International Journal of Information Security and Privacy
Volume 19 • Issue 1 • January-December 2025

FUR-HABE:
A Hierarchical CP-ABE Scheme With Traceable Fine-Grained User Revocation for Cloud Storage

Xiaohui Yang
https://orcid.org/0000-0003-0379-6326
Hebei University, China

Ya'nan Tao
https://orcid.org/0009-0002-3505-0051
Hebei University, China

ABSTRACT

An effective method to protect cloud data is access control. But, the efficiency of key distribution by a single authority is low, and it is difficult to achieve dynamic attribute revocation when system properties are shared by multiple users. Existing attribute revocation mechanisms face challenges in terms of functional complexity and computational efficiency, which hinder their practical application. To address these issues, this paper put forward a Hierarchical CP-ABE scheme with Traceable Fine-grained User Revocation for Cloud Storage (FUR-HABE). In this scheme, most of the decryption calculations are outsourced to cloud servers. It employs a layered key authorization mechanism to provide flexible and scalable key delegation. Additionally, the scheme supports key encapsulation key (KEK) attribute revocation and user revocation to accommodate different revocation needs, enabling flexible revocation.

KEYWORDS

Privacy Protection, Attribute Revocation, KEK, Fine-Grained Access Control, CP-ABE

INTRODUCTION

With the development of cloud computing technology, cloud systems provide users with convenient data services (Mell & Grance, 2011). Simultaneously, it has also facilitated improvements in various systems, from current healthcare and assisted living systems to intelligent city systems (Atzori et al., 2010). Seagate predicts that, by 2025, global data creation will grow to 175ZB, posing a significant challenge of explosive data growth for governments and businesses worldwide ("How Backblaze's Fireball B2 Moves Big Datasets to the Cloud Easily, Safely and Swiftly," 2021). For data storage in the cloud, data anonymization is the process of removing or replacing personally identifiable information through technical means so that data cannot be directly associated with a specific individual can prevent sensitive data from being exposed to unauthorized users, using data encryption to protect the data (Han et al., 2010). Encrypted data is always shared by multiple users, so as more sensitive data is in the cloud, fine-grained access control encryption is needed. Sahai and Waters (2005) proposed attribute-based encryption (ABE), which offers an encryption scheme with fine-grained access control. There are two kinds of ABE: key policy ABE (KP-ABE), which

DOI: 10.4018/IJISP.365602

was proposed by Goyal et al. (2006); and ciphertext-policy ABE (CP-ABE), which was proposed by Bethencourt et al. (2007). In CP-ABE, the ciphertexts are associated with access policies, the decryption key is associated with the attribute, and only users who satisfy this policy can decrypt the data, while, in KP-ABE, the ciphertext is associated with the attribute, data is then encrypted based on the user's attributes, and the decryption key is bound with the access policies. Considering the specificity environment of the cloud, we believe that it is more appropriate to use CP-ABE to protect data security in cloud systems because it allows users to customize access policies.

However, due to various issues, pure CP-ABE is not suitable for direct application in applications. The first of these issues is revocation. Revocation can be divided into user revocation, partial attribute revocation, and system attribute revocation. Partial attribute revocation, known as attribute-level user revocation, is the most finely-grained method of revocation (Han et al., 2020). Revoking any attribute from any user may affect other users in the system because every attribute in the system can be shared by multiple users. Additionally, although attribute revocation is more fine-grained, it is less effective when dealing with users because it requires revoking all attributes of that user. Yu et al. (2010) and Naor et al. (2001) focused on the complexity of attribute revocation. In the direct revocation model, revoking a user's identity to revoke all attributes owned by the user is coarse-grained. The second issue is the burden of authority. Hur et al. (2013), Rouselakis and Waters (2015), and K. Yang, X. Jia, & K. Ren, (2013) found that most existing ABE schemes relying on a single key authorization entity have key escrow issues. Having a single central authority (CA) responsible for all attribute management, key distribution, and revocation operations can lead to overwhelming pressure on the authority, increased latency, lack of efficiency, and potential pitfalls. J. Li et al. (2019) studied a hierarchical access control scheme based on attribute encryption in cloud computing. Their proposed scheme enables users to encrypt multiple files at the same access level. However, this led to performance bottlenecks in distributed cloud systems. In the paper, we solve the above issues by designing a hierarchical ABE scheme with traceable flexible revocation, suitable for practical large-scale cloud storage networks.

