IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INNOVATION AND APPLICATION OF INTELLIGENT PROCESSING OF DATA, INFORMATION AND KNOWLEDGE AS RESOURCES IN EDGE COMPUTING

Received February 27, 2019, accepted March 20, 2019, date of publication March 25, 2019, date of current version April 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907319

TMO: Time Domain Outsourcing Attribute-Based Encryption Scheme for Data Acquisition in Edge Computing

YOUHUIZI LI[1,2,3], ZEYONG DONG[1,2], KEWEI SHA[4], CONGFENG JIANG[1,2], JIAN WAN[2,5], AND YUAN WANG[6,7]

[1]School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
[2]Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou 310018, China
[3]Xi'an Key Laboratory of Mobile Edge Computing and Security, Xi'an 710071, China
[4]College of Science and Engineering, University of Houston–Clear Lake, Houston, TX 77058, USA
[5]School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China
[6]Engineering Research Center of Augmented Reality and Intelligent Interaction of Zhejiang Province, Hangzhou 310052, China
[7]NetEase (Hangzhou) Network Co., Ltd., Hangzhou 310052, China

Corresponding author: Jian Wan (wanjian@zust.edu.cn)

This work was supported in part by the Natural Science Foundation of Zhejiang Province under Grant LQ18F020003, in part by the Natural Science Foundation of China under Grant 61802093, Grant 61472109, and Grant 61572163, in part by the Xi'an Key Laboratory of Mobile Edge Computing and Security under Grant 201805052-ZD3CG36, and in part by the Key Research and Development Program of Zhejiang Province under Grant 2018C01098.

ABSTRACT With the rapid development of the Internet of Things and the ever-increasing demands of advanced services and applications, edge computing is proposed to move the computing and storage resources near the data source, which improves the response time and saves the bandwidth. However, due to the limited available resources and massive privacy-sensitive user data in edge nodes, there are huge challenges in data security and privacy protection in the edge computing environment. Hence, we propose an efficient time-domain multi-authority outsourcing attribute-based encryption (ABE) scheme (TMO) with a dynamic policy updating method for secure data acquisition and sharing in the edge computing. Specifically, considering that the time is a crucial factor in many real-world application scenarios, we add time-domain information in the encryption algorithm. Besides, to take full advantage of edge computing, TMO extends the multi-authority ABE approach by outsourcing the computation to edge nodes to enhance security and performance. Moreover, to tackle the mobility and frequently changing edge environment, TMO also provides an efficient online policy updating method to manage attribute information and to access policy with low overhead. The security analysis and the experimental results show that TMO can indeed efficiently enhance data security with low overhead in the edge computing environment.

INDEX TERMS Multi-authority, CP-ABE, time-based, security, edge computing.

I. INTRODUCTION

With the development of Internet of Things, we are surrounded by millions of smart devices which generate a huge amount of data. Gartner predicts that there will be 20.4 billion IoT devices by 2020 [1]. Besides, the increasing user demands accelerate the developing of advanced complex applications and services, such as augmented reality, virtual reality, intelligent transportation and smart city [2] as well as automated user friendly service reconfiguration [3].

To satisfy the requirements of the above applications and to improve the user experience, edge computing [4] is proposed as a new computing paradigm where data is processed at the nearest edge node with enough resources. Compared with cloud computing, the advantages of edge computing is two-fold. First, it leverages the available computing and storage resources on edge nodes which are close to users, so the response time can be improved. Second, since the massive data do not need to be transmitted to the cloud, the backbone bandwidth is also saved. However, the new computing paradigm also results in new research challenges. For example, the resource-limited edge nodes normally hold a lot

The associate editor coordinating the review of this manuscript and approving it for publication was Honghao Gao.

of privacy-sensitive user data, powerful security approaches used in cloud centers are not feasible in the edge computing environment [5]–[7]. Moreover, in order to adapt to the dynamic edge environment, applications and services are more complicated [8], [9]. The high mobility of users and devices in edge computing makes the security issue even more difficult [10], [11], users' privilege changes frequently and the attackers have more chances to join the group. Hence, we propose an efficient lightweight attribute-based encryption scheme in this paper to enhance the data security in the edge computing environment.

In many real-world applications, time is an important factor which defines the usefulness and effectiveness of data, especially in massive data situation [12]. For instance, when using video captured by road cameras to track a suspect, the data is more useful if it is close to the time that the crime happened. Time should also be considered in data acquisition and sharing. Take the fire disaster situation as an example, if the smart home device detects a fire in the house, it can temporarily share the private real-time video and smoke sensor data to the property managers and firefighters. After knowing the first-sight information, they can be better prepared and reduce disaster losses as much as possible. Besides, the high mobility of users and devices is also the nature of edge computing. With the changing of various factors, including location, time and user roles, we have to update the access policy frequently [13]. Considering the computational overhead and bandwidth costs of the update policy, it is necessary to provide an efficient and secure policy updating mechanism.

Data encryption is a commonly used solution to protect data security and privacy [14]. From the aspect of basic infrastructures, edge computing can better support multi-authority and outsourcing decryption, especially in attribute-based encryption schemes. In cloud computing, there is usually one single attribute authority. It is responsible for the authorization and distribution of all attributes, as well as the production of their corresponding key devices. There are several problems: firstly, it leads to a performance bottleneck. If a large number of users make a private key request to the authority at the same time, some users may get the private key much longer than the average time. An attacker even can occupy the key distribution service on purpose, other normal users will be greatly influenced. Secondly, there is also a security single point of failure. Since the authority knows all the private parameters, when it encounters a malicious attack, the intruder can make any key he wants. As a result, users' private data will be illegally exposed. To deal with this situation, multi-authority is proposed that one authority is only responsible for parts of the attributes [15]. The decentralized feature of edge computing perfectly supports the multi-authority scenario, and edge nodes can provide a more secure and flexible solution. Moreover, previous researches [16]–[18] also outsourced decryption to the cloud for resource-limited devices. Compared with the faraway cloud server, edge nodes, as the closest computing node, can perform the pre-decryption operation in a relatively short time. Besides, it can also reduce the result transmission distance and minimize the ciphertext exposure risk.

In this paper, we focus on how to efficiently and securely acquire/share data to a group of authorized users by satisfying various time restrictions in the edge computing environment. We propose TMO, a time domain multi-authority outsourcing attribute-based encryption scheme, to enhance the data security and protect user privacy. Specifically, TMO adds time as one of the encryption factors and splits attributes into universal attribute and time attribute, so that a flexible data access mechanism is supported. Besides, we extend the multi-authority attribute-based encryption approach in [16] with outsourcing the decryption to edge nodes, which can greatly reduce the cost. Moreover, TMO also provides an efficient access policy updating method. Instead of requiring the data owner to retrieve the ciphertext and re-encrypt it for distribution, edge storage nodes can update the access policy of the existing ciphertext online without wasting the network bandwidth. The contributions of this paper are summarized as follows:

- Based on the data and infrastructure characteristics, we propose TMO, an efficient and secure time domain multi-authority outsourcing attribute-based encryption scheme, to improve the data security and privacy in the edge computing environment.
- We develop a flexible update mechanism for time range updating and access attribute updating. Through modifying the time restriction and managing the access privilege, time attribute can be updated effectively. Besides, we designed a dynamic policy update method which leverages the data storage node to update the corresponding ciphertext online with minimum network overhead.
- We implement a prototype system and conduct a comprehensive performance evaluation. Compared with previous solutions, experimental results show that TMO can indeed improve both encryption and outsourcing decryption performance. With supporting time domain access control, TMO also provides acceptable performance in the key-generation stage.

The remainder of the paper is organized as follows. Related work is reviewed in Section II. Section III introduces the system architecture, definitions and security models. Section IV describes the proposed TMO scheme, its construction and security analysis. Policy update mechanism is presented in Section V, and the experiments and results are demonstrated in Section VI. Finally, we conclude this paper in Section VII.

II. RELATED WORK

Attribute-based encryption(ABE) is a widely used public-key encryption scheme which allows fine-grained access control to the encrypted data through the attributes management [19]. It is first proposed in a fuzzy identity-based encryption scheme [20]. According to the location of the attributes, ABE schemes can be categorized to Key-Policy Attribute-Based

Encryption(KP-ABE) [21] and Ciphertext-Policy Attribute-Based Encryption(CP-ABE) [22]. Due to its high flexibility and scalability, CP-ABE is employed in many scenarios. Different features are proposed to the classic CP-ABE scheme, such as constant size key ciphertext [23], attribute revocation [24], user anonymity [25], and so on. Two or three features are combined to provide a secure encryption method according to the specific requirements of the usage scenarios. Next, we briefly introduce the existing attribute-based encryption approaches from the aspects of authority, outsourcing and time domain encryption.

Authority: Attribute authority is a trusted role who is responsible for distributing and managing public and private keys. Based on the number of authority involved in the encryption system, ABE schemes are classified to single authority ABE [22] and multi-authority ABE [26]. In a single authority ABE, key structure is relatively simple and it is easy to manage existing keys. However, single authority is not very flexible when adding new attributes. Besides, the cooperation between different organization is inevitable in real scenarios, it is not feasible to manage the attributes definition and distribution across domains. Hence, multi-authority ABE is proposed and becomes more and more popular. For examples, Lewko and Waters [27] proposed a multi-authority ABE which does not require a global authority to coordinate decentralized attribute authorities. Follows that, Ruj *et al.* [25] proposed a decentralized access control scheme to support anonymous authentication; Belguith *et al.* [28] designed an encryption method that hides the access policy and outsources the decryption task to the cloud. Compared with [27] of which public parameters are composed of attributes, Yang *et al.* [16] added authority to the parameter group, the size of the public parameter is significantly reduced. In TMO, we further improve [16] with less computation and more features to make it more suitable in the edge computing environment.

Outsourcing: The computational complexity of most ABE schemes [27], [15], [29] increases linearly with the number of attributes involved in the ciphertext. In order to reduce the computing pressure for resource-limited devices, outsourcing decryption is proposed to offload the computation to the cloud or other proxy nodes. The first outsourced ABE was proposed by Green *et al.* [30], the cloud performs a pre-decryption calculation based on the transformable keys and returns a constant size ciphertext to the user. In the edge computing environment, the edge/fog nodewhich can provide computing and storage resources is more close to the terminal devices [31]. Hence, some recent works leverage the fog node to do the encryption/decryption calculation. Zhang *et al.* [32] proposed an access control scheme which outsources the encryption/decryption tasks to the fog node and efficiently updates the system attributes. Since the edge nodes are more vulnerable to attack than cloud, Zuo *et al.* [33] proposed a safer outsourcing decryption ABE method that can resist the chosen-ciphertext attacks (CCA). In our proposed TMO, outsourcing feature is also included to enhance the system performance and improve the user experience.

TABLE 1. Notations used in TMO.

Notation	Description
ECN	Edge Computing Node
ESN	Edge Storage Node
DO	Data Owner
DU	Data User
CA	Central Authority
AA	Attribute Authority
GP	Global Parameters
(sk_{CA}, vk_{CA})	A pair of CA signature and verification key
Apk_{AA_j}	Public key related to AA_j
Ask_{AA_j}	Secret key related to AA_j
$Cert_{uid}$	Certification of a user uid
CT_{FID}	Key Ciphertext numbered FID
k	A symmetric key
$[T_{begin}, T_{end}]$	Access time range
$TAAParam_{j,FID}$	Time parameters for AA_j
$TDOParam_{j,FID}$	Time parameters for DO
$S_{j,uid}$	A set of universal attributes belonging to a user uid received from AA_j
$Usk_{j,uid}$	The universal private keys belonging to a user uid received from AA_j
$ST_{j,uid}$	A set of time attributes belonging to a user uid received from AA_j
$UTsk_{j,uid,FID}$	The time private keys for CT_{FID} belonging to a user uid received from AA_j
$EK_{FID,uid}$	The edge key for CT_{FID} belonging to a user uid
ICT	An intermediate ciphertext
$LK_{FID,uid}$	The local key for CT_{FID} belonging to a user uid

Time domain encryption: With the development of the Internet of Things, the data generated daily is greatly increasing. Time is becoming an important factor that can decide the usefulness of the data. In order to share time-sensitive data, there are several ABE schemes that consider the time domain information. For examples, Liu *et al.* proposed a time-based proxy re-encryption scheme [34], which can control the user's access rights during a specified period. But with the changing of the time periods, it needs to re-encrypt the same data into different versions. Hong *et al.* [35] designed an access control scheme based on both time and attributes. If an attribute has a time restriction, the cloud obtains a token and updates the ciphertext online at the beginning of the time period. Without this operation, the unexposed ciphertext cannot be decrypted even if the user owns the private key of the corresponding attributes. Yang *et al.* [36] proposed a time domain multi-authority ABE method which embeds time into both ciphertext and secret key, so the encrypted data can only be decrypted when both access policy and access period are satisfied. In our TMO, the time domain information is also considered, furthermore, we integrate it with multi-authority and outsourcing to provide an efficient and secure ABE system for edge computing.

