A Framework for Secure Sharing of Hierarchical Data in Cloud

Deepa Maria Polson, Sabitha S, Dr. Rajasree M.S.

Abstract— Data Security is an important aspect in Cloud storage. Nowadays, there are numerous algorithms that ensure the security for the data placed in the Cloud storage.One way of modeling our data is to structure it in a hierarchical format and then upload it to the Cloud. This represent a less tedious way of organization and searching of our data becomes less cumbersome. Hierarchical structures can be used to model a plenty of situations like Role Based Access Control (RBAC) model, patient's medical records. We may want to share the data with others. This paper presents a study of various key management schemes used in hierarchical data models which focuses on key sharing. Also it provides a comparative study of different key management schemes. The key management schemes are divided into three sections: Key Derivation Schemes for predefined hierarchy, Compact-Sized key using symmetric encryption, and Key Management Schemes using Identity Based Encryption.

Index Terms— Hierarchical Data Models, Key Management, Secure Data Sharing, Cloud Storage.

1 Introduction

DATA sharing in Cloud storage is gaining popularity. Cloud Service providers offer Infrastructure as a Service (IaaS), which users have to pay per use only and which is very cheap. Hence, the use of cloud storage is increasing since the amount of data to be stored for each individual is increasing day by day in the current world. As the world is becoming more and more digital, more and more options have to be provided for the people to store data efficiently and securely, transmit the data securely in the networks, share the data securely. So in every aspect, whether the data be in motion or at the state of rest, Security is important for the data. The Cloud Service Providers like Amazon, Google are giving us proper security support that they encrypt the data using the schemes like AES [1] if and when needed and the hackers cannot decrypt our data. But in such a situation, we are trusting the cloud technicians, whom we would not know, with our personal and confidential data. This data may include our medical reports, office files, personal photos, videos etc.

The initial work for providing security while sharing was through access control mechanisms [3]. That is, when a user requests for a file, the cloud server will check whether the user have correct access permissions and grant the file only if the permissions are satisfied by the user. But this also had the issue of trusting the cloud technicians as mentioned before. So the solution was to encrypt the data and place it in cloud. The data owner can share the proper key with others with whom he/she wants to share the data. In this aspect the most straightforward method [1] will be to encrypt each file with a distinct key and share corresponding key (public key in case of asymmetric encryption and the same key in case of symmetric encryption).

But it would need more bandwidth and storage space to share the keys for each of the files. Now the other option would be to encrypt all keys with same key and share this. But this approach have a disadvantage that on reception of the decryption key users can decrypt all files encrypted with the key. The most viable option would be to share a compact key [1] with each users so that the recipients can decrypt only those portions which they have right to. Many works have been done in the literature for providing compact key for hierarchical structures.

- *Deepa Maria Polson is currently pursuing Master's degree program in Information Security in College of Engineering, Trivandrum, India, PH-09496460780. E-mail: deepamariapolson@gmail.com.*
- *Sabitha S is currently working as Associate Professor in College of Engineering, Trivandrum. E-mail: ssabithasureshkumar@gmail.com.*
- *Dr.Rajasree M.S. is currently Director of IIITMK, Trivandrum, India.*

1.1 Applications of hierarchical structures

1. RBAC models [2]: In RBAC models where each user is associated with a role, and the users can be modeled as a tree structure. For example, consider a college scenario there is a principal at the top of the institution then there are multiple departments headed by the HODs followed by the corresponding Staff Advisors and then the students of each semester. In this scenario there may be some files that can be viewed by all (say, notice about any events) and some files should not be accessed by students but can be accessed by all others (say, all mark details of the students).
2. Content Distribution [2]: Different users receive data in different quality and resolutions from satellites to which they have paid for. This is possible by dividing the complete users into a hierarchical structure and providing corresponding keys.
3. Cable Television [11]: In Direct to Home services, users receive the content to which they are subscribed to. But actually the base stations broadcast all the data. Then how do each user get only what they have subscribed to. This is done by representing the data in a hierarchical structures and providing suitable keys.
4. Project Development [2]: The different forms of information flow in the projects, positions in the

company, component at the managerial, development areas can be represented using hierarchical structures.
5. Defense in Depth: At each stage of intrusion defense there remains particular set of resources which can be efficiently represented using hierarchical structures.
6. Medical records [3]: The medical records can be divided into a hierarchical format one branch being about heart diseases, other branch being about allergies, and other being about prescriptions. Individuals must be able to share part of this files with others.
7. Personal files [1]: An individual may divide his/her files into personal and official files which may be again divided to photos, videos, music files or confidential documents, common spreadsheets etc.

