RESEARCH ARTICLE

Traceable ciphertext-policy attribute-based encryption scheme with attribute level user revocation for cloud storage

Shangping Wang¹, Keke Guo^{1*}, Yaling Zhang²

1 School of Science, Xi'an University of Technology, Xi'an, Shaanxi, China, **2** School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, Shaanxi, China

* guokeke127@163.com

OPEN ACCESS

Citation: Wang S, Guo K, Zhang Y (2018) Traceable ciphertext-policy attribute-based encryption scheme with attribute level user revocation for cloud storage. PLoS ONE 13(9): e0203225. <https://doi.org/10.1371/journal.pone.0203225>

Editor: Hua Wang, Victoria University, AUSTRALIA

Received: June 20, 2018

Accepted: August 16, 2018

Published: September 13, 2018

Copyright: © 2018 Wang et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the paper and its Supporting Information file.

Funding: This research is supported by the National Natural Science Foundation of China (No. 61572019, <http://www.nsf.gov.cn>), and the Key Project of Research Foundation of Natural Science Foundation of Shaanxi Province of China (No. 2016JZ001, <http://www.sninfo.gov.cn>). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Abstract

In a ciphertext-policy attribute-based encryption (CP-ABE) scheme, a user may have multiple attributes, and each attribute may be shared simultaneously by many users. The decryption key of an attribute can thus be shared by many users who all possess the attribute. For monetary gain, a malicious authorized user may reveal his/her decryption key to a third party, and it is difficult to trace the owner of primitive secret key from an exposed key. At the same time, this situation may also limit commercial applications of CP-ABE systems. To solve these problems and enable fine-grained access control for the encrypted data, we propose a traceable CP-ABE scheme with attribute-level user revocation for cloud storage (TUR-CPABE). Our scheme enjoys four advantages. First, it has the ability to trace malicious users who have leaked key information from the system. Second, it supports attribute-level user revocation for malicious users and allows ABE fine-grained access control. Third, it allows secret key updates and ciphertext updates to resist collusion attacks between users. Fourth, outsourcing encryption, decryption and attribute revocation are used to reduce the computational burden on data owners, data users and the trust authority, respectively. In addition, our scheme has been proven to be secure against chosen plaintext attacks under a selective access policy based on decisional q -BDHE assumption in the standard model.

1 Introduction

In a cloud storage system, the cloud server must be able to provide data storage and other services for end users. Increasingly, companies and individuals prefer to store their valuable data in cloud servers due to limited equipment resources and the need to process big data. Due to security and privacy concerns, data owners always encrypt their data before outsourcing it to the cloud server. Encrypting data is a valid way to prevent information leakage, but encrypting messages hampers the sharing of messages with fine-grained access control. To solve this problem, Sahai and Waters [1] proposed the concept of attribute-based encryption (ABE), which can provide a “one-to-many” encryption scheme with fine-grained access control.

Competing interests: The authors have declared that no competing interests exist.

ABE has two main categories: ciphertext-policy attribute-based encryption (CP-ABE) [2] and key-policy attribute-based encryption (KP-ABE) [3]. In CP-ABE, attributes are related to the user's decryption key, and access policy affects the ciphertext. KP-ABE is exactly the opposite. Not only can CP-ABE defend the privacy of data and provide ABE with fine-grained access control, it also allows the data owner to define flexible access policies for the data. To obtain the better expressivity, efficiency and security, Waters *et al.* [4–11] conducted a large number of studies of ABE systems. We will discuss two significant developments: trace and revocation.

Recently, several revocable ABE schemes have been presented [12–15]. To our knowledge, revocable ABE systems can be classified into two main types: direct revocable ABE systems and indirect revocable ABE systems. In a direct revocable ABE system, the trusted authority calls for renewal of the revocation list, and users have no need to communicate with the trusted authority after a revocation has taken place. There are some ABE schemes [16–19] that have revocation at the user level. In that case, the trusted authority must control the revocation list and distribute keys for non-revoked users so that revoked users cannot continue to decrypt in an indirect revocable ABE system. Certain ABE schemes [20–22] have been proposed with revocation at the attribute level. Compared with revocation at the user level, revocation at the attribute level offers more fine-grained access control.

In general, traceable CP-ABE is an applied encryption method that is traceable, and it can achieve fine-grained access control. However, some of the existing traceable CP-ABE schemes are less expressive, as the access structure is almost completely restricted to AND gates. To enable greater expressivity, Liu *et al.* [23] proposed a traceable CP-ABE scheme that supports any monotonous access structure. Considering the characteristics of ABE, it is hard to trace the owner of a primitive secret key when a malicious authorized user reveals his/her decryption key to a third party. Inspired by Boyen's [24] signature scheme, Liu added the user's identity in the private key generation phase and uses less expense to add malicious user traceability into the CP-ABE system. Inspired by [23], Ning *et al.* [25] who presented a white-box traceable ciphertext-policy attribute-based encryption scheme that supports flexible attributes. Considering the business applications of CP-ABE, the number of attributes is not polynomial-bound, and storing the cost of a betrayer trace is constant.

Recently, some cryptography researchers have proposed practical ABE schemes that can support traitor tracing and revocation. Wong *et al.* [26] proposed a betrayer tracing and revocation ABE for a large universe. Subsequently, Li *et al.* [27] proposed a traitor tracing and revocable ABE scheme. The scheme was proven to be safe under the prime order group, but it cannot resist a collusion attack between users. Although both Wong and Li's scheme support direct user revocation, their schemes can neither support a key update nor ciphertext updates, nor can they acquire ABE fine-grained access control. In 2017, Liu *et al.* [28] proposed traitor tracing the CP-ABE scheme, which can support ciphertext updates, but it can neither resist collusion attacks between users nor gain key update. In order to explain the above collusion attack, we provide the following specific example.

For instance, a sensitive data is encrypted through the access policy ("hospital" AND "hematology" AND "doctor"). We assume u_1 holds a secret key associated with the attributes set ("hospital" AND "hematology" AND "nurse") and u_2 holds a secret key associated with the attributes set ("hospital" AND "gastroenterology" AND "doctor"). In our assumption, u_1 cannot decrypt the ciphertext because of lacking attribute "doctor", u_2 cannot decrypt the ciphertext because of lacking attribute "hematology". However, they can collude to forge the attributes set ("hospital" AND "hematology" AND "doctor") and get the correct decryption key. Therefore, the scheme [28] suffers the collusion attack between users. To solve the above problem, we present a traceable CP-ABE scheme with attribute-level user revocation.

1.1 Our contribution

Scheme [25] only considered trace malicious users, and scheme [29] can support attribute-level user revocation. Inspired by [25] and [29], we provide a traceable ciphertext-policy attribute-based encryption scheme with attribute-level user revocation for cloud storage (TUR-CPABE). Our main contributions are as follows.

1. In this paper, we formally propose the definition of a traceable ciphertext-policy attribute-based encryption scheme with attribute-level user revocation for cloud storage. In our scheme, we adopt linear secret sharing schemes (LSSS) as an access structure. This provides attribute-level user revocation for malicious user and fine-grained access control for ABE. In our scheme, the trust authority can trace defectors and send the identity of a defector to the attribute manager. The attribute manager is responsible for revoking the defector's attributes and updating the related decryption key and ciphertext. A user in the system could decrypt ciphertext successfully if and only if his/her identity was absent from the revocation list and his/her attributes can satisfy the access policy.
2. In the scheme, we distribute the identity of each user on a leaf node in the *KEK* tree. We can revoke a user by revoking his/her attributes. When the attribute manager revokes a malicious user's attribute i , it will update i 's users group G_i and the corresponding group key GSK , re-encrypted ciphertext CT' and the header of the message Hdr . Thus, our scheme can resist collusion attacks between users.
3. In the proposed scheme, outsourcing encryption, decryption and attribute revocation are used to reduce the computational burden of data owners, users and trust authority, respectively. Moreover, the experimental results show that the time spent in the decryption phase is constant.
4. The scheme is proven to be secure against a chosen plaintext attack under decisional q -BDHE assumption in the standard model.

1.2 Related work

Sahai and Waters first presented the concept of ABE [1], and then, the ideas of KP-ABE and CP-ABE were formally proposed by Goyal *et al.* in [3]. After that, many cryptography researchers focused on KP-ABE and CP-ABE schemes [30–34].

In 2007, Abdalla *et al.* [35] proposed a traitor trace identity-based encryption scheme, and that was the original work on traitor tracing. In 2011, Wang *et al.* [36] presented another traitor tracing ABE scheme that can recognize a user's identity. This scheme could employ the technique of betrayer tracing and combine with a security coding technique to ensure the identity of the key abuser. Subsequently, Ning [37] proposed a traceable CP-ABE scheme; this scheme can catch malicious users effectively. A commitment scheme is used to trace defectors, but the method does not support the malicious user revocation.

To expand the commercial applications of ABE systems and combine them with user revocation mechanisms, Liu [38] presented a white-box traceable dynamic ABE scheme. The scheme can support user revocation and outsourcing decryption. However, it can neither resist collusion attacks between users nor support key and ciphertext updates. Jiang [39] proposed a traceable CP-ABE scheme that can resist key abuse. A betrayer who wants to leak a decryption key must abandon the whole key and give an exclusive dummy attribute set. Yang [40] presented a traceable scheme supporting search encryption and user revocation; it can perform efficient keyword search and provides fine-grained access control for encrypted data. At the same time, a large proportion of cryptographic computing is being outsourced to the cloud

server, and this alleviates the computational burden on end-user devices. Zhang *et al.* [41] used a composite order bilinear group to construct an effective white-box traceable CP-ABE scheme with a large universe and multiple authorities. Although this scheme can be used to trace malicious users and resist collusion attacks between users, it cannot support malicious user revocation.

1.3 Organization

The remainder of our paper is summarized as follows. We provide some necessary background knowledge that will be applied to our scheme in Section 2. We describe the system model and security model of our scheme in Section 3. We present a TUR-CPABE scheme and the proof of its security based on security games in Section 4 and 5, respectively. In Section 6, we provide a theoretical performance analysis of our scheme. Finally, we conclude the paper in Section 7.

2 Preliminaries

In this section, we give some necessary background information used in our scheme, including information about bilinear maps, binary trees, access structures and complexity assumptions.

