International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org || Volume 2 Issue 12 || December. 2013 || PP.29-34

Data defense in unpredictable Cloud Using Access Control and Access Time

[1]Pooja A.Uplenchwar, [2]Mrs. L.H.Patil
[1]Research Scholar Department of Computer Science Of Engineering and Technology, Priyadarshini Institute of Engineering and Technology, Nagpur University, India
[2]Assistance Professor Department of Computer Science Of Engineering and Technology, Priyadarshini Institute of Engineering and Technology, Nagpur University, India

ABSTRACT: *Cloud computing has been envisioned as the next-generation architecture of IT activity. In difference to established solution, wherever the IT services are under appropriate physical, logical and personnel controls, cloud computing moves the appliance software and databases to the large data centers, wherever the management of the data and services may not bes fully reliable. This unique attribute, however, poses many new security challenges which have not been well understood. In this project, focus on cloud data storage security, which has constantly been an important feature of quality of service Data owner's stores encrypted data in the cloud to ensure security for his data in the cloud computing environment and issues decryption key to only authorized user to access the data from cloud. As user is revoked, data owner has to re-encrypt the data so that revoked user cannot access the data again .To perform this operation data owner will issue re-encryption command to cloud so that data in cloud gets re-encrypted. Once re-encryption is done here is a need for generation of new decryption keys to legal user, so that they can go on to access the data. Within a cloud computing environment all such commands may not be received and executed by all of the cloud servers due to unreliable network communications. To solve this problem we are proposing time-based re-encryption scheme. In this method automatic re-encryption of data will takes place based on the internal clock value present at the cloud server. To execute this automatic re-encryption we will make use of encryption technique called Attribute Based Encryption (ABE) with DES (Data Encryption Standard). ABE provides fine –grain access control and easier user revoking system and DES will provide Encryption technique.*

GENERAL TERMS**:** *Security, Algorithms, Design, Data privacy*

KEYWORDS: *Attribute-based encryption, cloud computing, double decryption, fine grained access control, identity based encryption, proxy re-encryption, revocation, provable security.*

I. INTRODUCTION

Data owner's having huge amount of data will outsource their data to third party who has large storage capacity called "cloud Service providers "(CSP) due to problem of storage capacity , cost involved in storing data among them etc. Cloud Service Provider is a lone who offers storage and computational services to data. Before outsourcing data to CSP's the data owner must think about the security issue related to his data so he will encrypt the data before outsourcing data. To execute this encryption can make use of encryption scheme called "Attribute Based Encryption" process, which provides fine-grained access control. ABE allows data to be encrypted by an access structure comprised of different attributes. Instead of particular decryption keys for particular files, users are issued attribute keys. Users must have the essential attributes that satisfy the access structure in order to decrypt a file. As, a box file encrypted using the access structure $\{(\alpha1\alpha2)\ \alpha3\}$ means that either a user through attributes $\alpha1$ and $\alpha2$, or a user through attribute $\alpha3$, can decrypt the box file.

When an encrypted data is stored and decryption key is allocated to user they can access data from cloud but what is the case when particular user is revoked? While a user is revoked and he has decryption key he can access data still, thus to overcome from this problem here is a need of immediate re-encryption of data by data owner. When re-encryption is done the newly generated decryption keys are distributed to authorized users. This resolution will lead to a performance bottleneck, particularly when there are many user revocations.

An alternative solution is to apply the proxy re-encryption (PRE) technique. This approach takes advantage of the abundant resources in a cloud by delegating the cloud to re-encrypt data. This approach is also called command driven re-encryption scheme, wherever cloud servers execute re-encryption while receiving commands from the data owner.

However, command-driven re-encryption schemes do not consider the underlying system architecture of the cloud environment. A cloud is basically a large scale distributed system where a data owner’s data is replicated over multiple servers for high availability. The same as a distributed system, the cloud will understand failures common to such systems, for example server crashes and network outages. Accordingly, re-encryption commands sent by the data owner may not propagate to all of the servers in a timely fashion, hence creating security risks.

