Chosen Ciphertext Secure Attribute-Based Encryption with Outsourced Decryption

Cong Zuo, Jun Shao(✉), Guiyi Wei, Mande Xie, and Min Ji

School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, People's Republic of China
zuocong10@gmail.com, chn.junshao@gmail.com, {weigy,xiemd,jimin}@zjgsu.edu.cn

Abstract. Although attribute-based encryption (ABE) is a useful cryptographic tool to realize expressive access policy on ciphertexts, it is not quite suitable for mobile devices. The root cause lies in that the size of the ciphertext and the decryption cost are usually proportional to the complexity of the access policy. To solve this problem, a variant of ABE, named attribute-based encryption with outsourced decryption (OD-ABE), was proposed by Green, Hohenberger and Waters. Especially, OD-ABE allows a proxy with a transformation key delegated from the user to simplify ABE ciphertexts satisfied by the user's attributes. On the other hand, this transformation also makes it tricky to design an OD-ABE scheme achieving the CCA security that is generally considered as the standard notion of security for a cryptosystem. However, the existing OD-ABE schemes only achieve the re-randomizable (replayable) CCA security. In this paper, we propose the CCA security model for OD-ABE and a concrete scheme secure in our proposed security model. We believe that this improvement on the security of OD-ABE will lead to a wider spectrum of applications.

Keywords: Attribute-based encryption · Chosen ciphertext security · Outsourced decryption · Random oracle model

1 Introduction

Cloud storage is becoming the most popular solution for the data sharing due to its cost-effective property. Since the cloud server is not always fully trusted, the fine-grained access control on the encrypted data is quite desired from the viewpoint of users. Attribute-based encryption (ABE), introduced by Sahai and Waters [18], is a promising solution for this requirement. There exist two types of ABE schemes: key-policy ABE (KP-ABE) [7] and ciphertext-policy ABE (CP-ABE) [1,4,11,19]. In the former type, the access policy is embedded into the user's private key. While in the latter one, the access policy is embedded into the ciphertext. Only if the user's attributes satisfy the access policy, the ciphertext can be decrypted by the user's private key.

J.K. Liu and R. Steinfeld (Eds.): ACISP 2016, Part I, LNCS 9722, pp. 495–508, 2016.
DOI: 10.1007/978-3-319-40253-6_30

Although ABE is a very useful cryptographic tool to realize fine-grained access control, the inefficient performance is its Achilles' Heel. In particular, the size of ABE ciphertext and decryption cost are usually proportional to the complexity of the access policy. This impedes the use of ABE in mobile devices due to the limited resources [9]. However, it has been reported that mobile exceeded PC Internet usage in early 2014 [16]. To solve this conflict, Green et al. [8] proposed the concept of attribute-based encryption with outsourced decryption (OD-ABE), where it allows a proxy (such as the cloud) with a transformation key to transform ABE ciphertexts into simple and constant size ciphertexts, while the proxy cannot obtain the corresponding plaintext. By using OD-ABE, the most of the decryption cost on ABE ciphertexts can be moved from the user to the cloud.

It is well-known that the chosen ciphertext security is generally considered as the right security notion for a cryptosystem. However, none of the existing OD-ABE schemes achieve CCA security. The traditional CCA security of public key encryption guarantees that any ciphertext cannot be malleable. Nevertheless, the ciphertext transformation is expected as a regular functionality in OD-ABE, which makes the definition of CCA security tricky. In this paper, we will propose the CCA security for OD-ABE by following the spirit of the traditional CCA security as close as we can. Furthermore, we will propose a concrete CCA secure OD-ABE scheme by applying the CHK [3] and FO [5] techniques on the ABE scheme in [4]. As we can see later, the design approach used in this paper can be applied to other ABE schemes, such as [1,6,11,15,20–22].

1.1 Related Work

The history of OD-ABE is quite simple compared to that of ABE. The concept of OD-ABE is proposed by Green et al. [8]. The notions of CPA security and CCA security, along with several CPA-secure and RCCA-secure OD-ABE scheme, are also proposed in [8]. After that, many research efforts have been dedicated to design OD-ABE schemes [10,12–14,17]. However, none of them aims to promote the security from RCCA security to CCA security.

We note that OD-ABE has the similar situation with proxy re-encryption (PRE) [2], where a proxy with a re-encryption key can also do ciphertext transformation. However, the intending decryptor changes after the ciphertext transformation in PRE, which makes the CCA security of PRE cannot be used in OD-ABE.

1.2 Paper Organization

The remaining paper is organized as follows. In Sect. 2, the definition and CCA security models of attribute-based encryption with outsourced decryption are introduced. In Sect. 3, we present our proposal for CCA secure attribute-based encryption with outsourced decryption. In what follows, we give the security proof of our proposal in our proposed CCA security model. At last, we conclude the paper in Sect. 5.

2 Definitions

2.1 Attribute-Based Encryption with Outsourced Decryption

Definition 1 (Attribute-Based Encryption with Outsourced Decryption (OD-ABE)). *An attribute-based encryption scheme with outsourced decryption OD-ABE is a tuple of probabilistic polynomial time (PPT) algorithms (*$\mathtt{KeyGen}$, $\mathtt{Ext}$, $\mathtt{OKGen}$, $\mathtt{Enc}$, $\mathtt{TDec}$, $\mathtt{Dec}$*).*

