Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Reliable Proxy Re-encryption in Unreliable Clouds

M.Srujana* *Computer Science KLUniversity , India.*
S.Satya Narayana *Assoc prof, Computer Science KLUniversity,India.*
Y.Divya *Computer Science KLUniversity, India.*
M.Girvani *Computer Science KLUniversity, India.*

Abstract— ***In this paper, we propose an efficient data retrieval scheme using attribute-based encryption. The proposed scheme is best suited for cloud storage systems with substantial amount of data. It provides rich expressiveness as regards access control and fast searches with simple comparisons of searching entities. The proposed scheme also guarantees data security end-user privacy during the data retrieval process. A key approach to secure cloud computing is for the data owner to store encrypted data in the cloud, and issue decryption keys to authorized users. The cloud storage based information retrieval service is a promising technology that will form a vital market in the near future. Although there have been copious studies proposed about secure data retrieval over encrypted data in cloud services, most of them focus on providing the strict security for the data stored in a third party domain. However, those approaches require astounding costs centralized on the cloud service provider, this could be a principal hindrance to achieve efficient data retrieval in cloud storage.***

Keywords— ***Attribute-based encryption, cloud computing, Proxy re-encryption.***

I. Introduction

Cloud infrastructures can be roughly categorized as either private or public. In a private cloud, the infrastructure is managed and owned by the customer and located on-premise (i.e., in the customers region of control) [1]. In particular, this means that access to customer data is under its control and is only granted to parties it trusts. In a public cloud the infrastructure is owned and managed by a cloud service provider. This means that customer data is outside its control and could potentially be granted to entrusted parties. A key approach to secure cloud computing is for the data owner to store encrypted data in the cloud, and issue decryption keys to authorized users. Then, when a user is revoked, the data owner will issue re-encryption commands to the cloud to re-encrypt the data, to prevent the revoked user from decrypting the data, and to generate new decryption keys to valid users, so that they can continue to access the data. However, since a cloud computing environment is comprised of many cloud servers, such commands may not be received and executed by all of the cloud servers due to unreliable network communications.An alternative solution is to apply the proxy re-encryption (PRE) technique [2].Proxy re-encryption allows a proxy to convert 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.

For instance, Alice might wish to temporarily forward encrypted email to her colleague Bob, without giving him her secret key. In this case, Alice the delegator could designate a proxy to re-encrypt her incoming mail into a format that Bob the delegate can decrypt using his own secret key. Clearly, Alice could provide her secret key to the proxy but this requires an impracticable level of trust in the proxy.

