IBIMA Publishing
Communications of the IBIMA
http://www.ibimapublishing.com/journals/CIBIMA/cibima.html
Vol. 2013 (2013), Article ID 462966, 10 pages
DOI: 10.5171/2013.462966

Research Article

Enhancing Data Privacy and Access Anonymity in Cloud Computing

Nabil Giweli, Seyed Shahrestani and Hon Cheung

School of Computing, Engineering and Mathematics, University Of Western Sydney, Sydney, Australia

Received 7 September 2012; Accepted 30 September 2012; Published 16 January 2013

Abstract

There is a growing interest in cloud computing due to its various benefits such as the efficient utilization of computing resources. However, privacy and security concerns are among the main obstacles facing the widespread adoption of this new technology. For instance, it is more desirable for many potential organizations and users that privacy protections and access authorizations on their data stored in the cloud remain under their control and only authorized entities can have access to the data even for the cloud server. In this paper, we propose a method that enables cloud clients more control of data security requirements on their data stored in the cloud. The data is protected by a client before it is sent to the cloud in a secure manner that only authorized users can access it. To provide a complete protection from unauthorized access, even the cloud provider is prevented from revealing the data content and access control policies. The client or data owner has complete control on what methods to use to protect the data and on who can have access on the data. The proposed method is based on a combination of cryptography techniques, including the Chines Remainder Theorem, symmetric and asymmetric encryptions. The proposed method combines access control and key sharing in one mechanism. In addition, the proposed method allows a client to use a unique key to encrypt the data and attaches it securely to its encrypted data. Only authorized users can have access to the key in order to decrypt the encrypted data. The data has all the security requirements independently attached to it including the integrity proof. The proposed method is efficient and has its computational overheard minimized. With all the security requirements and metadata stored with the data itself, the proposed method is also flexible and suitable for protecting clients' data in the cloud computing environment.

Keywords: Cloud computing, data privacy, confidentiality, access control, security.

Introduction

Managing the access control and privacy protection on data are still among the major obstacles in the widespread acceptance of cloud computing services. These services are shared by various users and usually accessed via a public network, such as the Internet. Data stored and transmitted in such an environment requires privacy and integrity protections as well as access control. The control over these security measures remain one of the major concerns. Some cloud providers such as Google have started to address these concerns for applications that require high levels of security, to be managed by the clients (Claburn 2009). There are also some situations where storage and handling of data must comply with government privacy regulations. This is for instance the case when dealing with Electronic Health Records (Haas et al. 2011). Recent surveys of security breaches show that companies such as Google, EMC/ RSA, Sony, UK National

Healthcare System (NHS) and Amazon EC2 have all experienced security incidents (Ko, Kirchberg and Lee 2011). Despite these incidents, providers of cloud services still argue that their service are more secure than in-house computing resources (Kshetri 2012).

In addition to concerns about security attacks by external entities, in some circumstances the data and applications may need to be protected from breaches by the cloud providers themselves. Generally speaking, when the data protection is managed by the cloud provider, they will normally have access to the data. Additionally, during the data transmission to a cloud server, the provider may be able to observe the client's activity patterns or even intercept and modify the data. These may include queries of stored data by the client or results of such queries sent back to the client. It is also possible for a provider to manipulate the requested operations for gaining some advantages (Tan, Liu and Wu 2011). In some cases, a cloud provider may be an honest but a curious operator (Samarati and Vimercati 2010). In other cases, a provider is considered to be trustworthy. They provide the needed services, including data availability, enforcing basic security control requirements, and processing queries on stored data and returning correct results to the queries. Due to poor security management, malicious actions from inside the cloud by an administrator or employee with access to clients' data can be carried out (Kui, Cong and Qian 2012).

To address the problems associated with security protection of data stored in a cloud, preserving the privacy and integrity of data stored in a cloud should not depend on the cloud provider. In other words, the techniques, which are used to provide the required security measures, should allow the clients to have complete control on the protection of their data, even from the service provider (Pearson 2012). Our approach, achieves these conditions based on the data centric security concept. In this approach, all security measures are attached to the data and remain protected from unauthorized access, even from the cloud provider.

In the proposed method reported here, protection measures are created and managed by the data owner and are tightly coupled with the data itself. This will prevent unauthorized access to data, when the provider is only responsible for executing the security policies of the data owner. The proposed method avoids using two layers of encryption, reducing the computational overhead and accelerating response times. The encryption key for each data set is unique. The key itself is securely attached to the data and only revealed to authorized users. As we will discuss shortly, these measures result in strong security levels for the data stored in a cloud server.