Related Work

Sahai and Waters (2005) introduced the first ABE scheme, which has evolved into two forms: CP-ABE and KP-ABE. CP-ABE not only protects the data privacy but also allows data owners to customize flexible access policies. The revocation issue of CP-ABE is one of the main obstacles to its widespread use. Before ABE, the revocation issue was studied in IBE that is a public key encryption mechanism in which a user's public key is directly associated with their identity information, simplifying key management. Boldyreva et al. (2001) proposed a revocable IBE scheme by letting the authority broadcast update materials at each revocation period. Pirretti et al. (2006) solved the property revocation problem by associating an expiration date with each property. Sun et al. 2020) and Huang et al. (2021) proposed a revocation method with periods, conducting revocation updates periodically. However, these schemes cannot achieve real-time attribute revocation. To implement an instant revocation mechanism, Tysowski et al. (2013) and Yu et al. (2010) presented a new approach to perform revocation operations by applying CP-ABE and re-encryption. In this approach, after each revocation, the cloud service provider (CSP) re-encrypts the ciphertext to prevent the revoked user from being able to decrypt. Han et al. (2020) and Zhang et al. (2018) proposed an attribute-based CP-ABE scheme, where the ciphertext consists of two parts: one part is concerning to the access policy for attribute values encryption, and the other is related to revocation information updated during revocation. However, the above schemes only support coarse-grained user revocation. Hur et al. (2010) introduced the concept of attribute groups, providing a foundation for fine-grained attribute revocation. Wang et al. (2018) proposed a revocation method that involves a third party's assistance, and the user's key consists of two parts: user key and group key. Liu et al. (2020) proposed a scheme based on ciphertext policy attributes, utilizing key encapsulation key (KEK) and re-encryption for effective attribute revocation. Wang et al. (2021) studied KEK-based CP-ABE with efficient revocation, where the cloud server implements user logout at the attribute level, which can reduce the burden on

data owners and attribute authorities. Although much work has been done (K. Yang, X. Jia, K. Ren, B. Zhang, & R. Xie, 2013; J. Li et al., 2016; Akinyele et al., 2013; Jiang et al., 2022) on revocation issues, they have only addressed either user revocation or attribute revocation, and most have not resolved the burden issue. Praveen Kumar et al. (2018) proposed that, in practical applications, it is necessary to form a revocation mechanism that uses both attribute revocation and user revocation. Peñuelas-Angulo et al. (2023) conducted a qualitative comparison of numerous articles regarding revocation and quantitatively compared the relevant costs of different revocation methods.

In most existing ABE schemes, there is only one authority to issue and revoke user keys. With the rapid development of technology, the speed and scale of data generation and sharing are experiencing explosive growth. With the number of data sharers increasing, authoritative institutions face unprecedented challenges, with the complexity, redundancy, and quality issues of data adding to the difficulty and complexity of handling them. M. Li et al. (2012) proposed secure data outsourcing, reducing the complexity of owners and users key management. To address the significant computational overhead and security issues brought about by large-scale user management and frequent revocations in large networks, Wang et al. (2011) proposed the concept of hierarchical ABE (HABE) by integrating the concepts of IBE and ABE. Wei et al. (2019) proposed a scalable hierarchical access control scheme to securely share health records in cloud storage. Xiao et al. (2022) presented a HABE scheme with scalable policies to enhance both data sharing efficiency and security simultaneously. Miao et al. (2019) proposed a secure multi-authority CP-attribute-based keyword search system in order to reduce the computational and storage burden of resource-constrained devices. These aforementioned schemes have only partially implemented functionalities, such as key delegation, lightweight decryption, revocability, and KEK revocation updates. Building upon this, this paper introduces a privacy-preserving hierarchical CP-ABE scheme for cloud storage, which features traceable fine-grained user revocation, achieving traceable, scalable key delegation, and lightweight revocation mechanisms.

