



Traceable and revocable multi-authority ABE supporting decryption outsourcing and policy update for cloud data access control

Yan-Qing YAO^{1,2,3}✉, Yun-Jia ZHANG^{1,2}, Zhi-Yi LIU², Yu-Xuan WANG², Xin-Yu TAN², Zhengde ZHAI⁴

1. State Key Laboratory of Complex & Critical Software Environment, Beihang University, Beijing 100191, China

2. Key Laboratory of Aerospace Network Security, Ministry of Industry and Information Technology, School of Cyber Science and Technology, Beihang University, Beijing 100191, China

3. Shanghai Artificial Intelligence Laboratory, Shanghai 200232, China

4. Li Auto Inc, Beijing 101300, China

Received December 13, 2024; accepted March 27, 2025

E-mail: yaoyq@buaa.edu.cn

© Higher Education Press 2026

Abstract

Nowadays, vast and rapidly growing information acts as digital records of social activities and is widely collected and stored as economic assets. To reduce the difficulty and local data management's cost significantly, cloud storage services provide a highly available, high-performance, and low-cost solution for user data hosting, enabling remote access, backup, and sharing of data stored by the cloud. However, this service model is not without security risks, including user privacy exposure, low trustworthiness of data, and unauthorized access. To address these concerns, attribute-based encryption (ABE) schemes allow for the implementation of fine-grained access policies while ensure the confidentiality and availability of data stored under the cloud environment. The issues of collusion among authorities, excessive decryption computation overhead, and high complexity in attribute revocation have aroused many researchers' attention, and many works have emerged. However, expanding the functionality of ABE schemes to satisfy multiple requirements and improving existing functionality of ABE schemes are still urgent problems to be solved. Motivated by these problems, here we propose a novel multi-functional multi-authority ABE scheme that incorporates functional features such as multi-authority key generation, outsourced decryption, malicious user tracking, flexible attribute revocation, and real-time policy updates, thereby providing fine-grained access control as well as confidentiality for data stored under cloud environments. Similar to prior works, we have analyzed the static security, forward security, and resistance to collusion attacks of our proposed scheme for completeness. Storage and computational efficiency evaluation shows that our proposed scheme achieves lower storage costs and computational overhead compared to existing schemes with similar functionalities.

Keywords

multi-authority attribute-based encryption; CP-ABE; decryption outsourcing; policy update; attribute revocation; white-box traceability; access control

1 Introduction

The abundance of information and data resources is a prominent characteristic of the current age of big data. These data serve as digital records of societal activities and represent reusable intangible assets. Utilizing data-sharing technologies to exchange valuable information not only facilitates easier access to services for users but also plays a crucial role in supporting collaborative efforts among various data stakeholders. However, the data is often widely distributed across different physical locations, leading to data silos that are inadequate for the collaborative information needs of participants in the big data age. Consequently, as data continues to

proliferate at an unprecedented rate, cloud storage has garnered significant attention as a convenient and flexible service model. As a vital component of cloud computing, cloud storage offers users access to computing and storage resources via the internet, significantly reducing local data storage and management costs. This model allows multiple collaborators to achieve remote data sharing, management, and cooperation through the cloud.

Consequently, various industries have conducted research on data-sharing models in cloud environments, but a series of security risks and privacy protection issues persist. In a partially trusted cloud environment, malicious servers and users can exploit unprocessed

sensitive data stored in the cloud, compromising data confidentiality. Moreover, the physical separation between cloud servers and data owners restricts the ability of owners to fully control access permissions, resulting in unauthorized access to data. Therefore, for data sharing under cloud environments, ensuring data confidentiality and implementing fine-grained access control policies are major challenges. Currently, one method to ensure the security of data is through encryption before uploading it to the cloud. However, traditional encryption methods facilitate only “one-to-one” information exchange and fail to meet actual interaction requirements.

Attribute-based encryption (ABE) can provide robust technical support for addressing challenges related to privacy protection, trust delegation, and inefficient access control in data-sharing models within cloud environments. Specifically, ABE employs a set of attributes rather than a single identifier to describe the identity of a user, while incorporating access policies into ciphertext or keys to restrict decryption and conceal information. Additionally, ABE extends the concept of singular verification information to allow for freely definable and extensible combinations of attributes, thereby enriching the scalability of system while ensuring data confidentiality and user privacy. Fine-grained access control can be achieved through a “one-many” data-sharing communication model.

Nevertheless, current ABE schemes often face challenges related to authority and user management, permission changes, and computational burdens. From a management perspective, the traditional single-authority model poses significant risks to cloud storage systems, such as the inability to distribute computational load during key construction and the escalating management complexity as the number of attributes increases, resulting in system bottlenecks and constrained scalability. Additionally, malicious users within the system may leak their decryption keys, resulting in unauthorized data access. Therefore, it is essential to establish tracking mechanisms to expel such users from the system. In terms of permission changes, traditional schemes lack real-time and effective policy updates and attribute revocation mechanisms. Data owners may need to modify data access scopes based on evolving sharing requirements. However, traditional models fail to update access policies and corresponding ciphertexts promptly, which increases the data leakage risk. In addition, the identities of data users may change over time, meaning they may no longer possess the attributes required to access certain data. In such cases, traditional models cannot timely update the relevant keys and ciphertexts through attribute revocation, thereby compromising the security and timeliness of the system. Finally, many existing schemes incur significant computational overhead during the decryption process, imposing high performance demands on end-user devices. This limitation restricts the applicability of data-sharing solutions in cloud storage scenarios where terminal resources are constrained.

1.1 Our contributions

Focused on addressing the challenges within the aforementioned ABE schemes and aligning them more effectively with the practical demands of cloud storage data-sharing models, this work makes the

following key contributions:

1) We propose a multi-functional multi-authority ABE data sharing model. This model enables data owners and data users to share data through cloud servers, leveraging trusted third parties, multiple attribute authorities, outsourced servers, and tracking servers to achieve multi-authority key generation, outsourcing decryption, attribute revocation, access policy updates, and malicious user tracking on the basis of traditional ABE modes. Relevant analyses and validations demonstrate that this model provides improved privacy protection and effectively meets the practical needs of data sharing and access control.

2) Based on the above model, we present a multi-functional multi-authority ABE access control scheme which is traceable and revocable with high efficiency. This scheme adopts the multi-authority approach in ciphertext-policy ABE to achieve fine-grained access control while guaranteeing the confidentiality of cloud data. Here, data users can utilize the outsourcing servers to perform high-cost decryption operations, thereby reducing the computational burden on local terminals. By introducing attribute version keys in the scheme, authorities and cloud servers collaborate to execute efficient attribute revocation algorithms, achieving dual updates of keys and ciphertexts. Through delegating tracking servers to execute the white-box tracking algorithm, authorities can obtain the identities of users that maliciously leak decryption keys and expel them from the system. In addition, this scheme holds a flexible access policy update mechanism that allows data owners to change access permissions at any time and promptly updates the ciphertexts and access policies stored in the cloud.

3) We provide the correctness verification, security proofs and performance comparison results for our proposed scheme. The correctness verification confirms that our scheme can be performed correctly in algorithms such as encryption, decryption, and outsourcing decryption. Similar to prior works, we give security proofs under a q -type assumption called q -1 assumption. We prove that this scheme can resist collusion attacks while maintaining static and forward security. By testing the operating costs of each algorithm module under different data scales, this scheme can significantly improve time efficiency, reduce key size, and shorten ciphertext length compared to known schemes, effectively decreasing the computation and storage costs.

1.2 Related works

Sahai and Waters proposed the ABE scheme in 2005 [1] firstly, for obtaining fine-grained access control. Researchers divide ABE schemes into two types: Key Policy ABE (KP-ABE) and Ciphertext Policy ABE (CP-ABE) [2]. Among them, the access policy in KP-ABE is associated with the user's private key, and the attribute set is associated with information resource. While in CP-ABE the access policy and the attribute set are associated with the ciphertext and the private key of user respectively. Specifically, the data owners can formulate policies grounded on the conditions for accessing the data, and the data can be successfully accessed by the data users only when their own attribute set matches the access policy. In 2007, The first concrete and implementable CP-ABE scheme was proposed by

Bethencourt et al. [3] and a provably secure CP-ABE scheme was proposed by Cheung et al. [4]. Since then, the bounded CP-ABE schemes' emergence [5] has enriched the expression form of the access policy.

With a higher degree of flexibility, for data access control services, the CP-ABE mechanism is more applicable under the cloud environment. Consequently, it offers effective safeguards for user privacy and data security in numerous domains such as electronic healthcare and social networking websites [6,7]. Nevertheless, the key generation and distribution processes of the majority of ABE schemes are entirely accomplished by a central authority, presenting the problem of single-point failure and thus failing to satisfy the demands of practical applications.

To address the aforementioned problems, Chase [8] presented the first Multi-Authority ABE (MA-ABE) scheme, which assumes the inclusion of multiple attribute authorities, but the existence of a central authority still brings about a security risk of single-point failure for this scheme. Subsequently, Chase and Chow [9] designed an MA-ABE scheme, which resolves the single-point failure issue while preventing user collusion. Moreover, schemes [10], [11], and [12] have respectively proposed MA-ABE systems, achieving decentralization via multiple attribute authorities. In 2023, Li et al. [13] designed a secure MA-ABE scheme for assuring data deletion that resists collusion attack and overcomes key escrow issue.

Furthermore, the considerable expenditure generated by decryption operations renders ABE-based access control protocols to have high demands on the computing performance of terminals, which restricts data sharing when terminal resources are limited. In response to this problem, Yang et al. [14] presented a ciphertext policy MA-ABE scheme with outsourced decryption, enhancing system efficiency by entrusting decryption computations to a cloud server. Subsequently, under the cloud environment, some researchers [15–18] propose ABE mechanisms that support outsourced decryption and achieve fine-grained access control respectively. Among them, Sethi et al. [18] constructed a ciphertext policy MA-ABE system encompassing multiple functional modules such as white-box tracing, policy update, and outsourced decryption. Motivated by the single authority key escrow as well as the low search efficiency issues, [19] proposes an attribute-based searchable encryption (SE) scheme with multi-authority. However, when the attributes of users change, the corresponding access permissions should be modified or revoked accordingly. This process should consider both correctness and timeliness. It has been found through research that the above works do not involve the attribute revocation function.

The first ABE scheme supporting attribute revocation was presented by Pirretti et al. [20]. This scheme utilizes timestamps to set validity periods for each attribute to achieve the revocation function, but it fails to meet the real-time requirement. The work of Yu et al. [21] addressed this issue through proxy re-encryption technology and proposed an instant revocation CP-ABE scheme. In order to implement attribute revocation while also reducing the computational burden on users, Hur et al. [22] and Fan et al. [23] respectively put forward revocable and outsourcable ABE schemes. Subsequently, a revocable ABE scheme for cloud storage systems

was presented in [24]. However, its considerable computational volume leads to a relatively low system efficiency. Yang et al. [25] introduced the attribute version keys concept in the MA-ABE scheme and implemented the instant revocation of attributes through the update of version keys. A user-revocable MA-ABE scheme was constructed by Youcef et al. [26] by dividing users into groups. Nevertheless, the rules for user groups in this scheme are relatively complex and difficult to maintain, thereby causing the problem of system efficiency decline when the number of users increases. Thereafter, Ge et al. [27] and Hu et al. [28] presented revocable ABE schemes respectively for cloud service scenarios, among which [28] satisfies adaptive security and unboundedness. In 2024, Chen et al. [29] constructed an efficient attribute-based encryption scheme which offers data integrity and dynamic access. In that scheme, the cloud serves as the delegation of data owners and performs the revocation. In 2025, Li et al. [30] proposed a flexible policy-hidden multigroup attribute-based encryption scheme. In that scheme, the attribute values are concealed, arbitrary number of users are allowed to be revoked and joined, and decryption tasks are outsourced to cloud servers.

In accordance with the diverse requirements for privacy protection and information security in practical applications, apart from the aforementioned works such as the multi-authority model, outsourced decryption and attribute revocation, the research on ABE schemes also pertains to functions like access policy updates (e.g., [31–33]) and traceability (e.g., [34–43]). Notably, Yang et al. [33] first presented a CP-ABE scheme from lattices that supports access policy update and resists attacks including quantum, collusion, and chosen plaintext. He et al. [43] introduced a CP-ABE scheme with enhanced traceability, which intertwined white-box and black-box traceability. Li et al. [44] proposed a CP-ABE scheme allowing fine-grained access control of encrypted IoT data in the cloud environment. It also provides user privacy and accountability.

