An Efficient Access Control Scheme for Outsourced Data

Xiaoming Wang
Department of Computer Science, Jinan University, Guangzhou ,China
Email: wxmsq@eyou.com

Yanchun Lin
Department of Computer Science, Jinan University, Guangzhou ,China
Email: yhl@you.com

***Abstract*—In this paper, we analyze Liu et al.'s scheme and show that their scheme is not secure. Then we modify their scheme and present an efficient access scheme to outsourced data. Our scheme adopts two-layer encryption model and all users are divided into different groups according to their access privilege. We employ filter functions to construct the key derivation procedure to prevent the revoked users from getting encryption key. Our scheme can realize dynamic access control without shipping outsourced data back to the data owner when group membership is changed. Moreover, our scheme can grant and revoke the users without updating any secret key of other users. Our scheme avoids publishing many public tokens for deriving the encryption key. The analysis of security and performance shows that our scheme is secure and efficient.**

***Index Terms*—outsourced databases, dynamic access control, key derivation**

I. INTRODUCTION

The management of large databases is quite expensive, as it needs not only storage capacity, but also skilled personnel. An emerging solution to this problem is outsourced database. Providing secure and efficient access control to outsourced data is very important for outsourced databases.

Many research schemes[1-11] have dealt with the issues of access control to outsourced data. Recently, to manage evolving access control requirements and re-encrypted outsourced data without having to ship outsourced data back to the data owner, the over-encryption model[6] is presented. In the over-encryption model, the data owner and the server collaborate to encrypt data and the data owner only needs to publish some tokens. However, one potential limitation of the over-encryption scheme is that it may require publishing too many tokens when the number of users is large[8]. In 2008, Liu et al. proposed an over-encryption in outsourced databases using secret sharing[8]. Their scheme uses the two-layers encryption model[6] in order to avoid the need for shipping resources back to the data owner for re-encryption when security requirements change. In their scheme, resources are divided into different sets based on access control lists, and each set corresponds to a distinct encryption key. Every user can use his/her corresponding key to derive the encryption key in order to access the resource. However, we consider Liu et al's scheme is not secure. In their scheme, if two users share a resource, their scheme employs the two users as a subset to build a binary linear function for deriving the encryption key. They randomly choose some points (x, y) on this function and assign as a key pair to other users, and choose another a point (x_p, y_p) on this function as public token to publish. Each user uses the public token (x_p, y_p) together with his/her key pair to derive the encryption key. However, any user can also reconstruct the function using the public token (x_p, y_p) and his/her key pair, the user may compute many key pairs for many unauthorized users, thus the unauthorized users can access the resource. Obviously, it is not accepted in real world.

To deter the attack, we modify Liu et al.'s scheme and present an efficient access control scheme to outsourced data. Our scheme also adopts the two-layer encryption model[6,8] and all users are divided into different groups according to their access privilege. Each user of the group can directly derive the encryption key by a public token of the group and cannot obtain the encryption key of other groups. We employ filter functions to construct the key derivation procedure to prevent the revoked users from getting encryption key. Our scheme can realize dynamic access control without shipping outsourced data back to the data owner when group membership is changed. Our scheme can grant and revoke the users without updating any secret key of other users. Moreover, our scheme avoids publishing many public tokens for deriving the encryption key. The analysis of security and performance shows that our scheme is secure and efficient.

The rest of the paper is organized as follows. Section 2 gives a brief analysis for Liu et al.'s scheme. We will present an efficient access scheme to outsourced data in section 3. In section 4, the analysis of security and performance is provided. Finally, the concluding remarks are given.

doi:10.4304/jcp.7.4.918-922

II. LIU ET AL.'S SCHEME ANALYSIS

A. Liu et al.'s scheme

Liu et al.'s scheme[8] uses the over-encryption mode. Data is doubly encrypted at the base encryption layer and the surface encryption layer.