The rest of this paper is organized as follows. The next section establishes the motivations for this work. The third section identifies the security requirements for preserving privacy and integrity of data in the cloud. The fourth section describes our proposed approach and discusses how it meets the security requirements. The fifth section discusses the improvement of our work compared to previous solutions. The final section gives the conclusions.

Privacy in Cloud Environments

To preserve privacy of a client's data in the cloud, the data must be encrypted before it is sent to the cloud server (Hay, Nance and Bishop 2011; Hennessy et al. 2009; Tan, Liu and Wu 2011). The encrypted data is never decrypted at the cloud server to prevent any unauthorized entities from exposing the data contents and these unauthorized entities may include the cloud server (Agrawal, Abbadi and Wang 2012). For instance, Figure 1 shows that a user's data is protected by the data owner and shared with other users. In such a scheme, the privacy of data does not depend on an implicit assumption of trust on the cloud provider or on the service level of agreement (SLA). Instead, it depends on the cryptography techniques used by the client to protect the data (Vimercati et al. 2010). When an authorized user needs to access the data in the cloud, first he/she requests the data owner for an authorization to access the data. The data owner responds with the information that allows the user to request the data securely from the cloud provider. The access policies to the encrypted data are attached to the data and also protected. The cloud provider does not know the details of the policies and the role of the cloud provider is only to execute these policies. In addition, all these access activities, including the transmission of data between the cloud and the user are performed in a secure manner. Furthermore, attached to the data is the integrity proof for protecting the integrity of the data. This proof can be verified by the user who requests and receives the protected data from the cloud provider.

There are a number of approaches for securing the data on the cloud. Solutions that are based on encrypting the outsourced data before its transmission to the server are considered the most prevalent ones. Some of these solutions also tend to hide the access control parameters from the cloud provider. For instance, to protect the privacy of the outsourced data and to manage access to the data, a two-layered encryption model has been proposed by some researchers (Vimercati et al. 2007). In the first layer, encryption of data is performed by its owner. In the second layer, as part of the access control enforcement mechanism, another encryption is done by the cloud server. In this technique, the access control matrix is represented by a key derivation method. The structure of the key derivation method changes according to the variations in the access control policies. While this approach provides some clear benefits, its implementation faces several challenges. In particular, this approach is more computationally expensive as the encryption of data is performed twice. Also, the process of deriving the keys is considered to be complex. The two layer encryption approach is yet considered to achieve improved performance by reducing the need of re-encrypting the data when changing access control policies (Dai, Luo and Liu 2011; Kayem, Martin and Akl 2011). The method is also vulnerable to collusion attacks during access policy updates.

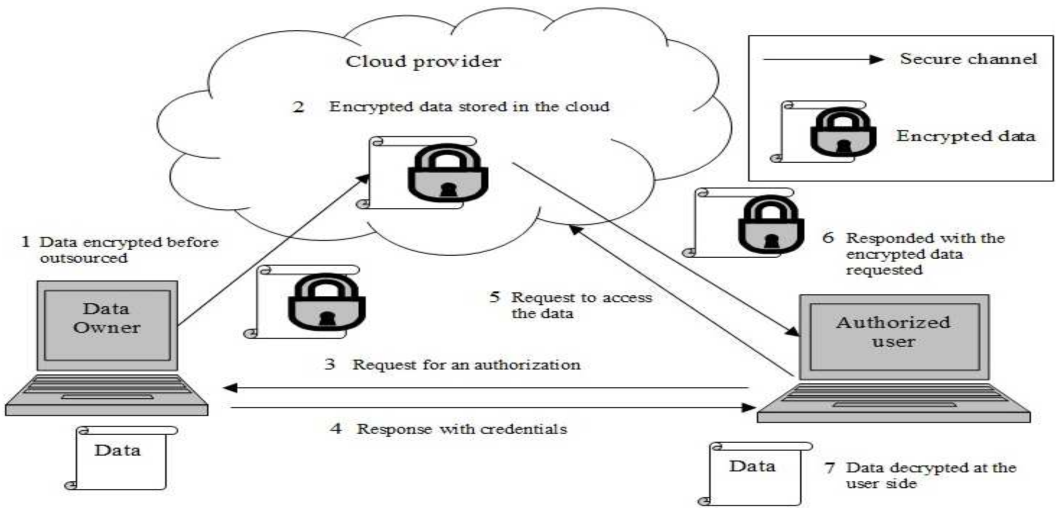
Cloud provider
2 Encrypted data stored in the cloud
Secure channel
Encrypted data
1 Data encrypted before outsourced
Data Owner
6 Responded with the encrypted data requested
5 Request to access the data
Authorized user
3 Request for an authorization
4 Response with credentials
Data
Data
7 Data decrypted at the user side

