




Research Article

Efficient Hierarchical and Time-Sensitive Data Sharing with User Revocation in Mobile Crowdsensing

Jiawei Zhang ¹, Jianfeng Ma,¹ Teng Li ¹ and Qi Jiang ^{1,2,3}

¹School of Cyber Engineering, Xidian University, Xi'an, China

²Communication Research Centre, Peng Cheng Laboratory, Shenzhen, China

³Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, China

Correspondence should be addressed to Teng Li; litengxidian@gmail.com

Received 18 November 2020; Revised 5 December 2020; Accepted 14 February 2021; Published 27 February 2021

Academic Editor: Athanasios V. Vasilakos

Copyright © 2021 Jiawei Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recently, cloud-based mobile crowdsensing (MCS) has developed into a promising paradigm which can provide convenient data sensing, collection, storage, and sharing services for resource-constrained terminators. Nevertheless, it also inflicts many security concerns such as illegal access toward user secret and privacy. To protect shared data against unauthorized accesses, many studies on Ciphertext-Policy Attribute-Based Encryption (CP-ABE) have been proposed to achieve data sharing granularity. However, providing a scalable and time-sensitive data-sharing scheme across hierarchical users with compound attribute sets and revocability remains a big issue. In this paper, we investigate this challenge and propose a hierarchical and time-sensitive CP-ABE scheme, named HTR-DAC, which is characteristics of time-sensitive data access control with scalability, revocability, and high efficiency. Particularly, we propose a time-sensitive CP-ABE for hierarchical structured users with recursive attribute sets. Moreover, we design a robust revocable mechanism to achieve direct user revocation in our scheme. We also integrate verifiable outsourced decryption to improve efficiency and guarantee correctness in decryption procedure. Extensive security and performance analysis is presented to demonstrate the security requirement satisfaction and high efficiency for our data-sharing scheme in MCS.

1. Introduction

As a promising paradigm, the adoption of the Mobile Crowdsensing (MCS), which can take advantage of individual resource-limited mobile terminals to sense, collect, and analyze data rather than massive static sensors deployment, grows rapidly, and a large number of mobile users are willing to enjoy the convenient services of MCS, such as smart health, smart cities, and intelligent transportation [1]. These days, the wide spread of Internet of Things [2] and the emerging 5G communication network which can support fast speed and massive access [3] also facilitate the application of MCS. Thus, the cloud-based mobile crowdsensing is proposed for optimizing data collection and minimizing the cost in both sensing and reporting [4]. As shown in Figure 1, the sensing data gathered by mobile terminals (e.g., laptops, vehicles, and IoT devices) are transmitted to cloud

via 5G communications network or even satellites for data collection, storage, and sharing. However, the sensitive and private in such sensing data may be breached in untrusted cloud, which may prevent the further participation of mobile users, especially when their data are illegally accessed.

There have been many data access control solutions, such as access control list (ACL), Bell-La Padula (BLP), BiBa, and role-based access control (RBAC). Nevertheless, all of these solutions suffer from different drawbacks, such as inflexibility, high computation complexity, and coarse-grained access control [5]. In recent years, prospective ciphertext-policy attribute-based encryption (CP-ABE) is proposed to control data access based on attributes to achieve flexibility and fine granularity [6–9]. In CP-ABE, data producers are required to designate a specific access policy to obtain the ciphertext of their data before outsourcing. When accessing these shared contents, users need

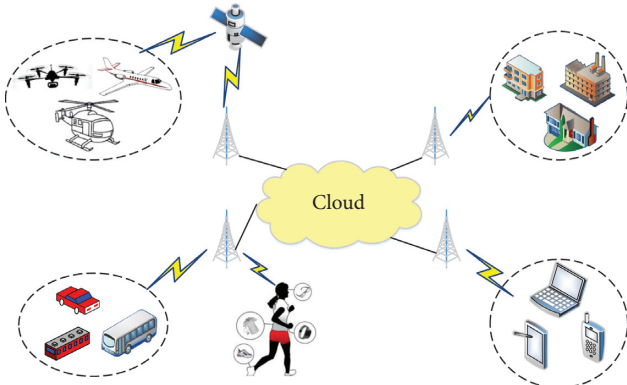


FIGURE 1: The application scenario of cloud-based mobile crowdsensing.

to decrypt the data with their secret keys and recover the plaintext if and only if they are authorized. However, these solutions cannot be directly utilized in the applications of time-sensitive data sharing across hierarchical users with revocability and recursive attribute set as some challenges remain unsolved.

Let us take an example of the smart health record (SHR) sharing system [10]. In general, the mobile users in the SHR system are usually in hierarchical structure and large scale which need a scalable architecture. In the meantime, these users may have the following attribute set:

$$\begin{aligned} &\{\text{Org: Central Hospital, Dept: Cardiac Surgery}\}, \\ &\{\text{Role: dean, Level: intermediate}\}, \quad (1) \\ &\{\text{Role: surgeon, Level: senior}\}. \end{aligned}$$

This recursive attribute set indicates the user that holds corresponding recursive key structure is both an intermediate dean and a senior surgeon of cardiac surgery in central hospital. Furthermore, the SHRs contain a lot of sensitive and private data, such as social security number, health condition, and disease, which need to be protected with fine-grained access control. Besides, many SHRs are time sensitive, that is, the data in high confidentiality are generally accessed by part of users (e.g., the hospital director) at the first time and following another part of users in a timed sequence. In addition, the mobile users in the system are dynamically changing, which requires an effective revocation mechanism. Therefore, it is the key to design a data-sharing scheme that deals with time-sensitive data and supports hierarchical and revocable users with recursive key structure efficiently in MCS.

Although these days, the authors in [11] have solved the issues of hierarchical users with compound key structure in CP-ABE by combining the scheme of HIBE and ASBE, and the scheme in [12] proposed a time-based ABE by introducing time-releasing encryption (TRE) into CP-ABE. However, these schemes cannot support user dynamicity. The work in [13] proposed an effective direct user revocation, but it cannot achieve effective security as the revoked users can neglect the related mechanism in decryption to recover the plaintext. Thus, the security cannot be guaranteed. Moreover, all the above

schemes incur high computation cost, which cannot adapt to the environment of MCS with resource-limited mobile terminals. Comprehensively, how to design an efficient access control scheme towards time-sensitive data with fine granularity, hierarchical users holding recursive attribute set, and user revocability simultaneously is still a big challenge.

In this paper, inspired by the above scenario and the challenges in existing related work, a hierarchical and time-sensitive revocable data access control (HTR-DAC) scheme is put forward. Specifically, we present a scalable CP-ABE scheme for time-sensitive data-sharing service across hierarchical users that holds compound attribute set and can be revoked directly if they conduct malicious activities for illicit purpose. Besides, our scheme can improve efficiency and guarantee the correctness of decryption. Our main contributions are three folds:

- (i) We put forward a scalable sharing scheme for time-sensitive data across hierarchical users with structured keys. The users' keys are issued by the delegation of the domain authorities they belong to. Moreover, a cloud user can access ciphertexts if and only if he is authorized with correct attribute sets and exposed time trapdoors, which ensures the fine-grained and time-sensitive data access control.
- (ii) We design a new approach to realize user revocability and high efficiency of our scheme. We integrate the verifiable outsourced decryption and a direct user revocation mechanism into our scheme. The approach can not only greatly improve the efficiency in decryption but also guarantee the correctness of decryption result.
- (iii) We present the security analysis for our proposal to show the scheme achieves its security goals. We also implement our scheme and conduct extensive experimental simulations with performance evaluations to display our proposal with better efficiency and practicality.

Our paper is outlined as follows. Section 2 reviews some related work, and Section 3 gives several relevant notations together with definitions. The system model, adversary model, and design goals are shown in Section 4 following the system definition and of our scheme with its security model presented in Section 5. Based on this, we show the concrete description for our scheme in Section 6. Section 7 presents a thorough analysis for security, and Section 8 displays the performance evaluation for our scheme. In the end, we make a summary for our work in Section 9.

2. Related Work

This section reviews some related works on attribute-based encryption (ABE) technique.

As a prospective technique, ABE was first introduced in [14] for access control in fine granularity. Later, Goyal et al. [15] divided ABE into two types: CP-ABE and KP-ABE (key-policy ABE). In the former, data owner can flexibly designate

the access policy for ciphertext. Thus, we focus on this technique for data access control in MCS. Subsequently, a great many studies were dedicated on CP-ABE, such as large universe CP-ABE [16], multiauthority CP-ABE [17], traceable CP-ABE, and revocable CP-ABE [18].

2.1. Hierarchical ABE and Time-Release Encryption. Currently, many ABE schemes only support single authority for managing users' secret keys. While, in a large scale, it is not suitable to fulfill the large number of user key management tasks. To find out a solution, the researchers in [19] raised the first hierarchical ABE scheme with the idea of hierarchical identity-based encryption. The scheme in [11] solves the problem of hierarchical user structure with key structure by combining the concept of HABE and ASBE (ABE with the attribute set). Recently, Wang and Gao [20] propose a CP-HABE scheme to solve the user privacy in Bitcoin Financial Systems. However, the scheme incurs continuous auxiliary input leakage. To solve the problem, the scheme [21] proposed a leakage-resilience CP-HABE scheme by introducing continuous leakage-resilience mechanism into the CP-ABE scheme. Later on, the work in [8] proposes a lightweight CP-HABE scheme to support flexibility and scalability and user revocation.

As many applications in cloud is time-sensitive, time-release encryption (TRE) was first introduced in [22] which introduces a trust time agent to release the access right at a specific time point uniformly. Later, many schemes [23, 24] have integrated TRE to different cryptographic schemes to adapt to different application scenarios. These days, some studies try to combine TRE with ABE. The scheme in [25] proposes a time-sensitive data-sharing scheme in cloud. However, it merely supports coarse-grained access control and incurs a heavy burden for cloud users. Then, the work in [26] proposes a CP-ABE-based data access control scheme with time domain by combining the attribute set of users and access time inspired by [27]. Although the scheme eliminates much of work for data owner, it brings extra overhead for authority. Recently, the work in [12] proposes a time-sensitive CP-ABE scheme with high efficiency and fine-grained access control.

2.2. Revocable ABE and Verifiable Outsourced Computing. Current revocable schemes can be classified as user-level and attribute-level ones in which the former involves direct revocation and indirect revocation. In direct user revocation schemes [13, 28–30], the data producer can integrate revocation list into ciphertext and requires no key updates in revocation. While, the indirect user revocation schemes [30–32] periodically update nonrevoked users' secret key without data producers knowing the revocation list. In particular, the work in [13] proposes a novel direct user revocation CP-ABE scheme. However, the scheme suffers from a low security as, in their revocation mechanism, any data user including revoked and nonrevoked user can recover the plaintext by skipping the revocation procedure in decryption. Later, the proposals in [28–30] extend the direct user revocation mechanisms, but they fail to solve the

problem in [13] and the low efficiency caused by direct revocation.

As the decryption cost in ABE is very high, many researchers have studied on this topic. The scheme in [33] introduced outsourced computing into ABE scheme to improve the decryption efficiency. Later, many studies are in this direction. In these schemes, the ciphertexts can be translated to a constant-sized ElGamal-typed ones and cloud gains nothing about the content. However, this kind of schemes cannot ensure the ciphertexts are decrypted by the cloud correctly. To overcome this flaw, Lai et al. [34] raise an outsourced decryption scheme with verifiability that ensures the decryption is executed correctly. Then, the scheme in [35] proposes another verifiable scheme by introducing token mechanism which eliminates the pairing computations. The work in [36] proposes a verifiable scheme with exculpability, while the scheme incurs heavy computational overhead. To improve efficiency, the scheme in [37] introduces a verification code for correctness of decryption with high efficiency.

