ANALYSIS

52(245), May 1, 2016

ISSN 2278–5469 EISSN 2278–5450

Discovery

Enhanced policy modernizing outsource for big data access control in the cloud

Nallakumar R[1], Sengottaiyan N[2], Ramya Vinodini R[3]

1.Assistant Professor, Department of Computer science and engineering, Anna University Regional Centre, Coimbatore, India
2.Professor, Department of Computer science and engineering, Anna University Regional Centre, Coimbatore, India
3.Department of Computer science and engineering, Anna University Regional Centre, Coimbatore, India

Article History
Received: 16 March 2016
Accepted: 23 April 2016
Published: 1 May 2016

Citation
Nallakumar R, Sengottaiyan N, Ramya Vinodini R. Enhanced Policy Modernizing Outsource For Big Data Access Control in the Cloud. *Discovery*, 2016, 52(245), 1039-1046

ABSTRACT

Cloud storage service allows data owner to host their data in the cloud and through which provide the data access to the users. Due to the high volume and velocity of big data, it is an effective option to store big data in the cloud, as the cloud has the capabilities of storing big data and also processing high volume of user access requests. Because of data outsourcing and untrusted cloud servers, the data access control is an effective way to ensure the data security in the cloud. ABE is a promising practice for access control of encrypted data. It allows the encryptor to specify access control in terms of any of the access formula over the attributes in the system. Policy updating is always being a challenging issue when ABE is used for the access control, because of high communication overhead and heavy computation burden on the data owners. In the proposed system instead of retrieving and re-encrypting the data, data owners only send policy updating queries to cloud server and let cloud server update the policies of encrypted data directly, which means that cloud server does not need to decrypt the data before or during the policy updating. This solution provides an efficient way of access control with the dynamic policy updating, which have the ability to use the previously encrypted data with old access policies and also it reduces the computation work of the data owners.

Keywords: Access control, Attribute Based Encryption (ABE), Policy Updating, Outsourcing, Big Data, Cloud.

1. INTRODUCTION

Cloud computing is an on-demand computing that provides access to a shared pool of configurable computing resources like networks, applications, servers, storage, and services. It mainly focuses on maximizing the effectiveness of the shared resources. Big data is the large volume of data, both structured and unstructured which usually includes data sets which have the sizes that beyond the ability of commonly used software tools to curate, capture, process and manage data within a passable elapsed time. Big data is defined as a set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are complex, diverse, and of a massive scale. When hosting big data into the cloud, the data security becomes a major concern as cloud servers cannot be fully trusted by data owners. Attribute Based Encryption **(**ABE) is a public-key based one to many encryptions that allows users to encrypt and decrypt data based on user attributes. In such a system, the decryption of a ciphertexts is possible only if the set of attributes of the user key matches the attributes of the ciphertext. Decryption is only possible when the number of matching is at least a threshold value d. Attribute-based access control (ABAC) defines an access control paradigm where access rights are granted to users through the use of policies which combine attributes together. It is attribute based access control normally considers identification, authentication, authorization and accountability. The policy updating is a difficult issue in attribute-based access control systems, because once the data owner outsourced data into the cloud, it would not keep a copy in local systems. When the data owner wants to change the access policy, it has to transfer the data back to the local site from the cloud, re-encrypt the data under the new access policy, and then move it back to the cloud server. By doing so, it incurs a high communication overhead and heavy computation burden on data owners. This paper, mainly focus on solving the policy updating problem in ABE systems and propose an enhanced policy updating outsourcing method. Instead of retrieving and re-encrypting the data, data owners only send policy updating queries to cloud server, and let cloud server update the policies of encrypted data directly, which means that cloud server does not need to decrypt the data before/during the policy updating. this scheme also avoid the transfer of encrypted data back and forth and minimize the computation work of data owners by making full use of the previously encrypted data under old access policies in the cloud.

The contributions of this paper include:

- Formulation of the policy updating problem in ABE systems and develop a new method to outsource the policy updating to the server.
- Propose an expressive and efficient data access control scheme for big data, which enables efficient dynamic policy updating.
- Reduce the storage space and computational time by making use of the previously encrypted data under old access policies.

