A Lightweight Access Control Mechanism for Mobile Cloud Computing

Xuanxia Yao, Xiaoguang Han
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing, China, 100083
yaoxuanxia@163.com

Xiaojiang Du
Department of Computer and Information Sciences
Temple University
Philadelphia PA, USA, 19122
dxj@ieee.org

***Abstract*—In order to meet the security requirement, most data are stored in cloud as cipher-texts. Hence, a cipher-text based access control mechanism is needed for data sharing in cloud. A popular solution is to use the attribute-based encryption. However, it is not suitable for mobile cloud due to the heavy computation overhead caused by bilinear pairing, which also makes it difficult to change the access control policy. In addition, attribute-based encryption can't achieve fine-grained access control yet. In this paper, we present a lightweight cipher-text access control mechanism for mobile cloud computing, which is based on authorization certificates and secret sharing. Only the certificate owner can reconstruct decryption keys for his/her files. Our analyses show that the mechanism can achieve efficient and fine-grained access control on cipher-text at a much lower cost than the attribute-based encryption solution.**

***Index Terms*—Authorization, certificate, access control, mobile cloud storage**

I. INTRODUCTION

With the rapid development of mobile applications and cloud computing, mobile cloud computing as a promising technology for mobile services is drawing more and more attentions. As a basic service, mobile cloud storage provides mobile users with large storage space and also supports data sharing with others. However, the growth of mobile cloud storage is limited by security and privacy concerns [1-2]. In order to make sensitive data secret from cloud server, data are stored in cloud in the form of cipher-texts. On the other hand, sharing data with authorized users is needed by many mobile cloud users, and hence a secure data sharing mechanism is required. In addition, authorized users may want to preserve their anonymity when accessing data. Hence, privacy protection may be needed. To sum up, a secure cloud storage should provide security, privacy, fine-grained access control, dynamic data sharing, scalability and accountability [3-5].

The basic idea of many existing solutions is to limit user's decryption power, and none of them can provide all the properties simultaneously. In addition, heavy computation overheads and inflexibility also make them not suitable for mobile cloud, because mobile devices have very limited resources and mobile users might be online or offline at any time. In this paper, we construct a lightweight cipher-text access control mechanism for mobile cloud storage from a different aspect.

Our contributions are three folds. First, an authorization certificate is introduced to simplify access control on cipher-text in cloud. Second, decryption key reconstruction is achieved via Lagrange interpolation polynomials, which makes decryption key distribution simple and fine-grained access control possible. Third, detailed evaluation and comparison analysis of the proposed cipher-text access control mechanism and the existing solutions are presented.

The remainder of this paper is organized as follows: In Section II, we review related works on secure cloud storage and offer our motivations. In Section III, we discuss the construction of the authorization certificate. Section IV gives the assumptions, notations and system model. Section V presents the lightweight access control mechanism. Section VI evaluates performance of the proposed mechanism and compares it with existing solutions. Section VII concludes the paper.

II. RELATED WORKS

There are three main methods to share confidential data in cloud, which are distributing decryption keys to data sharers directly, broadcast encryption based solutions [3], and attribute-based encryption (ABE) methods [4-7]. In fact, the latter two solutions are similar, because they are all based on bilinear pairing.

Distributing a file's decryption key to an authorized user directly (DDKD) is typically implemented by public key cryptosystem and this method only needs data sharers and files to have a small number of keys [8]. However, its authorization revocation is very complicated. In order to revoke a data sharer's access privilege on a file, the data owner has to update the encryption key, re-encrypt the file and re-distribute the new key to all other authorized users, which makes it not suitable for dynamic data sharing.

Broadcast encryption is used to achieve confidential data sharing due to its multi-recipient encryption. A typical broadcast encryption solution [3] is realized by employing verifier local-revocable group signature, identity-based broadcast encryption with constant size cipher-text and private key, which can provide dynamic secure cloud storage, support

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

dynamic users and data provenance. However, its main operation is the complicated bilinear pairing, which is too expensive for mobile users. Furthermore, the length of the system parameter depends on the maximum number of decrypt parties for a cipher-text, which causes poor scalability.