3. Preliminaries

This section presents several relevant notions and definitions employed in our paper.

3.1. Notations. We summarize several notations used in our scheme as well as their descriptions in Table 1.

3.2. Access Structure and Key Structure

Definition 1 (access structures, see [38]). Suppose $\{L_1, \dots, L_n\}$ is a party set. One of the collection $L \subseteq 2^{\{L_1, \dots, L_n\}}$ is considered to be monotone if $\forall M, N: M \in L$ and $M \subseteq N$; then, $N \in L$. An access structure that is monotone is defined as one of the nonempty subsets L of $\{L_1, \dots, L_n\}$, i.e., $(L \subseteq 2^{\{L_1, \dots, L_n\}}) / \{\emptyset\}$. The elements in L are defined as authorized sets and the other sets are defined as unauthorized sets. Without loss of generality, we can describe users with their attribute set.

Definition 2 (key structure, see [11]). As for most of the practical situations, each user's attribute set \mathcal{A} is organized in a tree-like recursive set structure in which each element of \mathcal{A} is an attribute set or a single attribute. As a result, the corresponding key structure of each user is similar to the attribute structure. The depth of \mathcal{A} (also, the key structure) is the number of levels for the recursive set. Suppose a 2 – depth key structure; the level 1 members can be either attribute sets or attributes, while the level 2 members may only be attributes. The key structure in Figure 2 is of depth = 2, denoting that the user is both an intermediate dean and a senior surgeon in cardiac surgery in central hospital. Moreover, in the key structure, each set is assigned a unique label which is a unique index of the recursive set, and each attribute in an element set is labeled by the set's index together with its name. Suppose a key structure of depth = 2 with m attribute sets is represented by $\mathcal{A} = \{A_i\}_{0 \leq i \leq m}$, where A_0 is the set at depth 1

TABLE 1: Notation description.

Notations	Descriptions
$[n]$	$\{1, \dots, n\}$
G_0, G_1	Two multiplicative cyclic groups
RL	User revocation list
SK_u	Secret attribute key for DU
TK_u	Transformation key pare for DU
\mathcal{A}	Key structure of DU
A_i	i th attribute set of DU
T	Access policy tree
X, \bar{X}	Leaf node set and translating node set of T
Tok	Time token released by TA
TD_x	The trapdoor associated with node $x \in T$
VC	Verification code for ciphertext
CT_s	The ciphertext encrypted with symmetric algorithms
CT	The whole ciphertext

and other sets A_i are at depth 2. As in Figure 2, $A_0 = \{\text{Org: Central Hospital, Dept: Cardiac Surgery}\}$, $A_1 = \{\text{Role: dean, Staff level: intermediate}\}$, and $A_2 = \{\text{Role:surgeon, Staff level: senior}\}$. Then, the attribute "Role:dean" can be denoted by $(1, \text{Role:dean})$.

3.3. Access Policy Tree

Definition 3 (access tree, see [39]). Similar to [39], suppose R is a policy tree with each node $x \in R$, where we use a threshold gate to represent nonleaf nodes and a leaf node is an attribute $\text{att}(x)$. As to a threshold gate $x \in R$, we use $\text{num}(x)$ which is the number of children and the threshold value $\text{th}_x \in [1, \text{num}(x)]$ to depict it. Specifically, if $\text{th}_x = 1$, it is an OR gate, and if $\text{th}_x = \text{num}(x)$, it is an AND gate. If $x \in T$ is a leaf node, its threshold value is $\text{th}_x = 1$.

Moreover, suppose $r \in R$ is the root node. If $x \in T$ is a nonleaf node, $\text{child}(x)$ is a collection of its children and $\text{parent}(x)$ denotes the parent node of x . Thus, we can infer that $|\text{child}(x)| = \text{num}(x)$. We use the function $\text{index}(x)$ to signify the unique index value of each node $x \in T$.

Access Tree Satisfaction. Suppose R is an access tree rooted from node r ; then, we use R_x to denote a subtree rooted from node $x \in R$. Here, we define $R_x(\mathcal{A}) = 1$ when and only when \mathcal{A} (a attribute set) is satisfactory to the subtree R_x , that is, when a leaf node x has $\text{att}(x) = \text{att}_i \in \mathcal{A}$, then $R_x(\mathcal{A}) = 1$, and when a nonleaf node has $\forall z_x \in \text{child}(x)$, the number of z satisfying $R_z(\mathcal{A}) = 1$ exceeds th_x , $R_x(\mathcal{A}) = 1$.

Moreover, consider the key structure $\mathcal{A} = \{A_0, A_1, \dots, A_m\}$ with depth = 2, where A_i denotes the i th attribute set and i ($0 \leq i \leq m$) denotes the index of each set. If \mathcal{A} satisfies R , then $R(\mathcal{A})$ will return a nonempty label set S . Here, $R(\mathcal{A})$ is also computed recursively as before and \mathcal{A} satisfies access tree T if and only if it consists of at least one set A_i ($0 \leq i \leq m$) having all elements required to be satisfactory to R . Generally, combining attributes in different attribute sets of \mathcal{A} that are satisfactory to R is impossible without translating nodes in R . Given a is a translating node $x \in R$, the attributes needed to meet R_x belonging to different sets in \mathcal{A} can be combined to make the predicate with $\text{parent}(x)$ hold.

3.4. Cryptographic Background

Definition 4 (bilinear maps, see [39]). We consider two p -ordered G_0 and G_1 groups that are multiplicative cyclic, where p is a prime. ε and ϵ are two generators of group G_0 . If $\hat{e}: G_0 \times G_0 \rightarrow G_1$ satisfies the following properties,

- (1) Bilinearity: $\hat{e}(\varepsilon^a, \varepsilon^b) = \hat{e}(\varepsilon, \varepsilon)^{ab}$, $\forall a, b \in Z_p$, $\varepsilon, \epsilon \in G$
 - (2) Nondegeneracy: $\hat{e}(\varepsilon, \varepsilon) \neq 1_{G_1}$, $\hat{e}(\varepsilon, \varepsilon)$ is a generator of G_1
 - (3) Computability: $\hat{e}(\varepsilon, \varepsilon)$ is efficiently computable for all $\varepsilon, \epsilon \in G_0$
- then, we call it a bilinear map.

Definition 5 (decisional bilinear Diffie–Hellman (DBDH) assumption, see [40]). Given two cyclic groups E and F and their orders are both the prime p . Suppose a generator $h \in E$ and a bilinear mapping $\hat{e}: E \times E \rightarrow F$. The DBDH problem is defined to find out the difference between $\hat{e}(h, h)^{\text{cdm}}$ and $\hat{e}(h, h)^v$ on inputting the tuple (h, h^c, h^d, h^m) , where $c, d, m, v \in_R Z_p$.

It is considered that DBDH assumption holds when no probabilistic polynomial time (PPT) adversaries can deal with the DBDH problem whose advantages are nonnegligible.

4. System Model

We present the system and adversary model as well as corresponding design goals for our proposal in this section.

4.1. System Model. Figure 3 shows the model of our system for HTR-DAC. It consists six entities: Cloud Service Provider (CSP), Trusted Authority (TA), several Domain Authorities (DAs), Data User (DU), and Data Owner (DO), which are described below:

- (i) CSP: the CSP has unlimited resources, such as computation and storage resources. It can provide cloud users with centralized service, e.g., storage service, data-sharing service, and outsourced computing.
- (ii) TA: this entity takes charge of initializing the system with parameters and the master key for the whole system. It also supports user authentication in its domain and enrollment of domain authorities of top level.
- (iii) DAs: DAs are in hierarchical structure involving several domain authorities of the top level and authorities of the low level. Each domain authority is responsible for managing the lower level domain authority, i.e., authenticating and generating master keys for the lower-level authorities in its domain. Moreover, each domain authority also takes charge of assigning secret attribute keys and transformation keys for cloud users.

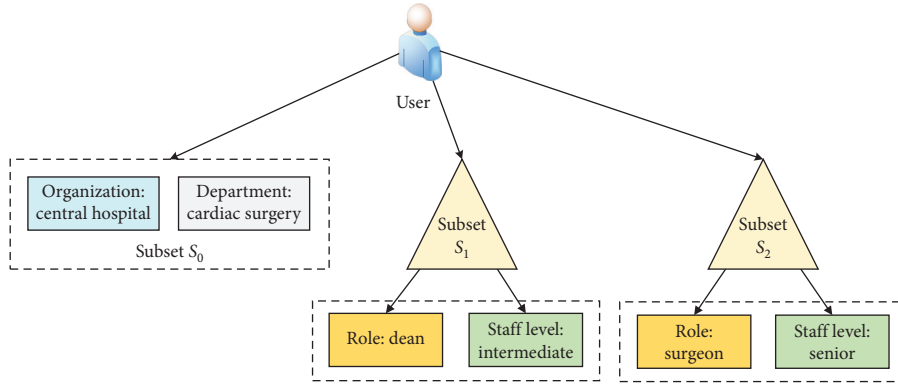


FIGURE 2: An example of key structure.

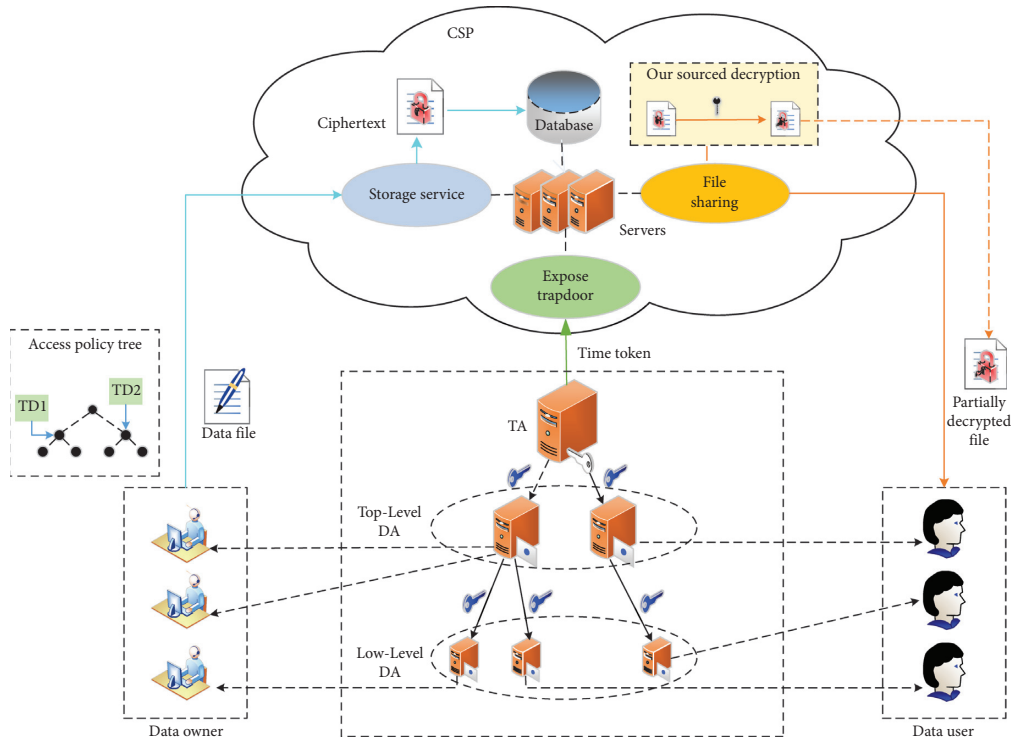


FIGURE 3: The system model of our proposal.

