Soft Computing (2019) 23:11409–11417
https://doi.org/10.1007/s00500-019-04088-y

FOUNDATIONS

Improvement of an outsourced attribute-based encryption scheme

Hongjie Chen[1] · Yongjian Liao[1]

Published online: 24 May 2019

Abstract
Attribute-based encryption (ABE) scheme has become an important technology in cryptography. However, the considerable overhead of encryption and decryption restricts its application on the devices with limited resource. Fortunately, as the development of cloud computing, someone presents that outsourcing decryption and encryption can be used to deal with the problem. This idea has drawn a lot of attention after being proposed. Recently, Li et al. put forward an efficient verifiable outsourced ABE scheme which outsources the encryption and decryption to the cloud server and enables its valid receivers to verify the correctness of the message after decrypting. However, in this paper, we prove that their scheme is not secure under chosen plaintext attack which is the basic security requirement of ABE scheme. Then, we improve their scheme and illustrate that the improved scheme is adaptive chosen ciphertext attack (CCA2) secure in the random oracle model. The final comparison shows that we do not add extra computational burden to improve the security.

Keywords Attribute-based encryption · Outsourced encryption · Outsourced decryption · Cloud computing · CCA2

1 Introduction

In 2005, Sahai and Waters (2005) constructed a fuzzy identity-based encryption (FIBE) scheme which is generally called attribute-based encryption (ABE) scheme now. Their original intention was to add error-tolerance into the identity-based encryption (IBE) scheme. Now, it is mainly used to realize the access control in the one-to-many system. Up to now, ABE schemes have been categorized into ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). For the former type (Bethencourt et al. 2007), the access structure is related to the ciphertext and the attributes are bound up with the private key. While in the later type (Goyal et al. 2006), the attribute is related to the ciphertext and the access structure is bound up with the private key.

ABE schemes have great value in practical applications, such as access control (Pooja et al. 2017; Huang et al. 2013), cloud computing (Shankar and Singh 2018; Havisha et al. 2017), keywords search (Wang et al. 2013), social networking (Yang et al. 2017; Li et al. 2017), Internet of Things (Li et al. 2018c, b) and so on. However, the considerable overhead of encryption and decryption is becoming the critical problem that impedes its development. Fortunately, as the development of cloud computing, outsourcing the computation to the cloud server becomes realistic.

The first ABE scheme with outsourcing property was put forward by Green et al. (2011). In their scheme, receivers are able to outsource most of their decryption operations to the cloud server without leaking the corresponding plaintext. Then, using similar method, many related schemes were proposed (Lai et al. 2013; Li et al. 2014; Zhang et al. 2017; Lin et al. 2015; Liao et al. 2018). Besides outsourcing the decryption, some scholars also realized the outsourcing of encryption (Li et al. 2012; Wang et al. 2017). Li et al. (2012) constructed a CP-ABE scheme with encryption outsourcing. In their scheme, sender just needs to do some simple operations to get partial ciphertext. Then, the cloud server will do most of the encryption operations to get the other part of ciphertext by using an outsourced encryption key. However, their scheme still needs the user to do a great amount of exponentiation operations which are related to the attributes. Wang et al. (2017) put forward a verifiable outsourced ABE scheme which realizes the encryption, decryption and key

Communicated by A. Di Nola.

✉ Yongjian Liao
liaoyj@uestc.edu.cn

Hongjie Chen
1547558367@qq.com

[1] University of Electronic Science and Technology of China, Chengdu, China

generation outsourcing. But in their scheme, the large ciphertext size will cause large waste in bandwidth.

Based on Green et al.'s scheme (Green et al. 2011), Li et al. (2018a) presented a replayable chosen ciphertext attack (RCCA) secure ABE scheme which drastically reduces the user's computation by outsourcing both the encryption and decryption to the cloud server. In their scheme, they adopt Green et al.'s method to outsource the decryption and try a new method to outsource the encryption. Assuredly, if their scheme is secure, their scheme will have great value in practical application. However, we find that their scheme is not secure even under CPA attack which is the basic security requirement of ABE scheme, though the author defined that their scheme satisfied RCCA security. We will show that the adversary could guess the right coin $b \in \{0, 1\}$ when the adversary only knows the challenge messages m_0, m_1 and the challenge ciphertext C_b in the indistinguishable game.

1.1 Contributions

In this paper, we will improve the security of Li et al.'s scheme (Li et al. 2018a). Our main contributions can be described as follows.

- We claim that Li et al.'s scheme (Li et al. 2018a) is not secure under CPA attack which is the basic security requirement of ABE scheme. Then, we show that a CPA adversary is able to break their scheme with probability 1.
- We improve Li et al.'s scheme such that the improved scheme is CCA2 secure in the random oracle model. We give the details of security proof which is omitted in Li et al.'s scheme.
- The comparison shows that the improvement just adds one element to the ciphertext and does not add extra calculative burden.

1.2 Organization

The major contents and structure are arranged below. We will recall some basic concepts in Sect. 2. After that, we introduce Li et al.'s outsourced ABE scheme and illustrate why their scheme is not secure under CPA attack in Sect. 3. Our improved scheme and its analysis are provided in Sect. 4. Finally, in Sect. 5, we conclude this paper.

2 Preliminaries

In this section, we will introduce some basic concepts which will be used in this paper.

