IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 7, 2018, accepted June 4, 2018, date of publication June 7, 2018, date of current version July 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2844829

Secure Multi-Authority Data Access Control Scheme in Cloud Storage System Based on Attribute-Based Signcryption

QIAN XU, CHENGXIANG TAN, ZHIJIE FAN, WENYE ZHU, YA XIAO, AND FUJIA CHENG
Department of Computer Science and Technology, Tongji University, Shanghai 200092, China

Corresponding author: Qian Xu (1062842783@qq.com)

This work was supported by the National Key Research and Development Program of China under Grants 2017YFC0803702 and 2017YFB0802302.

ABSTRACT Nowadays, secure data access control has become one of the major concerns in a cloud storage system. As a logical combination of attribute-based encryption and attribute-based signature, attribute-based signcryption (ABSC) can provide confidentiality and an anonymous authentication for sensitive data and is more efficient than traditional ''encrypt-then-sign'' or ''sign-then-encrypt'' strategies. Thus, ABSC is suitable for fine-grained access control in a semi-trusted cloud environment and is gaining more and more attention in recent years. However, in many previous ABSC schemes, user's sensitive attributes can be disclosed to the authority, and only a single authority that is responsible for attribute management and key generation exists in the system. In this paper, we propose PMDAC-ABSC, a novel privacy-preserving data access control scheme based on Ciphertext-Policy ABSC, to provide a fine-grained control measure and attribute privacy protection simultaneously in a multi-authority cloud storage system. The attributes of both the signcryptor and the designcryptor can be protected to be known by the authorities and cloud server. Furthermore, the decryption overhead for user is significantly reduced by outsourcing the undesirable bilinear pairing operations to the cloud server without degrading the attribute privacy. The proposed scheme is proven to be secure in the standard model and has the ability to provide confidentiality, unforgeability, anonymous authentication, and public verifiability. The security analysis, asymptotic complexity comparison, and implementation results indicate that our construction can balance the security goals with practical efficiency in computation.

INDEX TERMS Attribute based signcryption, multi-authority, public verifiability, anonymous authentication, attribute privacy protection.

I. INTRODUCTION

With the rapid development of cloud computing, more people are coming to prefer moving both the large burden of data storage and computation overhead to the cloud server in a cost-effective manner [1]. In spite of the benefits of cloud computing, secure data access control is still one of the major challenging obstacles since the cloud server is not fully trusted by the data user and the data stored in the cloud may contain sensitive information [2]. Hence, to protect the user's privacy and provide data confidentiality, data owner has to encrypt the data before outsourcing the data to the cloud [3]. Moreover, fine-grained access control measure on the outsourced sensitive data is also desired from the viewpoint of data owner in order to share the data with other users who have certain attributes [4]. For example, to alleviate the storage and computation burden, the personal health record (PHR) service provider may assist the third-party cloud server to store the data. As the PHR data may include sensitive information, it should be guaranteed that only the doctor who is treating the patient has the privilege to access the data. A possible and practical cryptographic tool to provide confidentiality and impose fine-grained access control on sensitive data is attribute-based encryption (ABE) which encrypts the plaintext with a set of attributes (Key-Policy ABE) or an access policy (Ciphertext-Policy ABE). However, in addition to data confidentiality and fine-grained access control, it is also essential that the access control mechanism has the ability to support anonymous authentication. For instance, when a data owner ''*Alice*'' uploads her health record to the cloud, the server can check that whether the

data is outsourced by a user with certain credentials such as "*Female* $\wedge$ *age* $\in$ [20, 30]". When a doctor of "*Hospital A*" accesses the data stored on the PHR cloud storage system, he can also verify that the data belongs to a patient with certain attributes such as "*Hospital A* $\wedge$ *age* $\in$ [20, 30]" while the patient's real identity "*Alice*" is kept secret.

One of the effective and promising strategies to address confidentiality, fine-grained access control, and anonymous authentication simultaneously is Attribute-based Signcryption (ABSC) [5], [6], which takes advantage of the attribute-based encryption and attribute-based signature (ABS) and is more efficient than "encrypt-then-sign" or "sign-then-encrypt" approach. Basically, as a logical combination of ABE and ABS, ABSC scheme adopts ABE to provide data confidentiality and fine-grained access control, and ABS to support anonymous authentication. Traditionally, there are two categories of ABE schemes: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE). In the former one, the access policy (predicate) is embedded in the secret key. While in CP-ABE, access policy is associated with the plaintext message. Similarly, there are also two types of ABS schemes: Signature-Policy ABS (SP-ABS) and Key-Policy ABS (KP-ABS). In SP-ABS, the access policy is assigned to the signature, whereas in KP-ABS, the access policy is embedded in the secret key. The Ciphertext-Policy ABSC (CP-ABSC) [5] is the combination of CP-ABE and SP-ABS. In CP-ABSC scheme, the data owner signcrypts (sign and encrypt) the plaintext under the signing and encryption predicates, and then outsources the resulting ciphertext to the cloud server. Correspondingly, the data user can designcrypt (verify and decrypt) the ciphertext to check that whether the ciphertext is uploaded by the data owner with certain attributes satisfying signing predicate, and recover the plaintext if the user's attributes satisfy the encryption predicate. Similarly, Key-Policy ABSC (KP-ABSC) [6] supports KP-ABE and KP-ABS, wherein the signing and encryption predicates are associated with the user's secret key.

Although ABSC scheme can provide many useful properties such as confidentiality, fine-grained access control, and anonymous authentication, two problems must be considered when implementing ABSC scheme in cloud storage system. The first one is multi-authority. In many previous ABSC schemes, as in [7]–[10], only one fully trusted central authority in the system is responsible for attribute management and secret key generation. However, in many real scenarios, the attributes of the data owner can be issued by different trust domains or authorities, and the predicates defined by the data owner for signing and encryption can also be written over the attributes monitored by multiple authorities. Hence, distributing the attribute management and the corresponding secret key generation in ABSC scheme from a single central authority over many independent authorities is necessary for practical application. The second one is attribute anonymity. For example, if the name, sex and address, which are used as the sensitive attributes of identity card [11], are disclosed or collected, then the user can be identified and impersonated in the network society. The existing ABSC schemes leak no information regarding the attributes of data owner to the data user and cloud server by employing the ABS technique. However, the attributes will be disclosed to the authority during the secret key generation phase. In the multi-authority scenario, it is difficult to ensure all the authorities are fully trusted. If some of these authorities are corrupted, the user's privacy cannot be guaranteed.

Recently, many multi-authority ABE (MA-ABE) schemes and multi-authority ABS (MA-ABS) schemes, as in [12]–[17], have been proposed to provide data access control and authentication in a multi-authority cloud storage setting. However, designing a multi-authority ABSC (MA-ABSC) scheme and proposing a data access control scheme based on MA-ABSC with attribute privacy protection have received very little attention so far. Meng and Meng [18] constructed a data access scheme in a decentralized setting based on identity signature and MA-ABE. However, the scheme does not consider attribute privacy protection and only supports threshold predicate in encryption protocol. Liu *et al.* [19] also constructed a KP-ABSC scheme supporting the key exposure protection. However, the scheme does not support multi-authority and public verification, and both the verification and decryption algorithms are required to be conducted by the data user to check and recover the plaintext, which results in a heavy computation overhead on the user side. Since Ciphertext-Policy can enable the data owner to define the predicate and determine who have the privilege to access the data [9], we focus on CP-ABSC instead of KP-ABSC to realize the data access control application.

A. CONTRIBUTIONS

In this paper, we propose PMDAC-ABSC, a novel privacy-preserving multi-authority data access control scheme based on CP-ABSC. Public verifiability and expressiveness are also considered in our scheme. To provide attribute anonymity and enable the authority to issue the secret key for the user without knowing user's attributes, the commitment scheme and set-member proof technique are employed. Meanwhile, we lighten the decryption cost by outsourcing costly operations to the cloud server without degrading the attribute privacy. Our scheme realizes the security in the standard model.

The main contributions of this work can be summarized as follows:

1) We propose a basic Multi-Authority CP-ABSC (MACP-ABSC) scheme and a privacy-preserving extension to protect the privacy of attributes. Based on the discrete logarithm assumption, the hiding property of the commitment scheme and the zero-knowledge property of the proof of knowledge technique, the privacy of attributes can be guaranteed throughout the data sharing process. To the best of our knowledge, we are the first to design a MACP-ABSC scheme with the ability of attribute privacy protection.

2) The verification mechanism of our construction does not require a plaintext message or the data owner's public key. Further, the scheme supports any monotone Boolean function predicates represented by monotone span programs (MSP) for signing and encryption.

3) We design an outsourcing paradigm to alleviate the computation overhead for users. By using the partial/full decryption approach, we outsource the costly operations to the cloud to save computational resources for users without degrading the attribute privacy. The security analysis, asymptotic complexity comparison and implementation results indicate that our construction can balance the security goals with practical efficiency in computation.

B. RELATED WORKS

1) ACCESS CONTROL SCHEMES BASED ON ABE

Attribute-based encryption, first introduced by Sahai and Waters [20], enables the sensitive data to be shared among different data users according to the predefined access policies (or predicates). Many works on ABE, as in [21]–[27], have been presented to propose secure and usable solutions to the problem of data access control in untrusted cloud servers. In [21], to improve the efficiency, the authors formulated outsourced ABE scheme and proposed several constructions with outsourced decryption and key generation. To verify the correctness of outsourced decryption, Lai *et al.* [22] proposed a verifiability mechanism to enable the user to efficiently check the partial decryption result computed by the cloud server. Zhang *et al.* [23] proposed an anonymous ABE scheme by hiding the access policy in ciphertext and performing test operation before decryption to check that if the secret keys match the hidden encryption predicate. However, the scheme only supports AND gate access policy and realizes security in the random oracle model. Moreover, the privacy of attributes cannot be guaranteed in key generation phase. In [24] and [25], ABE scheme with computation outsourcing is constructed for data access control in fog-computing system. The heavy computation overhead of encryption or decryption is outsourced to the fog nodes, which results in a significant efficiency improvement of the end user's resource-constraint devices. Rohit *et al.* [26] and Wang *et al.* [27] proposed hierarchical ABE to provide fine-grained access control, full privilege delegation and scalable attribute revocation.

In many ABE schemes, the attribute universe is assumed to be managed by a single authority. This premise may not be appropriate for the practical requirements, where users' attributes may be issued by different authorities. For example, the patient "*Alice*" may encrypts her sensitive health records under the encryption predicate as "$(Doctor \vee Researcher) \wedge Female \wedge age \in [40, 50]$", and uploads the ciphertext to the cloud. Then only the female doctors or researchers aged from 40 to 50 can access the Alice's health data. Since the attribute "*Doctor*" can only be certified by a hospital and the attribute "*Researcher*" can only be issued by a research organization, it is required that the access right of the data user should be authorized by different authorities. To solve this problem, Chase [28] proposed multi-authority ABE (MA-ABE) scheme. In MA-ABE, there are multiple independent attribute authorities, each of which monitors a set of attributes. The data user has the ability to access the encrypted data if and only if he holds a certain number of attributes from each attribute authority. Since a central authority is still employed in [28], an improved MA-ABE scheme which uses Pseudo Random Function and 2-party key exchange mechanism was proposed in [29] to remove the central authority. However, since each pair of authorities must involve in a key exchange protocol, a new joined attribute authority has to cooperate with all the other authorities, which incurs a heavy communication and computation overhead. Recently, many MA-ABE schemes have been proposed, as in [11]–[17] and [30]–[36]. Han *et al.* [11], [12] constructed a multi-authority CP-ABE (MACP-ABE) scheme to provide confidentiality and fine-grained control in a decentralized setting. The scheme does not require a central authority and can protect the user's privacy. Nevertheless, the scheme is not collusion resistant and cannot provide authentication. In [13], a secure data access control scheme with attribute revocation and decryption outsourcing is proposed. However, the security of [13] is realized in the random oracle model. Lewko and Waters [14] constructed a decentralized fully secure MA-ABE scheme. To resist collusion attack, the secret keys of the user issued by different authorities are all tied to his global identity. However, the security is proven in the random oracle model. The scheme is also inefficient due to the longer size of composite-order group elements. Ruj *et al.* [15] proposed a decentralized ABE scheme where multiple attribute authorities issue the secret keys to the user. They also adopted a ABS scheme to provide authentication. Sourya and Ruj [16] constructed an efficient data access control scheme for mobile cloud storage system by means of online/offline encryption and outsourced decryption. The scheme also supports user-level revocation. Yang *et al.* [17] proposed a data access control scheme for multi-authority cloud system. The most costly operations in a decryption algorithm are outsourced to the cloud. Thus the data user only needs to perform one exponentiation to recover the plaintext. However, the attribute privacy cannot be guaranteed since the cloud server has to know user's attributes to execute partial decryption for the user. In [30], a privilege control mechanism is constructed to provide confidentiality of user's identity and sensitive data. The scheme can resist compromise attack on the attribute authorities, whereas it only supports AND gate predicate and is provably secure in the random oracle model. Li *et al.* [31] proposed a ciphertext-policy MA-ABE scheme supporting decryption outsourcing and efficient user revocation. The scheme realizes adaptive security in the standard model and supports any monotone encryption predicate. However, the group order in [31] is a product of three primes, which incurs a significant computation overhead on the user side. Nomura *et al.* [32] proposed a CP-ABE scheme supporting immediate attribute revocation without secret key update, and realizes security

in the standard model. The length of the secret key and ciphertext is fixed. Nevertheless, the encryption predicate is expressed with only AND gate. Chow *et al.* [33] also considered the problem of attribute revocation in multi-authority ABE schemes. They proposed a framework for constructing MACP-ABE with attribute revocation and outsourced decryption from any pairing-based single-authority ABE. The framework needs the secret keys and ciphertext of underlying ABE can be divided into attribute-independent part and attribute-dependent part. Then by employing a two-round key generation, the attribute-dependent part of secret key can be generated by other authorities in the multi-authority setting. Zhou *et al.* [34] also proposed an efficient multi-authority ABE scheme supporting attribute revocation. To improve the revocation efficiency, only a minimal set of attributes should be updated in order to revoke a user. Meanwhile, the costly computations of encryption and revocation are outsourced to the cloud server, which makes the scheme more scalable. To overcome single-point bottleneck and low efficiency of multi-authority ABE scheme, Xue *et al.* [35] proposed a heterogeneous data access control framework with an auditing mechanism. User legitimacy verification is separated from secret key generation phase, and multiple attribute authorities are involved in the system to share the heavy load of user legitimacy verification. A central authority is only responsible for the secret key computation without verification. Since multiple attribute authorities can work in parallel, the efficiency of the scheme can be greatly improved. Single-point bottleneck problem is also considered in [36]. In [36], the authors constructed TMACS for public cloud storage system. Taking the advantage of (t, n) threshold secret sharing, the master secret key can be shared by multiple authorities, and a legal user can obtain the secret key from at least t authorities. Moreover, a hybrid scheme is proposed in [36] to achieve security and system-level robustness. In hybrid scheme, the attribute universe is divided into multiple disjoint sets, each of which is managed by n authorities. In secret key generation phase, t authorities monitoring the same attribute set containing the corresponding attribute jointly generate the secret key for the user.

In attribute-based cryptography with outsourced decryption, attribute privacy is involved in three phases: secret key generation, encryption and decryption. In our construction, we realize the attribute privacy protection in secret key generation and decryption. Since the signing and encryption predicates are contained in ciphertext, some attribute information can be obtained by data user or cloud server. However, due to the high policy expressiveness (any monotone Boolean function) and the distribution of attribute control over many independent authorities, the selection of signing and encryption predicates can preserve the attribute privacy. For example, assume there are three attribute authorities $\{AA_1, AA_2, AA_3\}$ supervising different attribute sets. $\widetilde{AA_1} = \{Faculty, Student, Male, Female\}$, $\widetilde{AA_2} = \{Computer\ Science, Engineering, Chemistry\}$ and $\widetilde{AA_3} = \{University\ A, University\ B\}$. When a student major in computer science of university A wants to upload a message to the cloud, he/she can define the signing predicates for each authority as $\mathcal{R}_{s,1} = (Student \wedge (Male \vee Female))$, $\mathcal{R}_{s,2} = (Computer\ Science \vee Engineering)$, $\mathcal{R}_{s,3} = (University\ A \vee University\ B)$. Although the cloud server can know that the message has been signed and uploaded by a data owner with attributes satisfying $\mathcal{R}_{s,1}$, $\mathcal{R}_{s,2}$ and $\mathcal{R}_{s,3}$, it cannot obtain the exact attributes of the data owner. Another solution to address the attribute privacy in ciphertext is anonymous ABE [23]. However, how to hide the attribute information in ciphertext while supporting high policy expressiveness, outsourced decryption and multi-authority is a challenge and thus will be our future work.

2) ATTRIBUTE-BASED SIGNATURE AND MULTI-AUTHORITY ATTRIBUTE-BASED SIGNATURE

ABS was first introduced by Maji *et al.* [37]. Due to anonymity and authentication properties, many ABS schemes have been proposed, such as [3] and [38]–[40]. Similar to ABE, to overcome the drawback that only a single authority exists in the system, the concept of MA-ABS was introduced in [41] and [42], in which there are multiple authorities and each authority is responsible for issuing a secret key associated with a category or sub-universe of attributes.

3) ACCESS CONTROL SCHEMES BASED ON ABSC

ABSC scheme, first introduced by Gagné *et al.* [5], is a logical mixture of ABE and ABS and can support many practical properties, including fine-grained access control, confidentiality and authentication. Recently, many data access control schemes based on ABSC have been proposed, as in [6]–[10]. Sreenivasa and Dutta [6] proposed a Key-Policy attribute-based signcryption scheme that supports any monotone Boolean function and constant size ciphertext. However, the message confidentiality and unforgeability of the scheme against selectively adversary are proven under decisional Bilinear Diffie-Hellman Exponent assumption in the random oracle model. Chen *et al.* [7] focused on the joint security of signature and encryption schemes and presented a CP-ABSC scheme in the joint security setting. However, it cannot support public verifiability since it requires the plaintext message to verify the signature. Liu *et al.* [8] proposed a secure sharing scheme for a PHR system based on CP-ABE [43] and ABS [37]. However, it is only provably secure in a random oracle model. In [9], a CP-ABSC based access control scheme with public verifiability is proposed. Although the security is realized in the standard model, the scheme does not consider the attribute privacy protection. Yu and Cao [10] proposed the hybrid access policy ABSC scheme that supports key policy signature and ciphertext policy encryption. The size of the ciphertext is constant and the scheme is proven to be secure in the standard model. Nevertheless, it only supports the threshold access structure in the encryption phase. Moreover, the above ABSC schemes only have a single authority and cannot serve for multi-authority system.

To realize the practical flexibility of MACP-ABSC scheme, to address the anonymous authentication, public verifiability and expressiveness, and also to support attribute privacy protection, we propose privacy-preserving multi-authority data access control scheme PMDAC-ABSC scheme based on the MACP-ABSC.

C. PAPER ORGANIZATION

The remainder of this paper is organized as follows. In section 2, we review the necessary notations and cryptographic background that are used throughout the paper. In section 3, we give the definition and system model of our scheme, and also the security notions and requirements. The details of the scheme and the security proof are elaborated in sections 4 and 5, respectively. Section 6 is dedicated to analyzing the scheme. Finally, we conclude this paper in section 7.

II. PRELIMINARIES

By $a \overset{R}{\leftarrow} A$, we denote that a is selected randomly from A. $|A|$ denotes the cardinality of a finite set A. $\mathbb{Z}_p$ denotes a finite field with prime order p, and $\mathbb{Z}_p^*$ stands for $\mathbb{Z}_p \backslash \{0\}$. $A(x) \rightarrow y$ denotes that y is computed by running algorithm A with input x. $[n]$ represents the set $\{1, 2, \ldots, n\}$. $\vec{a}^{(i)}$ denotes the ith element of the vector $\vec{a}$. A function $\epsilon : \mathbb{Z} \rightarrow R$ is negligible if, for any $z \in \mathbb{Z}$, there exists a k such that $\epsilon(x) < 1/x^z$ when $x > k$. We use s and e as subscripts for signing and encryption, respectively. $Pr[E]$ denotes the probability of an event E occurring.

The notion $Commit(params, m) \rightarrow \{com, decom\}$ describes the commit algorithm. $Decommit(params, m, com, decom) \rightarrow \{0, 1\}$ denotes the decommit algorithm, where the output 1 represents that the decommitment $decom$ can decommit com to m. In this paper, we employ the Pedersen commitment scheme [44], which exhibits hiding and binding properties. The hiding property means that the plaintext is unknown until the committer releases it. The binding property means that only $decom$ can decommit com.

We use a zero-knowledge proof of knowledge $\Sigma = PoK\left\{(\alpha, \beta, \gamma) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma\right\}$ to denote the knowledge of integers α, β and γ such that $y = g^\alpha h^\beta$ and $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma$ hold on the group $\mathbb{G} = \langle g \rangle = \langle h \rangle$ and $\tilde{\mathbb{G}} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$, respectively. The values in parentheses denote the elements to be proven, and all other letters denote the elements that are known to the verifier. Notably, there exists an efficient extractor that can be used to rewind the knowledge from the successful prover. In this paper, we employ the Schnorr proof of knowledge [45] to attest the knowledge of a discrete logarithm.

Definition 1 (Bilinear Maps [13]): Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups with the prime order p and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Then the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ can be defined as follows:

1. Bilinear. $\forall a, b \in \mathbb{Z}_p$, $e\left(g^a, g^b\right) = e(g, g)^{ab}$.
2. Non-degenerate. $e(g, g) \neq 1$.
3. Computable. There is an efficient algorithm to compute the map $e(g, h)$, where $g, h \in \mathbb{G}$.

Throughout the paper, $\mathcal{GG}(1^k) \rightarrow (e, p, \mathbb{G}, \mathbb{G}_T)$ denotes the bilinear generation algorithm that intakes a security parameter 1^k and outputs a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups with order p.

Definition 2 (Decisional Bilinear Diffie-Hellman Assumption [13]): Let g be a generator of $\mathbb{G}$ with prime order p and $a, b, c \in \mathbb{Z}_p^*$ be randomly chosen. Given a vector $\vec{\mathcal{Y}} = \left(g, g^a, g^b, g^c\right)$, the decisional BDH assumption holds if no PPT adversary $\mathcal{A}$ can distinguish $\left(\vec{\mathcal{Y}}, \Omega = e(g, g)^{abc}\right)$ from $\left(\vec{\mathcal{Y}}, \Omega \overset{R}{\leftarrow} \mathbb{G}_T\right)$ with the advantage

$$Adv_{\mathcal{A}} = \left| Pr\left[\mathcal{A}\left(\vec{\mathcal{Y}}, \Omega = e(g, g)^{abc}\right) = 1\right] - Pr\left[\mathcal{A}\left(\vec{\mathcal{Y}}, \Omega \overset{R}{\leftarrow} \mathbb{G}_T\right) = 1\right] \right| \geq \epsilon(k).$$

Definition 3 (Decisional q-Parallel Bilinear Diffie-Hellman Exponent (q-PBDHE) Assumption [43]): Suppose that $a, s, b_1, b_2, \ldots, b_q \overset{R}{\leftarrow} \mathbb{Z}_p$, $\mathcal{GG}(1^k) \rightarrow (e, p, \mathbb{G}, \mathbb{G}_T)$ and g is a generator of $\mathbb{G}$. Given

$$\vec{\mathcal{Y}} = \Bigg(g, g^s, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, \forall 1 \leq j \leq q, g^{sb_j}, g^{\frac{a}{b_j}}, \ldots, g^{\frac{a^q}{b_j}}, g^{\frac{a^{q+2}}{b_j}} \ldots, g^{\frac{a^{2q}}{b_j}}, \forall 1 \leq j, k \leq q, k \neq j, g^{\frac{asb_k}{b_j}}, \ldots, g^{\frac{a^q sb_k}{b_j}}\Bigg),$$

the decisional q-PBDHE assumption holds if no PPT adversary $\mathcal{A}$ can distinguish $\left(\vec{\mathcal{Y}}, \Omega = e(g, g)^{a^{q+1}s}\right)$ from $\left(\vec{\mathcal{Y}}, \Omega \overset{R}{\leftarrow} \mathbb{G}_T\right)$ with the advantage

$$Adv_{\mathcal{A}} = \left| Pr\left[A\left(\vec{\mathcal{Y}}, \Omega = e(g, g)^{a^{q+1}s}\right) = 1\right] - Pr\left[\mathcal{A}\left(\vec{\mathcal{Y}}, \Omega \overset{R}{\leftarrow} \mathbb{G}_T\right) = 1\right] \right| \geq \epsilon(k).$$

Definition 4 (Monotone Span Program (MSP) [46]): Assume $\{v_1, v_2, \ldots, v_m\}$ is a set of variables. An MSP is a labeled matrix $\Omega := (M_{\ell \times n}, \rho)$, where M is an $\ell \times n$ matrix over $\mathbb{Z}_p$ and ρ is the labeling function $\rho : [\ell] \rightarrow \{v_1, v_2, \ldots, v_m\}$.

