SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Adaptively secure multi-authority attribute-based encryption with verifiable outsourced decryption

ZHANG Kai[1], MA JianFeng[2*], LIU JiaJia [3] & LI Hui[3]

[1]*School of Telecommunications Engineering, Xidian University, Xi'an, Shaanxi 710071, China;*
[2]*School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi 710071, China;*
[3]*School of Cyber Engineering, Xidian University, Xi'an, Shaanxi 710071, China*

Appendix A Background

Appendix A.1 Access structures

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

In our context, we consider only monotone access structures and the role of parties is taken by attributes.

Appendix A.2 Linear secret sharing schemes

Definition 2 (Linear Secret-Sharing Schemes (LSSS) [1]). A secret-sharing scheme Π over a set of parties P is called linear (over Z_p) if

1. The shares for each party form a vector over Z_p.

2. There exists a matrix A with l rows and n columns called the share-generating matrix for Π. For all $i = 1, \cdots, l$, the i·th row of A we let the function ρ defined the party labeling row i as $\rho(i)$. When we consider the column vector $v = (s, r_2, \cdots, r_n)$, where $s \in Z_p$ is the secret to be shared, and $r_2, \cdots, r_n \in Z_p$ are randomly chosen, then Av is the vector of l shares of the secret s according to Π. The share $(Av)_i$ belongs to party $\rho(i)$.

It is shown in [1] that the linear secret-sharing scheme enjoys the linear reconstruction property, defined as follows. Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be an authorized set, and let $I \subset \{1, 2, \cdots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{\omega_i \in Z_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of a secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. These constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix A.

Appendix A.3 Composite order bilinear groups

Composite order bilinear groups were first introduced in [2]. Let $\mathcal{G}$ be an algorithm that takes as input a security parameter λ and outputs a tuple $(p_1, p_2, p_3, G, G_T, e)$, where p_1, p_2, p_3 are distinct primes, G and G_T are cyclic groups of order $N = p_1 p_2 p_3$, and the bilinear map $e : G \times G \longrightarrow G_T$ is a map such that:

1. Bilinear: $\forall g, h \in G, a, b \in Z_N, e(g^a, h^b) = e(g, h)^{ab}$.

2. Non-degenerate: $\exists g \in G$ such that $e(g, g)$ has order N in G_T.

We say that G is a bilinear group if the group operations in G and the bilinear map $e : G \times G \longrightarrow G_T$ are both efficiently computable. We let G_{p_1}, G_{p_2}, and G_{p_3} denote the subgroups of order p_1, p_2, and p_3 in G respectively. We note that when $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$, $e(h_i, h_j)$ is the identity element in G_T.

Appendix B The definition of multi-authority CP-ABE with outsourced decryption

We now give the definition and security model for multi-authority CP-ABE with outsourced decryption.

* Corresponding author (email: jfma@mail.xidian.edu.cn)

Appendix B.1 Multi-authority CP-ABE with outsourced decryption

A multi-authority CP-ABE scheme with outsourced decryption consists of seven algorithms:

Global Setup(λ). The global setup algorithm takes security parameter λ as input. It outputs the global parameters GP for the system.

Authority Setup(GP). Each authority runs the authority setup algorithm with the global parameters GP as input to produce its own public / secret key pair (PK_j, SK_j).

Encrypt($GP, \{PK_j\}, M, (A, \rho)$). The encryption algorithm takes in a message M, an access structure (A, ρ), the set of public keys for relevant authorities, and the global parameters GP. It outputs a ciphertext CT.

KeyGen($GID, S, \{SK_j\}, GP$). The key generation algorithm takes as input an identity GID, the global parameters GP, a set of attributes S, and the set of secret keys for relevant authorities. It outputs a private key $SK_{S,GID}$.

TKGen($SK_{S,GID}$). The transformation key generation algorithm takes as input a private key $SK_{S,GID}$. It outputs a transformation key $TK_{S,GID}$ and the corresponding retrieving key $RK_{S,GID}$.

Transform($TK_{S,GID}, CT$). The transformation algorithm takes as input a transformation key $TK_{S,GID}$ for a set of attributes S and a ciphertext CT for an access structure (A, ρ). If the set S satisfies the access structure (A, ρ), it outputs the partially decrypted ciphertext $CT^{'}$. Otherwise, it outputs the error symbol $\perp$.

Decrypt$_{out}$($RK_{S,GID}, CT^{'}$). The decryption algorithm takes as input a retrieving key $RK_{S,GID}$ and a partially decrypted ciphertext $CT^{'}$. It outputs a message M.

We now describe a security model for multi-authority CP-ABE with outsourced decryption. We allow the adversary to select several authorities in the model and corrupt them for malicious attack. Let T be the set of authorities and S the attribute universe. The formal security game consists of the following phases:

Setup. The challenger runs the global setup algorithm and gives the global parameters GP to the adversary. The adversary specifies a set $T' \subseteq T$ of corrupt authorities. For good (non-corrupt) authorities in $T - T'$, the challenger obtains public / secret key pairs by running the authority setup algorithm, and gives the public keys to the adversary.

Phase 1. The adversary can repeatedly make any of the following queries:

1) Private key query: The adversary makes private key queries by submitting pairs (S, GID) to the challenger, where S is a set of attributes belonging to good authorities and GID is an identity. The challenger responds with the corresponding private key $SK_{S,GID}$.

2) Transformation key query: The adversary makes transformation key queries by submitting pairs (S', GID') to the challenger, where S' is a set of attributes belonging to good authorities and GID' is an identity. The challenger responds with the corresponding transformation key $TK_{S',GID'}$.

Challenge. The adversary submits two equal length messages, M_0, M_1, and an access structure (A, ρ). The access structure (A, ρ) must satisfy the following constraint. We let $S_{T'}$ denote the set of all the attributes controlled by corrupt authorities. For each identity GID, S_i refers to the attributes set for which the adversary has queried the private key $SK_{S_i,GID}$. S_{GID} refers to the attributes set $\cup S_i$. For each GID, we require that the access structure (A, ρ) cannot be satisfied by $S_{T'} \cup S_{GID}$. The challenger flips a random coin b, and encrypts M_b under (A, ρ). The ciphertext CT^* is given to the adversary.

