sensors

MDPI

Article

Secure Data Access Control for Fog Computing Based on Multi-Authority Attribute-Based Signcryption with Computation Outsourcing and Attribute Revocation

Qian Xu *, Chengxiang Tan, Zhijie Fan, Wenye Zhu, Ya Xiao and Fujia Cheng

Department of Computer Science and Technology, Tongji University, Cao An Highway, Shanghai 201804, China; jerrytan@tongji.edu.cn (C.T.); 1310898@tongji.edu.cn (Z.F.); 1310513@tongji.edu.cn (W.Z.); 1710053@tongji.edu.cn (Y.X.); 1631585@tongji.edu.cn (F.C.)

* Correspondence: 1310512@tongji.edu.cn; Tel.: +86-135-1212-2032

Received: 23 April 2018; Accepted: 16 May 2018; Published: 17 May 2018

Abstract: Nowadays, fog computing provides computation, storage, and application services to end users in the Internet of Things. One of the major concerns in fog computing systems is how fine-grained access control can be imposed. As a logical combination of attribute-based encryption and attribute-based signature, Attribute-based Signcryption (ABSC) can provide confidentiality and anonymous authentication for sensitive data and is more efficient than traditional "encrypt-then-sign" or "sign-then-encrypt" strategy. Thus, ABSC is suitable for fine-grained access control in a semi-trusted cloud environment and is gaining more and more attention recently. However, in many existing ABSC systems, the computation cost required for the end users in signcryption and designcryption is linear with the complexity of signing and encryption access policy. Moreover, only a single authority that is responsible for attribute management and key generation exists in the previous proposed ABSC schemes, whereas in reality, mostly, different authorities monitor different attributes of the user. In this paper, we propose OMDAC-ABSC, a novel data access control scheme based on Ciphertext-Policy ABSC, to provide data confidentiality, fine-grained control, and anonymous authentication in a multi-authority fog computing system. The signcryption and designcryption overhead for the user is significantly reduced by outsourcing the undesirable computation operations to fog nodes. The proposed scheme is proven to be secure in the standard model and can provide attribute revocation 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.

Keywords: Internet of Things; fog computing; Attribute Based Signcryption; multi-authority; access control; anonymous authentication

1. 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 cloud servers in a cost-effective manner [1]. However, the advance of the Internet of Things (IoTs) has posed a challenge to the centralized cloud computing system due to its geo-distribution, location awareness, and low latency requirements. To solve the problem, Cisco proposed the concept of fog computing in 2014, where a layer consisting of fog devices (such as routers, access points, and IP video cameras) bridges between the cloud server and end users [2]. In a fog computing system, the fog devices, termed as fog nodes, are distributed and implemented at the edge of networks [3]. Since fog nodes are much closer to end users than the cloud server and have plentiful computing resources and wireless

communication facility, some of the computing tasks can be outsourced to fog nodes from the nearby end user, which alleviates the computation burden of the users and significantly improve the efficiency. Thus, the fog computing paradigm can be applied in many real-time and geographically distributed applications, such as wireless sensors, smart grids and health fog applications [4].

However, there are still various challenging obstacles in fog computing systems, such as the privacy and security of users' data [5,6]. Traditionally, a cloud server is not fully trusted by the data owner in cloud computing system, and the data uploaded may contain sensitive information; hence, the data should be encrypted before outsourcing to the cloud. In accord with cloud computing, message confidentiality should also be considered in fog computing systems. Moreover, since the fog nodes are more easily compromised than cloud servers [6], it is required that fog nodes should alleviate the computation burden of end devices without degrading the privacy in fog computing systems. In addition to confidentiality, data owners may wish to impose fine-grained access control such that only users with certain attributes have access to the data [7]. For example, in a health fog system, which combines the advantage of both the fog computing and original cloud-based healthcare services [8], personal health records usually contain abundant sensitive information, such as weight, heart rate, and blood type. After gathering by sensors, the personal health record may be uploaded to the cloud for the user's individual needs or to perform real-time analytics. To ensure the privacy of the health data, an access control system should guarantee that only the users authorized by the data owner can access the data. For instance, to analyze whether the blood pressure is normal, the owner "Alice" wants to share her health data to users with attributes "$Institution = Hospital \wedge Role = Doctor \wedge Gender = Female$". One of the effective techniques to address this fine-grained access requirement is attribute-based encryption (ABE) [9]. It realizes the confidentiality and access control on data based on encryption under an access policy defined over the set of attributes.

Besides the confidentiality and fine-grained access control, it is also necessary to provide anonymity authentication for data sharing between users in the access control mechanism. For instance, the owner "Alice", aged 20, would like to encrypt and store some sensitive health information in the cloud but does not want to be recognized. When a data user, such as the doctor or researcher, accesses the data, he/she can verify that the data is actually uploaded by a patient with certain credentials such as "$Gender = Female \wedge Age \in [18,30]$" without knowing the patient's real identity "Alice" or her real age.

A feasible and promising solution is the Attribute Based Signcryption (ABSC) scheme, which takes advantages of Attribute-Based Encryption (ABE) and Attribute-Based Signature (ABS), and is more efficient than do the traditional "encrypt-then-sign" or "sign-then-encrypt" strategies. ABSC employs ABE to provide confidentiality and fine-grained access control, and uses ABS to achieve authentication without revealing the data owner's sensitive attributes. Traditionally, ABE can be classified into two categories: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE). In KP-ABE, the secret key is associated with an access structure (predicate), and the message is encrypted with a set of attributes. While in CP-ABE, predicate is assigned to the plaintext message. Similarly, ABS has two categories: Signature-Policy ABS (SP-ABS) wherein the predicate is embedded in the signature, and Key-Policy ABS (KP-ABS) wherein the predicate is associated with the secret key. The Ciphertext-Policy ABSC (CP-ABSC) [10] supports CP-ABE and SP-ABS, and the Key-Policy ABSC (KP-ABSC) [11] supports KP-ABE and KP-ABS. Recently, many data access control schemes based on ABSC have been proposed, as in [12–15]. Although some of them are efficient, three problems must be considered when implementing ABSC scheme in fog computing environment. The first one is performance. The traditional ABSC scheme is typically computationally intensive. In particular, the cost of signcryption and designcryption on the user side are proportional to the complexity of predicates. One possible strategy to alleviate the computation overhead required on end user is to outsource the most computation-consuming job of signcryption and designcryption to the fog node. Although many ABE schemes with outsourcing encryption and decryption, as in [16–20], have been proposed in recent years for secure data sharing in fog computing system, realizing ABSC scheme with anonymous

authentication and efficient computation outsourcing is still a challenge since ABSC schemes contain both of the signing and encryption protocols. The second problem is multi-authority. In traditional ABSC schemes, as in [12–15], a central authority is responsible for attribute management and key generation. However, in many applications, the predicate embedded in the ciphertext or signature can be written over attributes issued by different trust domains and authorities. For example, the health data uploaded by "Alice" may contain the encryption predicate as "(*Doctor* $\vee$ *Researcher*) $\vee$ *Female*". Since only a hospital can authorize a person the attribute "*Doctor*" and only a research organization can certify that a person is a "*Researcher*", it is not practical to authorize access right to a person by a single authority. Therefore, it is necessary to distribute attribute management and secret key generation from a single central authority over many authorities. Some multi-authority ABE schemes for fog computing, as in [17], have been proposed, whereas constructing multi-authority ABSC scheme with outsourcing capability is still a blank. The third one is attribute revocation. For example, when the attributes of a doctor are updated from $A = \{Institution = Hospital \wedge Role = Doctor \wedge Gender = Female\}$ to $B = \{Institution = Hospital \wedge Gender = Female\}$, her access rights should be modified accordingly. Attribute revocation is not trivial and straightforward in ABE schemes. However, it has not been taken into account in multi-authority ABSC schemes with outsourcing capability.

The problem of designing a multi-authority data access control scheme based on ABSC with signcryption and designcryption outsourcing capabilities and attribute revocation for fog computing system, has received very little attention so far, although some schemes based on Multi-Authority ABE (MA-ABE) and ABS (MA-ABS) for cloud storage setting have been proposed, as in [21–26]. Meng et al. [27] proposed a decentralized KP-ABSC scheme for secure data sharing in the cloud. However, the scheme is just a combination of identity signature and MA-ABE, and only supports the threshold predicate. It also does not provide any security definition or computation outsourcing. Hong et al. [28] proposed a KP-ABSC scheme with outsourced designcryption and key exposure protection. However, the computation overhead of signcryption increases with the complexity of the predicate, and since the verification and decryption both have to be performed on the user side, the number of pairing operations evaluated on the user side is proportional to the sum of the required attributes, which is not acceptable to IoT devices. Moreover, the scheme in [28] does not support multi authorities and attribute revocation. We focus on CP-ABSC in access control application, as CP primitives are more suitable for the data owner to choose the predicate to determine who can access the sensitive data [14].

1.1. Contributions

In this paper, we propose OMDAC-ABSC, a novel data access control scheme for fog computing system based on Multi-Authority CP-ABSC (MACP-ABSC) supporting the computation outsourcing for both signcryptor (data owner) and designcryptor (data user). To the best of our knowledge, OMDAC-ABSC is the first scheme that significantly reduces computation burden from both data owners and data users in the multi-authority ABSC setting. Public verifiability, expressiveness and attribute revocation are also considered in our scheme. The main contributions can be summarized as follows:

(1) We propose a data access control scheme OMDAC-ABSC for fog computing system, in which fog nodes serve as a bridge between the cloud server and end users. In our scheme, heavy signcryption and designcryption operations can be outsourced from end users (e.g., tablet computers and smartphones) to fog nodes. In signcryption phase, the fog nodes are in charge of generating part of the ciphertext. In designcryption phase, the fog nodes can perform the partial decryption without degrading the data confidentiality, and the data user only requires a constant number of exponentiations to decrypt the ciphertext. Additionally, unlike other existing works such as [27,28], our scheme supports public verification, since the verification mechanism does not require the plaintext message or the data owner's public key. Thus the verification algorithm

can be performed by any trusted party, which alleviates the computation burden of the end user. Therefore, our construction is efficient from computation point of view.

(2) Unlike some existing ABE schemes for fog computing such as [16,18,19] and ABSC schemes such as [15,27,28], the proposed OMDAC-ABSC scheme is more expressiveness and supports any monotone Boolean function predicates represented by monotone span programs (MSP) for both signing and encryption. Moreover, we remove the limitation that the labeling functions ρ in signing and encryption predicates should be injective functions.

(3) Our OMDAC-ABSC scheme is proven to be secure in the standard model. We also formally prove that our construction satisfies the properties of signcryptor privacy and collusion resistance.

(4) We also consider the attribute revocation in our OMDAC-ABSC scheme. In attribute revocation phase, the authority supervising the revoked attribute only distributes the update keys to the non-revoked users and the cloud server to update the corresponding components. It is also proved that our scheme guarantees both the forward and backward revocation security.

1.2. Paper Organization

The remainder of this paper is organized as follows: in Section 2, we discuss some related works. Then in Section 3, we review the necessary notations and cryptographic background that are used throughout the paper. In Section 4, we give the definition of our scheme and the security requirements. The details of the scheme and the security proof are elaborated in Sections 5 and 6, respectively. Section 7 is dedicated to discussing the functionality and performance of the scheme. Finally, we conclude this paper in Section 8.

2. Related Works

2.1. Access Control Schemes Based on ABE

ABE was first introduced by Sahai and Waters [9]. In ABE, a data owner can share sensitive data with others according to predicates (or access policies). Several works on ABE have been presented to address data access control in untrusted cloud servers. Recently, the ABE scheme was adopted in fog-computing systems to guarantee confidentiality and fine-grained access control. Heavy computations of encryption or decryption are outsourced to fog nodes to improve the efficiency. In [16], an anonymous user authentication in ciphertext update phase was realized, whereas the scheme only supports AND-gate predicate. Zuo et al. [18] proposed a CCA-secure ABE scheme with decryption outsourcing. However, the encryption phase of the scheme in [18] incurs heavy computation cost. Additionally, the scheme in [18] is only provably secure in the random oracle model and only supports the AND-gate encryption predicate. Zhang et al. [19] presented an ABE-based access control scheme for fog computing with outsourced encryption and decryption. Although the computation operations (pairings and exponentiations) for users to encrypt and decrypt are irrelevant to the complexity of predicate, the scheme only supports threshold encryption predicate, and requires both the cloud server and fog nodes to be trusted. Lounis et al. [29] proposed a cloud-based architecture for medical wireless sensor networks, in which the resource-constrained end devices outsource the costly computations to the trusted gateway. However, the decryption phase incurs heavy computation cost. Xiao et al. [30] constructed a fine-grained hybrid scheme for fog computing with the advantages of efficient data search and access authorization through online/offline encryption, delegation of search task and decryption to fog nodes, and provable security. Mao et al. [20] proposed an ABE scheme with verifiable outsourced decryption, whereas it incurs a heavy computation overhead in encryption phase. Li et al. [31] also proposed a fully verifiable ABE scheme with outsourcing capability. However, Liao et al. [32] showed that the verification mechanism proposed in [31] is not always correct.

In many ABE schemes, the attribute universe is assumed to be managed by a single authority. In reality, however, users' attributes may be monitored by different authorities. To track this problem, MA-ABE scheme was proposed by Chase et al. [33]. In MA-ABE, the attribute universe is divided

into multiple disjoint sets, and each authority controls one of these attribute sets. The user can successfully decrypt the ciphertext if and only if the user possesses at least a pre-specified number of attributes from each authority. Furthermore, Chase et al. [34] proposed an improved MA-ABE scheme to remove the fully trusted central authority by adopting a Pseudo Random Function (PRF) and a secure 2-party anonymous secret-key-issuing protocol. However, the multiple authorities must cooperate with each other, and the number of authorities must be determined in the initialization phase. Recently, many distributed access control schemes based on MA-ABE have been proposed, such as [21–26,35,36]. Han et al. [21] proposed a privacy-preserving decentralized CP-ABE based access scheme (PPDCP-ABE) to protect the user's privacy. However, PPDCP-ABE cannot resist collusion attack or support anonymous authentication. Rui et al. [22] constructed a MA-ABE scheme with secure attribute-level immediate attribute revocation. The scheme is only provably secure under the random oracle model. Lewko et al. [23] proposed a decentralized attribute-based encryption using the dual system encryption methodology. The secret keys of the user are tied to his global identity in order to resist collusion attack. However, the scheme realizes the security in random oracle model using the composite-order bilinear group, which incurs great computation overhead. Sourya et al. [25] proposed a decentralized data sharing scheme with outsourced decryption and user revocation. They also proposed a decentralized data sharing scheme where multiple attribute authorities distribute secret keys to the user [24]. In [26], the authors outsourced the main computation overhead in a decryption algorithm to the cloud. However, the security cannot be guaranteed if the revoked user eavesdrops to obtain the update keys and retrieves the ability to decrypt as a non-revoked user. To implement multi-authority ABE in fog computing system, Fan et al. [17] proposed a VO-MAACS scheme with verification mechanism. Although the encryption and decryption algorithms are outsourced, the scheme cannot support anonymous authentication and attribute revocation, and does not have security proof. Jung et al. [35] presented an anonymous privilege control scheme to address data and identity privacy in multi-authority cloud storage system. To guarantee the confidentiality of user's identity information, the scheme in [35] decomposes the central authority to multiple ones while preserving tolerance to compromise attack on the authorities. However, the security is realized in random oracle model, and the encryption predicate is the AND gate. In [36], the authors constructed a multi-authority data access control scheme with decryption outsourcing and attribute-level user revocation. The scheme supports any monotone encryption predicate and is adaptively secure in the standard model. Nevertheless, the scheme in [36] needs to deal with large composite-order group elements and thus incurs heavy computation overhead.

2.2. Attribute-Based Signature and Multi-Authority Attribute-Based Signature

ABS was first introduced by Maji et al. [37]. Due to their anonymity and authentication properties, many ABS schemes have been proposed. Like ABE, to overcome the drawback that only a single authority exists in the system, the concept of MA-ABS was introduced in [38]. In MA-ABS, there are multiple authorities and each authority controls one of disjoint attribute sets. The user is able to successfully sign the plaintext if he/she possesses a pre-specified number of attributes from multiple authorities.

2.3. Access Control Schemes Based on ABSC

ABSC scheme, first introduced by Gagné et al. [10], is a logical combination of ABE and ABS and can support many practical properties, including confidentiality, fine-grained access control, and authentication. Recently, many data access control schemes based on ABSC have been proposed, as in [11–15,27,28]. Y. Sreenivasa [11] 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 in the random oracle model. Chen et al. [12] 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 plaintext is required in verification mechanism. Liu et al. [13] proposed a secure PHR data access control scheme based on CP-ABE [39] and ABS [37]. However, it is only provably secure in a random oracle model. In [14], the authors constructed a CP-ABSC based access control scheme with public verifiability, but the scheme does not support computation outsourcing. Yu et al. [15] proposed the hybrid access policy ABSC scheme that supports KP-ABS and CP-ABE. The size of the ciphertext is constant, and the scheme realizes security in the standard model. Nevertheless, it only supports the threshold predicate in the encryption phase. Moreover, the above ABSC schemes only have a single authority and cannot be applied in the multi-authority system.

3. Preliminaries

By $a \xleftarrow{R} A$, 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 $\mathbb{Z}_p \backslash \{0\}$. $y \leftarrow A(x)$ 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 $\epsilon : \mathbb{Z} \to R$ 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 superscripts for signing and encryption, respectively. $Pr[E]$ denotes the probability of an event E occurring. For an unambiguous presentation of the paper, we define the important notations used in our scheme in the Appendix A.

Definition 1. *Bilinear maps [22]: 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 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ can be defined as follows:*

- *Bilinear. For all $u, v \in \mathbb{G}$, $a, b \in \mathbb{Z}_p$, $e\left(u^a, v^b\right) = e(u, v)^{ab}$.*
- *Non-degenerate. $e(g, g) \neq 1$.*
- *Computable. There is an efficient algorithm to compute the map e.*

$\mathcal{GG}\left(1^k\right) \to (e, p, \mathbb{G}, \mathbb{G}_T)$ takes as input a security parameter 1^k and outputs a bilinear group $(e, p, \mathbb{G}, \mathbb{G}_T)$ with prime order p and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Definition 2. *Decisional Bilinear Diffie-Hellman (BDH) Assumption [22]: 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 $\left(\vec{\mathcal{Y}}, \Omega \xleftarrow{R} \mathbb{G}_T\right)$ with the advantage $Adv_{\mathcal{A}} = \left| Pr\left[\mathcal{A}\left(\vec{\mathcal{Y}}, \Omega = e(g, g)^{abc}\right) = 1\right] - Pr\left[\mathcal{A}\left(\vec{\mathcal{Y}}, \Omega \xleftarrow{R} \mathbb{G}_T\right) = 1\right] \right| \geq \epsilon(k)$.*

Definition 3. *Decisional q-Parallel Bilinear Diffie-Hellman Exponent (q-PBDHE) Assumption [21]: Suppose that $a, w, b_1, b_2, \ldots, b_q \xleftarrow{R} \mathbb{Z}_p$, $\mathcal{GG}\left(1^k\right) \to (e, p, \mathbb{G}, \mathbb{G}_T)$ and g is a generator of $\mathbb{G}$. Given $\vec{\mathcal{Y}} = \left(g, g^w, g^a, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}}, \forall 1 \leq j \leq q, g^{wb_j}, g^{\frac{a}{b_j}}, \ldots, g^{\frac{a^q}{b_j}}, g^{\frac{a^{q+2}}{b_j}} \ldots, g^{\frac{a^{2q}}{b_j}}, \forall 1 \leq j, k \leq q, k \neq j, g^{\frac{awb_k}{b_j}}, \ldots, g^{\frac{a^q wb_k}{b_j}}\right)$, the decisional q-PBDHE assumption holds if no PPT adversary $\mathcal{A}$ can distinguish $\left(\vec{\mathcal{Y}}, \Omega = e(g, g)^{a^{q+1}w}\right)$ from $\left(\vec{\mathcal{Y}}, \Omega \xleftarrow{R} \mathbb{G}_T\right)$ with the advantage $Adv_{\mathcal{A}} = \left| Pr\left[\mathcal{A}\left(\vec{\mathcal{Y}}, \Omega = e(g, g)^{a^{q+1}w}\right) = 1\right] - Pr\left[\mathcal{A}\left(\vec{\mathcal{Y}}, \Omega \xleftarrow{R} \mathbb{G}_T\right) = 1\right] \right| \geq \epsilon(k)$.*

Definition 4. *Monotone Span Program (MSP) [11]: 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 $\rho : [\ell] \to \{v_1, v_2, \ldots, v_m\}$.*

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

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

Lemma 1 [14]. *If* $\Omega\left(\vec{x}\right) = 0$, *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* $i \in X_1$.

Definition 5. *Predicates [14]: 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*

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

Lemma 2 [14]. *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 ([14]). *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 a 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,* $\mathbb{V} = \left\{\vec{0}\right\}$ *and the predicate is an AND gate. In our construction, we consider the signing and encryption predicates consisting of both AND and OR gates.*

4. Scheme and Security Definitions

Our OMDAC-ABSC scheme consists of a multi-authority attribute-based signcryption (MACP-ABSC) scheme.

4.1. Multi-Authority Attribute-Based Signcryption

The MACP-ABSC scheme consists of the following five algorithms:

$GlobalSetup\left(1^k\right)$. Taking as input a security parameter 1^k, the algorithm outputs the public parameters PP. It also generates the public key PK_{uid} for the user with identity uid.

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

$SecretKeyGen\left(PP, PK_{aid}, SK_{aid}, PK_{uid}, \widetilde{U}\right)$. Taking as input PP, $\{PK_{aid}, SK_{aid}\}$ of authority AA_{aid}, user's public key PK_{uid} and attribute set $\widetilde{U} = \widetilde{U^d} \cup \widetilde{U^s}$, where $\widetilde{U^d}$ denotes the set of decryption attributes, and $\widetilde{U^s}$ is the set of signing attributes. $\widetilde{U^d} \cap \widetilde{U^s} = \varnothing$. The algorithm outputs the secret signing and decryption keys $SK_{uid,aid} = \left\{SK^s_{uid,aid}, SK^d_{uid,aid}\right\}$ for the user.

$Signcryption\left(\mathcal{M}, PP, \mathcal{R}_s, \mathcal{R}_e, \left\{SK^s_{do,k}\right\}_{k\in I}\right)$. Taking as input the plaintext $\mathcal{M}$, public parameters PP, signing and encryption predicates $\mathcal{R}_s, \mathcal{R}_e$, and the set of signcryptor's secret signing keys $\left\{SK^s_{do,k}\right\}_{k\in I}$, where I is the set of involved authorities in signcryption and do is signcryptor's identity. The algorithm outputs the ciphertext CT.