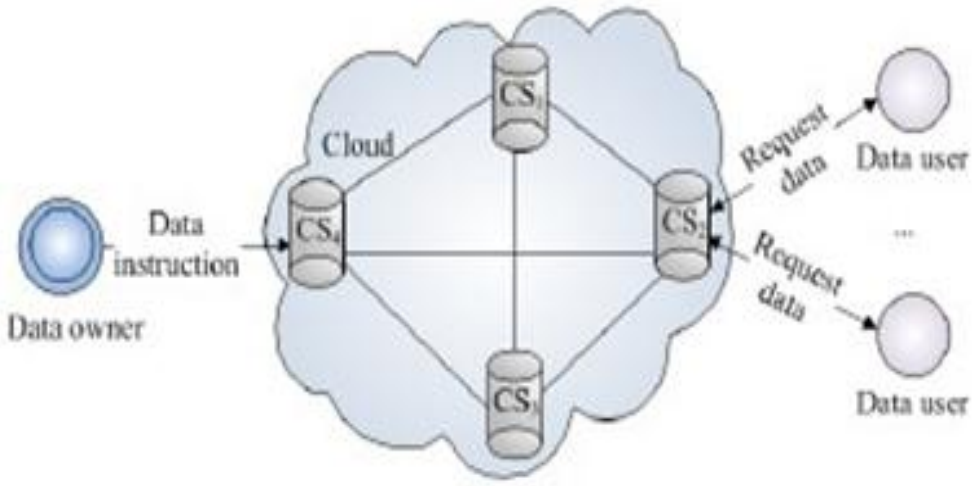

Fig. 1. A typical cloud environment

The primary advantage of PRE scheme 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 delegate – in order to allow a proxy to re-encrypt their cipher texts .In this schemes, only a inadequate amount of trust is placed in the proxy. For example, it is not able to decrypt the cipher texts it re-encrypts and we prove our schemes secure even when the proxy publishes all the re-encryption information it knows. This enables a number of applications that would not be sensible if the proxy needed to be fully trusted. To illustrate, let us consider a cloud environment shown in Fig. 1, where the data owner's data is stored on cloud servers CS_1, CS_2, CS_3, and CS_4. Assume that the data owner issues to CS_4 a re-encryption command, which should be propagated to CS_1, CS_2, and CS_3. Due to a network outage, CS_2 did not receive the command, and did not re-encrypt the data. At this time, if revoked users query CS_2, they can obtain the old cipher text, and can decrypt it using their old keys. A better solution is to allow each cloud server to autonomously re-encrypt data without receiving any command from the data owner. In this paper, a time based re-encryption scheme is proposed, which allows each cloud server to automatically Re-encrypt data based on its internal time. In this scheme the data is associated with an access control and an access time. Each user is issued keys associated with attributes and attribute effective times. The user can decrypt the data using the keys with attributes rewarding the access control, and access time. The data owner and the CSP share a secrete key with which each cloud server can re-encrypt data by updating the data access time. We propose a PRE scheme that achieves both the standard-model CCA security (i.e., without random oracles) and the proxy invisibility. To satisfy two security conceptions simultaneously, our construction is based on the ideas of the first standard-model CCA-secure (non-type-based) PRE scheme and the fully secure anonymous identity-based encryption scheme. In, the CHK methodology and the blind exponentiation technique that re-randomizes cipher texts via a blinding factor are employed for the standard-model CCA security. We use these techniques to achieve the standard-model CCA security. In, a private key generator of identity-based encryption combines the master key and an identity in the form of an inverse number for anonymity. Similarly, to generate a re-encryption key, we combine the delegator's private key and a type into an inverse form for proxy invisibility. The security proof of the proposed scheme is given in a formal security model.

II. Related Work

Many researchers have proposed storing encrypted data in the cloud to defend against the CSP. Their solution utilizes proxy re-encryption scheme (PRE),in which a delegator and a delegate generate a proxy key that allows a semi-trusted third party to convert cipher texts encrypted under delegator's public key into cipher texts which can be decrypted by delegate. Recently, proxy re-encryption has been shown very useful in a number of applications such as access control in file storage space, email forwarding, and law enforcement. ABE is a new cryptographic technique that resourcefully supports fine grained access control. Fine-grained access control systems assist granting differential access rights to a set of users and allow flexibility in specifying the access rights of individual users [3].Hierarchical attribute-based encryption (HABE) model by combining a HIBE system and a CP-ABE system, to provide fine-grained access control and full delegation. Our scheme relies on time to re-encrypt data. However, in a cloud, the internal clock of each cloud server may differ. There have been several solutions to this problem. For instance, proposed a probabilistic synchronization scheme, for reading remote clocks in networks subject to unbounded random message delays. The method can be used to improve the precision of both internal and external synchronization algorithms. Message delay used to synchronize the clocks of their host processors, time server processors communicate among themselves by sending messages via communication networks. To achieve loose synchronization in the cloud, and to determine the maximal time difference between the data owner and each cloud server by applying these techniques, our scheme can always achieve correct access control in unreliable clouds. ABE is a new cryptographic technique that efficiently supports fine grained access control. The combination of PRE and ABE was first introduced by, and extended by. In a hierarchical attribute-based encryption (HABE) scheme is proposed to achieve high performance and full delegation. The main difference between prior work and ours is that we do not require the underlying cloud infrastructure to be reliable in order to ensure correctness. Our scheme relies on time to re-encrypt data. However, in a cloud, the internal clock of each cloud server may differ. There have been several solutions to this problem. For instance, proposed a probabilistic synchronization scheme, which exchanges messages to get remote servers' accurate clocks with high probability [4]. Work by used message delay to estimate the maximal difference between two communicating nodes to synchronize the clocks.