2.1 Bilinear maps

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplication cyclic groups of prime order p , and g be a generator of $\mathbb{G}$. A map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map [42] with the following properties:

1. Bilinearity: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$;
2. Nondegeneracy: $e(g, g) \neq 1$;
3. Computability: There is an efficient polynomial time algorithm to compute the value of $e(u, v)$ for all $u, v \in \mathbb{G}$.

2.2 Access structure

Definition 1 (Access structure [43]). Let $P = \{P_1, P_2, \dots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^P$ is called monotone for $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An (monotone) access structure is a (monotone) collection $\mathbb{A}$ of non-empty subsets of P , i.e., $\mathbb{A} \subseteq 2^P \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, or else the sets are called the unauthorized sets. In this paper, we only consider the monotone access structure.

2.3 Linear secret sharing scheme

Definition 2 (linear secret sharing scheme (LSSS) [43]). A LSSS over a group of parties P can be defined as follows:

1. The shares for each party from a vector over $\mathbb{Z}_p$.
2. For a LSSS, there is a matrix M with l rows and n columns referred to as the share generating matrix. For $i \in [1, l]$, we define a function ρ that labels the i -th row of matrix M as attribute $\rho(i)$. We consider a column vector $v = (s, v_2, v_3, \dots, v_n)$, where $s \in \mathbb{Z}_p$ is a secret value to be shared and we choose random $v_2, v_3, \dots, v_n \in \mathbb{Z}_p$. Then Mv is the vector of l shares of the secret s , and the share $(Mv)_i$ belongs to attribute $\rho(i)$.

Each LSSS in the above definition also enjoys the linear reconstruction property, defined as follows: Let $\mathbb{A}$ be an access structure for the LSSS and $S \in \mathbb{A}$ be any authorized sets, and let $I \subseteq$

$\{1, 2, \dots, l\}$ be defined as $I = \{i: \rho(i) \in S\}$. There exists a set of constants $\{\omega_i | i \in I\}$ such that $\sum_{i \in I} \omega_i M_i = (1, 0, \dots, 0)$. Thus $\sum_{i \in I} \omega_i \lambda_i = s$ if $\lambda_i = M_i \cdot v$ is a valid share of any secret s .

2.4 Binary tree

Definition 3 (Binary tree [44]). We denote $\mathcal{U}$ be a group of users in the system, a user $u_k \in \mathcal{U}$ ($1 \leq k \leq |\mathcal{U}|$) and $G_x \subseteq \mathcal{U}$ be users group for attribute x . We describe the binary tree below:

1. We assign a user u_k for every leaf node and allocate a unique value v_j for every node in the tree.
2. We define $path(i)$ as a Dijkstra from the root node to the node i .
3. The minimal covering set $node(G_x)$ is the minimal set of nodes, and it is able to cover all of the leaf nodes that have users connected in G_x .
4. To consider the intersection of $path(u_k)$ and $node(G_x)$, we have $\beta_x = node(G_x) \cap path(u_k)$. As shown in Fig 1, we give an example: suppose that the users group for attribute x is $G_x = \{u_1, u_3, u_4, u_5, u_6, u_8\}$ and u_8 is a user connected with leaf node 14, then compute $node(G_x) = \{v_7, v_4, v_5, v_{14}\}$, $path(u_8) = \{v_0, v_2, v_6, v_{14}\}$, we have $\beta_x = \{v_{14}\}$.

2.5 Complexity assumptions

Now we briefly introduce the l -Strong Diffie-Hellman (l – SDH) assumption and the q -Bilinear Diffie-Hellman exponent (q – BDHE) assumption.

Assumption 1 (l – SDH assumption [24]). Let $\mathbb{G}$ be a bilinear group of prime order p and let g be a generator of $\mathbb{G}$. A l – SDH problem can be described as follows: Randomly choose exponent $x \in \mathbb{Z}_p^*$ and given a $l + 1$ -tuple $(g, g^x, g^{x^2}, \dots, g^{x^l})$, output a pair $(c, g^{1/(x+c)}) \in \mathbb{Z}_p \times \mathbb{G}$. Algorithm $\mathcal{A}$ can solve the l – SDH problem with the advantage ϵ if

$$\Pr \left| \mathcal{A}(g, g^x, g^{x^2}, \dots, g^{x^l}) = (c, g^{1/(x+c)}) \right| \geq \epsilon.$$

Definition 4. We say that the l – SDH assumption holds if no polynomial time algorithm $\mathcal{A}$ can solve the l – SDH problem with a non-negligible advantage.

Assumption 2 (q – BDHE assumption [4]). Let $\mathbb{G}$ be a bilinear group of prime order p and g be a generator of $\mathbb{G}$. A decisional q – BDHE problem can be described as follows: Randomly choose exponent $d, s \in \mathbb{Z}_p^*$, given

$$\vec{y} = (g, g^s, g^d, \dots, g^{dq}, g^{d^{q+2}}, \dots, g^{d^{2q}}).$$

It is difficult for the algorithm $\mathcal{A}$ to distinguish $e(g, g)^{d^{q+1}s} \in \mathbb{G}_T$ from the random element $Z \in \mathbb{G}_T$. Algorithm $\mathcal{A}$ can solve the q – BDHE problem with the advantage ϵ if

$$\left| \Pr[\mathcal{A}(\vec{y}, T = e(g, g)^{d^{q+1}s}) = 0] - \Pr[\mathcal{A}(\vec{y}, T = Z) = 0] \right| \geq \epsilon.$$

Definition 5. We say that the q – BDHE assumption holds if no polynomial time algorithm $\mathcal{A}$ has a non-negligible advantage in solving the q – BDHE problem.

3 System model

In this section, we first account the system architecture of TUR-CPABE and then provide the formal definition and security model of TUR-CPABE.

Fig 1. Binary tree.

<https://doi.org/10.1371/journal.pone.0203225.g001>

3.1 System architecture

As shown in Fig 2, the system architecture of the TUR-CPABE scheme has the following six entities.

Trust authority (TA): TA can produce the public parameters and the master key, and it is in charge of distributing private keys to users in the system. In our scheme, the TA is fully trusted.

Attribute manager (AM): The AM has users groups for every attribute, and it generates the manager public key, the manager master key and the group key for the users in each group. In addition, AM is responsible for re-encrypting local ciphertext, updating the key, updating the ciphertext and obtaining an attribute revocation list.

Data owner (DO): The DO takes charge of defining the access policy, encrypting data under the access policy and uploading the local ciphertext to the attribute manager.

Data user (DU): This is an entity who wants to access outsourced data. A user in the system can decrypt the ciphertexts successfully if and only if his/her identity is absent from the revocation list and his/her attributes can satisfy the access policy.

Cloud server provider (CSP): We suppose that a CSP is honest but curious. In other words, it can execute every authorization request honestly but it obtains as much information as possible in the process and from the result.

Outsourcing decryption server provider (ODSP): The server can provide outsourcing decryption service for the user to generate a partially decrypted ciphertext.

3.2 Formal definition

A TUR-CPABE involves eight algorithms: system setup, manager setup, key generation, encryption, decryption, key sanity check, trace and update.

Fig 2. System architecture of TUR-CPABE.

<https://doi.org/10.1371/journal.pone.0203225.g002>

System setup $(\lambda, U) \rightarrow (PP, MSK)$: TA runs this algorithm. The algorithm inputs the security parameter λ as well as the attributes universe U in the system, and it outputs the public parameters PP and the master key MSK .

Manager setup $(PP) \rightarrow (MPK, MMK)$: AM runs this algorithm. The algorithm inputs the public parameters PP , and it outputs the manager public key MPK and the manager master key MMK .

Key generation $(PP, MSK, MPK, MMK, S, id) \rightarrow SK$: In this algorithm, a decryption key SK consists of the user private key USK produced by the TA and the group key GSK generated by the AM. The algorithm inputs the public parameters PP , the master key MSK , the manager public key MPK , the manager master key MMK , a user's identity id as well as an attributes set S , then it outputs the user private key USK and the group key GSK .

Encryption $(PP, MPK, m(M, \rho)) \rightarrow (Hdr, CT')$: The algorithm inputs the public parameters PP , the manager public key MPK , a message m and an access policy (M, ρ) , then it outputs the

re-encrypted ciphertext CT' and the header of message Hdr . This algorithm is performed by the AM and the DO.

Decryption $(PP, CT', Hdr, SK) \rightarrow m$: This algorithm is carried out by the DU and the ODSP. The algorithm inputs the public parameters PP , the re-encrypted ciphertext CT' , the header of the message Hdr and the decryption key SK . The algorithm outputs plaintext m if and only if the user's identity is absent from the revocation list and his/her attributes can satisfy the access policy.

Key sanity check $(PP, SK) \rightarrow 1$ or 0 : TA performs this algorithm. If a user's decryption key SK is suspected, then it must be determined whether it is a well-formed decryption key by means of the key sanity check. If SK cannot pass the key sanity check, it outputs 0. Otherwise, it outputs 1.

Trace $(PP, SK) \rightarrow id$ or $\perp$: TA runs this algorithm. If a user's decryption key SK is a well-formed decryption key, then TA sends his/her identity id to the AM. Finally, the AM revokes the attribute for the malicious user.

Update $(CT', Hdr, GSK, RL) \rightarrow (\overline{CT'}, \overline{Hdr}, \overline{GSK}, RL')$: AM can perform attribute revocation for the malicious user, add the malicious user's id and revoke the attribute into the revocation list to get a new revocation list RL' . Finally, the AM inputs the group key GSK , the re-encrypted ciphertext CT' and the header of message Hdr to obtain the updated $\overline{CT'}$, $\overline{Hdr}$ and $\overline{GSK}$.

3.3 Security model

In this section, we propose the security model for our TUR-CPABE system.

3.3.1 Traceability. In this subsection, we present a traceability definition of our TUR-CPABE system. We describe it by considering a security game between an adversary $\mathcal{A}$ and a simulator $\mathcal{B}$ as follows:

Initialization: Simulator $\mathcal{B}$ executes System setup (λ, U) as well as Manager setup (PP) to obtain the public parameters PP and the manager public key MPK . After that, $\mathcal{B}$ sends PP as well as MPK to $\mathcal{A}$.