Nevertheless, at present, there is no MA-ABE mechanism that can concurrently fulfill the characteristics of outsourced decryption, attribute revocation, policy updates and traceability, so as to be applicable to multi-party data sharing and access control scenarios in cloud storage environments. Hence, our work is committed to filling this vacancy and conducting research on the multi-functional MA-ABE mechanism in the aforementioned cloud environment.

1.3 Organization

The remainder of the paper is organized as below: In Section 2, we recall several notations and concepts. In section 3, we present the system model, algorithm definitions and security model of our proposed multi-functional MA-ABE scheme for cloud data access control. Section 4 shows the concrete construction of our proposed scheme. The correctness and security analysis is presented in Section 5. In Section 6, we conduct functionality and performance evaluation. Finally, we conclude in Section 7.

■ 2 Preliminaries

Now we will present some notations and concepts involved in our scheme.

2.1 Prime order bilinear groups

Recall the concept of prime order bilinear groups as follows. Denote $\mathcal{G}_{\text{PBG}}(1^\lambda)$ as a generator, the input of which is the security parameter 1^λ , while the output of which is $\mathbb{G} := (p, G, G_T, e, g)$ as an explanation of prime order bilinear groups. In $\mathbb{G}$, p is a large prime, G and G_T are two cyclic groups of order p , g is the generator of G , and a function $e : G \times G \rightarrow G_T$ named with a bilinear map satisfies three properties below.

- 1) Bilinearity. $\forall x \in G, u, v \in \mathbb{Z}_p, e(x^u, x^v) = e(x, x)^{uv}$ holds.
- 2) Non-degeneracy. Denote 1_{G_T} as the identity element of G_T . $e(g, g) \neq 1_{G_T}$ holds.
- 3) Computability. For any given $x \in G$, $e(x, x)$ can be computed by a polynomial time algorithm.

2.2 Hardness assumption

We recall a q -type assumption on prime order bilinear groups, denoted by q -1 assumption [45], which is similar to the Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption [46]. Let G and G_T be two cyclic groups of prime order p . We assume that there exists a computable bilinear map $e : G \times G \rightarrow G_T$. The q -1 assumption [45] is described through a game between an attacker and a challenger below.

The challenger chooses a element g uniformly at random from the group G , and $q+2$ values $a, s, b_1, b_2, \dots, b_q$ uniformly at random from $\mathbb{Z}_p$. Subsequently, the challenger sends to the attacker a tuple $\mathbb{B}$ which consists of the following elements:

$$\begin{aligned} &g, g^s, p, G, G_T, e, \\ &g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, g^{a^i/b_j^2}, \quad \forall i, j \in \{1, 2, \dots, q\}, \\ &g^{a^i b_j/b_j'^2}, \quad \forall i \in \{1, 2, \dots, 2q\}, j \in \{1, 2, \dots, q\}, \\ &j' \in \{1, 2, \dots, q\}, \text{ where } j \neq j', \\ &g^{a^i/b_j} \quad \forall i \in \{1, 2, \dots, 2q\}, j \in \{1, 2, \dots, q\}, \\ &\text{where } i \neq q+1, \\ &g^{sa^i b_j/b_j'}, g^{sa^i b_j/b_j'^2}, \quad \forall i, j, j' \in \{1, 2, \dots, q\}, \text{ where } j \neq j'. \end{aligned}$$

Additionally, the challenger gives to the attacker the term $e(g, g)^{a^{q+1}s}$ as well as a term R picked uniformly at random from G_T .

Finally, the attacker distinguishes between R and $e(g, g)^{a^{q+1}s}$ via a algorithm $\mathcal{A}$, and the attacker has advantage ϵ in the above security game if

$$|Pr[\mathcal{A}(\mathbb{B}, e(g, g)^{a^{q+1}s}) = 0] - Pr[\mathcal{A}(\mathbb{B}, R) = 0]| \geq \epsilon.$$

The q -1 assumption is said to hold if for every probabilistic polynomial time algorithm $\mathcal{A}$ it has at most a negligible advantage in this game [45].

2.3 Access structures

Denote $P = \{P_1, P_2, \dots, P_n\}$ as the set of participants and $2^P = \{A | A \subseteq P\}$, where participants correspond to attributes in ABE. The set $\mathbb{A} \subseteq 2^P$ is called monotonic if for each $B, C \subseteq P$, in case $B \subseteq C$ and $B \in \mathbb{A}$, then $C \in \mathbb{A}$. We refer to $\mathbb{A}$ as a monotonic

access structure if it is a monotonic non-empty subset in $P = \{P_1, P_2, \dots, P_n\}$, that is, $\mathbb{A}$ is a subset of $2^{P_1, P_2, \dots, P_n} \setminus \{\emptyset\}$. For any set D , if $D \in \mathbb{A}$, it is termed an authorized set, or else it is considered unauthorized.

2.4 Linear secret sharing schemes

The ABE system in our contribution employs linear secret sharing schemes (LSSS) [10]. Given a set of participants $P = \{P_1, P_2, \dots, P_n\}$, a secret sharing scheme Π is called linear defined on P if:

- 1) For every participant, the shares of a secret $s \in \mathbb{Z}_p$ form a vector over $\mathbb{Z}_p$.
- 2) For every secret sharing scheme Π , there exists a $l \times n$ matrix M which is named with the share-generating matrix. Utilizing a function ρ , M 's rows are labeled with participants from P , that is, $\rho : \{1, \dots, l\} \rightarrow P$. Let $s \in \mathbb{Z}_p$ be the secret to be shared and $y_2, \dots, y_n \in \mathbb{Z}_p$ be randomly chosen elements, denote $v = (s, y_2, \dots, y_n)$ as a row vector. The vector Mv^T provides the secret s 's l shares depending on Π . The share $\lambda_i = (Mv^T)_i$ is assigned to participant $\rho(i)$.

LSSS conform to the property of linear reconstruction, detailed as follows. Assume Π is a LSSS for access structure $\mathbb{A}$. Denote $S \in \mathbb{A}$ as an authorized set. Denote $I = \{i | \rho(i) \in S\} \subseteq \{1, \dots, l\}$. Consequently, there exists an algorithm that can generate constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ for a secret s 's valid shares $\{\lambda_i\}$ in polynomial time satisfying $\sum_{i \in I} \omega_i \lambda_i = s$. It can be observed that $\sum_{i \in I} \omega_i M_i = (1, 0, \dots, 0)$, where M_i corresponds to the i th row of M . Nevertheless, for any unauthorized set there exists no such constants.

Based on the above description, we employ (M, ρ) to represent the access structure $\mathbb{A}$'s access policy in our ABE construction.

Access control structures are typically described using access control trees: every leaf node represents an attribute, and non-leaf nodes denote threshold nodes of the form (t, n) , in which n indicates the number of children of the threshold node which is satisfied if at least t (where $1 \leq t \leq n$) of its child nodes are satisfied. Specifically, monotone boolean formulas is able to be modeled as binary access trees, in which AND and OR gates correspond to threshold nodes of $(2, 2)$ and $(1, 2)$, respectively. Via utilizing the Lewko-Waters algorithm [10] and the Liu-Cao-Wong algorithm [47], access structures are possibly converted into LSSS matrices and construct the matching ρ functions over these matrices. The Lewko-Waters algorithm deals with access control trees that can be expressed using monotone (excluding NOT gates) boolean formulas, requiring that leaf nodes are attributes and non-leaf nodes are either AND or OR gates. In contrast, the Liu-Cao-Wong algorithm is more versatile, allowing non-leaf nodes to be threshold nodes of the form (t, n) rather than being restricted to AND or OR gates. This algorithm translates AND and OR gates into threshold nodes of $(2, 2)$ and $(1, 2)$, respectively, resulting in an LSSS matrix of the same size as that produced by the Lewko-Waters algorithm.

■ 3 The proposed multi-functional MA-ABE scheme for cloud data access control

Now we will provide a multi-functional MA-ABE scheme which is

traceable and revocable with decryption outsourcing and policy update, which can be applied to cloud data access control. We will present its system model, algorithm definitions, and security model in detail as follows.

3.1 System model

According to the requirements of practical cloud data access control, the data owner, data user, outsourced server, cloud server, the third party authority, attribute authorities, and tracking server comprise the main participant set of the system model constructed in this scheme.

Data owner (DO): DO is the data's provider, who defines the data's access policy and provides the encrypted data to the cloud server (CS). When the access policy changes, DO will upload the updated access policy to CS.

Data user (DU): DU is the data's user, who submits a data access request to CS. If the access policy is satisfied, the ciphertext returned by CS can be successfully decrypted, and at the same time, an outsourced server can be utilized to implement outsourced decryption operations.

Cloud server (CS): An incompletely trustworthy service provider, which is accountable for storing the ciphertext data received from DO along with offering data access control services in accordance with the information of DU, namely sending the requisite ciphertext data to it. If the data users' attributes are revoked or the access policy is updated, it is tasked with converting the ciphertext to effect attribute revocation or policy update.

Outsourced server (OS): OS decrypts the ciphertext received from DU and feeds back the result to it, thereby alleviating the computational burden on DU's terminal.

Attribute authority (AA): AA manages diverse sets of attributes and generates keys for DUs in accordance with their attributes. When attributes are updated, it updates the keys of DUs whose attributes

have not been revoked.

Third party authority (TPA): A trusted certification center, responsible for generating the global parameters. Meanwhile, it conducts identity authentication for both AA and DU, and issues them unique authority identity and user identity, respectively.

Tracking server (TS): It is responsible for providing white-box tracking services for malicious users, namely conducting sanity checks on the keys of malicious users and sending the tracking results to AA.

Figure 1 depicts the system model. TPA conducts system initialization to generate global parameters and simultaneously creates unique identities for AA and DU. AA produces the attribute public key as well as user attribute private key, and transmits them respectively to DO and DU. The plaintext data is encrypted by DO using the attribute public key and the corresponding ciphertext is uploaded by DO to CS. If the access policy needs to be changed, a policy update request can be issued to CS, following which CS updates the corresponding stored ciphertexts. After DU sends a data access request to CS, it is capable of converting the obtained ciphertext data and sending the converted ciphertext and key data to OS for decryption, thereby enhancing the computing efficiency of the terminal. When AA detects potential malicious users, it requests TS to implement tracking on the relevant malicious users. After TS completes the key check, it returns the malicious user's identity to AA. AA can then add the malicious user to the user revocation list to safeguard the security of the system.

3.2 Algorithm definitions

Our multi-functional MA-ABE scheme is implemented by executing a series of algorithms, consisting of eight steps: Setup, Key Generation, Data Encryption, Data Decryption, Outsourcing Decryption, Attribute Revocation, Policy Update, and Traceability.

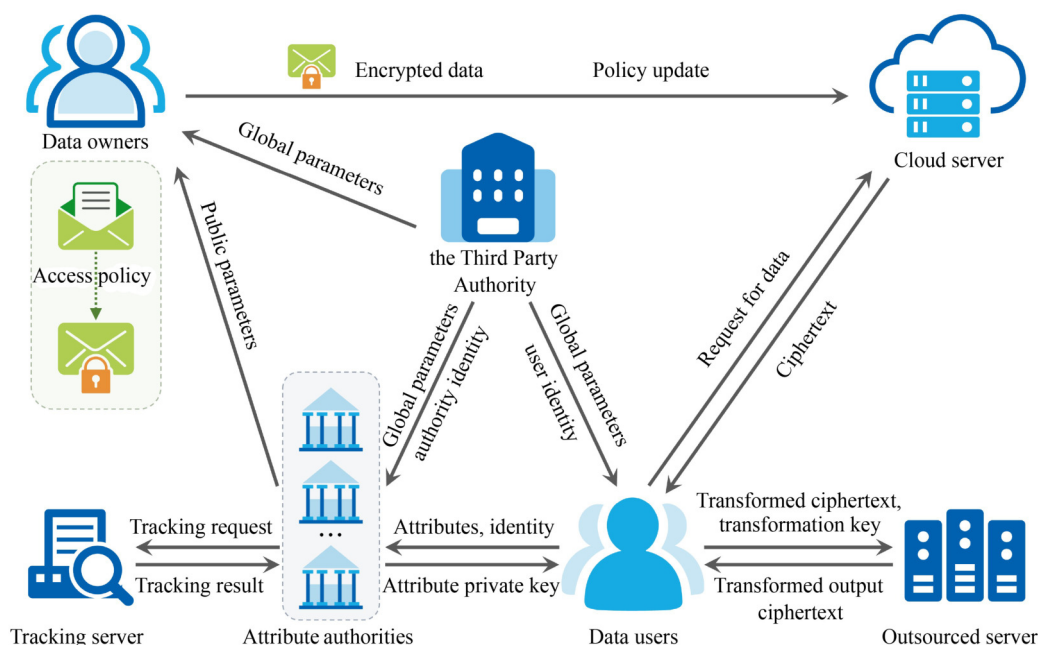


Fig. 1 System model of the multi-functional MA-ABE scheme for cloud data access control

- Setup

TPA and AA undertake the setup procedures respectively to initialize the system. This group encompasses two algorithms: GlobalSetup and AASetup.