Attribute-based encryption schemes allow any user to decrypt cipher-text as long as it has the required attributes. This feature makes them a popular solution and can avoid frequent key distribution in cipher-text access control. However, the main operation in ABE is also based on bilinear pairing and updating access control policy needs re-encryption. The heavy computation overheads make them not suitable for mobile cloud users. In addition, since the cipher-text length is proportional to the number of attributes, it is difficult to enforce fine-grained access control using ABE. Although some ABE schemes with constant size cipher-text have been proposed [9], increase of attributes will inevitably cause system parameters rapid grow and lead to large communication overheads.

All the existing solutions are from the point of restricting a user's decryption power but not making any restrictions on accessing cipher-text. There are still several unsolved challenges, such as poor scalability, complicated authorization management, heavy computation overheads, and so on. In this paper, we introduce an authorization certificate and use the secret sharing idea based on Lagrange interpolation polynomial to construct a lightweight cipher-text access control mechanism for mobile cloud storage. The authorization certificate is not only used to prove that its owner has the right to access the related materials but also used to distribute the secret share.

III. Authorization Certificate

An authorization certificate is issued by the data owner and expresses its access control policy. To ensure that only the owner of the certificate can get the related information of the authorized files and reconstruct their decryption keys, an authorization certificate should include 7 items, which is shown in Fig.1.

Fig.1: Authorization certificate

1) Authorization File List

The authorization file list is used to list the name f_i of each file i granted to the certificate owner and denoted by I_1.

2) Valid Period

The valid period is expressed in the form of "from the start date and time to the end date and time", which is denoted by I_2.

3) Mask Value List

The mask value list corresponds to the authorization file list and is denoted by I_3. For file i in the authorization file list, its mask value M_i is calculated by Eq. (1).

$$M_i = E_{PKB}(f(\text{Hash}(f_i \| ID_B)) \bmod p) \quad (1)$$

In Eq. 1, f() is the Lagrange interpolation function for file i.

4) Certificate Number

The certificate number CN is generated by the data owner according a specified rule and should be unique for each certificate. This item is denoted by I_4.

5) Certificate Owner's Digest

The certificate owner's digest COD is used to guarantee that the certificate will not be embezzled, and it is computed by Eq. (2) and denoted by I_5. In Eq. (2), ID_B is certificate owner B's identifier.

$$COD = E_{PKB}(\text{Hash}(CN \| ID_B)) \quad (2)$$

6) Data Owner's Signature

The data owner's signature is computed by Eq. (3) and denoted by I_6. Here, ID_B is certificate owner B's identifier.

$$DS = \text{Sign}_{SKA}(\text{Hash}(I_1 \| I_2 \| I_3 \| I_4 \| \text{Hash}(CN \| ID_B))) \quad (3)$$

7) Data Owner's ID

Data Owner's ID is used to indicate the certificate issuer and is denoted by I_7.

It can be seen that the authorization certificate is an anonymity certificate, because no information of the data owner is revealed, which makes anonymous access possible and can protect users' privacy in certain degree.