Key query: $\mathcal{A}$ submits a series of attributes sets $(id_1, S_1), \dots, (id_q, S_q)$ to request the corresponding decryption keys, where $(u_i, x) \in RL$ (it denotes revoke attribute x for user u_i whose identity is id_i) or $S_i \notin (M^*, \rho^*)(M^*, \rho^*)$ is a challenge access policy, $i = 1, 2, \dots, q$. Then simulator $\mathcal{B}$ performs the key generation algorithm and returns the result to $\mathcal{A}$.

Key forgery: Adversary $\mathcal{A}$ outputs a decryption key SK^* . If $\text{Trace}(PP, SK^*) \neq \perp$ (i.e., SK^* is a well-formed decryption key) and $\text{Trace}(PP, SK^*) \neq \{id_1, \dots, id_q\}$, then $\mathcal{A}$ wins the game. The advantage of $\mathcal{A}$ wins the game is defined as: $\Pr[\text{Trace}(PP, SK^*) \neq \{\perp, id_1, \dots, id_q\}]$.

Definition 6. A TUR-CPABE scheme is traceable if no polynomial time adversary has at most negligible advantage in this game.

3.3.2 IND-CPA security. In this subsection, we provide the IND-CPA security of our TUR-CPABE system. We describe it by a security game between an adversary $\mathcal{A}$ and a simulator $\mathcal{B}$ as follows:

Initialization: Adversary $\mathcal{A}$ chooses a revocation list RL^* as well as a challenge access policy (M^*, ρ^*) , where M^* is a $l^* \times n^*$ matrix with $n^* \leq q$. Then $\mathcal{A}$ sends them to $\mathcal{B}$.

Setup: Simulator $\mathcal{B}$ performs System setup (λ, U) as well as Manager setup (PP) to obtain the public parameters PP and the manager public key MPK , the master key MSK as well as the manager master key MMK . Finally $\mathcal{B}$ keeps MSK , MMK secret and sends PP , MPK to adversary $\mathcal{A}$.

Phase 1: $\mathcal{A}$ submits a series of tuples $(id_1, S_1), \dots, (id_q, S_q)$ to ask for corresponding to decryption keys.

1. If $S \in (M^*, \rho^*)$ and $(u, x) \notin RL^*$ (this denotes the revoke attribute x for the user u), the algorithm aborts.
2. If $S \notin (M^*, \rho^*)$ and $(u, x^*) \in RL^*$, simulator $\mathcal{B}$ generates a decryption key for each attributes set (id_i, S_i) and returns it to $\mathcal{A}$.

Challenge: Adversary $\mathcal{A}$ submits two equal length messages m_0, m_1 to $\mathcal{B}$. Then, $\mathcal{B}$ chooses $v \in \{0, 1\}$ randomly, encrypts the message m_v under the access policy (M^*, ρ^*) as well as computes the challenge ciphertext $\langle Hdr_v^*, CT_v^* \rangle$. Finally, $\mathcal{B}$ sends the challenge ciphertext to $\mathcal{A}$.

Phase 2: Same as Phase 1.

Guess: $\mathcal{A}$ outputs a guess v' of v . If $v = v'$, $\mathcal{A}$ wins the game. The advantage of adversary $\mathcal{A}$ wins the game is defined as: $Adv = |\Pr[v = v'] - 1/2|$.

Definition 7. If there is no polynomial time adversary who is able to break our scheme with a negligible advantage in this game, our scheme is said to be indistinguishable from a chosen plaintext attack under a selective access policy.

4 Construction of TUR-CPABE

In our scheme, user's decryption key consists of two parts. One is the user private key USK related to his/her attributes set, and the other is the group key GSK related to users group he/she belongs to. Only by combining the corresponding USK and GSK is the user able to decrypt the ciphertext. In the proposed scheme, the encryption algorithm also has two steps: First, the data owner encrypts the message to obtain the local ciphertext. Then, the attribute manager re-encrypts the local ciphertext to gain the re-encrypted ciphertext and the header of the message, and the attribute manager uploads them to the cloud server provider. A user can decrypt the ciphertext when and only when the user's identity id is absent from a revocation list and his/her attributes can satisfy the access policy. In addition, our decryption algorithm can be stated as follows: the outsourcing decryption server provider performs the outsourcing decryption algorithm and then sends the partially decrypted ciphertext to users; and users execute the local decryption algorithm to recover the plaintext.

System setup $(\lambda, U) \rightarrow (PP, MSK)$: The algorithm inputs the security parameter λ and the attributes universe $U = \{1, 2, \dots, x, \dots, n\}$. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplication cyclic groups of prime order p , g be a generator of $\mathbb{G}$. Function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map. As shown in Fig 1, $\mathcal{T}$ is a binary KEK tree, and for every leaf node in the tree to assign a user u whose identity is id . $RL = \{(id, x)\}$ is a revocation list (The initial is empty); this denotes the revoke attribute x for user u . The TA performs the following algorithms.

1. Randomly choose $\alpha, a \in \mathbb{Z}_p$ and $h \in \mathbb{G}$.
2. For every attribute $x \in U$, select $U_x \in \mathbb{G}$.
3. TA chooses a probabilistic encryption scheme (Enc, Dec) [45] from $\{0, 1\}^*$ to $\mathbb{Z}_p^*$, it is a symmetric encryption with secret key $\bar{k} \in \mathbb{Z}_p$, and the scheme encrypts the same plaintext generating different ciphertext each time.

Then, the TA sets the public parameters $PP = \langle p, e, \mathbb{G}, \mathbb{G}_T, g, h, g^a, e(g, g)^a, \{U_x\}_{x \in U} \rangle$ and the master key $MSK = \langle \alpha, a, \bar{k} \rangle$. Finally, the TA publishes PP and keeps the MSK secret.

Manager setup $(PP) \rightarrow (MPK, MMK)$: For every attribute $x \in U (1 \leq x \leq |U|)$, AM chooses a random number $w_x \in \mathbb{Z}_p$ and computes $W_x = g^{w_x}$. Then, it sets the manager public key as $MPK = \{W_x | 1 \leq x \leq |U|\}$ and the manager master key as $MMK = \{w_x | 1 \leq x \leq |U|\}$. Finally, AM publishes MPK and keeps MMK secret.

Key generation (PP, MSK, MPK, MMK, S, id) $\rightarrow SK$: In this algorithm, a decryption key SK consists of user private key USK produced by the TA and group key GSK generated by the AM. The specific steps are as follows:

1. **USK generation**: For the user $u_k \in \mathcal{U} (1 \leq k \leq |\mathcal{U}|)$, the TA authenticates his/her attributes set $S_k (S_k \subseteq U)$ and generates a USK_k connected with attributes set S_k . This algorithm can be stated as follows:

- a. For each attribute $x \in S_k (1 \leq x \leq |U|)$, the TA chooses $r', r'_x \in \mathbb{Z}_p$ randomly and computes $c = \text{Enc}_{\tau}(id_k)$, where the user u_k 's identity is id_k and there is no distinction between the result c and a random number in $\mathbb{Z}_p$. Then, the private key USK_k' is set as follows:

$$USK_k' = \langle K_1' = c, K_2' = g^{x/(a+c)} h^{r'}, L_1' = g^{r'}, L_2' = g^{ar'}, \{K_{x,1}' = g^{r'_x}, K_{x,2}' = U_x^{r'_x}\}_{x \in S_k} \rangle$$

- b. TA randomly selects $z \in \mathbb{Z}_p^*$ and sets the transformation key TK_k as

$$TK_k = \langle K_1' = c, K_2 = g^{x/z(a+c)} h^{r'/z} = g^{x/z(a+c)} h^r, L_1 = g^{r'/z} = g^r,$$

$$L_2 = g^{ar'/z} = g^{ar}, \{K_{x,1} = g^{r'_x/z} = g^{r_x}, K_{x,2} = U_x^{r'_x/z} = U_x^{r_x}\}_{x \in S_k} \rangle$$

and u_k 's user private key USK_k is set as $USK_k = (K_1 = z, TK_k)$.

- c. For every attribute $x \in S_k (1 \leq x \leq |U|)$, TA computes $kek_x = W_x^{r'_x/z} = W_x^{r_x}$ and adds (x, kek_x) into u_k 's group key GSK_k .
 - d. TA retains GSK_k , then it sends USK_k to the user and GSK_k to the AM by the secure channel, respectively.
2. **GSK generation**: Every node in the tree is assigned an exclusive value v_j and an exclusive sequence number $sequence(v_j)$. For the user $u_k \in \mathcal{U} (1 \leq k \leq |\mathcal{U}|)$, AM produces a group KEK_x that can compute path nodes from a leaf node to the root node. The detailed algorithm is as follows:

- a. For each attribute $x \in S_k (1 \leq x \leq |U|)$, AM computes a minimal covering set $node(G_x)$ for G_x and defines a Dijkstra $path(u_k)$ for the user $u_k \in \mathcal{U} (1 \leq k \leq |\mathcal{U}|)$, where G_x is the users group for attribute x .
 - b. For every attribute $x \in S_k (1 \leq x \leq |U|)$, AM executes an intersection operation $\beta_x = node(G_x) \cap path(u_k)$. If $\beta_x = \phi$, AM doesn't compute a KEK_x for the user u_k . Otherwise, it computes $KEK_x = (kek_x)^{1/v_j} = g^{w_x r_x / v_j}$, where $v_j \in \beta_x$. Then, AM adds $\{sequence(v_j), KEK_x\}$ into u_k 's group key GSK_k .
3. AM sends GSK_k to the user and G_x to the TA by the secure channel, respectively. Then, the user u_k obtains an unbroken decryption key $SK_k = \{USK_k, GSK_k\}$.

Encryption ($PP, MPK, m, (M, \rho)$) $\rightarrow (Hdr, CT')$: This algorithm has two steps. First, the user owner DO encrypts the message m to obtain the local ciphertext CT . Then, AM re-encrypts CT to produce the re-encrypted ciphertext CT' and the header of the message Hdr . The specific process is as follows:

1. **DO encrypts**: The algorithm inputs the public parameters PP , the message m and an access policy (M, ρ) . M is a matrix with $l \times n$ elements, and function ρ maps the rows of M into the attributes set. Then, the DO chooses a random column vector $v = (s, v_2, v_3, \dots, v_n)$, where $v_2, v_3, \dots, v_n$ are applied to share s . DO computes $\lambda_i = M_i \cdot v$ for $\forall i \in [1, l]$, where M_i is a vector

corresponding to the i -th row of matrix M . DO randomly chooses an exponent $\tau_i \in \mathbb{Z}_p$ and computes the local ciphertext CT as follows:

$$CT = \langle (M, \rho), C = m \cdot e(g, g)^{as}, C_0 = g^s, C_1 = g^{as},$$

$$\{C_{i,1} = h^{\lambda_i}, C_{i,2} = U_{\rho(i)}^{-\tau_i}, C_{i,3} = g^{\tau_i}\}_{i \in [1,l]} \rangle.$$

Then DO sends CT to the AM.