In a nutshell, hierarchical structures can be used to model a plethora of situations ranging from business applications to personal applications [2].

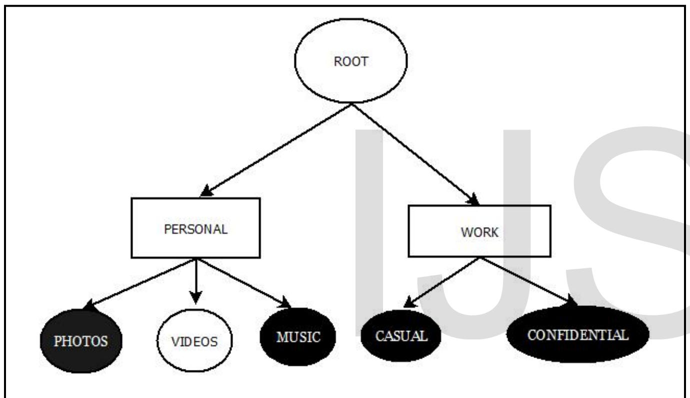


Fig. 1. Hierarchical Structures

Fig 1 shows hierarchical modeling of data. A person have some files which is classified into personal and work files. The personal files include images, videos, music files etc. Whereas the work related files include Confidential, Casual files. If the owner wants to share some of the files, say, Photos, Music, Confidential as well as Casual files, with another user, how the delegation process can effectively be done. This is the concern of key management in hierarchical structures. The works focusing on key management tries to effectively share keys such that it can only decipher what the user wants to share and also reduce communication and storage overheads.

Many works have been done about hierarchical access control through encryption. The rest of the paper is organized as follows. Section 2 presents a brief outline on key management schemes and its classification. Section 3 summarizes the various key management schemes proposed by various authors. Section 4 describes a key management scheme which is suitable for hierarchical structure which provide a fine grained access to the files. Section 5 provides a comparative study of all the works. Section 6 concludes the work.

2 Classification of Key Management Schemes - A Brief Overview

The works done for making cryptographic keys in hierarchical structures can be divided into three main categories [1].

1. Cryptographic schemes for predefined hierarchy.
2. Compact Key size in Symmetric-key encryption.
3. Compact key size in Identity based Encryption.

The works for key management in hierarchical structures was started by Akl and Taylor [6] in 1983 where they focused on providing access control in a hierarchical system. Later in 1988, R.S. Sandhu [10] proposed a method to generate a tree hierarchy of symmetric-keys by using repeated evaluation of pseudorandom functions on a fixed secret. Later in 1989, G.C. Chick and Stafford [7] extended this work of Akl and Taylor. In the first work proposed the keys of lower classes can be derived from the upper classes only. This was relaxed to create a master keys for classes those were not even hierarchically connected. This increased the flexibility and allows the system to control more services. In 2002, Tzeng [8] proposed a scheme in which not only the class in which the user belongs to, but also the time period in which the user belong to the class is considered and delegate the corresponding keys to the user depending on both the parameters. Each user holds some secret parameters which is independent of number of classes in the hierarchy and time period. In 2012, G. Ateneise et al [9] proposed another time bound key assignment scheme in hierarchical structures. This scheme was designed using bilinear maps and symmetric encryption schemes. This method also supported local changes that would change the public parameters but would not require private key re-distribution.

The above mentioned works didnot concentrate on providing a compact key size. In 2001, Benolah [12] proposed a key compression technique that can be used in digital finger printing as well as broadcast encryption. This work concentrated on a mechanism whereby the key set can be compressed into a single key whose size only depended on the security parameter. Later in 2009, Benolah et al [3] proposed a scheme for protecting medical records. This scheme proposed a patient controlled encryption scheme in which users can define an ontology of their medical reports and share a part of the key which the recipient can use for decryption and searching in the records. In 2009, Alomair et al [13] proposed a scheme which reduced the key size for achieving authentication in symmetric-key encryption. But the decryption scheme is not mentioned in this scheme and concentrates on authentication only.