III. SYSTEM ARCHITECTURE AND DEFINITIONS

In this section, we first introduce the related preliminary knowledge, then present the system architecture, definitions and security models. Some frequently used notations are summarized in Table 1.

A. PRELIMINARIES

1) BILINEAR MAPS [22]

Let G and G_T be two multiplicative cyclic groups of prime order p, while g is the generator of G. Algorithm $e : G \times G \rightarrow G_T$ is a bilinear map that satisfies the following three properties.

- Bilinearity: if $\forall\, u, v \in G$ and $x, y \in Z_p$, then $e(u^x, v^y) = e(u, v)^{xy}$.
- Non-degeneracy: $e(g, g) \neq 1$.
- Computability: $\forall\, u, v \in G, e(u, v)$ is an admissible algorithm.

2) LINEAR SECRET SHARING SCHEMES(LSSS) [37]

A secret sharing scheme π that consists of a set of participants is linear if it meets the following conditions.

- The shared value of each participant is an element in Z_p. And these elements form a vector over Z_p.
- The matrix A consisting of l rows and n columns is the sharing-generating matrix for π. The map ρ which is responsible for mapping each row of the matrix to an associated participant (*i.e.,* for $i = 1, \ldots, l$, $\rho(i)$ is the participant with row i).
- Consider the column vector $\vec{v} = (s, r_2, \ldots, r_n)$, where $s \in Z_p$ is the secret to be shared, and $r_2, \ldots, r_n \in Z_p$ are randomly chosen. Then $A \cdot \vec{v} = \vec{\lambda}$ is the vector of l shares of the secret s according to π, the share value $A_i \cdot \vec{v}$ belongs to the participant $\rho(i)$.

If an authorized set SP that consists of certain participants. Let $I \subseteq \{1, 2, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in SE\}$, then there exists constants $\{c_i \in Z_p\}_{i \in I}$, such that $\sum_{i \in I} c_i \lambda_i = s$, which can recover the secret s. Meanwhile, all these constants $\{c_i\}$ can be found in polynomial time, but the secret s can't be recovered if SP is unauthorized.

B. SYSTEM ARCHITECTURE

The system architecture of TMO scheme is illustrated in Figure 1. It is a three-tier architecture: cloud - edge - user. Edge layer contains a lot of edge nodes with computing and storage resources, so data can be stored (on ESNs) and pre-processed (on ECNs) near the user. Besides, the distributed nature of edge nodes makes it perfectly to support multi-authority which enhances the system security. Hence, the performance and security of encryption schemes can be greatly improved by leveraging edge nodes. The specific roles of the six participants and their interactions on key/data distribution are described as follows.

Cloud: The Cloud provides the storage capability for end users and distributes the stored ciphertext to the edge storage node(ESN) when receiving an access request. All ciphertext in the system will be uploaded to the cloud for permanent storage. We assume that the cloud is honest-but-curious, it will correctly execute the requests but be curious about the stored content. Besides, in the policy updating situation, the cloud will not only perform the update algorithm on the original ciphertext but also distribute the update key to the corresponding ESNs.

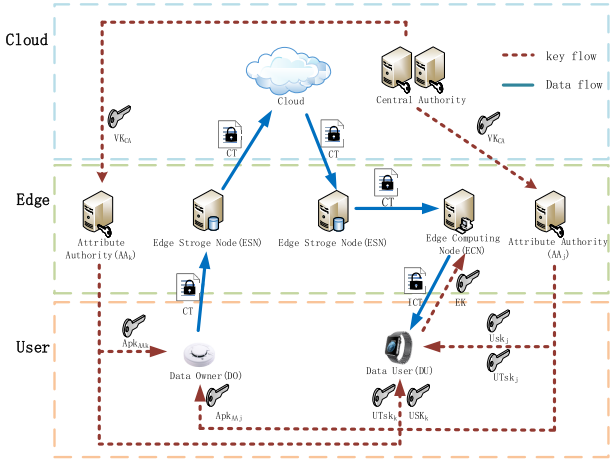


FIGURE 1. System architecture.

Edge Node: There are two types of edge nodes: Edge Storage Node (ESN) and Edge Computing Node (ECN). The ESN is the nearest storage node, it fills the gap between users and the cloud. On one hand, it receives the ciphertext from users and sends the data to the cloud for permanent storage based on the network situation. On the other hand, it also handles the access requests in real time, retrieves data from the cloud if necessary and sends the ciphertext to the authorized data users. The ECN is an edge node with more computing power. It is responsible for pre-decrypting the ciphertext that the user wants, but cannot get any content information. If an edge node has both powerful computing capability and large storage space, it can serve as both an ESN and an ECN.

Central Authority (CA): The CA is a trusted entity, which manages the settings of global parameters and the registration of users and authorized attribute authorities.

Attribute Authority (AA): The AA manages a group of attributes in its domain and creates corresponding private keys to users. In our scheme, there are two kinds of attributes: the universal attribute and the time attribute. For a universal attribute, its private key can be combined with other attribute private keys to access related ciphertext. For a time attribute, the AA will create a special private key with the time parameter embedded, and the time parameter was configured by data owner. The valid time range of multiple ciphertexts with the same time attribute may different, so a specific time attribute only corresponds to one ciphertext and cannot be shared among others. By comparing the valid accessing time range with the request time, the AA determines whether to make and distribute the private keys.

Data Owner(DO): The DO creates the ciphertext using the following two steps. First, the plain data is encrypted using symmetric encryption which is a widely used lightweight encryption method. Then, the symmetric key used in the first step is encrypted by our attribute-based encryption method to

enhance flexibility and security. The attribute-based encryption method requires a specific access policy, in which there may involve both universal attributes and time attributes. Noted that, in our TMO scheme, we provide the coarse-grained time domain restriction. If a set of time attributes in a ciphertext belong to the same AA, the time restrictions of these attributes should be the same. Because AA is responsible for the attributes in the same (or similar) professional field, the time restriction for the attributes in one ciphertext can be the same, and it also reduces the encryption complexity. After the encryption process is completed, the ciphertext and the encrypted key are transmitted to the nearest ESN, and the time parameters are sent to the corresponding AA for private key creation.

Data User (DU): Since the DU is normally a resource-constrained device, the ECN is used to improve the decryption process. The DU first initiates an access request to the closest ESN for the ciphertext and transforms its private key to the edge key. Then, the ESN forwards the ciphertext to the ECN, meanwhile, the DU also sends the edge key to the ECN. After performing the pre-decryption on the encrypted data, the ECN returns the intermediate ciphertext to the DU. Noted that only when the attribute set and access time of the DU satisfy the access policy, the symmetric key can be accurately decrypted. Finally, the DU can calculate the plaintext by itself.

C. DEFINITION

The TMO scheme mainly contains the following algorithms:

- $CASetUp\ (\lambda) \rightarrow (GP, (sk_{CA}, vk_{CA}))$. The CA setup algorithm takes an implicit security parameter λ as the input and outputs the global parameters GP and a pair of CA signature and verification key (sk_{CA}, vk_{CA}).
- $UserReg\ (GP, Info_{uid}) \rightarrow (uid, Cert_{uid}, K_{uid})$. The user registration algorithm is executed by the CA. The inputs are identity information $Info_{uid}$ submitted by the user and the global parameters GP. The CA authenticates the legality of the user's identity. If legal, it distributes a global unique identification number uid, an identity certificate $Cert_{uid}$ and an identity key K_{uid} to the user.
- $AAReg\ (GP, Info_{AID}) \rightarrow (AID, vk_{CA})$. The AA registration algorithm is also executed by the CA. With the input parameters of the authority information $Info_{AID}$ and the global parameters GP, the CA authenticates the legality of the AA. If legal, it distributes a global unique identification number AID and the verification key vk_{CA} to the AA.
- $AASetUp\ (GP, AID, S_{AA_j}) \rightarrow \left(Ask_{AA_j}, Apk_{AA_j}\right)$. The AA setup algorithm is executed independently by each $AA_j\ (j \in N)$, where N is the total number of AA in the system. Take the global parameters GP, the unique AID of the AA, and the corresponding attribute set S_{AA_j} as the inputs, this algorithm outputs the AA's private and public key pair $\left(Ask_{AA_j}, Apk_{AA_j}\right)$.
- $TimeParamGen\ (GP, FID, (T_{begin}, T_{end}), AID, ST_{j,FID}) \rightarrow (TAAParam_{j,FID}, TDOParam_{j,FID})$. The DO executes the time parameters generation algorithm to calculate the time parameter for AA ($TAAParam_{j,FID}$) and the embedded time parameter in encryption ($TDOParam_{j,FID}$). The inputs are the global parameters GP, a global unique ciphertext number FID, an access time range $[T_{begin}, T_{end}]$ and an attribute set $ST_{j,FID}$ that is composed of the time attributes that belongs to the authority AA_j, and used in the ciphertext FID. For the same set of time attributes with the same time range, the algorithm will still output different time parameters for different FID.
- $Encrypt\ \left(GP, k, (A, \rho), \left\{Apk_{AA_j}\right\}, \left\{TDOParam_{j,FID}\right\}\right) \rightarrow (CT_{FID})$. The encryption algorithm is executed by the DO. With the global parameters GP, the symmetric key k, an access policy (A, ρ), a set of public parameters of AAs $\left\{Apk_{AA_j}\right\}$ and a set of corresponding time parameters $\left\{TDOParam_{j,FID}\right\}$, this algorithm outputs a symmetric key ciphertext CT_{FID}.
- $UniversalKeyGen\ \left(GP, Ask_{AA_j}, S_{j,uid}, Cert_{uid}\right) \rightarrow \left(Usk_{j,uid}\right)$. The universal private key generation algorithm is executed by the authority AA_j. Take the global parameters GP, the private keys Ask_{AA_j}, the set of universal attributes $S_{j,uid}$ and the identity certificate $Cert_{uid}$ of DU as the inputs, the algorithm calculates a universal private key $Usk_{j,uid}$ for the user.
- $TimeKeyGen\ \left(GP, Ask_{AA_j}, FID, ST_{j,uid}, Cert_{uid}, TAAParam_{j,FID}\right) \rightarrow \left(UTsk_{j,uid,FID}\right)$. The time private key generation algorithm is also executed by the authority AA_j. Given the global parameters GP, the private keys Ask_{AA_j}, the FID of the ciphertext, the identity certificate $Cert_{uid}$ of DU, the set of the time attributes $ST_{j,uid}$ and the corresponding time parameter $TAAParam_{j,FID}$, the algorithm outputs the time private key $UTsk_{j,uid,FID}$ that can only be used to decrypt this specific ciphertext FID if the request time is within the access time range.
- $TransKeyGen\ \left(GP, \left\{Usk_{j,uid}\right\}, \left\{UTsk_{j,uid,FID}\right\}\right) \rightarrow \left(Ek_{uid,FID}, Lk_{uid,FID}\right)$. The transformable key generation algorithm is executed by the DU. On input the global parameters GP, the DU's universal private key set $\left\{Usk_{j,uid}\right\}$ and the time private key set $\left\{UTsk_{j,uid,FID}\right\}$ for the ciphertext FID, the algorithm outputs a pair of edge key $Ek_{uid,FID}$ for pre-decryption and local key $Lk_{uid,FID}$ for final decryption. If an attribute can be both universal and time, only its time private key is converted.
- $Decrypt.out\ \left(CT_{FID}, Ek_{uid,FID}\right) \rightarrow (ICT)$. The outsourcing decryption algorithm is executed by the ECN. With the symmetric key ciphertext CT_{FID} and the edge key $Ek_{uid,FID}$, the algorithm outputs an intermediate ciphertext ICT.
- $Decrypt.final\ \left(ICT, Lk_{uid,FID}\right) \rightarrow (k)$. The decryption algorithm is executed by the DU. Given the intermediate ciphertext ICT and the local key $Lk_{uid,FID}$, the algorithm

outputs the symmetric key k which is used to decrypt the data ciphertext CT_{data} to obtain the content that DU wants to access.

