Comments and Corrections

Comments on "DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems"/Security Analysis of Attribute Revocation in Multiauthority Data Access Control for Cloud Storage Systems

Jianan Hong, Kaiping Xue, *Member, IEEE*, and Wei Li

***Abstract*— In the above paper, Yang *et al.* have proposed a multi-authority ciphertext-policy attribute-based encryption-based data access control for cloud storage, in which the authors claimed that the mechanism in dealing with attribute revocation could achieve both forward security and backward security. Unfortunately, our further analysis and investigation show that their work adopts a bidirectional re-encryption method in ciphertext updating, so a security vulnerability appears. Our proposed attack method demonstrates that a revoked user can still decrypt new ciphertexts that are claimed to require the new-version secret keys to decrypt.**

***Index Terms*— Attribute based encryption, multi-authority, attribute revocation, backward security.**

I. Introduction

Ciphertext-Policy Attribute-based Encryption (CP-ABE) [1] is regarded as one of the most attractive cryptographic techniques for data access control in cloud storage system, because of its fine-grained data access control policy and direct control of data for data owners. In CP-ABE, the user can access the content of the ciphertext, only if his/her attributes satisfy the ciphertext's preset access policy.

To apply CP-ABE to data access control for cloud storage, some multi-authority CP-ABE schemes, such as [2] and [3], have been proposed. Specially, in DAC-MACS [2], besides proposing a multi-authority CP-ABE scheme for cloud storage, the authors claimed that the attribute revocation mechanism could satisfy the following two requirements: 1) The revoked user cannot decrypt the newly encrypted ciphertexts that require the revoked attributes to decrypt (*Backward Security*); 2) The newly joined user can also decrypt the previously published ciphertexts if he/she has got adequate attributes (*Forward Security*). However, the authors proved the security without the consideration that the cloud server would leak an important component (*CUK*, Cipheretxt Update Key) to revoked users, or the cloud server obtains some users' leaked *GSK* (Global Secret Key). Similar as the work in [4] and [5] et. al., we cannot ignore the potential collusion between the cloud and revoked users.

In this paper, we demonstrate that, with the component *CUK* a revoked user can transform the newly encrypted ciphertext to a previous version, which can be further decrypted with his/her revoked old-version secret keys. Our work shows that the revocation algorithm of DAC-MACS doesn't meet the backward security requirement.

Manuscript received November 20, 2014; accepted January 18, 2015. Date of publication February 26, 2015; date of current version April 29, 2015. This work was supported by the National Natural Science Foundation of China under Grant 61379129 and Grant 61170231. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jiangtao Li.

The authors are with the Department of Electrical Engineering and Information Science, University of Science and Technology of China, Hefei 230026, China (e-mail: hongjn@mail.ustc.edu.cn; kpxue@ustc.edu.cn; lwz159@mail.ustc.edu.cn).

Digital Object Identifier 10.1109/TIFS.2015.2407327

II. Brief Review of DAC-MACS

DAC-MACS contains five algorithms: *System Initialization*, *Secret Key Generation*, *Encryption*, *Decryption* and *Attribute Revocation*.

A. System Initialization

1) CA Setup: Let $\mathbb{G}$, $\mathbb{G}_T$ be two multiplicative cyclic groups with the order p, and g be a generator of $\mathbb{G}$. The bilinear map e is defined on $\mathbb{G}$ and $H : \{0, 1\}^* \rightarrow \mathbb{G}$ is a hash function. *CA* randomly chooses $a \in \mathbb{Z}_p$ as the master key *MSK*, and computes g^a. Then, *CA* generates a pair of signature and verification key (sk_{CA},vk_{CA}). The system parameters are set as: $SP = (g, g^a, \mathbb{G}, \mathbb{G}_T, H)$.

For each legal *AA*(attribute authority), *CA* assigns an authority identity *aid*. For each user(uid), *CA* randomly chooses u_{uid} and selects z_{uid} as GSK_{uid}, then computes $GPK_{uid} = g^{uid}$. Later, *CA* generates a certification $Cert(uid) = E(uid, u_{uid}, g^{1/z_{uid}})$. *CA* securely sends GPK_{uid}, GSK_{uid} and $Cert(uid)$ to the user.

2) AA Setup: Each AA_k randomly generates its secret key: $SK_k = (\alpha_k, \beta_k, \gamma_k)$. For each attribute x_k in AA_k's attribute subset S_{A_k}, AA_k generates the public attribute key as $PK_{x_k} = \left(g^{\upsilon_{x_k}} H(x_k)\right)^{\gamma_k}$, where υ_{x_k} denotes the random chosen attribute version of x_k. The public authority key of AA_k is computed as $PK_k = \left(e(g, g)^{\alpha_k}, g^{\frac{1}{\beta_k}}, g^{\frac{\gamma_k}{\beta_k}}\right)$.