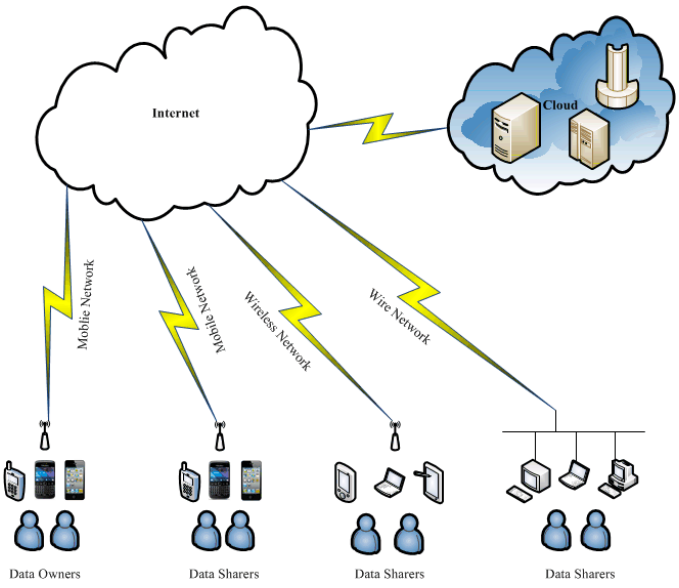

Fig.2: System Model

IV. System Model, Assumptions and Notations

A. System Model

We consider a mobile cloud storage system involving three participants, which are cloud server, data owners and data sharers.

The cloud server provides data storage service and enforces access control on the stored data. Data owners have files stored in cloud and can share them with other users. Authorized users have right to download files in cloud. Fig. 2 shows the relationship among the 3 parties.

Since the proposed scheme is mainly for mobile users with limited resources in mobile cloud computing environment, data owners ask users to access data in the Internet. Data sharers may access data via wireless networks or wired networks.

B. Notations

In order to facilitate the description of the proposed access control mechanism, Table I lists the notations used in this paper.

TABLE I. Notations

Notation	Description
f_i	The name of file *i*
F_i	The contents of file *i*
MD_i	The Message Digest (Hash Value) of F_i
p	A big prime number, its length should be not less than the required length of the symmetric key used in this mechanism.
KF_i	The encryption key based on symmetric cryptograph for file *i*
PK_i	The public key for traditional asymmetric cryptosystem of user *i*
SK_i	The private key for traditional asymmetric cryptosystem of user *i*
$E_k(m)$	Encrypt message *m* using key *k*
$D_k(C)$	Decrypt message *C* using key *k*
$Sign_k(m)$	Make signature on message *m* using key *k*
CF_i	The cipher-text of file *i*
MF_i	The mask value of the cipher-text for file *i*

C. Assumptions

We make the following assumptions.

- The cloud server is semi-trusted and performs access control faithfully.
- All cloud users can do symmetric encryption and decryption and traditional public key based encryption, decryption, signature and verification.
- The hash function used in this paper is secure in cryptology.
- Each data owner keeps the decryption key for every of its files stored in cloud.
- A data file *i* is stored in cloud in the format of Fig. 3.

f_i	CF_i	MF_i	MD_i

Fig.3: The format of a file stored in cloud

Operation type	Operation object	A Data owner's signature on the hash value of the first two items	ID or public key of a data owner

Fig.4: The format of a request message

- All operations made by a data owner should be done by sending a request message to the cloud server. The format of a request message is shown in Fig.4.

The first field is operation type. There are six kinds, which are creating, accessing, modifying, deleting, authorization revocation and registration. The first 4 types are operations on files. Authorization revocation refers to revoke the access privilege on a file from a specified data sharer. Registration means that a new data owner registers itself to the cloud server.

The second field is operation object, which changes with the operation type. When the operation type is creating and modifying, the operation object is the information required in Fig. 3. When the operation type is accessing or deleting, the operation object should be a file name. When the operation type is authorization revocation, the operation object is the number of an authorization certificate and the file names in it. And when the operation type is registration, the operation object is a data owner's ID.

For the first four operation types, the last field should be the ID of a data owner. For registration, the last field should be the public key of a data owner.

V. The Access Control Scheme

The proposed access control scheme consists of five function modules: setup, operations on a file, authorization, file accessing and authorization revocation. We describe each module of the scheme in the following.

A. Setup

The basic task of the setup module is to initialize the system and make each participant ready. There are four things: (1) A pair of public key and private key are generated for each data owner and data sharer in the mobile cloud computing environment. (2) The data owner registers itself to the cloud server by sending a registration request to it. (3) The cloud server verifies the data owner's registering request. If it passes the verification, the cloud server allocates storage space for the data owner and sends acknowledge message to the data owner. (4) After receiving the acknowledge message from the cloud server, the data owner sends all security parameters including information of the hash function, the symmetric crypto algorithm and their parameters to the cloud server.