2.1 Bilinear map

Boneh and Franklin (2001) introduced the bilinear map in detail. We will give a brief review about it. Pick two multiplicative cyclic groups G_1 and G_2, which satisfy the bilinear map $e : G_1 \times G_1 \rightarrow G_2$, with prime order p. Let g be a random generator of G_1. The bilinear map e must have the following properties:

- *Bilinear* The equation $e(g^a, g^b) = e(g, g)^{ab}$ holds for any $a, b \in \mathbb{Z}_p$ and $g \in G_1$.
- *Non-degenerate* If $g \neq 1_{G_1}$, we have $e(g, g) \neq 1_{G_2}$, where 1_{G_1} and 1_{G_2} represent identity element of G_1 and G_2, respectively.
- *Computability* For any $g \in G_1, e(g, g)$ can be computed efficiently by some algorithms.

2.2 Access structure

Definition 1 (*Access Structure*) For all B, C, if $B \in \mathbb{A}$ and $B \subseteq C$, we have $C \in \mathbb{A}$, where $\{P_1, P_2, \ldots, P_n\}$ are a series of participants. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone, if it satisfies the above condition. The access structure is a non-empty set $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \backslash \{\emptyset\}$. We define a set as an authorized set if it is in $\mathbb{A}$. Otherwise, it is an unauthorized set.

2.3 Linear secret-sharing scheme

Beimel (1996) gave a definition about the linear secret-sharing scheme (LSSS). The definition is described as follows.

Definition 2 (*Linear Secret-Sharing Scheme*) For a series of parties $\mathcal{P}$, we define Π as a linear secret-sharing scheme, if it satisfies the following conditions:

- All the parties' shares comprise a vector over $\mathbb{Z}_p$.
- For Π, there is a $l \times n$ matrix A called share-generating matrix. A function ρ maps each row of A to a party. Let $\mathbf{v} = (s, r_2, r_3, \ldots, r_l)^T$ be a column vector, where r_2, $r_3, \ldots, r_l$ are randomly chosen from $\mathbb{Z}_p$ and $s \in \mathbb{Z}_p$ is the sharing secret. Then, the l shares of secret s are $A\mathbf{v}$. For each party $\rho(i)$, the share is $\lambda_i = A_i\mathbf{v}$.

The linear secret-sharing scheme satisfies the following linear reconstruction feature. If Π is a LSSS scheme for access structure $\mathbb{A} = (A, \rho)$ and $S \in \mathbb{A}$ is an authorized set. Let $I = \{i : \rho(i) \in S\}$ be a subset of $\{1, 2, \ldots, l\}$. There exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ enabling $\omega A = (1, 0, \ldots, 0)$. That is to say, we can recover the shared secret as $\sum_{i \in I} \omega_i \lambda_i = s$ if $\{\lambda_i\}$ are the valid shares of s.

2.4 Decisional q-bilinear Diffie–Hellman exponent assumption

We adopt the definition of the decisional q-bilinear Diffie–Hellman exponent (q-BDHE) problem from Waters (2011). Choose a random cyclic group G_1 with prime order p. Pick two random elements $a, s \in \mathbb{Z}_p$ and a random generator g of G_1. The challenger is given $Y = (g, g^a, g^{a^2}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, g^s)$ and $T \in G_2$. The challenger must decide if the T is randomly chosen from G_2 or $T = e(g, g)^{a^{q+1}s} \in G_2$.

We define that a probabilistic polynomial time (PPT) algorithm can solve the decisional q-BDHE problem with advantage ε if:

$$\begin{aligned} \mathrm{Adv}_{\mathcal{B}}^{q-\mathrm{BDHE}} &= Pr[B(Y, T = e(g, g)^{a^{q+1}s}) = 0] \\ &\quad - Pr[B(Y, T = R) = 0] \\ &\geq \varepsilon. \end{aligned}$$

Definition 3 We define that the decisional q-BDHE assumption holds if there does not exist a PPT algorithm that can solve the decisional q-BDHE problem with non-negligible advantage.

2.5 System model

Our system model still follows Li et al.'s outsourced ABE model (Li et al. 2018a).

- *Setup*(k, U) The key generation center takes as input a security parameter k and the universal attribute set U. Then, it generates the system parameter $Param$ and the master key MK.
- *KeyGen*$(Param, MK, S)$ The key generation center takes as input the system parameter $Param$, the master key MK and a set $S \subseteq U$ of attributes. It outputs the private key SK and the decryption transformation key DTK.
- *ETKGen*$(U, Param)$ The user takes the universal attributes U and the system parameter $Param$ as input, then generates a encryption transformation key ETK.
- *Encrypt* The encryption algorithm is performed by two parties: the local user and the cloud server.
  - *Local*$(Param, m, ETK, \mathbb{A})$ With the input of the system parameter $Param$, the plaintext m, the access structure $\mathbb{A}$ and the encryption transformation key ETK, the user generates the partial ciphertext UC and the outsourcing parameter OP.
  - *Cloud*(OP) The cloud server outputs the total ciphertext C with the input of the outsourcing parameter OP.
- *Transform*(C, DTK) With the input of the decryption transformation key DTK and the ciphertext C, the cloud server generates the transformed ciphertext CT.
- *Decrypt*(SK, CT) The user takes as input the private key SK and the transformed ciphertext CT. Then, it does some simple decryption operations to get the plaintext.