Let $\vec{x} = (x_1, x_2, \ldots, x_m) \in \{0, 1\}^m$ and $X_\mu = \left\{i \in [\ell] : \left[\rho(i) = v_j\right] \wedge \left[x_j = \mu\right]\right\}$ where $\mu \in \{0, 1\}$. $X_1 \cup X_0 = [\ell]$. Let M^i be the ith row of M. We denote $\Omega(\vec{x}) = 1$ if Ω accepts the input $\vec{x}$. Likewise, $\Omega(\vec{x}) = 0$ means Ω rejects $\vec{x}$. Then $\Omega(\vec{x}) = 1 \Leftrightarrow \left[\exists (a_1, a_2, \ldots, a_\ell) \in \mathbb{Z}_p^\ell \text{ such that } \sum_{i \in [\ell]} a_i M^i = \vec{1}\right]$ where $a_i = 0$ for all $i \in X_0$.

An MSP Ω computes a monotone Boolean function $\mathcal{R}$: $\{0, 1\}^m \rightarrow \{0, 1\}$ if $\Omega(\vec{x}) = 1$ for all $\vec{x} \in \{\vec{x} : \mathcal{R}(\vec{x}) = 1\}$.

Lemma 1: If $\Omega(\vec{x}) = 0$, then there exists a vector $\overrightarrow{\omega} = (\omega_1, \omega_2, \ldots, \omega_n) \in \mathbb{Z}_p^n$ with $\omega_1 = -1$ such that $\overrightarrow{\omega} M^i = 0$ for all $i \in X_1$.

Definition 5 (Predicates [9]): Assume U is the universe of attributes. A predicate over U is a monotone Boolean function whose inputs are associated with the attributes of U. Let $W \subset U$ is a subset of attributes. A predicate $\mathcal{R}$ accepts $W \subset U$ if $\mathcal{R}(W) = 1$. If W does not satisfy $\mathcal{R}$ then $\mathcal{R}(W) = 0$. A predicate $\mathcal{R}$ is said to be monotone, if $\mathcal{R}(W) = 1 \Rightarrow \mathcal{R}(C) = 1$ for every attribute set $C \supset W$.

Suppose $\mathcal{R}$ is a predicate and $L_{\mathcal{R}}$ is the set of attributes utilized in $\mathcal{R}$. Then the corresponding MSP for $\mathcal{R}$ is a labeled matrix $\Omega := (M_{\ell \times n}, \rho)$, where $\rho: [\ell] \to L_{\mathcal{R}}$.

Define $X_1 = \{i \in [\ell] : [\rho(i) = a] \wedge [a \in W]\}$ and $X_0 = \{i \in [\ell] : [\rho(i) = a] \wedge [a \notin W]\}$. $X_1 \cup X_0 = [\ell]$. Then $R(W) = 1 \Leftrightarrow \Omega(W) = 1 \Leftrightarrow \left[\exists (a_1, a_2, \ldots, a_\ell) \in \mathbb{Z}_p^\ell \text{ such that } \sum_{i \in [\ell]} a_i M^i = \vec{1} \text{ and } a_i = 0 \forall i, \rho(i) \notin W\right]$.

Lemma 2: If $\mathcal{R}(W) = 0$, then there exists a vector $\vec{\omega} = (\omega_1, \omega_2, \ldots, \omega_n) \in \mathbb{Z}_p^n$ with $\omega_1 = -1$ such that $\vec{\omega} M^i = 0$ for all i where $\rho(i) \in W$.

Definition 6: Let $M_{\ell \times n}$ be a matrix of size $\ell \times n$ over a field $\mathbb{F}$. *rank* (M) is rank of $M_{\ell \times n}$. If *rank* $(M) < \ell$, then $\mathbb{V} = \left\{(b_1, b_2, \ldots, b_\ell) \in \mathbb{F}^\ell : \sum_{i \in [\ell]} b_i M^i = \vec{0}\right\}$ contains polynomial number of vectors $(b_1, b_2, \ldots, b_\ell)$ and the predicate for which MSP is $\Omega := (M_{\ell \times n}, \rho)$ consists of both AND and OR gates. Otherwise, $\mathbb{V} = \{\vec{0}\}$ and the predicate is an AND gate. In our construction, we consider the signing and encryption predicates consisting of both AND and OR gates [9].

Definition 7 (Set-Membership Proof [45]): Let $\mathcal{GG}(1^k) \to (e, p, \mathbb{G}, \mathbb{G}_T)$, g, h be the generators of $\mathbb{G}$. The proof $PK(\sigma : \sigma \in \Phi)$ with commitment $C = g^\sigma h^r$ is performed as follows.

Init. Suppose that $\Phi \subseteq \mathbb{Z}_p$ is a finite set. $x \overset{R}{\leftarrow} \mathbb{Z}_p$ and $Y = g^x$. For $i \in \Phi$, $A_i = g^{1/(x+i)}$. The verifier sends Y, A_i, g, h, e to the prover.

Blind. To prove $\sigma \in \Phi$, the prover selects $v \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes $V = A_\sigma^v = g^{v/(x+\sigma)}$.

*Step*1. The prover chooses $s, t, r, m \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes $D = g^s h^m$ and $a = e(V, g)^{-s} e(g, g)^t$. Send D, V, a to the verifier.

*Step*2. The verifier sends $c \overset{R}{\leftarrow} \mathbb{Z}_p$ to the prover.

*Step*3. The prover sends $z_\sigma = s - c\sigma, z_r = m - cr$ and $z_v = t - cv$ to the verifier.

*Step*4. The verifier verifies $D = C^c g^{z_\sigma} h^{z_r}$ and $a = e(Y, V)^c e(V, g)^{-z_\sigma} e(g, g)^{z_v}$.

III. SCHEME AND SECURITY DEFINITIONS

A. DEFINITION OF PMDAC-ABSC SCHEME

Our PMDAC-ABSC scheme consists of a MACP-ABSC scheme. The MACP-ABSC scheme consists of the following algorithms.

GlobalSetup $\left(1^k\right)$. Taking as input a security parameter 1^k, the algorithm outputs the public parameters *PP*. It also consists of registration phases for users and authorities.

AuthoritySetup (PP). It takes as input *PP* and outputs the public key and secret key pairs $\{PK, SK\}$ for the authority.

SecretKeyGen $\left(PP, PK_j, SK_j, PK_{uid}, \widetilde{U}\right)$. Taking as input *PP*, $\{PK_j, SK_j\}$ of authority AA_j, user's public key PK_{uid} and attribute set $\widetilde{U}$, the algorithm outputs the secret key $SK_{uid,j}$ for the user.

Signcryption $\left(\mathcal{M}, PP, \left\{SK_{do,j}, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j \in I}\right)$. Taking as input the message $\mathcal{M}$, *PP*, signing and encryption predicates $\left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j \in I}$ and the set of signcryptor's secret keys $\left\{SK_{do,j}\right\}_{j \in I}$, I is the set of involved authorities in the ciphertext. The algorithm outputs the ciphertext CT.

DeSigncryption $\left(PP, CT, PK_{du}, \left\{SK_{du,j}\right\}_{j \in I}\right)$. This algorithm contains two sub-algorithms: *Verify* (PP, CT) and *Decryption* $\left(PP, CT, PK_{du}, \left\{SK_{du,j}\right\}_{j \in I}\right)$.

Verify (PP, CT). Taking as input PP and CT, the algorithm verifies that whether CT is a valid ciphertext. Since the algorithm only intakes public parameters and ciphertext, any trusted party can perform the verification mechanism.

Decryption $\left(PP, CT, PK_{du}, \left\{SK_{du,j}\right\}_{j \in I}\right)$. Taking as input PP, CT and , $PK_{du}, \left\{SK_{du,j}\right\}_{j \in I}$ of data user, output the plaintext $\mathcal{M}$ or $\perp$.

Definition 8: Assume $\widetilde{AA_j}$ is attribute set of AA_j. MACP-ABSC scheme is correct if for each $j \in I$, $\mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_{do,s}}\right) = 1, \mathcal{R}_{e,j}\left(\widetilde{AA_j} \cap \widetilde{U_{du,d}}\right) = 1$, then $Pr\left[DeSigncryption\left(PP, CT, PK_{do}, \left\{SK_{do,j}\right\}_{j \in I}\right) \to \mathcal{M}\right] = 1$, where $SecretKeyGen\left(PP, PK_j, SK_j, PK_{do}, \widetilde{U_{do}}\right) \dashrightarrow SK_{do,j}$, $SecretKeyGen\left(PP, PK_j, SK_j, PK_{du}, \widetilde{U_{du}}\right) \dashrightarrow SK_{du,j}$, and $Signcryption\left(\mathcal{M}, PP, \left\{SK_{do,j}, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j \in I}\right) \to CT$.

The PMDAC-ABSC scheme has the same algorithms *GlobalSetup*, *AuthoritySetup*, *Signcryption* and *DeSign* − *cryption* as the MACP-ABSC scheme. However, we construct the *Enhanced_SecretKeyGen* instead of *SecretKeyGen* to compute the secret key for users.

Definition 9: Enhanced_SecretKeyGen Algorithm

Enhanced_SecretKeyGen contains two steps:

1) The user U commits the attribute $a \in \widetilde{U}$ to authority AA_j by *Commit* $(params, a) = \{com, decom\}$ and uses the set member proof technique $PK\left(a : a \in \widetilde{AA_j}\right)$ with commitment *com* in Definition 7 to prove that the attribute a is monitored by AA_j without revealing anything regarding the attributes to AA_j.

2) U proves in zero knowledge Σ_U to AA_j that U knows the secrets with which the secret key components can be computed. If Σ_U is correct, then AA_j generates the secret key according to the elements from U and proves in zero knowledge Σ_{AA_j} to U that the secret key is well formed. Finally, *Enhanced_SecretKeyGen* outputs the real secret key with the user's secrets and elements from AA_j.

B. SCHEME MODEL

As shown in Fig. 1, our PMDAC-ABSC scheme has four types of entities: global certificate authority (CA), users (including signcryptors and designcryptors), independent attribute authorities (AAs) and cloud server.

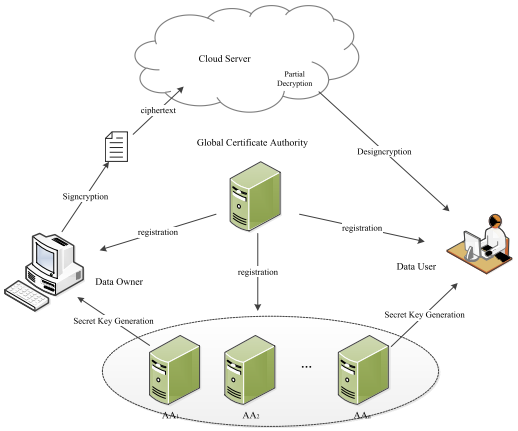


FIGURE 1. System architecture.

Global Certificate Authority (CA): CA generates the public parameters for the system by performing *GlobalSetup* algorithm. CA is also responsible for the users' and authorities' registrations. During the registration, CA can verify the identity of the legitimate user and authority.

Attribute Authority (AA): AA can initialize itself to setup its public and secret keys using *AuthoritySetup* algorithm. To compute the secret keys for users, the authority verifies the user's identity and generates the secret keys by running *SecretKeyGen* algorithm according to the user's attributes. *SecretKeyGen* algorithm can be further extended to *Enhanced_SecretKeyGen* algorithm to protect the attribute privacy.

From a high-level point of view, *Enhanced_SecretKeyGen* forms a two-party computation (2PC) protocol [29] for the functionality F between the user U and authority AA_j. F takes as public input the authority's public key PK_j, the user's public key PK_{uid} and the attribute set $\widetilde{AA_j}$. It also receives as secret input the user's attribute set $\widetilde{U}$ and the authority's secret key SK_j. It outputs the result of *SecretKeyGen* to the user.

1) CLOUD SERVER

Cloud server is responsible for storing sensitive data uploaded by data owners. It can also provide corresponding access to data users. Since our scheme supports public verification, the cloud server can verify that the ciphertext is valid and is signcrypted by the data owner whose attributes satisfy the signing predicates contained in the ciphertext. If the ciphertext is not valid, the cloud server can reject it. The cloud server can also perform *PartialDecryption* algorithm to help data user decrypt the ciphertext with user's transformed secret key.

2) DATA OWNER (SIGNCRYPTOR)

When the data owner signcrypts a message, he/she can select signing and encryption predicates for each authority and outsource the resulting ciphertext to the cloud server by running *Signcryption* algorithm. To successfully outsource the ciphertext, the data owner's signing attributes should satisfy the signing predicates. We assume that the ciphertext implicitly contains the signing and encryption predicates. Only legally registered users can endorse the data, and only users satisfying the predicates can decrypt the data.

3) DATA USER (DESIGNCRYPTOR)

To access the sensitive data, the decryption attributes of the data user should satisfy the encryption predicates specified by the data owner during the signcryption phase. Since the data users are always resource limited, we employ two techniques to improve the efficiency of the data user. The first one is public verification. The data user can use any trusted third party to check the validity and integrity of the ciphertext, and do not need to perform verification on his own device. The second one is outsourced decryption without degrading the attribute privacy. Since the most computation-consuming job of decryption is offloaded on the cloud server, the efficiency of *Decryption* algorithm can be significantly improved on the user side.

C. THREAT ASSUMPTION

Assume the global central authority CA is fully trusted. The attribute authorities can honestly issue the secret key for the user and will not collude with the user to access the sensitive data. However, the attribute authorities can be corrupted and disclose the information sent from the data user to the adversary. The cloud server is honest but curious. It will execute the protocol in general but will attempt to collect the user's private information and get illegal access privileges. The users are malicious, and can collude with other users and even the cloud server to sign or decrypt the unauthorized data.

D. SECURITY NOTIONS

1) SECURITY OF COLLUSION ATTACK

The PMDAC-ABSC scheme is secure against collusion attack of users and cloud server if no polynomial time adversaries can sign the plaintext or decrypt the ciphertext by cooperating with each other when they are individually unauthorized to sign or decrypt the data.

2) SECURITY MODELS OF PMDAC-ABSC SCHEME WITH *SecretKeyGen* ALGORITHM

The confidentiality, unforgeability and signcryptor privacy of our PMDAC-ABSC scheme are presented in Definition 10, 11 and 12 as follows by defining the security games between a challenger and an adversary $\mathcal{A}$.

Definition 10 (Message Confidentiality): Indistinguishability of ciphertext under selective encryption predicate and adaptive chosen ciphertext attack (IND-sEP-CCA2).

We say that the scheme is $(T, q_{sk}, q_{SC}, q_{DS}, \epsilon)$-IND-sEP-CCA2 secure if for any PPT adversary A running in time at most T that makes at most q_{sk} *SecretKey* queries, q_{SC} *Signcryption* queries and q_{DS} *DeSigncryption* queries,

the advantage $Adv_{\mathcal{A}}^{IND-sEP-CCA2}$ of $\mathcal{A}$ is at most ϵ in the following game.

Init. $\mathcal{A}$ submits the set consists of corrupted authorities with index I' and challenge encryption predicates $\mathcal{R}_{e,I^*} = \left\{\left(M_{e,j}^*, \rho_{e,j}^*\right)\right\}_{j\in I^*}$ where I^* is the set consisting of indexes of the authorities. $I^* \neq I'$. Assume $\mathcal{R}_e^* = \left(M_e^*, \rho_e^*\right)$ is specified by the authority $AA^* \in I^*, AA^* \notin I'$. $\mathcal{R}_e^* \in \mathcal{R}_{e,I^*}$. $\mathcal{R}_e^*$ cannot be satisfied by the attributes selected by $\mathcal{A}$ to query decryption secret keys.

Setup. The challenger $\mathcal{C}$ runs *GlobalSetup* to generate the public parameters and the public key of the user and adversary.

Authority Setup. $\mathcal{C}$ runs *AuthoritySetup* to generate the public and secret key pairs for the authorities.

Phase 1.

SecretKey query $\mathcal{O}^{sk}\left(\widetilde{U}, AA_j\right)$. $\mathcal{A}$ can adaptively query the secret key for a user U with identity *uid* and a set of attributes $\widetilde{U} = \widetilde{U_d} \cup \widetilde{U_s}$ to the authority AA_j. $\widetilde{U_d}$ does not satisfy $\mathcal{R}_e^*$. $\mathcal{C}$ runs *SecretKeyGen* and returns the secret key to the adversary.

Signcryption query $\mathcal{O}^{SC}\left(\mathcal{M}, \left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j\in I}\right)$. $\mathcal{A}$ submits a message $\mathcal{M} \in \mathbb{G}_T$, signing and encryption predict set $\left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j\in I'}$. $\mathcal{C}$ selects a signing attribute set $\widetilde{U_s}$ such that $\mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_s}\right) = 1$ for all $j \in I$ and returns the ciphertext to the adversary.

DeSigncryption query $\mathcal{O}^{DS}\left(CT, \widetilde{U_d}\right)$. $\mathcal{A}$ submits a ciphertext CT and a decryption attribute set $\widetilde{U_d}$. $\mathcal{C}$ returns the plaintext to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two messages $\mathcal{M}_0, \mathcal{M}_1$ with the same length and signing predicates $\mathcal{R}_{s,I^*} = \left\{\left(M_{s,j}^*, \rho_{s,j}^*\right)\right\}_{j\in I^*}$ to $\mathcal{C}$. $\mathcal{C}$ selects a signing attribute set $\widetilde{U_s}$ satisfying $\mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_s}\right) = 1$ for all $j \in I^*$. $\mathcal{C}$ randomly chooses a bit $\ell \in \{0, 1\}$ and runs the *Signcryption* algorithm and returns the ciphertext CT^* to $\mathcal{A}$ as the challenge ciphertext.

Phase 2. Phase 1 is repeated. In this phase, $\mathcal{A}$ cannot issue $\mathcal{O}^{DS}\left(CT^*, \widetilde{U_d}\right)$ where $\mathcal{R}_{e,j}\left(\widetilde{AA_j} \cap \widetilde{U_d}\right) = 1$ for all $j \in I^*$.

Guess. $\mathcal{A}$ outputs a guess bit ℓ' on ℓ. $\mathcal{A}$ wins the game if $\ell' = \ell$.

The advantage of $\mathcal{A}$ is defined by $Adv_{\mathcal{A}}^{IND-sEP-CCA2} = \left|Pr\left[\ell' = \ell\right] - 1/2\right|$.

Definition 11 (Ciphertext Unforgeability): Existential unforgeability under selective signing predicate and adaptive chosen message attack (EUF-sSP-CMA).

We say that the proposed scheme is $(T, q_{sk}, q_{SC}, q_{DS}, \epsilon)$-EUF-sSP-CMA secure if for any PPT adversary $\mathcal{A}$ running in time at most T that makes at most q_{sk} *SecretKey* queries, q_{SC} *Signcryption* queries and q_{DS} *DeSigncryption* queries, the advantage $Adv_{\mathcal{A}}^{EUF-sSP-CMA}$ of $\mathcal{A}$ is at most ϵ in the following game.

Init. The adversary $\mathcal{A}$ submits a set of corrupted authorities with index I' and challenge signing predicates $\mathcal{R}_{s,I^*} = \left\{\left(M_{s,j}^*, \rho_{s,j}^*\right)\right\}_{j\in I^*}$. Assume $\mathcal{R}_s^* = \left(M_s^*, \rho_s^*\right)$ is specified by the authority AA^*, $\mathcal{R}_s^* \in \mathcal{R}_{s,I^*}$. $\mathcal{R}_s^*$ cannot be satisfied by the attributes selected by $\mathcal{A}$ to query signing secret keys. $AA^* \in I^*, AA^* \notin I'$.

Setup, *Authority Setup*, *Signcryption query* and *DeSigncryption query* are the same as Definition 10.

SecretKey query $\mathcal{O}^{sk}\left(\widetilde{U}, AA_j\right)$. Basically the same as Definition 10 except that $\widetilde{U_s}$ does not satisfy $\mathcal{R}_s^*$.

Forgery. $\mathcal{A}$ outputs the forgery ciphertext CT^* for $\mathcal{R}_{s,I^*} = \left\{\left(M_{s,j}^*, \rho_{s,j}^*\right)\right\}_{j\in I^*}$ and $\left\{\mathcal{R}_{e,j}\right\}_{j\in I^*}$.

$\mathcal{A}$ wins the game if CT^* is a valid ciphertext and $\mathcal{A}$ has never issued $\mathcal{O}^{SC}\left(\mathcal{M}, \left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j\in I^*}\right)$. The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{EUF-sSP-CMA} = Pr\left[\mathcal{A}\ wins\right]$.

Definition 12: Signcryptor Privacy.

The MACP-ABSC scheme satisfies signcryptor privacy if for any adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l} PP \leftarrow GlobalSetup\left(1^k\right) \\ \left\{PK_j, SK_j\right\}_I \leftarrow AuthoritySetup\left(PP\right) \\ \left(\begin{array}{c}\widetilde{U_s^0}, \widetilde{U_s^1}, M, \\ \left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j\in I}\end{array}\right) \leftarrow \mathcal{A}\left(PP, \left\{PK_j, SK_j\right\}_I\right) \\ \forall j \in I, \mathcal{R}_{s,j}(\widetilde{AA_j} \cap \widetilde{U_s^0}) = 1 \\ \ell' = \ell\colon\ \mathcal{R}_{s,j}(\widetilde{AA_j} \cap \widetilde{U_s^1}) = 1 \\ \ell \overset{R}{\leftarrow} \{0, 1\} \\ CT_\ell \leftarrow Signcryption\left(\begin{array}{c}\mathcal{M}, PP \\ \{SK_{singer,j}^{\ell}\}_{j\in I'} \\ \left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j\in I}\end{array}\right) \\ \ell' \leftarrow \mathcal{A}\left(PP, CT_\ell, \left\{PK_j, SK_j\right\}\right) \end{array}\right] < \frac{1}{2}.$$

3) SECURITY MODEL OF *Enhanced_SecretKeyGen*

We consider the security of *Enhanced_SecretKeyGen* algorithm against two types of adversaries: the malicious user and malicious authority. For the malicious user U, it is required that U cannot obtain more information in *Enhanced_SecretKeyGen* than in *SecretKeyGen* algorithm. For the malicious authority AA_j, it is required that *Enhanced_SecretKeyGen* reveals nothing regarding the user's attributes.

Definition 13: Enhanced_SecretKeyGen is secure against the malicious user U if there exists a simulator *Sim* such that for the security parameter 1^k, there is no efficient distinguisher $\mathcal{D}$ can distinguish the real key generation protocol from the ideal key generation protocol with more than a non-negligible advantage.

Real Key Generation Protocol: The adversary U, as a malicious user with the attribute set $\widetilde{U} = \widetilde{U_d} \cup \widetilde{U_s}$, execute the interactive *Enhanced_SecretKeyGen* algorithm with authority AA_j. *Enhanced_SecretKeyGen* outputs the secret key for $\mathcal{A}$.

Ideal Key Generation Protocol: A honest user U_h with the same public key and attribute set $\widetilde{U}$ as U, execute the *SecretKeyGen* $\left(PP, PK_j, SK_j, PK_U, \widetilde{U}\right)$ with AA_j and obtains the secret key.

Definition 14: Enhanced_SecretKeyGen is secure against the malicious authority AA_j if there is no PPT AA_j can win the following game with more than a non-negligible advantage.

Init. AA_j submits $\left(PK_{U_0}, \widetilde{U_0}\right)$ and $\left(PK_{U_1}, \widetilde{U_1}\right)$ to the challenger $\mathcal{C}$.

