2012 International Conference on Industrial and Intelligent Information (ICIII 2012)
IPCSIT vol.31 (2012) © (2012) IACSIT Press, Singapore

Remote Data Checking and Dynamic Access Control schemes for Outsourced Data in Cloud

Purushothama B R and B B Amberker

Department of Computer Science and Engineering,
National Institute of Technology Warangal,
Andhra Pradesh - 506004, INDIA

Abstract. Cloud Computing paradigm provides resources of computing infrastructure as services over the Internet. Along with the several benefits it offers it suffers from various security risks. Data outsourced onto the honest-but-curious public cloud servers should be protected from unauthorized access including the cloud service providers. Data should be encrypted to maintain confidentiality. There is need for access control mechanism to provide data access selectively to the authorized users. We propose a dynamic access control scheme using *multilinear map* that requires users to store only a single key irrespective of the number of data items to which user has authorized access. The cloud server being dishonest might erase the data of the owner for its benefits. We provide a bandwidth saving scheme that provides the proof of data possession.

Keywords: Cloud Computing, Data Outsourcing, Access Control, Data Checking.

1. Introduction

Cloud Computing is a computing paradigm wherein resources of the computing infrastructure are provided as services over the Internet. Increase in demand for the computing resources have made several organizations (clients) to outsource their storage and computing needs. Data Outsourcing allows the clients to store their at an external cloud service provider which provides means for clients to access the outsourced data.

Cloud Computing being economic offers various types of services.

- ***Infrastructure as a Service (IaaS):*** The customer makes use of the cloud service provider's storage, computing and networking infrastructure. A good example of IaaS is Amazon's Elastic cloud (Amazon EC2) [1].
- ***Platform as a Service (PaaS):*** Platform comprises of development tools and runtime environment. The cloud customer leverages the providers development tools to run the custom applications. Examples include Microsoft Azure [2] and Google Apps.
- ***Software as a Service (PaaS):*** Customers use application software that is hosted on the provider's infrastructure. Examples include Google docs and web mail services.

There are two categories of cloud infrastructure.

1) ***Private cloud infrastructure:*** In this category the infrastructure is owned and managed by the customer (organization) and located on-premise (in the customer's region of control). The access to the data is controlled by the customer and the access is granted to the trusted parties. We hope, the use of users, customers, data owners, service providers will be clear from the context.
2) ***Public cloud infrastructure:*** In this category the cloud service provider owns and manages the infrastructure and the data is located off-premise (in the service provider's region of control). The access to the data could be potentially granted to the untrusted parties as the data is not controlled by the customer but the service provider.

The advantage of outsourcing data is related to the costs of building in-house infrastructure versus outsourced hosting. Outsourcing provides significant cost savings and service benefits. By outsourcing, the data owner can concentrate on the core business. Due to outsourcing of data on to the public cloud storage, the sensitive data will be no more in the control of the data owner. The confidentiality and integrity of the data will be at risk.

1.1. Our Contributions

1) The server often behaves maliciously for gaining benefits. It might erase the data of the owner in the process. There is a need to check the data possession at the server. We provide a method to check the data possession wherein the size of the proof is constant (essentially an element of the group under consideration) irrespective of the number of blocks queried. The size of the proof generated by the server is less in comparison with the scheme in [5].
2) Often the data owner wants to give access permission selectively to the data items among the authorized users [4]. It requires the enforcement of the access control policy. We design a scheme for enforcing the access control scheme that does not require the involvement of the data owner in processing the authorized users query.

2. Data Possession Checking Scheme

2.1. Notations and the scheme

Let $F = \{b_1, \dots, b_f\}$ be the data to be outsourced.

$$\theta : \{0,1\}^{k} \times \{0,1\}^{\log f} \rightarrow \{0,1\}^{l}$$

$$\pi: \{0,1\}^{k} \times \{0,1\}^{\log f} \rightarrow \{0,1\}^{\log f}$$

Suppose G_1 and G_2 be the multiplicative groups of order q. The generators of G_1 and G_2 are g and $e(g, g)$ respectively. Define $e : G_1 \times G_1 \rightarrow G_2$ be the bilinear map. $\forall a, b \in \mathbb{Z}$, $e(g^a, g^b) = e(g, g)^{ab}$.

