A Practical and Efficient Bidirectional Access Control Scheme for Cloud-Edge Data Sharing

Jie Cui, Bei Li, Hong Zhong, Geyong Min, Yan Xu, Lu Liu

Abstract—The cloud computing paradigm provides numerous tempting advantages, enabling users to store and share their data conveniently. However, users are naturally resistant to directly outsourcing their data to the cloud since the data often contain sensitive information. Although several fine-grained access control schemes for cloud-data sharing have been proposed, most of them focus on the access control of the encrypted data (e.g., restricting the decryption capabilities of the receivers). Distinct from the existing work, this paper aims to address this challenging problem by developing a more practical bidirectional fine-grained access control scheme that can restrict the capabilities of both senders and receivers. To this end, we systematically investigate the access control for cloud data sharing. Inspired by the access control encryption (ACE), we propose a novel data sharing framework that combines the cloud side and the edge side. The edge server is located in the middle of all the communications, checking and preventing illegal communications according to the predefined access policy. Next, we develop an efficient access control algorithm by exploiting the attribute-based encryption and proxy re-encryption for the proposed framework. The experimental results show that our scheme exhibits superior performance in the encryption and decryption compared to the prior work.

Index Terms—Cloud computing, data sharing, access control, encryption, edge computing

1 INTRODUCTION

THE cloud computing paradigm provides numerous tempting advantages, such as powerful computation capacity, flexible resource sharing and low cost. It enables users to obtain desired services in an unprecedented and convenient manner, regardless of time and location. Cloud storage service, such as iCloud, Dropbox and Google Drive, is one of the most fundamental services provided by cloud computing [1]. By migrating the local data into the cloud, users can enjoy convenient data management and sharing at a low cost. With the development of the information industry, the cloud storage will become more popular in the future.

Traditionally, a fine-grained cloud-data sharing system usually considers the receivers' access control instead of the senders'. Only the receiver who has the decryption right can recover the message. It seems sufficient for a cloud data sharing system. However, for the people who want to control the information flow in the system, this is not enough. Consider an application where a company stores its top-secret documents in the cloud and encrypts them using an access policy that determines which employees are allowed to decrypt the documents. Caused by mistakes or malice, a legitimate employee may share the plaintext directly with other employees who do not have the decryption rights. In addition, a doctor may share a file to a businessman and a man may share a video of campus violence to a student through the public cloud storage. If the cloud server is allowed to check the contents of the sharing file, it will violate the user's data privacy. Moreover, the senders can avoid checking the contents by processing the sharing file, e.g. changing the file type. As a result, how to construct a cloud-data sharing system that can restrict the capabilities of both senders and receivers while preserving the users' privacy is an important and challenging problem.

The general solution to protecting the data privacy is to employ a searchable encryption scheme (SE) to encrypt the data files and potential keywords before uploading them to the cloud. The encrypted data can be subsequently retrieved with a corresponding keyword and decrypted by those who have the decryption keys. However, how can the encrypted data be shared efficiently. To enable a search on the data file, it requires the sender to share his secret key with the receivers or stay online to generate the search trapdoor [2], [3]. To address this problem, Sun et al. [4] proposed an attribute-based keyword search scheme, which provides fine-grained search authorization. Then the file can be searched by those whose attributes satisfy the access policy.

If all the users are honest, using the above method is enough to address the aforementioned issue. But in reality, we must consider the more complicated situation where a sender may be dishonest. To cope with this issue, Damgård *et al.* [5] initialized a study on a novel primitive called access control encryption, which introduces a sanitizer to route messages from senders to receivers. It gives different privileges to different users, indicating which files they can read and which files they can share. To prevent a malicious sender from sending a message in plaintext or

- *J. Cui, B. Li, H. Zhong and Y. Xu are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230039, China, the Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China, and the Institute of Physical Science and Information Technology, Anhui University, Hefei 230039, China (e-mail: zhongh@ahu.edu.cn).*
- *G. Min is with the Department of Computer Science, College of Engineering, Mathematics, and Physical Sciences, University of Exeter, Exeter, EX4 4QF, U.K. (e-mail: g.min@exeter.ac.uk).*
- *L. Liu is with the School of Informatics, University of Leicester, LE1 7RH, UK (email: l.liu@leicester.ac.uk).*

in a screenshot, the sanitizer must secure the message before broadcasting it to the receivers. If the cloud server is used to perform as the sanitizer, it will put a heavy burden on the cloud since the sanitizer must process each message. Especially with the rapid development of Internet-of-Things (IoT) technology, data sharing on the cloud has incorporated many new features, such as real-time and data transmission intensiveness [6]. For instance, in an electronic health system, the data owner may upload health data to the cloud server in real time, and the uploaded data need to be sanitized and stored by the cloud server. If all the processing is handed over to the cloud server, it will cause huge computational pressure on the server. Furthermore, as the cloud server is usually far away from the data owner, there will be a large delay caused by data transmission. To mitigate the computing burden and reduce the transmission latency, we should simplify the sanitizer's operations and possibly outsource it to a third party, which is only semi-trusted.

Inspired by the access control encryption, we propose in this paper a novel bidirectional access control scheme by exploiting attribute-based encryption and proxy re-encryption. The concept of bidirectional was introduced in proxy re-encryption [7], and we use it to describe the access control that restricts the capabilities of both senders and receivers. Specifically, we introduce the edge server to perform as a sanitizer, which can reduce the burden of cloud server. When the edge server receives a data file from a sender, it will check the validity of the information flow by the predefined access policy. We differentiate access policy and access structure in our scheme. In the access control encryption, the access policy is denoted by a predicate $P : [n] \times [n] \rightarrow \{0, 1\}$. A sender S_i is allowed to send files to a receiver R_j only when $P(i, j) = 1$. In our scheme, the access policy is defined by the central authority and is formed as $P : S \times S \rightarrow \{0, 1\}$, where S stands for the attributes of the users (senders and receivers). In addition, the access structure is used for the attribute-based encryption. When a sender wants to send a file to a receiver, the attributes of the sender and the receiver must satisfy the access policy, e.g. the sender and the receiver must have the same attributes (the real access policy can be more complicated). If the information flow is valid, the edge server will perform a signature authentication to verify the identity of the sender and then re-encrypt the data file. The sanitized data file will be sent to the receivers, and be decrypted by those whose attributes satisfy the access structure in the ciphertext.

1.1 Motivation and Contribution

While numerous fine-grained access control schemes have been proposed for data sharing in the cloud [8], [9], most of them have concentrated on allowing fine-grained access to the encrypted data (e.g., restricting the decryption abilities of the receivers). However, when the sender is dishonest, the data may be shared with the receivers who should not get the ciphertext.

In this work, we are motivated to address the above mentioned problem and proposed a secure cloud data sharing scheme that can achieve more practical bidirectional access control. To achieve this goal, we construct an attribute-based access control encryption (AACE) that fulfills the aforementioned bidirectional fine-grained access control. Specifically, cryptographic alone is insufficient for the access control encryption [10], and requires a sanitizer to process all the communication between the senders and receivers. So we construct a secure fine-grained data sharing scheme by combining the cloud server and the edge server. The edge server is located in the middle of all communications, checking and preventing illegal communication. To the best of our knowledge, no systematic investigation of data sharing for cloud-edge computing has been conducted. To fill this gap, this paper makes the following major contributions:

- We develop a novel fine-grained data sharing framework that combines the cloud side and the edge side. In our scheme, we employ the edge server to perform as a sanitizer that re-encrypts and routes the messages between the senders and the receivers according to the predefined information flow access policy.
- We present a formal definition of the attribute-based access control encryption (AACE) method and its corresponding security model. We exploit the attribute based keyword search to realize a fine-grained data access control at the receiver side.
- In addition, we use proxy re-encryption and signature authentication to realize the information flow control of the sender side. This process is performed before the file reaches the receivers. Even if the sender sends a file in plaintext, the receivers will receive a re-encrypted ciphertext.
- Then we provide a concrete attribute based access control encryption (AACE) and a data sharing construction based on the proposed AACE. Compared with many existing cloud data sharing constructions, which assume the sender is fully honest, we use a more practical threat model where the sender can also be malicious, thus requiring a higher security guarantee.

1.2 Related Work

Sahai and Waters [11] invented the notion of attribute-based encryption (ABE). Unlike the traditional public-key encryption, ABE provides a more fine-grained access control of the encrypted data. Goyal *et al.* [12] introduced the definitions of key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). ABE has been shown to have applications in many areas, e.g., electronic health-record (EHR) systems [13] and searchable encryption [14], [15]. However, in public cloud storage, CP-ABE is more practical [16]. Although a variety of protocols [17], [18] have been designed for cloud access control using CP-ABE, many other challenges still exist.

Xue *et al.* [20] proposed an access control scheme for encrypted cloud storage by combining the owner-side and cloud side. They exploited a hybrid attribute-based encryption and a digital signature to ensure the access control on the data user and the integrity of the message, respectively. However, they did not consider the condition that the sender may be malicious. To enhance the access control functionality, Damgård *et al.* [5] conducted a study on access control encryption (ACE). They used a novel control on the information flow, both in terms of what the users could read, as well as what they could send. Additional

TABLE 1
The comparison between our scheme and the related access control encryption

Scheme	Assumption	Predicate	Ciphertext Size	Sanitizer's Key Size
Damgård *et al.* [5] Section 3	DDH or DCR	Arbitrary	$O(2^n)$	$O(n)$
Damgård *et al.* [5] Section 4	iO	Arbitrary	$O(n)$	$O(1)$
Kim and Wu [10]	DDH, RSA and LWE	Arbitrary	$O(n)$	$O(1)$
Han *et al.* [19]	DPBDHE and q-SDH	Arbitary	$O(N)$	$O(1)$
Our scheme	DLIN	Arbitrary	$O(N)$	$O(1)$

* we compare our scheme with related schemes for predicates $\pi : 0, 1^n \times 0, 1^n \rightarrow 0, 1$, where n and N denotes the number of identities and related attributes, respectively.

ACE applications have been proposed [21], [22]. Kim and Wu [10] reduced the construction of an ACE scheme to a proxy re-encryption scheme and presented a generic ACE construction for general policies. Motivated by ACE, Han *et al.* [19] proposed an information-flow control scheme based on ABE, which can prevent corrupt senders from sending messages to unauthorized receivers. However, it is inefficient and cannot be applied directly to cloud-data sharing. In Table 1, we provide the comparison between our scheme and the related access control encryption in terms of assumption, predicate, ciphertext size and the sanitizer's key size.

