This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

On the Security of Data Access Control for Multiauthority Cloud Storage Systems

Xianglong Wu, Rui Jiang, and Bharat Bhargava, Fellow, IEEE

Abstract—Data access control has becoming a challenging issue in cloud storage systems. Some techniques have been proposed to achieve the secure data access control in a semitrusted cloud storage system. Recently, K.Yang *et al.* proposed a basic data access control scheme for multiauthority cloud storage system (DAC-MACS) and an extensive data access control scheme (EDAC-MACS). They claimed that the DAC-MACS could achieve efficient decryption and immediate revocation and the EDAC-MACS could also achieve these goals even though nonrevoked users reveal their Key Update Keys to the revoked user. However, through our cryptanalysis, the revocation security of both schemes cannot be guaranteed. In this paper, we first give two attacks on the two schemes. By the first attack, the revoked user can eavesdrop to obtain other users' Key Update Keys to update its Secret Key, and then it can obtain proper Token to decrypt any secret information as a nonrevoked user. In addition, by the second attack, the revoked user can intercept Ciphertext Update Key to retrieve its ability to decrypt any secret information as a nonrevoked user. Secondly, we propose a new extensive DAC-MACS scheme (NEDAC-MACS) to withstand the above two attacks so as to guarantee more secure attribute revocation. Then, formal cryptanalysis of NEDAC-MACS is presented to prove the security goals of the scheme. Finally, the performance comparison among NEDAC-MACS and related schemes is given to demonstrate that the performance of NEDAC-MACS is superior to that of DACC, and relatively same as that of DAC-MACS.

Index Terms—Access control, attribute revocation, revocation security, CP-ABE, multiauthority cloud

1 Introduction

CLOUD computing extends the existing capabilities of Information Technology (IT) since cloud adaptively provides storage and processing services such as SaaS, IaaS, and PaaS that dynamically increase the capacity and add capabilities without investing in new infrastructure or licensing new software [1].

However, the data access control (DAC) issue of cloud computing systems has been escalated by the surge in attacks such as collusion, wiretapping and distort, so that DAC must be designed with sufficient resistance. DAC issues are mainly related to the security policies provided to the users accessing the uploaded data, and the techniques of DAC must specify their own defined security access policies and the further support of policy updates, based on which each valid user can have access to some particular sets of data whereas invalid users are **unauthorized** to access the data. One approach to alleviate attacks is to store the outsourcing data in encrypted form. However, due to the normally semitrusted cloud and its arrangement issues of administration rights, cloud-based access control approaches with traditional encryption are no longer applicable to cloud storage systems [2].

Sahai and Waters [4] laid a theoretical foundation for solving above encryption problem by introducing the new concept of attribute-based encryption (ABE) whose prototype is the identity-based encryption (IBE). The ABE notion has been the promising cryptographic approach on which more intensive research is based. V. Goyal *et al.* first proposed the key-policy attribute based encryption for fine-grained access control (KP-ABE) [5]. In KP-ABE, the data was encrypted by attribute set, and decryption was possible only when the user's policy tree matched the attribute set in the ciphertext. Shortly after KP-ABE, J. Bethencourt introduced the mechanism of ciphertext policy attribute-based encryption (CP-ABE) [6], in which the user received attributes and secret keys from the attribute authority and was able to decrypt ciphertext only if it held sufficient attributes that satisfied the access policy **embedded** in the ciphertext.

Furthermore, the constructed CP-ABE scheme is deemed as one of the most appropriate techniques for data access control in cloud storage systems, since it can be configured to some DAC schemes which do not require the data owners to distribute keys and furnish the data owners with more efficient and attribute-level control on defined access policies offline. **A myriad of data access control techniques based on CP-ABE (e.g. [2], [3], [7]-[19]) are proposed to construct the efficient, secure, fine-grained and attribute-level-revocable access schemes in a semi-trusted cloud storage system. However, based on the Dolev-Yao model [30], security goals such as active attack resistance, data confidentiality, anti-collusion, and attribute-revocation security of most solution designs cannot be all perfectly guaranteed since the capable Dolev-Yao adversaries can overhear, intercept, replay, and synthesis arbitrary information in the open communication channels. For example, in context of attribute revocation in the scenario of K.Yang *et***

- *Xianglong Wu is with the School of Information Science and Engineering, Southeast University, Nanjing, China (e-mail: wuxianglong-018@163.com)*
- *Rui Jiang is with the School of Information Science and Engineering, Southeast University, Nanjing, China (e-mail: R.Jiang@seu.edu.cn), corresponding author.*
- *Bharat Bhargava is with the Department of Computer Science, Purdue University, West Lafayette, IN, USA (e-mail: bbshail@purdue.edu).*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

***al.* proposed DAC-MACS and EDAC-MACS [2], due to the open and non-secure communication channel, the revoked users, as the Dolev-Yao adversaries, can still breach the backward revocation when they eavesdrop to obtain more than two valid users' Key Update Keys to update their own Secret Keys, or when they intercept the Ciphertext Update Key delivered from attribute authority to cloud. In both scenarios, each revoked user can retrieve its ability to decrypt any secret information as a non-revoked user.**

1.1 Our Contributions

In this paper, two attacks are first given on the DAC-MACS's and EDAC-MACS's revocation security which cannot be guaranteed through our cryptanalysis. Subsequently, a new extensive DAC-MACS scheme (NEDAC-MACS) is proposed to withstand above two attacks so as to support more secure attribute revocation. The main contributions of this paper are summarized as follows:

1. In this paper, two attacks are firstly constructed on the vulnerabilities of revocation security in DAC-MACS and EDAC-MACS. By the first attack, the revoked user can eavesdrop to obtain other users' Key Update Keys to update its Secret Keys, and then it can obtain proper Token to decrypt any secret information as a nonrevoked user as before. In addition, by the second attack, the revoked user can intercept the Ciphertext Update Key to retrieve its ability to decrypt any secret information as a nonrevoked user as before.
2. Secondly, we propose a new extensive DAC-MACS scheme, denoted as the NEDAC-MACS, to withstand above two attacks and support more secure attribute revocation. We modify some DAC-MACS's algorithms, and perform the vital ciphertext update communication between cloud server and AAs with some more secure algorithms. Our NEDAC-MACS scheme mainly includes two improvements on the DAC-MACS at *Secret Key Generation* phase and *Attribute Revocation* phase, and it can run correctly according to the correctness proof of NEDAC-MACS.
3. Then, formal cryptanalysis of the NEDAC-MACS is described to prove that the proposed NEDAC-MACS can guarantee collusion resistance, secure attribute revocation, data confidentiality, and provable security against static corruption of authorities based on the random oracle model.
4. Finally, performance analysis of our NEDAC-MACS are conducted by making an efficiency comparison among related CP-ABE schemes to testify that the NEDAC-MACS is security-enhanced without reducing more efficiency. The major overhead of decryption is also securely outsourced to the cloud servers, and the overall overheads of storage, communication and computation of the NEDAC-MACS are superior to that of DACC and relatively same as that of DAC-MACS.

1.2 Organizations

We first introduce related work in section 2. The system model and framework of DAC-MACS and EDAC-MACS are briefly reviewed in section 3. Then, two detailed attacks on the attribute revocation security of the two schemes are elaborated in section 4. Subsequently, a new extensive DAC-MACS scheme with enhanced revocation security is proposed in section 5. Section 6 and 7 present the formal cryptanalysis and performance simulation of our NEDAC-MACS scheme, respectively. Finally, the conclusion is given in Section 8.

2 Related Work

Data Access Control: A plurality of data access control systems (e.g. [2], [3], [7]-[19]) based on the promising CP-ABE technique are proposed to construct the efficient, secure, fine grained and revocable access schemes. S.Ruj *et al.* (2011) proposed a distributed access control scheme in clouds (DACC) [9] that supported attribute revocation. In DACC, one or more key distribution centers (KDCs) distributed keys to data owners and users. Technically, it requires not only forward security but more indispensable backward security in context of the attribute revocation. However, DACC supported attribute revocation with vulnerable forward security [2].

J.Hur *et al.* (2011) proposed an attribute-based DAC scheme [12] with efficient revocation in cloud storage systems, whereas it was designed only for the cloud systems with single trusted authority. In addition, the above two schemes both require data owners to reencrypt the outsourced ciphertext after revocation.

Liu *et al.* (2013) presented a secure multi-owner data sharing scheme called Mona [20]. It is claimed that the scheme can achieve fine-grained access control and secure revocation. However, the scheme will easily suffer from collusion attack by the revoked user and the cloud [21].

Recently, K.Yang *et al.* proposed a data access control scheme for multiauthority cloud storage system (DAC-MACS) [2] and [3] which both supported more efficient decryption and secure attribute revocation without reencryption by the data owners. In reference [2], due to a strong security assumption in DAC-MACS that the non-revoked users will not reveal their key update keys to the revoked user, the authors further removed the assumption and proposed the extensive data access control scheme (EDAC-MACS). In context of secure attribute revocation, DAC-MACS and EDAC-MACS could both achieve forward revocation security irrespective of active attacks. However, the backward revocation security both in DAC-MACS and EDAC-MACS still cannot be guaranteed when the revoked user eavesdrops to obtain more than two users' Key Update Keys to update its Secret Key, or when the revoked user intercepts the Ciphertext Update Key. In both scenarios, the revoked user can retrieve its ability to decrypt any secret information as a nonrevoked user just as before.

Efficiency of Outsourcing Decryption: Green *et al.* [22] (2011) introduced the notion of outsourcing ABE decryption, and presented two concrete ABE schemes with outsourced decryption, which outsourced the main computation of the decryption and only incurred a small overhead of plaintext recovery for the user by using a token-based

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

TABLE Ⅰ NOTATIONS AND DESCRIPTIONS

Notations	Descriptions
G_1, G_2, G_3	Multiplicative cyclic groups of prime order p
H	A hash function $H: \{0,1\}^* \rightarrow Z_q^*$
MSK	The system master key α
SP	The public system parameters
(sk_{CA}, vk_{CA})	The signature and verification key of CA
uid	An unique global identity of user
aid	An unique global identity of attribute authority
U_j	The user whose identity $uid = j$
S_A	The ID set of attribute authorities in the system
S_U	The ID set of users in the system
S_{A_k}	The set of attributes superviced by AA_k
I_A	The set of authorities who supervise the involved attributes in the access policy defined in CT
I_{A_k}	The index set of attributes which are assigned by AA_k and involved in the access policy of CT
κ	The content keys to encrypt data
TK	The decryption token generated by servers to reduce user's computation overhead
t_p	One pairing computation time
t_m	One scalar multiplication time
I_u	The set of attributes U_u holds

decryption method. When outsourcing the decryption of ABE ciphertext, data confidentiality against the curious but honest cloud servers or an adversary can be guaranteed; however, most ABE schemes provide no guarantee on the correctness of the outsourced transformation done by the cloud servers. Cloud service providers are postulated to be semi-trusted and may have profit motives to reduce the computation and return incorrect answers which are unlikely to be detected by valid users. Recently, Lai [23] (2013) modified the original model of Green's ABE schemes [22] to allow for verifiability of the outsourced transformations. However, the storage, computation and communication overheads of the additional redundancy in scheme [23] all scale linearly with the complexity of the transmitted ciphertext and cannot be practical and flexible in more general scenario.

3 BRIEF REVIEW OF DAC-MAC AND EDAC-MAC

3.1 Notations

Some notations used in the paper and their descriptions are briefly shown in Table I.

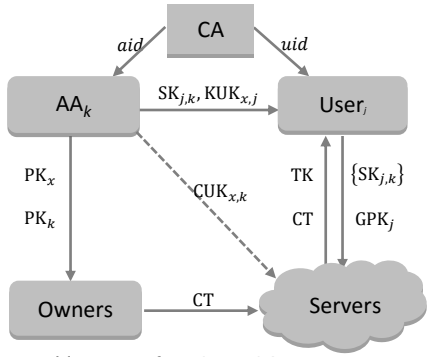


Fig. 1. System Architecture of DAC-MACS

TABLE Ⅱ ENTITIES AND DESCRIPTIONS

Entity	Descriptions of roles and behaviors
CA	A trusted entity to register each user and AA_k, and set up the system.
AA_k	The k-th attribute authority to issue, revoke and update user's attributes and attribute keys.
Server	It stores owners' data, provides DAC services and generates decryption token for users, and conducts CT update for attribute revocation.
User	It submits its attribute keys to the servers for a decryption token, and decrypts the CT.
Owner	It defines the access policies, encrypts content keys κ under the policies and encrypt data by the key κ. It then outsources CT to servers.

3.2 System Model of DAC-MACS

As shown in Fig. 1, a cloud storage system with multiple attribute authorities (DAC-MACS) has five types of entities involved: global certificate authority (CA), users, cloud servers, data owners, and attribute authority (AA). Table Ⅱ presents the roles and behaviors of all involved parties in DAC-MACS.

In DAC-MACS, the global certificate authority (CA) accepts both users' and attribute authorities' registrations to initialize the system by two steps CAsetup and AAsetup, and hence assign a global unique identity uid to each valid user and a global unique aid to each AA.

After registration, each $AA_k \in S_A$ runs Secret Key generation algorithm to compute valid user's secret keys $\{SK\}$ according to the user's role or hierarchy in a defined access policy to some sensitive data.

Then, for each data m, data owners first define an access structure [24], [25] $\mathbb{A} = (M, \rho)$, encrypt the data under this access structure and then outsource the encrypted data CT to the proxy cloud server.

Thereafter, the user $U_j \in S_U$ can upload $\mathbb{A}$-related secret keys $\{SK\}$ and its global public key GPK to cloud for a decryption token TK computed by cloud servers, then the user can decrypt the data m with the TK and its global secret key. The CA, AAs, and cloud servers cannot decrypt the data m without user's global secret key.