Phase 2. Phase 1 is repeated with the restriction that the adversary cannot violate the constraint on the challenge access structure (A, ρ).

Guess. The adversary outputs a guess β' for β.

The adversary wins if $\beta = \beta'$. The adversary's advantage in this game is defined as $Pr[\beta = \beta'] - \frac{1}{2}$.

Definition 3. A multi-authority CP-ABE scheme with outsourced decryption is adaptively CPA-secure (against static corruption of authorities) if all polynomial time adversaries have at most a negligible advantage in the above security game.

We say that a system is selectively CPA-secure if we add an Init stage before Setup where the adversary commits to the challenge access structure (A, ρ). Our constructions in Section 4 will be proved secure in the adaptive security model without the Init stage. A system is statically CPA-secure if all queries done by the adversary are sent to the challenger immediately after seeing the global parameters. We refer the reader to [3] for the details of static security.

Appendix B.2 Multi-authority CP-ABE with verifiable outsourced decryption

The definition and security model of multi-authority CP-ABE scheme with verifiable outsourced decryption are the same as above, except the Encrypt algorithm, Decrypt$_{out}$ algorithm, and Challenge phase. We now describe them as follows:

Encrypt($GP, \{PK_j\}, M, (A, \rho)$). The encryption algorithm takes in a message M, an access structure (A, ρ), the set of public keys for relevant authorities, and the global parameters GP. It outputs a ciphertext CT and a ciphertext tag Tag_{CT}.

Decrypt$_{out}$($RK_{S,GID}, CT^{'}, Tag_{CT}$). The decryption algorithm takes as input a retrieving key $RK_{S,GID}$, a partially decrypted ciphertext $CT^{'}$, and a tag Tag_{CT}. It outputs a message M or $\perp$.

Challenge. The adversary submits two equal length messages, M_0, M_1, and an access structure (A, ρ). The access structure (A, ρ) must satisfy the following constraint. We let $S_{T'}$ denote the set of all the attributes controlled by corrupt authorities. For each identity GID, S_i refers to the attributes set for which the adversary has queried the private key $SK_{S_i,GID}$. S_{GID} refers to the attributes set $\cup S_i$. For each GID, we require that the access structure (A, ρ) cannot be satisfied by $S_{T'} \cup S_{GID}$. The challenger flips a random coin b, and encrypts M_b under (A, ρ). The ciphertext CT^* and the tag Tag_{CT^*} are given to the adversary.

Verifiability of the multi-authority CP-ABE with outsourced decryption is also described by a game between a challenger and an adversary. The game proceeds as follows:

Setup. The challenger runs the global setup algorithm and gives the global parameters GP to the adversary. The adversary specifies a set $T' \subseteq T$ of corrupt authorities. For good (non-corrupt) authorities in $T - T'$, the challenger obtains public / secret key pairs by running the authority setup algorithm, and gives the public keys to the adversary.

Phase 1. The adversary can repeatedly make any of the following queries:

1) Private key query: The adversary makes private key queries by submitting pairs (S, GID) to the challenger, where S is a set of attributes belonging to good authorities and GID is an identity. The challenger responds with the corresponding private key $SK_{S,GID}$.

2) Transformation key query: The adversary makes transformation key queries by submitting pairs (S', GID') to the challenger, where S' is a set of attributes belonging to good authorities and GID' is an identity. The challenger responds with the corresponding transformation key $TK_{S',GID'}$.

Challenge. The adversary specifies a message M^* and an access structure (A, ρ). The challenger computes $(CT^*, Tag_{CT^*}) = Encrypt(GP, \{PK_j\}, M^*, (A, \rho))$ and sends (CT^*, Tag_{CT^*}) to the adversary.

Phase 2. The same as Phase 1.

Output. The adversary outputs a partially decrypted ciphertext CT', an identity GID^*, and a set of attributes S^* which satisfies the access structure (A, ρ).

The adversary wins in the above game if $\text{Decrypt}_{out}(RK_{S^*,GID^*}, CT', Tag_{CT^*}) \notin \{M^*, \perp\}$. The adversary's advantage in this game is defined as Pr[The adversary wins].

Definition 4. A multi-authority CP-ABE scheme with verifiable outsourced decryption is verifiable if all polynomial time adversaries have at most a negligible advantage in this game.

Appendix C Proof of Theorem 1

Theorem 1. The above multi-authority CP-ABE scheme with outsourced decryption is adaptively CPA-secure assuming that the scheme of Lewko and Waters [4] is an adaptively CPA-secure CP-ABE scheme.

Proof. Suppose there exists a polynomial-time adversary $\mathscr{A}$ that can attack the above scheme in the adaptive security model with advantage ϵ. We build a simulator $\mathscr{B}$ that can attack the LW scheme in the adaptive CPA-security model with advantage ϵ. Let $\mathscr{C}$ be the challenger corresponding to $\mathscr{B}$ in the security game of LW scheme. In [4] the LW scheme is proved adaptively CPA-secure based on three static assumptions in the random oracle model.

Setup. The simulator $\mathscr{B}$ receives the global parameters $GP = \{N, G, G_{p1}, g_1, H\}$ from $\mathscr{C}$ and sends these to the adversary $\mathscr{A}$. $\mathscr{A}$ specifies a set $T' \subseteq T$ of corrupt authorities. For good (non-corrupt) authorities in $T - T'$, $\mathscr{B}$ obtains their public keys and sends these to the adversary $\mathscr{A}$.

Phase 1. In this phase the simulator $\mathscr{B}$ answers private key queries and transformation key queries. The simulator $\mathscr{B}$ first initializes an empty table Q. Then it answers the adversary's queries as follows:

1) Private key query: $\mathscr{A}$ makes private key query for a pair (S, GID), where S is a set of attributes belonging to good authorities and GID is an identity. $\mathscr{B}$ calls the LW key generation oracle on (S, GID) to obtain the private key $SK_{S,GID}$. Then, for each $i \in S$, $\mathscr{B}$ chooses a random value $z_i \in Z_N^*$ and sets the transformation key $TK_{S,GID}$ as $\{K'_{i,GID}\}_{i \in S} = \{K_{i,GID}^{1/z_i}\}_{i \in S}$ and the retrieving key $RK_{S,GID}$ as $\{z_i\}_{i \in S}$. Finally, $\mathscr{B}$ stores $((S, GID), SK_{S,GID}, TK_{S,GID}, RK_{S,GID})$ in table Q and returns $SK_{S,GID}$ to $\mathscr{A}$.