Identity-based encryption was the first method which introduced public-key encryption in hierarchical structures. A fully functional identity based encryption scheme was published in 2002 by Boneh and Franklin [14]. The identity-based encryption would allow any arbitrary string to be the public-key of the user. There is a trusted PKG {Private Key Generator} which issues the corresponding private keys after validating the user using the identity. In 2005, Amit Sahai et al [15] extended the identity based encryption scheme to include biometrics as the identity of individual users. They used a fuzzy based system here. Later, a hierarchical IBE system was proposed by Gentry and Silverberg [17] which mirrors an organizational hierarchy. In 2007, F.Guo et al [16] proposed a scheme that dealt with having a compact key which would decrypt multiple ciphertexts. Then in 2008, F.Guo et al [4] extended the above scheme and proposed a scheme which provided a single key for multiple identities without random oracles.

3 Key Management in Hierarchical Structures

3.1 Key Derivation Schemes for predefined hierarchy

In 1983, Akl et al [6] proposed a scheme which concentrated on providing access control in a hierarchical system. A partially ordered set of users is constructed where a high level user can access the lower level user classes and vice-versa is not possible. In the primary solution, a centralized authority generate keys for users at a particular level and distributes the same. When a message at level m is encrypted and is sent, it is broadcasted as a pair (encrypted message x, level). Here the disadvantage is that higher level users would have to store a large number of keys because they rely on assumption that a higher level user can access all lower level classes. This work focused on this key management problem. This storage issue was solved in this work where each user have to store key of that level only and keys of its children can be derived from this key using one-way functions. This one-way functions used in this scheme involve DES encryption for totally ordered posets or a function composition for arbitrary posets.

In 1988, R.S.Sandhu [10] proposed a scheme which also concentrated on solving key management for hierarchical structures. The main problem for work proposed by Akl et al was that in order to add new branch to existing tree, all tree has to be re-keyed. The solution proposed in the work is also based on one-way functions. The scheme classifies the users into a rooted tree of security classes. The requirement is that users at high clearance level can read or create information with lower sensitivity level. Here $f_p(x)$ is a one-way function where p is the security parameter. An arbitrary key is assigned to the security class at the root. The key for the child of a security class i SC_i is computed as $K_j=f_{name}(SC_j(K_i))$. For each child of a security class in the tree, a different one-way function is used. To ensure that sizes of all the keys are same, the authors ensure that the names of security classes fit within the block size.

In 1989, G. C. Chick et al [7] extended the scheme proposed by Akl et al [6]. In this work, a master key is used which is a representation of all the keys. The security of the work lies in modular exponentiations. For the computation of the master keys, a small prime is assigned to each class. Then two values T and u_i is computed and using this values secret keys for each users is computed. The secret keys depends on u_i which is nothing but the product of primes of the subclasses of the concerned class. In this scheme, computation of a service key is feasible if and only if secret key is accessible using that master key. It is possible to add new services to the system, provided that new services do not depend on existing system. The system depends on a central authority, if it fails, the security of the complete system fails. The system can be made more secure by using a committee of members all of which forms a central authority. The main advantage of the system is providing a compact key size.

In 2002, W. G. Tzeng [8] proposed a scheme that manages keys for the members who have different access privileges and designs an integrated key graph which keeps the key details of all the users. This scheme presents a centralized key management method. The authors define a data group (DG) and service groups (SG) where DGs are the users that can access to a particular resource whereas SG are the users that are authorized to access exactly the same set of resources. Each users in each DG share a key. In this scheme, each node of the key tree is associated with a key. The root of the key tree is associated with the session key. Each leaf node is associated with user$’$s private key. The intermediate nodes have auxiliary keys. Each user stores his private key, session key and a set of auxiliary keys. This scheme doesnot consider the overlap that occurs in DG and hence makes inefficient use of keys. This work concentrates on Direct to Home subscriptions where users pay for different packages.