For attribute revocation, the corresponding AA, which supervises the revoked attribute, first assigns a version key to each attribute and then generates Ciphertext Update Key for cloud to update CT and Key Update Key for users to update SK. Only those CTs, SKs related to the revoked attribute need to be updated to implicitly contain the latest version key of the revoked attribute. After attribute revocation, all algorithms in system stay unaltered.

3.3 Framework of DAC-MACS

The framework of DAC-MACS mainly consists of five phases: *System initialization*, *Secret Key generation by AAs*, *Data encryption by data owners*, *Data decryption by users with the help of cloud*, and *Attribute revocation*.

3.3.1 System Initialization

The whole system can be set up with following steps:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

1. *CA setup*: The certificate authority initializes the system with the CAsetup algorithm:

$$\mathrm{CASetup}(1^{\lambda}) \longrightarrow (\mathrm{MSK}, \mathrm{SP}, (sk_{\mathrm{CA}}, vk_{\mathrm{CA}})).$$

It takes a security parameter λ as inputs and it outputs the system's master key MSK and the public parameters SP and a pair of signature and verification key $(sk_{\mathrm{CA}}, vk_{\mathrm{CA}})$.

2. *User Registration*: The users send their identity information to CA, then CA conducts UserReg algorithm:

$$\mathrm{UserReg}(\mathrm{SP}, sk_{\mathrm{CA}}, \mathrm{info}_u) \rightarrow \left(uid, \mathrm{GPK}_{uid}, \mathrm{GSK}_{uid}, \mathrm{cert}(uid)\right)$$

to compute and return each user's unique identity uid, global public key $\mathrm{GPK}_{uid} = g^{uid}$, a global secret key $\mathrm{GSK}_{uid} = z_{uid}$ and a user certification $\mathrm{cert}(uid) = Sign_{sk_{\mathrm{CA}}}(uid, u_{uid}, g^{1/z_{uid}})$.

3. *AA Registration*: Similar to the user registration, each AA sends their identity information to CA for its unique identity aid.

4. *AA Setup*: Each AA_{aid}, $aid \in S_{\mathrm{A}}$ initializes itself with the AASetup algorithm:

$$\mathrm{AASetup}(\mathrm{SP}, aid) \rightarrow \left(\mathrm{SK}_{aid}, \mathrm{PK}_{aid}, \{\mathrm{VK}_{x_{aid}}, \mathrm{PK}_{x_{aid}}\}\right).$$

The outputs $\mathrm{SK}_k = (\alpha_k, \beta_k, \gamma_k)$, $\mathrm{PK}_k = (e(g,g)^{\alpha_k}, g^{1/\beta_k}, g^{\gamma_k/\beta_k})$ are the secret and public authority key of AA_k, and $\{\mathrm{VK}_{x_{aid}} = v_{x_k}, \mathrm{PK}_{x_{aid}} = (g^{v_{x_k}}\mathrm{H}(x_k))^{\gamma_k}\}$ are the secret version keys and public key of each attribute x_k supervised by AA_k.

3.3.2 Secret Key Generation by AAs

Each attribute authority AA_k ($k \in S_{\mathrm{A}}$) assigns each valid user U_j ($j \in S_{\mathrm{U}}$) a set of attributes $S_{j,k}$, then performs the SKeyGen algorithm:

$$\mathrm{SKeyGen}(\mathrm{SK}_{aid}, \mathrm{SP}, \{\mathrm{PK}_{x_{aid}}\}, S_{uid,aid}, \mathrm{cert}(uid)) \rightarrow \mathrm{SK}_{uid,aid}$$

to generate the user's secret attribute key $\mathrm{SK}_{j,k}$:

For $\forall j \in S_{\mathrm{U}}$ and $\forall k \in S_{\mathrm{A}}$:

$$\mathrm{SK}_{j,k} = (\mathrm{K}_{j,k}, \mathrm{L}_{j,k}, R_{j,k} \forall x_k \in S_{j,k}: \mathrm{K}_{j,x_k})$$

$$= \begin{bmatrix} \mathrm{K}_{j,k} = g^{\frac{\alpha_k}{z_j}} g^{au_j} g^{\frac{a}{\beta_k} t_{j,k}}, \mathrm{L}_{j,k} = g^{\frac{t_{j,k}\beta_k}{z_j}}, \mathrm{R}_{j,k} = g^{at_{j,k}}, \\ \forall x_k \in S_{j,k}: \quad \mathrm{K}_{j,x_k} = g^{\frac{t_{j,k}\beta_k\gamma_k}{z_j}} \cdot \left(\mathrm{PK}_{x_k}\right)^{\beta_k u_j} \end{bmatrix},$$

where the value $t_{j,k}$ is randomly chosen in Z_p.

3.3.3 Data Encryption by Owners

For each data m, according to the data's logic attribute granularserities, data owners define a monotone access structure $\mathbb{A}$ which can be efficiently realized by a linear secret sharing schemes (LSSS [24]), then an efficient monotone span program (MSP) (M, ρ) can be constructed due to the proved equivalence between LSSS and MSP [24], [25]. Under $\mathbb{A}$, data owners perform the Encrypt algorithm:

$$\mathrm{Encrypt}\left(\mathrm{SP}, \ \{\mathrm{PK}_k\}_{k \in I_{\mathrm{A}}}, \ \{\mathrm{PK}_{x_k}\}_{x_k \in S_{\mathrm{A}_k}}^{k \in I_{\mathrm{A}}}, m, \mathbb{A}\right) \rightarrow \mathrm{CT}$$

to compute CT for the data m:

$$\mathrm{CT} = \left(En_{\kappa}(m), \mathrm{C}, \mathrm{C}', \mathrm{C}'', \forall i = 1 \ to \ l: \mathrm{C}_i, \mathrm{D}_{1,i}, \mathrm{D}_{2,i}\right)$$

$$= \begin{bmatrix} En_{\kappa}(m), \ \mathrm{C} = \kappa \cdot (\prod_{k \in I_{\mathrm{A}}} e(g,g)^{\alpha_k})^s, \ \mathrm{C}' = g^s, \ \mathrm{C}'' = g^{\frac{s}{\beta_k}}, \\ \forall i = 1 \ to \ l: \mathrm{C}_i = g^{a\lambda_i} \cdot \left(\mathrm{PK}_{x_{\rho(i)}}\right)^{-r_i}, \mathrm{D}_{1,i} = g^{\frac{r_i}{\beta_k}}, \mathrm{D}_{2,i} = g^{\frac{-r_i\gamma_k}{\beta_k}} \end{bmatrix}$$

where values $k \in I_{\mathrm{A}}, r_i, s$, and vector $\vec{v} = (s, y_2, \ldots, y_n)$ are randomly chosen, s is the secret value in LSSS, $\lambda_i = (\mathrm{M} \cdot \vec{v})_i$ is a share of secret s and belongs to $\rho(i)$, M is a $l \times n$ matrix in monotone span program, and ρ is a function from $\{1,2,\ldots,l\}$ to $\{x_k \in S_{\mathrm{A}_k}, \ k \in I_{\mathrm{A}}\}$.

3.3.4 Data Decryption by Users with the Help of Cloud Servers

1. *Token Generation by Cloud*

The user U_j ($j \in S_{\mathrm{U}}$) from the user set S_{U} queries for a decryption Token TK and CT by sending its secret keys $\{\mathrm{SK}_{j,k}\}_{k \in I_{\mathrm{A}}}$ and GPK_j. Then TK is computed by TKGen algorithm:

$$\mathrm{TKGen}\left(\mathrm{CT}, \mathrm{GPK}_{uid}, \{\mathrm{SK}_{uid,k}\}_{k \in I_{\mathrm{A}}}\right) \rightarrow \mathrm{TK},$$

and the output is

$$\mathrm{TK} = \prod_{k \in I_{\mathrm{A}}} \frac{e(\mathrm{C}', \mathrm{K}_{j,k}) \cdot e(\mathrm{R}_{j,k}, \mathrm{C}'')^{-1}}{\prod_{i \in I_{\mathrm{A}_k}} \left[e(\mathrm{C}_i, \mathrm{GPK}_j) \cdot e(\mathrm{D}_{1,i}, \mathrm{K}_{j,\rho(i)}) \cdot e(\mathrm{D}_{2,i}, \mathrm{L}_{j,k})\right]^{w_i N_{\mathrm{A}}}}$$

where $N_{\mathrm{A}} = |I_{\mathrm{A}}|$, $I_{\mathrm{A}_k} = \{i: \rho(i) \in S_{\mathrm{A}_k}\}$, $I = \{I_{\mathrm{A}_k}\}_{k \in I_{\mathrm{A}}}$, and $\{w_i\}_{i \in I}$ are the chosen constants which can reconstruct the secret s if $\{\lambda_i\}_{i \in I}$ are valid shares of s.

2. *Data Decryption by Users*

After receiving TK and CT, the user U_j can decrypt the ciphertext with its GSK_j by the Decrypt algorithm:

$$\mathrm{Decrypt}(\mathrm{CT}, \mathrm{TK}, \mathrm{GSK}_{uid}) \rightarrow m.$$

The user U_j first compute the content key:

$$\kappa = \mathrm{C}/\mathrm{TK}^{z_j}, \text{ where } \mathrm{GSK}_j = z_j,$$

then it can decrypt the ciphertext:

$$m = De_{\kappa}\left(En_{\kappa}(m)\right).$$

3.3.5 Attribute Revocation

Suppose $\tilde{x}_k$ of user U_{μ} is revoked from AA_k.

1. *Update Key Generation by AAs*

The $\tilde{x}_k$-corresponding authority AA_k first generates a new attribute version key $\mathrm{VK}'_{\tilde{x}_k}$, and then performs the UKeyGen algorithm:

$$\mathrm{UKeyGen}(\mathrm{SK}_{aid}, \{u_{uid}\}, \mathrm{VK}_{\tilde{x}_{aid}}) \rightarrow \mathrm{KUK}_{uid,\tilde{x}_{aid}}, \mathrm{CUK}_{\tilde{x}_{aid}}, \mathrm{VK}'_{\tilde{x}_{aid}}$$

to calculate the Attribute Update Key $\mathrm{AUK}_{\tilde{x}_k}$, the Key Update Key $\mathrm{KUK}_{j,\tilde{x}_k}$ and the Ciphertext Update Key $\mathrm{CUK}_{\tilde{x}_k}$:

$$\mathrm{AUK}_{\tilde{x}_k} = \gamma_k\left(\mathrm{VK}'_{\tilde{x}_k} - \mathrm{VK}_{\tilde{x}_k}\right),$$

$$\mathrm{KUK}_{j,\tilde{x}_k} = g^{u_j \beta_k \mathrm{AUK}_{\tilde{x}_k}}, \ \mathrm{CUK}_{\tilde{x}_k} = \beta_k \mathrm{AUK}_{\tilde{x}_k}/\gamma_k.$$

Then, AA_k sends $\mathrm{KUK}_{j,\tilde{x}_k}$, $\mathrm{CUK}_{\tilde{x}_k}$ to nonrevoked user U_j ($j \neq \mu$) and cloud server respectively. Meanwhile, the public key of the revoked attribute $\tilde{x}_k$ is changed to the latest version:

$$\mathrm{PK}'_{\tilde{x}_k} = \mathrm{PK}_{\tilde{x}_k} \cdot g^{\mathrm{AUK}_{\tilde{x}_k}}.$$

2. *Secret Key Update by Nonrevoked Users*:

Upon receiving $\mathrm{KUK}_{j,\tilde{x}_k}$, user $\mathrm{U}_j (j \neq \mu)$ can run the SKUpdate algorithm:

$$\mathrm{SKUpdate}\left(\mathrm{SK}_{uid,aid}, \mathrm{KUK}_{uid,\tilde{x}_{aid}}\right) \rightarrow \mathrm{SK}'_{uid,aid}$$

so as to update its $\mathrm{SK}_{j,k}$ to the latest version:

$$\mathrm{SK}'_{j,k} = (\mathrm{K}'_{j,k} = \mathrm{K}_{j,k}, \mathrm{L}'_{j,k} = \mathrm{L}_{j,k}, \mathrm{R}'_{j,k} = \mathrm{R}_{j,k},$$

$$\mathrm{K}'_{j,\tilde{x}_k} = \mathrm{K}_{j,\tilde{x}_k} \cdot \mathrm{KUK}_{j,\tilde{x}_k}, \forall x_k \in S_{j,k}, x_k \neq \tilde{x}_k: \mathrm{K}'_{j,x_k} = \mathrm{K}_{j,x_k})$$

3. *Ciphertext Update by Cloud*

Receiving $\mathrm{CUK}_{\tilde{x}_k}$ from AA_k, cloud servers can run the CTUpdate algorithm:

$$\mathrm{CTUpdate}(\mathrm{CT}, \mathrm{CUK}_{\tilde{x}_{aid}}) \rightarrow \mathrm{CT}'$$

to update its current ciphertext

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

$$\mathrm{CT} = (En_{\kappa}(m), \mathrm{C}, \mathrm{C}', \mathrm{C}'', \forall i = 1\ to\ l\colon \mathrm{C}_i, \mathrm{D}_{1,i}, \mathrm{D}_{2,i})$$

into the latest version:

$$\mathrm{CT}' = \left(En_{\kappa}(m), \mathrm{C}, \mathrm{C}', \mathrm{C}'', \forall i = 1\ to\ l\colon \mathrm{C}_i', \mathrm{D}_{1,i}, \mathrm{D}_{2,i}\right),$$

therein $\forall i = 1\ to\ l$: if $\rho(i) = \tilde{x}_k\colon \mathrm{C}_i' = \mathrm{C}_i \cdot \mathrm{D}_{2,i}^{\mathrm{CUK}_{\tilde{x}_k}} = g^{a\lambda_i} \cdot \left(\mathrm{PK}_{\tilde{x}_k}'\right)^{-r_i}$, else $\mathrm{C}_i' = \mathrm{C}_i$.