Challenge. Give the malicious authority AA_j two black-box oracles

$$O_0\left(\begin{array}{c} params, PK_j, PK_{U_0}, \widetilde{U_0}, \\ \{decom_{d,i}\}_{a_{d,i}\in\widetilde{U_{0,d}}\cap\widetilde{AA_j}}, \{decom_{s,i}\}_{a_{s,i}\in\widetilde{U_{0,s}}\cap\widetilde{AA_j}} \end{array}\right)$$

and

$$O_1\left(\begin{array}{c} params, PK_j, PK_{U_1}, \widetilde{U_1}, \\ \{decom_{d,i}\}_{a_{d,i}\in\widetilde{U_{1,d}}\cap\widetilde{AA_j}}, \{decom_{s,i}\}_{a_{s,i}\in\widetilde{U_{1,s}}\cap\widetilde{AA_j}} \end{array}\right).$$

$\mathcal{C}$ chooses a bit $b \in \{0, 1\}$, and executes *Enhanced_SecretKey Gen* $\left(U_b \leftrightarrow AA_j\right)$ and *Enhanced_SecretKeyGen* $\left(U_{1-b} \leftrightarrow AA_j\right)$. $\mathcal{C}$ returns the outputs $SK_{b,j}$ and $SK_{1-b,j}$ to AA_j.

Guess. AA_j outputs its guess b' on b. AA_j wins the game if $b' = b$. The advantage of $\mathcal{A}$ is defined to be $Adv_{AA_j} = \left|Pr\left[b' = b\right] - 1/2\right|$.

E. SECURITY REQUIREMENTS

Our PMDAC-ABSC scheme needs to satisfy the following five security requirements.

- Message Confidentiality: Unauthorized users cannot access the sensitive data if the decryption attributes of the user do not satisfy the encryption predicates formulated during the signcryption phase by the data owner.
- Ciphertext Unforgeability: Unauthorized users who do not have the privilege to signcrypt the data cannot create a valid ciphertext which can be successfully verified to satisfy the signing predicates.
- Signcryptor Privacy: The data user should not know the attributes used by the data owner to signcrypt the sensitive data.
- Collusion Resistance: The colluders (users and cloud server) cannot sign the plaintext (collusion resistance of signing) or decrypt the ciphertext (collusion resistance of decryption) by cooperating with each other when they are individually unauthorized to sign or decrypt the data.
- Attribute Privacy: To guarantee user's privacy, it is required that the attribute authorities and the cloud server should not know user's attributes during the secret key generation phase and designcryption phase.

IV. CONSTRUCTION OF PMDAC-ABSC SCHEME

In this section, we propose the construction of PMDAC-ABSC scheme in detail. TABLE 1 summarizes the notations used in our scheme.

A. HIGH-LEVEL OVERVIEW

Basically, the PMDAC-ABSC scheme contains four phases: system initialization, secret key generation, data outsourcing and data designcryption.

- System Initialization: In this phase, CA generates the system public parameters. CA is also responsible for the registrations of attribute authorities and users.
- Secret Key Generation: The legal user can request secret key from attribute authorities in this phase. The secret key can be divided into three parts: common secret key, decryption secret key, and signing secret key.
- Data Outsourcing: After define the signing and encryption predicates, the data owner can signcrypt the plaintext under the signing and encryption predicates, and then upload the ciphertext to the cloud server. To improve the efficiency, the data owner can first encrypts the data by a symmetric encryption scheme and then signcrypt the encryption key. Since our scheme is public verifiable, the cloud server can check that whether the ciphertext is valid and has been signed by the user whose signing attributes satisfy the signing predicates. If the ciphertext is not valid, the cloud server will not store the data.
- Data Designcryption: When the data user queries the sensitive data, the cloud server first runs a partial decryption algorithm and returns a partial ciphertext to user if user's identity is legal and the decryption attributes of user satisfy the encryption predicates contained in the ciphertext. The data user then performs a full decryption on the partial ciphertext and obtains the data.

The work flow of our PMDAC-ABSC scheme is shown in Fig. 2.

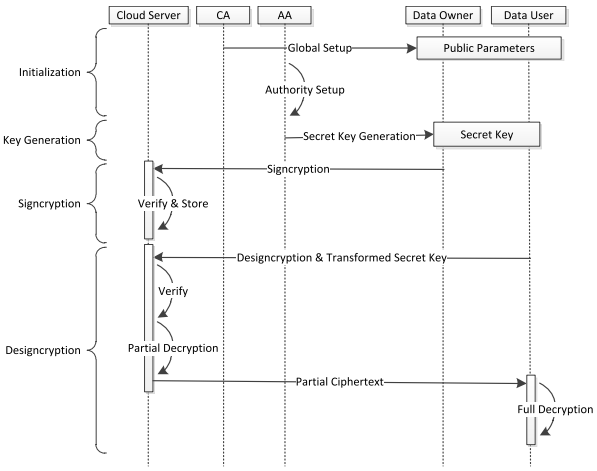


FIGURE 2. Work flow of our scheme.

Compared with some MA-ABE schemes [14]–[16] and MA-ABS schemes [37], our scheme employs a global certificate authority CA which is responsible for assigning unique identities and registering all users and authorities. CA is also required to publish the public-key-certificate of each valid user for identity verification. However, CA is not involved in any attribute management and the creations of the secret keys that are associate with attributes. Without CA, it is difficult to tie together components of a user's secret key and use the key randomization method to prevent collusion

TABLE 1. Notations.

Notations	Meaning
$PP = \{g, h, g_1, \gamma_1, \gamma_2, \{k_0, k_1, \ldots, k_l\}\}$	Public parameters.
H_1, H_2, H_3	Collision resistant hash functions.
uid/aid	Identity of user/authority.
do/du	Identity of data owner (signcryptor)/data user (designcryptor)
$\widetilde{U_d}/\widetilde{U_s}$	Decryption/Signing attribute set of the user.
$\widetilde{AA}_j$	Attribute set of authority AA_j.
$PK_j = \left\{ \begin{matrix} \Delta_j, X_j, Y_j^1, Y_j^2, Z_j^1, Z_j^2, \\ \left\{A_{j,a_{j,k}}, B_{j,a_{j,k}}\right\}_{a_{j,k} \in \widetilde{AA}_j} \end{matrix} \right\}$	Public key of authority AA_j.
$SK_j = \left\{\alpha_j, x_j, y_j, z_j, \left\{\varphi_{j,k}\right\}_{a_{j,k} \in \widetilde{AA}_j}\right\}$	Secret key of authority AA_j.
$a_{j,k}$	kth attribute of AA_j.
$\varphi_{j,k}$	Attribute version key of $a_{j,k}$.
$A_{j,a_{j,k}}, B_{j,a_{j,k}}$	Attribute public keys of $a_{j,k}$.
$CK_{uid,j} = \left\{ \begin{matrix} W_{uid,j}, \Theta_{uid,j}, \Theta'_{uid,j}, \\ R_{uid,j}, R'_{uid,j}, V_{uid,j}, V'_{uid,j} \end{matrix} \right\}$	Common secret key of U_{uid} generated by AA_j.
$DK_{uid,j} = \left\{K^d_{uid,j}, \left\{F^d_{uid,j,a_{j,k}}\right\}_{a_{j,k} \in \widetilde{AA}_t \cap \widetilde{U_d}}\right\}$	Decryption secret key of U_{uid} generated by AA_j.
$SGK_{uid,j} = \left\{K^s_{uid,j}, \left\{F^s_{uid,j,a_{j,k}}\right\}_{a_{j,k} \in \widetilde{AA}_t \cap \widetilde{U_s}}\right\}$	Signing secret key of U_{uid} generated by AA_j.
$SK_{uid,j} = \{CK_{uid,j}, DK_{uid,j}, SGK_{uid,j}\}$	Secret key of U_{uid} generated by AA_j.
$TSK_{uid,j}$	Transformed secret key of $SK_{uid,j}$ used for partial decryption.
$\ell_{s,j}/\ell_{e,j}$	Number of rows of $M_{s,j}/M_{e,j}$ of $\mathcal{R}_{s,j}/\mathcal{R}_{e,j}$.
$\ell_{s,max}/\ell_{e,max}$	Maximum number of rows of $M_{s,j}/M_{e,j}$ of $\mathcal{R}_{s,j}/\mathcal{R}_{e,j}$.
$n_{s,j}/n_{e,j}$	Number of columns of $M_{s,j}/M_{e,j}$ of $\mathcal{R}_{s,j}/\mathcal{R}_{e,j}$.
$M^i_{s,j}/M^i_{e,j}$	ith row of $M_{s,j}/M_{e,j}$.
$M^{(i,k)}_{s,j}/M^{(i,k)}_{e,j}$	(i,k)th element of $M_{s,j}/M_{e,j}$.
I	Set of the authorities whose attributes are selected for signcryption.
$\overrightarrow{v_j}/\overrightarrow{t_j}$	Vectors selected by data owner for signing protocol.
$thre_{tt}$	Time threshold.
s_j, s'_j	Secret shares for AA_j.
CT^p	Partial ciphertext returned by cloud server.
CT_R	Ciphertext component computed by data user for full decryption.
Σ_U/Σ_{AA_j}	Proof of knowledge of user U or authority AA_j
N	Number of authorities

attack [13], [17]. For example, in [12], the colluders can just use their secret keys to decrypt the unauthorized ciphertext encrypted with the subset of the colluding user's combined attribute set. In some MA-ABE schemes such as [14]–[16], a random oracle H (GID) is used to tie the secret keys issued by different authorities. Since our scheme realizes security in the standard model, we employ CA to assign global unique $PK_{uid} = g^{\mu}$ as the public key with which the secret keys generated by different attribute authorities can be tied together for decryption. Thus the collusion resistance can be guaranteed. Additionally, with the help of CA, we can improve the privacy of our scheme by realizing the identity authentication and preventing authorities from forging a virtual user to decrypt the ciphertext. In secret key generation phase, the attribute authority verifies user's certification using the verification key of CA and then generates the secret key for the user. In designcryption phase, the cloud server can verify user's identifier and perform the partial decryption algorithm if the user is valid.

B. SCHEME CONSTRUCTION

1) SYSTEM INITIALIZATION

During the initialization phase, the system generates the public parameters and registers for the data user and authority.

After registration, the authority initializes itself by running *AuthoritySetup* (PP) algorithm.

GlobalSetup $\left(1^k\right)$.

Output a bilinear group $\mathcal{GG}(1^k) \rightarrow (e, p, \mathbb{G}, \mathbb{G}_T)$, where the prime p is the order of group $\mathbb{G}$. CA generates a key pair $\{sk_{CA}, vk_{CA}\}$ for signing and verification.

Output public parameters $PP = \{g, h, g_1, \gamma_1, \gamma_2, \{k_0, k_1, \ldots, k_l\}\}$, where g, h, g_1 are the random generators of $\mathbb{G}$. $\gamma_1, \gamma_2, \{k_0, k_1, \ldots, k_l\} \xleftarrow{R} \mathbb{G}$. $H_1 : \{0,1\}^* \rightarrow \{0,1\}^l$, $H_2 : \mathbb{G} \rightarrow \mathbb{Z}_p^*$ and $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ are cryptographic collision resistant hash functions.

CA verifies the identity description info submitted by the user. If it is a valid user, CA selects a unique identity number uid and sends $PK_{uid} = g^{\mu}$ to the user as the public key. The public key certificate is set by CA as $cert\,(uid) = Sign_{sk_{CA}}(uid, PK_{uid})$. μ is difficult for each user and authority to compute or learn.

CA verifies the identity description info submitted by the authority. If it is a valid authority, CA selects a unique identity number $aid \in [1, N]$ for the authority.

AuthoritySetup (PP). Each authority AA_j runs this algorithm to initialize itself, where $j \in [1, N]$.

Choose α_j, x_j, y_j, z_j randomly from $\mathbb{Z}_p$.

Set $\Delta_j = e(g,g)^{\alpha_j}, X_j = g^{x_j}, Y_j^1 = g^{y_j}, Y_j^2 = g_1^{y_j}, Z_j^1 = g^{z_j}, Z_j^2 = h^{z_j}$.

For each attribute $a_{j,k} \in \widetilde{AA_j}$ where $k \in \left[1, n_j\right]$, Randomly pick $\varphi_{j,k} \xleftarrow{R} \mathbb{Z}_p$ and compute $A_{j,a_{j,k}} = g^{\varphi_{j,k}}$ and $B_{j,a_{j,k}} = h^{\varphi_{j,k}} g^{\frac{1}{z_j + a_{j,k}}}$.

Output the public key $PK_j = \left\{\Delta_j, X_j, Y_j^1, Y_j^2, Z_j^1, Z_j^2, \left\{A_{j,a_{j,k}}, B_{j,a_{j,k}}\right\}_{a_{j,k} \in \widetilde{AA_j}}\right\}$ and secret key $SK_j = \left\{\alpha_j, x_j, y_j, z_j, \left\{\varphi_{j,k}\right\}_{a_{j,k} \in \widetilde{AA_j}}\right\}$.

2) SECRET KEY GENERATION

After the registration, the newly joined user with identity uid needs to request a secret key from authority AA_j, AA_j verifies the user's $cert\,(uid)$ with verification key vk_{CA}. If it is a valid user, AA_j runs *SecretKeyGen* algorithm described as follows to generate the secret key for user.

SecretKeyGen $\left(PP, PK_j, SK_j, PK_{uid}, \widetilde{U}\right)$. AA_j samples $\omega_{uid,j}, \theta_{uid,j}, \delta_{uid,j}$ randomly from $\mathbb{Z}_p$. Then, the algorithm runs the following as follows:

ComKeyGen $\left(PP, \omega_{uid,j}, \theta_{uid,j}, \delta_{uid,j}\right)$. Output for user the common secret key

$$CK_{uid,j} = \Big\{ W_{uid,j} = g^{\omega_{uid,j}}, \Theta_{uid,j} = g_1^{\theta_{uid,j}}, \Theta'_{uid,j} = h^{\theta_{uid,j}}, R_{uid,j} = g_1^{\frac{1}{\theta_{uid,j}}}, R'_{uid,j} = h^{\frac{1}{\theta_{uid,j}}}, V_{uid,j} = g^{\delta_{uid,j}}, V'_{uid,j} = h^{\delta_{uid,j}} \Big\}.$$

DecKeyGen $\left(PP, PK_j, SK_j, PK_{uid}, \widetilde{U_d}\right)$. Output for user the decryption secret key

$$DK_{uid,j} = \left\{ K^d_{uid,j} = g^{\alpha_j} g^{x_j \omega_{uid,j}} g_1^{\theta_{uid,j}} g_1^{\frac{y_j + uid}{\theta_{uid,j}}}, \left\{ F^d_{uid,j,a_{j,k}} = A_{j,a_{j,k}}^{\omega_{uid,j}} (g^{\mu})^{\varphi_{j,k}} \right\}_{a_{j,k} \in \widetilde{AA_i} \cap \widetilde{U_d}} \right\}.$$

SignKeyGen $\left(PP, PK_j, SK_j, PK_{uid}, \widetilde{U_s}\right)$. Output for user the signing secret key

$$SGK_{uid,j} = \left\{ K^s_{uid,j} = g^{\alpha_j} g_1^{\delta_{uid,j}}, \left\{ F^s_{uid,j,a_{j,k}} = A_{j,a_{j,k}}^{\delta_{uid,j}} \right\}_{a_{j,k} \in \widetilde{AA_i} \cap \widetilde{U_s}} \right\}.$$

Finally, AA_joutputs the secret key $SK_{uid,j}$ for user as $SK_{uid,j} = \left\{CK_{uid,j}, DK_{uid,j}, SGK_{uid,j}\right\}$.

3) DATA OUTSOURCING

The data owner with identity uid runs the *Signcryption* algorithm as follows to signcrypt the sensitive data under the defined signing and encryption predicates before outsourcing the data to the cloud server. As in [13], the owner can also first encrypts the data by a symmetric encryption scheme and then signcrypt the encryption key with *Signcryption* algorithm.

Signcryption $\left(\mathcal{M}, PP, \left\{SK_{uid,j}, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j \in I}\right)$. The algorithm takes as input $\mathcal{M}, PP, \left\{SK_{uid,j}, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j \in I}$, where $\mathcal{M} \in \mathbb{G}_T$, $\mathcal{R}_{s,j}, \mathcal{R}_{e,j}$ are signing and encryption predicates respectively, and I is the set that consists of the indexes of the authorities whose attributes are selected to encrypt $\mathcal{M}$. $SK_{uid,j}$ is the signcryptor's secret key. For each authority AA_j where $j \in I$, $\mathcal{R}_{s,j}\left(\widetilde{AA_j^s} \cap \widetilde{U_s}\right) = 1$. Here, $\mathcal{R}_{s,j} := \left(M_{s,j}, \rho_{s,j}\right), \mathcal{R}_{e,j} := \left(M_{e,j}, \rho_{e,j}\right)$ where $M_{s,j}$ (resp. $M_{e,j}$) is an $\ell_{s,j} \times n_{s,j}$ (resp. $\ell_{e,j} \times n_{e,j}$) matrix. Let $M^i_{s,j}$ (resp. $M^i_{e,j}$) be the ith row of the matrix $M_{s,j}$ (resp. $M_{e,j}$). We remove the limitation that $\rho_{e,j}$ should be an injective function. For $j \in I$, the algorithm runs as follows:

Since $\mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_s}\right) = 1$, the algorithm computes a vector $\overrightarrow{v_j} = \left(v_{j,1}, v_{j,2}, \ldots, v_{j,\ell_{s,j}}\right) \in \mathbb{Z}_p^{\ell_{s,j}}$ such that $\overrightarrow{v_j} \cdot M_{s,j} = \vec{1}$. Then the algorithm chooses $\overrightarrow{t_j} = \left(t_{j,1}, t_{j,2}, \ldots, t_{j,\ell_{s,j}}\right) \in \mathbb{Z}_p^{\ell_{s,j}}$ such that $\overrightarrow{t_j} \cdot M_{s,j} = \vec{0}$. The existence of such a vector $\overrightarrow{t_j}$ is discussed in Definition 6.

Choose $\delta'_{uid,j} \xleftarrow{R} \mathbb{Z}_p^*$ and re-randomize the secret key as $K^s_{uid,j} = g^{\alpha_j} g_1^{\delta_{uid,j}} g_1^{\delta'_{uid,j}}, V_{uid,j} = g^{\delta_{uid,j}} g^{\delta'_{uid,j}}, \left\{F^s_{uid,j,k} = F^s_{uid,j,k} A_{j,a_{j,k}}^{\delta'_{uid,j}} = A_{j,a_{j,k}}^{\delta''_{uid,j}}\right\}_{a_{j,k} \in \widetilde{AA_i} \cap \widetilde{U_s}}$.

Pick $s_j, s'_j \xleftarrow{R} \mathbb{Z}_p$ for each $j \in I$ such that $\sum_I s'_j = 0$ and set $\overrightarrow{\varepsilon_j} = \left(s_j, \varepsilon_{j,2}, \ldots, \varepsilon_{j,n_{e,j}}\right) \in \mathbb{Z}_p^{n_{e,j}}$, $\overrightarrow{\varepsilon'_j} = \left(s'_j, \varepsilon'_{j,2}, \ldots, \varepsilon'_{j,n_{e,j}}\right) \in \mathbb{Z}_p^{n_{e,j}}, \lambda_{j,i} = M^i_{e,j} \overrightarrow{\varepsilon_j}, \lambda'_{j,i} = M^i_{e,j} \overrightarrow{\varepsilon'_j}$.

Select $\left\{r_{j,1}, r_{j,2}, \ldots, r_{j,\ell_{e,j}}\right\} \xleftarrow{R} \mathbb{Z}_p$.

The algorithm computes the following terms: $C_0 = \mathcal{M} \prod_{j \in I} \Delta_j^{s_j}, C_{j,1} = g^{s_j}, C_{j,2} = \left(Y_j^1\right)^{s_j}, C_{j,3} = \left(\gamma_1 \gamma_2^{\pi_j}\right)^{s_j}$

where $\pi_j = H_2\left(C_{j,1}\right)$.

$$\left\{C_{j,4,i} = g^{x_j\lambda_{j,i}} A_{j,\rho_{e,j}(i)}^{-r_{j,i}},\right.$$
$$\left. C_{j,5,i} = g_1^{\lambda'_{j,i}} A_{j,\rho_{e,j}(i)}^{-r_{j,i}}, D_{j,i} = g^{r_{j,i}}\right\}_{i\in[\ell_{e,j}]},$$
$$\left\{S_{1,j,i} = g^{v_{j,i}\delta''_{uid,j}+t_{j,i}}\right\}_{i\in[\ell_{s,j}]}.$$
$$H_1\left(\prod_{i\in[\ell_{s,j}]} S_{1,j,i}, tt, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right) = \left(b_{j,1}, b_{j,2}, \ldots, b_{j,l}\right) \in \{0,1\}^l.$$
$$H_3\left(C_{j,1}, C_{j,2}, C_{j,3}, \prod_{i\in[\ell_{e,j}]} C_{j,4,i}, \prod_{i\in[\ell_{e,j}]} C_{j,5,i}, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right) = \beta_j.$$
$$S_{2,j} = K^s_{uid,j}\left(k_0 \prod_{i=1}^{l} k_i^{b_{j,i}}\right)^{s_j} \cdot C_{j,3}^{\beta_j}\left(\prod_{i=1}^{\ell_{s,j}}\left(F^s_{uid,j,\rho_{s,j}(i)}\right)^{v_{j,i}}\left(A_{j,\rho_{s,j}(i)}\right)^{t_{j,i}}\right).$$

Output

$$CT = \left\{C_0, \left\{\begin{matrix} C_{j,1}, C_{j,2}, C_{j,3}, \left\{C_{j,4,i}, C_{j,5,i}, D_{j,i}\right\}_{i\in[\ell_{e,j}]}, \\ \left\{S_{1,j,i}\right\}_{i\in[\ell_{s,j}]}, S_{2,j}\end{matrix}\right\}_{j\in I}, tt\right\},$$

where tt is the current time.

4) DATA DESIGNCRYPTION

If the owner's attributes satisfy the signing predicates implicitly contained in the ciphertext, then any party can successfully verify the ciphertext (public verifiability). If the receiver's decryption attributes satisfy the encryption predicates embedded in the ciphertext, then the decryption phase can be launched to access the plaintext.

DeSigncryption $\left(PP, CT, PK_{uid}, \left\{SK_{uid,j}\right\}_{j\in I}\right)$. Assume that $thre_{tt}$ is a predefined time threshold for designcryption and $\widetilde{tt}$ is the current time. If $\left|\widetilde{tt} - tt\right| > thre_{tt}$ or $\mathcal{R}_{e,j}\left(\widetilde{AA}_j \cap \widetilde{U}_d\right) \neq 1$ for any $j \in I$, the algorithm returns $\perp$. Otherwise, the algorithm performs as follows.

Verify (PP, CT).

For $j \in I$, sample $\left\{\tau_2, \tau_3, \ldots, \tau_{n_{s,j}}\right\} \xleftarrow{R} \mathbb{Z}_p^*$ and compute $\varpi_{j,i} = \left(1, \tau_2, \tau_3, \ldots, \tau_{n_{s,j}}\right) \cdot M^i_{s,j}$, where $i \in \left[\ell_{s,j}\right]$. $H_1\left(\prod_{i\in[\ell_{s,j}]} S_{1,j,i}, tt, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right) = \left(b_{j,1}, b_{j,2}, \ldots, b_{j,l}\right)$, $H_2\left(C_{j,1}\right) = \pi_j$ and $H_3\left(C_{j,1}, C_{j,2}, C_{j,3}, \prod_{i\in[\ell_{e,j}]} C_{j,4,i}, \prod_{i\in[\ell_{e,j}]} C_{j,5,i}, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right) = \beta_j$.

Check the validity of the ciphertext using the following equation $\prod_I \Delta_j$, as shown at the bottom of this page.

If it is invalid, return $\perp$, otherwise, proceed as follows.

Decryption $\left(PP, CT, PK_{uid}, \left\{SK_{uid,j}\right\}_{j\in I}\right)$.

