege of accessing his documents. Specifically, data owner u_i revokes u_j's permission of accessing his documents of grade l. Similar to the above, u_j executes $\text{Revoke}(u_j, u_i, A_{ji})$ and sent the output value to the CSP. After receiving this value, the CSP updates π_{ji} into $\pi_{ji} - A_{ji}$.

(4) *Data Sharing Phase.*
The authenticated data user makes a request to the CSP for some data owner's documents. CSP picks up the object files and computes the relative decrypted key. Then the CSP transfer the documents and each partial key to the data user.
Let T_r denotes the request made by the data user. $T_r = (u_i, U_j)$, in which $U_j = \{u_i | i \in \{1, 2, \ldots, n\} \wedge i \neq j\}$. It means a data user can appoint one or more data owners as his own will. If he wants to get all the data owners' files he has privilege to access, he just uses U instead of U_j.
The data user sends T_r to CSP, and then the CSP executes step 1 to acquire corresponding files and partial decrypted key. After receiving the result, the data user executes step 2 to compute the decrypted key and then decrypts the files.
Step 1: $Z \leftarrow \text{Search}(T_r, \Lambda)$: The CSP executes this algorithm such as to get targeted documents and its corresponding trapdoor that is used for decryption. We describe this Algorithm in Table 1.

Table 1. Search algorithm

Algorithm 1 : Search algorithm

Input: access query T_r and authority matrix Λ

Output : the set of targeted documents and corresponding trapdoor used for decryption

1. initialize Z=Φ
2. for every $u_i \in U_j$ do
3. for every $A_{jil} \in \pi_{ji}$ do
4. for each $D_{ilk} \in D_i$ do
5. $DK'_{ilk} = e\left(g^{\frac{CK_{il}}{x_i x_j} \cdot r_{ilk}}, g^{\frac{CK_{il}}{x_i x_j}}\right)^{\alpha}; Z = Z \cup \{(D'_{ilk}, DK'_{ilk})\};$
6. end for
7. end for
8. end for

Step 2: $D_{ilk} \leftarrow \text{DecDoc}\ (SK_j, Z)$: The u_j computes $DK_{ilk} = H((DK'_{ilk})^{x_j^2})$, which is used to decrypt D'_{ilk} such as to get D_{ilk}.

3.2 Correctness

In this scheme, the data owner uploads his documents along with the document classification information to the cloud. Then in the authorization phase, if he permits a user

to access to a category of documents with higher grade, this implies he also permits this user to get access to other categories of documents with lower grade. Thus, if the set π_{ji} only contains one value, it indicates that the user u_j is only authorized to access to the documents with the lowest grade. In this case, the CSP just perform hash operation among the documents of the lowest grade, such as to recover the partial key of each document, which will latterly send to the client along with the corresponding documents. According to the bilinear property:

$$H\left((Dk'_{ilk})^{x_j^2}\right) = H\left(e\left(g^{\frac{CK_{il}}{x_i x_j} r_{ilk}}, g^{\frac{CK_{il}}{x_i x_j}}\right)^{\alpha \cdot x_j^2}\right) = H\left(e\left(g^{\frac{r_{ilk} \cdot CK_{il}}{x_i}}, g^{\frac{\alpha \cdot CK_{il}}{x_i}}\right)\right).$$

That is the secret key DK_{ijk} which is used for u_i to encrypt the document D_{ilk}. Because of the symmetry property of AES encryption algorithm, this key can be used for decryption correctly.

4 Security and Efficiency Analysis

4.1 Security Analysis

In our proposed scheme, each document is encrypted by AES with a unique symmetric key; thus, according to the security insurance of AES, only if this key is not recovered by adversaries, the document cannot be decrypted. Afterwards, we will regard the CSP and the legal data user as adversary, and explain separately why they cannot recover the secret keys.

Honest but curious CSP as adversary

The CSP can acquire the authority value $g^{\frac{CK_{il}}{x_i x_j}}$ of each user, as well as the random number r_{ilk} of each document, and accordingly, he can figure out $g^{\frac{r_{ilk} \cdot CK_{il}}{x_i x_j}}$. Using his secret key α, he can figure out $\left(g^{\frac{CK_{il}}{x_i x_j}}\right)^{\alpha}$, which result in $g^{\frac{\alpha \cdot CK_{il}}{x_i x_j}}$. Although the CSP also possesses each user's public key $g^{\frac{1}{x_j}}$, as he cannot acquire the data user's secret key x_j, the DLP hardness assumption ensure that he cannot figure out $g^{\frac{\alpha \cdot CK_{il}}{x_i}}$, and thus he cannot figure out the secret key $DK_{ilk} = H\left(e\left(g^{\frac{r_{ilk} \cdot CK_{il}}{x_i}}, g^{\frac{\alpha \cdot CK_{il}}{x_i}}\right)\right)$.