For the previous ciphertext CT' which is updated after *Attribute Revocation* phase, it is called *updated previous ciphertext* in this paper. Meanwhile, the newly outsourced data can also be denoted by CT' since they are both corresponding to the current version $\mathrm{PK}_{\tilde{x}_k}'$.

3.4 EDAC-MACS Description

In DAC-MACS [2], K.Yang *et al.* first gave DAC-MACS a strong security assumption that all the nonrevoked users will not send their received Key Update Keys to the revoked user, since they found the revoked user can technically update its secret key to the latest vision via using other user's Key Update Key.

Then they removed this assumption and propose the extensive data access control scheme (EDAC-MACS). Compared to DAC-MACS, three algorithms' outputs are modified: SKeyGen, TKGen and UKeyGen. With these fraction modifications, they claimed that the revoked user has no chance to update its Secret Key even if it can corrupt some AAs and collude with some nonrevoked users. However, this conclusion cannot be guaranteed according to the following section 4.

4 Vulnerability Analysis of DAC-MACS and EDAC-MACS

In this section, attack model and two attacks on the attribute revocation security of DAC-MACS and EDAC-MACS are described in detail. In 4.1, we present the adopted attack model. Then, the first attack is elaborated in section 4.**2** on the EDAC-MACS's vulnerability that the revoked user (attacker) can update its Secret Key with other users' Key Update Keys, and hence decrypt any secret information as a nonrevoked user. Then in section 4.**3**, the second attack on the vulnerability of both DAC-MACS and EDAC-MACS is presented that revoked user can intercept the Ciphertext Update Key to retrieve its ability to decrypt any secret information as a nonrevoked user as before.

4.1 Attack Model

In this paper, we make the cryptanalysis and propose our new extensive scheme based on the Dolev-Yao model [30], in which the adversary can overhear, intercept, insert arbitrary information into, synthesis, and replay any message delivered in the communication channels. Under the Delov-Yao model, the only way to protect the transmitted information from passive or active attacks by eavesdroppers or malicious adversaries is to design the effective security protocols. This means there is no "secure communication channels" assumption between all the involved communication entities. Therefore, it is reasonable that Delov-Yao model can be more appropriate and practical to describe the attackers and demonstrate the communication protocols in reality.

4.2 Attack I

The attack 1 includes two phases: attack preparation and attack implementation. At the preparation phase, the revoked user (attacker) eavesdrops to obtain any two nonrevoked users' Key Update Keys at *Attribute Revocation* phase of EDAC-MACS. Then at the implementation phase, the revoked user can update its own Secret Key SK and then successfully decrypt corresponding CT' as a nonrevoked user.

4.2.1 Attack Preparation Phase

At the *Attribute Revocation* phase of EDAC-MACS, when $\tilde{x}_k$ of user U_{μ} is revoked from AA_k, AA_k sends computed Key Update Keys to each nonrevoked user by implementing UKeyGen algorithm. In principle, the revoked user U_{μ} cannot decrypt any $\tilde{x}_k$-corresponding ciphertext. However, as an attacker in EDAC-MACS, the revoked U_{μ} can eavesdrop to obtain any two nonrevoked users' Key Update Keys: $\mathrm{KUK}_{p,\tilde{x}_k}$ of U_p and $\mathrm{KUK}_{q,\tilde{x}_k}$ of U_q $(p, q \neq \mu)$:

$$\mathrm{KUK}_{p,\tilde{x}_k} = g^{(u_p\beta_k+\gamma_k)\mathrm{AUK}_{\tilde{x}_k}}, \mathrm{KUK}_{q,\tilde{x}_k} = g^{(u_q\beta_k+\gamma_k)\mathrm{AUK}_{\tilde{x}_k}},$$

where $\mathrm{AUK}_{\tilde{x}_k} = \gamma_k\left(v_{\tilde{x}_k}' - v_{\tilde{x}_k}\right)$.

The revoked user (attacker U_{μ}) can also obtain the u_p, u_q of two users from the $\mathrm{cert}(uid)$ with the CA's verification key vk_{CA}.

$$\mathrm{cert}(uid) = Sign_{sk_{\mathrm{CA}}}\left(uid, u_{uid}, g^{1/z_{uid}}\right), uid = p, q.$$

Then U_{μ} can compute its Key Update Key $\mathrm{KUK}_{\mu,\tilde{x}_k}$ and successfully decrypts CT' at the following phase.

4.2.2 Attack Implementation Phase

Having obtained u_p, u_q, $\mathrm{KUK}_{p,\tilde{x}_k}$ and $\mathrm{KUK}_{q,\tilde{x}_k}$, the attacker U_{μ} starts generating its own $\mathrm{KUK}_{\mu,\tilde{x}_k}$ as follows.

Attacker U_{μ} first computes an interim parameter:

$$\Delta = \mathrm{KUK}_{p,\tilde{x}_k}/\mathrm{KUK}_{q,\tilde{x}_k} = g^{(u_p-u_q)\beta_k\gamma_k\left(v_{\tilde{x}_k}'-v_{\tilde{x}_k}\right)}.$$

Afterwards, it can compute its own Key Update Key:

$$\mathrm{KUK}_{\mu,\tilde{x}_k} = \Delta^{\frac{u_{\mu}}{(u_p-u_q)}} \cdot \left[\frac{\mathrm{KUK}_{p,\tilde{x}_k}}{\Delta^{\frac{u_p}{(u_p-u_q)}}}\right].$$

Then, attacker U_{μ} can update its current $\mathrm{SK}_{\mu,k} = (\mathrm{K}_{\mu,k}, \mathrm{L}_{\mu,k}, \mathrm{R}_{\mu,k}, \forall x_k \in S_{\mu,k}\colon \mathrm{K}_{\mu,x_k})$ to the latest version with following algorithm:

$$\mathrm{SKUpdate}(\mathrm{SK}_{\mu,k}, \mathrm{KUK}_{\mu,\tilde{x}_{aid}}) \rightarrow \mathrm{SK}_{\mu,k}'.$$

It outputs:

$$\mathrm{SK}_{\mu,k}' = \begin{bmatrix} \mathrm{K}_{\mu,k}' = \mathrm{K}_{\mu,k}, \mathrm{L}_{\mu,k}' = \mathrm{L}_{\mu,k}, \mathrm{R}_{\mu,k}' = \mathrm{R}_{\mu,k}, \\ \mathrm{K}_{\mu,\tilde{x}_k}' = \mathrm{K}_{\mu,\tilde{x}_k} \cdot \mathrm{KUK}_{\mu,\tilde{x}_k}, \\ \forall x_k \in S_{\mu,k}, x_k \neq \tilde{x}_k\colon \mathrm{K}_{\mu,x_k}' = \mathrm{K}_{\mu,x_k} \end{bmatrix}.$$

Then U_{μ} can upload the latest version $\mathrm{SK}_{\mu,k}'$ to freely query the cloud for proper Token TK and the objective CT':

$$\mathrm{TK} = \prod_{k\epsilon I_{\mathrm{A}}} \frac{e(\mathrm{C}', \mathrm{K}_{\mu,k}') \cdot e(\mathrm{R}_{\mu,k}', \mathrm{C}")^{-1}}{\prod_{i\epsilon I_{\mathrm{A}_k}}\left[e\left(\mathrm{C}_i', \mathrm{GPK}_{\mu}\right) \cdot e\left(\mathrm{D}_{1,i}, \mathrm{K}_{\mu,\rho(i)}'\right) \cdot e\left(\mathrm{D}_{2,i}, \mathrm{L}_{\mu,k}'\right)\right]^{w_i N_{\mathrm{A}}}}$$

$$= \frac{e(g,g)^{sau_{\mu}N_{\mathrm{A}}} \prod_{k\epsilon I_{\mathrm{A}}} e\,(g,g)^{s\frac{\alpha_k}{z_{\mu}}}}{e(g,g)^{au_{\mu}N_{\mathrm{A}}\sum_{i\epsilon I}\lambda_i w_i}} = \prod_{k\epsilon I_{\mathrm{A}}} e\,(g,g)^{s\frac{\alpha_k}{z_{\mu}}}.$$

Afterwards, the attacker U_{μ} can successfully calculate the symmetric encryption key κ:

$$\kappa = \mathrm{C}/\mathrm{TK}^{z_{\mu}}, \text{ where } \mathrm{GSK}_{\mu} = z_{\mu}.$$

Finally U_{μ} can successfully finish the attack for decrypt-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

ing the CT′, whether the CT′ is updated previous one or newly outsourced one, as follow:

$$m = De_{\kappa}\big(En_{\kappa}(m)\big).$$

4.3 Attack Ⅱ

The attack 2 also includes two phases: attack Preparation and attack Implementation. At the preparation phase, the revoked user (attacker U_{μ}) intercepts the previous $\mathrm{CUK}_{\tilde{x}_k}$ at the *Attribute Revocation* phase in DAC-MACS or EDAC-MACS. Then at the implementation phase, the revoked user can use the previous $\mathrm{CUK}_{\tilde{x}_k}$ to decrypt any secret information as a nonrevoked user. Furthermore the revoked user U_{μ} can properly complete all related operations on its own since it can learn the algorithms CTUpdate, TKGen and all the corresponding inputs.

4.3.1 Attack Preparation Phase

At *Attribute Revocation* phase of DAC-MACS or EDAC-MACS, when the AA_k sends Ciphertext Update Key $\mathrm{CUK}_{\tilde{x}_k}$ to cloud server after implementing the UKeyGen algorithm, the revoked user U_{μ}, as an attacker, can eavesdrop to obtain the transmitted $\mathrm{CUK}_{\tilde{x}_k} = \beta_k \mathrm{AUK}_{\tilde{x}_k}/\gamma_k$.

Then it can successfully decrypt CT′ at the following implementation phase.

4.3.2 Attack Implementation Phase

Having obtained $\mathrm{CUK}_{\tilde{x}_k}$, the revoked user (attacker U_{μ}) can freely obtain the objective CT′ anywhere and anytime from cloud servers, whether the CT′ is updated previous one or newly outsourced one:

$$\mathrm{CT}' = \begin{bmatrix} En_{\kappa}(m),\ \mathrm{C},\ \mathrm{C}',\ \mathrm{C}'',\ \forall i = 1\ to\ l\colon\ \mathrm{D}_{1,i},\ \mathrm{D}_{2,i}, \\ if\ \rho(i) = \tilde{x}_k\colon \mathrm{C}_i' = \mathrm{C}_i \cdot \mathrm{D}_{2,i}^{\mathrm{CUK}_{\tilde{x}_k}} = g^{a\lambda_i} \cdot \big(\mathrm{PK}_{\tilde{x}_k}'\big)^{-r_i}, \\ else\colon \mathrm{C}_i' = \mathrm{C}_i. \end{bmatrix}.$$

Then, U_{μ} starts invoking CTUpdate algorithm to reverse the received CT′ back to previous nonrevoked state for U_{μ}:

$$\mathrm{CTUpdate}(\mathrm{CT}', -\mathrm{CUK}_{\tilde{x}_k}) \rightarrow \mathrm{CT}.$$

It outputs

$$\mathrm{CT} = \begin{bmatrix} En_{\kappa}(m), & \mathrm{C},\ \mathrm{C}',\ \mathrm{C}'', & \forall i = 1\ to\ l\colon\ \mathrm{D}_{1,i}, \mathrm{D}_{2,i}, \\ if\ \rho(i) = \tilde{x}_k\colon C_i = g^{a\lambda_i} \cdot \big(\mathrm{PK}_{x_{\rho(i)}}\big)^{-r_i}, & else\ \mathrm{C}_i' = \mathrm{C}_i & \end{bmatrix}.$$

Correctness.

$$\text{If } \rho(i) = \tilde{x}_k\colon \mathrm{C}_i' \cdot \mathrm{D}_{2,i}^{-\mathrm{CUK}_{\tilde{x}_k}} = \mathrm{C}_i \cdot \mathrm{D}_{2,i}^{\mathrm{CUK}_{\tilde{x}_k}} \cdot \mathrm{D}_{2,i}^{-\mathrm{CUK}_{\tilde{x}_k}} = \mathrm{C}_i. \ \square$$

Afterwards, the attacker U_{μ} can successfully calculate TK by itself:

$$\begin{aligned} \mathrm{TK} &= \prod_{k \in I_{\mathrm{A}}} \frac{e\big(\mathrm{C}', \mathrm{K}_{\mu,k}'\big) \cdot e\big(\mathrm{R}_{\mu,k}', \mathrm{C}''\big)^{-1}}{\prod_{i \in I_{A_k}} \Big[e\big(\mathrm{C}_i', \mathrm{GPK}_{\mu}\big) \cdot e\big(\mathrm{D}_{1,i}, \mathrm{K}_{\mu,\rho(i)}'\big) \cdot e\big(\mathrm{D}_{2,i}, \mathrm{L}_{\mu,k}'\big)\Big]^{w_i N_{\mathrm{A}}}} \\ &= \prod_{k \in I_{\mathrm{A}}} e\,(g,g)^{s\frac{\alpha_k}{z_{\mu}}}. \end{aligned}$$

Hence the symmetric encryption key κ can be calculated with the TK:

$$\kappa = \mathrm{C}/\mathrm{TK}^{z_{\mu}},\ \text{where}\ \mathrm{GSK}_{\mu} = z_{\mu}.$$

Finally, U_{μ} can decrypt the CT′ as:

$$m = De_{\kappa}\big(En_{\kappa}(m)\big).$$

5 Our New Extensive DAC-MACS Scheme

In order to withstand above two attacks and to support more secure attribute revocation, a more robust extensive DAC-MACS scheme, denoted as the NEDAC-MACS, is proposed. We modify the vulnerable algorithms of DAC-MACS so that the vital ciphertext update communications between cloud and AAs are performed with security-enhanced algorithms. Our NEDAC-MACS scheme mainly includes two improvements on EDAC-MACS schemes at the *Secret Key Generation* phase and the *Attribute Revocation* phase.