2.6 Security model

In this subsection, we will introduce two security models: (selectively) chosen plaintext attack (CPA) security model and (selectively) adaptive chosen ciphertext attack (CCA2) security model. Both in these two models, the adversary $\mathcal{A}$ and the challenger $\mathcal{B}$ will play an indistinguishable game.

2.6.1 CPA security model

- *Init* The challenger $\mathcal{B}$ is given the decisional q-BDHE challenge parameters. The adversary $\mathcal{A}$ provides $\mathcal{B}$ with a challenge access structure $\mathbb{A}^* = (A^*, \rho^*)$.
- *Setup* $\mathcal{B}$ runs the setup algorithm and sends the system parameter $Param$ to the adversary.
- *Phase I* The challenger answers the adversary's queries for the private key SK and the decryption outsourced key DTK for the set S which does not satisfy the access structure (A^*, ρ^*).
- *Challenge* $\mathcal{A}$ chooses two messages m_0, m_1 of the same length and sends them to the challenger $\mathcal{B}$. Then $\mathcal{B}$ chooses $b \in \{0, 1\}$ randomly and encrypts m_b under (A^*, ρ^*). The ciphertext C_b is returned to $\mathcal{A}$.
- *Phase II* This stage is the same as Phase I.
- *Guess* $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b.

The advantage for $\mathcal{A}$ to win the game can be defined as:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{CPA}} = \left| Pr[b = b'] - \frac{1}{2} \right|.$$

2.6.2 CCA2 security model

- *Init and Setup* These two algorithms are the same as they are in the CPA security model, respectively.
- *Phase I* $\mathcal{B}$ builds a blank table $\mathcal{T}$ and answers the adversary's repeated queries as follows:
  - *Create*(S) For the attribute set S, it runs the algorithm KeyGen to compute the decryption transformation key DTK and the private key SK. Finally, it stores the tuple (S, SK, DTK) into the table $\mathcal{T}$ and sends the DTK to the adversary $\mathcal{A}$.
  - *Corrupt*(S) If there is a tuple (S, SK, DTK) in the table $\mathcal{T}$, $\mathcal{B}$ will return SK to $\mathcal{A}$. Otherwise, it returns $\perp$.

- *Decrypt*(S, C) For the adversary's decryption queries, the challenger firstly searches the table $\mathcal{T}$ to obtain the tuple (S, SK, DTK). If the tuple does not exist, it returns $\perp$. Otherwise, it decrypts the ciphertext C by running the Decrypt algorithm and returns the corresponding plaintext to $\mathcal{A}$.

- *Challenge* $\mathcal{A}$ sends two challenge messages m_0, m_1 of the same length to $\mathcal{B}$. Then, $\mathcal{B}$ flips a coin to get b and encrypts $m_b \in \{m_0, m_1\}$ into the challenge ciphertext C_b. After that, it sends C_b to $\mathcal{A}$. In this stage, the outsourcing parameter OP^* for the challenge message can be accessed by the adversary.
- *Phase II* For the adversary's query, the challenger repeats Phase I, but refuses to decrypt C_b.
- *Guess* $\mathcal{A}$ outputs its guess b' for b.

The advantage for $\mathcal{A}$ to win the game can be defined as:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{CCA2}} = \left| Pr[b = b'] - \frac{1}{2} \right|.$$

3 Li et al.'s scheme

We firstly recall Li et al.'s scheme (Li et al. 2018a) before analyzing it.

3.1 Construction of their scheme

Their scheme contains six phases: Setup, KeyGen, ETKGen, Encrypt, Transform, Decrypt. The details are described as follows:

- *Setup*(k, U) The key generation center takes as input a secure parameter k and the universal set $U = \{1, 2, \ldots, u\}$ of attributes. It generates two multiplicative cyclic groups G_1, G_2 of prime order p, where G_1 and G_2 satisfy the bilinear map $e : G_1 \times G_1 \to G_2$, and a hash function $H : G_1^3 \to \mathbb{Z}_p$. Then, it picks a generator g of G_1, $\alpha, a \in \mathbb{Z}_p^*$ and $h_1, h_2, \ldots, h_u \in G_1$ randomly, where $h_1, h_2, \ldots, h_u$ are related to the attributes. Finally, it publishes the system parameter:

  $$Param = (G_1, G_2, g, e(g, g)^{\alpha}, g^a, h_1, h_2, \ldots, h_u, H)$$

  and keeps the master key $MK = (\alpha, a)$ secretly.
- *KeyGen*$(Param, MK, S)$ With the input of the system parameter $Param$, the master key MK and a set $S \subseteq U$ of attributes, the key generation center randomly chooses $t, z \in \mathbb{Z}_p^*$ and computes $K = g^{\alpha} g^{at}$, $K_0 = g^t$, $K_x = h_x^t$ for all $x \in S$. Finally, it publishes the decryption transformation key $DTK = (K^z, K_0^z, \{K_x^z\}_{x \in S})$ and computes the private key $SK = z^{-1} (\mathrm{mod\ p})$.
- *ETKGen*$(U, Param)$ Each user takes the universal attribute set U and the system parameter $Param$ as input and picks a random element $\gamma \in \mathbb{Z}_p$. Then, it outputs the encryption transformation key $ETK = (\gamma, \{h_x^{\gamma}\})$, where $x \in U$. Unlike the DTK, the ETK should be kept secretly.
- *Encrypt* In this phase, the encryption results come from two parties: the local user and the cloud server.
  - *Local*$(Param, m, ETK, \mathbb{A})$: The user takes as input the system parameter $Param$, the plaintext $m \in G_2$, the encryption transformation key ETK and an access structure $\mathbb{A} = (A, \rho)$, where A is a $l \times n$ matrix and the function ρ maps each row of matrix A to an attribute. Then, it randomly picks elements $d, s \in \mathbb{Z}_p$ and a vector $\mathbf{v} = (s, r_2, \ldots, r_n)^T \in \mathbb{Z}_p^n$. After that, it computes the shares $\{\lambda_i = A_i \cdot \mathbf{v}\}_{i \in \{1, \ldots, l\}}$ of the secret s, where A_i denotes the i-th row of A. Finally, the user outputs the partial ciphertext:

    $$UC = \left(C_1 = m \cdot e(g, g)^{\alpha s}, C_2 = g^s, C_3 = g^{sr}\right),$$

    where $r = H(C_1, C_2, m)$. Meanwhile, the user generates the outsourcing parameter:

    $$OP = \left(\lambda_i - d, -sr - \gamma, g^{ad} \cdot h_{\rho(i)}^{\gamma}\right),$$

    where $i = 1, 2, \ldots, l$. Then, it sends OP to the cloud server.
  - *Cloud*(OP): After getting the outsourcing parameter OP, the cloud server computes the other part of the ciphertext $\{\overline{C}_i\}_{i \in \{1, 2, \ldots, l\}}$, where

    $$\overline{C}_i = (g^a)^{\lambda_i - d} \cdot g^{ad} \cdot h_{\rho(i)}^{\gamma} \cdot h_{\rho(i)}^{-sr-\gamma} = g^{a\lambda_i} \cdot h_{\rho(i)}^{-sr}.$$

    Finally, the full ciphertext C is composed as:

    $$C = \left(C_1, C_2, C_3, \{\overline{C}_i\}_{i \in \{1, 2, \ldots, l\}}\right).$$

- *Transform*(C, DTK) With the input of the ciphertext C for $\mathbb{A}$ and DTK for an attribute set S, the cloud server defines a set $I = \{i : \rho(i) \in S\}$. It then is able to compute $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \lambda_i = s$ if S satisfies $\mathbb{A}$. Finally, it computes the transformation ciphertext:

  $$CT = \frac{e(K^z, C_2)}{\prod_{i \in I} \left(e(K_0^z, \overline{C}_i) \cdot e(K_{\rho(i)}^z, C_3)\right)^{\omega_i}} = e(g, g)^{\alpha s z}.$$

– *Decrypt*(SK, CT) The user uses the private key SK and the transformation ciphertext CT to compute:

$$m = \frac{C_1}{CT^{SK}}$$

and if the result satisfies the equation:

$$C_3 = C_2^{H_1(C_1, C_2, m)},$$

it returns the message m.

3.2 Analysis of their scheme

We now prove that the above scheme is insecure under CPA attack.

From the definition of the CPA security model, after querying and receiving some private keys, the adversary $\mathcal{A}$ chooses two challenge messages m_0, m_1 of the same length and sends them to the challenger. The challenger encrypts the message m_b, where $b \in \{0, 1\}$, then it returns the ciphertext $C_b = (C_1, C_2, C_3, \{\overline{C}_i\})$ to the adversary. After receiving the challenge ciphertext C_b, the adversary $\mathcal{A}$ chooses a message from $\{m_0, m_1\}$. Suppose $\mathcal{A}$ chooses m_0. Then $\mathcal{A}$ computes $r' = H(C_1, C_2, m_0)$ and $C_3' = C_2^{r'}$. If $C_3 = C_3'$ holds, the adversary can determine that $b = 0$. Otherwise, it determines that $b = 1$. By this way, the adversary knows whether m_b is m_0 or m_1 with probability 1. That is to say, the scheme cannot satisfy the CPA security. As we know, if a scheme cannot satisfy CPA security, it cannot also satisfy RCCA security.

4 The improved scheme

Focusing on the security of Li et al.'s scheme(Li et al. 2018a), we propose the following CCA2 secure scheme.

4.1 Construction

– *Setup*(k, U) This phase is similar to the above scheme, except that our scheme picks two secure hash functions $H_1 : G_2 \times G_1 \times G_2 \times \mathbb{Z}_p \to \mathbb{Z}_p$ and $H_2 : G_2 \to \mathbb{Z}_p$. So, the system parameter is: $Param =$

$$(G_1, G_2, g, e(g, g)^{\alpha}, g^a, h_1, h_2, \ldots, h_u, H_1, H_2)$$

and the master key is: $MK = \alpha$.

– *KeyGen*$(Param, MK, S)$ This phase is similar to the KeyGen phase in Li et al.'s scheme. The key generation center also chooses a random element $z \in \mathbb{Z}_p^*$, and it computes the private key $SK = z$ and the decryption transformation key $DTK = \left(K^{1/z}, K_0^{1/z}, \{K_x^{1/z}\}_{x \in S}\right)$.

– *ETKGen*$(U, Param)$ This phase is the same as the ETKGen phase in the above scheme.

– *Encrypt* This phase is interactively implemented by the cloud server and the local user.

