Outsourcing Encryption of Attribute-Based Encryption with MapReduce

Jingwei Li[1], Chunfu Jia[1], Jin Li[2], and Xiaofeng Chen[3]

[1] College of Information Technical Science, Nankai University
lijw@mail.nankai.edu.cn,
cfjia@nankai.edu.cn
[2] School of Computer Science, Guangzhou University
lijin@gzhu.edu.cn
[3] State Key Laboratory of Integrated Service Networks, Xidian University
xfchen@xidian.edu.cn

Abstract. Attribute-based encryption (ABE) is a promising cryptographic tool for fine-grained access control. However, the computational cost in encryption commonly grows with the complexity of access policy in existing ABE schemes, which becomes a bottleneck limiting its application. In this paper, we formulize the novel paradigm of outsourcing encryption of ABE to cloud service provider to relieve local computation burden. We propose an optimized construction with MapReduce cloud which is secure under the assumption that the master node as well as at least one of the slave nodes is honest. After outsourcing, the computational cost at user side during encryption is reduced to approximate four exponentiations, which is constant. Another advantage of the proposed construction is that the user is able to delegate encryption for any policy.

1 Introduction

Recently, much attention has been attracted by a new public-key primitive called attribute-based encryption (ABE). ABE achieves flexible one-to-many encryption instead of one-to-one, which has significant advantage over the traditional public key primitives. ABE thus is envisioned as an important tool for addressing the problem of secure and fine-grained data sharing and access control on untrusted server in cloud computing. Until now, there are two kinds of ABE having been proposed: key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE). In KP-ABE, the access policy is assigned in private key, whereas, in CP-ABE, it is specified in ciphertext.

However, one of the main efficiency drawbacks of ABE is that the computational cost in encryption phase grows with the complexity of the access formula. Thus, before ABE can be widely deployed in cloud computing for the purpose of providing secure access control, there is an increasing need to improve its efficiency. To address this problem, outsourced ABE, which provides a way to outsource encryption or/and decryption to third party service providers without revealing data or private keys, was introduced [17][28]. Outsourced ABE has a

T.W. Chim and T.H. Yuen (Eds.): ICICS 2012, LNCS 7618, pp. 191–201, 2012.

wide range of applications. For example, in the cloud computing environment which has mobile devices or sensors as information collection nodes, user terminal (e.g. mobile device) has constrained computation ability to independently finish basic encryption or decryption to protect sensitive data residing in public cloud. Outsourced ABE allows user to perform heavy encryption or/and decryption through "borrowing" the computation resources from a third party service provider. Therefore, with this paradigm computation/storage intensive tasks can be performed even by resource-constrained users.

1.1 Contribution

In this paper, concerning on outsourcing encryption of ABE, we formulize the security definition for this novel paradigm. We propose an outsourced ABE construction with delegated encryption. In this construction, user is to control the trival policy while a two-leveled MapReduce paradigm is utilized to produce a partial ciphertext for the user specified policy. The proposed construction is secure under the assumption that the master node as well as at least one of the slave nodes in MapReduce cloud is honest. This assumption is weaker than those assumptions in previous work which require all the nodes in the cloud are honest. Furthermore, we state that another advantage of our construction is that through introducing trivial policy, it is able to delegate encryption for any policy, while in previous work [28], user is required to specify a hybrid one connected by an AND gate.

1.2 Related Work

The notion of ABE, which was introduced as fuzzy identity-based encryption in [25], was firstly dealt with by Goyal et al. [16]. Two different and complementary notions of ABE were defined. In KP-ABE, each ciphertext is labeled with a set of attributes and each key is associated with an access structure. On the contrary, In CP-ABE, each private key is identified by a set of attributes and each ciphertext is labeled with an access structure. A construction of KP-ABE was provided in the same paper [16], while the first CP-APE construction supporting tree-based structure in generic group model is presented by Bethencourt et al. [5]. Subsequently, a number of variants of ABE schemes have been proposed since its introduction [8][22][27][20][26][21][24]. However, almost all of them requires a large number of exponentiations at user side during encryption.