- $\mathtt{KeyGen}(\lambda) \rightarrow (mpk, msk)$. *On input a security parameter* λ*, the key generation algorithm* $\mathtt{KeyGen}$ *outputs the master public/private key pair* (mpk, msk) *of the private key generator (PKG). Note that the master public key* mpk *is included in all of the following algorithms implicitly.*
- $\mathtt{Ext}(msk, I_k) \rightarrow d$. *On input the master private key pair* msk *and an attribute set (resp. access structure)* I_k*, the key extraction algorithm* $\mathtt{Ext}$ *outputs a private key* d *corresponding to* I_k*.*
- $\mathtt{OKGen}(d) \rightarrow (tk, rk)$. *On input a private key* d *corresponding to* I_k*, the outsourced decryption key generation algorithm* $\mathtt{OKGen}$ *outputs an outsourced decryption key* ok *containing a transformation/retrieving key pair* (tk, rk) *corresponding to* I_k*.*
- $\mathtt{Enc}(I_e, m) \rightarrow C$. *On input an access structure (resp. attribute set)* I_e *and a message* m *from the message space* $\mathcal{M}$*, the encryption algorithm* $\mathtt{Enc}$ *outputs a ciphertext* C *corresponding to* I_e*.*
- $\mathtt{TDec}(tk, C) \rightarrow C_t$. *On input a transformation key* tk *corresponding to* I_k *and a ciphertext* C *encrypted under* I_e*, the outsourced decryption* $\mathtt{TDec}$ *outputs a transformed ciphertext* C_t *if* $f(I_e, I_k) = 1$[1]*;* $\perp$ *otherwise.*
- $\mathtt{Dec}(dk, C) \rightarrow m$. *There are two cases in this algorithm, since there are two types of ciphertexts: One is from* $\mathtt{Enc}$*, and the other is from* $\mathtt{TDec}$*.*
  - *If the input ciphertext* C *is from* $\mathtt{Enc}$ *encrypted under* I_e*, the decryption key* dk *is the private key* d *corresponding to* I_k*. It outputs* m *if* $f(I_e, I_k) = 1$*; or* $\perp$ *otherwise.*
  - *If the input ciphertext* C *is from* $\mathtt{TDec}$ *encrypted under* I_e*, the decryption key* dk *is the retrieving key* rk *corresponding to* I_k*. It outputs* m *if* $f(I_e, I_k) = 1$*; or* $\perp$ *otherwise.*

Correctness. For any message m in the message space $\mathcal{M}$, $(mpk, msk) \leftarrow \mathtt{KeyGen}(\lambda)$, $d \leftarrow \mathtt{Ext}(msk, I_k)$, $(tk, rk) \leftarrow \mathtt{OKGen}(d)$, the following conditions must hold:

$$\mathtt{Dec}(d, \mathtt{Enc}(I_e, m)) = m, \text{ if } f(I_e, I_k) = 1.$$

and

$$\mathtt{Dec}(rk, \mathtt{TDec}(tk, \mathtt{Enc}(I_e, m))) = m, \text{ if } f(I_e, I_k) = 1.$$

Remark 1 (Differences Between our Definition and that in [8]). The main difference between our definition and that in [8] is that we explicitly extract the key (retrieving key) used to decrypt the transformed ciphertexts from the private

[1] If I_e is satisfied by I_k, we have $f(I_e, I_k) = 1$; otherwise, we have $f(I_e, I_k) = 0$.

key d. It is because that the retrieving key is usually much shorter than the private key, and the user may only store the retrieving key but not the private key in the mobile device for saving storage. This small difference makes the OD-ABE scheme closer to underlying mobile device application. We note that the definition given in [13] has the similar setting.

2.2 Chosen Ciphertext Security for Attribute-Based Encryption with Outsourced Decryption

As we noticed before, in attribute-based encryption with outsourced decryption, there are two formats of ciphertexts. One is the original ciphertext generated from algorithm `Enc`, and the other is the transformed ciphertext computed from algorithm `TDec`. Hence, there are two cases in this definition.

The Challenge Ciphertext is an Original Ciphertext.

***Setup*:** The challenger $\mathcal{C}$ runs $\texttt{KeyGen}(\lambda)$ to get the master public/private key pair (mpk, msk). After that, $\mathcal{C}$ returns mpk to the adversary $\mathcal{A}$, while keeping msk secret.

Phase 1: $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of:

- Private key generation oracle $\mathcal{O}_{sk}$: On input I_k, $\mathcal{C}$ returns the corresponding private key d.
- Transformation key generation $\mathcal{O}_{tk}$: On input I_k and an index $\texttt{i}$, $\mathcal{C}$ returns the corresponding transformation key tk.
- Retrieving key generation $\mathcal{O}_{rk}$: On input I_k and an index $\texttt{i}$, $\mathcal{C}$ returns the corresponding retrieving key rk. Note that if tk and rk are corresponding to the same private key and the corresponding indexes are the same, then tk and rk are considered as a pair generated from `OKGen`.
- Outsourced decryption oracle $\mathcal{O}_{od}$: On input (C_i, I_k) and an index $\texttt{i}$, $\mathcal{C}$ returns $\texttt{TDec}(tk, C_i)$, where C_i is an original ciphertext, and tk is a transformation key corresponding to I_k and index $\texttt{i}$.
- Decryption oracle $\mathcal{O}_{dec}$: On input (C_i, I_k) and the optional input index $\texttt{i}$, $\mathcal{C}$ returns $\texttt{Dec}(dk, C_i)$, where dk is a decryption key corresponding to I_k, and it would be a private key d if the input ciphertext C_i is an original ciphertext, or rk corresponding to I_k and index $\texttt{i}$ if C_i is a transformed ciphertext. Note that if the transformed ciphertext is not generated from $\mathcal{O}_{od}$ or `TDec` with the transformation key tk from $\mathcal{O}_{tk}$, the challenger simply returns $\perp$.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0^*, m_1^* from the message space $\mathcal{M}$, and an attribute set (resp. access structure) I_e^* on which it wishes to challenge. There are two restrictions on I_e^*, (i) For all I_k's queried to $\mathcal{O}_{sk}$, $f(I_e^*, I_k) \neq 1$. (ii) For all I_k's queried to both $\mathcal{O}_{tk}$ and $\mathcal{O}_{rk}$, $f(I_e^*, I_k) \neq 1$. $\mathcal{C}$ picks a random bit $\mathbf{b} \in \{0, 1\}$ and sets $C^* = \texttt{Enc}(I_e^*, m_{\mathbf{b}}^*)$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: Same as Phase 1 but the challenger will output $\perp$ in the following cases, where the adversary can trivially obtain the message m corresponding to C^*.
- $\mathcal{O}_{sk}$: $f(I_e^*, I_k) = 1$.
- $\mathcal{O}_{tk}$: $f(I_e^*, I_k) = 1$ and $(I_k, \mathtt{i})$ has been queried to $\mathcal{O}_{rk}$.
- $\mathcal{O}_{rk}$: $f(I_e^*, I_k) = 1$ and $(I_k, \mathtt{i})$ has been queried to $\mathcal{O}_{tk}$.
- $\mathcal{O}_{od}$: $C_i = C^*$, $f(I_e^*, I_k) = 1$, and $(I_k, \mathtt{i})$ has been queried to $\mathcal{O}_{rk}$.
- $\mathcal{O}_{dec}$: There are the following cases.
  - $C_i = C^*$ and $f(I_e^*, I_k) = 1$.
  - $C_i \leftarrow \mathtt{TDec}(tk, C^*)$ and $f(I_e^*, I_k) = 1$, where $tk \leftarrow \mathcal{O}_{tk}(I_k, \mathtt{i})$.
  - $C_i \leftarrow \mathcal{O}_{od}(C^*, I_k, \mathtt{i})$ and $f(I_e^*, I_k) = 1$.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $\mathbf{b}' \in \{0, 1\}$ and wins the game if $\mathbf{b} = \mathbf{b}'$.