D. SECURITY MODEL

In our model, an adversary can query and get any universal key or time key, but these keys cannot be used directly to decrypt the challenge's ciphertext. Similar to [16], AAs can only be statically corrupted. Let S_{AA} denotes a set of all attribute authorities in the system, and $S'_{AA} \subset S_{AA}$ denotes a set of attribute authorities that are corrupted.

The security assumptions include:

- A user's own attribute keys cannot satisfy the access policy of the ciphertext, but he will cooperate with other users to obtain the private keys owned by the other party. The combined private keys can satisfy the access policy of the ciphertext, which can be used to access the content.
- Since AAs are at the edge level, they are more vulnerable than the cloud. They may be hacked to obtain the private keys of all the corresponding attributes.

The security game is defined as follows:

Setup: The challenger runs the *CASetUp* algorithm to generate the global parameters. The adversary specifies the set of AA that have been statically corrupted. Since *AASetUp* algorithm is performed by each AA, for the corrupted AAs, the adversary can obtain the public and private key; for the normal AAs, the adversary only gets the public key.

Phase 1: The adversary specifies a ciphertext number FID^* and an access policy (A^*, ρ^*) in which the set of time attributes and their corresponding time ranges also need to be determined. Then the adversary can make the following key query:

- **Universal key query:** The adversary submits the global uid, the corresponding identity certificate $Cert_{uid}$ and a set of universal attributes S_{uid} in which all the attributes belong to an uncorrupted AA. The challenger returns the universal private key $\{Usk_{j,uid}\}_{j \in S_{AA} - S'_{AA}}$ for the user uid to the adversary.
- **Time key query:** Similar to the universal key query, the adversary also submits uid, $Cert_{uid}$, an unique ciphertext FID, a time attribute set $ST_{j,uid}$ in which all the time attributes belong to an uncorrupted AA_j and an access time t to the challenger. If the time point t is valid, the challenger returns the time private key $\left(UTsk_{j,uid,FID}\right)_{j \in S_{AA} - S'_{AA}}$ to the adversary.
- **Transformable key query:** The adversary submits the universal private keys $Usk_{uid} = \left\{Usk_{j,uid}\right\}$ and the time private keys $UTsk_{uid,FID} = \left\{UTsk_{j,uid,FID}\right\}$ to the challenger, and the challenger executes the *TransKeyGen* algorithm to calculate the edge key $Ek_{uid,FID}$ and the local key $Lk_{uid,FID}$ which will then be sent to the adversary.

Challenge: The adversary submits two equal length symmetric keys k_0, k_1 to the challenger. Let S_{co} represents the set of any universal or time private keys that belong to corrupted AAs. For each uid, The game requires $Usk_{uid} \bigcup UTsk_{uid,FID} \bigcup S_{co}$ can't both match the access policy (A^*, ρ^*) and the ciphertext number FID^*. The challenger randomly selects a $\upsilon \in \{0, 1\}$ and encrypts k_υ under the access policy (A^*, ρ^*), then the ciphertext CT_{FID^*} will be sent to the adversary.

Phase 2: The adversary can repeat the query of phase 1. But there is a restriction, the new $Usk_{uid} \bigcup UTsk_{uid,FID}$ must still not satisfy the requirements of the ciphertext number FID^* and the universal and time attributes in the access policy.

Guess: The adversary outputs a guess υ'. If $\upsilon' = \upsilon$, the adversary wins the security game. The advantage of the adversary in the game is defined as follows:

$$Adv = \Pr\left[\upsilon' = \upsilon\right] - \frac{1}{2}$$

Theorem 1: The TMO scheme is secure against the static corruption of AAs, if the adversary has at most a negligible advantage in the above game in polynomial time.

Theorem 2: The TMO scheme is collusion resilience if there is no polynomial time solution that adversary can decrypt the symmetric key ciphertext by combining the private keys from other users who can't decrypt the data by his own.

IV. TIME DOMAIN OUTSOURCING MULTI-AUTHORITY ATTRIBUTE-BASED ENCRYPTION SCHEME

A. TMO OVERVIEW

We propose TMO, a time domain outsourcing multi-authority attribute-based encryption scheme in the edge computing environment. TMO provides secure and efficient fine-grained access control for data acquisition and data sharing. With the ever-increasing data and advanced applications, time is a deterministic factor for data usage. Hence, we embed time domain information into the encryption scheme to ensure the security and system flexibility. Besides, TMO also leverages the benefits of edge computing to support multi-authority and outsourcing features to improve efficiency. Outsourcing to edge nodes also reduces the security risks introduced by the key ciphertext transmission between the cloud and the terminal device. Specifically, plain data is split into data blocks and they are encrypted locally using symmetric encryption. Then, the encryption key is encrypted with the multi-authority attribute-based encryption scheme. The two types of ciphertext are sent to ESN for storage. The attributes in TMO consist of universal attributes and time attributes. Legal data users have correct attributes private keys and their accessing time is also in the valid time range. Then, data users can leverage ECNs to pre-decrypt the ciphertext by sending the corresponding transformed edge key. Finally, users can obtain the data content after performing the lightweight decryption locally.

B. CONSTRUCTION OF TMO

Based on the previous defined system models and algorithms, TMO consists of the following stages:

- **Central Authority Setup:** Let I_{AA} denote the set of attribute authorities in the system. With the security parameter λ, the CA runs the *CASetUp* algorithm which selects two multiplicative cyclic groups G and G_T with the same prime order P, a symmetric bilinear map $\hat{e}: G \times G \rightarrow G_T$, a generator g and a random element $h \in G$ and an anti-collision hash function $F: \{0,1\}^* \rightarrow G$ (mapping attributes to elements in G) to build the global parameters GP ($GP = (g, h, G, G_T, F)$). Besides, a pair of signature and verification key (sk_{CA}, vk_{CA}) that are used to sign and recover the user's credentials are also generated.
- **User registration:** When a user joins the system, he needs to register with the CA. The CA verifies the legality of the user's identity by analyzing $Info_{uid}$. If legal, the CA runs the *UserReg* algorithm to complete the registration for the user. A global unique *uid* is assigned to the user. Besides, the algorithm also randomly chooses an element $u_{uid} \in Z_p$ and signs it with the signature key sk_{CA} to build the user's credential $Cert_{uid}$. Then, the $Cert_{uid}$ and the user's identity key K_{uid} ($K_{uid} = g^{u_{uid}}$) are sent to the user.
- **Attribute authority registration:** Each $AA \in I_{AA}$ needs to register with the CA when joining the system. The CA verifies the legality of AA by analyzing the corresponding $Info_{AID}$. If legal, the CA runs the *AAReg* algorithm to complete the registration for the AA. The algorithm assigns a global unique *AID* and sends the verification key vk_{CA} that can recover the user's signature credentials to the AA.
- **Attribute authority setup:** Each authority needs to run the *AASetUp* algorithm at the initial stage. The algorithm randomly selects two elements $\alpha_j, \beta_j \in Z_p$ as the AA's private keys. And the corresponding public key Apk_{AA_j}, which is calculated as $Apk_{AA_j} = (e(g,g)^{\alpha_j}, g^{\beta_j})$ is published on the system's public bulletin board and open for all the users.
- **Encrypt time parameters generation:** For a ciphertext with a time restriction, the DO will first execute the *TimeParamGen* algorithm to calculate the time parameters of the time attributes before encrypting the symmetric key. Considering the efficiency and simplicity, we assume that the time restrictions of all the time attributes that belong to the same authority in an access policy should be the same, while attributes belonging to different AAs can be configured differently. Let $ST_{j,FID}$ represents the set of time attributes in the ciphertext *FID* that belong to the authority AA_j, and the valid time range is defined as $[T_{begin}, T_{end}]$. The algorithm randomly selects an element $r_j \in Z_p$ as the time parameters $TDOParam_{j,FID}$, which is embedded into the key ciphertext. Besides, it calculates another time parameters $TAAParam_{j,FID}$ for AA_j as $TAAParam_{j,FID} = (T_{begin}, T_{end}, ST_{j,FID}, g^{r_j})$, which is used to generate attribute private key in decryption phase.
- **Symmetric key ciphertext encryption:** To take advantages of both the efficiency of symmetric encryption and the security of public key encryption, TMO employs the following two-stage encryption: 1) Selecting a symmetric key k to encrypt data block M as $CT_{data} = Enc(M, k)$; 2) Executing the *Encrypt* algorithm to encrypt the symmetric key k. Before performing public key encryption on k, we need to specify the access policy (A, ρ), where A is the matrix of $l \times n$ and each row corresponds to an attribute. The l denotes the number of attributes contained in the access policy, the function ρ maps the matrix data to the corresponding attributes. The algorithm randomly selects an element $s \in Z_p$ as the secret value to be shared, and then uses a random vector $\vec{v} = (s, y_2, \ldots, y_n) \in Z_p$ to divide the secret value s. For $\forall x \in [1, l]$, the public shared value of s is λ_x, which is calculated as $\lambda_x = \vec{v} \cdot A_x$, and A_x is the x-th row in the matrix A. The elements $t_1, t_2, \ldots, t_l \in Z_p$ are also randomly selected, and the symmetric key ciphertext CT_{FID} is calculated as:

$$
\begin{aligned}
&C_0 = k \cdot e(g,g)^{s \cdot \prod \alpha_j}, \quad C_1 = g^s, \\
&\forall x \in [1, l], \quad C_{2,x} = h^{\lambda_x} \cdot g^{\beta_j t_x}, \; C_{3,x} = g^{-t_x}, \\
&C_{4,x} = \begin{cases} \left(F(\rho(x)) \cdot g^{\beta_j}\right)^{t_x} & \text{if } \rho(x) \text{ is universal} \\ \left(F(\rho(x)) \cdot g^{\beta_j r_j}\right)^{t_x} & \text{if } \rho(x) \text{ is time} \end{cases} \\
&CT_{FID} = (C_0, C_1, \{C_{2,x}, C_{3,x}, C_{4,x}\})
\end{aligned}
$$

- **Universal attribute private key generation:** Let $S_{j,uid}$ represents the set of universal attributes of which keys that the user needs to request in the authority AA_j. Noted that the universal key can be used to decrypt all the ciphertexts that contain the corresponding attributes. When AA receives the universal key request, it will run the *UniversalKeyGen* algorithm to generate the universal attribute private key. First, it verifies the user's identity by using the vk_{CA} to recover the u_{uid} from the certificate $Cert_{uid}$. Then, it selects a random element $z_j \in Z_p$ to calculate the universal private as:

$$
\begin{aligned}
Usk_{j,uid} &= (D_{1,j} = g^{\alpha_j} \cdot h^{u_{uid}}, D_{2,j} = g^{z_j}, \\
&\forall x_j \in S_{j,uid} : D_{j,x_j} = F(x_j)^{z_j} \cdot g^{\beta_j (z_j + u_{uid})})
\end{aligned}
$$

- **Time attribute private key generation:** If a ciphertext has access time restrictions, the DU needs to request time attribute private key from the corresponding authority in a valid time range and then combines it with the universal private key to decrypt the data correctly. To efficiently check the time information, the authority AA_j sets up a table to store the time parameters $(FID, TAAParam_{j,FID})$ of its ciphertexts. By comparing the accessing time with the valid time range retrieved via the ciphertext *FID*, the authority AA_j determines whether the user's key request is legal. If legal, it executes the

TimeKeyGen algorithm to generate the time private key. Let $ST_{j,uid}$ denotes the set of the corresponding time attributes, same as universal attribute private key generation, after verifying the user's identity, the algorithm randomly chooses an element $z_j' \in Z_p$ to calculate the time attribute key as:

$$UTsk_{j,uid,FID} = \left(D_{2,j}' = g^{z_j'},\right.$$
$$\left.\forall x_j \in ST_{j,uid} : D_{j,x_j}' = F\left(x_j\right)^{z_j'} \cdot \left(g^{r_j}\right)^{\beta_j z_j'} \cdot g^{u_{uid}\beta_j}\right)$$