To reduce the load at local, it always desires to deliver expensive computational tasks outside. Actually, the problem that how to securely outsource different kinds of expensive computations has drew much attention from theoretical computer science community [3][2][4][1]. But they are not suitable for reliving ABE computational overhead of exponentiations at user side. To achieve this goal, the traditional approach is to utilize server-aided techniques [6][19][18][7]. However, previous server-aided techniques are oriented to accelerating the speed of exponentiation using untrusted servers. Directly utilizing these techniques in ABE will not work efficiently.

Another approach might be to leverage recent general outsourcing technique or delegating computation [15][13][12][9][14] based on fully homomorphic encryption or interactive proof systems. However, Gentry [14] has shown that even for weak security parameters on "bootstrapping", the operation of homomorphic encryption would take at least 30 seconds on a high performance machine. Therefore, even if the privacy of the inputs and outputs can be preserved by utilizing these general techniques, the computational overhead is still huge and impractical.

Another two related work similar to us are [17] and [28]. In [17], a novel paradigm for outsourcing decryption of ABE is provided while in [28] the authors presented the PP-CP-ABE (privacy preserving cipher policy attribute-based encryption) which allows to securely outsource both decryption and encryption to third party service providers. Comparing with our work, i) the former has a different goal aiming at partial decryption delegation but we consider on outsourcing encryption of ABE; ii) the latter is the inspiration of this paper. Based on Zhou's work [28], we formulize the notion of outsourcing encryption of ABE and propose an optimized construction to enhance Zhou's scheme [28] in both security and functionality.

1.3 Organization

This paper is organized as follows. In Section 2 we describe the system model of our scheme. The construction and its security analysis are presented in Section 3. Finally, we draw conclusion in Section 4.

2 System Model

2.1 MapReduce

A two-leveled MapReduce [10] which is a software framework for supporting data-intensive computing, comprises a set of slave computer nodes and a master computer node. We now give an overview of MapReduce's workflow as follows.

- **Upload Phase**. User contacts with the master node to get knowledge which slave nodes are free. Then, he/she goes on to upload a set of data and "operations", i.e. complied Java classes, to these slave nodes to produce partially encrypted ABE ciphertext.
- **Map Phase**. After data and implementations have been uploaded, the MapReduce is triggered. Each involved slave node becomes a "mapper" node, which scans the information uploaded on it to obtain the key-value pair. Furthermore, the mapper node takes the key-value pair as input and executes the map function to generate an intermediate key-value pair. The phase can be denoted as $\mathsf{Map}(k, v) \rightarrow (k', v')$.
- **Reduce Phase**. After each slave node finishes its assigned task, MapReduce starts the "Reduce" phase, in which the master node is selected as the reducer. The reducer executes reduce function on the set of intermediate pairs (k', v') with the same key and outputs the final result. The phase can be denoted as $\mathsf{Reduce}(\{(k', v')\}) \rightarrow Output$.

2.2 System Model

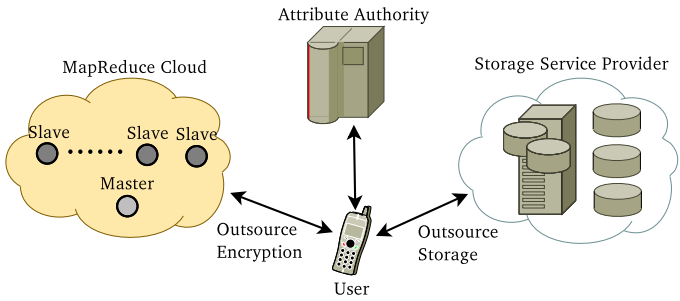

Fig. 1. System Model for Outsourcing Encryption of ABE

We present proposed system model for outsourcing encryption of ABE scheme in Fig. 1. Comparing with that for traditional ABE, a MapReduced cloud is involved to execute the delegated ABE encryption task.