The advantage $\mathbf{Adv}^{\mathtt{CCA\text{-}O}}_{\mathtt{OD\text{-}ABE}}(\lambda)$ is defined as $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$. The scheme `OD-ABE` is said to be *CCA-O* secure[2] if all efficient adversaries $\mathcal{A}$ specified as above, the advantage $\mathbf{Adv}^{\mathtt{CCA\text{-}O}}_{\mathtt{OD\text{-}ABE}}(\lambda)$ is negligible.

The Challenge Ciphertext is a Transformed Ciphertext.

Phase 1: Identical to that in the challenge original ciphertext case.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0^*, m_1^* from the message space, an attribute set (resp. access structure) I_e^*, an access structure (resp. attribute set) I_k^*, and an index $\mathtt{i}^*$. There is one restriction on $(I_k^*, \mathtt{i}^*)$: $(I_k^*, \mathtt{i}^*)$ has never been queried to $\mathcal{O}_{rk}$. $\mathcal{C}$ picks a random bit $\mathbf{b} \in \{0, 1\}$ and sets $C^* = \mathtt{TDec}(tk^*, \mathtt{Enc}(I_e^*, m_{\mathbf{b}}^*))$, where tk^* is the output of $\mathcal{O}_{tk}$ with input $(I_k^*, \mathtt{i}^*)$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: Almost the same as that in Phase 1 but with the following constraints.
- $\mathcal{O}_{rk}$: $(I_k, \mathtt{i}) = (I_k^*, \mathtt{i}^*)$.
- $\mathcal{O}_{dec}$: $C_i = C^*$ and $(I_k, \mathtt{i}) = (I_k^*, \mathtt{i}^*)$.

Guess: Identical to that in the challenge original ciphertext case.

The advantage $\mathbf{Adv}^{\mathtt{CCA\text{-}T}}_{\mathtt{OD\text{-}ABE}}(\lambda)$ is defined as $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$. The scheme `OD-ABE` is said to be *CCA-T* secure[3] if all efficient adversaries $\mathcal{A}$ specified as above, the advantage $\mathbf{Adv}^{\mathtt{CCA\text{-}T}}_{\mathtt{OD\text{-}ABE}}(\lambda)$ is negligible.

Remark 2 (Difference Between our Security Model and that in [8]). Compared to the RCCA security model in [8], the proposed security model has the following differences.

- We allow the adversary to query the decryption oracle with the ciphertext whose corresponding plaintext belongs to the challenge messages $\{m_0^*, m_1^*\}$, while we only disallow the adversary to issue the queries that can help it to win the game trivially. This difference makes our definition closer to the traditional CCA security definition.

[2] O stands for original ciphertexts.
[3] T stands for transformed ciphertexts.

- There two different security games in our definition corresponding to the two different formats of ciphertexts, which makes our definition closer to the nature of attribute-based encryption with outsourced decryption.
- As mentioned in Remark 1, the user may only store the retrieving key rk but not the private key d in the mobile device for some reasons (such as saving storage). In this case, the retrieving key might be obtained by the adversary for some incidents. For example, the mobile device is stolen or infected by computer virus. This kind of attack is captured by $\mathcal{O}_{rk}$, which makes the OD-ABE schemes proven secure in our security model have a wider spectrum of applications.

Due to the above differences, the scheme proposed in [8] cannot be proven secure in our security model.

Remark 3 (CPA Security and Selective Security). We can define CPA security as CCA security. In particular, we can obtain the CPA security game only if we remove the decryption oracle in the CCA security game.

Furthermore, if the adversary makes a decision on the challenge value I_e^* (and challenge index $\mathtt{i}^*$ for CCA-T security) before the **Setup** stage, we have the selective security.

2.3 Bilinear Groups

We say that a (multiplicative) cyclic group $\mathbb{G}$ with a prime order q is a bilinear group if it satisfies the following properties.

- The group action in $\mathbb{G}$ can be computed efficiently.
- There exists an admissible bilinear map $\hat{e}$ as follows.
  - $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}_T$ is a (multiplicative) cyclic group with prime order q.
  - For all $a, b \in \mathbb{Z}_q$, the generator g of $\mathbb{G}$, $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$.

We denote $\mathtt{BSetup}$ as an algorithm that, on input the security parameter λ, outputs the parameters for a bilinear map as $(q, g, \mathbb{G}, \mathbb{G}_T, \hat{e})$, where $q \in \Theta(2^\lambda)$.

2.4 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Let $(q, g, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \mathtt{BSetup}(\lambda)$, and a, b, c, z be random elements from $\mathbb{Z}_q^*$. The decisional Bilinear Diffie-Hellman (DBDH) problem is to distinguish between the tuples of the form $(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc})$ and $(g, g^a, g^b, g^c, \hat{e}(g, g)^z)$. An algorithm $\mathcal{A}$ has advantage ϵ in solving DBDH problem if

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^z) = 1]| \geq \epsilon.$$

We say the DBDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$ if ϵ is negligible.