5.1 Preliminaries

5.1.1 Bilinear Pairing

Definition 1. Let $\mathrm{G}_1, \mathrm{G}_2$ and G_3 be three multiplicative cyclic groups of the same prime order p. Let $e\colon \mathrm{G}_1 \times \mathrm{G}_2 \rightarrow \mathrm{G}_3$ denote a bilinear map defined with the following three properties:

- Bilinear: $\forall P \in \mathrm{G}_1,\ \forall Q \in \mathrm{G}_2,\ a, b \in Z_p$, we have $e(aP, bQ) = e(P,Q)^{ab}$.
- Nondegenerate: $\exists P \in \mathrm{G}_1,\ \exists Q \in \mathrm{G}_2$ such that $e(P,Q) \neq I$, where I is the identity element of G_3.
- Computable: There exits an efficient algorithm to compute $e(P,Q)$, for $\forall P \in \mathrm{G}_1,\ \forall Q \in \mathrm{G}_2$.

In this paper, we adopt the symmetric bilinear pairings on elliptic curves groups (let $\mathrm{G}_1 = \mathrm{G}_2$ denoted as G).

5.1.2 Decisional q-Parallel Bilinear Diffie-Hellman Exponent Problem

Definition 2 (q-parallel BDHE [9]). Let g be a generator of group G with prime order p and $a, s \in Z_p$ be randomly chosen. Given a vector $\vec{y}$:

$$\begin{aligned} \vec{y} = (&g, g^s, g^{1/z}, g^{a/z}, \dots, g^{\left(a^q/z\right)}, g^a, g^{a^2}, \dots, g^{a^q}, , g^{a^{q+2}}, \dots, g^{a^{2q}}, \\ &\forall 1 \le j \le q, g^{s \cdot b_j}, g^{a/b_j}, \dots, g^{a^q/b_j}, , g^{a^{q+2}/b_j}, \dots, g^{a^{2q}/b_j}, \\ &\forall 1 \le j, k \le q, k \neq j, g^{a \cdot s \cdot b_k/b_j}, \dots, g^{a^q \cdot s \cdot b_k/b_j}). \end{aligned}$$

It must be hard to distinguish a valid tuple $e(g,g)^{a^{q+1} \cdot s} \in \mathrm{G}_T$ from a random element $R \in \mathrm{G}_T$.

Definition 3. An algorithm $\mathcal{A}$ that outputs $z \in \{0,1\}$ has advantage ε in solving decisional q-parallel BDHE problem in group G if

$$\big|Pr\big[\mathcal{A}\big(\vec{y}, T = e(g,g)^{a^{q+2} \cdot s}\big) = 0\big] - Pr[\mathcal{A}(\vec{y}, T = R) = 0]\big| \ge \varepsilon.$$

5.1.3 Linear Secret Sharing Scheme (LSSS) [24]

A secret sharing scheme over a set of parties P is called linear over Z_p if:

- The shares for each party form a vector over Z_p.
- There exists a share-generating matrix M with l rows and n columns, for all $i = 1, \dots, l$, we define the function $\rho(i)$ labeled with the i-th row of M. **Let $s \in Z_p$ be the secret to be share, and randomly choose $r_2, \dots, r_n \in Z_p$ to contruct the column vector $\vec{v} = (s, r_2, \dots, r_n)$, the party $\rho(i)$ gets the share $\lambda_i = (\mathrm{M}\vec{v})_i$ of the secret s from $\mathrm{M}\vec{v}$.**

5.2 Security Model of NEDAC-MACS

Similar to DAC-MACS, the authorities can only be corrupted statically, whereas the adversary can query adaptively secret keys under condition that queried secret keys cannot be used in decrypting the challenge ciphertext. The security model of the NEDAC-MACS is presented by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

defining a game between a challenger and an adversary as following steps.

***Init*:** After performing the CAsetup algorithm, a set of corrupted attribute authorities S'_{A} are selected by the adversary in the set of all authorities S_{A}. The challenger generates the public keys and secret keys, then sends all public keys and secret keys to the querying adversary in authority set S'_{A}, whereas sends only public keys in S_{A}-S'_{A}.

***Phase 1*:** The adversary selectively refers (uid, S_{uid}) in S_{A} -S'_{A} to the challenger for obtaining corresponding secret keys $\{\mathrm{SK}_{uid,k}\}$ and update keys.

***Challenge*:** The adversary refers two messages m_0 and m_1 of equal length, and additionally gives a challenge access structure (M^*, ρ^*) under following requirement: the target vector $(1,0,\ldots,0)$ is not in the span of $V \cup V_{uid}$, where V denotes the set of all rows of M^* labeled by attributes from S'_{A}, and V_{uid} denotes the set of all rows of M^* labeled by all queried attributes. I.e., the adversary cannot properly decrypt the challenge ciphertext with queried keys and any other keys from S'_{A}. Then, the challenger randomly chooses a bit in $\{0,1\}$, encrypts m_b under (M^*, ρ^*), and finally sends the ciphertext CT^* to adversary.

Phase 2: Similar to Phase 1, more secret keys and update keys can be queried as long as they do not breach the defined constraints condition on (M^*, ρ^*) and the following additional constraint condition: the adversary is not able to query those update keys which can update the queried secret keys to latest version so that the updated keys can decrypt the challenge ciphertext finally.

***Guess*:** When the adversary ends Phase 2, it gives a guess b' of b.

Definition 4. The advantage of an adversary $\mathcal{A}$ in above game is defined as $Adv_{\mathcal{A}} = Pr[b' = b] - 1/2$.

Definition 5. When each one of the collusive user group $\widehat{S_{\mathrm{U}}}$ cannot decrypt the data CT with its own attributes alone, NEDAC-MACS scheme is secure against collusion resistance if no polynomial time adversary can decrypt the CT by the combining attributes of users in $\widehat{S_{\mathrm{U}}}$.

Definition 6. When the decisional q-parallel BDHE assumption holds, NEDAC-MACS scheme is secure against static corruption among authorities if all polynomial time adversaries with a challenge matrix of size $l^* \times n^*$, where $n^* < q$, have at most a negligible advantage in the security game.

5.3 NEDAC-MACS

Due to the open and non-secure communication channel in context of attribute revocation, the revoked user, as a Dolev-Yao attacker, can still breach the backward revocation security both in DAC-MACS and EDAC-MACS when it eavesdrops to obtain more than two users' Key Update Keys to update its Secret Key, or when it intercepts the Ciphertext Update Key.

Therefore, we modify the vulnerable algorithms on the EDAC-MACS schemes at Secret Key Generation phase and Attribute Revocation phase, so that the vital ciphertext update communications between cloud servers and AAs are performed with security-enhanced algorithms in our NEDAC-MACS scheme, which can ensure the real security goals on the open and non-secure communication channels. The two main improvements are inspired by the Green *et al.* [22] introduced notion of outsourcing ABE decryption. Specifically, all valid attribute authorities in NEDAC-MACS apply some components of randomness, such as $h_{j,k}$ on the exponent of bilinear pairing, to each user's secret attribute keys. Thus, when the discrete logarithm assumption holds, the malicious adversary or collusive users are blinded by the randomness, and it is hard for them to launch passive or active attacks such as adaptive chosen message attack or our attack 1 and 2 in section 4.

5.3.1 NEDAC-MACS Architecture

Similar to DAC-MACS, the NEDAC-MACS, new extensive data access control for multiple authorities cloud storage system, also has five types of entities involved: global certificate authority (CA), users, cloud servers, data owners, and attribute authorities (AAs).

The security assumptions of each entity are the same as EDAC-MACS.

The framework of the NEDAC-MACS model also consists of five phases: *System Initialization*, *Secret Key Generation by AAs*, *Data Encryption by Owners*, *Data Decryption by Users* with the help of cloud, and *Attribute Revocation*.

At *System Initialization* phase of NEDAC-MACS, all corresponding algorithms remain the same as in DAC-MACS.

Then at the *Secret Key Generation* phase, compared to DAC-MACS, the output of the Secret Key generation algorithm are modified in NEDAC-MACS by adding a randomly chosen number $h_{uid,aid}$ piece for AA to compute valid user U_{uid}'s secret keys SK. Meanwhile, the component $\mathrm{L}_{uid,aid}$ in SK is correspondingly changed to $\mathrm{L}_{uid,x_{aid}}$ linked with attribute.

Then at the *Data Encryption* and *Decryption* phase, the encryption algorithm by data owner and the decryption algorithm by users is the same as in DAC-MACS.

Finally at the *Attribute Revocation* phase, when attribute $\tilde{x}_{aid}$ of AA_{aid} is revoked from user U_{uid}, the corresponding update key generation algorithm takes as four inputs users' SK_{aid}, $\{u_{uid}\}$, current $\mathrm{VK}_{\tilde{x}_{aid}}$, plus the CT's components $\mathrm{D}_{2,i}$ $(\rho(i) = \tilde{x}_{aid})$ transmitted from cloud servers, and it outputs a new version key for $\tilde{x}_{aid}$, the ciphertext update keys for cloud to update CT, and the key update keys for users to update SK. Only those CTs, SKs related to the revoked attribute $\tilde{x}_{aid}$ need to be updated to implicitly contain the latest version key of $\tilde{x}_{aid}$. The update key generation and secret key update algorithms' outputs are correspondingly changed to contain the randomly chosen number $h_{uid,aid}$ piece, and the ciphertext update algorithm is converted into taking as inputs the ciphertext CT, $\mathrm{CUK}_{\tilde{x}_{aid}}$, $\tilde{x}_{aid}$, PK_{aid}, and a new randomly picked value $\tilde{r}_i$.

After attribute revocation, all the cryptography algorithms in the NEDAC-MACS also stay unaltered except the public key of the involved revoked attribute. Those modified or added fragments of DAC-MACS's algorithms are detailed as the two improvements below.

5.3.2 Improvement at Secret Key Generation Phase

At the *Secret Key Generation by AAs* phase, we add a randomly chosen number $h_{j,k}$ stored by the AA_k for future

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

attribute revocation from the user U_j.

Each AA_k ($k \in S_\mathrm{A}$) assigns each valid user U_j ($j \in S_\mathrm{U}$) a set of attributes $S_{j,k}$ after verifying user's cert(j) by using verification key vk_{CA}, then AA_k **performs the** SKeyGen **algorithm**:

$$\mathrm{SKeyGen}\begin{pmatrix}\mathrm{SK}_{aid}, \{\mathrm{PK}_{x_{aid}}\}, S_{uid,aid},\\ \mathrm{SP}, \mathrm{cert}(uid),\ h_{uid,aid}\end{pmatrix} \to \mathrm{SK}_{uid,aid}$$

to generate user's secret key $\mathrm{SK}_{j,k}$, for $\forall j \in S_U$, $\forall k \in S_A$:

$$\mathrm{SK}_{j,k} = (\mathrm{K}_{j,k}, \mathrm{R}_{j,k}, \forall x_k \in S_{j,k}: \mathrm{K}_{j,x_k}, \mathrm{L}_{j,x_k})$$
$$= \begin{bmatrix} \mathrm{K}_{j,k} = g^{\alpha_k/z_j} \cdot g^{au_j} \cdot g^{a \cdot t_{j,k}/\beta_k}, \quad \mathrm{R}_{j,k} = g^{at_{j,k}},\\ \forall x_k \in S_{j,k}: \quad \mathrm{L}_{j,x_k} = g^{\beta_k t_{j,k}/z_j} \cdot g^{v_{x_k}\beta_k u_j (h_{j,k}-1)},\\ \mathrm{K}_{j,x_k} = g^{\beta_k \gamma_k t_{j,k}/z_j} \cdot \left(g^{v_{x_k}(h_{j,k}-1)} g^{v_{x_k}} \mathrm{H}(x_k)\right)^{\gamma_k \beta_k u_j} \end{bmatrix}$$

where S_U denotes the set of all users, $t_{j,k}$ and $h_{j,k}$ are randomly chosen numbers in Z_p. Note that $h_{j,k}$ should be securely stored by AA_k for future revocation.

5.3.3 Improvement at Attribute Revocation Phase

Suppose the $\tilde{x}_k$ of user U_μ is revoked from AA_k.