B. Operations on a File

Here, operations on a file include creating, accessing deleting and modifying a file, which can only be made by the file's owner.

1) Creating a New File

Creating a new file is to upload a new file to cloud. For this purpose, the data owner *A* should use the following steps.

Step 1. Generate a unique name for the new file *i*, such as f_i;

Step 2. Generate an encryption key KF_i for file i according to the security policy;

Setp 3. Select an integer r from [1, p] randomly;

Step 4. Construct a linear function $f(x)=r•x+KF_i$ on finite field Z_p for the new file i;

Step 5. Use the stipulated secure symmetric crypto algorithm to encrypt F_i with KF_i and get its cipher-text CF_i;

Step 6. Calculate the hash value of CF_i and its corresponding function value according to Eq. (4) and Eq. (5), respectively;

$$HC_i=\text{Hash}(CF_i) \bmod p \quad (4)$$

$$MF_i=f(HC_i) \bmod p; \quad (5)$$

Step 7. Compute the message digest of f_i according to Eq. 6;

$$MD_i=\text{Hash}(F_i) \quad (6)$$

Step 8. Compute H_i according to Eq. (7) and create signature on H_i;

$$H_i=\text{Hash}(\text{Creating}\|f_i\|CF_i\|MF_i\|MD_i) \quad (7)$$

Step 9. Construct a creating request message "Creating$\|f_i\|$ $CF_i\|MF_i\|MD_i\|\text{Sign}_{SKA}(H_i)\|ID_A$" and send it to the cloud server.

After receiving this request, the cloud server verifies it using the data owner's public key. If it passes the verification, the file i will be stored in cloud in the form of "f_i || CF_i || $MF_i\|MD_i$" and an acknowledgement is sent to the data owner.

2) Accessing a File

For a data owner A, accessing its one file i is very simple, it just needs to send an accessing request to the cloud server. According to Fig. 4, the accessing request is "Accessing|| f_i || $\text{Sign}_{SKA}(\text{Hash}(\text{Accessing}\|f_i))\|ID_A$".

After receiving this request message, the cloud server verifies it using A's public key. If it passes the verification, the file i is sent to A in the form of "f_i || CF_i || MD_i".

3) Deleting a File

If a data owner A wants to delete the file i in cloud, it only needs to send a deleting request to the cloud server. The deleting request message is constructed according to Fig. 4, here it is "Deleting$\|f_i\|\text{Sign}_{SKA}(\text{Hash}(\text{Deleting}\|f_i))\|ID_A$".

After receiving the delete request, the cloud server verifies the signature, if it passes the verification, file f_i and its related information will be deleted from cloud by the cloud server.

4) Modifying a File

Here, we assume that the modification only involves the file's contents. If the file's name is modified, it will be treated as a new file. When a data owner A wants to modify its file i stored in cloud, it should use the following steps.

Step 1. If it has a copy of file i locally, go to step 4, otherwise, Access file i from cloud;

Step 2. Decrypt the cipher-text and verify its integrity. If the verification is not passed, go to step 7;

Step 3. Make modification on the file and re-encrypt it to get CF_i';

Step 4. Perform the operations on the modified file i from step 5 to step 7 in the subsection of "1) Creating a New File" to get MF_i' and MD_i';

Step 5. Compute $H_i'=\text{Hash}(\text{Modifying } \|f_i\|CF_i'\|\ MF_i'\|\ MD_i')$ and make signature on it, which is $\text{Sign}_{SKA}(\ H_i')$;

Step 6. Construct a modifying request "Modifying $\|f_i\|CF_i'\|$ $MF_i'\|\ MD_i'\|\text{Sign}_{SKA}(\ H_i')\|ID_A$" and sent it to the cloud server;

Step 7. Stop.

