Reverse Outsource: Reduce the Cloud's Workload in Outsourced Attribute-Based Encryption Scheme

Fei Meng, Leixiao Cheng, and Mingqiang Wang*

School of Mathematics, Shandong University, Jinan Shandong 250100, China
Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education
School of Mathematical Sciences, Fudan University, Shanghai 200433, China
wangmingqiang@sdu.edu.cn

Abstract. Ciphertext-policy attribute-based encryption (CP-ABE) is a cryptographic technique known for ensuring fine-grained access control on encrypted data. However, one of the main drawbacks of CP-ABE is that the time required to decrypt the ciphertext is considerably expensive, since it grows with the complexity of access policy. Outsourcing partial decryption operations to the cloud eliminates the overheads for users, but it also inevitably increases the computational burden of the cloud. When millions of users are enjoying cloud computing services, it may cause huge congestion and latency.
In this paper, we propose a heuristic primitive called reverse outsourcing. Specifically, users outsource part of the decryption work to the cloud, which splits it up and dispatches each to different idle users. Idle users are those whos has some smart devices connected to the internet and not in use. It's like, the cloud employs many idle users to accomplish its own computing tasks. Then, we proposed a reverse outsourced CP-ABE scheme which provable secure under the BDBH assumptions.

Keywords: Fine-grained access control · Attribute-based encryption · Attribute update · Cloud computing · Outsourcing.

1 Introduction

Cloud computing is widely used to store and manage shared data and provide computing services. However, data and services on the cloud are usually open and accessible to anyone, so users are often advised to encrypt data before uploading it to the cloud.

In many application scenarios, such as in industrial, academic and medical fields, it is necessary for the data owner to establish a specific access control policy to decide who can decrypt the ciphertext. Attribute-based encryption (ABE) [8] initially introduced by Sahai et al. achieves fine-grained access control

* Corresponding author

and data privacy simultaneously. There are two types of ABE schemes, key-policy ABE (KP-ABE) [3] and ciphertext-policy ABE (CP-ABE) [1]. Bethencourt et al. [1] proposed the first CP-ABE scheme, in which access policies are built into ciphertexts. This design is close to the real application scenario. Since then, various kind of CP-ABE schemes have been proposed to achieve different functions. Cheung et al. [2] proposed a CP-ABE scheme based on the AND gate access structure. Horvath et al. [6] proposed a multi-authority CP-ABE scheme with identity-based revocation. Wang et al. [9] devised a CP-ABE scheme with hierarchical data sharing.

In CP-ABE, the pairing and exponentiation operations grow with the complexity of access policy, which means the user's computation cost is quite expensive. A normal way to reduce the workload of resource-limited user is to outsource these complex operations to the cloud. Green et al. [4] shifted partial decryption procedure to the cloud. Zhang et al. [10] achieved fully outsourcing, including key generation, ciphertext encryption and decryption. Fog computing, as an extension of cloud computing, also provides computing services to resource-limited users.

Although in the outsourced CP-ABE schemes [4, 6, 9, 10], the cloud is supposed to have almost unlimited computing capabilities, it is obviously not the case in reality. Outsourceing technique will increase cloud computing pressure, but little attention has been paid to reducing cloud's workload.

There are countless intelligent devices connected to the Internet all over the world. They have certain computing power and are not in use most of the time. Is there any way to aggregate this computing resources to provide computing services for the cloud?

Motivated by the above issues, we initially introduced reverse outsourcing to reduce cloud's workload. Specifically, the main contributions of our paper are as follows:

- **Reverse outsourcing**: We initially introduce a heuristic concept called *Reverse outsourcing*, i.e., the cloud is allowed to outsource computing tasks to *idle users* (with online devices that have some computing power but are not in use) to reduce its workload. We assume that the cloud will reward idle users after they complete the corresponding computing tasks correctly, and propose the *Rational idle user model* [5] in which users are more willing to earn rewards than saving computing resources. To demonstrate how reverse outsourcing works, we apply Reverse outsourcing to our proposed outsourced CP-ABE scheme, and use Game theory [7] and Nash equilibrium to analyze each rational idle user's strategy: when and only when all idle users execute the scheme honestly, each one can get the maximum benefit.

This paper is organized as follows. Section 2 describes the necessary preliminaries. Section 3 presents the system and security model. We give a concrete construction and explicit analysis of our scheme in section 4 and section 5 respectively. In the end, section 6 summarizes the paper and prospects for the future research.

2 Preliminaries

2.1 Access Structures

Definition 1 (Access structure [1]). *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

In this paper, attributes take the role of the parties and we only focus on the monotone access structure $\mathbb{A}$, which consists of the authorized sets of attributes. Obviously, attributes can directly reflect a user's authority.