Let us consider a cloud environment shown in Fig. 1, wherever the data owner’s data is stored on cloud servers CS1, CS2, CS3, and CS4. Suppose that the data owner issues to CS4 a re- encryption command, which propagates to CS1, CS2, with CS3. Because of a network problem, CS2 did not obtain the command, and did not re-encrypt the data. By this time, if revoked users question CS2, they can obtain the old cipher text, and can decrypt it by their old keys. A improved solution is to allow each cloud server to independently re-encrypt data without receiving any command from the data owner.
Here in this project proposes a time –based reliable re-encryption scheme which allows each cloud server to automatically re-encrypt data based on its internal clock.

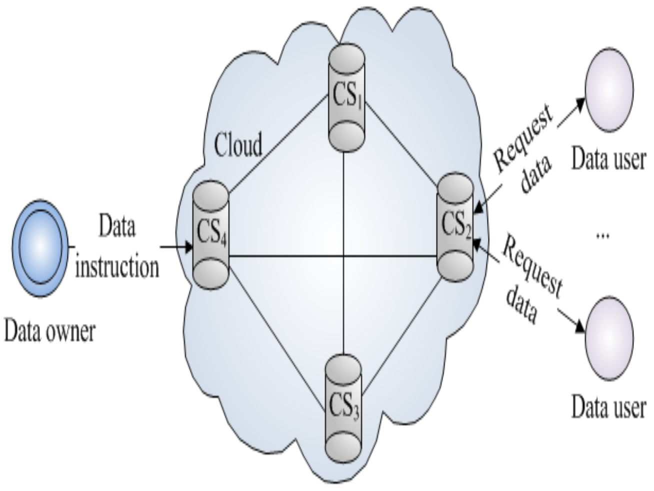


Fig.1 Related work to cloud architecture

II. LITERATURE SURVEY

In [**1**] consider the problem of building a secure cloud storage service on top of a public cloud infrastructure where the service provider is not completely trusted by the customer. Describe, at a high level, several architectures that combine recent and non-standard cryptographic primitives in order to achieve our goal. Survey the benefits such architecture would provide to both customers and service providers and give an overview of recent advances in cryptography motivated specially by cloud storage.

In **[2]** proposed a new method for managing access control on the data being stored on the cloud server, based on the cloud server’s internal clock. Our method does not rely on the cloud service provider to reliably propagate re encryption commands to all servers to ensure access control correctness. It showed that our solutions remain secure and it is strong enough without perfect clock synchronization which depicts the cloud behavior as long as we can bound the time difference between the servers and the data owner.

[3] In this paper, propose an efficient data retrieval scheme using attribute-based encryption. The proposed scheme is best suitable for cloud storage systems with substantial quantity of data. It provides rich expressiveness the same as regards access control and fast searches with simple comparisons of searching entities. The proposed method too guarantees data security end-user privacy during the data retrieval procedure.
A key approach to secure cloud computing is for the data owner to store encrypted data inside the cloud, and issue decryption keys to authorized users. The cloud storage based in order retrieval service is a promising technology that will form a vital market in the near future. However, those approaches require astounding costs centralized on the cloud service provider, this could be a principal interruption to accomplish efficient data retrieval in cloud storage.

In **[4]** DES is the archetypal block cipher — an algorithm that takes a fixed-length string of plaintext bit and transforms it through a series of complicated operations into another cipher text bit string of the same length. DES also uses a key to convert the transformation, in order that decryption can only be performed by those who know the particular key use to encrypt. In the case of DES, the block size is 64 bit, but only 56 bit are used and the remaining 8 bit can be used for equality, and then discarded in the algorithm. Hence, the effective key length of DES is 56 bit. The algorithm's overall structure there is 16 identical same processes, termed rounds. There is also an initial and final permutation, known as IP and FP (the FP is inverse function of IP (IP ―revocation‖ FP operation, and vice versa)).