- (iv) DO: the DO uploads his important information through all kinds of mobile devices to CSP. Before outsourcing these data, DO needs to encrypt the whole data for confidentiality and unauthorized access prevention by designating specific policy.
- (v) DU: the DU downloads the data from CSP according to his requirements and decrypts the data if he has enough rights. He then recovers the correct plaintext after verification. If any DU has malicious activities, it will be revoked directly.

Then, we depict an overview for our HTR-DAC scheme based on the above system model involving the following four phases:

- (i) *Initialization*. In this phase, TA initiates the whole system by generating the corresponding public key

and master key. All entities are able to get the system public key.

- (ii) *Enrollment*. In this phase, TA authenticates the top-level domain authorities and assigns them with the master keys. Recursively, each domain authority authenticates the domain authorities in the lower level within its domain and also creates master keys for them. Moreover, each domain authority manages cloud users in its domain by generating attribute keys and transformation keys for them after successful user authentication.
- (iii) *Encryption*. In this phase, DO encrypts the sensitive and important data using symmetric encryption algorithm before uploading them to the CSP. Moreover, DO designates an access policy for the ciphertexts outsourced in CSP.

- (iv) *Decryption*. In this phase, DU requests data files that he needs from CSP. After outsourced decryption, CSP returns the DU with the partially decrypted ciphertexts. Then, the DU verifies if the outsourced decryption is correct and recovers the plaintext according to the result of verification.

4.2. Adversary Model. In our proposal, the TA, DAs, and DO are regarded as the fully trusted entities, while the CSP is considered to be untrusted which may intentionally leak or modify the content of sensitive data. Moreover, some unauthorized DU may illegally access the sensitive data which will break the data security and privacy. Besides, some DU may conduct malicious activities for extrainternet revenues.

According to the ability of adversary, we consider classifying the attacks into two types:

- (1) Type-A attack: the adversary has insufficient privileges for data access, even he is not revoked or arrives at release time
- (2) Type-B attack: the adversary has enough rights but he conducts accesses before relevant releasing time arrives

Focusing on these attacks, we take the following security requirements into consideration:

- (i) *Data confidentiality*: the data generated by DO and outsourced in CSP should be secured against illegal sniffing and eavesdropping by attackers. Moreover, the outsourced data should also not be accessed by any malicious access without enough access rights.
- (ii) *Collusion-resistance*: the scheme should prevent users anyone of whom is not authorized from colluding through combining their secret keys to access the shared data.
- (iii) *Revocability*: any malicious cloud users that conduct malicious activities should be revoked and the revoked users should get rid of all the access rights.

In addition, the following aspects are also in our consideration:

- (i) *Efficiency*: as the resource-limited mobile devices are utilized in cloud-based mobile crowdsensing, it is preferable for DU to outsource the high-computational burden in decryption to CSP to improve efficiency
- (ii) *Verifiability*: due to the untrusted CSP, DU should have the ability to check if the result of outsourced decryption procedure is correct

5. Security Model

This section presents the formal definition and the security model for our proposed scheme.

5.1. Definition of Our HTR-DAC Scheme. We propose a CP-ABE scheme in which the authorities and users are in the hierarchical structure each of which manages the domain authorities at lower level and cloud users in its charge. Moreover, the attribute set of each user is formed in recursive attribute sets. Besides, the scheme can revoke users in a direct way. Specifically, the algorithms in our scheme are as follows:

- (i) *Setup*(λ, d) \longrightarrow {PK, MSK₀}: this procedure is executed by TA to initialize whole system. Given system security parameter λ and d -depth of key structure, it outputs the system public parameters PK and the master key MSK₀.
- (ii) *Create Top DA*(PK, MSK₀, \mathcal{A}) \longrightarrow MSK_{*i*}: the procedure is executed by TA. Given PK, MSK₀, and the compound attribute set \mathcal{A} of authority DA_{*i*} which is at the top level, it outputs the master key MSK_{*i*} for DA_{*i*}.
- (iii) *Create Entity*(MSK_{*i*}, ID, $\overline{\mathcal{A}}$) \longrightarrow MSK_{*i+1*}/SK_{*u*}: the procedure is run by top level domain authority DA_{*i*} or the lower level domain authority managing the cloud user. On inputting MSK_{*i*} of DA_{*i*} and the compound attribute set $\overline{\mathcal{A}}$ of the next-level domain authority DA_{*i+1*} or a cloud user ID, it returns the master key MSK_{*i+1*} for domain authority DA_{*i+1*} or the secret attribute key SK_{*u*} for cloud user ID.
- (iv) *TKey Gen*(PK, SK_{*u*}) \longrightarrow TK_{*u*}: the procedure is executed by the corresponding domain authority to create transformation key pair for the cloud user. Given PK and the secret attribute key of the cloud user, it outputs the transformation key pair TK_{*u*}.
- (v) *Encrypt*(PK, M, T, RL) \longrightarrow CT: this algorithm is run by DO to generate ciphertexts for fine-grained access control. It takes the system public parameters PK, the message M to be encrypted, the designated access policy tree T , and user revocation list R as input and returns the ciphertext CT of M .
- (vi) *Token Gen*(PK, t) \longrightarrow Tok_{*t*}: the procedure is executed by TA to create time token at different fixed time points orderly. On inputting the system public parameters PK and the time point t , the algorithm outputs the time token Tok_{*t*} of the time point t .
- (vii) *Trap*(PK, Tok_{*t*}) \longrightarrow TD_{*x*}[']: the procedure is run by CSP to expose the trapdoor in the access policy tree T of ciphertexts stored in it. Given the system public parameters and time token Tok_{*t*}, the algorithm outputs the exposed trapdoor TD_{*x*}['].
- (viii) *Decrypt*_{OUT}(PK, TPK_{*u*}, CT) \longrightarrow CT': the procedure is executed by CSP to partially decrypt the ciphertext. Given PK, the public transformation key TPK_{*u*} of DU, and the ciphertext CT to be decrypted, it outputs the partially decrypted ciphertext CT'.

- (ix) $\text{Decrypt}_U(\text{PK}, \text{TSK}_u, \text{CT}') \rightarrow M$ or null: this algorithm is executed by DU to recover the plaintext of the ciphertext. On inputting PK, the secret transformation key TSK_u of DU, and CT' denoting the partially decrypted ciphertext, the algorithm returns the plaintext M .
- (x) $\text{Dec Verify}(\text{PK}, R^*, \text{VC}, \text{CT}_s) \rightarrow \text{True/False}$: the procedure is executed by DU to verify if the outsourced decryption is correct. Given the system public parameters PK, the recovered random R^* , verification code VC, and the symmetric ciphertext CT_s , the algorithm checks if the recovered random is correct, i.e., if CSP correctly executes outsourced decryption. Finally, it outputs the result True/False of verification.

5.2. Security Model. Here, we describe the IND-CPA security model for our HTR-DAC scheme corresponding to the attacks described before and conduct a selective security game between an adversary \mathcal{A} and a challenger \mathcal{C} specified as follows:

Init: \mathcal{A} sends a challenge access policy tree \mathcal{T} and user revocation list RL to \mathcal{C} .

Setup 1: \mathcal{C} executes the Setup algorithm of our scheme and outputs the public parameters to \mathcal{A} .

Phase 1: \mathcal{A} issues a polynomial number of queries $\{q_i\}_{i \in [m]}$, where q_i belongs to the following queries:

- (1) SK query: \mathcal{A} requests \mathcal{C} for secret key with an identity $\text{ID} \notin \text{RL}$ and a recursive attribute set \mathcal{A} that is unsatisfactory to \mathcal{T} before arriving at a time point t_1 . As a response, \mathcal{C} outputs the secret key and publishes a series of time tokens before t_1 and returns them to \mathcal{A} .
- (2) TK query: \mathcal{A} issues queries for transformation key similar to that in SK Query. \mathcal{C} executes $T\text{KeyGen}$ to generate transformation key pairs and send it to \mathcal{A} .

Challenge: \mathcal{A} finishes the above phase and issues two equal-length data M_0 and M_1 to \mathcal{C} . Then, \mathcal{C} randomly picks a bit $\epsilon \in [0, 1]$ and encrypts M_ϵ according to \mathcal{T} and ID and sends it to \mathcal{A} .

Phase 2: it is similar to Phase 1 and before a later time point $t_2 > t_1$.

Guess: \mathcal{A} publishes his guess ϵ' for ϵ . If $\epsilon' = \epsilon$, he wins the security game. The advantage of \mathcal{A} is defined as $\text{Adv}_{\mathcal{A}} = |\Pr[\epsilon' = \epsilon] - (1/2)|$.

Definition 6. A HTR-DAC scheme is indistinguishable against chosen-plaintext attack (CPA) if all probabilistic polynomial adversaries cannot break the security game.

6. Proposed HTR-DAC Scheme

This section describes the concrete construction of our proposal. In particular, our scheme involves the following algorithms: Setup, Create Top DA, Create Entity, TKey Gen,

Encrypt, Token Gen, Trap, $\text{Decrypt}_{\text{OUT}}$, Decrypt_U , and Dec Verify, which are described in detail below.

6.1. Initialization Phase

- (i) $\text{Setup}(\lambda, d) \rightarrow \{\text{PK}, \text{MSK}_0\}$: on inputting the security parameter λ and the depth of the key structure d , the algorithm generates two multiplicative bilinear groups G_0 and G_1 of prime order p with a generator g of bilinear group G_0 and a bilinear map $\hat{e}: G_0 \times G_0 \rightarrow G_1$ and selects random numbers $\alpha, \beta_i, \gamma \in Z_p$, where $i \in [d]$ which is used to represent the depth of key structure. Here, we take $d = 2$ as an example. The algorithm chooses a probabilistic symmetric encryption scheme (Enc, Dec) from a binary string to Z_p^* and selects collision-resistant hash functions $H_0: G_1 \rightarrow \{0, 1\}^{l_1}$, $H_1: \{0, 1\} \rightarrow G_0$, $H_2: G_1 \rightarrow Z_p^*$, and $H_3: \{0, 1\}^* \rightarrow \{0, 1\}^{l_2}$, where l_1 and l_2 are the output length of hash function H_0 and H_3 , respectively. Then, the algorithm sets the time format F_T . Next, it computes and outputs the system public key and master key as follows:

$$\begin{aligned} \text{PK} &= (G_0, G_1, \hat{e}, g, \hat{e}(g, g)^\alpha, h_1 = g^{\beta_1}, h_2 = g^{\beta_2}), \\ \eta_1 &= g^{(1/\beta_1)}, \eta_2 = g^{(1/\beta_2)}, \delta = g^\gamma, H_0, H_1, H_2, H_3, \\ \text{MSK}_0 &= (g^\alpha, \beta_1, \beta_2, \gamma). \end{aligned} \quad (2)$$

Finally, TA publishes PK and stores MSK_0 locally in secret.