Blaze *et al.* [23] first introduced the proxy re-encryption. It allows a proxy to process the message without leaking the information of the encrypted message. Liang *et al.* [24] introduced a CP-ABE scheme that supports policy updating through the proxy re-encryption technique. After that, numerous CP-ABE with policy updating using the technique of proxy re-encryption have been proposed [25], [26]. Proxy re-encryption is also widely used in user revocation schemes [27], [28]. However, the proxy re-encryption in our scheme is different from the existing schemes because the senders do not need to generate a transformation key and the proxy just performs a re-randomization on the ciphertext. We exploit the proxy re-encryption to prevent a malicious sender from directly sending a plaintext or screenshot to the receivers.

Recently, edge computing has become a new computing paradigm [29]. It decentralizes the computing power and storage resources of the cloud to edge nodes closer to users to provide low latency and high quality services [30], [31]. In addition, it offers context awareness, mobility and scalability which makes it widely used in the IoT environment [32], [33]. Wang *et al.* [34] presented a novel cloud-edge computing framework, where the cloud server mainly processes large-scale data, while the edge side is used to provide real time service. Combining the cloud side and edge side can bring numerous benefits [35]. For example, applying them to a traditional data sharing system can alleviate the burden on the cloud, and users can enjoy a series of high-quality services at a low latency [36], [37]. Furthermore, it can reduce the network bandwidth usage by mitigating the data transmission from users to the cloud [38].

1.3 Outline

The rest of this paper is organized as follows. In Section II, we introduce the preliminaries and cryptographic primitives involved in our scheme. Then we provide formal definitions of the system and security models in Section III. Next, we present a concrete construction in Section IV. We then present the security proof and experimental results in Sections V and VI, respectively. Finally, we conclude this work in Section VII.

2 PRELIMINARIES

In this section we will describe the preliminaries that are associated with our scheme. All the notations used in this paper are summarized in Table 2.

TABLE 2
Notations

Notations	Definitions
A_S	A set of sender's attributes
A_R	A set of receiver's attributes
λ	A security parameter
p	A large prime
$\mathcal{M}_i$	The ith row of the matrix $\mathcal{M}$
$\mathcal{M}_{i,j}$	The (i, j)th element of the matrix $\mathcal{M}$
$\boldsymbol{c}$	N-dimensional vector $(c_1, \cdots, c_n)$
$g^{\boldsymbol{c}}$	$(g^{c_1}, \cdots, g^{c_2})$
$negl(\lambda)$	A negligible function of λ
π	An access policy
$\|\|$	Concatenation operations
H_1, H_2	Two secure hash function that map $\{0, 1\}^* \rightarrow G$
H_3	A secure hash function that map $\{0, 1\}^* \rightarrow Z_p$

2.1 Access Structure and Monotone Span Program

Definition 1. *(Access Structure [12]).* Let $P_1, \ldots, P_n$ be a set of attributes. A collection $\mathbb{A} \subseteq 2^{P_1,\ldots,P_n}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \subseteq \mathbb{A}$. A monotone access structure is a monotone collection $\mathbb{A}$ of non-empty subsets of $P_1, \ldots, P_n$, i.e. $\mathbb{A} \subseteq 2^{P_1,\ldots,P_n} \setminus \{\emptyset\}$.

If $\mathcal{U}$ stands for the universe of attributes, a Monotone Span Program (MSP) over Z_p is given by a labeled matrix $\mathcal{M}$ of $n_1 \times n_2$ and a mapping: $\rho : \{1, \ldots, n_1\} \rightarrow \mathcal{U}$, which maps the i-th row of $\mathcal{M}$ to an attribute in $\mathcal{U}$. Let $\mathcal{S}$ be a subset of $\mathcal{U}$ and $I = \{i | i \in \{1, \ldots, n_1\}, \rho(i) \in \mathcal{S}\}$ stands for the rows of $\mathcal{M}$(every row is labeled by one literal). The MSP$(\mathcal{M}, \rho)$ accepts $\mathcal{S}$ if there exists coefficients $\{\gamma_i\}_{i\in I}$ such that:

$$\sum_{i\in I} \gamma_i \mathcal{M}_i = (1, 0, 0, \ldots, 0) \tag{1}$$

2.2 Bilinear Map and Dual Paring Vector Spaces

Let G_1 and G_2 denote two cyclic multiplicative groups of prime order p, g_1 is the generators of G_1 and g_2 is the generator of G_2. The bilinear pairing is a map $e : G_1 \times G_2 \rightarrow G_T$ with the following properties:

- Bilinear: for all $g_1 \in G_1$, $g_2 \in G_2$ and $a, b \in Z_p$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$
- Non-degenerate: $e(g_1, g_2) \neq 1$.
- Computability: it is efficient to compute $e(g_1, g_2)$ for all $g_1 \in G_1$ and $g_2 \in G_2$. A pairing is asymmetric if $G_1 \neq G_2$ and no efficient computable homomorphism exists between them.

Dual Paring Vector Spaces$(p, G_T, \boldsymbol{V}, \boldsymbol{V^*}, \boldsymbol{C}, \boldsymbol{C^*})$ *[39]* is a tuple of a prime p, a cyclic group G_T of order p, two n-dimensional vectors $\boldsymbol{V}, \boldsymbol{V^*}$ and their canonical bases: $\boldsymbol{C} := (\boldsymbol{c_1}, \ldots, \boldsymbol{c_n})$ of $\boldsymbol{V}$ and $\boldsymbol{C}^* := (\boldsymbol{c_1^*}, \ldots, \boldsymbol{c_n^*})$ of $\boldsymbol{V^*}$ generated by $Dual(Z_p^n)$ that satisfy the "dual orthonormal", meaning that:

$$\boldsymbol{c_i} \cdot \boldsymbol{c_j^*} = \begin{cases} 0 \ (mod\ p), & i \neq j \\ \delta \ (mod\ p), & i = j \end{cases} \tag{2}$$

where δ is a random element of Z_p. Then for $g_1 \in G_1$ and $g_2 \in G_2$, we have that

$$e(g_1^{\boldsymbol{c_i}}, g_2^{\boldsymbol{c_j^*}}) = 1$$

whenever $i \neq j$, and 1 stands for the identity element in G_T.

2.3 Decision Linear Assumption

Let $p\mathcal{G} = (p, G_1, G_2, G_T, g_1, g_2, e) \leftarrow Gen(1^\lambda)$, where $Gen(1^\lambda)$ is an asymmetric group generator, g_1 is the generator of G_1 and g_2 is the generator of G_2. Let $D := (g_1^{x_1}, g_1^{x_2}, g_2^{x_1}, g_2^{x_2}, g_1^{x_1 r_1}, g_1^{x_2 r_2}, g_2^{x_1 r_1}, g_2^{x_2 r_2})$, $T_0 := (g_1^{r_1+r_2}, g_2^{r_1+r_2}), T_1 := (g_1^r, g_2^r)$ where $x_1, x_2, r_1, r_2, r \xleftarrow{R} Z_p$, the advantage for all PPT adversaries $\mathcal{A}$ in solving the decision linear problem is defined as

$$Adv_{DLIN}^{\mathcal{A}}(\lambda) := \Big| Pr[\mathcal{A}(1^\lambda, p\mathcal{G}, D, T_0) = 1] - Pr[\mathcal{A}(1^\lambda, p\mathcal{G}, D, T_1) = 1] \Big| \leq negl(\lambda)$$

The probability is taken over the randomness used by $\mathcal{A}$. Note that it is hard for all probabilistic polynomial time adversaries $\mathcal{A}$ to distinguish T_0 from T_1 for $p\mathcal{G} \leftarrow Gen, x_1, x_2, r_1, r_2, r \leftarrow Z_p$.

3 SYSTEM MODEL AND SECURITY MODEL

In this section, we describe the cloud-edge data sharing framework proposed in our scheme. Furthermore, the security definitions are provided.

3.1 System Model

As shown in Fig. 1, the access control system consists of five entities: cloud server, edge server, central authority, data owners and data users.

- **Central Authority (CA)** is a fully trusted party that initializes the system and generates the secret keys for users (senders and receivers). In addition, it will publish an information flow policy $\pi : S \times S \rightarrow \{0, 1\}$ to define the relationship between the attributes of senders and receivers.
- **Cloud Server (CS)** is a semi-trusted party that provides data storage service for users. When the receiver requests a data file, cloud server will check whether the receiver can decrypt the data file before downloading.
- **Edge Server (ES)** is a semi-trusted party that processes the requests from the senders. To prevent malicious senders from transmitting files to receivers, the edge server will check the access policy to ensure that the sender can share files with the receivers and re-randomize the ciphertext. The edge server can cache the data file for user downloading when it has sufficient storage.
- **Sender** is the publisher of files. All the communications must be routed by the edge server. Therefore, as shown in the system model, the data file cannot be uploaded to the cloud server directly.
- **Receiver** obtains data files from the edge server and the cloud server. When the sender and the receiver are located in the same area, the receiver can obtain the shared file from the edge server directly. Otherwise, the data file can be transferred to the edge server close to the receiver and then forward to the receiver.

Remark 1. *It is worth nothing that in practical applications, there will be multiple edge servers, and there will be senders and receivers in the area of each edge server. When the sender and the receiver are in the same edge area, the time to upload and download files is reduced. Otherwise, the receiver can get the shared files from the cloud server through the closer edge server. When the edge server has enough memory, it can cache the data files. Because a group of users have the same decryption rights in attribute based encryption, other users with the same decryption rights can directly get the cached shared files from the edge server when they download the files, thus reducing the download delay.*

To achieve the fine-grained access control (provided in Table 3), we consider three types of control among the entities in our system:

TABLE 3
Fine-grained access control in our scheme

Access control	Legal Receiver	Illegal Receiver
Legal Sender	✓	×
Illegal Sender	×	×

- Control I: Sender uses the attribute-based encryption to encrypt the data file, which ensures the fine-grained access control on receivers.
- Control II: The edge server ensures the validity of the information flow through the predefined access policy. If sender's and receiver's attributes satisfy the policy, the communication will go on.
- Control III: The cloud server verifies whether the receiver can decrypt the data file before downloading, which prevents a malicious receiver from downloading the data files.

Next, we give the formal definition of attribute-based access control encryption (AACE).