Fig 1. Basic Architecture of Preserving Data Privacy in the Cloud

Attribute-based encryption (ABE) is another approach proposed in the literature. It is considered in two types, namely Ciphertext-policy ABE, CP-ABE, and Key-policy ABE, KP-ABE. CP-ABE is more suitable than the other type for enforcing access control policies for outsourced data (Junbeom 2011). In the CP-ABE scheme, the encryption of data is limited to users who hold some specific attributes. These attributes are specified by the data owner during the process of the data encryption. However, efficient management of these attributes, particularly their revocations, is still one of the challenges facing this approach (Junbeom 2011). In addition, the ABE relies on the Bilinear Diffie-Hellman, which incurs a computational overhead during the encryption and decryption processes.

The Chinese Remainder Theorem (CRT) is proposed to enforce access control on outsourced data (Tourani, Hadavi and Jalili 2011). The CRT is mainly used for protecting the access control policies from being revealed by the server storing the data or by other users. Each user with his or her secret key can prove to the outsource server that they are authorized to access certain resources by computing a secret value from a shared value related to each resource. In comparison with the previous methods, the CRT approach has less computational complexity at both the client and server sides. That is because it does not require re-encrypting the data or a complex key derivation mechanism.

Several proposed solutions rely on using the same key for encryption of all resources (Dai, Luo and Liu 2011; Tourani, Hadavi and Jalili 2011). This can have serious security ramifications. Furthermore, sharing and distribution of the key in a secure manner, among all authorized users needs to be considered. In this situation, if this key is compromised, all resources lose their protection. Moreover, any user can collude with the cloud server to access a resource that they are not authorized to access. On the other hand, if the data owner uses a unique key to encrypt each resource, distribution of the keys to authorized users can be a problem. Our approach overcomes this problem, by enhancing the works previously reported (Kong et al. 2005; Tourani, Hadavi and Jalili 2011).

Cloud Security Requirements

To facilitate the development of a security approach to preserve privacy and integrity of data stored in a cloud server, we summarize the security requirements to be met in this section. These requirements can also be used for the development of similar approaches:

- The data is encrypted by the data owner and can be accessed by authorized users only.

- The data is self-protected and has the required security parameters attached to it.

- Access control parameters are hidden from the service provider and other users.

- The service provider does not know the number or identity of users who are authorized to access data sets.

- Unauthorized entities, including the service provider, cannot gain access to data or gain metadata about the data, when authorized operations are carried out on the data.

- The data set is self-contained for proving its integrity and authenticity to the authorized users.

- The key management should be such a way that the interactions between the data owner and authorized users are minimal.

In the next section, we describe our proposed approach and show how it achieves these requirements.

A Novel Approach to Cloud Data Privacy and Integrity

In our approach, to be discussed in this part, the CRT and the public key cryptosystem are utilized to share an encrypted value for each resource *r*, among authorized users. The resource is a data set or a file containing data of any type, including text, audio, image or video. This encrypted value contains two secret parameters. The first parameter, C_r, is a response to a challenge that only an authorized user can compute and provide to access the resource. The second parameter is a symmetric key, K_s, used to encrypt the resource. Parameters C_r and K_s are concatenated as $C_r||K_s$, and treated as one value. This value is encrypted with the public key of each authorized user, resulting in $a_i = E_{K_{pub\,i}}(C_r \parallel K_s)$ for the user *i*, where *i=1, 2, 3,….., k,* and *k* is the number of users authorized to access a particular resource *r* and $K_{pub\,i}$ is the public key of user *i*.