B. Secret Key Generation

The user U_j sends $Cert(j)$ to AA_k and requests AA_k to issue the secret keys associated to his/her attributes. After successfully verifying user U_j's legality, AA_k recovers u_j and g^{1/z_j} from *Cert(j)*. Let $S_{j,k}$ denote the set of attributes AA_k issues to U_j. Firstly, AA_k chooses a random number $t_{j,k} \in \mathbb{Z}_p$. Then, the secret key $SK_{j,k}$ is generated as:

$$SK_{j,k} = \Bigg(K_{j,k} = g^{\frac{\alpha_k}{z_j}} \cdot g^{a \cdot (u_j + \frac{t_{j,k}}{\beta_k})}, L_{j,k} = g^{\frac{\beta_k}{z_j} \cdot t_{j,k}},$$
$$R_{j,k} = g^{a \cdot t_{j,k}}, \forall x_k \in S_{j,k} : K_{j,x_k} = g^{\frac{\beta_k \gamma_k}{z_j} t_{j,k}}$$
$$\cdot \left(g^{\upsilon_{x_k}} \cdot H(x_k)\right)^{\gamma_k \cdot \beta_k \cdot u_j}\Bigg).$$

C. Encryption

The data owner encrypts the data M using a symmetric encryption algorithm with κ, denoted as $E_\kappa(M)$, where κ is a random chosen content key. Then the data owner encrypts κ using CP-ABE under the access policy defined by himself/herself.

The owner defines and gets an *LSSS* access structure $(\mathcal{M}, \rho)$ over all the selected attributes from the involved *AA*s. $\mathcal{M}$ is an $l \times n$ matrix, where l denotes the number of involved attributes. Let $\rho(i)$ denote the attribute associated to the i-th row of $\mathcal{M}$. In addition,

let I_A denote the set of AAs who manage the involved attributes. The algorithm selects a random encryption exponent $s \in \mathbb{Z}_p$ and a random vector $\vec{v} = (s, y_2, y_3, \ldots, y_n) \in \mathbb{Z}_p^n$, where $y_2, y_3, \ldots, y_n$ are used to share the parameter s. Each $\lambda_i = \vec{v} \cdot \mathcal{M}_i$ is calculated for i from 1 to l, where $\mathcal{M}_i$ denotes the i-th row of the matrix $\mathcal{M}$. Then the owner randomly selects $r_1, r_2, \ldots, r_l \in \mathbb{Z}_p$ and calculates the ciphertext CT as[1]:

$$CT = \Big(E_\kappa(M), C = \kappa(\prod_{k\in I_A} e(g,g)^{\alpha_k})^s, C' = g^s,$$
$$\forall k \in I_A: \ C_k'' = g^{\frac{s}{\beta_k}}, \forall i = 1 \ to \ l: \ C_i = g^{a\cdot\lambda_i}$$
$$\cdot ((g^{v_{\rho(i)}} \cdot H(\rho(i)))^{\gamma_k})^{-r_i},$$
$$D_{1,i} = g^{\frac{r_i}{\beta_k}}, D_{2,i} = g^{-\frac{\gamma_k}{\beta_k}\cdot r_i}\Big)$$

where, in computing C_i, $D_{1,i}$ and $D_{2,i}$, $k = j$ if the attribute $\rho(i)$ is maintained by AA_j.

D. Decryption

Let $I = \{I_{A_k}\}_{k\in I_A}$ denote the whole index set of attributes involved in the procedure of decryption, where I_{A_k} denotes the index set of attributes involved in this procedure managed by AA_k, and $N_A = |I_A|$ denote the number of AAs involved in the procedure. For a user U_j, let $\mathcal{M}_I$ be a sub-matrix of $\mathcal{M}$, where each row of $\mathcal{M}_I$ corresponds to a specific attribute in U_j's attribute set S_{U_j}. The algorithm firstly computes the reconstruction constant $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|I|}\}$, so that $\Omega \cdot \mathcal{M}_I = (1, 0, \ldots, 0)_n$. Then the encryption exponent s can be reconstructed as $s = \sum_{i\in I} \omega_i \cdot \lambda_i$. The algorithm computes a decryption token TK as

$$TK = \prod_{k\in I_A} \frac{e(C', K_{j,k})e(R_{j,k}, C_k'')^{-1}}{\prod_{i\in I_{A_k}} \Big(e(C_i, GPK_{U_j})e(D_{1,i}, K_{j,\rho(i)})e(D_{2,i}, L_{j,k})\Big)^{\omega_i N_A}}$$
$$= \prod_{k\in I_A} e(g,g)^{\frac{\alpha_k}{z_j}s}$$

Furthermore, the algorithm recovers the content key κ as: $\kappa = C/TK^{z_j}$. Finally, with κ, the user U_j can decrypt the ciphertext to get the final data M.