Legal user as adversary

Assuming that the data user have acquired his own authority value $g^{\frac{CK_{il}}{x_i x_j}}$, he can use his own secret key to calculate $g^{\frac{r_{ilk} \cdot CK_{il}}{x_i}}$ and $g^{\frac{CK_{il}}{x_i}}$. However, although he also processes the public key g^{α} of the CSP, the CDH hardness assumption ensures that he cannot figure our $g^{\frac{\alpha \cdot CK_{il}}{x_i}}$. Thus it is guaranteed that even if the internal user acquired additional information, he cannot perform any adversarial behavior, as long as he does not collude with the service provider.

4.2 Performance

We compare our proposed scheme with the schemes proposed in Refs. [8, 14]. Reference [8] based on ABE, so it must have a trusted center called attribute authorities (AA), from which user is entitled to a number of attributes and can obtain a decryption key corresponding to those attributes. Reference [14] based on key-aggregate, so it needs the data owner to be online at all time.

Our scheme only needs a trusted key distribution center for public and secret key generation and it does not need a trusted center to manage user. The users' addition and revocation is made by data owner himself, so it is more flexible. The users' authority information is stored in the CSP, making it convenient to revoke user by just deleting the privilege value. And the CSP transfers the partial decrypted key to the user for the further decryption using his private key, which avoids the data owners to distribute the keys. The details are shown in Table 2.

Table 2. Comparison between our scheme and Refs. [8, 14]

Scheme	Trusted center	Fine-grained access control	User revocation	Data owner online all time
Ref. [8]	Y	Y	S	N
Ref. [14]	N	Y	F	Y
Ours	N	Y	F	N

Y yes, N no, F fast, S slow

5 Conclusion and Future Work

We propose a scheme that allows multi-user to securely and flexibly share their data. We consider the scenario that the data owner wants to split the operation of updating data and changing the authority. The data owner can discretionally share his different data to different users, so it achieves fine-gained access authorization. In our scheme, all of the documents are encrypted by symmetric encryption algorithm, and each of them is encrypted with a unique and independent key. The key-distribution is integrated with the user authorization and data sharing procedure, which means it is not needed for the data owner to be online all time to distribute the encryption keys to the data users. In terms of security, except for semi-honest CSP and external adversary, which have been extensively adopted as adversarial model in most works, we also take the misbehavior of legal users into consideration. Getting rid of the dependency of TTP, this scheme is highly compatible for multi-user scenario, and in order to capture a wide range of application of cloud computing, we will extend our scheme to solve the problem of keyword searching over encrypted data as a future work.

Acknowledgements. This work is supported by the National Natural Science Foundation (NSF) under grant Nos. 61572294, 61602275 and NSF Key Project under grant No. 61632020.

References

1. Ardagna, C.A., Damiani, E., Frati, F., Rebeccani, D., Ughetti, M.: Scalability patterns for platform-as-a-service. In: 2012 IEEE 5th International Conference on, Cloud Computing (CLOUD), pp. 718–725. IEEE, June 2012
2. Motoyama, M., McCoy, D., Levchenko, K., Savage, S., Voelker, G.M.: An analysis of underground forums. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, pp. 71–80. ACM, November 2011
3. Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q., Fu, K.: Plutus: scalable secure file sharing on untrusted storage. In: Proceedings of the 2nd USENIX Conference on File and Storage Technologies USENIX Association, pp. 29–42, March 2003
4. Goh, E.J., Shacham, H., Modadugu, N., Boneh, D.: SiRiUS: securing remote untrusted storage. In: The Proceedings of the Internet Society (ISOC) Network and Distributed Systems Security Symposium (NDSS-03), vol. 3, pp. 131–145, February 2003
5. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
6. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP 2007), pp. 321–334. IEEE, May 2007
7. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM, October 2006
8. Li, J., Huang, Q., Chen, X., Chow, S.S., Wong, D.S., Xie, D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 386–390. ACM, March 2011
9. Li, J., Huang, X., Li, J., Chen, X., Xiang, Y.: Securely outsourcing attribute-based encryption with checkability. IEEE Trans. Parallel Distrib. Syst. **25**(8), 2201–2210 (2014)
10. Xie, X., Ma, H., Li, J., Chen, X.: An efficient ciphertext-policy attribute-based access control towards revocation in cloud computing. J. Univ. Comput. Sci. **19**(16), 2349–2367 (2013)
11. Liang, K., Susilo, W., Liu, J.K.: Privacy-preserving ciphertext multi-sharing control for big data storage. IEEE Trans. Inf. Forensics Secur. **10**(8), 1578–1589 (2015)
12. Liang, K., Susilo, W.: Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. IEEE Trans. Inf. Forensics Secur. **10**(9), 1981–1992 (2015)
13. Wang, S., Liang, K., Liu, J.K., Chen, J., Yu, J., Xie, W.: Attribute-based data sharing scheme revisited in cloud computing. IEEE Trans. Inf. Forensics Secur. **11**(8), 1661–1673 (2016)
14. Chu, C.K., Chow, S.S., Tzeng, W.G., Zhou, J., Deng, R.H.: Key-aggregate cryptosystem for scalable data sharing in cloud storage. IEEE Trans. Parallel Distrib. Syst. **25**(2), 468–477 (2014)
15. Cui, B., Liu, Z., Wang, L.: Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage. IEEE Trans. Comput. **65**(8), 2374–2385 (2016)