ISSN 2395-695X (Print)
ISSN 2395-695X (Online)

International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Issue 4, April 2016

BIG DATA ACCESS CONTROL IN THE CLOUD USING HIGHLY PROTECTED AND VALID POLICY UPDATE

R.Karthikeyan [1], G.Brindha Roselin[2]
HOD, CSE Department, Mohamed Sathak Engineering College, Kilakkarai, India [1]
ME(Student), CSE Department, Mohamed Sathak Engineering College, Kilakkarai, India [2]

***Abstract*- Because of large volume and velocity of big data, it is a proficient option to store big data in the cloud. A system's privacy and security controls are more likely to be compromised due to the misconfiguration of access control failure of cryptographic primitives or protocols. A trivial implementation is to let the data owners to get the data and re-encrypt it using the newly found access policy then store it back in the cloud. We also have proposed a novel scheme to perform the encryption to provide security and protection during access using the ABE with ElGamal Algorithm using the latest encryption technique known as Swiss army knife of cryptography. This method can avoid the retransmissions and loss of the data and reduce the work of the data owners in computation and communication. The crucial point is that all the transformations should be done without revealing the secret key until it reaches the correct recipient. A user can also use the outsourced policy updating algorithms for various types of access policies. Finally we propose an efficient and a secure method for checking the ciphertexts that are updated by the cloud server is correct. The analysis shows that our policy updating scheme is secure and and competent.**

Keywords— Policy updating, Outsourced, Access Control, ABE, Big Data and Cloud(key words)

I. INTRODUCTION

The term big data refers to the massive amount of digital information which many companies and Government collect about various fields, persons and their surroundings. It can also be referred to as high volume, high velocity and high variety of information. Security and Privacy issues are magnified by velocity, volume and variety of bigdata such as large scale cloud infrastructures, diversity of data sources and formats, streaming nature of data acquisition and high volume inter-cloud migration. An efficient option is to store the big data in the cloud as the cloud has capabilities of storing big data and processing high volume of user requests in an economic way. When hosting big data into the cloud, the data security becomes a major issue as cloud servers cannot be fully trusted by data owners.

This paper uses the ElGamal algorithm[8] with Swiss army knife of cryptography technique[23] to ensure data security in the system similar to it is shown in the Fig.1. An asymmetric key encryption algorithm that uses a pair of different cryptographic keys to encrypt and decrypt is used. It also allows the data owners to define access policies and encrypt the data under each policy such that only the users whose attributes that match with the corresponding policies will get access to the data to decrypt it. When more and more organizations and enterprises outsource data into the cloud, the policy updating becomes an important concern as data access policies may be changed dynamically and frequently by data owners. Also, this issue has not been considered in any of the existing attribute-based access control schemes [11]-[13][21].

The previous papers using ABE systems for the policy update suffers from various issues because once the data owner outsourced the data into the cloud, it would not keep a copy in local systems. When the data owner wants to change the access policy it must again transfer the data to the old site from cloud, re-encryption takes place then after that it moves back to the cloud server. That has lead to a high communication overhead and heavy computation burden on the data owners. In order to overcome these problems we have developed new method to outsource the job of policy updating to the cloud server.

The challenge of outsourcing policy updating to the cloud must provide the following

1) *Correctness:* Only those users who own the essential attributes must be able to decrypt the data encrypted under the new access policy by running the original decryption algorithm.
2) *Completeness:* The system must be ready to update and transform to any type of access policies.
3) *Security:* There should not be any damages to the existing security of access control scheme or introduce any new security problems.

The policy updating problem has been discussed in the key policy structure [2] and ciphertext-policy structure [21]. However, these methods cannot satisfy the completeness requirement and security requirement either. In this paper, we come up with the approaches in providing security also focusing on the policy updating problems in the ABE systems and propose a highly protected and valid policy updating outsourcing method. We ought a system that can scale to handle a large number of data and process massive amount of data. Local computers no longer have to take the entire burden when it comes to running applications. Cloud is used to minimize the usage cost of the computing resources.

We let the cloud server to update the policies of the encrypted data directly, which means that cloud server does not need to decrypt the data before / during the policy updating. We also provide a secure policy checking method

that enables data owners to check whether the ciphertexts have been updated by the cloud server correctly.