- $\mathbf{Setup}(\mathbf{1}^\lambda) \rightarrow (\mathbf{msk}, \mathbf{mpk})$: On inputting a security parameter λ, the algorithm returns the master public key mpk and the master secret key msk.
- $\mathbf{Keygen}(\mathbf{msk}, \mathbf{A_U}, \mathbf{id_i}) \rightarrow \mathbf{sk_i}$: On inputting the master secret key msk and a set of attributes A_U along with an identity id_i, the algorithm returns a secret key sk.
- $\mathbf{Encrypt}(\mathbf{mpk}, (\mathcal{M}, \rho), \mathbf{sk_i}, \mathbf{msg}) \rightarrow \mathbf{ct}$: On inputting the master public key mpk, access structure $(\mathcal{M}, \rho)$, a

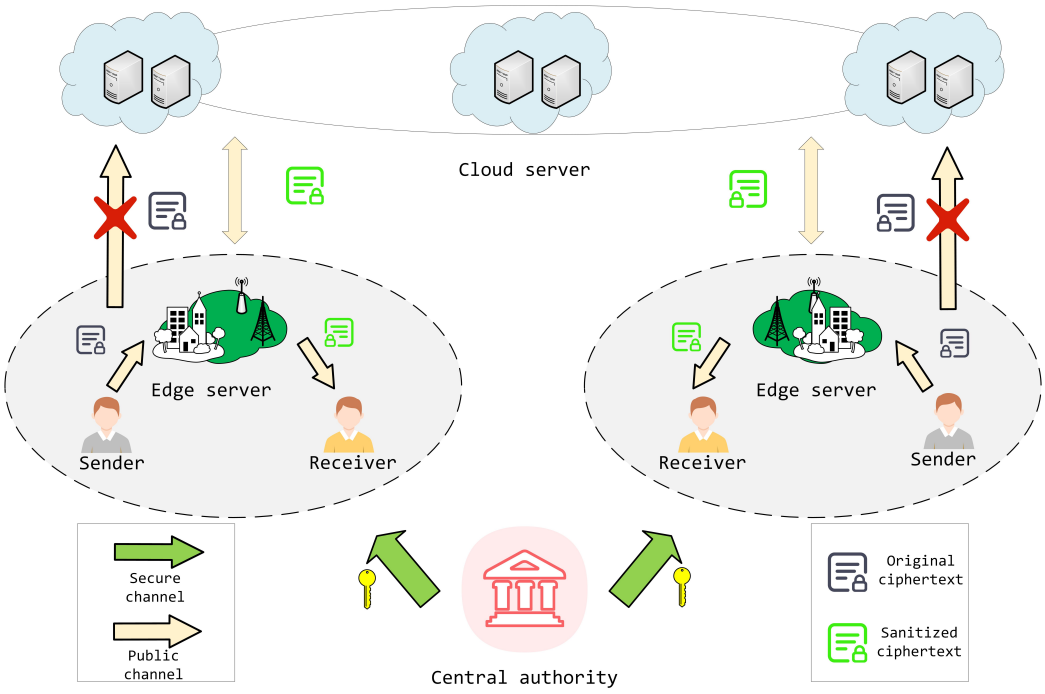


Fig. 1. System model

message $msg \in M$ and the sender's secret key sk_i, the algorithm returns a ciphertext ct.

- $\mathbf{Sanitize}(\mathbf{ct}, \mathbf{mpk}) \rightarrow \mathbf{ct}'$: On inputting a ciphertext ct and the master public key mpk, the algorithm returns the sanitized ciphertext ct' or a invalid symbol $\perp$.
- $\mathbf{Decrypt}(\mathbf{ct}', \mathbf{sk_j}) \rightarrow \mathbf{msg}'$: On inputting a sanitized ciphertext ct' and the receiver's secret key sk_j, the algorithm outputs a message msg' or an invalid symbol $\perp$.

Definition 2. *(Correctness).* An attribute based access control scheme Π_{AACE} is correct if for all messages $msg \in M$, and all attributes $A_S, A_R \in U$ where $\pi(A_S, A_R) = 1$. Let $(mpk, msk) \leftarrow setup(1^\lambda), sk \leftarrow keygen, ct \leftarrow encrypt(sk_i, msg)$, we have that

$$Pr[decrypt(sk_j, sanitize(mpk, ct)) = msg] = 1 - negl(\lambda).$$

3.2 Security Model

The security notions (*no-read up* and *no-write down*) were invented for the information flow control [10]. Recently, it was improved to suit for an ACE scheme: the *no-read rule* and the *no-write rule* [5]. In our scheme, the no-read rule guarantees that only the privileged receivers should be able to decrypt the data file. The no-write rule guarantees that a sender can only share files with the receivers when their attributes satisfy the access policy. Specifically, no sender with attributes A_S should be able to construct a valid ciphertext which can be accessed by a receiver with attributes A_R whenever $\pi(A_S, A_R) = 0$. Here we give the formal definitions.

Definition 3. *(no-read rule).* Let Π_{AACE} be an attribute based access control encryption scheme. Given a security parameter λ and a bit $b \in 0, 1$, we define the no-read rule experiment $\mathsf{Expt}^{\mathsf{read}}_{\Pi_{\mathsf{AACE}},\mathcal{A}}(\lambda, \mathsf{b})$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- setup: The challenger $\mathcal{C}$ runs $\mathsf{setup}(1^\lambda)$ to obtain mpk, msk, and gives mpk to $\mathcal{A}$.
- key query: On inputting a set of attributes A_U, the challenger $\mathcal{C}$ returns a secret key $\mathsf{sk} = \mathsf{KeyGen}(\mathsf{msk}, \mathsf{S})$ and gives it to $\mathcal{A}$.
- challenge: On inputting a pair of messages (msg_0, msg_1) and an access structure $\mathbb{A}$, the challenger $\mathcal{C}$ returns a ciphertext $\mathsf{ct} = \mathsf{Encrypt}(\mathsf{mpk}, \mathbb{A}, \mathsf{msg_b})$ and gives it to $\mathcal{A}$.

$\mathcal{A}$ outputs a bit b' as the output of the game. An attribute based access control encryption scheme is called no-read rule security if the advantage for all PPT adversaries $\mathcal{A}$ is,

$$Adv^{\mathcal{A}}_{\Pi}(\lambda) := \Big| Pr[Expt^{read}_{\Pi,\mathcal{A}}(\lambda, 0) = 1] - Pr[Expt^{read}_{\Pi,\mathcal{A}}(\lambda, 1) = 1] \Big| = negl(\lambda).$$

The no-read rule actually guarantees control I. An attribute based access control encryption scheme that satisfies the no-read rule security means that a non-privileged receiver cannot be able to decrypt the data file. And sender anonymity is also required in the no-read rule. In our scheme, we employ an anonymous identity to protect the senders' identity privacy.

Definition 4. *(no-write rule).* Let Π_{AACE} be an attribute based access control encryption scheme. Given a security parameter λ and a bit $b \in 0, 1$, we define the no-write rule experiment $\mathsf{Expt}^{\mathsf{write}}_{\Pi_{\mathsf{AACE}},\mathcal{A}}(\lambda, \mathsf{b})$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- setup: The challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda)$ to obtain mpk, msk, and gives mpk to $\mathcal{A}$.

- key query: On inputting a set of attributes A_U, the challenger $\mathcal{C}$ returns a secret key $\mathsf{sk} = \mathsf{KeyGen}(\mathsf{msk}, \mathsf{S})$ and gives it to $\mathcal{A}$.
- challenge: On inputting a ciphertext ct and an access structure $\mathbb{A}$, the challenger sets $\mathsf{ct_0} = \mathsf{ct}$. Then it uniformly selects a message $\mathsf{msg'} \leftarrow \mathsf{M}$ and returns a sanitized ciphertext $\mathsf{ct_1} = \mathsf{Sanitize}(\mathsf{Encrypt}(\mathsf{mpk}, \mathbb{A}, \mathsf{msg'}), \mathbb{A})$ and gives it to $\mathcal{A}$.

$\mathcal{A}$ outputs a bit b' as the output of the game. An attribute based access control encryption scheme is called no-write rule security if the advantage for all PPT adversaries $\mathcal{A}$ is,

$$Adv_{\Pi}^{\mathcal{A}}(\lambda) := \big|Pr[Expt_{\Pi,\mathcal{A}}^{write}(\lambda,0)=1] - Pr[Expt_{\Pi,\mathcal{A}}^{write}(\lambda,1)=1]\big| = negl(\lambda).$$

The no-write rule actually guarantees control II, and ensures that even a legal sender should not be able to share files with the specified receiver if they do not satisfy the access policy.

4 OUR CONSTRUCTION

In this section, we will describe the cloud-edge data sharing system design. Firstly, we will give a detailed description of the AACE algorithm. Then we will describe the specific operations of the system.

4.1 The Proposed AACE Algorithm

First, function $\mathbf{Setup}()$ is run by the central authority (CA) to generate the master public and secret key pair. The process of this function is given in Function 1.

Function 1: Setup

INPUT: The secret parameter 1^λ.
OUTPUT: The master public key mpk and master secret key msk.

1) CA runs an asymmetric group generator $Gen(1^\lambda)$ to obtain $(p, G_1, G_2, G_T, e, g_1, g_2)$, where g_1 and g_2 are the generators of G_1 and G_2, respectively.
2) Let S be a set of attributes. CA publishes a policy $\pi : S \times S$ to define which senders can communicate with the specified receivers. It then picks $a_1, a_2, b_1, b_2 \leftarrow Z_p^*, d_1, d_2, d_3 \leftarrow Z_p$.
3) Finally it returns $(g_1, g_2, h_1 = g_2^{a_1}, h_2 = g_2^{a_2}, T_1 = e(g_1,g_2)^{d_1a_1+d_3}, T_2 = e(g_1,g_2)^{d_2a_2+d_3})$ as the master public key mpk, and outputs $(a_1, a_2, b_1, b_2, g_1^{d_1}, g_1^{d_2}, g_1^{d_3})$ as the master secret key msk.

Second, function $\mathbf{KeyGen}()$ is run by the central authority (CA) to generate the public and secret key pair for the user. The process is given in Function 2.

Third, function $\mathbf{Encrypt}()$ is run by the sender to encrypt the message. The process of this function is given in Function 3.

Fourth, function $\mathbf{Sanitize}()$ is run by the edge server to sanitize the ciphertext. The process of this function is given in Function 4.

Finally, function $\mathbf{Decrypt}()$ is run by the data receiver to recover the message. The process of this function is shown in Function 5.

Function 2: KeyGen

INPUT: The master secret key msk, user identity id_i and his attribute sets A_U.
OUTPUT: User's secret key and public key.

