ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Public Auditing of Dynamic Data Sharing in Cloud Storage Using Aggregate Cryptosystem

Dr.L.Sankari, R.Keerthana

Associate Professor, Department of Computer Science, Sri Ramakrishna College of Arts & Science for Women Coimbatore, India.

Research Scholar, Department of Computer Science, Sri Ramakrishna College of Arts & Science for Women Coimbatore, India.

ABSTRACT: Cloud computing technology is widely used so that the data can be outsourced on cloud can accessed easily. Different members can share that data through different virtual machines but present on single physical machine. The need is to share data securely among users. The cloud service provider and users authentication is necessary to make sure no loss or leak of users data. Privacy preserving in cloud is important make sure the users identity is not revealed to everyone. On cloud anyone can share data as much they want to i.e. only selected content can be shared. Cryptography helps the data owner to share the data in a safe way. So user encrypts data and uploads on server. Different encryption and decryption keys are generated for different data. The encryption and decryption keys may be different for different set of data. Only those set of decryption keys are shared that the selected data can be decrypted. Here a public-key cryptosystems which generate a ciphertext which is of constant size. In this work, a special type of public-key encryption which calls Advanced Encryption Standard (AES) is proposed. In AES, user's encrypt a message not only under a public-key, but also under an identifier of ciphertext. The key owner holds a secret key called master-secret key. More important the extracted key have can be an aggregate key which is as compact as a secret key for a single class but aggregates the power of many such keys. And also the secured cryptographic techniques such as Rivest Shamir Adleman (RSA) and HASH algorithms are used to secure the files in the cloud.

KEYWORDS: cloud computing, cryptography, encryption, decryption, Advanced Encryption Standard, RSA, hash, public key, aggregate key, security.

I. INTRODUCTION

Storing data on cloud is gaining popularity recently. In enterprise, we see the increase in demand for data outsourcing, which assists in the planned management of business data. It is also used as a basic technology behind many online services for personal applications. Now, it is easy to apply for free accounts

for email, photo album, file sharing and/or remote access, with storage size more than 25GB.Together with the current wireless technology, users can retrieve almost all of their files and emails by a cell phone in any side of the world. data Confidentiality, a traditional way to ensure it is to rely on the server to enforce the access control after authentication, which means any unpredicted privilege rise will expose all data. Data from different clients can be present on separate virtual machines but reside on a single physical machine. Data in a destination VM could be stolen by instantiating another VM co-resident with the destination one. Regarding availability of files, there are a number of cryptographic schemes which go as far as allowing a third-person auditor to check the availability of files on behalf of the sender without leaking anything about the data, or without compromising the data owners secrecy. Likewise, cloud users possibly will not hold the strong conviction that the cloud server is doing a good job in terms of secrecy. A cryptographic solution, with Stated security relied on number theoretic assumptions is more attractive whenever the useris not perfectly happy with trusting the security of the Virtual Machine or the honesty of the technical member. These users are encouraged to encrypt their files with their own keys before uploading them on to the cloud. Sharing of data is an vital functionality in cloud storage.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

The demanding problem is how to effectively share cipher text. So the users can download the cipher text from the storage, decrypt them, then upload them on to the cloud for sharing, but it loses the value of cloud computing. Users should be able to hand over the access rights of the sharing data to others so that they can access these data from the cloud directly. Below we will take Dropbox as an example. Assume that 'A' puts all her private data on Dropbox, and she does not want to leak her photos to everyone. Due to various data leakage possibility A cannot feel comfortable by just relying on the privacy protection mechanisms provided by Dropbox, so she encrypts all the data using her own keys before uploading. Now, A's friend, B, told her to share the data taken over all these years which B appeared in. A can then use the share function of Dropbox, but the problem is how to hand over the decryption rights for these data to B. The possible option A can choose is to firmly send B the secret keys involved. obviously, there are two great ways for her under the traditional encryption concept: A encrypts all files with a one encryption key and gives B the corresponding private key directly. A encrypts files with Different keys and sends B the corresponding private keys. Clearly, the first option is not efficient since all unselected data may be also leaked to B. For the second option, there are practical concerns on efficiency. The number of such keys is as many as the number of the shared data, say, a Hundred. Transferring these private keys naturally requires a secure medium, and storing these keys requires rather costly secure storage. The costs and complexities involved generally boost with the number of the decryption keys to be shared. In short, it is very deep and costly to do that. Encryption keys also come with two flavors — symmetric key or asymmetric (public) key. Using symmetric encryption, when A wants the data to be originated from a third person, she has to give the encryptor of her key; clearly, this is not always desirable. By dissimilarity, the encryption and decryption key are different in public-key encryption. The use of public-key encryption gives more flexibility for our system. For example in business settings, every employee can upload ciphered data on the cloud storage server without the knowledge of the company's secret key. Therefore, the best solution for the above problem is that A encrypts files with different public-keys, but only sends B a one decryption key. Since the decryption key should be sent via a secure medium and kept secret, small key size is always useful.