GlobalSetup(1^λ) $\rightarrow GP$: TPA performs this algorithm with the security parameter 1^λ as input, and generates the global parameters GP for every authority and every user in the system to employ.

AASetup(GP, S_{aid}) $\rightarrow (PK_{aid}, SK_{aid}, \{VK_{x_{aid}}, PK_{x_{aid}}\}_{x_{aid} \in S_{aid}})$: This algorithm is executed by AA with identity aid . The inputs of this algorithm are the system parameters GP along with the attribute set S_{aid} managed by AA. The output of this algorithm is a pair (PK_{aid}, SK_{aid}) of AA where PK_{aid} (resp. SK_{aid}) is the public key (resp. private key), as well as the set $\{VK_{x_{aid}}, PK_{x_{aid}}\}_{x_{aid} \in S_{aid}}$ including version keys and public keys for the managed attributes. In this case, PK_{aid} and $\{PK_{x_{aid}}\}_{x_{aid} \in S_{aid}}$ are treated as public parameters, while SK_{aid} and $\{VK_{x_{aid}}\}_{x_{aid} \in S_{aid}}$ are considered as the components of master key.

- Key generation

Multiple attribute authorities perform this phase. The multiple attribute authorities generate corresponding attribute private key components for users based on the attribute sets they manage.

KeyGen($GP, (uid, S), \{SK_{aid}\}, \{VK_{x_{aid}}\}_{x_{aid} \in S}$) $\rightarrow SK_{S,uid}$: When inputting the system parameters GP , a data user identity uid , attribute set S owned by the user, the set of private keys $\{SK_{aid}\}$ about relevant attribute authorities, and version key set $\{VK_{x_{aid}}\}_{x_{aid} \in S}$ of all attributes in S , this algorithm produces $SK_{S,uid}$ which is the attribute private key for DU with identity uid .

- Data encryption

Through this step, the data is encrypted by DO and the corresponding ciphertext is stored in CS for access by other data users.

Encrypt($GP, Msg, (M, \rho), \{PK_{aid}\}, \{PK_{x_{aid}}\}$) $\rightarrow CT$: When inputting GP (i.e., the system parameters), Msg (i.e., a plaintext of the shared data), (M, ρ) (i.e., an access policy), $\{PK_{aid}\}$ (i.e., the attribute authorities' public key set related to the access policy), and $\{PK_{x_{aid}}\}$ (i.e., the attribute public key set), the $< italic > DO < /italic >$ calculates and obtains the ciphertext CT .

- Data decryption

The data user decrypts the ciphertext stored in the cloud server to be shared through this step. If its attributes satisfy the corresponding access policy, the ciphertext can be successfully decrypted by DU. Otherwise, decryption fails.

Decrypt($GP, CT, SK_{S,uid}$) $\rightarrow Msg$ or $\perp$: This algorithm is performed by DU with identity uid . The inputs of this algorithm are GP, CT obtained from the cloud server, and the attribute private key $SK_{S,uid}$ held by the user with identity uid . The output of this algorithm is the plaintext Msg if the access policy of CT is satisfied. Otherwise, a decryption failure symbol $\perp$ is returned.

- Outsourcing decryption

Data users have the option to delegate the ciphertext to an outsourcing server for decryption, mitigating local computational

expenses and enhancing the operational efficiency of the system. This phase incorporates three algorithms: GenTransKey, PDecrypt and OutsourcingDecrypt, which are executed in sequential order to realize the outsourcing decryption procedure.

GenTransKey($GP, CT, SK_{S,uid}$) $\rightarrow (TK_{S,uid}, RK_{S,uid}, CT')$: When inputting GP, CT along with its attribute private key $SK_{S,uid}$, the DU with identity uid produces the transformation key $TK_{S,uid}$, the reserved key $RK_{S,uid}$ and the transformed ciphertext CT' . And it transmits CT', uid and $TK_{S,uid}$ to OS.

PDecrypt($GP, CT', uid, TK_{S,uid}$) $\rightarrow CT'_{out}$: OS utilizes system parameters GP , the received ciphertext CT' , identity uid of DU and the transformation key $TK_{S,uid}$ to calculate the transformed output ciphertext CT'_{out} , and delivers it back to DU.

OutsourcingDecrypt($GP, CT'_{out}, RK_{S,uid}$) $\rightarrow Msg$ or $\perp$: DU executes this algorithm by taking system parameters GP , the ciphertext CT'_{out} and its reserved key $RK_{S,uid}$ as inputs. If the access policy of CT is satisfied, it outputs the plaintext Msg , or else a symbol $\perp$ meaning decryption failure is returned.

- Attribute revocation

In the case where a certain attribute of a user is revoked, aiming at preventing unauthorized access to related data, this user is not allowed to have the data access rights corresponding to this attribute anymore. That is, the private key about this attribute becomes invalid. The attribute revocation step comprises three algorithms: UKeyGen, SKUpdate, and CTUpdate, which update the attribute private keys for other data users that have not lost this attribute.

UKeyGen($x_{aid}, VK_{x_{aid}}$) $\rightarrow \widehat{VK}_{x_{aid}}, AUK_{x_{aid}}$: When inputting a revoked attribute x_{aid} as well as its current version key $VK_{x_{aid}}$, AA that manages this attribute with identity aid produces a new version key $\widehat{VK}_{x_{aid}}$ for x_{aid} as well as an update key $AUK_{x_{aid}}$.

SKUpdate($SK_{S,uid}, AUK_{x_{aid}}$) $\rightarrow \widehat{SK}_{S,uid}$: Assuming that the DU with identity uid has not lost the revoked attribute x_{aid} , this algorithm takes the current attribute private key $SK_{S,uid}$ of this DU and the update key $AUK_{x_{aid}}$ generated by AA as inputs to produce the updated attribute private key $\widehat{SK}_{S,uid}$ for DU. The original attribute private key becomes invalid.

CTUpdate($CT, AUK_{x_{aid}}$) $\rightarrow \widehat{CT}$: This algorithm is performed by CS. When inputting the original ciphertext CT associated with x_{aid} as well as the update key $AUK_{x_{aid}}$, it produces $\widehat{CT}$ as the new ciphertext and enables it to replace the original one.

- Policy update

If a DO intends to modify the access policy, it's unnecessary to re-encrypt the original plaintext suitable for the new access policy. Instead, by leveraging the relationship between the new and original policies, the update key can be generated, achieving the update of the ciphertext stored in CS. This phase encompasses three algorithms: PolicyCompare, PKUpdate, and PCTUpdate.

PolicyCompare((M', ρ'), (M, ρ)) $\rightarrow I_1, I_2, I_3$: This algorithm is executed by DO, with the new (M', ρ') and the original (M, ρ) as inputs. According to the corresponding relationship of attributes in the two policies, it obtains three sets of indexes, namely I_1, I_2 , and I_3 .

PKUpdate($GP, (M', \rho'), (M, \rho), \{I_1, I_2, I_3\}, Sha-resInfo(M), \{PK_{aid}\}, \{PK_{x_{aid}}\}) \rightarrow PUK_M$: DO inputs system parameters GP , the new access policy (M', ρ') , the original access policy (M, ρ) , three sets I_1, I_2 , and I_3 , the secret sharing information $Sha-resInfo(M)$ of M , the public key set $\{PK_{aid}\}$ of relevant authorities and the public key set $\{PK_{x_{aid}}\}$ of relevant attributes to generate the ciphertext update key PUK_M .

PCTUpdate($CT, (M', \rho'), \{I_1, I_2, I_3\}, PUK_M) \rightarrow CT'$: Cloud server takes as inputs CT which is stored in CS, the new (M', ρ') , three sets I_1, I_2, I_3 for M' and the update key PUK_M . It generates the updated ciphertext CT' . Subsequently, the original ciphertext is replaced with CT' .

• Traceability

In case AA detects the existence of malicious users within the system, it is capable of entrusting the tracking server to track them. Once these users pass the key sanity check, they would be added to the user revocation list by AA to safeguard the system security.

Trace($GP, SKS, uid, \{PK_{aid}\}, \{PK_{x_{aid}}\}) \rightarrow uid$ or $\perp$: The tracking server performs this algorithm, taking system parameters GP , the attribute private key SKS, uid of the malicious user, the set $\{PK_{aid}\}$ of related attribute authorities along with the set of related attribute public keys $\{PK_{x_{aid}}\}$ as inputs. If the key sanity check succeeds, the tracking server generates identity uid of the malicious user so that AA can add it to the revocation list. Or else, this algorithm returns a special symbol $\perp$, implying this user does not need to be tracked.

3.3 Definitions of correctness, forward security and resistance to collusion attacks

Now we present the definitions of correctness, forward security and resistance to collusion attacks about the above multi-functional MA-ABE scheme as follows.

Definition 1 The above multi-functional MA-ABE scheme is said to be correct if it satisfies the following conditions.

1) Assume that the resources of data users are not constrained. Whenever GP is obtained from the global setup algorithm, CT is obtained from the encryption algorithm on the message Msg , and $\{SKS, uid\}$ is the key obtained from the key generation algorithm for the user identity uid , attribute set S owned by the user satisfying the access structure of the ciphertext, then **Decrypt**(GP, CT, SKS, uid) $\rightarrow Msg$. Otherwise, data decryption fails.

2) Assume that the resources of data users are constrained. Whenever GP is obtained from the global setup algorithm, CT is obtained from the encryption algorithm on the message Msg , and $\{SKS, uid\}$ is the key obtained from the key generation algorithm for the user identity uid , attribute set S owned by the user satisfying the access structure of the ciphertext, we execute the outsourcing decryption phase including three algorithms: GenTransKey, PDecrypt and OutsourcingDecrypt, the outsourcing decryption phase can finally generate the plaintext Msg . Otherwise, it fails.

Definition 2 The above multi-functional MA-ABE scheme is said to be forward secure if any system user losing an attribute cannot

successfully access the associated plaintext data anymore unless their remaining attributes continue to satisfy the access policy for that data.

Definition 3 Collusion attack refers to that a subset of attribute authorities or data users collude to ex-change information so as to gain access to more plaintext data.

3.4 Static security model

On the basis of [48], we construct a security model, defining it as a game between the challenger and the attacker to provide a static security proof of the scheme. Here the set of corrupt authorities is chosen by the attacker and remains fixed throughout the game. It should be noted that one attribute is managed by one attribute authority, although one attribute authority may manage multiple attributes.

This security game is composed of four phases, as shown below.

1) Global setup.

The challenger invoke the algorithm *GlobalSetup* with input 1^λ and outputs GP (i.e., the global public parameter).

2) Attacker's queries.

The attacker provides the responses to the challenger as follows.

(i) A corrupt authority set $U_{co-AA} \subseteq U_{AA}$ selected by the attacker and their respective public keys $\{PK_{aid}\}_{aid \in U_{co-AA}}$, where U_{AA} is the set of all authorities in our cryptosystem.

(ii) A non-corrupt authority set $U_{no-co-AA} \subseteq U_{AA}$ for which the attacker asks for the public keys. Note that $U_{no-co-AA} \cap U_{co-AA} = \emptyset$.

(iii) A sequence $\{(uid_i, S_i)\}_{i \in [m_1]}$ for which the attacker requests the attribute private keys, where uid_i is a global identity for each $i \in [m_1]$, uid_i and uid_j are distinct as long as $i \neq j$ with $i, j \in [m_1]$, and S_i is the user's attribute set where the user owns identity uid_i . There exists a function T to associate attributes with their corresponding authorities, i.e., $T: U \rightarrow U_{AA}$, where U is the set of all attributes in our cryptosystem. Note that $S_i \subseteq U$ and $T(j) \notin U_{co-AA}, \forall j \in S_i$.

(iv) A sequence $\{(uid_i, S_i)\}_{i \in [m_2]}$ for which the attacker requests the transform keys, where the constraints on uid_i and S_i follow from those of $\{(uid_i, S_i)\}_{i \in [m_1]}$.

(v) Two messages Msg_0, Msg_1 of same length as well as a set $\{(M_k, \rho_k)\}_{k \in [q]}$ of challenge access policies. For every challenge $(M_k, \rho_k), k \in [q]$, $S_i \cup S_{U_{co-AA}}$ is an authorized set $\forall i \in [m]$, where $S_{U_{co-AA}}$ is the attribute set managed by U_{co-AA} .