6.2. Domain/User Enrollment Phase. In this phase, TA invokes the algorithm Create Top DA to generate the master key for a new valid top domain authority DA_i after receiving the request to take part in the system from DA_i . Subsequently, DA_i can manage the next-level domain authorities or users within its domain by calling the algorithm Create Entity. As to a cloud user, TA also takes charge of the generation of transformation keys for users after they get their secret attribute keys from their domain authority:

- (i) $\text{Create Top DA}(\text{PK}, \text{MSK}_0, \mathcal{A}) \rightarrow \text{MSK}_i$: on receiving the request from a domain authority at top level DA_i to take part in the system after being checked by TA, the algorithm generates master key for DA_i by selecting a random number $r_\Delta \in Z_p$. Then, it picks $r_{\Delta, i} \in_R Z_p$ according to $A_i \in \mathcal{A}$, where A_i is each attribute set. Moreover, the algorithm picks $r_{\Delta, i, j} \in_R Z_p$ according to $\text{att}_{i, j} \in A_i$, where $\text{att}_{i, j}$ is each attribute and $0 \leq i \leq m$ and $1 \leq j \leq n_i$. The algorithm computes the master key for DA_i as follows:

$$\text{MSK}_i = \begin{pmatrix} \mathcal{A}, D_0 = g^{(\alpha + r_\Delta/\beta_1)}, \forall \text{att}_{i, j} \in A_i: \\ D_{i, j} = g^{r_{\Delta, j}} \cdot H_1(\text{att}_{i, j})^{r_{\Delta, i, j}}, D'_{i, j} = g^{r_{\Delta, i, j}} \\ \forall A_i \in \mathcal{A}: E_i = g^{(r_\Delta + r_{\Delta, i}/\beta_2)} \end{pmatrix}. \quad (3)$$

In the master key of a top-level domain authority, E_i is used to translate $r_{\Delta,i}$ of A_i to r_{Δ} of A_0 at the translating node. Meanwhile, E_i and $E_{i'}$ can translate $r_{\Delta,i'}$ to $r_{\Delta,i}$ by computing $(E_i/E_{i'})$.

- (ii) Create Entity ($MSK_i, I, D, \overline{\mathcal{A}}$) \longrightarrow MSK_{i+1}/SK_u : the algorithm is used to generate master key for a new domain authority or the attribute secret key of a user. Given the master key $MSK_i = (\overline{\mathcal{A}}, D_0, D_{i,j}, D'_{i,j})$, for $0 \leq i \leq m, 1 \leq j \leq n, E_i$, for $0 \leq i \leq m$ of DA_i , the algorithm picks $\bar{r}_{\Delta} \in_R Z_p$ for the user or domain authority, $\bar{r}_{\Delta,i} \in_R Z_p$ for each attribute set $A_i \in \overline{\mathcal{A}}$, and $\bar{r}_{\Delta,i,j} \in_R Z_p$ for each attribute $att_{i,j} \in A_i$. For different kinds of entity, the algorithm generates corresponding key for the entity.

- (1) If the entity is a domain authority (i.e., DA_{i+1}), the algorithm generates the master key MSK_{i+1} for DA_{i+1} as follows:

$$\begin{aligned} MSK_{i+1} &= \left(\overline{\mathcal{A}}, \overline{D}_0 = D_0 \cdot \eta_1^{\bar{r}_{\Delta}}, \forall att_{i,j} \in A_i \right), \\ \overline{D}_{i,j} &= D_{i,j} \cdot g^{\bar{r}_{\Delta,i}} \cdot H_1(att_{i,j})^{\bar{r}_{\Delta,i,j}}, \\ \overline{D}'_{i,j} &= D'_{i,j} \cdot g^{\bar{r}_{\Delta,i,j}}, \\ \forall A_i \in \overline{\mathcal{A}}: \overline{E}_i &= E_i \cdot \eta_2^{\bar{r}_{\Delta} + \bar{r}_{\Delta,i}}. \end{aligned} \quad (4)$$

- (2) If the entity is a user with identity ID, the algorithm generates the attribute secret key SK_u for the user as follows:

- (iii) TKey Gen (PK, SK_u) \longrightarrow TK_u : TA selects $z \in_R Z_p^*$ as the secret transformation key TSK_u of the user and computes the public transform key $TPK_u = \left\{ \overline{\mathcal{A}}, K_0, K_1, \{K_{i,j}, K'_{i,j}\}_{att_{i,j} \in A_i}, \{\widehat{E}_i\}_{A_i \in \overline{\mathcal{A}}} \right\}$, where

$$\begin{aligned} K_0 &= D_0^{(1/z)}, K_1 = D_1^{(1/z)}, \\ \forall att_{i,j} \in A_i: K_{i,j} &= D_{i,j}^{(1/z)}, K'_{i,j} = D'_{i,j}{}^{(1/z)}, \\ \forall A_i \in \overline{\mathcal{A}}: \widehat{E}_i &= E_i^{(1/z)}. \end{aligned} \quad (6)$$

Finally, the algorithm outputs the transformation key pair $TK_u = (TPK_u, TSK_u)$ for the cloud user who keeps the TSK_u secret and publishes TPK_u .

6.3. Encryption Phase

- (1) Encrypt (PK, M, T, RL) \longrightarrow CT: on inputting PK, the data M to be encrypted, the designated access policy tree T , and the revocation list $RL = \{ID_1, \dots, ID_K\}$, the algorithm consists the following steps:

- (i) The algorithm chooses a random $R \in G_T$ and computes $ck = H_2(R)$ as the symmetric encryption key. Then, it encrypts the data M with

ck to get $CT_s = \text{Enc}(M, ck)$, where Enc is the symmetric encryption algorithm of our scheme. Moreover, the algorithm computes the verification code $VC = H_3(R' \| CT_s)$ for CT_s , where $R' = H_0(R)$.

- (ii) With the designate access policy tree T whose root node is denoted by R , the algorithm chooses a random number $s_R^0 \in Z_p$ as the base secret value of T and computes $C_0 = R \cdot \hat{e}(g, g)^{as_R^0}$. Then, for each node x in T , the algorithm picks two random number $s_x^1 \in Z_p$ and $s_x^2 \in Z_p^*$, which satisfy the following equation:

$$\begin{cases} s_x^1 \cdot s_x^2 = s_x^0, & x \text{ is linked to a time trapdoor,} \\ s_x^1 = s_x^0, s_x^2 = 1, & \text{otherwise.} \end{cases} \quad (7)$$

- (iii) Given the user revocation list $RL = \{ID_1, \dots, ID_K\}$, the algorithm selects a random δ_k for $ID_k \in RL$, where $\sum_{k=1}^K \delta_k = s_R^0$. Then, the algorithm computes the corresponding ciphertext component $\widetilde{C}, \widetilde{C}'$, where

$$\forall ID_k \in RL: \widetilde{C}_k = h_1^{\delta_k}, \widetilde{C}'_k = h_1^{-ID_k \cdot \delta_k}. \quad (8)$$

- (iv) Then, for a trapdoor TD_x related to the time release $t \in F_T$ and a secret parameter s_x^2 , DO picks a random number $r_t \in Z_p$ and generates TD_x of node x as follows:

$$TD_x = (A_x = g^{r_t}, B_x = s_x^2 + H_2(e(H_1(t), f)^{r_t})). \quad (9)$$

- (v) Next, the algorithm computes the ciphertext in a top-to-bottom way by executing the following steps:

- (1) For each nonleaf node x with s_x^1 , the DO chooses a polynomial q_x whose degree $d_x = \text{th}_x - 1$ and $q_x(0) = s_x^1$. For each of x 's child node $y \in \text{child}(x)$ with a unique index $\text{index}(y)$, DO sets $s_y^0 = q_x(\text{index}(y))$.
- (2) For a leaf node x with s_x^1 and related attribute $\text{att}(x)$, the algorithm generates corresponding ciphertext components C_x and C'_x as follows:

$$\forall x \in X: C_x = g^{s_x^1}, \quad C'_x = H_1(\text{att}(x))^{s_x^1}, \quad (10)$$

where X is the leaf node set in T .

- (3) For each translating node x' in T , the algorithm computes the corresponding ciphertext component $\overline{C}_{x'}$ as follows:

$$\forall x' \in \overline{X}: \overline{C}_{x'} = h_2^{s_{x'}^1}, \quad (11)$$

where \overline{X} is the translating node set in T .

Finally, the DO outputs the ciphertext $CT = \{T, RL, CT_s, VC, C_0, \{C_x, C'_x\}_{x \in X}, \{TD_x\}_{TD_x \in T}, \{\bar{C}_{x'}\}_{x' \in \bar{X}}, \{\tilde{C}_k, \tilde{C}'_k\}_{ID_k \in RL}\}$ and uploads it to CSP.

6.4. Time-Release Phase

- (i) Token Gen(PK, t) \longrightarrow Tok $_t$: as the system runs at a uniform time and the time is counted by the number of time point here. When each time point $t \in F_T$ arrives, TA published a time token Tok $_t = H_1(t)^y$ which can be received by each entity in the system.
- (ii) Trap(PK, Tok $_t$) \longrightarrow TD' $_x$: when CSP receives a Tok $_t$ at releasing time point t published by CA, it finds all trapdoors related to time point t in all access policies of files stored in CSP. For each of these trapdoors TD $_x = (A_x, B_x)$, the CSP computes the following equation:

$$\begin{aligned} TD'_x &= B_x - H_2(\widehat{e}(\text{Tok}_t, A_x)) \\ &= s_x^2 + H_2(e(H_1(t), g^y)^{r_t}) - H_2(\widehat{e}(H_1(t)^y, g^{r_t})) \\ &= s_x^2. \end{aligned} \quad (12)$$

Then, the CSP replaces these TD $_x$ with TD' $_x$ for the ciphertexts of related files. Thus, if the above equation is correctly executed, the related trapdoor will be exposed to be TD' $_x = s_x^2$.

6.5. Decryption Phase. The decryption phase involves the following algorithms:

- (i) Decrypt $_{\text{OUT}}$ (PK, TPK $_u$, CT) \longrightarrow CT': the algorithm is executed by CSP for outsourced decryption of CT. As to each node $x \in T$, we have the exposed trapdoor, i.e., TD' $_x = s_x^2$, as follows:

$$\begin{cases} s_x^2 = TD'_x, & x \text{ is related to an exposed trapdoor,} \\ s_x^2 = 1, & x \text{ is related to no trapdoor.} \end{cases} \quad (13)$$

The algorithm first runs $T(\mathcal{A})$ to check if the key structure \mathcal{A} in TPK $_u$ satisfies the policy tree T in CT. Then, the algorithm gets a label set S_x for each node $x \in T$ when it recursively runs $T_x(\mathcal{A})$. If \mathcal{A} does not satisfy T , then the algorithm returns null; Otherwise, the algorithm chooses i from label set returned by $T(\mathcal{A})$ and performs DecryptNode(CT, TPK $_u$, x, i) as follows.