E. Attribute Revocation

For a revoked attribute x_k with the version v_{x_k}, AA_k randomly selects a new version v'_{x_k}, and publishes its updated public key: $PK'_{x_k} = (g^{v'_{x_k}} H(x_k))^{\gamma_k}$. The algorithm further computes two update keys: 1) User's Key Update Key $KUK_{j,x_k} = g^{u_j\cdot\beta_k\cdot\gamma_k\cdot(v'_{x_k} - v_{x_k})}$ for each non-revoked user who has attribute x_k; 2) Ciphertext Update Key $CUK_{x_k} = \beta_k \cdot (v'_{x_k} - v_{x_k})$. Then the revocation algorithm is conducted as following:

- *Secret Key Update.* Upon receiving KUK_{j,x_k}, U_j computes new K_{j,x_k} partly in secret key SK_{j,x_k} for the attribute x_k as: $K'_{j,x_k} = K_{j,x_k} \cdot KUK_{j,x_k}$.
- *Ciphertext Update.* When the cloud server receives CUK_{x_k}, it updates each ciphertext, whose access policy consists of the revoked attribute x_k. For such ciphertext with access policy $(\mathcal{M}, \rho)$, we only need to update C'_i, when $\rho(i) = x_k$. The cloud computes as: $C'_i = C_i \cdot D_{2,i}^{CUK_{x_k}}$.

[1]In Yang et al.'s scheme, there are two loose mistakes in describing how to compute CT. Different C'' should be separately computed, when $\forall k \in I_A$. Here, we redefine this parameter as C''_k. Furthermore, the parameter k in computing C_i, $D_{1,i}$ and $D_{2,i}$ should be specified. We have further made appropriate modifications in the following formulas.

III. Security Vulnerability Analysis of DAC-MACS

To prove the security, the authors in [2] propose a game between a challenger and an adversary, and draw a conclusion that DAC-MACS is secure under the decisional q-parallel BDHE assumption. However, this game makes a connotative restriction that the adversary could not get CUK_{x_k} of any revoked attribute. However, this restriction doesn't suit the security assumption defined in [2], because the cloud service provider is assumed to be semi-trusted, and is likely to collude with any dishonest users or adversaries, and cooperate to calculate any unauthorized elements for their own benefits.

The authors in [2] claimed that the newly uploaded ciphertext could be only decrypted with the new-version secret keys of the attributes after implementing the attribute revocation procedure. However, we demonstrate that although the revoked user cannot update the secret keys of revoked attributes, he/she can collude with the cloud service provider to get CUKs of the updated attributes and further get unauthorized benefits.

The input parameters of our attack model are a new-version ciphertext(CT_0), and the CUKs of the involved updated attributes. The output is a re-encrypted cipheretext (CT_1), which is an old-version one. For an updated attribute $\rho(i)$ in the ciphertext, we assume that its version in CT_0 and CT_1 are $v_{\rho(i),0}$ and $v_{\rho(i),1}$ respectively, and its associated random number is r_i. The two associated components in CT_0 are given as: $C_i = g^{a\lambda_i} \cdot ((g^{v_{\rho(i),0}} H(\rho(i)))^{\gamma_k})^{-r_i}$ and $D_{2,i} = g^{-\frac{\gamma_k}{\beta_k} r_i}$.

For clarity, C'_i is a temporary notation for the component C_i in the old-version ciphertext CT_1. The algorithm computes C'_i as follows:

$$C'_i = C_i \cdot D_{2,i}^{-CUK_{\rho i}}$$
$$= g^{a\lambda_i} \cdot \big((g^{v_{\rho(i),0}} H(\rho(i)))^{\gamma_k}\big)^{-r_i} (g^{-\frac{\gamma_k}{\beta_k} r_i})^{\beta_k(v_{\rho(i),1} - v_{\rho(i),0})}$$
$$= g^{a\lambda_i} \cdot ((g^{v_{\rho(i),1}} H(\rho(i)))^{\gamma_k})^{-r_i}$$

After the re-encryption operation, the component C_i in CT_1 is associated with a revoked attribute and back to the old-version one. Any revoked users can collude with the cloud provider to get the CUKs and implement our proposed attack model to newly uploaded data. Then with their old-version secret keys, the revoked users can implement the algorithm of *Decryption* in Section II to decrypt to get the content of the unauthorized data, as long as the revoked attributes satisfy the access policy.

As discussed above, based on attribute revocation in DAC-MACS, revoked users can still collude with the cloud service provider to decrypt the newly uploaded ciphertext, which is claimed to be decrypted only with the new-version secret keys of the revoked attributes. Therefore, the revocation mechanism in DAC-MACS doesn't satisfy the property of *backward security*, and brings security vulnerability.

IV. Conclusion

In this paper, we analyze the shortcoming of DAC-MACS in dealing with attribute revocation, although the main construction of that work is proved secure. We find that, if a revoked user wants to access the unauthorized content whose access policy can be satisfied by his/her revoked attributes, the only thing to do is to use our proposed attack algorithm to transform the new-version ciphertext to the old-version one if he/she can collude with the cloud service provider to get enough ciphertext update keys. The security vulnerability exists because DAC-MACS wrongly uses a bidirectional re-encryption scheme in the ciphertext updating procedure. This vulnerability allows any party to re-encrypt the ciphertext between old-version and new-version, only if he/she can get the CUKs

between these two versions. In other words, if there's a unidirectional re-encryption scheme that can be embedded into DAC-MACS, the above mentioned security vulnerability will be prevented effectively.

REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.

[2] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.

[3] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.

[4] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[5] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 8, pp. 2201–2210, Aug. 2014.