3) Challenger's replies.

The challenger selects a bit b uniformly at random from $\{0, 1\}$ and replies with:

(i) The public key set $\{PK_{aid}\}$ corresponding to the non-corrupt authorities $aid \in U_{no-co-AA}$, where the public key PK_{aid} is generated by invoking *AASet-up*(GP, S_{aid}) algorithm.

(ii) The set of attribute private keys $\{SKS_{i, uid_i}\}_{i \in [m]}$, where SKS_{i, uid_i} is obtained via performing *KeyGen*($GP, (uid_i, S_i), \{SK_{aid}\}, \{VK_{x_{aid}}\}_{x_{aid} \in S_i}$) algorithm.

(iii) The set of challenge ciphertexts $\{CT_k\}_{k \in [q]}$, where CT_k is produced by running *Encrypt*($GP, Msg_b, (M_k, \rho_k), \{PK_{aid}\}$,

$\{PK_{x_{aid}}\}$ algorithm.

4) Guess.

The attacker outputs a guess $b' \in \{0, 1\}$.

Definition 4 A Multi-Authority CP-ABE system is called statically secure if in the above security game the attacker has a negligible advantage in guessing the bit b correctly, where the attacker's advantage is defined as $|\Pr[b = b'] - 0.5|$.

■ 4 Concrete construction of our proposed multi-functional MA-ABE scheme

Grounded on the formal algorithm definitions as well as the security model presented in the previous section, this section details our proposed MA-ABE scheme's concrete construction, which also consists of eight phases: Setup, Key Generation, Data Encryption, Data Decryption, Outsourcing Decryption, Attribute Revocation, Policy Update, and Traceability.

4.1 Setup

This step is made up of GlobalSetup and AASetup, with GlobalSetup performed by TPA and AASetup managed by AA.

GlobalSetup $(1^\lambda) \rightarrow GP$: This algorithm outputs the global public parameters $GP = \{G, G_T, p, g, e, H, F, T, \delta\}$ for the system when inputting the security parameter 1^λ . The elements in GP are defined as follows:

- (i) G and G_T are cyclic groups, both of which have prime order p .
- (ii) Let $e : G \times G \rightarrow G_T$ be a bilinear map.
- (iii) Let g be a generator of G .
- (iv) Let $H : \mathbb{Z}_p^* \rightarrow G$ be a hash function mapping a user's identity (i.e., uid) to an element in G .
- (v) Let $F : U \rightarrow G$ be a hash function mapping a user's attribute to an element in G .
- (vi) Let $T : U \rightarrow U_{AA}$ be a function mapping an attribute to its corresponding authority, i.e., aid .
- (vii) $\delta : [I] \rightarrow U_{AA}$ is a function mapping a row of a matrix with I rows to corresponding authority, i.e., $\delta(\cdot) = T(\rho(\cdot))$.

AASetup $(GP, S_{aid}) \rightarrow (PK_{aid}, SK_{aid}, \{VK_{x_{aid}}, PK_{x_{aid}}\}_{x_{aid} \in S_{aid}})$: Each authority $aid \in U_{AA}$ runs this algorithm to output PK_{aid} , SK_{aid} and $\{VK_{x_{aid}}, PK_{x_{aid}}\}_{x_{aid} \in S_{aid}}$ when inputting the global parameters GP as well as its managed attribute set S_{aid} . During this process, the algorithm chooses random values $\alpha_{aid}, \beta_{aid} \in \mathbb{Z}_p$ and $v_{x_{aid}} \in \mathbb{Z}_p$ for each attribute x_{aid} in S_{aid} . Here,

$$\begin{aligned} PK_{aid} &= (PK_{1,aid} = e(g, g)^{\alpha_{aid}}, PK_{2,aid} = g^{\beta_{aid}}), \\ SK_{aid} &= (\alpha_{aid}, \beta_{aid}), VK_{x_{aid}} = v_{x_{aid}}, \\ PK_{x_{aid}} &= g^{v_{x_{aid}} F(x_{aid})}. \end{aligned}$$

4.2 Key generation

The key generation algorithm *KeyGen* is executed by multiple attribute authorities to produce the corresponding attribute private key components for a specific user.

KeyGen $(GP, (uid, S), \{SK_{aid}\}, \{VK_{x_{aid}}\}_{x_{aid} \in S}) \rightarrow SK_{S,uid}$: The inputs of this algorithm are GP , a user identity $uid \in \mathbb{Z}_p^*$, the user's attribute set S , the private keys $\{SK_{aid}\}$ of corresponding authorities as well as version keys $\{VK_{x_{aid}}\}_{x_{aid} \in S}$ of the attributes in S . This

algorithm outputs an attribute private key $SK_{S,uid}$ for (uid, S) . For each attribute $i \in S$, the relevant authority with identity $aid = T(i)$ chooses an element t uniformly at random from $\mathbb{Z}_p^*$ and computes:

$$\begin{aligned} K_{1,i,uid} &= g^{\alpha_{aid}} \cdot H(uid)^{\beta_{aid}} \cdot (g^{VK_i} \cdot F(i))^t, \\ K_{2,i,uid} &= g^t, K_{3,uid} = uid. \end{aligned}$$

The user with identity uid combines the above private key components provided by the corresponding authorities for each attribute to obtain an attribute private key $SK_{S,uid}$ as

$$SK_{S,uid} = \{K_{3,uid}, \{K_{1,i,uid}, K_{2,i,uid}\}_{i \in S}\}.$$

4.3 Data encryption

In this step, the data encryption algorithm *Encrypt* is executed by a data owner as follows:

Encrypt $(GP, Msg, (M, \rho), \{PK_{aid}\}, \{PK_{x_{aid}}\}) \rightarrow CT$: The inputs of the encryption algorithm are GP (i.e., the global parameters), Msg (i.e., a message), (M, ρ) (i.e., an LSSS access policy with $M = (M_1, \dots, M_l)^T \in \mathbb{Z}_p^{l \times n}$), $\{PK_{aid}\}$ (i.e., the related authorities' public key set), and $\{PK_{x_{aid}}\}$ (i.e., the public keys of related attributes).

Firstly, it picks a random value $s \in \mathbb{Z}_p$ as well as a random vector $v = (s, v_2, v_3, \dots, v_n)$. For each $i \in [I]$, this algorithm calculates $\lambda_i = M_i v^T$ as a secret sharing of s corresponding to $\rho(i)$. And it selects a random value $r_i \in \mathbb{Z}_p$ for every row $i \in [I]$ of M and outputs

$$CT = \{(M, \rho), C_0, \{C_{1,i}, C_{2,i}, C_{3,i}, C_{4,i}\}_{i \in [I]}\},$$

as the ciphertext, where $C_0 = Msg \cdot e(g, g)^s$, $C_{1,i} = e(g, g)^{\lambda_i} \cdot PK_{1,\delta(i)}^{r_i}$, $C_{2,i} = g^{-r_i}$, $C_{3,i} = PK_{2,\delta(i)}^{r_i}$, $C_{4,i} = PK_{\rho(i)}^{r_i}$.

4.4 Data decryption

In this step, the data decryption algorithm *Decrypt* is executed by a data user.

Decrypt $(GP, CT, SK_{S,uid}) \rightarrow Msg$ or $\perp$: The input of the decryption algorithm is GP (i.e., the global parameters), CT (i.e., a ciphertext for an LSSS access policy (M, ρ) with $M = (M_1, \dots, M_l)^T \in \mathbb{Z}_p^{l \times n}$), and $SK_{S,uid}$ (i.e., attribute private key of the user with identity uid and an attribute set S). It correctly decrypts the message Msg if S satisfies (M, ρ) . Or else, it returns back a special symbol $\perp$, implying the decryption is failed.

According to the access policy (M, ρ) , denote $I \subseteq [I]$ as $I = \{i : \rho(i) \in S\}$. This algorithm calculates D_i for each row $i \in I$ as $D_i = C_{1,i} \cdot e(C_{2,i}, K_{1,\rho(i),uid}) \cdot e(C_{3,i}, H(K_{3,uid})) \cdot e(C_{4,i}, K_{2,\rho(i),uid})$.

It finds a constant value set $\{c_i \in \mathbb{Z}_p\}_{i \in I}$ satisfying $\sum_{i \in I} c_i M_i = (1, 0, \dots, 0)$ and calculates

$$e(g, g)^s = \prod_{i \in I} D_i^{c_i}.$$

The message is ultimately got back as $Msg = C_0 / e(g, g)^s$.

4.5 Outsourcing decryption

This step is made up of three algorithms: GenTransKey executed by a data user, PDecrypt performed by the outsourced server, as well as OutsourcingDecrypt handled by the data user.

GenTransKey($GP, CT, SK_{S,uid}$) $\rightarrow TK_{S,uid}, RK_{S,uid}, CT'$: or else

This algorithm takes as inputs the global parameters GP , a ciphertext CT for an LSSS access policy (M, ρ) where $M = (M_1, \dots, M_l)^\top \in \mathbb{Z}_p^{l \times n}$, and attribute private key $SK_{S,uid}$ of the user with an attribute set S and identity uid . It selects a random value $z \in \mathbb{Z}_p$ and outputs $RK_{S,uid}$, CT' and $TK_{S,uid}$ as

$$RK_{S,uid} = z,$$

$$CT' = \{(M, \rho), C_0, \{C'_{1,x} = C_{1,x}^{1/z}, C_{2,x}, C'_{3,x} = C_{3,x}^{1/z}, C_{4,x}\}_{x \in [l]}\},$$

$$TK_{S,uid} = \{TK_{1,i,uid}, TK_{2,i,uid}\}_{i \in S},$$

where $TK_{1,i,uid} = K_{1,i,uid}^{1/z}$, $TK_{2,i,uid} = K_{2,i,uid}^{1/z}$.

PDecrypt($GP, CT', uid, TK_{S,uid}$) $\rightarrow CT'_{out}$: The input of this algorithm is GP , a ciphertext CT' for (M, ρ) where $M = (M_1, \dots, M_l)^\top \in \mathbb{Z}_p^{l \times n}$, a user identity uid , and transformation key $TK_{S,uid}$ of the user with an attribute set S and identity uid .

According to (M, ρ) , denote $I = \{i : \rho(i) \in S\} \subseteq [l]$. The algorithm calculates D_i for each $i \in I$ as $D_i = C'_{1,i} \cdot e(C_{2,i}, TK_{1,\rho(i),uid}) \cdot e(C'_{3,i}, H(uid)) \cdot e(C_{4,i}, TK_{2,\rho(i),uid})$.

It finds a constant set $\{c_i \in \mathbb{Z}_p\}_{i \in I}$ satisfying $\sum_{i \in I} c_i M_i = (1, 0, \dots, 0)$ and produces the partially decrypted ciphertext CT'_{out} as $CT'_{out} = \{(M, \rho), C_0, D\}$ where $D = \prod_{i \in I} D_i^{c_i}$.

OutsourcingDecrypt($GP, CT'_{out}, RK_{S,uid}$) $\rightarrow Msg$ or $\perp$: The inputs of the outsourcing decryption algorithm are GP , a partially decrypted ciphertext CT'_{out} provided by the outsourced server, and reserved key $RK_{S,uid}$ of the user with an attribute set S and identity uid . This procedure returns back a special symbol $\perp$ if S fails to satisfy (M, ρ) emended in CT'_{out} , indicating that decryption is not possible. Otherwise, the algorithm correctly outputs the decrypted message Msg as

$$Msg = C_0 / (D^{RK_{S,uid}}).$$

4.6 Attribute revocation

This step is comprised of three algorithms: UKeyGen, SKUpdate, as well as CTUpdate.