In this paper, AES is proposed for encrypt a user's message not only under a public-key, but also under an identifier of ciphertext called class. That means the ciphertexts are further categorized into different classes. The key owner holds a master-secret called master-secret key, which can be used to extract secret keys for different classes. More importantly, the extracted key have can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys, i.e., the decryption power for any subset of ciphertext classes. And also the secured cryptographic techniques such as RSA and HASH algorithms are used to secure the decrypted messages from unauthorized users.

II. LITERATURE SURVEY

In this section, the various key assignment schemes are compared with the basic KAC algorithm for sharing in secure cloud storage.

In [1] presented new cryptosystem, cipher text is labelled by the encryptor with a set of descriptive attributes. Each private key is associated with an access structure that specifies which type of cipher texts the key can decrypt. We call such a scheme a Key-Policy Attribute-Based Encryption (KP-ABE), since the access structure is specified in the private key, while the cipher texts are simply labelled with a set of descriptive attributes. A user is able to decrypt a cipher text if the attributes associated with a cipher text satisfy the key's access structure. This setting is reminiscent of secret sharing schemes. Secret-sharing schemes (SSS) are used to divide a secret among a number of parties. The information given to a party is called the share (of the secret) for that party.
In [2] Proxy re-encryption allows a proxy to transform a cipher text computed under Alice's public key into one that can be opened by Bob's secret key. There are many useful applications of this primitive. It presented several efficient proxies re-encryption schemes that offer security improvements over earlier approaches. The primary advantage of our schemes is that they are unidirectional (i.e., Alice can delegate to Bob without Bob having to delegate to her) and do not require delegators to reveal all of their secret key to anyone – or even interact with the delegatee – in order to allow a proxy to re-encrypt their cipher texts. In this scheme, only a limited amount of trust is placed in the proxy.
In [3], proposed an aggregate signature, it presents security models for such signatures, and gives several applications for aggregate signatures. Aggregate signatures are useful for reducing the size of certificate chains (by aggregating all signatures in the chain) and for reducing message size in secure routing protocols such as SBGP. An aggregate

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

signature scheme enables us to achieve precisely this type of compression. Suppose each of n users has a public-private key pair (PK_i, SK_i). User ui signs message M_i to obtain a signature σ_i. Then there is a public aggregation algorithm that takes as input all of $\sigma_1 \ldots \sigma_n$ and outputs a short compressed signature σ.
In [4], decentralizing attribute-based encryption proposes a Multi-Authority Attribute-Based Encryption (ABE) system. In this system, any party can become an authority and there is no requirement for any global coordination other than the creation of an initial set of common reference parameters. A party can simply act as an ABE authority by creating a public key and issuing private keys to different users that react their attributes. A user can encrypt data in terms of any Boolean formula over attributes issued from any chosen set of authorities. Finally, the system does not require any central authority. Spreading a central authority's keys over several machines to alleviate performance pressures might simultaneously increase the risk of key exposure.

