Threshold Cryptography Based Data Security in Cloud Computing

Sushil Kr Saroj
CSE Department
GCET, Greater Noida

Sanjeev Kr Chauhan
CSE Department
MNNIT, Allahabad

Aravendra Kr Sharma
CSE Department
Galgotias Univ., Greater Noida.

Sundaram Vats
CSE Department
GCET, Greater Noida.

***Abstract*-Cloud computing is very popular in organizations and institutions because it provides storage and computing services at very low cost. However, it also introduces new challenges for ensuring the confidentiality, integrity and access control of the data. Some approaches are given to ensure these security requirements but they are lacked in some ways such as violation of data confidentiality due to collusion attack and heavy computation (due to large no keys). To address these issues we propose a scheme that uses threshold cryptography in which data owner divides users in groups and gives single key to each user group for decryption of data and, each user in the group shares parts of the key. In this paper, we use capability list to control the access. This scheme not only provides the strong data confidentiality but also reduces the number of keys.**

***Keywords*: Outsourced data, malicious outsiders, access control, authentication, capability list, threshold cryptography.**

I. Introduction

Cloud computing is a new and fast growing technology in field of computation and storage of data. It provides storage and computing as a service at very attractive cost. It provides services according to three fundamental service models: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). Storage as a service is basically a platform as a service. The five characteristics of cloud computing are: on-demand service, self service, location independent, rapid elasticity and measured scale service. These characteristics make cloud significant. Industries and institutions are exploiting these characteristics of cloud computing and increasing their profit and revenue [1]. That is why, industries are shifting their businesses towards cloud computing.

However, data security is a major obstacle in the way of cloud computing. People are still fearing to exploit the cloud computing. Some people believe that cloud is unsafe place and once you send your data to the cloud, you lose complete control over it [8][9]. They are more or less right. Data of data owners are processed and stored at external servers. So, confidentiality, integrity and access of data become more vulnerable. Since, external servers are operated by commercial service providers, data owner can't trust on them as they can use data for their benefits and can spoil businesses of data owner [4]. Data owner even can't trust on users as they may be malicious. Data confidentiality may violet through collusion attack of malicious users and service providers.

Many schemes are given to ensure these security requirements but they are suffering from collusion attack of malicious users and cloud service provider and heavy computation (due to large no keys). To address these issues we propose a scheme. In this scheme, there are basically three entities: Data Owner (DO), Cloud Service Provider (CSP) and Users. Users are divided in groups on some basis such as location, project and department and, corresponding to each group, there is a single key for encryption and decryption of data. Each user in the group shares parts of the key. Data can be decrypted when at least threshold number of users will present. This scheme not only provides data confidentiality by all means but also reduces the number of keys. To achieve fine-grained data access control, the approach has used capability list [6]. It is basically row-based decomposition of access matrix. In capability list authorized data and operations for a user are specified. It is better suit than Access Control List (ACL) [5][10][16] because ACL specifies users and their permitted operation for each data and file. It is practically inefficient that two users require same data and have same operations on it. In this paper, the approach has used the modified Diffie-Hellman algorithm to generate one time shared session-key between CSP and user to protect the data from outsiders. To ensure data integrity the approach has used MD5 [4].

The remaining portion of this paper is organized as follows. In Section II, we review the related work. In Section III, we describe the model and assumption. In Section IV, we present our proposed scheme. Section V shows Hierarchy management of access rights. In Section VI, we analyse our approach in terms of security and performance. Finally, we conclude the paper in section VII.

II. Related Work

Data confidentiality and access control are two basic security requirements for outsourced data in cloud computing. Sometime, when we emphasize more on security of data, we forget about performance of systems (DO, CSP, users). For example, to secure data, we sometime use too many keys. We know that keys are confidential, so there is need to secure and maintain these keys which are additional work. These additional works affect the performance of the system. So, it is desirable to reduce no of keys. So, there is need a scheme that provides not only data security but also maintain the performance. Many schemes are suggested to meet these requirements.