For each leaf node $x \in T$, if $\text{att}(x) = \text{att}_{i,j} \in A_i$, where $A_i \in \mathcal{A}$ belongs to the attribute set \mathcal{A} of DU and the trapdoor set upon the node has been correctly exposed, the CSP can execute DecryptNode(CT, TK $_u$, x, i) as follows:

$$\begin{aligned} P_x &= \text{DecryptNode}(CT, \text{TPK}_u, x, i) \\ &= \left(\frac{\widehat{e}(K_{i,j}, C_x)}{\widehat{e}(K_{i,j}', C'_x)} \right)^{s_x^2} \\ &= \widehat{e}(g, g)^{r_{\Delta,i} (s_x^0/z)}. \end{aligned} \quad (14)$$

For each nonleaf node $x \in T$ with label i , the algorithm directly executes DecryptNode(CT, TK $_u$, x, i) as follows:

- (i) Firstly, assume L_x be arbitrary th $_x$ -sized set of child nodes y of the node x , and we have $y \in L_x$ only if $i \in S_y$ or $i' \in S_y$ ($i' \neq i$) and x is a translating node. If no such set exists, it returns null.
- (ii) Moreover, for each node $y \in L_x$, if $i \in S_y$, the algorithm executes DecryptNode(CT, TK $_u$, y, i) and gets its output P_y , and if $i' \in S_y$ ($i' \neq i$), the algorithm executes DecryptNode(CT, TK $_u$, y, i') and gets P'_y . Then, the algorithm translates P'_y to P_y by computing the following:

$$\begin{cases} P_y = \widehat{e}\left(\bar{C}_y, \frac{\widehat{E}_i}{\widehat{E}_i'}\right)^{s_y^2} \cdot P'_y & i \neq 0, \\ = \widehat{e}(g, g)^{s_y^0 r_{\Delta,i}/z}, \\ P_y = \frac{\widehat{e}(\bar{C}_y, E_i)^{s_y^2}}{P'_y} = \widehat{e}(g, g)^{s_y^0 (r_{\Delta,i}/z)}, & i = 0. \end{cases} \quad (15)$$

- (iii) Furthermore, the algorithm computes P_x according to polynomial interpolation by $P_x = \prod_{y \in L_x} P_y^{\Delta_{j,x}}$, where $j = \text{index}(y)$ and $L_x = \{\text{index}(y): y \in L_x\}$. Thus, it gets the result as

$$\begin{cases} P_x = \widehat{e}(g, g)^{(s_x^0 r_{\Delta,i}/z)}, & i \neq 0, \\ P_x = \widehat{e}(g, g)^{(s_x^0 r_{\Delta,i}/z)}, & i = 0. \end{cases} \quad (16)$$

- (iv) In addition, the algorithm executes DecryptNode(CT, TK $_u$, R, i) on root node and gets P_R in a bottom-up way. Then, it outputs the final result P as follows:

$$\begin{cases} P = \frac{\widehat{e}(\bar{C}_R, E_i)^{s_R^2}}{P_R} = \widehat{e}(g, g)^{(s_R^0 r_{\Delta,i}/z)}, & i \neq 0, \\ P = P_R = \widehat{e}(g, g)^{(s_R^0 r_{\Delta,i}/z)}, & i = 0. \end{cases} \quad (17)$$

- (v) Later, the algorithm computes:

$$C' = \frac{\prod_{k=1}^K \left(\hat{e}(K_1, \tilde{C}_k) \hat{e}(K_0, \tilde{C}'_k) \right)^{(1/\text{ID} - \text{ID}_k)}}{P} \quad (18)$$

$$= \hat{e}(g, g)^{s_k^0 \alpha / z}.$$

Finally, the CSP sends partially decrypted ciphertext $CT' = \{C', CT_s, VC, C_0\}$ to the DU.

- (i) $\text{Decrypt}_U(\text{PK}, \text{TSK}_u, CT')$ \rightarrow M or null: after receiving the system public parameters PK, the secret transformation key TSK_u , and the partially decrypted ciphertext CT' , the algorithm gets the random element R^* by computing $R^* = C_0 / (C')^{\text{TSK}_u}$. If the algorithm Dec Verify returns True, it computes $\text{ck}^* = H_2(R^*)$ and recovers the plaintext $M = \text{Dec}(CT_s, \text{ck}^*)$. Otherwise, it outputs null.
- (ii) $\text{Dec Verify}(\text{PK}, R^*, VC, CT_s) \rightarrow \text{True/False}$: on inputting a recovered random element R^* , the DU computes $\bar{R}^* = H_0(R^*)$ and checks the following equations:

$$VC \stackrel{?}{=} H_3(\bar{R}^* CT_s). \quad (19)$$

If equation (19) holds, the algorithm outputs True. Otherwise, it outputs False.

7. Security Analysis

This section presents the formal security analysis for HTR-DAC.

Theorem 1. *Our scheme achieves soundness if and only if the data user has enough rights and correct exposed trapdoors.*

Proof. If and only if the data user has enough rights and correct exposed trapdoors, we have the following equations.

For each node $x \in T$, suppose $\bar{Y} = \hat{e}(g, g)$, and we have

$$P_x = \text{Decrypt Node}(CT, \text{TK}_u, x, i)$$

$$= \left(\frac{\hat{e}(K_{i,j}, C_x)}{\hat{e}(K_{i,j}, C'_x)} \right)^{s_x^2}$$

$$= \left(\frac{\hat{e}\left((g^{r_{\Delta,i}} \cdot H_1(\text{att}_{i,j})^{r_{\Delta,i,j}})^{(1/z)}, g^{s_x^1} \right)}{\hat{e}\left((g^{r_{\Delta,i,j}})^{(1/z)}, H_1(\text{att}(x))^{s_x^1} \right)} \right)^{s_x^2} \quad (20)$$

$$= \bar{Y}^{r_{\Delta,i} s_x^1 (s_x^2/z)} \& = \bar{Y}^{r_{\Delta,i} (s_x^0/z)}.$$

Then, for each $y \in L_x$, if $i' \neq i$, the translating procedure is performed as follows:

- (i) If $i \neq 0$,

$$P_y = \hat{e}\left(\bar{C}_y, \hat{E}_i / \hat{E}_i \right)^{s_y^2} \cdot P'_y$$

$$= \hat{e}\left(h_2^{s_y^1}, \frac{g^{(r_{\Delta} + r_{\Delta,i'} / \beta_2 z)}}{g^{(r_{\Delta} + r_{\Delta,i'} / \beta_2 z)}} \right)^{s_y^2} \cdot \hat{e}(g, g)^{r_{\Delta,i'} (s_y^0/z)}$$

$$= \hat{e}(g, g)^{s_y^1 s_y^2 (r_{\Delta,i} - r_{\Delta,i'}) / z} \cdot \hat{e}(g, g)^{r_{\Delta,i'} (s_y^0/z)}$$

$$= \hat{e}(g, g)^{s_y^0 r_{\Delta,i} / z}, \quad (21)$$

$$P = \hat{e} \frac{(\bar{C}_R, E_i)^{s_R^2}}{P_R} = \hat{e} \frac{\left(h_2^{s_R^1}, g^{(r_{\Delta} + r_{\Delta,i} / \beta_2 z)} \right)^{s_R^2}}{\bar{Y}^{s_R^0 r_{\Delta,i} / z}}$$

$$= \frac{\bar{Y}^{(r_{\Delta} + r_{\Delta,i}) s_R^1 s_R^2 / z}}{\bar{Y}^{s_R^0 r_{\Delta,i} / z}} = \bar{Y}^{r_{\Delta} s_R^0 / z}.$$

- (ii) If $i = 0$,

$$P_y = \frac{\hat{e}(\bar{C}_y, \hat{E}_i)^{s_y^2}}{P'_y}$$

$$= \hat{e} \frac{\left(h_2^{s_y^1}, g^{(r_{\Delta} + r_{\Delta,i'} / \beta_2 z)} \right)^{s_y^2}}{\bar{Y}^{r_{\Delta,i'} (s_y^0/z)}} \quad (22)$$

$$= \frac{\bar{Y}^{s_y^1 s_y^2 (r_{\Delta} + r_{\Delta,i'}) / z}}{\bar{Y}^{r_{\Delta,i'} (s_y^0/z)}}$$

$$= \bar{Y}^{s_y^0 r_{\Delta} / z}.$$

Finally, we have

$$Q = \prod_{k=1}^K \left(\hat{e}(K_1, \tilde{C}_k) \hat{e}(K_0, \tilde{C}'_k) \right)^{(1/\text{ID} - \text{ID}_k)}$$

$$= \prod_{k=1}^K \left(\hat{e}\left(g^{((\alpha+r_{\Delta})\text{ID}/\beta_1 z)}, h_1^{\delta_k} \right) \cdot \hat{e}\left(g^{(\alpha+r_{\Delta}/\beta_1 z)}, h_1^{-\text{ID}_k \delta_k} \right) \right)^{(1/\text{ID} - \text{ID}_k)}$$

$$= \prod_{k=1}^K \left(\bar{Y}^{(\alpha+r_{\Delta})(\text{ID} - \text{ID}_k) \delta_k / z} \right)^{(1/\text{ID} - \text{ID}_k)}$$

$$= \prod_{k=1}^K \left(\bar{Y}^{(\alpha+r_{\Delta}) \delta_k / z} \right) = \hat{e}(g, g)^{\alpha+r_{\Delta} / z \sum_{k=1}^K \delta_k}$$

$$= \bar{Y}^{(\alpha+r_{\Delta}) s_R^0 / z} C' = \frac{\sum_{k=1}^K \left(\hat{e}(K_1, \tilde{C}_k) \hat{e}(K_0, \tilde{C}'_k) \right)^{(1/\text{ID} - \text{ID}_k)}}{P}$$

$$= \frac{Q}{P} = \frac{\bar{Y}^{(\alpha+r_{\Delta}) s_R^0 / z}}{\bar{Y}^{s_R^0 r_{\Delta} / z}} = \bar{Y}^{s_R^0 \alpha / z}. \quad (23)$$

TABLE 2: Function comparison.

Scheme	Hierarchical authority	Time-based control	Large universe	Revocability	High efficiency	Verifiability
Scheme [41]	×	×	✓	✓	×	×
Scheme [12]	×	✓	✓	×	×	×
Scheme [42]	×	×	✓	✓	×	×
Scheme [37]	×	×	×	×	✓	✓
Scheme [43]	✓	×	×	✓	×	×
Scheme [8]	✓	×	×	×	✓	×
Our scheme	✓	✓	✓	✓	✓	✓

Therefore, the proposed HTR-DAC is sound if and only if the data user has enough rights and correct exposed trapdoors. \square

Theorem 2. *No PPT adversaries can selectively win the security game of our scheme with an advantage that is non-negligible on condition the DBDH assumption holds.*

Proof. When the advantage ζ of adversary \mathcal{A} is non-negligible when he selectively breaks the security game against our scheme, we can create a simulator \mathcal{B} who is able to distinguish a DBDH parameter from a random parameter with an identical advantage to that of \mathcal{A} . \square

Init: the simulator \mathcal{B} of DBDH game creates the bilinear group $\{G_0, G_1, \hat{e}, p, g\}$, where $\hat{e}: G_0 \times G_0 \rightarrow G_1$ and $g \in G_0$. It then selects randoms $c, d, m, v \in Z_p$ and $\epsilon \in [0, 1]$. If $\epsilon = 0$, the challenger \mathcal{B} generates a tuple $(C, D, M, V) = (g^c, g^d, g^m, \hat{e}(g, g)^{cdm})$; otherwise, it generates $(g^c, g^d, g^m, \hat{e}(g, g)^v)$. \mathcal{B} then sends the tuple to \mathcal{C} . In the meantime, the adversary \mathcal{A} submits a selected challenging access policy tree \mathcal{T} , a revocation list $RL = \{ID_1, \dots, ID_n\}$, and a time point $t_1 \in F_T$ to challenger \mathcal{C} of our scheme.

Setup 2: after the challenger \mathcal{C} gets the DBDH tuple (C, D, M, V) and bilinear group from \mathcal{B} , it randomly chooses $\alpha, \beta_1, \beta_2, \gamma \in Z_p$ and hash functions H_0, H_1, H_2 , and H_3 . Then, for Type-A attack, \mathcal{C} computes $\delta = g^v$, and for Type-B attack, it simulates $\delta = D$. \mathcal{C} also simulates $H_1(x) = g^{q_x}$, where $q_x \in Z_p$. Finally, \mathcal{C} generates system public parameters $PK = \{G_0, G_1, p, g, \hat{e}, \hat{e}(g, g)^\alpha, h_1 = g^{\beta_1}, h_2 = g^{\beta_2}, \eta_1 = g^{(1/\beta_1)}, \eta_2 = g^{(1/\beta_2)}, \delta\}$, and the master key $MSK = \{g^\alpha, \beta_1, \beta_2, \gamma, H_1\}$. It keeps the MSK privately and sends the PK to the adversary \mathcal{A} .