Definition 2 (Access tree [1]). *Let $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If n_x is the number of children of a node x and k_x is its threshold value, then $0 \leq k_x \leq n_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = n_x$, it is an AND gate. Each leaf node x of the tree is describe by an attribute and a threshold value $k_x = 1$.*

We introduce some functions that will be used in scheme construction and security proof. $\mathsf{parent}(x)$ denotes the parent of the node x in the access tree. $\mathsf{att}(x)$ is defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree. The access tree $\mathcal{T}$ also defines an ordering between the children of every node, that is, the children of a node are numbered from 1 to n_x. The function $\mathsf{index}(x)$ returns such a number associated with the node x, where the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

Definition 3 (Satisfying an access tree [1]). *Let $\mathcal{T}$ be an access tree with root r. Denote by $\mathcal{T}_x$ the subtree of $\mathcal{T}$ rooted at the node x. Hence $\mathcal{T}$ is the same as $\mathcal{T}_r$. If a set of attributes γ satisfies the access tree $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(\gamma) = 1$. We compute $\mathcal{T}_x(\gamma)$ recursively as follows. If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(\gamma) = 1$ for all children x' of node x. $\mathcal{T}_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 if and only if $\mathsf{att}(x) \in \gamma$.*

2.2 Bilinear Map and DBDH Assumption

Algorithms in our scheme are mainly implemented by bilinear maps, which is presented as follows:

Let $\mathbb{G}_0$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a efficient computable bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$. The bilinear map e has a few properties: (1) Bilinearity: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$. (2) Non-degeneracy: $e(g, g) \neq 1$. We say

that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

The security of our scheme is based on the decisional bilinear Diffie-Hellman (DBDH) assumption, which is defined as follows: Given the bilinear map parameter $(\mathbb{G}_0, \mathbb{G}_T, p, e, g)$ and three random elements $(x, y, z) \in \mathbb{Z}_p^3$, if there is no probabilistic polynomial time (PPT) adversary $\mathcal{B}$ can distinguish between $(g, g^x, g^y, g^z, e(g, g)^{xyz})$ and $(g, g^x, g^y, g^z, \vartheta)$, we can say that the DBDH assumption holds, where ϑ is randomly selected from $\mathbb{G}_T$. More specifically, the advantage ϵ of $\mathcal{B}$ in solving the DBDH problem is defined as

$$\Big|\Pr[\mathcal{A}(g, g^x, g^y, g^z, Z = e(g, g)^{xyz}) = 1] - \Pr[\mathcal{A}(g, g^x, g^y, g^z, Z = R) = 1]\Big|. \quad (1)$$

Definition 4 (DBDH). *We say that the DBDH assumption holds if no PPT algorithm has a non-negligible advantage ϵ in solving DBDH problem.*

3 System and Security Model of CP-ABE

This section describes the fundamental descriptions of CP-ABE scheme: system parties, system model, threat model and security model.

3.1 System Parties

As shown in Fig. 1, our proposed reverse outsourced CP-ABE scheme requires the following five parties: Key Generation Center (KGC), Data Owner (DO), Cloud Server (CS), End User (EU). The specific role of each party is given as follows:

- Key Generation Center (KGC): The KGC is a fully trusted third party which is in charge of generating public parameters and user secret keys.
- Data Owner (DO): The DO owns data files and defines the access structure to encrypt the data, then uploads the ciphertext to the his/her own cloud storage account.
- Cloud Server (CS): The CS is an entity that provides computing and storage services. In this paper, we assume that the computational power of the CS is not unlimited. The CS provides end users with computing services of partially decryption on the ciphertext and accomplish this computing task on its own or assign it to idle users. The CS is also responsible for checking whether the idle users return correct results.
- End User (EU): End user can add the encrypted data uploaded by the DO into his/her cloud storage account and would like to decrypt the encrypted data. If his/her attribute set satisfies the access structure embedded in the given ciphertext, he/she can decrypt it and obtain the data. Moreover, the EU is allowed to submit a partial decryption key to the CS and let CS help him/her decrypt the ciphertext.

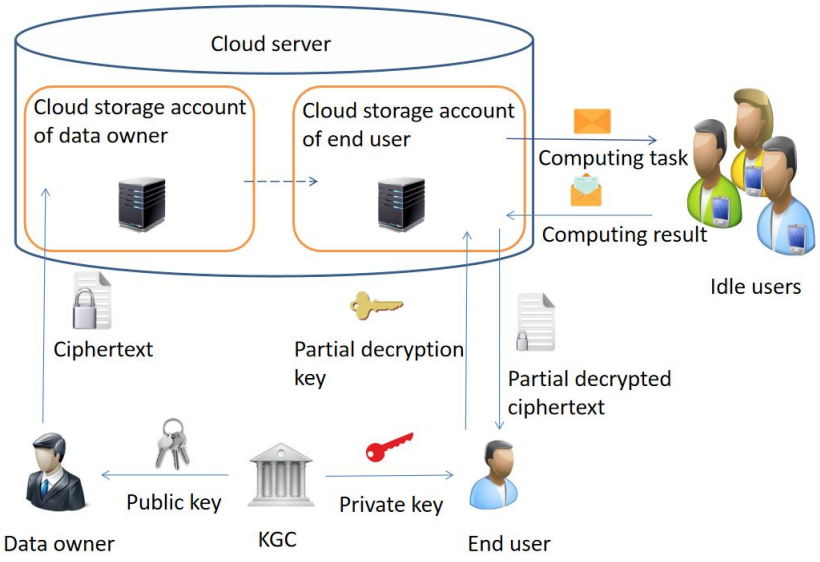

Fig. 1. System description of reverse outsourced CP-ABE scheme.

