P-MOD: Secure Privilege-Based Multilevel Organizational Data-Sharing in Cloud Computing

Ehab Zaghloul, Kai Zhou, *Student Member, IEEE*, and Jian Ren, *Senior Member, IEEE*

Abstract—Cloud computing has changed the way enterprises store, access and share data. Big data sets are constantly being uploaded to the cloud and shared within a hierarchy of many different individuals with different access privileges. With more data storage needs turning over to the cloud, finding a secure and efficient data access structure has become a major research issue. In this paper, a Privilege-based Multilevel Organizational Data-sharing scheme (P-MOD) is proposed that incorporates a privilege-based access structure into an attribute-based encryption mechanism to handle the management and sharing of big data sets. Our proposed privilege-based access structure helps reduce the complexity of defining hierarchies as the number of users grows, which makes managing healthcare records using mobile healthcare devices feasible. It can also facilitate organizations in applying big data analytics to understand populations in a holistic way. Security analysis shows that P-MOD is secure against adaptively chosen plaintext attack assuming the DBDH assumption holds. The comprehensive performance and simulation analyses using the real U.S. Census Income data set demonstrate that P-MOD is more efficient in computational complexity and storage space than the existing schemes.

Index Terms—Cloud computing, big data, hierarchy, privilege-based access, sensitive data, attribute-based encryption, mobile healthcare

1 Introduction

It was estimated that data breaches cost the United States' healthcare industry approximately $6.2 billion in 2016 alone [1]. To mitigate financial loss and implications on the reputation associated with data breaches, large multilevel organizations, such as healthcare networks, government agencies, banking institutions, commercial enterprises and etc., began allocating resources into data security research to develop and improve accessibility and storage of highly sensitive data.

One major way that large enterprises are adapting to increased sensitive data management is the utilization of the cloud environment. It was reported that more than half of all U.S. businesses have turned over to the cloud for their business data management needs [2]. The on-demand cloud access and data sharing can greatly reduce data management cost, storage flexibility, and capacity [3]. However, data owners have deep concerns when sharing data on the cloud due to security issues. Once uploaded and shared, the data owner inevitably loses control over the data, opening the door to unauthorized data access.

A critical issue for data owners is how to efficiently and securely grant privilege level-based access rights to a set of data. Data owners are becoming more interested in selectively sharing information with data users based on different levels of granted privileges. The desire to grant level-based access results in higher computational complexity and complicates the methods in which data is shared on the cloud. Research in this field focuses on finding enhanced schemes that can securely, efficiently and intelligently share data on the cloud among users according to granted access levels.

Based on a study conducted by the National Institute of Standards and Technology (NIST), Role-Based Access Control (RBAC) models are the most widely used to share data in hierarchical enterprises of 500 or more individuals [4]. RBAC models aim to restrict system access to authorized users as they provide access control mechanisms. The access control mechanisms are based on predefined and fixed roles making the models identity-centric. Each individual within the organization is assigned to a role that defines the privileges of the user. However, the limitations of this model are evident when presented with a large complex matrix of data users in an organization. The foundation of RBAC is based on abstract choices for roles. This would require a continuously increasing number of RBAC roles to properly encapsulate the privileges assigned to each user of the system. Managing a substantial number of rules can become a resource-intensive task, referred to as *role explosion* [5].

To better comprehend the importance of this study, consider a scenario where patients share their Public Health Records (PHR) on the cloud to be accessed by health providers and administrators of a hospital. In most cases, the patient wishes to grant the physician access to most parts of the PHR (including its most sensitive parts, e.g., medical history) while granting an administrator access to limited parts that are less sensitive (e.g., date of birth). In order to achieve that, the patient needs to define a hierarchy of data access privileges ranking various types of hospital employees. Next, the patient needs to clarify the privileges at each 7level to define the content that can be accessed by each data user. It is important to realize that patients have different conservative beliefs when it comes to their PHRs. For

• *The authors are with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824-1226. E-mail: {ebz, zhoukai, renjian}@msu.edu.*

Manuscript received 10 Oct. 2018; revised 4 Dec. 2018; accepted 19 Mar. 2019. Date of publication 25 Mar. 2019; date of current version 25 Nov. 2020. (Corresponding author: Ehab Zaghloul.)
Recommended for acceptance by H. Wang.
Digital Object Identifier no. 10.1109/TBDATA.2019.2907133

example, some would prefer granting access to the most sensitive parts of their PHRs to only specific physicians while denying others.

In this paper, a Privilege-based Multilevel Organizational Data-sharing scheme (P-MOD) is proposed. It builds on concepts presented in [6] to solve the problems of sharing data within organizations with complex hierarchies. The main contributions presented in this paper can be summarized as follows:

- We present multiple data file partitioning techniques and propose a privilege-based access structure that facilitate data sharing in hierarchical settings.
- We formally prove the security of P-MOD and show that it is secure against adaptively chosen plaintext attacks under the Decisional Bilinear Diffie-Hellman (DBDH) assumption.
- We present present a performance analysis for P-MOD and compare it to three existing schemes [7], [8], [9] that aim to achieve similar hierarchical goals.
- We implement P-MOD and conduct comprehensive simulations under various scenarios using the real U.S. Census Income data set [10]. We also compare our results to simulations we have conducted for two other schemes [7], [9] under the same conditions.

The rest of this paper is organized as follows. In Section 2, the related work is reviewed. In Section 3, preliminaries are introduced that summarize key concepts used in this research. Next, in Section 4, the problem formulation is described outlining the design goals and system model. In Section 5, the proposed P-MOD scheme is presented in detail. Following that, in Section 6 we formally prove the security of P-MOD based on the hardness of the Decisional Bilinear Diffie-Hellman (DBDH) problem. In Section 7, a performance analysis and evaluation of P-MOD is conducted and we support this analysis with some empirical results in Section 8. Finally, in Section 9, a conclusion is drawn to summarize the work done in this research.

2 Related Work

Fuzzy Identity-Base Encryption (Fuzzy IBE) was introduced in [11] to handle data sharing on the cloud in a flexible approach using encryption. The ciphertext is shared on the cloud to restrict access to authorized users. In order for an authorized individual to obtain the data, the user must request a private key from a key-issuer to decrypt the encrypted data. Fuzzy IBE is a specific type of function encryption [12] in which both the private key of the data user and ciphertext are affiliated with attributes. Attributes are descriptive pieces of information that can be assigned to any user or object. Since attributes can be any variable, they provide more flexibility when granting data access. The scheme enables a set of descriptive attributes to be associated with a private key and the ciphertext shared on the cloud. If the private key of the data user incorporates the minimum threshold requirement of attributes that match those integrated within the ciphertext, the data user can decrypt it. Although this scheme allows complex systems to be easily defined using attributes, it becomes less efficient when used to express large systems or when the number of attributes increases.

Attribute-Based Encryption (ABE) schemes later emerged to provide more versatility when sharing data. These schemes integrate two types of constructs: attributes and access policies. Access policies are statements that join attributes to express which users of the system are granted access and which users are denied. ABE schemes were introduced via two different approaches: Key-Policy Attribute-Based Encryption (KP-ABE) [13] and Ciphertext Policy Attribute-Based Encryption (CP-ABE) [7]. In KP-ABE, each ciphertext is labeled with a set of descriptive attributes, while each private key is integrated with an access policy. For authorized data users to decrypt the ciphertext, they must first obtain a private key from the key-issuer to use in decryption. The key-issuer integrates the access policy into the keys generated. Data users can successfully decrypt a ciphertext if the set of descriptive attributes associated with the ciphertext satisfies the access policy integrated within their private keys. KP-ABE can achieve fine-grained access control and is more flexible than Fuzzy IBE. However, the data owner must trust the key-issuer to only issue private keys to data users granted the privilege of access. This is a limitation since the data owner ultimately forfeits control over which data users are granted access.

On the other hand, CP-ABE is considered to be conceptually similar to Role-Based Access Control (RBAC) [14]. It gives the data owner control over which data user is able to decrypt certain ciphertexts. This is due to the access structure being integrated by the data owner into the ciphertext during encryption. It allows the private key generated by the key-issuer to only contain the set of attributes possessed by the data user. Some CP-ABE schemes [15], [16], [17], [18], [19] were later introduced that can provide higher flexibility and better efficiency.

Most attribute-based encryption schemes such as Fuzzy IBE, KP-ABE, and CP-ABE serve as a better solution when data users are not ranked into a hierarchy and each is independent of one another (i.e., no relationships). However, they share a common limitation of high computational complexity in the case of large multilevel organizations. These schemes require a single data file to be encrypted with a large number of attributes (from different levels) to grant them access to it.

Hierarchical Attribute-Based Encryption (HABE) that combines the Hierarchical Identity-Based Encryption (HIBE) [20] scheme and CP-ABE [7] was later introduced in [8], [21]. HABE is able to achieve fine-grained access control in a hierarchical organization. It consists of a root master that generates and distributes parameters and keys, multiple domain masters that delegate keys to domain masters at the following levels, and numerous users. In this scheme, keys are generated in the same hierarchical key generation approach as the HIBE scheme. To express an access policy, HABE uses a disjunctive normal form where all attributes are administered from the same domain authority into one conjunctive clause. This scheme becomes unsuitable for practical implementation when replicas of the same attributes are administered by other domain authorities. Synchronizing attribute administration might become a challenging issue with complex organizations that have multiple domain authorities. Examples of other hierarchical schemes were later introduced in [22], [23], [24], [25].