Phase 3: the adversary \mathcal{A} submits a series of queries q_i for secret key and transformation key as follows:

- (i) SK query: \mathcal{A} requests for secret key with his identity ID that $ID \notin RL$ and a recursive attribute set \mathcal{A} that is unsatisfactory to \mathcal{T} at time t_1 . Then, \mathcal{C} computes $D_0 = g^{(\alpha+m/\beta_1)} = (M \cdot g^\alpha)^{(1/\beta_1)}$ and $D_1 = D_0^{I \cdot D}$. For all $\text{att}_{i,j} \in \mathcal{A}$, it generates $D_{i,j} = g^{r_{\Delta,i}} \cdot (H_1(\text{att}_{i,j}))^{r_{\Delta,i}}$, $D_{i,j}' = g^{r_{\Delta,i}}$, and for all $A_i \in \mathcal{A}$, it computes $E_i = g^{(m+r_{\Delta,i}/\beta_2)}$, where $r_{\Delta,i}$ and $r_{\Delta,i,j}$ are randomly picked. For Type-B attack, \mathcal{C} designate a time point $t_1' > t_1$ at which the recursive attribute set \mathcal{A} satisfies \mathcal{T} and computes $H_1(t) = g^{q_t}$ and $\text{Tok}_t = B^{q_t}$, where $t < t_1$ and $q_t \in_R Z_p$. Finally, \mathcal{C} returns SK_u to \mathcal{A} and publishes $\{\text{Tok}_t\}_{t < t_1}$.

- (ii) TK query: similar to the SK query, the challenger \mathcal{C} runs $TKeyGen$ algorithm to generate transformation key pair and sends them to \mathcal{A} .

Challenge: the adversary \mathcal{A} finishes the Phase 1 and submits two data M_0 and M_1 with equal length to \mathcal{C} . First, \mathcal{C} picks $\epsilon \in_R [0, 1]$ and computes $C_0 = M_\epsilon \cdot (\hat{e}(M \cdot g^\alpha, C)/V)$. For each ID_k in the submitted user revocation list RL, \mathcal{C} chooses a random number $\delta_k \in Z_p$ so that $\sum_{k=1}^K \delta_k = c$ and computes $\tilde{C}_k = h_1^{\delta_k}$ and $\tilde{C}_k' = h_1^{-ID_k \cdot \delta_k}$. Then, according to the challenging access policy tree \mathcal{T} , $\forall x \in \mathcal{T}$, if it is related to a trapdoor, \mathcal{C} chooses a random $s_x^2 \in Z_p$; otherwise, it sets $s_x^2 = 1$. Next, \mathcal{C} computes $C_x = D^{q_x(0)}$, $C_x' = H_1(\text{att}(x))^{q_x(0)}$, and $\text{TD}_x = s_x^2$, where $q_x(0) = s_x/s_x^2$ and q_x is a th_x -sized polynomial. It also designates the translating node set \bar{X} in \mathcal{T} and computes $\tilde{C}_{x'} = h_2^{s_x}$ for each node $x' \in \bar{X}$. Finally, the challenger \mathcal{C} returns the ciphertext $CT^* = \{T, RL, C_0, \{C_x, C_x'\}, \{\text{TD}_x\}, \{\tilde{C}_{x'}\}, \{\tilde{C}_k, \tilde{C}_k'\}\}$ to \mathcal{C} . For Type-B attack, \mathcal{C} acts as [12].

With respect to the adversary \mathcal{A} , when $\epsilon = 0$, $V = \hat{e}(g, g)^{cdm}$, and according to the decryption procedure, the adversary can get M_ϵ from CT. Nevertheless, when $\epsilon = 1$, $V \in G_1$ is a random element. Thus, \mathcal{A} cannot get any information about M_ϵ from CT.

Phase 4: the adversary \mathcal{A} repeats the procedures in Phase 1 with the same restriction that the $ID \notin RL$ and the attribute set \mathcal{S} in queries do not satisfy T .

Guess: the adversary \mathcal{A} outputs the guess of bit ϵ . If $\epsilon = \epsilon$, the challenger \mathcal{C} guesses $Z = \hat{e}(g, g)^{cdm}$ with his output 0; otherwise, it guesses Z as a random element. If the adversary \mathcal{A} has the advantage of ζ , then the challenger \mathcal{C} can break the DBDH game with advantage $(\zeta/2)$ given that the variables ϵ and ϵ are independent. The computation of the advantage for \mathcal{C} is the same as in [12].

In conclusion, if an adversary \mathcal{A} can win the security game of our scheme with a nonnegligible advantage ζ , then the challenger \mathcal{C} can break the DBDH game with identical advantage. Therefore, our scheme is IND-CPA secure in our security model.

8. Performance Evaluation

Here, we analyze the performance for our scheme in functional, theoretical, and experimental respects.

First of all, we present the function comparison between our scheme and some existing related CP-ABE schemes, i.e., [8, 12, 37, 41–43], as shown in Table 2. We observe that our HTR-DAC scheme supports hierarchical authorities and users, time-sensitive data sharing, large universe, direct

TABLE 3: Storage complexity comparison.

Schemes	Encrypt	Decrypt	KeyGen	PP Size	UKey Size
Scheme [12]	$(2 X + TD + 1)E_{G_0} + (TD + 1)E_{G_1} + \text{Mul}_{G_1}$ $+ TD P + (X + 2 TD)H$	$(3 S + k)E_{G_1} + (2 S + 2)\text{Mul}_{G_1}$ $+ (2 S + 1)P$	$(3 S + 1)E_{G_0} + S \text{Mul}_{G_0} + S H$	$3 G_0 + G_1 $	$(2 S + 1) G_0 $
Our scheme	$(2 X + TD + \bar{X} + 2 RL)E_{G_0} + (TD + 1)E_{G_1}$ $+ \text{Mul}_{G_1} + TD P + (2 TD + X)H$	$E_{G_1} + \text{Mul}_{G_1} + 3H$	$(3 S + AS + 1)E_{G_0} + (3 S + AS + 1)\text{Mul}_{G_0}$ $+ S H$	$6 G_0 + G_1 $	$ Z_p $

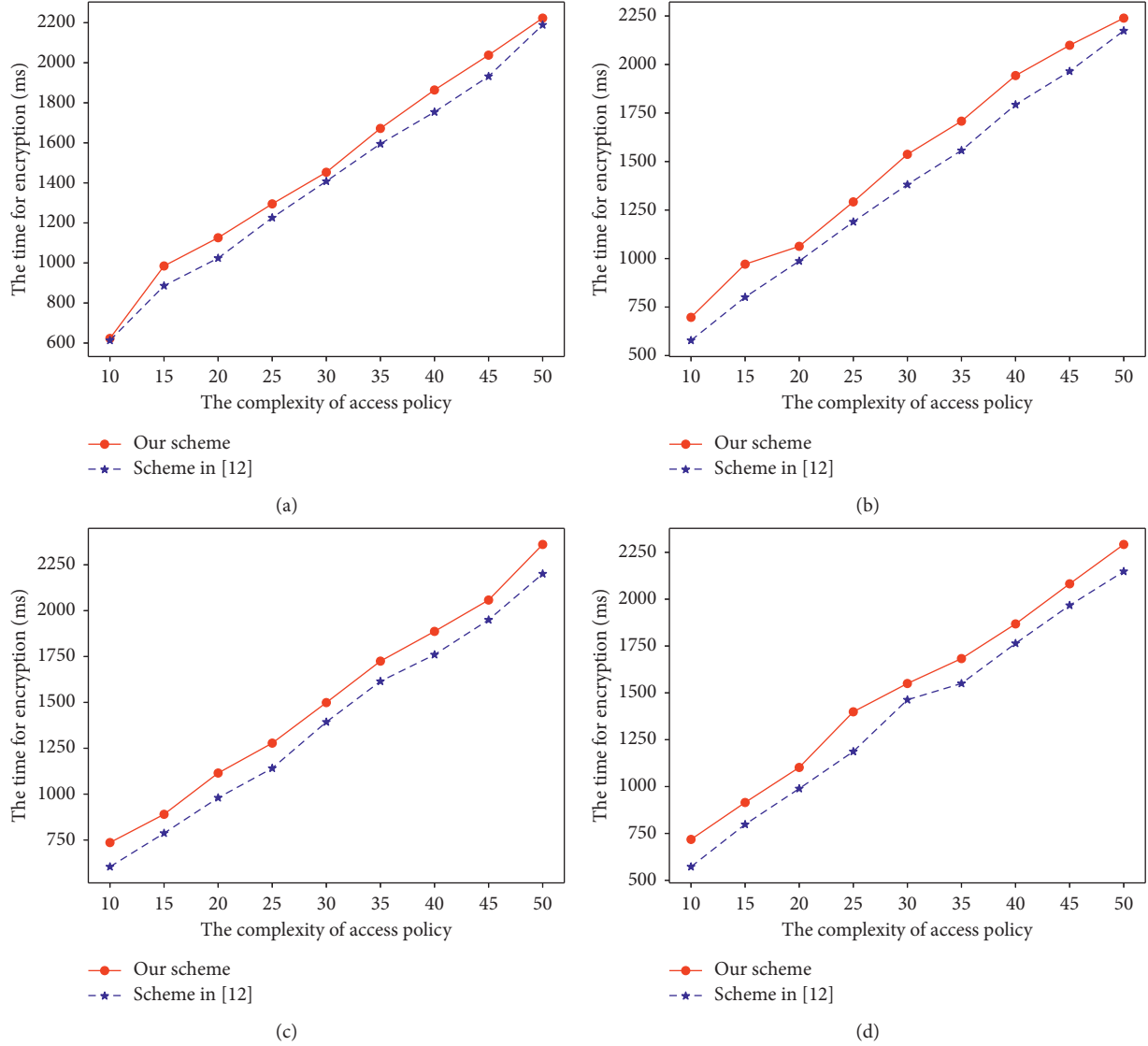


FIGURE 4: Encryption cost comparison for different numbers of translating nodes. (a) $|\bar{X}| = 0$. (b) $|\bar{X}| = 2$. (c) $|\bar{X}| = 4$. (d) $|\bar{X}| = 6$.

revocability, efficient decryption, and verifiability, simultaneously, which are more flexible and scalable than others. Then, to demonstrate the theoretical performance evaluation, we compare the computation complexity and storage complexity of our scheme with that of the state-of-the-art scheme in [12]. The comparison results are shown in Table 3, which summarizes the complexity of Encrypt, Decrypt, KeyGen, PPSize, and UKeySize denoting the computation complexity of encryption, decryption, and key generation algorithms, as well as the storage complexity of the public parameter size and user key size, respectively.

To obtain precise evaluation of the performance, we implement our scheme and the scheme [12] using Java Programming Language and Java Pairing-Based Cryptography library (JPBC) [44] which supports operations of pairing, exponential, addition, multiplication, and inversion in finite field and groups. In our development, we adopt the Type *A* curve with prime order. It is defined over a 160 bit elliptic curve group and a 512 bit finite field. Moreover, our experimental

simulations are conducted on Windows system equipped with CPU of Intel Core i5 CPU 2.13 GHz and RAM of 8.00 GB.