- **Private key transformation:** After obtaining the universal private keys $\left\{Usk_{j,uid}\right\}$ and the time private keys $\left\{UTsk_{j,uid,FID}\right\}$, the DU performs the *TransKeyGen* algorithm to generate an edge key for pre-decryption on ECN and a local key for final local decryption. The algorithm randomly selects an element $q \in Z_p$, and then the local key $Lk_{uid,FID} = q$ and the edge key $Ek_{uid,FID}$ is

$$Ek_{uid,FID} = \left(K_{uid}' = K_{uid}{}^{\frac{1}{q}}, F_{1,j} = D_1^{\frac{1}{q}},\right.$$
$$\left.\begin{cases} x_j \in S_{j,uid} : F_{2,j} = \left(D_{2,j}\right)^{\frac{1}{q}}, F_{j,x_j} = \left(D_{j,x_j}\right)^{\frac{1}{q}} \\ x_j \in ST_{j,uid} : F_{2,j} = \left(D_{2,j}'\right)^{\frac{1}{q}}, F_{j,x_j} = \left(D_{j,x_j}'\right)^{\frac{1}{q}} \end{cases}\right)$$

- **Outsourcing decryption:** The ECN performs a pre-decryption operation without acquiring any information about the encrypted content. Let SE_{att} denotes the set of all the attributes contained in the edge key $Ek_{uid,FID}$. I_{att} denotes the set of the row indexes (of access matrix A) that correspond to the attributes in the edge key, $I_{att} = \{x : \rho(x) \in SE_{att}\}$. Moreover, I_{att} can be further divided according to the authority, that is $I_{att} = \left\{I_{AA_j}\right\}_{j \in I_{AA}}$. Let $N_{AA} = |I_{AA}|$ represents the number of AAs involved in the access policy. There exists a constant set $\left\{c_x \in Z_p\right\}_{x \in I_{att}}$ that can be found in a polynomial time, so that $s = \sum_{x \in I_{att}} c_x \lambda_x$, where λ_x is the public shared value of the attribute corresponding to the secret value s on the x-th row in the access matrix. The ECN runs the *Decrypt.out* algorithm to calculate the intermediate ciphertext *ICT*. The concrete calculation process of pre-decryption is as follows:

$$\mathcal{R}_x = e\left(C_{2,x}, K_{uid}'\right) \cdot e\left(C_{3,x}, F_{j,\rho(x)}\right) \cdot e\left(C_{4,x}, F_{2,j}\right)$$
$$\times \text{ if } \rho(x) \text{ is a universal attribute}$$
$$\mathcal{R}_x = e\left(g^{-t_x}, \left(F\left(\rho(x)\right)^{z_j} \cdot g^{\beta_j(z_j+u_{uid})}\right)^{\frac{1}{q}}\right)$$
$$\cdot e\left(h^{\lambda_x} g^{\beta_j t_x}, \left(g^{u_{uid}}\right)^{\frac{1}{q}}\right)$$
$$\cdot e\left(\left(F\left(\rho(x)\right) \cdot g^{\beta_j}\right)^{t_x}, \left(g^{z_j}\right)^{\frac{1}{q}}\right)$$
$$= e\left(h^{\lambda_x}, \left(g^{u_{uid}}\right)^{\frac{1}{q}}\right) = e(h, g)^{\frac{\lambda_x u_{uid}}{q}}$$
$$\times \text{ if } \rho(x) \text{ is a time attribute}$$
$$\mathcal{R}_x = e\left(g^{-t_x}, \left(F\left(\rho(x)\right)^{z_j'} \cdot \left(g^{r_j}\right)^{\beta_j z_j'} \cdot g^{u_{uid}\beta_j}\right)^{\frac{1}{q}}\right)$$
$$\cdot e\left(h^{\lambda_x} g^{\beta_j t_x}, \left(g^{u_{uid}}\right)^{\frac{1}{q}}\right)$$
$$\cdot e\left(\left(F\left(\rho(x)\right)\right) g^{\beta_j r_j}\right)^{t_x}, \left(g^{z_j'}\right)^{\frac{1}{q}}\right)$$
$$= e\left(h^{\lambda_x}, \left(g^{u_{uid}}\right)^{\frac{1}{q}}\right) = e(h, g)^{\frac{\lambda_x u_{uid}}{q}}$$
$$\mathcal{T} = \prod_{j \in I_{AA}} \frac{e\left(C_1, F_{1,j}\right)}{\prod_{x \in I_{AA_j}} \left(\mathcal{R}_x\right)^{c_x N_{AA}}}$$
$$= \prod_{j \in I_{AA}} \frac{e\left(g^s, \left(g^{\alpha_j} \cdot h^{u_{uid}}\right)^{\frac{1}{q}}\right)}{\prod_{x \in I_{AA_j}} \left(e(g,h)^{\frac{\lambda_x u_{uid}}{q}}\right)^{c_x N_{AA}}}$$
$$= \frac{e(g,h)^{\frac{s u_{uid} N_{AA}}{q}} \cdot e(g,g)^{\frac{s \prod_{j \in I_{AA}} \alpha_j}{q}}}{e(g,h)^{\frac{u_{uid} N_{AA} \sum_{x \in I_{att}} c_x \lambda_x}{q}}}$$
$$= e(g,g)^{\frac{s \prod_{j \in I_{AA}} \alpha_j}{q}}$$
$$ICT = (\mathcal{T}, C_0).$$

- **Local decryption:** After obtaining the intermediate ciphertext *ICT*, the DU runs the *Decrypt.final* algorithm with the local key $Lk_{uid,FID}$ to generate the symmetric key k. The concrete calculation process of k is:

$$\frac{C_0}{\mathcal{T}^q} = \frac{k \cdot e(g,g)^{s \cdot \prod \alpha_j}}{\left(e(g,g)^{\frac{s \prod_{j \in I_{AA}} \alpha_j}{q}}\right)^q} = k$$

Then, the DU can take the symmetric key k to decrypt the data ciphertext CT_{data} and obtain the plain data M as $M = Dec\left(CT_{data}, k\right)$.

C. SECURITY ANALYSIS

Based on the security game proposed in Section III, we prove that the proposed TMO is secure and can resist collusion attacks.

Theorem 1: If there doesn't exist an adversary A can win the security game (the size of challenge matrix M^* is $l^* \times n^*$ $(n^* < q)$) with non-negligible advantage $\varepsilon > 0$ in polynomial time, then there isn't a polynomial time simulator that can selectively break the decisional q-parallel BDHE assumption.

Proof: Assuming that in our security game, the adversary A with non-negligible advantage ϵ can only select an access matrix M^* with a maximum number of $q - 1$ columns. However, A can perform an arbitrary universal private key query, time private key query and transformable key query. One restriction exists, that is, all the obtained keys (even combined with the keys in the corrupted AAs) still cannot satisfy the attributes requirements in the access policy. Under this restriction, the security game of the multi-authority system is equivalent to that of the single-authority system. (Detailed proof is presented in Appendix.)

Theorem 2: TMO scheme is still secure under collusion attacks.

Proof: Every user has a globally unique *uid*, and the user identity parameter u_{uid} is embedded in each attribute key (no matter it is a universal attribute key or a time attribute key). Without loss of generality, assuming there are two malicious users in a collusion attack, then the decryption key should contain two u_{uid}. In this situation, one of the linear pairing calculations $e\left(h^{\lambda_x} \cdot g^{\beta_j t_x}, g^{u_{uid}}\right)^{c_x}$ in the decryption phase will not correctly recover the component $e(h, g)^{su_{uid}}$ which contains the secret value s. Hence, when there are multiple users' private key, even if the attribute set satisfys the access policy requirements, the ciphertext still cannot be successfully decrypted.

V. POLICY UPDATE

Due to the high mobility of the users and devices in the edge computing environment, the access policy will be changed frequently. Based on the usage scenario, we proposed the following two efficient policy update approaches: Time Range Update and Access Policy Update.

A. TIME RANGE UPDATE

For time attributes, the valid access time is defined by the time range embedded in the ciphertext. When the effective time period expires, the time range should be quickly updated to prevent encrypted data from being exposed to users whose access rights have expired. Let $ST_{j,FID}$ denote the set of time attributes received from AA_j in the ciphertext *FID*. If the valid time expires, the component of the ciphertext which corresponds to attribute $x \in ST_{j,FID}$ will need to be updated. If the DO want to extend the access time, a new time range $[t_a, t_b]$ $(t_a \leq t_b)$ should be given, otherwise, they can set an unreasonable time range $[t_a, t_b]$ $(t_a > t_b)$ to refuse the request. Both situations require a new random time parameter $r'_j \in Z_P$, and the ciphertext update key is calculated as

$$UK_j = \Big(K_{1,j} = g^{r'_j}, \quad \forall \rho'(x) \in ST_{j,FID} : K_{j,x} = \left(g^{\beta_j}\right)^{t_x r'_j - t_x r_j}\Big)$$

Next,the DO send the new time parameter tuple $\left(FID, [t_a, t_b], K_{1,j}\right)$ to the corresponding AA_j which is responsible for generating new time private key for decryption later. The update keys $\left\{K_{j,x}\right\}$ are also sent to the cloud, and it will distribute the keys to the ESNs that store the original ciphertext. Then, they will perform the following update algorithm to calculate the new ciphertext:

$$C'_{4,x} = C_{4,x} \cdot K_{j,x} = F\left(\rho'(x)\right)^{t_x} \cdot g^{\beta_j r'_j t_x} \quad \text{where} \quad \forall \rho'(x) \in ST_{j,FID}$$

B. ACCESS POLICY UPDATE

In the proposed TMO scheme, the access policy is represented as a combination of an access matrix and a row mapping function (A, ρ) in a LSSS structure. Since the updating of the accessible attributes, a new access policy $\left(A', \rho'\right)$ should be applied in the system. In order to save the communication costs and the computing overhead, we extend the *PolicyCompare* algorithm proposed in [13] to TMO scheme and only modify the public shared value of the s to build the new ciphertext. Specifically, after knowing the difference between the original and new access policies, the DO executes the *CTUKGen* algorithm to generate the update key. Then, same with the *Time Range Update*, the cloud will distribute the update key to all ESNs. The *CTUpdate* algorithm will be performed to calculate the new ciphertext on all storage nodes.

Policy Compare: To compare the difference between the original and new access policies, we define the algorithm as:

$$PolicyCompare\left((A, \rho), \left(A', \rho'\right)\right) \rightarrow \left(\left(S_{1,A'}, S_{2,A'}, S_{3,A'}\right), \left(AS_{1,A'}, AS_{2,A'}\right)\right)$$

Take the original and new access policies as inputs, the algorithm divides the attributes of new access matrix A' into three sets $S_{1,A'}, S_{2,A'}, S_{3,A'}$ and generates two AA sets $AS_{1,A'}, AS_{2,A'}$.

From the access attribute aspect, the difference of access matrix can be classified into the following three cases:

- **Case 1 (Set $S_{1,A'}$):** The original matrix A has the same attribute *att* as the new matrix A', and the number of attributes in A' is less than or equal than the one in A.
- **Case 2 (Set $S_{2,A'}$):** The original matrix A has the same attribute *att* as the new matrix A', and the number of attributes in A' is more than the one in A.
- **Case 3 (Set $S_{3,A'}$):** The new attributes that appear in A' but don't exist in A.

From the attribute authority aspect, the difference of access matrix can be classified as $AS_{1,A'}$, which indicates the AAs that are included in both the original and new access policies, and $AS_{2,A'}$, which indicates the new AAs that only appear in the new access policy.

Update key generation: The ciphertext update key generation algorithm is defined as:

$$CTUKGen\left(Param_{CT_{FID}}, \left\{Apk_{AA_j}\right\}, (A, \rho), \left(A', \rho'\right)\right) \rightarrow \left(\left\{UK_{x,i}\right\}_{x \in [1,l']}, UK_{C_0}\right)$$

With the encrypted parameters of the key ciphertext, the public key of each involved AA, the original and new access policies, the algorithm outputs update keys.