1. *Update Key Generation by AAs*

The $\tilde{x}_k$-corresponding authority AA_k first queries the cloud servers for $\mathrm{D}_{2,i}$ ($\rho(i) = \tilde{x}_k$), and then performs the UKeyGen algorithm:

$$\mathrm{UKeyGen}\left(\mathrm{SK}_{aid}, \{u_j\}, \mathrm{VK}_{\tilde{x}_{aid}}, \mathrm{D}_{2,i}\right) \to \mathrm{KUK}_{j,\tilde{x}_{aid}}, \mathrm{CUK}_{\tilde{x}_{aid}}, \mathrm{VK}'_{\tilde{x}_{aid}}, \mathrm{LUK}_{j,\tilde{x}_{aid}}$$

to generate a new attribute version key $\mathrm{VK}'_{\tilde{x}_k} = v'_{\tilde{x}_k}$ for $\tilde{x}_k$, an Attribute Update Key

$$\mathrm{AUK}_{\tilde{x}_k} = \gamma_k\left(\mathrm{VK}'_{\tilde{x}_k} - \mathrm{VK}_{\tilde{x}_k}\right),$$

a Key Update Keys for nonrevoked users $\mathrm{U}_j (j \neq \mu)$ to update their Secret Keys {SK}:

$$\mathrm{KUK}_{j,\tilde{x}_k} = g^{h_{j,k}u_j\beta_k \mathrm{AUK}_{\tilde{x}_k}},\ \mathrm{LUK}_{j,\tilde{x}_k} = g^{\beta_k u_j (h_{j,k}-1)\mathrm{AUK}_{\tilde{x}_k}/\gamma_k},$$

and a Ciphertext Update Key for the cloud servers to update corresponding CT:

$$\mathrm{CUK}_{\tilde{x}_k} = \mathrm{D}_{2,i}^{\beta_k \mathrm{AUK}_{\tilde{x}_k}/\gamma_k}.$$

Then AA_k sends $(\mathrm{KUK}_{j,\tilde{x}_k}, \mathrm{LUK}_{j,\tilde{x}_k})$, $\mathrm{CUK}_{\tilde{x}_k}$ to each norevoked users U_j ($j \neq \mu$) and the cloud servers respectively. Meanwhile, the public key of the revoked attribute $\tilde{x}_k$ has been updated to the latest version:

$$\mathrm{PK}'_{\tilde{x}_k} = \mathrm{PK}_{\tilde{x}_k} \cdot g^{\mathrm{AUK}_{\tilde{x}_k}} = \left[g^{v'_{\tilde{x}_k}} \mathrm{H}(\tilde{x}_k)\right]^{\gamma_k}.$$

2. *Secret Key Update by Nonrevoked Users*

Upon receiving update key pair $(\mathrm{KUK}_{j,\tilde{x}_k}, \mathrm{LUK}_{j,\tilde{x}_k})$, the nonrevoked user U_j ($j \neq \mu$) can run the SKUpdate algorithm:

$$\mathrm{SKUpdate}\left(\mathrm{SK}_{uid,aid}, \mathrm{KUK}_{uid,\tilde{x}_{aid}}, \mathrm{LUK}_{uid,\tilde{x}_{aid}}\right) \to \mathrm{SK}'_{uid,aid}$$

to update its $\mathrm{SK}_{j,k}$ to the latest version:

$$\mathrm{SK}'_{j,k} = \begin{bmatrix} \mathrm{K}'_{j,k} = \mathrm{K}_{j,k}, \quad \mathrm{R}'_{j,k} = \mathrm{R}_{j,k},\\ \mathrm{K}'_{j,\tilde{x}_k} = \mathrm{K}_{j,\tilde{x}_k} \cdot \mathrm{KUK}_{j,\tilde{x}_k}, \quad \mathrm{L}'_{j,\tilde{x}_k} = \mathrm{L}_{j,\tilde{x}_k} \cdot \mathrm{LUK}_{j,\tilde{x}_k},\\ \forall x_k \in S_{j,k}, x_k \neq \tilde{x}_k: \mathrm{K}'_{j,x_k} = \mathrm{K}_{j,x_k},\ \mathrm{L}'_{j,x_k} = \mathrm{L}_{j,x_k} \end{bmatrix}.$$

3. *Ciphertext Update by Cloud*

Receiving $\mathrm{CUK}_{\tilde{x}_k}$, the cloud servers first randomly choose a value $\bar{r}_i$ in Z_p, and then they can perform the CTUpdate algorithm:

$$\mathrm{CTUpdate}(\mathrm{CT}, \mathrm{CUK}_{\tilde{x}_{aid}}, \mathrm{PK}'_{\tilde{x}_{aid}}, \mathrm{PK}_{aid}, \bar{r}_i) \to \mathrm{CT}'$$

to update current $\tilde{x}_k$-corresponding ciphertext CT:

$$\mathrm{CT} = (En_\kappa(m), \mathrm{C}, \mathrm{C}', \mathrm{C}'', \forall i = 1\ to\ l: \mathrm{C}_i, \mathrm{D}_{1,i}, \mathrm{D}_{2,i})$$

into the latest version:

$$\mathrm{CT}' = \left(En_\kappa(m), \mathrm{C}, \mathrm{C}', \mathrm{C}'', \forall i = 1\ to\ l: \mathrm{C}'_i, \mathrm{D}'_{1,i}, \mathrm{D}'_{2,i}\right),$$

therein $\forall i = 1\ to\ l$:

If $\rho(i) = \tilde{x}_k$: $\mathrm{C}'_i = \mathrm{C}_i \cdot \left(\mathrm{PK}'_{x_{\rho(i)}}\right)^{-\bar{r}_i} \cdot \mathrm{CUK}_{\tilde{x}_k}$,

$$\mathrm{D}'_{1,i} = \mathrm{D}_{1,i} \cdot g^{\frac{-\bar{r}_i}{\beta_k}},\ \mathrm{D}'_{2,i} = \mathrm{D}_{2,i} \cdot g^{\frac{-\bar{r}_i \gamma_k}{\beta_k}},$$

Else: $\mathrm{C}'_i = \mathrm{C}_i, \mathrm{D}'_{1,i} = \mathrm{D}_{1,i},\ \mathrm{D}'_{2,i} = \mathrm{D}_{2,i}$.

We note that $\bar{r}_i$ can be discarded by cloud servers after the ciphertext update.

In a NEDAC-MACS scheme, ciphertexts correspond to access structures $\mathbb{A}$, and private keys are associated with a set of attributes W. Decryption is possible when the attribute set W is authorized in the access structure $\mathbb{A}$, i.e., $\mathrm{W} \in \mathbb{A}$.

Definition 7. NEDAC-MACS scheme is correct if for any valid user U_{uid} in the system, any outputs of algorithm $\mathrm{CASetup}(1^\lambda) \to (\mathrm{MSK}, \mathrm{SP}, (sk_{\mathrm{CA}}, vk_{\mathrm{CA}}))$, any U_{uid}'s attribute sets $\mathrm{W} \in \{S_{\mathrm{A}_k}\}^{k \in S_\mathrm{A}}$ authorized in an access structure $\mathbb{A}$, any message $m \in \{0,1\}^*$ to be encrypted into CT under $\mathbb{A}$, and any AA_{aid}'s outputs of $\mathrm{SKeyGen}(\mathrm{SK}_{aid}, \mathrm{SP}, \{\mathrm{PK}_{x_{aid}}\}, \mathrm{W}, \mathrm{cert}(uid), h_{uid,aid}) \to \mathrm{SK}_{uid,aid}$, we have $\mathrm{TKGen}\left(\mathrm{CT}, \mathrm{GPK}_{uid}, \{\mathrm{SK}_{uid,k}\}^{k \in I_\mathrm{A}}\right) \to \mathrm{TK}$ and $\mathrm{Decrypt}(\mathrm{CT}, \mathrm{TK}, \mathrm{GSK}_{uid}) \to m$ with probability 1 over the randomness of all the algorithms.

Theorem 1. *NEDAC-MACS scheme is correct.*

Proof. If a valid user U_j holds sufficient attribute set W which satisfies the access policy $\mathbb{A}$ of the ciphertext CT, it can upload its Secret Keys $\{\mathrm{SK}_{j,k}: k \in I_\mathrm{A}\}$, which are generated by corresponding AA_k with the algorithm SKeyGen, and its global public key GPK_j to cloud server for the decryption token TK computed by the cloud with algorithm TKGen as follow:

$$\mathrm{TK} = \prod_{k \in I_\mathrm{A}} \frac{e(\mathrm{K}_{j,k}, \mathrm{C}') \cdot e(\mathrm{R}_{j,k}, \mathrm{C}")^{-1}}{\prod_{i \in I_{\mathrm{A}_k}} \left[e(\mathrm{C}_i, \mathrm{GPK}_j) \cdot e(\mathrm{D}_{1,i}, \mathrm{K}_{j,\rho(i)}) \cdot e(\mathrm{D}_{2,i}, \mathrm{L}_{j,x_k})\right]^{w_i N_\mathrm{A}}}$$

1. $$\prod_{k \in I_\mathrm{A}} e(\mathrm{K}_{j,k}, \mathrm{C}') \cdot e(\mathrm{R}_{j,k}, \mathrm{C}")^{-1}$$
$$= \prod_{k \in I_\mathrm{A}} e\left(g^{\frac{\alpha_k}{z_j}} g^{au_j} g^{\frac{a}{\beta_k} t_{j,k}}, g^s\right) \cdot e(g^{at_{j,k}}, g^{\frac{s}{\beta_k}})^{-1}$$
$$= e(g,g)^{sau_j N_\mathrm{A}} \prod_{k \in I_\mathrm{A}} e(g,g)^{s\frac{\alpha_k}{z_j}}.$$

2. $$\prod_{k \in I_\mathrm{A}} \prod_{i \in I_{\mathrm{A}_k}} \left[e\left(\mathrm{C}_i, \mathrm{GPK}_j\right) \cdot e\left(\mathrm{D}_{1,i}, \mathrm{K}_{j,\rho(i)}\right) \cdot e\left(\mathrm{D}_{2,i}, \mathrm{L}_{j,x_k}\right)\right]^{w_i N_\mathrm{A}}$$
$$= \prod_{k \in I_\mathrm{A}} \prod_{i \in I_{\mathrm{A}_k}} \begin{bmatrix} e(g^{a\lambda_i - v_{\rho(i)}\gamma_k r_i}, g^{u_j}) \cdot e(\mathrm{H}(\rho(i))^{-\gamma_k r_i}, g^{u_j}) \cdot\\ e\left(g^{\frac{r_i}{\beta_k}}, g^{\frac{\beta_k \gamma_k}{z_j} t_{j,k} + v_{\rho(i)}\gamma_k \beta_k u_j + v_{\rho(i)}(h_{j,k}-1)\gamma_k \beta_k u_j}\right) \cdot\\ e\left(g^{\frac{r_i}{\beta_k}}, \mathrm{H}(\rho(i))^{\gamma_k \beta_k u_j}\right) \cdot e(g^{-\frac{\gamma_k}{\beta_k} r_i}, g^{\frac{\beta_k}{z_j} t_{j,k}}) \cdot\\ e(g^{-\frac{\gamma_k}{\beta_k} r_i}, g^{v_{x_k}\beta_k u_j (h_{j,k}-1)}) \end{bmatrix}^{w_i N_\mathrm{A}}$$
$$= \prod_{k \in I_\mathrm{A}} \prod_{i \in I_{\mathrm{A}_k}} \left[e(g,g)^{a\lambda_i u_j}\right]^{w_i N_\mathrm{A}} = e(g,g)^{au_j N_\mathrm{A} \sum_{i \in I} \lambda_i w_i}$$
$$= e(g,g)^{sau_j N_\mathrm{A}}.$$

$$\mathrm{TK} = \frac{e(g,g)^{sau_j N_\mathrm{A}} \prod_{k \in I_\mathrm{A}} e(g,g)^{s\alpha_k/z_j}}{e(g,g)^{au_j N_\mathrm{A} \sum_{i \in I} \lambda_i w_i}} = \prod_{k \in I_\mathrm{A}} e(g,g)^{s\frac{\alpha_k}{z_j}}.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

Then the user U_j can perform the decryption algorithm Decrypt to obtain plaintext m:

$\kappa = C/TK^{z_j}$, $m = De_{\kappa}(En_{\kappa}(m))$, where $GSK_j = z_j$.

Therefore, U_j can successfully decrypt arbitrary outsourced ciphertext corresponding to its attribute set. □

6 Security Analysis of NEDAC-MACS

In this section, the formal security analysis of NEDAC-MACS is given to prove that our NEDAC-MACS can guarantee collusion resistance, revocation security, data confidentiality and provable security against static corruption of authorities under security model 5.2.

6.1 Collusion Resistance

Theorem 2 proves that our NEDAC-MACS can withstand the collusion attack between the legitimate users. For example, given that a valid user U_1 with attribute set S_1 and another user U_2 with S_2 , according to Theorem 2, it is infeasible for U_1 and U_2 to collude together for decrypting the ciphertext CT encrypted with $W = S_1 \cup S_2$.

Theorem 2. *NEDAC-MACS scheme is secure with users collusion resistance.*

Proof. In NEDAC-MACS, Secret Keys issued by different AA_k to each user is associated with the user's unique identity u_j, and meanwhile two random elements $t_{j,k}$, $h_{j,k}$ chosen by AA_k. Those collusive users are blinded by the random numbers $t_{j,k}$, $h_{j,k}$, and it is hard for them to calculate one user's secret key with other user's secret keys. Therefore, those collusive users cannot decrypt those ciphertext which each individual of them cannot decrypt alone, even though the whole attribute set of them satisfies the access policy. Moreover, those collusive users also cannot selectively replace the components of Secret Key issued by AA_k with the components of secret key issued by AA_l $(k \neq l)$. □

6.2 Revocation Security

In this section, formal cryptanalysis on the security of attribute revocation in NEDAC-MACS is given. Theorem 3 proves that our NEDAC-MACS can ensure the revocation security, which means in context of attribute revocation in NEDAC-MACS, the revoked users, as Dolev-Yao attackers, cannot launch attack 1 in section 4 and update their Secret Keys to breach revocation security and retrieve the ability to decrypt any secret information as non-revoked users as before, even though they intercept any valid users' Key Update Keys.

Theorem 3. *In the NEDAC-MACS, the revoked user has no chance to update its Secret Key even if it can corrupt some AAs (not the AA corresponding to the revoked attribute) and collude with some nonrevoked users.*

Proof. In NEDAC-MACS, when $\tilde{x}_k$ of user U_μ is revoked from AA_k, each key update key $KUK_{j,\tilde{x}_k} = g^{h_{j,k}u_j\beta_k AUK_{\tilde{x}_k}}$, $j \neq \mu$ is associated with both the user's unique identity u_j and an item $h_{j,k}\beta_k$ defined by corresponding AA_k. The item $h_{j,k}\beta_k$ in the secret key prevents users from updating their secret keys with the other users' update keys, since it is only known by the noncorrupted AA_k and kept different and secret to all the users. □

We describe the formal definitions of the backward and forward revocation security as following definition 8 and 9 respectively, which are the basis of proofs in theorem 4 and 5.