A data owner can use a different symmetric key, K_s, to encrypt a different file before sending it for storing in the cloud. They can then share this key securely and efficiently with authorized users. Neither the data owner nor the users need to keep or exchange this key, as each key will be securely attached to its relevant file. Therefore, if the symmetric key K_s of one resource gets compromised, other resources remain secure. The secret value C_r is used identify the user's authorized access to the resource *r*. The data owner will securely attach the parameter C_r to the resource and send it along with some shared value X_r. Only authorized users can calculate the parameter C_r using the shared value X_r. So, only authorized users can know and reveal C_r to the server and hence prove that they are authorized to access this particular resource. The parameter C_r is unique for each resource even if it relates to the same user for different resources. As such, if one C_r for a particular resource is compromised, no other resources will be affected. Moreover, this condition is useful if the data owner wants to change the access control list for a resource. For instance to grant access rights to a new user, the data owner only needs to change X_r for that resource. As the parameter C_r can remain the same, it is not necessary to resend a new C_r to the server. This in turn, will reduce the possibility of compromising this value while maintaining a dynamic mechanism for updating the access control list.

Applying CRT to Cryptography Access Control

The Chinese Remainder Theorem (CRT) can be used to securely share a secret key amongst authorized users. Based on the CRT, for any given integers, $a_1, a_2, \ldots, a_k$, the following system of simultaneous congruence has a unique solution *X*, such that $0 \le X < n = n_1 n_2 \ldots n_k$, provided the non-negative integers $n_1, n_2, \ldots, n_k$ are relatively prime.

$$\begin{gathered} X \equiv a_1 \; mod \; n_1 \\ X \equiv a_2 \; mod \; n_2 \\ \vdots \\ X \equiv a_k \; mod \; n_k \end{gathered} \tag{1}$$

Therefore, when X is known, each a_i can be calculated by the equation $a_i = X \; mod \; n_i$, for $i = 1,2,\ldots\ldots,k$. The proof of this theorem could be found in several number theory references, for example (Yan. 2003).

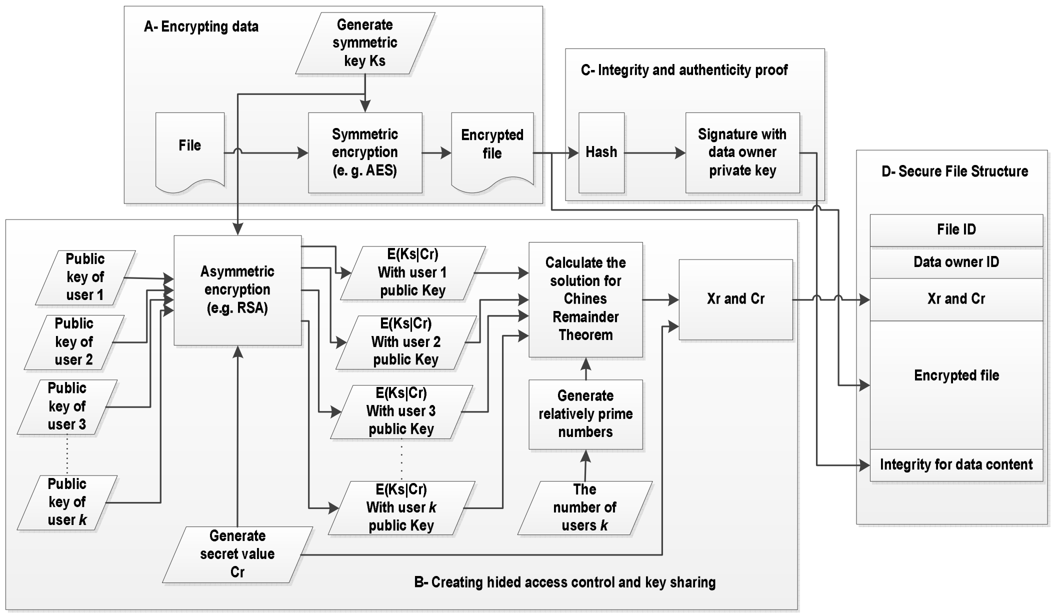
A- Encrypting data
Generate symmetric key Ks
File
Symmetric encryption (e. g. AES)
Encrypted file
C- Integrity and authenticity proof
Hash
Signature with data owner private key
D- Secure File Structure
File ID
Data owner ID
Xr and Cr
Encrypted file
Integrity for data content
Public key of user 1
Public key of user 2
Public key of user 3
Public key of user k
Asymmetric encryption (e.g. RSA)
E(Ks|Cr) With user 1 public Key
E(Ks|Cr) With user 2 public Key
E(Ks|Cr) With user 3 public Key
E(Ks|Cr) With user k public Key
Calculate the solution for Chines Remainder Theorem
Generate relatively prime numbers
The number of users k
Xr and Cr
Generate secret value Cr
B- Creating hided access control and key sharing

Fig 2. Preserving Data Privacy and Integrity: Processes at the Data Owner Side

To apply this theorem to our proposed method, for subjects $s_1, s_2, \ldots, s_k$ which in our case are authorized users, each user is provided with a unique relatively prime number $n_i = n_1, n_2, \ldots,$ or n_k, where k is the number of authorized users. We next describe the operations of the CRT at the data owner side.