1) Base encryption Layer

Suppose to grant the access for the data R to the users A, B and C. First the data owner randomly chooses two pairs of keys (X_a, Y_a) and (X_b, Y_b) as master keys, and assigns them to the users A and B. Next the data owner generates the derivation function based on these two pairs as $f(x) = \alpha x+K_{ab}$. Solving the equation, K_{ab} is the encryption key for the users A, B and C to access the data R. The data owner now can randomly choose (X_c, Y_c) on this function and assign as a key pair to the user C, and pick up another point (X_p, Y_p) to publish as a token. If there are more users who need to share the resource R, the data owner simply chooses more points and assigns them to the users. The each of A, B and C can derive the encryption key K_{ab} using his/her own key pair combined with the token and then access the data.

2) Surface encryption Layer

Assuming the users A, B, and C have the access to the same data R. The data owner randomly assigns (X_a, Y_a) , (X_b, Y_b) and (X_c, Y_c) as the key pairs to the users A, B and C, then regards (A, B) as a subset and defines a function $f(x) = \alpha x+K_{ab}$, and chooses another point (X_{p1}, Y_{p1}) on this function to publish as a token. The data owner uses a transfer key ($h(K_{ab})$, $h^2(K_{ab})$) as the key pair for the subset together with the key pair (X_c, Y_c) of the user C, and has another deriving function $g(x) = \beta x + K_{abc}$. The data owner chooses another point (X_{p2}, Y_{p2}) on the function $g(x)$ to publish as a token. The user A or B can use his/her own key together with the public token (X_{p1}, Y_{p1}) to regenerate this function $f(x)$ to get the encryption key K_{ab}, and then use their transfer key ($h(K_{ab})$, $h^2(K_{ab})$) and the public token (X_{p2}, Y_{p2}) to get the encryption key K_{abc}. The user C can use his/her own key together with the public token (X_{p2}, Y_{p2}) to get the encryption key K_{abc}.

B. Analysis of Liu et al.'s scheme

In Liu et al.'s scheme, any user can reconstruct the derivation function using the public token together with his/her key pair. However, if a user obtains the derivation function, he/her may choose many points on this function as the pair of keys and assigns these keys to unauthorized users for profits, thus the unauthorized users can derive the encryption key using the pair of keys combined with the public token, thereby the unauthorized users can access the data. Therefore, their scheme is not secure.

Suppose to grant the access for the data R to the users A and B. The data owner randomly chooses two pairs of keys (X_a, Y_a) and (X_b, Y_b) as master keys, and assigns them to the users A and B. Next the data owner generates the derivation function based on these two pairs as $f(x) = \alpha x+K_{ab}$. K_{ab} is the encryption key for the users A and B to access the data R. The data owner chooses another a point (X_p, Y_p) on this function as public token to publish. Therefore, the user A or B can reconstruct the derivation function $f(x)$ using the public token (X_p, Y_p) together with his/her key pair. The user A or B may choose more points on the derivation function $f(x)$ and assigns them to the unauthorized users for profits, thus the unauthorized users can access the restricted data. Obviously, it is not allowed in real world.

III. PROPOSED SCHEME

Our scheme uses the DAS (database as a service) model. This model is mostly suitable for one-to-many group where there is a data owner, a server and a large number of users. The data owner is responsible for producing, distributing, and updating encryption keys. The server is responsible for producing the query result on the encrypted outsourced data, and sending the encrypted result to the user. The user decrypts the result from the server using the decryption key in order to get the plaintext result.

We assume that the data owner defines an access control policy to regulate access to outsourced data. All the users of outsourced data are divided into different groups according to their access privilege. Table 1 illustrates an example of access control policy with six users (A, B, C, D, E, F) and five data (r_1, r_2, r_3, r_4, r_5), where there tree groups G_1 =(A, B, C), G_2=(D, E), G_3=(A, B, C,F). The users A, B, $C \in G_1$ can access the data (r_1, r_2), the users D, $E \in G_2$ can access the data r_3, the users A, B, C, $F \in G_3$ can access the data (r_4, r_5).