Symbol	Description
DO	Data Owner
CSP	Cloud Service Provider
Pu	Public Key
Pr	Private Key
K_T	Symmetric Key
Ek	Encryption
Dk	Decryption
PuCSP	Public Key of CSP
PrCSP	Private Key of CSP
PuDO	Public Key of DO
PrDO	Private Key of DO
PuUSR	Public Key of User
PrUSR	Private Key of User
Fi	i^{th} File
Di	i^{th} File Message Digest
d	Number of Shares
UID	User Identity
FID	File Identity
AR	Access Right
CPList	Capability List
M	Message
PKS	Partial Key Set
K_{Ti}	i^{th} User's partial symmetric key
OR	OR operation of Gate
MD5	Hash Algorithm
$X_{A/B}$	Chosen Secret Key
$Y_{A/B}$	Calculated Public Key
K_S	Secret Session Key
q	Prime number
p	Primitive root
MOD	Modulus Function

The scheme proposed in [13] is the group-key scheme. In group-key scheme, there is a single key corresponding to each group of users for decryption process and all users of the group know that key. Here, number of keys is reduced but there is a problem of collusion attack of CSP and a user because a single malicious user can leak whole data of the group to CSP. We know that CSP is not trusted party. It can use data owner's data for its commercial benefits.

The scheme proposed in [4] tried to achieve data confidentiality and access control. In this scheme, data are encrypted by symmetric keys and symmetric keys are known only to data owner and corresponding data users. The encrypted data are stored at CSP. CSP can't see data stored at it as data are encrypted. Data are further encrypted by one time secrete session-key shared between CSP and user by the modified Diffie-Hellman protocol to protect data from outsiders during the transmission between CSP and user. This scheme no doubt provides whole data security but there is associated a key corresponding to each user and users may be large in number in some applications. So, number of keys may increase. Hence, increases the maintenance and security concerns of keys

Communication model of the proposed scheme somehow matches with it [4] but proposed scheme is more secure and reduces number of keys. The proposed scheme is useful for those applications where works are done in team and group such as in software industries. You may think proposed scheme has limited applications but it is not as such. It is applicable all where you can group users on some basis and can apply threshold cryptography technique. Such as software and hardware industries, institutes, banks and medicals fields. There is provision of hierarchy of access in this scheme which makes this scheme more useful and realistic. For Example, an university has vice-chancellor, hods, teachers, clerklier-staff and students. Each one has different level of access right.

III. Model and Assumptions

We suppose that our model is composed of three entities: a CSP, a DO and many users associated with DO. Initially, all users are registered at DO. During registration users send their credentials to DO. We assume that user's credentials are sent securely to DO. DO then divides users in groups and provides encryption keys, tokens, algorithm (MD5) and other necessary things for secure communication to user groups in response of registration. A user can get data from CSP in a confidential manner after successful authentication of himself at CSP. We assume that CSP has a large capacity and computational power. We also assume that no one can breach the security of CSP. Further we assume that the algorithm which is used to generate the secrete keys for encryption, is secure at DO. DO

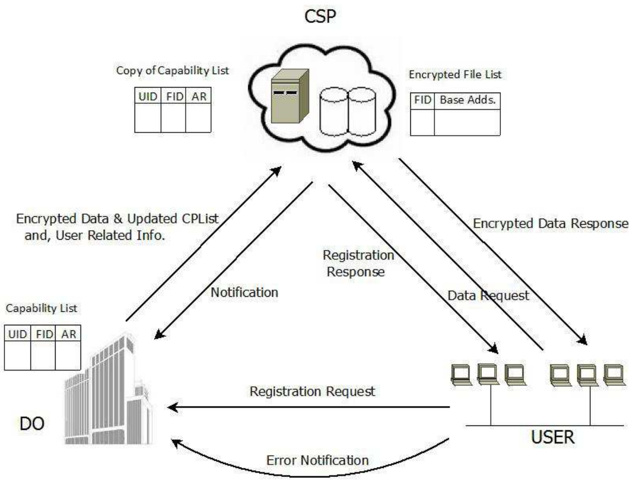

Fig. 1. Communication Model in the Proposed Scheme

has storage capacity to store some files and data and, he can execute programs also at CSP to manage his files and data. We are using modified Diffie-Hellman and public key cryptography to secure communication between CSP and user. Modified Diffie-Hellman protocol is used to create one time session-key between CSP and user. Fig.1 illustrates the secure communication between entities in the proposed scheme.