$DeSigncryption\left(PP, CT, PK_{du}, \left\{SK^d_{uid,k}\right\}_{k\in I}\right)$. This algorithm intakes the public parameters PP, ciphertext CT, public key PK_{du} of the data user U_{du} (designcryptor), and the set of designcryptor's secret decryption keys $\left\{SK^d_{uid,k}\right\}_{k\in I}$, outputs the plaintext $\mathcal{M}$ or $\perp$.

Definition 7. *Assume the signcryptor is denoted by U_{do} and designcryptor is denoted by U_{du}. We say that the MACP-ABSC scheme is correct if $\mathcal{R}_s\left(\widetilde{U^s_{du}}\right) = 1$, $\mathcal{R}_e\left(\widetilde{U^d_{du}}\right) = 1$, then $Pr\left[\mathcal{M} \leftarrow DeSigncryption\left(PP, CT, PK_{du}, \left\{SK^d_{du,k}\right\}_{k\in I}\right)\right] = 1$, where $\{PP, PK_{do}PK_{du}\} \leftarrow GlobalSetup\left(1^k\right)$, $\{PK_k, SK_k\} \leftarrow AuthoritySetup(PP)$, $SK^s_{do,k} \leftarrow SecretKeyGen\left(PP, PK_k, SK_k, PK_{do}, \widetilde{U_{do}}\right)$, $SK^d_{du,k} \leftarrow SecretKeyGen\left(PP, PK_k, SK_k, PK_{du}, \widetilde{U_{du}}\right)$, $CT \leftarrow Signcryption\left(\mathcal{M}, PP, \mathcal{R}_s, \mathcal{R}_e, \left\{SK^s_{do,k}\right\}_{k\in I}\right)$.*

4.2. High-Level Overview of OMDAC-ABSC Scheme

Based on MACP-ABSC scheme, we propose OMDAC-ABSC scheme, a novel data access control scheme for fog computing system supporting the computation outsourcing for both signcryptor and designcryptor.

4.2.1. Scheme Description

As shown in Figure 1, our OMDAC-ABSC scheme has five types of entities: the global certificate authority (CA), cloud server, users (including signcryptors and designcryptors), independent attribute authorities (AAs) and fog nodes.

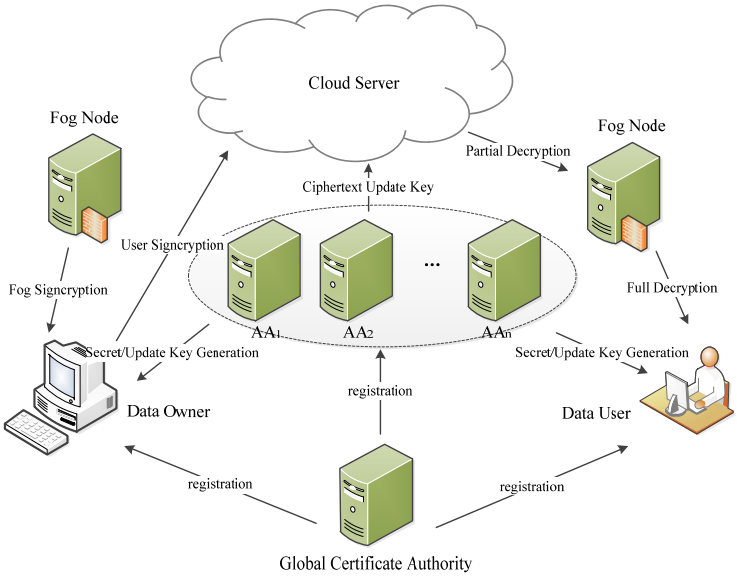

Figure 1. System Architecture.

Global Certificate Authority: The global certificate authority (CA) is fully trusted in the system and generates the public parameters for the system. CA is also responsible for the users' and authorities' registrations. However, CA is not involved in any attribute management and the creations of the secret keys that are associated with attributes. 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 return the ciphertext to the fog node if the user is valid.

Cloud Server: The cloud server is a semi-trusted party and also provides data storage and data access service to 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.

User: Users who are attached to fog nodes and equipped with IoT devices in our system include the signcryptor and designcryptor. When the signcryptor signcrypts a message, he/she can select the signing and encryption predicates over the attributes from multiple authorities and outsource the resulting ciphertext to the cloud server. 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 encryption predicate can decrypt the data.

Attribute Authority: The authority can initialize itself to setup its public and secret keys. To compute the secret keys for users, the authority verifies the user's identity and generates the secret keys according to the user's attributes.

Fog Node: Fog nodes, deployed at the edge of the network, offer a variety of services, such as low latency, location awareness, and real-time applications. Each of them is linked to the cloud server. Fog nodes are also in charge of part of signcryption and designcryption computations. Note that in designcryption phase, only if the data user's attributes satisfy the encryption predicate will the fog nodes partially designcrypt the ciphertext with the proxy secret keys.

The work flow of OMDAC-ABSC scheme is shown in Figure 2. The scheme consists of the following six phases.

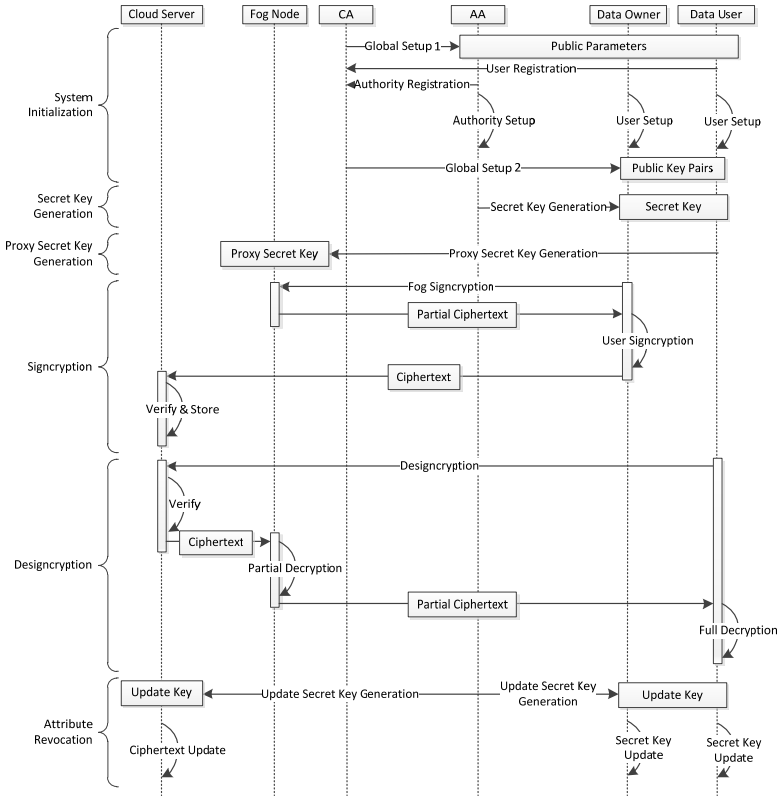

Figure 2. Work flow of OMDAC-ABSC scheme.

(1) System Initialization

In this phase, CA generates the public parameters for the system, and also accepts the registrations of the attribute authorities and the users. The initialization phase contains the following six algorithms:

$GlobalSetup1\left(1^k\right)$. This algorithm is run by CA. Taking as input the security parameter 1^k, the algorithm outputs the public parameters PP.

$UserReg(PP)$. This algorithm is run by CA and data user. Taking as input the public parameters, CA assigns the global identity uid and partial public key PPK_{uid} to the user.

$AuthorityReg(PP)$. This algorithm is run by CA and the attribute authority. Taking as input the public parameters, CA assigns the global identity aid and partial public key PPK_{aid} for the attribute authority.

$UserSetup(PP, PPK_{uid})$. Given the global identity uid, public parameters PP, and partial public key PPK_{uid}, the data user runs $UserSetup(PP, PPK_{uid})$ to initialize himself/herself. The algorithm outputs the public key PK_{uid} and secret key SK_{uid} for the user. Additionally, the public key certificate $cert(uid)$ generated by CA is sent to the user for identity authentication.

$AuthoritySetup(PP, PPK_{aid})$. Given the global identity aid, public parameters PP, and partial public key PPK_{aid}, the attribute authority runs $AuthoritySetup(PP, PPK_{aid})$ to initialize itself. The algorithm outputs the public key PK_{aid} , $PK^1_{uid,aid}$ and secret key SK_{aid} for the attribute authority AA_{aid}.

$GlobalSetup2\left(1^k, PP, \left\{PK_{aid}, PK^1_{uid,aid}\right\}_{U_{uid}\in S_U, AA_{aid}\in S_A}\right)$. This algorithm is run by CA to end the system initialization phase. Taking as input the public parameters PP and authorities' public keys $\left\{PK_{aid}, PK^1_{uid,aid}\right\}_{U_{uid}\in S_U, AA_{aid}\in S_A}$, CA generates the public key $PK_{uid,aid}$ for each pair of user U_{uid} and authority AA_{aid}.

(2) Secret Key Generation

After system initialization, the attribute authority AA_{aid} can verify the user's identity using the public key certificate $cert(uid)$ and then run $SecretKeyGen\left(PP, PK_{aid}, SK_{aid}, PK_{uid}, \widetilde{U}\right)$ algorithm to compute the secret signing and decryption keys for the valid user according to the user's attribute set $\widetilde{U}$.

$SecretKeyGen\left(PP, PK_{aid}, SK_{aid}, PK_{uid}, \widetilde{U}\right)$. The algorithm intakes the public parameters PP, the public key and secret key pair $\{PK_{aid}, SK_{aid}\}$ of the authority AA_{aid}, the public key PK_{uid} and user's attribute set $\widetilde{U}$, outputs the user's secret signing and decryption keys $SK_{uid,aid} = \left\{SK^s_{uid,aid}, SK^d_{uid,aid}\right\}$.

(3) Proxy Secret Key Generation

In this phase, the data user runs $PxSecretKeyGen(SK_{uid}, SK_{uid,aid})$ algorithm to compute the proxy secret signing and decryption keys $PSK_{uid,aid} = \left\{PSK^s_{uid,aid}, PSK^d_{uid,aid}\right\}$ and then sends $PSK_{uid,aid}$ to the fog nodes to outsource the signcryption and designcryption computation overhead.

$PxSecretKeyGen(SK_{uid}, SK_{uid,aid})$. Taking as input the secret key SK_{uid} and secret signing and decryption keys $SK_{uid,aid}$, this algorithm outputs the proxy secret signing and decryption keys $PSK_{uid,aid} = \left\{PSK^s_{uid,aid}, PSK^d_{uid,aid}\right\}$. $PSK_{uid,aid}$ are sent to the fog nodes.

(4) Data Signcryption

To achieve high efficiency, the signcryptor first encrypts the plaintext with a random content key by applying a symmetric encryption algorithm. Then the signcryptor defines the signing and encryption predicates $\mathcal{R}_s$ and $\mathcal{R}_e$, and signcrypts the content secret key with the following two algorithms:

$Fog\_Signcryption\left(PP, \left\{PSK^s_{uid,k}\right\}_{k\in I^s_A}, PK_{uid}, \mathcal{R}_s, \mathcal{R}_e\right)$. This algorithm is performed in the fog nodes. Taking as input the public parameters PP, proxy secret signing key $PSK^s_{uid,k}$ of the attribute

authority AA_k whose attributes are selected for signing, the public key PK_{uid} of signcryptor, the signing and encryption predicates $\mathcal{R}_s, \mathcal{R}_e$, the algorithm outputs part of the ciphertext CT'.

$User\_Signcryption\left(\mathcal{M}, PP, \{PK_{aid}\}_{aid\in I_A^e}, SK_{uid}, CT'\right)$. This algorithm intakes the message to be signcrypted, the public parameters PP, the public key PK_{aid} of attribute authorities whose attributes are selected for encryption, secret key SK_{uid} of signcryptor and partial ciphertext CT', outputs the ciphertext CT and sends CT to the cloud server.

(5) Data Designcryption

When the user queries the ciphertext, the cloud server verifies the user's identifier and returns the ciphertext to the fog node if the user is valid. If the decryption attribute set $\widetilde{U^d}$ satisfies the encryption predicate $\mathcal{R}_e$ embedded in ciphertext, the data user can obtain the plaintext by running $DeSigncryption\left(PP, CT, PK_{uid}, \left\{PSK^d_{uid,k}\right\}_{k\in I_A^e}, SK_{uid}\right)$ algorithm which includes the following three sub-algorithms: $Verify(PP, CT)$ run by any trusted party (public verifiability), $PartialDecryption\left(PP, CT, PK_{uid}, \left\{PSK^d_{uid,k}\right\}_{k\in I_A^e}\right)$ run by fog nodes and $FullDecryption(PP, CT^p, SK_{uid})$ performed by the user. I_A^s (resp. I_A^e) denotes the set of the indexes of the authorities involved in signing (resp. encryption). Note that I_A^s (resp. I_A^e) can be obtained from $\mathcal{R}_s$ (resp. $\mathcal{R}_e$) which is implicitly contained in CT.

$Verify(PP, CT)$. This algorithm takes as input the public parameters PP and ciphertext CT, outputs $\perp$ if CT contains an invalid signature corresponding to the signing predicate $\mathcal{R}_s$ embedded in CT. Otherwise, proceed $Decryption\left(PP, CT, PK_{uid}, \{PSK_{uid,k}\}_{k\in I_A^e}, SK_{uid}\right)$ algorithm as follows:

$Decryption\left(PP, CT, PK_{uid}, \{PSK_{uid,k}\}_{k\in I_A^e}, SK_{uid}\right)$. This algorithm contains two sub-algorithms:

$PartialDecryption\left(PP, CT, PK_{uid}, \left\{PSK^d_{uid,k}\right\}_{k\in I_A^e}\right)$. This algorithm intakes the public parameters PP, the ciphertext CT, the public key PK_{uid} of the user and the proxy secret decryption key $PSK^d_{uid,k}$, outputs the partial decryption result CT^p and returns CT^p to the user.

$FullDecryption(PP, CT^p, SK_{uid})$. Taking as input the public parameters PP, the partial decryption result CT^p and secret key SK_{uid}, the algorithm outputs the final plaintext $\mathcal{M}$ or $\perp$.

(6) Attribute revocation

In this phase, suppose the attribute x of the user U is revoked from AA_k. After randomly chooses a new attribute version key, the authority AA_k distributes the update keys implicitly containing the latest attribute version key to the non-revoked users and cloud server respectively. Only the x-related components of secret keys and ciphertext will be updated.

$UpSecretKeyGen(PK_{uid}, SK_k, SK_{uid,k})$. This algorithm is run by attribute authority AA_k. The algorithm intakes the public key PK_{uid} of non-revoked user U_{uid}, the secret key of AA_k, outputs the signing and decryption update keys $sUK_{uid,x}, dUK_{uid,x}$, and ciphertext update keys cUK, sUK.

$UpSecretKey(SK_{uid,k}, sUK_{uid,x}, dUK_{uid,x})$. This algorithm is run by the non-revoked user U_{uid}. Taking as input the secret signing and decryption key $SK_{uid,k}$, and the signing and decryption update keys $sUK_{uid,x}, dUK_{uid,x}$, the algorithm outputs the updated secret signing and decryption keys.

$UpCiphertext(CT, cUK, sUK)$. This algorithm is run by the cloud server. Taking as input the ciphertext tagged with the revoked attribute, and the ciphertext update keys cUK, sUK, the algorithm outputs the updated ciphertext.

4.2.2. Threat Assumption

Assume CA is fully trusted. The authorities can honestly issue the secret keys for the user and will not collude with the user to access the sensitive data. However, the authorities can be corrupted and disclose the information sent from the data user to the adversary. The fog nodes can also be corrupted

and leak the information such as proxy secret keys to the adversary. The cloud server is semi-trusted. It will execute the protocol in general but will leak the signcrypted data to some malicious users and get illegal access privileges. The data users (including the signcryptor and designcryptor) are malicious and can collude with other users and even the cloud server and fog nodes to sign or decrypt the unauthorized data.

4.2.3. Security Requirements

Following [12,14], the confidentiality, unforgeability and signcryptor privacy of OMDAC-ABSC scheme are presented in Definitions 8–10 as follows by defining the security games between a challenger and an adversary $\mathcal{A}$. Then in Definitions 11 and 12, we provide the definitions of collusion resistance and attribute revocation security.

Definition 8. *Indistinguishability of ciphertext under selective encryption predicate and adaptively chosen ciphertext attack (IND-sEP-CCA2).*

The scheme is $\left(T, q_{sk}, q_{psk}, q_{SC}, q_{DS}, \epsilon\right)$-IND-sEP-CCA2 secure if for any PPT adversary $\mathcal{A}$ which runs in time at most T and makes at most q_{sk} *SecretKey* queries, q_{psk} *Proxy SecretKey* queries, q_{SC} *Signcryption* queries, and q_{DS} *DeSigncryption* queries, the advantage $Adv_{\mathcal{A}}^{IND-sEP-CCA2}$ of $\mathcal{A}$ in the following game with a challenger $\mathcal{C}$ is at most ϵ.

Init. $\mathcal{A}$ specifies the space of attributes and the set of corrupted authorities. $\mathcal{A}$ submits the challenge encryption predicate $\mathcal{R}_e^* = (\mathrm{M}_e^*, \rho_e^*)$ over encryption attributes that will be used to encrypt the challenge ciphertext. Note that the adversary cannot decrypt the challenge ciphertext with any secret decryption keys queried from *SecretKey* queries and the keys directly generated from the corrupted authorities.

Setup. The challenger runs the algorithms in system initialization phase to generate the public parameters, and the pairs of public key and the secret key of the attribute authorities. Then the challenger sends the public keys to the adversary. For the corrupted authorities, the challenger sends the secret keys to the adversary.

Phase 1. In this phase, the challenger $\mathcal{C}$ answers the queries from $\mathcal{A}$ as follows:

SecretKey query $\mathcal{O}^{sk}\left(\widetilde{U}, AA_k, uid\right)$. $\mathcal{A}$ 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_k. $\widetilde{U^d}$ does not satisfy $\mathcal{R}_e^*$ together with any keys that can be obtained from corrupted authorities. The challenger runs *SecretKeyGen* and returns the secret key to the adversary.

Proxy SecretKey query $\mathcal{O}^{psk}\left(\widetilde{U}, AA_k, uid\right)$. $\mathcal{A}$ can adaptively query the proxy secret key for a user U with identity uid. The challenger runs *PxSecretKeyGen* and returns the proxy secret key to the adversary.

Signcryption query $\mathcal{O}^{SC}(\mathcal{M}, \mathcal{R}_s, \mathcal{R}_e)$. Upon receiving a message $\mathcal{M} \in \mathbb{G}_T$, signing and encryption predicts $\mathcal{R}_s, \mathcal{R}_e$, the challenger $\mathcal{C}$ selects a signing attribute set $\widetilde{U^s}$ such that $\mathcal{R}_s\left(\widetilde{U^s}\right) = 1$ 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 $\mathcal{A}$ if $\mathcal{R}_e\left(\widetilde{U^d}\right) = 1$ and CT contains a valid signature corresponding to the signing predicate $\mathcal{R}_s$, where $\mathcal{R}_e$ and $\mathcal{R}_s$ are implicitly contained in CT.

Challenge. $\mathcal{A}$ submits two messages $\mathcal{M}_0, \mathcal{M}_1$ with the same length and signing predicate $\mathcal{R}_s^* = (\mathrm{M}_s^*, \rho_s^*)$ to the challenger. $\mathcal{C}$ selects a signing attribute set $\widetilde{U^s}$ satisfying $\mathcal{R}_s^*\left(\widetilde{U^s}\right) = 1$. The challenger randomly chooses a bit $\lfloor \in \{0,1\}$ and runs the *Signcryption* algorithm to signcrypt the message $\mathcal{M}_{\lfloor}$ 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 $\mathcal{O}^{DS}$ with the challenge ciphertext CT^* obtained in Challenge phase and attribute set $\widetilde{U^d}$ such that $\mathcal{R}_e^*\left(\widetilde{U^d}\right) = 1$.

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

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

Definition 9. *Existential unforgeability under selective signing predicate and adaptively chosen message attack (EUF-sSP-CMA).*

The proposed scheme is $\left(T, q_{sk}, q_{psk}, q_{SC}, q_{DS}, \epsilon\right)$-EUF-sSP-CMA secure if for any PPT adversary $\mathcal{A}$ which runs in time at most T and makes at most q_{sk} *SecretKey* queries, q_{psk} *Proxy SecretKey* queries, q_{SC} *Signcryption* queries, and q_{DS} *DeSigncryption* queries, the advantage $Adv_{\mathcal{A}}^{EUF-sSP-CMA}$ of $\mathcal{A}$ in the following game with a challenger $\mathcal{C}$ is at most ϵ.

Init. $\mathcal{A}$ specifies the space of attributes and a set of corrupted authorities, and then submits the challenge signing predicate $\mathcal{R}_s^* = (\mathrm{M}_s^*, \rho_s^*)$ over signing attributes that will be used to forge the ciphertext. Note that the adversary cannot sign the plaintext under the signing predicate $\mathcal{R}_s^*$ with any secret signing keys queried from *SecretKey* queries and the keys directly generated from the corrupted authorities.

Setup, *Proxy SecretKey query*, *Signcryption query* and *DeSigncryption query* are the same as Definition 8.

SecretKey query $\mathcal{O}^{sk}\left(\widetilde{U}, AA_k, uid\right)$. $\mathcal{A}$ can adaptively query the secret key for a user U with a set of attributes $\widetilde{U} = \widetilde{U^d} \cup \widetilde{U^s}$ to the authority AA_k. $\widetilde{U^s}$ does not satisfy $\mathcal{R}_s^*$ together with any keys that can be obtained from corrupted authorities. The challenger runs *SecretKeyGen* and returns the secret key to the adversary.

Forgery. $\mathcal{A}$ outputs the forgery ciphertext CT^* for the selective signing predicate $\mathcal{R}_s^*$ and an arbitrary encryption predicate $\mathcal{R}_e^*$.

$\mathcal{A}$ wins the game if CT^* is a valid ciphertext and $\mathcal{A}$ has never issued $\mathcal{O}^{SC}(\mathcal{M}, \mathcal{R}_s^*, \mathcal{R}_e^*)$. The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{EUF-sSP-CMA} = Pr[\mathcal{A}\text{ wins}]$.