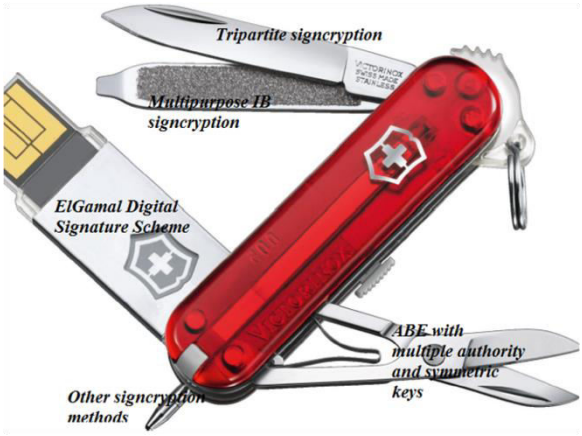

Fig.1.*Example for Swiss Army Knife of Cryptography*

The contributions of this paper include:

1) It offers access control to the data owners using ElGamal encryption algorithm and expand a new method to outsource the policy updating to the server.
2) It deals with the designs of policy updating algorithms for different types of access policies, e.g., Boolean Formulas, LSSS Structure.
3) Implementation of the Swiss army knife of cryptography technique will ensure more secure and valid access control schemes.

In this paper we do not require any help from the data users and owners to check the correctness of the ciphertext updating by their own secret keys and checking keys issued by each authority. More over we discuss some of the key features of the access control scheme based on identity and how it will be suitable for big data access control in the cloud. In addition we also add more performance evaluation on policy updating algorithms and the policy checking methods with encryption algorithms.

An asymmetric key encryption algorithm that uses a pair of different cryptographic keys to encrypt and decrypt is used. ElGamal abridged the Diffie-Hellman key exchange algorithm by introducing a random exponent of receiving entity [7]. Due to this simplification the algorithm can be used to encrypt in one direction without the necessity of the second party to take actively part. The key innovation here is that the use of Swiss army knife technique that combines the utilities of various encryption methods such as *ElGamal Digital Signature, Tripartite signcryption, and Multipurpose Identity based signcryption* in one single method.

II. SYSTEM AND SECURITY MODEL

A. System Model

We consider a cloud storage system with multiple authorities,as shown in Fig.2. The system model consists of the followingentities: authorities (AA), cloud server (server), data owners(owners) and data consumers (users).

Authority. The Authority delimits that, power is delegated formally. It includes the right to command a situation, commit resources and give orders. In previous works every authority is dependent with each other and is responsible for managing attributes of users in its own area. Here the secret key / public key pair is generated for each attribute in its domain and follows to generate the secret key for each user according to their attributes.

Cloud server. The A cloud server is a logical server that is built, compared and provided through a cloud computing platform over the Internet. Cloud servers possess and show similar capabilities and functionality to a typical server but are accessed remotely from a cloud service provider. A cloud server may also be called a virtual server or virtual private server. The cloud server stores the data for data owners and allows the data owners to access service to the users. The server is also in charge for updating cipher texts from old access policies to new access policies.

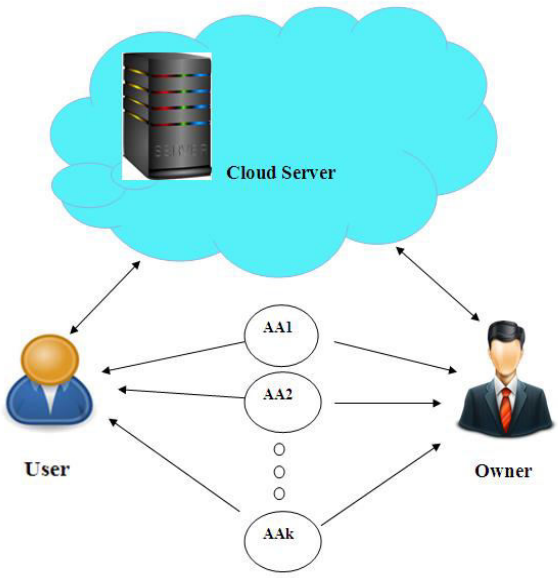

Fig. 2*. System Model*

Data owners. There are other concerns with regards to storing data in the cloud such as backups, data security, privacy and transfer of data. So despite the advantages of cloud services an enterprise must answer the most crucial question when going for any cloud hosted service, that is ‘who owns the data’. The actual ownership of data in the cloud may be reliant on the nature of data stored as well as where it was created. The data owners define access policies and encrypt data under these policies before hosting them in the cloud.