UKeyGen($x_{aid}, VK_{x_{aid}}$) $\rightarrow \widehat{VK}_{x_{aid}}, AUK_{x_{aid}}$: The inputs of this algorithm are an attribute x_{aid} to revoke and the version key $VK_{x_{aid}}$ of x_{aid} . The authority with identity $aid = T(x_{aid})$ chooses an element a uniformly at random from $\mathbb{Z}_p$ and produces the new version key $\widehat{VK}_{x_{aid}}$ of x_{aid} along with an updated key $AUK_{x_{aid}}$ as

$$\widehat{VK}_{x_{aid}} = a, \quad AUK_{x_{aid}} = a - VK_{x_{aid}}.$$

SKUpdate($SK_{S,uid}, AUK_{x_{aid}}$) $\rightarrow \widehat{SK}_{S,uid}$: The inputs of the key update algorithm are an updated key $AUK_{x_{aid}}$ about the revoked attribute x_{aid} , as well as the private key $SK_{S,uid}$ of the user with an attribute set S and identity uid . For each attribute $i \in S$, if i is the revoked attribute x_{aid} , the relevant authority with identity $aid = T(x_{aid})$ calculates

$$K'_{1,i,uid} = K_{1,i,uid} \cdot K_{2,i,uid}^{AUK_{x_{aid}}},$$

$$K'_{1,i,uid} = K_{1,i,uid}.$$

The user with identity uid utilizes the above private key components to obtain an updated private key $\widehat{SK}_{S,uid}$ as

$$\widehat{SK}_{S,uid} = \{K_{3,uid}, \{K'_{1,i,uid}, K_{2,i,uid}\}_{i \in S}\}.$$

CTUpdate($CT, AUK_{x_{aid}}$) $\rightarrow \widehat{CT}$: The inputs of the ciphertext update algorithm are CT for LSSS access policy (M, ρ) where $M = (M_1, \dots, M_l)^\top \in \mathbb{Z}_p^{l \times n}$ as well as an updated key $AUK_{x_{aid}}$ about the revoked attribute x_{aid} . For every row $i \in [l]$, if $\rho(i)$ is the revoked attribute x_{aid} , cloud server computes the ciphertext component $C'_{4,i}$ as

$$C'_{4,i} = C_{4,i} \cdot (1/C_{2,i})^{AUK_{x_{aid}}}.$$

Otherwise, $C'_{4,i} = C_{4,i}$.

It ultimately outputs the updated ciphertext $\widehat{CT}$ as $\widehat{CT} = \{(M, \rho), C_0, \{C_{1,i}, C_{2,i}, C_{3,i}, C'_{4,i}\}_{i \in [l]}\}.$

4.7 Policy update

This phase consists of three algorithms: PolicyCompare, PKUpdate and PCTUpdate.

PolicyCompare($(M', \rho'), (M, \rho)$) $\rightarrow I_1, I_2, I_3$: Referring to [31], this algorithm compares the new LSSS access policy (M', ρ') with the previous one (M, ρ) where $M' = (M'_1, \dots, M'_{l'})^\top \in \mathbb{Z}_p^{l' \times n'}$ and $M = (M_1, \dots, M_l)^\top \in \mathbb{Z}_p^{l \times n}$ respectively, and generates three sets of row indexes I_1, I_2 , and I_3 for M' (see Fig. 2 for details).

PKUpdate($GP, (M', \rho'), (M, \rho), \{I_1, I_2, I_3\}, Sha-resInfo(M)$, $\{PK_{\delta(j)}\}_{(j,0) \in I_3}, \{PK_{\rho'(j)}\}_{(j,0) \in I_3}$) $\rightarrow PUK_M$: The input of this algorithm is GP , a new policy (M', ρ') where $M' = (M'_1, \dots, M'_{l'})^\top \in \mathbb{Z}_p^{l' \times n'}$, the previous policy (M, ρ) with $M = (M_1, \dots, M_l)^\top \in \mathbb{Z}_p^{l \times n}$, three sets of row indexes I_1, I_2 , and I_3 for M' , the secret sharing information $Sha-resInfo(M)$ containing s and $\{\lambda_i\}_{i \in [l]}$ about M , the public keys $\{PK_{\delta(j)}\}_{(j,0) \in I_3}$ of corresponding authorities and public keys $\{PK_{\rho'(j)}\}_{(j,0) \in I_3}$ of corresponding attributes.

A random vector $v' = (s, v'_2, v'_3, \dots, v'_n)$ is chosen firstly. For every $j \in [l']$, this algorithm calculates $\lambda'_j = M'_j \cdot v'^\top$ as a secret sharing of s corresponding to $\rho'(j)$. Subsequently, it selects a value a_j uniformly at random from $\mathbb{Z}_p$ for every index $(j, i) \in I_2$ and a value r'_j uniformly at random from $\mathbb{Z}_p$ for every index $(j, 0) \in I_3$. This procedure ultimately outputs the ciphertext update key PUK_M as

$$PUK_M = \{UK_1, UK_2, UK_3\},$$

where

$$UK_1 = \{UK_{1,j,i,M} = g^{\lambda'_j - \lambda_i}\}_{(j,i) \in I_1},$$

$$UK_2 = \{(a_j, UK_{1,j,i,M} = g^{\lambda'_j - a_j \cdot \lambda_i})\}_{(j,i) \in I_2},$$

$$UK_3 = \{(UK_{1,j,i,M} = g^{\lambda'_j + \alpha_{\delta(j)} \cdot r'_j}, UK_{2,j,i,M} = g^{-r'_j},$$

$$UK_{3,j,i,M} = g^{\beta_{\delta(j)} \cdot r'_j}, UK_{4,j,i,M} = PK_{\rho'(j)}^{\lambda'_j})\}_{(j,0) \in I_3}.$$

Input: The new access policy (M', ρ') where M' is an $l' \times n'$ matrix; The previous access policy (M, ρ) where M is an $l \times n$ matrix;
Output: I_1, I_2, I_3 ; // 3 subsets of row indexes in M'

```

1:  $I_M \leftarrow$  index set of rows in  $M$ ;
2: for  $j = 1$  to  $l'$  do
3:   if  $\rho'(j)$  in  $M$  then
4:     if  $I_M \neq \emptyset$  &  $\exists i \in I_M$  s.t.  $\rho(i) == \rho'(j)$  then
5:       add  $(j, i)$  into  $I_1$ ;
6:       delete  $i$  from  $I_M$ ;
7:     else
8:       find any  $i \in [1, l]$  s.t.  $\rho(i) == \rho'(j)$ ;
9:       add  $(j, i)$  into  $I_2$ ;
10:    end if
11:  else
12:    add  $(j, 0)$  into  $I_3$ ;
13:  end if
14: end for

```

Fig. 2 PolicyCompare algorithm [31]

PCTUpdate $(CT, (M', \rho'), \{I_1, I_2, I_3\}, PUK_M) \rightarrow CT'$: The inputs of the algorithm are a ciphertext CT to update, a new (M', ρ') with $M' = (M'_1, \dots, M'_{l'})^T \in \mathbb{Z}_p^{l' \times n'}$, three sets of row indexes I_1, I_2 , and I_3 for M' , and the ciphertext update key PUK_M .

If $(j, i) \in I_1$, it computes

$$C'_{1,j} = C_{1,i} \cdot e(g, UK_{1,j,i,M}),$$

$$C'_{t,j} = C_{t,i} \quad \text{for each } t \in \{2, 3, 4\},$$

as the ciphertext components.

If $(j, i) \in I_2$, it computes

$$C'_{1,j} = C_{1,i}^{a_j} \cdot e(g, UK_{1,j,i,M}),$$

$$C'_{t,j} = C_{t,i}^{a_j} \quad \text{for each } t \in \{2, 3, 4\},$$

as the ciphertext components.

If $(j, 0) \in I_3$, it computes

$$C'_{1,j} = e(g, UK_{1,j,i,M}),$$

$$C'_{t,j} = UK_{t,j,i,M} \quad \text{for each } t \in \{2, 3, 4\},$$

as the ciphertext components.

It ultimately calculates

$$CT' = \{(M', \rho'), C_0, \{C'_{1,j}, C'_{2,j}, C'_{3,j}, C'_{4,j}\}_{j \in [l']}\},$$

as the updated ciphertext.

4.8 Traceability

In this step, the traceability algorithm Trace is performed by the tracking server.

Trace $(GP, SK_{S,uid}, \{PK_{T(i)}\}_{i \in S}, \{PK_i\}_{i \in S}) \rightarrow uid$ or $\perp$: The inputs of this algorithm are GP , a tracked user private key $SK_{S,uid}$, the public keys $\{PK_{T(i)}\}_{i \in S}$ of corresponding authorities and public keys $\{PK_i\}_{i \in S}$ of corresponding attributes. If $SK_{S,uid}$ is not in the following form

$$SK_{S,uid} = \{K_{3,uid}, \{K_{1,i,uid}, K_{2,i,uid}\}_{i \in S}\},$$

or cannot succeed in the key sanity check, it returns back “ $\perp$ ” implying $SK_{S,uid}$ does not need to be traced, or else this algorithm outputs a user identity uid holding the private key $SK_{S,uid}$ when it passes the key sanity check procedure as follows.

There exists an attribute $i \in S$ that satisfies:

$$K_{1,i,uid}, K_{2,i,uid} \in G, \quad K_{3,uid} \in \mathbb{Z}_p^*,$$

$$e(g, K_{1,i,uid}) = PK_{1,T(i)} \cdot e(PK_{2,T(i)}, H(K_{3,uid}))$$

$$\cdot e(K_{2,i,uid}, PK_i).$$

5 Correctness and security analysis

Now we provide the analysis of correctness and security for our proposed cryptosystem, including correctness of encryption and decryption, correctness of outsourcing decryption, static security, forward security, and collusion prevention.

5.1 Correctness of encryption and decryption

Assume the data users' resources are not constrained, based on the aforementioned scheme, we know that

$$D_i = C_{1,i} \cdot e(C_{2,i}, K_{1,\rho(i),uid}) \cdot e(C_{3,i}, H(K_{3,uid}))$$

$$\cdot e(C_{4,i}, K_{2,\rho(i),uid}).$$

Take the components in CT and $SK_{S,uid}$ into it for calculation, then

$$D_i = C_{1,i} \cdot e(C_{2,i}, K_{1,\rho(i),uid}) (C_{3,i}, H(K_{3,uid}))$$

$$\cdot e(C_{4,i}, K_{2,\rho(i),uid})$$

$$= e(g, g)^{\lambda_i} \cdot PK_{1,\delta(i)}^{r_i} \cdot e(g^{-r_i}, g^{\alpha_{aid}} \cdot H(uid))^{\beta_{aid}}$$

$$\cdot (g^{VK_{\rho(i)}} \cdot F(\rho(i)))^t \cdot e(PK_{2,\delta(i)}^{r_i}, H(uid)) \cdot e(PK_{\rho(i)}^{r_i}, g^t)$$

$$= e(g, g)^{\lambda_i} \cdot e(g, g)^{\alpha_{\delta(i)} r_i} \cdot e(g, g)^{-r_i \alpha_{aid}} \cdot e(g, H(uid))^{-r_i \beta_{aid}}$$

$$\cdot e(g, PK_{\rho(i)})^{-r_i t} \cdot e(g, H(uid))^{\beta_{\delta(i)} r_i} \cdot e(g, PK_{\rho(i)})^{r_i t}$$

$$= e(g, g)^{\lambda_i}.$$

If the attribute set S meets (M, ρ) , there are constants $\{c_i \in \mathbb{Z}_p\}_{i \in I}$ satisfying $\sum_{i \in I} c_i M_i = (1, 0, \dots, 0)$. From this, we can infer the following conclusion:

$$\sum_{i \in I} \lambda_i c_i = \sum_{i \in I} M_i v^\top c_i = (1, 0, \dots, 0) v^\top = s.$$

Accordingly,

$$\prod_{i \in I} D_i^{c_i} = \prod_{i \in I} e(g, g)^{\lambda_i c_i} = e(g, g)^s.$$

Therefore, the decryption algorithm can generate the plaintext correctly:

$$Msg = C_0 / \prod_{i \in I} D_i^{c_i}.$$

5.2 Correctness of outsourcing decryption

Assuming that the resources of data users are constrained, based on the aforementioned phase of outsourcing decryption in our scheme, we know that

$$D_i = C'_{1,i} \cdot e(C_{2,i}, TK_{1,\rho(i),uid}) \cdot e(C'_{3,i}, H(uid)) \cdot e(C_{4,i}, TK_{2,\rho(i),uid}).$$

Take the components in CT , CT' , $SK_{S,uid}$ and $TK_{S,uid}$ into it for calculation, then

$$\begin{aligned} D_i &= C'_{1,i} \cdot e(C_{2,i}, TK_{1,\rho(i),uid}) \cdot e(C'_{3,i}, H(uid)) \\ &\quad \cdot e(C_{4,i}, TK_{2,\rho(i),uid}) \\ &= C_{1,i}^{1/z} \cdot e(C_{2,i}, K_{1,\rho(i),uid}^{1/z}) \cdot e(C_{3,i}^{1/z}, H(uid)) \\ &\quad \cdot e(C_{4,i}, K_{2,\rho(i),uid}^{1/z}) \\ &= (C_{1,i} \cdot e(C_{2,i}, K_{1,\rho(i),uid}) \cdot e(C_{3,i}, H(uid)) \\ &\quad \cdot e(C_{4,i}, K_{2,\rho(i),uid}))^{1/z} \\ &= e(g, g)^{\lambda_i/z}. \end{aligned}$$