To allow k users to access a resource r, the resource r is encrypted with a symmetric key K_s, and then the solution for the following congruence is calculated.

$$
\begin{gathered}
X_r \equiv \left(E_{K_{pub\,1}}(C_r \parallel K_s)\right) mod\ n_1 \\
X_r \equiv \left(E_{K_{pub\,2}}(C_r \parallel K_s)\right) mod\ n_2 \\
\vdots \\
X_r \equiv \left(E_{K_{pub\,k}}(C_r \parallel K_s)\right) mod\ n_k
\end{gathered}
\qquad (2)
$$

The solution value X_r is the shared value for the resource r, C_r is the secret value for that resource and K_s is the symmetric key for encrypting this resource r, and $\left(E_{K_{pub\,i}}(C_r \parallel K_s)\right)$ is the ciphertext resulting from encryption with the public key $K_{pub\,i}$ of the user i. Each user is assigned their own n_i where all n_i for $1 \leq i \leq k$ are relatively prime. Clearly, if the secret parameters C_r or K_s related to the resource r get compromised, other resources remain secure. Additionally, the data owner can manage each resource independently of other resources. The processes explained in this section are illustrated in Figure 2.

Encrypted File Structure and Integrity Verifications

To provide integrity protection on outsourced data, a data owner signs the content digest $H(F)$ of each encrypted file F with his or her private key, $K_{prv\,0}$ and attaches the resulting digital signature to the file. This is shown in Figure 2. When the data owner wants to grant access rights to a new user for this file, it is not necessary to re-hash the encrypted file content and re-sign the resulting digest. In addition, this action is not necessary when the data owner needs to revoke the access privilege of a user for accessing this file. In the above cases, the symmetric encryption key K_s used to encrypt the file content needs to be changed. The symmetric key K_s, needs to be changed, if a revoked user has already accessed the file. In this case, the user knows K_s for this file and can collude with the server to access the file.

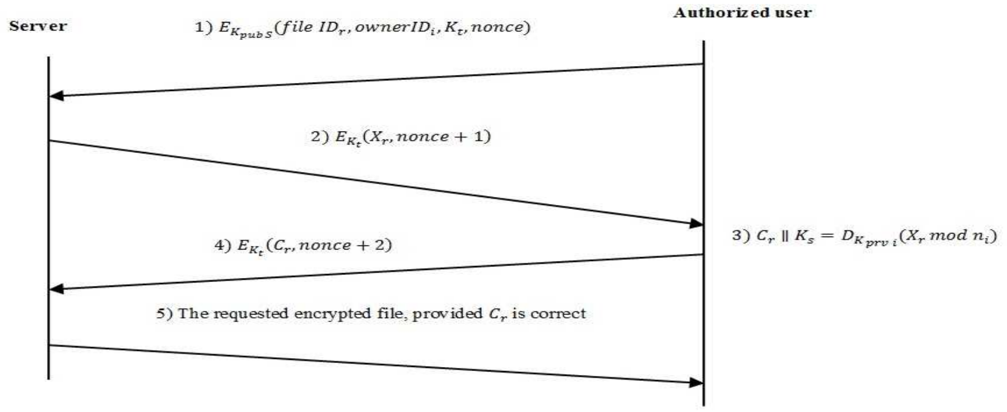
Server
Authorized user
1) $E_{K_{pub\,S}}(file\ ID_r, ownerID_i, K_t, nonce)$
2) $E_{K_t}(X_r, nonce+1)$
3) $C_r \parallel K_s = D_{K_{prv\,i}}(X_r\ mod\ n_i)$
4) $E_{K_t}(C_r, nonce+2)$
5) The requested encrypted file, provided C_r is correct

Fig 3: Proposed Access Control Procedure

The resulting encrypted file is shown in section D of Figure 2. It includes the file ID, owner ID, $X_r||C_r$, encrypted file content and digital signature of the encrypted file content digest. To achieve higher levels of security and efficiency, the data owner can add themselves as the user for all files too. In that way, the data owner does not need to keep all the symmetric keys and C_r parameters, as they can be calculated from its shared value X_r that is already stored in the cloud for each file. The data owner can easily verify the integrity of the shared value X_r for each file by calculating the $C_r||K_s$ using Equation (3) and checking that C_r matches the parameter attached to that file. The data owner can also do that by verifying that the key K_s can successfully decrypt the encrypted file.