File Hierarchy Ciphertext Policy Attribute-Based Encryption (FH-CP-ABE) [9] is one of the most recent hierarchical solutions available today. It proposes a leveled access structure to manage a hierarchical organization that shares data of various sensitivity. A single access structure was proposed that represents both the hierarchy and the access policies of an organization. This access structure consists of a root node, transport nodes, and leaf nodes. The root node and transport nodes are in the form of gates (i.e., AND or OR). The leaf nodes represent attributes that are possessed by data users. Based on the possession of certain attributes, each data user is mapped into specific transport nodes (certain levels within the hierarchy) based on the access structure that the user satisfies. If the data user satisfies a full branch of the access structure, then the data user is ranked at the root node (highest level within the hierarchy). Data users ranked at the highest level (root node) can decrypt a ciphertext of highest sensitivity and any other ciphertext with less sensitivity in the lower levels of the hierarchy. The nodes ranked in the lower levels (transport nodes) can not decrypt any ciphertexts in the levels above. The main advantage of this scheme is that it provides leveled access structures which are integrated into a single access structure. As a result, storage space is saved as only one copy of the ciphertext is needed to be shared on the cloud for all data users. However, since this scheme uses a single access structure to represent the full hierarchy, the higher levels are forced to accommodate attributes of all the levels below. As the number of levels increases in the hierarchy, the number of attributes grows exponentially making this scheme infeasible on a large scale. A simplified and reduced access structure is proposed to reduce the computational complexity by removing all branches of the single access structure while keeping one full branch. The full branch consists of the root node, a set of transport nodes (one for each level), and the leaf nodes (attributes). However, in real-life applications, relationships within an organization are often built in a cross-functional matrix, making this a complicated solution when assigning privileges.

3 Preliminaries

3.1 Cryptographic Hash Function

A cryptographic hash function h is a mathematical algorithm that maps data of arbitrary size to a bit string of fixed size. It is cryptographically secure if it satisfies the following requirements:

- *Preimage-Resistance*: It should be computationally infeasible to find any input for any pre-specified output which hashes to that output, i.e., for any given y, it should be computationally infeasible to find an x such that $h(x) = y$.
- *Week Collision Resistance*: For any given x, it should be computationally infeasible to find $x' \neq x$ such that $h(x') = h(x)$ [26].
- *Strong Collision-Resistance*: It should be computationally infeasible to find any two distinct inputs x and x', such that $h(x) = h(x')$.

3.2 Bilinear Maps

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of the same prime order p. The generator of $\mathbb{G}_0$ is denoted as g. A bilinear map from $\mathbb{G}_0 \times \mathbb{G}_0$ to $\mathbb{G}_1$ is a function $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ that satisfies the following properties:

- *Bilinearity*: $e(g^a, g^b) = e(g, g)^{ab}$ for any $a, b \in \mathbb{Z}_p$.
- *Symmetry*: $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$ for any $a, b \in \mathbb{Z}_p$.
- *Non-degeneracy*: $e(g, g) \neq 1$.
- *Computability*: $e(g, g)$ is an efficiently computable algorithm.

3.3 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

The DBDH assumption [27] is a computational hardness assumption and is defined as follows:

Let $\mathbb{G}_0$ be a group of prime order p, g be a generator, and $a, b, c \in \mathbb{Z}_p$ be chosen at random.

It is infeasible for the adversary to distinguish between any given $(g, g^a, g^b, g^c, e(g, g)^{abc})$ and (g, g^a, g^b, g^c, R), where $R \in_r \mathbb{G}_1$ is a random element and $\in_r$ denotes a random selection. An algorithm $\mathcal{A}$ that outputs a guess $z \in \{0, 1\}$, has advantage ε in solving the DBDH problem in $\mathbb{G}_0$ if:

$$\Big| Pr[\mathcal{A}(g, g^a, g^b, g^c, T = e(g, g)^{abc}) = 0] - Pr[\mathcal{A}(g, g^a, g^b, g^c, T = R) = 0] \Big| \geq \varepsilon. \tag{1}$$

The DBDH assumption holds if no polynomial algorithm has a non-negligible advantage in solving the DBDH problem.

3.4 Access Structure

An access structure represents access policies for a set of individuals interested in gaining individual access to a secret. The access structure defines sets of attributes that can be possessed by a single individual to allow access to the secret. It is defined as follows:

Let $\{P_1, \ldots, P_n\}$ be a set of parties. A set of parties that can reconstruct the secret is defined as a collection. The collection is monotone meaning that, if $\mathscr{A} \subseteq 2^{\{P_1, \ldots, P_n\}}$ then $\forall B \in \mathscr{A}$ and $B \subseteq C$ implies $C \in \mathscr{A}$. An access structure is a monotone collection $\mathscr{A}$ of non-empty subsets $\{P_1, \ldots, P_n\}$, i.e., $\mathscr{A} \subseteq 2^{\{P_1, \ldots, P_n\}} \setminus \emptyset$. The sets in $\mathscr{A}$ are called the authorized sets and the sets not in $\mathscr{A}$ are called the unauthorized sets.

In this paper, we use parties to represent the attributes. This means that an access structure $\mathscr{A}$ may consist of both authorized and unauthorized sets of attributes.

3.5 Leveled Access Tree $\mathcal{T}_i$

An access tree $\mathcal{T}_i$ at level $\mathcal{L}_i$ represents an access structure that determines whether a data user can decrypt the ciphertext esk_i at that level or not. A $\mathcal{T}_i$ may consist of multiple nodes. We use $x_l^i \in \mathcal{T}_i$ to represent the lth node of $\mathcal{T}_i$. The non-leaf nodes of $\mathcal{T}_i$ are in the form of threshold gates, while the leaf nodes represent possible attribute values possessed by data users.

For every node $x_l^i \in \mathcal{T}_i$, a threshold value $k_{x_l^i}$ is assigned. A node in the form of an AND gate is associated with a threshold value $k_{x_l^i} = num_{x_l^i}$, where $num_{x_l^i}$ represents the number of children of node x_l^i. A node in the form of an OR gate or any leaf node representing an attribute is associated with a threshold value $k_{x_l^i} = 1$.

TABLE 1
Notations Summary

Symbol	Definition
DO	A data owner
H	The hierarchical layout of the O
$\mathcal{L}_i$	ith level within H where $1 \leq i \leq k$
$\mathcal{T}_i$	ith access tree at $\mathcal{L}_i$ where $1 \leq i \leq k$
F_i	ith data file part where $1 \leq i \leq k$
sk_i	ith symmetric key used to encrypt F_i where $1 \leq i \leq k$
EF_i	ith sym. encrypted F_i under sk_i where $1 \leq i \leq k$
esk_i	ith cp-abe encrypted sk_i under $\mathcal{T}_i$ at $\mathcal{L}_i$ where $1 \leq i \leq k$
DU_j	jth data user in set DU where $1 \leq j \leq m$
SK_j	jth data user's private key where $1 \leq j \leq m$
$A_{j,u}$	uth attribute within the jth data user's attribute set $\mathbb{A}_j$ where $1 \leq u \leq n$ and $1 \leq j \leq m$
x_l^i	lth node of $\mathcal{T}_i$ where $1 \leq l \leq \infty$ and $1 \leq i \leq k$
$k_{x_l^i}$	Threshold value of x_l^i where $1 \leq l \leq \infty$ and $1 \leq i \leq k$

The root node x_1^i of each $\mathcal{T}_i$ carries a secret sk_i. The data user that possesses the correct set of attributes can satisfy $\mathcal{T}_i$ and obtain sk_i.

4 Problem Formulation

Consider a data owner that possesses a data file $\mathcal{F}$ and wishes to selectively share different segments of it on the cloud among a set of data users based on certain access privileges. We assume that the data users can be ranked into a hierarchy that defines their access privileges.

Selectively sharing data files on the cloud becomes a burden on the data owner as the hierarchy grows (the access privileges increase in number) and/or as the access restrictions become more complex due to an increase in the sensitivity of the file segments. A trivial solution involves the data owner to use public key encryption. This solution would require the data owner to encrypt the same part of the data file once for each data user being granted access then upload the resulting ciphertexts to the cloud. The data users would then fetch their uniquely encrypted parts of the file from the cloud and utilize their private keys to decrypt them. This method ensures that no unprivileged data user will gain access to any part of the data file even if that user is able to download the ciphertexts from the cloud. However, on a large scale, public key encryption becomes an inefficient solution due to the increase in the number of encryptions and large storage spaces required. Therefore, the challenge is to provide the data owners with an efficient, secure and privilege-based method that allows them to selectively share their data files among multiple data users while minimizing the required cloud storage space needed to store the encrypted data segments.

4.1 Design Goals

Based on the problem described above, we have the following design goals:

Privilege-Based Access: Data is shared in a hierarchical manner based on user privileges. Data users with more privileges (ranked at the higher levels of the hierarchy) are granted access to more sensitive parts of $\mathcal{F}$ than those with fewer privileges (ranked at the lower levels of the hierarchy).

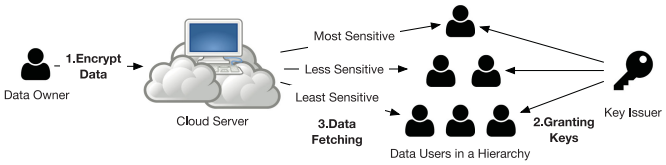

Fig. 1. General scheme of privilege-based data sharing. (1) Data owner encrypts the data under P-MOD. (2) Data users are granted keys based on their level. (3) Data users fetch the data from the cloud and decrypt it with their keys.