2. RELATED WORK

Table 1 Comparison of encryption methods

Techniques/ parameter	ABE	KP-ABE	CP-ABE	HABE	MA-ABE
Fine Grained Access Control	Low	Low, High If There Is Re-encryption Technique	Average Realization Of Complex Access Control	Good Access Control	Better Access Control
Efficiency	Average	Average, High For Broadcast Type System	Average, Not Efficient For Modern Enterprise Environment	Flexible	Scalable
Computational Overhead	High	Most Of Computational Overheads	Average Computational Overheads	Some Of Overhead	Average
Collusion Resistant	Average	Good	Good	Good	High Collusion Resistant

Sahai and Waters introduced attribute-based encryption (ABE) as a new means for encrypted access control. In an attribute-based encryption system ciphertexts are not necessarily encrypted to one particular user as in traditional public key cryptography. Instead both users' private keys and ciphertexts will be associated with a set of attributes or a policy over attributes. A user is able to

decrypt a ciphertext if there is a "match" between his private key and the ciphertext. In their original system Sahai and Waters presented a Threshold ABE system in which ciphertexts were labelled with a set of attributes S and a user's private key was associated with both a threshold parameter k and another set of attributes S'. In order for a user to decrypt a ciphertext at least k attributes must overlap between the ciphertext and his private keys. The primary drawback of the Sahai-Waters threshold ABE system is that the threshold semantics are not very expressive and therefore are limiting for designing more general systems. Goyal et al. Introduced the idea of a more general key-policy attribute based encryption system. In their construction a ciphertext is associated with a set of attributes and a user's key can be associated with any monotonic tree access structure.

The construction of Goyal et al. can be viewed as an extension of the Sahai-Waters techniques where instead of embedding a Shamir secret sharing scheme in the private key, the authority embeds a more general secret sharing scheme for monotonic access trees. Goyal et. al. also suggested the possibility of a ciphertext-policy ABE scheme, but did not offer any constructions. Pirretti et al gave an implementation of the threshold ABE encryption system, demonstrated different applications of attribute-based encryption schemes and addressed several practical notions such as key-revocation. In recent work, Chase gave a construction for a multi-authority attribute-based encryption system, where each authority would administer a different domain of attributes. The primary challenge in creating multi-authority ABE is to prevent collusion attacks between users that obtain key components from different authorities. While the Chase system used the threshold ABE system as its underlying ABE system at each authority, the problem of multi-authority ABE is in general orthogonal to finding more expressive ABE systems. The defining property of ABE systems are their resistance to collusion attacks. This property is critical for building cryptographic access control systems; otherwise, it is impossible to guarantee that a system will exhibit the desired security properties as there will exist devastating attacks from an attacker that manages to get a hold of a few private keys. While considering ABE systems with different flavours of expressibility, prior work made it clear that collusion resistance is a required property of any ABE system. Before attribute-based encryption was introduced there were other systems that attempted to address access control of encrypted data by using secret sharing schemes combined with identity-based encryption however, these schemes did not address resistance to collusion attacks.

3. SYSTEM MODEL

When hosting big data into the cloud, the data security becomes a major concern as cloud servers cannot be fully trusted by data owners. Most of the previous methods cannot satisfy the completeness requirement, because they can only delegate key/ciphertext with a new access policy that should be more restrictive than the previous policy. Existing systems uses the separate algorithms for each process the time conception will be more and also each data is encrypted separately so the decrypted data will also be more this will increase the storage space more. It creates the data over flow and also the chance for denial of service (DoS) attack. DoS is the attack that may cause of overloading of the data and also this makes the system slower or even create the downtime. The contribution of proposed scheme includes the policy updating problem in ABE systems and develops a new method to outsource the policy updating to the server. Also propose an expressive and efficient data access control scheme for big data, which enables efficient dynamic policy updating. Compared to the previous version, this scheme also propose an efficient and secure policy checking method that enables data owners to check whether the cipher texts have been updated correctly by cloud server. In this method, do not require any help of data users and data owners can check the correctness of the cipher text updating by their own secret keys and checking keys issued by each authority. This method can also provides guarantee data owners cannot use their secret keys to decrypt any cipher texts encrypted by other data owners, although their secret keys contain the components associated with all the attributes. Moreover, this discuss gives some key features of the attribute-based access control scheme and show how it is suitable for big data access control in the cloud.