- Idle User (IU): Idle Users are those who have smart devices connected to the network and not in use. Every idle user can use his/her smart devices to provide some computing resources, accomplish computing assignment for the CS, and earn rewards.

3.2 System Model

The proposed reverse outsourced CP-ABE scheme mainly consists of the following five fundamental algorithms:

- **Setup**$(1^\lambda, \mathcal{L}) \rightarrow (PK, MSK)$: Given security parameter λ and a set of all possible attributes $\mathcal{L}$, the KGC runs this algorithm to generate the public key PK and the master secret key MSK.
- **KeyGen**$(MSK, S) \rightarrow SK$: On input the master secret key MSK and an attribute set S, the KGC runs this algorithm and returns SK to the EU, where the partial decryption key SK_{pd} in contained in SK.
- **Enc**$(PK, \mathbb{A}, M) \rightarrow CT$: On input an access structure $\mathbb{A}$ and the message M, the DO runs this algorithm to generate the ciphertext CT.
- **PreDec**$(CT, SK_{pd}) \rightarrow CT'$: On input CT and SK_{pd}, the CS runs this algorithm to partially decrypt CT and outputs CT', if the attribute set S contained in SK_{pd} satisfies the access structure $\mathbb{A}$ embedded in CT. This algorithm consists of the following two steps:
  1. **Reverse outsource**: The CS divides the partial decryption task into several parts and assigns each to an idle user;
  2. **Verification**: Gathering the results returned by each idle user, the CS checks whether the partial decryption task is accomplished correctly.

- **Dec**$(CT', SK) \rightarrow M$: On input CT' and SK, the EU can decrypt CT and outputs M if his/her attribute set S satisfies the access structure $\mathbb{A}$ embedded CT.

3.3 Threat Model

In this paper, we assume that the KGC is a fully trusted third party, while the CS is an honest-but-curious entity, which exactly follows the protocol specifications but also are curious about the sensitive information of ciphertext. End users are not allowed to collude with CS. Nevertheless, malicious users may collude with each other to access some unauthorized ciphertexts. Idle users are rational and selfish, which means that they may cheat, but they are more willing to get rewards than to save computing resources and they also won't tease the cloud by submitting wrong results after they have calculated the correct results.

3.4 Security Model

Our proposed scheme achieves chosen plaintext security, and the security game between a PPT adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ is as follows.

- *Initialization*: $\mathcal{A}$ chooses and submits a challenge access structure $\mathbb{A}^*$ to its challenger $\mathcal{C}$.
- *Setup*: $\mathcal{C}$ runs **Setup** algorithm and returns the public key PK to $\mathcal{A}$.
- *Phase* 1: $\mathcal{A}$ adaptively submits any attribute set S to $\mathcal{C}$ with the restriction that S doesn't satisfy $\mathbb{A}^*$. In response, $\mathcal{C}$ runs **KeyGen** algorithm and answers $\mathcal{A}$ with the corresponding SK.
- *Challenge*: $\mathcal{A}$ submits two equal-length challenge messages (m_0, m_1) to $\mathcal{C}$. Then $\mathcal{C}$ picks a random bit $\vartheta \in \{0, 1\}$ and runs **Enc** algorithm to generate the challenge ciphertext CT^* of m_ϑ.
- *Phase* 2: This phase is the same as Phase 1.
- *Guess*: $\mathcal{A}$ outputs a guess bit ϑ' of ϑ. We say that $\mathcal{A}$ wins the game if and only if $\vartheta' = \vartheta$. The advantage of $\mathcal{A}$ to win this security game is defined as $Adv(\mathcal{A}) = \left| \Pr[\vartheta' = \vartheta] - \frac{1}{2} \right|$.

Definition 5. *The proposed scheme achieves IND-CPA security if there exist no PPT adversary winning the above security game with a non-negligible advantage ϵ under the DBDH assumption.*

4 Reverse Outsourcing

4.1 Definitions

We found that some computing tasks of the CS could be assigned to idle users to reduce the computing pressure in the cloud, which we called Reverse outsourcing. Idle users are those users with smart devices that are not in use and

connected to the Internet. Each idle user can provide a small amount of computing resources for the cloud, but if millions of idle users are brought together, their total computing power will be considerable. Here, we formally introduce the concept of Reverse outsourcing.

Definition 6 (Reverse Outsourcing). *For relieving the cloud's computing burden, the cloud is allowed to divide a computing task into several sub-tasks and assign each subtask to the corresponding idle user. Each idle user returns a sub-result after completing his/her assignment. The cloud combines all sub-results to get the result of the original computing task, and then checks whether it is valid or not.*

When the CS assigns computing tasks to idle users, they should follow protocol specifications. If the results they returned are valid, they will be rewarded by the CS. In this paper, the Reverse outsourcing is applied to the Rational idle user model, which is defined as follows.