Data Confidentiality: All parts of $\mathcal{F}$ are completely protected from unprivileged data users (including the storage space). Data users are entitled to access the parts of $\mathcal{F}$ corresponding to the levels they fall in and/or any other parts corresponding to the levels below with respect to their own.

Fine-grained access control: The data owner has the capability to encrypt any part of $\mathcal{F}$ using any set of descriptive attributes he/she wishes, limiting access to only authorized data users. The set of descriptive attributes is defined by the data owner at the time of encryption and can be selected from an infinite pool.

Collusion resistant: Two or more data users at the same/different level can not combine their private keys to gain access to any part of $\mathcal{F}$ they are not authorized to access independently.

4.2 System Model

The general model of privilege-based data sharing among hierarchically-ranked data users is illustrated in Fig. 1. The system consists of four main entities:

Data owner (DO). An individual that owns a data file and wishes to selectively share it with multiple data users based on certain desired privacy preferences.

Data users (DU): A set of hierarchically-ranked individuals $\{DU_j \in DU \mid 1 \leq j \leq \infty\}$ interested in obtaining different segments of a shared data file. Data users fall into different levels within the hierarchy based on specific sets of attributes $\mathbb{A}_j = \{A_{j,1}, A_{j,2} \ldots A_{j,n}\}$ they possess.

Key-issuer: A fully trusted entity that generates private keys for the data users that possess a correct set of attributes.

Cloud server: A non-trusted entity used to store the encrypted segments of the data file.

As shown by Fig. 1, the data users ranked at the higher levels are granted access to more sensitive segments of file $\mathcal{F}$ than those ranked at lower levels. In our proposed scheme, the hierarchy is not fixed nor predefined. It is defined by the data owners as they encrypt their files to be shared with a set of data users. A hierarchy can consist of multiple levels $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$, where $1 \leq k \leq \infty$. Level $\mathcal{L}_1$ represents the highest rank while level $\mathcal{L}_k$ represents the lowest rank. At each level $\mathcal{L}_i$ of the hierarchy, the data owner defines the desired leveled access tree $\mathcal{T}_i$, where $1 \leq i \leq k$. The access tree identifies the policies required in order for a user to gain access to the data file at a certain level.

A summary of the notations used in this paper is presented in Table 1.

5 The Proposed P-MOD Scheme

This section presents the construction of P-MOD. We assume that file $\mathcal{F}$ is partitioned into k parts based on data sensitivity

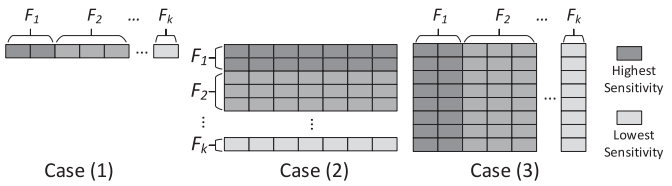

Fig. 2. File partitioning.

defined by its owner. Each part of $\mathcal{F}$ is then independently encrypted and shared among the data users of the system under our proposed privilege-based access structure.

5.1 Data File Partitioning and Encryption

The DO partitions file $\mathcal{F}$ into a set of k data sections, that is $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$. Each $F_i \in \mathcal{F}$ is treated as a new file that is associated with a sensitivity value used to assign access rights to the data users based on their privileges. The process of partitioning $\mathcal{F}$ is performed based on the structure of $\mathcal{F}$. We assume that $\mathcal{F}$ consists of at least one record, resulting in multiple ways to partition it as shown in Fig. 2.

If $\mathcal{F}$ consists of a single record, then each $F_i \in \mathcal{F}$ represents one or more record attribute(s) associated with the record, as shown in case (1) in Fig. 2. However, if $\mathcal{F}$ consists of multiple records, then the DO has flexibility in choosing how to partition it. One approach is to handle each record as a whole, where records are clustered into groups of similar sensitivity. In this case, each $F_i \in \mathcal{F}$ represents one or more record(s), as shown case (2) in Fig. 2. Alternatively, partitioning can be performed over specific record attributes, versus the whole record. In this case, $F_i \in \mathcal{F}$ represents one or more record attribute(s) of the records, as shown case (3) in Fig. 2.

Regardless of how partitioning is performed, each $F_i \in \mathcal{F}$ is then treated as a new data file. Suppose F_1 contains the most sensitive information of $\mathcal{F}$ that can only be accessed by a data user at the highest level $\mathcal{L}_1$ and F_k contains the least sensitive information of $\mathcal{F}$ that can be accessed by all data users at any level of the hierarchy. Before the DO uploads $\{F_1, F_2, \ldots, F_k\}$ to the cloud, each $F_i \in \mathcal{F}$ is encrypted separately using a symmetric encryption algorithm such as the Advanced Encryption Algorithm with a secret key sk_i to produce an encrypted file

$$EF_i = \mathsf{Enc}_{sk_i}(F_i). \tag{2}$$

For key selection, the DO randomly selects sk_1. The remaining symmetric keys $\{sk_2, \ldots, sk_k\}$ are then derived from sk_1 using a one-way cryptographic hash function h, that is

$$sk_{i+1} = h(sk_i). \tag{3}$$

Next, the symmetric keys are encrypted as discussed in Section 5.3, to be accessed only by the data users that have been granted the privilege of access. The privileged data users that are successful in obtaining sk_i corresponding to level $\mathcal{L}_i$ can derive $\{sk_{i+1}, \ldots, sk_k\}$ using Equation (3). However, given the properties of hash function h, sk_i cannot be used to derive any of the symmetric keys $\{sk_1, \ldots, sk_{i-1}\}$.

5.2 The P-MOD Privilege-Based Access Structure

Fig. 3 illustrates the general privilege-based access structure of P-MOD. Data users are ranked into k levels of privileges, $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$. The DO defines an access tree $\mathcal{T}_i$ at each

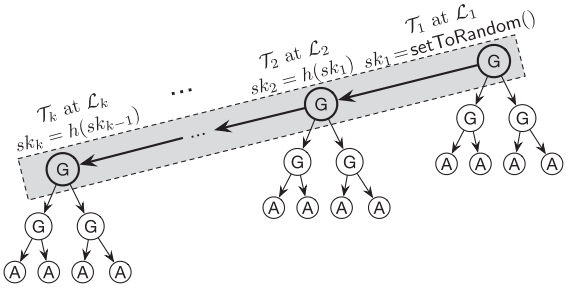

Fig. 3. Privilege-based multilevel access structure.

corresponding level $\mathcal{L}_i$. Each $\mathcal{T}_i$ is associated with the appropriate leaf nodes (attributes) that define the privileges of the level. Data users that possess the correct sets of attributes which can satisfy $\mathcal{T}_i$ at $\mathcal{L}_i$ are granted access to symmetric key sk_i, hence, can derive $\{sk_{i+1}, \ldots, sk_k\}$ and are able to decrypt parts $\{EF_i, EF_{i+1}, \ldots, EF_k\}$.

As shown in Fig. 3, an access tree $\mathcal{T}_i$ may consist of non-leaf nodes and leaf nodes. The non-leaf nodes are threshold gates, represented as 'G', while the leaf nodes are attributes, represented as 'A'. The DO may construct access trees from any number and layout of nodes that satisfy the privacy preferences desired.

One of the advantages of our proposed privilege-based access structure is the ability to reduce attribute replication when defining the hierarchy. Data users that possess attributes which can satisfy access tree $\mathcal{T}_i$ are granted access to sk_i, however, they do not need to possess attributes that can also satisfy the access trees $\{\mathcal{T}_{i+1}, \ldots, \mathcal{T}_k\}$ in order to obtain $\{sk_{i+1}, \ldots, sk_k\}$. This helps simplify the process of defining a hierarchy as the number of users and/or access constraints grow. With reduced-size access trees, we can greatly reduce the computational complexity when encrypting file partitions, generating private keys for privileged users and decrypting ciphertexts.

5.3 The P-MOD Construction

The scheme is based on the construction presented in [7] and formally divides the process into four main functions:

$\mathsf{Setup}(1^\kappa)$: This is a probabilistic function carried out by the key-issuer. The Setup function takes a security parameter κ and randomly chooses values $\alpha, \beta \in \mathbb{Z}_p$.

$$PK = \{\mathbb{G}_0, g, B = g^\beta, e(g,g)^\alpha\}, \tag{4}$$

$$MK = \{\beta, g^\alpha\}. \tag{5}$$

$\mathsf{KeyGen}(MK, \mathbb{A}_j)$: This is a probabilistic function carried out by the key-issuer. The inputs to this function are MK generated by the Setup function, and the attribute set $\mathbb{A}_j$ of jth data user, where $A_{j,u} \in \mathbb{A}_j$ represents the uth attribute within the set. The KeyGen function outputs a unique private key SK_j for the data user. In order to guarantee a unique SK_j, it generates a random value $r_j \in \mathbb{Z}_p$ and incorporates it within the private key. Based on the number of attributes in the input set $\mathbb{A}_j$, the KeyGen function also generates a random value $r_{j,u} \in_r \mathbb{Z}_p$ for each attribute within the set. The SK_j is defined as:

$$SK_j = \big(D_j = g^{(\alpha+r_j)/\beta}, \{D_{j,u} = g^{r_j} \cdot h(u)^{r_{j,u}}, D'_{j,u} = g^{r_{j,u}} \,|\, \forall A_{j,u} \in \mathbb{A}_j\}\big). \tag{6}$$

The purpose of the randomly selected r_j is to ensure that each SK_j is unique and the attribute components within the SK_j are associated. It should be infeasible for data users to collude by combining components of their private keys ($D_{j,u}$ and $D'_{j,u}$) to decrypt data beyond their individual access rights. This means, the attribute components from different private keys cannot be combined to access unauthorized data.