Cryptographic Keys for a Predefined Hierarchy: We start by discussing the most relevant study in the literature of cryptography/security. Cryptographic key assignment schemes (e.g., [5]) aim to minimize the expense in storing and managing secret keys for general cryptographic use. Data management is the important aspects of cloud. So this area must be checked clearly. Securing huge amount of data is very much risky. So our proposed scheme is made to solve all these problems. The below tabular column depicts our scheme efficiency with other various schemes.
Utilizing a tree structure, a key for a given branch can be used to derive the keys of its decreasing nodes. Just giving the parent key implicitly grants all the keys of its descendant nodes. Sandhu [6] proposed a method to generate a tree hierarchy of symmetric keys by using repeated evaluations of pseudorandom function/ block-cipher on a fixed secret. The concept can be generalized from a tree to a graph. More advanced cryptographic key assignment schemes support access policy that can be modeled by an acyclic graph or a cyclic graph [7], [8]. Most of these schemes produce keys for symmetric-key cryptosystems, even though the key derivations may require modular arithmetic as used in public-key cryptosystems, which are generally more expensive than “symmetric-key operations” such as pseudorandom function.

Sharing data or information among users is an important functionality in cloud storage. In [9] author Proposed new public-key cryptosystems that produce constant-size ciphertexts such that efficient delegations of decryption rights for any set of ciphertexts are possible. The one can aggregate any set of secret keys and make them as compact as a single key, but the power of all the keys being aggregated. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. More important is that the extracted key can be an aggregate key which is as condense as a secret key for a single class, but combines the power of many such keys that is the decryption power for any subset of ciphertext classes. Implementation of the KAC system in C with the pairing-based cryptography (PBC) Library.
In [10] authors specialize in a privacy-preserving public auditing system for information storage security in cloud computing. privacy-preserving public auditing theme uses the homomorphic linear critic and random masking to ensure that the third -party administrator wouldn't learn any data regarding the information content hold on the cloud server throughout the economical auditing method, that not solely eliminates the burden of cloud user from the tedious and presumably costly auditing task, however conjointly overcome the users worry of their externalization information outpouring.

In [11] authors provide a framework for Provable Data Possession (PDP). A PDP protocol checks that associate degree outsourced storage website retains a file that consists of f blocks. The consumer C (data owner) preprocesses the file, generating a tiny low piece of data that's keep domestically, transmits the file to the server S, and should delete its native copy. The server stores the file and responds to challenges issued by the consumer. Storage at the server is Ω (f) and storage at the consumer is O (1), orthodox to our notion of associate degree outsourced storage relationship. As a part of preprocessing, the consumer could alter the file to be kept at the server. The consumer could encipher, encode, or expand the file, or could embody further data to be keep at the server. Before deleting its native copy of the file, the consumer could execute a data possession challenge to for m certain the server has with success keep the file.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

III. PROPOSED METHODOLOGY

In this paper, a special type of public-key encryption is proposed which it is called as key-aggregate cryptosystem (KAC). In KAC, users encrypt a message not only under a public-key, but also under an identifier of ciphertext called class. That means the ciphertexts are further categorized into different classes. The key owner holds a master-secret called master-secret key, which can be used to extract secret keys for different classes. More importantly, the extracted key have can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys, i.e., the decryption power for any subset of ciphertext classes. The proposed modules are discussed in given below.

A. Cloud Storage

Cloud Storage is a model of networked computer data storage where data is stored on multiple virtual servers, generally hosted by third parties, rather than being hosted on dedicated servers. Hosting companies operate large data centers; and people who require their data to be hosted buy or lease storage capacity from them and use it for their storage needs. The data center operators, in the background, virtualize the resources according to the requirements of the customer and expose them as virtual servers, which the customers can themselves manage. Physically, the resource may span across multiple servers.

B. Data Control System

In our scheme, a data encryptor specifies an access structure for a cipher text which is referred to as the cipher text policy. Only users with decryption keys whose associated attributes, specified in their key structures, satisfy the access structure can decrypt the cipher text. CP-ABE with verifiable outsourced decryption. The same approach applies to KP-ABE with verifiable outsourced decryption. To assess the performance of our ABE scheme with verifiable outsourced decryption, we implement the CP-ABE scheme with verifiable outsourced decryption and conduct experiments on both an ARM-based mobile device and an Intel-core personal computer to model a mobile user and a proxy, respectively.