– *Local* $(Param, m, ETK, \mathbb{A})$ The input parameters are the same as the above scheme, and the message m belongs to $\mathbb{Z}_p$. The user randomly picks elements $W \in G_2, d, s, r_2, r_3, \ldots, r_n \in \mathbb{Z}_p$ and constructs vector $\mathbf{v} = (s, r_2, \ldots, r_n)^T \in \mathbb{Z}_p^n$. After that, it computes the shares $\{\lambda_i = A_i \cdot \mathbf{v}\}_{i \in \{1,2,\ldots,l\}}$ of the secret s. Finally, it outputs the user's ciphertext: $UC = (C_1, C_2, C_3, C_4)$, where

$$C_1 = W \cdot e(g, g)^{\alpha s},$$
$$C_2 = g^s,$$
$$C_3 = g^{sr},$$
$$C_4 = H_2(W) \oplus m,$$

and $r = H_1(C_1, C_2, m, W)$. At the same time, it computes the blinding factors $(g^a)^d \cdot h_{\rho(i)}^{\gamma}$, $-\frac{s}{r} - \gamma$ and $\frac{\lambda_i}{r} - d$ for all $i = 1, 2, \ldots, l$ and sends the outsourcing parameter

$$OP = \left(r, \frac{\lambda_i}{r} - d, -\frac{s}{r} - \gamma, g^{ad} \cdot h_{\rho(i)}^{\gamma}\right)$$

to the cloud server.

– *Cloud*(OP) After receiving the outsourcing parameter OP, the cloud server can compute the other part of the ciphertext $\{\overline{C}_i\}_{i \in \{1,2,\ldots,l\}}$, where

$$\begin{aligned}\overline{C}_i &= \left((g^a)^{\frac{\lambda_i}{r} - d} \cdot g^{ad} \cdot h_{\rho(i)}^{\gamma} \cdot h_{\rho(i)}^{-\frac{s}{r} - \gamma}\right)^r \\ &= g^{a\lambda_i} \cdot h_{\rho(i)}^{-s}.\end{aligned}$$

Finally, the full ciphertext is composed as:

$$C = \left(C_1, C_2, C_3, C_4, \{\overline{C}_i\}_{i \in \{1,\ldots,l\}}\right).$$

– *Transform*(C, DTK) This phase is also similar to the above scheme, but C_3 in their equation is replaced by C_2. That is to say the method to compute CT is:

$$\begin{aligned}CT &= \frac{e(K^{1/z}, C_2)}{\prod_{i \in I}\left(e(K_0^{1/z}, \overline{C}_i) \cdot e(K_{\rho(i)}^{1/z}, C_2)\right)^{\omega_i}} \\ &= e(g, g)^{\alpha s/z}.\end{aligned}$$

– *Decrypt*(SK, CT) After receiving the transformed ciphertext CT, the user computes:

$$W = \frac{C_1}{CT^{SK}}.$$

Then, it computes: $m = H_2(W) \oplus C_4$. If the result satisfies the equation $C_3 = C_2^{H_1(C_1, C_2, m, W)}$, it returns the plaintext m.

4.2 Security analysis

We now prove that our improved scheme is CCA2 secure in the random oracle model.

Theorem *Suppose $\mathcal{A}$ is a CCA2 attacker who can attack our improved scheme with advantage ε. Then, there will be a challenger $\mathcal{B}$ that has the same advantage to solve the decisional q-BDHE problem.*

- *Init* The challenger $\mathcal{B}$ is given the decisional q-BDHE challenge parameters Y, T. The adversary $\mathcal{A}$ provides a challenge access structure $\mathbb{A}^* = (A^*, \rho^*)$ to B, where A^* is a $l^* \times n^*$ matrix and $n^* \leq q$. The challenger's goal is to determine whether T is $e(g, g)^{a^{q+1}s} \in G_2$ or is chosen from G_2 randomly.
- *Setup* The challenger $\mathcal{B}$ picks $\alpha' \in \mathbb{Z}_p$ randomly and implicitly sets $\alpha = \alpha' + a^{q+1}$. In this phase, the challenger does not know the value of α exactly, but it can compute: $e(g, g)^{\alpha} = e(g^a, g^{a^q})e(g, g)^{\alpha'}$. And for each $x \in U$, it randomly chooses $z_x \in \mathbb{Z}_p$. If for one particular i, the equation $\rho^*(i) = x$ holds, then the challenger sets $h_x = g^{z_x} \cdot g^{aA^*_{i,1}} \cdot g^{a^2 A^*_{i,2}} \ldots g^{a^{n^*} A^*_{i,n^*}}$. Otherwise, it sets $h_x = g^{z_x}$.
- *Phase I* $\mathcal{B}$ firstly initializes three blank tables $\mathcal{T}_0$, $\mathcal{T}_1$ and $\mathcal{T}_2$. The following steps are the challenger's answers to the adversary's queries.
  - $H_1(C_1, C_2, m, W)$ $\mathcal{B}$ searches the table $\mathcal{T}_1$ to find if there exists a tuple (C_1, C_2, m, W, r). If not, it picks a random element $r \in \mathbb{Z}_p$ which is different from all the values of r in the table $\mathcal{T}_1$. Then, it stores the tuple (C_1, C_2, m, W, r) into the table $\mathcal{T}_1$ and returns r. Otherwise, it returns the corresponding r directly.
  - $H_2(W)$ $\mathcal{B}$ searches the table $\mathcal{T}_2$ to find if there exists a tuple (W, ω). If not, it picks a random element $\omega \in \mathbb{Z}_p$ which is different from all the values of ω in the table $\mathcal{T}_2$. Then, it stores the tuple (W, ω) into the table $\mathcal{T}_2$ and returns ω. Otherwise, it returns the corresponding ω directly.
  - *Create*(S) Because $\mathcal{A}$ cannot ask for the private key of the attributes set S which satisfies the challenge structure $\mathbb{A}^*$. So for the attribute set S which does not satisfy $\mathbb{A}^*$, $\mathcal{B}$ computes the private key as follows.
    According to the construction of the access structure, the challenger can find a vector $\mathbf{w} = (w_1, w_2, \ldots, w_{n^*})^T \in \mathbb{Z}_p^{n^*}$, where $w_1 = -1$ such that $A_i^* \cdot \mathbf{w} = 0$ for all i where $\rho^*(i) \in S$. Then, it randomly chooses $r \in \mathbb{Z}_p$ and defines t as:

$$t = r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{n^*} a^{q-n^*+1}.$$

After that, for the set S of attributes, it computes:

$$K_0 = g^r \prod_{i=1,\ldots,n^*} (g^{a^{q+1-i}})^{w_i} = g^t,$$
$$K = g^{\alpha'} g^{ar} \prod_{i=2,\ldots n^*} (g^{a^{q+2-i}})^{w_i} = g^{\alpha} g^{at}.$$

If there exist a i such that the equation $\rho^*(i) = x$ holds, the challenger creates $K_x =$

$$K_0^{z_x} \prod_{j=1,\ldots,n^*} \Big(g^{a^j \cdot r} \prod_{\substack{k=1,\ldots,n^* \\ k \neq j}} (g^{a^{q+1+j-k}})^{w_k}\Big)^{A^*_{i,j}} = h_x^t.$$

Otherwise, it computes $K_x = K_0^{z_x} = h_x^t$. It randomly picks $z \in \mathbb{Z}_p$ as the private key and computes the decryption transformation key: $DTK = (K^{1/z}, K_0^{1/z}, K_x{}^{1/z})$. After that, it stores the tuple (S, SK, DTK) into the table $\mathcal{T}_0$ and sends DTK to $\mathcal{A}$.
- *Corrupt*(S) Suppose the adversary does not query the private key for a set S, where S satisfies $\mathbb{A}^*$. The challenger firstly searches the table $\mathcal{T}_0$ to get the tuple (S, SK, DTK). If the tuple does not exist, it returns $\perp$. Otherwise, it returns SK.
- *Decrypt*(S, C) For the adversary's decryption queries, the challenger performs the following steps.
  - It searches the table $\mathcal{T}_1$ to get the tuples $(C_1, C_2, m_i, W_i, r_i)$. If there is not such a tuple it returns $\perp$.
  - Otherwise, for each i, it compares C_3 with $C_2^{r_i}$. If there is an i such that $C_3 = C_2^{r_i}$, it stamps the tuple $(C_1, C_2, m_i, W_i, r_i)$. We note that the i is unique because the challenger always chooses different r for $H_1(C_1, C_2, m, W)$.
  - Then, it searches the table $\mathcal{T}_2$ to get the tuple (W_i, ω_i). If $m_i = \omega_i \oplus C_4$, it returns m_i. Otherwise, it returns $\perp$.
- *Challenge* In this phase, we will introduce how to generate the challenge ciphertext. The adversary $\mathcal{A}$ sends two messages m_0, m_1 of the same length to the challenger $\mathcal{B}$. Then, $\mathcal{B}$ flips a coin b and encrypts $m_b \in \{m_0, m_1\}$. It chooses $W \in G_2$ and $r, \omega \in \mathbb{Z}_p$ randomly. The partial ciphertext is $C_b = (C_1^*, C_2^*, C_3^*, C_4^*)$, where

$$C_1^* = WTe(g^{\alpha'}, g^s),$$
$$C_2^* = g^s,$$
$$C_3^* = g^{sr},$$
$$C_4^* = \omega \oplus m_b.$$

The challenger picks $y_2, y_3, \ldots, y_{n^*} \in \mathbb{Z}_p$ randomly and computes:

$$\mathbf{v} = (s, sa + y_2, sa^2 + y_3, \ldots, sa^{n^*-1} + y_{n^*})^T.$$

Then, it picks $y, r_1, r_2, \ldots, r_{l^*} \in \mathbb{Z}_p$ and sets $\gamma = -\frac{s}{r} - y$, $d_i = \frac{\lambda_i}{r} - r_i$. That is to say, $y = -\frac{s}{r} - \gamma$ and $r_i = \frac{\lambda_i}{r} - d_i$. For all $i \in \{1, \ldots, l^*\}$, it can compute:

$$\begin{aligned}(g^a)^{d_i} \cdot h_{\rho(i)}^{\gamma} &= (g^a)^{\frac{\lambda_i}{r} - r_i} \cdot h_{\rho(i)}^{-\frac{s}{r} - y} \\ &= (g^a)^{\frac{\lambda_i}{r}} \cdot h_{\rho(i)}^{-\frac{s}{r}} \cdot g^{-ar_i} \cdot h_{\rho(i)}^{-y} \\ &= \Big(\big(\prod_{j=1,\ldots,n^*} (g^a)^{A_{i,j}^* y_j} \big) (g^s)^{-z_{\rho^*(i)}} \Big)^{\frac{1}{r}} \cdot \\ &\quad g^{-ar_i} \cdot h^{-y}.\end{aligned}$$