Definition 8. NEDAC-MACS scheme supports backward security in context of attribute revocation if the $\tilde{x}_k$-revoked user has no chance to passively retrieve its ability to decrypt any $\tilde{x}_k$-corresponding ciphertext CT as a nonrevoked user, whether the CT is updated previous ciphertext or the newly outsourced ciphertext.

Definition 9. NEDAC-MACS scheme supports forward security in context of attribute revocation if the newly recruited user $\mathbf{U_n}$ who has been assigned the attribute $\tilde{x}_k$ **(soppose $\tilde{x}_k$ is revoked from other user U_μ, $\mu \neq n$)**, is able to decrypt any authorized $\tilde{x}_k$-corresponding ciphertext CT, whether the CT is updated previous ciphertext or newly outsourced ciphertext.

Theorem 4 gives the proof that our NEDAC-MACS can ensure the backward revocation security, which means in context of attribute revocation in NEDAC-MACS, the revoked users cannot launch attack 1 and 2 in section 4 and breach the backward revocation security even though they eavesdrop to intercept any Ciphertext Update Keys delivered from AAs to cloud servers on open and non-secure communication channel. For example, suppose that the AA_k-mornitoring attribute $\tilde{x}_k$ is revoked from user Alice U_μ, the NEDAC-MACS is able to guarrentee that Alice cannot decrypt any $\tilde{x}_k$-related ciphertext CT whether or not the CT is authorized to Alice before the $\tilde{x}_k$ revocation.

Theorem 4. *NEDAC-MACS characterizes backward security in context of attribute revocation.*

Proof. When $\tilde{x}_k$ of user U_μ is revoked from AA_k:

1. For the previous ciphertext CT′ which is updated after the *Attribute Revocation* phase:

   $CT' = (En_{\kappa}(m), C, C', C'', \forall i = 1\ to\ l: C'_i, D'_{1,i}, D'_{2,i})$,

   if $\rho(i) = \tilde{x}_k$:

   $C'_i = C_i (PK'_{\tilde{x}_k})^{-\bar{r}_i} CUK_{\tilde{x}_k}$, $D'_{1,i} = g^{\frac{-(r_i+\bar{r}_i)}{\beta_k}}$, $D'_{2,i} = g^{\frac{-(r_i+\bar{r}_i)}{\beta_k}}$.

   We note that the transmitted $CUK_{\tilde{x}_k} = D_{2,i}^{\beta_k AUK_{\tilde{x}_k}/\gamma_k}$, $\rho(i) = \tilde{x}_k$, and assume that the revoked user has not stored the previous CT. Then it is hard for the revoked users to calculate the exponent $\beta_k AUK_{\tilde{x}_k}/\gamma_k$. Meanwhile, due to those revoked users' blindness by the random number $\bar{r}_i$ chosen by cloud servers, the component $[PK'_{\tilde{x}_k}]^{-\bar{r}_i}$ cannot be canceled out by the revoked user itself.

   Therefore, even though the revoked user can obtain all involved communication information like $D_{2,i}$, $CUK_{\tilde{x}_k}$ in NEDAC-MACS, it still cannot stretch the updated previous CT′ back to the previous version CT the revoked user can properly decrypt.

2. For the newly outsourced ciphertext CT′:

   $CT' = (En_{\kappa}(m), C, C', C'',\ \forall i = 1\ to\ l: C'_i, D_{1,i}, D_{2,i})$,

   $\forall i = 1\ to\ l$:

   If $\rho(i) = \tilde{x}_k$: $C_i = g^{a\lambda_i} \cdot (PK'_{x_{\rho(i)}})^{-r_i}$, $D_{1,i} = g^{r_i/\beta_k}$, $D_{2,i} = g^{-r_i\gamma_k/\beta_k}$, else: $C_i = g^{a\lambda_i} \cdot (PK_{x_{\rho(i)}})^{-r_i}$, $D_{1,i} = g^{r_i/\beta_k}$, $D_{2,i} = g^{-r_i\gamma_k/\beta_k}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

The revoked user cannot construct $(\mathrm{D}_{2,i})^{\beta_k \mathrm{AUK}_{\tilde{x}_k}/\gamma_k}$, since only the uncorrupted attribute authority AA_k who supervises $\tilde{x}_k$ can calculate exponent $\beta_k \mathrm{AUK}_{\tilde{x}_k}/\gamma_k$. Therefore, the revoked user cannot transform the $\mathrm{C}_i = g^{a\lambda_i}(\mathrm{PK}'_{\tilde{x}_k})^{-r_i}$ into $\mathrm{C}_i = g^{a\lambda_i}(\mathrm{PK}_{\tilde{x}_k})^{-r_i}$.

Overall, the revoked user cannot reverse any previous published ciphertext CT' and the newly outsourced ciphertext CT' back to nonrevoked state when U_μ can properly decrypt the ciphertext. □

Theorem 5 proves that our NEDAC-MACS can ensure the forward revocation security, which means when the attribute revocation period ended in NEDAC-MACS, each newly recruited user U_n who has been assigned the attribute $\tilde{x}_k$ (soppose $\tilde{x}_k$ is revoked from user U_μ, $\mu \neq n$), is able to decrypt any authorized $\tilde{x}_k$-corresponding ciphertext CT. The proof of theorem 5 can be be derived on the basis of the Lemma 1 which describes the correctness of our modification at the "Attribute Revocation" phase.

Lemma 1. *In NEDAC-MACS, the attribute revocation phase is correct, and still retain the proper running of whole NEDAC-MACS.*

Proof. At the step *Secret Key Update by Nonrevoked Users* of the attribute revocation in NEDAC-MACS, the secret attribute keys of the nonrevoked user U_j who was assigned the revoked attribute $\tilde{x}_k$, are updated to

$$\mathrm{SK}'_{j,k} = (\mathrm{K}_{j,k}, \mathrm{R}_{j,k}, \forall x_k \in S_{j,k}: \mathrm{K}'_{j,x_k}, \mathrm{L}'_{j,x_k}),$$

$$\text{if } x_k = \tilde{x}_k: \mathrm{K}'_{j,\tilde{x}_k} = g^{\beta_k \gamma_k t_{j,k}/z_j} \cdot \left(g^{v'_{\tilde{x}_k}(h_{j,k}-1)} g^{v'_{\tilde{x}_k}} \mathrm{H}(\tilde{x}_k)\right)^{\gamma_k \beta_k u_j},$$

$$\mathrm{L}'_{j,\tilde{x}_k} = g^{\beta_k t_{j,k}/z_j} \cdot g^{v'_{\tilde{x}_k} \beta_k u_j (h_{j,k}-1)}$$

Then, at the step *Ciphertext Update by Cloud*, the $\tilde{x}_k$-corresponding CT is updated to

$$\mathrm{CT}' = \left(En_\kappa(m), \mathrm{C}, \mathrm{C}', \mathrm{C}'', \forall i = 1 \; to \; l: \mathrm{C}'_i, \mathrm{D}'_{1,i}, \mathrm{D}'_{2,i}\right),$$

If $\rho(i) = \tilde{x}_k$, we have:

$$\mathrm{C}'_i = \mathrm{C}_i \cdot \left(\mathrm{PK}'_{x_{\rho(i)}}\right)^{-\bar{r}_i} \cdot \mathrm{CUK}_{\tilde{x}_k} = g^{a\lambda_i} \cdot \left(\mathrm{PK}'_{x_{\rho(i)}}\right)^{-(r_i+\bar{r}_i)},$$

$$\mathrm{D}'_{1,i} = \mathrm{D}_{1,i} \cdot g^{\frac{-\bar{r}_i}{\beta_k}} = g^{\frac{-(r_i+\bar{r}_i)}{\beta_k}},$$

$$\mathrm{D}'_{2,i} = \mathrm{D}_{2,i} \cdot g^{\frac{-\bar{r}_i \gamma_k}{\beta_k}} = g^{\frac{-(r_i+\bar{r}_i)\gamma_k}{\beta_k}}.$$

All above operations are equivalent to assigning a new random number $r'_i = r_i + \bar{r}_i$ in Z_p to the ciphertext, since $\bar{r}_i$ is randomly chosen in Z_p.

Then, if nonrevoked user has the attribute subset authorized in the above CT', the result of token TK is

$$\mathrm{TK} = \frac{e(g,g)^{sau_j N_\mathrm{A}} \prod_{k \in I_\mathrm{A}} e(g,g)^{s\alpha_k/z_j}}{e(g,g)^{au_j N_\mathrm{A} \sum_{i \in I} \lambda_i w_i}} = \prod_{k \in I_\mathrm{A}} e(g,g)^{s\frac{\alpha_k}{z_j}}.$$

Then the user U_j can obtain the plaintext m:

$$\kappa = \mathrm{C}/\mathrm{TK}^{z_j}, m = De_\kappa\left(En_\kappa(m)\right), \text{ where } \mathrm{GSK}_j = z_j.$$

Therefore, these update operations of revocation still maintain the formal consistency of all parameters and algorithms in NEDAC-MACS. □

Theorem 5. *NEDAC-MACS characterizes forward security in context of attribute revocation.*

Proof. The proof of NEDAC-MACS's forward security is similar to Lemma 1, since, after the *Attribute Revocation* phase, the newly joined user's secret keys and any ciphertexts on cloud servers are all corresponding to the latest version public key of the revoked attribute, just as nonrevoked U_j with revoked $\tilde{x}_k$ does in lemma 1. □

6.3 Data Confidentiality

In NEDAC-MACS, even though the cloud servers learn user's secret keys SK and perform the operation of outsourced decryption computation, the cloud servers cannot properly decrypt any ciphertext uploaded by data owners since the full decryption algorithm involves user's global secret key GSK_{uid}. Furthermore, at the ciphertext update step of *Attribute Revocation* phase, cloud servers update any corresponding ciphertext CT without the ability to decrypt them. Therefore, data confidentiality against the curious but honest cloud servers is guaranteed.

Invalid users who hold insufficient attributes to satisfy access policy, cannot receive proper Token TK from cloud servers for decryption. Due to the users' blindness of the random numbers $t_{j,k}, h_{j,k}$ according to theorem 2 and 3, the invalid user cannot fabricate and upload proper set of Secret Keys for decrypting objective ciphertext. Therefore, data confidentiality against invalid users is guaranteed.

6.4 Provable Security against Static Corruption of Authorities

Under the security model defined in 5.2, the NEDAC-MACS can enjoy the same provable security against static corruption of authorities as DAC-MACS, which is reduced to the hardness of the decisional q-parallel BDHE assumption [28], [29], [30].

Theorem 6. *When the decisional q-parallel BDHE assumption holds, no polynomial time adversary can selectively break the NEDAC-MACS with a challenge matrix of size $l^* \times n^*$, where $n^* < q$.*

Proof. We adopt proof by contradiction like DAC-MACS. Suppose there is an adversary algorithm $\mathcal{A}$ chooses a challenge matrix M^* with at most $q-1$ columns and can selectively break the NEDAC-MACS with nonnegligible advantage $Adv_\mathcal{A}$ in the selective security game. Then, based on random oracle model, we can construct a simulator algorithm $\mathcal{B}$ that plays the decisional q-parallel BDHE with a nonnegligible advantage as follows.

***Init*:** $\mathcal{B}$ takes as inputs $\vec{y}$ and T of the decisional q-parallel BDHE problem. The adversary sends the challenge access structure (M^*, ρ^*) to the $\mathcal{B}$, where M^* has $n^* < q$ columns.

***Setup*:** The simulator runs the initialization algorithms CASetup and AASetup. The adversary specifies the corrupted authority set $S'_\mathrm{A} \subset S_\mathrm{A}$, and reveals S'_A to the simulator. For each $\mathrm{AA}_k \in S_\mathrm{A} - S'_\mathrm{A}$, the simulator randomly assigns the corresponding α'_k, β_k, γ_k to each $\mathrm{AA}_k \in S_\mathrm{A} - S'_\mathrm{A}$ by letting $\alpha_k = \alpha'_k + a^{q+1}$ and $e(g,g)^{\alpha_k} = e\left(g^a, g^{a^q}\right) \cdot e(g,g)^{\alpha'_k}$.

Let $X = \{i \,|\rho^*(i) = x\}$. The random oracle H is defined by simulator as

$$\mathrm{H}(x) = g^{d_x} \prod_{i \in X} g^{\frac{a^2 M^*_{i,1}}{b_i}} \cdot g^{\frac{a^3 M^*_{i,2}}{b_i}} \cdots g^{\frac{a^{n^*+1} \cdot M^*_{i,n^*}}{b_i}}.$$

We note that the outputs of the random oracle are randomly distributed due to a randomly chosen value

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

g^{d_x} and also note $\mathrm{H}(x) = g^{d_x}$ for $X = \emptyset$

For each $\mathrm{AA}_k \in S_\mathrm{A} - S'_\mathrm{A}$, the simulator randomly selects a version number $v_{x_k} \in Z_p$ then simulates the public key PK_k and the public attribute keys PK_{x_k} as

$$\mathrm{PK}_k = \left(e(g,g)^{\alpha'_k}, g^{\frac{1}{\beta_k}}, g^{\frac{\gamma_k}{\beta_k}}\right),$$

$$\mathrm{PK}_{x_k} = \left(g^{v_{x_k}+d_{x_k}} \prod_{i\in X} g^{\frac{a^2 M^*_{i,1}}{b_i}} \cdot g^{\frac{a^3 M^*_{i,2}}{b_i}} \cdots g^{\frac{a^{n+1}\cdot M^*_{i,n}}{b_i}}\right)^{\gamma_k}.$$

After assigning a user identity uid to the adversary $\mathcal{A}$, the simulator $\mathcal{B}$ randomly selects u'_{uid}, $z_{uid} \in Z_p$ then lets

$$\mathrm{GSK}_{uid} = z_{uid}, u_{uid} = u'_{uid} - a^q / z_{uid},$$

$$\mathrm{GPK}_{uid} = g^{u'_{uid}} \cdot \left(g^{a^q}\right)^{-1/z_{uid}}.$$

The simulator $\mathcal{B}$ then sends the $(\mathrm{GPK}_{uid}, \mathrm{GSK}_{uid})$ to the adversary $\mathcal{A}$.