2) Transformation key query: The adversary $\mathscr{A}$ makes transformation key query for a pair (S', GID'), where S' is a set of attributes belonging to good authorities and GID' is an identity. $\mathscr{B}$ searches the entry $((S', GID'), SK_{S',GID'}, TK_{S',GID'}, RK_{S',GID'})$ in table Q. If such entry exists, $\mathscr{B}$ returns the transformation key $TK_{S',GID'}$. Otherwise, for each $i \in S'$, $\mathscr{B}$ chooses a random element $g_i \in G$ and sets the transformation key $TK_{S',GID'}$ as $\{g_i\}_{i \in S'}$. Note that $SK_{S',GID'} = \{K_{i,GID'} = g_1^{\alpha_i} H(GID')^{y_i}\}_{i \in S'}$ and every $g_1^{\alpha_i} H(GID')^{y_i}$ is an element in the cyclic group G, so for each $i \in S'$, there exists an unknown value $z_i \in Z_N^*$ such that $g_i = K_{i,GID'}^{1/z_i}$. Finally, $\mathscr{B}$ stores $((S', GID'), *, TK_{S',GID'}, *)$ in table Q and returns $TK_{S',GID'}$ to $\mathscr{A}$.

Challenge. The adversary $\mathscr{A}$ submits two equal length messages, M_0, M_1, and an access structure (A, ρ). The access structure (A, ρ) must satisfy the following constraint. We let $S_{T'}$ denote the set of all the attributes controlled by corrupt authorities. For each identity GID, S_i refers to the attributes set for which the adversary has queried the private key $SK_{S_i,GID}$. S_{GID} refers to the attributes set $\cup S_i$. For each GID, we require that the access structure (A, ρ) cannot be satisfied by $S_{T'} \cup S_{GID}$. $\mathscr{B}$ sends M_0, M_1, and (A, ρ) to $\mathscr{C}$ to obtain the challenge ciphertext CT^*. Then, $\mathscr{B}$ sends the challenge ciphertext CT^* to $\mathscr{A}$.

Phase 2. The adversary $\mathscr{A}$ continues to adaptively queries as in Phase 1, but with the restriction that the adversary cannot violate the constraint on the challenge access structure (A, ρ). $\mathscr{B}$ responds the queries as in Phase 1.

Guess. Eventually, $\mathscr{A}$ outputs a bit β', then $\mathscr{B}$ outputs β'.

This ends the description of the simulation. Thus, if $\mathscr{A}$ has advantage ϵ in the adaptive security game against our scheme, then $\mathscr{B}$ breaks the LW scheme with the same probability.

Appendix D Adaptively secure multi-authority ABE scheme with verifiable outsourced decryption

We extend our adaptive secure construction to achieve verifiability by the method from [5]. We refer the reader to [5] for the details of symmetric encryption (SE) and randomness extractor which will be used in our construction.

Global Setup(λ). The global setup algorithm takes as input a security parameter λ. It then chooses a bilinear group G of composite order $N = p_1p_2p_3$, a generator g_1 of G_{p1}, a hash function $H : \{0,1\}^* \to G$, two collision-resistant hash functions $H_1 : M \to \{0,1\}^{l_{H_1}}$, $H_2 : \{0,1\}^* \to \{0,1\}^{l_{H_2}}$, a symmetric encryption scheme $SE = (SE.Enc, SE.Dec)$ with key space $\{0,1\}^{l_{SE}}$, and an extractor $h \in \mathcal{H}$ where $\mathcal{H}$ is a family of pairwise independent hash functions from M to $\{0,1\}^{l_{SE}}$. The global public parameters are published as $GP = \{N, G, G_{p1}, g_1, H, H_1, H_2, h, SE\}$. We view H as a random oracle.
Authority Setup(GP). For each attribute i belonging to the authority A_j, A_j chooses two random exponents $\alpha_i, y_i \in Z_N$. The public key is published as $PK_j = \{e(g_1, g_1)^{\alpha_i}, g_1^{y_i} \forall i\}$. The authority A_j sets $SK_j = \{\alpha_i, y_i \forall i\}$ as its secret key.
Encrypt($GP, \{PK_j\}, M, (A, \rho)$) The encryption algorithm takes as input the global parameters GP, the public keys of the relevant authorities, a message M, and an LSSS access structure (A, ρ). A is an $n \times l$ matrix and ρ maps its rows to attributes. The algorithm first selects a random value $R \in G_T$. It then chooses a random $s \in Z_N$ and a random vector $v \in Z_N^l$ with s as its first entry. For $x = 1$ to l, it calculates $\lambda_x = A_x \cdot v$, where A_x is row x of A. It also chooses a random vector $\omega \in Z_N^l$ with 0 as its first entry. For $x = 1$ to l, it calculates $\omega_x = A_x \cdot \omega$. For each $x \in \{1, 2, \cdots, l\}$, it chooses a random $r_x \in Z_N$ and computes:

$$C_0 = Re(g_1, g_1)^s, C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x}, C_{2,x} = g_1^{r_x}, C_{3,x} = g_1^{y_{\rho(x)} r_x} g_1^{\omega_x}.$$