In 2004, Y.Sun [11] published a work that solves the problem of assigning cryptographic keys to a set of partially ordered classes so that cryptographic key of higher class can be used to derive the cryptographic key of a lower class. A basic and straightforward way to achieve a time-based access control is to require each user to memorize encryption keys assigned to all classes lower down in the hierarchy for each time period. In this work, cryptographic keys of a class are different for each time period. There is a central authority which generates and assigns keys to the hierarchy. A one way function H is used which outputs 56 bits or 128 bits depending on the encryption scheme (DES or AES). It chooses 4 primes and computes two values n_1 and n_2. CA randomly chooses e_1, e_2… e_m and computes d_1, d_2 … d_m which is the multiplicative inverse of e_i modulo n_1. Based on these values and a hash function it derives K_{It} which is the key for class I at time t. Using the public parameters and class is and time interval a user can recover the key for the id using modular exponentiation and Lucas function. Later in 2012, G. Ateneise [9] proposed a scheme to assign time-dependent encryption keys to a set of classes in a partially ordered hierarchy. In this work users are divided into a number of disjoint classes and a partially ordered hierarchy is defined between the users depending on the authority, position or power. This method assigns a private information $s_{v,T}$ for each class v and time sequence T and encryption key $K_{u,t}$ can be used by the users belonging to class u in the time period to protect the sensitive data by means of symmetric cryptosystems. This work also proposes a bilinear pairing method for deriving the encryption keys.

3.2 **Compact-Sized Keys in Symmetric Encryption**

The above proposed methods concentrated on providing cryptographic keys for predefined hierarchy. But this methods do have a deficiency that, as the number of classes to be shared increases, the number of keys to be increases and hence more bandwidth and more confidential storage space is needed. Thus later works were extended to provide a compact key size. Initially the work was done by Benolah [12] in April 2001. This work describes a mechanism whereby each keyset can effectively be compressed into a single key whose size is dependent only on the security parameters. The recovery of an individual key requires time proportional to the security parameter and the number of keys compressed for that subscriber, the additional time to compute more than one key is small and independent of the number of keys compressed. Generally, cryptographic keys are incompressible and distinct keys should be chosen independently. However, in many situations, computational independence rather than the information theoretic independence is considered to be more important. Key compression have a major application in broadcast scenario. Because the bandwidth required would be affected by the number of keys to be shared with the users. Key compression method involves associating a prime with each data set initially. For each subset of the data available to the user, product of the primes to which the user is not entitled is computed. Also, the key is derived using modular exponentiation using the primes associated with each datum. To recover a key, a subscriber needs to takes its compressed key set and raise it to product of all primes except those which are entitled to users.

In 2007, Atallah [2] proposed a dynamic key management scheme for access hierarchy. The work modelled the user population as a set of directed graph. A user with access privileges for a class obtains access to objects stored at that class and its descendants. Here, the problem of key management is to efficiently derive key for the child nodes from its parent nodes. The system makes two assumptions. One, they rely on a trusted server which generate and distribute keys. Second, the security of the model relies on pseudo random functions. Here the key derivation is done using a hash function. This method does not incorporate search ability. Later in 2009, Benolah proposed a scheme for providing compact key which also incorporated search ability. This work was explained for medical records. In this system, medical records are divided into hierarchical structure and keys for each section is given so that the recipient can decrypt that particular branch only. A root secret key is used from which the subkeys are derived. The patient selectively distribute subkeys for decryption. The work discusses two variants: Public key PCE and Symmetric key PCE. In public key PCE, the doctors or other persons can upload files to the records without knowing the corresponding decryption key; only some basic information are used. In Symmetric PCE, one has to know the decryption key in order to encrypt data. This scheme uses Hierarchical Identity Based Encryption in which an encryptor encrypt a file using tags.