Work by proposed a clock synchronization scheme for cloud environments, which uses an authoritative time source shared by all participants in a transaction to achieve clock synchronization between virtual cloud policy enforcement points. By applying these techniques to achieve loose synchronization in the cloud, and to determine the maximal time difference between the data owner and each cloud server, our R3 scheme can always achieve correct access control in unreliable clouds.

III. Preliminary

A. System Model

We assume that the system is composed of the following parties: the CSP, the trusted third party (TTP), enterprise users, end users, and internal trusted parties (ITPs). The first two parties: the CSP operates a large number of interconnected cloud servers with abundant storage capacity and computation power to provide high- quality services and the TTP is responsible for generating keys for the CSP and enterprise users [5].

B. Design model

The main design goal is to achieve scalable user revocation while protecting data security in cloud computing. Specifically, we categorize our goal into the following points:

- **Fine Grained Access control:** The data owner can specify expressive access structure for each data.
- **Data consistency:** This requires that all data users who request file F, should obtain the same content in the same time slice.
- **Data confidentiality:** The CSP and malicious users cannot recover data without the data owner's permission.
- **Cost Efficiency:** The re-encryption cost on the CSP is relatively low.

C. Adversary Model

There are two kinds of adversaries in the system: CSP, and malicious users. The CSP will correctly execute the protocol defined previously, but may try to gain some additional information about the stored data. The malicious user wants to access the data, to which he is not eligible to access. The communication channels are assumed to be secured under existing security protocols such as SSL to protect data security during information transferring. Note that both a CSP, and malicious users, can exist together. We assume that the CSP will not collude with any malicious user. However, malicious users may collude to obtain additional information.

This assumption is reasonable, and has also been made known in previous research, e.g., the proxy re-encryption system, where the semi-trusted proxy server is assumed to not between the CSP and the data owner. A data user, after being authenticated by the data owner, is granted a set of keys, each 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 T $S_1$to T S_n , she will be issued keys as is shown in Table I.

Key	Description
$_1SK_{a1}$	Keys for attributes a_1 to TS_1
$_1SK_{am}$	Keys for attributes a_m to TS_1
$_nSK_{a1}$	Keys for attributes a_1 to TS_n
nSK_{am}	Keys for attributes a_m to TS_n

The security requirements of the R3 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 same content in the same 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 that has not been requested by any data user should not be re-encrypted.

IV. Basic Reliable Proxy Re-Encryption Scheme

In the basic R3 scheme, we consider ideal conditions, 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.

A. Intuition

The data owner will first generate a shared secret key to the CSP. Then, after the data owner encrypts each file with the appropriate attribute structure and time slice, the data owner uploads the file in the cloud. The CSP will replicate the file to various cloud servers. Each cloud server will have a copy of the shared secret key [6]. Let us assume that a cloud server stores an encrypted file F with A and TS_i. When a user queries that cloud server, the cloud server first uses its own clock to determine the current time slice. Assuming that the current time slice is TS_i+k, the cloud servers will automatically re-encrypt F with TS_i+k without receiving any command from the data owner. During the process, the cloud server cannot gain the contents of the cipher text and the new decryption keys. Only users with keys satisfying A and TS_i+k decrypt F.

B. Protocol Description

We divide the description of the basic R3 scheme into three components: data owner initialization, data user read data and data owner write data. We will rely on the following functions.