They also ask the server to update access policies of the encrypted data stored in the cloud. After that, they will check whether the server has updated the policies correctly.

Data Users. Each user is assigned with a global user identity with set of their corresponding attributes and can freely get the cipher texts from the server. The user can decrypt the cipher text, only when its attributes satisfy the access policy defined in the cipher text.

B Framework

To accomplish all the requirements of policy updating[1], the framework of our access control scheme must be as follows.

Definition 1 (Framework).*Our dynamic policy access control scheme is a collection of the following algorithms:* GlobalSetup, AuthoritySetup, SKeyGen, Encrypt, Decrypt, UKeyGen *and* CTUpdate.

- **GlobalSetup**(λ)$\rightarrow$GP.The global setup algorithm takesno input other than the implicit security parameter λ. It outputs the global parameter GP for the system.
- **AuthoritySetup**(GP,AID)$\rightarrow$(SK,PK). The authority setup algorithm is run by each authority AID with GP and the authority identity AID as inputs and its secret/public key pair (SK_{AID},PK_{AID}) as outputs.
- **SKeyGen**(GID,GP,$S_{GID,AID}$,SK_{AID})$\rightarrow$$SK_{GID,AID}$. Each authority AID runs the secret key generation algorithm to generate a secret key $SK_{GID,\ AID}$ for user GID. It takes as inputs the global identity GID, the global parameter GP, a set of attributes $S_{GID,\ AID}$ issued by this authority AID and the secret key SK_{AID} of this authority. It outputs a secret key $SK_{GID,AID}$ for this user GID.
- **Encrypt**({PK},GP,m,A)$\rightarrow$CT. The encryption algorithm takes as inputs a set of public keys {PK} of relevant authorities, the global parameter GP, the message m and an access policy A. It outputs a ciphertext CT.
- **Decrypt**(CT,GP,{$SK_{GID,AID}$})$\rightarrow$m. The decryption algorithm takes as inputs the ciphertext, the global parameter GP and a collection of secret keys from relevant authorities for user GID. It outputs the message m when the user's attributes satisfy the access policy associated with the ciphertext. Otherwise, the decryption fails.
- **UKeyGen**({PK},EnInfo(m),A,A')$\rightarrow$$UK_m$. The updatekey generation algorithm is run by the data owner. It takes as inputs the relevant public keys, the encryption information EnInfo(m) of the message m, the previous access policy A and the new access policy A'. It outputs the update key UK_m of m used to update the ciphertext CT from the previous access policy to the new one.
- **CTUpdate**(CT,UK_m)$\rightarrow$CT'.The ciphertext updating algorithm is run by cloud server. It takes as inputs the previous ciphertext CT and the update key UKm. It outputs a new ciphertext CT' corresponding to the new access policy A'.

C Security Model

The cloud server is curious about the stored data and messages it received during the services. But it is assumed that the cloud server will not collude with users, i.e., it will not send the ciphertexts under previous policies to users, whose attributes can satisfy previous access policies but fail to satisfy new access policies. The users are assumed to be dishonest, i.e., they may collude to access unauthorized data. The authorities can be corrupted or compromised by the attackers. We assume that the adversary can corrupt authorities only statically, but key queries can be made adaptively.

We now describe the security model of our system by the following game between a challenger and an adversary:

Setup. The global setup algorithm is run. The adversary specifies a set S'_AS_A of corrupted authorities. The challenger generates secret/public key pairs by running the authority setup algorithm. For uncorrupted authorities in $S_AS'_A$, the challenger sends only public keys to the adversary. For corrupted authorities in S'_A, the challenger sends both public keys and secret keys to the adversary.

Phase 1. The adversary makes secret key queries by submitting pairs (GID,$S_{GID,AID}$) to the challenger, where GID is an identity and $S_{GID,AID}$ is a set of attributes belonging to an uncorrupted authority AID. The challenger gives the corresponding secret keys $SK_{GID,AID}$ to the adversary.

Challenge. The adversary submits two equal length messagesm_0 and m_1. In addition, the adversary gives a set of challenge access structure which must satisfy the constraint that the adversary cannot ask for a set of keys that allow decryption, in combination with any keys that can be obtained from corrupted authorities. The challenger then flips a random coin b, and encrypts m_b under all access structures. Then, the ciphertext {$CT^*_1,\ldots,CT^*_q$} are given to the adversary.