***Phase 1*:** The adversary $\mathcal{A}$ refers (uid, S_k) to the simulator for obtaining secret keys and update keys. Thereinto S_k denotes attributes set from $\mathrm{AA}_k \in S_\mathrm{A} - S'_\mathrm{A}$ and S_k does not satisfy M^* in combination with any keys of $\mathrm{AA}_k \in S'_\mathrm{A}$.

Since S_k does not satisfy M^*, a vector $\vec{\omega} = (\omega_1, \omega_2, \ldots, \omega_{n^*}) \in Z_p^{n^*}$ can be found by the simulator $\mathcal{B}$ where $\omega_1 = -1$, and for each $i, \rho^*(i) \in S_k$: $\vec{\omega} \cdot M^*_i = 0$.

The simulator $\mathcal{B}$ then randomly selects a number $r \in Z_p$ and sets t as

$$t_{uid,k} = r + \omega_1 a^{q-1} + \omega_2 a^{q-2} + \cdots + \omega_{n^*} a^{q-n^*}.$$

Then component $\mathrm{R}_{uid,k}$, $\mathrm{K}_{uid,k}$ can be calculated as

$$\mathrm{R}_{uid,k} = g^{ar} \cdot \prod_{i=1,2,\ldots n^*} \left(g^{a^{q+1-i}}\right)^{\omega_i},$$

$$\mathrm{K}_{uid,k} = g^{\frac{\alpha'_k}{z_{uid}}} \cdot g^{a u'_{uid}} \cdot g^{\frac{ar}{\beta_k}} \cdot \prod_{i=1,2,\ldots n^*} \left(g^{a^{q+1-i}}\right)^{\frac{\omega_i}{\beta_k}}.$$

In the NEDAC-MACS, the component K_{uid,x_k} and L_{uid,x_k} in the secret key are modified by adding some fractions. For those $x_k \in S_k$ used in the access structure ($\exists i$, such that $\rho^*(i) = x_k$), L_{uid,x_k} and $\mathrm{K}_{x_k,uid}$ can be constructed by the simulator as follows.

$\forall x_k \in S_k$:

$$\mathrm{L}_{uid,x_k} = g^{v_{x_k}\beta_k u'_{uid}(h_{j,k}-1)} \cdot \left(g^{\frac{\beta_k}{z_{uid}}}\right)^r \cdot \prod_{i=1,2,\ldots n^*} \left(g^{a^{q-i}}\right)^{\omega_i \frac{\beta_k}{z_{uid}}},$$

$$\mathrm{K}_{uid,x_k} = \left(\mathrm{L}_{uid,k}\right)^{\gamma_k} \cdot \left((g^{v_{x_k}\gamma_k(h_{j,k}-1)} \cdot \mathrm{PK}_{x_k}\right)^{\beta_k u'_{uid}} \cdot \left(g^{a^q}\right)^{\frac{-\beta_k\gamma_k\left(v_{x_k}+d_{x_k}\right)}{z_{uid}}} \cdot \prod_{i\in X} \prod_{j=1,2,\ldots n^*} \left(g^{\frac{a^{q+1+j}}{b_i}}\right)^{-\beta_k\gamma_k M^*_{i,j}}.$$

For those attributes $x \in S_{aid}$ not used in the access structure, L_{uid,x_k} and $\mathrm{K}_{x_k,uid}$ can be constructed as

$$\mathrm{K}_{uid,x_k} = \left(\mathrm{L}_{uid,k}\right)^{\gamma_k} \cdot (\mathrm{GPK}_{uid})^{\beta_k\gamma_k(v_{x_k}+d_{x_k})} \cdot g^{\gamma_k{}^2(v_{x_k}+d_{x_k})}.$$

The adversary can submit some pairs $\{(uid, x_k)\}$ to query update keys. When uid is a nonrevoked user and x_k is assigned a new version key v'_{x_k}, the simulator then responds corresponding keys $\mathrm{KUK}_{uid,x_k}, \mathrm{LUK}_{uid,x_k}$ to adversary:

$$\mathrm{KUK}_{uid,x_k} = g^{u_j\beta_k\gamma_k\left(v'_{x_k}-v_{x_k}\right)}, \mathrm{LUK}_{uid,x_k} = g^{\beta_k u_{uid}\mathrm{AUK}_{x_k}/\gamma_k}.$$

Otherwise, it sends "$\perp$" back.

***Challenge*:** After receiving two equal length messages m_0, m_1 and a challenging access structure from the adversary, simulator $\mathcal{B}$ randomly chooses a bit b in $\{0,1\}$. It first generates

$$\mathrm{C} = m_b T \prod_{k\in I_A} e(g^s, g^{\alpha'_k}), \mathrm{C}' = g^s, \mathrm{C}'' = g^{s/\beta_k}.$$

Randomly choosing $y'_2, \ldots, y'_{n^*} \in Z_p$, the simulator shares secret s by a vector $\vec{v} = (s, sa + y'_2, s\cdot a^2 + y'_3, \ldots, s\cdot a^{n^*-1} + y'_{n^*}) \in Z_p^{n^*}$, then $\mathcal{B}$ can simulate each share λ_i, $i = 1,2,\ldots n^*$ of the secret s as

$$\lambda_i = s\cdot \mathrm{M}_{i,1} + \textstyle\sum_{\mathrm{j}=2,\ldots n^*} (sa^{j-1} + y'_j)\mathrm{M}^*_{i,j}.$$

For each $i = 1,2,\ldots n^*$, let $R_i = \{t \neq i \mid \rho^*(i) = \rho^*(t)\}$. $\mathcal{B}$ randomly chooses $r'_1, \ldots, r'_l$, and simulates the C_i as

$$\mathrm{C}_i = \left(g^{v_{\rho^*(i)}}\mathrm{H}(\rho^*(i))\right)^{\gamma_k r'_l} \cdot \left(\prod_{j=1,2,\ldots n^*} g^{a\mathrm{M}_{i,j}y_j}\right) \cdot (g^{sb_i})^{-\gamma_k\left(v_{\rho^*(i)}+d_{\rho^*(i)}\right)} \cdot \prod_{k\in R_i} \prod_{j=1,2,\ldots n^*} \left(g^{\frac{aj_{sb_i}}{b_k}}\right)^{\gamma_k M^*_{k,j}}.$$

The rest components of the challenge ciphertext CT^* can be simulated as

$$\mathrm{D}_{1,i} = \left(g^{r'_i} g^{sb_i}\right)^{\frac{1}{\beta_k}}, \mathrm{D}_{2,i} = \left(g^{r'_i} g^{sb_i}\right)^{\frac{-\gamma_k}{\beta_k}}.$$

***Phase 2*:** Same as Phase 1.

***Guess*:** The adversary $\mathcal{A}$ finally ends Phase 2 and gives a guess b' of b. If $b' = b$, and the simulator $\mathcal{B}$ outputs 0 to predicate that $T = e(g,g)^{a^{q+1}\cdot s} \in G_T$; otherwise, it outputs 1 to indicate that it believes T is a random element in G_T.

When T results in a tuple, the simulator $\mathcal{B}$ gives a perfect simulation and we have that

$$Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{a^{q+1}\cdot s}\right) = 0\right] = 1/2 + Adv_{\mathcal{A}}.$$

When T results in a random group element in G_T, the message m_b is completely hidden from the adversary $\mathcal{A}$ and $Pr\left[\mathcal{B}\left(\vec{y}, T = e(g,g)^{a^{q+1}\cdot s}\right) = 0\right] = 1/2$.

Therefore, the simulator $\mathcal{B}$ can play the decisional q-parallel BDHE game with nonnegligible advantage. □

TABLE Ⅲ

SECURITY COMPARISON OF CP-ABE SCHEMES

Scheme	Co Res	Revocation		Confidentiality		Pr Sec
		B	F	Ag Cloud	Ag User	
DACC	√	√	×	√	√	√
DAC-MACS	×	×	√	√	×	√
NEDAC-MACS	√	√	√	√	√	√

Co Res = Collusion Resistance, B = Backward, F = Forward, Ag = Against, Pr Sec = Provable Security.

6.5 Security Comparison

Table Ⅲ details the comprehensive security comparison among schemes of S.Ruj *et al.*'s DACC [9], K.Yang *et al.*'s DAC-MACS [2] and our NEDAC-MACS in terms of collusion resistance, revocation security, data confidentiality and provable security against static corruption of authorities. Therein, "√" represents the scheme's capability to achieve the corresponding index, whereas "×" means the opposite.

7 PERFORMANCE ANALYSIS

To validate the efficiency of our NEDAC-MACS, performance comparisons are carried out in terms of storage overhead, computation overhead and communication overhead among CP-ABE schemes of DACC [9], DAC-MACS [2] and our NEDAC-MACS.

7.1 Storage Overhead

Table Ⅳ details the storage comparison among the three

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

TABLE IV
STORAGE OVERHEAD COMPARISON OF CP-ABE SCHEMES

Scheme	Authority (AA_k/KDC_k)	Data Owners	User	Cloud
DACC	$2n_{a,k}\|p\|$	$(n_c + 2\sum_{k=1}^{N_A} n_{a,k})\|p\|$	$(n_{c,x} + \sum_{k=1}^{N_A} n_{a,k,uid})\|p\|$	$(3t_c + 1)\|p\|$
DAC-MACS	$(n_{a,k} + 3)\|p\|$	$(3N_A + 1 + \sum_{k=1}^{N_A} n_{a,k})\|p\|$	$(3N_A + 1 + \sum_{k=1}^{N_A} n_{a,k,uid})\|p\|$	$(3t_c + 3)\|p\|$
NEDAC-MACS	$(n_{a,k} + 3 + n_u)\|p\|$	$(3N_A + 1 + \sum_{k=1}^{N_A} n_{a,k})\|p\|$	$(2N_A + 1 + 2\sum_{k=1}^{N_A} n_{a,k,uid})\|p\|$	$(3t_c + 3)\|p\|$

schemes, where $|p|$ is the size of element in the groups G, G_T, Z_p with prime order p, t_c denotes the total number of attributes associated with a ciphertext, n_c denotes the total number of ciphertext on cloud, t_u denotes the total-number of attributes of a user, x is the revoked attribute, $n_{non,x}$ denotes the total number of nonrevoked users who have the revoked x, $n_{c,x}$ is the number of ciphertext associated with the revoked attribute x, $n_{a,k,uid}$ is the number of attributes assigned from AA_k to user U_{uid}, $n_{a,k}$ is the number of attributes managed by AA_k, N_A is the number of AA involved in system.

Table IV shows that the overall storage overhead of NEDAC-MACS is relatively same as that of DAC-MACS and has advantage over DACC when n_c the number of ciphertext or $n_{c,x}$ the number of ciphertext associated with the revoked x is large in the system.

It is illustrated in Table IV that, on the authority side, DAC-MACS and NEDAC-MACS incur less storage overhead than DACC since both schemes requires each attribute authority to store the version key of each held attribute and the secret authority key, whereas DACC needs to store the secret keys for all attributes. Moreover, the components need be stored in NEDAC-MACS are similar to DAC-MACS except those added $h_{j,k}$ need to be securely stored in users' secret keys by the corresponding AA_k for each user. However, adding $h_{j,k}$ results in a $n_u|p|$ reducing of storage overhead on authority side comparing to that of DAC-MACS.

On the data owners side, DAC-MACS and NEDAC-MACS incur the same storage overhead better than that of DACC when n_c is large in the system. The reason is that DACC requires the data owners to hold the encryption secret for each ciphertext, whereas in DAC-MACS and NEDAC-MACS, public keys of attribute and AA_k are mainly needed to be stored.

On each user side, the storage overheads of DAC-MACS and NEDAC-MACS also stay identical and both require less overhead than that of DACC when $n_{c,x}$ is large in the system. This is due to the reason that the storage overhead in DAC-MACS and NEDAC-MACS mainly comes from the global secret keys and the secret keys of users, whereas DACC requires each user to store both the secret keys issued by all the AAs and the ciphertext components which are associated with the revoked attribute.

The three schemes require almost the same storage overhead on the cloud server side since the storage mainly comes from the ciphertext, where we do not consider the plaintext size encrypted by symmetric keys.

7.2 Computation Overhead

Table V details the computation overhead comparison among the schemes and it indicates that NEDAC-MACS incurs less computation overhead than DACC and is comparable to DAC-MACS. DACC needs one pairing computation to encrypt each plaintext and requires more for decryption so that it incurs the largest amount of computation overhead both in encryption on data owners and decryption on user side. Moreover, since the computationally intensive and storage demanding jobs of decryption process (TKGen) in DAC-MACS and NEDAC-MACS scheme are partitioned and offloaded on traditional cloud resources, it can greatly reduce the workload level on user side. However, DACC requires the data owners to change all stored ciphertext containing $x \in I_u$, thus incurs a heavy computation overhead for attribute operations off cloud due to the huge amount of involved ciphertext.