In some previous works, such as [13], [16], and[17], the user sends the secret keys corresponding to the attributes directly to the cloud server. If the server knows the user's attributes, it can calculate the constant numbers $\{\sigma_i\}_{i\in[\ell_{e,j}]}$ which can reconstruct the shared secret s as $\sum_{i=1}^{\ell_{e,j}} \sigma_i\lambda_i = s$, and then perform the bilinear pairing operation on the pair of secret key and ciphertext component for each row of the encryption predicate matrix. However, this technique cannot be applied to our scheme, as the cloud server cannot know the user's attributes. In our construction, we embed the constant numbers $\{\sigma_i\}_{i\in[\ell_{e,j}]}$ into secret keys as exponentials before sending them to the cloud. However, according to Definition 5, for all $\rho_{e,j}(i) \notin \widetilde{AA}_j \cap \widetilde{U}_d$ we have $\sigma_i = 0$. If $\{\sigma_i\}_{i\in[\ell_{e,j}]}$ are directly embedded, the secret keys associated with $\rho_{e,j}(i) \in \widetilde{AA}_j \cap \widetilde{U}_d$ are distinguishable from the keys associated with $\rho_{e,j}(i) \notin \widetilde{AA}_j \cap \widetilde{U}_d$; thus, the cloud server can infer the user's attributes. Therefore, to make the cloud server's view on secret keys corresponding to each $\rho_{e,j}(i)$ computationally indistinguishable, we randomize the secret keys with some random numbers chosen from $\mathbb{Z}_p^*$.

The designcryptor randomly chooses secret values ϕ, $r \xleftarrow{R} \mathbb{Z}_p^*$.

If for each $j \in I$, $\mathcal{R}_{e,j}\left(\widetilde{AA}_j \cap \widetilde{U}_d\right) = 1$, compute $\overrightarrow{\sigma_j} = \left(\sigma_{j,1}, \sigma_{j,2}, \ldots, \sigma_{j,\ell_{e,j}}\right) \in \mathbb{Z}_p^{\ell_{e,j}}$ such that $\sum_{i=1}^{\ell_{e,j}} \sigma_{j,i} M^i_{e,j} = \vec{1}$, where $\sigma_{j,i} = 0$ for all i where $\rho_{e,j}(i) \notin \widetilde{AA}_j \cap \widetilde{U}_d$.

For each i, randomly choose $a_{j,i}, b_{j,i}, c_{j,i} \xleftarrow{R} \mathbb{Z}_p^*$.

Compute the transformed secret key as $TSK_{uid,j} = \left\{\Theta^{\phi}_{uid,j}, R^{\phi}_{uid,j}, K^{d\ \phi}_{uid,j}, \{W_i, P_i, F_i\}_{i\in[\ell_{e,j}]}\right\}$, where $W_i = W^{\phi\sigma_{j,i}}_{uid,j}$, $P_i = PK^{\phi\sigma_{j,i}}_{uid}$, $F_i = F^{d\ \phi\sigma_{j,i}}_{uid,j,\rho_{e,j}(i)}$ for i that $\rho_{e,j}(i) \in \widetilde{AA}_j \cap \widetilde{U}_d$ and $W_i = g^{a_{j,i}r}$, $P_i = g^{b_{j,i}r}$, $F_i = g^{c_{j,i}r}$ for i that $\rho_{e,j}(i) \notin \widetilde{AA}_j \cap \widetilde{U}_d$.

The designcryptor sends $\left\{TSK_{uid,j}, \left\{a_{j,i}, b_{j,i}, c_{j,i}\right\}_{i\in[\ell_{e,j}]}\right\}_{j\in I}$ to the cloud server. For the transformed secret key $TSK_{uid,j}$, it is infeasible for the cloud server to distinguish the components $\{W_i, P_i, F_i\}_{i\in[\ell_{e,j}]}$ associated with $\rho_{e,j}(i) \in \widetilde{AA}_j \cap \widetilde{U}_d$ from the components associated with $\rho_{e,j}(i) \notin \widetilde{AA}_j \cap \widetilde{U}_d$, due to the randomly chosen numbers $\phi, r, a_{j,i}, b_{j,i}, c_{j,i}$.

Then *Decryption* can be divided into the following two sub-algorithms.

PartialDecryption $\left(PP, CT, PK_{uid}, \left\{SK_{uid,j}\right\}_{j\in I}\right)$. This algorithm is performed by the cloud server. The cloud sends $CT^p = \left\{C_0, \left\{\left\{C^{a_{j,i}}_{j,4,i}, C^{b_{j,i}}_{j,5,i}, D^{c_{j,i}}_{j,i}\right\}_{i\in[\ell_{e,j}]}\right\}_{j\in I}, CT_x\right\}$ to the user,

$$\prod_I \Delta_j = \frac{\prod_I e\left(S_{2,j}, g\right)}{\prod_I \left(e\left(k_0 \prod_{i=1}^{l} k_i^{b_{j,i}}, C_{j,1}\right) e\left(\left(\gamma_1\gamma_2^{\pi_j}\right)^{\beta_j}, C_{j,1}\right)\left(\prod_{i=1}^{\ell_{s,j}} e\left(g_1^{\varpi_{j,i}} A_{j,\rho_{s,j}(i)}, S_{1,j,i}\right)\right)\right)}.$$

where $CT_x = CT_R \frac{\prod_I e\left(\Theta^{\phi}_{uid,j}, C_{j,1}\right) e\left(R^{\phi}_{uid,j}, C_{j,2}\right) e\left(R^{\phi}_{uid,j}, C_{j,1}\right)^{uid}}{\prod_I e\left(K^{d\phi}_{uid,j}, C_{j,1}\right)}$ and $CT_R = \prod_I \prod_{i\in[\ell_{e,j}]} \left(e\left(W_i, C_{j,4,i}\right) e\left(F_i, D_{j,i}\right) e\left(P_i, C_{j,5,i}\right)\right)$.

FullDecryption $\left(PP, CT^p, PK_{uid}, \left\{SK_{uid,j}\right\}_{j\in I}\right)$. This algorithm is performed on the user side. After receiving CT^p, the data user computes $CT_R = e\left(g^r, \prod_{j\in I} \prod_{\rho_{e,j}(i)\notin \widetilde{AA_j}\cap\widetilde{U_d}} C^{a_{j,i}}_{j,4,i} D^{c_{j,i}}_{j,i} C^{b_{j,i}}_{j,5,i}\right)$. Then the message $\mathcal{M}$ can be recovered $\mathcal{M} = C_0 \left(\frac{CT_x}{CT_R}\right)^{\frac{1}{\phi}}$.

C. CORRECTNESS

Assume the signcryptor's identity is *do*. If $\left|\widetilde{tt} - tt\right| \le thre_{tt}$ and $\mathcal{R}_{e,j}\left(\widetilde{AA_j} \cap \widetilde{U_d}\right) = 1$ for each j ∈ I, then the ciphertext can be verified and decrypted as explained subsequently.

$$\begin{aligned} S_{2,j} &= K^s_{do,j} \left(k_0 \prod_{i=1}^{l} k_i^{b_{j,i}}\right)^{s_j} \\ &\quad \cdot C^{\beta_j}_{j,3} \left(\prod_{i=1}^{\ell_{s,j}} \left(F^s_{do,j,\rho_{s,j}(i)}\right)^{v_{j,i}} \left(A_{j,\rho_{s,j}(i)}\right)^{t_{j,i}}\right) \\ &= g^{\alpha_j} g_1^{\delta_{do,j}} \left(k_0 \prod_{i=1}^{l} k_i^{b_{j,i}}\right)^{s_j} \left(\gamma_1 \gamma_2^{\pi_j}\right)^{s_j\beta_j} \\ &\quad \cdot \left(\prod\nolimits_{i=1}^{\ell_{s,j}} \left(A_{j,\rho_{s,j}(i)}\right)^{\left(\delta_{do,j} v_{j,i} + t_{j,i}\right)}\right). \end{aligned}$$

Since $\overrightarrow{v_j} \cdot M_{s,j} = \vec{1}$ and $\overrightarrow{t_j} \cdot M_{s,j} = \vec{0}$, we have

$$\begin{aligned} &\sum\nolimits_{i=1}^{\ell_{s,j}} \varpi_{j,i} \left(\delta_{do,j} v_{j,i} + t_{j,i}\right) \\ &\quad = \sum\nolimits_{i=1}^{\ell_{s,j}} \left(1, \tau_2, \tau_3, \ldots, \tau_{n_{s,j}}\right) \cdot M^i_{s,j} \left(\delta_{do,j} v_{j,i} + t_{j,i}\right) \\ &\quad = \left(1, \tau_2, \tau_3, \ldots, \tau_{n_{s,j}}\right) \delta_{do,j} \sum\nolimits_{i=1}^{\ell_{s,j}} M^i_{s,j} v_{j,i} \\ &\qquad + \left(1, \tau_2, \tau_3, \ldots, \tau_{n_{s,j}}\right) \sum\nolimits_{i=1}^{\ell_{s,j}} M^i_{s,j} t_{j,i} = \delta_{do,j}. \end{aligned}$$

Hence,

This demonstrates the correctness of *Verify* algorithm. If $\sum_{i=1}^{\ell_{e,j}} \sigma_{j,i}\lambda_{j,i} = s_j$, $\sum_{i=1}^{\ell_{e,j}} \sigma_{j,i}\lambda'_{j,i} = s'_j$, $\sum_I s'_j = 0$, then

$$\begin{aligned} &\prod\nolimits_{j\in I} \prod\nolimits_{i\in[\ell_{e,j}]} \left(e\left(W_i, C_{j,4,i}\right) e\left(F_i, D_{j,i}\right) e\left(P_i, C_{j,5,i}\right)\right) \\ &\quad = CT_R \prod\nolimits_{j\in I} e\left(g, g\right)^{\phi\omega_{uid,j} x_j s_j} e\left(g, g_1\right)^{\phi\mu s'_j} \\ &\quad = CT_R \prod\nolimits_{j\in I} e\left(g, g\right)^{\phi\omega_{uid,j} x_j s_j}, \end{aligned}$$

where

$$CT_R = e\left(g^r, \prod\nolimits_{j\in I} \prod\nolimits_{\rho_{e,j}(i)\notin \widetilde{AA_j}\cap\widetilde{U_d}} C^{a_{j,i}}_{j,4,i} D^{c_{j,i}}_{j,i} C^{b_{j,i}}_{j,5,i}\right).$$

Therefore, $C_0 \left(\frac{CT_x}{CT_R}\right)^{\frac{1}{\phi}} = \frac{\mathcal{M} \prod_{j\in I} \Delta_j^{s_j}}{\prod_{j\in I} e(g,g)^{\alpha_j s_j}} = \mathcal{M}$. This exhibits the correctness of *Decryption* algorithm.

D. PUBLIC VERIFIABILITY

The *Verify* (PP, CT) algorithm only takes as input the public parameters and ciphertext. Therefore, any user or cloud server who can access the ciphertext can check whether the signcryptor's attributes satisfy the signing predicate. Thus our construction is public verifiability.

E. ATTRIBUTE PRIVACY PROTECTION IN SECRET KEY GENERATION PHASE

To generate the secret key of the user U, the authority AA_j randomly chooses three numbers $\omega_{uid,j}, \theta_{uid,j}, \delta_{uid,j} \xleftarrow{R} \mathbb{Z}_p$ to tie $SK_{uid,j}$ to the user's identity *uid* and uses them to generate the components $F^d_{uid,j,a_{j,k}}$ and $F^s_{uid,j,a_{j,k}}$ for the attributes $a_{j,k} \in \widetilde{AA_j} \cap \widetilde{U}$. Then AA_j can infer $A_{j,a_{j,k}}$ by computing $\left(\frac{F^d_{uid,j,a_{j,k}}}{(g^{\mu})^{\varphi_{j,k}}}\right)^{\frac{1}{\omega_{uid,j}}}$, $\left(F^s_{uid,j,a_{j,k}}\right)^{\frac{1}{\delta_{uid,j}}}$. To protect the privacy of the attributes, we design *Enhanced_SecretKeyGen* to generate the secret key for the user to prevent the authority from directly obtaining $a_{j,k}$ or recording $\omega_{uid,j}, \theta_{uid,j}, \delta_{uid,j}$.

Enhanced_SecretKeyGen $\left(PP, PK_j, SK_j, PK_{uid}, \widetilde{U}\right)$. Taking as input the public parameter PP, $\left\{PK_j, SK_j\right\}$ of AA_j, PK_{uid} of user U with attributes $\widetilde{U}$, the algorithm outputs the secret key as follows.

1) U selects $a_1, a_2, d_1, d_2, d_3, a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, d'_1, d'_2, d'_3 \xleftarrow{R} \mathbb{Z}_p$. Let $D_1 = d_1 d_2$, $D_2 = d_2 d_3$, $D'_1 = d'_1 d'_2$, $D'_2 = d'_2 d'_3$.

To commit the attribute a_x, U computes $\Psi^1_{a_x} = g^{a_x} h^{a'_4}$.

To implement the set member proof defined in Definition 7, U computes $\left\{\Psi^{2d}_{a_x} = B^{D_1}_{a_x}, \Psi^{3d}_{a_x} = A^{D_1}_{a_x}, \Psi^{4d}_{a_x} = e\left(\Psi^{3d}_{a_x}, h\right)^{a'_5} e\left(\Psi^{2d}_{a_x}, g\right)^{-a'_5} e\left(g, g\right)^{D'_1}\right\}_{a_x\in\widetilde{U_d}}$ and $\left\{\Psi^{2s}_{a_x} = B^{D_2}_{a_x}, \Psi^{3s}_{a_x} = A^{D_2}_{a_x}, \Psi^{4s}_{a_x} = e\left(\Psi^{3s}_{a_x}, h\right)^{a'_5} e\left(\Psi^{2s}_{a_x}, g\right)^{-a'_5} e\left(g, g\right)^{D'_2}\right\}_{a_x\in\widetilde{U_s}}$. $\left\{\Psi^5_{a_x} = g^{a'_5} h^{a'_6}\right\}_{a_x\in\widetilde{AA_j}\cap\widetilde{U}}$. Namely, in Definition 7, we set $C = \Psi^1_{a_x}$, $D = \Psi^5_{a_x}$, $Y = A_x$, $\sigma = a_x$, $r = a'_4$, $s = a'_5$, $m = a'_6$ For $a_x \in \widetilde{AA_j} \cap \widetilde{U_d}$, $v = D_1$, $t = D'_1$, $V = \Psi^{2d}_{a_x}$, $a = \Psi^{4d}_{a_x}$. For $a_x \in \widetilde{AA_j} \cap \widetilde{U_s}$, $v = D_2$, $t = D'_2$, $V = \Psi^{2s}_{a_x}$, $a = \Psi^{4s}_{a_x}$.

Finally, U computes the following terms: $\Theta_1 = X_j^{d_1}$, $\Theta_2 = g^{D_1}$, $\Theta_3 = h^{a_1} g_1^{uid}$, $\Theta_4 = \Theta_3^{a_2}$, $\Theta_5 = \left(Y_j^2\right)^{a_2}$, $\Theta_6 = g_1^{\frac{1}{a_2}}$, $\Theta_7 = g^{D_2}$, $\Theta_8 = h^{a_1} g_1^{d_3}$, and $\Theta'_1 = X_j^{d'_1}$, $\Theta'_2 = g^{D'_1}$, $\Theta'_3 = h^{a'_1} g_1^{a'_3}$, $\Theta'_4 = \Theta_3^{a'_2}$, $\Theta'_5 = \left(Y_j^2\right)^{a'_2}$, $\Theta'_6 = g_1^{\frac{1}{a'_2}}$, $\Theta'_7 = g^{D'_2}$, $\Theta'_8 = h^{a'_1} g_1^{d'_3}$ to construct the proof of knowledge $\Sigma_U =$

$$Pok \left\{ \begin{array}{c} (a_1, a_2, d_1, d_3, D_1, D_2, a_x) : \Theta_1 = X_j^{d_1} \wedge \Theta_2 = g^{D_1} \\ \wedge \Theta_3 = h^{a_1} g_1^{uid} \wedge \Theta_4 = \Theta_3^{a_2} \wedge \Theta_5 = \left(Y_j^2\right)^{a_2} \\ \wedge \Theta_6 = g_1^{\frac{1}{a_2}} \wedge \Theta_7 = g^{D_2} \wedge \Theta_8 = h^{a_1} g_1^{d_3} \wedge \\ \left\{ \begin{array}{c} \frac{e\left(Z_j^1, \Psi^{2d}_{a_x}\right)}{e\left(Z_j^2, \Psi^{3d}_{a_x}\right)} = \\ e\left(g, \Psi^{2d}_{a_x}\right)^{-a_x} e\left(h, \Psi^{3d}_{a_x}\right)^{a_x} e\left(g, g\right)^{D_1} \end{array} \right\}_{a_x\in\widetilde{AA_j}\cap\widetilde{U_d}} \\ \left\{ \begin{array}{c} \frac{e\left(Z_j^1, \Psi^{2s}_{a_x}\right)}{e\left(Z_j^2, \Psi^{3s}_{a_x}\right)} = \\ e\left(g, \Psi^{2s}_{a_x}\right)^{-a_x} e\left(h, \Psi^{3s}_{a_x}\right)^{a_x} e\left(g, g\right)^{D_2} \end{array} \right\}_{a_x\in\widetilde{AA_j}\cap\widetilde{U_s}} \end{array} \right\}.$$

$$\frac{\prod_I e\left(S_{2,j}, g\right)}{\prod_I \left(e\left(k_0 \prod_{i=1}^{l} k_i^{b_{j,i}}, C_{j,1}\right) e\left(\left(\gamma_1 \gamma_2^{\pi_j}\right)^{\beta_j}, C_{j,1}\right) \left(\prod_{i=1}^{\ell_{s,j}} e\left(g_1^{\varpi_{j,i}} A_{j,\rho_{s,j}(i)}, S_{1,j,i}\right)\right)\right)}$$
$$= \prod_I \frac{e\left(g^{\alpha_j}, g\right) e\left(g_1^{\delta_{do,j}}, g\right) e\left(\left(\prod_{i=1}^{\ell_{s,j}} \left(A_{j,\rho_{s,j}(i)}\right)^{\left(\delta_{do,j} v_{j,i} + t_{j,i}\right)}\right), g\right)}{\prod_{i=1}^{\ell_{s,j}} e\left(A_{j,\rho_{s,j}(i)}, g^{\delta_{do,j} v_{j,i} + t_{j,i}}\right) e\left(g_1, g\right)^{\delta_{do,j}}} = \prod_I \Delta_j.$$

U sends $\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6, \Theta_7, \Theta_8, \Theta'_1, \Theta'_2, \Theta'_3, \Theta'_4, \Theta'_5, \Theta'_6, \Theta'_7, \Theta'_8$, and $\left\{\Psi^1_{a_x}, \Psi^{2d}_{a_x}, \Psi^{2s}_{a_x}, \Psi^{3d}_{a_x}, \Psi^{3s}_{a_x}, \Psi^{4d}_{a_x}, \Psi^{4s}_{a_x}, \Psi^5_{a_x}\right\}_{a_x \in \widetilde{AA_j} \cap \widetilde{U}}$ to AA_j.

2) AA_j sends $c \xleftarrow{R} \mathbb{Z}_p$ to U.

3) U computes $a''_1 = a'_1 - ca_1, a''_2 = a'_2 - ca_2, a''_3 = a'_3 - cuid, a''_4 = a'_6 - ca'_4, a''_5 = a'_5 - ca_x, a''_6 = \frac{1}{a'_2} - c\frac{1}{a_2}, d''_1 = d'_1 - cd_1, d''_3 = d'_3 - cd_3, D''_1 = D'_1 - cD_1, D''_2 = D'_2 - cD_2$. U sends $a''_1, a''_2, a''_3, a''_4, a''_5, a''_6, d''_1, d''_3, D''_1, D''_2$ to AA_j.

4) AA_j checks whether $\Theta'_1 = X_j^{a''_1} \Theta_1^c, \Theta'_2 = g^{D''_1} \Theta_2^c, \Theta'_3 = h^{a''_1} g_1^{a''_3} \Theta_3^c, \Theta'_4 = \Theta_3^{a''_2} \Theta_4^c, \Theta'_5 = \left(Y_j^2\right)^{a''_2} \Theta_5^c, \Theta'_6 = g_1^{a''_6} \Theta_6^c, \Theta'_7 = g^{D''_2} \Theta_7^c, \Theta'_8 = h^{a''_1} g_1^{d''_3} \Theta_8^c, \Psi^5_{a_x} = g^{a''_5} h^{a''_4} \left(\Psi^1_{a_x}\right)^c$, $e\left(\Theta'_5, \Theta'_6\right) = e\left(\Theta_5, \Theta_6\right) = e\left(Y_j^2, g_1\right)$ hold and $\Psi^{4d}_{a_x} = e\left(g, \Psi^{2d}_{a_x}\right)^{-a''_5} e\left(h, \Psi^{3d}_{a_x}\right)^{a''_5} e\left(g, g\right)^{D''_1} \left(\frac{e\left(Z_j^1, \Psi^{2d}_{a_x}\right)}{e\left(Z_j^2, \Psi^{3d}_{a_x}\right)}\right)^c$ for $a_x \in \widetilde{AA_j} \cap \widetilde{U_d}$ and $\Psi^{4s}_{a_x} = e\left(g, \Psi^{2s}_{a_x}\right)^{-a''_5} e\left(h, \Psi^{3s}_{a_x}\right)^{a''_5} e\left(g, g\right)^{D''_2} \left(\frac{e\left(Z_j^1, \Psi^{2s}_{a_x}\right)}{e\left(Z_j^2, \Psi^{3s}_{a_x}\right)}\right)^c$ for $a_x \in \widetilde{AA_j} \cap \widetilde{U_s}$.

If any of the above equations does not hold, AA_j aborts. Otherwise, AA_j selects $c_{uid}, e_{uid}, m_{uid}, c'_{uid}, e'_{uid}, m'_{uid}, c''_{uid}, l_{uid}, x_{uid} \xleftarrow{R} \mathbb{Z}_p$. To commits $c_{uid}, e_{uid}, m_{uid}, \alpha_j, x_j$ and construct proof of knowledge Σ_{AA_j} of $c_{uid}, e_{uid}, m_{uid}, \alpha_j, x_j$, AA_j computes $\Upsilon_1 = g_1^{c_{uid}}, \Upsilon_2 = g_1^{\frac{1}{c_{uid}}}, \Upsilon_3 = h^{c_{uid}}, \Upsilon_4 = h^{\frac{1}{c_{uid}}}, \Upsilon_5 = g^{e_{uid}}, \Upsilon_6 = g^{m_{uid}}, \Upsilon_7 = h^{m_{uid}}, \Upsilon'_1 = g_1^{c'_{uid}}, \Upsilon'_2 = g_1^{c''_{uid}}, \Upsilon'_3 = h^{c'_{uid}}, \Upsilon'_4 = h^{c''_{uid}}, \Upsilon'_5 = g^{e'_{uid}}, \Upsilon'_6 = g^{m'_{uid}}, \Upsilon'_7 = h^{m'_{uid}}$ and $K^{d*}_{uid,j} = g^{\alpha_j - \mu x_j} \Theta_1^{e_{uid}} \Theta_6^{c_{uid}} \left(\Theta_4 \Theta_5\right)^{\frac{1}{c_{uid}}}, K^{d*}_{uid,j}{}' = g^{l_{uid} - \mu x_{uid}} \Theta_1^{e'_{uid}} \Theta_6^{c'_{uid}} \left(\Theta_4 \Theta_5\right)^{c''_{uid}}, K^{s*}_{uid,j} = g^{\alpha_j} \Theta_8^{m_{uid}}, K^{s*}_{uid,j}{}' = g^{l_{uid}} \Theta_8^{m_{uid}\prime}$. Then AA_j sends $\Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4, \Upsilon_5, \Upsilon_6, \Upsilon_7, \Upsilon'_1, \Upsilon'_2, \Upsilon'_3, \Upsilon'_4, \Upsilon'_5, \Upsilon'_6, \Upsilon'_7, K^{d*}_{uid,j}, K^{d*}_{uid,j}{}', K^{s*}_{uid,j}, K^{s*}_{uid,j}{}', \left\{\Phi^d_{a_x} = \Psi^{3d^{e_{uid}}}_{a_x}, \Phi^d_{a_x}{}' = \Psi^{3d^{e'_{uid}}}_{a_x}\right\}_{a_x \in \widetilde{AA_j} \cap \widetilde{U_d}}$ and $\left\{\Phi^s_{a_x} = \Psi^{3s^{m_{uid}}}_{a_x}, \Phi^s_{a_x}{}' = \Psi^{3s^{m'_{uid}}}_{a_x}\right\}_{a_x \in \widetilde{AA_j} \cap \widetilde{U_s}}$ to U.