Due to the different number of the rows in the access matrixes A' and A, the data owners need to reassign the shared values of the secret value s. Thus a random vector $\vec{y}' = \left(s, y'_2, \ldots, y'_{n'}\right) \in Z_p^{n'}$ is been selected to generate the new shared values λ'_k $\left(k \in \left[1, l'\right]\right)$ as: $\lambda'_k = A'_k \cdot \vec{y}'$. Let $T(att)$ be the function that maps the attribute *att* to its corresponding AA. The updated key for each $x \in \left[1, l'\right]$ is calculated based on its classification in the *PolicyCompare* phase. In detail:

- Case 1: $(x, i) \in S_{1,A'}$, Let $t'_x = t_i$ and part of the update keys is $K_{1,x} = h^{\lambda'_x - \lambda_i}$. Based on the states of the same

attribute in the original and new matrixes, the update keys can be calculated as:

1) $\rho(i)$ and $\rho'(x)$ are the same type: $UK_{x,i} = K_{1,x}$.
2) $\rho(i)$ and $\rho'(x)$ are different types:

$$UK_{x,i} = \left(K_{1,x}, K_{2,x}\right)$$

a) $\rho(i)$ is a time attribute, $\rho'(x)$ is a universal attribute:

$$K_{2,x} = \left(g^{\beta_{T(\rho(i))}}\right)^{t'_x - r_{T(\rho(i))} t_i}$$

where $r_{T(\rho(i))}$ is the time parameter of the time attributes received from the authority AA_j in the ciphertext.

b) $\rho(i)$ is a universal attribute, $\rho'(x)$ is a time attribute: If the time parameter of this ciphertext for the authority has not been determined, the DO needs to randomly select a time parameter $r'_{T(\rho'(x))} \in Z_p$ and set a new range of access time $\left[T_{begin}, T_{end}\right]$. The tuple of the time parameters $\left(T_{begin}, T_{end}, g^{r'_{T(\rho'(x))}}\right)$ is then sent to the corresponding authority $T\left(\rho'(x)\right)$.

$$K_{2,x} = \left(g^{\beta_{T(\rho(i))}}\right)^{t'_x r'_{T(\rho'(x))} - t_i}$$

- Case 2: $(x, i) \in S_{2,A'}$, the DO first needs to randomly select $v_x \in Z_p$ to re-randomize the original component and let $t'_x = v_x \cdot t_i$. Then, based on the states of the same attribute, part of the update keys is same $\left(K_{1,x} = h^{\lambda'_x - v_x \lambda_i}\right)$ and the remaining update keys can be calculated as:

1) $\rho(i)$ and $\rho'(x)$ are the same type:

$$UK_{x,i} = \left(K_{1,x}, v_x\right)$$

2) $\rho(i)$ and $\rho'(x)$ are different types:

$$UK_{x,i} = \left(K_{1,x}, v_x, K_{2,x}\right)$$

a) $\rho(i)$ is a time attribute, $\rho'(x)$ is a universal attribute:

$$K_{2,x} = \left(g^{\beta_{T(\rho(i))}}\right)^{t'_x - r_{T(\rho(i))} v_x t_i}$$

b) $\rho(i)$ is a universal attribute, $\rho'(x)$ is a time attribute: If the time parameter of this ciphertext for the authority has not been determined, then DO repeats the same operations as in Case 1. The $K_{2,x}$ is different and the update key is

$$K_{2,x} = \left(g^{\beta_{T(\rho(i))}}\right)^{t'_x r'_{T(\rho'(x))} - v_x t_i}$$

- Case 3: $(x, i) \in S_{3,A'}$, the DO first needs to randomly select an element $t'_x \in Z_p$, and part of the keys are the same

$$UK_{x,i} = \left(K_{1,x}, K_{2,x}, K_{3,x}\right)$$
$$K_{1,x} = h^{\lambda'_x}\left(g^{\beta_{T(\rho'(x))}}\right)^{t'_x}, K_{2,x} = g^{-t'_x}$$

Only new attributes are added, so there are only two types of the update key:

1) $\rho'(x)$ is a universal attribute:

$$K_{3,x} = F\left(\rho'(x)\right)^{t'_x}\left(g^{\beta_{T(\rho'(x))}}\right)^{t'_x}$$

2) $\rho'(x)$ is a time attribute: Same with the previous cases, data owners need to create time parameters and send to corresponding AAs. Then the update key can be calculated as:

$$K_{3,x} = F\left(\rho'(x)\right)^{t'_x}\left(g^{\beta_{T(\rho'(x))}}\right)^{r'_{T(\rho'(x))} t'_x}$$

Since the component C_0 of the key ciphertext involves the public key $e(g,g)^{\alpha_j}$ of the authority, we also need to update the AA-related part. Let AS_A denote the set of AAs involved in the original access policy. Specifically, the public key of the AA that in the set $\left(AS_A - AS_{1,A'}\right)$ need to be removed, and the ones that in the set $AS_{2,A'}$ should be added. The update key of the C_0 component is calculated as:

$$UK_{C_0} = \left(K_1, K_2\right)$$
$$K_1 = \left(\prod_{j \in AS_A - AS_{1,A'}} e(g,g)^{\alpha_j}\right)^{-s}$$
$$K_2 = \left(\prod_{j \in AS_{2,A'}} e(g,g)^{\alpha_j}\right)^{s}$$

Ciphertext update: With the update keys, storage nodes can update the ciphertext based on the following algorithm: $CTUpdate\left(CT_{FID}, \left\{UK_{x,i}\right\}, UK_{C_0}\right) \rightarrow CT'_{FID}$
With the original key ciphertext CT_{FID}, the update key $\left\{UK_{x,i}\right\}$ which correspond to the three sets $S_{1,A'}, S_{2,A'}, S_{3,A'}$ and the AA update key UK_{C_0}, the algorithm generates the new ciphertext.

For the attribute component part, the update parameters are defined based on the attribute classification:

- Case 1 $(S_{1,A'})$: $C'_{2,x} = C_{2,i} \cdot K_{1,x}$, $C'_{3,x} = C_{3,i}$.
  If the update key $K_{2,x}$ is not empty, then the $C_{4,i}$ also should be updated as $C'_{4,x} = C_{4,i} \cdot K_{2,x}$.
  Otherwise, $C'_{4,x} = C_{4,i}$.
- Case 2 $(S_{2,A'})$: $C'_{2,x} = \left(C_{2,i}\right)^{v_x} \cdot K_{1,x}$, $C'_{3,x} = \left(C_{3,i}\right)^{v_x}$
  If there is $K_{2,x}$, then the $C'_{4,x} = \left(C_{4,i}\right)^{v_x} \cdot K_{2,x}$. Otherwise $C'_{4,x} = \left(C_{4,i}\right)^{v_x}$.
- Case 3 $(S_{3,A'})$: $C'_{2,x} = K_{1,x}$, $C'_{3,x} = K_{2,x}$, $C'_{4,x} = K_{3,x}$

For the attribute authority part, the updated operation of the C_0 ciphertext component is $C'_0 = C_0 \cdot UK_1 \cdot UK_2$.

The concrete structure of the new key ciphertext is:

$$CT'_{FID} = \left(C'_0, C_1, \forall x \in \left[1, l'\right] : C'_{2,x}, C'_{3,x}, C'_{4,x}\right)$$

TABLE 2. Notations used for performance evaluation.

Notations	Description
E_1	Exponentiation in group G.
F_1	Storage size of an element in group G.
EM_1	Multiplication in group G.
E_2	Exponentiation or multiplication in group G_T.
F_2	Storage size of an element in group G_T.
e	Pairing operation in group G_T.
l	Number of attributes in the access policy.
S_u	Set of attributes contained in the user's private key.
I_{AA}	Set of AA included in key ciphertext.
$\|*\|$	Number of elements in the set $*$.

C. SECURITY ANALYSIS

Theorem 3. After the ciphertext update algorithm is executed, the updated ciphertext is still secure.

Proof: We demonstrate the time range update and the access policy update respectively.

Time range update: The DO only modifies the time parameters of the time attributes and the corresponding ciphertext components. Therefore, we believe that the DO is trustable. Otherwise, the security discussion of the ciphertext is meaningless. In this scenario, only the time parameters and the update key that are sent to AA may expose the ciphertext. For time parameters, even if a malicious opponent intercepts the time parameter $g^{r_j'}$, the decryption key still cannot be generated without the authority private key β_j. Besides, the update key is sent to the node where the ciphertext is stored, and it cannot be used to decrypt the ciphertext. Therefore, the time range update operation is secure.

Access policy update: The DO re-defines a new access policy A', and the original ciphertext structure is leveraged as much as possible to reduce the communication and calculation overheads. In the ciphertext update phase, the storage nodes only know the relationships between the original access policy and the new access policy. Besides, the update keys which are transmitted in the network are not helpful in decryption. Moreover, similar to the time range update case, interception of the time parameters also cannot contribute to the ciphertext decryption. Therefore, the access attribute update operation is also secure.

VI. PERFORMANCE EVALUATION

To comprehensively evaluate the performance of the proposed TMO scheme, we compared its storage cost and computing overhead with the two popular encryption methods: DACMACS [16] and OOMADO [29]. Both of them are multi-authority outsourcing ABE schemes, DACMACS is suitable for cloud storage situations and OOMADO with an offline/online encryption feature is preferred in a mobile environment. Besides, the influences of time attribute and time AA on our proposed scheme is also evaluated. Moreover, we analyzed the performance of policy updating under various cases. The notations used in the theoretical analysis are listed in Table 2.

TABLE 3. The storage comparison of the three schemes.

Scheme	Key Ciphertext Size	AA Private Key Size
TMO	$(3l+1)F_1+F_2$	$(2\|I_{AA}\|+\|S_u\|+1)F_1$
DACMACS	$(3l+\|I_{AA}\|+1)F_1+F_2$	$(3\|I_{AA}\|+\|S_u\|)F_1$
OOMADO	$3lF_1+(2l+1)F_2$	$\|S_u\|F_1$

A. STORAGE COST

The storage size is a key factor that greatly influences the large-scale deployment of an access control scheme. Specifically, the ciphertext and the keys are the most frequently transmitted data among data owners, data storage nodes (cloud or ESN) and data users. Hence, as shown in Table 3, we compare the key ciphertext size and the AA private key size of the three encryption schemes. In TMO, the size of a symmetric key ciphertext is $(3l+1)F_1+F_2$, which is smaller than the other two schemes. Besides, the size of the private key obtained from each AA is $(2|I_{AA}|+|S_u|+1)F_1$, which is slightly smaller than DACMACS scheme and larger than the OOMADO scheme. This is due to the different encryption structures, OOMADO only stores one key per attribute, while our scheme needs to record more keys to support efficient outsourcing.

B. COMPUTING OVERHEAD

The computing overhead is mainly generated in the encryption phase(data owner), the pre-decryption phase(cloud or ECN) and the local decryption phase(data user). Noted that attribute-based encryption methods are employed to encrypt the symmetric key, so the data blocks encryption is not considered. Specifically, We compared the computing overheads of the three schemes from the aspects of algorithm, authority and attribute.

1) EXPERIMENTAL SETUP

The operating system used in the experiments is the Windows 10 Professional (x64) platform, the processor is Intel(R) Core(TM) i7-7700 CPU @3.60GHz and the memory is 8.00GB. The JPBC 2.0.0 library is applied to build the three encryption schemes, and a 160 bits type A elliptic curve which is based on the 512-bit finite field is the base curve of pairing.

2) ALGORITHM ANALYSIS

Generally, the encryption algorithm is composed of exponential operation and multiplication operation. In our experiments, each operation is executed 100 times, and we take the average value as the result. In the group G, the time of exponential operation and multiplication operation are 10.897ms and 3.114ms respectively. In the group G_T, the corresponding time are 1.964ms and 2.277ms. Since the time difference is relatively small in G_T, we use the same notation E_2 to represent the multiplication and the exponential operation for simplicity. In Z_p, a multiplication operation only takes 0.00779ms, so we ignored such small costs. In addition, the time of the linear pairing operation is 8.379 ms.