Proposed Access Control Procedure

The cloud server stores the encrypted data and the relevant metadata prepared by the data owner. The provider has access to the metadata, but not the contents of the file, integrity proof and access control parameters. The provider can also read the secret parameter C_r for each file, but the number or identities of users that can access the file remain hidden to the provider. Moreover, this information remains hidden even when a user is accessing the file. When a user requests to access a file, the server will provide the requester with the shared value X_r, as a challenge. If the user is authorized to access this file, he or she is able to calculate the secret parameter C_r based on the challenge X_r and to return the secret parameter, C_r, to the server. The server compares the value of the C_r received with the one attached to the file. If they are the same, the server will send the file for this user.

To facilitate the needed calculations, it can be noted that when a User i is authorized to access resource r, the data owner must have already sent the user the relatively prime n_i to the user. After the user receives X_r from the server, the user can determine K_s and C_r from the following equation:

$$C_r \parallel K_s = D_{K_{prv\,i}}(X_r \, mod \, n_i) \tag{3}$$

Where $D_{K_{prv\,i}}$ is the decryption operation using the private key of user i, and n_i is the relatively prime number given to this user by the data owner.

The messages exchanged between an authorized user and the cloud server hosting shared files are shown in Figure 3. In our proposed scheme, when User i needs to access the resource r, the user sends a request to the server, containing the ID of the resource, ID_r, the file owner's ID_i, a *nonce*, and a session key K_t. all encrypted with the public key of the server $E_{K_{pub\,S}}$. The result is represented as $E_{K_{pub\,S}}(ID_r,\ ID_i, K_t\, , nonce)$. The session key K_t is created by the user for securing the information exchange between the user and the server. The server locates the file with ID_r, reads its shared value X_r, increment the *nonce* to (*nonce* +1) and encrypts them with the session key K_t. The result, represented as $E_{K_t}(X_r,\ \ nonce + 1)$ is returned to the user. After receiving the requested information from the server, the user calculates the secret parameter C_r using Equation (3) and returns C_r and (*nonce* + 2) to the server encrypted with the session key K_t, i.e., $E_{K_t}(C_r,\ \ nonce + 2)$. The server checks C_r and, if there is a match, sends the encrypted file to the user. The user can decrypt the file by key K_s calculated by using Equation (3).

Granting Access and Revoking User Privileges

To grant a new user access to a resource r, a new congruence is added to the CRT, as shown in Equation (4). In such a case, the owner needs to recalculate the shared value X'_r:

$$\begin{aligned} X'_r &\equiv \left(E_{K_{pub\,1}}(C_r \parallel K_s)\right) mod\ n_1 \\ X'_r &\equiv \left(E_{K_{pub\,2}}(C_r \parallel K_s)\right) mod\ n_2 \\ &\vdots \\ X'_r &\equiv \left(E_{K_{pub\,k}}(C_r \parallel K_s)\right) mod\ n_k \\ X'_r &\equiv \left(E_{K_{pub\,k+1}}(C_r \parallel K_s)\right) mod\ n_{k+1} \end{aligned} \tag{4}$$

Where $K_{pub\,k+1}$ is the public key of the new user. It can be shown that the solution of a set of congruence can be used efficiently to find a new solution for the same set plus a new modulus (Kong et al. 2005), i.e,, it is more efficient to find the X'_r by solving

$$\begin{aligned} X'_r &\equiv X_r \text{mod } n_1 n_2 \ldots n_k \text{ and} \\ X'_r &\equiv \left(E_{K_{pub\,k+1}}(C_r \parallel K_s)\right) \text{mod } n_{k+1} \end{aligned} \tag{5}$$

The data owner sends the new shared value X'_r with the file ID and data owner ID to the server to replace the old ones. The granting mechanism is efficient, as it needs only one asymmetric encryption operation for a fixed length of information, i.e. $C_r \parallel K_s$, and a solution for only two congruences. Moreover, the granting mechanism is not disclosing any secret value as the exchanged values are not considered to be secret. Although, the shared value X'_r is based on secret values, only authorized users can reveal them.