Randomized generation is used for the key generation so it further reduces the work complexity of the data owner and also this kind of key generation can also a improve the security by avoiding the user to use their keys to decrypt the data of other authority. Here single algorithm is used for the entire implementation so it avoids the complexity of the process flow. It allows the automatic encryption of the entering data it increases the performance of the method. Since the system uses the automatic encryption the data are not encrypted separately so the memory space required for storage data will be reduced, by this DoS attack can be avoided hence this attack occurs because of data overflow. Cipher Text Attribute Based Encryption is used for security purpose of the data that are stored in cloud this method avoids the collusion attack. Dynamic policy access control scheme is a collection of the following algorithms: GlobalSetup, Authority Setup, SKeyGen, Encrypt, Decrypt, UKeyGen and CTUpdate.

GlobalSetup (λ)= GP. The global setup algorithm takes no input other than the implicit security parameter λ. It outputs the global parameter GP for the system.

Authority Setup (GP,AID) = (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})=$SK_{GID,AID}$.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)=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,{SKGID,AID})=m. 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')= UKm. The update key 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 UKm of m used to update the ciphertext CT from the previous access policy to the new one.

CTUpdate (CT,UKm) = 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 CT0 corresponding to the new access policy A'.

To update the access policy of the encrypted data in the cloud, it 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 UKm by running the update key generation algorithm UKGen, and then sends the update key UKm to the cloud server. Upon receiving the update key from the data owner, the cloud server will run the ciphertext updating algorithm CT Update to update the ciphertext from the previous access policy A to the new one A0.However, the update key generation algorithm UKGen and the ciphertext updating algorithm CT Update are related to the structure relationship between the previous access policy A and the new access policy A0.