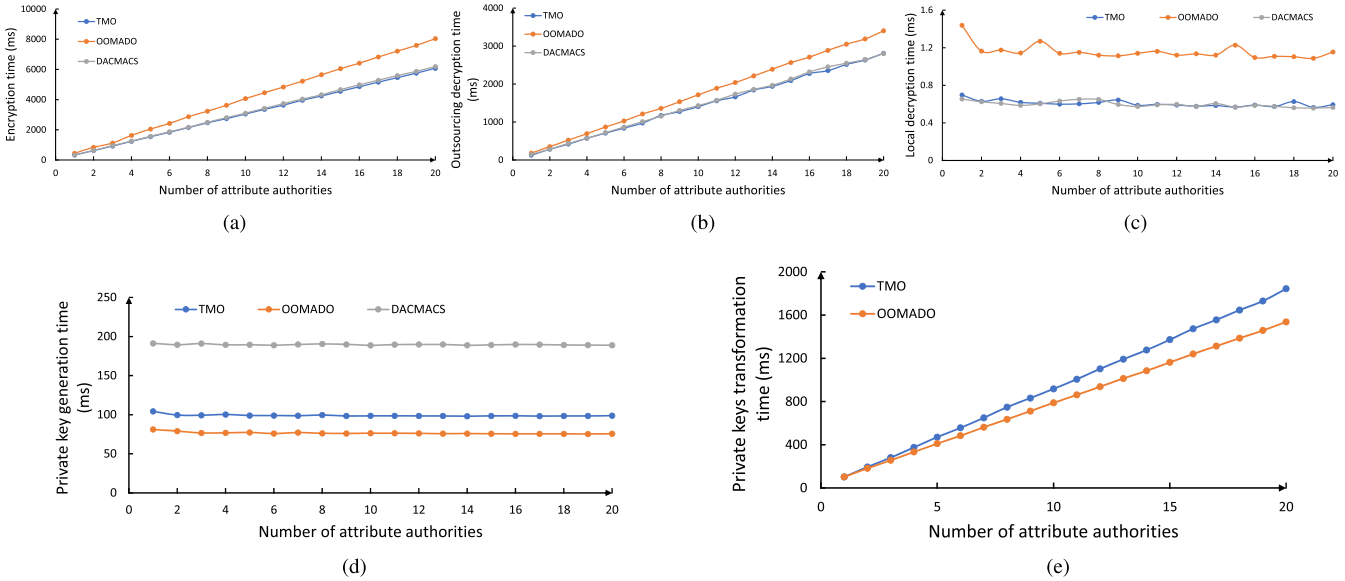


FIGURE 2. Comparison of the three schemes' performance under different number of attribute authorities. (a) Comparison of encryption time (OOMADO's result. (b) Comparison of outsourcing decryption time. (c) Comparison of local decryption time. (d) Comparison of AA private key generation time. (e) Comparison of keys transformation time.

TABLE 4. The computing overhead comparison of the three schemes.

Scheme	Encryption Cost	Outsourcing Decryption Cost	Local Decryption Cost
TMO	$(\|I_{AA}\|+1)\,E_2+(5l+1)\,E_1+2lEM_1$	$(\|I_{AA}\|+3\,\|S_u\|)\,e+(4\,\|S_u\|+\|I_{AA}\|-1)\,E_2$	$2E_2$
DACMACS	$(\|I_{AA}\|+1)\,E_2+(4l+\|I_{AA}\|+1)\,E_1+lEM_1$	$(2\,\|I_{AA}\|+3\,\|S_u\|)\,e+(4\,\|S_u\|+4\,\|I_{AA}\|-1)\,E_2$	$2E_2$
OOMADO	Offline: $6lE_2+5lE_1+e+2lEM_1$ Online: $2E_2$	$2lE_1+lEM_1+3lE_2+(2\,\|S_u\|)\,e+(5\,\|S_u\|-2)\,E_2$	$4E_2$

These basic operations are used to compare the computing overheads of the three encryption schemes. As Table 4 presents, TMO performs $|I_{AA}|+1$ exponential(multiplication) operations in G_T, $2l$ multiplication and $5l+1$ exponential operation in G when encrypting the symmetric key, the computing overhead is between the DACMACS and the OOMADO. In the pre-decryption phase, the overhead of TMO is less than the other two schemes. For the local decryption, data users only perform two multiplications in G_T to obtain the data, the cost is half of the OOMADO case.

3) ATTRIBUTE AUTHORITY IMPACT

Since DACMACS and OOMADO do not support time attribute, in the following experiments, all the attributes are universal attributes. To better evaluate the influence of AAs, asides from the three metrics(the encryption time, the outsourcing decryption time and the local decryption time) used in the algorithm analysis section, we also compared the AA private key generation time and the user key transformation time. In the experiments, each AA is responsible for 10 attributes, and the number of AA increased from 1 to 20. The results are shown in Figure 2.

As demonstrated in Figure 2(a), the encryption time of TMO is the smallest, while the result of DACMACS should be less than ours based on Table 4. The main reason is that the process of mapping an attribute to an element in the group G is implemented differently. TMO calculates the element by a hash function in real time, whereas DACMACS calculates the result in the AA setup phase and then stores in a HashMap for searching later. In the outsourcing pre-decryption case (Figure 2(b)) and the local decryption case (Figure 2(c)), the results are consistent with the theoretical analysis in Table 4. For the AA private key generation time(Figure 2(d)), the time of TMO is between the DACMACS case and the OOMADO case. The results can also be verified as the storage size difference showed in Table 3. Besides, due to the size of the private key set, the key transformation time of TMO is greater than the OOMADO case (Figure 2(e)). When the number of AA is 1, the size of the key set in the OOMADO case and the TMO case are 10 and 12 respectively. With the number of AA increase, the difference of key set size also increases. Moreover, Figure 2 also presents that the time of encryption, key transformation and outsourcing pre-decryption are proportional to the number of AA.

4) ATTRIBUTE IMPACT

Similarly, all the attributes are universal attributes in this group of experiments. The number of AA is set to 10, and the number of attributes increased from 1 to 10 in each AA. We also evaluated the five metrics' data, since the results/trends of the encryption time, outsourcing decryption

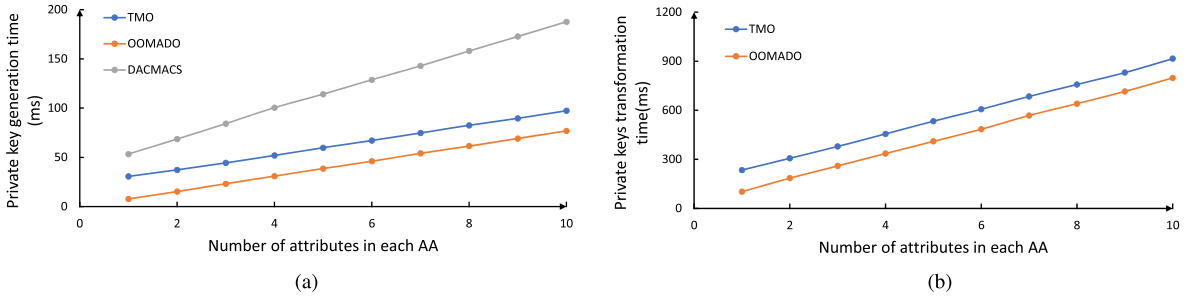


FIGURE 3. Comparison of the three schemes' performance under different number of attributes. (a) Comparison of AA private key generation time. (b) Comparison of keys transformation time.

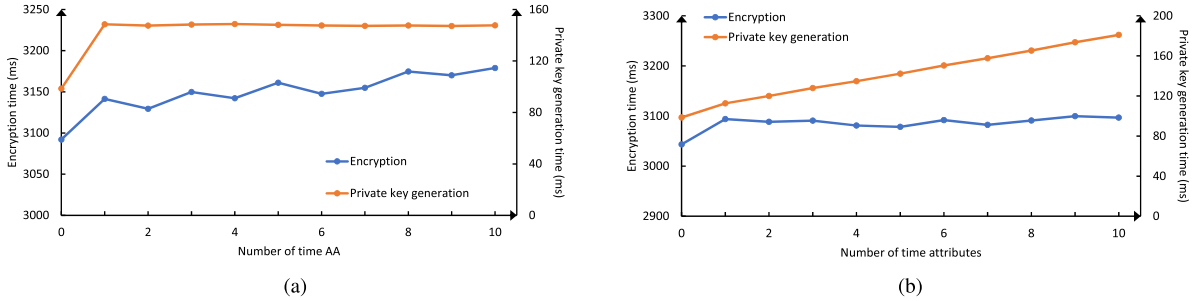


FIGURE 4. Encryption time and private key generation time under the impact of time attributes. (a) Operation time distribution in the time AA variation case. (b) Operation time distribution in the time attribute variation case.

time and local decryption time are the same with the previous AA evaluation case, Figure 3 only demonstrates the AA private key generation time and the user private keys transformation time data. As Figure 3(a) shows, TMO's AA private key generation time is still between the DACMACS case and the OOMADO case, and the AA private key generation time is proportional to the number of attributes. For keys transformation time (Figure 3(b)), the trend is also the same, OOMADO is faster than our scheme. When the number of attributes is 1, the size of the key set in the OOMADO case and the TMO case are 10 and 30 respectively. Hence, there is an obvious time difference. When the number of AA is fixed, the time difference is almost stable. Additionally, the time of encryption, AA key generation, keys transformation, outsourcing decryption and local decryption are all proportional to the number of attributes.

C. TIME ATTRIBUTE INFLUENCE

To evaluate the time attribute influence, we analyzed the results from the aspects of encryption time and AA key generation time. Since there is no difference between the private key structure of time attribute and universal attribute, their decryption time (outsourcing pre-decryption and local decryption) are close. So the decryption process is not considered in this evaluation. Specifically, the number of AA is set to 10, and the total number of attributes in each AA is also 10. The baseline case is that the attributes in every AA are all universal attributes.

Time AA variation: Generally, assuming the attributes of Time AA are composed of 5 time attributes and 5 universal attributes. As we can see from Figure 4(a), the encryption time tends to increase slowly with the number of Time AA increases. The DO needs to set a time parameter for time attributes in each AA and embeds it in the ciphertext. So the encryption process becomes longer with more time AA involving. For AA private key generation time, Time AA cost 50 ms more on average than the baseline case that with all universal attributes. The extra time is spent on executing a multiplication and exponential operation of the group G according to the structure of the encryption scheme. Besides, since we calculate the average time, the increasing of Time AA will not affect the key generation time.

Time attribute variation: In this experiment, every AA has time attributes which can change from 1 to 10. As Figure 4(b) presents, since the number of time parameters is mainly related to the number of AA, the time attribute variation in each AA does not affect the encryption time. Besides, due to that the computing overhead of each time private key is increased by an extra calculation in group G, the AA key generation time is proportional to the number of time attributes.

The experimental results show that TMO can support time attribute with acceptable encryption performance and no effect on decryption process.

D. POLICY UPDATE EVALUATION

For the policy update schemes, we analyzed the following two updating situations:

Time range update: The number of time parameters is directly related to the number of AA (not the time attributes),

TABLE 5. Attribute state updating information.

	Old Attribute		New Attribute	
	Universal	Time	Universal	Time
Case 1/Case 2 Type 1	✓		✓	
Case 1/Case 2 Type 2		✓		✓
Case 1/Case 2 Type 3		✓	✓	
Case 1/Case 2 Type 4	✓			✓
Case 3 Type 1			✓	
Case 3 Type 2				✓

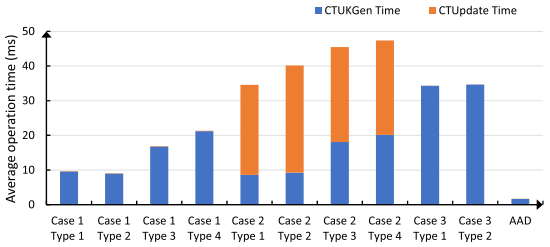


FIGURE 5. Comparison of the updating performance. (AAD: the ciphertext component updating caused by the difference of AA).

and the DO creates the update key for each AA. So, without loss of generality, we set up the experiment with four AAs: three universal AA (each contains 8 universal attributes) and one time AA which contains 4 time attributes and 4 universal attributes. After running the experiments 10 times, the update key generation time of the time parameter is 40.517ms, and the update time of the ciphertext is 0.243ms. The results show that the time range update is efficient in TMO scheme.

Access policy update: This approach is used in large scale policy updating situations. Specifically, the update key is calculated with considering both attributes difference and authority difference. In the experiments, we created an original access policy and a new access policy based on the following configurations: 1) Each of the three attribute sets $(S_{1,A'}, S_{2,A'}, S_{3,A'})$ contains 4 attributes; 2) Adding 2 new authorities and deleting 2 old authorities, and the total number of AA in access policy is 10. That is, $AS_{1,A'} = 8$, and $AS_{2,A'} = 2$. The changing of the attribute states in the three attribute sets (three cases) are shown in Table 5, for example, Case 1 Type 3 refers that the time attributes in the original access policy change to the universal attributes in the new access policy, and the attributes belong to $S_{1,A'}$ set.

As Figure 5 shows, when the attribute state is the same, the update key generation time is relatively short regardless of the cases. Besides, if the attribute state changes, the update key generation time will be increased due to the adding/deleting of a time parameter. For the ciphertext update algorithm, it takes a long time in Case 2 because the original ciphertext component needs to be re-randomized. Fortunately, Case 2 did not happen frequently in the real scenario. The ciphertext update time is very short in other cases, it takes about 0.1ms in Case 1 and 0.002ms in both Case 2 and Case 3. Moreover, comparing with the time spend in attribute updating, the ciphertext component corresponding to the authority difference only spends a very short time to finish the updating. Hence, our policy update algorithm can efficiently update a ciphertext with a low cost.