C. Key Exchange

Alice encrypts files with distinct public-keys, but only sends Bob a single (constant-size) decryption key. Since the decryption key should be sent via a secure channel and kept secret, small key size is always desirable. For example, we cannot expect large storage for decryption keys in the resource-constraint devices like smart phones, smart cards, or wireless sensor nodes. Especially, these secret keys are usually stored in the tamper-proof memory, which is relatively expensive.

D. Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) [12] may be a symmetric-key square figure calculation and U.S. government ordinary for secure and ordered encoding and unraveling. In December 2001, the National Institute of Standards (NIST) affirmed the AES as Federal experimental control Standards Publication (FIPS PUB) 197, which points out application of the Rijndael calculation to all or any touchy ordered information. After a compelling assessment, the Rijndael configuration, made by two Belgian cryptographers, was the last decision. The AES supplanted the DES with new and upgraded gimmicks:

- Block encryption usage
- 128-bit bunch encryption with 128, 192 and 256-bit key lengths.
- 20-30 years for data security.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

E. Secured RSA and Hashing Algorithm

1. RSA algorithm

RSA [13] is an algorithm for public-key cryptography, involves a public key and a nonpublic key. The overall public keys are regularly known to everybody and are utilized for scrambling messages. Messages encoded with the overall population key will exclusively be unscrambled abuse the particular key. Client information incorporates encryption before capacity, client verification methodology before capacity or recovery, and building secure channels for information transmission. RSA crypto framework understand the properties of the multiplicative Homomorphic encryption. Ronald Rivest, Adi Shamir and Leonard Adleman have imagined the RSA calculation and named after its creators. RSA utilizes measured exponential for encryption and decoding. RSA utilizes two examples, a and b, where a is public and b is private. Let the plaintext is P and C is cipher text, then at encryption C=P mod n and at decryption side P = C mod n n is a very large number, created during key generation process. n is a very large number, created during key generation process. It is used to checks third parties are interfere are opened the messages.

RSA key generation steps

1. Choose 2 primes call them p, q.
2. Multiply them call product n.
3. Multiply their "predecessors" (p-1,q-1) call product φ
4. Pick some integer call it e – between 1 and φ(exclusive) –sharing no prime factor with φ.
5. Find the integer (there's only one) that call it d – times e divided by φ leaves 1 then your keys are: – public: e together with n (e is for "encryption") – private: d together with n (d is for "decryption").

Encrypting with public key {e,n} (c = me mod n)

1. Choose a clear text message call it m – in the form of a number less than n
2. Raise it to power e
3. Divide that by n call remainder c then your cipher text result is c

Decrypting with private key {d,n} (m = c d mod n)

1. Take cipher text c
2. Raise it to power d
3. Divide that by n call remainder r then your recovered result is r is identically the original clear text message m

2. HASH algorithm

There are several similarities in the evolution of hash function and that of symmetric block ciphers. We have seen that the increasing power of brute-force attacks and advances in cryptanalysis have led to the decline in the popularity of DES and in the design of newer algorithm with longer key lengths and with features designed to resist specific cryptanalytic attacks. Similarly, advances in computing power and hash function cryptanalysis have led to the decline in the popularity of first MD4 and then MD5, two very popular hash functions. In response, newer hash algorithm have been developed with longer hash code length and with features designed to resist specific cryptanalytic attacks. Another point of similarity is the reluctance to depart from a proven structure.

AES is based on the Feistel cipher, which in turn is based on the Substitution-permutation network proposal of Shannon. Many important subsequent block ciphers follow the feistel design because the design can be adapted to resist newly discovered cryptanalytic threats. If, instead, an entirely new design were used for a symmetric block cipher, there would be concern that the structure itself opened up new avenues of attack not yet thought of. Similarly, most important modern hash functions follow the basic structure. This has proved to be a fundamentally sound structure and newer designs simply refine the structure and add to the hash code length. MD5, SHA-1, and RIPEMD- 160. We then look at an internet-standard message authentication code. A hash function H is a transformation that takes a variable-size input m and returns a fixed-size string, which is called the hash value h (that is, h = H(m)). Hash functions with just this property have a variety of general computational uses, but when employed in cryptography the hash functions are usually chosen to have some additional properties.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

A key aggregate encryption has five polynomial-time algorithms as

Setup Phase

The data owner executes the setup phase for an account on server which is not trusted. The setup algorithm only takes implicit security parameter.