After receiving the modifying request, the cloud server verifies it using the data owner's public key. If it passes the verification, the cloud server will replace $CF_i\|MF_i\|MD_i$ with $CF_i'\|\ MF_i'\|\ MD_i'$ for file i.

C. Authorization

In our access control mechanism, authorization module issues an authorization certificate to a specified user. If data owner A wants to grant the access privilege on one or some of its files to data sharer B, it only needs to construct an authorization certificate in accordance with the requirements in Section III and send it to user B by a short message or other security channels over Internet.

D. File Accessing

Here, file accessing refers to an authorized data sharers accessing the authorization files. When an authorized user B wants to access an authorized file i of data owner A, it should do three things: (1) To get file i from cloud; (2) To obtain the decryption key of file i; and (3) To decrypt the cipher-text and verify its integrity.

1) Getting File i from Cloud

In order to get file i from cloud, data sharer B needs to construct the access request message "accessing || f_i || COD || authorization certificate" and sent it to cloud server.

After receiving the user's accessing request, the cloud server will make access control on the requested file i according to following steps.

Step 1. Check whether the requested file is an authorization file or not according to authorization file list. If not, go to step 7;

Step 2. Check whether the current time is between the starting time and the end time of the authorization certificate, if it is not in the validity period, go to step 7;

Step 3. Check whether the authorization certificate is in the authorization revoked list or not, if not, go to step 5;

Step 4. Check whether the authorization certificate is revoked completely, if yes, go to step 7, otherwise check whether the request file is revoked or not, if yes, go to step 7;

Step 5. Verify authenticity of the signature on the authorization certificate according to OCD and the certificate, if the verification isn't passed, go to step 7;

Step 6. Send file i's related information ($f_i\|CF_i\|MF_i\|MD_i$) stored in cloud to the data sharer;

Step 7. Stop.

2) Obtaining the Decryption Key

After receiving the related materials of the requested file i, data sharer B should do the following steps so as to obtain the decryption key.

Step 1. Compute $x0_i=\text{Hash}(CF_i) \bmod p$, and lets $y0_i= MF_i$;

Step 2. Compute $x1_i$ according to Eq.8;

$$x1_i = \text{Hash}(f_i\|ID_B) \bmod p \quad (8)$$

Step 3. Compute $y1_i$ according to Eq.9, where M_i is from the authorization certificate;

$$y1_i=\text{D}_{SKB}(M_i) \quad (9)$$

Step 4. Compute the decryption key for file f_i according to Eq.10.

$$KF_i=((x1_i y0_i - x0_i y1_i)/(x1_i - x0_i)) \bmod p \quad (10)$$

3) Decrypting the Cipher-Text and Verifying Its Integrity

After obtaining the decryption key KF_i for the requested data file i, data sharer B can decrypt the cipher-text of it. The decryption result is denoted by F'_i, there should be Eq.11.

$$F_i' = D_{KFi}(CF_i) \quad (11)$$

In order to verify the integrity of F_i', the data sharer should check whether Eq.12 is true or not.

$$\text{Hash}(F_i')=MD_i \quad (12)$$

If the Eq.12 is true, it indicates that the F_i' is equal to F_i and the data sharer B obtains the correct file i.

E. Authorization Revocation

In our scheme, authorization revocation is very simple, by which fine-grained revoking can be realized. For instance, when data owner A wants to revoke data sharer B's accessing privilege on its file i, A just needs to send a revoking request "Revoking$\|CN\|f_i\|\text{Sign}_{SKA}(\text{Hash}(\text{Revoking}\|CN\|f_i))\|ID_A$" to the cloud server.

After receiving the revoking request message, cloud server knows that A wants to revoke the access privilege of f_i in the authorization certificate with number of CN. It will verify the signature in the revoking request, if it passes the verification, the certificate number CN and the revoked file's name are recorded in the authorization revocation list.

It needs to state that the file name part can be empty if A wants to revoke all the authorization in the certificate with number of CN.