1) Select $r_1, r_2 \leftarrow Z_p$ and compute:
$$sk_0 = (sk_{0,1}, sk_{0,2}, sk_{0,3}) = (g_2^{b_1 r_1}, g_2^{b_2 r_2}, g_2^{r_1+r_2})$$
2) For attributes $y \in A_U$ and $t = 1, 2$, CA selects $\sigma_y \leftarrow Z_p$ and computes:
$$sk_{y,t} = H_1(y1t)^{\frac{b_1 r_1}{a_t}} \cdot H_1(y2t)^{\frac{b_2 r_2}{a_t}} \cdot H_1(y3t)^{\frac{r_1+r_2}{a_t}} \cdot g_1^{\frac{\sigma_y}{a_t}}$$
and sets $sk_y = (sk_{y,1}, sk_{y,2}, sk_{y,3})$, where $sk_{y,3} = g_1^{-\sigma_y}$.
3) Select σ' and compute:
$$sk'_t = g_1^{d_t} \cdot H_2(11t)^{\frac{b_1 r_1}{a_t}} \cdot H_2(12t)^{\frac{b_2 r_2}{a_t}} \cdot H_2(13t)^{\frac{r_1+r_2}{a_t}} \cdot g_1^{\frac{\sigma'}{a_t}}$$
and set $sk' = (sk'_1, sk'_2, sk'_3)$, where $sk'_3 = g_1^{d_3} \cdot g_1^{-\sigma'}$.
4) Run $Dual(Z_p^4)$ algorithm to obtain two orthonormal bases $\boldsymbol{C}$ and $\boldsymbol{C^*}$. And then choose $\alpha \in Z_p^*$ to compute $pk = (T_3 = e(g_1,g_2)^{\alpha \mathbf{c_1} \mathbf{c_1^*}}, h_3 = g_1^{\mathbf{c_1}}, h_4 = g_1^{\mathbf{c_2}}, h_5 = g_1^{H_3(id_i)})$ as the public key and $sk_\theta = (\alpha, g_2^{\mathbf{c_1^*}}, g_2^{\mathbf{c_2^*}})$ as the signature key.
5) Output the secret key $(sk_0, sk_{y\{y\in A_U\}}, sk', sk_\theta)$ and the public key pk.

Function 3: Encrypt

INPUT: The master public key mpk, access structure $(\mathcal{M}, \rho)$ and a message msg.
OUTPUT: The ciphertext CT.

1) The sender selects $s_1, s_2 \leftarrow Z_p$ and computes:
$$ct_0 = (ct_{0,1}, ct_{0,2}, ct_{0,3}) = (g_2^{a_1 s_1}, g_2^{a_2 s_2}, g_2^{s_1+s_2})$$
2) Suppose $\mathcal{M}$ is a $n_1 \times n_2$ rows matrix. Then for $i \in \{1, \ldots, n_1\}$ and $l \in \{1, 2, 3\}$ it computes:
$$ct_{i,l} = H_1(\rho(i)l1)^{s_1} \cdot H_1(\rho(i)l2)^{s_2} \cdot \prod_{j=1}^{n_2} [H_2(jl1)^{s_1} \cdot H_2(jl2)^{s_2}]^{\mathcal{M}_{i,j}}$$
and sets $ct_i = (ct_{i,1}, ct_{i,2}, ct_{i,3})$.
3) Compute $ct' = T_1^{s_1} T_2^{s_2} \cdot msg$ and set $ct = (ct_0, ct_1, \ldots, ct_{n_1}, ct')$ as the ciphertext.
4) To prove the sender's identity, it selects $r \in Z_p^*$ and a set of attributes A_R satisfying the access structure $(\mathcal{M}, \rho)$ then computes:
$$\beta = H_3(h_5||T||CT||A_S||A_R), \theta = g_2^{(\alpha+r\beta)\mathbf{c_1^*} - r\mathbf{c_2^*}}$$
where A_S is a subset of the sender's attributes that satisfy the access policy $\pi(A_S, A_R) = 1$ and T is the current time.
5) Then the sender sends $(ct, pk, T, A_S, A_R, \theta, (\mathcal{M}, \rho))$ as the ciphertext CT to the edge server.

4.2 System Operations

The AACE scheme is designed for cloud-edge data sharing. The whole process of AACE includes system initialization, user registration, file sharing, file sanitize, and file access operations. The main operations are shown in Fig.2.

1) System initialization: In this phase, CA runs the setup algorithm to obtain a master public key mpk and master secret key msk.
2) User registration: When a user enters the data sharing system for the first time, CA will run keygen $\rightarrow$ (pk, sk)

Function 4: Sanitize

INPUT: The ciphertext CT.
OUTPUT: The sanitized ciphertext CT'.

1) Edge server first checks the freshness of the message, and rejects the message if it is not fresh.
2) Edge server then checks whether the sender can share data files with the receiver through the access policy $\pi(A_S, A_R) \stackrel{?}{=} 1$. If the sender's attributes and receiver's attributes satisfy the policy, edge server will verify the validity of the received message.
3) Edge server checks whether the equation: $e(g_1^{\mathbf{c_1}+\beta \mathbf{c_2}}, \theta) = T_3$ holds to verify the validity of the received message. If it does not hold, the sanitizer rejects the message; Otherwise, the edge server accepts the message.
4) If the above process succeeds, the edge server selects $s_1', s_2' \leftarrow Z_p$ and rerandomizes the ciphertext:

$$\begin{aligned} ct_0' &= (ct_{0,1} \cdot h_1^{s_1'}, ct_{0,2} \cdot h_2^{s_2'}, ct_{0,3} \cdot g_2^{s_1'+s_2'}) \\ &= (g_2^{a_1(s_1+s_1')}, g_2^{a_2(s_2+s_2')}, g_2^{s_1+s_2+s_1'+s_2'}) \end{aligned}$$

5) For $i \in \{1, \ldots, n_1\}$ and $l \in \{1, 2, 3\}$ it computes:

$$\begin{aligned} ct_{i,l}' = ct_{i,l} &\cdot H_1(\rho(i)l1)^{s_1'} \cdot H_1(\rho(i)l2)^{s_2'} \\ &\cdot \prod_{j=1}^{n_2} [H_2(jl1)^{s_1'} \cdot H_2(jl2)^{s_2'}]^{\mathcal{M}_{i,j}} \end{aligned}$$

$ct'' = ct' \cdot T_1^{s_1'} \cdot T_2^{s_2'}$ and set $ct_i' = (ct_{i,1}', ct_{i,2}', ct_{i,3}')$.
6) Finally it outputs the sanitized ciphertext $CT' = (ct_0', ct_1', \ldots, ct_{n_1}', ct'')$.

Function 5: Decrypt

INPUT: The master public key mpk, the sanitized ciphertext CT' and the secret key sk.
OUTPUT: The recorded message msg.

1) The receiver parses the ciphertext CT' as $CT = (ct_0, ct_1, \ldots, ct_{n_1}, ct')$ and the secret key $sk = (sk_0, sk_{y_{y \in S}}, sk', sk_\sigma)$.
2) If the attributes in the secret key sk satisfy the access structure $(\mathcal{M}, \rho)$ in ciphertext CT, then according to Eq. (1) we can always find a set of constants $\gamma_{i\{i=1,\ldots,n_1\}}$ that satisfy the equation, then compute:

$$\begin{aligned} C = ct' &\cdot e(\prod_{i=1}^{n_1} ct_{i,1}^{\gamma_i}, sk_{0,1}) \cdot e(\prod_{i=1}^{n_1} ct_{i,2}^{\gamma_i}, sk_{0,2}) \\ &\cdot e(\prod_{i=1}^{n_1} ct_{i,3}^{\gamma_i}, sk_{0,3}) \end{aligned}$$

$$\begin{aligned} D = e(sk_1' &\cdot \prod_{i=1}^{n_1} sk_{\rho(i),1}^{\gamma_i}, ct_{0,1}) \cdot e(sk_2' \cdot \prod_{i=1}^{n_1} sk_{\rho(i),2}^{\gamma_i}, ct_{0,2}) \\ &\cdot e(sk_3' \cdot \prod_{i=1}^{n_1} sk_{\rho(i),3}^{\gamma_i}, ct_{0,3}) \end{aligned}$$

3) Then it recovers and outputs the message as $msg = \frac{C}{D}$.

to generate a pair of keys, and return them to the user (senders and receivers).

3) File sharing: The sender employs a searchable encryption scheme to encrypt data file and then uses the proposed AACE algorithm to encrypt the encryption key k and the associated file tag t. Let $ct_m = SE(k, m)$ denote the encryption of the sharing data and $ct_k = AACE.Encrypt(pp, (k, t))$ to be the encryption of the keys and the file tag. After that, the sender computes a token $tkn = PRF(k, t)$ where PRF is a pseudo-random function keyed by k. The token will be used for the downloading authentication, since only the receivers whose attributes satisfy the access policy can decrypt the ciphertext ct_k. Then the token along with (ct_m, ct_k) will be sent to the edge server.
4) File sanitizing: When the edge server receives tkn and (ct_m, ct_k) from the sender, it first runs the sanitize algorithm to process the ct_k. If the sender can share the data with the specified receivers, the edge server will send (tkn, ct_m) to the cloud server and broadcast the sanitized ciphertext ct_k' to all receivers.
5) File access: The receiver uses his secret key to decrypt the ct_k', when the attributes in his keys satisfy the access structure, then he can retrieve the message (k, t). Then he computes $tkn = PRF(k, t)$ and sends this to the cloud server. If the token is valid, the cloud server will retrieve the data file and send it to the edge server closer to the receiver.

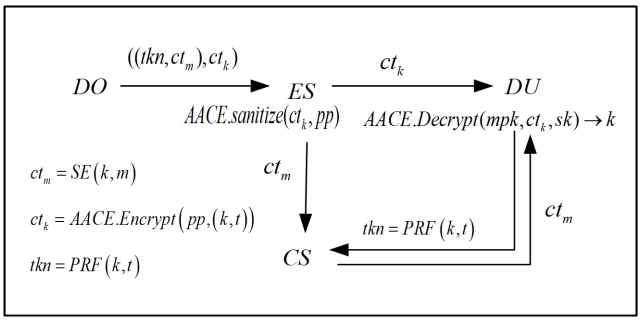


Fig. 2. System operations

5 SECURITY ANALYSIS OF OUR SCHEME

In this section, we first analyze the correctness of our construction and then provide a formal security proof of our scheme. Specifically, we deduce the correctness and security to the underlying primitive. From the proof we can conclude that our scheme satisfies the no-read rule security and the no-write rule security.