Phase 2. The adversary may query more secret keys, as long as they do not violate the constraints on the challenge access structures. The adversary can also makes update key queries by submitting the pair the simulator returns the update key UK_{mb} to the adversary.

Definition 2.*Our scheme is secure against static corruption of authorities if all polynomial time adversaries have at most a negligible advantage in the above security game.*

III. Attribute Based Access Control with Dynamic Policy Updating for Big Data using Elgamal

The construction of our dynamic-policy access control scheme is based on an adapted ElGamal method [22]. Our

scheme consists of five phases: *Signature scheme, Signature algorithm, Data encryption, Verification and Policy Updating*.

A ElGamal Signature scheme

Assume that Alice has an El-Gamal key for which the public part is (g, b, P),and the private part is the number a. Recall:

- P is a prime number.
- 1 < g < P is a primitive root of P.
- b = ga mod P.

Typically there would also be a hash-function H involved in digitally signing a message M with an El-Gamal signature. One would first compute H(M),the hash of the message, and then digitally sign the hash. For purposes of ex-position, we may denote the quantity (whether it is the hash or just the raw message) which will be signed also by M. The M we sign must be less than P. We describe here how a message M would be signed, assuming that M < P.

B Signature algorithm

- Select randomly a number r < P − 1 such that gcd(r, P − 1) = 1.
- Compute y = gr mod P.
- Compute s = (M − ay)(r−1) mod (P − 1).

Alice's El-Gamal signature on M is (y, s).

C Data Encryption

The owner first encrypts the data *m* by running the encryption algorithm Encrypt. The algorithm takes as inputs a set of public keys {P,K} for relevant authorities, the global parameters, the data *m* and an *n×l* access matrix M with ρ mapping its rows to attributes. There is Swiss army knife of cryptography technique [23] which uses any of the encryption process that is random method. It chooses a random encryption exponents Z_p and a random vector = (s, y_2,…,y_l) Z_p^l, wherey_2,…,y_l are used to share the encryption exponent s. For i = 1to n, it computes $\lambda_i = M_i \cdot$ where M_i is the vector corresponding to the i-th row of M. It also chooses a random vector Z_p^l with 0 as its first entry and computes $w_i = M_i$. .For each row i of M, it chooses a random r_i Z_p and computes the ciphertext as The encryption information EnInfo(m) of the data m contains all the random numbers ri, i.e., EnInfo(m) = {r1,…,r_n}.

D Verification algorithm

The verifier knows the following things: Alice's public key (g, b, P), the message M and presented signature (y, s). The verifier does NOT know Alice's private key A and the random number r chosen by Alice.

The verifier now computes:

- V1 = ys · by P.
- V2 = gM mod P.

If V1 = V2, and if y, s < P, then the signature (y, s) is accepted as Alice's; otherwise, the signature is not accepted.

E Policy Updating

To update the access policy of the encrypted data in the cloud, we delegate the ciphertext update from the data owner to the cloud server, such that the heavy communication overhead of the data retrieval can be eliminated and the computation cost on data owners can also be reduced.

When the data owner wants to update the ciphertext from the previous access policy A to the new access policy A', it first generates an update key UK_m by running the updatekey generation algorithm UK_{Gen}, and then sends the updatekey UK_m to the cloud server. Upon receiving the update key from the data owner, the cloud server will run the ciphertextupdating algorithm CTUpdate to update the ciphertext from the previous access policy A to the new one A'.

However, the update key generation algorithm UKGen and the ciphertextupdating algorithm CTUpdate are related to the structure relationship between the previous access policy A and the new access policy A'. For different types of updating operation, we have different design of UKGen and CTUpdate, which will be described in detail in the next section.

F Features of Attribute-based Access Control

In big data era, the volume of data is high and it is increasing in a high velocity. The proposed attribute-based access control(ABAC) method [2] is quite suitable for controlling big data than traditional access control methods due to the following features:

1) *Policy Checking Entity Free*: In ABAC, access policies are defined by data owners but do not require any entity (e.g., the server) to check these policies. Instead, access policies in ABAC are enforced implicitly by the cryptography. Due to this key feature, ABAC is widely applied to control big data in cloud environments, where cloud servers are not trusted to enforce access policies.

2) *Storage Efficiency*: In traditional Public Key Cryptography, for each data, multiple copies of ciphertexts are produced whose number is proportional to the number of users. Considering the high volume of big data, it incurs a huge storage overhead even when only doubling the volume of big data. Fortunately, in ABAC, only one copy of ciphertext is generated for each data, which can reduce the storage overhead significantly.