2. AM encrypts: For $\forall i \in [1, l]$, AM randomly chooses $k_i \in \mathbb{Z}_p$ and re-encrypts CT :

$$CT' = \langle (M, \rho), C = m \cdot e(g, g)^{as}, C_0 = g^s, C_1 = g^{as},$$

$$\{C_{i,1} = h^{\lambda_i}, C_{i,2}' = U_{\rho(i)}^{-\tau_i} \cdot g^{k_i}, C_{i,3} = g^{\tau_i}\}_{i \in [1,l]} \rangle.$$

For $\forall i \in [1, l]$, AM computes $node(G_{\rho(i)})$ and sets the header of the message Hdr as:

$$Hdr = \langle \forall i \in [1, l], \{sequence(v_j), E(k_i) = g^{k_i v_j / w_{\rho(i)}}\}_{v_j \in node(G_{\rho(i)})} \rangle.$$

Finally, AM uploads the ciphertext $\langle CT', Hdr \rangle$ to the CSP.

Decryption $(PP, CT', Hdr, SK) \rightarrow m$: This algorithm has two steps. First, ODSP executes the outsourcing decryption operation. Second, DU performs the local decryption algorithm. For the user $u_k \in \mathcal{U} (1 \leq k \leq |\mathcal{U}|)$, given the ciphertext $\langle CT', Hdr \rangle$ and the decryption key $SK_k = \{USK_k, GSK_k\}$, there are two cases:

1. If the user u_k 's attributes set $S_k \notin (M, \rho)$ (in other words, the user u_k 's attributes set S_k cannot satisfy the access policy (M, ρ)) or $u_k \in RL$, the algorithm aborts.
2. If the user u_k 's attributes set $S_k \in (M, \rho)$ and $u_k \notin RL$, let $I = \{i: \rho(i) \in S\}$ and $I \subseteq [1, l]$, there exists a set of constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in [1, l]}$ so that $\sum_{i \in I} \omega_i \lambda_i = s$. Then, DU sends the ciphertext $\langle CT', Hdr \rangle$, the transformation key TK_k and the group key GSK_k to the ODSP. Finally, ODSP sends the partially decrypted ciphertext CT_1 to the DU, then DU recovers the message m . The algorithms are stated as follows.

- a. ODSP decryption: First, the algorithm computes

$$\begin{aligned} B &= e(KEK_{\rho(i)}, E(k_i)) = e(g^{w_{\rho(i)} r_{\rho(i)} / v_j}, g^{k_i v_j / w_{\rho(i)}}) = e(g, g)^{k_i r_{\rho(i)}} \\ D &= e(L_1^{K_1'} L_2, C_{i,1}) e(K_{\rho(i),1}, C_{i,2}') e(K_{\rho(i),2}, C_{i,3}) = e(g, h)^{(a+c) r_{\lambda_i}} e(g, g)^{r_{\rho(i)} k_i} \\ E &= \prod_{i \in I} \left(\frac{D}{B}\right)^{\omega_i} = \prod_{i \in I} (e(g, h)^{(a+c) r_{\lambda_i}})^{\omega_i} = e(g, h)^{(a+c) rs} \\ F &= e(K_2, C_0^{K_1'} \cdot C_1) = e(g^{s/z(a+c)} h^r, g^{cs} \cdot g^{as}) = e(g, g)^{as/z} e(h, g)^{(a+c) rs} \end{aligned}$$

Then, the ODSP computes the partially decrypted ciphertext $CT_1 = F/E = e(g, g)^{as/z}$ and sends it to users.

- b. DU decryption: The algorithm inputs $CT_1 = e(g, g)^{as/z}$ and $C = m \cdot e(g, g)^{as}$, then recovers the message $m = C / (CT_1)^{K_1} = m \cdot e(g, g)^{as} / (e(g, g)^{as/z})^z$.

Key sanity check $(PP, SK) \rightarrow 1$ or 0 : If the user u_k 's decryption key SK_k is suspected, TA will check whether user private key USK_k satisfies the key sanity check:

1. USK_k is in the form of $USK_k = \langle K_1, K_1', K_2, L_1, L_2, \{K_{x,1}, K_{x,2}\}_{x \in S_k} \rangle$ and $K_1, K_1' \in \mathbb{Z}_p^*$, $K_2, L_1, L_2, K_{i,1}, K_{i,2} \in \mathbb{G}$.
2. $e(L_2, g) = e(L_1, g^a)$.
3. $e(K_2^{K_1}, g^a g^{K_1'}) = e(g, g)^a e(L_2 L_1^{K_1'}, h^{K_1})$.
4. $\exists x \in S_k, s.t. e(K_{x,2}, g) = e(U_x, K_{x,1}) \neq 1$.

If USK_k can pass the key sanity check, the algorithm outputs 1. Otherwise, it outputs 0.

Trace $(PP, SK) \rightarrow id$ or $\perp$: This algorithm is performed by the TA. If the user u_k 's user private key USK_k cannot pass the key sanity check, the algorithm outputs $\perp$. Otherwise, the algorithm should be done as follows:

1. It extracts the user u_k 's identity id_k from $Dec_K(K_1')$.
2. Search id_k from attribute x 's users group G_x . If TA can find id_k , the algorithm outputs the corresponding malicious user u_k . Otherwise, it outputs a user u^* who is never appear in G_x .
3. TA sends the malicious user's id_k to the AM.

Update $(CT', Hdr, GSK, RL) \rightarrow (\overline{CT'}, \overline{Hdr}, \overline{GSK}, \overline{RL})$: In an ABE scheme, a user has multiple attributes. Usually, each attribute can be shared by many users. Thus, the decryption key of an attribute can be shared by many users. When the user u_k 's attribute x is revoked, we can update other unrevoked users corresponding to GSK_x . In the meantime, we need to update the ciphertext related to this attribute to make sure the user's decryption key connected to the attribute is useless. Thus, the user u_k loses his/her decryption privilege. The attribute revocation algorithm includes the following steps.

1. Key update: In our system, AM performs the attribute revocation algorithm and updates the user's GSK_k . The particular steps are as follows:
 - a. For each tuple $(u_k, x) \in RL' (1 \leq k \leq |U|, 1 \leq x \leq |U|)$, AM randomly chooses $\delta_x \in \mathbb{Z}_p$ and for every tuple $(u_k, x) \notin RL' (1 \leq k \leq |U|, 1 \leq x \leq |U|)$, let $\delta_x = 1$. Then, for each attribute $x \in U (1 \leq x \leq |U|)$, AM computes $\overline{W}_x = W_x^{\delta_x}$. Finally, AM updates the manager public key as $\overline{MPK} = \{\overline{W}_x | 1 \leq x \leq |U|\}$, and updates the manager master key as $\overline{MMK} = \{\overline{w}_x = w_x \cdot \delta_x | 1 \leq x \leq |U|\}$.
 - b. For each tuple $(u_k, x) \in RL' (1 \leq k \leq |U|, 1 \leq x \leq |U|)$, AM updates the users group for attribute x as $\overline{G}_x$ and computes $node(\overline{G}_x)$.
 - c. For every tuple $(u_k, x) \notin RL' (1 \leq k \leq |U|, 1 \leq x \leq |U|)$, AM performs an intersection operation $\overline{\beta}_x = node(\overline{G}_x) \cap path(u_k)$. Then, it computes $\overline{kek}_x = (kek_x)^{\delta_x}$ and $\overline{KEK}_x = (\overline{kek}_x)^{1/\overline{v}_j}$, where $\overline{v}_j \in \overline{\beta}_x$.
 - d. AM replaces the group key GSK_k with the updated group key $\overline{GSK}_k = \{x, sequence(\overline{v}_j), \overline{kek}_x, \overline{KEK}_x\}$.
2. Ciphertext update: After updating the group key, AM continues executing the ciphertext update algorithm, and the algorithm is described as follows.

- a. First, AM randomly chooses an exponent $\bar{s} \in \mathbb{Z}_p$. Then AM selects a random $\bar{k}_i \in \mathbb{Z}_p$ for each tuple $(u, \rho(i)) \in RL'$ and updates CT' as

$$\overline{CT'} = \langle (M, \rho), \overline{C} = C \cdot e(g, g)^{\bar{s}}, \overline{C}_0 = C_0 \cdot g^{\bar{s}}, \overline{C}_1 = C_1 \cdot g^{a\bar{s}},$$

$$(u, \rho(i)) \in RL' : \{\overline{C}_{i,1} = C_{i,1} \cdot h^{\bar{s}}, \overline{C}_{i,2} = C_{i,2} \cdot g^{\bar{k}_i - k_i}, \overline{C}_{i,3} = C_{i,3}\}_{i \in [1, l]},$$

$$(u, \rho(i)) \notin RL' : \{\overline{C}_{i,1} = C_{i,1} \cdot h^{\bar{s}}, \overline{C}_{i,2} = C_{i,2}', \overline{C}_{i,3} = C_{i,3}\}_{i \in [1, l]} > .$$

- b. b. AM updates Hdr as

$$\overline{Hdr} = \langle \forall i \in [1, l], \text{if } (u, \rho(i)) \in RL', \{sequence(\overline{v}_j), E(\overline{k}_i) = g^{\overline{k}_i \overline{v}_j / w_{\rho(i)}}\}_{\overline{v}_j \in node(\overline{G_{\rho(i)}})} \rangle$$

$$\text{if } (u, \rho(i)) \notin RL', \{sequence(v_j), E(k_i) = g^{k_i v_j / w_{\rho(i)}}\}_{v_j \in node(G_{\rho(i)})} > .$$

5 Security analysis

In this section, we first provide a proof of traceability based on the $l - SDH$ hardness assumption. Then, we prove that our scheme is able to achieve $IND - CPA$ security if the $q - BDHE$ assumption holds.