KeyGen Phase

This phase is executed by data owner to generate the public or the master key pair (pk, msk).

Encrypt Phase

This phase is executed by anyone who wants to send the encrypted data. Encrypt (pk, m, i), the encryption algorithm takes input as public parameters pk, a message m, and i denoting ciphertext class. The algorithm encrypts message m and produces a ciphertext C such that only a user that has a set of attributes that satisfies the access structure is able to decrypt the message.

1. Input= public key pk, an index i, and message m
2. Output = ciphertext C.

Extract Phase

This is executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate.

1. Input = master-secret key mk and a set S of indices corresponding to different classes
2. Outputs = aggregate key for set S denoted by kS.

Decrypt Phase

This is executed by the candidate who has the decryption authorities. Decrypt (kS, S, i, C), the decryption algorithm takes input as public parameters pk, a ciphertext C, i denoting ciphertext classes for a set S of attributes.

1. Input = kS and the set S, where index i = ciphertext class
2. Outputs = m if i element of S

IV. RESULTS AND DISCUSSION

KEY GENERATION TIME

Figure 1 illustrates the key generation time chart. It shows that the time required for generation of key using standard key aggregate algorithm and the proposed AES algorithm. Compared to both the proposed algorithm key generation time is lesser than the existing algorithm. A package is been developed to calculate the key generation time using the formula. And the time is measured in milliseconds.

Key Gen Time = Aggregate Key -- File Uploaded Time

Where
File Uploaded Time is time taken for uploading any file.
Aggregate key is the key generated by combining the key of other files.
Example
Existing = 200 – 110 = 90
Proposed= 150 – 140 = 10

The above example shows the values that received during the time calculation of the time taken to generate a key.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

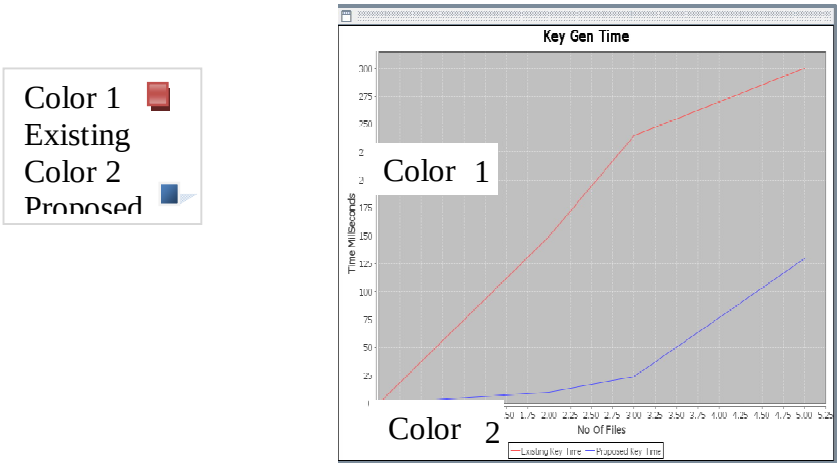

Fig 1: key gen time comparison

Table 1.1 Time comparison of key generation

S.NO	Existing (Milli Sec)	Proposed (Milli Sec)	NO OF FILES
1	90	10	2
2	230	25	3
3	265	75	4
4	300	130	5

The encryption time is used to calculate the time taken to encrypt the files. Figure 2 illustrates the encrypting time comparison. It is clear that the comparison of time required for encryption between standard key aggregate algorithm and the proposed AES algorithm. Compared to than existing algorithm the proposed algorithm encryption time is less. It is calculated using the formula.

Encryption Time = Cipher text - Uploaded file

Where

Cipher text is the file than is been encrypted.

File Uploaded Time is time taken for uploading any file.