Before our analysis, we let $|X|, k, |\bar{X}|, |TD|$, and $|RL|$ denote the size of leaf node set, nonleaf node set, translating node set, trapdoor set in access policy tree T , and the size of revocation list. $E_{G_0}, \text{Mul}_{G_0}, E_{G_1}, \text{Mul}_{G_1}, P$, and H denote the exponential operation and the multiplication operation in G_0 and G_1 as well as the pairing and hash function operations. $|G_0|, |G_1|$, and $|Z_p|$ denote the length of elements in G_0, G_1 , and Z_p , respectively. The symbols $|AS|$ and $|S|$ denote the number of sets and attributes in a recursive attribute set.

Figure 4 shows the actual performance of Encrypt in both schemes. We notice that the computational cost of encryption in the other scheme is affected by the factors of the number of leaf nodes and the number of trapdoors in access policy tree T , while our scheme is affected by the size of translating node set and revocation list in addition. These two additional factors are brought about by the features of

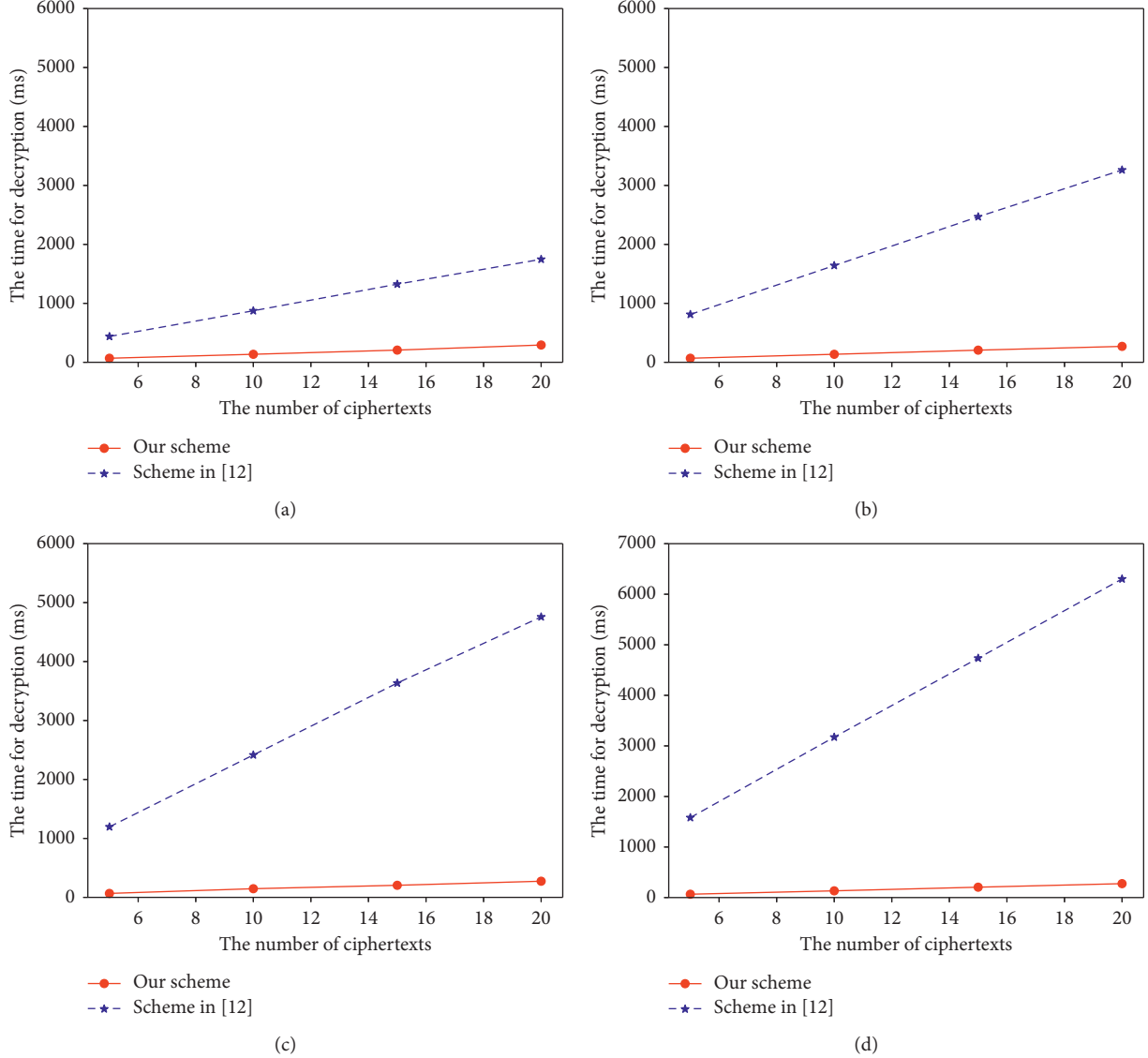


FIGURE 5: Decryption cost comparison for different sizes of user attribute set. (a) $|S| = 5$. (b) $|S| = 10$. (c) $|S| = 15$. (d) $|S| = 20$.

user key structure and direct revocation in our scheme and incur extracomputation in encryption. In our experiments, we set $|\text{RL}| = 5$ and adjust the value of $|\overline{X}|$. Figure 4 shows the actual time cost in different numbers of translating nodes. With the same access policy and fixed size of revocation list, the time cost in our scheme is more, and as the number of translating nodes grows, the gap becomes larger. However, we notice that the difference between two schemes is tiny on the whole.

In Figure 5, we describe the comparison of time cost in Decrypt algorithm for the two schemes. We notice that, within the same size of user attribute set, the decryption time cost in our scheme is smaller and nearly constant as the number of ciphertexts grows, while in scheme [12], it is far more and linear with the growth of the number of ciphertexts. In theory, the computation complexity in our scheme and the other are $E_{G_1} + \text{Mul}_{G_1} + 3H$ and $(3|S| + k)E_{G_1} + (2|S| + 2)\text{Mul}_{G_1} + (2|S| + 1)P$, respectively.

In our experiments, we set $k = 2$. It is obvious from the figure that the computation complexity in our scheme is constant and far smaller while that of the other scheme is affected by the size of the user attribute set. Moreover, within the same size of the user attribute set, the time cost of decryption for one data file is far more than that of our scheme, and as the number of ciphertexts grows, the gap becomes larger.

Figure 6 shows the actual performance of key generation procedure in the two schemes. From the theoretical analysis, the computation complexity of our scheme and the other are $(3|S| + |AS| + 1)E_{G_0} + (3|S| + |AS| + 1)\text{Mul}_{G_0} + |S|H$ and $(3|S| + 1)E_{G_0} + |S|\text{Mul}_{G_0} + |S|H$, respectively, which means the time cost in key generation in our scheme is affected by not only the number of attributes of a user but also the attribute sets of a user in his key structure \mathcal{A} , which will incur extra time cost. Moreover, as the figure shows, the time cost in our scheme for key generation in the same number of user attributes is nearly the same as that of the scheme [12].

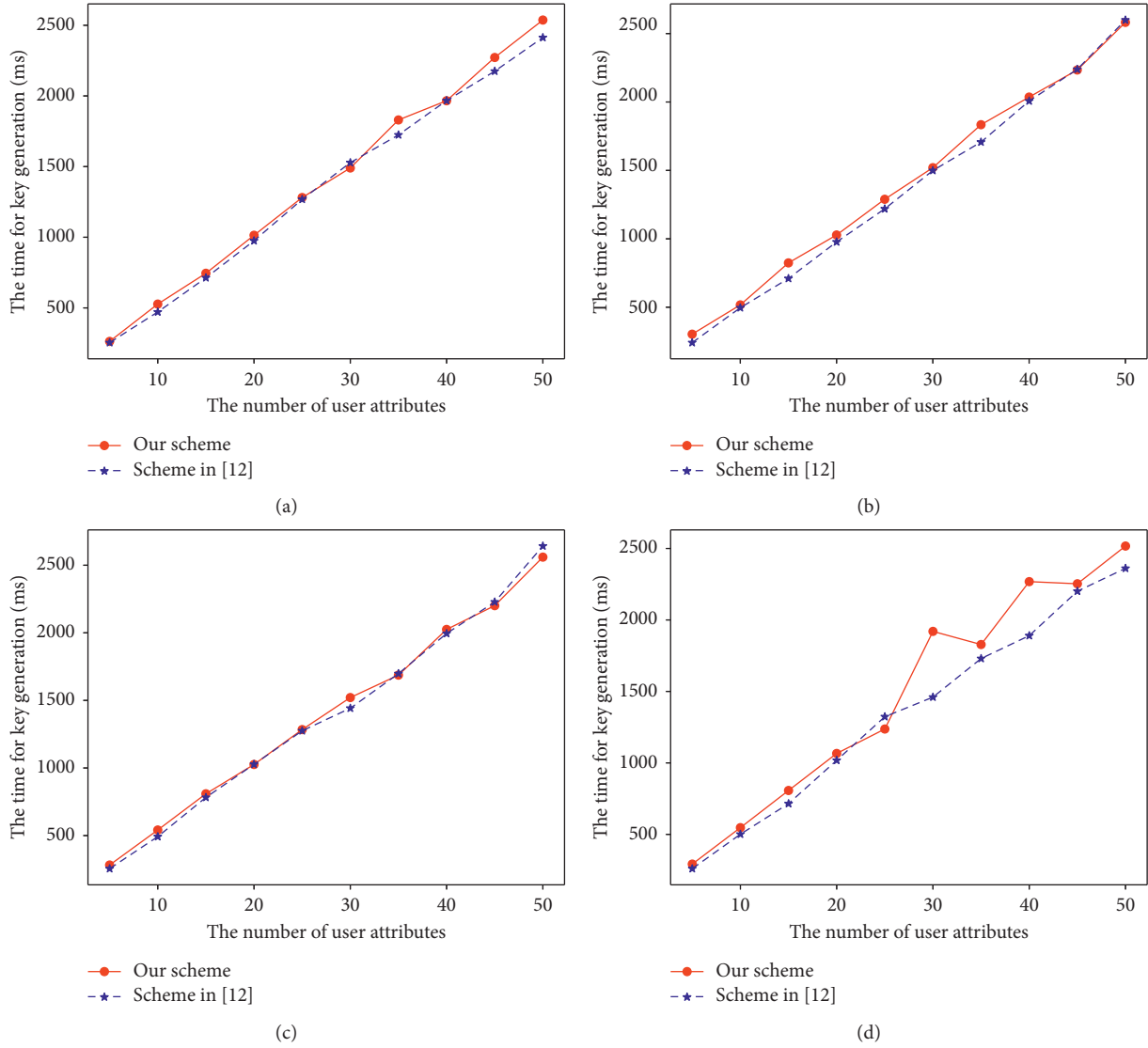


FIGURE 6: Key generation cost comparison for different numbers of attribute set. (a) $|AS| = 1$. (b) $|AS| = 2$. (c) $|AS| = 3$. (d) $|AS| = 4$.