5.1 Traceability

Theorem 1. Suppose that $q < l$, our TUR-CPABE system is traceable if $l - SDH$ assumption holds. Where q is the number of key queries that the attacker makes.

Proof: Suppose there is a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ capable of winning this traceability game with advantage ϵ , w.l.o.g., suppose $l = q + 1$, we can establish a PPT algorithm $\mathcal{B}$ to break the $l - SDH$ hardness problem with a non-negligible advantage.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplication cyclic groups of prime order p , let g be a generator of $\mathbb{G}$ and let function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map. The algorithm $\mathcal{B}$ receives a $l - SDH$ challenge problem $(\bar{g}, \bar{g}^a, \bar{g}^{a^2}, \dots, \bar{g}^{a^l})$, where $a \in \mathbb{Z}_p^*$, and the goal of $\mathcal{B}$ is to output a tuple $(c_r, \omega_r = \bar{g}^{1/(a+c_r)})$. Let $A_i = \bar{g}^{a^i}$ ($i = 0, 1, \dots, l$). To solve the $l - SDH$ problem, algorithm $\mathcal{B}$ can imitate a challenger's role for adversary $\mathcal{A}$. The specific processes are stated below:

Setup: Algorithm $\mathcal{B}$ selects q different values $c_1, c_2, \dots, c_q \in \mathbb{Z}_p^*$ randomly. Let

$f(y) = \prod_{i=1}^q (y + c_i) = \sum_{i=0}^q \alpha_i y^i$, where $\alpha_0, \alpha_1, \dots, \alpha_q \in \mathbb{Z}_p$ are the coefficient of polynomial $f(y)$. Then $\mathcal{B}$ performs the following algorithm:

1. Let $g = \prod_{i=0}^q (A_i)^{z_i} = \bar{g}^{f(a)}$ and $g^a = \prod_{i=0}^q (A_i)^{z_{i-1}} = \bar{g}^{f(a) \cdot a}$.
2. $\mathcal{B}$ randomly picks $\alpha, \theta \in \mathbb{Z}_p, h \in \mathbb{G}$. For each attribute $x \in U$, $\mathcal{B}$ chooses a random number $u_x \in \mathbb{Z}_p$ and establishes $U_x = g^{u_x}$. Finally, $\mathcal{B}$ publishes the public parameters as $PP = \langle p, e, \mathbb{G}, \mathbb{G}_T, g, h = g^\theta, g^a, e(g, g)^\alpha, \{U_x\}_{x \in U} \rangle$.
3. For each $x \in U$, $\mathcal{B}$ randomly chooses $w_x \in \mathbb{Z}_p$ and computes $W_x = g^{w_x}$, then publishes the manager public key as $MPK = \{W_x | 1 \leq x \leq |U|\}$.
4. Set up a binary KEK tree and assign a user u for every leaf node, and the user's identity is id . Every node possesses an exclusive value v_j as well as an exclusive sequence number $sequence(v_j)$ in the binary KEK tree.

Key query: Adversary $\mathcal{A}$ submits a set of attributes (id_i, S_i) to $\mathcal{B}$ and requests the corresponding decryption keys. When it goes on the i -th query, we suppose $i \leq q$, let polynomial $f_i(y) = \frac{f(y)}{y+c_i}$, we can write $f_i(y)$ as $f_i(y) = \sum_{j=0}^{q-1} \beta_j y^j$. $\mathcal{B}$ computes

$$\sigma_i = \prod_{j=0}^{q-1} (A_j)^{\beta_j} = \prod_{j=0}^{q-1} (\bar{g}^x)^{\beta_j} = \bar{g}^{f(a)} = \bar{g}^{f(a)/(a+c_i)} = g^{1/(a+c_i)},$$

and $\mathcal{B}$ randomly selects $z, r', r'_x \in \mathbb{Z}_p$, computes DSK_i related to (id_i, S_i) as follows: First, $\mathcal{B}$ computes

$$DSK_i' = \langle K_1' = c_i, K_2' = (\sigma_i)^z h^{r'} = g^{z/(a+c_i)} h^{r'}, L_1' = g^{r'} \rangle,$$

$$L_2' = g^{ar'}, \{K_{x,1}' = g^{r'_x}, K_{x,2}' = (U_x)^{r'_x} = U_x^{r'_x}\}_{x \in S_i} > .$$

Then $\mathcal{B}$ sets the transformation key TK_i as

$$TK_i = \langle K_1' = c_i, K_2 = g^{z/(a+c_i)} h^{r'/z} = g^{z/(a+c_i)} h^r, L_1 = g^{r'/z} = g^r \rangle,$$

$$L_2 = g^{ar'/z} = g^{ar}, \{K_{x,1} = g^{r'_x/z} = g^{r_x}, K_{x,2} = U_x^{r'_x/z} = U_x^{r_x}\}_{x \in S_i} > ,$$

and builds the user private key as $USK_i = (K_1 = z, TK_i)$. Finally, $\mathcal{B}$ computes $kek_x = W_x^{r'_x/z} = W_x^{r_x}$.

To define a function $node(G_x)$ for attribute x 's users group G_x , where $x \in S_i$. For every user $u_i \in \mathcal{U} (1 \leq i \leq |\mathcal{U}|)$, $\mathcal{B}$ defines a function $path(u_i)$ and performs an intersection operation $\beta_x = node(G_x) \cap path(u_i)$. If $\beta_x = \phi$, $\mathcal{B}$ sets $GSK_i = \{x, kek_x\}$. Otherwise, $\mathcal{B}$ computes $KEK_x = (kek_x)^{1/v_j} = g^{w_x r_x / v_j}$, where $v_j \in \beta_x$. Then, $\mathcal{B}$ sets the group key as $GSK_i = \{x, sequence(v_j), kek_x, KEK_x\}$.

Finally, $\mathcal{B}$ returns the decryption key $SK_i = \{DSK_i, GSK_i\}$ to $\mathcal{A}$.

Key forgery: Adversary $\mathcal{A}$ submits a forged decryption key SK^* to $\mathcal{B}$. Here the distribution of the decryption key SK and the public parameters PP in the above game are the same as in the real system.

Let $\varepsilon_{\mathcal{A}}$ denote the event that $\mathcal{A}$ wins the game, i.e., SK^* can pass the key sanity check and $K_1' \notin \{c_1, c_2, \dots, c_q\}$. If the event $\varepsilon_{\mathcal{A}}$ does not happen, $\mathcal{B}$ chooses a random tuple $(c_r, \omega_r) \in \mathbb{Z}_p \times \mathbb{G}$ as a solution to the l -SDH hardness problem. If event $\varepsilon_{\mathcal{A}}$ takes place, $\mathcal{B}$ writes the polynomial f as $f(y) = \gamma(y)(y + K_1') + \gamma - 1$, where $\gamma(y) = \sum_{i=0}^{q-1} (\gamma_i y^i)$, $\gamma - 1 \in \mathbb{Z}_p^*$. Note that $\gamma - 1 \neq 0$, since $f(y) = \prod_{i=1}^q (y + c_i)$, $c_i \in \mathbb{Z}_p^*$ and $K_1' \notin \{c_1, c_2, \dots, c_q\}$, $f(y)$ cannot be divisible by $y + K_1'$. $\mathcal{B}$ computes the tuple $(c_r, \omega_r) \in \mathbb{Z}_p \times \mathbb{G}$ as follows:

Suppose $L_1 = g^r$, where $r \in \mathbb{Z}_p$ is unknown, and let $K_1 = z$, where $z \in \mathbb{Z}_p^*$. According to the equality $e(L_2, g) = e(L_1, g^a)$ from the key sanity check, we have $L_2 = g^{ar}$. On the basis of the equality $e(K_2^{K_1}, g^a g^{K_1'}) = e(g, g)^z e(L_2 L_1^{K_1'}, h^{K_1})$, we have $K_2 = g^{z/(a+K_1')} h^r$. Then $\mathcal{B}$ continues performing the following algorithm.

$$\sigma = (K_2 / L_1^0)^{K_1'^{-1}} = g^{1/(a+K_1')} = \bar{g}^{f(a)/(a+K_1')} = \bar{g}^{\gamma(a)} \bar{g}^{(\gamma-1)/(a+K_1')}$$

$$\omega_r = (\sigma \cdot \prod_{i=0}^{q-1} A_i^{-\gamma_i})^{1/(\gamma-1)} = (\bar{g}^{\gamma(a)} \bar{g}^{(\gamma-1)/(a+K_1')} \cdot \bar{g}^{-\gamma(a)})^{1/(\gamma-1)} = \bar{g}^{1/(a+K_1')}$$

$$c_r = K_1' \bmod p \in \mathbb{Z}_p.$$

As $e(\bar{g}^a \cdot \bar{g}^{c_r}, \omega_r) = e(\bar{g}^a \cdot \bar{g}^{K_1'}, \bar{g}^{1/(a+K_1')}) = e(\bar{g}, \bar{g})$, the tuple (c_r, ω_r) is a solution to l -SDH hardness problem.

At present, we assess the superiority of $\mathcal{B}$ to break the l -SDH hardness problem.

Suppose ζ denotes the event that (c_r, ω_r) is the solution to the l -SDH hardness problem and this solution can be checked by verifying whether the equality $e(\bar{g}^a \cdot \bar{g}^{c_r}, \omega_r) = e(\bar{g}, \bar{g})$ holds. When $\mathcal{B}$ randomly selects (c_r, ω_r) , ζ can happen with a negligible advantage. We denote this as 0 for simplicity. When the event $\mathcal{A}win \wedge \gcd(\gamma - 1, p) = 1$ occurs, $\mathcal{B}$ outputs a tuple (c_r, ω_r) and the probability of tuple (c_r, ω_r) satisfies the equality $e(\bar{g}^a \cdot \bar{g}^{c_r}, \omega_r) = e(\bar{g}, \bar{g})$ is 1. Hence, the possibility of $\mathcal{B}$ solves l -SDH challenge problem is as follows:

$$\begin{aligned} \Pr[\zeta] &= \Pr[\zeta | \overline{\mathcal{A}win}] \cdot \Pr[\overline{\mathcal{A}win}] + \Pr[\zeta | \mathcal{A}win \wedge \gcd(\gamma - 1, p) \neq 1] \cdot \Pr[\mathcal{A}win \wedge \gcd(\gamma - 1, p) \neq 1] \\ &\quad + \Pr[\zeta | \mathcal{A}win \wedge \gcd(\gamma - 1, p) = 1] \cdot \Pr[\mathcal{A}win \wedge \gcd(\gamma - 1, p) = 1] \\ &= 0 + 0 + \Pr[\mathcal{A}win \wedge \gcd(\gamma - 1, p) = 1] \\ &= \Pr[\mathcal{A}win] \cdot \Pr[\gcd(\gamma - 1, p) = 1] \\ &= \varepsilon. \end{aligned}$$

5.2 IND – CPA security of the TUR-CPABE

Theorem 2. If the decisional q -BDHE assumption holds, then there are no PPT adversaries that have non-negligible advantages in breaking our TUR-CPABE scheme under selective access policy and chosen plaintext attacks, where $q > 2|\mathcal{U}| - 2$ and $|\mathcal{U}|$ is the number of users in the system.