Encrypt$(PK, sk_i, \mathcal{T}_i)$: This is a probabilistic function carried out by the DO to encrypt the symmetric keys that are to be shared with the privileged data users. The inputs to this function are PK, the public key generated by the Setup function, the symmetric key sk_i derived in Equation (3) representing the data that will be encrypted, and the access tree $\mathcal{T}_i$ that defines the authorized set of attributes at $\mathcal{L}_i$. The output of this function is the encrypted symmetric key esk_i.

For $\{sk_1, \ldots, sk_k\}$, the Encrypt function will run k times, once for each sk_i. At each run, the Encrypt function chooses a polynomial $q_{x_l^i}$ with degree $d_{x_l^i} = k_{x_l^i} - 1$ for each node $x_l^i \in \mathcal{T}_i$. The process of assigning polynomials to each x_l^i occurs in a top-bottom approach starting from the root node in $\mathcal{T}_i$. The Encrypt function chooses a secret $s_i \in \mathbb{Z}_p$ and sets the value of $q_{x_1^i}(0) = s_i$. Next, it randomly chooses the remaining points of the polynomial to completely define it. For any other node $x_l^i \in \mathcal{T}_i$, the Encrypt function sets the value $q_{x_l^i}(0) = q_{\text{parent}}(\text{index}(x_l^i))$, where q_{parent} is the parent node polynomial of x_l^i. The remaining points of those polynomials are then randomly chosen.

Let X_i be the set of leaf nodes in $\mathcal{T}_i$. The encrypted symmetric key esk_i at $\mathcal{L}_i$ is then constructed as:

$$esk_i = \big(\mathcal{T}_i, \tilde{C}_i = sk_i \cdot e(g,g)^{\alpha \cdot s_i}, C_i = g^{\beta \cdot s_i}, \{C_{x_l^i} = g^{q_{x_l^i}(0)}, C'_{x_l^i} = h(x_l^i)^{q_{x_l^i}(0)} \,|\, \forall x_l^i \in X_i\}\big). \tag{7}$$

Decrypt(esk_i, SK_j): This is a deterministic function carried out by the data user. The inputs to this function are an encrypted symmetric key esk_i corresponding to $\mathcal{L}_i$, and the private key SK_j of the jth data user.

The Decrypt function operates in a recursive manner propagating through the nodes in $\mathcal{T}_i$ by calling a recursive function defined as DecryptNode(esk_i, SK_j, x_l^i). The inputs to this function are esk_i, SK_j and a node x_l^i within $\mathcal{T}_i$. If the attributes incorporated within SK_j satisfy the rules within $\mathcal{T}_i$, the data user can decrypt esk_i and obtain sk_i.

DecryptNode(esk_i, SK_j, x_l^i) performs differently depending on whether x_l^i is a leaf or non-leaf node. If x_l^i is a leaf node and $\text{att}(x_l^i) \notin \mathbb{A}_j$, DecryptNode$(esk_i, SK_j, x_l^i)$ returns $\emptyset$, otherwise, $\text{att}(x_l^i) = A_{j,u} \in \mathbb{A}_j$ and DecryptNode(esk_i, SK_j, x_l^i) is defined as:

$$\begin{aligned}\text{DecryptNode}(esk_i, SK_j, x_l^i) &= \frac{e(D_{j,u}, C_{x_l^i})}{e(D'_{j,u}, C'_{x_l^i})} \\ &= \frac{e(g^{r_j} \cdot h(u)^{r_{j,u}}, g^{q_{x_l^i}(0)})}{e(g^{r_{j,u}}, h(x_l^i)^{q_{x_l^i}(0)})} \\ &= e(g,g)^{r_j \cdot q_{x_l^i}(0)}.\end{aligned} \tag{8}$$

If x_l^i is a non-leaf node, DecryptNode(esk_i, SK_j, x_l^i) operates recursively. For each node $z_{l,c}^i$ that is a child of x_l^i, DecryptNode$(esk_i, SK_j, z_{l,c}^i)$ is computed and the output is stored in $F_{z_{l,c}^i}$.

This recursive function is based on Lagrange interpolation. The Lagrange coefficient $\Delta_{a,\mathbb{A}_j}$ for $a \in \mathbb{Z}_p$ and the set of attributes $\mathbb{A}_j$ is defined as:

$$\Delta_{a,\mathbb{A}_j}(x) = \prod_{k \in \mathbb{A}_j, k \neq a} \frac{x-k}{a-k}. \tag{9}$$

Let $\mathbb{A}_{j,x_l^i}$ be an arbitrary $k_{x_l^i}$-sized set of child nodes $z_{l,c}^i$ such that $F_{z_{l,c}^i} \neq \emptyset$. If no such set exists then the function returns $F_{z_{l,c}^i} = \emptyset$. Otherwise, $F_{z_{l,c}^i}$ is computed using Lagrange interpolation as follows:

$$\begin{aligned}F_{x_l^i} &= \prod_{z_{l,c}^i \in \mathbb{A}_{j,x_l^i}} F_{z_{l,c}^i}^{\Delta_{a,\mathbb{A}'_{j,x_l^i}}(0)} \\ &= \prod_{z_{l,c}^i \in \mathbb{A}_{j,x_l^i}} \left(e(g,g)^{r_j \cdot q_{z_{l,c}^i}(0)}\right)^{\Delta_{a,\mathbb{A}'_{j,x_l^i}}(0)} \\ &= \prod_{z_{l,c}^i \in \mathbb{A}_{j,x_l^i}} \left(e(g,g)^{r_j \cdot q_{\text{parent}(z_{l,c}^i)}(\text{index}(z_{l,c}^i))}\right)^{\Delta_{a,\mathbb{A}'_{j,x_l^i}}(0)} \\ &= \prod_{z_{l,c}^i \in \mathbb{A}_{j,x_l^i}} e(g,g)^{r_j \cdot q_{x_l^i}(i) \cdot \Delta_{a,\mathbb{A}'_{j,x_l^i}}(0)} \\ &= e(g,g)^{r_j \cdot q_{x_l^i}(0)},\end{aligned} \tag{10}$$

where $a = \text{index}(z_{l,c}^i)$ and $\mathbb{A}'_{j,x_l^i} = \{\text{index}(z_{l,c}^i) : z_{l,c}^i \in \mathbb{A}_{j,x_l^i}\}$.

If the attributes in $\mathbb{A}_j$ satisfy $\mathcal{T}_i$, then the following is computed at the root node x_1^i.

$$\begin{aligned}R_i &= \text{DecryptNode}(esk_i, SK_j, x_1^i) \\ &= e(g,g)^{r_j \cdot q_{x_1^i}(0)} \\ &= e(g,g)^{r_j \cdot s_i}.\end{aligned} \tag{11}$$

To obtain sk_i from the result derived in Equation (11), we compute the following:

$$\frac{\tilde{C}_i}{\frac{e(C_i, D_j)}{R_i}} = \frac{sk_i \cdot e(g,g)^{\alpha \cdot s_i}}{\frac{e\left(g^{\beta \cdot s_i}, g^{(\alpha+r_j)/\beta}\right)}{e(g,g)^{r_j \cdot s_i}}} = sk_i. \tag{12}$$

At this point, the data user can simply decrypt EF_i using the sk_i derived from Equation (12) to obtain the plaintext F_i as follows:

$$F_i = \mathsf{Dec}_{sk_i}(EF_i). \tag{13}$$

If the data user is interested in attaining data files $\{F_{i+1} \dots F_k\}$ belonging to levels $\{\mathcal{L}_{i+1} \dots \mathcal{L}_k\}$ respectively, the user can compute the symmetric keys $\{sk_{i+1}, \dots, sk_k\}$ of the lower level using the derived sk_i as previously discussed in Equation (3). Finally, any $\{EF_{i+1} \dots EF_k\}$ at the lower levels can be decrypted as in Equation (13).

6 Security Analysis

In this section, a formal proof of security for P-MOD is presented. It is assumed that a symmetric encryption technique such as AES is used to secure each individual data file $F_i \in \mathcal{F}$. It is also assumed that the process of attribute authentication between a data user and the key-issuer, in order for the data user to obtain a private key, is secure and efficient.

Theorem 1. *P-MOD is secure against unprivileged accesses assuming the hash function is collision resistant.*

Proof. Let $\mathbb{A}_j = \{A_{j,1}, A_{j,2} \dots A_{j,n}\}$ be a set of attributes possessed by user DU_j and $\mathcal{T}_i$ an access tree at level $\mathcal{L}_i$ of a hierarchy. If $\mathbb{A}_j \in \mathcal{T}_i$, the user can obtain sk_i as discussed in Equations (8-12). Given hash function h, the user cannot derive $\{sk_1 \dots sk_{i-1}\}$. Therefore, P-MOD is immune to unprivileged accesses. □

Following the work presented in [27], we also provide a security proof based on ciphertext indistinguishability which proves that the adversary is not able to distinguish pairs of ciphertexts. A cryptosystem is considered to be secure under this property if the probability of an adversary to identify a data file that has been randomly selected from a two-element data file chosen by the adversary and encrypted does not significantly exceed $\frac{1}{2}$. We first present an Indistinguishability under Chosen-Plaintext Attack (IND-CPA) security game. Next, based on the IND-CPA security game, a formal proof of security is provided for P-MOD.