5.1 Correctness Analysis

For a valid signature $\theta = g_2^{(\alpha+r\beta)\mathbf{c_1^*} - r\mathbf{c_2^*}}$, it holds

$$\begin{aligned} &e(g_1^{\mathbf{c}_1+\beta \mathbf{c}_2}, \theta) = e(g_1^{\mathbf{c}_1+\beta \mathbf{c}_2}, g_2^{(\alpha+r\beta)\mathbf{c_1^*} - r\mathbf{c_2^*}}) \\ &= e(g_1, g_2)^{(\alpha+r\beta)\mathbf{c}_1\mathbf{c}_1^* - r\beta \mathbf{c}_2\mathbf{c}_2^*} = e(g_1, g_2)^{\alpha \mathbf{c}_1\mathbf{c}_1^*} = T_3 \end{aligned}$$

For a sanitized ciphertext CT',

$$\begin{aligned} D = e(sk_1' &\cdot \prod_{i=1}^{n_1} sk_{\rho(i),1}^{\gamma_i}, ct_{0,1}) \cdot e(sk_2' \cdot \prod_{i=1}^{n_1} sk_{\rho(i),2}^{\gamma_i}, ct_{0,2}) \\ &\cdot e(sk_3' \cdot \prod_{i=1}^{n_1} sk_{\rho(i),3}^{\gamma_i}, ct_{0,3}) \end{aligned}$$

$$\begin{aligned} &= e(sk_1', ct_{0,1}) \cdot e(sk_2', ct_{0,2}) \cdot e(sk_3', ct_{0,3}) \cdot e(\prod_{i=1}^{n_1} sk_{\rho(i),1}^{\gamma_i}, ct_{0,1}) \\ &\cdot e(\prod_{i=1}^{n_1} sk_{\rho(i),2}^{\gamma_i}, ct_{0,2}) \cdot e(\prod_{i=1}^{n_1} sk_{\rho(i),3}^{\gamma_i}, ct_{0,3}) \end{aligned}$$

Then we take the first component $g_1^{d_t}$ of sk'_i and compute,

$$\begin{aligned} & e(g_1^{d_1}, ct_{0,1}) \cdot e(g_1^{d_2}, ct_{0,2}) \cdot e(g_1^{d_3}, ct_{0,3}) \\ & = e(g_1, g_2)^{d_1 a_1 (s_1 + s'_1)} \cdot e(g_1, g_2)^{d_2 a_2 (s_2 + s'_2)} \\ & \cdot e(g_1, g_2)^{d_3 (s_1 + s'_1 + s_2 + s'_2)} = T_1^{s_1 + s'_1} \cdot T_2^{s_2 + s'_2} \end{aligned}$$

which can be divided by the ct' in the sanitized ciphertext, and for the rest components in C and D,

$$\begin{aligned} \prod_{i=1}^{n_1} ct_{i,l}^{\gamma_i} &= \prod_{i=1}^{n_1} \cdot H_1(\rho(i)l1)^{(s_1 + s'_1)\gamma_i} \cdot H_1(\rho(i)l2)^{(s_2 + s'_2)\gamma_i} \\ & \cdot \prod_{j=1}^{n_2} [H_2(jl1)^{(s_1 + s'_1)} \cdot H_2(jl2)^{(s_2 + s'_2)}]^{\gamma_i \mathcal{M}_{i,j}} \\ & = \prod_{j=1}^{n_2} [H_2(jl1)^{(s_1 + s'_1)} \cdot H_2(jl2)^{(s_2 + s'_2)}]^{\sum_{i=1}^{n_1} \gamma_i \mathcal{M}_{i,j}} \\ & \cdot \prod_{i=1}^{n_1} \cdot H_1(\rho(i)l1)^{\gamma_i (s_1 + s'_1)} \cdot H_1(\rho(i)l2)^{\gamma_i (s_2 + s'_2)} \\ & = H_2(1l1)^{(s_1 + s'_1)} \cdot H_2(1l2)^{(s_2 + s'_2)} \\ & \cdot \prod_{i=1}^{n_1} \cdot H_1(\rho(i)l1)^{\gamma_i (s_1 + s'_1)} \cdot H_1(\rho(i)l2)^{\gamma_i (s_2 + s'_2)} \end{aligned}$$

Then it is easy to see that the rest components in C and D are equal.

5.2 Security Analysis

Firstly, we will use some compact representations to simplify the proof. Following [40], in our scheme, $[x]_1$ stands for g_1^x, $[y]_2$ stands for g_2^y and $[z]_T$ stands for $e(g_1, g_2)^z$. For a column vector $\boldsymbol{v} := (v_1, \ldots, v_n)^T$, $[\boldsymbol{v}]_1$ is a n-dimensional tuple $(g_1^{v_1}, \ldots, g_1^{v_n})^T$. It is similar for a matrix $\mathcal{M}$. And for two matrices $\boldsymbol{A}, \boldsymbol{B}$, $[\boldsymbol{A}^T \boldsymbol{B}]_T$ denotes $e([\boldsymbol{A}]_1, [\boldsymbol{B}]_2)$. The outputs of $\mathsf{Samp}(\lambda)$ is

$$\mathrm{Z} := \begin{bmatrix} u_1 & 0 \\ 0 & u_2 \\ 1 & 1 \end{bmatrix}, z^{\perp} := \begin{bmatrix} u_1^{-1} \\ u_2^{-1} \\ -1 \end{bmatrix},$$

where $u_1, u_2 \xleftarrow{R} Z_p^*$. If we set

$$\boldsymbol{A} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{bmatrix}, \boldsymbol{r} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \boldsymbol{r'} = \begin{bmatrix} r_1 \\ r_2 \\ r \end{bmatrix}$$

then we can rewrite the DLIN assumption as

$$([\boldsymbol{A}]_1, [\boldsymbol{A}]_2, [\boldsymbol{A}\boldsymbol{r}]_1, [\boldsymbol{A}\boldsymbol{r}]_2) \approx ([\boldsymbol{A}]_1, [\boldsymbol{A}]_2, [\boldsymbol{r'}]_1, [\boldsymbol{r'}]_2)$$

where the symbol $\approx$ means the former is indistinguishable from the latter.

Theorem 1. *Our attribute based access control encryption scheme satisfies the no-read rule if no adversary can efficient break the experiment* $\mathsf{Expt}^{\mathsf{read}}_{\Pi_{\mathsf{AACE}},\mathcal{A}}(\lambda, \mathsf{b})$ *with a nonnegligible probability.*

Proof. We use a series of hybrid experiments to prove the security. The zeroth hybrid experiment, Hyb_0, is of course the AACE security experiment $\mathsf{Expt}^{\mathsf{read}}_{\Pi_{\mathsf{AACE}},\mathcal{A}}(\lambda, \mathsf{b})$. At first $\mathcal{C}$ runs the setup algorithm to initialize the system and obtain mpk and msk. It then generates the public-secret key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{keygen}$ and gives $\mathsf{mpk}, \mathsf{pk}$ to $\mathcal{A}$. Upon receiving a query from $\mathcal{A}$, $\mathcal{C}$ runs the keygen algorithm as in the real scheme to interact with $\mathcal{A}$. When $\mathcal{A}$ makes a challenge query with a pair of messages $\mathsf{msg}_0, \mathsf{msg}_1 \in \mathsf{M}$, the challenger $\mathcal{C}$ answers the queries by computing the $\mathsf{encrypt}(\mathsf{sk}, \mathsf{msg}_\mathsf{b})$ algorithm.

We first revise the experiment. This modified form will be the first hybrid experiment, Hyb_1. The modified experiment $\mathsf{Expt}^{\mathsf{read}}_{\Pi_{\mathsf{AACE}},\mathcal{A}}(\lambda, \mathsf{b})$ is defined as follows

setup: Run the group generator to obtain the public parameters as before. Then use the $\mathsf{Samp}(\mathsf{p})$ algorithm to obtain $(\boldsymbol{A}, \boldsymbol{a}^{\perp}), (\boldsymbol{B}, \boldsymbol{b}^{\perp})$. Select $d_1, d_2, d_3 \xleftarrow{R} Z_p$ and set $\boldsymbol{d} = (d_1, d_2, d_3)^T$ be a column vector. Finally, it outputs $mpk := ([\boldsymbol{A}]_2, [\boldsymbol{d}^T \boldsymbol{A}]_T)$ and $msk := (pp, \boldsymbol{A}, \boldsymbol{B}, [\boldsymbol{d}]_1)$.

key query: The challenger $\mathcal{C}$ maintains two lists L_1 and L_2 to simulate the random oracle. The entries of L_1 is formed by $(x, \boldsymbol{W}_x)$ or $(j, \boldsymbol{U}_j)$ where $x \in \{0,1\}^*$ and $j \leftarrow Z_p^*$, and $\boldsymbol{W}_x, \boldsymbol{U}_j$ are 3×3 matrices over Z_p. And the entries of L_2 is formed by (q, r) where q denotes the query that $\mathcal{A}$ will make, and r is an element in G. When A makes a query of xlt, for $l \in \{1,2,3\}$ and $t \in \{1,2\}$, $\mathcal{C}$ first checks whether the query (xlt, r) has been queried in L_2. If the query exists, $\mathcal{C}$ returns r, otherwise $\mathcal{C}$ checks whether $(x, \boldsymbol{W}_x)$ in L_1. If yes, $\mathcal{C}$ will compute $r := [(\boldsymbol{W}_x^T \boldsymbol{A})_{l,t}]_1$, then returns r and appends (xlt, r) to L_2. Otherwise, it picks a random 3×3 matrices $\boldsymbol{W}_x$ and appends $(x, \boldsymbol{W}_x)$ to L_1, then $\mathcal{C}$ computes r as the former case and appends (xlt, r) to L_2. Finally, r is given to $\mathcal{A}$. When the query is $0xlt$, $\mathcal{C}$ checks whether the query $(0jlt, r)$ can be found in L_2. If the query exists, $\mathcal{C}$ returns r, otherwise $\mathcal{C}$ checks whether $(j, \boldsymbol{U}_j)$ in L_1. If yes, $\mathcal{C}$ will compute $r := [(\boldsymbol{U}_j^T \boldsymbol{A})_{l,t}]_1$, then append $(0jlt, r)$ to L_2 and return r. Otherwise, it picks a random 3×3 matrices $\boldsymbol{U}_j$ and appends $(j, \boldsymbol{U}_j)$ to L_1, then $\mathcal{C}$ computes r as the former case and appends $(0jlt, r)$ to L_2. Finally, r is given to $\mathcal{A}$. When the query is anything else, C checks whether (q, r) has been queried in L_2. If yes, then $\mathcal{C}$ returns r. Else, $\mathcal{C}$ selects $r' \in G$ and appends (q, r') to L_2. Finally, r' is given to $\mathcal{A}$.