Proof: Suppose there is a PPT adversary $\mathcal{A}$ able to break our TUR-CPABE scheme with an advantage ε . In this case, we could set up a simulator $\mathcal{B}$ who has an advantage $\varepsilon/2$ to break the decisional q -BDHE hardness problem. The simulation processes are described as follows:

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplication cyclic groups of prime order p , g be a generator of $\mathbb{G}$ and function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map. Given $\vec{y} = (g, g^s, g^d, \dots, g^{d^q}, g^{d^{q+2}}, \dots, g^{d^{2q}})$ then simulator $\mathcal{B}$ casts a fair coin μ . If $\mu = 0$, $\mathcal{B}$ sets $T = e(g, g)^{d^{q+1}s}$. Otherwise, $\mathcal{B}$ sets $T = Z$, where Z is a random element in $\mathbb{G}_T$.

Initialization: Adversary $\mathcal{A}$ chooses a challenge access policy (M^*, ρ^*) as well as a revocation list RL^* , where M^* is a $l^* \times n^*$ matrix and $n^* \leq q$.

Setup: To simulate public parameters as well as manager public key, simulator $\mathcal{B}$ needs to execute the following algorithms.

1. $\mathcal{B}$ chooses a random $\alpha' \in \mathbb{Z}_p^*$ such that $e(g, g)^{\alpha} = e(g, g)^{\alpha'} e(g^d, g^{d^q})$, where implicit sets $\alpha = \alpha' + d^{q+1}$.
2. Randomly select a value $z_x \in \mathbb{Z}_p$ for every attribute $x \in U(1 \leq x \leq |U|)$, then every group element $U_x \in \mathbb{G}$ is generated as follows. If there exists $i \in \{1, 2, \dots, l^*\}$ such that $\rho^*(i) = x$, set $U_x = g^{z_x} g^{dM_{i,1}^*} g^{d^2M_{i,2}^*} \dots g^{d^{n^*}M_{i,n^*}^*}$. Otherwise, let $U_x = g^{z_x}$.
3. $\mathcal{B}$ randomly chooses $a \in \mathbb{Z}_p$, then computes g^a and sets $h = g^d$.
4. Given a revocation list RL^* . If $(u, x) \in RL^*$, $\mathcal{B}$ randomly chooses $\theta_x \in \mathbb{Z}_p^*$ and sets $W_x = (g^d)^{\theta_x} = g^{d\theta_x}$, where implicit sets $w_x = d\theta_x$. Otherwise, builds $W_x = g^{\theta_x}$, where $w_x = \theta_x$.

- Set up a binary *KEK* tree and assign a user u for every leaf node, and the user's identity is id . Every node has an exclusive value v_j as well as an exclusive sequence number $sequence(v_j)$ in the binary *KEK* tree.

Hence, the simulator publishes the public parameters as

$$PP = \langle p, e, \mathbb{G}, \mathbb{G}_T, g, h, g^a, e(g, g)^z, \{U_x\}_{x \in U} \rangle$$

and the manager public key as $MPK = \{W_x | 1 \leq x \leq |U|\}$.

Phase 1: Adversary $\mathcal{A}$ commits a sequence of tuples $(u_1, S_1), \dots, (u_q, S_q)$ to ask for the corresponding decryption keys, where (u, x) denotes revoke the user u 's attribute x . $\mathcal{B}$ does the following in response:

Case 1. If $S \in (M^*, \rho^*)$ and $(u, x) \notin RL^*$, the algorithm aborts.

Case 2. If $S \in (M^*, \rho^*)$ and $(u, x^*) \in RL^*$, $\mathcal{B}$ chooses a random $c, r_x' \in \mathbb{Z}_p$ and implicit sets $r' = -\frac{d^q}{a+c} + \frac{d^{q-1}}{a+c} \cdot \frac{M_{i,1}^*}{M_{i,2}^*}$. Then, the simulator performs the following algorithm to setting decryption key.

- $\mathcal{B}$ computes USK' as follows:

$$K_1' = c$$

$$K_2' = g^{z/(a+c)} (g^{d^q/(a+c)})^{M_{x,1}^*/M_{x,2}^*} = g^{z/(a+c)} h^r$$

$$L_1' = [(g^{d^q})^{1/(a+c)}]^{-1} [(g^{d^{q-1}})^{1/(a+c)}]^{M_{x,1}^*/M_{x,2}^*} = g^{r'}$$

$$L_2' = (L)^a = g^{ar'}$$

$$K_{x,1}' = g^{r_x'}$$

$$K_{x,2}' = g^{r_x' z} \prod_{j=1}^{n^*} g^{d^j M_{ij}^* r_x'} = (g^{z x} \prod_{j=1}^{n^*} g^{d^j M_{ij}^*})^{r_x'} = U_x^{r_x'}.$$

- Choose a random exponent $z \in \mathbb{Z}_p$, then let $K_1 = z$ and set TK as

$$TK = \langle K_1' = c, K_2 = (K_2')^{1/z} = g^{z/(a+c)} h^r, L_1 = (L_1')^{1/z} = g^r, L_2 = (L_2')^{1/z} = g^{ar'},$$

$$\{K_{x,1} = (K_{x,1}')^{1/z} = g^{r_x}, K_{x,2} = (K_{x,2}')^{1/z} = U_x^{r_x}\}_{x \in S} \rangle.$$

Finally, obtain $USK = (z, TK)$.

- For have been revoked attribute x^* , $\mathcal{B}$ selects the value $v^* \in path(u)$, then computes $kek_{x^*} = (g^{r_{x^*}'/z})^{d\theta_{x^*}} = (g^{r_{x^*}'})^{d\theta_{x^*}} = g^{w_{x^*} r_{x^*}}$ and $KEK_{x^*} = (g^{r_{x^*}'/z})^{d\theta_{x^*}/v^*} = g^{w_{x^*} r_{x^*}/v^*}$. Otherwise, $\mathcal{B}$ computes $kek_x = (g^{r_x'/z})^{\theta_x} = g^{w_x r_x}$, then computes a minimal covering set $node(G_x)$ for the attribute x 's users group G_x and defines a Dijkstra $path(u)$ for the user u , finally performs an intersection operation $\beta_x = node(G_x) \cap path(u)$. If $\beta_x = \phi$, $\mathcal{B}$ does not compute KEK_x for u . Otherwise, $\mathcal{B}$ computes $KEK_x = (kek_x)^{1/v_j} = g^{w_x r_x/v_j}$, where $v_j \in \beta_x$.

Case3. If $S \notin (M^*, \rho^*)$ and $(u, x^*) \in RL^*$, $\mathcal{B}$ responds in the following.

1. Find a vector $\vec{\omega} = (\omega_1, \omega_2, \dots, \omega_{n^*}) \in \mathbb{Z}_p^{n^*}$ such that $\omega_1 = -1$ and $M_i^* \cdot \vec{\omega} = 0$, where $\rho^*(i) \in S$. As $S \notin (M^*, \rho^*)$, such a vector must exist.
2. $\mathcal{B}$ selects a random number $c, r_x' \in \mathbb{Z}_p$, builds $K_1' = c$.
3. $\mathcal{B}$ chooses a random number $t \in \mathbb{Z}_p$, implicit defines

$$r' = \frac{1}{a+c} (t + \omega_1 d^q + \omega_2 d^{q-1} + \dots + \omega_{n^*} d^{q-n^*+1}).$$

4. $\mathcal{B}$ computes $K_2', L_1', L_2', K_{x,1}'$ are as follows:

$$K_2' = g^{(x'+dt)/(a+c)} \left(\prod_{i=2}^{n^*} g^{\omega_i d^{q+2-i}} \right)^{1/(a+c)} = g^{x/(a+c)} h^{r'}$$

$$L_1' = g^{t/(a+c)} \left(\prod_{i=1}^{n^*} g^{\omega_i d^{q+1-i}} \right)^{1/(a+c)} = g^{r'}$$

$$L_2' = g^{at/(a+c)} \left(\prod_{i=1}^{n^*} g^{\omega_i d^{q+1-i}} \right)^{a/(a+c)} = g^{ar'}$$

$$K_{x,1}' = g^{r_x'}.$$

For $\forall x \in S$, $\mathcal{B}$ computes $K_{x,2}'$. If there no exists i such that $\rho^*(i) = x$, $\mathcal{B}$ computes

5. $K_{x,2}' = g^{z r_x'}$. Otherwise, sets $K_{x,2}' = g^{z r_x'} \left(\prod_{j=1}^{n^*} g^{d^{ij} M_{ij}^*} \right)^{r_x'}$.

$\mathcal{B}$ chooses a random exponent $z \in \mathbb{Z}_p^*$, then builds $K_1 = z$. Compute the transformation key as

$$TK = \langle K_1' = c, K_2 = (K_2')^{1/K_1} = g^{x/z(a+c)} h^{r'/z} = g^{x/z(a+c)} h^r, L_1 = g^{r'/z} = g^r,$$

$$L_2 = g^{ar'/z} = g^{ar}, \{K_{x,1} = g^{r_x'/z} = g^{r_x}, K_{x,2} = (K_{x,2}')^{1/z}\}_{x \in S} \rangle$$

and set $USK = (K_1 = z, TK)$. Then, $\mathcal{B}$ computes kek_{x^*} and KEK_{x^*} as in case 2.