After that, it computes $Tag_0 = H_1(R)$, a symmetric key $K_{SE} = h(R)$, and $C_{SE} = SE.Enc(K_{SE}, M)$. It outputs the ciphertext as $CT = \{C_0, \{C_{1,x}, C_{2,x}, C_{3,x}\}_{x \in \{1,2,\cdots,l\}}, C_{SE}\}$ and the tag of the ciphertext CT as $Tag_{CT} = H_2(Tag_0 \| C_{SE})$.
KeyGen($GID, S, \{SK_j\}, GP$). The key generation algorithm takes as input an identity GID, the global parameters GP, the secret keys of the relevant authorities, and a set S of attributes. To create a key for GID for attribute i belonging to an authority, the authority computes: $K_{i,GID} = g_1^{\alpha_i} H(GID)^{y_i}$. The private key is $SK_{S,GID} = \{K_{i,GID} = g_1^{\alpha_i} H(GID)^{y_i}\}_{i \in S}$.
TKGen($SK_{S,GID}$). For each $i \in S$, it chooses a random value $z_i \in Z_N^*$. It sets the transformation key $TK_{S,GID}$ as $\{K'_{i,GID}\}_{i \in S} = \{K_{i,GID}^{1/z_i}\}_{i \in S}$ and the retrieving key $RK_{S,GID}$ as $\{z_i\}_{i \in S}$.
Transform($TK_{S,GID}, CT$). The transformation algorithm takes as input a transformation key $TK_{S,GID}$ for a set S and a ciphertext $(C_0, \{C_{1,x}, C_{2,x}, C_{3,x}\}_{x \in \{1,2,\cdots,l\}}, C_{SE})$ for an access structure (A, ρ). If S does not satisfy the access structure (A, ρ), it outputs $\perp$. Suppose that S satisfies the access structure (A, ρ) and let $I \subset \{1, 2, \cdots, l\}$ be defined as $I = \{x : \rho(x) \in S\}$. The transformation algorithm chooses constants $\{c_x \in Z_N\}_{x \in I}$ such that $\sum_{x \in I} c_x A_x = (1, 0, \ldots, 0)$ and computes:

$$T_1 = \prod_{x \in I} (C_{1,x} e(H(GID), C_{3,x}))^{c_x}, \{T_{2,x} = e(K'_{\rho(x),GID}, C_{2,x})^{c_x}\}_{x \in I}.$$

It outputs the partially decrypted ciphertext CT' as $(T_0 = C_0, T_1, \{T_{2,x}\}_{x \in I}, C_{SE})$.
Decrypt$_{out}$($RK_{S,GID}, CT', Tag_{CT}$). The decryption algorithm takes as input a retrieving key $RK_{S,GID} = \{z_i\}_{i \in S}$, a partially decrypted ciphertext $CT' = (T_0, T_1, \{T_{2,x}\}_{x \in I}, C_{SE})$, and a tag Tag_{CT}. It computes $R = T_0 \prod_{x \in I} T_{2,x}^{z_{\rho(x)}} / T_1$ and $Tag_0 = H_1(R)$. If $Tag_{CT} = H_2(Tag_0 \| C_{SE})$, it computes $K_{SE} = h(R)$ and outputs $M = SE.Dec(K_{SE}, C_{SE})$. Otherwise, it outputs $\perp$.

By the theorems in [5] and Theorem 1, we can easily obtain the following two corollaries.

Corollary 1. Suppose the scheme in [4] is adaptively CPA-secure, SE is a semantically secure one-time symmetric encryption scheme, the parameters satisfy $0 < l_{SE} \leqslant (log \mid K \mid -l_{H_1}) - 2log(1/\varepsilon_{\mathcal{H}})$ where $\varepsilon_{\mathcal{H}}$ is a negligible value. Then, the above construction is adaptively CPA-secure.

Corollary 2. The above construction is verifiable assuming that H_1 and H_2 are collision-resistant hash functions.

Appendix E Efficient statically secure multi-authority ABE scheme with outsourced decryption

In this section, we propose an efficient statically secure multi-authority CP-ABE scheme with verifiable outsourced decryption in a bilinear group G of primer order p. $e : G \times G \longrightarrow G_{T'}$ is a bilinear map, where $G_{T'}$ is another group of primer order p. Let (A, ρ) be an access policy, U be the attribute universe, and U_Θ be the authority universe. Each attribute is controlled by a unique authority $\theta \in U_\Theta$. $T : U \to U_\Theta$ is a publicly computable function that maps each attribute to a unique authority and $\delta(\cdot) = T(\rho(\cdot))$ is a function that maps rows to authorities.

Appendix E.1 An efficient construction

We now present our efficient multi-authority CP-ABE scheme with outsourced decryption construction based on the Rouselakis-Waters scheme [3], in which neither any central authority nor any coordination between different authorities is required. Compared with our adaptively secure construction, this construction uses the significantly faster primer order bilinear group and supports large universe.
Global Setup(λ). The global setup algorithm takes security parameter λ as input. It then chooses a bilinear group G of prime order p with generator g, a hash function H that hashes global identities to group elements, and another hash function F that hashes attributes to group elements. The global public parameters are published as $GP = \{p, G, g, H, F, U, U_\Theta, T\}$. We view H and F as random oracles.
Authority Setup(GP). For each authority $\theta \in U_\Theta$, θ chooses two random exponents $\alpha_\theta, y_\theta \in Z_p$. The public key is published as $PK_\theta = \{e(g, g)^{\alpha_\theta}, g^{y_\theta}\}$. The authority θ sets $SK_\theta = \{\alpha_\theta, y_\theta\}$ as its secret key.
Encrypt($GP, \{PK_\theta\}, M, (A, \rho)$). The encryption algorithm takes in a message M, an $n \times l$ access matrix A with ρ mapping its rows to attributes, the global parameters GP, and the public keys of the relevant authorities. The algorithm first chooses

a random $s \in Z_p$ and a random vector $v \in Z_p^l$ with s as its first entry. For $x = 1$ to l, it calculates $\lambda_x = A_x \cdot v$, where A_x is row x of A. It also chooses a random vector $\omega \in Z_p^l$ with 0 as its first entry. For $x = 1$ to l, it calculates $\omega_x = A_x \cdot \omega$. For each $x \in \{1, 2, \cdots, l\}$, it chooses a random $r_x \in Z_p$. The ciphertext is computed as:

$$C_0 = Me(g,g)^s, C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\delta(x)} r_x}, C_{2,x} = g^{-r_x}, C_{3,x} = g^{y_{\delta(x)} r_x} g^{\omega_x}, C_{4,x} = F(\rho(x))^{r_x}.$$

KeyGen$(GID, S, \{SK_\theta\}, GP)$. The key generation algorithm takes as input an identity GID, the global parameters GP, the secret keys of the relevant authorities, and a set S of attributes. To create a key for GID for attribute i belonging to an authority θ, the authority θ chooses a random $t_{i,GID} \in Z_p$ and computes:

$$K_{i,GID} = g^{\alpha_\theta} H(GID)^{y_\theta} F(i)^{t_{i,GID}}, \overline{K_{i,GID}} = g^{t_{i,GID}}.$$