If the attribute set S satisfies (M, ρ) , there are constants $\{c_i \in \mathbb{Z}_p\}_{i \in I}$ satisfying $\sum_{i \in I} c_i M_i = (1, 0, \dots, 0)$. From this, we can infer the following conclusion:

$$\sum_{i \in I} \lambda_i c_i = \sum_{i \in I} M_i v^\top c_i = (1, 0, \dots, 0) v^\top = s.$$

Accordingly,

$$\prod_{i \in I} D_i^{c_i} = \prod_{i \in I} e(g, g)^{\lambda_i/z \cdot c_i} = e(g, g)^{s/z}.$$

It is proved that the outsourcing decryption algorithm can correctly generate the plaintext correctly:

$$M = \begin{bmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,n} \\ M_{2,1} & M_{2,2} & \cdots & M_{2,n} \\ M_{3,1} & M_{3,2} & \cdots & M_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{l,1} & M_{l,2} & \cdots & M_{l,n} \end{bmatrix} \longrightarrow M' = \begin{bmatrix} 0 & \cdots & 0 & M'_{1,n-c+1} & \cdots & M'_{1,n} \\ M'_{2,1} & \cdots & M'_{2,n-c} & M'_{2,n-c+1} & \cdots & M'_{2,n} \\ 0 & \cdots & 0 & M'_{3,n-c+1} & \cdots & M'_{3,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M'_{l,1} & \cdots & M'_{l,n-c} & M'_{l,n-c+1} & \cdots & M'_{l,n} \end{bmatrix}$$

Fig. 3 Transformation of M used by the simulator [48]

$$Msg = C_0 / \left(\prod_{i \in I} D_i^{c_i} \right)^z.$$

5.3 Static security

The static security proof of our cryptosystem is demonstrated by the following theorem using the static security model given in Subsection 3.4. We borrow the proof idea of Theorem 1 in [48] except that the static security here is based on a q -type assumption called q -1 assumption while the static security in [48] is based on the q -DPBDHE2 assumption.

Theorem 1 Assume that the q -1 assumption holds. Then under the random oracle model, all probabilistic polynomial-time attackers with a challenge matrix of size at most $q \times q$ have a negligible advantage in statically breaking our scheme constructed in Section 4.

Proof Assume there exists an attacker $\mathcal{A}$ breaking our scheme in no more than probabilistic polynomial-time with a non-negligible advantage ϵ , then an algorithm $\mathcal{B}$ which is probabilistic polynomial-time computable can be constructed. The static security game with $\mathcal{A}$ is simulated by $\mathcal{B}$ to solve the q -type problem with advantage ϵ' .

For an access structure $\mathbb{A}$, denote $M \in \mathbb{Z}_p^{l \times n}$ as a linear secret sharing scheme's secret sharing matrix, and assume that $C \subseteq [l]$ is a non-authorized set of rows. Spanned by C 's rows, we get the subspace, the dimension of which is denoted as $c \in N$. Then the transformation matrix of M to be used by the simulator is M' as shown in Fig. 3, where $M'_{x,j} = 0$ for every $(x, j) \in C \times [n-c]$ ($n' = n - c$) and we highlight the rows belonging to corrupted authorities [48].

Let H and F be two random oracles programmed by the simulator and the simulator works below.

Initialization. The simulator receives (D, T) from the q -type challenger, sends $GP = (G, p, g, U, U_{AA}, T)$ to the attacker $\mathcal{A}$. Grounded on the static security model, $\mathcal{A}$ produces the components U_{co-AA} , $U_{no-co-AA}$, $Q = \{(uid_i, S_i)\}_{i \in [m]}$, $(Msg_0, Msg_1) \in G_T^2$, and (M, ρ) . Furthermore, $\mathcal{A}$ generates two sequences L_I including global identities, $L \subseteq U$ including attributes for the H oracle and F oracle queries respectively. Subsequently, the secret sharing matrix M is replaced with the transformation matrix M' by the simulator, where $C = U_{co-AA}$ and proceeds to calculate each of the inputs to the attacker $\mathcal{A}$.

Authority public keys. $\mathcal{B}$ provides each non-corrupted authority's public key in $U_{no-co-AA}$, which can be divided into the following two cases:

- (i) If authority $aid \in U_{no-co-AA}$ isn't in the challenge policy

(M, ρ) , that is, $aid \notin \delta[l]$, $\mathcal{B}$ chooses random values $\alpha_{aid}, \beta_{aid} \in \mathbb{Z}_p$ and generates $(e(g, g)^{\alpha_{aid}}, g^{\beta_{aid}})$ as the public key.

(ii) If authority $aid \in U_{no-co-AA}$ is in the challenge policy (M, ρ) , that is, $aid \in \delta[l] \setminus U_{co-AA}$, denote $X = \{x | \delta(x) = aid\} \subseteq [l]$. $\mathcal{B}$ selects random values $\alpha'_{aid}, \beta'_{aid} \in \mathbb{Z}_p$, sets $\alpha_{aid} = \alpha'_{aid} + \sum_{x \in X} b_x a^{q+1} M'_{x,1}$ as well as $\beta_{aid} = \beta'_{aid} + \sum_{x \in X} \sum_{j=2}^{n'} b_x a^{q+2-j} M'_{x,j}$ implicitly. Then, it outputs

$$(e(g, g)^{\alpha_{aid}}, g^{\beta_{aid}}) = \left(e(g, g)^{\alpha'_{aid}} \prod_{x \in X} e(g^{b_x a}, g^{a^q})^{M'_{x,1}}, g^{\beta'_{aid}} \prod_{x \in X} \prod_{j=2}^{n'} (g^{b_x a^{q+2-j}})^{M'_{x,j}} \right),$$

as the public key.

H-oracle queries. We divide this step into three cases below:

(i) When the queried identity $uid \in L_I$ and $uid \notin \{uid_i\}_{i \in [m]}$, $\mathcal{B}$ picks a random $h \in G$ for $H(uid)$, which won't be used anywhere else.

(ii) When the queried identity uid equals $uid_i \in \{uid_i\}_{i \in [m]}$ and there does not exist a row x satisfying $\rho(x) \in S_i$, $\mathcal{B}$ picks a random value $h_i \in \mathbb{Z}_p$ and generates

$$H(uid_i) = g^{h_i} \cdot g^a \cdot g^{a^2} \cdot \dots \cdot g^{a^{n'-1}} = g^{h_i} \prod_{k=2}^{n'} g^{a^{k-1}}.$$

(iii) When the queried uid equals $uid_i \in \{uid_i\}_{i \in [m]}$ and there are some rows $X' \subseteq [l]$ satisfying $\rho(X') \subseteq S_i$, there exists a vector $d_i \in \mathbb{Z}_p^{1 \times n}$ with $d_{i,1} = 1$ as its first element. The inner product of each of the aforementioned rows with d_i equals zero. Then $\mathcal{B}$ chooses a random value $h_i \in \mathbb{Z}_p$ and generates

$$H(uid_i) = g^{h_i} \cdot (g^a)^{d_{i,2}} \cdot (g^{a^2})^{d_{i,3}} \cdot \dots \cdot (g^{a^{n'-1}})^{d_{i,n'}} = g^{h_i} \prod_{k=2}^{n'} (g^{a^{k-1}})^{d_{i,k}}.$$

F-oracle queries. We divide this step into two cases below:

(i) Given a queried attribute u and its authority $aid = T(u)$, if $aid \notin \delta[l]$ or $aid \in U_{co-AA}$, the simulator $\mathcal{B}$ picks a random $f \in G$ for $F(u)$, stores f in order that it can be reused in a secret key query.

(ii) When $aid \in \delta[l]$, denote $X'' = \{x | \delta(x) = aid\} \setminus \{x | \rho(x) = u\} \subseteq [l]$, which is the row set belonging to authority aid Other than having u as the corresponding attribute. Subsequently, $\mathcal{B}$ picks a random value $f_u \in \mathbb{Z}_p$ and computes

$$F(u) = g^{f_u} g^{\sum_{x \in X''} \sum_{j \in [n']} b_x a^{q+1-j} M'_{x,j}} = g^{f_u} \prod_{x \in X''} \prod_{j \in [n']} (g^{b_x a^{q+1-j}})^{M'_{x,j}}.$$

Secret keys. Given a query pair (uid_i, S_i) where S_i is a subset of U , $\mathcal{B}$ calculates the components $\{K_{1,u,uid_i}, K_{2,u,uid_i}\}$ of attribute private key for each $u \in S_i$. This step can be divided into three cases below.

(i) Assume that $aid = T(u)$ is the attribute u 's authority. If $aid \notin \delta[l]$, $\mathcal{B}$ knows α_{aid} and β_{aid} . Then it picks a random value

$t \in \mathbb{Z}_p$ and generates $K_{1,u,uid_i} = g^{\alpha_{aid}} H(uid_i)^{\beta_{aid}} (g^{VK_u} \cdot F(u))^t$ and $K_{2,u,uid_i} = g^t$.

(ii) If $aid \in \delta[l]$ and $S_i \cap \rho[l] = \emptyset$, it is known that $H(uid_i) = g^{h_i} g^{\sum_{k=2}^{n'} a^{k-1}}$ and $F(u) = g^{f_u} g^{\sum_{x \in X} \sum_{j \in [n']} b_x a^{q+1-j} M'_{x,j}}$ according to the H -oracle and F -oracle, where $X = \{x | \delta(x) = aid\}$. Subsequently, $\mathcal{B}$ sets $t = -\sum_{k \in [n']} a^k$ implicitly and calculates the key components:

$$\begin{aligned} K_{1,u,uid_i} &= g^{\alpha_{aid}} \cdot H(uid_i)^{\beta_{aid}} \cdot (g^{VK_u} \cdot F(u))^t \\ &= g^{\sum_{x \in X} b_x a^{q+1} M'_{x,1}} g^{\sum_{x \in X} \sum_{j=2}^{n'} \sum_{k=2}^{n'} b_x a^{q+1+k-j} M'_{x,j}} \\ &\quad \cdot g^{-\sum_{x \in X} \sum_{j \in [n']} \sum_{k \in [n']} b_x a^{q+1+k-j} M'_{x,j}} \\ &\quad \cdot g^{\alpha'_{aid}} H(uid_i)^{\beta'_{aid}} (g^{\beta_{aid}-\beta'_{aid}})^{h_i} (g^t)^{f_u} (g^t)^{VK_u} \\ &= g^{-\sum_{x \in X} \sum_{j=2}^{n'} b_x a^{q+2-j} M'_{x,j}} g^{-\sum_{x \in X} \sum_{k=2}^{n'} b_x a^{q+k} M'_{x,1}} \\ &\quad \cdot g^{\alpha'_{aid}} H(uid_i)^{\beta'_{aid}} (g^{\beta_{aid}-\beta'_{aid}})^{h_i} (g^t)^{f_u} (g^t)^{VK_u} \\ &= g^{\alpha'_{aid}} H(uid_i)^{\beta'_{aid}} (g^{\beta_{aid}-\beta'_{aid}})^{h_i} (g^t)^{f_u} (g^t)^{VK_u} \\ &\quad \cdot \prod_{x \in X} \prod_{j=2}^{n'} (g^{b_x a^{q+2-j}})^{-M'_{x,j}} \cdot \prod_{x \in X} \prod_{k=2}^{n'} (g^{b_x a^{q+k}})^{-M'_{x,1}}, \\ K_{2,u,uid_i} &= g^t = \prod_{k \in [n']} (g^{a^k})^{-1}. \end{aligned}$$