Note that in our scheme, the fog nodes can be corrupted. In this case, the proxy secret keys sent from the users might be obtained by the adversary. This kind of attack is captured by the proxy secret key query $\mathcal{O}^{psk}\left(\widetilde{U}, AA_k, uid\right)$, which makes the access control scheme proven secure in our security model have a wider spectrum of applications.

Definition 10. *Signcryptor Privacy.*

It is required that the signature of the proposed scheme reveals nothing about the attributes of the data owner except that the attributes satisfy the signing predicate. We define signcryptor privacy as a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Assume the public parameters PP and public and secret key pairs $\{PK_k, SK_k\}_{I_A}$ of attribute authorities are given to $\mathcal{A}$. $\mathcal{A}$ submits two signing attribute sets $\widetilde{U_0^s}, \widetilde{U_1^s}$ satisfying $\mathcal{R}_s\left(\widetilde{U_0^s}\right) = \mathcal{R}_s\left(\widetilde{U_1^s}\right) = 1$ to the challenger. The challenger then chooses a bit $\lfloor \xleftarrow{R} \{0,1\}$ and signcrypts the plaintext $\mathcal{M}$ with the signing and encryption predicates $\mathcal{R}_s, \mathcal{R}_e$, and secret signing key $SK_{uid,k}^{s,\lfloor}$ for $\widetilde{U_{\lfloor}^s}$. $\mathcal{C}$ sends the ciphertext $CT_{\lfloor}$ to $\mathcal{A}$. $\mathcal{A}$ then outputs a guess bit $\lfloor'$ on $\lfloor$. $\mathcal{A}$ wins the game if $\lfloor' = \lfloor$. We say OMDAC-ABSC scheme satisfies signcryptor privacy if for any adversary $\mathcal{A}$,

$$Pr\left[\begin{array}{c} \lfloor' = \lfloor \ : \end{array}\begin{array}{c} PP \leftarrow GlobalSetup1\left(1^k\right) \\ \{PK_k, SK_k\}_{I_A} \leftarrow AuthoritySetup(PP, PPK_k) \\ \left(\widetilde{U_0^s}, \widetilde{U_1^s}, \mathcal{M}, \mathcal{R}_s, \mathcal{R}_e\right) \leftarrow \mathcal{A}\left(PP, \{PK_k, SK_k\}_{I_A}\right) \\ \mathcal{R}_s\left(\widetilde{U_0^s}\right) = 1 = \mathcal{R}_s\left(\widetilde{U_1^s}\right) \\ \lfloor \overset{R}{\leftarrow} \{0,1\} \\ CT_{\lfloor} \leftarrow \mathcal{C}\left(\mathcal{M}, PP, \mathcal{R}_s, \mathcal{R}_e, \widetilde{U_{\lfloor}^s}, \left\{SK_{uid,k}^{s,\lfloor}\right\}_{k \in I_A^s}\right) \\ \lfloor' \leftarrow \mathcal{A}\left(PP, CT_{\lfloor}, \{PK_k, SK_k\}_{I_A}\right) \end{array}\right] = \frac{1}{2}$$

Definition 11. *Collusion Resistance.*

OMDAC-ABSC scheme is secure against collusion attack of two or more communication entities (e.g., data users, fog nodes, and cloud server) if there does not exist a set of polynomial time adversaries that can sign the plaintext (collusion resistance of signing) or decrypt the ciphertext (collusion resistance of decryption) by cooperating with each other when none of adversaries is authorized to sign or decrypt the data.

Definition 12. *Suppose the attribute x is revoked.*

Forward Security. If x is the signing attribute, then OMDAC-ABSC scheme supports forward revocation security if the newly joined user can successfully sign the plaintext with the x-corresponding signing attribute set. Otherwise, the forward revocation security guarantees if each newly joined user can decrypt x-corresponding ciphertext if the decryption attributes of the user satisfy the encryption predicate contained in the ciphertext.

Backward Security. If x is the signing attribute, then OMDAC-ABSC scheme supports backward revocation security if the updated ciphertext cannot be reversed back to the non-revoked state while maintaining the verification algorithm holds. Otherwise, the backward revocation security guarantees if the attribute revoked user cannot decrypt the x-corresponding ciphertext as a non-revoked user.

5. Construction of OMDAC-ABSC Scheme

In this section, we propose the construction of OMDAC-ABSC scheme in detail. The notations of the scheme are listed in Appendix A.

5.1. System Initialization

5.1.1. System Setup 1

$GlobalSetup1\left(1^k\right)$. Taking as input a security parameter 1^k, the algorithm outputs the public parameters PP as follows.

(1) Generate a bilinear group $\mathcal{GG}\left(1^k\right) \rightarrow (e, p, \mathbb{G}, \mathbb{G}_T)$, where the prime p is the order of group $\mathbb{G}$. Let g, θ be the random generators of $\mathbb{G}$. Randomly select γ_1, γ_2, $\{k_0, k_1, \ldots, k_l\}$, $\{V_1, V_2 \ldots, V_{\ell_m}\}$ from $\mathbb{G}$. Choose three cryptographic collision resistant hash functions $H_1 : \mathbb{G} \rightarrow \mathbb{Z}_p^*$, $H_2 : \{0,1\}^* \rightarrow \{0,1\}^l$ and $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.

(2) CA generates a pair of keys $\{sk_{CA}, vk_{CA}\}$ for signing and verification in identity authentication.

(3) Output $PP = \{g, \theta, \gamma_1, \gamma_2, \{k_0, k_1, \ldots, k_l\}, \{V_1, V_2 \ldots, V_{\ell_m}\}\}$ as the system public parameter. CA accepts both user registration $UserReg(PP)$ and authority registration $AuthorityReg(PP)$.

$UserReg(PP)$. CA verifies user U's identity information then runs this algorithm to register U. CA selects a unique identity number uid and sends $PPK_{uid} = \left\{g^{s_{uid}}, g^{d_{uid}}, \{V_i^{s_{uid}}\}_{i\in[\ell_m]}\right\}$ as the partial public key to user. s_{uid} and d_{uid} are kept secret in the system.

$AuthorityReg(PP)$. CA verifies the identity information of the authority then runs this algorithm to register the authority. CA selects a unique identity number $aid \in [1, N_A]$, then selects α_{aid} and publishes the partial public key $PPK_{aid} = \Delta_{aid} = e(g,g)^{\alpha_{aid}}$ to AA_{aid}.

$UserSetup(PP, PPK_{uid})$. Given the global identity uid, the user runs $UserSetup(PP, PPK_{uid})$ to initialize itself and compute the public key PK_{uid} and secret key SK_{uid} as follows.

1. Set $SK_{uid} = z_{uid}$ where $z_{uid} \xleftarrow{R} \mathbb{Z}_p$.
2. Set $PK_{uid} = \left\{g^{s_{uid}}, g^{d_{uid}}, g^{1/z_{uid}}, \theta^{z_{uid}}, g^{z_{uid}}, \{V_i^{s_{uid}}\}_{i\in[\ell_m]}\right\}$.
3. CA sets $cert(uid) = Sign_{sk_{CA}}(uid, PK_{uid})$ as the public key certificate.

$AuthoritySetup(PP, PPK_{aid})$. Each authority AA_{aid} runs this algorithm to initialize itself and compute the public key $PK_{aid}, PK^1_{uid,aid}$ and secret key SK_{aid} as follows:

(1) Set $SK_{aid} = \{\beta_{aid}, \gamma_{aid}, \{\varphi_x\}_{x\in\widetilde{AA_{aid}}}\}$, where $\beta_{aid}, \gamma_{aid}, \varphi_x \xleftarrow{R} \mathbb{Z}_p$.

(2) Set $PK_{aid} = \left\{\Delta_{aid}, X_{aid}, Y_{aid}, Z_{aid}, \{A_x\}_{x\in\widetilde{AA_{aid}}}\right\}$, where $A_x = g^{\varphi_x}, X_{aid} = g^{1/\beta_{aid}}$, $Y_{aid} = \theta^{1/\beta_{aid}}, Z_{aid} = \theta^{1/\gamma_{aid}}$.

(3) Set $PK^1_{uid,aid} = g^{1/(\gamma_{aid} z_{uid})}$ for each user $U_{uid} \in S_U$.

5.1.2. System Setup 2

$GlobalSetup2\left(1^k, PP, \{PK_{aid}, PK^1_{uid,aid}\}_{U_{uid}\in S_U, AA_{aid}\in S_A}\right)$. Taking as input the public parameters PP and authorities' public keys $\{PK_{aid}, PK^1_{uid,aid}\}_{U_{uid}\in S_U, AA_{aid}\in S_A}$, CA generates the public key $PK_{uid,aid}$ for each pair of user U_{uid} and authority AA_{aid} as follows:

For $U_{uid} \in S_U, AA_{aid} \in S_A$, $PK_{uid,aid} = \{PK^1_{uid,aid}, PK^2_{uid,aid}, PK^3_{uid,aid}\}$, where $PK^2_{uid,aid} = (PK^1_{uid,aid})^{\alpha_{aid}} Z_{aid}^{d_{uid}} = g^{\alpha_{aid}/(\gamma_{aid} z_{uid})} \theta^{d_{uid}/\gamma_{aid}}$ and $PK^3_{uid,aid} = X_{aid}^{\alpha_{aid}} Y_{aid}^{s_{uid}} = g^{\alpha_{aid}/\beta_{aid}} \theta^{s_{uid}/\beta_{aid}}$.

5.2. Secret Key Generation

AA_{aid} runs the secret key generation algorithm $SecretKeyGen$ to generate the secret signing and decryption keys for the user U_{uid}.

$SecretKeyGen(PP, PK_{aid}, SK_{aid}, PK_{uid}, \widetilde{U})$. AA_{aid} first verifies the user's $cert(uid)$ with verification key vk_{CA}. If the user is a legal user, AA_{aid} computes the user's secret signing and decryption keys $SK_{uid,aid} = \{SK^s_{uid,aid}, SK^d_{uid,aid}\}$ as:

(1) $SK^s_{uid,aid} = \{K^s_{uid,aid} = (PK^3_{uid,aid})^{\beta_{aid}} = g^{\alpha_{aid}} \theta^{s_{uid}}, \{F^s_{uid,x} = (g^{s_{uid}})^{\varphi_x} = A_x^{s_{uid}}\}_{x\in\widetilde{U^s}\cap\widetilde{AA_{aid}}}\}$.

(2) $SK^d_{uid,aid} = \{K^d_{uid,aid} = (PK^2_{uid,aid})^{\gamma_{aid}} = g^{\alpha_{aid}/z_{uid}} \theta^{d_{uid}}, \{F^d_{uid,x} = (g^{d_{uid}})^{\varphi_x} = A_x^{d_{uid}}\}_{x\in\widetilde{U^d}\cap\widetilde{AA_{aid}}}\}$.

5.3. Proxy Secret Key Generation

Each user U_{uid} runs the $PxSecretKeyGen(SK_{uid}, SK_{uid,aid})$ to generate the proxy secret key $PSK_{uid,aid} = \left\{PSK^s_{uid,aid}, PSK^d_{uid,aid}\right\}$ as:

(1) $PSK^s_{uid,aid} = \left\{PS_{uid,aid} = \left(K^s_{uid,aid}\right)^{z_{uid}}, PV_{uid} = g^{z_{uid} s_{uid}}, \left\{PF^1_{uid,x} = \left(F^s_{uid,x}\right)^{z_{uid}}, PF^2_{uid,x} = (A_x)^{z_{uid}}\right\}_{x\in\widetilde{U^s}\cap\widetilde{AA_{aid}}}, \left\{V_i^{z_{uid}}, V_i^{s_{uid} z_{uid}}\right\}_{i\in[\ell_m]}\right\}$.

(2) $PSK^d_{uid,aid} = SK^d_{uid,aid}$.

The transformed secret keys $\left\{PSK_{uid,aid}\right\}$ are sent to the fog node.

5.4. Data Signcryption

The data owner first encrypts the data component with a content secret key k by using symmetric encryption algorithm En_k, then it runs *Signcryption* to signcrypt the secret key. *Signcryption* contains two phases: fog signcrypt *Fog_Signcryption* and user signcrypt *User_Signcryption*.

$Signcryption\left(\mathcal{M}, PP, \left\{PSK^s_{uid,k}\right\}_{k\in I^s_A}, PK_{uid}, \{PK_{aid}\}_{aid\in I^e_A}, SK_{uid}, \mathcal{R}_s, \mathcal{R}_e\right)$. Assume that $\mathcal{R}_s(\mathrm{M}_s, \rho_s)$ (resp. $\mathcal{R}_{e,j}(\mathrm{M}_e, \rho_e)$) is the signing predicate (resp. encryption predicate) over all the attributes selected from the set of attribute authorities I^s_A (resp. I^e_A), where M_s (resp. M_e) is a $\ell_s \times n_s$, $\ell_s \le \ell_m$ (resp. $\ell_e \times n_e$) matrix with row labeling function $\rho_s : [\ell_s] \to \mathbb{Z}_p$ (resp. $\rho_e : [\ell_e] \to \mathbb{Z}_p$). Note that we remove the limitation that ρ_s (resp. ρ_e) should be an injective function (i.e., an attribute can associate with more than one rows of M_s (resp. M_e)). Let M^i_s (resp. M^i_e) be the ith row of the matrix M_s (resp. M_e). Assume the signing attribute set is $\widetilde{U^s}$ and $\mathcal{R}_s\left(\widetilde{U^s}\right) = 1$. The algorithm contains two phases as follows:

(1) $Fog\_Signcryption\left(PP, \left\{PSK^s_{uid,k}\right\}_{k\in I^s_A}, PK_{uid}, \mathcal{R}_s, \mathcal{R}_e\right)$. This algorithm is performed in the fog node FD as follows:

- It first computes a vector $\vec{a} = (a_1, a_2, \ldots, a_{\ell_s}) \in \mathbb{Z}^{\ell_s}_p$ such that $\vec{a}\cdot \mathrm{M}_s = \vec{1}$ since $\mathcal{R}_s\left(\widetilde{U^s}\right) = 1$. Note that $a_i = 0$ for all i where $\rho_s(i) \notin \widetilde{U^s}$. Then the algorithm chooses $\vec{b} = (b_1, b_2, \ldots, b_{\ell_s}) \in \mathbb{Z}^{\ell_s}_p$ such that $\sum_{i\in[\ell_s]} b_i \mathrm{M}^i_s = \vec{0}$.
- The algorithm randomly chooses $s'_{uid} \xleftarrow{R} \mathbb{Z}^*_p$ and re-randomize the proxy secret key $PSK^s_{uid,aid}$ as

$$PS_{uid,k} = PS_{uid,k}(\theta^{z_{uid}})^{s'_{uid}} = g^{\alpha_k z_{uid}}\theta^{z_{uid}s''_{uid}},$$

$$PV_{uid} = PV_{uid}(g^{z_{uid}})^{s'_{uid}} = g^{z_{uid}s''_{uid}},$$

$$\left\{PF_{uid,x} = PF^1_{uid,x}\left(PF^2_{uid,x}\right)^{s'_{uid}} = A_x^{z_{uid}s''_{uid}}\right\}_{x\in\widetilde{U^s}},$$

$$V_i^{z_{uid}s''_{uid}} = V_i^{s_{uid}z_{uid}}\left(V_i^{z_{uid}}\right)^{s'_{uid}}, \text{ where } s''_{uid} = s_{uid} + s'_{uid}.$$

- The fog node randomly picks $w' \xleftarrow{R} \mathbb{Z}^*_p$. Then it selects $\left\{r'_1, r'_2, \ldots, r'_{\ell_e}\right\} \xleftarrow{R} \mathbb{Z}_p$, $\left\{\lambda'_1, \lambda'_2, \ldots, \lambda'_{\ell_e}\right\} \xleftarrow{R} \mathbb{Z}_p$, and computes the following terms: $\left\{C'_{2,i} = \theta^{\lambda'_i}A^{-r'_i}_{\rho_e(i)}, D'_i = g^{r'_i}\right\}_{i\in[\ell_e]}$, $\left\{S'_{1,i} = PV^{a_i}_{uid}g^{z_{uid}b_i} = g^{z_{uid}(a_i s''_{uid}+b_i)}\right\}_{i\in[\ell_s]}$. $S'_2 = \left(\prod_{I^s_A} PS_{uid,k}\right)\left(\prod_{i\in[\ell_s]}\left(PF_{uid,\rho_s(i)}V_i^{z_{uid}s''_{uid}}\right)^{a_i}\left(PF^2_{uid,\rho_s(i)}V_i^{z_{uid}}\right)^{b_i}\right)$.

FD outputs the partially signcrypted ciphertext $CT' = \left\{w', \left\{C'_{2,i}, D'_i, \lambda'_i, r'_i\right\}_{i\in[\ell_e]}, \left\{S'_{1,i}\right\}_{i\in[\ell_s]}, S'_2\right\}$ to the user.

(2) $User\_Signcryption\left(\mathcal{M}, PP, \{PK_{aid}\}_{aid\in I^e_A}, SK_{uid}, CT'\right)$. The user randomly picks $w \xleftarrow{R} \mathbb{Z}^*_p$ and $\{r_1, r_2, \ldots, r_{\ell_e}\} \xleftarrow{R} \mathbb{Z}_p$. Then the user computes $\lambda_i = \mathrm{M}^i_e\vec{\varepsilon}$ where $\vec{\varepsilon} = (w, \varepsilon_2, \ldots, \varepsilon_{n_e}) \in \mathbb{Z}^{n_e}_p$. The algorithm computes the following terms:

$C_0 = \mathcal{M}\prod_{k\in I^e_A}\Delta_k{}^w$, $C_1 = g^w$, $\left\{C''_{2,i} = \lambda_i - \lambda'_i, D''_i = r_i - r'_i\right\}_{i\in[\ell_e]}$, $\pi = H_1(C_1)$, $C_3 = (\gamma_1\gamma_2{}^\pi)^w$, $\left\{S_{1,i} = \left(S'_{1,i}\right)^{1/z_{uid}}\right\}_{i\in[\ell_s]}$, $H_2\left(\prod_{i\in[\ell_s]} S_{1,i}, tt, \mathcal{R}_s, \mathcal{R}_e\right) = (c_1, c_2, \ldots, c_l) \in \{0,1\}^l$, $H_3(C_0, C_1, C_3, \mathcal{R}_s, \mathcal{R}_e) = \beta$ and $S_2 = (S'_2)^{1/z_{uid}}\left(k_0\prod_{i=1}^l k_i{}^{c_i}\right)^w C_3^\beta$.

The ciphertext is $CT = \left\{C_0, C_1, \left\{C'_{2,i}, C''_{2,i}, D'_i, D''_i\right\}_{i\in[\ell_e]}, \{S_{1,i}\}_{i\in[\ell_s]}, S_2, tt\right\}$.

5.5. Data Designcryption

If the owner's attributes satisfy the signing predicate 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.

$DeSigncryption\left(PP, CT, PK_{uid}, \left\{PSK^{d}_{uid,k}\right\}_{k\in I^{e}_{A}}, SK_{uid}\right)$. 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 $\mathcal{R}_e\left(\widetilde{U^d}\right) \neq 1$, the algorithm returns $\perp$. Otherwise, the algorithm performs as follows. Note that I^{s}_{A} (resp. I^{e}_{A}) can be obtained from the implicitly contained predicate $\mathcal{R}_s$ (resp. $\mathcal{R}_e$).

$Verify(PP, CT)$. This verification algorithm can be performed in FD or other trusted third party since it only takes the ciphertext and public parameter PP as the input.

The algorithm samples $\{\tau_2, \tau_3, \ldots, \tau_{n_s}\} \xleftarrow{R} \mathbb{Z}^{*}_{p}$ and computes $\varpi_i = (1, \tau_2, \tau_3, \ldots, \tau_{n_s}) \cdot \mathrm{M}^{i}_{s}$, where $i \in [\ell_s]$. $H_1(C_1) = \pi$ and $H_2\left(\prod_{i\in[\ell_s]} S_{1,i}, tt, \mathcal{R}_s, \mathcal{R}_e\right) = (c_1, c_2, \ldots, c_l)$, and $H_3(C_0, C_1, C_3, \mathcal{R}_s, \mathcal{R}_e) = \beta$. Then the algorithm checks the validity of the ciphertext using the following equation:

$\prod_{I^{s}_{A}} \Delta_k = \frac{e(S_2, g)}{e\left(k_0 \prod_{i=1}^{l} k_i{}^{c_i}, C_1\right) e\left((\gamma_1\gamma_2{}^{\pi})^{\beta}, C_1\right)\left(\prod_{i=1}^{\ell_s} e\left(A_{\rho_s(i)} V_i \theta^{\varpi_i N^{s}_{A}}, S_{1,i}\right)\right)}$, where $N^{s}_{A} = \left|I^{s}_{A}\right|$.

If it is invalid, return $\perp$, otherwise, proceed $Decryption\left(PP, CT, PK_{uid}, \left\{PSK_{uid,k}\right\}_{k\in I^{e}_{A}}, SK_{uid}\right)$ algorithm as follows:

- $PartialDecryption\left(PP, CT, PK_{uid}, \left\{PSK^{d}_{uid,k}\right\}_{k\in I^{e}_{A}}\right)$. If the user's attributes satisfy the encryption predicate, the cloud server sends the ciphertext to the FD. FD chooses a set of constants $\vec{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_{\ell_e}) \in \mathbb{Z}^{\ell_e}_{p}$ such that $\sum_{i=1}^{\ell_e} \sigma_i \mathrm{M}^{i}_{e} = \vec{1}$, where $\sigma_i = 0$ for all i where $\rho_e(i) \notin \widetilde{U^d}$. Then it computes: $CT_x = \frac{\prod_{k\in I^{e}_{A}} e\left(K^{d}_{uid,k}, C_1\right)}{\prod_{k\in I^{e}_{A}} \prod_{i\in I_{A_k}} \left[e\left(C'_{2,i}\theta^{C''_{2,i}} A^{-D''_i}_{\rho_e(i)}, g^{d_{uid}}\right) e\left(D'_i g^{D''_i}, F^{d}_{uid,\rho_e(i)}\right)\right]^{\sigma_i N^{e}_{A}}}$, where I_{A_k} is defined as $I_{A_k} = \left\{i : \rho_e(i) \in \widetilde{AA_k}\right\}$. FD sends $CT^{p} = \{C_0, CT_x\}$ to the user.
- $FullDecryption(PP, CT^{p}, SK_{uid})$. This algorithm is performed on the user side. After receiving CT^{p}, the data user recovers the message $\mathcal{M}$ as: $\mathcal{M} = \frac{C_0}{(CT_x)^{z_{uid}}}$.