3) *Dynamic Policies but Same Keys*: Data owners can use the same public key to encrypt data under different access policies, and users do not need to change their secret keys either. What's more, data owners can change access policies of existing ciphertexts by simply sending a request to the cloud server, and let the server do the policy change without leaking out any sensitive information of the data as well as the keys.

IV. DYNAMIC POLICY UPDATING

Every access policy can be demonstrated by either LSSS structure or Access Tree Structure. In this section, we only consider monotonic structures, and non-monotonic structures can be similarly achieved betaking NOT operation as another attribute. Specifically, we first design the policy updating algorithms for monotonic Boolean formulas. Then, we present the algorithms to update LSSS structures. Finally, we consider general threshold access tree structures by designing algorithms of updating a threshold gate.

A Updating a Boolean Formula

Access policies with monotonic Boolean formulas can be represented as the simplest threshold access trees, where then on-leaf nodes are AND and OR gates, and the leaf nodes correspond to attributes. The monotonic boolean formulas can be easily converted to LSSS structure, because the number of leaf nodes in the access tree is the same as the number of rows in the corresponding LSSS matrix. As shown in Fig. 3, there are four basic operations: Attr2OR, Attr2AND, AttrRmOR and AttrRmAND.

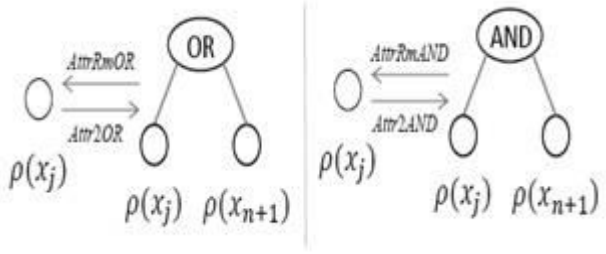

Fig.3. *Operations of Boolean Formula*

B Updating a LSSS Structure

Access policies can also be expressed in LSSS structure as in our access control scheme. To convert a LSSS structure (M,ρ)to a new LSSS structure (M',ρ'), it is too costly to choose anew encryption secret s' and re-encrypt the data under the new access policy. In order to save the communication cost and the computation cost on data owners, in our method, we do not change the encryption secret s, such that we can make full use of the previous ciphertext encrypted under the old policy(M,ρ). To enable the data owner to re-randomize the encryption secret *s*, the encryption information EnInfo(m) of the data m should also contain two random vectors and, and the public key of each attribute *x* is known to the data owner as($g^{\alpha x}$,$g^{\beta x}$). The data owner will run the update key generation algorithm to construct the update keys and send them to the cloud server. Upon receiving update keys, the cloud server will run the ciphertext update algorithm to update ciphertext from the previous access policy to the new policy.

C Updating a Threshold Gate

The problem of updating a threshold gate from *(t,n)*-gate to*(t'n')*-gate has been discussed in key-policy structure [2] and ciphertext-policy structure [21]. However, the existing methods would introduce a security problem in the new threshold gate.

For example, when increasing the threshold value from *t* to*t +1*, existing methods will set the *t +1* share λ_{t+1} of the secret s to be 0, such that the secret s can be reconstructed by using *t +1* shares as *s+0 = s*. In this case, any t shares are still be able to reconstruct the secre*ts*, which should not be allowed in a *(t +1, n)*-gate.

To solve the security problem, instead of setting the value of the new share to be 0, our method is to re-randomize the secret *s* under the new policy *(t',n')*-gate, as shown in Fig. 4. The data owner first transforms the threshold gate into LSSS structure by running the policy converting algorithmThreshold2LSSS, i.e., transforming (*t,n*)-gate and *(t',n')*-gate to (M,ρ) and (M',ρ') respectively. Then, we can apply the DNF2LSSS,SSS2MSP[12] and DNF2SSS algorithms to update the LSSS structure (M,ρ) to the new one (M',ρ').To convert a threshold gate to LSSS structure, the algorithmThreshold2LSSS first converts the threshold gate into DNF boolean formulas, and then converts the DNF boolean formulas into LSSS structure by calling the algorithm DNF2LSSS[13].For example, a (2,3)-gate on attributes A, B, C can be simply represented as (A^B)_(B^C)_(A^C).