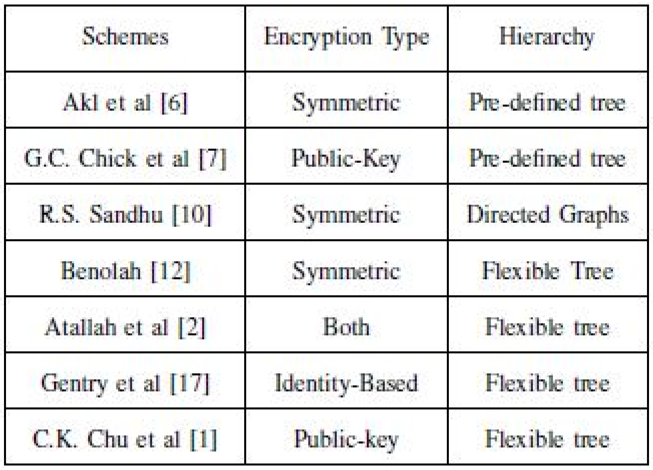
3.3 **Key Management Using Identity based Encryption**

Identity-based encryption is a type of public-key encryption in which the public-key of a user can be set as an identity-string of the user. The identity based encryption was proposed by Shamir [14] and the first fully functional model was developed by D.Boneh et al [14] in the year 2001. This IBE system is built from bilinear maps and the authors uses Weil pairing for bilinear maps. The scheme is specified by four randomized algorithms. One, Setup phase which takes a security parameter and returns system parameter and master key. Second phase is the Extract phase which takes as input system parameters and identity of the user and returns private key. Third phase is encryption which takes as input system parameters, identity and message M and outputs the ciphertext. The last phase is Decryption which takes system parameters, private key, ciphertext and outputs the decrypted message M. This scheme uses hash functions as well as XOR functions for encryption and decryption. Later Fuzzy-Based IBE systems were developed by Amit Sahai et al [15] in 2005 which was an extension of IBE. This system used biometrics as the identity of each user. The use of biometrics had advantages of being unique and inherent for each and every user. But there may occur some differences when biometrics is measured each time. Thus, instead of straightforward comparison a threshold difference is allowed. Hence “fuzzy” measurements are used. The number of exponentiations in the group to encrypt an identity will be linear with the number of elements in the identity’s description.

In the year 2002, a hierarchical IBE system was proposed by Gentry and Silverberg [17] which mirrors an organizational hierarchy. An identity at level k can issue private keys to its descendants but cannot decrypt those files for other identities. This work was later extended by F. Guo et al [16] in the year 2007. This IBE was developed to conveniently handle public keys which is the identities which are used as public keys. When a user possess ‘n’ identities the public key handling become messier. This work follows a hierarchical IBE by which users can derive keys from upper classes. Later in the year 2008, F.Guo et al [4] extended the above scheme and proposed a scheme which provided a single key for multiple identities without random oracles. This was done because the schemes without random oracles had more provable security.

The schemes discussed above focused on generating key that could be used for sharing data with other users. But the above proposed methods didnot discuss how to share data that have distinct root values in the tree. For eg, from Fig 1, how to effectively share a single key such that it would delegate the decryption rights for Photos and Videos. This key derivation is discussed in the following section.

TABLE 1
KEY MANAGEMENT SCHEMES



Schemes	Encryption Type	Hierarchy
Akl et al [6]	Symmetric	Pre-defined tree
G.C. Chick et al [7]	Public-Key	Pre-defined tree
R.S. Sandhu [10]	Symmetric	Directed Graphs
Benolah [12]	Symmetric	Flexible Tree
Atallah et al [2]	Both	Flexible tree
Gentry et al [17]	Identity-Based	Flexible tree
C.K. Chu et al [1]	Public-key	Flexible tree

4 Proposed Solution

We have seen that hierarchical structures can be used to model a myriad of situations in real life scenarios. The question that arises in the modeling of hierarchical structures is how to securely share the data with other users. As specified earlier, the security can be ensured using encryption and proper decryption key can be shared. The method of sharing the keys in such a way that it is space-efficient as well as it ensures security is the matter of concern in the hierarchical model arena. Many works have been proposed to effectively transmit the decryption key.

A key aggregate cryptosystem was published by C.K. Chu et al [1] in 2014 which provided a public-key cryptosystem which was based on ID allotted to files. It uses bilinear pairing for encryption as well as decryption. It keeps ciphertext size as well as key size constant. In this work, an aggregate key is computed based on the id allotted in each branch and a single key is delegated to each user which can decrypt corresponding id. But still there remains a problem that if we get a decryption key of a particular id, we can decrypt all the files encrypted with that id. This problem can be solved by associating policies with the files. During Encryption, we can associate the attributes of the user who can decrypt the file. Then we generate an aggregate key based on the id and attributes for each user. During decryption, we check whether the policies match and decryption key have the id of the file to be decrypted. If both are satisfied, only then the file is decrypted. Thus, a fine grained access can be provided through this method while providing an aggregate key as well. The access policies which determine who all can decrypt the files is included as a tree structure with the ciphertext. The attributes of the user are included in the aggregate key from which a particular secret is generated only using that key.