The computation overhead comparison is also conducted by simulating the whole architectures of DACC, DAC-MACS, and NEDAC-MACS with PBC library version 0.5.12 [27], on an Ubuntu system 14.04 with a 2.5 GHz processor and 2G RAM. We adopt the ordinary symmetric elliptic curve (type D internals) with elliptic curve group size 159-bit and embedding degree 6. Each value in Figures 2, 3, 4 is the mean of 10 simulation trials.

As shown in Fig.2, Fig.3, and Fig.4, the consuming time comparison of both encryption and decryption are conducted according to two parameters: the number of authorities and the number of attributes per authority. The revocation computation is based on the number of revoked attributes.

In Fig.2, suppose each user holds the same number of assigned attributes from each attribute. In Fig.2, we set 10 as the involved number of attributes from each attribute authority, and also the involved number of authority. Fig.2 illustrates that the three schemes nearly have the same efficiency in encryption time for data owners, since they are all based on CP-ABE.

In Fig.3 a), we set 10 as the number of involved attributes of user from each AA, and the number of involved authorities linked to the ciphertext is also set to be 10 in Fig.3 b). Fig.3 shows that NEDAC-MACS incurs less computation overhead than DACC and is relatively same as DAC-MACS in efficiency of decryption time for users. The reason is the most computation-consuming job of decryption is offloaded on cloud server in DAC-MACS and NEDAC-MACS scheme, which greatly reduces the workload level on user side. Moreover, the secret keys of users in in NEDAC-MACS and DAC-MACS systems can all be available in public for the cloud servers, which enhances the computation efficiency at the *Data Decryption* phase when comparing with the DACC.

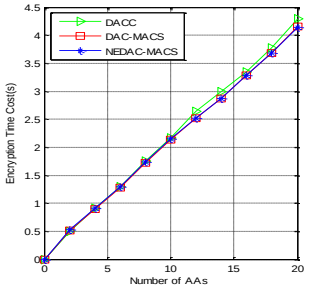
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2441698, IEEE Transactions on Services Computing

TABLE V
COMPUTATION OVERHEAD COMPARISON OF CP-ABE SCHEMES

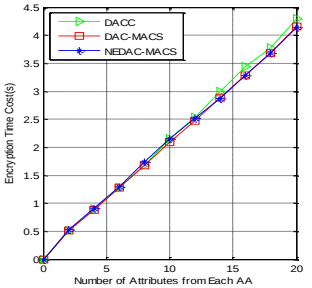
Scheme	Crypt-Computation		Revocation Computation off Cloud
	Encryption	Decryption	
DACC	$t_p + (4t_c + 1)t_m$	$2t_c t_p + t_c t_m$	$\sum_{x \in I_u} 3n_{non,x}\, t_m$
DAC-MACS	$(t_c + \lvert I_A \rvert + 1)t_m$	t_m	$(n_{non,x} + n_{c,x} + 1)\, t_m$
NEDAC-MACS	$(t_c + \lvert I_A \rvert + 1)t_m$	t_m	$(2n_{non,x} + n_{c,x} + 1)\, t_m$

TABLE VI
COMMUNICATION OVERHEAD COMPARISON OF CP-ABE SCHEMES

Scheme	Attribute Revocation		Encryption	Decryption
	Key Update	CT Update		
DACC	N/A	$(n_{c,x} n_{non,x} + 1)\lvert p \rvert$	$(3t_c + 1)\lvert p \rvert$	$(3t_c + 1)\lvert p \rvert$
DAC-MACS	$n_{non,x}\lvert p \rvert$	$\lvert p \rvert$	$(3t_c + 3)\lvert p \rvert$	$(3t_c + 4)\lvert p \rvert$
NEDAC-MACS	$2n_{non,x}\lvert p \rvert$	$3\lvert p \rvert$	$(3t_c + 3)\lvert p \rvert$	$(3t_c + 4)\lvert p \rvert$

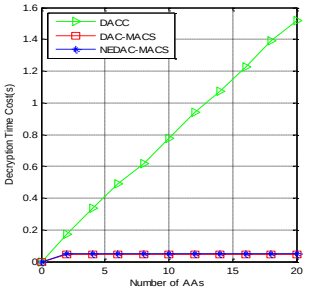


a) Comparison 1

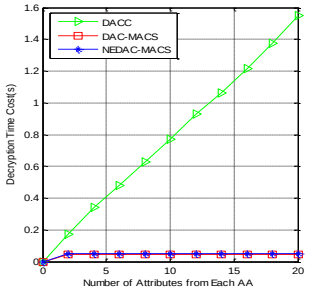


b) Comparison 2

Fig. 2. Comparison of Encryption Time on Data Owners.



a) Comparison 1



b) Comparison 2

Fig. 3. Comparison of Decryption Time on Users.

Fig.4 gives the comparison of revocation computation time off cloud (include secret key update by nonrevoked users and update key generation by authority) according to the number of revoked attributes appeared in the ciphertext. It indicates that NEDAC-MACS performs better than DACC and incurs a slight efficiency reducing than DAC-MACS on the revocation computation off cloud.

7.3 Communication Overhead

The communication overhead comparison is conducted among the three schemes regardless of the common fields (M, ρ) overhead in the ciphertext. Table VI details the communication overhead comparison.

It is easy to find that the three schemes incur almost the same communication overhead at both *Encryption* and *Decryption* phase since they all need to send the ciphertext in the two phases. At *Attribute Revocation* phase, when the ciphertext is reencrypted in DACC, some of its components related to the revoked attributes should be sent to each nonrevoked user who holds the revoked attributes, which increases the overhead of communication compared with DAC-MACS and NEDAC-MACS. We note that in NEDAC-MACS, $\mathrm{L}_{uid,x_{aid}}$ of secret keys of U_{uid} are linked with attribute x_{aid}, thus it requires the transmitted update message LUK for updating when x_{aid} of U_{uid} is revoked from AA_{aid}, which results in corresponding reducing of communication efficiency compared with DAC-MACS. However, the overall communication overhead of NEDAC-MACS is relatively the same as that of DAC-MACS and has advantage over DACC.

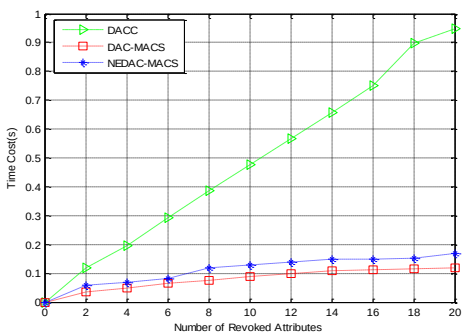


Fig. 4. Comparison of Decryption Time Off-Cloud.

8 CONCLUSION

In this paper, we first give two attacks on DAC-MACS and EDAC-MACS for their backward revocation security. Then, a new effective data access control scheme for multiauthority cloud storage systems (NEDAC-MACS) is proposed to withstand the two vulnerabilities in section 3 and thus to enhance the revocation security. NEDAC-MACS can withstand the two vulnerabilities even though the nonrevoked users reveal their received key update keys to the revoked user. In NEDAC-MACS, the revoked user has no chance to decrypt any objective ciphertext even if it actively eavesdrop to obtain an arbitrary number of nonrevoked users' Key Update Keys KUK or collude with some nonrevoked users or obtain any transmitted information such as Ciphertext Update Keys CUK. Then, formal cryptanalysis of NEDAC-MACS is presented to prove its improved security. Finally, the performance simulation shows the overall storage, computation, and communication overheads of the NEDAC-MACS are superior to that of DACC and relatively same as that of DAC-MACS.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (NO.61202448), and the Key Laboratory Program of Information Network Security of Ministry of Public Security (No.C14610).

REFERENCES

[1] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *J. Network and Computer Applica-*

tions, vol. 34, no. 1, pp. 1-11, Jul. 2010

[2] K. Yang, X. Jia, and K. Ren, "DAC-MACS: Effective data access control for multiauthority cloud storage systems," *IEEE Trans. Information Forensics and Security*, vol. 8, no. 11, pp. 1790-1801, Nov. 2013

[3] Kan Yang and Xiaohua Jia, "Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage," *IEEE Trans. Parallel and Distributed Systems*, vol.25, no.7, pp.1735-1744, July 2014

[4] A. Sahai and B. Waters, "Fuzzy identity-based encryption," *Proc. EUROCRYPT' 05*, pp. 457-473, 2005

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," *Proc. ACM Conf. Computer and Comm.* Security, pp. 89-98, 2006

[6] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proc.IEEE Symp.Security & Privacy*, pp. 321-334, 2007

[7] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-Based Encryption with Non-Monotonic Access Structures," *Proc. ACM Conf. Computer and Comm.* Security, pp. 195-203, 2007

[8] L. Cheung and C. C. Newport, "Provably secure ciphertext policy ABE," *Proc. ACM Conf. Computer & Communications Security*, pp. 456-465, 2007

[9] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: distributed access control in clouds," *Proc. TrustCom'11*, pp. 91-98, IEEE, 2011

[10] Zhiguo Wan, Jun'e Liu, and Deng, R.H., "HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing," *IEEE Trans. Information Forensics and Security*, vol.7, no.2, pp. 743-754, April 2012

[11] Junzuo Lai, Deng, R.H., Chaowen Guan, and Jian Weng, "Attribute-Based Encryption With Verifiable Outsourced Decryption," *IEEE Trans. Information Forensics and Security*, vol.8, no.8, pp. 1343-1354, Aug. 2013

[12] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 22, no. 7, pp.1214-1221, Jul. 2011

[13] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Trans. Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2271-2282, Oct. 2013

[14] M. Chase and S. S. M. Chow, "Improving privacy and security in multiauthority attribute-based encryption," *Proc. CCS'09*, pp.121-130, 2009

[15] M. Chase, "Multiauthority attribute-based encryption," *Proc.TCC'07*, pp. 515-534, Springer, 2007

[16] S. Müller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," *Proc. 11th Int. Conf. Information Security and Cryptology*, pp. 20-36, Springer, 2008

[17] A. B. Lewko and B. Waters, "Decentralizing Attribute-based Encryption," *Proc. EUROCRYPT'11*, pp. 568-588, Springer, 2011

[18] H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure threshold multiauthority attribute based encryption without a central authority," *Inf. Sci.*, vol.180, no. 13, pp. 2618-2632, 2010

[19] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext policy attribute-based encryption with accountability," *Proc. ASIACCS'11*, pp. 386-390, ACM, 2011

[20] Xuefeng Liu, Yuqing Zhang, Boyang Wang, and Jingbo Yang, "Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud," *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1182-1191, June 2013

[21] Zhongma Zhu, Zemin Jiang, Rui Jiang, "The Attack on Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud," *Proc. ISCC 2013*, Guangzhou, Dec.7, 2013, pp. 185-189

[22] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," *in Proc. USENIX Security Symp.*, San Francisco, CA, USA, 2011

[23] J. Z. Lai, R. H. Deng, C. W. Guan, and J. Weng, "Attribute-Based Encryption With Verifiable Outsourced Decryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, pp. 1343-1354, Aug 2013

[24] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Dept. Inst. of Tech., Technion Univ., Haifa, 1996

[25] J. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," *Advances in Cryptology-CRYPTO*, vol. 403, pp. 27-36, 1988

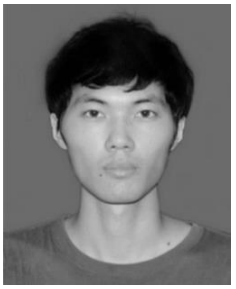
[26] L. Xu, X. Wu, and X. Zhang, "CL-PRE: a Certificateless Proxy Re-Encryption Scheme for Secure Data Sharing with Public Cloud,"in *the Proceedings of ACM ASIACCS 2012*, 2012

[27] Pairing Based Cryptography (PBC) Library. [Online]. Available: http://crypto.stanford.edu/pbc/

[28] W. Mao, "Modern Cryptography: Theory and Practice," New Jersey: Prentice Hall PTR, 2003

[29] M. Bellare, P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," *Pro. CCS. ACM Press, Springer*, 1993

[30] Dolev, D., Yao A. C., "On the security of public key protocols", *IEEE Trans. Information Theory*, vol. IT-29, no. 2, pp. 198–208, 1983



Xianglong Wu is a student in the Department of Information Science and Engineering, Southeast University, China. He majors in information security, and mainly engages in cloud storage security protocols research.

Rui Jiang is now an associate Professor at Southeast University, China. He received his Ph D degree at Shanghai Jiaotong University, Shanghai, China in 2005. His current research interests include secure analysis and design of communication protocols, secure mobile cloud computing, secure network and systems communications, mobile voice end-to-end secure communications, and applied cryptography.

Bharat Bhargava is a Professor of Computer Science at Purdue University. He is conducting research in security and privacy issues in distributed systems and sensor networks. This involves identity management, secure routing and dealing with malicious hosts, adaptability to attacks, and experimental studies. His recent work involves attack graphs for collaborative attacks. Prof. Bhargava has won five best paper awards in addition to the technical achievement award and golden core award from IEEE, and is a fellow of IEEE. He received Outstanding Instructor Awards from the Purdue chapter of the ACM in 1996 and 1998. He has graduated the largest number of Ph.D students in CS department and is active in supporting/mentoring minority students. In 2003, he was inducted in the Purdue's Book of Great Teachers. He has graduated the largest number of women Ph.D students and the first African American student Ph.D in CS department. He is editor-in-chief of three journals and serves on over ten editorial boards of international journals. Professor Bhargava is the founder of the IEEE Symposium on Reliable and Distributed Systems, IEEE conference on Digital Library, and the ACM Conference on Information and Knowledge Management. Bhargava has worked extensively at research laboratories of Air Force and Naval. He has successfully completed several Darpa and Navy STTR proposals. He is working with General Motor Corporation in analyzing use of sensors in cars and other vehicle. He has organized an NSF workshop on V2V wireless network.