sensors

MDPI

Article

A Secure and Verifiable Outsourced Access Control Scheme in Fog-Cloud Computing

Kai Fan [1,*], Junxiong Wang [1], Xin Wang [1], Hui Li [1] and Yintang Yang [2]

1. State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China; wangjx921210@163.com (J.W.); xwang2011@stu.xidian.edu.cn (X.W.); lihui@mail.xidian.edu.cn (H.L.)
2. Key Laboratory of the Ministry of Education for Wide Band-Gap Semiconductor Materials and Devices, Xidian University, Xi'an 710071, China; ytyang@xidian.edu.cn

* Correspondence: kfan@mail.xidian.edu.cn; Tel.: +86-139-9193-6634

Received: 31 May 2017; Accepted: 21 July 2017; Published: 24 July 2017

Abstract: With the rapid development of big data and Internet of things (IOT), the number of networking devices and data volume are increasing dramatically. Fog computing, which extends cloud computing to the edge of the network can effectively solve the bottleneck problems of data transmission and data storage. However, security and privacy challenges are also arising in the fog-cloud computing environment. Ciphertext-policy attribute-based encryption (CP-ABE) can be adopted to realize data access control in fog-cloud computing systems. In this paper, we propose a verifiable outsourced multi-authority access control scheme, named VO-MAACS. In our construction, most encryption and decryption computations are outsourced to fog devices and the computation results can be verified by using our verification method. Meanwhile, to address the revocation issue, we design an efficient user and attribute revocation method for it. Finally, analysis and simulation results show that our scheme is both secure and highly efficient.

Keywords: fog computing; cloud computing; access control; attribute-based encryption; verifiable outsource; revocation

1. Introduction

Recently, fog computing has drawn a great deal of attention. It is a quite novel computing paradigm that extends cloud computing facilities and services to the edge of the network to provide computing, networking, and storage services between end devices and data centers [1,2]. Fog computing devices are located between endpoints and the traditional cloud, thus resources and services are available and are closer to the end-users, and the delays induced by service deployments can be reduced [3,4]. Compared with the cloud computing concept, which is more centralized, fog computing provides resources and services in a distributed way. Combined with the traditional cloud, faster and more convenient computing services are provided to nearby devices based on their own computing, storage and network capacity [5]. Since fog devices are localized, it provides low-latency communication and more context awareness [6]. With all these advantages, the fog computing paradigm is well positioned for big data and real time analytics.

Fog computing is a quite novel computing paradigm that aims at moving the cloud computing (CC) facilities and services to the access network, in order to reduce the delays induced by service deployments. Although big data and the Internet of things (IOT) still rely on cloud computing, as the number of networking devices and data volume are increasing dramatically, fog-cloud computing can effectively solve the bottleneck problem of data transmission and data storage. However, since fog devices are located at the edge of the network and are of much lower cost than cloud servers, they are more easily compromised and have lower trustworthiness [7,8], especially in the process of data sharing. Therefore, secure and efficient access control schemes in fog-cloud computing environment need to be implemented [9,10]. Compared with traditional data access control schemes in cloud computing, the network structures and system models in the fog-cloud computing environment are different. Fog devices can provide computing, networking, and storage services for users, such that less communication and computational cost is left for users to do, therefore, cloud, fog and end-users should be considered in the new access control scheme.

Ciphertext-policy attribute-based encryption (CP-ABE) [11] is regarded as one of the most suitable technologies to realize fine-grained access control. This technique allows data owners to implement access control by setting up access structures. Compared with single-authority CP-ABE schemes, in multi-authority CP-ABE schemes, attributes are from different domains and managed by different authorities. Moreover, it does not have the single point of failure and system bottleneck problem, which makes multi-authority CP-ABE schemes more practical for data access control in the fog-cloud computing environment.

However, the processes of encryption and decryption in CP-ABE systems are time-consuming. The computation for data owners and users is a great overhead. To outsource part of the encryption and decryption computation to a cloud server is a solution. However, the server may be "lazy". It may not follow the algorithm, and only execute part of the computations or deliberately return incorrect results. Therefore, a verification method of the outsourced encryption and decryption needs to be proposed. Besides, user and attribute revocation is another issue in CP-ABE systems. On the one hand, the users in the system may change frequently, and on the other hand, the attributes of users may also change, and revocation of any attribute may affect other users who share the same attribute. However, most existing schemes cannot support flexible and efficient user and attribute revocation in multi-authority cloud storage systems. The key update and ciphertext re-encryption operations are time-consuming. Therefore, verifiable outsourced multi-authority CP-ABE schemes with efficient and flexible user and attribute revocation need to be proposed.

1.1. Related Work

In 2007, Bethencourt et al. [11] put forward the first CP-ABE scheme. Over the last decade, many CP-ABE schemes [12–17] were proposed. However, most of them are time-consuming and lack efficiency. To improve the efficiency and reduce the overhead of users, several schemes which support outsourced computation and revocation are proposed:

1.1.1. Outsourced Computation

Green et al. [18] proposed an outsourcing decryption ABE scheme. In their scheme, the traditional private keys are divided into user keys and transformation keys. Thus, complex decryption computations are outsourced to the cloud server, and users only need one exponentiation operation to recover the plaintext. However, their scheme cannot be applied to multi-authority systems. Based on this method, Yang et al. [19,20] put forward two multi-authority CP-ABE schemes which support outsourced decryption. Li et al. [21] also proposed an outsourced ABE scheme which supports both outsourced key-issuing and decryption. However, they did not consider the correctness of results from the cloud server.

To solve this problem, Lai et al. [22] introduced the verifiability of ABE and proposed a verifiable outsourced decryption ABE scheme. But in their scheme, both the length of the ciphertext and the computational of encryption are doubled. Later, Li et al. [23] presented an outsourcing ABE scheme with checkability which supports both outsourced key-issuing and decryption. However, the length of ciphertext and the amount of expensive pairing computations grow with the number of attributes.

To address this problem, two ABE schemes [24,25] in which the length of ciphertext is constant are put forward. However their constructions cannot be applied to ABE schemes with Linear Secret Sharing Schemes (LSSS). Mao et al. [26] proposed a generic construct of attribute-based encryption with verifiable outsourced decryption. Their CPA-secure construct has more compact ciphertext and less computational costs. Users only need a constant number of simple computations to decrypt the ciphertext.

1.1.2. Revocation

Ostrovsky et al. [27] first proposed a fine-grained user revocation scheme based on CP-ABE that supports negative clauses. With the help of a semi-trusted service provider, Ibraimi et al. [28] put forward a CP-ABE scheme which achieved immediate attribute revocation for the first time, but their construct cannot be applied to an outsourcing environment. Yu et al. [29] presented a CP-ABE scheme where proxy encryption technology was introduced. The scheme achieves immediate attribute revocation, at the same time, the proxy server also share the authority job, hoowever, the proxy server needs to be online all the time. Another CP-ABE scheme with fine-grained attribute revocation was put forward by Hur et al. [30]. They use attribute group keys to re-encrypt the ciphertext, but their scheme cannot prevent collusion attacks. Another revocable CP-ABE scheme was proposed by Xie et al. [31]. In their scheme, the key update computations are greatly reduced. Later, Yang et al. [32] put forward a proxy-assisted CP-ABE scheme which provides efficient cloud data sharing and user revocation.