The public cryptographic hash function h.

2.2. Scheme

1) Data owner publishes $\{\boldsymbol{g^{b_i}}\}$, $1 \leq i \leq f$ and sends $\boldsymbol{F}$ to the server.
2) The owner randomly chose $\boldsymbol{t_1, t_2} \in \mathbb{Z}_q^*$ and prepares the challenge $\boldsymbol{C} = (\boldsymbol{c}, \boldsymbol{g^{t_1}}, \boldsymbol{g^{t_2}}, \boldsymbol{k_1})$. Sends the challenge to the service provider and $\boldsymbol{t_1, t_2}$ are kept secret by owner.
3) Service provider generates the proof as below: $\boldsymbol{For\ 1 \leq j \leq c}$
   - Compute indices $\boldsymbol{i_j = \pi_{k_1}(j)}$. Compute :

$$\boldsymbol{V = e(g^{t_1}, g^{t_2})^{b_{i_1}} \dots e(g^{t_1}, g^{t_2})^{b_{i_c}}}$$

$$\boldsymbol{= e(g, g)^{t_1 t_2 (b_{i_1} + \dots + b_{i_c})}}$$

   Sends $\boldsymbol{h(V)}$ to the user.
4) The user checks the proof as below: $\boldsymbol{T = 1;\ For\ 1 \leq j \leq c}$

$$\boldsymbol{i_j = \pi_{k_1}(j)}.$$

$$\boldsymbol{T = Te(g^{t_1}, g^{b_{i_j}})^{t_2}}$$

Verification: If $\boldsymbol{h(T) = h(V)}$ then it is a proof that the server possesses the data blocks queried in the challenge.

The size of the proof generated by the server is less in comparison with the scheme in [5].

2.3. Adversarial argument:

The security of the scheme is based on the intractability of solving the discrete log problem. If server solve discrete log problem then he can "cheat" the data owner. We will not be able to provide the proof due to space constraints.

3. Dynamic access control to the outsourced data using multilinear map

In this section, we present a scheme that supports dynamic access control in which the access privileges of the users to the data objects can be granted and revoked. The definitions used in this section are taken from [3].

The scheme comprises of following algorithms.

Setup (A,t): Let U be the set of users and D be the set of data items such that $|U| = n$ *and* $|D| = m$. A is the access control matrix of order $n \times m$ each row corresponding to a user in U and is defined by

$$A[u_i, d_j] = \begin{cases} 1, & if\ ui\ is\ not\ authorized\ to\ access\ dj \\ 0, & if\ ui\ is\ \ authorized\ to\ access\ dj \end{cases}$$

It lists all acl_{d_i}, where $d_i \in D$ and sets

$$k = max_i\{|acl_{d_i}|\}$$

Run the randomized *multilinear map generator algorithm* $G = G(t, n)$ to obtain (Γ, g, q) . Let $e: G_1^n \rightarrow G_2$ be the *multilinear map* defined by Γ. G_1 and G_2 are multiplicative groups. The generator of G_1 is g and q is the order of G_1 and G_2. We assume *discrete log problem* in G_1 is intractable.

Let $U = \{u_1, \dots u_n\}$ be the user set. Let $D = \{d_1, \dots, d_m\}$ such that $\forall\, i\,, d_i \in G_2$ be the data set to be outsourced. As acl_{d_i} being the set of users authorized to access the data item d_i, let acl_{d_i} be the number of users who have authorized access to d_i. Set

$$l = min_i\{|acl_{d_i}|\}$$

Choose $n - l$ numbers randomly from $\mathbb{Z}_q$. Let these numbers be $\{s_1, \dots, s_{n-l}\}$. Output the public parameters $P_{pub} = (\Gamma, g, q, g^{s_1}, \dots, g^{s_{n-l}})$.

User Key Generation, KGU (g,q): A user $u's$ key pair is of the form $pk_u = g^{\alpha}$ and $sk_u = \alpha$, where $\alpha \leftarrow \mathbb{Z}_q \backslash \{0\}$. Let Alice be the owner of the data to be outsourced. The key pairs corresponding to the Alice be $(sk_O, pk_O) = (a, g^a)$.