5) U picks $c' \xleftarrow{R} \mathbb{Z}_p$ and sends it to AA_j.

6) AA_j computes $c'''_{uid} = c'_{uid} - c'c_{uid}, e''_{uid} = e'_{uid} - c'e_{uid}, c^*_{uid} = c''_{uid} - \frac{c'}{c_{uid}}, l'_{uid} = l_{uid} - c'\alpha_j, m''_{uid} = m'_{uid} - c'm_{uid}, x''_{uid} = x_{uid} - c'x_j$. Then it sends $c'''_{uid}, e''_{uid}, c^*_{uid}, l'_{uid}, m''_{uid}, x''_{uid}$ to U.

7) U checks $\Upsilon'_1 = \Upsilon_1^{c'} g_1^{c'''_{uid}}, \Upsilon'_2 = \Upsilon_2^{c'} g_1^{c^*_{uid}}, \Upsilon'_3 = \Upsilon_3^{c'} h^{c'''_{uid}}, \Upsilon'_4 = \Upsilon_4^{c'} h^{c^*_{uid}}, \Upsilon'_5 = \Upsilon_5^{c'} g^{e''_{uid}}, \Upsilon'_6 = \Upsilon_6^{c'} g^{m''_{uid}}, \Upsilon'_7 = \Upsilon_7^{c'} h^{m''_{uid}}$ and $K^{d*}_{uid,j}{}' = g^{l'_{uid} - \mu x''_{uid}} \Theta_1^{e''_{uid}} \Theta_6^{c'''_{uid}} \left(\Theta_4 \Theta_5\right)^{c^*_{uid}} K^{d*}_{uid,j}{}^{c'}, K^{s*}_{uid,j}{}' = g^{l'_{uid}} \Theta_8^{m''_{uid}} K^{s*}_{uid,j}{}^{c'}, e\left(\Upsilon_1, \Upsilon_2\right) = e\left(g_1, g_1\right), e\left(\Upsilon_3, \Upsilon_4\right) = e\left(h, h\right)$.

If all of the above equations hold, U computes the secret key as: $W_{uid,j} = \frac{\Upsilon_5^{d_1}}{g^\mu} = g^{e_{uid} d_1 - \mu} = g^{\omega_{uid,j}}, \Theta_{uid,j} = \Upsilon_1^{\frac{1}{a_2}} = g_1^{\frac{c_{uid}}{a_2}} = g_1^{\theta_{uid,j}}, \Theta'_{uid,j} = \Upsilon_3^{\frac{1}{a_2}} = h^{\frac{c_{uid}}{a_2}} = h^{\theta_{uid,j}}, R_{uid,j} = \Upsilon_2^{a_2} = g_1^{\frac{a_2}{c_{uid}}} = g_1^{\frac{1}{\theta_{uid,j}}}, R'_{uid,j} = \Upsilon_4^{a_2} = h^{\frac{a_2}{c_{uid}}} = h^{\frac{1}{\theta_{uid,j}}}, V_{uid,j} = \Upsilon_6^{d_3} = g^{m_{uid} d_3} = g^{\delta_{uid,j}}, V'_{uid,j} = \Upsilon_7^{d_3} = h^{m_{uid} d_3} = h^{\delta_{uid,j}}. K^d_{uid,j} = \frac{K^{d*}_{uid,j} \Upsilon_1^{\frac{1}{a_2}}}{\Upsilon_4^{a_1 a_2}} = g^{\alpha_j} g^{x_j (e_{uid} d_1 - \mu)} g_1^{\theta_{uid,j}} g_1^{\frac{y_j + uid}{\theta_{uid,j}}} = g^{\alpha_j} g^{x_j \omega_{uid,j}} g_1^{\theta_{uid,j}} g_1^{\frac{y_j + uid}{\theta_{uid,j}}}, K^s_{uid,j} = \frac{K^{s*}_{uid,j}}{\Upsilon_7^{a_1}} = g^{\alpha_j} g_1^{\delta_{uid,j}}$.

For attributes $a_x \in \widetilde{AA_j} \cap \widetilde{U_d}, F^d_{a_x} = \left(\Phi^d_{a_x}\right)^{\frac{1}{d_2}} = \Psi^{3d^{\frac{e_{uid}}{d_2}}}_{a_x} = A^{d_1 e_{uid}}_{a_x} = A^{\omega_{uid,j} + \mu}_{a_x}$, and for $a_x \in \widetilde{AA_j} \cap \widetilde{U_s}, F^s_{a_x} = \left(\Phi^s_{a_x}\right)^{\frac{1}{d_2}} = A^{d_3 m_{uid}}_{a_x} = A^{\delta_{uid,j}}_{a_x}$. Namely, we implicitly set $\omega_{uid,j} = e_{uid} d_1 - \mu, \theta_{uid,j} = \frac{c_{uid}}{a_2}, \delta_{uid,j} = m_{uid} d_3$.

V. SECURITY ANALYSIS

In this section, we prove the security of our PMDAC-ABSC scheme. In Theorem 1, 2, 3 and Theorem 4, we analyze the security of PMDAC-ABSC scheme with basic *SecretKeyGen* algorithm. Then in Theorem 5 and Theorem 6, we demonstrate the security of *Enhanced_SecretKeyGen* algorithm.

Assume AA^* is the j^*th authority. For all $j \in [N]$, $l_{e,max}, n_{e,max}, l_{s,max}, n_{s,max}$ are the maximum values of $\left\{\ell_{e,j}, n_{e,j}, \ell_{s,j}, n_{s,j}\right\}_{j \in [N]}$. u_{max} is the maximum number of $\left|\widetilde{AA_j}\right|$. T^e is the cost time for one exponentiation in group $\mathbb{G}$ or $\mathbb{G}_T$, and T^p is the cost time for one pairing operation. Suppose that the Hash functions H_1, H_2, H_3 are collision resistant.

A. MESSAGE CONFIDENTIALITY

Based on the security model defined in Definition 10, the proposed scheme guarantees the message confidentiality against static corruption of authorities, which can be reduced to the hardness of the q-PBDHE assumption.

Theorem 1: If an adversary $\mathcal{A}$ can break $(T, q_{sk}, q_{SC}, q_{DS}, \epsilon)$-IND-sEP-CCA2 security of our scheme, then there is an algorithm $\mathcal{B}$ that can solve the q-PBDHE assumption with an advantage $\epsilon' = \frac{1}{2}\epsilon - \frac{q_{DS}}{p}$ in a time

$T' = T + O\left(\left(1 + l_{e,max} n_{e,max} u_{max}\right)(1 + q_{SC}) N + q_{sk} + n_{e,max} + l_{e,max} n_{e,max}^2 + q_{SC} N \left(l + l_{s,max}\right)\right) T^e + O\left((1+q_{SC}) N + q_{DS}\right) T^p$.

Proof: Assume $\mathcal{A}$ can $(T, q_{sk}, q_{SC}, q_{DS}, \epsilon)$ break our scheme, we will construct the algorithm $\mathcal{B}$ as follows: $\mathcal{B}$ is given with the q-PBDHE challenge instance $\vec{y}$. The challenger $\mathcal{C}$ runs $\mathcal{GG}(1^k) \rightarrow (e, p, \mathbb{G}, \mathbb{G}_T)$ to generate the bilinear group and chooses $\ell \in \{0, 1\}$. If $\ell = 0$, $\mathcal{C}$ sends $\left(\vec{y}, \Omega = e(g, g)^{a^{q+1} s}\right)$ to $\mathcal{B}$; otherwise it sends $\left(\vec{y}, \Omega \overset{R}{\leftarrow} \mathbb{G}_T\right)$ to $\mathcal{B}$.

Init. The same as defined in Definition 10. Assume M_e^* is a $\ell_e^* \times n_e^*$ matrix and $n_e^* < q$.

Setup. Let $\pi, \psi, \mu \overset{R}{\leftarrow} \mathbb{Z}_p$, $h = g^{\pi}, g_1 = g^a, \{\ell_0, \ell_1, \ldots, \ell_l\} \overset{R}{\leftarrow} \mathbb{Z}_p, \left\{k_i = g^{\ell_i}\right\}_{i \in [l]}$. $\gamma_1 = g^{\psi} \left(g^{a^q}\right)^{-1}, \gamma_2 = \left(g^{a^q}\right)^{\frac{1}{\pi^*}}$, where $\pi^* = H_2(g^s)$. $\mathcal{B}$ sends $PP = \{g, h, g_1, \gamma_1, \gamma_2, \{k_0, k_1, \ldots, k_l\}\}$ and H_1, H_2, H_3 to $\mathcal{A}$. Set g^{μ} for $\mathcal{A}$ as the public key and assign the identity *uid* to $\mathcal{A}$.

Authority Setup.

1) For the authority AA_j with $j \in I'$, $\mathcal{B}$ chooses $\alpha_j, x_j, y_j, z_j \overset{R}{\leftarrow} \mathbb{Z}_p$ and sets $\Delta_j = e(g, g)^{\alpha_j}, X_j = g^{x_j}, Y_j^1 = g^{y_j}, Y_j^2 = g_1^{y_j}, Z_j^1 = g^{z_j}, Z_j^2 = h^{z_j}$, and $A_{j,a_{j,k}} = g^{\varphi_{j,k}}, B_{j,a_{j,k}} = h^{\varphi_{j,k}} g^{\frac{1}{z_j + a_{j,k}}}$ for each $a_{j,k} \in \widetilde{AA_j}$, where $\varphi_{j,k} \overset{R}{\leftarrow} \mathbb{Z}_p$ and $k \in [n_j]$. The public key of AA_j is $PK_j = \left\{\Delta_j, X_j, Y_j^1, Y_j^2, Z_j^1, Z_j^2, \left\{A_{j,a_{j,k}}, B_{j,a_{j,k}}\right\}_{a_{j,k} \in \widetilde{AA_j}}\right\}$ and secret key is $SK_j = \left\{\alpha_j, x_j, y_j, z_j, \left\{\varphi_{j,k}\right\}_{a_{j,k} \in \widetilde{AA_j}}\right\}$. $\mathcal{B}$ sends $\left\{PK_j, SK_j\right\}$ to $\mathcal{A}$.

2) For the authority AA_j with $j \in I'I^*$, $\mathcal{B}$ does the same as 1) and generates the public key and secret key pair $\left\{PK_j, SK_j\right\}$ for AA_j. B sends PK_j to $\mathcal{A}$.

3) For the authority AA^*, $\mathcal{B}$ chooses $\alpha', \eta, y, z \overset{R}{\leftarrow} \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}, x = \eta a$. Then $\Delta^* = e(g, g)^{\alpha} = e\left(g^a, g^{a^q}\right) e(g, g)^{\alpha'}$. $X^* = g^{\eta a}, Y^{*1} = g^y, Y^{*2} = g_1^y, Z^{*1} = g^z, Z^{*2} = h^z$. Let $\mathcal{X}$ be the set consisting of the indexes $i \in \left[\ell_e^*\right]$ with $\rho_e^*(i) = a_x \in \widetilde{AA^*}$. For the attribute a_x where $\mathcal{X} \neq \emptyset$, $\mathcal{B}$ chooses $\varphi_x \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes $A_{a_x}^* = g^{\varphi_x} \prod_{i \in \mathcal{X}} \prod_{k \in [n_e^*]} g^{\frac{a^k M_e^{*(i,k)}}{b_i}}$, where $M_e^{*(i,k)}$ is the (i,k)th element of M_e^*. $B_{a_x}^* = \left(A_{a_x}^*\right)^{\pi} g^{\frac{1}{z + a_x}}$. If $\mathcal{X} = \emptyset$, $\mathcal{B}$ chooses $\varphi_x \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes $A_{a_x}^* = g^{\varphi_x}, B_{a_x}^* = h^{\varphi_x} g^{\frac{1}{z + a_x}}$. This assignment describes that $A_{a_x}^* = g^{\varphi_x}, B_{a_x}^* = h^{\varphi_x} g^{\frac{1}{z + a_x}}$ for each signing attribute as the signing attributes are different from encryption attributes. $\mathcal{B}$ sends $PK^* = \left\{\Delta^*, X^*, Y^{*1}, Y^{*2}, Z^{*1}, Z^{*2}, \left\{A_{a_x}^*, B_{a_x}^*\right\}_{a_x \in \widetilde{AA^*}}\right\}$ to $\mathcal{A}$.

4) For the authority AA_j with $j \in I^*$ and $AA_j \neq AA^*$, $\mathcal{B}$ does the same as 2) except that for each $a_x \in \widetilde{AA_j}$, $A_{j,a_x} = g^{\varphi_{j,x}} \prod_{i \in X} g^{\frac{a M_e^{*(i,1)}}{b_i}}$ where $\mathcal{X}$ is the set consisting of the indexes $i \in \left[\ell_{e,j}\right]$ with $\rho_{e,j}(i) = a_x$ and $\varphi_{j,x} \overset{R}{\leftarrow} \mathbb{Z}_p$. If $\mathcal{X} = \emptyset$, $A_{j,a_x} = g^{\varphi_{j,x}}$. $\mathcal{B}$ sends PK_j to $\mathcal{A}$.

Phase 1.

SecretKey query $\mathcal{O}^{sk}\left(\widetilde{U}, AA_j\right)$. $\mathcal{A}$ adaptively queries the secret key for a user U with identity *uid* and a set of attributes $\widetilde{U} = \widetilde{U_d} \cup \widetilde{U_s}$ to the authority AA_j. $\widetilde{U_d}$ does not satisfy $\mathcal{R}_e^*$.

1) For the authority $AA_j \neq AA^*$, $\mathcal{B}$ samples $\omega_{uid,j}, \theta_{uid,j}, \delta_{uid,j} \overset{R}{\leftarrow} \mathbb{Z}_p$. Then $\mathcal{B}$ computes

$$CK_{uid,j} = \Big\{ W_{uid,j} = g^{\omega_{uid,j}}, \Theta_{uid,j} = g_1^{\theta_{uid,j}}, \Theta'_{uid,j} = h^{\theta_{uid,j}}, R_{uid,j} = g_1^{\frac{1}{\theta_{uid,j}}}, R'_{uid,j} = h^{\frac{1}{\theta_{uid,j}}}, V_{uid,j} = g^{\delta_{uid,j}}, V'_{uid,j} = h^{\delta_{uid,j}} \Big\},$$

$$DK_{uid,j} = \Big\{ K^d_{uid,j} = g^{\alpha_j} g^{x_j \omega_{uid,j}} g_1^{\theta_{uid,j}} g_1^{\frac{y_j + uid}{\theta_{uid,j}}}, \left\{F^d_{uid,j,a_{j,k}} = A_{j,a_{j,k}}^{\omega_{uid,j}} \left(g^{\mu}\right)^{\varphi_{j,k}}\right\}_{a_{j,k} \in \widetilde{AA_j} \cap \widetilde{U_d}} \Big\}$$

and

$$SGK_{uid,j} = \left\{K^s_{uid,j} = g^{\alpha_j} g_1^{\delta_{uid,j}}, \left\{F^s_{uid,j,a_{j,k}} = A_{j,a_{j,k}}^{\delta_{uid,j}}\right\}_{a_{j,k} \in \widetilde{AA_j} \cap \widetilde{U_s}}\right\}.$$

Send $SK_{uid,j} = \left\{CK_{uid,j}, DK_{uid,j}, SGK_{uid,j}\right\}$ to $\mathcal{A}$.

2) For the authority AA^*, $\mathcal{B}$ chooses $r, t, \theta^*, \delta^{*\prime} \overset{R}{\leftarrow} \mathbb{Z}_p$ and a vector $\vec{f} = \left(f_1, f_2, \ldots, f_{n_e^*}\right) \in \mathbb{Z}_p^{n_e^*}$ such that $f_1 = -1$ and $\vec{f} M_e^{*i} = 0$ for all $\rho_e^*(i) \in \widetilde{U_d} \cap \widetilde{AA^*}$. $\mathcal{B}$ computes $W^* = g^r \prod_{i=1}^{n_e^*} g^{\left(f_i a^{q-i+1}\right)/\eta} = g^{\omega^*}$, namely $\mathcal{B}$ implicitly sets $\omega^* = r + \left(\sum_{i=1}^{n_e^*} f_i a^{q-i+1}\right) \Big/ \eta$. Then $\mathcal{B}$ computes $\Theta^* = g_1^{\theta^*}, \Theta^{*\prime} = h^{\theta^*}, R^* = g_1^{\frac{1}{\theta^*}}, R^{*\prime} = h^{\frac{1}{\theta^*}}, V^* = g^{\delta^{*\prime}} g^{-a^q}, V^{*\prime} = V^{*\pi}$. By this, $\mathcal{B}$ implicitly defines $\delta^* = \delta^{*\prime} - a^q$. The common secret key is $CK^* = \left\{W^*, \Theta^*, \Theta^{*\prime}, R^*, R^{*\prime}, V^*, V^{*\prime}\right\}$. $\mathcal{B}$ computes $K^{d*} = g^{\alpha'} g^{r \eta a} \prod_{i=2}^{n_e^*} g^{f_i a^{q-i+2}} g^{a \theta^*} g^{\frac{a(y+uid)}{\theta^*}}$. For the attribute $a_x \in \widetilde{U_d} \cap \widetilde{AA^*}$ for which there is no i such that $\rho_e^*(i) = a_x$, $\mathcal{B}$ computes $F_{a_x}^{d*} = A_{a_x}^{*\ \omega^*} \left(g^{\mu}\right)^{\varphi_x} = W^{*\varphi_x} \left(g^{\mu}\right)^{\varphi_x}$; otherwise, $\mathcal{B}$ computes

$$F_{a_x}^{d*} = W^{*\varphi_x} \left(g^{\mu}\right)^{\varphi_x} \prod_{i \in X} \prod_{k \in [n_e^*]} \cdot \left(g^{\frac{\eta a^k}{b_i}} \prod_{j=1, k \neq j}^{n_e^*} g^{\frac{f_j a^{q+k-j+1}}{\eta b_i}}\right)^{M_e^{*(i,k)}}.$$

The decryption secret key is $DK^* = \left\{K^{d*}, \left\{F_{a_x}^{d*}\right\}_{a_x \in \widetilde{U_d} \cap \widetilde{AA^*}}\right\}$. $\mathcal{B}$ computes $K^{s*} = g^{\alpha} g_1^{\delta^*} = g^{\alpha'} g^{a \delta^{*\prime}}$. $F_{a_x}^{s*} = A_{a_x}^{*\ \delta^*} = g^{\varphi_x \delta^{*\prime}} \left(g^{a^q}\right)^{-\varphi_x}$. The signing secret key is $SGK^* = \left\{K^{s*}, \left\{F_{a_x}^{s*}\right\}_{a_x \in \widetilde{U_s} \cap \widetilde{AA^*}}\right\}$. $\mathcal{B}$ sends $SK^* = \{CK^*, DK^*, SGK^*\}$ to $\mathcal{A}$.

Signcryption query $\mathcal{O}^{SC}\left(\mathcal{M}, \left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j\in I}\right)$. $\mathcal{B}$ selects a signing attribute set $\widetilde{U_s}$ such that $\mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_s}\right) = 1$ for all $j \in I$ and computes the secret key CK_j, SGK_j and returns the ciphertext $CT \leftarrow$ *Signcryption* $\left(M, PP, \left\{CK_j, SGK_j, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j\in I}\right)$ to $\mathcal{A}$.

DeSigncryption query $\mathcal{O}^{DS}\left(CT, \widetilde{U_d}\right)$. If $\left|tt - \bar{tt}\right| > thre_{tt}$ or for any $j \in I$, $\mathcal{R}_{e,j}\left(\widetilde{AA_j} \cap \widetilde{U_d}\right) = 0$, then $\mathcal{B}$ returns $\perp$. If $C_{j,1} = g^{s_j} = g^s$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ carries out the following steps.

If AA^* is not involved in the generation of CT, or $\widetilde{U_d}$ does not satisfy $\mathcal{R}_e^*$, then $\mathcal{B}$ obtains the secret key from $\mathcal{O}^{sk}\left(\widetilde{U}, AA_j\right)$, and returns the output of *DeSigncryption*(PP, CT, PK, SK) to $\mathcal{A}$. Otherwise, assume $AA_j = AA^*$ is jth authority, $\pi_j = H_2\left(C_{j,1}\right)$. If signature is invalid, $\mathcal{B}$ returns $\perp$. Otherwise, if $\mathcal{R}_e^*\left(\widetilde{U_d}\right) = 1$, compute

$$
\begin{aligned}
& e\left(g^{\alpha'}, g^{s_j}\right) e\left(\frac{C_{j,3}}{C_{j,1}^{\psi}}, g^{a}\right)^{\left(\frac{\pi_j}{\pi^*}-1\right)^{-1}} \\
&= e\left(g^{\alpha'}, g^{s_j}\right) e\left(\frac{\left(\gamma_1\gamma_2^{\pi_j}\right)^{s_j}}{g^{s_j\psi}}, g^{a}\right)^{\left(\frac{\pi_j}{\pi^*}-1\right)^{-1}} \\
&= e\left(g^{\alpha'}, g^{s_j}\right) e\left(\frac{\left(g^{\psi}\left(g^{a^q}\right)^{-1}\left(g^{a^q}\right)^{\frac{\pi_j}{\pi^*}}\right)^{s_j}}{g^{s_j\psi}}, g^{a}\right)^{\left(\frac{\pi_j}{\pi^*}-1\right)^{-1}} \\
&= e\left(g^{\alpha'}, g^{s_j}\right) e\left(g^{a^q}, g^{as_j}\right) = \Delta^{*s_j}.
\end{aligned}
$$

Thus $\mathcal{B}$ can return $\mathcal{M} = \frac{C_0}{\prod_{j\in I}\Delta_j^{s_j}}$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two messages $\mathcal{M}_0, \mathcal{M}_1$ with the same length and signing predicate $\mathcal{R}_{s,I^*} = \left\{\left(M_{s,j}^*, \rho_{s,j}^*\right)\right\}_{j\in I^*}$ to $\mathcal{B}$. Assume M_s^* is a $\ell_s^* \times n_s^*$ matrix, and $\mathcal{R}_s^* = \left(M_s^*, \rho_s^*\right)$ is specified by the authority AA^* with $\mathcal{R}_s^* \in \mathcal{R}_{s,I^*}$. $\mathcal{B}$ chooses $\hat{\ell} \in \{0, 1\}$. $\mathcal{B}$ chooses $s_j, s_j' \xleftarrow{R} \mathbb{Z}_p$ such that $\sum_I^* s_j' = 0$. For $j \in I^*$, $\mathcal{B}$ selects a signing attribute set $\widetilde{U_s}$ satisfying $\mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_s}\right) = 1$.