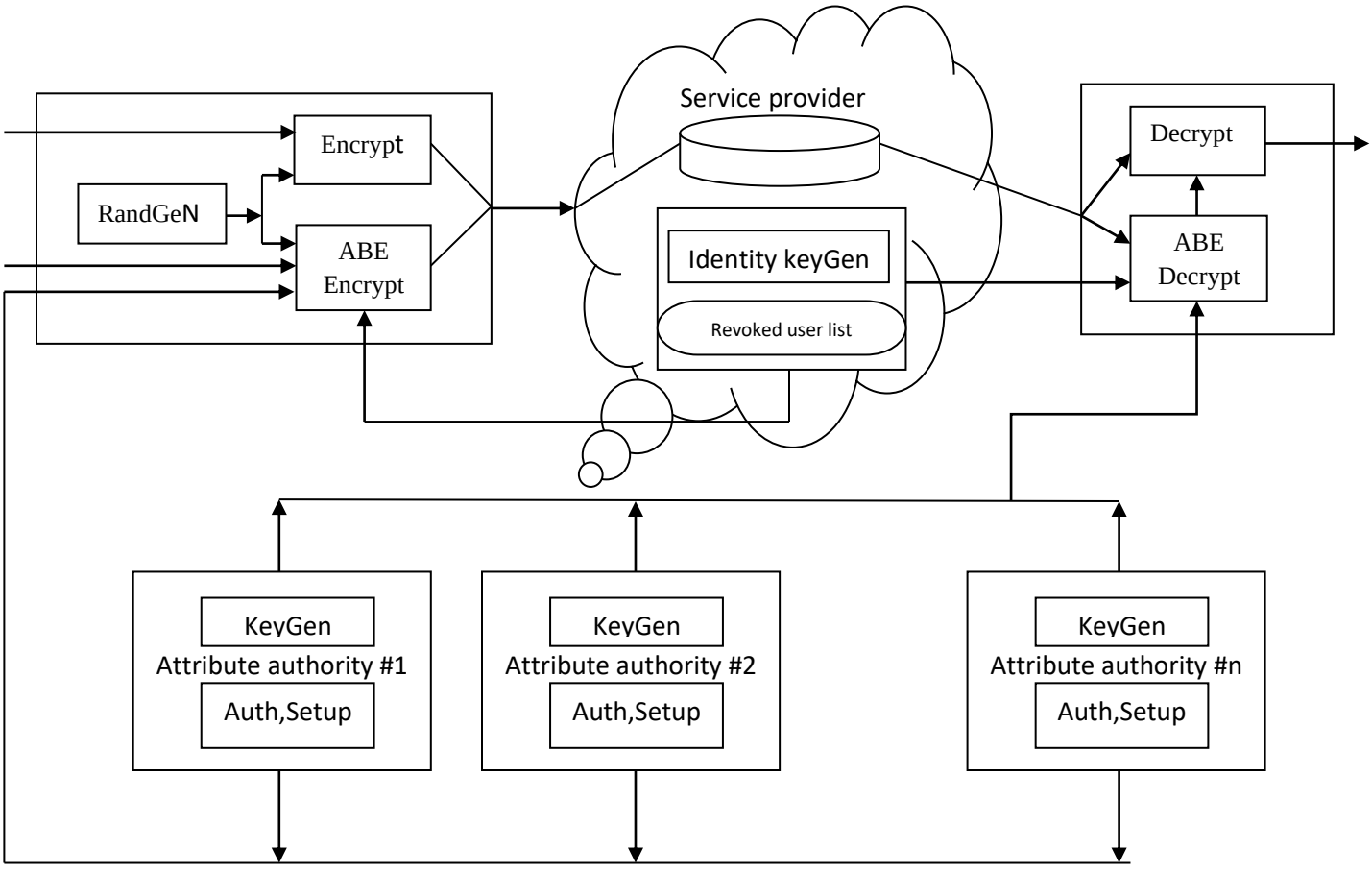

Figure 1 System Architecture

4. SYSTEM ARCHITECTURE

The system consists of four important sections they are encryption part, decryption part, cloud provider and the authority setup. In this encryption will be done by the data owner and also in the proposed scheme the data will be encrypted automatically if the owner once enters the data. The data owner can also get the revoked user list from the cloud provider for the secure distribution of the key. Randomized key generation is used in the ABE encryption part so that it reduces the complexity of the data owner to find the revoked user. The job of cloud provider is to store the encrypted data and key information, and also cloud server is responsible for the policy update here. Owner will just provide the policies and encrypted data. If the user once decrypted the data under a particular policy they cannot use the same key to decrypt the data of others. Authority setup will provide two keys a public key and the secret key. The randomized public key is given to the data owner for the encryption and secret key will be given to user for the decryption. And also the access structure is given to the data owner, it allows data owners to define an access structure on attributes and encrypt the data under this access structure, such that data owners can define the attributes that the user needs to possess in order to decrypt the ciphertext.

5. RSA ALGORITHM

RSA algorithm is designed by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT in 1978. It can be broadly classified into three steps; a) key generation, b) encryption, and c) decryption. RSA has two keys; public key and private key. Both keys are used for encryption and decryption purpose. Sender encrypts the message using receiver's public key and when the message gets transmit to receiver, then receiver can decrypt it using his own private key. It uses two prime numbers and to generate private key and public key. It uses two prime numbers and to generate private key and public key. For small values of & , the designing of key in the encryption process becomes too weak and one could be able to decrypt the data using random probability theory. On the other hand, if large value of and are selected, it consumes more time and its performance degrades. In this algorithm, we use four prime numbers instead of two so that it becomes very difficult to factor n. This increases the security of the system and also increases the computation complexity. To decrease the complexity we improved the time taken by modulo operation. Both encryption and decryption in RSA involve raising an integer to another integer mod n. Since the integers are large numbers, if the exponentiation is done before modulo operation the size of the intermediate result would be very large. To make it practical to implement the RSA algorithm the following property of modular arithmetic is exploited.