IND-CPA is a game used to test for security of asymmetric key encryption algorithms. In this game, the adversary is modeled as a probabilistic polynomial-time algorithm. The algorithms in the game must be completed and the results returned within a polynomial number of time steps. The adversary will choose to be challenged on an encryption under a leveled access tree $\mathcal{T}^*$. The adversary can impersonate any data user and request many private keys SK_j. However, the game rules require that any attribute set $\mathbb{A}_j$ that the adversary claims to possess does not satisfy $\mathcal{T}^*$. The security game is divided into the following steps:

Initialization. The adversary selects an access tree $\mathcal{T}^*$ to be challenged against and commits to it.

Setup. The challenger runs the Setup function and sends the public key PK to the adversary.

Phase 1. The adversary requests multiple private keys $(SK_1, \dots, SK_{q_1})$ corresponding to q_1 different sets of attributes $(\mathbb{A}_1, \dots, \mathbb{A}_{q_1})$.

Challenge. The adversary submits two equal length data files F_0 and F_1 to the challenger. The adversary also sends $\mathcal{T}^*$ such that none of $(SK_1, \dots, SK_{q_1})$ generated from Phase 1 contain correct sets of attributes that satisfy it. The challenger flips a coin μ randomly and encrypts F_μ under $\mathcal{T}^*$. Finally, the challenger sends the ciphertext CT^* generated according to Equation (7) to the adversary.

Phase 2. Repeat phase 1 with the restriction that none of the newly generated private keys $(SK_{q_1+1}, \dots, SK_q)$ corresponding to the different sets of attributes $(\mathbb{A}_{q_1+1}, \dots, \mathbb{A}_q)$ contain correct sets of attributes that satisfy $\mathcal{T}^*$.

Guess. The adversary outputs a guess μ' of μ. The adversary wins the security game if $\mu' = \mu$ and loses otherwise.

Definition 1 (Secure against adaptively chosen plaintext attack.). *P-MOD is said to be secure against an adaptively chosen plaintext attack if any polynomial-time adversary has only a negligible advantage in the security game, where the advantage is defined as* $Adv = Pr[\mu' = \mu] - \frac{1}{2}$.

The security of P-MOD is reduced to the hardness of the DBDH problem. Based on Theorem 1 and by proving that a single esk_i at any $\mathcal{L}_i$ is secure, the whole system is proved to be secure since all ciphertexts at any level follow the same rules.

Theorem 2. *P-MOD is secure against adaptively chosen plaintext attack if the DBDH assumption holds.*

Proof. Assume there is an adversary that has non-negligible advantage $\varepsilon = \mathsf{Adv}_{\mathcal{A}}$. We construct a simulator that can distinguish a DBDH element from a random element with advantage ε. Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ be an efficiently computable bilinear map and $\mathbb{G}_0$ is of prime order p with generator g. The DBDH challenger begins by selecting the random parameters: $a, b, c \in_r \mathbb{Z}_p$. Let $g \in \mathbb{G}_0$ be a generator and T is defined as $T = e(g,g)^{abc}$ if $\mu = 0$, and $T = R$ otherwise, where $\mu \in_r \{0,1\}$ and $R \in_r \mathbb{G}_1$. The simulator acts as the challenger in the following game:

Initialization. The simulator accepts the DBDH challenge requested by the adversary who selects the $\mathcal{T}^*$.

Setup. The simulator runs the Setup function. It chooses a random $\alpha^* \in_r \mathbb{Z}_p$ and computes the value $\alpha = \alpha^* + ab$. Next, it simulates $e(g,g)^{\alpha} \leftarrow e(g,g)^{\alpha^*+ab} = e(g,g)^{\alpha^*} e(g,g)^{ab}$ and $B = g^{\beta} \leftarrow g^b$, where b represents a simulation of the value β. Finally, it sends all components of $PK = \{\mathbb{G}_0, g, B = g^{\beta}, e(g,g)^{\alpha}\}$ to the adversary.

Phase 1. In this phase, the adversary requests multiple private keys $(SK_1, \dots, SK_{q_1})$ corresponding to q_1 different sets of attributes $(\mathbb{A}_1, \dots, \mathbb{A}_{q_1})$. After receiving an SK_j query for a given set $\mathbb{A}_j$ where $\mathbb{A}_j \notin \mathcal{T}^*$ (i.e., $\forall A_{j,u} \in \mathbb{A}_j$ does not satisfy $\mathcal{T}^*$), the simulator chooses a random $r'_j \in_r \mathbb{Z}_p$ and defines $r_j = r'_j - b$. Next, it simulates $D_j = g^{(\alpha + r_j)/\beta} \leftarrow g^{\alpha/\beta} g^{r_j/\beta} = g^{(\alpha^*+ab)/b} g^{(r'_j - b)/b}$. Then, $\forall A_{j,u} \in \mathbb{A}_j$, it selects a random $r_{j,u} \in_r \mathbb{Z}_p$ and simulates $D_{j,u} = g^{r_j} \cdot h(u)^{r_{j,u}} \leftarrow g^{r'_j - b} h(u)^{r_{j,u}}$ and $D'_{j,u} \leftarrow g^{r_{j,u}}$. Finally, the simulated values of $SK_j = (D_j, \{D_{j,u}, D'_{j,u} \,|\, r_{j,u} \in_r \mathbb{Z}_p,\ \forall A_{j,u} \in \mathbb{A}_j\})$ are sent to the adversary.

Challenge. The adversary sends two plaintext data files sk_0 and sk_1 to the simulator who randomly chooses a $\mu \in_r \{0,1\}$ by flipping a coin to select one of the files. The simulator then runs the Encrypt function and derives a

ciphertext CT^*. It simulates $\tilde{C} = sk_\mu \cdot e(g,g)^{\alpha \cdot sk_\mu} \leftarrow sk_\mu \cdot e(g,g)^{(\alpha^* + ab)c} = sk_\mu \cdot Te(g,g)^{\alpha^* c}$, where c represents a simulation of the value sk_μ and $T = e(g,g)^{abc}$. Next, it simulates $C = g^{\beta \cdot sk_\mu} \leftarrow g^{bc}$. Finally, for each attribute $x \in X^*$ (set of leaf nodes in $\mathcal{T}^*$) it computes $C_x = g^{q_x(0)}$ and $C'_x = h(x)^{q_x(0)}$. The simulated values of $CT^* = \{\mathcal{T}^*, \tilde{C}, C, \forall x \in X^* : C_x, C'_x\}$ are then sent to the adversary.

Phase 2. Repeat *Phase 1* with the restriction that the requested private keys are associated with attribute sets such that, $\forall \mathbb{A}_j \,|\, q_1 + 1 \leq j \leq q$ and $\mathbb{A}_j \notin \mathcal{T}^*$.

Guess. The adversary tries to guess the value μ. If the adversary guesses the correct value, the simulator outputs 0 to indicate that $T = e(g,g)^{abc}$, or 1 to indicate that $T = R$, a random group element in $\mathbb{G}_1$.

Given a simulator $\mathcal{A}$, if $T = e(g,g)^{abc}$, then CT^* is a valid ciphertext, $Adv = \varepsilon$ and

$$Pr\Big[\mathcal{A}\Big(g, g^a, g^b, g^c, T = e(g,g)^{abc}\Big) = 0\Big] = \frac{1}{2} + \varepsilon. \tag{14}$$

If $T = R$ then $\tilde{C}$ is nothing more than a random value to the adversary. Therefore,

$$Pr\big[\mathcal{A}\big(g, g^a, g^b, g^c, T = R\big) = 0\big] = \frac{1}{2}. \tag{15}$$

From Equations (14) and (15), we can conclude that

$$\begin{aligned} &\Big|Pr\Big[\mathcal{A}\Big(g, g^a, g^b, g^c, T = e(g,g)^{abc}\Big) = 0\Big] \\ &-Pr\big[\mathcal{A}\big(g, g^a, g^b, g^c, T = R\big) = 0\big]\Big| = \varepsilon. \end{aligned} \tag{16}$$

Therefore, the simulator plays the DBDH game with a non-negligible advantage and the proof is complete. □

7 Performance Analysis and Evaluation

In this section, we present a performance analysis for P-MOD and compare it with three existing schemes, CP-ABE [7], HABE [21] and FH-CP-ABE [9].

7.1 Traditional CP-ABE in a Hierarchical Setting

CP-ABE [7] handles sharing of independent pieces of data based on independent access policies. It was not designed to support a privilege-based access structure (i.e., hierarchical organization). Therefore, to adapt CP-ABE to a privilege-based access structure, the Encrypt function runs once for each level. However, if it were to be used in a hierarchical organization, there would be a trade off between the key management and the complexity of the encryption and decryption processes. Fig. 4 shows the two general cases in which CP-ABE is utilized to share data with users in a hierarchical organization.

In case (1), key management is favored over encryption and decryption complexities. The root node of each access tree at levels $\mathcal{L}_2 \ldots \mathcal{L}_k$ consist of an OR gate to accommodate the policies of the levels above. When generating keys for the data users, the key issuer needs to incorporate only a subset of the attributes rather than the entire set belonging to the access tree. As a result, the size of each private key is therefore optimized and the key management process becomes less resource-intensive. However, since levels are independent, attributes must be repeatedly incorporated into each access tree. As a result, the sizes of the access trees at lower levels will increase, as shown in case (1) of Fig. 4. For complex organizations with a large number of levels, the access trees will become even larger. This results in an increase in encryption and decryption complexities.