Upon receiving a key query Q from $\mathcal{A}$, $\mathcal{C}$ first checks whether the query has been made. For every $y \in Q$, if $(y, \boldsymbol{W}_y)$ or $\boldsymbol{U}_1$ cannot be found in list L_1, then $\mathcal{C}$ generates them in the above way. Otherwise $\mathcal{C}$ computes $sk_0 = [\boldsymbol{B}\boldsymbol{r}]_2, sk_y = [\boldsymbol{W}_y \boldsymbol{B}\boldsymbol{r} + \sigma_y \boldsymbol{a}^{\perp}]$, and $sk' = [\boldsymbol{d} + \boldsymbol{U}_1 \boldsymbol{B}\boldsymbol{r} + \sigma' \boldsymbol{a}^{\perp}]_1$, where $r_1, r_2, \sigma', \sigma_y$ are randomly picked from Z_p, and $\boldsymbol{r}$ denotes a 2-dimensional vector $(r_1, r_2)^T$. Finally, $(sk_0, \{sk_y\}_{y \in S}, sk')$ is given to $\mathcal{A}$.

encryption query: When $\mathcal{C}$ receives a message msg and an access policy $(\mathcal{M}, \rho)$ from $\mathcal{A}$, $\mathcal{C}$ first checks whether the query has been made. If $[(\boldsymbol{W}_{\rho(i)}^T \boldsymbol{A})_{l,t}]_1$ or $[(\boldsymbol{U}_j^T \boldsymbol{A})_{l,t}]_1$ cannot be found in list L_2, then $\mathcal{C}$ generates them in the above way. Otherwise $\mathcal{C}$ computes $ct_0 = [\boldsymbol{A}\boldsymbol{s}]_2, ct_i = [\boldsymbol{W}_{\rho(i)}^T \boldsymbol{A}\boldsymbol{s} + \sum_{j=1}^{n_2} (\mathcal{M})_{i,j} \boldsymbol{U}_j^T \boldsymbol{A}\boldsymbol{s}]_1$, and $ct' = [\boldsymbol{d}^T \boldsymbol{A}\boldsymbol{s}]_T \cdot msg$, where s_1 and s_2 are randomly picked from Z_p, and $\boldsymbol{s}$ denotes a 2-dimensional vector $(s_1, s_2)^T$. Finally, $(ct_0, \{ct_i\}_{i=1,\ldots,n_1}, ct')$ is given to $\mathcal{A}$.

sanitize query: When $\mathcal{C}$ receives a query of $(ct, A_S, A_R, \theta, (\mathcal{M}, \rho))$, if A_S and A_R satisfies $\pi(A_S, A_R) = 1$ and the signature is valid, then he checks whether the query has been made. If $[(\boldsymbol{W}_{\rho(i)}^T \boldsymbol{A})_{l,t}]_1$ or $[(\boldsymbol{U}_j^T \boldsymbol{A})_{l,t}]_1$ cannot be found in list L_2, then $\mathcal{C}$ generates them as the Encryption query does. Otherwise $\mathcal{C}$ computes $ct'_0 = ct_0[\boldsymbol{A}\boldsymbol{s'}]_2, ct'_i = ct_i[\boldsymbol{W}_{\rho(i)}^T \boldsymbol{A}\boldsymbol{s'} +$

$\sum_{j=1}^{n_2}(\mathcal{M})_{i,j}\boldsymbol{U}_j^T\boldsymbol{As'}]_1$, and $ct'' = ct'[\boldsymbol{d}^T\boldsymbol{As'}]_T$, where s_1 and s_2 are randomly picked from Z_p, and $\boldsymbol{s'}$ denotes a 2-dimensional vector $(s_1', s_2')^T$. Finally, $(ct_0', \{ct_i'\}_{i=1,\ldots,n_1}, ct'')$ is given to $\mathcal{A}$.

challenge: $\mathcal{A}$ requests a pair of messages $(\mathsf{msg}_0, \mathsf{msg}_1)$, the challenger selects a random bit b and runs the encrypt and sanitize algorithm to obtain a sanitized ciphertext ct'. Finally, $\mathcal{A}$ outputs a bit b'.

Lemma 1. *If Π_{ABE} is a fully secure attribute scheme, then the above construction is a secure attribute based access control encryption scheme.*

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ that can break the no-read rule experiment. Then we can construct an adversary $\mathcal{A}'$. $\mathcal{A}'$ is given the input 1^λ and access to all the query oracle. When $\mathcal{A}$ makes an encryption query on a message $\mathsf{msg} \in \mathsf{M}$, $\mathcal{A}'$ runs a encryption query and returns a ciphertext. When $\mathcal{A}$ queries its sanitize oracle on a ciphertext ct, $\mathcal{A}'$ runs a sanitize query and returns the sanitized ciphertext. When $\mathcal{A}$ makes a challenge query and outputs a bit b'. $\mathcal{A}'$ outputs 1 if $\mathcal{A}$ succeeds, and 0 otherwise. We assume all the query oracle is random oracle, the view of $\mathcal{A}$ when run as a subroutine of $\mathcal{A}'$ is distributed identically to the view of $\mathcal{A}$ in $\Pi^{read}_{\mathsf{AACE},\mathcal{A}}$. Thus

$$Pr|\Pi^{read}_{\mathsf{AACE},\mathcal{A}'} = 1| - Pr|\Pi^{read}_{\mathcal{A}} = 1| = negl(\lambda)$$

Since the security of our attribute based access control encryption can be reduced to the DLIN assumption, and the detailed proof can be found in [41]. That means $Pr|\Pi^{read}_{\mathcal{A}} = 1| = negl(\lambda)$ and thus $Pr|\Pi^{read}_{AACE,\mathcal{A}'} = 1| = negl(\lambda)$. We can conclude that our attribute based access control encryption satisfies the no-read rule.

Theorem 2. *Our attributes access control encryption satisfies the no-write rule if no adversary can efficiently break the experiment $\mathsf{Expt}^{\mathsf{write}}_{\Pi_{\mathsf{AACE}},\mathcal{A}}(\lambda, \mathsf{b})$ with a nonnegligible probability.*

Lemma 2. *If no efficient adversary can forge a valid signature with a nonnegligible probability, then our Construction satisfies the no-write rule.*

At the beginning, we will describe the experiment in the same way to simplify the proof.

key query: When receiving an identity id_i, $\mathcal{C}$ runs $\mathsf{Dual}(\cdot)$ to obtain two orthonormal bases $\boldsymbol{C}, \boldsymbol{C^*}$, and computes $pk = ([\alpha \boldsymbol{d}_1\boldsymbol{d}_1^*]_T, [\boldsymbol{d}_1]_1, [\boldsymbol{d}_2]_1), sk = (\alpha, [\boldsymbol{d}_1^*]_2, [\boldsymbol{d}_2^*]_2)$. Then it returns (pk, sk), and pk is given to $\mathcal{A}$.

signature query: When $\mathcal{A}$ makes a signature query, $\mathcal{C}$ picks $r \xleftarrow{R} Z_p$ and computes $\theta = [(\alpha + r \cdot H(msg))\boldsymbol{d}_1^* - r\boldsymbol{d}_2^*]_2$ as the signature.

verify: On inputting a key pair (pk, sk), a message $msg \in M$, and a signature θ, it outputs 1 if and only if $e([\boldsymbol{d}_1 + H(msg)\boldsymbol{d}_2]_1, \theta) = [\alpha \boldsymbol{d}_1\boldsymbol{d}_1^*]_T$.

challenge: $\mathcal{A}$ is given pk and the access to the signature query oracle. We use Q to denote the query set that $\mathcal{A}$ makes. Note that $\mathcal{A}$ can make as many queries as it wants. Finally, $\mathcal{A}$ outputs (msg, θ). $\mathcal{A}$ succeeds if and only if $\mathsf{verify}(\mathsf{msg}, \theta) = 1$ and $msg \notin Q$. In this case, the experimental output is defined to be 1.

Proof. It is clear that for any malicious encryptor, if he cannot forge a valid signature, the edge server would terminate the communication and drop the message. The messages will not be delivered to the receivers. So the no-write rule is reduced to the security of the signature. Next, we will provide the proof.

Lemma 3. *If Π is a secure signature scheme and H is a secure hash function, then the above construction is a secure signature scheme.*

Proof. Let Π' denote the above construction, and $\mathcal{A}'$ be an efficient adversary. Q denotes a set of queries $\mathcal{A}$ has made, whose entries are formed by (msg, θ), and let (msg', θ') denote the final output of $\mathcal{A}'$. We assume that $msg' \notin Q$. We define col to be the event that $H(msg) = H(msg')$. Then we have

$$\begin{aligned} &Pr|\Pi^{'(forge)}_{\mathcal{A}'} = 1| \\ &= Pr|\Pi^{'(forge)}_{\mathcal{A}'} = 1 \wedge col| + Pr|\Pi^{'(forge)}_{\mathcal{A}'} = 1 \wedge \overline{col}| \\ &\leq Pr|col| + Pr|\Pi^{'(forge)}_{\mathcal{A}'} = 1 \wedge \overline{col}| \end{aligned} \quad (3)$$

Subsequently, we show that both terms in the above equation are negligible to complete the proof. Intuitively, $Pr|col|$ is negligible by the collision resistant of H, and the second term is negligible. Firstly, we construct the following algorithm to find a collision in H.

- Run key query to get pk and give it to $\mathcal{A}'$.
- $\mathcal{A}'$ makes a signature query of $\mathsf{msg_i}$, the algorithm computes $\theta = [(\alpha + r \cdot H(msg_i))\boldsymbol{d}_1^* - r\boldsymbol{d}_2^*]_2$ and adds (msg_i, θ) to the query list Q. Finally θ is given to $\mathcal{A}'$.
- When $\mathcal{A}'$ outputs (msg', θ'), if there exists a message $msg_i \in Q$ that $H(msg') = H(msg_i)$, then the algorithm outputs (msg', msg_i).