2.5 Modified Decisional Diffie-Hellman (MDDH) Assumption

Let $(q, g, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \texttt{BSetup}(\lambda)$, and a, b, c be random elements from $\mathbb{Z}_q^*$. The modified decisional Diffie-Hellman (MDDH) problem is to distinguish between the tuples of the form $(g, g^a, \hat{e}(g,g)^b, \hat{e}(g,g)^{ab})$ and $(g, g^a, \hat{e}(g,g)^b, \hat{e}(g,g)^c)$. An algorithm $\mathcal{A}$ has advantage ϵ in solving MDDH problem if

$$|\Pr[\mathcal{A}(g, g^a, \hat{e}(g,g)^b, \hat{e}(g,g)^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, \hat{e}(g,g)^b, \hat{e}(g,g)^c) = 1]| \geq \epsilon.$$

We say the MDDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$ if ϵ is negligible.

3 Our Proposal

Before giving our new OD-ABE scheme, we would like to present some intuitions of our construction. The main idea to realize CCA security for non-transformable public key encryption is to allow the decryptor to have the ability to check the validity of the ciphertext. In the OD-ABE scheme, the situation is a little bit more complex than the non-transformable public key encryption. It additionally needs two other decryptions. One is the ciphertext's transformation by the proxy, and the other is the transformed ciphertext's decryption by the decryptor. For the verifiability of the former, we will use the CHK transformation [3], which supports the public verifiability. Hence, the proxy can check the validity before doing transformation. For the latter one, the FO transformation [5] is applied instead of the CHK transformation. Although the ciphertext has been transformed, the decryptor can still check the validity of the transformed ciphertext by using the FO technique [5]. Combining the above ideas, we have our CCA-secure OD-ABE scheme. Our proposal is based on the ABE scheme due to Cheung and Newport [4], while the ideas used in this paper can be also applied to other ABE schemes, such as [1,6,11,15,20–22].

- `KeyGen`: We let the set of attributes be $N = \{1, \ldots, n\}$ for some nature number n. On input a security parameter λ, the PKG runs $\texttt{BSetup}(\lambda)$ to obtain $(q, g, \hat{e}, \mathbb{G}, \mathbb{G}_T)$. It selects random elements $y, t_1, \cdots, t_{3n}$ from $\mathbb{Z}_q$. Let $Y := \hat{e}(g,g)^y$ and $T_i = g^{t_i}$ for each $i \in \{1, \cdots, 3n\}$. It also selects hash functions $H_1 : \{0,1\}^{2\ell} \to \mathbb{Z}_q$, $H_2 : \mathbb{G}_T \to \{0,1\}^\ell$, $H_3 : \{0,1\}^\ell \to \{0,1\}^\ell$, and $H_4 : \{0,1\}^* \to \mathbb{G}$, where ℓ is the bit length of messages. The resulting master public key is

$$mpk := (\hat{e}, g, q, \mathbb{G}, \mathbb{G}_T, Y, T_1, \cdots, T_{3n}, H_1, H_2, H_3, H_4),$$

  and the corresponding master secret key is

$$msk := (y, t_1, \cdots, t_{3n}).$$

- `Ext`: On input the master private key msk and an attribute set $S \subseteq \{1, \cdots, n\}$, the PKG selects random r_i from $\mathbb{Z}_q$ for $i \in [1, n]$ and sets $r := \sum_{i=1}^{n} r_i \bmod q$, and $\hat{D} = g^{y-r}$. It also computes

$$D_i = \begin{cases} g^{r_i/t_i}, & \text{if } i \in S \\ g^{r_i/t_{n+i}}, & \text{if } i \notin S \end{cases} \quad \text{and} \quad F_i = g^{r_i/t_{2n+i}},\ i \in [1, n].$$

The resulting private key corresponding to the attribute set S is

$$d = (\hat{D}, \{(D_i, F_i)|i \in [1, n]\}).$$

– `OKGen`: On input a private key $d = (\hat{D}, \{(D_i, F_i)|i \in [1, n]\})$ corresponding to S, the user selects a random element z from $\mathbb{Z}_q$. The resulting transformation key is

$$tk = (\hat{D}^z, \{(D_i^z, F_i^z)|i \in [1, n]\})$$

and the retrieving key

$$rk = z.$$

– `Enc`: On input an `AND` gate $W = \bigwedge_{i \in I} \underline{i}$, where $I \subseteq \{1, \cdots, n\}$, and $\underline{i}$ is the attribute i or its negation $\neg i$, the encryptor selects a random μ from $\{0,1\}^\ell$ and computes $s = H_1(\mu||m)$, $\tilde{C}_1 = \mu \oplus H_2(Y^s)$, $\tilde{C}_2 = m \oplus H_3(\mu)$, $\hat{C}_1 = g^s$, and

$$C_i = \begin{cases} T_i^s, & \text{if } i \in I \wedge \underline{i} = i \\ T_{n+i}^s, & \text{if } i \in I \wedge \underline{i} = \neg i \\ T_{2n+i}^s, & \text{if } i \in \{1, \cdots, n\} \setminus I \end{cases}$$

At last, the encryptor computes $\hat{C}_2 = H_4(W||\tilde{C}_1||\tilde{C}_2||\hat{C}_1||C_1|| \cdots ||C_n)^s$. The resulting ciphertext is $CT = (W, \tilde{C}_1, \tilde{C}_2, \hat{C}_1, \hat{C}_2, \{C_i|i \in \{1, \cdots, n\}\})$.