VII. CONCLUSION

With the blooming of IoT and ever-increasing demands of users, edge computing is proposed to leverage the computing and storage resources on the edge to process the massive data. With the benefit of reducing response time and saving bandwidth, edge computing also encounters huge challenges in security and privacy. Hence, in this paper, we proposed TMO, a time domain multi-authority outsourcing attribute-based encryption scheme, to enhance data security in the edge computing environment. The proposed TMO takes time as a key encryption factor to provide a flexible data acquisition mechanism. Besides, edge nodes are used to support multi-authority and outsourcing features, which can greatly improve the security and reduce the costs. Moreover, an efficient dynamic policy updating method is also developed, which updates the access policy online without wasting the network bandwidth. The security analysis and the comprehensive performance experimental results show that TMO can indeed improve the data security with less overhead in edge computing environment. In the future, we will explore the usage of historical data accessible time range, it is helpful to profile data users and predict data open possibilities. Moreover, we will improve TMO to support a large number of users and devices with more features in the edge environment.

APPENDIX

SECURITY ANALYSIS OF THEOREM 1

In Theorem 1, we states that there isn't a polynomial time simulator that can selectively break the decisional q-parallel BDHE assumption. The decisional q-parallel Bilinear Diffie-Hellman Exponent Assumption (q-BDHE) [22] is defined as follows: Let a challenge randomly choose $a, s, z, b_1, b_2, \ldots, b_q \in Z_P$. Suppose the adversary can get the following terms

$$\vec{\gamma} = (g, g^s, g^{\frac{1}{z}}, g^{\frac{a}{z}}, \ldots, g^{\frac{a^q}{z}}, g^a, \ldots, g^{(a^q)}, g^{(a^{q+2})}, \ldots g^{(a^{2q})},$$
$$\forall 1 \le j \le q : g^{s \cdot b_j}, g^{\frac{a}{b_j}}, \ldots, g^{\frac{a^q}{b_j}}, g^{\frac{a^{q+2}}{b_j}}, \ldots, g^{\frac{a^{2q}}{b_j}},$$
$$\forall 1 \le j, l \le q, l \ne j : g^{\left(a \cdot s \cdot \frac{b_l}{b_j}\right)}, \ldots, g^{\left(a^q \cdot s \cdot \frac{b_l}{b_j}\right)})$$

No polynomial time adversary can distinguish the term $e(g, g)^{a^{q+1}s}$ over G_T from a random element in G_T.

Based on the proposed security models, we challenge the decisional q-parallel BDHE assumption by constructing a simulator B.

Init: The simulator B receives the following data: 1) tuple γ in the decisional q-parallel BDHE assumption sent by the challenger C; 2) a unique ciphertext FID^* with the access policy (M^*, ρ^*) from the adversary A, and the size of the access matrix M^* is $l^* \times n^*$ ($n^* \le q - 1$).

Setup: Assume S_{AA} denotes a set of all AA in the system, S'_{AA} ($S'_{AA} \subset S_{AA}$) refers to the set of AAs which have

been statically corrupted. The simulator executes the *CASetup* algorithm to generate the global parameters *GP*, and sends the public parameters g, h to the adversary. The adversary shares S'_{AA} with the simulator. Then, simulator executes the *AASetup* algorithm for all uncorrupted authorities $AA_j \left(j \in S_{AA} - S'_{AA}\right)$, and sends the public parameter Apk_{AA_j} to adversary. For the corrupted authorities $AA_j \left(j \in S'_{AA}\right)$, the *AASetup* algorithm is executed by the adversary itself to obtain the corresponding public and private keys. The simulator randomly selects two elements $\alpha'_j, \beta_j \in Z_p$ and implicitly sets $\alpha_j = \alpha'_j + a^{q+1}$. Then the public key of the uncorrupted authority is calculated as:

$$e(g,g)^{\alpha_j} = e\left(g^a, g^{a^q}\right) \cdot e(g,g)^{\alpha'_j}$$
$$APK_{AA_j} = \left(e(g,g)^{\alpha_j}, g^{\beta_j}\right)$$

The simulator builds the random prediction oracle F of attributes through a table. If the element $F(x)$ corresponding to the attribute x already exists in the table, it directly returns $F(x)$. Otherwise, let R_x denote the set of all the row indexes i in the access matrix M^* that corresponds to the attribute x, then it selects a random element $h_x \in Z_p$. The random prediction oracle F of the attribute is calculated as:

$$F(x) = g^{h_x} \prod_{i \in R_x} g^{a^2 M^*_{i,1}/b_i} \cdot g^{a^3 M^*_{i,2}/b_i} \cdots g^{\frac{a^{n^*+1} M^*_{i,n}}{b_i}}$$

Note that if $R_x = \emptyset$, then $F(x) = g^{h_x}$ is still randomly distributed because the element g^{h_x} is random in group G. The result tuple $(x, F(x))$ is recorded on the table.

Phase 1: At this phase, the adversary can perform the three queries: 1) arbitrary universal private key query via (uid, S_{uid}); 2) time private key query through the tuple $\left(uid, FID', ST_{uid,FID'}\right)$; and 3) transformable keys query by $\left(uid, FID', \left\{Usk_{j,uid}\right\}, \left\{UTsk_{j,uid,FID'}\right\}\right)_{j \in S_{AA} - S'_{AA}}$. Similarly, there is a restriction on these key queries. Combining with the private keys from the corrupted authorities, the adversary's private keys for the ciphertext FID^* still cannot satisfy the requirements in the access policy M^*.

The simulator first selects a vector $\vec{w} = (w_1, w_2, \cdots, w_{n^*}) \in Z_p^{n^*}$, $(w_1 = -1)$. For all indexes i that $\rho^*(i) \in S_{uid} \cup ST_{uid,FID'}$, there exists $\vec{w} \cdot M^*_i = 0$. According to the definition of LSSS structure, such a vector must exist because $S_{uid} \cup ST_{uid,FID'}$ cannot simultaneously satisfy the ciphertext FID^* and the attribute requirements in M^*.

The simulator selects two random elements $r_{uid}, u'_{uid} \in Z_p$, and implicitly sets $u_{uid} = u'_{uid} - a^q$. Then, the user's identity key is $K_{uid} = g^{u_{uid}} = g^{u'_{uid} - a^q}$. Note that according to the definition of $g^{u_{uid}}$, there is an element $g^{a^{q+1}}$ that we cannot simulate. However, this element will be canceled by g^{α_j} in $D_{1,j}$.

For the universal private key query, the simulator randomly selects an element $r_j \in Z_P$ for each uncorrupted AA_j. The authority key is calculated as:

$$z_j = r_j + w_1 a^{q-1} + w_2 a^q + \cdots + w_{n^*} a^{q-n^*}$$

$$D_{1,j} = g^{\alpha'_{aid_k} + a^{q+1}} \cdot g^{a\left(u'_{uid} - a^q\right)} = g^{\alpha'_{aid_k} + a u'_{uid}}$$
$$D_{2,j} = g^{r_j} \cdot \prod_{i=1}^{n^*} g^{w_i \cdot a^{q-i}}$$

To calculate the universal private key D_{j,x_j} $(\forall x_j \in S_{j,uid})$, where $S_{j,uid}$ is the universal attribute set provided by the adversary, and $S_{uid} = \left\{S_{j,uid}\right\}$ which consists of the attributes from each uncorrupted authorities AA_j. If the attribute x_j is used in the access matrix M^*(*i.e.* there is a row index i in the matrix, such that $\rho^*(i) = x_j$), then it contains the element $g^{\frac{a^{q+1}}{b_k}}$ that we can't simulate. Fortunately, due to $\vec{w} \cdot M^*_i = 0$, this element can be cancelled. Hence, we have

$$D_{j,x_j} = g^{\beta_j u_{uid}} \cdot \left(g^{h_{x_j}} \cdot g^{\beta_j}\right)^{r_j} \cdot \left(\prod_{i=1}^{n^*} g^{w_i \cdot a^{q-i}}\right)^{\left(\beta_j + h_{x_j}\right)} \cdot \prod_{k \in R_x} \prod_{i=1}^{n^*} \left(g^{\frac{a^{i+1}}{b_k} r_j}\right)^{M^*_{k,i}} \cdot \prod_{k \in R_x} \prod_{i=1}^{n^*} \left(\prod_{j=1, j \neq i}^{n^*} g^{\frac{a^{q+1+i-j}}{b_k} w_j}\right)^{M^*_{k,i}}$$

For the universal attribute $x_j \notin S_{j,uid}$ that does not exist in M^*, its corresponding D_{j,x_j} is calculated as:

$$D_{j,x_j} = g^{\beta_j u_{uid}} \cdot \left(g^{h_{x_j}} \cdot g^{\beta_j}\right)^{r_j} \cdot \left(\prod_{i=1}^{n^*} g^{w_i \cdot a^{q-i}}\right)^{\left(\beta_j + h_{x_j}\right)}$$

For the time private key query, the simulator also re-randomly selects an element $r'_j \in Z_p$ for each uncorrupted AA_j, thus the authority key $D'_{2,j}$ in which r_j is embedded needs to be recalculated as

$$z_j = r'_j + w_1 a^{q-1} + w_2 a^q + \cdots + w_{n^*} a^{q-n^*}$$
$$D'_{2,j} = g^{r'_j} \cdot \prod_{i=1}^{n^*} g^{w_i \cdot a^{q-i}}$$

Then, after receiving the ciphertext number FID' submitted by the adversary, the simulator randomly selects a time parameter $t_{j,FID'} \in Z_p$ for each AA_j. The time attribute key is $D_{j,x_j}{}' \left(\forall x_j \in ST_{j,uid,FID'}\right)$, and $ST_{j,uid,FID'}$ refers to the set of time attributes. Similarly, $ST_{uid,FID'} = \left\{ST_{j,uid,FID'}\right\}$ consists of time attributes from each uncorrupted authority AA_j. If the time attribute x_j exists in the access matrix M^*, its time attribute key $D_{j,x_j}{}'$ is calculated as

$$D_{j,x_j}{}' = g^{\beta_j u_{uid}} \cdot \left(g^{h_{x_j}} \cdot g^{\beta_j t_{j,FID'}}\right)^{r_j} \cdot \left(\prod_{i=1}^{n^*} g^{w_i \cdot a^{q-i}}\right)^{\left(\beta_j t_{j,FID'} + h_{x_j}\right)} \cdot \prod_{k \in R_x} \prod_{i=1}^{n^*} \left(g^{\frac{a^{i+1}}{b_k} r_j}\right)^{M^*_{k,i}} \cdot \prod_{k \in R_x} \prod_{i=1}^{n^*} \left(\prod_{j=1, j \neq i}^{n^*} g^{\frac{a^{q+1+i-j}}{b_k} w_j}\right)^{M^*_{k,i}}$$

For the time attribute $x_j \notin ST_{j,uid,FID'}$ that does not exist in M^*, its $D_{j,x_j}{}'$ is calculated as

$$D_{j,x_j}{}' = g^{\beta_j u_{uid}} \cdot \left(g^{h_{x_j}} \cdot g^{\beta_j t_{j,FID'}}\right)^{r_j} \cdot \left(\prod_{i=1}^{n^*} g^{w_i \cdot a^{q-i}}\right)^{\left(\beta_j t_{j,FID'} + h_{x_j}\right)}$$

For the transformable keys query, the adversary submits the universal private key set $Usk_{uid} = \{Usk_{j,uid}\}$ and the time private key set $UTsk_{uid,FID'} = \{UTsk_{j,uid,FID'}\}$ for the ciphertext FID'. The simulator queries whether such transformable keys already exists on the table T. If exists, the transformable key $Tk_{uid,FID'}$ is directly returned to the adversary. Otherwise, the simulator generates the transformable keys by running the *TransKeyGen* algorithm. The result tuple $\left(Usk_{uid}, UTsk_{uid,FID'}, Tk_{uid,FID'}\right)$ will be saved to the table T, and the transformable keys will also be sent to the adversary. The simulator selects a random element $q \in Z_p$ as the local key, and the $Tk_{uid,FID'}$ is calculated as

$$Ek_{uid,FID'} = \left(\left\{D_{1,j}{}^{\frac{1}{q}}\right\}, \left\{D_{2,j}{}^{\frac{1}{q}}\right\}, \left\{D_{j,x_j}{}^{\frac{1}{q}}\right\}_{x_j \in S_{uid}}, \left\{D'_{2,j}{}^{\frac{1}{q}}\right\}, \left\{D_{j,x_j}{}'^{\frac{1}{q}}\right\}_{x_j \in ST_{uid,FID'}} \right)$$

$$Lk_{uid,FID'} = q$$

$$Tk_{uid,FID'} = \left(Ek_{uid,FID'}, Lk_{uid,FID'}\right)$$

Challenge: At this stage, the simulator mainly encrypts the symmetric key. The adversary submits two equal length symmetric keys k_0, k_1. The simulator randomly selects a secret value s and a symmetric key k_υ ($\upsilon \in \{0, 1\}$) to encrypt with the access policy specified by the adversary. The ciphertext CT_{key} of the symmetric key is calculated as follows:

$$C_0 = k_\upsilon \cdot \textstyle\prod_{j \in I_A} e\,(g, g)^{s \cdot \alpha_j}, C_1 = g^s$$

The public shared value of s can be constructed as:

$$\lambda_x = s \cdot M^*_{x,1} + \textstyle\sum_{j=2}^{n^*} \left(s a^{j-1} + y_j\right) M^*_{x,j}$$

Meanwhile, the simulator also randomly selected $l_1, l_2, \ldots, l_l \in Z_P$, and for any attribute x, its ciphertext component is:

$$C_{2,x} = g^{\beta_j l_x} \cdot g^{a\lambda_x} = g^{\beta_j l_x} \cdot g^{asM^*_{x,1}} \cdot \prod_{j=2}^{n^*} g^{\left(sa^j + ay_j\right) M^*_{x,j}}$$

$$C_{3,x} = g^{-l_x}$$

Besides, the ciphertext component $C_{4,x}$ is calculated according to the state of the attribute. Let S_T denote the set of time attributes contained in the access policy.