Figure 2 gives the overall architecture of this solution. In this solution, Data owner stores the data in the hierarchical model explained above. She encrypts the files using the id allotted to each file and the access policies and places it in the cloud. She also computes an aggregate key for the files and shares with other person. The other one receiving the key downloads the required file from cloud and attempts to decrypt it using the aggregate key $K_{1,2,3}$ (as shown in figure). During this process, the policies needed to be satisfied is validated against the policies in the aggregate key and the ids decryptable by the aggregate is checked and if both are satisfied, the file is decrypted.

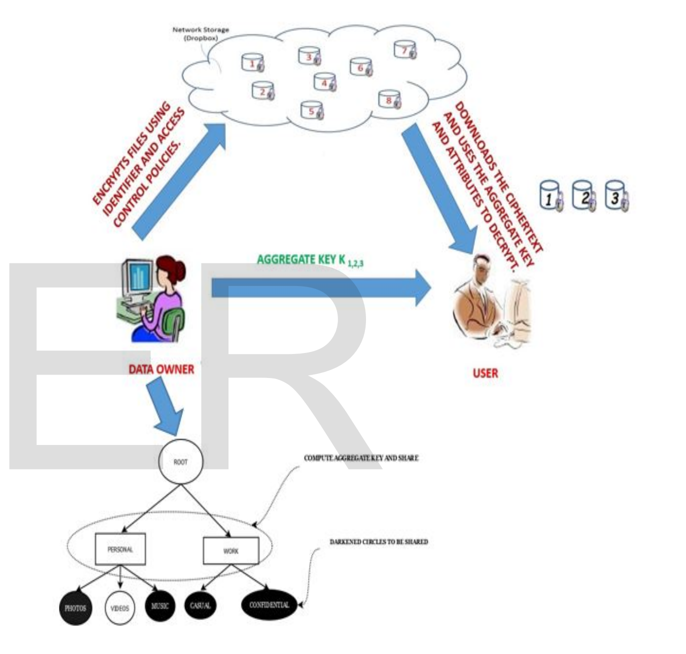

Network Storage (Dropbox)
1
2
3
4
5
6
7
8
ENCRYPTS FILES USING IDENTIFIER AND ACCESS CONTROL POLICIES.
DOWNLOADS THE CIPHERTEXT AND USES THE AGGREGATE KEY AND ATTRIBUTES TO DECRYPT.
AGGREGATE KEY K 1,2,3
DATA OWNER
USER
ROOT
COMPUTE AGGREGATE KEY AND SHARE
PERSONAL
WORK
DARKENED CIRCLES TO BE SHARED
PHOTOS
VIDEOS
MUSIC
CASUAL
CONFIDENTIAL

Fig. 2 Proposed Architecture

5 Comparative Study

The works proposed for the predefined hierarchy ([6],[7],[8],[9],[10],[11]) always assumed that if we get the key for the upper classes then lower classes must be decryptable by the user. This may not always be the case. Sometimes when a person wants to share a part of files with the upper class users and a different set of files to other users. Also those works did not aim at providing a compact key. Later the works done by Benolah and M.Chase et al ([3], [12], and [7]) could provide a compact key. But the schemes were symmetric.

Then came the Identity-Based encryption schemes which used the string as the public key. Later, in 2014, C.K.Chu [1] described a Key Aggregate Cryptosystem [KAC] which concentrated on giving a aggregate key which would decipher only those files for which the id of the file and the key matches. It allotted an id for each branch in the hierarchy. It had an advantage from above schemes that it used a public-key scheme and for sharing multiple files a single key had to be shared with the other user. Table 1 shows the comparative studies of different schemes used in key management for hierarchical structures.