VI. Evaluation and Analysis

We conduct evaluation and analysis from two aspects. Firstly, we use the requirements for secure cloud storage as the criterions to evaluate it. Then we conduct a comparison analysis on overheads for the proposed scheme and several popular solutions.

A. Performance Evaluation

The basic performance requirements for secure cloud storage include five aspects. They are security, fine-grained access control, dynamic data sharing, scalability and accountability. We evaluate the proposed mechanism from the five aspects as following.

1) Security

The security requirement includes confidentiality and integrity.

For confidentiality, we guarantee it from two aspects. Firstly, the secure symmetric cryptographic algorithm is used to encrypt files stored in cloud so as to keep them security from cloud server. Secondly, only the authorized user can access and decrypt the cipher-text of the authorization file by introducing the authorization certificate. Since the authorization certificate can bound the privilege and the authorized user, only the authorized user or the certificate owner can pass the verification of the cloud server and get the cipher-text and its corresponding secret share of the authorization file. At the same time, the authorization certificate also help to distribute the other secret share to the authorized user securely. Any non-certificate-owner can neither pass the cloud server's verification for getting the related information of the files in the authorization list nor get the secret share in the certificate owing to not knowing the private key of the certificate owner.

For integrity, we use hash functions to generate the digest of the file, by which the file owner and authorized data sharer can verify the integrity of the file in cloud.

2) Fine-Grained Access Control

In the proposed mechanism, access control on the cipher-text in cloud is essentially achieved on the basis of authorization certificate. An authorization certificate is a credential issued by a data owner to prove that the owner has the privilege to access the specified file in cloud. Since data owner can specify who can access which file of it according to actual necessary, we say that the fine-grained access control can be achieved.

3) Dynamic Data Sharing

Dynamic data sharing means the data sharers are variable. Since the data sharers' changing has no effects on the cipher-text and its related materials, moreover, a data owner can grant or revoke the access privilege of a file to or from a user as needed, which means dynamic data sharing is well supported.

4) Scalability

In our mechanism, the size of the cipher-text and its related security parameters has nothing to do with the number of the data sharers. Adding a data share is only to issue an authorization certificate to it, which is very easy for the data owner and doesn't cause any decline in system performance, so It can be said that our mechanism has good scalability.

5) Accountability

Since the owner of an authorization certificate is anonymous, the cloud server can't get anything about it. For realizing accountability, we let the cloud server record the certificate number in an access log. The anonymous access can be revoked by the collaboration of the data owner and the cloud server, because the identity of an authorization certificate owner is known to data owner.

B. Comparison Analysis

A cipher-text access control mechanism for cloud storage mainly involves key management and authorization management two aspects. The authorization management can be further classified into authorization and authorization revoking two basic operations. So we analyze and compare the overheads in our mechanism and the typical existing solutions from the three aspects.

1) Key Management

Here, key management refers to how to distribute the decryption key.

In our mechanism, the decryption key distribution is essentially the reconstruction of the decryption key, which is achieved on the basis of Lagrange interpolation polynomial. It needs 2 decryption based on public key cryptographic graphic, 1 hash computing and some simple arithmetic computing.

In DDKD solutions, the decryption key is usually distributed by the trusted third party or PKI. Depending on the trusted third party always leads to high communications and

some unsecure factors. And using PKI needs one decryption based on traditional public key cryptosystem, the overhead is a little lower than our mechanism.

In the attribute-based encryption solutions, the decryption key distribution is implicit, which is based on the very expensive bilinear pairing. The overhead is much more than our mechanism and DDKD solutions.

2) Authorization

Authorization refers to grant the access privilege of the file to the specified user.

In our mechanism, authorization is to issue an authorization certificate to a data sharer, in which the most expensive operation is the traditional public key based encryption. This authorization mechanism makes it very easy for a data owner to perform fine-grained and dynamic authorization.

In DDKD solutions, authorization is bound with decryption key distribution and needs one encryption based on public key.