The algorithm DNF2LSSS used to change DNF boolean formulas to LSSS structures[1] is a combination of two algorithms:

- DNF2SSS
- SSS2MSP

V. CHECKING ON POLICY UPDATING

After sending the policy updating request to the cloud server, the data owner waits for the cloud server to finish the updating of all the relevant ciphertexts in the cloud. Then, the data owner will check whether the cloud server has done the updating operation correctly by a challenge-proof policy checking protocol. Specifically, the data owner sends a Checking Challenge to the cloud server. Then, the cloud server sends back a Checking Proof ***P*** to the data owner. Upon receiving the proof ***P***, the data owner verifies the correctness of the proof from the cloud server. If the proof is correct, it means the cloud server has updated the ciphertext correctly.

To enable the data owner to check the updating[1], we assume that each owner also has a global identity GID_0. During the system initialization, each owner GID_0 can receive a secret key $SK_{O,AID}$ and a checking key $CK_{O,AID}$ from the corresponding authority AID[1].*Challenge* The data owner generates a checking challenge $\boldsymbol{\delta}$ as

$$\boldsymbol{\delta} = (H(GID_0),S')$$

And send it to the cloud server.

Proof The cloud server then generates a proof ***P*** according to the challenge as

$$\boldsymbol{P}= \{pi=C_{1,i} \bullet e(H(GID_0),C_{3,i})\} I \in I_{s'}$$

and sends back to the data owner. Finally upon receiving the proof ***P*** the data owner checks whether *S'* can satisfy the new access policy.

VI. Analysis of Our Scheme

In this section, we give the performance analysis of our scheme.

A Performance Analysis

In our method, the data owner only needs to send the update keys to the cloud server, instead of the whole encrypted big data. Therefore, our method can significantly reduce the communication cost during the policy updating. Suppose |p| is the element size in the G,GT,Zp. Table 1 shows the size of update keys in our scheme.

We can see that Type1 operation incurs the smallest size of update keys. When updating an access policy to a new one, the most common operation is Type 1 operation such that our scheme incurs a small communication cost.

Operation	Size(UK)
Attr2OR	4|p|
Attr2AND	5|p|
Type1	2|p|
Type2	3|p|
Type3	3|p|

Table 1*.Sizes of Update keys*

Compared with SSW's scheme [21], our scheme makes full use of the previous ciphertexts encrypted under the old access structure. That is if an attribute in the new access policy has ever appeared in the previous access policy, the new ciphertext component of this attribute can be derived from the previous ciphertext component with the update key.

The data owner only needs to compute ciphertext components for new attributes. Moreover, in our scheme, we also delegate all the pairing operations to the server, such that the workload of the data owner can be further reduced.

To evaluate the computation time, we conduct the simulation on a Linux system with an Intel Core 2 Duo CPU at 3.16GHzand 4.00GB RAM. The code uses the Pairing-Based Cryptography library version 0.5.12 to simulate the access control schemes. We use a symmetric elliptic curve a-curve, where the base field size is 512-bit and the embedding degree is 2.The a-curve has a 160-bit group order, which means p is a160-bit length prime.

VII. Related Works

The attribute-based encryption (ABE) technique [2]–[4], [6]is regarded as one of the most suitable technologies for data access control in cloud storage systems. There are two complementary forms of ABE, Key-Policy ABE (KP-ABE)[2] and Ciphertext-Policy ABE (CP-ABE) [3], [4]. In KPABE ,attributes are used to describe the encrypted data and access policies over these attributes are built into user's secret keys; while in CP-ABE, attributes are used to describe the user's attributes and the access policies over these attributes are attached to the encrypted data.

A variety of issues associated with the big data in the cloud is studied inorder to bring out methods to overcome the issues are provided [19]. A complex access control scheme used is CP-ABE is untrusted storage server and secure against collision attacks and the methods used are closer to traditional access control methods and also provided a system implementation [20]. The data access control is an efficient way to ensure the data security in the cloud [11].To deal with the security problems various schemes based on the ABE have been proposed recently as in [18].

However all the above works cannot satisfy any of the few requirements such as completeness or correctness since they can only delegate key, ciphertext with a new access policy which must be restrictive. Furthermore they cannot satisfy the security requirements either they proposed that ciphertext can be re-encrypted by any valid users by decrypting it first.