To revoke the access privileges of a user *k* to resource *r*, the owner has to change the secret value C_r to C'_r and to re-calculate X''_r for the remaining users from 1 to *k-1* as follows:

$$\begin{aligned} X''_r &\equiv \left(E_{K_{pub\,1}}(C'_r \parallel K_s)\right) mod\ n_1 \\ X''_r &\equiv \left(E_{K_{pub\,2}}(C'_r \parallel K_s)\right) mod\ n_2 \\ &\vdots \\ X''_r &\equiv \left(E_{K_{pub\,k-1}}(C'_r \parallel K_s)\right) mod\ n_{k-1} \end{aligned} \tag{6}$$

The data owner sends the new C'_r and X''_r to the server with the file ID and data owner ID. If the user has accessed the data before revocation, it is possible that the user and the server can collude to access the data after revocation. Even if the data owner re-encrypts the data with a new symmetric key, the collusion attack is still possible as the server may have a copy of the data encrypted with the previous secret key. However, the scheme does not reveal the identity of the users to the server and that may reduce the chance of this type of collusion attacks.

Discussions

Our method, compared to the previous work in (Kong et al. 2005; Tourani, Hadavi and Jalili 2011), is more secure because we use a different symmetric key to encrypt each file and different secret value to enforce the access control for each file. This improvement reflects in several security and privacy aspects as follows:

- A user cannot access the encrypted data before he/she provides securely to the secret value to the server and without revealing his or her identity.
- If a symmetric key K_s and/or secret value C_r of one file got compromised other files remains secure.
- Our approach is more robust against possible collusion attacks between revoked users and the cloud server. Because each file is encrypted with a different symmetric

key, the server could not collude with a revoked user to reveal the data content if the revoked user has not accessed it before revocation. However, if the revoked user has accessed the file and revealed the symmetric key K_s, the data owner could delete the previous file and uses a new key K_s to create the new one. Therefore, if the server is prevented from keeping a copy of the deleted files, then it cannot collude with revoked users.

- When granting a new user to a file, the owner does not need to exchange the secret value C_r with the server, but only the shared value X_r has to be exchanged. Hence, the possibility of compromising the value C_r is reduced. In addition, for revoking a user, the data owner only needs to change the C_r of that file because there are no other files using the same old value.

- When the data owner adds himself as a user for all files, as we recommend, the data owner and authorized users only needs to keep their public-key pairs safe.

- The method provides integrity and authenticity verifications for each file and the parameters do not require changing during revoking and granting process.

- All the security requirements are attached to each file independently, which allows both cloud provider and client to deal with each file more flexibly and efficiently.

The computational overhead for all these improvements depend on the length in bits of the secret value Cr and the encryption algorithms used in the system. These factors could be carefully designed to get less impact on the efficiency of our method. In general, computing the Chines Reminder Theorem solution is efficient compared to other cryptography methods used in cryptography access control as it uses simple modulus operations without exponential operation (Wang, Wei and Hu 2009).

Conclusions

Cloud computing paradigm offers on-demand cost-effective computing and storage resources. However, preserving privacy of data outsourced to the cloud and giving the clients more control over the security of their data are required. In this paper, we proposed a method to achieve these goals. Our approach allows the cloud users to manage the privacy and integrity protections for their data stored in the cloud without relying on the trustworthiness of the provider. Moreover, the users manage the access control policies for their data without revealing them to the provider. The cloud server is only responsible for enforcing the access control policies without revealing the authorized users' identities. In our method, we increase the security of data by allowing users to encrypt each file, before outsourced to the cloud, with a unique symmetric key without an additional overhead to manage these keys. Our approach attaches the secure information needed to the file so authorized users could share, access and verify the integrity and authenticity of each file independently. Therefore, our method provides the flexibility to deal individually with each protected file, which is expected to be more suitable for the cloud environment. Our future work will be concentrated in adding more security features, such as the capability to search securely the encrypted data, and on investigating implementation issues for the practical use of the proposed method.