Definition 7 (Rational Idle User Model). *Rational idle user are selfish and lazy, and always attempt to maximize their profits, which means that they prefer to get rewards from the* CS*, rather than save the computational resource of their smart devices. Therefore, for each rational idle user* IU_i*, it holds that* $ut_i^{++} > ut_i^{+} > ut_i^{-} > ut_i^{--}$*, where*

- ut_i^{++} *is the utility of* IU_i *when he/she gets rewards without following the protocol specification.*
- ut_i^{+} *is the utility of* IU_i *when he/she follows the protocol specification and gets rewards.*
- ut_i^{-} *is the utility of* IU_i *when he/she doesn't get rewards without following the protocol specification.*
- ut_i^{--} *is the utility of* IU_i *when he/she follows the protocol specification but doesn't get rewards.*

In the Rational idle user model, any user is independent from each other. Since the performance of IU_i satisfies $ut_i^{++} > ut_i^{+} > ut_i^{-} > ut_i^{--}$, he/she may try to defraud rewards without following the protocol honestly. So each IU_i has two strategies: follow the protocol or not. In order to analyze the best strategy for each IU_i, we formalize the Reverse Outsourcing Game by means of Game theory and introduce the notion of Nash equilibrium.

Definition 8 (Reverse Outsourcing Game). *The reverse outsourcing game is a tuple* $G_{RO} = \{\mathsf{IU}, T, ST, R, V\}$*, where*

- $\mathsf{IU} = \{\mathsf{IU}_1, \mathsf{IU}_2, \ldots, \mathsf{IU}_k\}$ *is the set of* k *rational idle users, where* $k \geq 1$*.*
- T *is a computing task, which can be divided into* k *sub-tasks* $\{T_1, T_2, \ldots, T_k\}$*. Each sub-task* T_i *can be assigned to* IU_i *and the entire task* T *completed by completing each sub-task.*
- $ST = \{st_1, st_2, \ldots, st_k\}$ *is the set of rational idle users' strategies in* G_{RO}*. In particular,* $st_i \in \{st_i^0, st_i^1\}$ *is the set of* IU_i*'s strategies.* st_i^0 *denotes that* IU_i *wants to be rewarded without following the protocol specification;* st_i^1 *denotes that* IU_i *follows the protocol honestly.*

- *R is the computational result of T, which can be obtained by gathering each sub-result R_i corresponding with T_i.*
- *V is a verification algorithm to check whether R is valid or not. If R is valid, every rational idle user will get the same reward. Otherwise, none of them will get reward.*

Definition 9 (Nash Equilibrium of G_{RO}). *For a given strategy $ST^* = (st_1^*, st_2^*, \ldots, st_k^*)$, ST^* is Nash equilibrium for G_{RO}, if and only if for any rational idle user $\mathsf{IU}_i \in \mathsf{IU}$, when the game G_{RO} is finished, for any $st_j \in \{st_j^0, st_j^1\}$, it holds that*

$$ut_i(st_i^* \mid ST^* \setminus \{st_i^*\}) \geq ut_i(st_i \mid ST^* \setminus \{st_i^*\}), \tag{2}$$

where $st_i^* \in \{st_i^0, st_i^1\}$.

4.2 Construction of Reverse Outsourced CP-ABE Scheme

Here, we show how to apply the Reverse outsourcing to an CP-ABE scheme with outsourced decryption.

Without loss of generality, we suppose that there are n possible attributes in total and $\mathcal{L} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$ is the set of all possible attributes. Assume $\mathbb{G}_0, \mathbb{G}_T$ are multiplicative cyclic groups with prime order p and the generator of $\mathbb{G}_0$ is g. Let λ be the security parameter which determines the size of groups. Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ be a bilinear map and $H : \{0,1\}^* \longrightarrow \mathbb{Z}_p$ be a hash function which maps any string to a random element of $\mathbb{Z}_p$. Set the Lagrange coefficient as $\Delta_{i,L}(x) = \prod_{j \in L, j \neq i} \frac{x-j}{i-j}$, where $i \in \mathbb{Z}_p$ and a set, L, of elements in $\mathbb{Z}_p$.

We extract the notion that we called **Lagrange-route product** from [1]. Actually in traditional CP-ABE schemes, the calculation of the lagrange-route product is implied in the decryption process of ciphertexts.

Definition 10 (Lagrange-Route Product). *For an access tree $\mathcal{T}$ and each leaf node y of $\mathcal{T}$, there is only one route from y to the root node R. We define the route as a set $S_{y \to R} = (y_o, y_1, y_2, \ldots, y_{R-1})$, where $y_0 = y$ and y_{i-1} is the child node of y_i, y_R is the root node R. Then, the Lagrange-route product is defined as:*

$$\pi_y = \prod_{x \in S_{y \to R}} \Delta_{i,q_x}(0), \tag{3}$$

where q_x is a polynomial and its definition will be given in the following **Enc** algorithm. The details of our improved scheme are shown as follows.