With the custom in [17], we denote $(I_{\text{enc}}, I_{\text{key}})$ as the input to encryption and key generation in ABE. Therefore, in CP-ABE, $(I_{\text{enc}}, I_{\text{key}}) = (\mathbb{A}, \omega)$ while that is $(\omega, \mathbb{A})$ in KP-ABE, where ω and $\mathbb{A}$ are attribute set and access structure respectively. Then, based on the proposed system model, we provide algorithm definitions as follows.

- **Setup**(λ) : The setup algorithm takes as input – a security parameter λ. It outputs the public key PK and the master key MK.
- **KeyGen**$(I_{\text{key}}, \text{MK})$: For each user's private key request on I_{key}, the key generation algorithm takes as input – an access structure (or attribute set) I_{key} and the master key MK. It outputs the private key SK.
- **Encrypt**$_{\text{U}}(I_{\text{enc}}, \text{M})$: The encryption algorithm at user side takes as input – an attribute set (or access structure) I_{enc} and the message M. It outputs the partially encrypted ciphertext at local CT_{U} and the set of outrourcing encryption keys $\{\text{OEK}_i\}_{i=1}^{n}$ where n is the number of slave nodes in MapReduce cloud to be assigned.
- **Encrypt**$_{\text{MR}}(I_{\text{enc}}, \{\text{OEK}_i\}_{i=1}^{n})$: The delegated encryption algorithm at the MapReduce cloud takes as input – an attribute set (or access structure) I_{enc} and the set of outsourcing encryption keys $\{\text{OEK}_i\}_{i=1}^{n}$. It outputs the partially encrypted ciphertext at MapReduce CT_{MR}.
- **Decrypt**(CT, SK) : The decryption algorithm takes as input – a ciphertext $\text{CT} = (\text{CT}_{\text{U}}, \text{CT}_{\text{MR}})$ which was assumed to be encrypted under the attribute set (or access structure) I_{enc} and the private key SK for access structure (or attribute set) I_{key}. It outputs the message M if $\gamma(I_{\text{key}}, I_{\text{enc}}) = 1$, otherwise outputs $\perp$, where γ is a predicate predefined.

2.3 Adversary Model

In the system, we assume that the master node as well as at least one of the slave nodes in MapReduce cloud is "honest-but-curious". Specifically, they will follow our proposed protocol but try to find out as much private information as possible based on their possession. We note that this adversary model is weaker than that in [28], which requires the whole encryption service provider is "honest-but-curious".

Therefore, two types of adversaries are considered: i) The MapReduce service provider, who can potentially access all the information including encrypted message, the outsourcing encryption keys, etc; ii) A curious user, who can obtain his individual private key and share his authentication with others.

Having such intuition, we will follow the replayable chosen-ciphertext attack (RCCA) security in [17] and define RCCA security for our setting.

Setup. The challenger runs setup algorithm and gives the public key PK to the adversary.

Phase 1. The challenger initializes an empty set D_{key}. Proceedingly, adversary is allowed to make the following queries for several times.

- *Private key query.* Upon receiving I_{key}, challenger runs key generation algorithm on I_{key} to obtain SK and returns SK after setting $D_{\text{key}} = D_{\text{key}} \cup \{I_{\text{key}}\}$.
- *Encryption query.* Upon receiving M, j and I_{enc}, challenger runs encryption algorithm totally to obtain CT and $\{\text{OEK}_i\}_{i=1}^{n}$. But only return CT and $\{\text{OEK}_i\}_{i=1,i\neq j}^{n}$ to adversary.
- *Decryption query.* Upon receiving CT encrypted under I_{enc}, challenger generates SK for I_{key} and performs decryption on CT to obtain M. Finally return M.

Challenge. Adversary submits two messages M_0 and M_1. In addition the adversary gives I^*_{enc} satisfying that $\gamma(I_{\text{key}}, I^*_{\text{enc}}) = 0$ for all $I_{\text{key}} \in D_{\text{key}}$. Challenger flips a random coin b and encrypts M_b under I^*_{enc} totally. Finally return the resulting ciphertext CT^*.