Our Contribution

In most ABE schemes, only a single authority provides key delegation and user revocation services, leading to significant delays and security risks when numerous users query for keys. During the attribute revocation process, distributing the updated keys individually to users results in substantial overhead. Therefore, in order to solve these issues, this paper proposes a privacy-preserving, traceable hierarchical key delegation and efficient KEK revocation ciphertext policy attribute encryption scheme (fine-grained user revocation FUR-HABE). The main contributions are as follows:

- Lightweight decryption mechanism: The most computationally expensive operations during decryption can be operated using a cloud server without leaking any data content or user key information, reducing the decryption costs on the user side.
- Flexible and scalable key delegation: Multiple key generation authorities can handle key delegation and revocation mechanisms independently, allowing users to request keys from any compliant authority. Additionally, when the system requires more computational power, the CA can authorize the addition of new key generators, reducing the burden on individual authorities compared to other schemes.
- Flexible revocation: To meet the requirements of different revocation granularities, this scheme supports both attribute and user revocation. For user revocation, users are directly added to the revocation list instead of revoking each attribute separately; for attribute revocation, efficient fine-grained KEK revocation is implemented. When revoking an attribute, the system updates the KEK tree for user attribute sets, encrypts key update components, updates user keys, and dynamically restricts user permissions.

Taking into account the user and attribute revocation mechanisms, simultaneous user and attribute revocations do not affect each other, and this independence ensures system flexibility and usability, as well as data security and user access continuity. We believe that any available cloud server is feasible and credible. CA acts as a central coordinator responsible for the resource scheduling and state management of all domain authorities (DA)s and notifies all DAs to update their states. Synchronization is through distributed principles.

PRELIMINARIES

The background information of this scheme is described in this section, including the access structures, linear secret sharing schemes (LSSS), KEK tree, and assumptions.

Bilinear Maps

Let G and G_T be two multiplication cyclic groups of prime order p, where g is the generator of G, and A map e: $G \times G \rightarrow G_T$ is a bilinear map (Boneh & Franklin, 2001) and has the following properties:

- Bilinear: $\forall P, \forall Q \subseteq G \text{ and } \forall a, \forall b \subseteq Z_p,\ e(P^a, Q^b) = e(P,Q)^{ab}$.
- Non-degeneracy: $e(g,g) \neq 1$.
- Computability: for ∀P, ∀Q ⊆G, there is an efficient algorithm to compute e(P,Q).

Access Structure

Let P={ P_1, P_2, …, P_n} be parties; a collection $A \subseteq 2^P$ is monotonous for $\forall B, \forall C: \text{if } B \subseteq A \text{ and } B \subseteq C \text{ then } C \subseteq A$. A monotone access structure is a monotone collection A of non-empty subsets of P, i.e., $A \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\varnothing\}$. The sets in A are called authorized sets.

Linear Secret Sharing Scheme

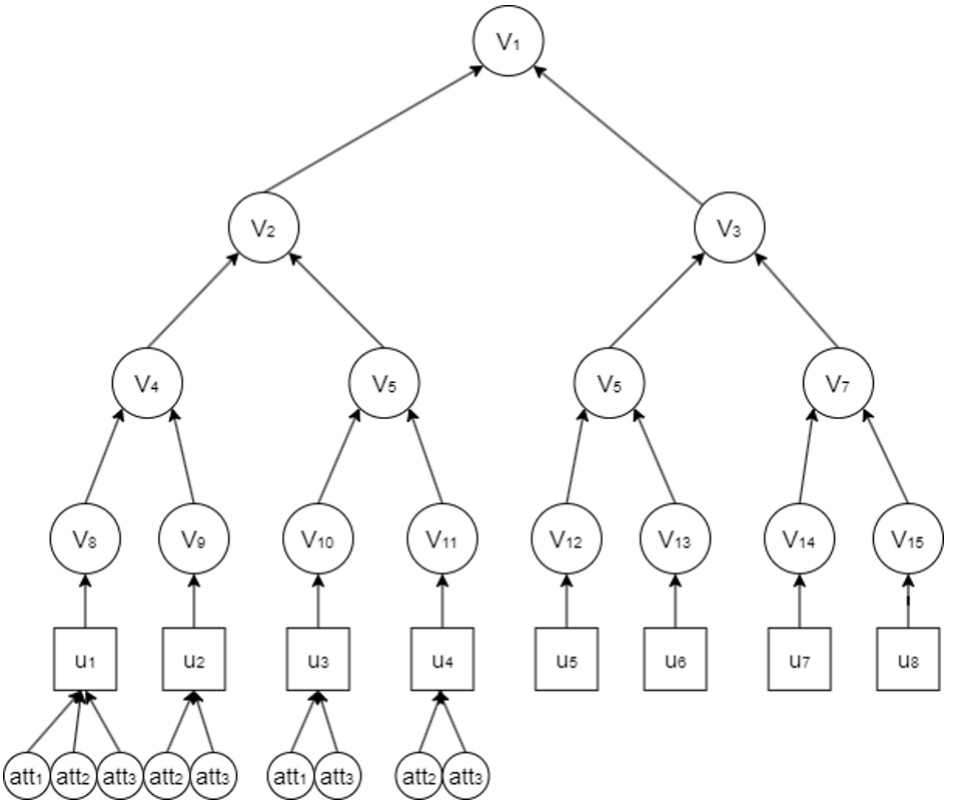
Let $A = \{A_1, A_2, \ldots, A_n\}$ be the attribute name collections, and, for $A_i \subseteq A$, the set of attribute value is $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,n}\}$, where n_i is the order of A_i. A LSSS is used to express the access policy by (M,ρ), where M is a $l \times n$ matrix, and ρ maps a row of matrix M to an attribute in A, which consists of two algorithms:

- *Share* $((M,\rho), s)$: It is used to share a secret value s based on A. A vector $v = (s, v_2, \ldots, v_n)^T$, where $s \subseteq Z_p$ is the secret value, and $v_2, \ldots, v_n \subseteq Z_p$, is randomly chosen; then, $\lambda_i = M_i \cdot v$ is the share of the secret s related to the attribute by $\rho(i)$.
- *Reconstruction* $(\lambda_i, \ldots, \lambda_l, (M,\rho))$: It is used to reconstruct s from secret shares. Let $S \subseteq A$be an authorized set and $I = \{i : \rho(i) \subseteq S\} \subseteq \{1,2,\ldots,l\}$. Then, there exist coefficients $\{c_i \mid i \subseteq I\}$ such that $\sum_{i \subseteq I} c_i M_i = (1,0,\ldots,0)$, and we then have $\sum_{i \subseteq I} c_i \lambda_i = s$.

KEK Tree

The KEK tree is a complete binary tree, which is built by the data manager based on the user set. It provides key update functionality for non-revoking users, as shown in Figure 1. Assume that the set of system users is $U = \{u_1, u_2, \ldots, u_N\}$, and the system attribute set is $S = \{att_1, att_2, \ldots, att_N\}$. Follow the steps below to construct a KEK tree to assigns relevant parameters in the attribute group key for the user.

Figure 1. KEK tree



1. Each leaf node uniquely corresponds to a user in the user set, and each node stores a random value θ_j as the KEK key.
2. Path node algorithm *Path(u)*: For user u, the path node is defined as all nodes from the leaf node to the root node. For example, in Figure 1, the path node of u_1 is $\{V_1,V_2,V_4,V_8\}$.
3. Minimum covering set algorithm *cover(R)*: The minimum coverage set of the attribute user set G_i of attribute att_i is the smallest set in the KEK tree that can cover all user nodes in the attribute set. For example, in Figure 1, the minimum coverage set of attribute att_3 is $\{u_1,u_2,u_3,u_4\}$.
4. For the attribute att_i owned by the user, according to *Path(u)* and *cover(R)*, run *Path(u)*∩ *cover(R)*, and there is a unique intersection point between the two sets with random values stored in the nodes. If the user does not have this attribute, the intersection is empty.

Complexity Assumptions

Definition 1 (q-BDHE Assumption)

The q-BDHE hardness assumption is described as follows: Let G and G_T be two multiplication cyclic groups of prime order p, where g is the generator of G, and A map e: G×G→G_T is a bilinear map (Boneh & Franklin, 2001). Randomly choose $s,\alpha \in Z_p$ and get $Y = \left(g, g^s, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^q}, g^{\alpha^{q+2}} \ldots, g^{\alpha^{2q}}\right)$, if there is no polynomial time algorithm that can distinguish the $T = e(g,g)^{\alpha^{q+1}s}$ from random element $T \in G_T$ with non-negligible advantage ε.