1) For the authority AA_j with $j \in I^*$ and $AA_j \neq AA^*$, $\mathcal{B}$ selects a vector $\overrightarrow{v_j} = \left(v_{j,1}, v_{j,2}, \ldots, v_{j,\ell_{s,j}}\right) \in \mathbb{Z}_p^{\ell_{s,j}}$ such that $\overrightarrow{v_j} \cdot \mathrm{M}_{s,j} = \vec{1}$, and $\overrightarrow{t_j} = \left(t_{j,1}, t_{j,2}, \ldots, t_{j,\ell_{s,j}}\right) \in \mathbb{Z}_p^{\ell_{s,j}}$ such that $\overrightarrow{t_j} \cdot \mathrm{M}_{s,j} = \vec{0}$. $\mathcal{B}$ implicitly sets $\overrightarrow{\varepsilon_j} = \left(s_j - \frac{s}{|I^*|-1}, \varepsilon_{j,2}, \ldots, \varepsilon_{j,n_{e,j}}\right) \in \mathbb{Z}_p^{n_{e,j}}$, $\overrightarrow{\varepsilon_j'} = \left(s_j' - \frac{s}{|I^*|-1}, \varepsilon_{j,2}', \ldots, \varepsilon_{j,n_{e,j}}'\right) \in \mathbb{Z}_p^{n_{e,j}}$, $\lambda_{j,i} = \mathrm{M}_{e,j}^i \overrightarrow{\varepsilon_j}$, $\lambda_{j,i}' = \mathrm{M}_{e,j}^i \overrightarrow{\varepsilon_j'}$, and $r_i = r_i' - \frac{sb_i}{|I^*|-1}$ for all $i \in \left[\ell_{e,j}\right]$. Then $\mathcal{B}$ computes CT using *Signcryption* algorithm. Note that:

$$
\begin{aligned}
C_{j,4,i} &= g^{a\lambda_{j,i}} A_{j,\rho_{e,j}(i)}^{-r_i} \\
&= g^{as_j\mathcal{M}_{e,j}^{(i,1)}} g^{\frac{sb_i\varphi_{j,\rho_{e,j}(i)}}{|I^*|-1}} A_{\rho_{e,j}(i)}^{-r_i'} \prod_{k=2}^{n_{e,j}} g^{a\varepsilon_k\mathcal{M}_{e,j}^{(i,k)}} \\
&\quad \cdot \prod_{l\in X\setminus i} g^{\frac{sab_i\mathcal{M}_{e,j}^{(i,1)}}{b_l(|I^*|-1)}}. \\
C_{j,5,i} &= g_1^{\lambda_{j,i}'} A_{j,\rho_{e,j}(i)}^{-r_i} \\
&= g^{as_j'M_{e,j}^{(i,1)}} g^{\frac{sb_i\varphi_{j,\rho_{e,j}(i)}}{|I^*|-1}} A_{\rho_{e,j}(i)}^{-r_i'} \prod_{k=2}^{n_{e,j}} g^{a\varepsilon_k' M_{e,j}^{(i,k)}} \\
&\quad \cdot \prod_{l\in X\setminus i} g^{\frac{sab_iM_{e,j}^{(i,1)}}{b_l(|I^*|-1)}}.
\end{aligned}
$$

2) For the authority AA^*, let $\vec{v} = \left(v_1, v_2, \ldots, v_{\ell_s^*}\right) \in \mathbb{Z}_p^{\ell_s^*}$ such that $\vec{v} \cdot M_s^* = \vec{1}$, $\vec{t} = \left(t_1, t_2, \ldots, t_{\ell_s^*}\right) \in \mathbb{Z}_p^{\ell_s^*}$ such that $\vec{t} \cdot M_s^* = \vec{0}$. Re-randomize the signing secret key with $\delta^{*\prime\prime}$ and implicitly set $\delta^* = \delta^{*\prime} - a^q + \delta^{*\prime\prime}$. $C_1 = g^s$, $C_2 = C_1^y$, $C_3 = g^{\psi s}$. Let $\vec{\varepsilon} = \left(s, sa + \varepsilon_2, \ldots, sa^{n_e^*-1} + \varepsilon_{n_e^*}\right) \in \mathbb{Z}_p^{n_e^*}$, $\overrightarrow{\varepsilon_j'} = \left(s' + s, sa + \varepsilon_2', \ldots, sa^{n_e^*-1} + \varepsilon_{n_e^*}'\right) \in \mathbb{Z}_p^{n_e^*}$. Thus $\sum_I^* \overrightarrow{\varepsilon_j'}(0) = \sum_I^* s_j' = 0$. $\left\{r_1', r_2', \ldots, r_{\ell_e^*}'\right\} \xleftarrow{R} \mathbb{Z}_p$. $\mathcal{B}$ implicitly sets $r_i = r_i' + sb_i$ for all $i \in \left[\ell_e^*\right]$ and computes:

$$
\begin{aligned}
C_{4,i} &= g^{a\lambda_i} A_{\rho_e^*(i)}^{*\,-r_i} = A_{\rho_e^*(i)}^{*\,-r_i'} g^{-sb_i\varphi_{\rho_e^*(i)}} \\
&\quad \cdot \prod_{k=2}^{n_e^*} g^{a\varepsilon_k M_e^{*(i,k)}} \prod_{l\in\mathcal{X}\setminus i} \prod_{k\in[n_e^*]} g^{\frac{-sa^k b_i M_e^{*(l,k)}}{b_l}}. \\
C_{5,i} &= A_{\rho_e^*(i)}^{*\,-r_i'} g^{-sb_i\varphi_{\rho_e^*(i)}} g^{as'M_e^{*(i,1)}} \\
&\quad \cdot \prod_{k=2}^{n_e^*} g^{a\varepsilon_k' M_e^{*(i,k)}} \prod_{l\in\mathcal{X}\setminus i} \prod_{k\in[n_e^*]} g^{\frac{-sa^k b_i M_e^{*(l,k)}}{b_l}}. \\
D_i &= g^{r_i} = g^{r_i'} g^{sb_i}. \\
S_{1,i} &= g^{v_i\delta^* + t_i} = g^{v_i\left(\delta^{*\prime} - a^q + \delta^{*\prime\prime}\right) + t_i}, \\
S_2 &= g^{\alpha'} g^{a\left(\delta^{*\prime} + \delta^{*\prime\prime}\right)} \left(g^s\right)^{k_0 + \sum_{i=1}^{\ell} k_i b_i^*} g^{\psi s\beta^*} \left(\prod_{i=1}^{\ell_s^*} S_{1,i}^{\varphi_{\rho_s^*(i)}}\right).
\end{aligned}
$$

Finally, $\mathcal{B}$ computes $C_0 = \mathcal{M}_{\hat{\ell}}\, \Omega e\left(g, g\right)^{\alpha' s} \prod_{i\in I^*\setminus AA^*} e(g, g)^{\alpha_i s}$. $\mathcal{B}$ sends the challenge ciphertext CT^* to $\mathcal{A}$.

Phase 2. Phase 1 is repeated. In this phase, $\mathcal{A}$ cannot issue *DeSigncryption query* with the challenge ciphertext CT^* and attribute set $\widetilde{U_d}$ such that $\mathcal{R}_{e,j}\left(\widetilde{AA_j} \cap \widetilde{U_d}\right) = 1$ for all $j \in I^*$.

Guess. $\mathcal{A}$ outputs his guess $\tilde{\ell}$ on $\hat{\ell}$. If $\tilde{\ell} = \hat{\ell}$, $\mathcal{B}$ outputs 0 and guess that $\Omega = e\left(g, g\right)^{a^{q+1}s}$; otherwise, $\mathcal{B}$ outputs 1 to indicate that Ω is a random element in $\mathbb{G}_T$.

If $\mathcal{A}$ issues *DeSigncryption query* with the ciphertext satisfying $C_{j,1} = g^{s_j} = g^s$, then the simulation aborts. The probability of this type of event is at most $\frac{q_{DS}}{p}$. If $\hat{\ell} = 0$, $\Omega = e\left(g, g\right)^{a^{q+1}s}$ and $\mathcal{B}$ does not abort, then CT^* is a valid ciphertext of $\mathcal{M}_0$. Thus, in this case, the advantage of $\mathcal{A}$ is $Pr\left[\tilde{\ell} = \hat{\ell} | \hat{\ell} = 0\right] > \frac{1}{2} + \epsilon - \frac{q_{DS}}{p}$. If $\Omega$is a random element in $\mathbb{G}_T$, then $\mathcal{A}$ has no information about $\mathcal{M}_{\hat{\ell}}$, namely the advantage in this case is $Pr\left[\tilde{\ell} \neq \hat{\ell} | \hat{\ell} = 1\right] = \frac{1}{2}$. Therefore, the advantage of $\mathcal{B}$ which can break the q-PBDHE assumption is at least $\frac{1}{2}\epsilon - \frac{q_{DS}}{p}$. The runtime of $\mathcal{B}$ is at most $T' = T + O\left(\left(1 + l_{e,max}n_{e,max}u_{max}\right)\left(1 + q_{SC}\right)N + q_{sk} + n_{e,max} + l_{e,max}n_{e,max}^2 + q_{SC}N\left(l + l_{s,max}\right)\right)T^e + O\left(\left(1 + q_{SC}\right)N + q_{DS}\right)T^p$.

B. CIPHERTEXT UNFORGEABILITY

The ciphertext unforgeability of our scheme is formulated through existential unforgeability under selectively signing predicates and adaptive chosen message attack model as defined in Definition 11.

Theorem 2: If an adversary $\mathcal{A}$ can break $(T, q_{sk}, q_{SC}, q_{DS}, \epsilon)$-EUF-sSP-CMA security of our scheme, then there is an algorithm $\mathcal{B}$ that can solve the q-PBDHE assumption with an advantage $\epsilon' = \frac{\epsilon}{8N(l+1)q_{SC}}$ in a time $T' = T + O\left(\left(1 + l_{s,max}n_{s,max}u_{max}\right)(1 + q_{SC})N + q_{sk} + n_{s,max} + l_{s,max}n_{s,max}^2 + (1 + q_{SC}N)\left(l + l_{s,max}\right)\right)T^e + O\left((1 + q_{SC})N + q_{DS}\right)T^p$.

Proof: Assume $\mathcal{A}$ can $(T, q_{sk}, q_{SC}, q_{DS}, \epsilon)$ break our basic scheme, we will construct the algorithm $\mathcal{B}$ as follows: $\mathcal{B}$ is given with the q-PBDHE challenge instance $\vec{y}$. The challenger $\mathcal{C}$ runs $\mathcal{GG}(1^k) \rightarrow (e, p, \mathbb{G}, \mathbb{G}_T)$ to generate the bilinear group and chooses $\ell \in \{0, 1\}$. If $\ell = 0$, $\mathcal{C}$ sends $\left(\vec{y}, \Omega = e(g, g)^{a^{q+1}s}\right)$ to $\mathcal{B}$; otherwise it sends $\left(\vec{y}, \Omega \xleftarrow{R} \mathbb{G}_T\right)$ to $\mathcal{B}$.

Init. The same as defined in Definition 11. Assume M_s^* is a $\ell_s^* \times n_s^*$ matrix and $n_s^* < q$.

Setup. Sample $\pi, \varepsilon_1, \varepsilon_2, \mu \xleftarrow{R} \mathbb{Z}_p$ and set $h = g^{\pi}, g_1 = g^a$. $\left\{k_0, k_1, \ldots, k_l\right\} \xleftarrow{R} \mathbb{Z}_p$. Assume $\varpi = 4q_{SC}$ and $\varpi(l+1) < p$. $\mathcal{B}$ chooses $m \xleftarrow{R} \{0, 1, \ldots, l\}, \varrho_0, \varrho_1, \ldots, \varrho_l \xleftarrow{R} \{0, 1, \ldots, \varpi - 1\}, \sigma_1, \sigma_2, k_0, k_1, \ldots, k_l \xleftarrow{R} \mathbb{Z}_p^*$. Set $k_0 = \left(g^{a^q}\right)^{\varrho_0 - \varpi m} g^{k_0}$ and $\left\{k_i = \left(g^{a^q}\right)^{\varrho_i} g^{k_i}\right\}_{i \in [l]}$. $\gamma_1 = g^{\sigma_1}, \gamma_2 = g^{\sigma_2}$. $\mathcal{B}$ defines two functions $L_1\left(\vec{b}\right) = p - \varpi m + \varrho_0 + \sum_{i=1}^{l} b_i\varrho_i$ and $L_2\left(\vec{b}\right) = k_0 + \sum_{i=1}^{l} b_i k_i$ for each $\vec{b} = (b_1, b_2, \ldots, b_l) \in \{0, 1\}^l$. Thus $k_0 \prod_{i=1}^{l} k_i^{b_i} = \left(g^{a^q}\right)^{L_1\left(\vec{b}\right)} g^{L_2\left(\vec{b}\right)}$. Let $L\left(\vec{b}\right) = \begin{cases} 0, \varrho_0 + \sum_{i=1}^{l} b_i\varrho_i = 0 \ mod \ \varpi \\ 1, otherwise \end{cases}$. Then $L\left(\vec{b}\right) = 1$ implies $L_1\left(\vec{b}\right) \neq 0 \ mod \ p$. $\mathcal{B}$ sends $PP = \{g, h, g_1, \gamma_1, \gamma_2, \{k_0, k_1, \ldots, k_l\}\}$ and H_1, H_2, H_3 to $\mathcal{A}$. Set g^{μ} for $\mathcal{A}$ as the public key and assign the identity *uid* to $\mathcal{A}$.

Authority Setup.

1) For the authority AA^*, $\mathcal{B}$ chooses $\alpha', x', y, z \xleftarrow{R} \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}, x = x' - a$. Then $\Delta^* = e(g, g)^{\alpha} = e\left(g^a, g^{a^q}\right)e(g, g)^{\alpha'}$. $X^* = g^{x'}g^{-a}, Y^{*1} = g^y, Y^{*2} = g_1^y, Z^{*1} = g^z, Z^{*2} = h^z$. If there exists $i \in \left[\ell_s^*\right]$ with $\rho_s^*(i) = a_x \in \widetilde{AA^*}$, $\mathcal{B}$ chooses $\varphi_x \xleftarrow{R} \mathbb{Z}_p$ and computes $A_{a_x}^* = g^{\varphi_x} \prod_{k \in [n_s^*]} g^{-a^k M_s^{*(i,k)}}$, where $M_s^{*(i,k)}$ is the (i, k)th element of M_s^*. $B_{a_x}^* = \left(A_x^*\right)^{\pi} g^{\frac{1}{z + a_x}}$. Otherwise, $\mathcal{B}$ chooses $\varphi_x \xleftarrow{R} \mathbb{Z}_p$ and computes $A_{a_x}^* = g^{\varphi_x}, B_{a_x}^* = h^{\varphi_x} g^{\frac{1}{z+a_x}}$. This assignment describes that $A_{a_x}^* = g^{\varphi_x}, B_{a_x}^* = h^{\varphi_x} g^{\frac{1}{z+a_x}}$ for each decryption attribute. $\mathcal{B}$ sends $PK^* = \left\{\Delta^*, X^*, Y^{*1}, Y^{*2}, Z^{*1}, Z^{*2}, \left\{A_{a_x}^*, B_{a_x}^*\right\}_{a_x \in \widetilde{AA^*}}\right\}$ to $\mathcal{A}$.

2) Otheriwise, $\mathcal{B}$ performs the same as in Theorem 1.

SecretKey query $\mathcal{O}^{sk}\left(\widetilde{U}, AA_j\right)$. $\mathcal{A}$ adaptively queries the secret key for a user U with identity *uid* and a set of attributes $\widetilde{U} = \widetilde{U_d} \cup \widetilde{U_s}$ to the authority AA_j. $\widetilde{U_s}$ does not satisfy $\mathcal{R}_s^*$.

1) For the authority $AA_j \neq AA^*$, $\mathcal{B}$ performs the same as in Theorem 1.

2) For the authority AA^*, $\mathcal{B}$ chooses $\omega', r, t, \theta^*, \delta^{*\prime} \xleftarrow{R} \mathbb{Z}_p$ and a vector $\vec{f} = \left(f_1, f_2, \ldots, f_{n_s^*}\right) \in \mathbb{Z}_p^{n_s^*}$ such that $f_1 = -1$ and $\vec{f}M_s^{*i} = 0$ for all $\rho_s^*(i) \in \widetilde{U_s} \cap \widetilde{AA^*}$. $\mathcal{B}$ implicitly sets $\delta^* = \delta^{*\prime} + \sum_{i=1}^{n_s^*} f_i a^{q-i+1}$. Then $\mathcal{B}$ computes $W^* = g^{\omega'}g^{a^q}, \Theta^* = g_1^{\theta^*}, \Theta^{*\prime} = h^{\theta^*}, R^* = g_1^{\frac{1}{\theta^*}}, R^{*\prime} = h^{\frac{1}{\theta^*}}, V^* = g^{\delta^{*\prime}} \prod_{i=1}^{n_s^*} g^{f_i a^{q-i+1}}, V^{*\prime} = V^{*\pi}$. The common secret key is $CK^* = \left\{W^*, \Theta^*, \Theta^{*\prime}, R^*, R^{*\prime}, V^*, V^{*\prime}\right\}$.

$\mathcal{B}$ computes $K^{d*} = g^{\alpha'}(g^a)^{-\omega'}\left(g^{a^q}\right)^{x'} g_1^{\theta^*} g_1^{\frac{y + uid}{\theta^*}}$. For the attribute $a_x \in \widetilde{U_d} \cap \widetilde{AA^*}$, $F_{a_x}^{d*} = A_{a_x}^{*\ \omega^*}(g^{\mu})^{\varphi_x} = (W^*)^{\varphi_x}(g^{\mu})^{\varphi_x}$. The decryption secret key is $DK^* = \left\{K^{d*}, \left\{F_{a_x}^{d*}\right\}_{a_x \in \widetilde{U_d} \cap \widetilde{AA^*}}\right\}$.

$K^{s*} = g^{\alpha}g_1^{\delta^*} = g^{\alpha'}(g^a)^{\delta^{*\prime} + \sum_{i=2}^{n_s^*} f_i a^{q-i+2}}$. For which there is no i such that $\rho_s^*(i) = a_x$, $F_{a_x}^{s*} = V^{*\varphi_x}$. Otherwise, $F_{a_x}^{s*} = V^{*\varphi_x} \prod_{k \in [n_s^*]} \left(g^{-\delta^{*\prime}a^k} \prod_{j=1, k \neq j}^{n_s^*} g^{-f_j a^{q+k-j+1}}\right)^{M_s^{*(i,k)}}$.

The signing secret key is $SGK^* = \left\{K^{s*}, \left\{F_{a_x}^{s*}\right\}_{a_x \in \widetilde{U_s} \cap \widetilde{AA^*}}\right\}$. $\mathcal{B}$ sends $SK^* = \{CK^*, DK^*, SGK^*\}$ to $\mathcal{A}$.

Signcryption query $\mathcal{O}^{SC}\left(\mathcal{M}, \left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j \in I}\right)$. $\mathcal{B}$ selects a signing attribute set $\widetilde{U_s}$ such that $\mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_s}\right) = 1$ for all $j \in I$. If $AA^* \notin I$, $\mathcal{B}$ issues $\mathcal{O}^{sk}$ and returns the ciphertext $CT \leftarrow Signcryption\left(\mathcal{M}, PP, \left\{CK_j, SGK_j, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j \in I}\right)$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ first computes $s_j' \xleftarrow{R} \mathbb{Z}_p$ for each $j \in I$ such that $\sum_I s_j' = 0$. Then $\mathcal{B}$ computes

$$\left\{C_0, \left\{\begin{matrix} C_{j,1}, C_{j,2}, C_{j,3}, \left\{C_{j,4,i}, C_{j,5,i}, D_{j,i}\right\}_{i \in [\ell_{e,j}]}, \\ \left\{S_{1,j,i}\right\}_{i \in [\ell_{s,j}]}, S_{2,j}, \mathcal{R}_{s,j}, \mathcal{R}_{e,j} \end{matrix}\right\}_{j \in I \setminus j^*}\right\}.$$

For AA^*, $s_j' = s'$, $\mathcal{B}$ performs as follows:

1) It first computes a vector $\vec{v} = \left(v_1, v_2, \ldots, v_{\ell_s^*}\right) \in \mathbb{Z}_p^{\ell_s^*}$ such that $\vec{v} \cdot M_s^* = \vec{1}$ and chooses $\vec{t} = \left(t_1, t_2, \ldots, t_{\ell_s^*}\right) \in \mathbb{Z}_p^{\ell_s^*}$ such that $\vec{t} \cdot M_s^* = \vec{0}$.

2) $\mathcal{B}$ randomly chooses $\delta \xleftarrow{R} \mathbb{Z}_p^*$ and computes $\left\{S_{1,i} = g^{v_i\delta + t_i}\right\}_{i \in [\ell_s^*]}$.

3) Assume $H_1\left(\prod_{i \in [\ell_s^*]} S_{1,i}, tt, \mathcal{R}_s^*, \mathcal{R}_e^*\right) = (b_1, b_2, \ldots, b_l) = \vec{b} \in \{0, 1\}^l$. If $L\left(\vec{b}\right) = 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ implicitly sets $s = s_x - \frac{a}{L_1\left(\vec{b}\right)}$ where $s_x \xleftarrow{R} \mathbb{Z}_p^*$. Then $C_1 = g^s = g^{s_x}(g^a)^{-\frac{1}{L_1\left(\vec{b}\right)}}$, $C_2 = C_1^y$, $C_3 = (g^{\sigma_1}g^{\pi\sigma_2})^s$, where $\pi = H_2(C_1)$.

4) $\mathcal{B}$ chooses $\left\{r_1, r_2, \ldots, r_{\ell_e^*}\right\} \xleftarrow{R} \mathbb{Z}_p$, $\vec{\varepsilon} = \left(s_x - \frac{a}{L_1\left(\vec{b}\right)}, \varepsilon_2, \ldots, \varepsilon_{n_e^*}\right) \in \mathbb{Z}_p^{n_e^*}$, $\overrightarrow{\varepsilon'} = \left(s', \varepsilon_2', \ldots, \varepsilon_{n_e^*}'\right) \in$

$\mathbb{Z}_p^{n_e^*}$, $\lambda_i = M_e^{*i}\vec{\varepsilon}$, $\lambda_i' = M_e^{*i}\overrightarrow{\varepsilon'}$. Then for $i \in [\ell_e^*]$, $\mathcal{B}$ computes

$$C_{4,i} = g^{x\lambda_i} A_{\rho_e^*(i)}^{*\,-r_i} = g^{(x'-a)\left(\left(s_x - \frac{a}{L_1(\vec{b})}\right)M_e^{*(i,1)} + \sum_{j=1}^{n_e^*} \varepsilon_j M_e^{*(i,j)}\right)} g^{-\varphi_{\rho_e^*(i)} r_i},$$

$$C_{5,i} = g_1^{\lambda_i'} A_{\rho_e^*(i)}^{*\,-r_i} = g^{a\left(s' M_e^{*(i,1)} + \sum_{j=1}^{n_e^*} \varepsilon_j' M_e^{*(i,j)}\right)} g^{-\varphi_{\rho_e^*(i)} r_i},$$

$$D_i = g^{r_i}.$$

5) $\mathcal{B}$ computes $H_3\left(C_1, C_2, C_3, \prod_{i\in[\ell_e^*]} C_{4,i}, \prod_{i\in[\ell_e^*]} C_{5,i}, \mathcal{R}_s^*, \mathcal{R}_e^*\right) = \beta$,

$$\begin{aligned} S_2 &= K^{s*}\left(k_0 \prod_{i=1}^{l} k_i^{b_i}\right)^s C_3^{\beta}\left(\prod_{i=1}^{\ell_s^*}\left(F_{\rho_s^*(i)}^{s*}\right)^{v_i}\left(A_{\rho_s^*(i)}^{*}\right)^{t_i}\right) \\ &= g^{\alpha'+a^{q+1}}\left(g^a\right)^{\delta} C_3^{\beta}\left(g^{a^q L_1(\vec{b}) + L_2(\vec{b})}\right)^{s_x - \frac{a}{L_1(\vec{b})}} \cdot \left(\prod_{i=1}^{\ell_s^*}\left(A_{\rho_s^*(i)}^{*}\right)^{v_i\delta + t_i}\right) \\ &= g^{\alpha'}\left(g^a\right)^{\delta} C_3^{\beta}\left(\left(g^{a^q}\right)^{L_1(\vec{b})} g^{L_2(\vec{b})}\right)^{s_x} g^{\frac{-aL_2(\vec{b})}{L_1(\vec{b})}} \cdot \left(\prod_{i=1}^{\ell_s^*}\left(A_{\rho_s^*(i)}^{*}\right)^{v_i\delta + t_i}\right). \end{aligned}$$

Finally, $\mathcal{B}$ sends CT to $\mathcal{A}$.

DeSigncryption query $\mathcal{O}^{DS}\left(CT, \widetilde{U_d}\right)$. If $|tt - \bar{tt}| > thre_{tt}$ or for any $j \in I$, $\mathcal{R}_{e,j}\left(\widetilde{AA_j} \cap \widetilde{U_d}\right) = 0$, then $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ issues the *SecretKey query* and returns the output of *DeSigncryption* $(PP, CT, PK_{uid}, SK_{uid})$ to $\mathcal{A}$.

Forgery. $\mathcal{A}$ submits a valid ciphertext CT^*. If *DeSigncryption* $(PP, CT^*, PK, SK) \rightarrow \mathcal{M}$ and $\mathcal{A}$ has never issued $\mathcal{O}^{SC}\left(M, \left\{\mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right\}_{j\in I^*}\right)$, $\mathcal{B}$ performs as follows.