In [5] introduces a new type of Identity-Based Encryption (IBE) scheme that we call Fuzzy Identity-Based Encryption. In Fuzzy IBE outlook an identity as set of expressive attributes. A Fuzzy IBE scheme allows for a private key for an identity, ω, to decrypt a cipher text encrypted with an identity,ω0 , if and only if the identities ω and ω0 are close to each other as measured by the "set overlap" distance metric. A Fuzzy IBE scheme can be applied to allow encryption using biometric inputs as identities; the error-tolerance property of a Fuzzy IBE scheme is precisely what allows for the use of biometric identities, which naturally will have some noise each time they are sampled. Additionally, shows that Fuzzy-IBE can be used for a type of application that we term "attribute-based encryption".

In **[6]** Cloud computing is an emerging computing paradigm in which resources of the computing infrastructure are provided as services over the Internet. As promising as it is, this model also brings forth many new challenges for data security and access control when users outsource sensitive data for sharing on cloud servers, which are not inside the similar trusted domain like data owners. To maintain sensitive user data confidential in opposition to untrusted servers, existing solutions generally concern cryptographic methods by disclosing data decryption keys only to authorized users. But, in doing so, these solutions predictably introduce a heavy computation overhead on the data owner for key distribution and data management when fine- grained data access control is desired, and therefore do not balance well. The problem of simultaneously achieving fine-grainedness, scalability, and data secrecy of access control actually still remains unsolved. In this paper addresses all this challenging open issue by, on one hand, major and enforcing access policies based on data attributes, and, alternatively, allowing the data owner to delegate most of the computation tasks involved in fine grained data access control to untrusted cloud servers without disclosing the underlying data contents.

III. PROPOSED RESEARCH METHODOLOGY

In this paper consider a cloud computing environment consisting of a data owner, a cloud service provider (CSP) and multiple data users. The data owner outsources his data in the form of a set of files F1…; Fn to the CSP. Each file is encrypted by the data owner before uploading to the CSP. Data users that want to access a particular file must first obtain the necessary keys from the data owner in order to decrypt the file. The data owner can also update the contents of a file after uploading it to the CSP. This is termed a write command.
Each file, F, is encrypted with two parameters, time slice and attributes. And divide time into time slices, and every time slice is of equal length. Then denote a particular time slice, TS, with a subscript, where TSi = [ti; ti+1). Fig.2 illustrates this conception. Attributes are ordered into an access structure, A, which regulates access to a file. For example, a file with attributes 1; 2; 3 and A = {(1∧2)∨3}, requires either both attributes 1 and 2, or just 3, to satisfy the access structure. A file F can only be decrypted with keys that satisfy both the access structure and time slice.

A data user, after being authenticated by the data owner, is decided a set of keys, each one of which is associated with an attribute and an effective time that denotes the length of time the user is authorized to possess the attributes. For example, if Alice is authorized to possess attributes *a1; : ..... am from TS1* to *TSn,* she will be issued keys as is shown in Table I.

The security necessities of this scheme are as follows:
1) Access control correctness. This requires that a data user with invalid keys cannot decrypt the file.
2) Data consistency. This requires that all data users who request file F, should obtain the similar content in the similar time slice.
3) Data confidentiality. The file content can only be known to data users with valid keys. The CSP is not considered a valid data user.
4) Efficiency. The cloud servers should not re-encrypt any file unnecessarily. This means that a file has not been requested by any data user should not be re-encrypted.

In this method considers two types of adversaries:-

The first type of adversary is the CSP. The CSP adversary is considered honest-but-curious. This means that the CSP will always correctly execute a given protocol, but may try to gain some additional information about the stored data.

The second type of adversary is malicious data users. The data user adversary will try to learn the file content that he is not authorized to access. This adversary is assumed to possess invalid keys (either with incorrect attributes or time). We also assume the data user adversary can query any server in the cloud. Note that both an honest-but-curious CSP and malicious data users can exist together.