IV. Proposed Scheme

In this section, we present a complete model for secure communication between different entities and secure access to data. There are four algorithms in the proposed scheme. Algorithm 1 describes secure communication of data between DO and CSP moreover this algorithm insures data confidentiality and, authentication of DO and CSP. Algorithm 2 describes procedures which DO and CSP apply after a new file creation in respect. Algorithm 3 describes about secure communication of data between CSP and user. In this algorithm user's authorization is also checked. Algorithm 4 describes the threshold cryptography technique for decryption of a user's file. Algorithm 4 is applied at user side where number of keys is reduced (one key corresponding to one group) and no threat of collusion attack as in group-key scheme.

To understand proposed scheme better we take an example of real life scenario, DO may be a software industry who stores its data on to the CSP and the users may be its employees who view their data from the CSP. DO divides users in groups on some basis such as project basis and encrypts the data of each group with a single symmetric key (K_T) and, it gives parts of the symmetric key (K_T) to each user of the group. DO computes digest of data by using 128-bit MD5 hash algorithm and then encapsulates the digest and data using the symmetric key (K_T). This in turn, provides strong data confidentiality and integrity. DO then fills the entries such as UID, FID and AR in Capability List corresponding to each new user. DO then encrypts Capability List and encapsulated things with its private key after that public key of CSP and, then sends all things to CSP. These encryptions ensure confidentiality and authentication between DO and CSP.

Algorithm 1: Procedure to be followed by CSP after getting encrypted File and Capability List from DO

```
Step 1: CSP stores Encrypted Data and Capability List
which are received from DO
Array ← Rece($Ek_{PuCSP}(Ek_{PrDO}(Ek_{K_T}(Fi)) || (CPList))$

$CPList || Ek_{K_T}(Fi) ← Dk_{PrCSP}(Dk_{PuDO}((Array))$

Step 2: CSP updates the Encrypted File List
Encptd. File List ← Encptd. File List (FID, Base Adds.)

Step 3: CSP updates Capability List
CPList ← CPList(UID, FID, AR)
```

Algorithm 1 describes the process what CSP do after getting encrypted data and Capability List from the DO. CSP decrypts the message using its own private key and the public key of data owner and stores the encrypted data and Capability List in its storage. CSP then updates the encrypted File List and Capability List. Since, data are encrypted using symmetric key (K_T) which is known only to DO and respected user group, CSP can't see data even though user's credential comes through it.

Algorithm 2: Procedure to be followed after a new File creation

```
Step 1: DO updates Capability List
CPList ← Add.(CPList, (UID, FID, AR))

Step 2: Now, DO encrypts the CPList, Encrypted File,
symmetric key and sends these to the CSP
Send($Ek_{PuCSP}(Ek_{PrDO}(CPList, Ek_{K_T}(Fi), Ek_{PrDO}$

$(Ek_{PuUSR}(K_T, N+1, TimeStamp)))))$

Step 3: CSP Updates its copy of the Capability List,
Encrypted File List and sends symmetric key to indented
user group
Send($Ek_{PuUSR}(Ek_{PrDO}(Ek_{PuUSR}(K_T, N+1, TimeStamp))))$

Step 4: Now, the user can send actual access request for
that File directly to CSP
```

Algorithm 2 illustrates the procedure required after a new File creation. When a new File is created, DO fills entries for that File in Capability List containing UID, FID and AR. DO generates a symmetric key (K_T) and encrypts File with that symmetric key (K_T). Now, DO encrypts the updated CPList, Encrypted File and symmetric key (K_T) with its private key after that public key of CSP and sends these to the CSP. When CSP receives these, it updates Capability List, Encrypted File List and sends encrypted symmetric key (K_T) to respective user group. Users of the user group then decrypt the message and get their own parts of the symmetric key (K_T). To avoid man-in-middle and replay attack we use nonce and timestamp in each message. After getting the details, user can request to CSP for data.