Outsourced Data Encryption Key Generation, $\boldsymbol{EKG(A, k, l, d_i, sk_O, P_{pub})}$: Let $k = |acl_{d_i}|$. Suppose $k = n$. W.l.o.g let $acl_{d_i} = \{u_{i_1}, \dots, u_{i_k}\}$. Let $\{g^{\alpha_{i_1}}, \dots, g^{\alpha_{i_k}}\}$ be corresponding public keys of the users in acl_{d_i}. Compute key K_i

$$\begin{aligned} K_i &= e(g^{\alpha_{i_1}}, \dots, g^{\alpha_{i_k}})^a \\ &= e(g, \dots, g)^{a\alpha_{i_1} \dots \alpha_{i_k}} \end{aligned}$$

Suppose $k \neq n$ and $k < n$ and $acl_{d_i} = \{u_{i_1}, \dots, u_{i_j}\}$, $j < n$. Compute K_i as,

$$\begin{aligned} K_i &= e\left(g^{\alpha_{i_1}}, \dots, g^{\alpha_{i_j}}, g^{s_1}, \dots, g^{s_{n-j}}\right)^a \\ &= e(g, \dots, g)^{a\alpha_{i_1} \dots \alpha_{i_j} s_1 \dots s_{n-j}} \end{aligned}$$

Output K_i .

***Data Encryption*, $E(d_i)$:** The data owner runs $EKG(A, k, l, d_i, sk_O, P_{pub})$ to get K_i. Encrypt d_i using K_i as,

$$C_i = K_i d_i$$

***Outsourced Data Decryption Key Generation*, $DKG(A, k, l, C_i, sk_{u_j}, P_{pub})$:** W.l.o.g let $acl_{d_i} = \{u_{i_1}, \dots, u_{i_{k-1}}, u_j\}$. Let $\{g^{\alpha_{i_1}}, \dots, g^{\alpha_{i_{k-1}}}\}$ be the corresponding public keys of the users in $\{u_{i_1}, \dots, u_{i_{k-1}}\}$. Note that C_i is the encrypted object of the data d_i. Let $|acl_{d_i}| = k$ and suppose $k = n$. Compute the key K_i as

$$\begin{aligned} K_i &= e(g^a, g^{\alpha_{i_1}}, \dots, g^{\alpha_{i_{k-1}}})^{\alpha_j} \\ &= e(g, \dots, g)^{a\alpha_{i_1} \dots \alpha_{i_{k-1}}\alpha_j} \end{aligned}$$

Suppose $k \neq n$ and $k < n$ and $acl_{d_i} = \{u_j, u_{i_1}, \dots, u_{i_s}\}$, s+1< n. Compute K_i as,

$$\begin{aligned} K_i &= e(g^a, g^{\alpha_{i_1}}, \dots, g^{\alpha_{i_s}}, g^{s_1}, \dots, g^{s_{k-s-1}})^{\alpha_j} \\ &= e(g, \dots, g)^{a\alpha_j\alpha_{i_1} \dots \alpha_{i_j} s_1 \dots s_{k-s-1}} \end{aligned}$$

Output K_i .

***Data Decryption*, $D(C_i)$:** Run Outsourced Data Decryption Key Generation algorithm, $DKG(A, k, l, C_i, sk_{u_j}, P_{pub})$ to get the key K_i used to decrypt C_i. Compute as below to get the d_i.

$$\begin{aligned} d_i &= (C_i / K_i) \\ &= K_i d_i / K_i. \end{aligned}$$

3.1. Granting access privileges to the authorized users

We address the following access privilege changes to the data items stored on the *honest-but-curious* server and give the corresponding algorithms to handle the access privilege changes.

The users in acl_{d_i} are authorized to access the item $d_i \in D$ along with the Alice the data owner. The data object is encrypted as detailed above to enforce this access control on d_i. Any user $u \notin acl_{d_i}$ \$ will not able to access d_i. Let $acl_{d_i} = \{u_{i_1}, \dots, u_{i_k}\}$ such that $k \leq n$. If $k = n$, then all the users have privileges to access d_i. Suppose $k < n$, it means not all users in U have access to d_i. In other words, there are users $u_{i_j} \notin acl_{d_i}$ but $u_{i_j} \in U$.