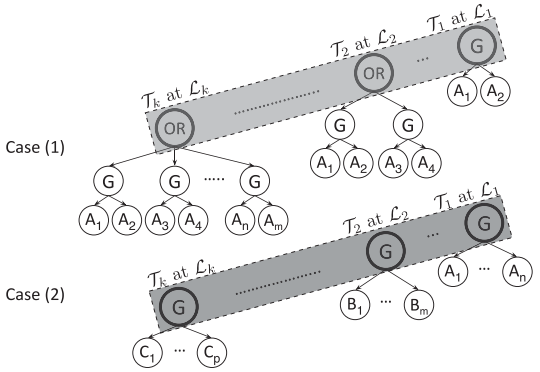

Fig. 4. CP-ABE utilized in a hierarchical organization.

On the other hand, case (2) favors minimizing encryption and decryption complexities over key management. Each data file is encrypted under an access policy with a set of unique attributes at each level without considering privileges and relationships. This results in simpler access trees at each level of the hierarchy and therefore lower encryption and decryption complexities. However, to grant access to an individual at a specific level, the key-issuer must generate a private key for that individual that incorporates the attributes at that level and all the levels below. Complex hierarchies that include a large number of attributes, could result in complicated key management. As a result, private keys will require incorporating a large number of attributes.

7.2 Computational Cost

We formulate the encryption and decryption costs based on the number of group operations $f_{\mathbb{G}_0}$, $f_{\mathbb{G}_1}$ for groups $\mathbb{G}_0$, $\mathbb{G}_1$ respectively and the number of bilinear mapping operations e involved in the Encrypt and the Decrypt functions for each scheme. Table 2 summarizes the number of operations for each scheme.

7.2.1 Encryption Cost

Encryption cost is measured as the number of basic operations involved in generating the ciphertext from the plaintext. It is formulated based on the Encrypt function which involves group operations $f_{\mathbb{G}_0}$ and $f_{\mathbb{G}_1}$.

The number of operations involved in sharing an independent piece of data using CP-ABE is $(2|X| + 1)$ and 2 for $f_{\mathbb{G}_0}$ and $f_{\mathbb{G}_1}$ respectively, where $|X|$ denotes the number of leaf nodes (attributes) of the access tree $\mathcal{T}$. For a hierarchical organization, we present the encryption complexity of case (1) in Fig. 4 as it involves a relationship between all levels. In a real life application, case (2) would not satisfy a hierarchical organization as it requires attributes to be shared by all users regardless which level they belong to. As shown in Table 2, the number of operations involved in the encryption process is formulated as $(2(|X_1| + \cdots + |X_k|) + k)$ and $2k$ for $f_{\mathbb{G}_0}$ and $f_{\mathbb{G}_1}$ respectively, where $|X_1|, |X_2|, \ldots, |X_k|$ are the number of leaf nodes (attributes) associated with access trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ respectively. However, the reuse of

TABLE 2
Comparison of Number of Operations

Function	Operation	CP-ABE [7]	HABE [8], [21]	FH-CP-ABE [9]	P-MOD
Encrypt	f_{G_0}	$2(\|X_1\| + \cdots + \|X_k\|) + k$	$\|X\|$	$2\|X\| + k$	$2(\|Y_1\| + \cdots + \|Y_k\|) + k$
	f_{G_1}	$2k$	1	$2v\|\mathbb{A}_T\| + 2k$	2
Decrypt	e	$k(2\|\mathbb{A}_j\| + 1)$	$3\|\mathbb{A}_j\|$	$2\|\mathbb{A}_j\| + 1$	$2\|\mathbb{A}_j\|$
	f_{G_1}	$2(\|S_1\| + \cdots + \|S_k\|) + 2k$	$\|\mathbb{A}_j\| - 1$	$2\|S\| + v\|\mathbb{A}_T\| + 2k$	$2\|S\| + 2$

attributes needed at each level in this scheme increases the computational complexity, making it an overall inefficient solution for hierarchical organizational structures.

Similarly, P-MOD generates a ciphertext for each level of the hierarchy. The number of operations involved is $(2(|Y_1| + \cdots + |Y_k|) + k)$ and $2k$ for $f_{\mathbb{G}_0}$ and $f_{\mathbb{G}_1}$ respectively, where $|Y_1|, |Y_2|, \ldots, |Y_k|$ are the number of leaf nodes (attributes) associated with access trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ respectively. However, P-MOD leverages a privilege-based access structure as discussed in Section 5.2. This results in smaller sized and level-specific access trees which minimize the number of attributes where, $|Y_i| < |X_i|, \forall i \in k$.

When comparing P-MOD with hierarchical schemes such as HABE and FH-CP-ABE, P-MOD minimizes the overall number of operations. This is because the encryption process for schemes such as HABE and FH-CP-ABE involve more complex hierarchies and access trees that contain all the access policies for all the levels. On the contrary, P-MOD involves smaller access trees, each one limited to level-specific attributes and policies.

In HABE, a single hierarchy represents the root master, domain masters, users and attributes. This may result in complex hierarchies as the number of users increase. Therefore, the size of $|X|$ may end up being large making the encryption process expensive. Similary, FH-CP-ABE [9] uses a single access tree $\mathcal{T}$ to encrypt all data files to be shared. The number of operations involved are $(2|X| + k)$ and $(2v|\mathbb{A}_T| + 2k)$ for operations $f_{\mathbb{G}_0}$ and $f_{\mathbb{G}_1}$ respectively, where $|\mathbb{A}_T|$ is the number of transport nodes (levels), and v is the number of children nodes associated with a transport node. This may also result in large sets $|X|$, $|\mathbb{A}_T|$ and v and lead to an expensive encryption process.

7.2.2 Decryption Cost

Decryption cost is measured as the number of basic operations involved in decrypting the ciphertext into plaintext. It is formulated based on the Decrypt function which involves bilinear operations e and group operations $f_{\mathbb{G}_1}$.

For a single Decrypt run, CP-ABE [7] involves $(2|\mathbb{A}_j|)$ and $(2|S| + 2)$ number of operations e and $f_{\mathbb{G}_1}$ respectively, where $|\mathbb{A}_j|$ is the number of attributes possessed by the jth data user and $|S|$ is the least number of interior nodes that satisfy $\mathcal{T}$. However, to adapt CP-ABE to a privilege-based access structure, the Decrypt function is run as many times as the number of ciphertexts a data user wishes to decrypt. The number of operations involved are $k(2|\mathbb{A}_j| + 1)$ and $(2[|S_1| + \ldots + |S_k|] + 2k)$ for operations e and $f_{\mathbb{G}_1}$ respectively, where $|S_1|, |S_2|, \ldots, |S_k|$ are the least number of interior nodes that satisfy the access trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ respectively.

The number of operations involved in the decryption process for P-MOD is similar to a single CP-ABE decryption run. P-MOD needs to run the Decrypt function only one time, even if the data user needs to obtain more than one data file. The Decrypt function requires $(2|\mathbb{A}_j|)$ and $(2|S| + 2)$ for operations e and $f_{\mathbb{G}_1}$ respectively. Once the data user successfully decrypts the ciphertext (obtains the symmetric key at his/her level), the user can derive the remaining lower level keys as described in Equation (3). The complexity of the operations involved in deriving the symmetric keys and decrypting ciphertexts at the lower levels are negligible in comparison with the group and bilinear operations involved in running the Decrypt function, and therefore could be ignored.

When comparing P-MOD to HABE, the e number of operations involved are slightly similar. However, the $f_{\mathbb{G}_1}$ number of operations required are $|\mathbb{A}_j - 1|$ and $2|S| + 2$ respectively. Therefore, for both schemes, encryption complexity would greatly depend on the number of attributes possessed by the user and the access tree generated during encryption.

FH-CP-ABE [9] aims to satisfy a certain transport node (level) of the single access tree $\mathcal{T}$ allowing the data user to decrypt certain encrypted files up to that level. The number of operations involved in this process are $(2|\mathbb{A}_j| + 1)$ and $(2|S| + v|\mathbb{A}_T| + 2k)$ for operations e and $f_{\mathbb{G}_1}$ respectively. Again, since FH-CP-ABE uses a single access tree $\mathcal{T}$, the number of transport nodes $|\mathbb{A}_T|$ can be large, resulting in higher decryption complexity. The least number of interior nodes $|S|$ that satisfy $\mathcal{T}$ can also be large if the construction of the access tree $\mathcal{T}$ is not optimized. Constructing a single access tree that accommodates a large number of attributes is resource-intensive and could become complicated as the access rules become more sophisticated. When comparing the decryption costs of P-MOD and FH-CP-ABE, the decryption cost of P-MOD depends on $|\mathbb{A}_j|$ and $|S|$ while the decryption complexity of FH-CP-ABE depends on $|S|$, v, $|\mathbb{A}_T|$ and k. The size of the sets in FH-CP-ABE will always be greater than the size of the sets in P-MOD due to the different constructions of access trees in each scheme.

7.3 Storage Cost

To evaluate the storage efficiency, we formulate the bit-length of the private keys and ciphertexts generated by each scheme. Table 3 represents a comparison of the storage costs in bit-length for all schemes. The bit-length of a single element in $\mathbb{G}_0$, $\mathbb{G}_1$ and $\mathbb{Z}_p$ are denoted as $L_{\mathbb{G}_0}$, $L_{\mathbb{G}_1}$ and $L_{\mathbb{Z}_p}$ respectively.