Phase 2. Phase 1 is repeated with the restrictions: i) Adversary cannot issue *private key query* on I_{key} where $\gamma(I_{\text{key}}, I^*_{\text{enc}}) = 1$. ii) *Decryption query* will be answered normally except that the response would be either M_0 or M_1, then challenger responds with a special message instead.

Guess. Adversary outputs a guess b' of b.

Definition 1. *A CP-ABE or KP-ABE scheme with outsourced encryption is secure against replayable chosen-ciphertext attack if all polynomial time adversaries have at most a negligible advantage in the game defined above.*

Finally, beyond the RCCA security, we also specify that i) An ABE with delegated encryption is CPA-secure (or secure against chosen-plaintext attack) if

no polynomial time adversary has non-negiligible advantage in a modified game, in which the *decryption query* in both phase 1 and phase 2 is removed; ii) An ABE with delegated encryption is secure in selective model if no polynomial time adversary has non-negiligible advantage in a modified game, in which the I^*_{enc} submission is advanced to an additional stage before setup.

3 Proposed Construction

3.1 Access Structure

In the proposed construction, private keys will be identified with a set ω of descriptive attributes, while the encryption policy is specified as an access tree $\mathcal{T}$. We will briefly review the concept of access tree in [5] as well as [28] before providing our construction.

Let $\mathcal{T}$ be a tree representing an access structure, in which each interior node is a threshold gate (i.e. AND gate or OR gate) while the leaves are associated with attributes. A user is able to decrypt a ciphertext with a given key if and only if there is an assignment of attributes from the private key to leaf nodes of the tree such that the tree is satisfied.

To facilitate working with the access tree, we define a few notations and functions as follows.

- num_x is the number of children of an interior node x. In order to uniquely identify each child, an ordering between the children of every node is defined, that is, the children of node x is numbered from 1 to num_x. Therefore, if assuming y is the child of node x, we could denote $\mathsf{index}(y)$ as such number associated with the node y.
- k_x is the threshold value of an interior node x, specifically, when $k_x = 1$, the threshold gate at x is OR gate and when $k_x = num_x$, that is an AND gate. We note that if x is a leaf node it is described by an attribute and a threshold value $k_x = 1$.
- The function $\mathsf{parent}(x)$ returns the parent of the node x in the tree. $\mathsf{attr}(x)$ returns the attribute associated with the leaf node x.

3.2 Our Construction

We utilize the MapReduce paradigm to split the secret s used in ciphertext into n pieces and each of them is dealt by slave node separately. Another trick used is to introduce a trival policy $\mathcal{T}_\theta$ consisting of a single leaf node θ to improve [28]. Specifically, our scheme supports to delegate encryption with any generic tree-based access policy.

Before providing construction, we define some basic tools used later.

Definition 2 (Bilinear Map). *Suppose $\mathbb{G}, \mathbb{G}_T$ be cyclic groups of prime order q, writing the group action multiplicatively. g is a generator of $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a map with the following properties:*

- *Bilinearity:* $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ *for all* $g_1, g_2 \in \mathbb{G}$, *and* $a, b \in_R \mathbb{Z}_q$;
- *Non-degeneracy: There exists* $g_1, g_2 \in \mathbb{G}$ *such that* $e(g_1, g_2) \neq 1$, *in other words, the map does not send all pairs in* $\mathbb{G} \times \mathbb{G}$ *to the identity in* $\mathbb{G}_T$;

We also define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_q$ and a set S of elements in $\mathbb{Z}_q$:

$$\Delta_{i,S} = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$$

Then, the proposed construction in detail is shown as follows.