Algorithm 3 describes how data are exchanged securely between CSP and the user by use of modified Diffie-Hellman algorithm. We called it modified D-H algorithm as we encrypt the D-H parameters using the public key of one side and, using nonce in each direction during session key (K_S) generation and data transfer. It helps to counter the man-in-the middle attack. After available of keys and tokens, the user may request for data to CSP. CSP initiates modified D-H key exchange with the user, if request is authentic. We assume that the session key (K_S) is shared between CSP and the user by modified Diffie-Hellman algorithm. Now, CSP encrypts the encrypted File (Fi) and its digest (Di) with the shared session key (K_S) and sends it to the user. This over encryption ensures the confidentiality of the message between cloud service provider and the user. The user then decrypts the message (user decrypts the message according to algorithm 4) and calculates the digest of File and then matches it with stored digest. If digest matches, File is original otherwise File is modified by outsiders and user then sends an error notification message to DO.

Algorithm 3: Algorithm for secure data exchange between CSP and User by using Modified D-H key exchange

```
Step 1: User sends data access request to CSP
Send(UID, FID, AR))

Step 2: CSP matches UID, FID, AR with CPList stored at it.
If( match)
    Go to step (3)
else
    Go to step (6)

Step 3: CSP initiates D-H exchange with that User and shares one time shared session key( $K_S$)

Step 4: CSP encrypts the encrypted File with shared session key and sends it to User
Send($Ek_{K_S}$($Ek_{K_T}$(Fi)))

Step 5: User decrypts the File and calculates the message digest of that File
If
   Calculated digest matches with stored digest then File is original
else
   File is modified and User sends Error Notification to DO

Step 6: CSP sends 'invalid request' message to User
```

Algorithm 4 which resembles the threshold cryptography technique [2][14], describes the procedure how a File is decrypting for User 1. After getting encrypted message, user's main concern how to decrypt it because he alone can't decrypt. So, he first updates PKS Vector (Initially, all bits of it are zero) with his key component and then sends PKS Vector and encrypted message to next user of same group. The next user then decrypts the message and updates the PKS Vector with his key component. This is continuing until all bits of PKS Vector are one. Here, we can see that application is not using all key components (Only threshold no of key components). After this, data are sent back to initiator user. Initiator user then decrypts the message and gets it. Initially, User 1does not decrypt message (M), he just updates the PKS Vector.

Algorithm 4: Algorithm for Decryption of a File for User 1

```
Step 1: User 1 receives Encrypted File
M ← Rece.($Ek_{K_T}$(Fi))

Step 2: Initially, all bits of PKS Vector is zero. Here, PKS Vector indicates parts of the key. User 1 will update this PKS Vector with the components he has
$PKS = PKS\ OR\ A_1$

Step 3: User 1 forwards M and PKS to ith user of the group. Who will decrypt it and update the PKS Vector
M = $Dk_{K_{T_i}}$(M)
$PKS = PKS\ OR\ A_i$
The ith user then forwards M and PKS to next user in the group.  The next user performs same operations as ith user did. This process is continued.

Step 4: if(PKS = = 11111............................up to d bits)
    Go to step (5)
else
    Go to step (3)

Step 5: Forward M to the User 1
User 1 then decrypts message (M) and get File
Fi = $Dk_{K_{T_1}}$(M)
```