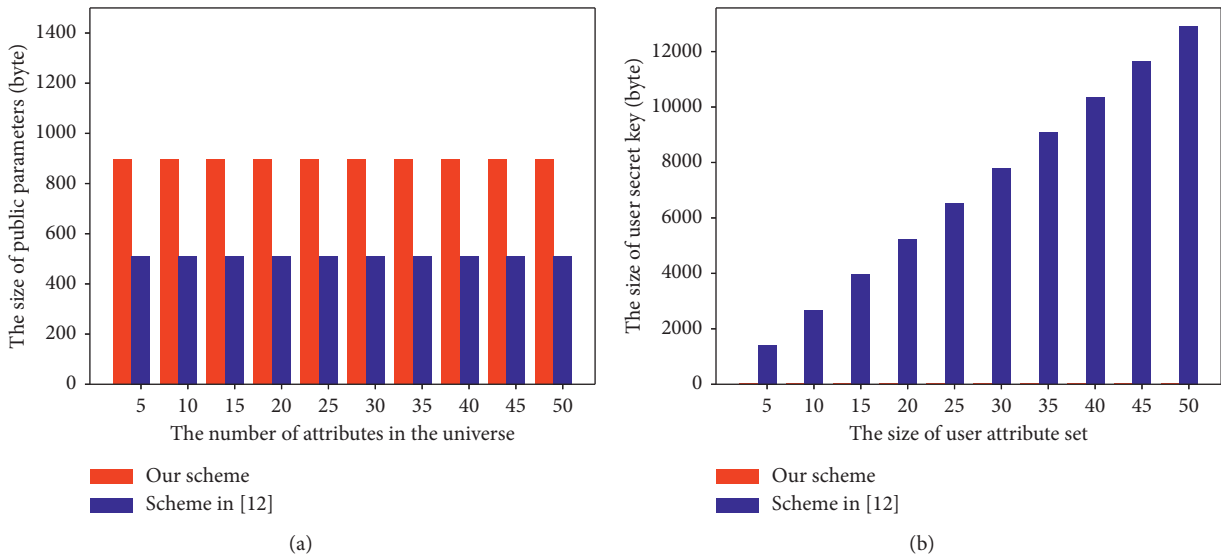


FIGURE 7: Comparison of the storage cost. (a) Public parameter size. (b) User key size.

As the number of attribute set in key structure grows, the gap is still very small. We can infer that the difference in time cost for key generation between the two schemes is tiny.

Figure 7 depicts the storage overhead for our scheme and the scheme [12]. We notice that the actual storage overhead in system public parameters, as Figure 7(a) shows, is larger in our scheme as we takes the storage complexity of $6|G_0| + |G_1|$ in our scheme which is more than $3|G_0| + |G_1|$ of the other scheme. This is because, in our scheme, to adapt to hierarchical users, we need more parameters for each level of user hierarchy. Moreover, in Figure 7(b), the storage overhead of user key size in our scheme is both smaller and constant while that of the other scheme is far more and proportional to the size of user attribute set. As we introduce outsourced decryption, the user key size just costs $|Z_p|$ in storage while that of the other scheme costs $(2|S| + 1)|G_0|$. Thus, we greatly lower the storage cost for key size.

In conclusion, our scheme outperforms existing related schemes no matter from the respects of efficiency or storage overhead. Thus, it is more suitable for the environment of mobile crowdsensing.

9. Conclusion

Ensuring fine granularity for time-sensitive data-sharing service in MCS across hierarchical users with recursive attribute sets and revocability is a big challenge. In our work, we propose a hierarchical and time-sensitive data access control scheme with revocability in cloud-based mobile crowdsensing. Our proposal realizes the properties of fine-grained access control, large attribute universe, hierarchical user, and revocability, which suits for data-sharing applications in MCS. Besides, we discuss the security and display the precise performance evaluation by implementing our scheme and conducting extensive experimental simulations which demonstrates the efficiency and practicality.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (nos. 61902291 and 62072352), China Postdoctoral Science Foundation Funded Project (2019M653567), National Natural Science Foundation of Shaanxi Province (2019JM-425), and Fundamental Research Funds for the Central Universities (JB191507).

References

[1] L. Ma, X. Liu, Q. Pei, and Y. Xiang, "Privacy-preserving reputation management for edge computing enhanced mobile

crowdsensing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 786–799, 2018.

- [2] Y. Liu, X. Ma, L. Shu et al., "Internet of things for noise mapping in smart cities: state-of-the-art and future directions," *IEEE Network*, vol. 34, no. 4, pp. 112–118, 2020.
- [3] X. Wang, S. Garg, H. Lin et al., "An intelligent privacy-preserving mobile edge crowdsensing strategy for industrial iot," *IEEE Internet of Things Journal*, 2020.
- [4] A. Capponi, C. Fiandrino, D. Kliazovich, P. Bouvry, and S. Giordano, "A cost-effective distributed framework for data collection in cloud-based mobile crowd sensing architectures," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 1, pp. 3–16, 2017.
- [5] N. H. Sultan, V. Varadharajan, L. Zhou, and F. A. Barbhuiya, "A role-based encryption scheme for securing outsourced cloud data in a multi-organization context," 2020, <https://arxiv.org/2004.05419>.
- [6] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for cloudiot," *IEEE Transactions on Cloud Computing*, 2020.
- [7] M. Rasori, P. Perazzo, and G. Dini, "A lightweight and scalable attribute-based encryption system for smart cities," *Computer Communications*, vol. 149, pp. 78–89, 2020.
- [8] M. Ali, M.-R. Sadeghi, and X. Liu, "Lightweight revocable hierarchical attribute-based encryption for internet of things," *IEEE Access*, vol. 8, p. 23951, Article ID 23964, 2020.
- [9] H. Deng, Z. Qin, Q. Wu, Z. Guan, and Y. Zhou, "Flexible attribute-based proxy re-encryption for efficient data sharing," *Information Sciences*, vol. 511, pp. 94–113, 2020.
- [10] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: efficient policy-hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130–2145, 2018.
- [11] Z. Wan, J. Liu, and R. H. Deng, "Hasbe: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 743–754, 2011.
- [12] J. Hong, K. Xue, Y. Xue et al., "Tafc: time and attribute factors combined access control for time-sensitive data in public cloud," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 158–171, 2017.
- [13] P. Zhang, Z. Chen, K. Liang, S. Wang, and T. Wang, "A cloud-based access control scheme with user revocation and attribute update," in *Proceedings of the Australasian Conference on Information Security and Privacy*, pp. 525–540, Springer, Melbourne, Australia, July 2016.
- [14] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Springer, Aarhus, Denmark, May 2005.
- [15] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 89–98, Acm, Alexandria, VA, USA, October 2006.
- [16] Z. Zhang, P. Zeng, B. Pan, and K.-K. R. Choo, "Large-universe attribute-based encryption with public traceability for cloud storage," *IEEE Internet of Things Journal*, vol. 7, no. 10, 2020.
- [17] K. Sethi, A. Pradhan, and P. Bera, "Practical traceable multi-authority cp-abe with outsourcing decryption and access policy updation," *Journal of Information Security and Applications*, vol. 51, Article ID 102435, 2020.

- [18] P. K. Premkamal, S. K. Pasupuleti, and P. Alphonse, "Dynamic traceable CP-ABE with revocation for outsourced big data in cloud storage," *International Journal of Communication Systems*, vol. 34, Article ID e4351, 2020.
- [19] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 735–737, Chicago, IL, USA, October 2010.
- [20] Y. Wang and J. Gao, "A regulation scheme based on the ciphertext-policy hierarchical attribute-based encryption in bitcoin system," *IEEE Access*, vol. 6, pp. 16 267–316 278, 2018.
- [21] J. Li, Q. Yu, and Y. Zhang, "Hierarchical attribute based encryption with continuous leakage-resilience," *Information Sciences*, vol. 484, pp. 113–134, 2019.
- [22] R. L. Rivest, A. Shamir, and D. A. Wagner, *Time-lock Puzzles and Timed-release Crypto*, ACM, New York, NY, USA, 1996.
- [23] K. Yuan, Z. Liu, C. Jia, J. Yang, and S. Lv, "Public key timed-release searchable encryption," in *Proceedings of the 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*, pp. 241–248, IEEE, Xi'an, China, September 2013.
- [24] L. Xu, F. Zhang, and S. Tang, "Timed-release oblivious transfer," *Security and Communication Networks*, vol. 7, no. 7, pp. 1138–1149, 2014.
- [25] E. Androulaki, C. Soriente, L. Malisa, and S. Capkun, "Enforcing location and time-based access control on cloud-stored data," in *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems*, pp. 637–648, IEEE, Madrid, Spain, July 2014.
- [26] 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 Transactions on Multimedia*, vol. 18, no. 5, pp. 940–950, 2016.
- [27] Y. Zhu, H. Hu, G.-J. Ahn, D. Huang, and S. Wang, "Towards temporal access control in cloud computing," in *Proceedings of the 2012 IEEE INFOCOM*, pp. 2576–2580, IEEE, Orlando, FL, USA, March 2012.
- [28] J. K. Liu, T. H. Yuen, P. Zhang, and K. Liang, "Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 516–534, Springer, Leuven, Belgium, July 2018.
- [29] K. Fan, J. Wang, X. Wang, H. Li, and Y. Yang, "Secure, efficient and revocable data sharing scheme for vehicular fogs," *Peer-to-Peer Networking and Applications*, vol. 11, no. 4, pp. 766–777, 2018.
- [30] P. K. Premkamal, S. K. Pasupuleti, and P. J. A. Alphonse, "Efficient revocable cp-abe for big data access control in cloud computing," *International Journal of Security and Networks*, vol. 14, no. 3, pp. 119–132, 2019.
- [31] B. Qin, Q. Zhao, D. Zheng, and H. Cui, "(Dual) server-aided revocable attribute-based encryption with decryption key exposure resistance," *Information Sciences*, vol. 490, pp. 74–92, 2019.
- [32] H. Cui, T. Hon Yuen, R. H. Deng, and G. Wang, "Server-aided revocable attribute-based encryption for cloud computing services," *Concurrency and Computation: Practice and Experience*, John Wiley & Sons, Hoboken, NJ, USA, Article ID e5680, 2020.
- [33] M. Green, S. Hohenberger, B. Waters et al., "Outsourcing the decryption of abe ciphertexts," in *Proceedings of the USENIX Security Symposium*, vol. 3, San Francisco, CA, USA, 2011.
- [34] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343–1354, 2013.
- [35] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "Dac-macs: effective data access control for multiauthority cloud storage systems," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1790–1801, 2013.
- [36] 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 Transactions on Dependable and Secure Computing*, vol. 14, no. 6, pp. 679–692, 2015.
- [37] 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," *Computer Networks*, vol. 133, pp. 141–156, 2018.
- [38] X. Fu, X. Nie, T. Wu, and F. Li, "Large universe attribute based access control with efficient decryption in cloud storage system," *Journal of Systems and Software*, vol. 135, pp. 157–164, 2018.
- [39] 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 Generation Computer Systems*, vol. 78, pp. 753–762, 2018.
- [40] L. Li, Z. Wang, and N. Li, "Efficient attribute-based encryption outsourcing scheme with user and attribute revocation for fog-enabled iot," *IEEE Access*, vol. 8, pp. 176738–176749, 2020.
- [41] J. Ning, X. Dong, Z. Cao, L. Wei, and X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1274–1288, 2015.
- [42] K. Zhang, H. Li, J. Ma, and X. Liu, "Efficient large-universe multi-authority ciphertext-policy attribute-based encryption with white-box traceability," *Science China Information Sciences*, vol. 61, no. 3, Article ID 032102, 2018.
- [43] X. Yan, X. He, J. Yu, and Y. Tang, "White-box traceable ciphertext-policy attribute-based encryption in multi-domain environment," *IEEE Access*, vol. 7, pp. 128298–128312, 2019.
- [44] A. De Caro and V. Iovino, "jpbcc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, pp. 850–855, Kerkyra, Greece, July 2011.