Suppose a user $u_{i_j} \notin acl_{d_i}$ but $u_{i_j} \in U$ be given "privilege/authorization" to access d_i. The data owner Alice should modify the access privileges so that d_i can be accessed by $acl_{d_i} \cup u_{i_j}$. The following algorithm is executed by the Alice.

***Grant Privilege Update*, $PU(A, d_i, sk_O, u_{i_j})$:** Suppose $k = |acl_{d_i}| < n$ for d_i and the user $u_{i_j} \notin acl_{d_i}$ is to be given access to d_i. The data owner Alice updates the access control matrix A to A' by making $A[u_{i_j}, d_i] = 1$. It should be noted that this entry was 0 in A. A' is the modified access control matrix. To enforce the changed access control to d_i wherein the users in $acl_{d_i}' = acl_{d_i} \cup u_{i_j}$ should access d_i , Alice does the following.

Alice decrypts $K_i d_i$ where K_i is computed using the private key of Alice and the public key information of users in acl_{d_i}. The new key K_i' for acl_{d_i}' is computed as below.

$$\begin{aligned} K_i' &= e(g^{\alpha_{i_1}}, \dots, g^{\alpha_{i_k}}, g^{\alpha_{i_j}}, g^{s_1}, \dots, g^{s_{n-k-1}})^a \\ &= e(g, \dots, g)^{a\alpha_{i_1} \dots \alpha_{i_k}\alpha_{i_j} s_1 \dots s_{n-k-1}} \end{aligned}$$

Alice computes $C_i' = d_i K_i'$ and sends the same to the server. C_i' can be accessed only by the users in acl_{d_i}' .

3.2. Revoking access privileges of an user

Another privilege change operation we address is removing the existing authorization/privilege of the user $u_{i_j} \in acl_{d_i}$ to access the data d_i. The user u_{i_j} should be made unauthorized to access d_i. Alice executes the following algorithm.

Revoke Privilege Update, $DP\left(A, d_i, P_{pub}, sk_O, u_{i_j}\right)$: The data owner Alice updates the access control matrix A to A' by making $A\,[u_{i_j}, d_i\,] = 0$. It should be noted that this entry was 1 in A. A' is the modified access control matrix. To enforce the changed access control to d_i wherein the users in ${acl_{d_i}}' = acl_{d_i} - u_{i_j}$ should access d_i, Alice does the following.

- Data owner picks $\mathrm{l} = \min_{\mathrm{i}}\{|\mathrm{acl_{d_i}}|\}$ random numbers from $\mathbb{Z}_{\mathrm{q}}$. Say, $\mathrm{s_{n-l+1,}} \ldots \mathrm{s_n}$.
- Data owner decrypts d_i by computing K_i using his private key and the public information (elaborated in the decryption key generation algorithm).
- Let $k = |acl_{d_i}|$. Suppose d_i was encrypted for the users in acl_{d_i} such that $k = n$. Alice computes for $K_i' = {acl_{d_i}}'$ as,

$$K_i' = e\left(g^{\alpha_{i_1}}, \ldots, g^{\alpha_{i_{j-1}}}, g^{\alpha_{i_{j+1}}}, \ldots, g^{\alpha_{i_k}}, g^{s_1}\right)^a$$
$$= e(g, \ldots, g)^{a\alpha_{i_1} \ldots \alpha_{i_{j-1}}\alpha_{i_{j+1}} \ldots \alpha_{i_k} s_1}$$

The data item d_i is encrypted as $C_i' = d_i K_i'$ and is stored on the server.