(iii) If $aid \in \delta[l]$ and $S_i \cap \rho[l] \neq \emptyset$, it is known that $H(uid_i) = g^{h_i} g^{\sum_{k=2}^{n'} a^{k-1} d_{i,k}}$ and $F(u) = g^{f_u} g^{\sum_{x \in X''} \sum_{j \in [n']} b_x a^{q+1-j} M'_{x,j}}$ according to the H -oracle and F -oracle. Subsequently, the simulator $\mathcal{B}$ sets implicitly $t = -\sum_{k \in [n']} a^k d_{i,k}$ and calculates the key components:

$$\begin{aligned} K_{1,u,uid_i} &= g^{\alpha_{aid}} \cdot H(uid_i)^{\beta_{aid}} \cdot (g^{VK_u} \cdot F(u))^t \\ &= g^{\sum_{x \in X} b_x a^{q+1} M'_{x,1}} g^{\sum_{x \in X} \sum_{j=2}^{n'} \sum_{k=2}^{n'} b_x a^{q+1+k-j} M'_{x,j} d_{i,k}} \\ &\quad \cdot g^{-\sum_{x \in X''} \sum_{j \in [n']} \sum_{k \in [n']} b_x a^{q+1+k-j} M'_{x,j} d_{i,k}} \\ &\quad \cdot g^{\alpha'_{aid}} H(uid_i)^{\beta'_{aid}} (g^{\beta_{aid}-\beta'_{aid}})^{h_i} (g^t)^{f_u} (g^t)^{VK_u} \\ &= g^{\sum_{x \in X \setminus X''} b_x a^{q+1} M'_{x,1}} g^{\sum_{x \in X \setminus X''} \sum_{j=2}^{n'} \sum_{k=2}^{n'} b_x a^{q+1+k-j} M'_{x,j} d_{i,k}} \\ &\quad \cdot g^{-\sum_{x \in X''} \sum_{j=2}^{n'} b_x a^{q+2-j} M'_{x,j} d_{i,1}} g^{-\sum_{x \in X''} \sum_{k=2}^{n'} b_x a^{q+k} M'_{x,1} d_{i,k}} \\ &\quad \cdot g^{\alpha'_{aid}} H(uid_i)^{\beta'_{aid}} (g^{\beta_{aid}-\beta'_{aid}})^{h_i} (g^t)^{f_u} (g^t)^{VK_u} \\ &= \prod_{x \in X \setminus X''} \prod_{j=2}^{n'} \prod_{k=2}^{n'} (g^{b_x a^{q+1+k-j}})^{M'_{x,j} d_{i,k}} \\ &\quad \cdot \prod_{x \in X''} \prod_{j=2}^{n'} (g^{b_x a^{q+2-j}})^{-M'_{x,j} d_{i,1}} \\ &\quad \cdot \prod_{x \in X''} \prod_{k=2}^{n'} (g^{b_x a^{q+k}})^{-M'_{x,1} d_{i,k}} \\ &\quad \cdot g^{\alpha'_{aid}} H(uid_i)^{\beta'_{aid}} (g^{\beta_{aid}-\beta'_{aid}})^{h_i} (g^t)^{f_u} (g^t)^{VK_u}, \\ K_{2,u,uid_i} &= g^t = \prod_{k \in [n']} (g^{a^k})^{-d_{i,k}}. \end{aligned}$$

Challenge ciphertext. The simulator $\mathcal{B}$ sets implicitly $s = s'a^{q+1}$ and C_0 is calculated as $C_0 = M s g_b \cdot T$, where T is the challenge term, and a bit b is random from $\{0, 1\}$. Simultaneously, $\mathcal{B}$ sets $v = (s'a^{q+1}, 0, \dots, 0) \in \mathbb{Z}_p^n$ implicitly.

The remaining components of the ciphertext are calculated in the following two cases:

(i) Given a row $x^* \in [I]$, let $aid = \delta(x^*)$ be the authority of the row x^* . If $aid \in U_{co-AA}$, the simulator $\mathcal{B}$ chooses an element r_{x^*} randomly from $\mathbb{Z}_p$, employing aid 's public key $(e(g, g)^{\alpha_{aid}}, g^{\beta_{aid}})$ it generates

$$\begin{aligned} C_{1,x^*} &= e(g, g)^{\lambda_{x^*}} \cdot PK_{1,aid}^{r_{x^*}} = (e(g, g)^{\alpha_{aid}})^{r_{x^*}}, \\ C_{2,x^*} &= g^{-r_{x^*}}, \\ C_{3,x^*} &= PK_{2,aid}^{r_{x^*}} = (g^{\beta_{aid}})^{r_{x^*}}, \\ C_{4,x^*} &= PK_{\rho(x^*)}^{r_{x^*}} = (g^{VK_{\rho(x^*)}} F(\rho(x^*)))^{r_{x^*}}. \end{aligned}$$

(ii) If $aid \notin U_{co-AA}$, we have that $\lambda_{x^*} = s'a^{q+1} \cdot M'_{x^*,1}$. The simulator $\mathcal{B}$ sets $r_{x^*} = -s'/b_{x^*}$ implicitly and generates

$$\begin{aligned} C_{1,x^*} &= e(g, g)^{\lambda_{x^*}} \cdot (e(g, g)^{\alpha_{aid}})^{r_{x^*}} \\ &= e(g, g)^{s'a^{q+1} M'_{x^*,1}} e(g, g)^{-\sum_{x \in X} s' b_x a^{q+1} M'_{x^*,1} / b_{x^*}} \\ &= \prod_{x \in X \setminus \{x^*\}} e(g, g^{s' b_x a^{q+1} / b_{x^*}})^{-M'_{x^*,1}}, \\ C_{2,x^*} &= g^{-r_{x^*}} = g^{s' / b_{x^*}}, \\ C_{3,x^*} &= (g^{\beta_{aid}})^{r_{x^*}} = g^{-\sum_{x \in X} \sum_{j=2}^{n'} s' b_x a^{q+2-j} M'_{x^*,j} / b_{x^*}} \\ &= \prod_{x \in X} \prod_{j=2}^{n'} (g^{s' b_x a^{q+2-j} / b_{x^*}})^{-M'_{x^*,j}}, \\ C_{4,x^*} &= (g^{VK_{\rho(x^*)}} F(\rho(x^*)))^{r_{x^*}} \\ &= g^{-s' VK_{\rho(x^*)} / b_{x^*}} \cdot g^{-\sum_{x \in X''} \sum_{j \in [n']} s' b_x a^{q+1-j} M'_{x^*,j} / b_{x^*}} \\ &= g^{-s' VK_{\rho(x^*)} / b_{x^*}} \prod_{x \in X''} \prod_{j \in [n']} (g^{s' b_x a^{q+1-j} / b_{x^*}})^{-M'_{x^*,j}}. \end{aligned}$$

Guess. If $\mathcal{A}$ guesses b correctly, $\mathcal{B}$ will output that the challenge term is $e(g, g)^{s'a^{q+1}}$, effectively simulating the static security game accurately. If $\mathcal{A}$ does not guess b correctly, $\mathcal{B}$ will indicate that T is a random group element of G_T , meaning it generates a random message's encryption. Hence, if the attack $\mathcal{A}$ succeeds with an advantage that is not negligible, so is the simulator $\mathcal{B}$. $\square$

5.4 Forward security

In our proposed multi-functional MA-ABE scheme, we have the following analysis.

(i) The users that have lost the revoked attribute are unable to construct new attribute private keys. In the attribute revocation mechanism proposed in this paper, the update of attribute private keys relies on an updated key, which is generated by the relevant authority using both the old version key and new version key, which are retained as private parameters exclusively controlled by the authority. Consequently, users who have lost the revoked attribute cannot construct new attribute private keys with the updated key, thus preventing them from decrypting the associated ciphertexts.

(ii) The users that have lost the revoked attribute can no longer access data associated with that attribute unless their remaining attributes still satisfy the data's access policy. Following attribute revocation, the users that have lost the revoked attribute cannot timely update and share the ciphertext and keys. As a result, these users can only reconstruct the secret values corresponding to the previously held attributes, but these values cannot be used for subsequent decryption. Consequently, users that have lost the revoked attribute are unable to decrypt the relevant ciphertext with their previous keys, guaranteeing the forward security of our cryptosystem according to Definition 2.

5.5 Resistance to collusion attacks

In our proposed multi-functional MA-ABE, we have the following facts.

(i) The attribute authorities of our proposed scheme produce private keys for users within their management scope, such that the private key components corresponding to each attribute of the user are associated with a random value t .

(ii) In our presented cryptosystem, every user holds a unique identity uid .

Consequently, there is a unique relationship between each user and its private key in our scheme. Even if the private key components of multiple users are combined together, there will be no additional plaintext information that can be extracted, thus making the system resistant to collusion attacks according to Definition 3.

5.6 Security comparison

Based on the aforementioned security metrics, we compared our proposed scheme with existing schemes [11], [18], and [40], as summarized in Table 1. It is evident that while all four multi-authority schemes meet static security requirements, schemes [11] and [18] lack forward security, and scheme [40] is vulnerable to collusion attacks. In contrast, our proposed scheme simultaneously provides static security, forward security, and resistance to collusion attacks, offering a superior level of system security. Consequently, it

Table 1 Security comparison of our scheme with [11], [18], and [40]

Scheme	Static security	Forward security	Collusion prevention
[11]	✓	×	✓
[18]	✓	×	✓
[40]	✓	✓	×
Our scheme	✓	✓	✓

more effectively addresses the privacy requirements for access control in cloud storage environments and demonstrates enhanced applicability.

6 Functionality and performance evaluation

Now we compare several existing ABE access control schemes with our scheme from the perspectives of functionalities and performance.

6.1 Practical functionality evaluation

With multi-authority, outsourcing decryption, attribute revocation, policy update and traceability as functional indicators, we analyze the functionalities of the schemes in [15], [18], [19], [30], [41], [42], and our proposed scheme respectively. The results are presented in Table 2.

Our proposed scheme differs from the above existing schemes in practical availability. To address the issues of a single authority's untrustworthiness and single-point failure, our ABE scheme is constructed in a multi-authority context. Moreover, we reduce the local users' computational cost by outsourcing the complex decryption to server, enhancing the operational efficiency of system. Simultaneously, to prevent unauthorized access to data due to failed attributes, our scheme offers a timely attribute revocation function to guarantee its forward security. If an access policy of the data stored in cloud undergoes changes, our policy update algorithm can update the ciphertext and keys promptly without needing repetitive encryption that would increase the cost. When a malicious user exists in the system, our scheme can accurately trace its unique identity, which is then added into the list of untrusted users. The analytical results in the table indicate that none of the existing schemes possess

all the aforementioned functional functions simultaneously, and these functions are essential when deploying the ABE scheme in practice. Consequently, our scheme is more functionally complete and efficient in practical applications.

6.2 Storage and computational efficiency evaluation

Storage and computational efficiency are two main aspects of the scheme performance. Hence, we analyze the performance of our scheme and other existing related schemes by taking the public key size, the private key size, the ciphertext size, and the overhead of bilinear operations in decryption as indicators. We use the following symbols to represent different indicator parameters, which are referenced from [18].

- $|U|$: Cardinal number of attribute universe
- $|U_{AA}|$: Cardinal number of authority universe
- $|S|$: Number of the user's attributes
- $|U_u|$: Number of users
- $|I|$: Number of rows used for decryption
- l : Number of rows in the LSSS matrix for the access policy

It should be noted that $|U_{AA}| \leq |U|$, $|S| \leq |U|$, and $|I| \leq l$. Table 3 presents the performance of our scheme and those of schemes [15], [18], [19], [41], and [42] under different indicators. From this, we observe that in the scheme of [19], whether in terms of ciphertext size as well as private key size, or decryption overhead, it all depends on $|U|$, indicating that the system performance will decline as the number of attributes rises, which limits the scalability of cryptosystem. However, the performance parameters in schemes

Table 2 Functionality comparison of our scheme with [15], [18], [19], [30], [41], and [42]

Scheme	Multi-authority	Outsourcing decryption	Attribute revocation	Policy update	Traceability
[15]	×	✓	×	×	×
[18]	✓	✓	×	✓	✓
[19]	✓	×	×	×	×
[30]	×	✓	✓	×	×
[41]	✓	×	×	×	✓
[42]	×	✓	×	×	✓
Our scheme	✓	✓	✓	✓	✓

Table 3 Efficiency comparison of our scheme with [15], [18], [19], [41], and [42]

Scheme	Public key size	Private key size	Ciphertext size	Number of bilinear pairings in decryption
[15]	$ U_u + 1$	$2 S + 3$	$3l + 1$	$3 I + 2$
[18]	$4 U_{AA} $	$4 S + 1$	$5l + 1$	$4 I $
[19]	3	$2 U + 2$	$ U + 3$	$2 U + 1$
[41]	$4 U_{AA} $	$4 S + 1$	$6l + 1$	$3 I $
[42]	7	$2 S + 4$	$3l + 3$	$3 I + 1$
Our scheme	$2 U_{AA} $	$2 S + 1$	$4l + 1$	$3 I $

[15], [18], [41], [42], and our scheme are not dependent on $|U|$, thus enabling a large attribute universe. The public parameter in scheme [15] is reliant on $|U_u|$. Since this value is dynamically changing, it will increase the system storage overhead significantly when the number of users is large. Compared with schemes [18] and [41], it is obvious that our scheme performs better and has higher storage and computational efficiency. While enhancing the computational efficiency, our scheme possesses several practical functions that are lacking in scheme [42] and resolves the security issue resulting from a single authority. In conclusion, our scheme holds greater advantages both in practicality and performance, providing a potent solution to the cloud data sharing problem.