Case4. If $S \notin (M^*, \rho^*)$ and $(u, x) \notin RL^*$, the simulator computes USK as in case 3. First, $\mathcal{B}$ computes $kek_x = (g^{r_x'/z})^{\theta_x} = g^{r_x \theta_x} = g^{w_x r_x}$. Then, $\mathcal{B}$ computes a minimal covering set $node(G_x)$ for the attribute x 's users group G_x and defines a Dijkstra $path(u)$ for the user u . Finally, $\mathcal{B}$ performs an intersection operation $\beta_x = node(G_x) \cap path(u)$. If $\beta_x = \emptyset$, $\mathcal{B}$ doesn't compute KEK_x for u . Otherwise, $\mathcal{B}$ computes $KEK_x = (kek_x)^{1/v_j} = g^{w_x r_x / v_j}$, where $v_j \in \beta_x$.

Challenge: Finally, we set the challenge ciphertext. Adversary $\mathcal{A}$ submits two equal length messages m_0, m_1 to $\mathcal{B}$. Then, $\mathcal{B}$ casts a fair coin $v \in \{0, 1\}$ and performs the following algorithms.

Local encryption: First, $\mathcal{B}$ sets $C = m_v \cdot T \cdot e(g^x, g^s)$, $C_0 = g^s$, $C_1 = (g^s)^a$. Then, $\mathcal{B}$ randomly chooses $y_2', y_3', \dots, y_{n^*}' \in \mathbb{Z}_p^*$ and builds

$$\vec{v} = (s, sd + y_2', sd^2 + y_3', \dots, sd^{n^*-1} + y_{n^*}') \in \mathbb{Z}_p^n.$$

Third, $\mathcal{B}$ computes

$$C_{i,1} = \prod_{j=2}^{n^*} ((g^d)^{y_j'})^{M_{ij}^*} \prod_{j=1}^{n^*} ((g^s)^{d^j})^{M_{ij}^*},$$

$$C_{i,2} = (g^{z_{\rho^*(i)}})^{-ad^i} (\prod_{j=1}^{n^*} g^{d^j M_{ij}^*})^{-ad^i},$$

$$C_{i,3} = g^{ad^i}.$$

Finally, $\mathcal{B}$ sets the local ciphertext as $CT^* = \langle (M^*, \rho^*), C, C_0, C_1, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i \in [1, l^*]} \rangle$.

Re-encryption: For each non-revoked attribute $\rho^*(i) (1 \leq i \leq l^*)$, $\mathcal{B}$ randomly chooses $k_i \in \mathbb{Z}_p$. Otherwise, $\mathcal{B}$ sets $k_i^* = d \cdot k_i$. Then, $\mathcal{B}$ re-encrypts the local ciphertext to obtain

$$\overline{CT_v^*} = \langle (M^*, \rho^*), C, C_0, C_1,$$

$$\rho^*(i) \neq x^*, \{C_{i,1}, C_{i,2} = (g^{z_{\rho^*(i)}})^{-ad^i} (\prod_{j=1}^{n^*} g^{d^j M_{ij}^*})^{-ad^i} \cdot g^{k_i}, C_{i,3}\}_{i \in [1, l^*]}$$

$$\rho^*(i) = x^*, \{C_{i,1}, C_{i,2} = (g^{z_{\rho^*(i)}})^{-ad^i} (\prod_{j=1}^{n^*} g^{d^j M_{ij}^*})^{-ad^i} \cdot g^{k_i^*}, C_{i,3}\}_{i \in [1, l^*]} \rangle.$$

In addition, for every $\rho^*(i) (1 \leq i \leq l^*)$ corresponding to the attribute x , $\mathcal{B}$ computes a minimal covering set $node(G_x)$ for x 's users group G_x . Then, $\mathcal{B}$ sets the header of message as

$$Hdr_v^* = \{\forall i \in [1, l^*] : \rho^*(i) \neq x^*, \langle sequence(v_j), E(k_i) = g^{k_i v_j / w_{\rho^*(i)}} \rangle_{v_j \in node(G_x)}\}$$

$$\rho^*(i) = x^*, \langle sequence(v^*), E(k_i^*) = g^{k_i^* v^* / w_{\rho^*(i)}} \rangle_{v^* \in node(G_{x^*})}\}.$$

Finally, $\mathcal{B}$ sends the challenge ciphertext $\langle Hdr_v^*, CT_v^* \rangle$ to $\mathcal{A}$.

Phase 2: Same as Phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess v' of v . If $v' = v$, the simulator outputs $\mu' = 0$ to show that T is a valid q -BDHE tuple. Otherwise, $\mathcal{B}$ outputs $\mu' = 1$ to indicate that T is a random group element from $\mathbb{G}_T$.

The decryption key and public parameters generated by simulations in the above game are the same as those in the real system.

When $\mu = 1$, the adversary cannot acquire any information about v . Therefore, we have $\Pr[v \neq v' | \mu = 1] = \frac{1}{2}$. $\mathcal{B}$ guesses $\mu' = 1$ as $v' \neq v$, we have $\Pr[\mu = \mu' | \mu = 1] = \frac{1}{2}$.

When $\mu = 0$, $\mathcal{A}$ receives m_v 's ciphertext. If the advantage of $\mathcal{A}$ is ϵ , we have $\Pr[v = v' | \mu = 0] = \frac{1}{2} + \epsilon$. $\mathcal{B}$ guesses $\mu' = 0$ as $v' = v$, and we have $\Pr[\mu = \mu' | \mu = 0] = \frac{1}{2} + \epsilon$.

The advantage of simulator $\mathcal{B}$ in the decisional q -BDHE game is defined as:

$$Adv = \left| \frac{1}{2} \Pr[\mu = \mu' | \mu = 0] + \frac{1}{2} \Pr[\mu = \mu' | \mu = 1] - \frac{1}{2} \right| = \left| \frac{1}{2} \left(\frac{1}{2} + \epsilon \right) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} \right| = \frac{1}{2} \epsilon.$$

6 Performance Analysis

In this section, our scheme is compared with several related schemes in terms of functionality and performance. The comparisons are listed in Tables 1 and 2. Our experiment is realized by using of the Pairing Cryptography (PBC) library [46]. Our pairing is structured on an ellipse curve $y^2 = x^3 + x$ in a finite field $\mathbb{F}_q$ (q is a prime number and $q \equiv 3 \pmod{4}$). The environment

Table 1. Functionality comparisons.

Scheme	Resist collusion	User revocation	Attribute revocation	Outsourcing decryption	Key update	Ciphertext update
[25]	×	×	×	×	×	×
[37]	×	×	×	×	×	×
[26]	✓	✓	×	×	×	×
[38]	×	✓	×	✓	×	×
[41]	✓	×	×	×	×	×
ours	✓	✓	✓	✓	✓	✓

✓: The scheme has the function.

×: The scheme does not have this function.

<https://doi.org/10.1371/journal.pone.0203225.t001>

of the hardware runtime is Intel Core i5-3470 CPU @ 3.20GHz, and RAM is 4.00GB. The software runtime environment is JDK 1.7.5, JPBC 2.0.0 and MyEclipse 10.

We compared our scheme with the other schemes [25,26,37,38,41] in Table 1. Those scheme can support traceability, and schemes [25,37,41] cannot support malicious user revocation. To achieve ABE fine-grained access control, our scheme can sustain the attribute level user revocation. In addition, we find that schemes [26,41] and our scheme both can resist collusion attacks between users. Only scheme [38] and our scheme can support an outsourcing decryption algorithm. Finally, Table 1 shows that our scheme can obtain a key update algorithm and a ciphertext update algorithm, and schemes [25,26,37,38,41] do not possess those functionalities.

From Table 2, we can find that our scheme is more efficient than those in [25,26,37,38,41] for the encryption algorithm. As our scheme only conducts one exponentiation operation and one multiplication operation in the decryption algorithm, our scheme and scheme [38] are much better than are other schemes [25,26,37,41]. In the trace algorithm, it is obvious that the schemes in [25,37,38] are less efficient than our scheme, and due to the cost of multiplication operations, it is much less expensive than a bilinear pairing operation so that our scheme is slightly lower in efficiency than that of the scheme [41]. Although our scheme is more efficient

Table 2. Performance comparisons.

Scheme	Key gen	Encryption	Decryption	Trace
[25]	$(5 + 3S)E + (1 + 2S)M$	$(3 + 5l)E + P + 2lM$	$(1 + 3n)P + 3E + 2M + (1 + 2n)M_T$	$(5 + 3S)P + (1 + S)E + (2 + 3S)M + M_T$
[37]	$(10 + S)E + 4M$	$(4 + 3l)E + P + lM$	$(2 + 2n)P + 3E + 3M + (2 + n)M_T$	$(7 + 2S)P + E + M + 2M_T$
[26]	$(7 + 3S)E + (4 + S)M$	$(11 + 5l)E + P(4 + 2l)M$	$(5 + 3n)P + E + 5M_T$	-
[38]	$(5 + 3S)E + (1 + 2S)M$	$(3 + 5l)E + P + 2lM$	$M_T + E$	$(5 + 2S)P + 5E + 3M + 2SM_T$
[41]	$(4 + S)E + (1 + S)M$	$8lE + P + E + l(M + M_T)$	$3n(P + M_T) + 2E + M_T + 2n(M + E) + M$	$(3 + 2S)P + 3E + 2M + (1 + S)M_T$
Ours	$(4 + 3S)E + M$	$(3 + 3l)E + P$	$M_T + E$	$(4 + 2S)P + 4E + M_T$

E : an exponent operation in $\mathbb{G}, \mathbb{G}_T$

P : a bilinear pairing operation

M : a multiplication operation in $\mathbb{G}$

M_T : a multiplication operation in $\mathbb{G}_T$

S : the number of attributes in the system

l : the number of rows in the access policy

n : the number of attributes that satisfy the access policy related to the ciphertext in the decryption key.

<https://doi.org/10.1371/journal.pone.0203225.t002>

Fig 3. Comparisons of computational overhead. (a)Key generation time (b)Encryption time (c)Decryption time (d)Trace time.

<https://doi.org/10.1371/journal.pone.0203225.g003>

than are those schemes in [25,26,38,41] for the key generation algorithm, it is slightly less efficient than scheme [37]. It is worth noting that our scheme can perform the attribute-level user revocation and can resist collusion attacks with a lower key generation expenditure.