- **Setup**(λ) : The setup algorithm is executed by authority. Select a bilinear group $\mathbb{G}$ of prime order q with generator g and two random integers $\alpha, \beta \in \mathbb{Z}_q$, and define a hash function $H : \{0,1\}^* \to \mathbb{G}$ modeled as a random oracle. Finally, output the public key $\mathrm{PK} = (\mathbb{G}, H(\cdot), g, h = g^\beta, e(g,g)^\alpha)$ and the master key $\mathrm{MK} = (\beta, g^\alpha)$.
- **KeyGen**(ω, MK) : For each user's private key request, the authority runs the key generation algorithm. Choose $r \in_R \mathbb{Z}_q$ and $r_j \in_R \mathbb{Z}_q$ for each attribute $j \in \omega \cup \{\mathsf{attr}(\theta)\}$. Finally compute and output the private key as $\mathrm{SK} = (d = g^{\frac{\alpha+r}{\beta}}, \{d_{j0} = g^r \cdot H(j)^{r_j}, d_{j1} = g^{r_j}\}_{j \in \omega \cup \{\mathsf{attr}(\theta)\}})$.
- **Encrypt**$_{\mathrm{U}}(\mathcal{T}_{\mathrm{U}}, \mathrm{M})$: To encrypt a message M with access policy $\mathcal{T}_{\mathrm{U}}$, firstly pick an integer $s \in_R \mathbb{Z}_q$ and randomly select a 1-degree polynomial $q_R(\cdot)$ such that $q_R(0) = s$. Furthermore, let $s_1 = q_R(1)$ and $s_2 = q_R(2)$. Then, make an n-splits on s_1 by randomly selecting $s_{11}, \ldots, s_{1n} \in \mathbb{Z}_q$ with $s_{11} + \ldots + s_{1n} = s_1$ and set $\mathrm{OEK}_i = s_{1i}$ for $i = 1, \ldots, n$. Finally, it outputs the partially encrypted ciphertext at local $\mathrm{CT}_{\mathrm{U}} = (\mathcal{T}_{\mathrm{U}} \wedge \mathcal{T}_\theta, \widetilde{E} = \mathrm{M}e(g,g)^{\alpha s}, E = h^s, (E_{\theta 0} = g^{s_2}, E_{\theta 1} = H(\mathsf{attr}(\theta))^{s_2}))$ and the set of outsourcing keys $\{\mathrm{OEK}_i\}_{i=1}^n$.
- **Encrypt**$_{\mathrm{MR}}(\mathcal{T}_{\mathrm{U}}, \{\mathrm{OEK}_i\}_{i=1}^n)$: The delegated encryption algorithm is executed by the MapReduce cloud. As described in Section 2.1, user uploads $(\mathcal{T}_{\mathrm{U}}, \mathrm{OEK}_i)$ which is scaned as the key-value pair to the i-th slave node. Then the MapReduce is triggered.
  - **Map.** The slave node i finishes the "partial encryption" task by choosing a $(k_x - 1)$-degree polynomial $q_x^{(i)}(\cdot)$ for each node x (including the leaves) in the tree $\mathcal{T}_{\mathrm{U}}$ in a top-down manner. The selected polynomial $q_x^{(i)}(\cdot)$ must satisfy the restriction that $q_x^{(i)}(0) = s_{1i}$ if x is the root node in $\mathcal{T}_{\mathrm{U}}$, otherwise $q_x^{(i)}(0) = q_{\mathsf{parent}(x)}^{(i)}(\mathsf{index}(x))$. Let Y_{U} denote the set of leaf nodes in $\mathcal{T}_{\mathrm{U}}$, then the partially encrypted ciphertext at slave node i is computed as $\mathrm{CT}_{\mathrm{MR}}^{(i)} = (\{E_{y0}^{(i)} = g^{q_y^{(i)}(0)}, E_{y1}^{(i)} = H(\mathsf{attr}(y))^{q_y^{(i)}(0)}\}_{y \in Y_{\mathrm{U}}})$. The map function for outsourced encryption is described as $\mathsf{Map}(\mathcal{T}_{\mathrm{U}}, \mathrm{OEK}_i) \to (\mathcal{T}_{\mathrm{U}}, \mathrm{CT}_{\mathrm{MR}}^{(i)})$.
  - **Reduce.** Let $q_y(x) = \sum_{i=0}^n q_y^{(i)}(x)$ for $y \in Y_{\mathrm{U}}$. The master node is selected as the reducer. Then, after gathering all the intermediate key-value pairs $\{(\mathcal{T}_{\mathrm{U}}, \mathrm{CT}_{\mathrm{MR}}^{(i)})\}_{i=1}^n$ sent from the other slave nodes, reducer