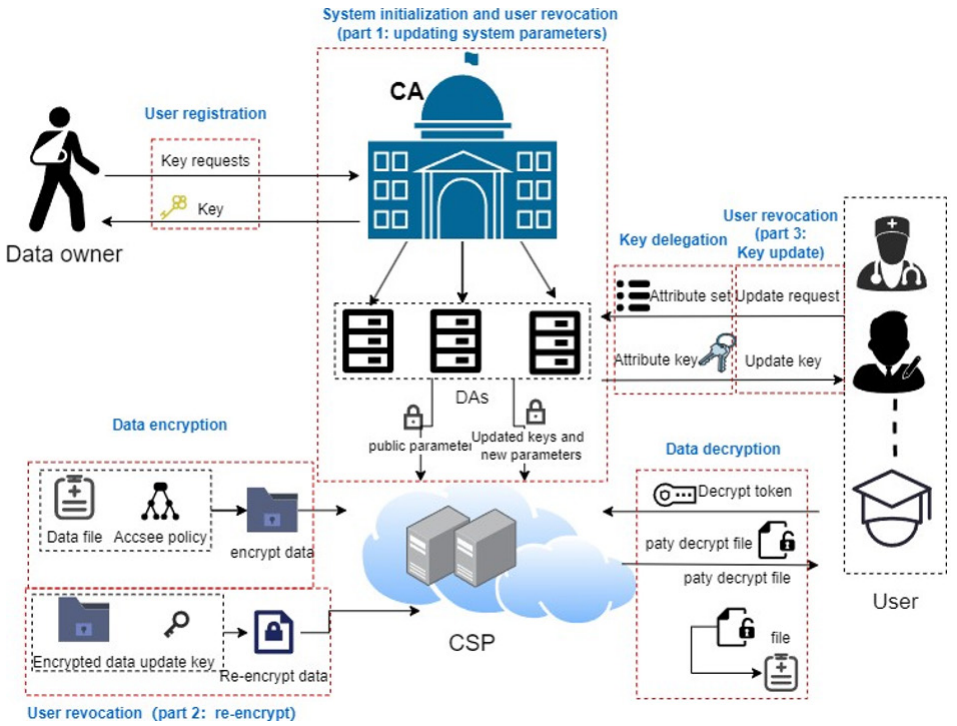
SYSTEM MODEL

This section defines the system architecture of FUR-HABE and security model.

System Architecture

The system model includes five entity parts: CA, DA) CSP, data owner (owner), and data user (user). The system architecture with five entities is described below in Figure 2 as:

Figure 2. System architecture of FUR-HABE



- CA: It includes functions, such as generating system parameters, user registration, and DA initialization. The CA assigns a globally unique user ID and generates a global key pair to each legal user in the system.
- DA: Each DA is an independent attribute authority generating a key for the data users based on user's attributes and performing revocation operations when data users lose certain attributes. Each DA maintains a binary tree of user attributes, including users requesting keys from them, providing key delegation, and user revocation capabilities related to specified sets of attributes. When a user requests attribute keys, they can choose any DA that supports that attribute and obtain the corresponding key.
- CSP: The CSP has almost immeasurable storage and computational resources and it provides storage, computation, and data access services. It generates decryption tokens for ciphertext based on keys distributed by authorities to users, assisting users in pre-decrypting ciphertext. Most

expensive decryption operations can be outsourced to CSP, ensuring no leakage of information. After the attribute revocation is revoked, the CSP updates the ciphertext.

- Data owner: The owner first operates on the data using symmetric encryption techniques; then, the user customizes the data access policy and encrypts the symmetric key according to the policy; finally, the owner stores all ciphertexts to the cloud server.
- Data user: Users who want to access data for different purposes can use their keys to have the CSP generate decryption tokens. After receiving the token, users use it to decrypt. Data access control does not depend on the CSP; instead, the ciphertext can be accessed for any legitimate users, allowing any legitimate user to freely query any ciphertext of interest from the cloud server. However, access control occurs in cryptography, meaning the server can generate correct decryption tokens, and the user can only decrypt the ciphertext if its property meets the access policy defined in the ciphertext and is not on the revocation list.

Formal Definition of FUR-HABE

FUR-HABE is mainly composed of a set of algorithms: CASetup, DASetup, RegisterUser, SKeyGen, Encrypt, TKGen, Decrypt, UKeyGen, KEKUpdate, and CTUpdate. The form is defined as the following.