– `TDec`: On input a transformation key $tk = (\hat{D}^z, \{(D_i^z, F_i^z)|i \in [1, n]\})$ corresponding to S and a ciphertext C encrypted under $W = \bigwedge_{i \in I} \underline{i}$, the proxy first checks $\hat{e}(\hat{C}_1, H_4(W||\tilde{C}_1||\tilde{C}_2||\hat{C}_1||C_1|| \cdots ||C_n)) \stackrel{?}{=} \hat{e}(g, \hat{C}_2)$, and $\hat{e}(\hat{C}_1, T_i) \stackrel{?}{=} \hat{e}(g, C_i)$ for $i \in I$ and $\underline{i} = i$, $\hat{e}(\hat{C}_1, T_{n+i}) \stackrel{?}{=} \hat{e}(g, C_i)$ for $i \in I$ and $\underline{i} = \neg i$, and $\hat{e}(\hat{C}_1, T_{2n+i}) \stackrel{?}{=} \hat{e}(g, C_i)$ for $i \in \{1, \cdots, n\} \setminus I$. If one of them does not hold, the proxy aborts the algorithm; otherwise, it computes $\tilde{C}_3 = \hat{e}(\hat{C}_1, \hat{D}^z) \cdot \prod_{i \in I} \hat{e}(C_i, D_i^z) \cdot \prod_{i \in \{1^{\cdots}, n\} \setminus I} \hat{e}(C_i, F_i^z)$. The resulting transformed ciphertext is $CT' = (\tilde{C}_1, \tilde{C}_2, \tilde{C}_3)$.

– `Dec`: There are two cases in this algorithm, since there are two types of ciphertexts: One is from `Enc`, and the other is from `TDec`.
  - If the input ciphertext CT is from `Enc` encrypted under $W = \bigwedge_{i \in I} \underline{i}$, the decryption key dk is the private key $d = (\hat{D}, \{(D_i, F_i)|i \in [1, n]\})$ corresponding to S. The decryptor first checks the validity of the ciphertext as in `TDec`. If it does not pass, the decryptor aborts the algorithm; otherwise, it computes $\tilde{C}_3 = \hat{e}(\hat{C}_1, \hat{D}) \cdot \prod_{i \in I} \hat{e}(C_i, D_i) \cdot \prod_{i \in \{1^{\cdots}, n\} \setminus I} \hat{e}(C_i, F_i)$. Note that if S satisfies W, we have that $\tilde{C}_3 = e(g, g)^{y \cdot s}$. The decryptor can obtain m by computing $\mu = \tilde{C}_1 \oplus H_2(\tilde{C}_3)$ and $m = \tilde{C}_2 \oplus H_3(\mu)$. If $\tilde{C}_1 = \mu \oplus H_2(Y^{H_1(\mu||m)})$ holds, then the decryptor outputs m; otherwise, it outputs $\perp$.
  - If the input ciphertext CT' is from `TDec` encrypted under W, the decryption key dk is the retrieving key $rk = z$ corresponding to S. Note that if S satisfies W, we have that $\tilde{C}_3 = e(g, g)^{y \cdot s \cdot z}$. The decryptor computes m by computing $\mu = \tilde{C}_1 \oplus H_2(\tilde{C}_3^{1/z})$ and $m = \tilde{C}_2 \oplus H_3(\mu)$. If both of $\tilde{C}_1 = \mu \oplus H_2(Y^{H_1(\mu||m)})$ and $Y^{H_1(\mu||m)} = \tilde{C}_3^{1/z}$ hold, then the decryptor outputs m; otherwise, it outputs $\perp$.

It is easy to see that with the help of outsourced decryption, the decryption cost is reduced from $3n + 3$ pairings and 1 exponentiation in $\mathbb{G}$ to only 2 exponentiations in $\mathbb{G}$.

Correctness. The correctness of the OD-ABE scheme can be easily obtained by the correctness of the scheme in [4] and the scheme in [8]. Hence, we omit it here.

4 Security Analysis

Now we will show that our OD-ABE scheme is CCA-secure in the sense of our proposed CCA security.

Theorem 1. *The proposed OD-ABE scheme is selective CCA-O secure in the random oracle model if the DBDH assumption holds in* $\mathbb{G}$ *and* $\mathbb{G}_T$.

Proof. We show that if there exists an adversary $\mathcal{A}$ that can break the selective CCA-O security of the proposed OD-ABE scheme, we can build an algorithm $\mathcal{B}$ that can solve the DBDH problem, i.e., given the tuple $(g, \mathbf{A} = g^a, \mathbf{B} = g^b, \mathbf{C} = g^c, \mathbf{Z}) \in \mathbb{G}^4 \times \mathbb{G}_T$, it decides whether $\mathbf{Z} = \hat{e}(g,g)^{abc}$. In particular, $\mathcal{B}$ will act as a challenger with $\mathcal{A}$ to play the following selective CCA-O security game.

Init: $\mathcal{A}$ sends the challenge gate $W^* = \bigwedge_{i \in I^*} \underline{i}$ to $\mathcal{B}$.

Setup: $\mathcal{B}$ sets $Y = \hat{e}(\mathbf{A}, \mathbf{B}) = \hat{e}(g,g)^{ab}$, and chooses random $\alpha_i, \beta_i, \gamma_i$ from $\mathbb{Z}_q$ for each $i \in [1, n]$. The public elements T_i, T_{n+i} and T_{2n+i} are computed as follows.

$$T_i = \begin{cases} g^{\alpha_i}, & \text{if } i \in I^* \wedge \underline{i} = i, \\ B^{\alpha_i}, & \text{otherwise;} \end{cases} \quad T_{n+i} = \begin{cases} g^{\beta_i}, & \text{if } i \in I^* \wedge \underline{i} = \neg i, \\ B^{\beta_i}, & \text{otherwise;} \end{cases}$$

and

$$T_{2n+i} = \begin{cases} g^{\gamma_i}, & \text{if } i \notin I^*, \\ B^{\gamma_i}, & \text{otherwise.} \end{cases}$$

It also builds the following hash oracles.