computes $\mathrm{CT}_{\mathrm{MR}} = (\{E_{y0} = \prod_{i=1}^{n} E_{y0}^{(i)} = g^{q_y(0)}, E_{y1} = \prod_{i=1}^{n} E_{y1}^{(i)} = H(\mathsf{attr}(y))^{q_y(0)}\}_{y \in Y_{\mathrm{U}}})$. The reduce function for outsourced encryption can be described as $\mathsf{Reduce}(\{(\mathcal{T}_{\mathrm{U}}, \mathrm{CT}_{\mathrm{MR}}^{(i)})\}_{i=1}^{n}) \rightarrow \mathrm{CT}_{\mathrm{MR}}$.

Finally, such ciphertext $\mathrm{CT}_{\mathrm{MR}}$ is sent back to user.

– **Decrypt**$(\mathrm{CT}, \mathrm{SK})$: The recursive decryption algorithm is executed by user. Suppose $\mathrm{CT} = (\mathrm{CT}_{\mathrm{U}}, \mathrm{CT}_{\mathrm{MR}})$ is encrypted under the policy corressponding to SK, then the decryption is followed in a down-top manner.

- For each leaf node y in the hybrid access tree $\mathcal{T}_{\mathrm{U}} \wedge \mathcal{T}_{\theta}$, let $i = \mathsf{attr}(y)$ and the decryption is presented as follows.

$$F_y = \frac{e(E_{y0}, d_{i0})}{e(d_{i1}, E_{y1})} = \frac{e(g^{q_y(0)}, g^r H(i)^{r_i})}{e(g^{r_i}, H(\mathsf{attr}(y))^{q_y(0)})} = e(g,g)^{rq_y(0)}$$

- For each interior node y, let S_y be an arbitrary k_y-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node is not satisfied and the function returns $\perp$. Then, the decryption is presented as follows.

$$\begin{aligned} F_y &= (\prod_{z \in S_y} F_z)^{\Delta_{i,S_y}(0)} \\ &= (e(g,g)^{\sum_{z \in S_y} rq_z(0)})^{\Delta_{i,S_y}(0)} = e(g,g)^{\sum_{z \in S_y} rq_{\mathsf{parent}(z)}(\mathsf{index}(z))\Delta_{i,S_y}(0)} \\ &= e(g,g)^{r\sum_{z \in S_y} q_y(i)\Delta_{i,S_y}(0)} \\ &= e(g,g)^{rq_y(0)} \end{aligned}$$

Finally, we are able to decrypt the root node by computing $F_R = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$. Then, the ciphertext can be decrypted by computing

$$\mathrm{M} = \frac{\widetilde{E}}{\frac{e(E,d)}{F_R}} = \frac{\mathrm{M}e(g,g)^{\alpha s}}{\frac{e(h^s, g^{\frac{\alpha+r}{\beta}})}{e(g,g)^{rs}}}$$

3.3 Security Analysis

Theorem 1. *The proposed construction is CPA-secure under the assumption that the master node as well as at least one of the slave nodes in MapReduce cloud is "honest-but-curious" in the generic group model.*

Proof. We observe that in the security game shown in Section 2.3, the challenge ciphertext has a component $\widetilde{E}$ which is randomly either $M_0 e(g,g)^{\alpha s}$ or $M_1 e(g,g)^{\alpha s}$. We can instead consider a modified game in which $\widetilde{E}$ is either $e(g,g)^{\alpha s}$ or $e(g,g)^{\nu}$, where ν is selected uniformly at random from $\mathbb{Z}_q$, and the adversary must decide which is the case. It is clear that any adversary has advantage ϵ in the origianl game can be transformed into an adversary that has advantage at least $\frac{\epsilon}{2}$ in the modified game. Then, we would like to bound the adversary's advantage in the modified game.