1.2. Our Contribution

In this paper, we propose a verifiable outsourced multi-authority access control scheme, named VO-MAACS. In our construct, most of the encryption and decryption computation is outsourced to fog devices and the computation results can be verified by using our verification method. Meanwhile, to address the revocation issue, we design an efficient user and attribute revocation method for it. Our contributions can be summarized as follows:

(1) We propose the verifiable outsourced multi-authority access control scheme (VO-MAACS), which is secure against collusion attacks. Most of the encryption and decryption computation is outsourced to fog devices, which greatly reduces the computation on the user side.
(2) We provide a verification method for the outsourced encryption and decryption. If a fog device returns incorrect results, users can notice it immediately by running the corresponding verification algorithm.
(3) We design an efficient user and attribute revocation method for our scheme. During the process of attribute revocation, most of the update and re-encryption operations are outsourced to the cloud server, and only a few components which are associated with the revoked attribute need to be updated, while the other components are not changed.
(4) We provide a security and performance analysis of our scheme, which shows that our scheme is both secure and highly efficient.

1.3. Organization

The remainder of this paper is organized as follows: we first give some preliminaries in Section 2. Then, we give the definition of the system model and framework in Section 3. In Section 4, we propose our VO-MAACS construct. Section 5 describes the security and performance analysis of our scheme. Finally, the conclusions are given in Section 6.

2. Preliminaries

In this section, some fundamental background used in this paper is provided, including bilinear maps, access structure and linear secret sharing scheme (LSSS).

2.1. Bilinear Maps

Definition 1. (Bilinear Maps). *Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:*

(1) *Bilinearity: for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.*

(2) *Non-degeneracy: there exists $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2) = 1$.*

(3) *Computability: there is an efficient algorithm to compute $e(g_1, g_2)$ for any $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.*

2.2. Access Structure

Definition 2. (Access Structure [33]). *Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\cdots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,P_2,\cdots,P_n\}} \backslash \{\phi\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

2.3. Linear Secret Sharing Schemes

Definition 3. (Linear Secret-Sharing Schemes (LSSS) [33]). *We recall the description of LSSS as follows [33]. Let Π be a secret sharing scheme over a set of parties $\mathcal{P}$ with realizing an access structure $\mathbb{A}$. We say that Π is a linear secret sharing scheme over $\mathbb{Z}_p$ if:*

(1) *The piece for each party forms a vector over $\mathbb{Z}_p$.*

(2) *During the generation of the pieces, the dealer chooses independent random variables, denoted $r_2, \cdots, r_n$,* each one distributed uniformly over $\mathbb{Z}_p$. *Each coordinate of the piece of every party is a linear combination of $r_2, \cdots, r_n$ and the secret s. That is, let M denotes a matrix with l rows and n columns. For the vector $\vec{v}^T = (s, r_2, \cdots, r_n)$ and any authorized set, there exist constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} w_i \lambda_i = s$, where $\lambda_i = (M\vec{v})_i$ and $I \subset \{1, 2, \cdots, l\}$.*

3. System Model and Framework

In this section, the system model and framework of our scheme are described.

3.1. System Model

A simple three level hierarchy is adopted in our fog-cloud system as illustrated in Figure 1. In this framework, each terminal device is connected to a nearby fog device. Fog devices are interconnected and each of them is linked to the cloud.

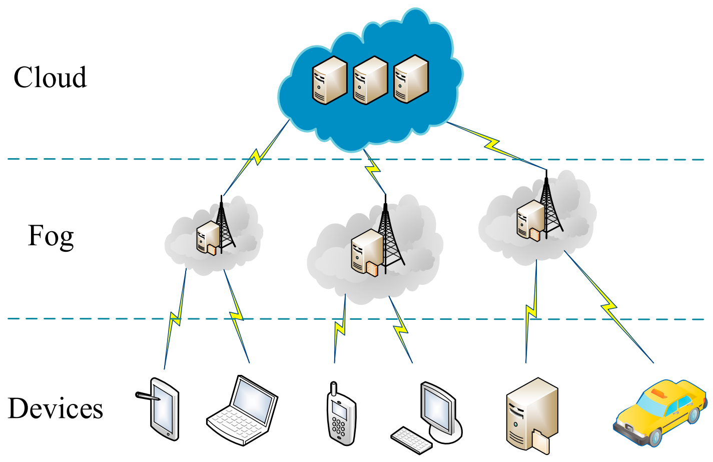

Figure 1. System model in fog-cloud computing environment.

In general, a layer of fog is added between the cloud server and terminal devices so that some computations on the cloud server can be delegated to the fog devices which are closer to the terminal devices. Thus, different tasks from different regions can be executed by the corresponding fog devices simultaneously, which greatly improves the efficiency. Fog devices are responsible for data transmission and data storage. Moreover, they are also in charge of part of the encryption and decryption computations. The cloud server is responsible for storing the ciphertext and the user proxy keys, as well as the ciphertext re-encryption operations and user proxy keys update operations when revocation occurs.

Our multi-authority fog-cloud system consists of six entities: a cloud service provider (CSP), fog devices (FDs), a global certificate authority (CA), attribute authorities (AAs), data owners (DOs) and data users (DUs), as shown in Figure 2.

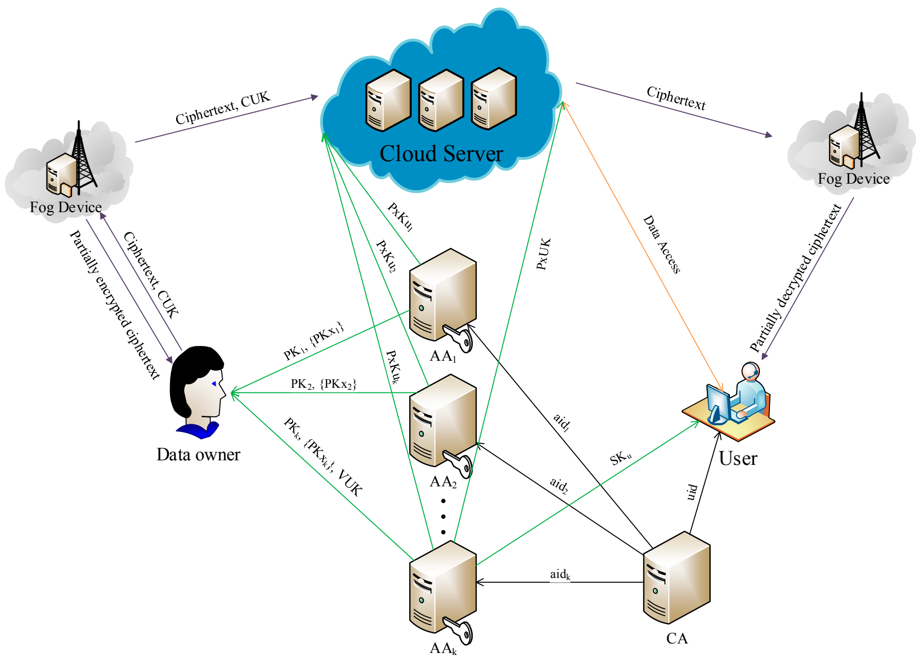

Figure 2. System model of multi-authority access control in fog-cloud system.

CA is a fully trusted global certificate authority in the system. It accepts the registration of all AAs and DUs in the system, and it is responsible for issuing a global unique identity *uid* for each DU and a unique identity *aid* for each AA. However, it does not participate in any attribute management and any generation of secret keys associated with attributes.

Each AA is an independent attribute authority that is responsible for issuing, revoking and updating users' attributes within its administration domain. In our scheme, each AA is responsible for generating a public attribute key PK_x for each attribute it manages and a user private key which consists of user proxy key PxK and user secret key SK for each DU. Especially, PxK is stored at CSP and SK is kept by DU.