1) Setup () → (P_K, M_K, s): At TS0, the data owner publishes the system public key PK, keeps the system master key MK secret, and sends the shared secret key s to the cloud.
2) GenKey(P_K,M_K,s,P_KAlice,A,T)→(S_KAlice,{S_KT Alice, A}): When the data owner wants to grant data user Alice attributes A with valid time period T , the data owner generates S_KAlice and {S_KT Alice, 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(P_K,A, s, TS_t, F) → (C_tA) : At TS_t, the data owner encrypts file F with access structure A, and produces cipher text CtA using the system public key, access structure, the system secret key, time slice, and plaintext file .
4) Decrypt(P_K, C_tA , S_KAlice, {S_Kt Alice,aij}1≤j≤ni) → F : At TS_t, 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.

5) REncrypt (C_tA, s, TS_{t+k}) → C_{t+k} A: When the cloud server wants to return a data user with the file at TSt+k, it updates the cipher text from C_tA to C_t+ A using the shared secret key.

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 first defines an access control A for F, and then determines the current time sliceT Si. Finally, it runs the Encrypt function with A and T S_i to output the cipher text [7]. When the data owner wants to grant a set of attributes in a period of time to data user Alice, it runs the GenKey function with attributes and effective times to generate keys for Alice.

2) Data user read data: When data user Alice wants to access file F at T S_i, she sends a read command R (F) to the cloud server, where F is the file name. On receiving the read command R (F), the cloud server runs the REncrypt function to re-encrypt the file with T S_i. On receiving the cipher text, Alice runs the Decrypt function using keys satisfying A and T S_i to recover F.

3) Data owner write data: When the data owner wants to write file F at T S_i, it will send a write command to the cloud server in the form of: W (F, seqnum), where 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 commit it at the end of T S_i.

V. Security Analysis

The Encrypt algorithm in the Time scheme is the same as the Encryption algorithm in HABE, which has been proven to be semantically secure [8]. Therefore, we consider that the Time scheme is secure if the following propositions hold:

Proposition 1: The keys produced by the GenKey algorithm are secure.

Proposition 2: The cipher text produced by the REncrypt algorithm is semantically secure.

Proposition 3: given the root secret key and the original cipher text, the CSP can know neither the underlying data, nor UAKs while executing re-encryption.

Access control correctness: It is clear that the correctness of access control is most vulnerable when a T S changes. Let us consider the case where Alice has keys with effective time up to T Si , and Bob has keys with effective time starting from T Si+1. Assuming that the data owner updates file F to F such that a user querying the file at T Si should obtain F, and a user querying the file at T Si+1 should obtain F'. The property of access control correctness fails if Alice is able to read F' (attack 1), or if Bob is able to read F (attack 2).

In attack 1, Alice's best time to launch an attack is just before ti+1, since she only has the keys to decrypt data up to T Si. However, the cloud server will commit the write command at ti+1 as long as its own clock is consistent with the data owner's clock, so that Alice never reads F', and thus her attack fails.

In attack 2, Bob's best time to launch an attack is just after ti+1. Querying earlier than ti+1 does not help Bob since he does not have the keys to decrypt the data. However, since the cloud server will commit the write command at ti+1 as long as its own clock is consistent with the data owner's clock, Bob will never access F, but only F'. Therefore, our scheme provides correct access control [9].

Data consistency: This property requires users that query within the same T S must receive the same data. Let us assume that both Alice and Bob have valid keys for the appropriate time slices, and we now want to show that so long as both Alice and Bob query within the same time slice, they must obtain the same data. Assuming that Alice and Bob both pick T Si to attack our scheme, the best attack time for Alice is to query just after t_i , and for Bob is to query just before t_i+1 . This attack is depicted in Fig. 3. According to the R3 scheme, the cloud server will return data that has been committed in t_i to both Alice and Bob. We first note that any write command that occurs after Alice and before Bob does not affect the correctness of the R3 scheme since this command will be committed at ti+1 .Furthermore, we have to ensure that all writes committed at t_i (what we are returning to Alice and Bob) must have already arrived before t_i . Since all parties' clocks are consistent and there are no delays, any write command committed at t_i can only be received by the cloud server before t_i. Thus, the data returned to Alice and Bob is consistent.