Decryption process of a File for User 1 is shown in Fig. 2

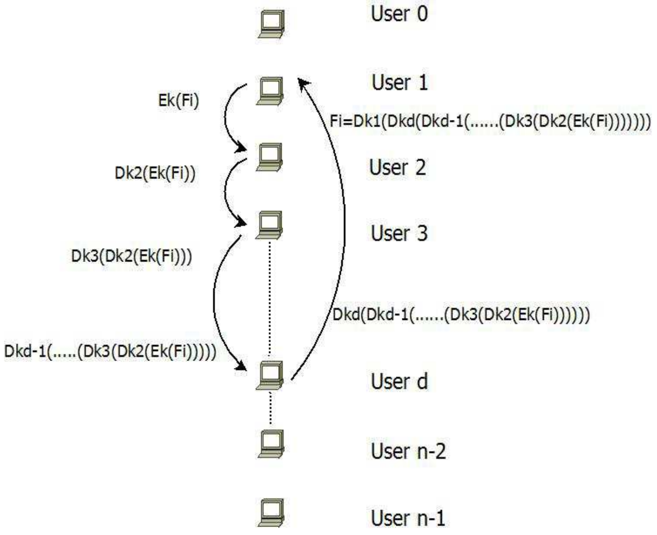
User 0
User 1
Ek(Fi)
Fi=Dk1(Dkd(Dkd-1(......(Dk3(Dk2(Ek(Fi)))))))
User 2
Dk2(Ek(Fi))
User 3
Dk3(Dk2(Ek(Fi)))
Dkd(Dkd-1(......(Dk3(Dk2(Ek(Fi))))))
Dkd-1(.....(Dk3(Dk2(Ek(Fi)))))
User d
User n-2
User n-1

Fig. 2. Process of Decryption for User 1

V. Hierarchy Management of Access Rights

In this section, we describe hierarchy of access rights of users. Fig. 3 illustrates about it. Here, users encircle in same group have same access rights. In other way, we can say that they are sharing key components of the key with which data of that group are encrypted. Along with this, each user can also see data of all the groups which are formed with users beneath him in the hierarchy. Here, we are calling child group for such users group but the user uses dummy key for it and this key is not shared with any others. When a user wants to see data of his child group, he encrypts the encrypted data of child group with his dummy key and sends encrypted data to that child group for decryption. Here, decryption is same as Algorithm 4. After all bits of PKS Vector are one, data returns to the user (initiator). The user then decrypts it with his dummy key and gets data.

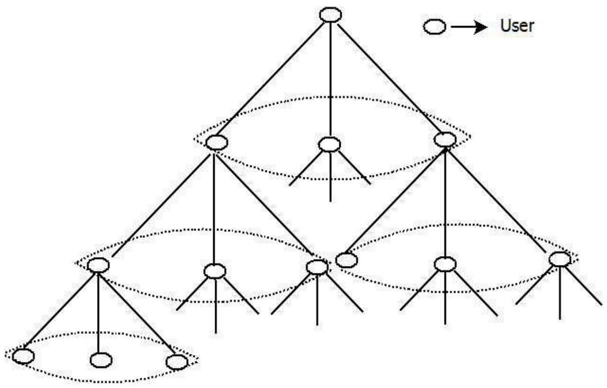

Fig. 3. Hierarchy of Access Rights

VI. Security and Performance Analysis

A. Security Analysis

In this section, we analyse the approach in terms of security strength and scalability.

1) *Data confidentiality:* In the proposed scheme, DO stores its data at CSP in encrypted form. Since, data are encrypted by the symmetric keys which are known only to DO and respective users group, CSP can't see the data. After validation of user request, CSP sends encrypted data to the user. To protect data from outsiders CSP again encrypts the encrypted data by the onetime session key (K_S) shared between CSP and user. Here, we can see that key length is increased due to applying double encryption, which helps against brute-force attack. In the proposed scheme, no member of any group knows about whole key (due to threshold cryptography). They know only about for what they are authorised. Hence, collusion attack of CSP and users is not possible.
2) *Entity Authentication:* In the proposed scheme, user is authenticated at DO when he sends his personal details to DO by encrypting its own private key during registration. DO is authenticated at CSP when it sends capability list and encrypted message digest and data to CSP by encrypting its own private key. User is authenticated at CSP when user's ID and password match with user's ID and password stored at the database of CSP.
3) *Data Integrity:* The proposed scheme uses MD5 to calculate the message digest of data. DO encrypts data and message digest of data with a symmetric key K_T which is known only to respected user group. When user gets encrypted data and message digest from CSP, he decrypts it first and then calculates the message digest of received data. Data integrity is ensured when received and calculated digest match.
4) *Data Access Control:* To ensure data access control, the proposed scheme uses capability list in the approach. Capability list basically contains the UID, FID and AR. Only DO has rights to perform any operation on it. CSP only can read it for the purpose of secure data access. CSP sends only those data to users what are in their access rights. In other way, users can access those data which are in their capability [15].