References

Agrawal, D., El Abbadi, A. & Wang, S. (2012). "Secure and Privacy-Preserving Data Services in the Cloud: A Data Centric View," *Proc. VLDB Endow.,* 5 (12), 2028-9.

Claburn, T. (2009). "Google Plans Private Government Cloud," Informationweek - Online. [Online], *via OxResearch; ProQuest Central; ProQuest Nursing & Allied Health Source* [Retrieved June 8, 2012], http://www.informationweek.com/government/cloud-saas/google-plans-private-government-cloud/220000732.

Dai, J., Luo, S. & Liu, H. (2011). "A Privacy-Preserving Access Control in Outsourced Storage Services," In Proceedings of the Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference On, 3 (247-51).

Di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S. & Samarati, P. (2007). "A Data Outsourcing Architecture Combining Cryptography and Access Control," *Proceedings of the Proceedings of the 2007 ACM Workshop on Computer Security Architecture, Fairfax, Virginia, USA.*

Haas, S., Wohlgemuth, S., Echizen, I., Sonehara, N. & Müller, G. (2011). "Aspects of Privacy for Electronic Health Records," *International Journal of Medical Informatics,* 80 (2), E26-E31.

Hay, B., Nance, K. & Bishop, M. (2011). "Storm Clouds Rising: Security Challenges for Iaas Cloud Computing," In Proceedings of the System Sciences (HICSS), 2011 44th Hawaii International Conference On, (1-7).

Hennessy, S. D., Lauer, G. D., Zunic, N., Gerber, B. & Nelson, A. C. (2009). "Data-Centric Security: Integrating Data Privacy and Data Security," *IBM Journal of Research and Development,* 53 (2), 2:1-2:12.

Hur, J. & Noh, D. K. (2011). "Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems," *IEEE Transactions on Parallel and Distributed Systems,* 22-1214-21.

Kayem, A. V. D. M., Martin, P. & Akl, S. G. (2011). "Efficient Enforcement of Dynamic Cryptographic Access Control Policies for Outsourced Data," *In Proceedings of the Information Security South Africa (ISSA),* 2011, (1-8).

Ko, R. K. L., Kirchberg, M. & Lee, B. S. (2011). "From System-Centric to Data-Centric Logging - Accountability, Trust & Amp; Security in Cloud Computing," *In Proceedings of the Defense Science Research Conference and Expo (DSR),* 2011, (1-4).

Kong, Y., Seberry, J., Getta, J. R. & Yu, P. (2005). A Cryptographic Solution for General Access Control Information Security, In J. Zhou, J. Lopez, R. Deng and F. Bao (Eds), *Springer Berlin / Heidelberg,* 3650 (461-73).

Kshetri, N. (2012). "Privacy and Security Issues in Cloud Computing: The Role of Institutions and Institutional Evolution," *Telecommunications Policy.*

Pearson, S. (2012). "Privacy, Security and Trust in Cloud Computing," P*rivacy and Security for Cloud Computing,* 3-42.

Ren, K., Wang, C. & Wang, Q. (2012). "Security Challenges for the Public Cloud," *Internet Computing, IEEE,* 16 (1), 69-73.

Samarati, P. & di Vimercati, S. D. C. (2010). "Data Protection in Outsourcing Scenarios: Issues and Directions," *Proceedings of the Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China.*

Tan, C. C., Liu, Q. & Wu, J. (2011). "Secure Locking for Untrusted Clouds," In Proceedings of the Cloud Computing (CLOUD), 2011 IEEE International Conference on, (131-8).

Tourani, P., Hadavi, M. A. & Jalili, R. (2011). "Access Control Enforcement on Outsourced Data Ensuring Privacy of Access Control Policies," In Proceedings of the High Performance Computing and Simulation (HPCS), 2011 International Conference on, (491-7).

Vimercati, S. D. C. D., Foresti, S., Jajodia, S., Paraboschi, S. & Samarati, P. (2010). "Encryption Policies for Regulating Access to Outsourced Data," *ACM Trans. Database Syst.,* 35 (2), 1-46.

Wang, B., Wei, Y. & Hu, Y. (2009). "Fast Public-Key Encryption Scheme Based on Chinese Remainder Theorem," *Frontiers of Electrical and Electronic Engineering in China,* 4 (2), 181-5.

Yan., S. Y. (2002). Number Theory for Computing, In Computers & Mathematics with Applications, Second Edition edn, *Elsevier Ltd,* 46 (502-3).