After that it publishes the outsourcing parameter:

$$OP^* = \big(r, \{r_i\}, y, \{(g^a)^{d_i} \cdot h_{\rho(i)}^{\gamma}\}\big)$$

and computes:

$$\begin{aligned}\{\overline{C}_i^* &= \Big((g^a)^{r_i} (g^a)^{d_i} \cdot h_{\rho(i)}^{\gamma} \cdot h_{\rho(i)}^{y} \Big)^r \\ &= g^{a\lambda_i} \cdot h_{\rho(i)}^{-s}\}_{i \in \{1,\ldots,l^*\}}.\end{aligned}$$

Finally, the challenger returns the ciphertext:

$$C_b = \big(C_1^*, C_2^*, C_3^*, C_4^*, \{\overline{C}_i^*\}_{i \in \{1,\ldots,l^*\}}\big).$$

- *Phase II* This phase is similar to phase I, but $\mathcal{A}$ cannot query for decrypting the challenge ciphertext C_b.
- *Guess* The adversary $\mathcal{A}$ gives its guess b' for b. The challenger determines that T is the tuple $e(g, g)^{a^{q+1}s}$ if $b' = b$. Otherwise, it guesses that T is chosen randomly from G_2.

If T is the tuple $e(g, g)^{a^{q+1}s}$, the challenger provides the adversary with a perfect attack environment. So the probability for the challenger to win the challenge is:

$$Pr[\mathcal{B}(Y, T = e(g, g)^{a^{q+1}s}) = 1] = \frac{1}{2} + \varepsilon.$$

If T is randomly chosen from G_2, the message m_b is hidden from the adversary completely. And the advantage for the challenger to win the game is:

$$Pr[\mathcal{B}(Y, T = R) = 1] = \frac{1}{2}.$$

Accordingly, the challenger has non-negligible advantage ε to solve the decisional q-BDHE problem, because:

$$\begin{aligned}Adv_{\mathcal{B}}^{q-BDHE} &= Pr[B(Y, T = e(g, g)^{a^{q+1}s}) = 1] \\ &\quad - Pr[B(Y, T = R) = 1] \\ &= \varepsilon.\end{aligned}$$

4.3 Performance

Now, we will analyze the performance of our improved scheme. We first compare the communication cost, the computation cost, the checkability and the security among our scheme and other outsourced ABE schemes (Green et al. 2011; Lin et al. 2015; Qin et al. 2015; Wang et al. 2017; Li et al. 2018a). Because we mainly change the encryption stage of Li et al.'s scheme, the computation and communication cost will be simplified to encryption time and ciphertext size, respectively. In Green et al.'s scheme, we choose their CP-ABE scheme which satisfies RCCA security. For the encryption time, we only compare the exponentiation time which is the main operation. The details are shown in Table 1, where l, l_1, l_2, t_1, t_2 denote the number of rows of matrix A, the element length of G_1, the element length of G_2, the exponentiation time over G_1 and the exponentiation time over G_2, respectively. Comparing with Li et al.'s scheme, to achieve CCA2 security, we just add one element to the ciphertext and the encryption time is the same, which means we do not increase the complexity of the ciphertext and computational burden.

In order to make the comparison more concrete, we compute the specific value of the user's encryption time according to Liao et al.'s experimental result (Liao et al. 2017) in Table 1 of their paper. The author implemented their protocol using PBC library on a computer of Windows XP operating system with Intel Core i5-3210M (2.50 GHz) and 4 GB RAM. We provide the details for $l = 10$ in Table 2. Besides the computations in Table 1, the computation results in Table 2 also include the multiplication over G_1 and G_2. We find that, among these schemes, our scheme and Li et al.'s scheme perform better. Compared with Li et al.'s scheme, our scheme will not add extra computational burden while increasing security.

5 Conclusion

In our work, we firstly reviewed Li et al.'s scheme and proved that their scheme is insecure under CPA attack. After that, we proposed an improved scheme which is CCA2 secure in the random oracle model. Finally, by comparing with other

Table 1 Performance comparison

Scheme	Ciphertext size	Encryption time	Checkability	Security
Green et al. (2011)	$(2+l)l_1+l_2$	$(1+3l)t_1+t_2$	Yes	RCCA
Lin et al. (2015)	$(3+2l)l_1$	$(3+3l)t_1+t_2$	Yes	CPA
Qin et al. (2015)	$(1+l)l_1+l_2$	$(1+3l)t_1+t_2$	Yes	RCCA
Wang et al. (2017)	$(2+3l)l_1+l_2$	$(3+l)t_1+t_2$	No	CPA
Li et al. (2018a)	$(2+l)l_1+l_2$	$3t_1+t_2$	Yes	No
Our scheme	$(3+l)l_1+l_2$	$3t_1+t_2$	Yes	CCA2

Table 2 The user's encryption time comparison

Scheme	Encryption time
Green et al. (2011)	250.461 ms
Lin et al. (2015)	266.498 ms
Qin et al. (2015)	250.461 ms
Wang et al. (2017)	106.353 ms
Li et al. (2018a)	26.293 ms
Our scheme	26.293 ms

schemes, our improved scheme did not add extra computational burden to the user while improving security.

Acknowledgements This study was funded by the National Natural Science Foundation of China (NSFC) under Grant No. 61472066, and the Sichuan Science and Technology Program (Nos. 2018GZ0180, 2018GZ0085, 2017GZDZX002).