IV. RELIABLE RE-ENCRYPTION SCHEME

Within the basic R3 scheme, consider ideal situation, where the data owner and all of the cloud servers in the cloud share a synchronized clock, and there are no transmission and queuing delays when executing read and write commands.

4.1 Intuition

The data owner will first generate a shared secret key to the CSP. Afterward, the statistics owner encrypts every file with the appropriate attribute structure and time slice, the statistics owner uploads the file in the cloud. The CSP will imitate the file to different cloud servers. Every cloud server will contain a copy of the shared secret key.

Let us assume that a cloud server stores an encrypted file *F* through A and *TSi*. At what time a user queries that cloud server, the cloud server initially uses its *own* clock to establish the current time slice. Let us considering that the current time slice is *TSi+k*, the cloud server will automatically re-encrypt *F* with *TSi+k* without receiving any command from the data owner. For the duration of the process, the cloud server cannot get the contents of the cipertext and the new decryption keys. Only users with keys satisfying A and *TSi+k* will be able to decrypt *F*.

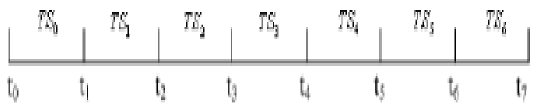
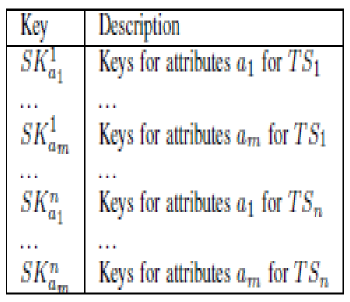


Fig. 2. Sample time slice

TABLE I
ALICE'S KEYS

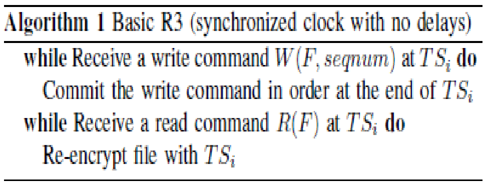


Key	Description
$SK^1_{a_1}$	Keys for attributes a_1 for TS_1
…	…
$SK^1_{a_m}$	Keys for attributes a_m for TS_1
…	…
$SK^n_{a_1}$	Keys for attributes a_1 for TS_n
…	…
$SK^n_{a_m}$	Keys for attributes a_m for TS_n

4.2 Protocol Description

Divide the description of the basic R3 scheme into three works: data owner initialization, data user read data and data owner write data. It will rely lying on the following functions. Table above shows the notations used in the description.

1) *Setup*() → (*PK;MK; s*) : At *TS*0, the data owner publishes the method public key *PK*, keeps the method master key *MK* secret, and sends the shared secret key *s* to the cloud.
2) *GenKey*(*PK;MK; s; PKAlice;A; T*) → (*SKAlice; {SKTAlice;A}*) : When the data owner wants to grant data user Alice attributes *A* with valid time period *T* , the data owner generates *SKAlice* and *{SKTAlice;A}* using the system public key, the system master key, the shared secret key, Alice's public key, Alice's attributes and eligible time.
3) *Encrypt*(*PK;*A*; s; TSt; F*) → (*Ct*A) : At *TSt*, the data owner encrypts file *F* with access structure A, and produces ciphertext *Ct*A using the system public key, access structure, the system secret key, time slice, and plaintext file.
4) *Decrypt*(*PK* ; *Ct*A ; *SKAlice* ; *{SKtAlice* ; *aij }* 1<= *j*<= *ni*) →*F* : At *TSt*, user *U*, who possesses version *t* attribute secret keys on all attributes in *CCi*, recovers *F* using the system public key, the user identity secret key, and the user attribute secret keys.