It outputs the private key $SK_{S,GID}$ as

$$\{K_{i,GID} = g_1^{\alpha_\theta} H(GID)^{y_\theta} F(i)^{t_{i,GID}}, \overline{K_{i,GID}} = g^{t_{i,GID}}\}_{i \in S}.$$

TKGen$(SK_{S,GID})$. For each $i \in S$, it chooses a random value $z_i \in Z_p^*$. It sets the transformation key $TK_{S,GID}$ as $\{K'_{i,GID}, \overline{K_{i,GID}}'\}_{i \in S} = \{K_{i,GID}^{1/z_i}, \overline{K_{i,GID}}^{1/z_i}\}_{i \in S}$ and the retrieving key $RK_{S,GID}$ as $\{z_i\}_{i \in S}$.

Transform$(TK_{S,GID}, CT)$. The transformation algorithm takes as input a transformation key $TK_{S,GID}$ for a set S and a ciphertext $(C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\}_{x \in \{1,2,\cdots,l\}})$ for an access structure (A, ρ). If S does not satisfy the access structure (A, ρ), it outputs $\perp$. Suppose that S satisfies the access structure (A, ρ) and let $I \subset \{1, 2, \cdots, l\}$ be defined as $I = \{x : \rho(x) \in S\}$. The transformation algorithm chooses constants $\{c_x \in Z_p\}_{x \in I}$ such that $\sum_{x \in I} c_x A_x = (1, 0, \ldots, 0)$ and computes:

$$T_1 = \prod_{x \in I} (C_{1,x} e(H(GID), C_{3,x}))^{c_x}, \{T_{2,x} = \{e(K'_{\rho(x),GID}, C_{2,x}) e(\overline{K_{\rho(x),GID}}', C_{4,x})\}^{c_x}\}_{x \in I}.$$

It outputs the partially decrypted ciphertext CT' as $(T_0 = C_0, T_1, \{T_{2,x}\}_{x \in I})$.

Decrypt$_{out}(RK_{S,GID}, CT')$. The decryption algorithm takes as input a retrieving key $RK_{S,GID} = \{z_i\}_{i \in S}$ and a partially decrypted ciphertext $CT' = (T_0, T_1, \{T_{2,x}\}_{x \in I})$. It computes $M = T_0 / (T_1 \cdot \prod_{x \in I} T_{2,x}^{z_{\rho(x)}})$.

Theorem 2. The above multi-authority CP-ABE scheme with outsourced decryption is statically CPA-secure assuming that the Rouselakis-Waters scheme [3] is a statically CPA-secure CP-ABE scheme.

Proof. Suppose there exists a polynomial-time adversary $\mathscr{A}$ that can attack the above scheme in the static CPA-security model with advantage ϵ. We build a simulator $\mathscr{B}$ that can attack the RW scheme in the static CPA-security model with advantage ϵ. Let $\mathscr{C}$ be the challenger corresponding to $\mathscr{B}$ in the security game of RW scheme. In [3] the RW scheme is proved statically CPA-secure in the random oracle model under a non-interactive q-type assumption.

Setup. The simulator $\mathscr{B}$ receives the global parameters $GP = \{p, G, g, H, F, U, U_\Theta, T\}$ from $\mathscr{C}$ and sends these to the adversary $\mathscr{A}$.

Adversary's Queries. Without loss of generality, we assume the adversary $\mathscr{A}$ makes private key queries for a sequence $\{(S_j, GID_j)\}_{j=1}^m$, and transformation key queries for a sequence $\{(S'_j, GID_j)\}_{j=1}^m$. If $\mathscr{A}$ has not queried private (resp., transformation) key for the identity GID_j, let the set $S_j = \emptyset$ (resp., $S'_j = \emptyset$). The adversary $\mathscr{A}$ responds with:

1) A set $C_\theta \subseteq U_\theta$ of corrupt authorities and their respective public keys.

2) A set $N_\theta \subseteq U_\theta$ of the good authorities for which the adversary requests the public keys.

3) A sequence $\{(S_j, GID_j)\}_{j=1}^m$ of the secret key queries, where the global identities GID_j are distinct and $S_j \subseteq U$ with $T(S_j) \cap C_\theta = \emptyset$. A pair (S_j, GID_j) in this sequence denotes that the adversary requests the secret key for the user GID_j with attributes from the set S_j.

4) A sequence $\{(S'_j, GID_j)\}_{j=1}^m$ of the transformation key queries, where the global identities GID_j are distinct and $S'_j \subseteq U$ with $T(S'_j) \cap C_\theta = \emptyset$.

5) Two equal length messages, M_0, M_1, and an access structure (A, ρ). The access structure (A, ρ) must satisfy the following constraint. We let S_{C_θ} denote the set of all the attributes controlled by corrupt authorities. For each identity GID_j, we require that the access structure (A, ρ) cannot be satisfied by $S_{C_\theta} \cup S_j$.

Challenge's Replies. The simulator $\mathscr{B}$ obtains the above responds from $\mathscr{A}$ and sends these responds except the transformation key queries to $\mathscr{C}$. Then, $\mathscr{C}$ replies with the public keys $\{PK_\theta\}$ for all $\theta \in N_\theta$, the secret keys $\{SK_{S_j,GID_j}\}$ for all $j \in [m]$, and the challenge ciphertext CT^*. For each $j \in [m]$, $\mathscr{B}$ creates the transformation key for the pair (S'_j, GID_j) as follows:

1) If $S'_j \subseteq S_j$, then for each $i \in S'_j$, it chooses a random value $z_i \in Z_p^*$. It sets the transformation key $TK_{S'_j,GID_j}$ as $\{K'_{i,GID_j}, \overline{K_{i,GID_j}}'\}_{i \in S'_j} = \{K_{i,GID_j}^{1/z_i}, \overline{K_{i,GID_j}}^{1/z_i}\}_{i \in S'_j}$ and the retrieving key RK_{S,GID_j} as $\{z_i\}_{i \in S'_j}$.