- H_1: On input R from $\{0,1\}^{2\ell}$, $\mathcal{B}$ first finds the pair (R, h_1) in the list L_1. If it exists, it simply returns h_1 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a random element from $\mathbb{Z}_q$ and sets as h_1. After that, $\mathcal{B}$ records (R, h_1) in the list L_1 and returns h_1 to $\mathcal{A}$.
- H_2: On input R from $\mathbb{G}_T$, $\mathcal{B}$ first finds the pair (R, h_2) in the list L_2. If it exists, it simply returns h_2 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a random element from $\{0,1\}^{\ell}$ and sets as h_2. After that, $\mathcal{B}$ records (R, h_2) in the list L_2 and returns h_2 to $\mathcal{A}$.
- H_3: On input R from $\{0,1\}^{\ell}$, $\mathcal{B}$ first finds the pair (R, h_3) in the list L_3. If it exists, it simply returns h_3 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a random element from $\{0,1\}^{\ell}$ and sets as h_3. After that, $\mathcal{B}$ records (R, h_1) in the list L_3 and returns h_3 to $\mathcal{A}$.

– H_4: On input R from $\{0,1\}^*$, $\mathcal{B}$ first finds the pair (R, h_4) in the list L_4. If it exists, it simply returns g^{h_4} to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a random element from $\mathbb{Z}_q$ and sets as h_4. After that, $\mathcal{B}$ records (R, h_4) in the list L_4 and returns g^{h_4} to $\mathcal{A}$.

Other elements in the public key are generated as in the real execution.

Phase 1: $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of:

– $\mathcal{O}_{sk}$: $\mathcal{A}$ submits an attribute set S that does not satisfy W^*. There must exist $j \in I^*$ such that: either $j \in S$ and $\underline{j} = \neg j$, or $j \notin S$ and $\underline{j} = j$. $\mathcal{B}$ selects such a j. As in [4], we assume that $j \notin S$ and $\underline{j} = j$. $\mathcal{B}$ selects random r'_i from $\mathbb{Z}_q$ for every $i \in [1, n]$, and implicitly sets $r_j := ab + r'_j \cdot b \bmod q$ for each $i \neq j$, $r_i := r'_i \cdot b \bmod q$ for $i = j$, and $r := \sum_{i=1}^{n} r_i \bmod q = ab + \sum_{i=1}^{n} r'_i \cdot b \bmod q$. Hence, we can compute $\hat{D} = \prod_{i=1}^{n} 1/B^{r'_i} = g^{ab-r}$,

$$D_i = \begin{cases} A^{1/\beta_j} \cdot g^{r'_j/\beta_j}, & i = j \\ B^{r'_i/\alpha_i}, & (i \neq j) \wedge (i \in S \wedge i \in I \wedge \underline{i} = i) \\ g^{r'_i/\alpha_i}, & (i \neq j) \wedge ((i \in S \wedge i \in I \wedge \underline{i} = \neg i) \vee (i \in S \wedge i \notin I)) \\ B^{r'_i/\alpha_i}, & (i \neq j) \wedge (i \notin S \wedge i \in I \wedge \underline{i} = \neg i) \\ g^{r'_i/\alpha_i}, & (i \neq j) \wedge ((i \notin S \wedge i \in I \wedge \underline{i} = i) \vee (i \notin S \wedge i \notin I)) \end{cases}$$

and

$$F_i = \begin{cases} A^{1/\gamma_j} \cdot g^{r'_j/\gamma_j}, & i = j \\ g^{r'_i/\gamma_i}, & (i \neq j) \wedge (i \in I) \\ B^{r'_i/\gamma_i}, & (i \neq j) \wedge (i \notin I) \end{cases}$$

The validity of $(\hat{D}, \{(D_i, F_i) | i \in [1, n]\})$ can be verified as that in [4].

– $\mathcal{O}_{tk}$: On input an attribute set S and an index $\mathtt{i}$, $\mathcal{B}$ first checks if $(\mathtt{i}, ok, S)$ in the list L_{ok}. If it exists, $\mathcal{B}$ returns the corresponding tk. Otherwise, $\mathcal{B}$ does the following steps. If S does not satisfy W^*, then $\mathcal{B}$ can obtain the corresponding private key d by querying $\mathcal{O}_{sk}$ with S, and the corresponding outsourced decryption key $ok = (tk, rk) = ((\hat{D}^z, \{(D_i^z, F_i^z) | i \in [1, n]\}), z)$ by running $\mathtt{OKGen}(d)$. If S satisfies W^*, then $\mathcal{B}$ chooses random elements z from $\mathbb{Z}_q$, $R, \{(R_i, R'_i) | i \in [1, n]\}$ from $\mathbb{G}$, and sets the outsourced decryption key $ok = (tk, rk) := ((R^z, \{(R_i^z, R_i'^z) | i \in [1, n]\}), z)$. At last, $\mathcal{B}$ records $(\mathtt{i}, ok, S)$ into the list L_{ok}, and returns tk to $\mathcal{A}$.

– $\mathcal{O}_{rk}$: It is almost the same as $\mathcal{O}_{tk}$. The only difference is that the returned value is the retrieving key rk.

– $\mathcal{O}_{od}$: On input $(C_i, S, \mathtt{i})$, $\mathcal{B}$ first checks whether S satisfies W embedded in C_i. If it does not, $\mathcal{B}$ outputs $\perp$; otherwise, it returns $\mathtt{TDec}(tk, C_i)$, where tk is the transformation key from $\mathcal{O}_{tk}$ with input $(S, \mathtt{i})$.

– $\mathcal{O}_{dec}$: On input $(C_i, S, \mathtt{i})$, $\mathcal{B}$ first checks whether S satisfies W embedded in C_i. If it does not, $\mathcal{B}$ outputs $\perp$; otherwise, $\mathcal{B}$ does the following steps.

- If the ciphertext is an original ciphertext, $\mathcal{B}$ first checks the validity of the ciphertext as the real execution. If it does not pass, it outputs $\perp$ and aborts. If it passes, it finds the pairs (R_1, h_1), (R_2, h_2), (R_3, h_3) in lists L_1, L_2, L_3, respectively. These pairs should satisfy $R_1 = \mu || m$,

$\tilde{C}_1 = \mu \oplus h_2$, $R_2 = Y^{h_1}$, $\tilde{C}_2 = m \oplus h_3$, and $R_3 = \mu$. If there does not exist such pairs, then $\mathcal{B}$ outputs $\perp$; otherwise, it outputs m.