DOs define access control policies over attributes from multiple attribute authorities and then encrypts the data following those policies. After that, they upload the encrypted data to the CSP.

The CSP is responsible for storing the ciphertext and the user proxy keys, and provides data access service to DUs. It is also in charge of the ciphertext re-encryption operations and user proxy key update operations when revocation occurs.

FDs are responsible for data transmission and data storage. Moreover, they are also in charge of part of the encryption and decryption computations. They can help generate part of the ciphertext for DOs, as well as decrypt part of the ciphertext for DUs. Only for those DUs whose attributes satisfy the access policy will FDs decrypt the ciphertext with their proxy keys. After that, they send the partially decrypted data to the corresponding DUs.

DUs can request their secret keys from the relevant authorities. After downloading any encrypted data from the CSP, a DU first asks a FD to decrypt it with his proxy key. If the attribute set of the DU meets the access policy, then the FD decrypts the ciphertext and sends the partially decrypted data to the DU. Upon receiving the partially decrypted data from the FD, the DU can recover the data with his secret key.

In our multi-authority fog-cloud system, we assume that the CA is fully trusted in the system. Each AA is also trusted, but it can be corrupted by an adversary. The CSP and FDs are semi-trusted. They may leak the encrypted data to some malicious users, but will execute the tasks assigned by each authority. DUs are assumed dishonest and may collude to obtain unauthorized access to data.

3.2. Framework

Definition 4. **(VO-MAACS: Verifiable Outsourced Multi-Authority Access Control Scheme).** ***Global Setup*** $(\lambda, U) \rightarrow \{GP, uid, aid\}$. *The global setup algorithm is run by CA. On input the security parameter* λ *and attribute universe description* U*, it outputs the global parameter* GP*, the user identity* uid *and the authority identity* aid*.*

Authority Setup $(aid) \rightarrow \left\{PK_{aid}, SK_{aid}, \{PK_{x_k}\}_{aid \in I_A}\right\}$. *The authority setup algorithm is run by each authority. On input of the authority identity* aid*, it outputs public attribute keys* $\{PK_{x_k}\}_{aid \in I_A}$ *for all attributes issued by each authority and a pair of authority public key* PK_{aid} *and authority secret key* SK_{aid}*. Here* I_A *denotes the involved authority set.*

Encrypt_out $(GP, \{PK_{x_k}\}_{aid \in I_A}) \rightarrow CT_{out}$. *The outsourced encryption algorithm is run by the FD. On input of the global parameter* GP *and a set of public attribute keys* $\{PK_{x_k}\}_{aid \in I_A}$*, it outputs the partially encrypted ciphertext* CT_{out}*.*

Verify_enc $(GP, CT_{out}) \rightarrow b$. *The outsourced encryption verification algorithm is run by a DO. On input of the global parameter* GP *and a partially encrypted ciphertext* CT_{out}*, it outputs a bit* $b \in \{0, 1\}$*,* $b = 1$ *indicates the FD outputs the correct result,* $b = 0$ *indicates the FD outputs the incorrect result.*

Encrypt_user $(GP, PK_{aid}, \{PK_{x_k}\}_{aid \in I_A}, CT_{out}, M, (\mathbb{A}, \rho)) \rightarrow CT$. *The user encryption algorithm is run by a DO. On input of the global parameter* GP*, a set of authority public keys* PK_{aid}*, a set of public attribute keys* $\{PK_{x_k}\}_{aid \in I_A}$*, a partially encrypted ciphertext* CT_{out}*, a message* M *and an access structure* $\mathbb{A}$*, it outputs the ciphertext* CT*.*

KeyGen $(GP, uid, S_{uid,aid}, SK_{aid}, \{PK_{x_k}\}_{aid \in I_A}) \rightarrow \{PxK_{uid,aid}, SK_{uid}\}$. *The key generation algorithm is run by each authority. On input of the global parameter* GP*, the user identity* uid*, a set of user attributes* $S_{uid,aid}$*, the authority secret key* SK_{aid}*, and a set of attribute public keys* $\{PK_{x_k}\}_{aid \in I_A}$*, it outputs user proxy key* $PxK_{uid,aid}$ *and user secret key* SK_{uid}*.*

Decrypt_out $(GP, CT, PxK_{uid,aid}, \{PK_{x_k}\}_{aid \in I_A}) \rightarrow CT'$. *The outsourced decryption algorithm is run by a FD. On input of the global parameter* GP*, the proxy keys* $PxK_{uid,aid}$ *and the ciphertext* CT*, it outputs the partially decrypted ciphertext* CT'*.*

Verify_dec $(GP, CT, CT') \rightarrow b$. *The outsourced decryption verification algorithm is run by a DU. On input of the global parameter* GP*, the ciphertext* CT *and a partially decrypted ciphertext* CT'*, it outputs a bit* $b \in \{0, 1\}$*,* $b = 1$ *indicates the FD has output the correct result,* $b = 0$ *indicates the FD has output the incorrect result.*

Decrypt_user $(CT, CT', SK_{uid}) \rightarrow M$. *The user decryption algorithm is run by a DU. On input of the ciphertext* CT*, the partially decrypted ciphertext* CT' *and the user secret key* SK_{uid}*, it outputs the message* M*.*

URev $(uid, L_{PxK}) \rightarrow L'_{PxK}$. *The user revocation algorithm is run by the CSP. On input of the revoked user identity* uid *and the proxy key list* L_{PxK}*, it outputs the updated proxy key list* L'_{PxK}*.*

ReKeyUpdate $(uid, PxK_{uid,aid}, v_{\tilde{x}_k}) \rightarrow \{VUK_{\tilde{x}_k}, PxUK_{\tilde{x}_k}\}$. *The key update algorithm is run by the involved authorities. On input of the uid of each non-revoked user, the proxy key* $PxK_{uid,aid}$ *and the current attribute version key* $v_{\tilde{x}_k}$*, it outputs the version update key* $VUK_{\tilde{x}_k}$ *and the proxy update key* $PxUK_{\tilde{x}_k}$*.*

CTUpdate $(VUK_{\tilde{x}_k}, CT_{out}) \rightarrow \{CUK_{\tilde{x}_k}\}$. *The ciphertext update algorithm is run by a DO. On input of the version update key* $VUK_{\tilde{x}_k}$ *and the partially encrypted ciphertext* CT_{out}*, it outputs the ciphertext update key* $CUK_{\tilde{x}_k}$*.*

PxKUpdate $(uid, PxK_{uid,aid}, PxUK_{\widetilde{x}_k}) \rightarrow PxK^*_{uid,aid}$. *The proxy key update algorithm is run by the CSP. On input of the uid of each non-revoked user, the current proxy key* $PxK_{uid,aid}$ *and the proxy update key* $PxUK_{\widetilde{x}_k}$, *it outputs a new proxy key* $PxK^*_{uid,aid}$ *for each non-revoked user who has the attribute* $\widetilde{x}_k$.

ReEnc $(CT, CUK_{\widetilde{x}_k}) \rightarrow CT^*$. *The re-encryption algorithm is run by the CSP. On input of the current ciphertext CT and the ciphertext update key* $CUK_{\widetilde{x}_k}$, *it outputs a new ciphertext* CT^*.

4. VO-MAACS: Verifiable Outsourced Multi-Authority Access Control Scheme

In this section, we give the concrete construction of VO-MAACS which is based on [14], together with the verification method and revocation scheme.