Correctness

Assume the identity of signcryptor (data owner) is do. If $\left|\widetilde{tt} - tt\right| \leq thre_{tt}$ and $\mathcal{R}_e\left(\widetilde{U^d}\right) = 1$, then the ciphertext can be verified and decrypted as explained subsequently.

$$
\begin{aligned}
S_2 &= (S'_2)^{1/z_{do}} \left(k_0 \prod_{i=1}^{l} k_i{}^{b_i}\right)^{w} C_3^{\beta} \\
&= \left(\prod_{I^{s}_{A}} g^{\alpha_k} \theta^{s''_{do}}\right) \left(\prod_{i\in[\ell_s]} \left(A^{s''_{do}}_{\rho_s(i)} V_i^{s''_{do}}\right)^{a_i} \left(A_{\rho_s(i)} V_i\right)^{b_i}\right) \left(k_0 \prod_{i=1}^{l} k_i{}^{c_i}\right)^{w} (\gamma_1\gamma_2{}^{\pi})^{w\beta} \\
&= \left(\prod_{I^{s}_{A}} g^{\alpha_k}\right) \theta^{s''_{do} N^{s}_{A}} \left(\prod_{i\in[\ell_s]} \left(A_{\rho_s(i)} V_i\right)^{a_i s''_{do} + b_i}\right) \left(k_0 \prod_{i=1}^{l} k_i{}^{c_i}\right)^{w} (\gamma_1\gamma_2{}^{\pi})^{w\beta}
\end{aligned}
$$

Since $\vec{a} \cdot \mathrm{M}_s = \vec{1}$ and $\sum_{i\in[\ell_s]} b_i \mathrm{M}^{i}_{s} = \vec{0}$, we have

$\sum_{i=1}^{\ell_s} \omega_i \left(s''_{do} a_i + b_i\right) = \sum_{i=1}^{\ell_s} (1, \tau_2, \tau_3, \ldots, \tau_{n_s}) \cdot \mathrm{M}_s^i \left(s''_{do} a_i + b_i\right) = (1, \tau_2, \tau_3, \ldots, \tau_{n_s}) s''_{do} \sum_{i=1}^{\ell_s} \mathrm{M}_s^i a_i + (1, \tau_2, \tau_3, \ldots, \tau_{n_s}) \sum_{i=1}^{\ell_s} \mathrm{M}_s^i b_i = s''_{do}$. Thus we have

$$\begin{aligned}
&\frac{e(S_2, g)}{e\left(k_0 \prod_{i=1}^{l} {k_i}^{c_i}, C_1\right) e\left((\gamma_1\gamma_2{}^{\pi})^{\beta}, C_1\right) \left(\prod_{i=1}^{\ell_s} e\left(A_{\rho_s(i)} V_i \theta^{\omega_i N_A^s}, S_{1,i}\right)\right)} \\
&= \frac{e\left(\left(\prod_{I_A^s} g^{\alpha_k}\right) \theta^{s''_{do} N_A^s} \left(\prod_{i \in [\ell_s]} \left(A_{\rho_s(i)} V_i\right)^{a_i s''_{do} + b_i}\right) \left(k_0 \prod_{i=1}^{l} {k_i}^{c_i}\right)^{w} (\gamma_1\gamma_2{}^{\pi})^{w\beta}, g\right)}{e\left(k_0 \prod_{i=1}^{l} {k_i}^{c_i}, g^{w}\right) e\left((\gamma_1\gamma_2{}^{\pi})^{\beta}, g^{w}\right) \left(\prod_{i=1}^{\ell_s} e\left(A_{\rho_s(i)} V_i, g^{a_i s''_{do} + b_i}\right)\right) \left(\prod_{i=1}^{\ell_s} e\left(\theta^{\omega_i}, g^{a_i s''_{do} + b_i}\right)\right)^{N_A^s}} \\
&= \frac{e\left(\left(\prod_{I_A^s} g^{\alpha_k}\right) \theta^{s''_{do} N_A^s}, g\right)}{\left(e(\theta, g)^{s''_{do}}\right)^{N_A^s}} = \prod_{I_A^s} \Delta_k
\end{aligned}$$

This demonstrates the correctness of $Verify$ algorithm. Assume the identity of designcryptor (data user) is uid. If $\sum_{i=1}^{\ell_e} \sigma_i \lambda_i = w$, then

$$\begin{aligned}
\prod_{k \in I_A^e} \prod_{i \in I_{A_k}} &\left[e\left(C'_{2,i} \theta^{C''_{2,i}} A_{\rho_e(i)}^{-D''_i}, g^{d_{uid}}\right) e\left(D'_i g^{D''_i}, F^d_{uid, \rho_e(i)}\right)\right]^{\sigma_i N_A^e} \\
&= \prod_{k \in I_A^e} \prod_{i \in I_{A_k}} \left[e\left(\theta^{\lambda_i} A_{\rho_e(i)}^{-r_i}, g^{d_{uid}}\right) e\left(g^{r_i}, F^d_{uid, \rho_e(i)}\right)\right]^{\sigma_i N_A^e} \\
&= \prod_{k \in I_A^e} \prod_{i \in I_{A_k}} \left[e\left(\theta^{\lambda_i}, g^{d_{uid}}\right)\right]^{\sigma_i N_A^e} = e(\theta, g)^{w N_A^e d_{uid}}
\end{aligned}$$

Hence $CT_x = \dfrac{\prod_{k \in I_A} e\left(K^d_{uid,k}, C_1\right)}{\prod_{k \in I_A^e} \prod_{i \in I_{A_k}} \left[e\left(C'_{2,i} \theta^{C''_{2,i}} A_{\rho_e(i)}^{-D''_i}, g^{d_{uid}}\right) e\left(D'_i g^{D''_i}, F^d_{uid, \rho_e(i)}\right)\right]^{\sigma_i N_A^e}} = \dfrac{\prod_{k \in I_A^e} e\left(g^{\alpha_k / z_{uid}} \theta^{d_{uid}}, g^{w}\right)}{e(\theta, g)^{w N_A^e d_{uid}}} = \prod_{k \in I_A^e} e(g^{\alpha_k}, g)^{w / z_{uid}} = \prod_{k \in I_A^e} {\Delta_k}^{w / z_{uid}}$ and $\dfrac{C_0}{(CT_x)^{z_{uid}}} = \dfrac{\mathcal{M} \prod_{k \in I_A^e} {\Delta_k}^{w}}{\prod_{k \in I_A^e} {\Delta_k}^{w}} = \mathcal{M}$. This exhibits the correctness of *Decryption* algorithm.

5.6. Attribute Revocation

Suppose the attribute x of user U is revoked from AA_k.

$UpSecretKeyGen(PK_{uid}, SK_k, SK_{uid,k})$. AA_k randomly chooses a new attribute version key $\varphi'_x \xleftarrow{R} \mathbb{Z}_p$ and computes the updated attribute public key $A'_x = g^{\varphi'_x}$. AA_j sets $dUK_{uid,x} = g^{d_{uid}(\varphi'_x - \varphi_x)}$, $sUK_{uid,x} = g^{s_{uid}(\varphi'_x - \varphi_x)}$ for the non-revoked users to update their secret decryption and signing keys.

If there exists i such that $\rho_e(i) = x$, namely the attribute x of AA_k is selected as the encryption attribute, then AA_k queries D'_i where $\rho_e(i) = x$. Then it computes $cUK = \left\{cUK_i = \left(D'_i\right)^{\varphi_x - \varphi'_x}\right\}_{\rho_e(i) = x}$, and sets $sgUK = \perp$.

Otherwise, if x is selected as the signing attribute, AA_k sets $cUK = \perp$ and $sgUK = \prod_{i=1}^{\mathcal{L}} {S_{1,i}}^{\varphi'_x - \varphi_x}$, where $\mathcal{L}$ is the set consisting of all the rows that $\rho_s(i) = x$.

AA_k sends ciphertext update keys cUK, sUK to the cloud server to update the corresponding ciphertext.

$UpSecretKey(SK_{uid,k}, sUK_{uid,x}, dUK_{uid,x})$. Upon receiving the update keys $sUK_{uid,x}$ and $dUK_{uid,x}$, the non-revoked user $U_{uid} \neq U$ then update his/her secret signing key or decryption key as follows:

If $x \in \widetilde{U^s}$, ${F^s_{uid,x}}' = F^s_{uid,x} sUK_{uid,x} = \left(A'_x\right)^{s_{uid}}$.
If $x \in \widetilde{U^d}$, ${F^d_{uid,x}}' = F^d_{uid,x} dUK_{uid,x} = \left(A'_x\right)^{d_{uid}}$.

$UpCiphertext(CT, cUK, sUK)$. Upon receiving cUK, sUK, the cloud server updates the ciphertext to contain the latest attribute version key as follows:

If $cUK = \left\{cUK_i = \left(D_i'\right)^{\varphi_x - \varphi_x'}\right\}_{\rho_e(i)=x}$ and $sgUK = \perp$, the server randomly chooses $\left\{r_i'' \in \mathbb{Z}_p\right\}_{\rho_e(i)=x}$ and computes $C_{2,i}' = C_{2,i}' cUK_i A_{\rho_e(i)}'^{-r_i''} = \theta^{\lambda_i'} A_{\rho_e(i)}'^{-(r_i'+r_i'')}$.

$D_i' = D_i' g^{r_i''} = g^{r_i'+r_i''}$, where $\rho_e(i) = x$.

Otherwise, the cloud server updates the signature component S_2 as: $S_2 = S_2 sgUK = \left(\prod_{I_A^s} g^{\alpha_k} \theta^{s_{do}''}\right)\left(\prod_{i\in[\ell_s]}\left(A_{\rho_s(i)} V_i\right)^{a_i s_{do}'' + b_i}\right)\left(k_0 \prod_{i=1}^{l} k_i^{c_i}\right)^{w} C_3^{\beta} sgUK = \left(\prod_{I_A^s} g^{\alpha_k} \theta^{s_{do}''}\right)\left(\prod_{i\in[\ell_s]\setminus\mathcal{L}}\left(A_{\rho_s(i)}\right)^{a_i s_{do}'' + b_i}\right)$ $\left(\prod_{i\in\mathcal{L}}\left(A_{\rho_s(i)}'\right)^{a_i s_{do}'' + b_i}\right)\left(\prod_{i\in[\ell_s]} V_i^{a_i s_{do}'' + b_i}\right)\left(k_0 \prod_{i=1}^{l} k_i^{c_i}\right)^{w} C_3^{\beta}$.

Correctness of Attribute Revocation.

By running $UpSecretKey(SK_{uid,k}, sUK_{uid,x}, dUK_{uid,x})$, the secret signing and decryption keys of non-revoked user U_{uid} are associated with the new attribute version key φ_x', which is the same as the updated ciphertext components $\left\{C_{2,i}' = \theta^{\lambda_i'} A_{\rho_e(i)}'^{-(r_i'+r_i'')}\right\}_{\rho_e(i)=x}$ or $S_2 = \left(\prod_{I_A^s} g^{\alpha_k} \theta^{s_{do}''}\right)\left(\prod_{i\in[\ell_s]\mathcal{L}}\left(A_{\rho_s(i)}\right)^{a_i s_{do}'' + b_i}\right)\left(\prod_{i\in\mathcal{L}}\left(A_{\rho_s(i)}'\right)^{a_i s_{do}'' + b_i}\right)\left(\prod_{i\in[\ell_s]} V_i^{a_i s_{do}'' + b_i}\right)\left(k_0 \prod_{i=1}^{l} k_i^{c_i}\right)^{w} C_3^{\beta}$.

For verification, since the updated signature component S_2 is associated with $A_{\rho_s(i)}'$ for i such that $\rho_s(i) = x$, we have $\frac{e(S_2, g)}{e\left(k_0 \prod_{i=1}^{l} k_i^{c_i}, C_1\right) e\left((\gamma_1\gamma_2^{\pi})^{\beta}, C_1\right)\left(\prod_{i\in[\ell_s]\mathcal{L}} e\left(A_{\rho_s(i)}, S_{1,i}\right)\right)\left(\prod_{i\in\mathcal{L}} e\left(A_{\rho_s(i)}', S_{1,i}\right)\right)\left(\prod_{i\in[\ell_s]} e\left(V_i \theta^{\omega_i N_A^s}, S_{1,i}\right)\right)} = \prod_{I_A^s} \Delta_k$, which exhibits the correctness of $Verify$ algorithm.

Additionally, the operations $C_{2,i}' = C_{2,i}' cUK_i A_{\rho_e(i)}'^{-r_i''}$ and $D_i' = D_i' g^{r_i''} = g^{r_i'+r_i''}$ are equivalent to assigning a new random number $r_i' + r_i''$ to the corresponding components of ciphertext. Then in $PartialDecryption\left(PP, CT, PK_{uid}, \left\{PSK_{uid,k}^{d}\right\}_{k\in I_A^e}\right)$ algorithm, we have $e\left(C_{2,i}' \theta^{C_{2,i}''} A_{\rho_e(i)}'^{-D_i''}, g^{d_{uid}}\right) e\left(D_i' g^{D_i''}, F_{uid,\rho_e(i)}^{d}\right) = e\left(\theta^{\lambda_i} A_{\rho_e(i)}'^{-r_i - r_i''}, g^{d_{uid}}\right) e\left(g^{r_i + r_i''}, \left(A_{\rho_e(i)}'\right)^{d_{uid}}\right) = e\left(\theta^{\lambda_i}, g^{d_{uid}}\right)$ for i such that $\rho_e(i) = x$, which exhibits the correctness of $Decryption$ algorithm.

6. Security Analysis

In this section, we state the security of our OMDAC-ABSC scheme in the following theorems. In Theorems 1 and 2, we prove the message confidentiality and ciphertext unforgeability of our scheme respectively. In Theorem 3 we demonstrate the signcryptor privacy. Then in Theorems 4 and 5, we analyze the collusion resistance and revocation security.

Throughout this section, assume 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. $\ell_{e,m}, n_{e,m}, \ell_{s,m}, n_{s,m}$ are the maximum values of $\{\ell_e, n_e, \ell_s, n_s\}$. Suppose that the Hash functions H_1, H_2, H_3 are collision resistant.

6.1. Message Confidentiality

Based on the security model defined in Definition 8 and Theorem 1, we can prove that our proposed scheme guarantees the message confidentiality under the hardness of the q-PBDHE assumption.

Theorem 1. *If an adversary $\mathcal{A}$ can break $\left(T, q_{sk}, q_{psk}, q_{SC}, q_{DS}, \epsilon\right)$-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 $T' = T + O\left(\ell_{e,m} n_{e,m} u_m + \left(n_{e,m} + \left|\widetilde{U}\right| \ell_{e,m} n_{e,m}^2\right) q_{sk} + \left(\left|\widetilde{U}\right| + \ell_{s,m}\right) q_{psk} + \left(\left|\widetilde{U}\right| + l + \ell_{s,m} + \ell_{e,m}\right) q_{SC} + q_{DS}\right) T^e$ $+ O(q_{DS}) T^p$.*

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

Init. The same as defined in Definition 8. Assume $\mathcal{R}_e^* = (\mathrm{M}_e^*, \rho_e^*)$ is the challenge encryption access structure over all the attributes selected from the set of authorities I_A^{*e}. Assume M_e^* is a $\ell_e^* \times n_e^*$ matrix and $n_e^* \leq q$.

Setup. The adversary chooses a set $S'_A \subset S_A$ consisting of the corrupted authorities, and sends S'_A to the simulator $\mathcal{B}$. For each uncorrupted authority $AA_k \in S_A - S'_A$, $\mathcal{B}$ randomly chooses $\alpha'_k \overset{R}{\leftarrow} \mathbb{Z}_p$ and implicitly sets $\alpha_k = \alpha'_k + a^{q+1}$. $\mathcal{B}$ publishes $\Delta_k = e(g,g)^{\alpha_k} = e\left(g^a, g^{a^q}\right) e(g,g)^{\alpha'_k}$.

Let $\psi \overset{R}{\leftarrow} \mathbb{Z}_p, \theta = g^a, \{\|_0, \|_1, \ldots, \|_l\}, \{v_1, v_2 \ldots, v_{\ell_{s,m}}\} \overset{R}{\leftarrow} \mathbb{Z}_p, \left\{k_i = g^{\|_i}\right\}_{i\in[l]}, \{V_i = g^{v_i}\}_{i\in[\ell_{s,m}]}$. $\gamma_1 = g^{\psi}\left(g^{a^q}\right)^{-1}, \gamma_2 = \left(g^{a^q}\right)^{\frac{1}{\pi^*}}$, where $\pi^* = H_1(g^w)$.

$\mathcal{B}$ sends $PP = \left\{e, p, \mathbb{G}, \mathbb{G}_T, g, \theta, \gamma_1, \gamma_2, \{k_0, k_1, \ldots, k_l\}, \{V_1, V_2 \ldots, V_{\ell_{s,m}}\}, H_1, H_2, H_3\right\}$ to $\mathcal{A}$. $\mathcal{B}$ initializes the empty list L_{sk}.

For the authority $AA_k \in S_A - S'_A$, $\mathcal{B}$ chooses $\beta_k, \gamma_k \overset{R}{\leftarrow} \mathbb{Z}_p$ and sets $X_k = g^{1/\beta_k}, Y_k = \theta^{1/\beta_k}, Z_k = \theta^{1/\gamma_k}$. Let $\mathcal{X}$ be the set consisting of the indexes $i \in [\ell_e^*]$ with $\rho_e^*(i) = x \in \widetilde{AA_k}$. For the attribute x where $\mathcal{X} \neq \varnothing$, $\mathcal{B}$ chooses $\varphi_x \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes $A_x = g^{\varphi_x} \prod_{i\in\mathcal{X}} \prod_{k\in[n_e^*]} g^{\frac{a^k \mathrm{M}_e^{*(i,k)}}{b_i}}$, where $\mathrm{M}_e^{*(i,k)}$ is the (i,k)th element of M_e^*. If $\mathcal{X} = \varnothing$, $\mathcal{B}$ chooses $\varphi_x \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes $A_x = g^{\varphi_x}$. This assignment describes that $A_x = g^{\varphi_x}$ for each signing attribute as the signing attributes are different from encryption attributes. $\mathcal{B}$ sends $PK_k = \left\{X_k, Y_k, Z_k, \{A_x\}_{x\in\widetilde{AA_k}}\right\}$ to $\mathcal{A}$. For the authority $AA_k \in S'_A$, $\mathcal{B}$ generates the public keys and secret keys of AA_k as in the real scheme and sends both the public keys and secret keys to $\mathcal{A}$.

Phase 1.

SecretKey query $\mathcal{O}^{sk}\left(\widetilde{U}, AA_k, uid\right)$. $\mathcal{A}$ adaptively queries the secret keys for the attribute set $\widetilde{U} = \widetilde{U^d} \cup \widetilde{U^s}$ with identity uid to the authority AA_k. $\widetilde{U^d}$ does not satisfy $\mathcal{R}_e^*$ together with any keys that can be obtained from corrupted authorities.

$\mathcal{B}$ checks the list L_{sk} that whether the entry $\left(uid, \widetilde{U}, PK_{uid}, SK_{uid}, PK_{uid,k}, SK_{uid,k}\right)$ exists. If it does, $\mathcal{B}$ sends SK_{uid} and $SK_{uid,k}$ to the adversary and publishes the public key PK_{uid} and $PK_{uid,k}$.

(1) Otherwise, $\mathcal{B}$ randomly picks $d'_{uid}, s'_{uid}, z_{uid}$ from $\mathbb{Z}_p^*$ and chooses a vector $\vec{f} = (f_1, f_2, \ldots, f_{n_e^*}) \in \mathbb{Z}_p^{n_e^*}$ such that $f = -1$ and $\vec{f}\mathrm{M}_e^{*i} = 0$ for all $\rho_e^*(i) \in \widetilde{U^d}$ since $\mathcal{R}_e^*\left(\widetilde{U^d}\right) = 0$. $\mathcal{B}$ sets $g^{d_{uid}} = g^{d'_{uid}} \prod_{i=1}^{n_e^*} g^{(f_i a^{q-i+1})/z_{uid}}$, $g^{s_{uid}} = g^{s'_{uid}} g^{-a^q}$, and computes $g^{1/z_{uid}}, \theta^{z_{uid}}, g^{z_{uid}}, \{g^{s_{uid}v_i}\}_{i\in[\ell_{s,m}]}$ as the public key PK_{uid}. Then $\mathcal{B}$ computes

$$PK_{uid,k} = \left\{ PK^1_{uid,k} = g^{1/(\gamma_k z_{uid})}, PK^2_{uid,k} = g^{\alpha_k/(\gamma_k z_{uid})} \theta^{d_{uid}/\gamma_k} = \left(g^{\frac{\alpha'_k}{z_{uid}}} g^{a d'_{uid}} g^{\Sigma_2^{n_e^*} f_i a^{q-i+2}/z_{uid}} \right)^{\frac{1}{\gamma_k}}, PK^3_{uid,k} = g^{\alpha_k/\beta_k} \theta^{s_{uid}/\beta_k} = \left(g^{\alpha'_k} g^{a s'_{uid}}\right)^{\frac{1}{\beta_k}} \right\},$$

and sets $SK_{uid,k}$ as $K^d_{uid,k} = \left(PK^2_{uid,k}\right)^{\gamma_k} = g^{\frac{\alpha'_k}{z_{uid}}} g^{a d'_{uid}} g^{\Sigma_2^{n_e^*} f_i a^{q-i+2}/z_{uid}}$, $K^s_{uid,k} = \left(PK^3_{uid,k}\right)^{\beta_k} = g^{\alpha'_k} g^{a s'_{uid}}$, $\left\{F^s_{uid,x} = \left(g^{s'_{uid}} g^{-a^q}\right)^{\varphi_x}\right\}_{x\in\widetilde{U^s}\cap\widetilde{AA_k}}$. For the attribute $x \in \widetilde{U^d} \cap \widetilde{AA_k}$ such that $\mathcal{X} = \emptyset$, $\mathcal{B}$ computes $F^d_{uid,x} = (g^{d_{uid}})^{\varphi_x}$. Otherwise, $F^d_{uid,x} =$

$g^{d_{uid}\varphi_x}\prod_{i\in\mathcal{X}}\prod_{k\in[n_e^*]}\left(g^{\frac{d'_{uid}a^k}{b_i}}\prod_{j=1,k\neq j}^{n_e^*}g^{\frac{f_j a^{q+k-j+1}}{z_{uid}b_i}}\right)^{\mathrm{M}_e^{*(i,k)}}$. $\mathcal{B}$ sends $SK_{uid}=z_{uid}$ and $SK_{uid,k}$ to the adversary and publishes the public key PK_{uid} and $PK_{uid,k}$. $\mathcal{B}$ inserts $\left(uid,\widetilde{U},PK_{uid},SK_{uid},PK_{uid,k},SK_{uid,k}\right)$ into L_{sk}.