Table 1 access control policy

Group	User	Data R
Group G_1	A,B,C	r_1, r_2
Group G_2	D,E	r_3
Group G_3	A,B,C,F	r_4, r_5

Let p, q be distinct large primes and $q|(p-1)$, g be a generator which is an element of Z_p^* with an order q (i.e., $g^q = 1 \bmod p$). $h(.)$ is a secure one-way hash function. The data owner, the server and each user u_i have respectively a pair of keys such as (ε_j , $y_j = g^{\varepsilon_j} \bmod p$) for $j=o, s, i$. For the sake of simplicity, we illustrate our scheme by table 1 definition example.

A. Encryption data

To delegate policy changes enforcement to the server, avoiding re-encryption for the data owner, we adopt the two-layer encryption model [6,8]. The data owner encrypts the data and sends them to the server in encrypted form. The server can impose another layer of encryption.

1)Base encryption

The base encryption provides initial protection for preventing the server from reading data. The data owner performs following base encryption before transmitting the data to the server.

a) The data owner first chooses a symmetric encryption key $K_d \in Z_q^*$ and encrypts respectively $R_1=(r_1, r_2)$, $R_2=(r_3)$ and $R_3=(r_4, r_5)$ such as $E_{K_d}(r_1)$, $E_{K_d}(r_2)$, $E_{K_d}(r_3)$, $E_{K_d}(r_4)$, $E_{K_d}(r_5)$. Where $E_{K_d}(\cdot)$ denotes an encryption operation with a symmetric encryption key K_d using a symmetrical encryption algorithm such as AES.

b) The data owner chooses secret keys k_i (i=*A, B, C, D, E, F*), and computes $\lambda_i = y_i^{\varepsilon_O} \bmod p$, $v_i = E_{\lambda_i}(k_i \| K_d)$ for i=*A,B,C,D,E,F*. Then the data owner sends respectively v_i to the authorization users *A,B,C,D,E* and *F*.

c) The data owner chooses the symmetric encryption keys K_1, K_2, K_3 and random numbers $\zeta, \alpha, \beta \in Z_q^*$, and generates public tokens as follows

The group G_1: $F_{G_1}(x) = K_1 + \zeta \prod_{i \in G_1}(x - h(k_i))$,

the group G_2: $F_{G_2}(x) = K_2 + \alpha \prod_{i \in G_2}(x - h(k_i))$,

the group G_3: $F_{G_3}(x) = K_3 + \beta \prod_{i \in G_3}(x - h(k_i))$,

Publishes the tokens $(F_{G_1}(x), F_{G_2}(x), F_{G_3}(x))$.

d) The data owner computes

$$\lambda_s = y_s^{\varepsilon_O} \bmod p, v_s = E_{\lambda_s}(K_1 \| K_2 \| K_3),$$

then sends v_s to the server.

e) The server computes $\lambda_s = y_O^{\varepsilon_s} \bmod p$, and decrypts v_s to get the symmetric encryption keys K_1, K_2 and K_3.

2) Surface Encryption

The server performs the surface encryption over the base encryption in order to enforce the dynamic changes over the policy.

The server computes respectively

$$E_{K_1}(E_{K_d}(r_1) \| E_{K_d}(r_2)), \quad E_{K_2}(E_{K_d}(r_2)),$$
$$E_{K_3}(E_{K_d}(r_4) \| E_{K_d}(r_5)),$$

then deposited them in the server.

B. Data Access

The authorization users can derive the encryption key by using the public tokens. For the users *A,B,C* $\in G_1$, they can derive the encryption key K_1 by the token

$$F_{G_1}(x) = K_1 + \zeta \prod_{i \in G_1}(x - h(k_i)),$$

that is, the users *A,B,* and *C* computes respectively

$$F_{G_1}(h(k_A)) = K_1 + \zeta \prod_{i \in G_1}(h(k_A) - h(k_i)) = K_1,$$
$$F_{G_1}(h(k_B)) = K_1 + \zeta \prod_{i \in G_1}(h(k_B) - h(k_i)) = K_1,$$
$$F_{G_1}(h(k_C)) = K_1 + \zeta \prod_{i \in G_1}(h(k_C) - h(k_i)) = K_1.$$

Thus the users *A, B* and *C* get the surface encryption key K_1. Because they already have the base encryption key K_d, therefore they can obtain data $R_1=(r_1, r_2)$.