Compliance with ethical standards

Conflict of interest The authors declare that there are no conflicts of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Beimel A (1996) Secure schemes for secret sharing and key distribution. Technion-Israel Institute of technology, Faculty of computer science

Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy (SP '07) (2007), pp 321–334. https://doi.org/10.1109/SP.2007.11

Boneh D, Franklin M (2001) Identity-based encryption from the Weil pairing. In: Advances in cryptology—CRYPTO 2001, vol 2139. Springer, Berlin, pp 213–229. https://doi.org/10.1007/3-540-44647-8_13

Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security. ACM, pp 89–98. https://doi.org/10.1145/1180405.1180418

Green M, Hohenberger S, Waters B (2011) Outsourcing the decryption of abe ciphertexts. In: SEC'11 Proceedings of the 20th USENIX conference on Security, p 34

Havisha V, Padmavathi P, Ramanamurthy S (2017) Cloud security-random attribute based encryption. Innov Comput Sci Eng 8:113–120. https://doi.org/10.1007/978-981-10-3818-1_13

Huang J, Chiang C, Liao I (2013) An efficient attribute-based encryption and access control scheme for cloud storage environment. Grid Pervasive Comput 7861:453–463. https://doi.org/10.1007/978-3-642-38027-3_48

Lai J, Deng R, Guan C, Weng J (2013) Attribute-based encryption with verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 8(8):1343–1354. https://doi.org/10.1109/TIFS.2013.2271848

Li J, Jia C, Li J, Chen X (2012) Outsourcing encryption of attribute-based encryption with mapreduce. Inf Commun Secur 7618:191–201. https://doi.org/10.1007/978-3-642-34129-8_17

Li J, Huang X, Li J, Chen A, Xiang Y (2014) Securely outsourcing attribute-based encryption with checkability. IEEE Trans Parallel Distrib Syst 25(8):2201–2210. https://doi.org/10.1109/TPDS.2013.271

Li Y, Qi F, Tang Z (2017) Traceable and complete fine-grained revocable multi-authority attribute-based encryption scheme in social network. Secur Priv Anonymity Comput Commun Storage 10656:87–92. https://doi.org/10.1007/978-3-319-72389-1_8

Li J, Li X, Wang L, He D, Ahmad H, Niu X (2018a) Fuzzy encryption in cloud computation: efficient verifiable outsourced attribute-based encryption. Soft Comput 22(3):707–714. https://doi.org/10.1007/s00500-017-2482-1

Li X, Niu J, Kumari S, Wu F, Choo KKR (2018b) A robust biometrics based three-factor authentication scheme for global mobility networks in smart city. Future Gener Comput Syst 83:607–618. https://doi.org/10.1016/j.future.2017.04.012

Li X, Peng J, Niu J, Wu F, Liao J, Choo KKR (2018c) A robust and energy efficient authentication protocol for industrial internet of things. IEEE Internet Things J 5(3):1606–1615. https://doi.org/10.1109/JIOT.2017.2787800

Liao Y, He Y, Li F, Zhou S (2017) Analysis of a mobile payment protocol with outsourced verification in cloud server and the improvement. Comput Stand Interfaces 56:101–106. https://doi.org/10.1016/j.csi.2017.09.008

Liao Y, He Y, Li F, Jiang S, Zhou S (2018) Analysis of an abe scheme with verifiable outsourced decryption. Sensors 18:176. https://doi.org/10.3390/s18010176

Lin S, Zhang R, Ma H, Wang S (2015) Revisiting attribute-based encryption with verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 10(10):2119–2130. https://doi.org/10.1109/TIFS.2015.2449264

Pooja R, Pavan B, Apoorva P (2017) Access control with anonymous authentication of data stored in clouds using abe algorithm. In: 2017 International conference on communication and signal processing (ICCSP), IEEE, pp 909–912. https://doi.org/10.1109/ICCSP.2017.8286501

Qin B, Deng R, Liu S, Ma S (2015) Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 10(7):1384–1393. https://doi.org/10.1109/TIFS.2015.2410137

Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Advances in cryptology—EUROCRYPT 2005, vol 3494. Springer, Berlin, pp 457–473. https://doi.org/10.1007/11426639_27

Shankar V, Singh K (2018) Applications of attribute-based encryption in cloud computing environment. Big Data Anal 654:687–692. https://doi.org/10.1007/978-981-10-6620-7_66

Wang C, Li W, Li Y, Xu X (2013) A ciphertext-policy attribute-based encryption scheme supporting keyword search function. Cyberspace Saf Secur 8300:377–386. https://doi.org/10.1007/978-3-319-03584-0_28

Wang H, He D, Shen J, Zheng Z, Zhao C, Zhao M (2017) Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing. Soft Comput 21(24):7325–7335. https://doi.org/10.1007/s00500-016-2271-2

Waters B (2011) Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Public key cryptography—PKC 2011, vol 6571. Springer, Berlin, pp 53–70. https://doi.org/10.1007/978-3-642-19379-8_4

Yang X, Yang M, Yang P, Q L (2017) A multi-authority attribute-based encryption access control for social network. In: 2017 3rd IEEE international conference on control science and systems engineering (ICCSSE), IEEE, pp 671–674. https://doi.org/10.1109/CCSSE.2017.8088017

Zhang J, Wang B, Xhafa F, Wang X, Li C (2017) Energy-efficient secure outsourcing decryption of attribute based encryption for mobile device in cloud computation. J Ambient Intell Humaniz Comput 10:1–10. https://doi.org/10.1007/s12652-017-0658-2