Proxy SecretKey query $\mathcal{O}^{psk}\left(\widetilde{U},AA_k,uid\right)$. $\mathcal{B}$ checks the list L_{sk} that whether the entry $\left(uid,\widetilde{U},PK_{uid},SK_{uid},PK_{uid,k},\ SK_{uid,k}\right)$ exists. If it does not exist, $\mathcal{B}$ issues $\mathcal{O}^{sk}\left(\widetilde{U},AA_k,uid\right)$ query to compute SK_{uid} and $SK_{uid,k}$, and then runs $PxSecretKeyGen(SK_{uid},SK_{uid,k})$ and returns $PSK_{uid,k}$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ directly performs $PxSecretKeyGen(SK_{uid},SK_{uid,k})$ and returns $PSK_{uid,k}$ to $\mathcal{A}$.

Signcryption query $\mathcal{O}^{SC}(\mathcal{M},\mathcal{R}_s,\mathcal{R}_e)$. $\mathcal{A}$ submits a message $\mathcal{M}\in\mathbb{G}_T$, signing and encryption predicts $\mathcal{R}_s=(\mathrm{M}_s,\rho_s),\mathcal{R}_e=(\mathrm{M}_e,\rho_e)$. $\mathcal{B}$ selects a signing attribute set $\widetilde{U^s}$ such that $\mathcal{R}_s\left(\widetilde{U^s}\right)=1$. For each $k\in I_A^s$, $\mathcal{B}$ computes the secret signing key $SK_{uid,k}^s$ and PK_{uid},SK_{uid} from $\mathcal{O}^{sk}\left(\widetilde{U},AA_k,uid\right)$, and $PSK_{uid,k}^s\leftarrow PxSecretKeyGen\left(SK_{uid},SK_{uid,k}^s\right)$, where uid is an arbitrary identity. Then $\mathcal{B}$ returns the ciphertext $CT\leftarrow Signcryption\left(\mathcal{M},PP,\left\{PSK_{uid,k}^s\right\}_{k\in I_A^s},PK_{uid},\{PK_k\}_{k\in I_A^e},SK_{uid},\mathcal{R}_s,\mathcal{R}_e\right)$ to $\mathcal{A}$.

DeSigncryption query $\mathcal{O}^{DS}\left(CT,\widetilde{U^d}\right)$. If $\left|tt-\overline{tt}\right|>thre_{tt}$ or $\mathcal{R}_e\left(\widetilde{U^d}\right)=0$, then $\mathcal{B}$ returns $\perp$. If $C_1=g^s$, $\mathcal{B}$ aborts. If $Verify$ algorithm is invalid, $\mathcal{B}$ returns $\perp$.Otherwise, $\mathcal{B}$ carries out the following steps.

Assume the encryption predicate contained in CT is $\mathcal{R}_e$ and I_A^e is the set which consists of the indexes of the authorities whose attributes are associated with rows of M_e.

If $\widetilde{U^d}$ does not satisfy the challenge encryption predicate $\mathcal{R}_e^*$, then $\mathcal{B}$ can obtain SK_{uid} and secret decryption key $SK_{uid,k}^d$ from $\mathcal{O}^{sk}\left(\widetilde{U},AA_k,uid\right)$, and $PSK_{uid,k}^d\leftarrow PxSecretKeyGen\left(SK_{uid},SK_{uid,k}^d\right)$. $\mathcal{B}$ returns the output of $DeSigncryption\left(PP,CT,PK_{uid},\left\{PSK_{uid,k}^d\right\}_{k\in I_A},SK_{uid}\right)$ to $\mathcal{A}$.

Otherwise, if $\mathcal{R}_e^*\left(\widetilde{U^d}\right)=1$, assume $\pi=H_1(C_1=g^{w_1})$, where w_1 is the secret value chosen to generate CT in signcryption phase. Then for $k\in I_A^e$, $\mathcal{B}$ compute $e\left(g^{\alpha'_k},C_1\right)e\left(\frac{C_3}{C_1^{\psi}},g^a\right)^{\left(\frac{\pi}{\pi^*}-1\right)^{-1}}=$

$e\left(g^{\alpha'_k},g^{w_1}\right)e\left(\frac{(\gamma_1\gamma_2{}^{\pi})^{w_1}}{g^{sx\psi}},g^a\right)^{\left(\frac{\pi}{\pi^*}-1\right)^{-1}}=e\left(g^{\alpha'_k},g^{w_1}\right)e\left(\frac{\left(g^{\psi}\left(g^{a^q}\right)^{-1}\left(g^{a^q}\right)^{\frac{\pi}{\pi^*}}\right)^{w_1}}{g^{w_1\psi}},g^a\right)^{\left(\frac{\pi_j}{\pi^*}-1\right)^{-1}}=$

$e\left(g^{\alpha'_k},g^{w_1}\right)e\left(g^{a^q},g^{aw_1}\right)=\Delta_k^{w_1}$. Thus $\mathcal{B}$ can return $\mathcal{M}=\frac{C_0}{\prod_{k\in I_A^e}\Delta_k^{w_1}}$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits two messages $\mathcal{M}_0,\mathcal{M}_1$ with the same length and signing predicate $\mathcal{R}_s^*=(\mathrm{M}_s^*,\rho_s^*)$ to $\mathcal{B}$. Assume I_A^{*s} is the set which consists of the indexes of the authorities whose attributes are associated with rows of M_s^* and M_s^* is a $\ell_s^*\times n_s^*$ matrix. $\mathcal{B}$ chooses $\lceil\in\{0,1\}$. $\mathcal{B}$ selects a signing attribute set $\widetilde{U^s}$ satisfying $\mathcal{R}_s^*\left(\widetilde{U^s}\right)=1$ and an arbitrary identity $uid_{\mathcal{A}}$.

Let $\overrightarrow{a}=\left(a_1,a_2,\ldots,a_{\ell_s^*}\right)\in\mathbb{Z}_p^{\ell_s^*}$ such that $\overrightarrow{a}\cdot\mathrm{M}_s^*=\overrightarrow{1}$, $\overrightarrow{b}=\left(b_1,b_2,\ldots,b_{\ell_s^*}\right)\in\mathbb{Z}_p^{\ell_s^*}$ such that $\sum_{i\in[\ell_s^*]}b_i\mathrm{M}_s^{*i}=\overrightarrow{0}$. Implicitly set $s_{uid_{\mathcal{A}}}=s''_{uid_{\mathcal{A}}}-a^q$. Then compute the challenge ciphertext as follows:

Let $\overrightarrow{\varepsilon}=\left(w,wa+\varepsilon_2,\ldots,sa^{n_e^*-1}+\varepsilon_{n_e^*}\right)\in\mathbb{Z}_p^{n_e^*}$ and implicitly sets $r_i=r'_i+wb_i$ for all $i\in[\ell_e^*]$. Select $\left\{r''_1,r''_2,\ldots,r''_{\ell_e^*}\right\}\xleftarrow{R}\mathbb{Z}_p$, $\left\{\lambda'_1,\lambda'_2,\ldots,\lambda'_{\ell_e^*}\right\}\xleftarrow{R}\mathbb{Z}_p$.

$C_0=\mathcal{M}_{\lceil}\prod_{k\in I_A^{*e}}\Omega e(g^w,g)^{\alpha'_k}$, $C_1=g^w$.

$C'_{2,i}=g^{a(\lambda_i-\lambda'_i)}A_{\rho_e^*(i)}{}^{-(r_i-r''_i)}=A_{\rho_e^*(i)}{}^{-(r'_i-r''_i)}g^{-wb_i\varphi_{\rho_e^*(i)}}g^{-a\lambda'_i}\prod_{k=2}^{n_e^*}g^{a\varepsilon_k\mathrm{M}_e^{*(i,k)}}\prod_{l\in\mathcal{X}\setminus i}\prod_{k\in[n_e^*]}g^{\frac{-wa^kb_i\mathrm{M}_e^{*(l,k)}}{b_l}}$, $C''_{2,i}=\lambda'_i$.

$D'_i=g^{r_i-r''_i}=g^{r'_i-r''_i}g^{wb_i}$, $D''_i=r''_i$. $C_3=g^{\psi w}$.

$$S_{1,i} = g^{a_i s_{uid_{\mathcal{A}}} + b_i} = g^{a_i (s''_{uid_{\mathcal{A}}} - a^q) + b_i},$$

$$S_2 = \left(\prod_{I_A^{*s}} g^{\alpha'_k} g^{a s''_{uid_{\mathcal{A}}}}\right)\left(\prod_{i\in[\ell_s]}\left(g^{\varphi_{\rho_s^*(i)}+v_i}\right)^{a_i(s''_{uid_{\mathcal{A}}}-a^q)+b_i}\right)(g^w)^{\|_0+\sum_{i=1}^{l}\|_i c_i+\psi\beta^*},$$

where $H_2\left(\prod_{i\in[\ell_s]} S_{1,i}, tt, \mathcal{R}_s, \mathcal{R}_e\right) = (c_1, c_2, \ldots, c_l)$ and $H_3(C_0, C_1, C_3, \mathcal{R}_s, \mathcal{R}_e) = \beta^*$.

Finally, $\mathcal{B}$ sends the challenge ciphertext $CT^* = \left\{C_0, C_1, \left\{C'_{2,i}, C''_{2,i}, D'_i, D''_i\right\}_{i\in[\ell_e^*]}, \{S_{1,i}\}_{i\in[\ell_s^*]}, S_2, tt\right\}$ 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 $\mathcal{R}_e^*\left(\widetilde{U^d}\right) = 1$.

Guess. $\mathcal{A}$ outputs his guess $\widetilde{\lfloor}$ on $\hat{\lfloor}$. If $\widetilde{\lfloor} = \hat{\lfloor}$, $\mathcal{B}$ outputs 0 and guess that $\Omega = e(g,g)^{a^{q+1}w}$; 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 $C_1 = g^w$, then the simulation aborts. The probability is at most $\frac{q_{DS}}{p}$. If $\lfloor = 0$, $\Omega = e(g,g)^{a^{q+1}w}$ and $\mathcal{B}$ does not abort, then CT^* is a valid ciphertext of $\mathcal{M}_0$. In this case, we have $Pr\left[\widetilde{\lfloor} = \hat{\lfloor} \middle| \lfloor = 0\right] > \frac{1}{2} + \epsilon - \frac{q_{DS}}{p}$. If Ω is a random element in $\mathbb{G}_T$, then C_0 is a random element and $\mathcal{A}$ cannot obtain $\mathcal{M}_{\lfloor}$, namely the advantage in this case is $Pr\left[\widetilde{\lfloor} \neq \hat{\lfloor} \middle| \lfloor = 1\right] = \frac{1}{2}$. 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 $T' = T + O\left(\ell_{e,m} n_{e,m} u_m + \left(n_{e,m} + \left|\widetilde{U}\right| \ell_{e,m} n_{e,m}^2\right) q_{sk} + \left(\left|\widetilde{U}\right| + \ell_{s,m}\right) q_{psk} + \left(\left|\widetilde{U}\right| + l + \ell_{s,m} + \ell_{e,m}\right) q_{SC} + q_{DS}\right) T^e + O(q_{DS}) T^p$. □

6.2. Ciphertext Unforgeability

Based on the security model defined in Definition 9 and Theorem 2, we can prove that our proposed scheme guarantees the ciphertext unforgeability under the hardness of the q-PBDHE assumption.

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

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

Init. The same as defined in Definition 9. Assume $\mathcal{R}_s^* = (\mathrm{M}_s^*, \rho_s^*)$ is the challenge signing access structure over all the attributes selected from the involved set of authorities I_A^{*s}. M_s^* is a $\ell_s^* \times n_s^*$ matrix and $n_s^* \leq q$.

Setup. The adversary chooses a set of $S'_A \subset S_A$ consisting of the corrupted authorities, and sends S'_A to the simulator $\mathcal{B}$.

For each uncorrupted authority $AA_k \in S_A - S'_A$, $\mathcal{B}$ randomly chooses $\alpha'_k \xleftarrow{R} \mathbb{Z}_p$ and implicitly sets $\alpha_k = \alpha'_k + a^{q+1}$. $\mathcal{B}$ publishes $\Delta_k = e(g,g)^{\alpha_k} = e\left(g^a, g^{a^q}\right) e(g,g)^{\alpha'_k}$.

Let $\sigma_1, \sigma_2 \xleftarrow{R} \mathbb{Z}_p, \theta = g^a$. $\mathcal{B}$ chooses $m \xleftarrow{R} \{0,1,\ldots,l\}, \varrho_0, \varrho_1, \ldots, \varrho_l \xleftarrow{R} \{0,1,\ldots,\varpi-1\}, \sigma_1, \sigma_2, \|_0, \|_1, \ldots, \|_l \xleftarrow{R} \mathbb{Z}_p^*$. Set $k_0 = \left(g^{a^q}\right)^{\varrho_0 - \varpi m} g^{\|_0}$ and $\left\{k_i = \left(g^{a^q}\right)^{\varrho_i} g^{\|_i}\right\}_{i\in[l]}$. $\gamma_1 = g^{\sigma_1}, \gamma_2 = g^{\sigma_2}$. For $i \in [\ell_s^*]$, $V_i = \prod_{l\in\mathcal{X}\setminus i}\prod_{k\in[n_s^*]} g^{a^k \mathrm{M}_s^{*(l,k)} N_A^{*s}}$ where $N_A^{*s} = \left|I_A^{*s}\right|$. For $i \in [\ell_s^*+1, \ell_{s,m}]$, $V_i = g^{v_i}$ where $v_i \xleftarrow{R} \mathbb{Z}_p$.

Assume $\varpi = 4q_{SC}$ and $\varpi(l+1) < p$. $\mathcal{B}$ defines two functions $L_1\left(\vec{c}\right) = p - \varpi m + \varrho_0 + \sum_{i=1}^{l} c_i\varrho_i$ and $L_2\left(\vec{c}\right) = \lfloor 0 + \sum_{i=1}^{l} c_i \rfloor_i$ for each $\vec{c} = (c_1, c_2, \ldots, c_l) \in \{0,1\}^l$. Thus $k_0 \prod_{i=1}^{l} k_i^{c_i} = \left(g^{a^q}\right)^{L_1(\vec{c})} g^{L_2(\vec{c})}$. Let $L\left(\vec{c}\right) = \begin{cases} 0, \ \varrho_0 + \sum_{i=1}^{l} c_i\varrho_i = 0 \ mod \ \varpi \\ 1, \ otherwise \end{cases}$. Then $L\left(\vec{c}\right) = 1$ implies $L_1\left(\vec{c}\right) \neq 0 \ mod \ p$.

$\mathcal{B}$ sends $PP = \left\{e, p, \mathbb{G}, \mathbb{G}_T, g, \theta, \gamma_1, \gamma_2, \{k_0, k_1, \ldots, k_l\}, \left\{V_1, V_2 \ldots, V_{\ell_{s,m}}\right\}, H_1, H_2, H_3\right\}$ to $\mathcal{A}$. $\mathcal{B}$ initializes the empty list L_{sk}.

For the authority $AA_k \in S_A - S_A'$, $\mathcal{B}$ chooses $\beta_k, \gamma_k \xleftarrow{R} \mathbb{Z}_p$ and sets $X_k = g^{1/\beta_k}, Y_k = \theta^{1/\beta_k}, Z_k = \theta^{1/\gamma_k}$. Let $\mathcal{X}$ be the set consisting of the indexes $i \in [\ell_s^*]$ with $\rho_s^*(i) = x \in \widetilde{AA_k}$. For the attribute x where $\mathcal{X} \neq \varnothing$, $\mathcal{B}$ chooses $\varphi_x \xleftarrow{R} \mathbb{Z}_p$ and computes $A_x = g^{\varphi_x}\prod_{i\in\mathcal{X}}\prod_{k\in[n_s^*]} g^{-a^k \mathrm{M}_s^{*(i,k)} N_A^{*s}}$. If $\mathcal{X} = \varnothing$, $\mathcal{B}$ chooses $\varphi_x \xleftarrow{R} \mathbb{Z}_p$ and computes $A_x = g^{\varphi_x}$. This assignment describes that $A_x = g^{\varphi_x}$ for each encryption attribute as the signing attributes are different from encryption attributes. $\mathcal{B}$ sends $PK_k = \left\{X_k, Y_k, Z_k, \{A_x\}_{x\in\widetilde{AA_k}}\right\}$ to $\mathcal{A}$. For the authority $AA_k \in S_A'$, $\mathcal{B}$ generates the public keys and secret keys of AA_k as in the real scheme and sends both the public keys and secret keys to $\mathcal{A}$.

SecretKey query $\mathcal{O}^{sk}\left(\widetilde{U}, AA_k, uid\right)$. $\mathcal{A}$ adaptively queries the secret keys for the attribute set $\widetilde{U} = \widetilde{U^d} \cup \widetilde{U^s}$ with identity uid to the authority AA_k. $\widetilde{U^s}$ does not satisfy $\mathcal{R}_s^*$ together with any keys that can be obtained from corrupted authorities.

(1) $\mathcal{B}$ checks the list L_{sk} that whether the entry $\left(uid, \widetilde{U}, PK_{uid}, SK_{uid}, PK_{uid,k}, SK_{uid,k}\right)$ exists. If it does, $\mathcal{B}$ sends SK_{uid} and $SK_{uid,k}$ to the adversary and publishes the public key PK_{uid} and $PK_{uid,k}$.

(2) Otherwise, $\mathcal{B}$ randomly picks $d_{uid}', s_{uid}', z_{uid}$ from $\mathbb{Z}_p^*$ and chooses a vector $\vec{f} = \left(f_1, f_2, \ldots, f_{n_s^*}\right) \in \mathbb{Z}_p^{n_s^*}$ such that $f = -1$ and $\vec{f}\mathrm{M}_s^{*i} = 0$ for all $\rho_s^*(i) \in \widetilde{U^s}$. since $\mathcal{R}_s^*\left(\widetilde{U^s}\right) = 0$. $\mathcal{B}$ computes $g^{d_{uid}} = g^{d_{uid}'} g^{-a^q/z_{uid}}$, $g^{s_{uid}} = g^{s_{uid}'}\prod_{i=1}^{n_s^*} g^{f_i a^{q-i+1}}$, and $\left\{V_i^{s_{uid}} = g^{s_{uid}v_i}\right\}_{i\in[\ell_s^*+1,\ell_{s,m}]}$. For $i \in [\ell_s^*]$, $V_i^{s_{uid}} = V_i^{s_{uid}'}\prod_{l\in\mathcal{X}\setminus i}\prod_{k\in[n_s^*]}\left(\prod_{j=1,k\neq j}^{n_s^*} g^{f_j a^{q+k-j+1}}\right)^{\mathrm{M}_s^{*(l,k)} N_A^{*s}}$. Set $PK_{uid} = \left\{g^{s_{uid}}, g^{d_{uid}}, g^{1/z_{uid}}, \theta^{z_{uid}}, g^{z_{uid}}, \left\{V_i^{s_{uid}}\right\}_{i\in[\ell_{s,m}]}\right\}$ and $PK_{uid,k} = \left\{PK_{uid,k}^1 = g^{1/(\gamma_k z_{uid})}, PK_{uid,k}^2 = g^{\alpha_k/(\gamma_k z_{uid})}\theta^{d_{uid}/\gamma_k} = \left(\frac{\alpha_k'}{g^{z_{uid}}} g^{a d_{uid}'}\right)^{\frac{1}{\gamma_k}}, PK_{uid,k}^3 = g^{\alpha_k/\beta_k}\theta^{s_{uid}/\beta_k} = \left(g^{\alpha_k'} g^{a s_{uid}'} g^{\Sigma_2^{n_s^*} f_i a^{q-i+2}}\right)^{\frac{1}{\beta_k}}\right\}$.

Then $\mathcal{B}$ sets $K_{uid,k}^d = \left(PK_{uid,k}^2\right)^{\gamma_k} = \frac{\alpha_k'}{g^{z_{uid}}} g^{a d_{uid}'}, K_{uid,k}^s = \left(PK_{uid,k}^3\right)^{\beta_k} = g^{\alpha_k'} g^{a s_{uid}'} g^{\Sigma_2^{n_s^*} f_i a^{q-i+2}}, \left\{F_{uid,x}^d = \left(g^{d_{uid}'} g^{-a^q/z_{uid}}\right)^{\varphi_x}\right\}_{x\in\widetilde{U^d}\cap\widetilde{AA_k}}$. For the attribute $x \in \widetilde{U^s} \cap \widetilde{AA_k}$ such that $\mathcal{X} = \emptyset$, $\mathcal{B}$ computes $F_{uid,x}^s = (g^{s_{uid}})^{\varphi_x}$. Otherwise, $F_{uid,x}^s = g^{s_{uid}\varphi_x}\prod_{i\in\mathcal{X}}\prod_{k\in[n_s^*]}\left(g^{-s_{uid}' a^k}\prod_{j=1,k\neq j}^{n_s^*} g^{-f_j a^{q+k-j+1}}\right)^{\mathrm{M}_s^{*(i,k)} N_A^s}$. $\mathcal{B}$ sends $SK_{uid} = z_{uid}$ and $SK_{uid,k}$ to the adversary and publishes the public key PK_{uid} and $PK_{uid,k}$. $\mathcal{B}$ inserts $\left(uid, \widetilde{U}, PK_{uid}, SK_{uid}, PK_{uid,k}, SK_{uid,k}\right)$ into L_{sk}.

Proxy SecretKey query $\mathcal{O}^{psk}\left(\widetilde{U}, AA_k, uid\right)$. The same as Theorem 1.

Signcryption query $\mathcal{O}^{SC}(\mathcal{M}, \mathcal{R}_s, \mathcal{R}_e)$. $\mathcal{A}$ submits a message $\mathcal{M} \in \mathbb{G}_T$, signing and encryption predicts $\mathcal{R}_s = (\mathrm{M}_s, \rho_s), \mathcal{R}_e = (\mathrm{M}_e, \rho_e)$. $\mathcal{B}$ selects a signing attribute set $\widetilde{U^s}$ such that $\mathcal{R}_s\left(\widetilde{U^s}\right) = 1$. $\mathcal{B}$ performs as follows:

(1) It first computes a vector $\overrightarrow{a} = (a_1, a_2, \ldots, a_{\ell_s}) \in \mathbb{Z}_p^{\ell_s}$ such that $\overrightarrow{a} \cdot \mathrm{M}_s = \overrightarrow{1}$. Then $\mathcal{B}$ chooses $\overrightarrow{b} = (b_1, b_2, \ldots, b_{\ell_s}) \in \mathbb{Z}_p^{\ell_s}$ such that $\sum_{i\in[\ell_s]} b_i \mathrm{M}_s^i = \overrightarrow{0}$.