The system initialization stage consists of three steps: CA initialization, DA initialization, and user registration. CA manages this stage. First, it generates the public parameters of the system and its master key; then, it initializes multiple DAs to alleviate the burden of creating the key and removing the user; additionally, it provides registration function for data owners to generate their private keys and public keys.

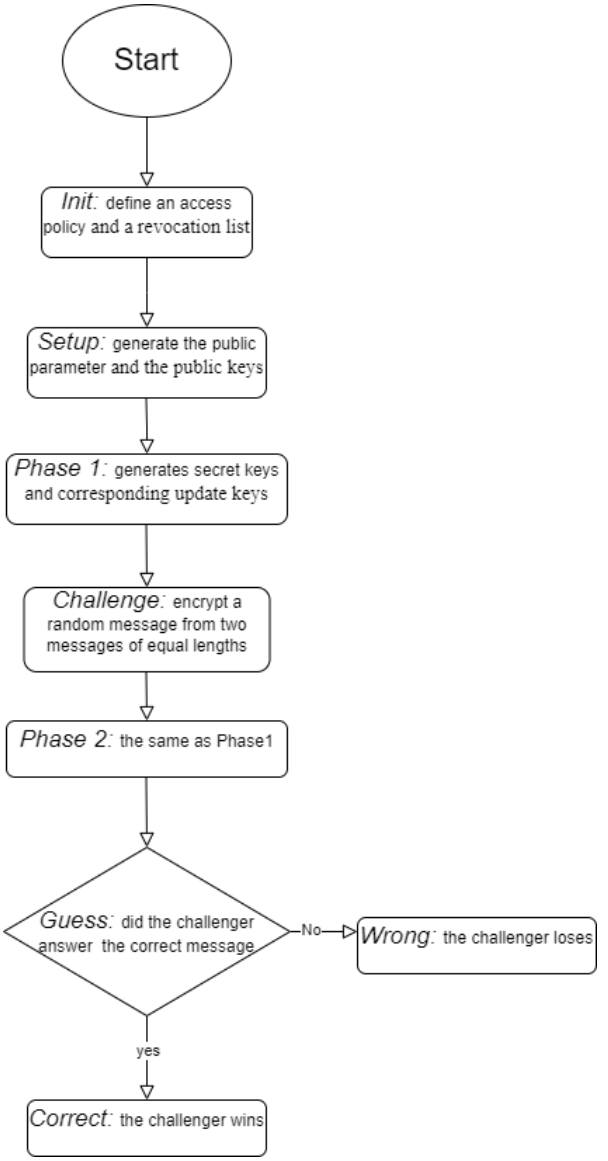
- CASteup $(1^{\lambda}, S) \rightarrow (PP, MSK)$: The algorithm is run by the CA. It generates global public parameters PP and a master secret-key MSK by inputting security parameter and universal attribute set S.
- $DASetup\left(PP, MSK, \left\{id_j\right\}_{j=1}^{n}, \left\{S_j\right\}_{j=1}^{n}, T\right) \rightarrow \left\{SK_j, PK_j\right\}_{j=1}^{n}$: This algorithm is run by the CA. Input PP, MSK, DA's identity $\left\{id_j\right\}_{j=1}^{n}$, subset of S $\left\{S_j\right\}_{j=1}^{\bar{n}}$, and binary tree T; then, output the set of DA key pairs
- RegisterUser(PP,u,MSK)$\rightarrow GPK_u, GSK_u$: The algorithm is run by the CA. It generates a global key pair for each user uid.
- SKeyGen(PP,SK_j,GPK_u,$S_{u,j}$)$\rightarrow \left(USK_{u,j}\right)$: Every DA can run the secret key generation algorithm. It uses PP, the authority secret key SK_j, user public key GPK_u, and attributes set $S_{u,j}$, which describes the key. It outputs the user's secret key, $USK_{u,j}$.
- Encrypt$\left(PP, \left\{PK_j\right\}_{j\in IA}, M, W, RL\right)\rightarrow$ (CT): Encryption algorithm is executed by data owners. It takes as inputs the PP, a set of public keys $\left\{PK_j\right\}_{j\in IA}$ from the involved authority set I_A, a message M, access structure W, and the user revocation list. Based on the access structure W, the message is encrypted, and the revocation list is added to the ciphertext. It outputs a ciphertext CT.
- TokenGen$\left(PP, CT, GPK_u, \left\{USK_{u,j}\right\}_{j\in IA}\right) \rightarrow$ (TK): The cloud server runs decryption token generation algorithm. It inputs PP, the ciphertext CT, user's global public key GPK_u, and a set of user's secret keys, $\left\{USK_{u,j}\right\}_{j\in IA}$. If the attribute set S meets the access structure A, the user can successfully obtain the correct decryption token TK.
- Decrypt(CT,TK,GSK_u)$\rightarrow$ (M): It is up to the user to run the decryption algorithm. It takes in the CT, the decryption token TK, and the user's global secret key GSK_u, and outputs message m.
- UKGen(PP,SK_j,PK_j,G_{att_i})$\rightarrow\left(KUKS_{att_i,j}, CUK_{att_i,j}\right)$: The DA runs the update key generation algorithm about the revoked attribute att_i. It inputs PP, the authority public key PK_j, the authority secret key SK_j, and the set of KEK tree user nodes G_{att_i} that have this attribute and have not been revoked. It outputs the user's key update key component $KUKS_{att_i,j}$ encrypted by the KEK key and the cipertext update key component $CUK_{att_i,j}$.