For the users *D, E* $\in G_2$, they can derive the encryption key K_2 by the token

$$F_{G_2}(x) = K_2 + \alpha \prod_{i \in G_2}(x - h(k_i)),$$

that is, the users *D* and *E* computes respectively

$$F_{G_2}(h(k_D)) = K_2 + \alpha \prod_{i \in G_2}(h(k_D) - h(k_i)) = K_2,$$
$$F_{G_2}(h(k_E)) = K_2 + \alpha \prod_{i \in G_2}(h(k_E) - h(k_i)) = K_2,$$

so that each of them can derive the encryption key K_2 using their own key k_i by the token and then access the data $R_2=(r_3)$.

For the users *A, B, C, F* $\in G_3$, they can derive the encryption key K_3 by the token

$$F_{G_3}(x) = K_3 + \beta \prod_{i \in G_3}(x - h(k_i))$$

that is, for the users *A,B,C,F* $\in G_3$, they can derive the encryption key K_3 by using their key such as

$$F_{G_3}(h(k_A)) = K_3 + \beta \prod_{i \in G_3}(h(k_A) - h(k_i)) = K_3,$$
$$F_{G_3}(h(k_B)) = K_3 + \beta \prod_{i \in G_3}(h(k_B) - h(k_i)) = K_3,$$
$$F_{G_3}(h(k_C)) = K_3 + \beta \prod_{i \in G_3}(h(k_C) - h(k_i)) = K_3,$$
$$F_{G_3}(h(k_F)) = K_3 + \beta \prod_{i \in G_3}(h(k_F) - h(k_i)) = K_3,$$

so that each of them can derive the encryption key K_3 by the token and then access the data $R_3=(r_4, r_5)$.

C. Grant Access Right

When a user is authorized to read a data, the data owner will change its access control policy and update corresponding public tokens.

For example, to grant the user *W* to data r_4 and r_5, we need to add the secret key $h(k_W)$ of the user *W* to the token $F_{G_3}(x)$ and modify $F_{G_3}(x)$ as follows.

$$F_{G_3}(x) = K_3 + \beta \prod_{i \in G_3 \cup W}(x - h(k_i))$$

Then the data owner updates the public token $F_{G_3}(x)$ of group G_3, The all encryption keys and other public tokens do not need to be changed.

The data owner computes

$$\lambda_W = y_W^{\varepsilon_O} \bmod p, \; v_W = E_{\lambda_W}(k_W \| K_d),$$

then sends v_W to the user *W*.

D. Revocation Access Right

To delete some users or revoke malicious users from some data, the surface encryption layer has to be over-encrypted by a new encryption key, but there is no operation on base encryption layer.

For example, to delete the user *C* from data (r_4, r_5), that is, the only users *A, B* and *F* can access the data (r_4, r_5). There is no operation on base encryption layer, but the surface encryption layer of the data (r_4, r_5) has to be over-encrypted by a new encryption key. The data owner

generates the new key K_3' and encrypts K_3' such as $v_s = E_{\lambda_s}(K_3')$, and sends v_s to the server. Then the server computes $\lambda_s = y_o^{\varepsilon_s} \bmod p$ and decrypts v_s to get the symmetric encryption key K_3'. The server encrypts the data ($E_{K_d}(r_4)$, $E_{K_d}(r_5)$) with K_3' such as

$$E_{K_3'}(E_{K_d}(r_4) \| E_{K_d}(r_5)).$$

The data owner modifies public token $F_{G_3}(x)$ such as

$$F_{G_3}(x) = K_3' + \beta \prod_{i \in G_3 \setminus C} (x - h(k_i))$$

Then data owner updates the public token $F_{G_3}(x)$ of the group G_3, The other encryption keys and public tokens do not need to be changed.