(2) $\mathcal{B}$ randomly chooses $s'_{uid} \xleftarrow{R} \mathbb{Z}_p^*$ and computes $\left\{S_{1,i} = g^{a_i s'_{uid} + b_i}\right\}_{i\in[\ell_s]}$.

(3) Assume $H_2\left(\prod_{i\in[\ell_s]} S_{1,i}, tt, \mathcal{R}_s, \mathcal{R}_e\right) = (c_1, c_2, \ldots, c_l) = \overrightarrow{c} \in \{0,1\}^l$. If $L\left(\overrightarrow{c}\right) = 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ implicitly sets $w_{\mathcal{A}} = w_1 - \frac{aN_A^s}{L_1\left(\overrightarrow{c}\right)}$ where $w_1 \xleftarrow{R} \mathbb{Z}_p^*$. Then $C_0 = \mathcal{M}\prod_{k\in I_A^e} \Delta_k{}^{w_{\mathcal{A}}}$, $C_1 = g^{w_{\mathcal{A}}} = g^{w_1}(g^a)^{-\frac{N_A^s}{L_1(\overrightarrow{c})}}$, $C_3 = (g^{\sigma_1} g^{\pi\sigma_2})^{w_{\mathcal{A}}}$, where $\Delta_k{}^{w_{\mathcal{A}}} = \Delta_k{}^{w_1}\left(e\left(g^{a^2}, g^{a^q}\right)e\left(g^{\alpha'_k}, g^a\right)\right)^{-N_A^s/L_1(\overrightarrow{c})}$ and $\pi = H_1(C_1)$.

(4) $\mathcal{B}$ chooses $\left\{r_1, r_2, \ldots, r_{\ell_e}\right\} \xleftarrow{R} \mathbb{Z}_p$, $\overrightarrow{\varepsilon} = \left(w_1 - \frac{aN_A^s}{L_1\left(\overrightarrow{c}\right)}, \varepsilon_2, \ldots, \varepsilon_{n_e}\right) \in \mathbb{Z}_p^{n_e}$, $\lambda_i = \mathrm{M}_e^i \overrightarrow{\varepsilon}$. Then $\mathcal{B}$ selects $\left\{r'_1, r'_2, \ldots, r'_{\ell_e}\right\} \xleftarrow{R} \mathbb{Z}_p$, $\left\{\lambda'_1, \lambda'_2, \ldots, \lambda'_{\ell_e}\right\} \xleftarrow{R} \mathbb{Z}_p$. For $i \in [\ell_e]$, $\mathcal{B}$ computes $C'_{2,i} = g^{a(\lambda_i - \lambda'_i)} A_{\rho_e(i)}{}^{-(r_i - r'_i)} = g^{a((w_1 - \frac{aN_A^s}{L_1(\overrightarrow{c})})\mathrm{M}_e^{(i,1)} + \sum_{j=2}^{n_e^*} \varepsilon_j \mathrm{M}_e^{(i,j)} - \lambda'_i)} g^{-\varphi_{\rho_e(i)}(r_i - r'_i)}$, $C''_{2,i} = \lambda'_i$, $D'_i = g^{r_i - r'_i}$, $D''_i = r'_i$.

(5) $\mathcal{B}$ computes $H_3(C_0, C_1, C_3, \mathcal{R}_s, \mathcal{R}_e) = \beta$, $S_2 = \left(\prod_{I_A^s} g^{\alpha_k}\right)\theta^{s'_{uid} N_A^s}\left(\prod_{i\in[\ell_s]}\left(A_{\rho_s(i)} V_i\right)^{a_i s'_{uid} + b_i}\right)$ $\left(k_0 \prod_{i=1}^l k_i{}^{c_i}\right)^{w_{\mathcal{A}}}(\gamma_1\gamma_2{}^{\pi})^{w_{\mathcal{A}}\beta} = \left(\prod_{I_A^s} g^{\alpha'_k + a^{q+1}}\right)\theta^{s'_{uid} N_A^s} C_3^{\beta}\left(\left(g^{a^q}\right)^{L_1(\overrightarrow{c})} g^{L_2(\overrightarrow{c})}\right)^{w_1 - \frac{aN_A^s}{L_1(\overrightarrow{c})}}$ $\left(\prod_{i=1}^{\ell_s}\left(A_{\rho_s(i)} V_i\right)^{a_i s'_{uid} + b_i}\right) = \left(\prod_{I_A^s} g^{\alpha'_k}\right) g^{a^{q+1} N_A^s} \theta^{s'_{uid} N_A^s} C_3^{\beta}\left(\left(g^{a^q}\right)^{L_1(\overrightarrow{c})} g^{L_2(\overrightarrow{c})}\right)^{w_1}\left(g^{aN_A^s}\right)^{\frac{-L_2(\overrightarrow{c})}{L_1(\overrightarrow{c})}} g^{-a^{q+1} N_A^s}$ $\left(\prod_{i=1}^{\ell_s}\left(A_{\rho_s(i)} V_i\right)^{a_i s'_{uid} + b_i}\right) = \left(\prod_{I_A^s} g^{\alpha'_k}\right)\theta^{s'_{uid} N_A^s} C_3^{\beta}\left(\left(g^{a^q}\right)^{L_1(\overrightarrow{c})} g^{L_2(\overrightarrow{c})}\right)^{w_1}\left(g^{aN_A^s}\right)^{\frac{-L_2(\overrightarrow{c})}{L_1(\overrightarrow{c})}}$ $\left(\prod_{i=1}^{\ell_s}\left(A_{\rho_s(i)} V_i\right)^{a_i s'_{uid} + b_i}\right)$. Finally, $\mathcal{B}$ sends $CT = \left\{C_0, C_1, \left\{C'_{2,i}, C''_{2,i}, D'_i, D''_i\right\}_{i\in[\ell_e]}, \{S_{1,i}\}_{i\in[\ell_s]}, S_2, tt\right\}$ to $\mathcal{A}$.

DeSigncryption query $\mathcal{O}^{DS}\left(CT, \widetilde{U^d}\right)$. If $|tt - \overline{tt}| > thre_{tt}$ or $\mathcal{R}_e\left(\widetilde{U^d}\right) = 0$, then $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ issues the $\mathcal{O}^{\mathrm{sk}}$ query to get the secret decryption key and returns the output of *DeSigncryption* to $\mathcal{A}$.

Forgery. $\mathcal{A}$ submits a valid ciphertext CT^* for the challenge signing predicate $\mathcal{R}_s^*$ and an encryption predicate $\mathcal{R}_e$. If $\mathcal{M} \leftarrow DeSigncryption(PP, CT^*, PK, SK)$ and $\mathcal{A}$ has never issued $\mathcal{O}^{\mathrm{SC}}(\mathcal{M}, \mathcal{R}_{\mathrm{s}}^*, \mathcal{R}_{\mathrm{e}})$. $\mathcal{B}$ performs as follows:

(1) $\mathcal{B}$ computes $H_2\left(\prod_{i\in[\ell_s^*]} S_{1,i}, tt, \mathcal{R}_s^*, \mathcal{R}_e\right) = (c_1, c_2, \ldots, c_l) = \overrightarrow{c} \in \{0,1\}^l$. If $\varpi m \neq \varrho_0 + \sum_{i=1}^l c_i \varrho_i$, $\mathcal{B}$ aborts. Otherwise, $L_1\left(\overrightarrow{c}\right) = 0 \ mod \ p$.

(2) If CT^* is a valid ciphertext, then $H_3(C_0, C_1, C_3, \mathcal{R}_s^*, \mathcal{R}_e) = \beta$ and $\pi = H_1(C_1)$. Then

$$S_2 = \left(\prod_{I_A^{*s}} g^{\alpha_k}\right)\theta^{s''_{uid} N_A^{*s}}\left(\prod_{i\in[\ell_s^*]}\left(A_{\rho_s^*(i)} V_i\right)^{a_i s''_{uid} + b_i}\right)\left(k_0 \prod_{i=1}^l k_i{}^{c_i}\right)^{w}(\gamma_1\gamma_2{}^{\pi})^{w\beta} =$$

$$\left(\prod_{I_A^{*s}} g^{\alpha'_k + a^{q+1}}\right)\theta^{s''_{uid} N_A^{*s}} C_1^{L_2(\overrightarrow{c}) + \beta(\sigma_1 + \pi\sigma_2)}\left(\prod_{i=1}^{\ell_s^*}\left(g^{\varphi_{\rho_s^*(i)}}\prod_{j\in[n_s^*]} g^{-a^j \mathrm{M}_s^{*(i,j)} N_A^{*s}}\right)^{a_i s''_{uid} + b_i}\right) =$$

$\left(\prod_{I_A^{*s}} g^{\alpha'_k}\right) g^{N_A^{*s} a^{q+1}} \theta^{s''_{uid} N_A^{*s}} C_1^{L_2(\vec{c})+\beta(\sigma_1+\pi\sigma_2)} \prod_{i=1}^{\ell_s^*} S_{1,i}^{\varphi_{\rho_s^*(i)}} g^{N_A^{*s}(\sum_{i\in[\ell_s^*]}\sum_{j\in[n_s^*]} -a^j \mathrm{M}_s^{*(i,j)}(a_i s''_{uid}+b_i))} =$
$\left(\prod_{I_A^{*s}} g^{\alpha'_k}\right) g^{N_A^{*s} a^{q+1}} \theta^{s''_{uid} N_A^{*s}} C_1^{L_2(\vec{c})+\beta(\sigma_1+\pi\sigma_2)} \prod_{i=1}^{\ell_s^*} S_{1,i}^{\varphi_{\rho_s^*(i)}} \theta^{-s''_{uid} N_A^{*s}} = \left(\prod_{I_A^{*s}} g^{\alpha'_k}\right) g^{N_A^{*s} a^{q+1}} C_1^{L_2(\vec{c})+\beta(\sigma_1+\pi\sigma_2)}$
$\prod_{i=1}^{\ell_s^*} S_{1,i}^{\varphi_{\rho_s^*(i)}}$, where $\vec{a}\cdot \mathrm{M}_s^* = \vec{1}$, $\vec{b}\cdot \mathrm{M}_s^* = \vec{0}$ and $\sum_{i\in[\ell_s^*]}\sum_{j\in[n_s^*]} -a^j \mathrm{M}_s^{*(i,j)}(a_i s''_{uid}+b_i) = -a s''_{uid}$.

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

6.3. Signcryptor Privacy

Based on the security model defined in Definition 10, we prove that our scheme guarantees signcryptor privacy in Theorem 3.

Theorem 3. *Our scheme guarantees the signcryptor privacy.*

Proof. The challenger sends $PP, PK, \{PK_k, SK_k\}_{I_A}$ to the adversary $\mathcal{A}$. Then $\mathcal{A}$ outputs two signing attribute sets $\widetilde{U_0^s}, \widetilde{U_1^s}$ satisfying $\mathcal{R}_s\left(\widetilde{U_0^s}\right) = 1 = \mathcal{R}_s\left(\widetilde{U_1^s}\right)$. The challenger selects $\lfloor \xleftarrow{R} \{0,1\}$ and computes $CT_{\lfloor}$ with the secret signing key $SK_{uid,k}^{s,\lfloor} \leftarrow SecretKeyGen\left(PP, PK_k, SK_k, PK_{uid}, \widetilde{U_{\lfloor}^s}\right)$. Note that both the challenger and $\mathcal{A}$ can compute $SK_{uid,k}^{s,\lfloor}$ for $\widetilde{U_{\lfloor}^s}$, where $k \in I_A$. Specifically, $K_{uid,k}^{s,\lfloor} = g^{\alpha_k} \theta^{s_{uid}^{\lfloor}}, F_{uid,x}^{s,\lfloor} = A_x^{s_{uid}^{\lfloor}}$, where $s_{uid}^{\lfloor} \xleftarrow{R} \mathbb{Z}_p^*$.

If the challenger uses $SK_{uid,k}^{s,0}$, then it can generate the ciphertext $CT_0 = \left\{C_0, C_1^0, \left\{C_{2,i}'^0, C_{2,i}''^0, D_i'^0, D_i''^0\right\}_{i\in[\ell_e]}, \left\{S_{1,i}^0\right\}_{i\in[\ell_s]}, S_2^0, tt\right\}$ as follows.

$C_1^0 = g^{\varpi_0}$, $C_3^0 = (\gamma_1\gamma_2^{\pi_0})^{\varpi_0}$ where $\pi_0 = H_1(C_1^0)$.

$\left\{C_{2,i}'^0 = \theta^{\lambda_i'^0} A_{\rho_e(i)}^{-r_i'^0}, D_i'^0 = g^{r_i'^0}, C_{2,i}''^0 = \lambda_i^0 - \lambda_i'^0, D_i''^0 = r_i - r_i'^0\right\}_{i\in[\ell_e]}, \left\{S_{1,i}^0 = g^{a_i^0 s''^0_{uid} + b_i^0}\right\}_{i\in[\ell_s]}$.

$H_2\left(\prod_{i\in[\ell_s]} S_{1,i}^0, tt, \mathcal{R}_s, \mathcal{R}_e\right) = (c_1, c_2, \ldots, c_l) \in \{0,1\}^l$. $H_3(C_0, C_1^0, C_3^0, \mathcal{R}_s, \mathcal{R}_e) = \beta$.

$S_2^0 = \left(\prod_{I_A^s} g^{\alpha_k}\right) \theta^{s''^0_{uid} N_A^s} \left(\prod_{i\in[\ell_s]} \left(A_{\rho_s(i)} V_i\right)^{v_i^0 s''^0_{uid} + t_i^0}\right) \left(k_0 \prod_{i=1}^{l} k_i^{c_i}\right)^{\varpi_0} (C_3^0)^{\beta}$, where $s''^0_{uid} = s_{uid}^0 + s'^0_{uid}$ and $s'^0_{uid} \xleftarrow{R} \mathbb{Z}_p^*$.

If the challenger uses $SK_{uid,k}^{s,1}$, and sets $w_0 = w_1, \lambda_i'^0 = \lambda_i'^1, r_i^0 = r_i^1, r_i'^0 = r_i'^1, s'^1_{uid} = s''^0_{uid} - s^1_{uid}$, then $\lambda_i^0 = \lambda_i^1$, $s''^0_{uid} = s''^1_{uid} = s''_{uid}$. Thus $C_1^0 = C_1^1, \pi_0 = \pi_1, C_{2,i}'^0 = C_{2,i}'^1, C_{2,i}''^0 = C_{2,i}''^1, C_3^0 = C_3^1$, $D_i'^0 = D_i'^1, D_i''^0 = D_i''^1$. The challenger sets $\vec{a^1}\cdot \mathrm{M}_s = \vec{1}$ and sets $b_i^1 = \left(a_i^0 - a_i^1\right) s''_{uid} + b_i^0$. Then $\vec{b^1}\cdot \mathrm{M}_s = \vec{0}$ and $a_i^0 s''_{uid} + b_i^0 = a_i^1 s''_{uid} + b_i^1$. Hence $S_{1,i}^0 = S_{1,i}^1, S_2^0 = S_2^1$, and $CT_0 = CT_1$.

Similarly, if the challenger firstly uses $SK_{uid,k}^{s,1}$ to generate $CT_1 = \left\{C_0, C_1^1, \left\{C_{2,i}'^1, C_{2,i}''^1, D_i'^1, D_i''^1\right\}_{i\in[\ell_e]}, \left\{S_{1,i}^1\right\}_{i\in[\ell_s]}, S_2^1, tt\right\}$, then it can generate CT_0 with $SK_{uid,k}^{s,0}$ and $CT_1 = CT_0$. Therefore, $\mathcal{A}$ can only outputs a random guess $\lfloor'$ and the probability is at most $\frac{1}{2}$. □

6.4. Collusion Resistance

High-Level Overview

In our scheme, the secret keys of each user are associated the random elements d_{uid}, s_{uid} picked by CA which are difficult for each user, fog node, authority and cloud server to compute or learn. Therefore, the colluders such as the user, fog node, and cloud server cannot selectively replace or convert the components of the secret keys under the discrete logarithm assumption. Additionally, since *uid* chosen by CA is globally unique in the system and d_{uid} and s_{uid} are kept secret, secret keys generated from different authorities for the same *uid* can be tied together for signcryption and designcryption, and the secret keys generated for different users cannot be combined.

Let S_c denote the set of colluders, and $\widetilde{U^d}$ is the combined decryption attribute set of S_c. Recall that the message $\mathcal{M}$ is blinded by $\prod_{k\in I_A^e} \Delta_k{}^w = \prod_{k\in I_A^e} e(g,g)^{\alpha_k w}$. It is infeasible to directly reconstruct $\prod_{k\in I_A^e} e(g,g)^{\alpha_k w}$ due to the blindness of α_k and the hardness of discrete logarithm assumption. Thus the colluders have to compute $\prod_{k\in I_A^e} e\left(K^d_{uid,k}, C_1\right)$ and have to cancel the redundant element $e(\theta,g)^{wN_A^e d_{uid}} = \prod_{k\in I_A^e} e(g,g)^{whd_{uid}}$, where $\theta = g^h$. Due to BDH assumption, the only way to cancel $e(\theta,g)^{wN_A^e d_{uid}}$ is to compute the denominator $\prod_{k\in I_A^e}\prod_{i\in I_{A_k}}\left[e\left(C'_{2,i}\theta^{C''_{2,i}}A_{\rho_e(i)}^{-D''_i}, g^{d_{uid}}\right)e\left(D'_i g^{D''_i}, F^d_{uid,\rho_e(i)}\right)\right]^{\sigma_i N_A^e}$ in *PartialDecryption* algorithm, which means $F^d_{uid,\rho_e(i)} = A^{d_{uid}}_{\rho_e(i)}$ with the same d_{uid} holds for all $\rho_e(i) \in \widetilde{U^d}$. However, since the colluders are individually unauthorized for decryption, none of the colluders holds $A^{d_{uid}}_{\rho_e(i)}$ for all $\rho_e(i) \in \widetilde{U^d}$ simultaneously. Moreover, since the secret key cannot be replaced, converted or combined, $\left\{A^{d_{uid}}_{\rho_e(i)}\right\}_{U_{uid}\in S_c, \rho_e(i)\in\widetilde{U^d}}$ are associated with different d_{uid}. Hence the colluders cannot successfully decrypt the ciphertext even though $\widetilde{U^d}$ satisfies the encrypt predicate defined in the ciphertext. Specifically, according to Theorems 1 and 2, we can prove that our scheme guarantees the collusion resistance under q-PBDHE assumption in Theorem 4.

Theorem 4. *The proposed data access control scheme is collusion resistance.*

Proof. For the designcryptor, we state that the security game defined in Definition 9 implies the collusion resistance. Suppose that S_c denotes the set of colluders who are unauthorized for decryption and $\widetilde{U^d} = \cup\left\{\widetilde{U_i^d}\right\}_{i\in S_c}$. If the colluders can decrypt CT^* when $\mathcal{R}_e^*\left(\widetilde{U^d}\right) = 1$, then the algorithm $\mathcal{B}$ which can solve the q-PBDHE assumption can be constructed as follows.

In the initialization phase, the challenger sets $\mathcal{R}_e^*$ as the selected challenge encryption predicate. In $\mathcal{O}^{sk}$, $\mathcal{A}$ queries for the secret decryption key corresponding to the colluder's individual attribute set $\widetilde{U_i^d}$. Since the colluders are individually unauthorized for decryption, we have $\mathcal{R}_e^*\left(\widetilde{U_i^d}\right) = 0$, which satisfies the constraint of $\mathcal{O}^{sk}$ defined in Definition 8. Then in challenge phase, the challenger encrypts $\mathcal{M}_{\lfloor}$ under $\mathcal{R}_e^*$. If the colluders can decrypt the ciphertext, then $\mathcal{A}$ can guess the bit $\hat{\lfloor}$, and thus $\mathcal{B}$ can solve the q-PBDHE assumption with non-negligible probability.

Similarly, for the signcryptor, the Theorem 2 guarantees that no colluders such as users, fog nodes 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 9 and break q-PBDHE assumption.

Therefore, the colluding users, fog nodes, and cloud server cannot sign or decrypt the data, and our OMDAC-ABSC scheme guarantees collusion resistance. □

6.5. Revocation Security

Assume the attribute x of U is revoked from AA_k. AA_k issues the update secret keys $dUK_x = g^{d_{uid}(\varphi'_x - \varphi_x)}, sUK_x = g^{s_{uid}(\varphi'_x - \varphi_x)}$ and sends the keys to the non-revoked users. dUK_x and sUK_x are associated with the secret value d_{uid}, s_{uid} chosen by CA and attribute version key φ'_x, φ_x chosen by AA_k. Therefore, due to the blindness of $d_{uid}, s_{uid}, \varphi'_x$, and φ_x, the revoked user U cannot update his/her secret signing or decryption key, even though he/she can corrupt some attribute authorities (not the authority AA_k corresponding to x) or collude with the non-revoked user.

Theorem 5. *Our OMDAC-ABSC scheme guarantees the forward and backward revocation security.*

Proof.

Forward Security. If there exists i such that $\rho_s(i) = x$, the newly joined user can sign the plaintext and generate the signature component S_2 associated with A'_x, which is the same as the updated attribute public key of AA_k. Thus the $Verify$ algorithm holds if user's signing attributes satisfy the signing predicate. Otherwise, the newly joined user's secret decryption keys are all associated with A'_x, which is the same as that in the components $C'_{2,i}$. Thus the newly joined user can decrypt ciphertext if his/her attribute set satisfies the embedded encryption predicate.

Backward security. If there exists i such that $\rho_s(i) = x$, and the revoked user reverse the signature component S_2 back to the non-revoked state which is associated with A_x, then the $Verify$ algorithm cannot hold since the attribute public key of AA_k has been updated to A'_x.

Otherwise, assume CT'_{old} denotes the ciphertext which is updated from CT_{old} in attribute revocation phase, we have $C'_{2,i} = \theta^{\lambda'_i} A'^{-(r'_i + r''_i)}_{\rho_e(i)}$ and $D'_i = g^{r'_i + r''_i}$. It is hard for the revoked user to cancel cUK_i and $g^{r''_i}$ since they are associated with the values φ'_x, φ_x which are secretly chosen by AA_k and r''_i randomly picked by cloud server. Therefore, the revoked user cannot reverse the CT'_{old} back to CT_{old}.