VIII. Conclusion

In this paper we have investigated the policy updating problems in the big data access control system and proposed some of the challenging requirements of the problem. A new outsourced policy updating method using ElGamal encryption algorithm with use of a latest encryption technique called Swiss army knife of cryptography[23] is introduced.

This method avoids the transmission of encrypted data and minimizes the computational work for the data owners with the use of the old access policies for proposing the policy updating algorithms to the different access policies.

Here they design a CP-ABE technique for the policy updation process. Finally this scheme proposes an efficient and secure method for the data owners to check the updation of cipher texts in the cloud.

In our Future work, we use pure homomorphic algorithm for the policy updation process. This is a highly secure process and also outsources the policy updating to the server and we design the policy updation process with in the algorithm. This covers the authorities, cloud server, data owners and data consumers in the cloud storage system

Reference

[1] Kan Yang, XiaohuaJia ,KuiRen"Secure and Verifiable Policy Update Outsourcingfor Big Data Access Control in the Cloud,"in1045-9219 (c) 2013 IEEETransactions on Parallel and Distributed Systems.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryptionfor fine-grained access control of encrypted data," in CCS'06.ACM, 2006, pp. 89–98.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute based encryption," in S&P'07. IEEE, 2007, pp. 321–334.

[4] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive,efficient, and provably secure realization," in PKC'11. Springer, 2011,pp. 53–70.

[5] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters,"Fully secure functional encryption: Attribute-based encryption and(hierarchical) inner product encryption," in EUROCRYPT'10. Springer,2010, pp. 62–91.

[6] A. B. Lewko and B. Waters, "Decentralizingattribute-based encryption,"in EUROCRYPT'11. Springer, 2011, pp. 568–588.

[7] Andreas, V. Meier, "The ElGamal Cryptosystem," June 8, 2005

[8] Whit_eldDi_e and Martin E. Hellman. New directions in cryptography. IEEETransactions on Information Theory, IT-22(6):644-654, 1976.

[9] K. Yang, X. Jia, and K. Ren, "Attribute-based fine-grained access controlwith efficient revocation in cloud storage systems," in AsiaCCS'13.ACM, 2013, pp. 523–528.

[10] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: EffectiveData Access Control for Multiauthority Cloud Storage Systems," IEEETrans. Info. Forensics Security, vol. 8, no. 11, pp. 1790–1801, 2013.

[11] K. Yang and X. Jia, "Expressive, efficient, and revocable data accesscontrol for multi-authority cloud storage," IEEE Trans. Parallel Distrib.Syst., vol. 25, no. 7, pp. 1735–1744, July 2014.

[12] A. Beimel, "Secure schemes for secret sharing and key distribution,"DSc dissertation, 1996.

[13] J. C. Benaloh and J. Leichter, "Generalized secret sharing and monotonefunctions," in CRYPTO'88, 1988, pp. 27–35.

[14] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity basedencryption with constant size ciphertext," in EUROCRYPT'05, 2005,pp. 440–456.

[15] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigmfor designing efficient protocols," in CCS'93, 1993, pp. 62–73.

[16] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, andfine-grained data access control in cloud computing," in INFOCOM'10.IEEE, 2010, pp. 534–542.

[17] K. Yang and X. Jia, "Attributed-based access control for multi-authoritysystems in cloud storage," in ICDCS'12. IEEE, 2012, pp. 1–10.

[18] T. Jung, X.-Y.Li, Z. Wan, and M. Wan, "Privacy preserving cloud dataaccess with multi-authorities," in INFOCOM'13.IEEE, 2013, pp. 2625–2633.

[19] VenkataInukollu, Arsi, Ravuri, "Security issues associated with big data in the cloud computing" in IJNSA, vol.6, No.3, May 2014.

[20] John, Sahai, B.Waters, "Ciphertext-Policy Attribute based encryption," in S&P '07, IEEE, 2007, pp.321-334.

[21] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials andciphertext delegation for attribute-based encryption," in CRYPTO'12.Springer, 2012, pp. 199–217.

[22] Daniel Bleichenbacher," Generating ElGamal Signature without knowing the secret key," in Institute for theoretical computer science,CH-8092 Zurich, Switzerland on April 16,1996

[23] Xavier Boyen,"Multipurpose Identity-Based SigncryptionA Swiss Army Knife for Identity-Based Cryptography",Proc. 23rd Int'l.Conf. Advances in Cryptology (CRYPTO '03), LNCS series,Springer Verlag, 2003.