- **Setup**$(1^\lambda, \mathcal{L}) \to (PK, MSK)$: Given a security parameter λ and all possible attributes $\mathcal{L}$, this algorithm will choose a bilinear group $\mathbb{G}_0$ of prime order p with generator g. Next it will pick random exponents $\alpha, \beta \in \mathbb{Z}_p$, and $v_j \in \mathbb{Z}_p$. The public key is published as: $PK = \{\mathbb{G}_0, g, h, g^\alpha, e(g,g)^\beta, e(g,h)^\beta, \{PK_j = g^{v_j} \mid \forall \mathbf{a}_j \in \mathcal{L}\}\}$ and the master key $MSK = (\alpha, \beta, v_j)$.
- **KeyGen**$(MSK, S) \to SK$: While receiving an attribute set S from the EU, the key generation algorithm is run by $\mathsf{KGC}$to output a key that identifies

with S. The algorithm randomly chooses $r, x', y', z' \in \mathbb{Z}_p$. Then it returns the partial decryption key SK_{pd} as

$$SK_{pd} = \left\{ \begin{array}{l} SK_0 = x' + y', SK_1 = g^{(\beta+\alpha r)x'+z'}, SK_2 = g^{(\beta+\alpha r)y'}, \\ SK_3 = (gh)^{z'}, \left\{ SK_{1,j} = g^{\frac{\alpha r x'}{v_j}}, SK_{2,j} = g^{\frac{\alpha r y'}{v_j}} \cdot h^{\frac{\alpha r (x'+y')}{v_j}} \mid \forall \mathbf{a}_j \in S \right\} \end{array} \right\} \tag{4}$$

and the user secret key $SK = \{x', z', SK_{pd}\}$.

- **Enc**$(PK, M, \mathcal{T}) \rightarrow CT$: The encryption algorithm encrypts a message M under the tree access structure $\mathcal{T}$. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in $\mathcal{T}$. These polynomials are chosen from the root node R in a top-down manner: for each node x of $\mathcal{T}$, the degree of q_x is $d_x = k_x - 1$, where k_x is the threshold value of x; beginning with root node R, it picks a random $s_1 \in \mathbb{Z}_p$, sets $q_R(0) = s_1$ and randomly chooses $d_R = k_R - 1$ other points of q_R to define the polynomial completely; for any other node x, it sets $q_x(0) = q_{\mathsf{parent}(x)}(\mathsf{index}(x))$ and chooses d_x other points to define q_x completely. Let, Y be the set of all leaf nodes in $\mathcal{T}$ and $\mathcal{X}$ be the set of all attributes in $\mathcal{T}$. The algorithm computes $\{\pi_y \mid \forall y \in Y\}$. Then the ciphertext is constructed as $CT = \{\mathcal{T}, \tilde{C} = M \cdot e(g,g)^{\beta s}, C = e(g,gh)^{\beta s}, C' = g^s, C'' = h^s, \{C_y = g^{v_j q_y(0) \pi_y} \mid \forall y \in Y, \mathbf{a}_j = \mathsf{att}(y) \in \mathcal{X}\}\}$.
- **PreDec**$(SK_{pd}, CT) \rightarrow CT'$: At first, for each $i \in [1,2]$, we define the function $F_{i,y}$ as $F_{i,y}(C_y, SK_{i,j}, y) = e(C_y, SK_{i,j})$, intaking C_y, y of CT and $SK_{i,j}$ of SK_{pd}. In $\mathcal{T}$, for each leaf node $y \in Y$, let $\mathbf{a}_j = \mathsf{att}(y) \in \mathcal{X}$. The the CS picks out the minimized set $Y' \subseteq Y$ such that $\{\mathsf{att}(y)\}_{y \in Y'} \subseteq S \bigcap \mathcal{X}$ and $\{\mathsf{att}(y)\}_{y \in Y'}$ satisfies $\mathcal{T}$. If there is no such set, the algorithm aborts. Otherwise, it conducts the following steps:
  1. **Reverse outsource**: The CS sets the computing task $T = \{T_i\}_{i \in [1,2]}$, where each sub-task $T_i = (F_{i,y}, \{C_y, SK_{1,j}\}_{\mathbf{a}_j = \mathsf{att}(y), y \in Y'})$ is assigned to the corresponding idle user IU_i.
     Given assignment as above, the IU_j computes and returns the sub-result R_i to the CS, where
     $$R_i = \prod_{\mathbf{a}_j = \mathsf{att}(y), y \in Y'} F_{i,y}(C_y, SK_{i,j}, y). \tag{5}$$
     Receiving each R_i outputted by IU_i, the CS computes the result $R = R_1 \cdot R_2$ corresponding with the original task T.
  2. **Verification**: The CS checks whether the result R is valid or not as follows:
     (1) The CS interacts CT and SK_{pd} to compute A as follows.
     $$\begin{aligned} A &= e(C' \cdot C'', SK_1) \cdot e(C' \cdot C'', SK_2) \\ &= e(g^s \cdot h^s, g^{(\alpha r+\beta)x'+z'}) \cdot e(g^s h^s, g^{(\alpha r+\beta)y'}) \\ &= e(g, gh)^{(\alpha r+\beta)(x'+y')s+z's}. \end{aligned} \tag{6}$$
     (2) The result R is valid only if the following equation holds,
     $$\frac{A}{R} = C^{SK_0} \cdot e(C', SK_3). \tag{7}$$