For the ciphertext CT_{new} which is uploaded after the attribute revocation phase, we have $C'_{2,i} = \theta^{\lambda'_i} A'^{-r'_i}_{\rho_e(i)}$ for i such that $\rho_e(i) = x$. The revoked user cannot transform these components into the ones associated with $A_{\rho_e(i)}$ due to the blindness of the attribute version keys φ'_x, φ_x chosen by AA_k and random element r'_i picked by fog node. Therefore, our OMDAC-ABSC scheme guarantees the forward and backward revocation security. □

7. Scheme Analysis

7.1. Security and Functionality

In this subsection, we detail the comprehensive security and functionality comparison among the proposed scheme and some MA-ABE schemes [21–26], CP-ABSC schemes [12–15] and ABE based schemes used for fog computing [16–20] in Tables 1–3. 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 1. Security and Functionality Comparison of MACP-ABE Schemes.

Schemes	**[21]**	**[22]**	**[23]**	**[24]**	**[25]**	**[26]**	**Ours**
Collusion Resistance	✕	✓	✓	✓	✓	✓	✓
Standard Model	✓	✕	✕	✕	✕	✕	✓
Encryption Predicate	MBF	MBF	MBF	MBF	MBF	MBF	MBF
Encryption Outsourcing	✕	✕	✕	✕	✓	✕	✓
Decryption Outsourcing	✕	✓	✕	✕	✓	✓	✓
Anonymous Authentication	✕	✕	✕	✓	✕	✕	✓
Attribute Revocation	✕	✓	✕	✕	✕	✓	✓

Table 2. Security and Functionality Comparison of CP-ABSC Schemes.

Schemes	[12]	[13]	[14]	[15]	**Ours**
Collusion Resistance	✓	✓	✓	✓	✓
Standard Model	✓	×	✓	×	✓
Signcryptor Privacy	✓	✓	✓	×	✓
Signing Predicate	MBF	MBF	MBF	MBF	MBF
Encryption Predicate	MBF	MBF	MBF	TG	MBF
Signcryption Outsourcing	×	×	×	×	✓
Designcryption Outsourcing	×	×	×	×	✓
Multi-Authority	×	×	×	×	✓
Public Verifiability	×	×	✓	✓	✓
Attribute Revocation	×	×	×	×	✓

Table 3. Security and Functionality Comparison of ABE based Schemes for Fog Computing.

Schemes	[16]	[17]	[18]	[19]	[20]	**Ours**
Collusion Resistance	✓	✓	✓	✓	✓	✓
Standard Model	×	×	×	✓	✓	✓
Encryption Predicate	TG	MBF	TG	TG	MBF	MBF
Encryption Outsourcing	✓	✓	×	✓	×	✓
Decryption Outsourcing	✓	✓	✓	✓	✓	✓
Multi-Authority	×	✓	×	×	×	✓
Anonymous Authentication	✓	×	×	×	×	✓
Attribute Revocation	×	×	×	✓	×	✓

Tables 1–3 show that our scheme supports many useful properties, such as multi-authority, collusion resistance, computation outsourcing, anonymous authentication, expressiveness, public verifiability and attribute revocation. Our scheme also realizes the security in the standard model.

7.2. Asymptotic Complexity and Performance

This section numerically analyzes the asymptotic complexity and performance of the proposed OMDAC-ABSC scheme against some MACP-ABE schemes [21,22,24–26], CP-ABSC schemes [12–15], and ABE based schemes [16–20] used for fog computing in terms of the size of secret key, ciphertext and update key, and computation overhead (exponentiations and pairing computations) of *Signcryption*, *DeSigncryption* and *UpCiphertext* algorithms. We focus on the computation overhead on the user side because of the limited computation resources. For simplicity, in asymptotic complexity analysis we ignore the cost time of Hash functions and operations in $\mathbb{Z}_p$. Table 4 summarizes the notations used in this section.

Table 4. Notations.

Notations	**Meaning**
$T^e_{\mathbb{G}}/T^e_{\mathbb{G}_T}$	Running time required for one exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$.
T^p	Running time for one pairing operation.
N_A	Number of involved authorities.
$\lvert\mathbb{G}\rvert/\lvert\mathbb{G}_T\rvert/\lvert\mathbb{Z}_p\rvert$	Size of the element in $\mathbb{G}$, $\mathbb{G}_T$, and $\mathbb{Z}_p$.
l_e /l_s	Number of required attributes in decryption and verification.
$\lvert\widetilde{U^d}\rvert$	Number of decryption attributes.
$\lvert\widetilde{U}\rvert$	Number of signing and decryption attributes.
S	Least interior nodes satisfying the access policy tree.

7.2.1. Asymptotic Complexity

Table 5 details the storage comparison on MACP-ABE schemes. It is clear that the size of the secret decryption key in our OMDAC-ABSC is larger than that in [24,25] due to the components $\left\{K^d_{uid,k}\right\}_{k\in I_A}$. Table 5 also illustrates that the size of ciphertext in our scheme is larger than that in [21,22,26], and has the advantage over [25]. Since our scheme supports public verification of signcryptor's attributes, the ciphertext contains the signature components $\{S_{1,i}\}_{i\in[\ell_s]}, S_2$, which result in a reducing $(1+l_s)|\mathbb{G}|$ of storage overhead. Although the scheme in [24] can also verify the data owner's attributes, it requires $2+2l_s$ signature group elements and is not publicly verifiable since it needs the plaintext message in verification algorithm. Additionally, both of our scheme and [25] requires the data owner to compute the ciphertext components $\left\{C''_{2,i}, D''_i\right\}_{i\in[\ell_e]}$ when performing *User_Signcryption* algorithm. This cost is $2l_e|\mathbb{Z}_p|$.

For attribute revocation, it is apparent that our scheme and [22] incur relatively the same storage overhead. Compared with [26], our scheme requires the attribute authority supervising the revoked attribute x to compute the ciphertext update key $cUK = \left\{\left(D'_i\right)^{\varphi_x-\varphi'_x}\right\}_{\rho_e(i)=x}$ when x is selected as an encryption attribute, and thus incurs at most l_e group elements, whereas the scheme [26] only sends $\varphi_x - \varphi'_x$ to the cloud. However, as shown in [22], DAC-MACS [26] cannot guarantee backward revocation security.

Table 6 shows the computation overhead comparison of *Signcryption* and *Decryption* algorithms on the user side and *UpCiphertext* algorithm on the cloud. From the table, we can see that the encryption and decryption cost of our scheme are both irrelevant to the number of attributes. In data signcryption phase, our scheme asks fog nodes to compute and generate part of the ciphertext which is associated with the signing and encryption predicates. Thus the signcryption cost of data owner can be reduced as $T^e_{\mathbb{G}_T} + 3T^e_{\mathbb{G}}$ in encryption and $(l_s + l + 2)T^e_{\mathbb{G}}$ in signing. In decryption phase, our scheme only incurs the cost of one exponentiation in $\mathbb{G}_T$. Hence the performance of ours is better than most schemes except for [25]. To guarantee the CCA security in the standard model (see Theorem 1), our scheme requires the data owner to compute the components C_1 and C_3, which results in a slight reducing $3T^e_{\mathbb{G}}$ of computation efficiency compared with [25]. However, our scheme performs better than [25] with respect to attribute revocation. Moreover, the DAC-MACS scheme in [26] only incurs the cost of l_e exponentiations in $\mathbb{G}$ in ciphertext update phase, while our scheme incurs twice this cost. The reason is that we re-randomize $C'_{2,i}$ and D'_i in *UpCiphertext* algorithm to realize the backward revocation security.

If we set $N_A = 1$, then the proposed scheme is a traditional CP-ABSC scheme. In Table 7, we compare the asymptotic complexity of OMDAC-ABSC with CP-ABSC schemes [12–15]. As seen from Table 7, the size of the secret key is linear to the size of the attribute universe, which is not different between our scheme and others. Our scheme incurs a slight reducing $l_e|\mathbb{G}| + 2l_e|\mathbb{Z}_p|$ of storage overhead than other schemes on the ciphertext. The reason is that we add $\left\{C''_{2,i}, D'_i, D''_i\right\}_{i\in[\ell_e]}$ to realize the attribute revocation and outsourced encryption, which are not considered in other schemes. Meanwhile, the ciphertext in our scheme consists of $l_s + 1$ group elements for verification, while that in [12] is $2l_s + 2$. Table 7 also indicates that our scheme incurs less computation overhead of *DeSigncryption* on the user side than do the other schemes since most costly job of decryption is outsourced to fog nodes. Compared with [14], our construction requires $3 + l_s$ pairing operations in total in decryption (user side) and verification, whereas in [14], $(5 + l_s)$ pairings are needed. Moreover, since our scheme supports public verifiability, the verification algorithm can be performed by a trusted intermediate party. Thus the user can recover the plaintext within one exponentiation in $\mathbb{G}_T$. In contrast, the schemes in [12,13,15] are not publicly verifiable, and thus incur large amount of computation overhead in verification and decryption on the user side. In [12,13], the number of pairings is linear to the number of attributes. In [15], although the size of ciphertext is only $6|\mathbb{G}|$, eight pairings are required to recover the plaintext.

Table 8 details the storage and computation overhead comparison of our scheme and some ABE based data access control schemes for fog computing. Since the schemes in [16,18–20] do not support multi-authority, we set $N_A = 1$ in our scheme for comparison. It is illustrated that the size of secret decryption key in OMDAC-ABSC is less than that in others. Since our scheme enables any trusted third party to verify the data owner's attributes, the ciphertext contains the signature components $\{S_{1,i}\}_{i\in[\ell_s]}, S_2$, which result in a reducing $(1+l_s)|\mathbb{G}|$ of storage overhead on the cloud side. For encryption, on the user side, our scheme incurs $3T^e_{\mathbb{G}}$ to compute C_1 and C_3 and thus is less efficient than [17]. However, our scheme guarantees the CCA security, which is not considered in [17]. For decryption, on the user side, our scheme and [17] both incurs less computation overhead than other schemes since the two schemes only require one exponentiation in $\mathbb{G}_T$. Therefore, our scheme is efficient from a computation point of view.

Table 5. Storage Comparison of MACP-ABE based Schemes.

Schemes	Secret Decryption Key	Ciphertext	Update Key	
			Secret Key Update	Ciphertext Update Key
[21]	$\left(6N_A+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+(3N_A+2l_e)\|\mathbb{G}\|$	-	-
[22]	$\left(2+2\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+(3+N_A+3l_e)\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$l_e\|\mathbb{G}\|$
[24]	$\left\|\widetilde{U^d}\right\|\|\mathbb{G}\|$	$(l_e+1)\|\mathbb{G}_T\|+(2+2l_e+2l_s)\|\mathbb{G}\|$	-	$l_e\|\mathbb{G}_T\|$
[25]	$\left\|\widetilde{U^d}\right\|\|\mathbb{G}\|$	$(3l_e+1)\|\mathbb{G}_T\|+4l_e\|\mathbb{G}\|+2l_e\|\mathbb{Z}_p\|$	$(2\|\mathbb{G}\|+\|\mathbb{G}_T\|)l_e$	$l_e\|\mathbb{G}_T\|$
[26]	$\left(2N_A+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+(1+3l_e)\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{Z}_p\|$
Ours	$\left(N_A+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+(2+2l_e+l_s)\|\mathbb{G}\|+2l_e\|\mathbb{Z}_p\|$	$\|\mathbb{G}\|$	$l_e\|\mathbb{G}\|$

Table 6. Time Comparison of *Signcryption*, *Decryption* and *UpCiphertext*.

Schemes	*Signcryption* (User Side)		*Decryption* (User Side)	*UpCiphertext*
	Encryption	Signing		
[21]	$N_A T^e_{\mathbb{G}_T}+(3N_A+3l_e)T^e_{\mathbb{G}}$	-	$(4N_A+2l_e)T^p+(N_A+l_e)T^e_{\mathbb{G}_T}$	-
[22]	$2N_A T^e_{\mathbb{G}_T}+(3+N_A+4l_e)T^e_{\mathbb{G}}$	-	$T^e_{\mathbb{G}_T}$	$2l_e T^e_{\mathbb{G}}$
[24]	$(1+2l_e)T^e_{\mathbb{G}_T}+3l_e T^e_{\mathbb{G}}$	$(2+3l_s+2l_s n_s)T^e_{\mathbb{G}}$	$2l_e T^p$	$(1+2l_e)T^p$
[25]	$T^e_{\mathbb{G}_T}$	-	$2T^e_{\mathbb{G}_T}$	$l_e T^p$
[26]	$N_A T^e_{\mathbb{G}_T}+(1+5l_e)T^e_{\mathbb{G}}$	-	$T^e_{\mathbb{G}_T}$	$l_e T^e_{\mathbb{G}}$
Ours	$T^e_{\mathbb{G}_T}+3T^e_{\mathbb{G}}$	$(l_s+l+2)T^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}$	$2l_e T^e_{\mathbb{G}}$

Table 7. Asymptotic Complexity Comparison of CP-ABSC based schemes.

Schemes	Secret Key	Ciphertext	*DeSigncryption*	
			Verification	Decryption (User Side)
[12]	$\left(4+\left\|\widetilde{U}\right\|\right)\|\mathbb{G}\|$	$\begin{pmatrix}4+l_e\\+2l_s\end{pmatrix}\|\mathbb{G}\|$	$(2+2l_s)T^p+(2l_s+3)T^e_{\mathbb{G}}$	$(2+2l_e)T^p+l_e T^e_{\mathbb{G}_T}$
[13]	$\left(4+\left\|\widetilde{U}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+\begin{pmatrix}5+l_e\\+l_s+\\n_s\end{pmatrix}\|\mathbb{G}\|$	$(6+l_s n_s+2l_s)T^p+l_s T^e_{\mathbb{G}_T}+2l_s n_s T^e_{\mathbb{G}}$	$2l_e T^p+l_e T^e_{\mathbb{G}_T}$
[14]	$\left(4+\left\|\widetilde{U}\right\|\right)\|\mathbb{G}\|$	$\begin{pmatrix}5+l_e\\+l_s\end{pmatrix}\|\mathbb{G}\|$	$(3+l_s)T^p+(l_s+l+1)T^e_{\mathbb{G}}$	$2T^p+3l_e T^e_{\mathbb{G}}$
[15]	$(6l_e+3l_s)\|\mathbb{G}\|$	$6\|\mathbb{G}\|$	$6T^p$	$2T^p+(4l_e+4l_e^2)T^e_{\mathbb{G}}$
Ours	$\left(2+\left\|\widetilde{U}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+(2+2l_e+l_s)\|\mathbb{G}\|+2l_e\|\mathbb{Z}_p\|$	$(3+l_s)T^p+(l_s+l+1)T^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}$

Table 8. Storage and Computation Overhead Comparison of ABE based Schemes for Fog Computing.

Schemes	Secret Decryption Key	Ciphertext	Encryption		Decryption	
			Fog Node	User	Fog Node	User
[16]	$\left(2+2\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+\left(3+2\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\left(2+2\left\|\widetilde{U^d}\right\|\right)T^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}+3T^e_{\mathbb{G}}$	$\left(2\left\|\widetilde{U^d}\right\|+4\right)T^p+(2S+2)T^e_{\mathbb{G}_T}$	T^p
[17]	$\left(2+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+(1+2l_e)\|\mathbb{G}\|+(2+2l_e)\|\mathbb{Z}_p\|$	$4l_e T^e_{\mathbb{G}}+T^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}$	$(1+2l_e)T^p+(1+7l_e)T^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}$
[18]	$\left(1+2\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\left(2+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|+2l\|\mathbb{Z}_p\|$	-	$T^e_{\mathbb{G}_T}+\left(2+\left\|\widetilde{U^d}\right\|\right)T^e_{\mathbb{G}}$	$\left(3+3\left\|\widetilde{U^d}\right\|\right)T^p$	$4T^e_{\mathbb{G}_T}$

Table 8. *Cont.*

[19]	$\left(3+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+\left(3+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\left(2+\left\|\widetilde{U^d}\right\|\right)T^e_{\mathbb{G}}$	$2T^e_{\mathbb{G}_T}+4T^e_{\mathbb{G}}$	$\left(\left\|\widetilde{U^d}\right\|+2\right)T^p+(2S+2)T^e_{\mathbb{G}_T}$	T^p
[20]	$\left(2+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$(2+2l_e)\|\mathbb{G}\|+l\|\mathbb{Z}_p\|$	-	$T^e_{\mathbb{G}_T}+(3+3l_e)T^e_{\mathbb{G}}$	$(1+2l_e)T^p+l_eT^e_{\mathbb{G}_T}$	$T^e_{\mathbb{G}_T}+2T^e_{\mathbb{G}}$
Ours	$\left(1+\left\|\widetilde{U^d}\right\|\right)\|\mathbb{G}\|$	$\|\mathbb{G}_T\|+(2+2l_e+l_s)\|\mathbb{G}\|+2l_e\|\mathbb{Z}_p\|$	$3l_eT^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}+3T^e_{\mathbb{G}}$	$(1+2l_e)T^p+l_eT^e_{\mathbb{G}_T}+3l_eT^e_{\mathbb{G}}$	$T^e_{\mathbb{G}_T}$

7.2.2. Performance

We implement the whole architectures of MACP-ABE schemes [21,22,24–26], CP-ABSC schemes [12–15] and our scheme 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 $y^2 = x^3 + x$ over a 512-bit finite field. The computation cost for one pairing operation is 2.9 ms, and that of exponentiation on $\mathbb{G}$ and $\mathbb{G}_T$ are 0.7 and 0.2 ms, respectively. Each value in Figures 3–8 is the mean of 10 simulation trials.

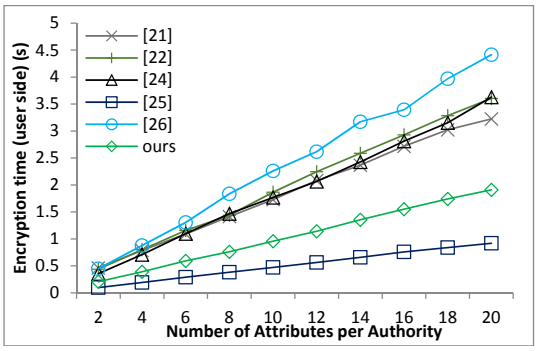

Figure 3. Encryption (user side).

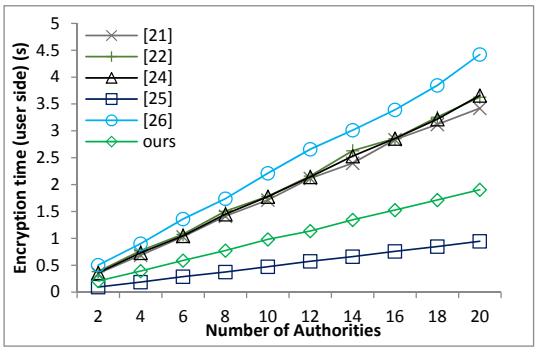

Figure 4. Encryption (user side).

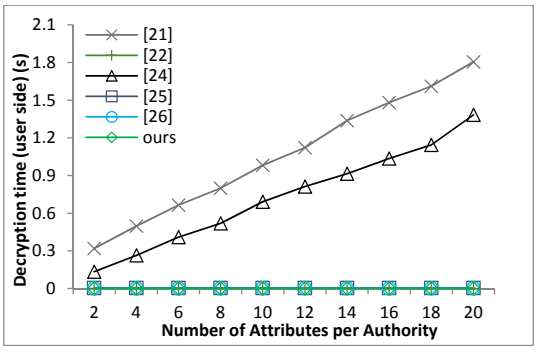

Figure 5. Decryption (user side).

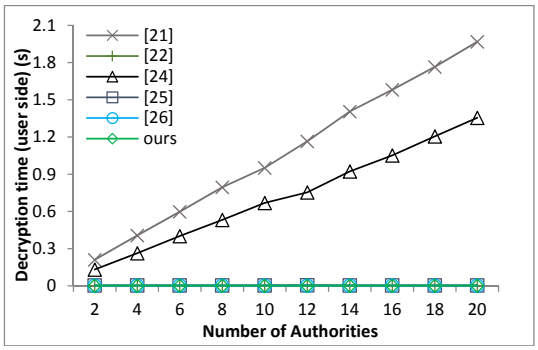

Figure 6. Decryption (user side).

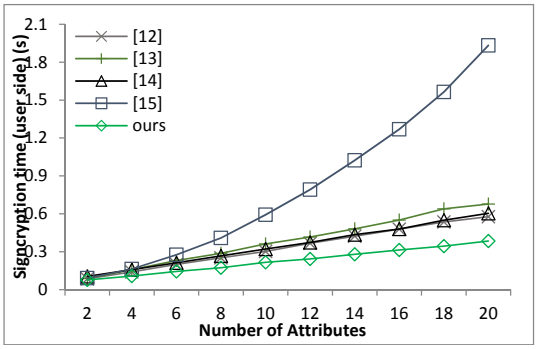

Figure 7. Signcryption (user side).

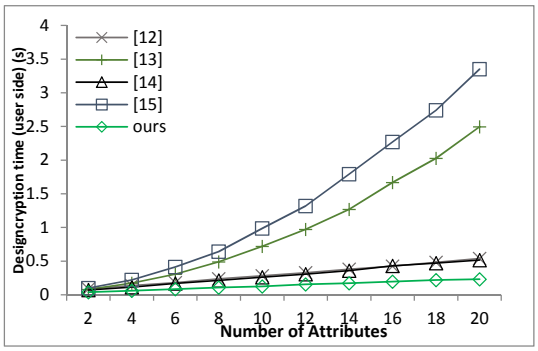

Figure 8. Designcryption (user side).

For simplicity, suppose each user holds the same number of attributes N_{AA} from each authority and $\left|\widetilde{AA_k}\right| = \left|\widetilde{AA_k} \cap \widetilde{U^d}\right| = N_{AA}$, where $k \in [N_A]$. $N_A^e = N_A^s = N_A$. Then, in signcryption we set $l_e = l_s = N_{AA} \times N_A$, and thus the comparison of computation overhead of *Signcryption* (without signing) and *Decryption* algorithms on the user side between our scheme and [21,22,24–26] can be conducted according to parameters N_A and N_{AA}. We also generate the signing and encryption predicates as AND-gate in the form of $(a_1$ and a_2 and ... and $a_{l_s})$ and $(a_1$ and a_2 and ... and $a_{l_e})$. In Figures 3 and 5, we set $N_A = 10$, while in Figures 4 and 6, we assume $N_{AA} = 10$. During the comparison between our scheme and the ones in [21,22,24–26], we do not take into account the signing protocol since the schemes in [21,22,25,26] do not support attribute-based signature.