Example

Existing= 250 – 500 = 250

Proposed = 500 – 360 = 140

The table below contains the value of the graph by comparing the existing and the proposed. And in the graph the red color of line represents the existing system and the blue line represents the proposed system. The encryption time is been calculated by the time of uploading of file and the time the cipher text is received.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

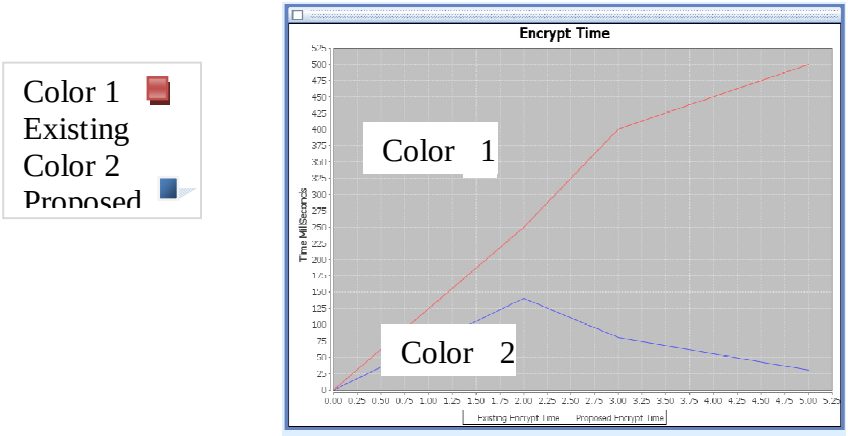

Fig 2: encryption time comparison

Table 5.2 Time comparison of Encryption

S.NO	Existing (Milli Sec)	Proposed (Milli Sec)	NO OF FILES
1	250	140	2
2	400	85	3
3	450	55	4
4	500	30	5

V. CONCLUSION AND FUTURE WORK

To share data flexibly is vital thing in cloud computing. Users prefer to upload there data on cloud and among different users. Outsourcing of data to server may lead to leak the private data of user to everyone. Encryption is a one solution which provides to share selected data with desired candidate. Sharing of decryption keys in secure way plays important role. Public-key cryptosystems provides delegation of secret keys for different ciphertext classes in cloud storage. The delegate gets securely an aggregate key of constant size. In AES, users encrypt a message not only under a public-key, but also under an identifier of ciphertext called class. And also the secured cryptographic techniques such as Ronald Rivest, Adi Shamir and Leonard Adleman (RSA) and HASH algorithms are used to secure the decrypted messages from unauthorized users. The experimental results show that formal security analysis of proposed schemes in the standard model.In cloud storage, the number of cipher texts usually grows rapidly without any restrictions. So we have to reserve enough cipher text classes for the future extension. Otherwise, we need to expand the public-key. Although the parameter can be downloaded with cipher texts, it would be better if its size is independent of the maximum number of cipher text classes.

REFERENCES

1. V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,” Proc. 13th ACM Conf. Computer and Comm. Security (CCS ’06), pp. 89-98, 2006.
2. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage,” ACM Trans. Information and System Security, vol. 9, no. 1, pp. 1-30, 2006.
3. D. Boneh, C. Gentry, B. Lynn, and H. Shacham, “Aggregate and Verifiably Encrypted Signatures from Bilinear Maps,” Proc. 22nd Int’l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT ’03), pp. 416-432, 2003.
4. B. Lewko and B. Waters (2011) “Decentralizing attribute-based encryption” in Proc. EUROCRYPT, pp. 568–588.
5. S. G. Akl and P. D. Taylor, “Cryptographic Solution to a Problem of Access Control in a Hierarchy,” ACM Transactions on Computer Systems (TOCS), vol. 1, no. 3, pp. 239–248, 1983.
6. R. S. Sandhu, “Cryptographic Implementation of a Tree Hierarchy for Access Control,” Information Processing Letters, vol. 27, no. 2, pp. 95–98, 1988.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

7. Y. Sun and K. J. R. Liu, "Scalable Hierarchical Access Control in Secure Group Communications," in Proceedings of the 23th IEEE International Conference on Computer Communications (INFOCOM '04). IEEE, 2004.
8. Q. Zhang and Y. Wang, "A Centralized Key Management Scheme for Hierarchical Access Control," in Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '04). IEEE, 2004, pp. 2067– 2071.