2) If $S'_j \nsubseteq S_j$, proceed as follows:

For each $i \in S'_j \cap S_j$, it chooses a random value $z_i \in Z_p^*$ and sets $\{K'_{i,GID_j}, \overline{K_{i,GID_j}}'\} = \{K_{i,GID_j}^{1/z_i}, \overline{K_{i,GID_j}}^{1/z_i}\}$.

For each $i \in S'_j - S'_j \cap S_j$, it chooses a random element $g_i \in G$ and a random value $t_i \in Z_p$. It sets $\{K'_{i,GID_j}, \overline{K_{i,GID_j}}'\} = \{g_i F(i)^{t_i}, g^{t_i}\}$. Note that $g^{\alpha_\theta} H(GID_j)^{y_\theta}$ is an element in the cyclic group G, so for each i, there exists an unknown value $z_i \in Z_p^*$ such that $g_i = (g^{\alpha_\theta} H(GID_j)^{y_\theta})^{1/z_i}$. Hence, $\{g_i F(i)^{t_i}, g^{t_i}\} = \{(g^{\alpha_\theta} H(GID_j)^{y_\theta})^{1/z_i} F(i)^{t_i}, , g^{t_i}\} = \{(g^{\alpha_\theta} H(GID_j)^{y_\theta} F(i)^{z_i t_i})^{1/z_i}, (g^{z_i t_i})^{1/z_i}\}$.

It sets the transformation key $TK_{S'_j,GID_j}$ as $\{K'_{i,GID_j}, \overline{K_{i,GID_j}}'\}_{i \in S'_j}$.

Finally, $\mathscr{B}$ sends the public keys $\{PK_\theta\}_{\theta \in N_\theta}$, the secret keys $\{SK_{S_j,GID_j}\}_{j=1}^m$, the transformation keys $\{TK_{S'_j,GID_j}\}_{j=1}^m$, and the challenge ciphertext CT^* to $\mathscr{A}$.

Guess. Eventually, $\mathscr{A}$ outputs a bit β', then $\mathscr{B}$ outputs β'.

This ends the description of the simulation. Thus, if $\mathscr{A}$ has advantage ϵ in the static security game against our scheme, then $\mathscr{B}$ breaks the RW scheme with the same probability.

Appendix E.2 A verifiable construction

Similar to Appendix D, we extend the above construction to achieve verifiability.

Global Setup(λ). The global setup algorithm takes as input a security parameter λ. It then chooses a bilinear group G of prime order p with generator g, a hash functions H that hashes global identities to group elements, another hash function F that hashes attributes to group elements, two collision-resistant hash functions $H_1 : M \to \{0,1\}^{l_{H_1}}$, $H_2 : \{0,1\}^* \to \{0,1\}^{l_{H_2}}$, a symmetric encryption scheme $SE = (SE.Enc, SE.Dec)$ with key space $\{0,1\}^{l_{SE}}$, and an extractor $h \in \mathcal{H}$ where $\mathcal{H}$ is a family of pairwise independent hash functions from M to $\{0,1\}^{l_{SE}}$. The global public parameters are published as $GP = \{p, G, g, H, F, U, U_\Theta, T, H_1, H_2, h, SE\}$. We view H and F as random oracles.

Authority Setup(GP). For each authority $\theta \in U_\Theta$, θ chooses two random exponents $\alpha_\theta, y_\theta \in Z_p$. The public key is published as $PK_\theta = \{e(g,g)^{\alpha_\theta}, g^{y_\theta}\}$. The authority θ sets $SK_\theta = \{\alpha_\theta, y_\theta\}$ as its secret key.

Encrypt($GP, \{PK_\theta\}, M, (A, \rho)$) The encryption algorithm takes as input the global parameters GP, the public keys of the relevant authorities, a message M, and an LSSS access structure (A, ρ). A is an $n \times l$ matrix and ρ mapping its rows to attributes. The algorithm first selects a random value $R \in G_T$. It then chooses a random $s \in Z_p$ and a random vector $v \in Z_p^l$ with s as its first entry. For $x = 1$ to l, it calculates $\lambda_x = A_x \cdot v$, where A_x is row x of A. It also chooses a random vector $\omega \in Z_p^l$ with 0 as its first entry. For $x = 1$ to l, it calculates $\omega_x = A_x \cdot \omega$. For each $x \in \{1, 2, \cdots, l\}$, it chooses a random $r_x \in Z_p$ and computes:

$$C_0 = Re(g,g)^s, C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\delta(x)} r_x}, C_{2,x} = g^{-r_x}, C_{3,x} = g^{y_{\delta(x)} r_x} g^{\omega_x}, C_{4,x} = F(\rho(x))^{r_x}.$$

After that, it computes:

$$Tag_0 = H_1(R), K_{SE} = h(R), C_{SE} = SE.Enc(K_{SE}, M).$$

Finally, It outputs the ciphertext as $CT = \{C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\}_{x \in \{1,2\cdots,l\}}, C_{SE}\}$ and the tag of the ciphertext CT as $Tag_{CT} = H_2(Tag_0 \| C_{SE})$.

KeyGen($GID, S, \{SK_\theta\}, GP$). The key generation algorithm takes as input an identity GID, the global parameters GP, the secret keys of the relevant authorities, and a set S of attributes. To create a key for GID for attribute i belonging to an authority θ, the authority θ chooses a random $t_{i,GID} \in Z_p$ and computes:

$$K_{i,GID} = g^{\alpha_\theta} H(GID)^{y_\theta} F(i)^{t_{i,GID}}, \overline{K_{i,GID}} = g^{t_{i,GID}}.$$

It outputs the private key $SK_{S,GID}$ as

$$\{K_{i,GID} = g_1^{\alpha_\theta} H(GID)^{y_\theta} F(i)^{t_{i,GID}}, \overline{K_{i,GID}} = g^{t_{i,GID}}\}_{i \in S}.$$

TKGen($SK_{S,GID}$). For each $i \in S$, it chooses a random value $z_i \in Z_p^*$. It sets the transformation key $TK_{S,GID}$ as $\{K'_{i,GID}, \overline{K_{i,GID}}'\}_{i \in S} = \{K_{i,GID}^{1/z_i}, \overline{K_{i,GID}}^{1/z_i}\}_{i \in S}$ and the retrieving key $RK_{S,GID}$ as $\{z_i\}_{i \in S}$.