For each $i \in [1,2]$, if each IU_i follows the protocol specification and outputs the correct sub-result R_i, which means that

$$\begin{aligned} R_1 &= \prod_{y \in Y'} F_{1,y}(C_y, SK_{1,j}, y) = \prod_{y \in Y'} e(g^{v_j q_y(0) \pi_y}, g^{\frac{\alpha r x'}{v_j}}) \\ &= e(g,g)^{\alpha r x' \sum\limits_{y \in Y'} q_y(0) \prod\limits_{x \in S_{y \to R}} \Delta_{i,q_x}(0)} = e(g,g)^{\alpha r x' s}, \end{aligned} \tag{8}$$

and similarly,

$$\begin{aligned} R_2 &= \prod_{y \in Y'} F_{2,y}(C_y, SK_{1,j}, y) = \prod_{y \in Y'} e(g^{v_j q_y(0) \pi_y}, g^{\frac{\alpha r y'}{v_j}} \cdot h^{\frac{\alpha r (x'+y')}{v_j}}) \\ &= e(g,g)^{\alpha r y' s} \cdot e(g,h)^{\alpha r (x'+y') s} \end{aligned} \tag{9}$$

Then R is valid, since

$$\begin{aligned} \frac{A}{R} &= \frac{e(g,gh)^{(\alpha r+\beta)(x'+y')s+z's}}{e(g,g)^{\alpha r x' s} \cdot e(g,g)^{\alpha r y' s} \cdot e(g,h)^{\alpha r (x'+y') s}} \\ &= e(g,gh)^{\beta s (x'+y')} \cdot e(g,gh)^{z's} \\ &= C^{SK_0} \cdot e(C', SK_3). \end{aligned} \tag{10}$$

3. Finally, the CS submits the partial decrypted $CT' = \{\tilde{C}, C', \frac{R_1}{A_1}\}$ to the EU.

- **Dec**$(CT', SK) \to M$: The EU derives M as

$$\frac{\tilde{C}}{(\frac{R_1}{A_1} \cdot e(C', g^{z'}))^{\frac{1}{x'}}} = \frac{M \cdot e(g,g)^{\beta s}}{(\frac{e(g,g)^{\alpha r x' s} \cdot e(g^s, g^{z'})}{e(g,g)^{(\alpha r+\beta) x' s + z' s}})^{\frac{1}{x'}}} = M. \tag{11}$$

In our reverse outsourced CP-ABE scheme, the probability that IU_i does not follow the protocol but returns the correct sub-result R_i is negligible. If IU_i cheats, for example, by randomly generating an incorrect sub-result, the cheating behavior can be detected by the CS. Then, all participants $\{\mathsf{IU}_i\}_{i \in [1,2]}$ will not get rewarded. Thus, the utility of IU_i for choosing a strategy $st_i \in \{st_i^0, st_i^1\}$ satisfies

$$ut_i(st_i \mid ST \setminus \{st_i\}) = \begin{cases} ut_i^+, & st_i = st_i^1 \text{ and } \forall st_j = st_j^1 \text{ for } st_j \in ST \setminus \{st_i\}; \\ ut_i^-, & st_i = st_i^0 \text{ or } \exists st_j = st_j^0 \text{ for } st_j \in ST \setminus \{st_i\}. \end{cases}$$

The utility of IU_i reaches the maximum when all participants $\{\mathsf{IU}_i\}_{i \in [1,2]}$ follow the protocol specification. According to Nash equilibrium theory, following the protocol honestly and returning the correct sub-result, is the best strategy for each IU_i . Any other strategy will not only decrease the utility of each IU_i but waste his/her computational resources. Therefore, in the Rational idle user model, in order to maximize it's own profits, each idle user has to follow the protocol specification of our reverse outsourced CP-ABE scheme.

5 Security Analysis

In this section, we provide a security analysis of our proposed reverse outsourced CP-ABE scheme.

Theorem 1. *Supposed that a PPT adversary $\mathcal{A}$ can break the IND-CPA security of our proposed scheme with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can distinguish a DBDH tuple from a random tuple with an advantage $\frac{\epsilon}{2}$.*

Proof. Given the bilinear map parameter $(\mathbb{G}_0, \mathbb{G}_T, p, e, g)$. The DBDH challenger $\mathcal{C}$ selects $a', b', c' \in \mathbb{Z}_p$, $\theta \in \{0,1\}$, $\mathcal{R} \in \mathbb{G}_T$ at random. Let $\mathcal{Z} = e(g,g)^{a'b'c'}$, if $\theta = 0$, $\mathcal{R}$ else. Next, $\mathcal{C}$ sends $\mathcal{B}$ the tuple $\langle g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z}\rangle$. Then, $\mathcal{B}$ plays the role of challenger in the following security game.