Let us analyze the above algorithm. When running the above algorithm to get a signature, the view of $\mathcal{A}'$ is distributed identically to the view of $\mathcal{A}'$ in the experiment $\Pi^{'(\mathsf{forge})}_{\mathcal{A}'}$. Particularly, the signature given to $\mathcal{A}'$ in the above algorithm has the same distribution as the signature that $\mathcal{A}'$ obtained in the experiment $\Pi^{'(\mathsf{forge})}_{\mathcal{A}'}$. Thus, when the collision occurs, we have

$$Pr|Hash - col_H = 1| = Pr|col|.$$

Since H is a secure hash function, we can conclude that $Pr|col|$ is negligible. We then show that the second term is negligible. Let $\mathcal{A}$ be an adversary that attacks Π_{sig} in $\Pi^{'(\mathsf{forge})}_{\mathcal{A}'}$, the adversary $\mathcal{A}$ is given access to the signature query as the above algorithm. When $\mathcal{A}'$ makes a query of $\mathsf{msg_i}$, $\mathcal{A}$ computes $\widehat{msg_i} = H(msg_i)$ and requests a signature θ on $\widehat{msg_i}$. Finally, θ is given to $\mathcal{A}'$. When $\mathcal{A}'$ outputs (msg', θ'), $\mathcal{A}$ outputs $(H(\widehat{msg'}), \widehat{\theta'})$.

Consider the above experiment $\Pi^{(\mathsf{forge})}_{\mathcal{A}}$, the view of $\mathcal{A}'$ when run as a subroutine by $\mathcal{A}$ is distributed identically to the view in the experiment $\Pi^{'(\mathsf{forge})}_{\mathcal{A}'}$. Whenever both $\Pi^{'(\mathsf{forge})}_{\mathcal{A}'} = 1$ and the collision col does not occur, $\mathcal{A}$ outputs a valid forgery. That means,

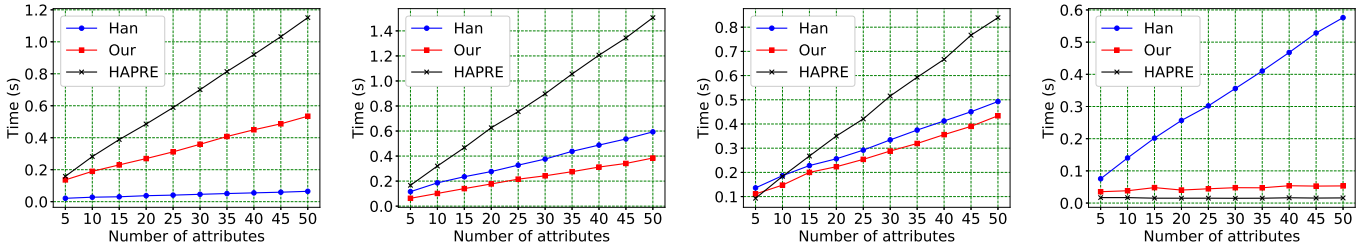
$$Pr|\Pi^{(\mathsf{forge})}_{\mathcal{A}} = 1| = Pr|\Pi^{'(\mathsf{forge})}_{\mathcal{A}'} \wedge \overline{col}|$$

Since Π is a secure signature scheme, the detail proof can be found in [42]. we can conclude that the former probability is negligible. This concludes the proof of the lemma.

TABLE 4
Comparisons of computation overhead

Schemes	Keygen	Encrypt	Sanitize	Decrypt
Han *et al.* [19]	$(T+2)e_1+2e_2$	$(2n_1+5)e_1+(n_1+5)e_2+2e_t+2p$	$(n_1+2)e_1+(n_1+1)e_2+14p$	$(I+3)p$
HAPRE [26]	$2e_1+(2T+6)e_2+e_t$	$(n_1+1)e_1+(4n_1)e_2+e_t$	$n_1e_t+(3n_1+1)p$	$3p$
Our scheme	$(9T+20)e_1+11e_2+e_t$	$6(n_1n_2+n_1)e_1+7e_2+2e_t$	$6(n_1n_2+n_1)e_1+3e_2+2e_t$	$6p$

* e, p indicate the exponentiation and pairing operations, the subscripts of them indicate which group they operate on. T denotes the number of attributes used in key generation, I denotes the number of attributes used in decryption.



(a) keygen (b) encrypt (c) sanitize (d) decrypt

Fig. 3. Computation overhead under the same settings.

TABLE 5
benchmark time of different operations (ms)

Groups	Multiplication	Exponentiation	Hash	Pairing
G_1	0.002	0.879	0.062	4.778
G_2	0.021	6.118	16.322	
G_T	0.007	1.598	-	

6 PERFORMANCE EVALUATION

In this section, the computation and communication overhead of the proposed scheme are evaluated and compared with other schemes [19], [26]. In Table 4, we provide a comparison of computation overhead to analyze the performance of these schemes in the key generation, encryption, sanitization and decryption. Moreover, a comparison of communication overhead is provided in Table 6 with regard to the size of the user key, the size of the ciphertext and the size of the sanitized ciphertext. They are the theoretical analysis of the proposed scheme and the comparative schemes, and they indicate the reasons for the difference between these schemes.

In order to verify the comparison in Table 4 and Table 6, we implement these schemes using the charm 0.50 framework in Python 3.6 on a laptop with Intel(R) Core(TM) i5-4210M CPU and 8 GB memory running Ubuntu 18.04. Particularly, since the symmetric bilinear pairings have serious security issues [43], we use the MNT224 curve for pairings. We use the access policy like 'A_1 and A_2 and $\cdots$ and A_n' which ensures that all the n attributes are involved in the decryption procedure. In our experiments, the running time is computed by calculating the average of running each procedure 10 times with the same input. The experimental results can be found in Fig. 3 and Fig. 4. In Table 5 we list the benchmark time (in millisecond) of different operations on MNT224 curve.

First, we analyze the computation overhead of these schemes. As shown in Table 5, the exponentiation and pairing operations are the most time-consuming. And the exponentiation operation on G_2 requires much longer time than the exponentiation operation on G_1 and G_T.

It can be seen from Table 4 that the scheme proposed by Han *et al.* [19] requires the least exponential operation on G_2. The proposed scheme requires more exponentiation operations on G_1. And HAPRE [26] requires the most exponentiation operations on G_2. Therefore, the experimental results depicted in Fig. 3(a) show that the scheme of Han *et al.* [19] performs best in key generation. The proposed scheme does not perform very well, but it is fully secure under standard assumption which achieves better security. In addition, it is admissible for a central authority with relatively large computation power.

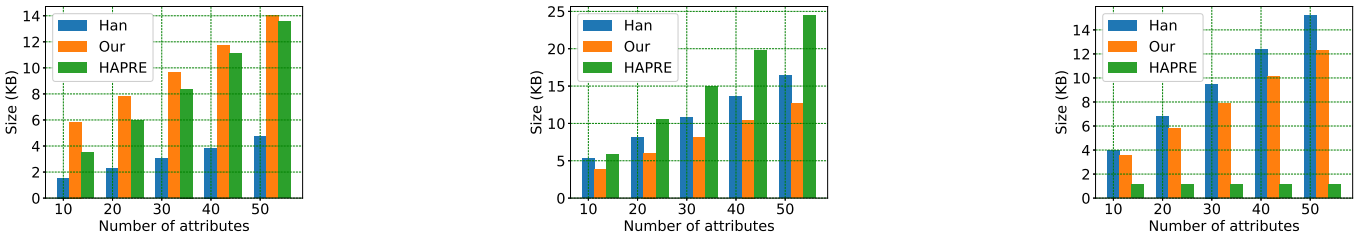
From Table 4 we can find that the proposed scheme requires the least exponentiation operations on G_2. However, HAPRE [26] requires the most exponentiation operations on G_2. Therefore, from the experimental results depicted in Fig. 3(b) we can see that the proposed scheme performs best in encryption. Since the sanitization of the proposed scheme and Han's scheme [19] are similar to the encryption, we will not analyze it separately. As HAPRE [26] is actually an outsourcing decryption scheme, the sanitization requires a large number of pairing operations to partially decrypt the ciphertext. The experimental results can be found in Fig. 3(c). We can see that the proposed scheme still has superior performance.

It can be seen from Table 4 that the decryption time is mainly related to the number of pairing operations. HAPRE [26] outsources a large number of pairing operations to the sanitization step, so it needs the least decryption time. The required pairing operation of decryption of Han *et al.* [19] is linear with the number of attributes used in the decryption, so it takes the most time. The pairing operation required in decryption of the proposed scheme is independent of the number of attributes, so it also has the better performance. The experimental results can be found in Fig. 3(d). We can see that the decryption of the proposed scheme only takes about 0.02s. It is almost as good as HAPRE [26] which uses outsourcing decryption. Besides, the computation overhead of the proposed scheme in setup takes about 0.02s and it

TABLE 6
Comparisons of communication overhead

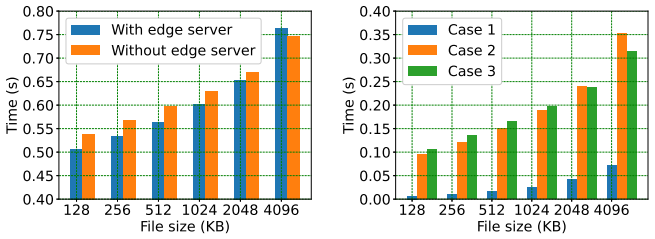
Schemes	User key size	Ciphertext size	Sanitized ciphertext size
Han *et al.* [19]	$(T+1)\tau G_1 + 2\tau G_2$	$(n_1+5)\tau G_1 + (n_1+5)\tau G_2 + 2\tau G_T$	$(n_1+2)\tau G_1 + (n_1+1)\tau G_2 + \tau G_T$
HAPRE [26]	$(T+1)\tau G_1 + (T+2)\tau G_2$	$(n_1+1)\tau G_1 + (2n_1+1)\tau G_2 + \tau G_T$	$2\tau G_1 + 2\tau G_2 + \tau G_T$
Our scheme	$3(T+1)\tau G_1 + 5\tau G_2$	$3n_1\tau G_1 + 4\tau G_2 + \tau G_T$	$3n_1\tau G_1 + 3\tau G_2 + \tau G_T$

* $\tau G_1, \tau G_2, \tau G_T$ are the sizes of elements in group G_1, G_2 and G_T, respectively. T is the number of attributes. n_1 is the rows of the access matrix $\mathcal{M}$. Note that the element on G_2 is 3 times the size of the element on G_1 in MNT224 curve.



(a) user key size (b) ciphertext size (c) sanitized ciphertext size

Fig. 4. Communication overhead under the same settings.



(a) upload file (b) download file

Fig. 5. Transmission overhead

takes only about 0.04s for the edge server to determine whether a sender can share files with the receiver, which are both very small constants.