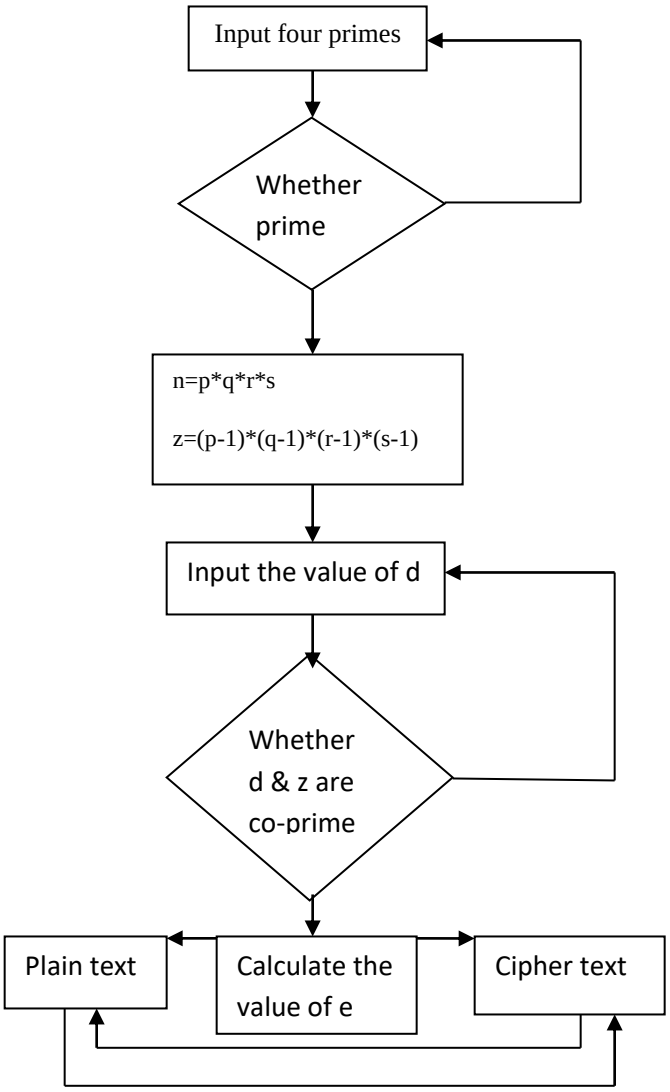

Figure 2 Flowchart of Proposed Algorithm

$$(a \bmod n)*(b \bmod n) = (a*b) \bmod n$$

Using this property along with successive multiplication scheme it is possible to compute x^6 with less than (e-1) multiplications.

Key Generation Algorithm:

There are two types of keys in RSA; public key and private key. The steps for key generation are given as:

- Generate four large prime numbers p,q,r and s.
- Compute n=p*q*r*s
- Compute z=(p-1)*(q-1)*(r-1)*(s-1)
- Choose a number relatively prime to z and call it d .
- Find e such that e*d= 1 mod z .
- Public key is (n , e)
- Private key is (n, d).