IV. Analysis of Security and Performance

A. Analysis of security

1) In our scheme, the server cannot read the data owner's outsourced data in any cases since the outsourced data are protected by the base encryption. Each authorized user can directly get the base encryption key from the data owner and directly decrypt the base encryption, but the server cannot get the base encryption key. In same reason, the outsiders cannot read the data owner's outsourced data. Therefore, the base encryption prevents the server and the outsider from accessing the data.

2) In our scheme, the surface encryption enforces the dynamic control. An authorized user can only access the part that the data owner allowed him to read and cannot access whole database. Because all users are divided into different groups according to their access privilege, and the data of different group is encrypted by different encryption key. Each user of the group can directly derive the surface encryption key by the public token of the group and cannot obtain the surface encryption key of other groups. For an example, the surface encryption key of the group G_1' data is K_1, an authorized user $A \in G_1$ can directly derive the encryption key K_1 from the public token $F_{G_1}(x)$ since

$$F_{G_1}(h(k_A)) = K_1 + \zeta \prod_{i \in G_1} (h(k_A) - h(k_i)) = K_1.$$

But a user $F \notin G_1$ cannot derive the encryption key K_1 from the public token $F_{G_1}(x)$ since

$$F_{G_1}(h(k_F)) = K_1 + \zeta \prod_{i \in G_1} (h(k_F) - h(k_i)) \neq K_1$$

Without K_1, the user $F \notin G_1$ cannot decrypt the surface encryption, so the user F cannot get the data even if he has already the base encryption key K_d. Therefore, our scheme assures that no one except the permitted users can read the data owner's outsourced data and has data confidentiality.

3) In our scheme, a revoked user will never be able to access restricted data. When a user $A \in G_1$ is revoked from the group G_1, the server has to re-perform the surface encryption with a new encryption key K_1', and modifies the public token $F_{G_1}(x)$ as

$$F_{G_1}(x) = K_1' + \zeta \prod_{i \in G_1 \setminus A} (x - h(k_i)).$$

It is computationally infeasible for the revoked user A to get any information about K_1' since his secret key $h(k_A)$ is excluded from $F_{G_1}(x)$. Therefore, the revoked user A cannot get the encryption key K_1', thus the revoked user A is prevented from accessing constrained data.

B. Analysis of performance

1) In our scheme, all users are divided into different groups according to their access privilege. Consider a general case with one group of n users. At initial time, in the base encryption, our method does not need any public token and each user can directly get the encryption key from the data owner. However, Liu et al. scheme needs to construct one linear function and one public token, each user must get a secret key from the data owner and directly derives the encryption key from the public token using the secret key. So the average step for each user is 1. In surface encryption, our scheme just needs a public token and each user can directly derive the surface encryption key by the public token. So the average step for each user is 1. Liu et al. scheme needs to construct n-1 linear functions, so the public tokens' number is n-1. So the average steps to get surface encryption key for each user is $(n^2+n-2)/2n$.

For granting a new user to a data, that is, a new user is added to a group G. In the base encryption, our scheme directly assigns the base encryption key to the new user. It does not need to any action and public tokens. However, Liu et al. scheme needs to add a new function for the new user to derive the encryption key and a new public token. The average step is 1. In surface encryption, our scheme only needs to add the hash value of the new user's secret key to the public token of the group G. And for the new user, it needs 1 step to get the surface encryption key. For the users in previous group, their surface encryption key is no change. But Liu et al. scheme needs to consider the previous set as a subset, and create a new function for the subset and new user. And for the user in previous set, it need two steps to get the surface encryption key, for the new user, need 1 step, so average steps is $(2n + 1)/(n + 1)$.

For revoking in the base encryption layer, our scheme does not any action as Liu et al. scheme. For revoking in the surface encryption layer, it needs a new encryption key to replace the surface encryption key. Our scheme only needs to remove the revoked user's secret keys from the public token of the group and update the public token of the group. So the total token number of updating and re-publishing is 1. Each user only needs 1 step to get the new surface encryption key. However, Liu et al.'s scheme needs to rebuild some the transfer keys and some tokens should recomputed and re-public. In this situation, the average token number of updating and re-publishing is $(3n^2-3n-2)/2n$. Each user needs $(3n^2+4n-6)/4n$ average steps to get the new surface encryption key.