- *Initialization*:$\mathcal{A}$ submits a challenge access structure $\mathbb{A}^*$ to $\mathcal{B}$.
- *Setup*: $\mathcal{B}$ chooses $\beta', t \in \mathbb{Z}_p$ at random and sets $h = g^t$, $g^\alpha = g^{a'}$, $e(g,g)^\beta = e(g,g)^{\beta'+a'b'} = e(g,g)^{\beta'} e(g^{a'}, g^{b'})$, $e(g,h)^\beta = (e(g,g)^\beta)^t$. For each attribute $a_j \in \mathcal{L}$, $\mathcal{B}$ picks a random $s_j \in Z_p$. If $a_j \in \mathbb{A}^*$, set $PK_j = g^{v_j} = g^{\frac{a'}{s_j}}$; otherwise, $PK_j = g^{v_j} = g^{s_j}$. Then, $\mathcal{B}$ sends $PK = \{\mathbb{G}_0, g, h, g^\alpha, e(g,g)^\beta, e(g,h)^\beta, \{PK_j \mid \forall a_j \in \mathcal{L}\}\}$ to $\mathcal{A}$.
- *Phase* 1: $\mathcal{A}$ adaptively submits any attribute set $S \in \mathcal{L}$ to $\mathcal{B}$ with the restriction that $S \not\models \mathbb{A}^*$. In response, $\mathcal{B}$ picks $r', x', y', z' \in \mathbb{Z}_p$ at random, and computes $g^r = \frac{g^{r'}}{g^{b'}} = g^{r'-b'}$, $SK_0 = x' + y'$, $SK_1 = g^{(\beta+\alpha r)x'+z'} = g^{(\beta'+a'b'+a'(r'-b'))x'+z'} = (g^{\beta'+a'r'})^{x'} \cdot g^{z'}$, $SK_2 = g^{(\beta+\alpha r)y'} = (g^{\beta'+a'r'})^{y'}$, $SK_3 = (gh)^{z'} = g^{(1+t)z'}$. For each $\mathbf{a}_j \in S$, if $\mathbf{a}_j \in \mathbb{A}^*$, $\mathcal{B}$ computes $SK_{1,j} = g^{\frac{\alpha r' x'}{v_j}} = g^{s_j x' r'}$ and $SK_{2,j} = g^{\frac{\alpha r' y'}{v_j}} h^{\frac{\alpha r'(x'+y')}{v_j}} = g^{s_j y' r'} h^{s_j(x'+y')r'}$; otherwise, $SK_{1,j} = g^{\frac{\alpha r' x'}{v_j}} = g^{\frac{a' r' x'}{s_j}}$ and $SK_{2,j} = g^{\frac{\alpha r' y'}{v_j}} h^{\frac{\alpha r'(x'+y')}{v_j}} = g^{\frac{a' r' y'}{s_j}} h^{\frac{a' r'(x'+y')}{s_j}}$. Afterwards, $\mathcal{B}$ answers $\mathcal{A}$ with the corresponding secret key $SK = (x', z', SK_{pd})$, where the partial decryption key $SK_{pd} = \{SK_0, SK_1, SK_2, SK_3, \{SK_{1,j}, SK_{2,j} \mid \forall \mathbf{a}_j \in S\}\}$.
- *Challenge*: $\mathcal{A}$ submits two equal-length challenge messages (M_0, M_1) to $\mathcal{B}$. Then, $\mathcal{B}$ describes the challenge access structure $\mathbb{A}^*$ as an access tree $\mathcal{T}^*$, sets $g^s = g^{c'}, h^s = g^{tc'}$, $e(g,g)^{\beta s} = \mathcal{Z} \cdot e(g,g)^{\beta' c'}$, $e(g,h)^{\beta s} = \mathcal{Z}^t \cdot e(g,g)^{\beta' c' t}$ and generates the challenge ciphertext CT^* in the following top-down manner:
  1. For the root node R of $\mathcal{T}^*$, $\mathcal{B}$ sets $g^{q_R(0)} = g^s$ and $L_R = [1, n_R]$, where n_R is the number of children of R.
     (1) If R is an AND gate, $\mathcal{B}$ randomly chooses $n_R - 1$ elements $r_1, \ldots, r_{n_R-1} \in \mathbb{Z}_p$. For each child nodes x_i of R, $i \in [1, n_R - 1]$, $\mathcal{B}$ sets $g^{q_{x_i}(0)} = g^{q_{\mathsf{parent}(x_i)}(\mathsf{index}(x_i))} = g^{q_R(i)} = g^{r_i}$ and for the n_R'th child node x_{n_R}, it sets $g^{q_{x_{n_R}}(0)} = g^{q_{\mathsf{parent}(x)}(\mathsf{index}(x_{n_R}))} = g^{q_R(n_R)} = (\frac{g^{q_R(0)}}{\prod_{i\in[1,n_R-1]} g^{q_R(i)}})^{\frac{1}{\Delta_{i,L_R}(n_R)}} = (\frac{g^s}{\prod_{i\in[1,n_R-1]} g^{r_i}})^{\frac{1}{\Delta_{i,L_R}(n_R)}}$.
     (2) If R is an OR gate, for $i \in [1, n_R]$, $\mathcal{B}$ sets $g^{q_{x_i}(0)} = g^{q_{\mathsf{parent}(x_i)}(\mathsf{index}(x_i))} = g^{q_R(i)} = g^s$.