Then, we are to denote $\phi_0 : \mathbb{Z}_q \to \{0,1\}^{\log q}$ as a random encoding for the group operation $g^x \in \mathbb{G}$ and $\phi_1 : \mathbb{Z}_q \to \{0,1\}^{\log q}$ as a random encoding for $e(g,g)^x \in \mathbb{G}_T$. Let $g = \phi_0(1)$.

At setup time, simulator chooses $\alpha, \beta \in_R \mathbb{F}_q$. Note that if $\beta = 0$ which happens with probability $\frac{1}{q}$, then setup is aborted. Finally, publish public parameters $h = g^\beta$ and $e(g,g)^\alpha$ to adversary.

When the adversary calls for the evaluation of H on any string i, simulator maintains a list $\mathcal{L}$ to store the response to hash query. Upon receiving a string i, if the entry (i, g^{t_i}) exists in $\mathcal{L}$, straightforwardly return g^{t_i}. Otherwise, pick $t_i \in_R \mathbb{F}_q$ and return g^{t_i} after adding the entry (i, g^{t_i}) into $\mathcal{T}$.

When the adversary makes its j-th private key request on ω_j of attributes, challenger picks $r^{(j)} \in_R \mathbb{F}_q$ and comptutes $d = g^{\frac{\alpha + r^{(j)}}{\beta}}$ and we have $d_{i0} = g^{r^{(j)} + t_i r_i^{(j)}}$ and $d_{i1} = g^{r_i^{(j)}}$ for $i \in \omega_j \cup \{\mathsf{attr}(\theta)\}$ and $r_i^{(j)} \in_R \mathbb{Z}_q$.

When the adversary makes encryption request on M, j as well as $\mathcal{T}_\mathrm{U}$, the simulator chooses $s \in_R \mathbb{F}_q$. Then it splits s into s_1 and s_2 with linear secret sharing. i) For s_1, the simulator continues to split it into $s_{11}, \ldots, s_{1n}$ and uses the linear secret sharing scheme associated with $\mathcal{T}_\mathrm{U}$ to construct shares $\lambda_i^{(j)}$ of s_{1j} $(j = 1, \ldots, n)$ for all relevant attributes i. Then, the simulator makes a reduce by computing $E_{i0} = g^{\sum_{j=1}^k \lambda_i^{(j)}}$ and $E_{i1} = g^{t_i \sum_{j=1}^k \lambda_i^{(j)}}$ for each relevant attribute i. ii) For s_2, the simulation perform computation like Section 3.2 to obtain $E_{\theta 0} = g^{s_2}$ and $E_{\theta 1} = g^{t_\theta s_2}$. Finally, these values along with $\widetilde{E} = Me(g,g)^{\alpha s}, E = g^s$ and $\{s_{1i}\}_{i=1, i \neq j}^n$ are sent to adversary.

When adversary asks for challenge on $\mathrm{M}_0, \mathrm{M}_1$ and $\mathcal{T}^*$, simulator's action is identical to the response to encryption query, except that it picks $u \in_R \mathbb{Z}_q$ and constructs the encryption as follows: $\widetilde{E} = e(g,g)^u$ and $E = h^s$. For each relevant attribute i, $E_{i0} = g^{\lambda_i}$ and $E_{i1} = g^{t_i \lambda_i}$.

Subsequently, the response to the group operation is identical to that in [5]. Therefore, using the generic bilinear group model, it is able to be shown that with probability $1 - O(\frac{p^2}{q})$ taken over the randomness of the choice of variable values in the simulation, adversary's view in this simulation is identically distributed to what its view would have been if it had been given $\widetilde{E} = e(g,g)^{\alpha s}$, where p is the bound on the total number of group elements received from queries to hash functions, group $\mathbb{G}, \mathbb{G}_T$ and the bilinear map e, and from its interaction with security game. Therefore, the proposed construction is secure in the proposed model.