As shown in table, the bit-length of the private keys for all schemes are similar. In terms of space complexity, this can be reduced to $\mathcal{O}(\mathbb{A}_j)$. On the other hand, the size of ciphertexts differ. The size of a ciphertext is based on the output of the Encrypt function for each scheme.

TABLE 3
Comparison of Private Key and Ciphertext Sizes

Component	Length	CP-ABE [7]	HABE [8], [21]	FH-CP-ABE [9]	P-MOD
Private Key	$L_{\mathbb{G}_0}$	$2\lvert\mathbb{A}_j\rvert+1$	$\lvert\mathbb{A}_j\rvert$	$2\lvert\mathbb{A}_j\rvert+1$	$2\lvert\mathbb{A}_j\rvert+1$
Ciphertext	$L_{\mathbb{G}_0}$	$2(\lvert X_1\rvert+\cdots+\lvert X_k\rvert)+k$	$\lvert X\rvert$	$2\lvert X\rvert+k$	$2(\lvert Y_1\rvert+\cdots+\lvert Y_k\rvert)+k$
	$L_{\mathbb{G}_1}$	k	1	$v\lvert\mathbb{A}_T\rvert+k$	k

For CP-ABE, the total size of all generated ciphertexts from all levels consists of $(2(|X_1|+\cdots+|X_k|)+k)$ elements from $\mathbb{G}_0$ and k elements from $\mathbb{G}_1$. Using this scheme, the lower levels must accommodate attributes of the higher levels. Ciphertext size can potentially end up large in size due to attribute replication at each level.

For HABE, the ciphertext consists of $|X|$ elements from $\mathbb{G}_0$ and 1 element from $\mathbb{G}_1$. Similar to the discussion in the encryption cost of HABE, the size of X can be large due to the complexity of how the hierarchy is defined. Likewise, the single ciphertext generated by FH-CP-ABE [9] consists of $(2|X|+k)$ elements from $\mathbb{G}_0$ and $(v|\mathbb{A}_T|+k)$ elements from $\mathbb{G}_1$. In this scheme, the ciphertext size depends on $|X|$, $|\mathbb{A}_T|$, v and k. As the size of these sets grow, the ciphertext size can grow exponentially based on how the tree $\mathcal{T}$ is constructed.

P-MOD generates ciphertexts in a similar approach to those generated by CP-ABE. The total size of all generated ciphertexts consists of $(2[|Y_1|+\cdots+|Y_k|]+k)$ elements from $\mathbb{G}_0$ and k elements from $\mathbb{G}_1$. However, the size of the ciphertext generated by P-MOD is shown to be smaller in size than CP-ABE in all instances. This is based on the composition of our proposed access structure which does not duplicate attributes, therefore generates smaller ciphertexts.

8 Empirical Results

In this section, the results of various simulations are presented to support the performance analysis discussed in Section 7. P-MOD is implemented and simulated in Java using the CP-ABE toolkit [28] and the Java Pairing-Based Cryptography library (JPBC) [29]. For comparison, simulations are also conducted for CP-ABE [7] and FH-CP-ABE [9] under the same conditions as P-MOD. All simulations are conducted on an Intel(R) Core(TM) i5-4200M at 2.50 GHz and 4.00 GB RAM machine running the Windows 10 OS.

The data set used in our simulations is the real U.S. Census Income data set [10]. It consists of 30,163 records, and each record is composed of 9 different record attributes. The records are stored in a Microsoft Excel file $\mathcal{F}$, with a total size of 2.42MB. For simulation purposes, it is assumed that data sensitivity is defined over record attributes (vertical columns), corresponding to the third approach described in Section 5.1. That means that $\mathcal{F}$ can be partitioned up to 9 different segments where each segment contains the entire vertical column of the file of different sensitivity.

In the simulations, the number of levels k of the hierarchy H is equivalent to the number of file partitions being shared. We also define the total number of different user attributes applied to users across all levels as N. While alternating the values of k and N, we compare the key generation, encryption, and decryption time-costs of all schemes. The experiments include applying the values for $k=\{3,6,9\}$, to each value of $N=\{10,100\}$, resulting in 6 experimental cases. Within each case, it is possible to distribute the user attributes among the levels in numerous ways. For example, when testing the schemes with a hierarchy H for $k=3$ and $N=10$, there are 36 different ways to distribute the $N=10$ user attributes among the $k=3$ levels, excluding the cases of having zero user attributes at any level. A possible way of distributing them could be $\mathbb{A}_{\mathcal{L}_1}=3$, $\mathbb{A}_{\mathcal{L}_2}=3$ and $\mathbb{A}_{\mathcal{L}_3}=4$, where $\mathbb{A}_{\mathcal{L}_1}$ represents the set of user attributes possessed by users in the highest level within H and $\mathbb{A}_{\mathcal{L}_3}$ the set of user attributes possessed by users in the lowest level. As the values of k and N increase, the possible ways of user attribute distribution among all levels increases. Without loss of generality, we assume that the number of attributes for each level follows the uniform distribution in our simulations.

$\mathcal{F}$ is partitioned into k sections, where $\forall F_i \in \mathcal{F}$ and $1 \le i \le k$, each F_i represents one or more full columns of $\mathcal{F}$. Table 4 represents the 9 record attributes distribution of $\mathcal{F}$ into each partition F_i, based on the given value for k in each scenario.

All simulations are performed in consideration of users at the most sensitive (highest) level within the hierarchy. This approach considers the most complicated applications, where a user needs to gain access to the entire file. Fig. 5 summarizes the time expended by the three schemes to generate a private key for the user, encrypt all partitions of $\mathcal{F}$ and decrypt all partitions respectively, in all 6 scenarios. By analyzing the figures, some observations can be derived. The results are summarized in the following subsections.

8.1 Key Generation Time-Cost

To measure the time to generate a private key for a user, the same attribute and level conditions are applied to all three schemes. P-MOD outperforms CP-ABE and FH-CP-ABE in all experimental evaluations. As illustrated in Fig. 5a, the time taken to generate a private key for a user at the highest level in CP-ABE and FH-CP-ABE, is independent of the value of k and remains nearly constant when N is kept constant, for both values of $N=\{10,100\}$ tested.

In comparison, P-MOD reacts differently. When the total number of user attributes is normally distributed among the levels, the time taken to generate a private key by P-MOD is approximately the reciprocal of the value of k multiplied by the equivalent time taken by CP-ABE or FH-CP-ABE to perform the same function. For example, the time

TABLE 4
Attribute Distribution in Each Partition F_i

k	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
3	2	3	4	-	-	-	-	-	-
6	1	1	1	2	2	2	-	-	-
9	1	1	1	1	1	1	1	1	1

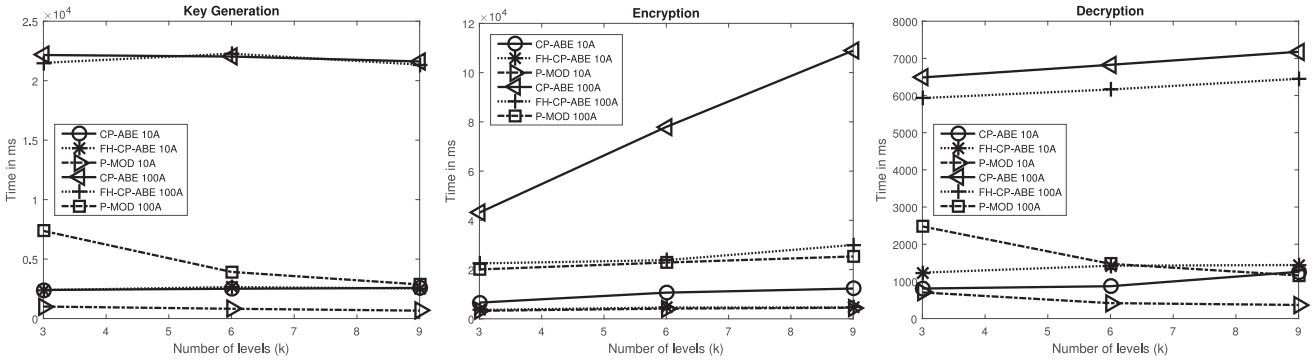

Fig. 5. Performance comparison using the real U.S. Census Income Data [10]: (a) Key generation time, (b) Encryption time, and (c) Decryption time.

taken by CP-ABE and FH-CP-ABE in the case where $k = 6$ and $N = 100$ is approximately 2.2×10^4 seconds. The time taken by P-MOD in the same conditions is approximately 0.35×10^4 seconds, which is nearly $\frac{1}{6}$ times the time-cost of the other schemes. Therefore, the time taken to generate a private key by P-MOD is inversely proportional to the value of k in the hierarchy. This is true for a constant value N with normally distributed attributes.

Another observation that can be made from Fig. 5a is the effect of the values N on the time expended. The time-cost is directly related to the value N and the degree of this proportional increase is different for each scheme as the value N increases.