2. In a similar way, $\mathcal{B}$ can calculate $g^{q_y(0)}$ and π_y for each leaf node of $\mathcal{T}^*$. Finally, $\mathcal{B}$ randomly picks $\theta' \in \{0,1\}$ and the challenge ciphertext CT^* is constructed as $\Big\{\mathcal{T}^*, \tilde{C}^* = M_{\theta'} \cdot e(g,g)^{\beta s}, C^* = e(g,gh)^{\beta s}, C'^* = g^s, C''^* = h^s, \{C_y^* = g^{v_j q_y(0)\pi_y} \mid \forall y \in Y^*, \mathbf{a}_j = \mathsf{att}(y) \in \mathcal{X}^*\}\Big\}$, where $e(g,g)^{\beta s} = \mathcal{Z} \cdot e(g,g)^{\beta' c'}$ and $e(g,gh)^{\beta s} = e(g,g)^{\beta s(1+t)}$.

- *Phase* 2: This phase is the same as Phase 1.
- *Guess*: $\mathcal{A}$ outputs a guess bit θ'' of θ'. If $\theta'' = \theta'$, $\mathcal{B}$ guesses $\theta = 0$ which indicates that $\mathcal{Z} = e(g,g)^{a'b'c'}$ in the above game. Otherwise, $\mathcal{B}$ guesses $\theta = 1$ i.e., $\mathcal{Z} = \mathcal{R}$.

If $\mathcal{Z} = e(g,g)^{a'b'c'}$, then CT^* is available and $\mathcal{A}'s$ advantage of guessing θ' is ϵ. Therefore, $\mathcal{B}'s$ probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = e(g,g)^{a'b'c'}\right) = 0\right] = \frac{1}{2} + \epsilon. \tag{12}$$

Else $\mathcal{Z} = \mathcal{R}$, then CT^* is random from the view of $\mathcal{A}$. Hence, $\mathcal{B}'s$ probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = \mathcal{R}\right) = 1\right] = \frac{1}{2}. \tag{13}$$

In conclusion, $\mathcal{B}'s$ advantage to win the above security game is

$$\begin{aligned} Adv(\mathcal{B}) = \frac{1}{2}\Big(&\Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = e(g,g)^{a'b'c'}\right) = 0\right] \\ &+ \Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = \mathcal{R}\right) = 1\right]\Big) - \frac{1}{2} = \frac{1}{2}\epsilon. \end{aligned} \tag{14}$$

□

6 Conclusion

In this paper, we propose a heuristic primitive called reverse outsourcing. Specifically, users outsource part of the decryption work to the cloud, which splits it up and dispatches each to different idle users. Idle users are those whos has some smart devices connected to the internet and not in use. It's like, the cloud employs many idle users to accomplish its own computing tasks. Then, we proposed a reverse outsourced CP-ABE scheme which provable secure under the BDBH assumptions.

Acknowledgments

The authors are supported by National Cryptography Development Fund (Grant No. MMJJ20180210) and National Natural Science Foundation of China (Grant No. 61832012 and No. 61672019).

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA. pp. 321–334 (2007)
2. Cheung, L., Newport, C.C.: Provably secure ciphertext policy ABE. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007. pp. 456–465 (2007)
3. 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, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006. pp. 89–98 (2006)
4. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: 20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings (2011)
5. Halpern, J.Y., Teague, V.: Rational secret sharing and multiparty computation: extended abstract. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004. pp. 623–632 (2004)
6. Horváth, M.: Attribute-based encryption optimized for cloud computing. In: SOFSEM 2015: Theory and Practice of Computer Science - 41st International Conference on Current Trends in Theory and Practice of Computer Science, Pec pod Sněžkou, Czech Republic, January 24-29, 2015. Proceedings. pp. 566–577 (2015)
7. Kol, G., Naor, M.: Cryptography and game theory: Designing protocols for exchanging information. In: Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. pp. 320–339 (2008)
8. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. pp. 457–473 (2005)
9. Wang, S., Zhou, J., Liu, J.K., Yu, J., Chen, J., Xie, W.: An efficient file hierarchy attribute-based encryption scheme in cloud computing. IEEE Trans. Information Forensics and Security **11**(6), 1265–1277 (2016)
10. Zhang, R., Ma, H., Lu, Y.: Fine-grained access control system based on fully outsourced attribute-based encryption. Journal of Systems and Software **125**, 344–353 (2017)