Transform($TK_{S,GID}, CT$). The transformation algorithm takes as input a transformation key $TK_{S,GID}$ for a set S and a ciphertext $(C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}\}_{x \in \{1,2\cdots,l\}}, C_{SE})$ for an access structure (A, ρ). If S does not satisfy the access structure (A, ρ), it outputs $\perp$. Suppose that S satisfies the access structure (A, ρ) and let $I \subset \{1, 2, \cdots, l\}$ be defined as $I = \{x : \rho(x) \in S\}$. The transformation algorithm chooses constants $\{c_x \in Z_p\}_{x \in I}$ such that $\sum_{x \in I} c_x A_x = (1, 0, \ldots, 0)$ and computes:

$$T_1 = \prod_{x \in I} (C_{1,x} e(H(GID), C_{3,x}))^{c_x}, \{T_{2,x} = \{e(K'_{\rho(x),GID}, C_{2,x}) e(\overline{K_{\rho(x),GID}}', C_{4,x})\}^{c_x}\}_{x \in I}.$$

It outputs the partially decrypted ciphertext CT' as $(T_0 = C_0, T_1, \{T_{2,x}\}_{x \in I}, C_{SE})$.

Decrypt$_{out}$($RK_{S,GID}, CT', Tag_{CT}$). The decryption algorithm takes as input a retrieving key $RK_{S,GID} = \{z_i\}_{i \in S}$, a partially decrypted ciphertext $CT' = (T_0, T_1, \{T_{2,x}\}_{x \in I}, C_{SE})$, and a tag Tag_{CT}. It computes $R = T_0 / (T_1 \cdot \prod_{x \in I} T_{2,x}^{z_{\rho(x)}})$ and $Tag_0 = H_1(R)$. If $Tag_{CT} = H_2(Tag_0 \| C_{SE})$, it computes $K_{SE} = h(R)$ and outputs $M = SE.Dec(K_{SE}, C_{SE})$. Otherwise, it outputs $\perp$.

By the theorems in [5] and Theorem 2, we can easily obtain the following corollaries.

Corollary 3. Suppose the scheme in [3] is statically CPA-secure, SE is a semantically secure one-time symmetric encryption scheme, the parameters satisfy $0 < l_{SE} \leqslant (log \mid K \mid - l_{H_1}) - 2log(1/\varepsilon_{\mathcal{H}})$ where $\varepsilon_{\mathcal{H}}$ is a negligible value. Then, the above construction is statically CPA-secure.

Corollary 4. The above construction is verifiable assuming that H_1 and H_2 are collision-resistant hash functions.

Appendix F Efficiency analysis

We summarize in Table F1 and Table F2 the efficiency analysis results of our multi-authority CP-ABE schemes. Note that the main purpose of outsourced decryption is to save the local computation time for the user decryption on mobile devices, so we focus on the user's computation costs of decryption in the multi-authority CP-ABE schemes. We denote by $|U|$ the number of attributes in the scheme, by $|U_\Theta|$ ($|U_\Theta| \leqslant |U|$ in our schemes) the number of authorities, by l an LSSS access structure with an $l \times n$ matrix, by $l_{C_{SE}}$ the size of the SE ciphertext C_{SE}, by $|S|$ the number of attributes in the use's

Table F1 Summary of adaptively secure multi-authority CP-ABE outsourcing results, where both G_T and G are cyclic groups of composite order $N = p_1p_2p_3$, G_{p_1} denotes the subgroup of order p_1 in G, P and E_T denote the maximum time required to compute a pairing in G and an exponentiation in G_T, respectively.

Scheme	Pk size	Full ct size	Pd ct size	Tf key size	User dec ops	Vrf
LW [4]	$\|U\|(\|G_T\|+\|G_{p_1}\|)$	$(1+l)\|G_T\|+2l\|G_{p_1}\|$	–	–	$2\|I\|P+\|I\|E_T$	–
Letter	$\|U\|(\|G_T\|+\|G_{p_1}\|)$	$(1+l)\|G_T\|+2l\|G_{p_1}\|$	$(2+\|I\|)\|G_T\|$	$\|S\|\|G\|$	$\|I\|E_T$	×
Appendix D	$\|U\|(\|G_T\|+\|G_{p_1}\|)$	$(1+l)\|G_T\|+2l\|G_{p_1}\|+l_{C_{SE}}$	$(2+\|I\|)\|G_T\|+l_{C_{SE}}$	$\|S\|\|G\|$	$\|I\|E_T$	✓

Table F2 Summary of statically secure multi-authority CP-ABE outsourcing results, where $G_{T'}$ and G_p are cyclic groups of prime order p, P' and $E_{T'}$ denote the maximum time to compute a pairing in G_p and an exponentiation in $G_{T'}$, respectively.

Scheme	Pk size	Full ct size	Pd ct size	Tf key size	User dec ops	Vrf
RW [3]	$\|U_\Theta\|(\|G_T\|+\|G_{p_1}\|)$	$(1+l)\|G_{T'}\|+3l\|G_p\|$	–	–	$3\|I\|P'+\|I\|E_{T'}$	–
Appendix E.1	$\|U_\Theta\|(\|G_T\|+\|G_{p_1}\|)$	$(1+l)\|G_{T'}\|+3l\|G_p\|$	$(2+\|I\|)\|G_{T'}\|$	$2\|S\|\|G_p\|$	$\|I\|E_{T'}$	×
Appendix E.2	$\|U_\Theta\|(\|G_T\|+\|G_{p_1}\|)$	$(1+l)\|G_{T'}\|+3l\|G_p\|+l_{C_{SE}}$	$(2+\|I\|)\|G_{T'}\|+l_{C_{SE}}$	$2\|S\|\|G_p\|$	$\|I\|E_{T'}$	✓

key, and by $|I|$ ($|I| \leqslant l$ in our schemes) the number of rows used in decryption. Let "Full ct size" (resp., "Pd ct size", "Tf key size", "Pk size") denote the size of ABE ciphertext (resp., partially decrypted ciphertext, transformation key, public key). Let "User dec ops" represent the user's computation costs of decryption. We ignore non-dominant operations, like multiplication and hash operations.