- Suppose d_i was encrypted for users in acl_{d_i} such that $k < n$. Then, the K_i generated to encrypt d_i had used public information of $k < n$ users in $acl_{d_i} = \{u_{i_1}, \ldots, u_{i_k}\}$ along with the public information $\{g^{s_j}\}, j = 1, \ldots, n-k$. To enforce the access privileges as in ${acl_{d_i}}'$, Alice computes K_i' as below,

$$K_i' = e\left(g^{\alpha_{i_1}}, \ldots, g^{\alpha_{i_{j-1}}}, g^{\alpha_{i_{j+1}}}, \ldots, g^{\alpha_{i_k}}, g^{s_1}, \ldots, g^{s_{n-k+1}}\right)^a$$
$$= e(g, \ldots, g)^{a\alpha_{i_1} \ldots \alpha_{i_{j-1}}\alpha_{i_{j+1}} \ldots \alpha_{i_k} s_1 \ldots s_{n-k+1}}$$

The data item d_i is encrypted as $C_i' = d_i K_i'$ and is stored on the server.

3.3. Dynamic access control enforcement by data owner without downloading the data

In the scheme presented in previous section, Alice the data owner downloads the encrypted data item to alter the access privileges. The process of downloading the data item from the server to make changes to update the access privileges and again storing the data after encryption to the server is costly. In this section, we focus on avoiding the need of Alice downloading the data item. We have assumed the server to be *honest-but-curious*. The server is honest in carrying out the operations specified by Alice. Under this assumption, we explain how the access privileges of the data items can be changed.

Suppose $acl_{d_i} = \{u_{i_1}, \ldots, u_{i_k}\}$ such that $k < n$. The key K_i used to encrypt the data item d_i is

$$K_i = e(g^{\alpha_{i_1}}, \ldots, g^{\alpha_{i_k}}, g^{s_1}, \ldots, g^{s_{n-k}})^a$$
$$= e(g, \ldots, g)^{a\alpha_{i_1} \ldots \alpha_{i_k} s_1 \ldots s_{n-k}}$$

The access privileges changes are carried out as explained below.

- Suppose Alice wants to authorize $u_{i_j} \notin acl_{d_i}$ but $u_{i_j} \in U$ to access d_i. Alice computes,

$$K_i' = e\left(g^{\alpha_{i_1}}, \ldots, g^{\alpha_{i_k}}, g^{\alpha_{i_j}}, g^{s_1}, \ldots, g^{s_{n-k-1}}\right)^a$$
$$= e(g, \ldots, g)^{a\alpha_{i_1} \ldots \alpha_{i_k}\alpha_{i_j} s_1 \ldots s_{n-k-1}}$$

Alice computes,

$$\delta = \frac{K_i'}{K_i}$$

Alice sends $\delta \in G_2$ to the server. Server computes the following

$$C_i' = C_i\, \delta = K_i d_i \delta = d_i K_i'$$

The server won't learn any information about d_i as server will not be able to compute either K_i or K_i'. The C_i' can be decrypted only by the users in $acl_{d_i}\ U\ u_{i_j}$ to get d_i.

The same procedure can be used to remove the access privilege of a user $\boldsymbol{u_{i_j} \in acl_{d_i}}$ to the data item $\boldsymbol{d_i}$.

3.4. Security

The security of the scheme is based on the *multilinear Diffie-Hellman Assumption* [3]. The proof is not provided because of space constraints.

4. Acknowledgements

This work was supported by *Ministry of Human Resource Development*, Government of India.

5. Conclusion

We have proposed two schemes to address security problems associated with outsourced data in cloud. The data possession proof for checking whether the data is kept or erased by the server. The scheme generates a proof which saves network bandwidth. The dynamic access control scheme using *multilinear map* is designed to enforce access control on the outsourced data. Only authorized users will be able to access the data. The users are supposed to store only single key to access the authorized data items.

6. References

[1] "Amazon elastic compute cloud (amazon ec2)." [Online]. Available: http://aws.amazon.com/ec2/.

[2] "Microsoft windows azure platform." [Online]. Available: http://www.microsoft.com/windowsazure.

[3] D. Boneh and A. Silverberg. "Applications of Multilinear Forms to Cryptography". In: *Contemporary Mathematics, American Mathematical Society.* 2003, Vol. 324, pp. 71-90.

[4] Nigel P. Smart. "Access Control using Pairing Based Cryptography". *In. Proc. of the 2003 RSA conference on the cryptographer's track*, 2003.

[5] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson and Dawn Song, ``Provable data possession at untrusted stores". In: *Proc. of the 14th ACM conference on Computer and communications security*. 2007, pp. 598-609.