4.1. Construction of VO-MAACS

Global Setup $(\lambda, U) \rightarrow \{GP, uid, aid\}$. The global setup algorithm takes a security parameter λ and a small attribute universe description U as input. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be the multiplicative groups with the same prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be the bilinear map. Let g_1 be the generator of $\mathbb{G}_1$ and g_2 be the generator of $\mathbb{G}_2$. Let $G : \mathbb{G}_T \rightarrow \mathbb{Z}_p$ be a hash function and $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ be a hash function which maps attributes to an element in $\mathbb{G}_2$, such that the security will be modeled in the random oracle. CA then chooses a random number $a \in \mathbb{Z}_p$ and sets the global parameter as $GP = \{p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_2^a, H, G\}$. Each authority, fog device and user should register itself with the global authority during the global setup process. CA then assigns a unique global authority identity *aid* to each legitimate authority and a unique global user identity *uid* to each legitimate user.

Authority Setup $(aid) \rightarrow \left\{PK_{aid}, SK_{aid}, \{PK_{x_k}\}_{aid \in I_A}\right\}$. Let $S_{A_{aid}}$ denote the set of all attributes managed by AA_{aid} and I_A denote the involved authority set. AA_{aid} first chooses two random exponents $\alpha_{aid}, \beta_{aid} \in \mathbb{Z}_p$. For each attribute $x_k \in S_{A_{aid}}$, AA_{aid} chooses an attribute version key as $VK_{x_k} = v_{x_k}$ and generates the public attribute keys as $\{PK_{x_k}\}_{aid \in I_A} = g_2^{v_{x_k}} \cdot g_2^{H(x_k)}$. Then it publishes $PK_{aid} = e(g_1, g_2)^{\alpha_{aid}}$ as its public key and keeps $SK_{aid} = \{\alpha_{aid}, \beta_{aid}\}$ as its secret key.

Encrypt_out $(GP, \{PK_{x_k}\}_{aid \in I_A}) \rightarrow CT_{out}$. FD first chooses a random number $s' \in \mathbb{Z}_p$ and computes $C_0 = g_1^{s'}$. For $i \in \{1, \cdots, l\}$, it randomly picks $\lambda'_i, \gamma' \in \mathbb{Z}_p$ and computes:

$$C_{i,1} = g_2{}^{a\lambda'_i} \cdot \left(g_2^{v_{x_i}} g_2^{H(x_i)}\right)^{-\gamma'_i}, C_{i,2} = g_1^{\gamma'_i}, s', \lambda'_i, \gamma'_i \tag{1}$$

Then, it outputs the partially encrypted ciphertext $CT_{out} = \left\{s', C_0, (C_{i,1}, C_{i,2}, \lambda'_i, \gamma'_i)_{i \in \{1, \cdots, l\}}\right\}$.

Encrypt_user $(GP, PK_{aid}, \{PK_{x_k}\}_{aid \in I_A}, CT_{out}, M, (\mathbb{A}, \rho)) \rightarrow CT$. Let $\mathbb{A}$ be a $l \times n$ matrix, where l denotes the total number of all the attributes. The function ρ maps rows of the matrix $\mathbb{A}$ to attributes. DO first chooses a random secret exponent $s \in \mathbb{Z}_p$ and a random vector $\vec{v} = (s, y_2, \cdots, y_n) \in \mathbb{Z}_p^n$ with s as its first entry, where $y_2, \cdots, y_n$ are used to share the secret exponent s. For $i = 1, \cdots, l$, it computes $\lambda_i = A_i \cdot \vec{v}$, where A_i is the vector corresponding to the i-th row of $\mathbb{A}$. After that, it randomly chooses $\gamma_1, \gamma_2, \cdots, \gamma_l \in \mathbb{Z}_p$ and computes:

$$CT_{user} = \left\{ \begin{array}{l} C = M \cdot e(g_1, g_2)^{\sum\limits_{aid \in I_A} \alpha_{aid} s}, \ C' = s - s', \ (C_{i,3} = \lambda_i - \lambda'_i, C_{i,4} = \gamma_i - \gamma'_i)_{i \in \{1, \cdots, l\}}, \\ C_v = G(e(g_1, g_2)^{\sum\limits_{aid \in I_A} \alpha_{aid} s}), \ (\mathbb{A}, \rho) \end{array} \right\} \tag{2}$$

$C', C_{i,3}, C_{i,4}$ are used to correct the shares of s and randomize γ_i. C_v is used to verify the result of outsourced decryption. Then, it outputs the intact ciphertext $CT = \{C, C', C_0, (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})_{i \in \{1, \cdots, l\}}, C_v, (\mathbb{A}, \rho)\}$.

KeyGen $(GP, uid, S_{uid,aid}, SK_{aid}, \{PK_{x_k}\}_{aid \in I_A}) \rightarrow \{PxK_{uid,aid}, SK_{uid}\}$. AA_{aid} first assigns a set of attributes $S_{uid,aid}$ to each legal user, then chooses a random number $z_{uid} \in \mathbb{Z}_p$ for each user and let

$SK_{uid} = \{z_{uid}\}$ as the user secret key. Then, AA_{aid} runs the key generation algorithm to generate the user proxy key as:

$$PxK_{uid,aid} = \left\{ K_{uid,aid} = g_2^{\frac{\alpha_{aid}}{z_{uid}}} g_2^{\frac{a\beta_{aid}}{z_{uid}}}, L_{uid,aid} = g_1^{\frac{\beta_{aid}}{z_{uid}}}, \{K_{uid,x_i}\}_{x_i \in S_{uid,aid}} = (g_2^{v_{x_i}} g_2^{H(x_i)})^{\frac{\beta_{aid}}{z_{uid}}}, S_{uid,aid} \right\} \quad (3)$$

The proxy keys $\{PxK_{uid,aid}\}$ are sent to CSP who will add them in its proxy key list L_{PxK} as $L_{PxK} = L_{PxK} \cup \{uid, PxK_{uid,aid}\}$, and the user secret keys are sent to the corresponding DUs.

Decrypt_out $(GP, CT, PxK_{uid,aid}, \{PK_{x_k}\}_{aid \in I_A}) \rightarrow CT'$. When a user queries the encrypted data in the system, CSP will first check his attribute set. If his attributes does not satisfy the access policy, CSP outputs $\perp$. Otherwise, it sends the ciphertext and the corresponding proxy keys to FD. FD first chooses a set of constants $w_i \in \mathbb{Z}_p$ such that, if λ_i are valid shares of the secret s according to $\mathbb{A}$, then $\sum_{i \in I} w_i \lambda_i = s$, where $I = \{1, \cdots, l\}$. Then it computes:

$$CT' = \prod_{aid \in I_A} \frac{e(C_0 \cdot g_1^{C'}, K_{uid,aid})}{\prod_{i \in I} \left(e((C_{i,1} \cdot g_2^{aC_{i,3}} \cdot (g_2^{v_{x_i}} g_2^{H(x_i)})^{-C_{i,4}})^{\omega_i}, L_{uid,aid}) \cdot e((C_{i,2} \cdot g_1^{C_{i,4}})^{\omega_i}, K_{uid,x_i}) \right)} = e(g_1, g_2)^{\sum_{aid \in I_A} \frac{\alpha_{aid} s}{z_{uid}}} \quad (4)$$

After that, FD sends the partially decrypted ciphertext $CT' = e(g_1, g_2)^{\sum_{k \in I_A} \frac{\alpha_k s}{z_j}}$ to the user.