We provide in Table F1 a comparison between our adaptively secure CP-ABE schemes and the LW scheme [4], all of which are constructed in composite order groups and adaptively CPA-secure in the random oracle model. We observe from Table F1 that in order to decrypt a ciphertext, the user in our schemes need to send proxy a transformation key once which size is linear with the user attributes, but only need to compute $|I|$ exponentiations in G_T, whereas the user in LW scheme has to compute $2|I|$ pairings in G and $|I|$ exponentiations in G_T. Furthermore, the partially decrypted ciphertext size in our schemes is much smaller than that of the LW scheme. Note that none of the costs in Table F1 is grows with the the number of authorities.

In Table F2, we compare our statically secure CP-ABE schemes with the RW scheme [3]. All of these schemes are constructed in prime order groups and statically CPA-secure in the random oracle model. Similarly, one can easily observe from Table F2 that a partially decrypted ciphertext in our schemes is always smaller and faster to decrypt than a ciphertext in the RW scheme. In order to decrypt a ciphertext, the user in our statically-secure CP-ABE schemes only need to compute $|I|$ exponentiations in $G_{T'}$, which is a prime order group. Note that the pairing and exponentiation operations in prime order groups are $1 \sim 2$ orders of magnitude faster than that in composite order groups, so our schemes in Table F2 is significantly faster than the schemes in Table F1. The timings in prime and composite order groups exhibits a significant gap, which has been discussed in detail in [3]. In addition, the public key size in Table F2 only grows linearly with the number of authorities, whereas the public key size in Table F1 grows linearly with number of attributes in the scheme.

Appendix G An analysis of the ABE schemes in [6]

Three multi-authority ABE schemes are presented in [6]: a multi-authority ABE scheme, a multi-authority CP-ABE scheme with outsourced decryption and a multi-authority KP-ABE scheme with outsourced decryption, which appear in Section 3, 4 and 5 of [6], respectively. In the sequel, for ease of discussion, we refer to these schemes as LM1, LM2 and LM3, respectively.

We firstly show that the LM2 scheme is not secure by constructing an effective attack. According to LM2, the private key of the user is $SK'_j = (K' = g^{y_j}g^{at'_j}, L' = g^{t'_j}, K'_x = F(x)^{t'_j}{}_{x\in S_0}, \{K'_{j,y} = F(y)^{t'_j}\}_{y\in S_j}), 1 \leqslant j \leqslant k$, the transformation key TK is $(PK, K_j = K_j'^{1/z} = g^{y_j/z}g^{at}, L_j = L'^{1/z} = g^t, \{K_x\}_{x\in S_0} = \{K_x'^{1/z}\}_{x\in S_0}, \{K'_{j,y}\}_{y\in S_j} = \{K_{j,y}'^{1/z}\}_{y\in S_j}), 1 \leqslant j \leqslant k$. We notice there exist some typos with respect to the private key and the transformation key in LM2. Note that the transformation algorithm need to employ K_0 and $K_{0,\rho(i)}$, which are not contained in the transformation key TK, so the proxy cannot run the transformation algorithm. Additionally, $\{K_x\}_{x\in S_0} = \{K_x'^{1/z}\}_{x\in S_0}$ is never used in the transformation algorithm. Following the description of LM2, we can easily see that the private key and the transformation key in LM2 scheme should be corrected as $SK'_j = (K'_j = g^{y_j}g^{at'_j}, L'_j = g^{t'_j}, \{K'_{j,y} = F(y)^{t'_j}\}_{y\in S_j}), 0 \leqslant j \leqslant k$ and $(PK, K_j = K_j'^{1/z} = g^{y_j/z}g^{at_j}, L_j = L_j'^{1/z} = g^{t_j}, \{K'_{j,y}\}_{y\in S_j} = \{K_{j,y}'^{1/z}\}_{y\in S_j}), 0 \leqslant j \leqslant k$. In that case, the transformation algorithm can be completely executed by the proxy. However, LM2 is not resistant to collusion attacks even though these typos are corrected. Assume each user $u_j (0 \leqslant j \leqslant k)$ obtains private key $SK'_j = (K'_j = g^{y_j}g^{at'_j}, L'_j = g^{t'_j}, \{K'_{j,y} = F(y)^{t'_j}\}_{y\in S_j})$, which is associated with his attribute S_j. If $\{S_j\}_{0\leqslant j\leqslant k}$ satisfies the access structure (M, ρ), then all the users $\{u_j\}_{0\leqslant j\leqslant k}$ can combine their private keys to decrypt the ciphertext, which is associated with the access structure (M, ρ). Hence, the LM2 scheme is not secure against collusion attacks.

Due to the same reason, LM1 scheme is not resistant to collusion attacks either, when the attributes are controlled by more than one authorities.

In LM3 scheme, different authorities generate different secret keys for a single user, but these keys are associated

with the same access policy. It means that different authorities control the same attributes set and generate secret keys associated with the same access policy for a user. However, in the well-defined multi-authority ABE scheme, each attribute is controlled by a unique authority and a user's keys from different authorities are associated with different attributes sets or access policies. In this regard, LM3 scheme cannot be seen as a proper multi-authority ABE. Hence, [6] fails to achieve outsourced decryption for multi-authority ABE scheme.

References

1 Beimel A. Secure schemes for secret sharing and key distribution. Dissertation for Ph.D. Degree. Haifa, Israel: Technion-Israel Institute of technology, 1996

2 Boneh D, Goh E J, Nissim K. Evaluating 2-DNF formulas on ciphertexts. In: Proceedings of the 2nd Theory of Cryptography Conference, Cambridge, MA, USA, 2005. 325-341

3 Rouselakis Y, Waters B. Efficient statically-secure large-universe multi-authority attribute-based encryption. Cryptology ePrint Archive, Report 2015/016, 2015

4 Lewko A, Waters B. Decentralizing attribute-based encryption. In: Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 2011. 568-588

5 Qin B, Deng R, Liu S, et al. Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Transactions on Information Forensics and Security, 2015, 10(7): 1384-1393

6 Li K, Ma H. Outsourcing decryption of multi-authority ABE ciphertexts. International Journal of Network Security, 2014, 16(4): 286-294