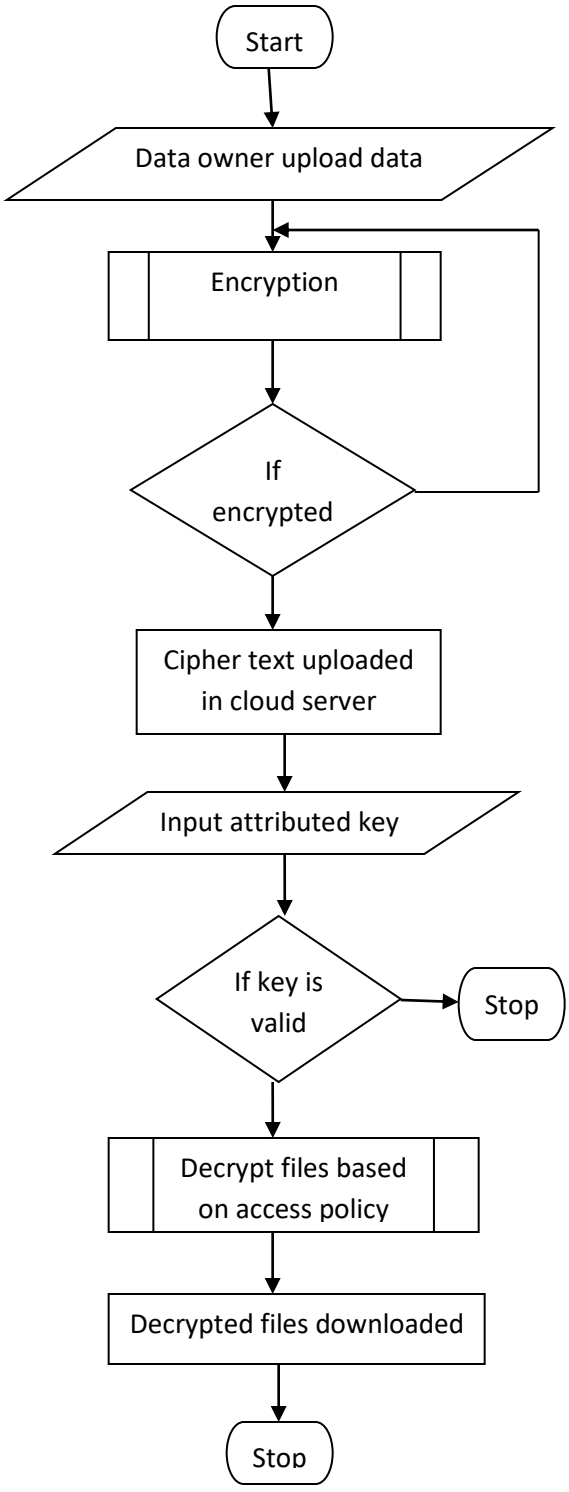

Figure 3 Flow Chart for Process Flow

Encryption Algorithm:

In RSA encryption is done with the help of public key to generate the cipher text. The steps for encryption are given as:

- Obtains the recipient public key (n,e) .
- Represents the plain text message as positive integer.
- Compute the cipher text c= m^ e mod n.
- Sends the cipher text.
- Using d for decryption
- Using e for encryption

Decryption Algorithm:

In RSA decryption is done with the help of private key to get the plain text. The steps for decryption are given as:

- Compute m= c^ e mod n by using private key.
- Extracts the plain text from integer representing m.

6. IMPLEMENTATION

Implementation of the proposed scheme can be carry out by following steps:

1. Big Data populate to Cloud Server
2. ABE Security Enhancement
3. User Interface Designing
4. Cloud Storage Enhancement
5. Accessor Verify Standards & Get Big Data

A. BigData populate to Cloud Server

The application admin can upload the data contained files to cloud server. The server act as interface between cloud and ABE processing. The data which is uploading by the admin is to stored cloud server interconnected to database server. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. With cloud computing, multiple users can access a single server to retrieve and update their data without purchasing licenses for different applications. The architectural elements described in this document will help you understand the components for leveraging various cloud deployment models.

B. ABE Security Enhancement

Attribute encryption standard plays vital role in which is populated by the admin is verified and converted to an encrypted format. ABE converts the Big Data to an unsigned format. The un-authorized used not able to view the ABE format. User revocation is a challenge issue in ABE as attributes are shared among unlimited number of users. Revocation of one user may involve key update for other non-revoked users and/or re-encryption of data files on the data servers. To facilitate user revocation on untrusted storage, this dissertation proposes a novel scheme in which the data owner is able to revoke any user in the timely fashion. The proposed scheme makes it possible for the data owner to securely offload most computation-intensive tasks pertained to user revocation to data servers which are envisaged to be powerful. In practical systems, it would be difficult for the data owner to obtain a copy of the pirate's decryption key and check its validity. This is because the data owner may not be able to get physical access to the pirate's key storage device or the pirate may have randomized the key storage memory. To address this issue, we propose a black-box tracing mechanism, i.e., tracing the pirate device only by observing its outputs on some inputs. Such a solution also enables the data owner to remotely trace suspicious users by tricking them into decrypting tracing ciphertexts and thus makes the tracing process very convenient.