1) Assume the components of CT^* corresponding to AA^* is $\left\{C_1, C_2, C_3, \left\{C_{4,i}, C_{5,i}, D_i\right\}_{i\in[\ell_e^*]}, \left\{S_{1,i}\right\}_{i\in[\ell_s^*]}, S_2\right\}$. $\mathcal{B}$ computes $H_1\left(\prod_{i\in[\ell_s^*]} S_{1,i}, tt, \mathcal{R}_s^*, \mathcal{R}_e^*\right) = (b_1, b_2, \ldots, b_l) = \vec{b} \in \{0,1\}^l$. If $\varpi m \neq \varrho_0 + \sum_{i=1}^{l} b_i \varrho_i$, $\mathcal{B}$ aborts. Otherwise, $L_1\left(\vec{b}\right) = 0 \bmod p$.

2) If CT^* is a valid ciphertext, then $H_3\left(C_1, C_2, C_3, \prod_{i\in[\ell_e^*]} C_{4,i}, \prod_{i\in[\ell_e^*]} C_{5,i}, \mathcal{R}_s^*, \mathcal{R}_e^*\right) = \beta$ and $\pi = H_2(C_1)$. Then

$$\begin{aligned} S_2 &= K^{s*}\left(k_0 \prod_{i=1}^{l} k_i^{b_i}\right)^s C_3^{\beta}\left(\prod_{i=1}^{\ell_s^*}\left(F_{\rho_s^*(i)}^{s*}\right)^{v_i}\left(A_{\rho_s^*(i)}^{*}\right)^{t_i}\right) \\ &= g^{\alpha'} g^{a^{q+1}}\left(g^a\right)^{\delta} C_1^{L_2(\vec{b}) + \beta(\sigma_1 + \pi\sigma_2)} \prod_{i=1}^{\ell_s^*} S_{1,i}^{\varphi_{\rho_s^*(i)}} \cdot \prod_{i=1}^{\ell_s^*}\left(\prod_{k\in[n_s^*]} g^{-a^k M_s^{*(i,k)}}\right)^{v_i\delta + t_i} \\ &= g^{\alpha'} g^{a^{q+1}}\left(g^a\right)^{\delta} C_1^{L_2(\vec{b}) + \beta(\sigma_1 + \pi\sigma_2)} \prod_{i=1}^{\ell_s^*} S_{1,i}^{\varphi_{\rho_s^*(i)}} \left(g^a\right)^{-\delta} \\ &= g^{\alpha'} g^{a^{q+1}} C_1^{L_2(\vec{b}) + \beta(\sigma_1 + \pi\sigma_2)} \prod_{i=1}^{\ell_s^*} S_{1,i}^{\varphi_{\rho_s^*(i)}}, \end{aligned}$$

where $\vec{v}\cdot M_s^* = \vec{1}$, $\vec{t}\cdot M_s^* = \vec{0}$ and $\sum_{i\in[\ell_s^*]}\sum_{k\in[n_s^*]} -a^k M_s^{*(i,k)}(v_i\delta + t_i) = -a\delta$.

Thus, $\mathcal{B}$ can calculate $g^{a^{q+1}} = \frac{S_2}{g^{\alpha'} C_1^{L_2(\vec{b}) + \beta(\sigma_1+\pi\sigma_2)} \prod_{i=1}^{\ell_s^*} S_{1,i}^{\varphi_{\rho_s^*(i)}}}$ and then break the q-PBDHE assumption by computing $e\left(g^{a^{q+1}}, g^s\right)$. Let E_1 be the event that $j^* \notin I$ in the forgery phase, E_2 be the event that $L\left(\vec{b}\right) = 0$ in some *Signcryption query* and E_3 be the event that $\varpi m \neq \varrho_0 + \sum_{i=1}^{l} b_i \varrho_i$ in the forgery phase. Then we have $Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] = Pr[\neg E_1] Pr[\neg E_2 \wedge \neg E_3] = \frac{1}{N}\frac{1}{(l+1)\varpi}\left(1 - \frac{2q_{SC}}{\varpi}\right)$. If $\varpi = 4q_{SC}$, then $Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] = \frac{1}{8N(l+1)q_{SC}}$. Thus the advantage of $\mathcal{B}$ solving the q-PBDHE assumption is at least $\mathcal{A}dv_{\mathcal{B}} \geq \frac{\epsilon}{8N(l+1)q_{SC}}$. The runtime of $\mathcal{B}$ is at most $T' = T + O\left(\left(1 + l_{s,max} n_{s,max} u_{max}\right)(1+q_{SC})N + q_{sk} + n_{s,max} + l_{s,max} n_{s,max}^2 + (1+q_{SC}N)\left(l + l_{s,max}\right)\right) T^e + O\left((1+q_{SC})N + q_{DS}\right) T^p$.

C. SIGNCRYPTOR PRIVACY

Based on the security model defined in Definition 12, our scheme guarantees signcryptor privacy and ensures that one cannot guess the set of signing attributes of a signcryptor used to sign a plaintext.

Theorem 3: Our scheme guarantees the signcryptor privacy.

Proof: The challenger sends $PP, PK, \left\{PK_j, SK_j\right\}_I$ to the adversary $\mathcal{A}$. Then $\mathcal{A}$ outputs two signing attribute sets $\widetilde{U_s^0}, \widetilde{U_s^1}$ satisfying $\forall j \in I, \mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_s^0}\right) = 1 = \mathcal{R}_{s,j}\left(\widetilde{AA_j} \cap \widetilde{U_s^1}\right)$. The challenger selects $\ell \xleftarrow{R} \{0,1\}$ and computes CT_ℓ with the secret key $SK_{do,j}^{\ell}$. Note that both the challenger and $\mathcal{A}$ can compute $SK_{do,j}^{\ell}$ for $\widetilde{U_s^\ell}$. Specifically, $V_{do,j}^{\ell} = g^{\delta_{do,j}^{\ell}}$, $K_{do,j}^{s\ell} = g^{\alpha_j} g_1^{\delta_{do,j}^{\ell}}$, $F_{do,j,a_{j,k}}^{s\ell} = A_{j,a_{j,k}}^{\delta_{do,j}^{\ell}}$, where $\delta_{do,j}^{\ell} \xleftarrow{R} \mathbb{Z}_p^*$.

If the challenger uses $SK_{do,j}^{0}$, then $C_{j,1}^0 = g^{s_j^0}$, $C_{j,2}^0 = \left(Y_j^1\right)^{s_j^0}$, $C_{j,3}^0 = \left(\gamma_1\gamma_2^{\pi_j}\right)^{s_j^0}$ where $\pi_j = H_2\left(C_{j,1}^0\right)$.

$$\left\{C_{j,4,i}^0 = g^{x_j\lambda_{j,i}^0} A_{j,\rho_{e,j}(i)}^{-r_{j,i}^0}, C_{j,5,i}^0 = g_1^{\lambda_{j,i}^{\prime 0}} A_{j,\rho_{e,j}(i)}^{-r_{j,i}^0}, D_{j,i}^0 = g^{r_{j,i}^0}\right\}_{i\in[\ell_{e,j}]},$$

$$\left\{S_{1,j,i}^0 = g^{v_{j,i}^0 \delta_{do,j}^{0\prime\prime} + t_{j,i}^0}\right\}_{i\in[\ell_{s,j}]}.$$

$$H_1\left(\prod_{i\in[\ell_{s,j}]} S_{1,j,i}^0, tt, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\right) = \left(b_{j,1}, b_{j,2}, \ldots, b_{j,l}\right) \in \{0,1\}^l.$$

$$H_3\left(C_{j,1}^0, C_{j,2}^0, C_{j,3}^0, \prod_{i\in[\ell_{e,j}]} C_{j,4,i}^0,\right.$$

$$\prod_{i\in[\ell_{e,j}]} C^0_{j,5,i}, \mathcal{R}_{s,j}, \mathcal{R}_{e,j}\Big) = \beta_j.$$

$$S^0_{2,j} = g^{\alpha_j} g_1^{\delta^{0''}_{do,j}} \left(k_0 \prod_{i=1}^{l} k_i^{b_{j,i}}\right)^{s^0_j} \cdot C^{0\ \beta_j}_{j,3} \left(\prod_{i=1}^{\ell_{s,j}} \left(A_{j,\rho_{s,j}(i)}\right)^{v^0_{j,i}\delta^{0''}_{do,j}+t^0_{j,i}}\right),$$

where $\delta^{0''}_{do,j} = \delta^0_{do,j} + \delta^{0'}_{do,j}$ and $\delta^{0'}_{do,j} \overset{R}{\leftarrow} \mathbb{Z}^*_p$.

If the challenger uses $SK^1_{do,j}$, and sets $s^0_j = s^1_j, s^{'0}_j = s^{'1}_j, r^0_{j,i} = r^1_{j,i}, \delta^{1'}_{do,j} = \delta^{0''}_{do,j} - \delta^1_{do,j}$, then $\lambda^0_{j,i} = \lambda^1_{j,i}, \lambda^{'0}_{j,i} = \lambda^{'1}_{j,i}$, $\delta^{0''}_{do,j} = \delta^{1''}_{do,j} = \delta''_{do,j}$. Thus $C^0_{j,1} = C^1_{j,1}, C^0_{j,2} = C^1_{j,2}, C^0_{j,3} = C^1_{j,3}, C^0_{j,4,i} = C^1_{j,4,i}, C^0_{j,5,i} = C^1_{j,5,i}, D^0_{j,i} = D^1_{j,i}$. The challenger sets $\overrightarrow{v^1_j} \cdot M_{s,j} = \vec{1}$ and sets $t^1_{j,i} = \left(v^0_{j,i} - v^1_{j,i}\right)\delta^{0''}_{do,j} + t^0_{j,i}$. Then $\overrightarrow{t^1_j} \cdot M_{s,j} = \vec{0}$ and $v^0_{j,i}\delta^{0''}_{do,j} + t^0_{j,i} = v^1_{j,i}\delta^{0''}_{do,j} + t^1_{j,i}$. Hence $S^0_{1,j,i} = S^1_{1,j,i}, S^0_{2,j} = S^1_{2,j}$, and $CT_0 = CT_1$. Similarly, if the challenger firstly uses $SK^1_{do,j}$ to generate CT_1, then it can generate CT_0 with $SK^0_{do,j}$ and $CT_1 = CT_0$. Therefore, $\mathcal{A}$ can only outputs a random guess ℓ' and the probability is at most $\frac{1}{2}$.

D. COLLUSION RESISTANCE

In our scheme, the secret keys of each user are associated the random elements $\omega_{uid,j}, \theta_{uid,j}, \delta_{uid,j}$ chosen by AA_j, and the g^μ picked by CA where μ is difficult for each user, authority and cloud server to compute or learn. Therefore, the colluders cannot selectively replace or convert the components of the secret key under the discrete logarithm assumption.

Theorem 4: Our scheme is secure against the collusion attack of unauthorized users and cloud server.

Proof: For the signcryptor, the Theorem 2 guarantees that no colluders such as data user or cloud server can generate the signature by combining their information if they are individually unauthorized to sign the plaintext. Otherwise, the colluders can build an adversary and output a forgery to win the game in Definition 11.

For the designcryptor, beside the Theorem 1, recall that the message $\mathcal{M}$ is blinded by $\prod_{j\in I} \Delta_j^{s_j} = \prod_{j\in I} e(g,g)^{\alpha_j s_j}$. To recover such a message, the colluders have to construct $e(g,g)^{\alpha_j s_j}$, and thus have to cancel the redundant element $e(g,g)^{\omega_{uid,j} x_j s_j}$ in executing $e\left(K^d_{uid,j}, C_{j,1}\right)$ for all AA_j, $j \in I$. Due to BDH assumption and colluders' blindness of secret values $\omega_{uid,j}, x_j$ secretly chosen by AA_j, the only way to cancel $e(g,g)^{\omega_{uid,j} x_j s_j}$ is to compute $\prod_{i=1}^{l_{e,j}} \left(e\left(W_i, C_{j,4,i}\right) e\left(F_i, D_{j,i}\right) e\left(P_i, C_{j,5,i}\right)\right)$, which will additionally introduce $e(g,g_1)^{\mu_j s'_j}$ for each AA_j. If the secret keys belong to the same designcryptor, which means $\mu_j = \mu$ holds for all $j \in I$, then from $\sum_I s'_j = 0$ we have $\prod_{j\in I} e(g,g_1)^{\mu s'_j} = 1$. Otherwise, $\prod_{j\in I} e(g,g_1)^{\mu_j s'_j}$ cannot be cancelled since μ_j and s'_j are secretly chosen and kept secret in the system.

Therefore, the colluding users and cloud server cannot sign or decrypt the data, even though their combined attribute set satisfies the access policy.

E. SECURE ANALYSIS OF Enhanced_SecretKeyGen

Enhanced_SecretKeyGen algorithm is secure against malicious user means that by executing *Enhanced_SecretKeyGen* with a honest authority, the malicious user cannot know anything which he/she cannot know by executing the basic *SecretKeyGen* algorithm.

Theorem 5: Enhanced_SecretKeyGen is secure against malicious user U.

Proof: The simulator *Sim* simulates the communication between U and AA_j and uses the outputs of U in *Enhanced_SecretKeyGen* algorithm to construct the outputs of AA_j in *Enhanced_SecretKeyGen* and *SecretKeyGen*, respectively. When U outputs $\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6, \Theta_7, \Theta_8$ and $\left\{\Psi^1_{a_x}, \Psi^{2d}_{a_x}, \Psi^{2s}_{a_x}, \Psi^{3d}_{a_x}, \Psi^{3s}_{a_x}, \Psi^{4d}_{a_x}, \Psi^{4s}_{a_x}, \Psi^5_{a_x}\right\}_{a_x\in\widetilde{AA_j}\cap\widetilde{U}}$, *Sim* checks the proof and obtains $a_1, a_2, d_1, d_3, D_1, D_2, a_x$ by rewind technology. *Sim* sends $\left\{\Psi^{3d\ \frac{1}{D_1}}_{a_x} = A_{a_x}\right\}_{a_x\in\widetilde{AA_j}\cap\widetilde{U_d}}$ and $\left\{\Psi^{3s\ \frac{1}{D_2}}_{a_x} = A_{a_x}\right\}_{a_x\in\widetilde{AA_j}\cap\widetilde{U_s}}$ to a honest user U_h. U_h then executes *SecretKeyGen* with AA_j and obtains the secret key

$$SK = \left\{W_{uid,j}, \Theta_{uid,j}, \Theta'_{uid,j}, R_{uid,j}, R'_{uid,j}, V_{uid,j}, V'_{uid,j}, K^d_{uid,j}, K^s_{uid,j}, \left\{F^d_{a_x}, F^s_{a_x}\right\}_{a_x\in\widetilde{AA_j}\cap\widetilde{U}}\right\}$$

with which *Sim* can generate the outputs of AA_j in *Enhanced_SecretKeyGen* by computing $\Upsilon_1 = \Theta^{a_2}_{uid,j}$, $\Upsilon_2 = R^{\frac{1}{a_2}}_{uid,j}$, $\Upsilon_3 = \Theta'^{\ a_2}_{uid,j}$, $\Upsilon_4 = R'^{\ \frac{1}{a_2}}_{uid,j}$, $\Upsilon_5 = \left(W_{uid,j} g^\mu\right)^{\frac{1}{d_1}}$, $\Upsilon_6 = V^{\frac{1}{d_3}}_{uid,j}$, $\Upsilon_7 = V'^{\ \frac{1}{d_3}}_{uid,j}$, $K^{d*}_{uid,j} = \frac{K^d_{uid,j}\Upsilon_4^{a_1 a_2}}{\Upsilon_1^{\frac{1}{a_2}}}$, $K^{s*}_{uid,j} = K^s_{uid,j}\Upsilon_7^{a_1}$, $\Phi^d_{a_x} = F^{d\ \frac{D_1}{d_1}}_{a_x}$, $\Phi^s_{a_x} = F^{s\ \frac{D_2}{d_3}}_{a_x}$. Since a_1, a_2, d_1, d_2, d_3 are randomly chosen from $\mathbb{Z}_p$, the outputs of AA_j are computationally indistinguishable. Hence, there is no efficient distinguisher D can distinguish the real key generation protocol from the ideal key generation protocol.

Enhanced_SecretKeyGen algorithm is secure against malicious authority means that the malicious authority cannot know anything about the user's attributes when executing *Enhanced_SecretKeyGen*algorithm.

Theorem 6: Enhanced_SecretKeyGen is secure against malicious authority AA_j.

Proof:

Init. AA_j submits $\left(PK_{U_0}, \widetilde{U_0}\right)$ and $\left(PK_{U_1}, \widetilde{U_1}\right)$ to the challenger $\mathcal{C}$.

Challenge. Give the malicious authority AA_j two black-box oracles $\mathcal{O}_0$ and $\mathcal{O}_1$. $\mathcal{C}$ chooses a bit $b \in \{0,1\}$, and executes *Enhanced_SecretKeyGen* $\left(U_b \leftrightarrow AA_j\right)$ and *Enhanced_SecretKeyGen* $\left(U_{1-b} \leftrightarrow AA_j\right)$. $\mathcal{C}$ returns the outputs $SK_{b,j}$ and $SK_{1-b,j}$ to AA_j.

Upon receiving the two secret keys from $\mathcal{C}$, AA_j can interact with the two black-box oracles $\mathcal{O}_0$ and $\mathcal{O}_1$, and then execute *Enhanced_SecretKeyGen* $(\mathcal{O}_0 \leftrightarrow AA_j)$ and *Enhanced_SecretKeyGen* $(\mathcal{O}_1 \leftrightarrow AA_j)$ in which AA_j acts as the authority and uses the same secret values c_b, e_b, m_b as interacting with $\mathcal{C}$ in Challenge phase. Assume $\Sigma_{\mathcal{O}_0}$ and $\Sigma_{\mathcal{O}_1}$ are proofs of knowledge sent from AA_j respectively in the *Enhanced_SecretKeyGen* algorithm.

If AA_j can distinguish the two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$ with the advantage $Adv_{\mathcal{O}}$, and the two proofs $\Sigma_{\mathcal{O}_0}$ and $\Sigma_{\mathcal{O}_1}$ are both correct, then AA_j can execute *Enhanced_SecretKeyGen* $(AA_j \leftrightarrow AA_j)$ with itself by using $(PK_{U_0}, \widetilde{U_0})$ and $(PK_{U_1}, \widetilde{U_1})$ as the users and $\Sigma_{\mathcal{O}_0}, \Sigma_{\mathcal{O}_1}$ as the proofs sent from the authority, and outputs $SK'_{0,j}$ and $SK'_{1,j}$ for U_0 and U_1, respectively. Since $(PK_{U_0}, \widetilde{U_0})$ and $(PK_{U_1}, \widetilde{U_1})$ have the identical distribution with the oracles, the outputs $SK'_{0,j}$ and $SK'_{1,j}$ are the same as $SK_{0,j}$ and $SK_{1,j}$ sent from the challenger. Hence, AA_j can output its guess on b according to $SK'_{0,j}$ and $SK'_{1,j}$, and the advantage Adv_{AA_j} is the same as $Adv_{\mathcal{O}}$. Due to the hiding property of commitment scheme and zero knowledge property of proof of knowledge scheme, the two oracles are computationally indistinguishable from AA_j's view. Thus $Adv_{\mathcal{O}}$ and Adv_{AA_j} are negligible. Therefore, *Enhanced_SecretKeyGen* is secure against malicious authority AA_j.

VI. SCHEME ANALYSIS

A. SECURITY AND FUNCTIONALITY

In this subsection, we detail the comprehensive security and functionality comparison among the proposed scheme and some MACP-ABE based schemes [12]–[17] and ABSC schemes [7]–[10] in Table 2 and Table 3. Therein, ✓ represents the capability to achieve the corresponding index, whereas ✗ denotes the opposite. MBF represents monotone Boolean function, and TG represents the threshold gate.

Table 2 and Table 3 show that our scheme supports many useful properties, such as collusion resistance, privacy protection, expressiveness, computation outsourcing, anonymous authentication, multi-authority and public verification. Our scheme also realizes the security in the standard model.

B. PERFORMANCE

This subsection numerically analyzes the performance of the proposed scheme against some existing CP-ABSC based schemes [7]–[10] and MACP-ABE based schemes [12]–[17] in terms of the size of secret key and ciphertext and computation overhead (number of required exponentiations and pairing computations) of *Signcryption* and *DeSigncryption* algorithms. In the rest of this section, assume that U^d_{max} is the maximum number of decryption attributes of a user. $T^e_{\mathbb{G}}$ and $T^e_{\mathbb{G}_T}$ denote the running time required for one exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$, T^p is the running time for one pairing operation. $|\mathbb{G}|$ and $|\mathbb{G}_T|$ denote the size of the element in $\mathbb{G}$ and $\mathbb{G}_T$. For simplicity, we assume that the number of decryption (verifying) attributes required in *DeSigncryption* phase is also $l_{e,max}$ $(l_{s,max})$ and ignore the cost time of Hash functions and operations in $\mathbb{Z}_p$.

TABLE 2. Security and functionality comparison of MACP-ABE based schemes.

Schemes	[12]	[13]	[14]	[15]	[16]	[17]	Ours
Collusion Resistance	×	✓	✓	✓	✓	✓	✓
Standard Model	✓	×	×	×	×	×	✓
Privacy Protection	✓	×	×	×	×	×	✓
Encryption Predicate	MBF	MBF	MBF	MBF	MBF	MBF	MBF
Computation Outsourcing	×	✓	×	×	✓	✓	✓
Anonymous Authentication	×	×	×	✓	×	×	✓

TABLE 3. Security and functionality comparison of ABSC based schemes.

Schemes	[7]	[8]	[9]	[10]	Ours
Collusion Resistance	✓	✓	✓	✓	✓
Standard Model	✓	×	✓	×	✓
Privacy Protection	×	×	×	×	✓
Signcryptor Privacy	✓	✓	✓	×	✓
Signing Predicate	MBF	MBF	MBF	MBF	MBF
Encryption Predicate	MBF	MBF	MBF	TG	MBF
Computation Outsourcing	×	×	×	×	✓
Multi-Authority	×	×	×	×	✓
Public Verifiability	×	×	✓	✓	✓

Table 4 details the storage comparison on MACP-ABE based schemes. It is clear that the size of the decryption secret key of our scheme is larger than other schemes because to protect the attributes of the user, we employ the commitment scheme and zero knowledge proof in key generation phase and hence introduce $\left\{\Theta'_{uid,j}, R'_{uid,j}, V_{uid,j}, V'_{uid,j}\right\}_{j\in[N]}$, whereas the schemes in [13]–[17] do not consider the privacy of attributes. Compared with [12], we consider the anonymity authentication that incurs $3N\,|\mathbb{G}|$ in the common secret key. Table 4 also illustrates that our scheme incurs more storage overhead of ciphertext than others, except for those in [15] and [16]. This occurs because the ciphertext in our scheme consists of $\left\{\left\{S_{1,j,i}\right\}_{i\in[\ell_{s,j}]}, S_{2,j}\right\}_{j\in[N]}$ and $\left\{\left\{C_{j,5,i}\right\}_{i\in[l_{e,j}]}\right\}_{j\in[N]}$, which are used for verification and collusion attack resistance.