6 Conclusion

Data Security is an important aspect that we have to deal in our day-to-day lives. This paper concentrated on security of data modeled as a hierarchy uploaded in Cloud. We focused on key management schemes that would allow efficient and secure sharing of data uploaded in Cloud with other users. Many works have been done by researchers for effective key derivation and delegation in the hierarchical structures. This paper reviewed on the works done by the researchers and also proposed a solution that would allow to share hierarchical data efficiently and securely. Also it provided a comparative study on various key management schemes and broadly classified the schemes into three categories which are a) Key Derivation Schemes for Pre-Defined hierarchy b) Schemes that provided a Constant-Sized key sing Symmetric key Encryption and c) Key Management Schemes which used Identity Based Encryption. The comparison focused on factors like type of encryption and hierarchy. Then a scheme is proposed which aims at providing an aggregate key which would reduce the communication overhead and at the same time ensure the security of the files uploaded. The fine-grained access is provided by considering the attributes of the user and specifying it while encryption and key generation.

7 References

[1] Cheng-Kang Chu, Sherman S.M. Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H. Deng, (2014), "Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage", IEEE Transactions On Parallel And Distributed Systems, Vol. 25, No. 2, pp 468-477.

[2] M.J. Atallah, M. Blanton, N. Fazio, and K.B. Frikken, (2009), “Dynamic and Efficient Key Management for Access Hierarchies,” ACM Trans. Information and System Security, vol. 12, no. 3, pp. 18:1-18:43.

[3] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, (2009), “Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records,” Proc. ACM Workshop Cloud Computing Security (CCSW ’09), pp. 103-114.

[4] F. Guo, Y. Mu, Z. Chen, and L. Xu, (2007), “Multi-Identity Single-Key Decryption without Random Oracles,” Proc. Information Security and Cryptology (Inscrypt ’07), vol. 4990, pp. 384-398.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, (2006), “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,” Proc. 13th ACM Conf. Computer and Comm. Security (CCS ’06), pp. 89-98.

[6] S.G. Akl and P.D. Taylor, (1983), “Cryptographic Solution to a Problem of Access Control in a Hierarchy,” ACM Trans. Computer Systems, vol. 1, no. 3, pp. 239-248.

[7] G.C. Chick and S.E. Tavares, (1989), “Flexible Access Control with Master Keys,” Proc. Advances in Cryptology (CRYPTO ’89), vol. 435, pp. 316-322.

[8] W.-G. Tzeng, (2002), “A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy,” IEEE Trans. Knowledge and Data Eng., vol. 14, no. 1, pp. 182-188.

[9] G. Ateniese, A.D. Santis, A.L. Ferrara, and B. Masucci, (2012), “Provably- Secure Time-Bound Hierarchical Key Assignment Schemes,” J. Cryptology, vol. 25, no. 2, pp. 243-270.

[10] R.S. Sandhu, (1988), “Cryptographic Implementation of a Tree Hierarchy for Access Control,” Information Processing Letters, vol. 27, no. 2, pp. 95-98.

[11] Y. Sun and K.J.R. Liu, (2004), “Scalable Hierarchical Access Control in Secure Group Communications,” Proc. IEEE INFOCOM ’04 pp. 1296-1306.

[12] J. Benaloh, (2009), “Key Compression and Its Application to Digital Fingerprinting,” Technical report, Microsoft Research.

[13] B. Alomair and R. Poovendran, (2009), “Information Theoretically Secure Encryption with Almost Free Authentication”, J. Universal Computer Science, vol. 15, no. 15, pp. 2937-2956.

[14] D. Boneh and M.K. Franklin, (2001), “Identity-Based Encryption from the Weil Pairing,” Proc. Advances in Cryptology (CRYPTO ’01), vol. 2139, pp. 213-229.

[15] Sahai and B. Waters, (2005), “Fuzzy Identity-Based Encryption,” Proc. 22nd Int’l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT ’05), vol. 3494, pp. 457-473.

[16] F. Guo, Y. Mu, and Z. Chen, (2007), “Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key,” Proc. Pairing-Based Cryptography Conf. (Pairing ’07), vol. 4575, pp. 392-406.

[17] Gentry, C., Silverberg, (2002), “Hierarchical ID-based cryptography” In: Zheng, Y., Zheng, Y. (eds.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548-566.