Second, we analyze the communication overhead of these schemes. Before the analysis, we need to recall that the element on G_2 is 3 times the size of the element on G_1 in the MNT224 curve. From Table 6, we can see that the user key of Han *et al.* [19] has the fewest elements on G_2. The proposed scheme has the fewest elements on G_2 in the ciphertext. Furthermore, the sanitized ciphertext of HAPRE [26] is partially decrypted, so the sanitized ciphertext in their scheme is the smallest and does not change with the increase of the number of attributes. The experimental results can be found in Fig. 4. Note that the results are both evaluated by encrypting a random element in G_T. Because the sharing data usually is very large, the typical method is to use a KEM method wherein ABE is used to encrypt a random element in G_T and use the random element to generate a key K. Then the shared data is encrypted using the key K through an efficient symmetric encryption scheme. Thus the communication cost is reduced to the cost of transmitting an encryption of key K. Although the users' key size in our scheme is the largest in these schemes, in fact, even when the number of attributes is 50, the key size is only about 14 KB.

In order to evaluate the overhead in the actual transmission process, we used three cloud servers to simulate the local user, edge server and cloud server, respectively. The local user and the edge server are located in the same city, the cloud server is located in the other city. The bandwidth is set to be 50Mbps and the size of the test file is from 128 KB to 4096 KB. The results are generated by caculating the average of 100 statistics obtained every 5 minutes. It can be seen from Fig. 5(a) that the upload time that the file is sanitized by the edge server and then uploaded to the cloud server is basically the same as the time that the file is directly uploaded from the local to the cloud server for sanitization. In Fig. 5(b), Case 1 represents that the data sender and the data receiver are in the same area, and the data user can download the file directly from the edge server; In Case 2, the data sender and the data receiver are not in the same area. In this case, the data file can be obtained from the cloud server through the edge server close to the receiver; In Case 3, when the edge server is not used, users need to download files directly from the cloud server. The results show that using the edge server to forward the file will significantly reduce the time of downloading a data file in Case 1. And in Case 2, the transmission time is basically the same as the time of Case 3. Although the download process in Case 2 may increase the communication overhead, it is worth noting that in the attribute-based encryption, a group of users usually have the same decryption rights. Therefore, when the cloud server files are sent to the edge server, other users with the same attributes can get the data directly from the edge serve, thus reducing the transmission cost of data download. The above analysis of experimental results shows that we can use edge servers to process messages and forward them, leading to the lower communication delay. Therefore, our construction is efficient and practical.

7 CONCLUSION

In this paper, we proposed a novel practical attribute-based access control encryption scheme for cloud-edge data sharing, which not only enforces the access control of the encrypted data, but also restricts the information flow of the shared data. The scheme satisfied the no-read and no-write

rules, which means it is secure against malicious senders and non-privileged receivers. Then, we presented the operations of the cloud-edge data sharing system. Finally, we implemented the proposed construction to investigate its performance. The experimental results showed that the proposed scheme is efficient and practical.

ACKNOWLEDGMENTS

The work was supported by the National Natural Science Foundation of China (No. U1936220, No. 62011530046, No. 61872001), the Special Fund for Key Program of Science and Technology of Anhui Province, China (No. 202003A05020043), the Open Fund for Discipline Construction, Institute of Physical Science and Information Technology, Anhui University. The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

REFERENCES

[1] J. Wei, W. Liu, and X. Hu, "Secure data sharing in cloud computing using revocable-storage identity-based encryption," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2016.

[2] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2012, p. 965–976.

[3] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, pp. 222–233, 2014.

[4] W. Sun, S. Yu, W. Lou, Y. Thomas, and H. Li, "Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *IEEE INFOCOM 2014*, 2014.

[5] I. Damgård, H. Haagh, and C. Orlandi, "Access control encryption: Enforcing information flow with cryptography," in *Theory of Cryptography Conference*. Springer, 2016, pp. 547–576.

[6] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for cloudiot," *IEEE Transactions on Cloud Computing*, 2020.

[7] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Acm Conference on Computer & Communications Security*, 2007.

[8] S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2101–2113, 2018.

[9] D. Zheng, B. Qin, Y. Li, and A. Tian, "Cloud-assisted attribute-based data sharing with efficient user revocation in the internet of things," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 18–23, 2020.

[10] S. Kim and D. J. Wu, "Access control encryption for general policies from standard assumptions," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 471–501.

[11] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2005, pp. 457–473.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.

[13] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 1, pp. 131–143, 2012.

[14] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *IEEE Infocom*, 2015.

[15] J. Cui, H. Zhou, H. Zhong, and Y. Xu, "Akser: Attribute-based keyword search with efficient revocation in cloud computing," *Information Sciences*, pp. 343–352, 2017.

[16] Y. Zhang, R. H. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based encryption for cloud computing access control: A survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–41, 2020.

[17] W. Li, K. Xue, Y. Xue, and J. Hong, "Tmacs: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Transactions on parallel and distributed systems*, vol. 27, no. 5, pp. 1484–1496, 2015.

[18] J. Ning, X. Huang, W. Susilo, K. Liang, X. Liu, and Y. Zhang, "Dual access control for cloud-based data storage and sharing," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[19] J. Han, L. Chen, W. Susilo, X. Huang, A. Castiglione, and K. Liang, "Fine-grained information flow control using attributes," *Information Sciences*, vol. 484, pp. 167–182, 2019.

[20] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2062–2074, 2018.

[21] G. Fuchsbauer, R. Gay, L. Kowalczyk, and C. Orlandi, "Access control encryption for equality, comparison, and more," in *IACR International Workshop on Public Key Cryptography*. Springer, 2017, pp. 88–118.

[22] G. Tan, R. Zhang, H. Ma, and Y. Tao, "Access control encryption based on lwe," in *Proceedings of the 4th ACM International Workshop on ASIA Public-Key Cryptography*. ACM, 2017, pp. 43–50.

[23] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," *Lecture Notes in Computer Science*, vol. 1403, pp. 127–144, 1998.

[24] X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, 2009, pp. 276–286.

[25] Y. Yang, H. Zhu, H. Lu, J. Weng, Y. Zhang, and K.-K. R. Choo, "Cloud based data sharing with fine-grained proxy re-encryption," *Pervasive and Mobile Computing*, vol. 28, pp. 122–134, 2016.

[26] H. Deng, Z. Qin, Q. Wu, Z. Guan, and Y. Zhou, "Flexible attribute-based proxy re-encryption for efficient data sharing," *Information Sciences*, vol. 511, pp. 94–113, 2020.

[27] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM symposium on information, computer and communications security*, 2010, pp. 261–270.

[28] S. Xu, G. Yang, and Y. Mu, "Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation," *Information Sciences*, vol. 479, pp. 116–134, 2019.

[29] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[30] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2019.

[31] X. Xia, F. Chen, Q. He, J. C. Grundy, M. Abdelrazek, and H. Jin, "Cost-effective app data distribution in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 31–44, 2020.

[32] T. Wang, L. Qiu, A. K. Sangaiah, A. Liu, M. Z. A. Bhuiyan, and Y. Ma, "Edge-computing-based trustworthy data collection model in the internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4218–4227, 2020.

[33] L. U. Khan, I. Yaqoob, N. H. Tran, S. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge computing enabled smart cities: A comprehensive survey," *IEEE Internet of Things Journal*, 2020.

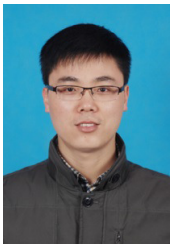
[34] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 80–85, 2017.

[35] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for iot-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[36] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Data processing and sharing for hybrid cloud-edge analytics," *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[37] F. A. Salaht, F. Desprez, and A. Lebre, "An overview of service placement problem in fog and edge computing," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–35, 2020.

[38] E. Ahmed, A. Ahmed, I. Yaqoob, J. Shuja, A. Gani, M. Imran, and M. Shoaib, "Bringing computation closer toward the user network: Is edge computing the solution?" *IEEE Communications Magazine*, vol. 55, no. 11, pp. 138–144, 2017.
[39] T. Okamoto and K. Takashima, "Hierarchical predicate encryption for inner-products," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2009, pp. 214–231.
[40] J. Chen, R. Gay, and H. Wee, "Improved dual system abe in prime-order groups via predicate encodings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 595–624.
[41] S. Agrawal and M. Chase, "Fame: fast attribute-based message encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 665–682.
[42] J. Chen, H. W. Lim, S. Ling, H. Wang, and H. Wee, "Shorter ibe and signatures via asymmetric pairings," in *International Conference on Pairing-Based Cryptography*. Springer, 2012, pp. 122–140.
[43] S. Galbraith, "New discrete logarithm records, and the death of type 1 pairings," 2014.



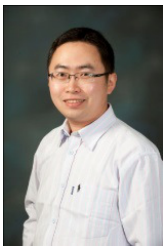
Jie Cui was born in Henan Province, China, in 1980. He received his Ph.D. degree in University of Science and Technology of China in 2012. He is currently a professor and Ph.D. supervisor of the School of Computer Science and Technology at Anhui University. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). He has over 100 scientific publications in reputable journals (e.g. IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, IEEE Journal on Selected Areas in Communications, IEEE Transactions on Computers, IEEE Transactions on Vehicular Technology, IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Network and Service Management, IEEE Transactions on Emerging Topics in Computing, IEEE Transactions on Cloud Computing and IEEE Transactions on Multimedia), academic books and international conferences.

Bei Li is now a research student in the School of Computer Science and Technology, Anhui University. His research focuses on cloud computing security and edge computing security.

Hong Zhong was born in Anhui Province, China, in 1965. She received her PhD degree in computer science from University of Science and Technology of China in 2005. She is currently a professor and Ph.D. supervisor of the School of Computer Science and Technology at Anhui University. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). She has over 150 scientific publications in reputable journals (e.g. IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Parallel and Distributed Systems, IEEE Journal on Selected Areas in Communications, IEEE Transactions on Multimedia, IEEE Transactions on Vehicular Technology, IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Network and Service Management, IEEE Transactions on Cloud Computing and IEEE Transactions on Big Data), academic books and international conferences.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems, Modelling and Performance Engineering.

Yan Xu is currently an associate professor of School of Computer Science and Technology at Anhui University. She received the BS and MS degrees from Shandong University in 2004 and 2007, respectively, and the PhD degree from University of Science and Technology of China in 2015. Her research interests include information security and applied cryptography.



Lu Liu is the Professor of Informatics and Head of Department of Informatics in the University of Leicester, UK. Prof Liu received the Ph.D. degree from University of Surrey, UK and MSc in Data Communication Systems from Brunel University, UK. Prof Liu's research interests are in areas of cloud computing, service computing, computer networks and peer-to-peer networking. He is a Fellow of British Computer Society (BCS).