- If the ciphertext is a transformed ciphertext, $\mathcal{B}$ finds $(\mathtt{i}, ok, S)$ in the list L_{ok}. If it does not exist, it outputs $\perp$. Otherwise, it finds the pairs (R_1, h_1), (R_2, h_2), (R_3, h_3) in lists L_1, L_2, L_3, respectively. These pairs should satisfy $R_1 = \mu || m$, $\tilde{C}_1 = \mu \oplus h_2$, $R_2 = Y^{h_1}$, $\tilde{C}_2 = m \oplus h_3$, and $R_3 = \mu$. If there does not exist such pairs, then $\mathcal{B}$ outputs $\perp$. Otherwise, it checks whether $\tilde{C}_3 = (\hat{e}(g, R) \cdot \prod_{i \in I} \hat{e}(g, R_i) \cdot \prod_{i \in \{1,\cdots,n\} \setminus I} \hat{e}(g, R'_i))^{z \cdot h_1}$, where R, R_i, R'_i are used to generate the corresponding transformation key recorded in the list L_{ok}, and $W = \bigwedge_{i \in I} \underline{i}$. If it does not, $\mathcal{B}$ outputs $\perp$; otherwise, it outputs m.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0^*, m_1^* from the message space $\{0,1\}^\ell$, $\mathcal{B}$ sets $\tilde{C}_1 = \mu \oplus H_2(\mathbf{Z})$, $\tilde{C}_2 = m_{\mathbf{b}} \oplus H_3(\mu)$, $\hat{C}_1 = \mathbf{C}$, $\hat{C}_2 = \mathbf{C}^{h_4}$, $\{\mathbf{C}^{\alpha_i} | i \in I^* \wedge \underline{i} = i\}$, $\{\mathbf{C}^{\beta_i} | i \in I^* \wedge \underline{i} = \neg i\}$, and $\{\mathbf{C}^{\gamma_i} | i \in \{1, \cdots, n\} \setminus I^*\}$, where $\mathbf{b}$ is randomly chosen from $\{0,1\}$, μ is a random element from $\{0,1\}^\ell$, and h_4 is the corresponding value in the list L_4. According to the analysis in [4] and [5], the resulting challenge ciphertext is valid if $\mathbf{Z} = \hat{e}(g,g)^{abc}$.

Phase 2: Almost the same as Phase 1 but with the specified restrictions.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $\mathbf{b}' \in \{0,1\}$ on $\mathbf{b}$. If $\mathbf{b} = \mathbf{b}'$, $\mathcal{B}$ decides $\hat{e}(g,g)^{abc} = \mathbf{Z}$; otherwise, it decides $\hat{e}(g,g)^{abc} \neq \mathbf{Z}$.

According to the analysis in [5], the above simulation will abort with a negligible probability. Hence, we obtain the theorem. □

Theorem 2. *The proposed OD-ABE scheme is selective CCA-T secure in the random oracle model if the MDDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$.*

Proof. We show that if there exists an adversary $\mathcal{A}$ that can break the selective CCA-T security of the proposed OD-ABE scheme, we can build an algorithm $\mathcal{B}$ that can solve the MDDH problem, i.e., given the tuple $(g, \mathbf{A} = g^a, \mathbf{B} = \hat{e}(g,g)^b, \mathbf{Z}) \in \mathbb{G}^2 \times \mathbb{G}_T^2$, it decides whether $\mathbf{Z} = \hat{e}(g,g)^{ab}$. In particular, $\mathcal{B}$ will act as a challenger with A to play the following selective CCA-T security game.

Init: $\mathcal{A}$ sends the challenge gate $W^* = \bigwedge_{i \in I^*} \underline{i}$ and challenge index $\mathtt{i}^*$ to $\mathcal{B}$.

Setup: $\mathcal{B}$ selects random $y, t_1, \cdots, t_{3n}$ from $\mathbb{Z}_q$, and computes the public elements as follows: $Y = \hat{e}(g,g)^y$ and $T_i = g^{t_i}$ for every $i \in [1, 3n]$. It also builds hash oracles for H_1, H_2, and H_3 as that in the proof of Theorem 1, while it chooses H_4 as the real execution. It is easy to see that the resulting master secret key is $msk = (y, t_1, \cdots, t_{3n})$ that is known to $\mathcal{B}$.

Phase 1: $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of:

- $\mathcal{O}_{sk}$: $\mathcal{A}$ submits an attribute set S, $\mathcal{B}$ runs $\mathtt{Ext}(msk, S)$ to obtain d and returns it to $\mathcal{A}$.
- $\mathcal{O}_{tk}$: On input an attribute set S and an index $\mathtt{i}$, $\mathcal{B}$ first finds $(\mathtt{i}, ok, S)$ in the list L_{ok}. If it exists, $\mathcal{B}$ returns the corresponding tk and aborts. Otherwise, if $\mathtt{i} \neq \mathtt{i}^*$ or S does not satisfy W^*, $\mathcal{B}$ obtains the corresponding private key d by querying $\mathcal{O}_{sk}$ with S, and the corresponding outsourced

decryption key $ok = (tk, rk) = ((\hat{D}^z, \{(D_i^z, F_i^z)|i \in [1,n]\}), z)$ by running $\texttt{OKGen}(d)$; if $\texttt{i} = \texttt{i}^*$ and S satisfies W^*, $\mathcal{B}$ obtains the corresponding private key d by querying $\mathcal{O}_{sk}$ with S, and the corresponding outsourced decryption key $ok = (tk, rk) = ((\mathbf{A}^{y-r}, \{\mathbf{A}^{r_i/t_i}|i \in S\}, \{\mathbf{A}^{r_i/t_{n+i}}|i \notin S\}, \{\mathbf{A}^{r_i/t_{2n+i}}|i \in [1,n]\}), \blacksquare)$, where $r = \sum_{i=1}^{n} r_i, \{r_i \in Z_q|i \in [1,n]\}$ are the random elements used to generate the corresponding private key d, and $\blacksquare$ means that the value of rk is unknown. Note that, in the case of that $\texttt{i} = \texttt{i}^*$ and S satisfies W^*, $\mathcal{B}$ implicitly sets $z = a$ that is unknown. At last, $\mathcal{B}$ records $(\texttt{i}, ok, S)$ into the list L_{ok} and returns tk to $\mathcal{A}$.