Fig 3 compares the computational overheads in key generation time, encryption time, decryption time and trace time. Fig 3(A) compares key generation times between our scheme and the above schemes. We find that the key generation time in the proposed scheme is much less than that in other schemes [25,26,38,41]. Fig 3(B) shows the time required for the data owner to encrypt a message. Our scheme takes much less time than that of the others [25,26,37,38,41]. Fig 3(C) shows the time required for the data user to decrypt a message. Because our scheme and scheme [38] make use of outsourcing decryption algorithm, the decryption time is a constant. Compared with schemes [25,26,37,41], our scheme has an obvious advantages in the decryption time. Fig 3(D) compares the trace times in the above scheme. Our scheme's trace time is much less than those of other schemes [25,37,38]. Compared with scheme [41], although our scheme does not have a clear advantage in its trace time, our scheme can achieve user revocation and can resist user collusion. In brief, the results of our experiment agree with the above theoretical analysis.

7 Conclusion

In this paper, we propose a scheme called traceable CP-ABE with attribute-level user revocation for cloud storage (TUR-CPABE). In our construction, a user's decryption key and ciphertext both have two parts. A secret key update and a ciphertext update are used to resist collusion attacks between users. In addition, outsourcing encryption, decryption and attribute revocation are used to reduce the computational burden of data owners, users and the trust authority, respectively. Finally, the security of our scheme is demonstrated under a chosen plaintext attack based on a decisional q -BDHE hardness problem in the standard model.

Because a black-box traceable tool is much better than a white-box traceable tool, our future work will focus on constructing a black-box traceable CP-ABE tool with attribute-level user revocation.

Supporting information

S1 Appendix.
(DOCX)

Acknowledgments

We would like to thank the anonymous reviewer for comments and suggestions. This work was supported by the National Natural Science Foundation of China under Grants 61572019, the Key Project of Research Foundation of Natural Science Foundation of Shaanxi Province of China under Grant No. 2016JZ001.

Author Contributions

Formal analysis: Shangping Wang.

Methodology: Shangping Wang.

Writing – original draft: Keke Guo.

Writing – review & editing: Yaling Zhang.

References

1. Sahai A, Waters B. Fuzzy Identity-Based Encryption. In: Cramer R, editor; 2005 2005//; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 457–473.
2. Bethencourt J, Sahai A, Waters B. Ciphertext-Policy Attribute-Based Encryption; 2007 20–23 May 2007. pp. 321–334.
3. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. Proceedings of the 13th ACM conference on Computer and communications security. Alexandria, Virginia, USA: ACM. pp. 89–98.
4. Waters B. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano D, Fazio N, Gennaro R, Nicolosi A, editors; 2011 2011//; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 53–70.
5. Liu Je, Wan Z, Gu M (2011) Hierarchical attribute-set based encryption for scalable, flexible and fine-grained access control in cloud computing. Proceedings of the 7th international conference on Information security practice and experience. Guangzhou, China: Springer-Verlag. pp. 98–107.
6. Au MH, Huang Q, Liu JK, Susilo W, Wong DS, et al. (2008) Traceable and retrievable identity-based encryption. Proceedings of the 6th international conference on Applied cryptography and network security. New York, NY, USA: Springer-Verlag. pp. 94–110.
7. Lewko A, Waters B. New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques. In: Safavi-Naini R, Canetti R, editors; 2012 2012//; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 180–198.

8. Green M, Hohenberger S, Waters B (2011) Outsourcing the decryption of ABE ciphertexts. Proceedings of the 20th USENIX conference on Security. San Francisco, CA: USENIX Association. pp. 34–34.
9. Rouselakis Y, Waters B (2013) Practical constructions and new proof methods for large universe attribute-based encryption. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. Berlin, Germany: ACM. pp. 463–474.
10. Li J, Chen X, Li J, Jia C, Ma J, et al. Fine-Grained Access Control System Based on Outsourced Attribute-Based Encryption. In: Crampton J, Jajodia S, Mayes K, editors; 2013 2013//; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 592–609.
11. Deng H, Wu Q, Qin B, Domingo-Ferrer J, Zhang L, et al. (2014) Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts. *Information Sciences* 275: 370–384.
12. Zhang P, Chen Z, Liang K, Wang S, Wang T. A Cloud-Based Access Control Scheme with User Revocation and Attribute Update. *Information Security and Privacy*; 2016; Cham. Springer International Publishing. pp. 525–540.
13. Ma H, Dong E, Liu Z, Zhang L. Privacy-Preserving Multi-authority Ciphertext-Policy Attribute-Based Encryption with Revocation. *Advances on Broad-Band Wireless Computing, Communication and Applications*; 2018; Cham. Springer International Publishing. pp. 811–820.
14. Zhang P, Chen Z, Liu JK, Liang K, Liu H (2018) An efficient access control scheme with outsourcing capability and attribute update for fog computing. *Future Generation Computer Systems* 78: 753–762.
15. Li D, Chen J, Liu J, Wu Q, Liu W. Efficient CCA2 Secure Revocable Multi-authority Large-Universe Attribute-Based Encryption. *Cyberspace Safety and Security*; 2017; Cham. Springer International Publishing. pp. 103–118.
16. Li J, Yao W, Zhang Y, Qian H, Han J (2017) Flexible and Fine-Grained Attribute-Based Data Storage in Cloud Computing. *IEEE Transactions on Services Computing* 10: 785–796.
17. Shi Y, Zheng Q, Liu J, Han Z (2015) Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation. *Information Sciences* 295: 221–231.
18. Ma H, Peng T, Liu Z (2017) Directly revocable and verifiable key-policy attribute-based encryption for large universe. 272–284 p.
19. Wang H, Zheng Z, Wu L, Li P (2017) New directly revocable attribute-based encryption scheme and its application in cloud storage environment. *Cluster Computing* 20: 2385–2392.
20. Qian H, Li J, Zhang Y, Han J (2015) Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation. *International Journal of Information Security* 14: 487–497.
21. Huang X, Tao Q, Qin B, Liu Z. Multi-Authority Attribute Based Encryption Scheme with Revocation; 2015 3–6 Aug. 2015. pp. 1–5.
22. Li X, Tang S, Xu L, Wang H, Chen J (2017) Two-Factor Data Access Control With Efficient Revocation for Multi-Authority Cloud Storage Systems. *IEEE Access* 5: 393–405.
23. Liu Z, Cao Z, Wong DS (2013) White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Any Monotone Access Structures. *IEEE Transactions on Information Forensics and Security* 8: 76–88.
24. Boneh D, Boyen X (2004) Short Signatures Without Random Oracles. 56–73 p.
25. Ning J, Dong X, Cao Z, Wei L, Lin X (2015) White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Flexible Attributes. *IEEE Transactions on Information Forensics and Security* 10: 1274–1288.
26. Liu Z, Wong DS. Practical Ciphertext-Policy Attribute-Based Encryption: Traitor Tracing, Revocation, and Large Universe. *Applied Cryptography and Network Security*; 2015; Cham. Springer International Publishing. pp. 127–146.
27. Li X, Liang K, Liu Z, Wong D (2017) Attribute based Encryption: Traitor Tracing, Revocation and Fully Security on Prime Order Groups. 309–320 p.
28. Liu Z, Duan S, Zhou P, Wang B (2017) Traceable-then-revocable ciphertext-policy attribute-based encryption scheme. *Future Generation Computer Systems* <https://doi.org/10.1016/j.future.2017.09.045>.
29. Li J, Yao W, Han J, Zhang Y, Shen J (2018) User Collusion Avoidance CP-ABE With Efficient Attribute Revocation for Cloud Storage. *IEEE Systems Journal* 12: 1767–1777.
30. Wang H, Dong X, Cao Z (2017) Multi-value-Independent Ciphertext-Policy Attribute Based Encryption with Fast Keyword Search. *IEEE Transactions on Services Computing* <https://doi.org/10.1109/TSC.2017.2753231> 1–1.
31. Qiu S, Liu J, Shi Y, Zhang R (2017) Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack.

32. Li J, Lin X, Zhang Y, Han J (2017) KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage. *IEEE Transactions on Services Computing* 10: 715–725.
33. Zhu H, Wang L, Ahmad H, Niu X (2017) Key-Policy Attribute-Based Encryption With Equality Test in Cloud Computing. *IEEE Access* 5: 20428–20439.
34. Yang K, Zhang K, Jia X, Hasan MA, Shen X (2017) Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms. *Information Sciences* 387: 116–131.
35. Abdalla M, Dent AW, Malone-Lee J, Neven G, Phan DH, et al. Identity-Based Traitor Tracing. *Public Key Cryptography—PKC 2007*; 2007; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 361–376.
36. Wang Y, Chen K-F, Chen J-H (2011) Attribute-Based Traitor Tracing. 181–195 p.
37. Ning J, Cao Z, Dong X, Wei L (2017) White-Box Traceable CP-ABE for Cloud Storage Service: How to Catch People Leaking Their Access Credentials Effectively. *IEEE Transactions on Dependable and Secure Computing* <https://doi.org/10.1109/TDSC.2015.2484326> 1–1.
38. Liu Z, Wang X, Cui L, Jiang ZL, Zhang C. White-box traceable dynamic attribute based encryption; 2017 15–17 Dec. 2017. pp. 526–530.
39. Jiang Y, Susilo W, Mu Y, Guo F (2018) Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing. *Future Generation Computer Systems* 78: 720–729.
40. Yang Y, Liu X, Deng RH, Li Y (2017) Lightweight Sharable and Traceable Secure Mobile Health System. *IEEE Transactions on Dependable and Secure Computing* <https://doi.org/10.1109/TDSC.2015.2484326> 1–1.
41. Zhang K, Li H, Ma J, Liu X (2018) Efficient large-universe multi-authority ciphertext-policy attribute-based encryption with white-box traceability. *Science China Information Sciences* 61: 032102.
42. Boneh D, Franklin M. Identity-Based Encryption from the Weil Pairing. *Advances in Cryptology—CRYPTO 2001*; 2001; Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 213–229.
43. Beimel A (1996) Secure Schemes for Secret Sharing and Key Distribution.
44. Hur J, Noh DK (2011) Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems. *IEEE Transactions on Parallel and Distributed Systems* 22: 1214–1221.
45. Goldwasser S, Micali S (1984) Probabilistic encryption. *Journal of Computer and System Sciences* 28: 270–299.
46. Duquesne S, Lange T (2005) Pairing-based cryptography. *Mathiiscernetin volume* 22: 573–590.