Figures 3 and 4 show that the encryption algorithm in our scheme is more efficient than that in [21,22,24,26]. The reason is that the most costly job of encryption has been outsourced to the fog nodes. Although our scheme incurs more computation overhead than the one in [25], we realize CCA security in the standard model and attribute-level revocation. Figures 5 and 6 give the comparison

of decryption time on the user side. It is illustrated that the performance of our scheme is relatively the same as that of [22,25,26], and is better than that of [21,24] because our scheme only incurs one exponentiation and one multiplication in $\mathbb{G}_T$.

Assume that $N_A = 1$ and $\ell_e = \ell_s = N_{AA}$. Figures 7 and 8 describe the comparison of computation overhead of *Signcryption* and *DeSigncryption* algorithms among the schemes [12–15] and ours. It is clear that our *Signcryption* algorithm incurs less computation overhead than other schemes because of the outsourced signcryption. Since our scheme and Y. Sreenivasa's scheme [14] are publicly verifiable, the $Verify(PP, CT)$ algorithm can be outsourced to a trusted party, and then our scheme needs only one exponentiation and one multiplication in $\mathbb{G}_T$ on the user side to recover the plaintext message.

Moreover, we simulate the schemes in [16–20] and our scheme on an android phone (MEIZU m1 note platform with an ARM Cortex A53-based processor MT6752@1.7 GHz, Android 5.1, and 2 GB RAM) as user's IoT device and a laptop (2.6 GHz processor, Ubuntu system 14.04, and 4G RAM) as the fog node. The underlying curve for pairings is also Type A curve in JPBC 2.0.0 [18], where the running time for pairing is 6 ms in Ubuntu system and 175 ms in Android. For comparison, we set $N_A = 1$ in our scheme and do not consider the signing protocol since the schemes in [16,18–20] do not support multi-authority and the schemes in [16–20] do not support attribute-based signature. Figures 9 and 10 show the comparison of computation overhead of encryption algorithm and Figures 11 and 12 show the comparison of decryption algorithm. The results are the average number of 10 runs. In Figure 9 we only compare the cost time of encryption on fog node between ours and the schemes in [16,17,19] since the schemes in [18,20] do not support encryption outsourcing.

It is illustrated in Figure 10 that the computation time of encryption algorithm on data owner in our scheme is basically the same as that in [17], and is smaller than that in [18,20] because of the encryption outsourcing. Compared with [16,19], the encryption algorithm in our scheme incurs slightly more computation overhead since our scheme requires the data owner to sample $\left\{C''_{2,i}, D''_i\right\}_{i\in[\ell_e]}$ and perform one Hash function $\pi = H_1(C_1)$ (we do not take into account the Hash functions H_2 and H_3 here since they are involved in signing protocol). However, the encryption time is approximately 0.14–0.8 s, which is acceptable to the end users.

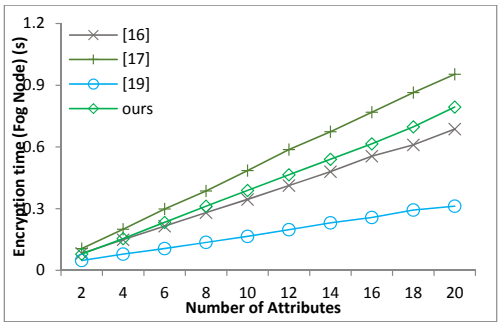

Figure 9. Encryption (fog node side).

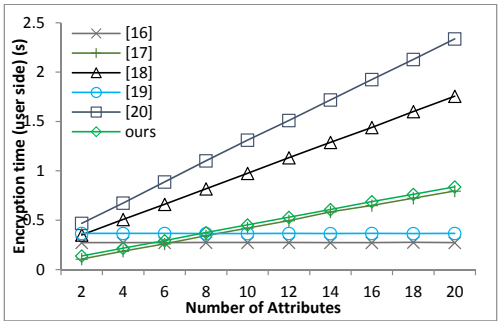

Figure 10. Encryption (user side).

Figure 11 indicates that on the fog node side, the decryption algorithm of our scheme incurs more computation overhead than the schemes in [16,18–20]. However, Figure 12 shows that our scheme performs better than other schemes except for [17] in efficiency of decryption time on the user side. This is because our scheme outsources the most computation-consuming job of decryption to the fog node and only incurs the cost of one exponentiation and one multiplication in $\mathbb{G}_T$ on the user side. In Figure 11, the decryption time of our scheme one the fog node is approximately 0.1–1 s, which increases almost linearly with the number of attributes.

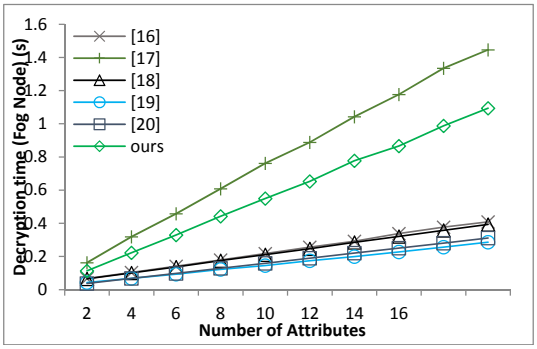

Figure 11. Decryption (fog node side).

However it is shown in Figure 12 that the running time of *FullDecryption* algorithm is nearly 0.03 s, which is acceptable for the end user. Since our scheme is public verifiable, the verification can be performed on any trusted third party and does not increase the computation burden of the user. Additionally, Huang et al. [16] and Zhang et al. [19] only support threshold access policy, while our scheme supports any monotone Boolean function. Overall, our scheme performs well in encryption and decryption on the user side and supports additional useful properties such as multi authorities, anonymous authentication, and public verifiability.

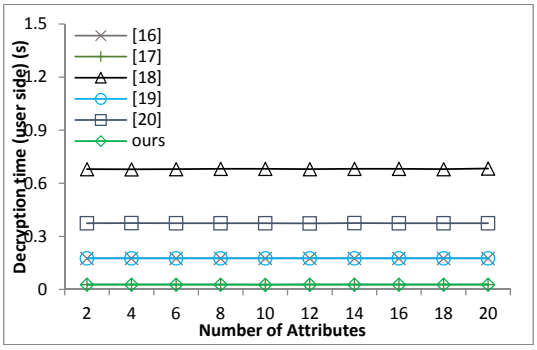

Figure 12. Decryption (user side).

8. Conclusions

In this paper, we proposed OMDAC-ABSC scheme for data sharing in fog computing system. The proposed scheme realizes the security in the standard model and supports many practical properties, such as confidentiality, fine-grained access control, anonymous authentication, attribute revocation, and public verifiability. The heavy computation operations of the signcryption and designcryption algorithms are outsourced to the fog nodes making our scheme more efficient and more suitable for fog computing than the existing ABSC schemes. The security analysis, asymptotic complexity, and performance comparisons indicate that our construction hits a good balance between the security and overhead efficiency.

One problem with outsourced decryption is to verify that whether the partial decryption performed by fog nodes is correct. In ABE scheme, verifiable outsourcing has been adopted to overcome this problem, as in [17,30–32]. A similar technique can be used in our ABSC construction to address verifiable outsourcing, which will be our future work. Moreover, realizing a fully secure MACP-ABSC based access control scheme instead of a selectively secure scheme will be another challenge.

Author Contributions: Q.X. and C.T. conceived the scheme. Q.X. designed the scheme, analyzed the data and wrote the paper. W.Z. and F.C. performed the experiments. Z.F. and Y.X. modified the manuscript.

Funding: This research was funded by [National Key Research and Development Program of China] grant number [2017YFC0803702] and grant number [2017YFB0802302].

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Appendix A

Table A1. Notations used in OMDAC-ABSC scheme.

Notations	Meaning
S_A, S_U	Set of attribute authorities and the set of users.
N_A	Number of attribute authorities.
uid/aid	Identity of user/authority.
do/du	Identity of data owner (signcryptor)/data user (designcryptor).
$\widetilde{U}$	Attribute set of the user.
$\widetilde{AA}$	Attribute set of the attribute authority.
$\widetilde{U^d}/\widetilde{U^s}$	Decryption/Signing attribute set of the user. $\widetilde{U} = \widetilde{U^d} \cup \widetilde{U^s}$
I_A^s/I_A^e	Set of the indexes of the authorities involved in signing/encryption. $I_A = I_A^s \cup I_A^e$. $N_A^s = \left\|I_A^s\right\|$. $N_A^e = \left\|I_A^e\right\|$.
H_1, H_2, H_3	Collision resistant hash functions.
$\mathcal{R}_s(\mathrm{M}_s, \rho_s)$	Signing and Encryption Predicate
$\mathcal{R}_{e,j}(\mathrm{M}_e, \rho_e)$	
$\mathrm{M}_s^i/\mathrm{M}_e^i$	ith row of $\mathrm{M}_s^i/\mathrm{M}_e^i$.
$\mathrm{M}_s^{(i,k)}/\mathrm{M}_e^{(i,k)}$	(i,k)th element of $\mathrm{M}_s^i/\mathrm{M}_e^i$.
ℓ_s/ℓ_e	Number of rows of $\mathrm{M}_s/\mathrm{M}_e$ of $\mathcal{R}_s/\mathcal{R}_e$.
n_s/n_e	Number of columns of $\mathrm{M}_s/\mathrm{M}_e$ of $\mathcal{R}_s/\mathcal{R}_e$.
ℓ_m	Maximum value of ℓ_s.
$PP = \left\{ \begin{array}{c} g, \theta, \gamma_1, \gamma_2, \\ \{k_0, k_1, \ldots, k_l\}, \\ \{V_1, V_2 \ldots, V_{\ell_m}\} \end{array} \right\}$	Public parameters.
s_{uid}, d_{uid}	Secret values chosen by CA for each user with identity uid.
φ_x, φ'_x	Attribute version key for attribute x.
A_x, A'_x	Attribute public key for attribute x.
$PPK_{uid} = \left\{ \begin{array}{c} g^{s_{uid}}, g^{d_{uid}}, \\ \{V_i^{s_{uid}}\}_{i\in[\ell_m]} \end{array} \right\}$	Partial public key generated by CA for each user U_{uid}.
$PPK_{aid} = \Delta_{aid}$	Partial public key generated by CA for each attribute authority AA_{aid}.
$PK_{uid} = \left\{ \begin{array}{c} g^{s_{uid}}, g^{d_{uid}}, g^{1/z_{uid}}, \\ \theta^{z_{uid}}, g^{z_{uid}}, \\ \{V_i^{s_{uid}}\}_{i\in[\ell_m]} \end{array} \right\}$	Public key of the user U_{uid}.
$SK_{uid} = z_{uid}$	Secret key of the user U_{uid}.
$PK_{aid} = \left\{ \begin{array}{c} \Delta_{aid}, \\ X_{aid}, Y_{aid}, Z_{aid}, \\ \{A_x\}_{x\in\widetilde{AA_{aid}}} \end{array} \right\}$	Public key of the authority AA_{aid}.
$SK_{aid} = \left\{\beta_{aid}, \gamma_{aid}, \{\varphi_x\}_{x\in\widetilde{AA_{aid}}}\right\}$	Secret key of the authority AA_{aid}.
$PK_{uid,aid} = \left\{ \begin{array}{c} PK^1_{uid,aid}, PK^2_{uid,aid}, \\ PK^3_{uid,aid} \end{array} \right\}$	Public key for each pair of user U_{uid} and authority AA_{aid}.
$F^s_{uid,x}/F^d_{uid,x}$	Signing/Decryption attribute key of U_{uid} for attribute x.
$SK^s_{uid,aid} = \left\{ \begin{array}{c} K^s_{uid,aid}, \\ \{F^s_{uid,x}\}_{x\in\widetilde{U^s}\cap\widetilde{AA_{aid}}} \end{array} \right\}$	Secret signing key of U_{uid} generated by AA_{aid}.
$SK^d_{uid,aid} = \left\{ \begin{array}{c} K^d_{uid,aid}, \\ \{F^d_{uid,x}\}_{x\in\widetilde{U^s}\cap\widetilde{AA_{aid}}} \end{array} \right\}$	Secret decryption key of U_{uid} generated by AA_{aid}.

Table A1. *Cont.*

$PSK^{s}_{uid,aid} = \left\{ PS_{uid,aid}, PV_{uid}, \left\{ PF^{1}_{uid,x}, PF^{2}_{uid,x} \right\}_{x \in \widetilde{U^{s}} \cap \widetilde{AA_{aid}}}, \left\{ V_i^{z_{uid}}, V_i^{s_{uid} z_{uid}} \right\}_{i \in [\ell_m]} \right\}$	Proxy secret key for signing.
$PSK^{d}_{uid,aid} = SK^{d}_{uid,aid}$	Proxy secret key for decryption.
$sUK_{uid,x}, dUK_{uid,x}$	Signing and decryption update keys for attribute x.
cUK, sUK	Ciphertext update keys.
$\overrightarrow{a}$ / $\overrightarrow{b}$	Vectors chosen by fog node for signing protocol.
s'_{uid}	Secret value randomly chosen by fog node to randomize proxy secret key.
$\left\{ r'_1, r'_2, \ldots, r'_{\ell_e} \right\}, \left\{ \lambda'_1, \lambda'_2, \ldots, \lambda'_{\ell_e} \right\}, w'$	Random values chosen by fog node for signcrypion.
$\{ r_1, r_2, \ldots, r_{\ell_e} \}, \{ \varepsilon_2, \ldots, \varepsilon_{n_e} \}, w$	Random values chosen by data owner for signcrypion.
$thre_{tt}$	Time threshold.
$\{ \tau_2, \tau_3, \ldots, \tau_{n_s} \}$	Random values used for verification. $\omega_i = (1, \tau_2, \tau_3, \ldots, \tau_{n_s}) \cdot \mathrm{M}^{i}_{s}$
$\{ \sigma_1, \sigma_2, \ldots, \sigma_{\ell_e} \}$	Random values chosen by fog node for designcryption.
CT'	Partial ciphertext computed by fog node in signcryption.
CT^{p}	Partial ciphertext computed by fog node in designcryption.
CT	Ciphertext.

References

1. Rong, C.M.; Nguyen, S.T.; Jaatun, M.G. Beyond lightning: A survey on security challenges in cloud computing. *Comput. Electr. Eng.* **2013**, *39*, 47–54. [CrossRef]
2. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 13–17 August 2012.
3. Stojmenovic, I.; Wen, S.; Huang, X.Y.; Luan, H. An overview of fog computing and its security issues. *Concurr. Comput. Pract. Exp.* **2016**, *28*, 2991–3005. [CrossRef]
4. Ahmad, M.; Amin, M.B.; Hussain, S.; Kang, B.H.; Cheong, T.; Lee, S.Y. Health fog: A novel framework for health and wellness applications. *J. Supercomput.* **2016**, *72*, 3677–3695. [CrossRef]
5. Yang, Y.J.; Liu, J.K.; Liang, K.T.; Choo, K.K.; Zhou, J.Y. Extended proxy-assisted approach: Achieving revocable fine-grained encryption of cloud data. In Proceedings of the Computer Security-ESORICS 2015, LNCS 9327, Vienna, Austria, 21–25 September 2015; Springer: Heidelberg, Germany, 2015.
6. Yi, S.H.; Qin, Z.R.; Li, Q. Security and privacy issues of fog computing: A survey. In Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications, Qufu, China, 10–12 August 2015.
7. Ren, K.; Wang, C.; Wang, Q. Security challenges for the public cloud. *IEEE Internet Comput.* **2012**, *16*, 69–73. [CrossRef]
8. Gia, T.N.; Jiang, M.Z.; Rahmani, A.M.; Westerlund, T.; Liljeberg, P.; Tenhunen, H. Fog computing in healthcare Internet of things: A case study on ECG feature extraction. In Proceedings of the IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), Liverpool, UK, 26–28 October 2015.
9. Sahai, A.; Waters, B. Fuzzy identity based encryption. *Lect. Notes Comput. Sci.* **2004**, *3494*, 457–473.
10. Gagné, M.; Narayan, S.; Naini, R.S. Threshold attribute based signcryption. In Proceedings of the Security and Cryptography for Networks, LNCS 6280, Amalfi, Italy, 13–15 September 2010; Springer: Berlin/Heidelberg, Germany, 2010.
11. Rao, Y.S.; Dutta, R. Expressive attribute-based signcryption with constant-size ciphertext. In Proceedings of the Progress in Cryptology-AFRICACYPT 2014, LNCS 8469, Marrakesh, Morocco, 28–30 May 2014; Springer: Cham, Switzerland, 2014.
12. Chen, C.; Chen, J.; Lim, H.W.; Zhang, Z.F.; Feng, D.G. Combined public-key schemes: The case of ABE and ABS. In Proceedings of the Provable Secure, LNCS 7496, Chengdu, China, 26–28 September 2012; Springer: Berlin/Heidelberg, Germany, 2012.
13. Liu, J.H.; Huang, X.Y.; Liu, J.K. Secure sharing of personal health records in cloud computing: Ciphertext policy attribute based signcryption. *Future Gener. Comput. Syst.* **2015**, *52*, 67–76. [CrossRef]
14. Rao, Y.S. A secure and efficient ciphertext policy attribute-based signcryption for personal health records sharing in cloud computing. *Future Gener. Comput. Syst.* **2017**, *67*, 133–151. [CrossRef]

15. Yu, G.; Cao, Z.F. Attribute-based signcryption with hybrid access policy. *Peer Peer Netw. Appl.* **2015**, *20*, 1–9. [CrossRef]
16. Huang, Q.L.; Yang, Y.X.; Wang, L.C. Secure data access control with ciphertext update and computation outsourcing in fog computing for Internet of Things. *IEEE Access* **2017**, *5*, 12941–12950. [CrossRef]
17. Fan, K.; Wang, J.X.; Wang, X.; Li, H.; Yang, Y.T. A secure and verifiable outsourced access control scheme in fog-cloud computing. *Sensors* **2017**, *17*, 1695. [CrossRef] [PubMed]
18. Zuo, C.; Shao, J.; Wei, G.Y.; Xie, M.D.; Ji, M. CCA-secure ABE with outsourced decryption for fog computing. *Future Gener. Comput. Syst.* **2018**, *78*, 730–738. [CrossRef]
19. Zhang, P.; Chen, Z.H.; Liu, J.K.; Liang, K.T.; Liu, H.W. An efficient access control scheme with outsourcing capability and attribute update for fog computing. *Future Gener. Comput. Syst.* **2018**, *78*, 753–762. [CrossRef]
20. Mao, X.P.; Lai, J.Z.; Mei, Q.X.; Chen, K.F.; Weng, J. Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption. *IEEE Trans. Dependable Secur.* **2016**, *13*, 533–546. [CrossRef]
21. Han, J.G.; Susilo, W.; Mu, Y.; Zhou, J.Y.; Au, M.H.A. Improving privacy and security in decentralized CP-ABE. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 665–678.
22. Jiang, R.; Wu, X.; Bhargava, B. SDSS-MAC: Secure data sharing scheme in multi-authority cloud storage systems. *Comput. Secur.* **2016**, *62*, 193–212. [CrossRef]
23. Lewko, A.; Waters, B. Decentralizing attribute-based encryption. In Proceedings of the Advances in Cryptology-EUROCRYPT 2011, LNCS 6632, Tallinn, Estonia, 15–19 May 2011; Springer: Berlin/Heidelberg, Germany, 2011.
24. Ruj, S.; Stojmenovic, M.; Nayak, A. Decentralized access control with anonymous authentication of data stored in clouds. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *20*, 384–394. [CrossRef]
25. Sourya, J.D.; Ruj, S. Efficient decentralized attribute-based access control for mobile clouds. *IEEE Trans. Cloud Comput.* **2017**, *99*, 1–14.
26. Yang, K.; Jia, X.H.; Ren, K. DAC-MACS: Effective data access control for multi-authority cloud storage systems. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 1790–1801. [CrossRef]
27. Meng, X.Y.; Meng, X.Y. A novel attribute-based signcryption scheme in cloud computing enviroments. In Proceedings of the IEEE International Conference on Information and Automation, Ningbo, China, 1–3 August 2016.
28. Hong, H.S.; Xia, Y.H.; Sun, Z.X.; Liu, X.M. Provably secure attribute based signcryption with delegated computation and efficient key updating. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 2646–2659.
29. Lounis, A.; Hadjidj, A.; Bouabdallah, A.; Challal, Y. Healing on the cloud: Secure cloud architecture for medical wireless sensor networks. *Future Gener. Comput. Syst.* **2016**, *55*, 266–277. [CrossRef]
30. Xiao, M.; Zhou, J.; Liu, X.J.; Jiang, M.D. A hybrid scheme for fine-grained search and access authorization in fog computing environment. *Sensors* **2017**, *17*, 1423. [CrossRef] [PubMed]
31. Li, J.G.; Wang, Y.; Zhang, Y.C.; Han, J.G. Full verifiability for outsourced decryption in attribute based encryption. *IEEE Trans. Serv. Comput.* **2017**. [CrossRef]
32. Liao, Y.J.; He, Y.C.; Li, F.G.; Jiang, S.Q.; Zhou, S.J. Analysis of an ABE scheme with verifiable outsourced decryption. *Sensors* **2018**, *18*, 176. [CrossRef] [PubMed]
33. Chase, M. Multi-authority attribute-based encryption. In Proceedings of the 4th Theory of Cryptography Conference on Theory of Cryptography, TCC 2007, Amsterdam, The Netherlands, 21–24 February 2007.
34. Chase, M.; Chow, S. Improving privacy and security in multi-authority attribute-based encryption. In Proceedings of the ACM Conference on Computer and Communications Security, Chicago, IL, USA, 9–13 November 2009.
35. Jung, T.; Li, X.Y.; Wan, Z.G.; Wan, M. Privacy preserving cloud data access with multi-authorities. In Proceedings of the IEEE INFOCOM, Turin, Italy, 14–19 April 2013.
36. Li, Q.; Ma, J.F.; Rui, L.; Liu, X.M.; Xiong, J.B.; Chen, D.W. Secure, efficient and revocable multi-authority access control system in cloud storage. *Comput. Secur.* **2016**, *59*, 45–59. [CrossRef]
37. Maji, H.K.; Prabhakaran, M.; Rosulek, M. Attribute-Based Signature: Achieving Attribute Privacy and Collusion Resistance; IACR Cryptology Eprint Archive. 2008. Available online: http://eprint.iacr.org/2008/328 (accessed on 23 April 2018).

38. Okamoto, T.; Takashima, K. Decentralized attribute-based signature. In Proceedings of the Public Key Cryptography, Nara, Japan, 26 Febrary–1 March 2013; Springer: Berlin/Heidelberg, Germany, 2013.
39. Waters, B. Ciphertext-policy attribute-based encryption: An expressive, efficient and provable secure realization. In Proceedings of the Public Key Cryptography, Taormina, Italy, 6–9 March 2011; Springer: Berlin/Heidelberg, Germany, 2011.