Decrypt_user $(CT, CT', SK_{uid}) \rightarrow M$. Upon receiving the partially decrypted ciphertext from FD, the user runs the user decryption algorithm to decrypt the ciphertext by using its secret key SK_{uid}. It computes M as:

$$M = \frac{C}{CT'^{z_{uid}}} \quad (5)$$

4.2. Verification Method

There exists such a situation that the FD may be "lazy". It may not follow the algorithm, only execute part of the computations or deliberately returns incorrect results. If this happens, a DO cannot notice the error, and large part of the computations will be affected. Therefore, we propose a verification method which can verify the result of outsourced encryption and outsourced decryption.

Our verification method includes two algorithms:

Verify_enc $(GP, CT_{out}) \rightarrow b$. Upon receiving the partially encrypted ciphertext CT_{out} from FD, DO first verify whether $C_0 = g^{s'}$ holds. If it does not holds, DO outputs $b = 0$, which indicates FD returns incorrect result. Otherwise, DO computes $t_i = (a\lambda'_i - v_{\rho(i)} \cdot \gamma_i - H(\rho(i)) \cdot \gamma_i) \bmod p$ and $C_i = C_{i,1} \cdot C_{i,2} = g_1^{\gamma'_i} \cdot g_2^{t_i}$, where $i \in \{1, \cdots, l\}$. Then, it picks a security parameter r, and randomly chooses $s_1, \cdots, s_l \in \{0,1\}^r$. After that, it computes $x = \sum_{i=1}^{n} \gamma'_i s_i \bmod p, y = \sum_{i=1}^{n} t_i s_i \bmod p$ and $\hat{C} = \prod_{i=1}^{n} C_i^{s_i} \bmod p$. If $\hat{C} = g_1^x \cdot g_2^y$, DO outputs $b = 1$, which indicates the FD returned the correct result. Otherwise, it outputs $b = 0$.

Here, we adopt the idea of [34]. We do not check the results in the ciphertext one by one. Instead, we use the batch verification algorithm to check $C_{i,1}, C_{i,2}$ together. Obviously, our solution is much more efficient than the normal verification method.

Verify_dec $(GP, CT, CT') \rightarrow b$. Upon receiving the partially decrypted ciphertext CT' from FD, the user computes $G(CT'^{z_{uid}})$, if $G(CT'^{z_{uid}}) = C_v$ holds, it outputs $b = 1$, which indicates FD returns the correct result. Otherwise, it outputs $b = 0$, which indicates that the FD has returned an incorrect result.

4.3. Revocation Scheme

4.3.1. User Revocation

In our scheme, when user revocation happens, we do not need to re-encrypt the ciphertext and update other non-revoked users' secret keys. The only operation we need is to send a user revocation message which contains the *uid* of the revoked user to the CSP, and then let the CSP delete the revoked user's proxy key $PxK_{uid,aid}$. Without the correct $PxK_{uid,aid}$, the FD cannot perform the outsourced decryption algorithm for the revoked user. Thus, the revoked user cannot recover the original data. The user revocation algorithm is described as follows:

$URev\ (uid, L_{PxK}) \rightarrow L'_{PxK}$. When the CSP receives the user revocation message from a DO, it then deletes the proxy key $PxK_{uid,aid}$ corresponding to the *uid* from the list and outputs the updated proxy key list L'_{PxK}.

4.3.2. Attribute Revocation

There are two phases in attribute revocation: Key update and Ciphertext re-encryption.

Phase 1: Key update

The key update in turn includes three steps: RekeyUpdate, CTUpdate and PxKUpdate.

(1) $ReKeyUpdate\ (uid, PxK_{uid,aid}, v_{\widetilde{x}_k}) \rightarrow \{VUK_{\widetilde{x}_k}, PxUK_{\widetilde{x}_k}\}$.

Let *uid* denotes all other non-revoked users except the revoked user with uid'. The involved authority AA_{aid} first generates a new attribute version key $v'_{\widetilde{x}_k}$. It then computes the version update key as $VUK_{\widetilde{x}_k} = v'_{\widetilde{x}_k} - v_{\widetilde{x}_k}$. After that, it applies $VUK_{\widetilde{x}_k}$ to compute the proxy update key as $PxUK_{\widetilde{x}_k} = g_2^{\frac{\beta_{aid}}{z_{uid}} \cdot VUK_{\widetilde{x}_k}}$ for each non-revoked user who has the attribute $\widetilde{x}_k$. Then AA_{aid} updates the public attribute key of the revoked attribute as $PK^*_{\widetilde{x}_k} = PK_{\widetilde{x}_k} \cdot g_2^{VUK_{\widetilde{x}_k}}$, and broadcast a message for each DO such that they can get the updated public attribute key of the revoked attribute. After that, $PxUK_{\widetilde{x}_k}$ is sent to the CSP to update $PxK_{uid,aid}$ and $VUK_{\widetilde{x}_k}$ is sent to the DO.

(2) $CTUpdate\ (VUK_{\widetilde{x}_k}, CT_{out}) \rightarrow \{CUK_{\widetilde{x}_k}\}$.

Upon receiving the version update key $VUK_{\widetilde{x}_k}$ and the partially encrypted ciphertext CT_{out}, DO computes the ciphertext update key as $CUK_{\widetilde{x}_k} = g_2^{-\gamma'_i \cdot AUK_{\widetilde{x}_k}}$. Then, $CUK_{\widetilde{x}_k}$ is sent to the CSP to update the ciphertext.

(3) $PxKUpdate\ (uid, PxK_{uid,aid}, PxUK_{\widetilde{x}_k}) \rightarrow PxK^*_{uid,aid}$.

Upon receiving the proxy update key $PxUK_{\widetilde{x}_k}$, CSP updates the corresponding proxy keys as $K^*_{x=\widetilde{x}_k,uid} = K_{x=\widetilde{x}_k,uid} \cdot PxUK_{\widetilde{x}_k}$ for each non-revoked user who has the attribute $\widetilde{x}_k$. Then the proxy keys $PxK_{uid,aid}$ are updated as:

$$PxK^*_{uid,aid} = \left\{ \begin{array}{l} K_{uid,aid} = g_2^{\frac{\alpha_{aid}}{z_{uid}}} g_2^{\frac{a\beta_{aid}}{z_{uid}}}, L_{uid,aid} = g_1^{\frac{\beta_{aid}}{z_{uid}}} \\ K_{uid,x \neq \widetilde{x}_k} = (g_2^{v_x} g_2^{H(x)})^{\frac{\beta_{aid}}{z_{uid}}} \\ K^*_{uid,x=\widetilde{x}_k} = (g_2^{v'_{\widetilde{x}_k}} g_2^{H(\widetilde{x}_k)})^{\frac{\beta_{aid}}{z_{uid}}} \end{array} \right\} \tag{6}$$

Phase 2: Ciphertext re-encryption

$ReEnc\ (CT, CUK_{\widetilde{x}_k}) \rightarrow CT^*$.