2) In terms of efficiency in our scheme, the computation cost for deriving the surface encryption key is only multiplication operations for each user. Because using symmetrical encryption algorithm, the actual encryption and decryption computational complexity is thus minimized, so the use of a double decryption does not appear as a significant computational burden for each user. Therefore, the efficiency of our scheme is high.

The storage overhead only includes a secret key of constant size for each user in our scheme. Each user can use the secret key to derive an encryption key or more encryption keys by the public tokens, thus avoids a user having to store and manage a huge number of keys. Therefore, the storage overhead of the scheme is very low.

It is very clear that low requirements for computation and storage allow a scheme to be used in a much wider spectrum of devices and applications than costly schemes. So our scheme has a wide range of applications.

V. CONCLUSION

Access control is a very important issue in outsourced data. In this paper, we propose a scheme to achieve secure and efficient access to outsourced data. We propose to construct the key derivation procedure with filter functions so that dynamic access control can be achieved. Through the adoption of key derivation method, the user needs to maintain only a secret key. Analysis of security and performance shows that our scheme is secure and efficient.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China under Grant (61070164); Science and Technology Planning Project of Guangdong Province, China (2010B010600025); Natural Science Foundation of Guangdong Province, China (81510632010 000022).

REFERENCES

[1] J.Birget, X.Zou, G.Noubir, and B.Ramamurthy. Hierarchy-based access control in distributed environments. In Proc. of IEEE InternationalConference on Communications, Finland, USA, June 2002.

[2] G.Miklau, D.Suciu. Controlling access to published data using cryptography. In Proc. of the 29th VLDB conference, Berlin, Germany, Sept. 2003.

[3] Y.Sun, K.Liu. Scalable hierarchical access control in secure group communications. In Proc. of the IEEE Infocom, Hong Kong, China, March 2004.

[4] J.Crampton, K.Martin, i P.Samarat. On key assignment for hierarchical access control. In 19th IEEE CSFW'06, 2006.

[5] Y.Kim, A.Perrig, G.Tsudik. Tree-based group key agreement. Technical Report 2002/009, 2002.

[6] S.De Capitani di Vimercati, S.Foresti, S.Jajodi a, S.Paraboschi, and P.Samarati. Over-encryption: Management of access control evolution on outsourced data. In VLDB, 2007.

[7] A.Lanovenko, H.Guo. Dynamic Group Key Management in Outsourced Databases. Proceedings of the World Congress on Engineering and Computer Science,San Francisco, USA ,October, 24-26,2007,pp.22-28.

[8] S Liu., W.Li, L.Y.Wang. Towards Efficient Over-Encryption in Outsourced Databases Using Secret Sharing. New Technologies, Mobilety and Security, Tangier, Morocco, November,5-7 , 2008, pp.1-5.

[9] J.Weng, M.R.Chen, K.F.Chen,R.H.Deng. Cryptanalysis of a Hierarchical Identity-Based Encryption Scheme. IEICE Transaction. 2010,94-A(4): 854-856.

[10] W.Wang, Z.W.Li, R.Owens, B.Bhargava. Secure and efficient access to outsourced data. Proceedings of the 2009 ACM workshop on Cloud computing security, Chicago,USA, November 9-13,2009,pp.55-56.

[11] J.Dai, Q.Zhou. A PKI-based mechanism for secure and efficient access to outsourced data.. Networking and digital society(ICNDS), St. Maarten, Netherlands Antilles, February, 10-16, 2010, pp. 640-643.

Xiaoming Wang received her Ph.D from the College of Mathematics, Nankai University, China, in 2003. Currently she is a professor in the Computer Science Department, Jinan University. Her research interests include database security, network security, cryptography, etc.

Yanchun Lin is a M.S. Degree candidate in Department of Computer Science, Jinan University, China. Her research interests include database security, network security, etc.