B. Performance Analysis

We have seen that DO transferred maximum of its load and computation to CSP and did only necessary things by itself. No of keys (because in threshold cryptography, there is a single key corresponding to each group) have reduced in the proposed scheme. Hence, reduces the maintenance and security concerns of keys and reduces the additional computation time.

VII Conclusion

In this paper, we presented a new approach which provides security for data outsourced at CSP. Some approaches are given to secure outsourced data but they are suffering from having large number of keys and collusion attack. By employing the threshold cryptography at the user side, we protect outsourced data from collusion attack. Since, DO stores its data at CSP in encrypted form and, keys are known only to DO and respected users group, data confidentiality is ensured. To ensure fine-grained access control of outsourced data, the scheme has used capability list. Public key cryptography and MD5 ensure the entity authentication and data integrity respectively. Public key cryptography and D-H exchange protected the data from outsiders in our approach. No of keys (because in threshold cryptography, there is a single key corresponding to each group) have reduced in the proposed scheme.

References

[1] J. Do, Y. Song, and N. Park, "*Attribute Based Proxy Re-encryption for Data Confidentiality in Cloud Computing Environments*," Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First

ACIS/JNU International Conference on, vol., no., pp.248-251, 23-25 May 2011.

[2] A. Shamir, "*How to share a secret*," Communications of the ACM, v.22 n.11, p.612-613, Nov. 1979. [Online]. Available: http://portal.acm.org/citation.cfm?id=359168.359176.

[3] N. Bennani, E. Damiani, and S. Cimato, "*Toward Cloud-Based Key Management for Outsourced Databases*," Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual, vol., no., pp.232-236, 19-23 July 2010.

[4] S. Sanka, C. Hota, and M. Rajarajan, "*Secure data access in cloud computing,*" Internet Multimedia Services Architecture and application (IMSAA), 2010 IEEE 4th International Conference on, vol., no., pp.1-6, 15-17 Dec. 2010.

[5] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "*Improved proxy re-encryption schemes with applications to secure distributed storage*," in Proc. of NDSS'05, 2005.

[6] C. Hota, S. Sanka, M. Rajarajan, and S. Nair, "*Capability-Based Cryptographic Data Access Control in Cloud Computing*," Int. J. Advanced Networking and Applications Volume: 01 Issue: 01 Page: (2011).

[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "*Attribute-based encryption for fine-grained access control of encrypted data,*" Association for Computing Machinery, in Proc. of CCS'06, 2006.

[8] T. Mather, S. Kumaraswamy, and S. Latif, "*Cloud Security and Privacy*," O'Reilly Media, Sep. 2009.

[9] A. T. Velte, T. J. Velte, and R. Elsenpeter, "*Cloud computing a practical approach,*" Tata McGraw-Hill Edition, 2010, ISBN-13:978-0-07-068351-8.

[10] W. Stallings, "*Cryptography and network security*," LPE Forth Edition, ISBN-978-81-7758-774-6.

[11] G. Miklau, and D. Suciu, "*Controlling access to published data using cryptography*," in Proc. of 29th VLDB, Germany, Sept 2003.

[12] S. Yu, C. Wang, K. Ren, and W. Lou, "*Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing*," in Proc. of IEEE INFOCOM 2010, 2010.

[13] H. Zhong, and H. Zhen, "*An Efficient Authenticated Group Key Agreement Protocol*," Security Technology, 2007 41st Annual IEEE International Carnahan Conference on, vol., no., pp.250-254, 8-11 Oct. 2007.

[14] S. K. Harit, S. K. Saini, N. Tyagi, and K. K. Mishra, "*RSA Threshold Signature Based Node Eviction in Vehicular Ad Hoc Network,*" Information Technology Journal, 2012, ISSN 1812-5638, in Asian Network for Scientific Information.

[15] R. S. Fabry, "*Capability-Based Addressing*," in Communications of the ACM, 17(7), July 1974, pp. 403-412.

[16] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "*Over-encryption: Management of access control evolution on outsourced data*," in Proc. of VLDB'07, 2007.