For a universal attribute x,

$$\forall x \notin S_T : C_{4,x} = g^{(\beta_j + h_x) l_x} \cdot \prod_{k \in R_x} \prod_{i=1}^{n^*} \left(g^{\frac{a^i}{b_k}}\right)^{M^*_{k,i} \cdot l_x}$$

For a time attribute x,

$$\forall x \in S_T : \; C_{4,x} = g^{(\beta_j t_{j,FID} + h_x) l_x} \cdot \textstyle\prod_{k \in R_x} \prod_{i=1}^{n^*} \left(g^{\frac{a^i}{b_k}}\right)^{M^*_{k,i} \cdot l_x}$$

Phase 2: Same as Phase 1.

Guess: The adversary guesses which symmetric key is encrypted. There are only two possibilities:

1) If the guess is correct, $\upsilon' = \upsilon$, the simulator returns 0, which means $T = e\,(g, g)^{a^{q+1}s}$. The simulator perfectly runs our security game. The advantage of the adversary is ϵ, so the advantage of B wining the secure game is

$$Pr\left[B\left(\vec{y}, T = e\,(g, g)^{a^{q+1}s}\right) = 0\right] = \frac{1}{2} + \epsilon$$

2)If the guess is wrong, $\upsilon' \neq \upsilon$, the simulator returns 1, which means T is a random element in the group G_T. The symmetric key k_υ is completely hidden from the adversary, so the advantage of B winning the game is

$$Pr\left[B\left(\vec{y}, T = R\right) = 0\right] = \frac{1}{2}$$

Finally, B's advantage in the game is

$$\begin{aligned} B &= \frac{1}{2}\left(Pr\left[B\left(\vec{y}, T = e\,(g, g)^{a^{q+1}s}\right) = 0\right] \right. \\ &\quad \left. + Pr\left[B\left(\vec{y}, T = R\right) = 0\right]\right) - \frac{1}{2} \\ &= \frac{\epsilon}{2} \end{aligned}$$

Therefore, our scheme is secure under the static corruption of attribute authorities.

REFERENCES

[1] M. Hung. (2017). *Leading the Iot—Gartner Insights on How to Lead in a Connected World.* [Online]. Available: https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf

[2] G. Rolph, A. Stein, and B. Stunder, "Real-time environmental applications and display system: Ready," *Environ. Model.Softw.*, vol. 95, pp. 210–228, Sep. 2017.

[3] H. Gao, W. Huang, X. Yang, Y. Duan, and Y. Yin, "Toward service selection for workflow reconfiguration: An interface-based computing solution," *Future Gener. Comput. Syst.*, vol. 87, pp. 298–311, Oct. 2018.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[5] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601–628, 1st Quart., 2018.

[6] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Comput. Secur.*, vol. 72, pp. 1–12, Jan. 2018. doi: 10.1016/j.cose.2017.08.007.

[7] H. Gao, Y. Duan, H. Miao, and Y. Yin, "An approach to data consistency checking for the dynamic replacement of service process," *IEEE Access*, vol. 5, pp. 11700–11711, 2017.

[8] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven Iot service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.

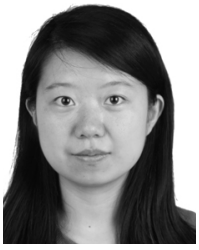
[9] H. Gao, K. Zhang, J. Yang, F. Wu, and H. Liu, "Applying improved particle swarm optimization for dynamic service composition focusing on quality of service evaluations under hybrid networks," *Int. J. Distrib. Sensor Netw.*, vol. 14, no. 2, pp. 1550–1583, Feb. 2018.

[10] K. Sha, W. Wei, T. A. Yang, Z. Wang, and W. Shi, "On security challenges and open issues in internet of things," *Future Gener. Comput. Syst.*, vol. 83, pp. 326–337, Jun. 2018.

[11] R. Roman *et al.*, "Mobile edge computing, fog: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[12] M. Satyanarayanan *et al.*, "Edge analytics in the internet of things," *IEEE Pervas. Comput.*, vol. 14, no. 2, pp. 24–31, Apr./Jun. 2015.

[13] K. Yang, X. Jia, K. Ren, R. Xie, and L. Huang, "Enabling efficient access control with dynamic policy updating for big data in the cloud," in *Proc. INFOCOM*, Apr. 2014, pp. 2013–2021.
[14] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[15] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* New York, NY, USA: Springer, 2015, pp. 315–332.
[16] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective data access control for multiauthority cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.
[17] H. Ma, R. Zhang, Z. Wan, Y. Lu, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 6, pp. 679–692, Dec. 2017.
[18] H. Wang, D. He, J. Shen, Z. Zheng, C. Zhao, and M. Zhao, "Verifiable outsourced ciphertext-policy attribute-based encryption in cloud computing," *Soft Comput.*, vol. 21, no. 24, pp. 7325–7335, Jun. 2017.
[19] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 89–98.
[20] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2005, pp. 457–473.
[21] S. Yu, K. Ren, W. Lou, and J. Li, "Defending against key abuse attacks in KP-ABE enabled broadcast systems," in *Proc. Int. Conf. Security Privacy Commun. Syst.* New York, NY, USA: Springer, 2009, pp. 311–329.
[22] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.*, Mar. 2011, pp. 53–70.
[23] J. Lai, R. H. Deng, Y. Li, and J. Weng, "Fully secure key-policy attribute-based encryption with constant-size ciphertexts and fast decryption," in *Proc. 9th ACM Symp. Inf., Comput. Commun. Secur.*, Jun. 2014, pp. 239–248.
[24] Q. Xu, C. Tan, Z. Fan, W. Zhu, Y. Xiao, and F. Cheng, "Secure data access control for fog computing based on multi-authority attribute-based signcryption with computation outsourcing and attribute revocation," *Sensors*, vol. 18, no. 5, p. 1609, 2018.
[25] S. Ruj, M. Stojmenovic, and A. Nayak, "Decentralized access control with anonymous authentication of data stored in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 384–394, Feb. 2014.
[26] M. Chase, "Multi-authority attribute based encryption," in *Proc. Theory Cryptogr. Conf.* New York, NY, USA: Springer, 2007, pp. 515–534.
[27] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* New York, NY, USA: Springer, 2011, pp. 568–588.
[28] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia, "PHOABE: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted iot," *Comput. Netw.*, vol. 133, pp. 141–156, Mar. 2018.
[29] S. J. De and S. Ruj, "Efficient decentralized attribute based access control for mobile clouds," *IEEE Trans. Cloud Comput.*, to be published.
[30] M. Green *et al.*, "Outsourcing the decryption of abe ciphertexts," in *Proc. USENIX Secur. Symp.*, Jun. 2011, no. 3, 2011.
[31] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for iot devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73520–73530, 2018.
[32] P. Zhang, Z. Chen, J. K. Liu, K. Liang, and H. Liu, "An efficient access control scheme with outsourcing capability and attribute update for fog computing," *Future Generat. Comput. Syst.*, vol. 78, pp. 753–762, Jan. 2018.
[33] C. Zuo, J. Shao, G. Wei, M. Xie, and M. Ji, "CCA-secure ABE with outsourced decryption for fog computing," *Future Generat. Comput. Syst.*, vol. 78, pp. 730–738, Jan. 2018.
[34] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Inf. Sci.*, vol. 258, pp. 355–370, Feb. 2014.
[35] J. Hong *et al.*, "Tafc: Time and attribute factors combined access control for time-sensitive data in public cloud," *IEEE Trans. Services Comput.*, to be published.
[36] K. Yang, Z. Liu, X. Jia, and X. S. Shen, "Time-domain attribute-based access control for cloud-based video content sharing: A cryptographic approach," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 940–950, May 2016. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7422115/
[37] A. Beimel, *Secure Schemes for Secret Sharing Key Distribution*. Haifa, Israel: Technion-Israel Institute of technology, 1996.



YOUHUIZI LI received the B.E. degree in computer science from Xidian University, in 2010, and the Ph.D. degree in computer science from Wayne State University, in 2016. She is currently an Assistant Professor with the Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou. She is also with the School of Compute Science and Technology, Hangzhou Dianzi University, China. Her research interests include energy efficiency, edge computing, and mobile systems. She is a member of CCF.

ZEYONG DONG is currently pursuing the M.S. degree with the School of Computer Science and Technology, Hangzhou Dianzi University, China. His research interests include edge computing and access control.

KEWEI SHA received the Ph.D. degree in computer science from Wayne State University, in 2008. He was the Department Chair and an Associate Professor with the Department of Software Engineering, Oklahoma City University (OCU). He is currently an Associate Director of the Cyber Security Institute and an Assistant Professor of computer science with the University of Houston–Clear Lake (UHCL). His research has been supported by NSF, NSFC, UHCL, and OCU. His research interests include the Internet of Things, cyber-physical systems, edge computing, network security and privacy, and data management and analytics. He is a Senior Member of the ACM. He received the IEEE Outstanding Leadership Award, in 2015, and the 2018 Albert Nelson Marquis Lifetime Achievement Award.

CONGFENG JIANG received the B.E. degree in hydro-electrical engineering from the North China University of Water Resources and Electric Power, in 2002, and the Ph.D. degree in hydro-electrical engineering from the Huazhong University of Science and Technology, in 2007. He is currently an Associate Professor with the Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou. He is also with the School of Compute Science and Technology, Hangzhou Dianzi University, China.

Since 2007, he has been an Assistant Professor with the School of Compute Science and Technology, Hangzhou Dianzi University. He has published more than 50 articles in grid computing, cloud computing, virtualization, and big data systems. His research interests include system optimization and performance evaluation, and distributed system benchmarking.

Dr. Jiang is a member of ACM and CCF.

JIAN WAN received the B.S. degree in mechanical engineering, the M.S. degree in mathematics, and the Ph.D. degree in computer science from Zhejiang University, in 1990, 1993, and 1996, respectively. From 2000 to 2015, he was the Dean of the School of Compute Science and Technology, Hangzhou Dianzi University. He is currently a Professor with the School of Information Engineering, Zhejiang University of Science and Technology, Hangzhou, China. He is also the Vice President of the Zhejiang University of Science and Technology and the Vice Director of the Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou. He has published more than 150 articles in grid and services computing, cloud computing, virtualization, and distributed systems. His research interests include distributed systems, computer networks, and big data analytics.

YUAN WANG received the Ph.D. degree in computer science from Zhejiang University, in 2006. He is currently the Vice President and an Executive Director of NetEase (Hangzhou) Network Co., Ltd., where he is fully responsible for public technical support work, cloud computing, and big data business of NetEase group, mainly including cloud computing and server-side architecture, front-end technology, big data mining and analysis, information security, multimedia, operation and maintenance, quality assurance, and so on.