Upon receiving the ciphertext update key $CUK_{\widetilde{x}_k}$, CSP updates the corresponding ciphertext as $C^*_{i,1} = C_{i,1} \cdot CUK_{\widetilde{x}_k}$. Then the new ciphertext CT^* is published as:

$$CT^* = \left\{ \begin{array}{c} C, C', C_0, (C_{i,2}, C_{i,3}, C_{i,4})_{i \in \{1,\cdots,l\}}, C_v, (\mathbb{A}, \rho) \\ \left(C_{i,1} = g_2{}^{a\lambda'_i} \cdot (g_2^{v x_i} g_2^{H(x_i)})^{-\gamma'_i} \right)_{x_i \neq \widetilde{x}_k, i \in \{1,\cdots,l\}} \\ \left(C^*_{i,1} = g_2{}^{a\lambda'_i} \cdot (g_2^{v' x_k} g_2^{H(x_k)})^{-\gamma'_i} \right)_{x_i = \widetilde{x}_k, i \in \{1,\cdots,l\}} \end{array} \right\} \tag{7}$$

Apparently, we can conclude that most of the update and re-encryption work is outsourced to the CSP, which greatly reduces the overhead of DOs. Meanwhile, we do not need to update the entire ciphertext and user proxy keys. Only those components which are involved with the revoked attribute need to be updated. In this way, our scheme can greatly improve the efficiency of attribute revocation.

5. Analysis of Our Scheme

In this section, a comprehensive analysis of VO-MAACS is provided, including security analysis and performance analysis.

5.1. Security Analysis

5.1.1. Correctness

The correctness of our scheme can be easily proved by the following equations:

When there is no attribute revocation:

$$\begin{aligned} CT' &= \prod_{aid \in I_A} \frac{e(C_0 \cdot g_1^{C'}, K_{uid,aid})}{\prod_{i \in I} \left(e((C_{i,1} \cdot g_2{}^{aC_{i,3}} \cdot (g_2^{v x_i} g_2^{H(x_i)})^{-C_{i,4}})^{\omega_i}, L_{uid,aid}) \cdot e((C_{i,2} \cdot g_1{}^{C_{i,4}})^{\omega_i}, K_{uid,x_i}) \right)} \\ &= \prod_{aid \in I_A} \frac{e(g_1^s, g_2^{\frac{\alpha_{aid}}{z_{uid}}} g_2^{\frac{a\beta_{aid}}{z_{uid}}})}{\prod_{i \in I} \left(e((g_2{}^{a\lambda_i \omega_i} \cdot (g_2^{v x_i} g_2^{H(x_i)})^{-\gamma_i \omega_i}, g_1^{\frac{\beta_{aid}}{z_{uid}}}) \cdot e(g_1^{\gamma_i \omega_i}, (g_2^{v x_i} g_2^{H(x_i)})^{\frac{\beta_{aid}}{z_{uid}}}) \right)} \\ &= e(g_1, g_2)^{\sum_{aid \in I_A} \frac{\alpha_{aid} s}{z_{uid}}} \end{aligned}$$

When the attribute $\widetilde{x}_k$ is revoked from a user whose identity is uid:

For $x_i \neq \widetilde{x}_k$:

$$\begin{aligned} CT_i &= \frac{e(C_0 \cdot g_1^{C'}, K_{uid,aid})}{\left(e((C_{i,1} \cdot g_2{}^{aC_{i,3}} \cdot (g_2^{v x_i} g_2^{H(x_i)})^{-C_{i,4}})^{\omega_i}, L_{uid,aid}) \cdot e((C_{i,2} \cdot g_1{}^{C_{i,4}})^{\omega_i}, K_{uid,x_i}) \right)} \\ &= \frac{e(g_1^s, g_2^{\frac{\alpha_{aid}}{z_{uid}}} g_2^{\frac{a\beta_{aid}}{z_{uid}}})}{e((g_2{}^{a\lambda_i \omega_i} \cdot (g_2^{v x_i} g_2^{H(x_i)})^{-\gamma_i \omega_i}, g_1^{\frac{\beta_{aid}}{z_{uid}}}) \cdot e(g_1^{-\gamma_i \omega_i}, (g_2^{v x_i} g_2^{H(x_i)})^{\frac{\beta_{aid}}{z_{uid}}})} \\ &= e(g_1, g_2)^{\frac{\alpha_{aid} s}{z_{uid}}} \end{aligned}$$

For $x_i = \widetilde{x}_k$:

$$\begin{aligned} CT_i &= \frac{e(C_0 \cdot g_1^{C'}, K_{uid,aid})}{\left(e((C_{i,1} \cdot g_2{}^{aC_{i,3}} \cdot (g_2^{v' x_i} g_2^{H(x_i)})^{-C_{i,4}})^{\omega_i}, L_{uid,aid}) \cdot e((C_{i,2} \cdot g_1{}^{C_{i,4}})^{\omega_i}, K_{uid,x_i}) \right)} \\ &= \frac{e(g_1^s, g_2^{\frac{\alpha_{aid}}{z_{uid}}} g_2^{\frac{a\beta_{aid}}{z_{uid}}})}{e((g_2{}^{a\lambda_i \omega_i} \cdot (g_2^{v' x_i} g_2^{H(x_i)})^{-\gamma_i \omega_i}, g_1^{\frac{\beta_{aid}}{z_{uid}}}) \cdot e(g_1^{\gamma_i \omega_i}, (g_2^{v' x_i} g_2^{H(x_i)})^{\frac{\beta_{aid}}{z_{uid}}})} \\ &= e(g_1, g_2)^{\frac{\alpha_{aid} s}{z_{uid}}} \end{aligned}$$

Therefore:

$$CT' = \prod_{aid \in I_A} CT_i = \prod_{aid \in I_A} e(g_1, g_2)^{\frac{\alpha_{aid} s}{z_{uid}}} = e(g_1, g_2)^{\sum_{aid \in I_A} \frac{\alpha_{aid} s}{z_{uid}}}$$

Then:

$$\frac{C}{CT'^{z_{uid}}} = \frac{M \cdot e(g_1, g_2)^{\sum_{aid \in I_A} \alpha_{aid} s}}{e(g_1, g_2)^{\sum_{aid \in I_A} \alpha_{aid} s}} = M$$

Therefore, VO-MAACS satisfies correctness.

5.1.2. Data Confidentiality

In our system, only for users whose attributes satisfy the access policy, will the FD decrypt the ciphertext for them by using their proxy keys. Users whose attributes do not satisfy the access policy, cannot receive the partially decrypted ciphertext from the FD. Thus, they are not able to recover the original data. When a user is revoked, his proxy key will be deleted by the CSP. Without the proxy key, he cannot obtain the partially decrypted ciphertext either. Therefore, for users whose attributes do not satisfy the access policy, our solution satisfies the data confidentiality.

In addition, although the CSP and FD can get user proxy keys, however, if they do not obtain the user secret keys, they still cannot decrypt the ciphertext. Similarly, they cannot collude with other users to recover the data either. Therefore, for the CSP and FD, our solution also satisfies the data confidentiality.

5.1.3. Collusion tolerance

In our system, each user is assigned with a unique identity *uid*, and each key issued by different AA is associated with a *uid*. Therefore, only the keys associated with the same *uid* can be used to decrypt the ciphertext. Other users cannot collude to decrypt the ciphertext. In addition, there exists a situation that some AAs may issue the same attributes. Since each AA has a unique identity *aid*, all attributes are distinguishable. Therefore, users cannot replace some of the components in the keys from the AA by using the component in the key from another AA.

5.2. Performance Analysis

We implement our scheme in Charm [35], a framework developed to facilitate the rapid prototyping of cryptographic schemes and protocols. It is based on the Python language which allows the programmer to write code similar to the theoretical implementations. Charm also provides routines for applying and using LSSS schemes needed for Attribute-Based systems. All our implementations are executed on an Intel® Pentium® CPU G630@270 GHz with 4.00 GB RAM running Ubuntu14.04 64-bit system and Python 2.7.