Algorithm 1 Basic R3 (synchronized clock with no delays)

```
while Receive a write command $W(F, seqnum)$ at $TS_i$ do
    Commit the write command in order at the end of $TS_i$
while Receive a read command $R(F)$ at $TS_i$ do
    Re-encrypt file with $TS_i$
```

5) *REncrypt*(*Ct*A*; s; TSt+k*) → *Ct+k*A : When the cloud server wants to return a data user with the file at *TSt+k*, it updates the ciphertext from *Ct*A to *Ct+k*A using the shared secret key.

4.2.1 Data owner initialization:

The data owner runs the *Setup* function to initiate the system. When the data owner wants to upload file *F* to the cloud server, it initially defines an access control A for *F*, and determines the current time slice *TSi*. Finally, it runs the *Encrypt* function with A and *TSi* to output the ciphertext. When the data owner wants to grant a set of attributes in a period of time to data customer Alice, it runs the *GenKey* function with *attributes* and *effective times* to create keys for Alice.

4.2.2 Data user read data:

When data user Alice wants to access file *F* at *TSi*, alice sends a read command *R*(*F*) to the cloud server, anywhere *F* is the file name. On getting the read command *R*(*F*), the cloud server runs the *REncrypt* task to re-encrypt the file with *TSi*. On receiving the ciphertext, Alice runs the *Decrypt* function using keys satisfying A and *TSi* to recover *F*.

4.2.3 Data owner write data:

When the data owner wants to write file *F* at *TSi*, it will send a write command to the cloud server in the form of: *W*(*F; seqnum*), everyplace *seqnum* is the order of the write command. This *seqnum* is necessary for ordering when the data owner issues multiple write commands that have to take place in one time slice. On receiving the write command, the cloud server will hand over it at the end of *TSi*. Algorithm 1 shows the actions of the cloud server.

V. CONCLUSION

In this paper propose that by using Reliable Re-encryption Scheme(R3 scheme), remove the condition of unreliability in cloud by using access control and access time with that a new addition of DES algorithm of cryptography that is using hybrid cryptography.

VI. ACKNOWLEDGMENTS

We like to express our appreciation to Mr. S. Deshmukh, for his guidance and encouragement.

REFERENCE

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage*," Financial Cryptography and Data Security,* 2010.

[2] Loknath S, Shivamurthy S, Bhaskar S and Shantgouda S, "Strong and Secure Re-Encryption Technique to Protect Data Access by Revoked Users in Cloud," *International Conference on Advances in Computer and Electrical Engineering* (ICACEE'2012) Nov. 17-18, 2012 Manila (Philippines).

[3] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," *in Proc. Of ACM CCS* (Poster 2010).

[4] Nimmi Gupta, "Implementation of Optimized DES Encryption," *ISSN 2249-6343 International Journal of Computer Technology and Electronics Engineering (IJCTEE)*2012.

[5] M.Srujana* S.Satya Narayana Y.Divya M.Girvani, "Reliable Proxy Re-encryption in Unreliable Clouds," *International Journal of Advanced Research in Computer Science and Software Engineering,* March 2013.

[6] A.Sahai and B. Waters, "Fuzzy identity-based encryption," *Advances in Cryptology–EUROCRYPT,* 2005.

[7] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and Fine grained data access control in cloud computing," *in Proc. of IEEE INFOCOM,* 2010.

[8] N. Antonopoulos and L. Gillam, "Cloud Computing: Principles, Systems and Applications," *Springer Publishing Company,* 2010.

[9] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM,* 2010.

[10] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute based encryption," *in Proc. of IEEE Symposium on S&P,* 2007.

[11] F. Cristian, "Probabilistic clock synchronization," *Distributed Computing,* 1989.

[12] K. Romer, "Time synchronization in ad hoc networks," *in Proc. of ACM MobiHoc*, 2001.

[13] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," *in Proc. Of ACM CCS,* 2006.

[14] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," *in Proc. of USENIX FAST*, 2003.

[15] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security,* 2006.