C. User Interface Designing

By using this module End-user can register to access the application. User should enrol their details like their user name, IP address and other necessary details to get the access authority. The user profiles are maintained by the Database server. DB server verifies the end-user details for security; if the user has given valid information then the server will provide an accessing authority to the users

D. Cloud Storage Enhancement

In this module the purpose is to identify the performance of cloud storage. Since the big data is used here the storage capacity should be more and reliable. Here not only the plaintext is stored, the encrypted data should also be stored hence in this method the data are encrypted as a whole set it reduces the memory space required for the storage. this module can also used to see the Big data content and the memory occupied by the data. The database is used to store all data. Databases are used to support internal operations of organizations and to underpin online interactions with users and service provider .Here SQL is used to communicate with the database. As the cloud provides the distributed storage, the data are stored in multiple servers so by using this module the number of data stored in each server and path of each data can also be clearly viewed.

E. Accessor Verify Standards and Get Big Data

This module is designed for End-to-End client-server architecture. The authorised Client gives the request to server and server response back to the client with encrypted data and key information. Client receives the encrypted content and using that secret key he will decrypt the data. User's private key will be associated with an arbitrary number of attributes expressed as strings. A user will only be able to decrypt a ciphertext if that user's attributes pass through the cipher text's access structure.

7. DYNAMIC 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. The proposed attribute-based access control (ABAC) method 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.

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 C 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, we assume that each owner also has a global identity. During the system initialization, each owner can receive a secret key and a checking key from the corresponding authority. The secret key can only be used to verify the checking proofs from the data owner together with the checking key, while the owner's secret key cannot be used to decrypt any ciphertexts encrypted by other data owners.

8. CONCLUSION

Data storage in cloud computing and the security in cloud system is given as an overview, here the main idea is to give integrity to the cloud storage area with different data models and security algorithm. The cloud data storage architecture along with the cloud data models and also the algorithm for cloud security using RSA algorithm is presented. Some important security services including key generation, encryption and decryption are provided in Cloud Computing system is enforced by the proposed scheme, the main goal is to securely store and manage data that is not controlled by the owner of the data. An important issue of secure data sharing on untrusted storage can be avoided by using the Attribute-Based Encryption to provide cryptographically enforced data access control, with ABE it can be able to enjoy fine-grained access control. Key idea is to investigate the policy updating problem in big data access control systems and an efficient method is developed to outsource the policy updating to the cloud server, which can satisfy all the requirements of the data owner as well as the user. Proposed scheme gives an expressive ABAC scheme for big data in the cloud, and designed policy updating algorithms for different types of access policies. Furthermore, it enables data owners to check the correctness of the ciphertext updating, also it allows policies to be expressed as any monotonic tree access structure and is resistant to collusion attacks in which an attacker might obtain multiple private keys. It also provides a fine-grained data access control scheme based on CP-ABE approach, where the owner was in charge of defining and enforcing the access policy, which can greatly reduce the cost of attribute revocation.

REFERENCE

1. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in CCS'06. ACM, 2006, pp. 89–98.
2. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute based encryption," in S&P'07. IEEE, 2007, pp. 321–334.
3. B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in PKC'11. Springer, 2011,pp. 53–70.
4. 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.
5. A. B. Lewko and B. Waters, "Decentralizing attribute-based encryption," in EUROCRYPT'11. Springer, 2011, pp. 568–588.
6. S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in INFOCOM'10. IEEE, 2010, pp. 534–542.
7. 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.
8. K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective Data Access Control for Multi authority Cloud Storage Systems," IEEE Trans. Info. Forensics Security, vol. 8, no. 11, pp. 1790–1801, 2013.
9. K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 7, pp. 1735–1744, July 2014.
10. A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in CRYPTO'12. Springer, 2012, pp. 199–217.
11. J. C. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in CRYPTO'88, 1988, pp. 27–35.
12. A. Beimel, "Secure schemes for secret sharing and key distribution," DSc dissertation, 1996.
13. D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in EUROCRYPT'05, 2005, pp. 440–456.
14. M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in CCS'93, 1993, pp. 62–73.
15. K. Yang and X. Jia, "Attributed-based access control for multi-authority systems in cloud storage," in ICDCS'12. IEEE, 2012, pp. 1–10.
16. T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in INFOCOM'13. IEEE, 2013, pp. 2625–2633.