In our experiment, access policies are generated in the form of $a_1, a_2, \ldots, a_n$, where a_i is an attribute. We set 20 distinct access policies in this form with N increasing from 20 to 200, and repeat each instance 20 times and take the average values as the experiment results. We simulate the computing time incurred in encryption and decryption. Since our scheme is based on Lewko's scheme [14], we compare our scheme with [14] in user encryption time and user decryption time. In our experiments, the number of attributes per authority is set to 10. The times for outsourced encryption are shown in Figure 3a,b, respectively.

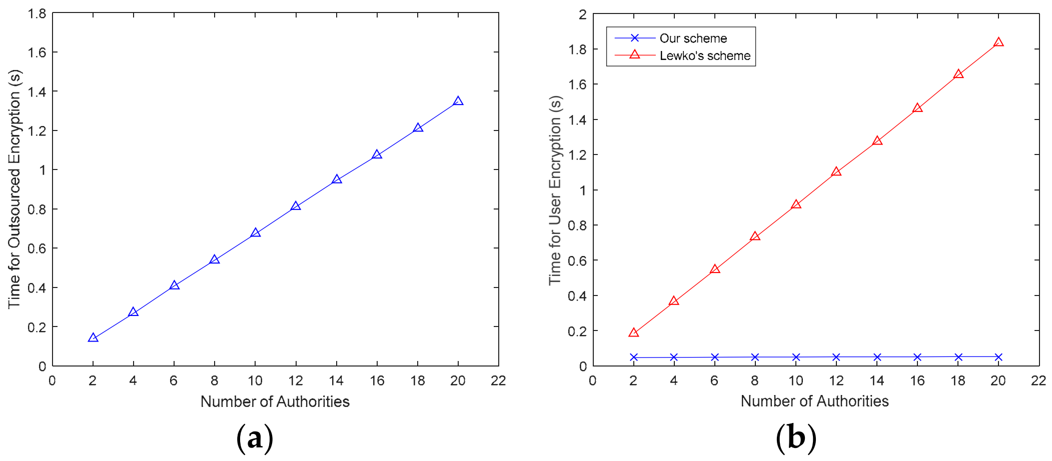

Figure 3. Comparison of encryption time with different number of authorities. (**a**) Encrypt_out time; (**b**) Encrypt_user time.

In Figure 3a, the Encrypt_out time is approximately 0.1~1.4 s, and it increases almost linearly with the number of attributes. In Figure 3b, since major computations are outsourced to the FDs, only a few operations are left for DOs. Therefore, the Encrypt_user time in our scheme is much less than that in [14]. Similarly, Figure 4 describes the time for outsourced decryption and user decryption. In Figure 4a, the Decrypt_out time is approximately 0.3~3 s, and like the Encrpyt_time, it also increases linearly with the number of attributes. In Figure 4b, as major computations are outsourced to FDs, only a few operations are left for DUs, therefore, the Decrypt_user time in our scheme is much less than that in [14].

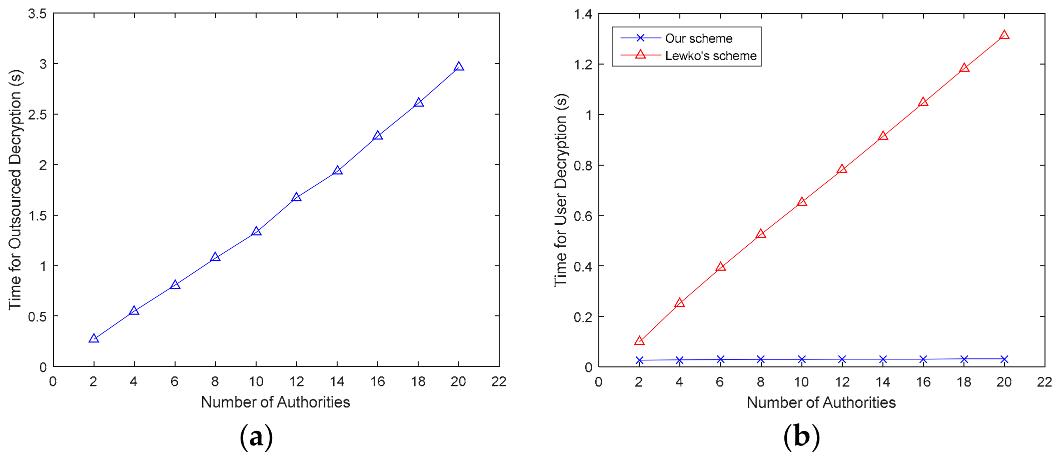

Figure 4. Comparison of decryption time with different number of authorities. (**a**) Decrypt_out time; (**b**) Decrypt_user time.

The computing cost for verification of outsourced encryption is shown in Figure 5. The time for Verify_enc is approximately 0.1~0.8 s and it increases almost linearly with the number of attributes. Figure 6 describes the comparison of computing cost of CSP, AA and DO in the attribute revocation process.

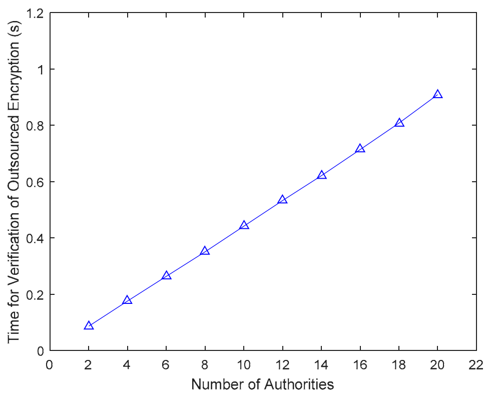

Figure 5. Computing cost for verification of outsourced encryption.

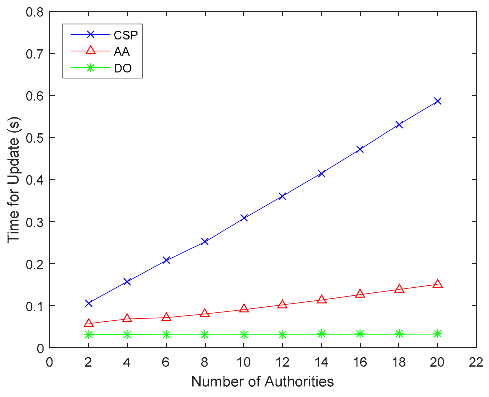

Figure 6. Comparison of computing cost of CSP, AA and DO in the attribute revocation process.

In fact, most computing overhead, such as proxy keys update and ciphertext re-encryption are outsourced to the CSP, and only a few computations are left for AAs and DOs. Therefore, the computing cost for DOs can be greatly reduced. Apparently, our scheme requires less time for both encryption and decryption than Lewko's scheme, and the computing cost for DOs in the attribute revocation process is greatly reduced. Therefore, we can conclude that our scheme's computation efficiency is much better than that of Lewko's scheme.

6. Conclusions

To realize data access control in fog-cloud computing system, we have proposed a verifiable outsourced multi-authority access control scheme, named VO-MAACS. In our construct, most encryption and decryption computations are outsourced to fog devices and the computation results can be verified by using our verification method. Meanwhile, to address the revocation issue, we designed an efficient user and attribute revocation method for this. Finally, the analysis and simulation results show that our scheme is both secure and highly efficient.

Acknowledgments: This work has been financially supported by the National Key Research and Development Program under Grant 2017YFB0802304, the National Natural Science Foundation of China (No. 61303216 and No. 61272457), Natural Science Basic Research Plan in Shaanxi Province of China (No. 2017JM6004), the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No. ICT170312), the Research Project of National Time Service Center, Chinese Academy of Sciences, China (No. 2017FWCGCZ0124) and National 111 Program of China B16037 and B08038.