Data confidentiality: In our scheme, we only store encrypted data in the cloud. Since the R3 scheme preserves the data confidentiality operations from HABE scheme, and retain the same confidentiality properties, the cloud without knowledge of keys cannot learn any useful information about the stored data.

Data efficiency: The cloud server does not re-encrypt a file until a data user requests that file. Based on the properties of function REncrypt, when k > 1, we see that the cloud server can combine the re-encrypt operations until receiving a file access request.

Algorithm: Extended R3 (a synchronized clock with delays)

While Receive write commands W (F, t_i+1, seqnum) do
  If Current time is earlier than t_i+1 + α then
    Build Window i for file F
    Commit the write command in Window i at t_i+1 + α
  Else
    Reject the write command
    Inform the data owner to send write command earlier
While Receive a read request R (F, T S_i) do
  If Current time is later than t_i+1 + α then

Re-encrypt the file in Window i with T S_i

Else

Hold on the read command until $t_i+1 + \alpha$

VI. Additional Discussions

One concern with the R3 scheme design is that associating a different cipher text for every time slice will require users to manage a lot of keys. The number of keys that the R3 scheme requires is related to the actual length of the time slice. This length can be set according to different application requirements. Thus, an application that expects to revoke users on a monthly basis will have a longer time slice, and hence have far fewer keys, than an application where membership changes by the hour. Furthermore, issuing multiple keys up- front is actually more efficient. Consider an alternative design where each valid user was issued just one key.

Now, every time any user is revoked, the owner has to inform the CSP to re-encrypt the previous cipher text so as to prevent the revoked user from decrypting it again. The owner then has to update all of the remaining valid users with the new keys to allow them to decrypt the new cipher text. We argue that any scheme that stores encrypted data in the cloud has to deal with the issue of re-encryption and re-keying [10]. Since the remaining users may not be online all of the time, the re-keying process is arguably more costly.

A possible improvement is to let the owner issue a valid user a special seed value which the user can then use to generate keys on his own. The challenge here is to prevent the user from generating additional keys beyond what is authorized. This remains part of our future research.

Furthermore, we only let the data owner perform data updates. This is inflexible for applications where users may need to update the data as well. Our solution can be extended to allow users to perform data updates in addition to data owners. A ticketing scheme can be used. The data owner will issue and sign a timestamp to authorize the user to perform a write. The user will submit the ticket together with his updates to the CSP, which will then apply the updates. The challenge here is the time lag between when the data owner issues the ticket and when the user's request reaches the CSP. This time lag may be unknown since the user may delay sending his update to the cloud. An additional protocol will be required to allow the CSP to reject update requests that are too close to the time slice borders.

VII. Conclusion

In this paper, we proposed the Time based scheme to achieve fine-grained access control and scalable user revocation in a cloud environment. Our scheme enables each user's access right to be effective in a pre-determined period of time, and enable the CSP to re-encrypt cipher texts automatically, based on its own time. Thus, the data owner can be offline in the process of user revocations. The main problem with our scheme is that it requires the effective time periods to be the same for all attributes associated with a user. Although we provide a possible improvement, the users will be issued more UAKs. Our future work is to allow different effective time periods for different attributes associated with a user, without increasing the number of UAKs associated with each user.

References

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

[2] 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.

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

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

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

[6] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," Advances in Cryptology–EUROCRYPT, 1998.

[7] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in Proc. of ACM CCS, 2008.

[8] 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.

[9] 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

[10] S.Satyanarayana, T.Gopikiran, B.Rajkumar. "Cloud Business Intelligence" international journal of advanced and innovative research (ijair) volume 1 issue 6, 2012.