In our construction, the reduce operation is run by the master node which is honest. Moreover, a further split on s_1 is performed to "map" the "partial encryption" task onto n slave nodes to allow for concurrent execution. Since at least one of the slave nodes is honest, they are not able to recover s_1 to fake access policy even if $n - 1$ slave nodes collude.

Finally, we specify that though the proposed construction is secure against chosen-plaintext attack, it is allowed to be extended to the stronger RCCA-security guarantee by using simulation-sound NIZK proofs [23]. Alternatively, if we are willing to use random oracle, then we can use standard techniques such as the Fujisaki-Okamoto transformation [11].

4 Conclusion

In this paper, we formulize the paradigm of outsourcing encryption of ABE in cloud computing. We utilize MapReduce to propose a security enhanced construction which is secure under the assumption that the master node as well as at least one of the slave nodes in cloud is honest. Another advantage of the proposed construction is that it is able to delegate encryption for any access policy, instead of a special hybrid access policy. With our proposed outsourcing method, the computational cost at user side in encryption algorithm is reduced to four exponentiations, which is constant and does not grow with the number of attributes included in the ciphertext.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Nos. 61272423, 60973141, 61100224 and 60970144), Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20100031110030), Natural Science Foundation of Guangdong Province (No. 10451009101004573), and Foundation for Distinguished Young Talents in Higher Education of Guangdong Province (No. LYM10106).

References

1. Atallah, M.J., Frikken, K.B.: Securely outsourcing linear algebra computations. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, pp. 48–59. ACM, New York (2010)
2. Atallah, M.J., Li, J.: Secure outsourcing of sequence comparisons. International Journal of Information Security 4, 277–287 (2005)
3. Atallah, M.J., Pantazopoulos, K., Rice, J.R., Spafford, E.E.: Secure outsourcing of scientific computations. In: Zelkowitz, M.V. (ed.) Trends in Software Engineering. Advances in Computers, vol. 54, pp. 215–272. Elsevier (2002)
4. Benjamin, D., Atallah, M.J.: Private and cheating-free outsourcing of algebraic computations. In: Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust, PST 2008, pp. 240–245. IEEE Computer Society, Washington, DC (2008)
5. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy 2007, pp. 321–334 (May 2007)
6. Bicakci, K., Baykal, N.: Server Assisted Signatures Revisited. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 143–156. Springer, Heidelberg (2004)
7. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New Algorithms for Secure Outsourcing of Modular Exponentiations. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 541–556. Springer, Heidelberg (2012)
8. Cheung, L., Newport, C.: Provably secure ciphertext policy abe. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 456–465 (2007)
9. Chung, K.M., Kalai, Y., Liu, F.H., Raz, R.: Memory Delegation. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 151–168. Springer, Heidelberg (2011)
10. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008)
11. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

12. Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
13. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 169–178. ACM, New York (2009)
14. Gentry, C., Halevi, S.: Implementing Gentry's Fully-Homomorphic Encryption Scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
15. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008, pp. 113–122. ACM, New York (2008)
16. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
17. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of abe ciphertexts. In: Proceedings of the 20th USENIX Conference on Security, SEC 2011, p. 34. USENIX Association, Berkeley (2011)
18. Hohenberger, S., Lysyanskaya, A.: How to Securely Outsource Cryptographic Computations. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg (2005)
19. Jakobsson, M., Wetzel, S.: Secure Server-Aided Signature Generation. In: Kim, K.-C. (ed.) PKC 2001. LNCS, vol. 1992, pp. 383–401. Springer, Heidelberg (2001)
20. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
21. Li, J., Ren, K., Zhu, B., Wan, Z.: Privacy-Aware Attribute-Based Encryption with User Accountability. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C. (eds.) ISC 2009. LNCS, vol. 5735, pp. 347–362. Springer, Heidelberg (2009)
22. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 195–203. ACM, New York (2007)
23. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science, pp. 543–553 (1999)
24. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012)
25. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
26. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
27. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
28. Zhou, Z., Huang, D.: Efficient and secure data storage operations for mobile cloud computing. Cryptology ePrint Archive, Report 2011/185 (2011)