Author Contributions: K.F. and J.W. conceived and designed the experiments; K.F. and J.W. performed the experiments; X.W. contributed analysis tools; H.L. and Y.Y. analyzed the data; K.F. and J.W. wrote the manuscript. All authors read and approved the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. 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, 17 August 2012; pp. 13–16.
2. Shojafar, M.; Cordeschi, N.; Baccarelli, E. Energy-efficient adaptive resource management for real-time vehicular cloud services. *IEEE Trans. Cloud Comput.* **2016**, *99*, 1. [CrossRef]
3. Zaghdoudi, B.; Ayed, H.K.B.; Harizi, W. Generic Access Control System for Ad Hoc MCC and Fog Computing. In Proceedings of the International Conference on Cryptology and Network Security, Milan, Italy, 14–16 November 2016; pp. 400–415.
4. Baccarelli, E.; Naranjo, P.G.V.; Scarpiniti, M.; Shojafar, M.; Abawajy, J.H. Fog of Everything: Energy-efficient Networked Computing Architectures, Research Challenges, and a Case Study. *IEEE Access* **2017**, *5*, 9882–9910. [CrossRef]
5. Hajibaba, M.; Gorgin, S. A review on modern distributed computing paradigms: Cloud computing, jungle computing and fog computing. *J. Comput. Inf. Technol.* **2014**, *22*, 69–84. [CrossRef]
6. Aazam, M.; Huh, E.N. Fog Computing and Smart Gateway Based Communication for Cloud of Things. In Proceedings of the International Conference on Future Internet of Things and Cloud IEEE, Barcelona, Spain, 27–29 August 2014; pp. 464–470.
7. Stojmenovic, I.; Wen, S.; Huang, X.; Luan, H. An overview of Fog computing and its security issues. *Concurr. Comput. Pract. Exp.* **2015**, *28*, 2991–3005. [CrossRef]
8. Yi, S.; Qin, Z.; 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; pp. 685–695.
9. Lu, R.; Rahulamathavan, Y.; Zhu, H.; Xu, C.; Wang, M. Security and Privacy Challenges in Vehicular Cloud Computing. *Mob. Inf. Syst.* **2016**, *2016*, 1–2. [CrossRef]
10. Lu, R.; Heung, K.; Lashkari, A.H.; Ghorbani, A.A. A Lightweight Privacy-Preserving Data Aggregation Scheme for Fog Computing-Enhanced IoT. *IEEE Access* **2017**, *5*, 3302–3312. [CrossRef]
11. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-policy attribute-based encryption. In Proceedings of the Security and Privacy, Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
12. Chase, M. Multi-authority attribute based encryption. *Theory Cryptogr. Conf.* **2007**, *4392*, 515–534. [CrossRef]
13. Waters, B. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. *Public Key Cryptogr. PKC* **2011**, *6571*, 53–70. [CrossRef]
14. Lewko, A.; Waters, B. Decentralizing attribute-based encryption. In Proceedings of the Advances in Cryptology–EUROCRYPT, Tallinn, Estonia, 15–19 May 2011; pp. 568–588.
15. Ruj, S.; Nayak, A.; Stojmenovic, I. DACC: Distributed access control in Clouds. In Proceedings of the TrustCom, Changsha, China, 16–18 November 2011; pp. 91–98.
16. Zhou, Z.; Huang, D.; Wang, Z. Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption. *IEEE Trans. Comput.* **2015**, *64*, 126–138. [CrossRef]
17. Wang, S.; Zhou, J.; Liu, J.K.; Yu, J.; Chen, J.; Xie, W. An efficient file hierarchy attribute-based encryption scheme in cloud computing. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1265–1277. [CrossRef]
18. Green, M.; Hohenberger, S.; Waters, B. Outsourcing the decryption of abe ciphertexts. In Proceedings of the USENIX Security Symposium, San Francisco, CA, USA, 8–12 August 2011; p. 34.
19. Yang, K.; Jia, X. Attributed-based access control for multi-authority systems in cloud storage. In Proceedings of the 32nd International Conference on Distributed Computing Systems (ICDCS), Macau, China, 18–21 June 2012; pp. 536–545.
20. Yang, K.; Jia, X. Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 1735–1744. [CrossRef]
21. Li, J.; Chen, X.; Li, J.; Jia, C.; Ma, J.; Lou, W. Fine-grained access control system based on outsourced attribute-based encryption. In Proceedings of the European Symposium on Research in Computer Security, Egham, UK, 9–13 September 2013; pp. 592–609.
22. Lai, J.; Deng, R.H.; Guan, C.; Weng, J. Attribute-based encryption with verifiable outsourced decryption. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 1343–1354. [CrossRef]

23. Li, J.; Huang, X.; Li, J.; Chen, X.; Xiang, Y. Securely outsourcing attribute-based encryption with checkability. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2201–2210. [CrossRef]
24. Chen, Y.; Song, L.; Yang, G. Attribute-Based Access Control for Multi-Authority Systems with Constant Size Ciphertext in Cloud Computing. *China Commun.* **2016**, *13*, 146–162. [CrossRef]
25. Li, X.; Tang, S.; Xu, L.; Wang, H.; Chen, J. Two-Factor Data Access Control With Efficient Revocation for Multi-Authority Cloud Storage Systems. *IEEE Access* **2017**, *5*, 393–405. [CrossRef]
26. Mao, X.; Lai, J.; Mei, Q.; Chen, K.; Weng, J. Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 533–546. [CrossRef]
27. Ostrovsky, R.; Sahai, A.; Waters, B. Attribute-based encryption with non-monotonic access structures. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; pp. 195–203.
28. Ibraimi, L.; Petkovic, M.; Nikova, S.; Hartel, P.; Jonker, W. Mediated Ciphertext-Policy Attribute-Based Encryption and Its Application. In *Information Security Applications*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 309–323.
29. Yu, S.; Wang, C.; Ren, K.; Lou, W. Attribute based data sharing with attribute revocation. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China, 13–16 April 2010; pp. 261–270.
30. Hur, J.; Noh, D.K. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 1214–1221. [CrossRef]
31. Xie, X.; Ma, H.; Li, J.; Chen, X. New ciphertext-policy attribute-based access control with efficient revocation. In Proceedings of the Information and Communication Technology-EurAsia Conference, Yogyakarta, Indonesia, 25–29 March 2013; pp. 373–382.
32. Yang, Y.; Liu, J.K.; Liang, K.; Choo, K.R.; Zhou, J. Extended proxy-assisted approach: Achieving revocable fine-grained encryption of cloud data. In Proceedings of the European Symposium on Research in Computer Security, Vienna, Austria, 21–25 September 2015; pp. 146–166.
33. Beimel, A. *Secure Schemes for Secret Sharing and Key Distribution*; Technion-Israel Institute of Technology, Faculty of Computer Science: Haifa, Israel, 1996; pp. 22–28.
34. Bellare, M.; Garay, J.A.; Rabin, T. Fast batch verification for modular exponentiation and digital signatures. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Espoo, Finland, 31 May–4 June 1998; pp. 236–250.
35. Akinyele, J.A.; Garman, C.; Miers, I.; Pagano, M.W.; Rushanan, M.; Green, M.; Rubin, A.D. Charm: A framework for rapidly prototyping cryptosystems. *J. Cryptogr. Eng.* **2013**, *3*, 111–128. [CrossRef]