In the ABE-based solutions, authorization is implicit. The access control policy is associated with the user's private key (KP-ABE) or cipher-text (CP-ABE), which is realized on the basis of the expensive bilinear pairing. In addition, since one attribute may be held by many users and a set of attribute may correspond to more than one user, which makes it difficult to achieve fine-grained access control.

3) Authorization Revocation

Authorization revocation is to revoke the access privilege on a file from a data sharer.

In our mechanism, since the access control on files in cloud is made by the loud server, authorization revoking is very easy for a data owner, who just sends a revoking request message to tell the cloud server to revoke the access privilege on which file in which certificate. The expensive overhead is one signature based on public key cryptosystem.

In DDKD solutions, Authorization revocation is very complicated. In order to revoke the access privilege on a file, the data owner has to do three things. The first is to update the file's key; the second is to re-encryption the file with the new key and upload it; and the third is to send the new key to all the rest authorized users. The computation overhead depends on the number of the remainder authorized users.

In ABE based solutions, Authorization revocation is also very complicated. Since the data owner has to re-encrypt the file's key using ABE algorithm, it will lead to heavy computation overhead.

VII. Conclusion

In order to meet the requirements for mobile cloud storage, a lightweight access control mechanism based on authorization certificate and secret sharing is proposed. Since all the operations in our mechanism are based on the traditional symmetric (or public key) cryptosystem, hash and simple arithmetic computing, it avoids problems in existing solutions, such as heavy computation overhead, and difficulties in authorization revoking. Hence, our mechanism is very suitable for mobile cloud storage. The detailed evaluation and comparison analysis showed that the proposed mechanism has significant advantages in authorization revocation over the existing solutions.

Acknowledgment

This work is supported by Chinese National Scholarship Fund and Chinese National High Technology Research and Development Program 863 under Grant No. 2012AA121604, as well as the US National Science Foundation (NSF) under grant CNS-1065444.

References

[1] A. N. Khan, M.L. Mat Kiah, S. U. Khan, S. A. Madani. "Towards secure mobile cloud computing: A survey. Future Generation Computer Systems". July 2013, vol. 29, no.5, pp. 1278-1299.

[2] Z. Zhou, D. Huang, "Efficient and secure data storage operations for mobile cloud computing", Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm). 22-26 Oct. 2012. Las Vegas, NV. pp. 37-45.

[3] S. S.M. Chow, C. Chu, X. Huang, J. Zhou, R. H. Deng. "Cryptography and security-dynamic secure cloud storage with provenance". Cryptography and Security: From Theory to Applications, Lecture Notes in Computer Science, vol 6805, 2012, pp. 442-464.

[4] W. Jia, H. Zhu, Z. Cao, L. Wei, X. Lin, "SDSM: a secure data service mechanism in mobile cloud computing", in: Proc. IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, Shanghai, China, Apr. 2011. pp. 1060-1065.

[5] S. Yu, C. Wang, K. Ren, W. Lou. "Achieving secure scalable and fine-grained data access in cloud computing". INFORM'10 Proceedings of the 29th conference on Information Communications.2010, pp. 534-542.

[6] V. Goyal, O. Pandey, A. Sahai, B. Waters. "Attribute-based encryption for fine-grained access control of encrypted data". CCS'06 Proceedings of the 13th ACM conference on Computer and communications security.2006, pp. 89-98.

[7] N. Attrapadung, B. Libert, E. de Panafieu. "Expressive key-policy attribute-Based encryption with constant-size ciphertexts". 14th International Conference on Practice and Theory in Public Key Cryptography(Public Key Cryptography-PKC 2011), Taormina, Italy, March 6-9, 2011.pp. 90-108.

[8] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-policy attribute-based encryption". Proceeding SP '07 proceedings of 2007 IEEE Symposium on Security and Privacy. 20-23 May 2007, Berkeley, CA. pp. 321-334.

[9] A. Lewko, B. Waters, "New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts," in 7th Theory of Cryptography Conference(TCC 2010), Zurich, Switzerland, February 9-11, 2010, pp. 455-479.