We define variable δ as the difference in time-cost of two experimental evaluations of the same scheme at $N = 10$ and $N = 100$, while keeping k fixed. In Fig. 5a, consider the simulated values for each scheme at $k = 9$. Both CP-ABE and FH-CP-ABE result in large time-cost changes, approximately $\delta_{\mathrm{CP-ABE}} = 19039$ms and $\delta_{\mathrm{FH-CP-ABE}} = 18821$ms, while P-MOD results in a smaller value, $\delta_{\mathrm{P-MOD}} = 2204$ms. Based on these values, $\delta_{\mathrm{P-MOD}}$ is approximately 11.7 percent of $\delta_{\mathrm{CP-ABE}}$ and $\delta_{\mathrm{FH-CP-ABE}}$, resulting in an approximately 88.3 percent improvement. The significance of this observation can be seen in applications where N is a large value. Efficiency can be gained in time expenditure by utilizing P-MOD, versus CP-ABE and FH-CP-ABE.

8.2 Encryption Time-Cost

The encryption time-cost is the time it takes each scheme to perform the encryption function over all partitions of $\mathcal{F}$. Fig. 5b represents the time expenditure of each scheme under all six experimental scenarios. P-MOD surpasses both CP-ABE and FH-CP-ABE in every experimental case. For example, compare the time duration of the three schemes at $k = 9$ and $N = 100$. The time duration for CP-ABE is approximately 4.3 times of P-MOD to perform encryption. Similarly, FH-CP-ABE is approximately 1.2 times of P-MOD to perform encryption.

All schemes follow a direct proportional pattern as the values k and N increase. These results prove the correctness of the encryption computational complexity analysis presented in Section 7.2.1. The encryption function in all schemes involves a number of f_{G_0} and f_{G_1} operations that are dependent on both the values k and N. However, based on the proposed hierarchical access structure, P-MOD is able to outperform both CP-ABE and FH-CP-ABE. In addition to this, the effect of changing the value N is also illustrated clearly in Fig. 5b. For example, when $k = 9, \delta_{\mathrm{P-MOD}}$ is approximately 21.6 percent of $\delta_{\mathrm{CP-ABE}}$ and approximately 82 percent of $\delta_{\mathrm{FH-CP-ABE}}$.

8.3 Decryption Time-Cost

As previously discussed, the experiments are performed in the perspective of a user that appears at the highest level of the hierarchy. Taking this into account, the decryption time-cost is defined as the time for the user to successfully decrypt all ciphertexts EF_i corresponding to all partitions F_i, if the user possesses the correct set of user attributes. Fig. 5c illustrates the time to perform the decryption function by each scheme. The decryption function of both CP-ABE and FH-CP-ABE both involve e and f_{G_1} operations that are dependent on the values k and N. As these values increase, the decryption time-cost increases linearly for both schemes, proving the correctness of the decryption complexity analysis in Section 7.2.2.

When measuring the decryption time-cost, P-MOD outperforms both CP-ABE and FH-CP-ABE. This is due to the time-cost being inversely proportional to the value of k. The time-cost decreases as the value of k increases while N is kept constant.

The decryption time-cost of P-MOD does not severely increase while the value N changes from 10 to 100. In contrast to this, CP-ABE and FH-CP-ABE are greatly affected, as seen in Fig. 5c. For example, when $k = 9$, $\delta_{\mathrm{P-MOD}}$ is approximately 13.4 percent of $\delta_{\mathrm{CP-ABE}}$ and approximately 15.8 percent of $\delta_{\mathrm{FH-CP-ABE}}$. The decryption time-cost of P-MOD is expected to drop when k increases while keeping the file size constant.

In summary, for a hierarchical organization with many levels, the simulation results show that P-MOD is significantly more efficient at generating keys, encryption, and decryption than that of both CP-ABE and FH-CP-ABE schemes.

9 Conclusion

The numerous benefits provided by the cloud have driven many large multilevel organizations to store and share their data on it. This paper begins by pointing out major security concerns data owners have when sharing their data on the cloud. Next, the most widely implemented and researched data sharing schemes are briefly discussed revealing points of weakness in each. To address the concerns, this paper proposes a Privilege-based Multilevel Organizational

Data-sharing scheme (P-MOD) that allows data to be shared efficiently and securely on the cloud. P-MOD partitions a data file into multiple segments based on user privileges and data sensitivity. Each segment of the data file is then shared depending on data user privileges. We formally prove that P-MOD is secure against adaptively chosen plaintext attack assuming that the DBDH assumption holds. Our comprehensive performance and simulation comparisons with the three most representative schemes show that P-MOD can significantly reduce the computational complexity while minimizing the storage space. Our proposed scheme lays a foundation for future attribute-based, secure data management and smart contract development.

REFERENCES

[1] P. Institute, "Sixth annual benchmark study on privacy and security of healthcare data," Ponemon Institute LLC, Traverse City, MI, 2016, Tech. rep. no. 6.

[2] R. Cohen, "The cloud hits the mainstream: More than half of U.S. businesses now use cloud computing," Apr. 2013. [Online]. Available: http://www.forbes.com, posted on: Jan. 10, 2017

[3] P. Mell and T. Grance, "The NIST definition of cloud computing," Computer Security Division Information Technology Laboratory, MD 20899-8930, 2011.

[4] A. C. OConnor and R. J. Loomis, "2010 economic analysis of role-based access control," National Institute of Standards and Technology, Gaithersburg, MD, RTI Project Number 0211876, vol. 20899, 2010.

[5] A. Elliott and S. Knight, "Role explosion: Acknowledging the problem," in *Proc. Int. Conf. Softw. Eng. Res. Practice*, 2010, pp. 349–355.

[6] E. Zaghloul, T. Li, and J. Ren, "An attribute-based distributed data sharing scheme," in *Proc. IEEE Global Commun. Conf.*, Dec. 9–13, 2018, pp. 1–6.

[7] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.

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

[9] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Trans. In. Forensics Security*, vol. 11, no. 6, pp. 1265–1277, Jun. 2016.

[10] M. Lichman, "UCI machine learning repository," Irvine, CA, 2013.

[11] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 457–473.

[12] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Proc. Theory Cryptography Conf.*, 2011, pp. 253–273.

[13] 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. Security*, 2006, pp. 89–98.

[14] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur. (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[15] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 456–465.

[16] S. Roy and M. Chuah, "Secure data retrieval based on ciphertext policy attribute-based encryption (CP-ABE) system for the DTNs," Lehigh CSE Tech. Rep., Citeseer, 2009.

[17] J. Lai, R. H. Deng, and Y. Li, "Fully secure cipertext-policy hiding CP-ABE," in *Proc. Int. Conf. Inf. Security Practice Experience*, 2011, pp. 24–39.

[18] J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proc. 7th ACM Symp. Inf. Comput. Commun. Security*, 2012, pp. 18–19.

[19] F. Guo, Y. Mu, W. Susilo, D. S. Wong, and V. Varadharajan, "CP-ABE with constant-size keys for lightweight devices," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 5, pp. 763–771, May 2014.

[20] C. Gentry and A. Silverberg, "Hierarchical id-based cryptography," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Security*, 2002, pp. 548–566.

[21] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 735–737.

[22] R. Bobba, H. Khurana, and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," in *Proc. Eur. Symp. Res. Comput. Security*, 2009, pp. 587–604.

[23] Z. Wan, J. Liu, and R. H. Deng, "Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.

[24] H. Deng, Q. Wu, B. Qin, J. Domingo-Ferrer, L. Zhang, J. Liu, and W. Shi, "Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts," *Inf. Sci.*, vol. 275, pp. 370–384, 2014.

[25] J. Li, Q. Wang, C. Wang, and K. Ren, "Enhancing attribute-based encryption with attribute hierarchy," *Mobile Netw. Appl.*, vol. 16, no. 5, pp. 553–561, 2011.

[26] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proc. 21st Annu. ACM Symp. Theory Comput.*, 1989, pp. 33–43.

[27] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptography*, 2011, pp. 53–70.

[28] J. Wang, "ciphertext-policy attribute based encryption java toolkit," 2015. [Online]. Available: https://github.com/junwei-wang/cpabe

[29] A. De Caro and V. Iovino, "JPBC: Java pairing based cryptography," in *Proc. IEEE Symp. Comput. Commun.*, 2011, pp. 850–855.

Ehab Zaghloul received the BS and MSc degrees in computer engineering from Arab Academy for Science and Technology (AAST), Alexandria, Egypt, in 2012 and 2015 respectively. He is currently working toward the PhD degree in electrical and computer engineering at Michigan State University (MSU), East Lansing. His research interests include applied cryptography, secure and private data sharing, distributed systems, and blockchain.

Kai Zhou (S'16) received the BS degree in electrical engineering from Shanghai Jiao Tong University, China, in 2013, and the PhD degree in electrical and computer engineering from Michigan State University, in 2018. His research interests include applied cryptography, data security and privacy, adversarial social network analysis, and game theory. He is currently a postdoctoral research associate in the department of computer science and engineering, Washington University in St. Louis. He is a student member of the IEEE.

Jian Ren (SM'09) received the PhD degree in electrical engineering from Xidian University, China. Currently, he is an associate professor with the Department of Electrical and Computer Engineering Michigan State University. Prior to joining MSU, Dr. Ren was the Leading Secure Architect at Avaya Lab (2000-2002), Bell Lab (1998-2000) and Racal Datacom (1997-1998). His research interests include network security, distributed storage, cloud computing, big data security and privacy, and wireless communications. He received the National Science Foundation (NSF) CAREER award in 2009. He served as the TPC chair of ICNC'17 and General chair of ICNC'18. He serves as the Executive Chair of ICNC'19. He serves as an associate editor of ACM Transactions on Sensor Networks. He is a senior member of the IEEE.