Table 5 shows the computation overhead comparison of *Signcryption* (without signing) and *Decryption* algorithms. Compared with [12], our scheme introduces $2Nl_{e,max}T^e_{\mathbb{G}}$ to construct $C_{j,5,i}$ for collusion resistance. For decryption, our

TABLE 4. Storage comparison of MACP-ABE based schemes.

schemes	decryption secret key	ciphertext
[12]	$(6N + U^d_{max})\|\mathbb{G}\|$	$\|\mathbb{G}_T\| + (3N + 2Nl_{e,max})\|\mathbb{G}\|$
[13]	$(2 + N + U^d_{max})\|\mathbb{G}\| + N\|\mathbb{G}_T\|$	$\|\mathbb{G}_T\| + (2 + 3Nl_{e,max})\|\mathbb{G}\|$
[14]	$U^d_{max}\|\mathbb{G}\|$	$(1 + Nl_{e,max})\|\mathbb{G}_T\| + 2Nl_{e,max}\|\mathbb{G}\|$
[15]	$U^d_{max}\|\mathbb{G}\|$	$(1 + Nl_{e,max})\|\mathbb{G}_T\| + (2 + 2Nl_{e,max} + 2Nl_{s,max})\|\mathbb{G}\|$
[16]	$(1 + U^d_{max})\|\mathbb{G}\|$	$(3Nl_{e,max} + 1)\|\mathbb{G}_T\| + 4Nl_{e,max}\|\mathbb{G}\|$
[17]	$(2N + U^d_{max})\|\mathbb{G}\|$	$\|\mathbb{G}_T\| + (2 + 3Nl_{e,max})\|\mathbb{G}\|$
Ours	$(9N + U^d_{max})\|\mathbb{G}\|$	$\|\mathbb{G}_T\| + (4N + 3Nl_{e,max} + Nl_{s,max})\|\mathbb{G}\|$

TABLE 5. Time comparison of *Signcryption* and *Decryption*.

schemes	***Signcryption*** (without signing)	***Decryption*** (user side)
[12]	$NT^e_{\mathbb{G}_T} + (3N + 3Nl_{e,max})T^e_{\mathbb{G}}$	$(4N + 2Nl_{e,max})T^p + Nl_{e,max}T^e_{\mathbb{G}_T}$
[13]	$2NT^e_{\mathbb{G}_T} + (3 + N + 4Nl_{e,max})T^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}$
[14]	$(1 + 2Nl_{e,max})T^e_{\mathbb{G}_T} + 3Nl_{e,max}T^e_{\mathbb{G}}$	$2Nl_{e,max}T^p + Nl_{e,max}T^e_{\mathbb{G}_T}$
[15]	$(1 + 2Nl_{e,max})T^e_{\mathbb{G}_T} + 3Nl_{e,max}T^e_{\mathbb{G}}$	$2Nl_{e,max}T^p$
[16]	$(1 + 3Nl_{e,max})T^e_{\mathbb{G}_T} + 5Nl_{e,max}T^e_{\mathbb{G}}$	$2T^e_{\mathbb{G}_T}$
[17]	$NT^e_{\mathbb{G}_T} + (1 + 5Nl_{e,max})T^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}$
Ours	$NT^e_{\mathbb{G}_T} + (4N + 5Nl_{e,max})T^e_{\mathbb{G}}$	$T^p + T^e_{\mathbb{G}_T} + (3N + 3Nl_{e,max})T^e_{\mathbb{G}}$

construction requires one pairing operation on the user side, which is more efficient than [12], [14], and [15]. Compared with [13], [16], and [17], our scheme additionally incurs $T^p + (3N + 3Nl_{e,max})\,T^e_{\mathbb{G}}$ computation overhead on the user side to protect the privacy of attributes when offloading *PartialDecryption* on cloud server resources.

Assume that U_{max} is the maximum total number of attributes of a user. If we set $N = 1$, then the proposed scheme is a traditional CP-ABSC scheme. In Table 6, we compare the asymptotic complexity of our scheme with CP-ABSC schemes [7]–[10]. As seen from Table 6, the size of the secret key is linear to the size of the attribute universe, which is not different between our scheme and others. To protect the privacy of the attributes and to resist the collusion attack, we also introduce $4\,|\mathbb{G}|$ and $(4 + l_{e,max})\,|\mathbb{G}|$ storage overhead in the secret key and ciphertext, respectively. However, Table 6 indicates that our scheme incurs less computation overhead of *DeSigncryption* on the user side than do the other schemes. Our scheme only requires one pairing computation, but the number of pairings in [7] and [8] is linear to the sum of required decryption and signing attributes.

TABLE 6. Asymptotic complexity comparison of CP-ABSC based schemes.

schemes	Secret key	Ciphertext	***DeSigncryption*** user side
[7]	$(4 + U_{max})\|\mathbb{G}\|$	$\binom{4 + l_{e,max}}{+2l_{s,max}}\|\mathbb{G}\|$	$(4 + 2l_{s,max} + 2l_{e,max})T^p + l_{e,max}T^e_{\mathbb{G}_T} + (2l_{s,max} + 3)T^e_{\mathbb{G}}$
[8]	$(4 + U_{max})\|\mathbb{G}\|$	$\|\mathbb{G}_T\| + \binom{5 + l_{e,max}}{+l_{s,max} + n_{s,max}}\|\mathbb{G}\|$	$(4 + l_{s,max}n_{s,max} + 2l_{e,max} + n_{s,max})T^p + (l_{e,max} + l_{s,max})T^e_{\mathbb{G}_T} + (2l_{s,max}n_{s,max} + n_{s,max})T^e_{\mathbb{G}}$
[9]	$(4 + U_{max})\|\mathbb{G}\|$	$\binom{5 + l_{e,max}}{+l_{s,max}}\|\mathbb{G}\|$	$2T^p + 3l_{e,max}T^e_{\mathbb{G}}$
[10]	$\binom{6l_{e,max}}{+3l_{s,max}}\|\mathbb{G}$	$6\|\mathbb{G}\|$	$8T^p + (l_{s,max} + 4l_{e,max} + 4l^2_{e,max})T^e_{\mathbb{G}}$
Ours	$(9 + U_{max})\|\mathbb{G}\|$	$\|\mathbb{G}_T\| + \binom{4 + 3l_{e,max}}{+l_{s,max}}\|\mathbb{G}\|$	$T^p + T^e_{\mathbb{G}_T} + (3 + 3l_{e,max})T^e_{\mathbb{G}}$

Compared with [9], our construction requires $4 + l_{s,max}$ pairing operations in decryption and verification, whereas in [9], $(5 + l_{s,max})$ pairings are needed. Moreover, if the ciphertext verification is performed by a trusted intermediate party, the receiver in our scheme can decrypt the ciphertext within one pairing operation. Our scheme also employs $3 + 3l_{e,max}$ exponentiations to compute the transformed secret key and protect the attribute privacy which is not considered in [9]. In [10], although the size of ciphertext is only $6\,|\mathbb{G}|$, eight pairings are required to recover the plaintext. Additionally, the number of exponentiation in our scheme is $3 + 3l_{e,max}$, whereas that for [10] is $l_{s,max} + 4l_{e,max} + 4l^2_{e,max}$. Therefore, our scheme performs well concerning privacy protection and is efficient from a computation point of view.

We implement the whole architectures of MACP-ABE based schemes [12]–[17], CP-ABSC based schemes [7]–[10] and ours with Pairing-based Cryptography (PBC) library version 0.5.14 on an Ubuntu system 14.04 with a 2.6 GHz processor and 4G RAM. We employ 160-bit Type A elliptic curve group constructed on $y^2 = x^3 + x$, and the computation cost for one pairing operation is 2.1 ms, and that of exponentiation on $\mathbb{G}$ and $\mathbb{G}_T$ are 0.7 and 0.2 ms, respectively. Each value in Figures 3, 4, 5, 6, 7, and 8 is the mean of 10 simulation trials.

Let N_{AA} be the number of attributes of data user monitored by each authority. For simplicity, assume $|\widetilde{AA}_j| = |\widetilde{AA}_j \cap \widetilde{U}_d| = \ell_{e,max} = N_{AA}$. Then, the computation overhead comparison of *Signcryption* (without signing) and *DeSigncryption* (without verifying) algorithms between our scheme and [12]–[17] can be conducted according to the number of authorities N and the number of attributes per authority N_{AA}. In Fig. 3 and Fig. 5, we set $N = 10$, while in Fig. 4 and Fig. 6, we assume $N_{AA} = 10$.

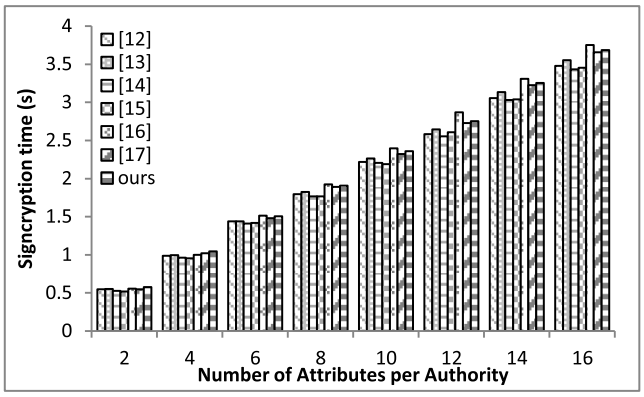


FIGURE 3. Signcryption (without signing).

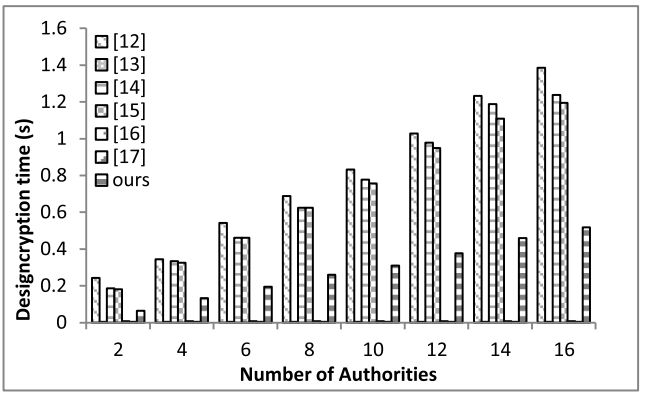


FIGURE 6. Designcryption (user side).

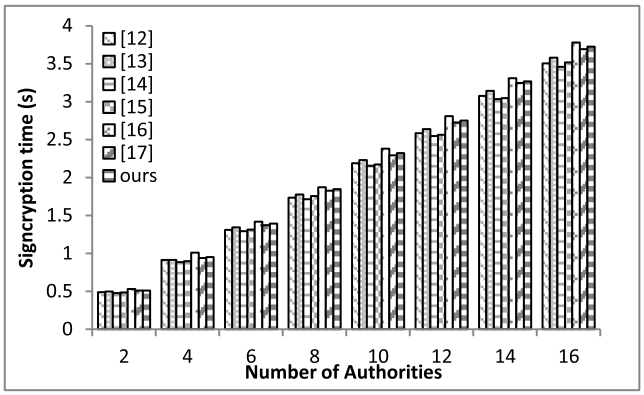


FIGURE 4. Signcryption (without signing).

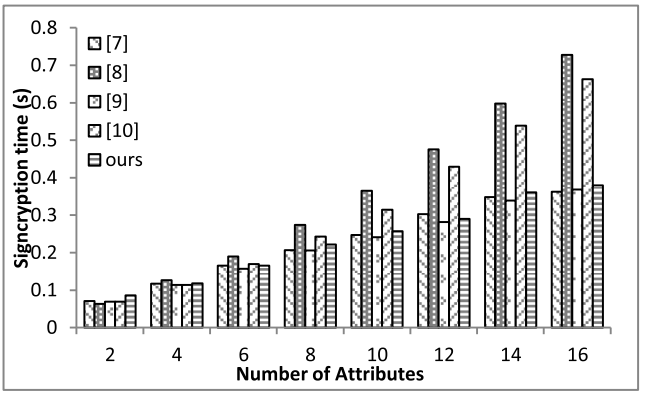


FIGURE 7. Signcryption.

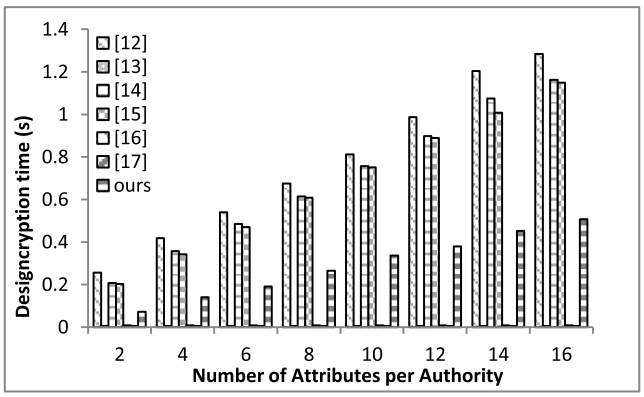


FIGURE 5. Designcryption (user side).

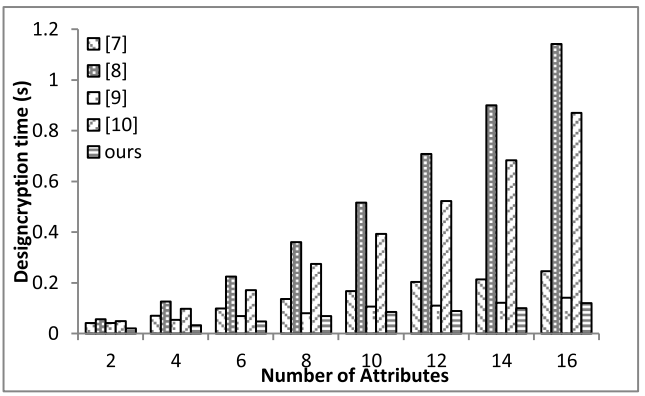


FIGURE 8. Designcryption.

Fig. 3 and Fig. 4 show that our scheme and [16], [17] nearly have the same efficiency in the encryption algorithm and that all incur more computation overhead than do the other schemes. The reason is that our scheme needs $(4N + 5Nl_{e,max})\, T^e_{\mathbb{G}}$ and more multiplication in $\mathbb{G}$ to compute the ciphertext to achieve attribute privacy protection. Fig. 5 and Fig. 6 illustrate that the performance of our scheme is better than [12], [14], and [15] because the most computation-consuming job of decryption is offloaded on cloud server. Compared with [13], [16], and [17], our scheme incurs $(3N + 3Nl_{e,max})\, T^e_{\mathbb{G}}$ to compute a transformed secret key and one pairing operation to compute CT_R. However, [13], [16], and [17] do not consider the privacy of attributes.

Assume that $N = 1$ and $\ell_{e,max} = \ell_{s,max} = N_{AA}$. Fig. 7 and Fig. 8 describe the computation overhead comparison of *Signcryption* and *DeSigncryption* algorithms among the schemes [7]–[10]. It is clear that the performance of *Signcryption* of our scheme is nearly the same as that of [7] and [9] and is better than [8] and [10]. Since our

scheme and Sreenivasa's scheme [9] are publicly verifiable, the *Verify* (PP, CT) algorithm can be outsourced to a trusted party, and the efficient of *DeSigncryption* on the user side can be greatly improved. Moreover, our scheme needs only one pairing operation on the user side. Overall, our scheme performs well in decryption on the user side and supports additional useful properties such as multiple authorities, anonymous authentication and attribute privacy protection.

VII. CONCLUSION

In this paper, we proposed a PMDAC-ABSC scheme for data sharing in the cloud storage system and the privacy preserving extension to protect the attribute privacy. In our construction, multiple authorities can work independently and issue the secret key for users without knowing a user's attributes. The proposed scheme realizes the security in the standard model and supports many practical properties, such as fine-grained access control, confidentiality, unforgeability, anonymous authentication and public verifiability. The computation overhead of the decryption algorithm is also alleviated by outsourcing the costly operations to the cloud server without degrading the attribute privacy. The security analysis, asymptotic complexity and performance comparisons indicate that our construction can balance the security with overhead efficiency.

In future work, it would be interesting to realize a fully secure MACP-ABSC based access control scheme instead of a selectively secure scheme. Additionally, how to reduce the storage overhead while achieving the same security level is another challenge.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *Commun. ACM*, vol. 53, no. 6, pp. 50–53, 2010.

[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. 14th Int. Conf. Financial Cryptogr. Data Secur.*, Tenerife, Spain, Jan. 2010, pp. 136–149.

[3] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Topics in Cryptology—CT-RSA*. Berlin, Germany: Springer, Feb. 2011, pp. 376–392.

[4] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan. 2012.

[5] M. Gagné, S. Narayan, and R. Safavi-Naini, "Threshold attribute-based signcryption," in *Security and Cryptography for Networks*. Berlin, Germany: Springer, Sep. 2010, pp. 154–171.

[6] Y. S. Rao and R. Dutta, "Expressive attribute based signcryption with constant-size ciphertext," in *Progress in Cryptology—AFRICACRYPT*. Cham, Switzerland: Springer, May 2014, pp. 398–419.

[7] C. Chen, J. Chen, H. W. Lim, Z. Zhang, and D. Feng, "Combined public-key schemes: The case of ABE and ABS," in *Provable Security*. Berlin, Germany: Springer, Sep. 2012, pp. 53–69.

[8] H. Liu, Y. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption," *Future Gener. Comput. Syst.*, vol. 52, pp. 67–76, Nov. 2015.

[9] Y. S. Rao, "A secure and efficient ciphertext-policy attribute-based signcryption for personal health records sharing in cloud computing," *Future Gener. Comput. Syst.*, vol. 67, pp. 133–151, Feb. 2017.

[10] G. Yu and Z. Cao, "Attribute-based signcryption with hybrid access policy," *Peer-to-Peer Netw. Appl.*, vol. 10, no. 1, pp. 253–261, 2017.

[11] J. Han, W. Susilo, Y. Mu, Y. Zhou, and M. H. Au, "PPDCP-ABE: Privacy-preserving decentralized ciphertext-policy attribute-based encryption," in *Computer Security—ESORICS*. Cham, Switzerland: Springer, 2014, pp. 73–90.

[12] G. Han, W. Susilo, Y. Mu, Y. Zhou, and A. Au, "Improving privacy and security in decentralized CP-ABE," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 665–678, Mar. 2015.

[13] R. Jiang, X. Wu, and B. Bhargava, "SDSS-MAC: Secure data sharing scheme in multi-authority cloud storage systems," *Comput. Secur.*, vol. 62, pp. 193–212, Sep. 2016.

[14] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, May 2011, pp. 568–588.

[15] S. Ruj, M. Stojmenovic, and A. Nayak, "Decentralized access control with anonymous authentication of data stored in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 384–394, Feb. 2014.

[16] J. De Sourya and S. Ruj, "Efficient decentralized attribute based access control for mobile clouds," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2017.2754255.

[17] K. Yang, H. Jia, K. Ren, and B. Zhang, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.

[18] Y. Meng and Y. Meng, "A novel attribute-based signcryption scheme in cloud computing environments," in *Proc. IEEE Int. Conf. Inf. Automat.*, Ningbo, China, Aug. 2016, pp. 1976–1979.

[19] X. Liu, Y. Xia, Z. Sun, and X. Liu, "Provably secure attribute based signcryption with delegated computation and efficient key updating," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 5, pp. 2646–2659, 2017.

[20] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, May. 2005, pp. 457–473.

[21] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou, "Fine-grained access control system based on outsourced attribute-based encryption," in *Computer Security—ESORICS*. Berlin, Germany: Springer, Sep. 2013, pp. 592–609.

[22] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.

[23] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, and I. You, "Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing," *Inf. Sci.*, vol. 379, pp. 42–61, Feb. 2017.

[24] A. Lounis, A. Hadjidj, A. Bouabdallah, and Y. Challal, "Healing on the cloud: Secure cloud architecture for medical wireless sensor networks," *Future Gener. Comput. Syst.*, vol. 55, pp. 266–277, Feb. 2016.

[25] Q. Huang, Y. Yang, and L. Wang, "Secure data access control with ciphertext update and computation outsourcing in fog computing for Internet of Things," *IEEE Access*, vol. 5, pp. 12941–12950, Jul. 2017.

[26] A. Rohit, K. M. Sraban, and S. Kouichi, "A scalable attribute-set-based access control with both sharing and full-fledged delegation of access privileges in cloud computing," *Comput. Elect. Eng.*, vol. 57, pp. 241–256, Jan. 2017.

[27] G. Wang, Q. Liu, J. Wu, and M. Guo, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," *Comput. Secur.*, vol. 30, no. 5, pp. 320–331, 2011.

[28] M. Chase, "Multi-authority attribute based encryption," in *Proc. 4th Theory Cryptogr. Conf. (TCC)*, Amsterdam, The Netherlands. Berlin, Germany: Springer, Feb. 2007, pp. 515–534.

[29] M. Chase and S. S. M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, Chicago, IL, USA, Nov. 2009, pp. 121–130.

[30] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 2625–2633.

[31] Q. Li, J. Ma, R. Li, X. Liu, J. Xiong, and D. Chen, "Secure, efficient and revocable multi-authority access control system in cloud storage," *Comput. Secur.*, vol. 59, pp. 45–59, Jun. 2016.

[32] K. Nomura, M. Mohri, Y. Shiraish, and M. Morii, "Attribute revocable multi-authority attribute-based encryption with forward secrecy for cloud storage," *IEICE Trans. Inf. Syst.*, vol. 100, no. 10, pp. 2420–2431, 2017.

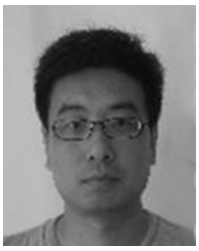
[33] S. S. M. Chow, "A framework of multi-authority attribute-based encryption with outsourcing and revocation," in *Proc. 21st ACM Symp. Access Control Models Technol.*, New York, NY, USA, Jun. 2016, pp. 215–226.

[34] J. Zhou *et al.*, "Securing outsourced data in the multi-authority cloud with fine-grained access control and efficient attribute revocation," *Comput. J.*, vol. 60, no. 8, pp. 1210–1222, Aug. 2017.

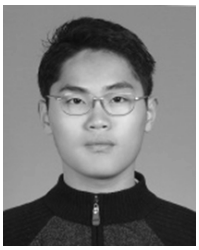
[35] K. Xue *et al.*, "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 953–967, Apr. 2017.
[36] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1484–1496, May 2016.
[37] H. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signature: Achieving attribute privacy and collusion resistance," IACR Cryptol. ePrint Arch., Tech. Rep. 328, 2008. [Online]. Available: http://eprint.iacr.org/2008/328
[38] J. Herranz, F. Laguillaumie, B. Libert, and C. Ràfols, "Short attribute-based signatures for threshold predicates," in *Topics in Cryptology—CT-RSA*. Berlin, Germany: Springer, Feb. 2012, pp. 51–67.
[39] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, New York, NY, USA, Apr. 2010, pp. 60–69.
[40] S. F. Shahandashti and R. Safavi-Naini, "Threshold attribute-based signatures and their application to anonymous credential systems," in *Progress in Cryptology—AFRICACRYPT*. Berlin, Germany: Springer, 2009, pp. 198–216.
[41] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in *Public-Key Cryptography—PKC*. Berlin, Germany: Springer, Feb. 2013, pp. 125–142.
[42] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," in *Public Key Cryptography—PKC*. Berlin, Germany: Springer, 2011, pp. 35–52.
[43] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography—PKC*. Berlin, Germany: Springer, 2011, pp. 53–70.
[44] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 1992, pp. 129–140.
[45] J. Camenisch, R. Chaabouni, and A. Shelat, "Efficient protocols for set membership and range proofs," in *Advances in Cryptology—ASIACRYPT*. Berlin, Germany: Springer, 2008, pp. 234–252.
[46] J. Chen, H. W. Lim, S. Ling, L. Su, and H. Wang, "Spatial encryption supporting non-monotone access structure," *Des., Codes Cryptogr.*, vol. 73, no. 3, pp. 731–746, 2014.

QIAN XU received the M.S. degree from the Department of Computer Science and Technology, Tongji University, China, in 2012, where he is currently pursuing the Ph.D. degree. His research interests include cryptography and cloud security.

CHENGXIANG TAN received the Ph.D. degree in engineering from Northwestern Polytechnic University, China, in 1994. He is currently a Professor of computer science with Tongji University. His research interests include cyber security and privacy preservation.

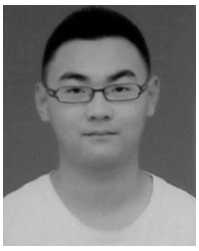


ZHIJIE FAN received the M.S. degree from Zhejiang University, Hangzhou, China, in 2009. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tongji University. His research interests include cloud and mobile security.



WENYE ZHU received the B.S. degree from the Department of Computer Science and Technology, Tongji University, China, in 2013, where he is currently pursuing the Ph.D. degree. His research interest is in cloud security.

YA XIAO received the B.S. degree from the Department of Computer Science and Technology, Tongji University, China, in 2015, where she is currently pursuing the Ph.D. degree. Her research interests include cloud security and machine learning.



FUJIA CHENG received the B.S. degree from the Department of Computer Science and Technology, Tongji University, China, in 2016, where he is currently pursuing the M.S. degree.