– $\mathcal{O}_{rk}$: It is almost the same as $\mathcal{O}_{tk}$. The only difference is that the returned value is the retrieving key rk.

– $\mathcal{O}_{od}$: On input $(C_i, S, \texttt{i})$, $\mathcal{B}$ first checks whether S satisfies W corresponding to C_i. If it does not, $\mathcal{B}$ outputs $\perp$; otherwise, it returns $\texttt{TDec}(tk, C_i)$, where tk is the transformation key from $\mathcal{O}_{tk}$ with input $(S, \texttt{i})$.

– $\mathcal{O}_{dec}$: On input $(C_i, S, \texttt{i})$, there are two cases.

- If the ciphertext is an original ciphertext, $\mathcal{B}$ runs $d \leftarrow \texttt{KeyGen}(msk, S)$ and $m \leftarrow \texttt{Dec}(d, C_i)$, and returns m to $\mathcal{A}$.
- If the ciphertext is a transformed ciphertext, $\mathcal{B}$ finds $(\texttt{i}, ok, S)$ in the list L_{ok}. If it does not exist, it outputs $\perp$ and aborts. Otherwise, $\mathcal{B}$ does the following steps.
  - *· If $\texttt{i} \neq \texttt{i}^*$ or S does not satisfy W^*, it uses the corresponding rk to obtain the message m and returns it to $\mathcal{A}$.
  - *· If $\texttt{i} \neq \texttt{i}^*$ and S satisfies W^*, it finds the pairs (R_1, h_1), (R_2, h_2), (R_3, h_3) in lists L_1, L_2, L_3, respectively. These pairs should satisfy $R_1 = \mu||m$, $\tilde{C}_1 = \mu \oplus h_2$, $R_2 = Y^{h_1}$, $\tilde{C}_1 = m \oplus h_3$, and $R_3 = \mu$. If there does not exist such pairs, then $\mathcal{B}$ outputs $\perp$. Otherwise, it checks whether $\tilde{C}_3 = \hat{e}(A, g)^{y \cdot h_1}$. If it does not, $\mathcal{B}$ outputs $\perp$; otherwise, it outputs m.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0^*, m_1^* from the message space $\{0,1\}^\ell$, $\mathcal{B}$ sets $\tilde{C}_1 = \mu \oplus H_2(\mathbf{B}^y)$, $\tilde{C}_2 = m_{\mathbf{b}} \oplus H_3(\mu)$, and $\tilde{C}_3 = \mathbf{Z}^y$, where μ is chosen randomly from $\{0,1\}^*$. According to the analysis in [5], if $\mathbf{Z} = \hat{e}(g,g)^{ab}$, then the resulting challenge transformed ciphertext is valid.

Phase 2: Almost the same as Phase 1 but with the specified restrictions.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $\mathbf{b}' \in \{0,1\}$ on $\mathbf{b}$. If $\mathbf{b} = \mathbf{b}'$, $\mathcal{B}$ decides $\hat{e}(g,g)^{ab} = \mathbf{Z}$; otherwise, it decides $\hat{e}(g,g)^{ab} \neq \mathbf{Z}$.

According to the analysis in [5], the above simulation will abort with a negligible probability. Hence, we obtain the theorem. □

5 Conclusions

In this paper, we investigated the CCA security of attribute-based encryption with outsourced decryption (OD-ABE). In particular, we proposed the CCA security model for OD-ABE and the first OD-ABE scheme with CCA security.

However, the proposal is proven secure in the random oracle model. It is interesting design a new OD-ABE scheme that can be proven secure in the standard model.

Acknowledgements. We would like to thank all the anonymous reviewers for their helpful comments. This work was supported by the National Natural Science Foundation of China [grant numbers 61472364, 61472365, and 61572435]; and the Natural Science Foundation of Zhejiang Province [grant numbers LR13F020003, LZ16F020001, and LR15G010001].

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE S&P 2007, pp. 321–334 (2007)
2. Blaze, M., Bleumer, G., Strauss, M.J.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
3. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
4. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: ACM CCS 2007, pp. 456–465 (2007)
5. 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)
6. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine grained access control of encrypted data. In: ACM CCS 2006, pp. 89–98 (2006)
7. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM Conference on Computer and Communications Security, pp. 89–98. ACM (2006)
8. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: USENIX Security Symposium (2011)
9. He, S., Chen, J., Jiang, F., Yau, D.K.Y., Xing, G., Sun, Y.: Energy provisioning in wireless rechargeable sensor networks. IEEE Trans. Mob. Comput. **12**(10), 1931–1942 (2013)
10. Lai, C.G.J., Deng, R.H., Weng, J.: Attribute-based encryption with verifiable outsourced decryption. IEEE Trans. Inf. Forensics Secur. **8**(8), 1343–1354 (2013)
11. 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)
12. Li, J., Huang, X., Li, J., Chen, X., Xiang, Y.: Securely outsourcing attribute-based encryption with checkability. IEEE Trans. Parallel Distrib. Syst. **25**(8), 2201–2210 (2014)
13. Lin, S., Zhang, R., Ma, H., Wang, M.: Revisiting attribute-based encryption with verifiable outsourced decryption. IEEE Trans. Inf. Forensics Secur. **10**(10), 2119–2130 (2015)

14. Mao, X., Lai, J., Mei, Q., Chen, K., Weng, J.: Generic and efficient constructions of attribute-based encryptionwith verifiable outsourced decryption. IEEE Trans. Dependable Secur. Comput. (99), 1 (2015)
15. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures.In: ACM CCS 2007, pp. 195–203 (2007)
16. O'Toole, J.: Mobile apps overtake PC internet usage in US CNN Money. http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet/, February 2014
17. Qin, B., Deng, R.H., Liu, S., Ma, S.: Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Trans. Inf. Forensics Secur. **10**(7), 1384–1393 (2015)
18. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
19. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
20. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012)
21. 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)
22. 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)