From the comparison results, we conclude that our scheme not only ensures privacy security but also has more comprehensive functions and better performance, making it more suitable for data sharing applications in cloud storage environments.

7 Conclusion

Dedicated to addressing the data sharing issues existing in the cloud storage environment and achieving fine-grained access control, this work proposed a novel multi-functional multi-authority CP-ABE scheme with functionalities including decryption outsourcing, attribute revocation, policy update and traceability. To enhance system efficiency and scalability, we employed the multi-authority feature to distribute the computational burden of key generation across multiple attribute authorities. In our scheme, the modules of policy update and outsourcing decryption can notably decrease the computational expense on user terminals and the communication overhead in this cryptosystem. Among them, a cloud server is capable of independently updating the ciphertext through policy update module, without the requirement for data owner to recalculate the ciphertext according to a new policy locally and subsequently upload it to the cloud server. Simultaneously, data users entrust the complex bilinear decryption to outsourced servers, liberating local storage, and computing resources. Through the construction of attribute revocation mechanism, our scheme can update keys and ciphertexts in a timely manner, preventing unauthorized access to relevant data resulting from variations in the scope of user attributes. Moreover, our scheme is able to track malicious users that might have leaked their attribute private keys of the scheme and add them to the list of untrusted users. Regarding the correctness and security of the proposed cryptosystem, this paper presented formal demonstrations of static security, forward security and resistance to collusion attacks. We evaluated the functionality and performance of our cryptosystem and other existing related cryptosystems. The results make clear that our scheme holds more powerful functions and higher execution efficiency while satisfying the large attribute universe, which can be feasibly applied to data sharing applications in the actual cloud environment.

Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2022ZD0161901), the National Natural Science Foundation of China (Grant No. 62072023), Beijing Natural Science Foundation (Grant No. 4242024), the Open Project Fund of the State

Key Laboratory of Cryptology (No. MMKFKT202120), the Exploratory Optional Project Fund of the State Key Laboratory of Complex & Critical Software Environment, and the Fundamental Research Funds of Beihang University (Nos. YWF-21-BJ-J-1041, YWF-23-L-1033, etc.).

Competing interests

The authors declare that they have no competing interests or financial conflicts to disclose.

References

- [1] Sahai A, Waters B. Fuzzy identity-based encryption. In: Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. 2005, 457–473
- [2] Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. 2006, 89–98
- [3] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: Proceedings of 2007 IEEE Symposium on Security and Privacy. 2007, 321–334
- [4] Cheung L, Newport C. Provably secure ciphertext policy ABE. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. 2007, 456–465
- [5] Goyal V, Jain A, Pandey O, Sahai A. Bounded ciphertext policy attribute based encryption. In: Proceedings of the 35th International Colloquium on Automata, Languages and Programming. 2008, 579–591
- [6] Chen N, Li J, Zhang Y, Guo Y. Efficient CP-ABE scheme with shared decryption in cloud storage. IEEE Transactions on Computers, 2022, 71(1): 175–184
- [7] Liu Z, Ding Y, Yuan M, Wang B. Black-box accountable authority CP-ABE scheme for cloud-assisted e-health system. IEEE Systems Journal, 2023, 17(1): 756–767
- [8] Chase M. Multi-authority attribute based encryption. In: Proceedings of the 4th Theory of Cryptography Conference on Theory of Cryptography. 2007, 515–534
- [9] Chase M, Chow S S M. Improving privacy and security in multi-authority attribute-based encryption. In: Proceedings of the 16th ACM Conference on Computer and Communications Security. 2009, 121–130
- [10] Lewko A, Waters B. Decentralizing attribute-based encryption. In: Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. 2011, 568–588
- [11] Li W, Xue K, Xue Y, Hong J. TMACS: a robust and verifiable threshold multi-authority access control system in public cloud storage. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(5): 1484–1496
- [12] Datta P, Komargodski I, Waters B. Fully adaptive decentralized multi-authority ABE. In: Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. 2023, 447–478
- [13] Li J, Zhang R, Lu Y, Han J, Zhang Y, Zhang W, Dong X. Multiauthority attribute-based encryption for assuring data deletion. IEEE Systems Journal, 2023, 17(2): 2029–2038

- [14] Yang K, Jia X, Ren K, Zhang B. DAC-MACS: effective data access control for multi-authority cloud storage systems. In: Proceedings of 2013 Proceedings IEEE INFOCOM. 2013, 2895–2903
- [15] Ning J, Cao Z, Dong X, Liang K, Ma H, Wei L. Auditable σ -time outsourced attribute-based encryption for access control in cloud computing. *IEEE Transactions on Information Forensics and Security*, 2018, 13(1): 94–105
- [16] Arthur Sandor V K, Lin Y, Li X, Lin F, Zhang S. Efficient decentralized multi-authority attribute based encryption for mobile cloud data storage. *Journal of Network and Computer Applications*, 2019, 129: 25–36
- [17] De S J, Ruj S. Efficient decentralized attribute based access control for mobile clouds. *IEEE Transactions on Cloud Computing*, 2020, 8(1): 124–137
- [18] Sethi K, Pradhan A, Bera P. Practical traceable multi-authority CP-ABE with outsourcing decryption and access policy updation. *Journal of Information Security and Applications*, 2020, 51: 102435
- [19] Wu Y, Li X, Liu Z. Attribute-based keyword searchable encryption scheme for multi-authority in cloud storage. In: Proceedings of 2022 IEEE 22nd International Conference on Communication Technology. 2022, 933–939
- [20] Piretti M, Traynor P, McDaniel P, Waters B. Secure attribute-based systems. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. 2006, 99–112
- [21] Yu S, Wang C, Ren K, Lou W. Attribute based data sharing with attribute revocation. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. 2010, 261–270
- [22] Hur J, Noh D K. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 2011, 22(7): 1214–1221
- [23] Fan K, Liu T, Zhang K, Li H, Yang Y. A secure and efficient outsourced computation on data sharing scheme for privacy computing. *Journal of Parallel and Distributed Computing*, 2020, 135: 169–176
- [24] Yang K, Jia X, Ren K. Attribute-based fine-grained access control with efficient revocation in cloud storage systems. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. 2013, 523–528
- [25] Yang K, Jia X. Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE Transactions on Parallel and Distributed Systems*, 2014, 25(7): 1735–1744
- [26] Imine Y, Lounis A, Bouabdallah A. Revocable attribute-based access control in mutli-authority systems. *Journal of Network and Computer Applications*, 2018, 122: 61–76
- [27] Ge C, Susilo W, Baek J, Liu Z, Xia J, Fang L. Revocable attribute-based encryption with data integrity in clouds. *IEEE Transactions on Dependable and Secure Computing*, 2022, 19(5): 2864–2872
- [28] Xiong H, Huang X, Yang M, Wang L, Yu S. Unbounded and efficient revocable attribute-based encryption with adaptive security for cloud-assisted internet of things. *IEEE Internet of Things Journal*, 2022, 9(4): 3097–3111
- [29] Chen S, Li J, Zhang Y, Han J. Efficient revocable attribute-based encryption with verifiable data integrity. *IEEE Internet of Things Journal*, 2024, 11(6): 10441–10451
- [30] Li J, Zhang E, Han J, Zhang Y, Shen J. PH-MG-ABE: a flexible policy-hidden multigroup attribute-based encryption scheme for secure cloud storage. *IEEE Internet of Things Journal*, 2025, 12(2): 2146–2157
- [31] Yang K, Jia X, Ren K. Secure and verifiable policy update outsourcing for big data access control in the cloud. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 26(12): 3461–3470
- [32] Liu Z, Jiang Z L, Wang X, Yiu S. Practical attribute-based encryption: outsourcing decryption, attribute revocation and policy updating. *Journal of Network and Computer Applications*, 2018, 108: 112–123
- [33] Yang M, Wang H, Wan Z. PUL-ABE: an efficient and quantum-resistant CP-ABE with policy update in cloud storage. *IEEE Transactions on Services Computing*, 2024, 17(3): 1126–1139
- [34] Li J, Huang Q, Chen X, Chow S S M, Wong D S, Xie D. Multi-authority ciphertext-policy attribute-based encryption with accountability. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. 2011, 386–390
- [35] Wang Y, Chen K, Long Y, Liu Z. Accountable authority key policy attribute-based encryption. *Science China Information Sciences*, 2012, 55(7): 1631–1638
- [36] Zhou J, Cao Z, Dong X, Lin X. TR-MABE: white-box traceable and revocable multi-authority attribute-based encryption and its applications to multi-level privacy-preserving e-healthcare cloud computing systems. In: Proceedings of 2015 IEEE Conference on Computer Communications (INFOCOM). 2015, 2398–2406
- [37] Liu Z, Cao Z, Wong D S. Traceable CP-ABE: how to trace decryption devices found in the wild. *IEEE Transactions on Information Forensics and Security*, 2015, 10(1): 55–68
- [38] Ning J, Dong X, Cao Z, Wei L. Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud. In: Proceedings of the 20th European Symposium on Research in Computer Security on Computer Security. 2015, 270–289
- [39] Liu Z, Wong D S. Practical attribute-based encryption: traitor tracing, revocation and large universe. *The Computer Journal*, 2016, 59(7): 983–1004
- [40] Liu Z, Duan S, Zhou P, Wang B. Traceable-then-revocable ciphertext-policy attribute-based encryption scheme. *Future Generation Computer Systems*, 2019, 93: 903–913
- [41] Zhang K, Li H, Ma J, Liu X. Efficient large-universe multi-authority ciphertext-policy attribute-based encryption with white-box traceability. *Science China Information Sciences*, 2018, 61(3): 032102
- [42] Li Q, Zhu H, Ying Z, Zhang T. Traceable ciphertext-policy attribute-based encryption with verifiable outsourced decryption in eHealth cloud. *Wireless Communications and Mobile Computing*, 2018, 2018(1): 1701675
- [43] He X, Li L, Peng H. An enhanced traceable CP-ABE scheme against various types of privilege leakage in cloud storage. *Journal of Systems Architecture*, 2023, 136: 102833
- [44] Li J, Zhang Y, Ning J, Huang X, Poh G S, Wang D. Attribute based encryption with privacy protection and accountability for CloudIoT. *IEEE Transactions on Cloud Computing*, 2022, 10(2): 762–773
- [45] Rouselakis Y, Waters B. Practical constructions and new proof methods for large universe attribute-based encryption. In: Proceedings of 2013 ACM SIGSAC Conference on Computer & Communications Security. 2013, 463–474
- [46] Waters B. Ciphertext-policy attribute-based encryption: an

expressive, efficient, and provably secure realization. In: Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography on Public Key Cryptography. 2011, 53–70

[47] Liu Z, Cao Z, Wong D S. Efficient generation of linear secret sharing scheme matrices from threshold access trees. Cryptology ePrint Archive, Paper 2010/374, 2010

[48] Rouselakis Y, Waters B. Efficient statically-secure large-universe multi-authority attribute-based encryption. In: Proceedings of the 19th International Conference on Financial Cryptography and Data Security. 2015, 315–332



Yan-Qing YAO is currently an associate professor and doctoral/master's supervisor of Beihang University, China. She received her PhD degree in computer science from Beihang University, China in 2015. From 2012 to 2014, she was a visiting student of New York University in the United States. Her research interests include cryptography, privacy preserving techniques, and artificial intelligence security. Her work won the CACR Excellent Doctoral Dissertation Award in 2018.

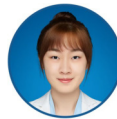


Yun-Jia ZHANG is currently pursuing her PhD degree with the School of Cyber Science and Technology, Beihang University, China. She received her MS degree in Cyberspace Security from the School of Computer

Science and Technology, Beijing Jiaotong University, China in 2021. Her research interests include privacy preserving techniques, asymmetric cryptography, and blockchain security.



Zhi-Yi LIU is an undergraduate student at the School of Cyber Science and Technology, Beihang university, China. Her research interests include computer security, privacy protection, and IoT/AIoT security.



Yu-Xuan WANG is a fourth year undergraduate student at the School of Cyber Science and Technology, Beihang University, China. Her research interests include cryptography and information security.



Xin-Yu TAN is a fourth year undergraduate student at the School of Cyber Science and Technology, Beihang University, China. Her research interests include cryptography and information security.



Zheng-De ZHAI is a senior expert of Li Auto Inc. He received his PhD degree in information security from the Institute of Software of software, Chinese Academy of Sciences, China in 2008. His research interests include data security, access control, and cryptography.