- Skupdate $\left(PP, USK_{u,j}, att_i, PK_j, KUKS_{att_i,j}, KEK_u\right) \rightarrow \left(USK'_{u,j}\right)$: All unrevoked users run the key update algorithm. It inputs PP, the current secret key $USK_{u,j}$, revocation attribute att_i, authoritative public key PK_j, key update key $KUK_{att_i,j}$, and the user path secret key KEK_u. Then, output is the new secret key $USK'_{u,j}$.
- CTUpdate $(CT, CUK_{att_i,j}) \rightarrow$ (CT'): The cloud server runs the ciphertext update algorithm. It inputs the current ciphertext CT and the ciphertext update key component $CUK_{att_i,j}$. Then, output is a new ciphertext CT'.

Security Model

The FUR-HABE security model is now described in the following game between a challenger *B* and an adversary *A*.

Figure 3. Security model



The concrete steps are as follows:

Init: A defines an access policy (M^*, ρ^*), and a revocation list RL^*, and ρ will each row M^* one-to-one correspondence with the attribute in S.

Setup: Challenger B executes the CASetup and DASetup algorithm and makes a request to authority DA_k, then generates the public parameter PP and the public keys PK_k, and sends to A.

Phase 1: A asks B about the decryption keys of attribute sets (u, S_u). B gives the generated secret keys to A. A asks B about update user attributes, and B responds with the corresponding update keys to A.

Challenge: A gives B two messages of equal length, m_0 and m_1, and B randomly picks a message m_b $(b \in 0,1)$ and then encrypts it with the access policy (M^*, ρ^*) and the revocation list RL^*. Lastly, B sends the ciphertex CT_b to A.
Phase 2: Phase 2 is the same as Phase 1.
Guess: A outputs the guessed b' of b. If $b = b'$, then A wins.

A's advantage in winning the game is defined as: $\varepsilon = \left| Pr[b = b'] - \frac{1}{2} \right|$.

Definition 2

If the advantage in the game for all polynomial-time adversaries is at most negligible, then the FUR-HABE scheme is indistinguishable from a chosen-plaintext attacks under a selective access policy.

CONSTRUCTION OF FUR-HABE

In this part, the constitution and concrete steps of the scheme are introduced. The scheme utilizes CP-ABE access control policies for privacy protection, and KEK trees are used for revocability. There are two parts of ciphertext: one is related to the revocation list, and the other is related to the access policy. Decryption and recovery of messages are only possible if the user's attributes meet the access policy and they are not in the revocation list. As depicted in Figure 2, the scheme comprises five stages: system initialization, key delegation, data encryption, data decryption, and revocation. For ease of exposition, see Table 1 for the meaning of the symbols. Algorithms are described in detail in the following sections.

Table 1. System parameter notion

Notation	Meaning
p	The large prime
$\mathbf{e}$	The bilinear map
$\mathbf{G}$	The cyclic groups
Z_p	The group of integers with prime order p
H	The secure hash function
1^λ	Security parameters
$\mathbf{PP}$	The system parameters
S	The system attribute set
att_i	The attribute in system attribute set
u	The system user
MSK	The master secret key
T	The binary tree in DA
id	The authoritative identifiers of DA
SK_j	The authority secret key of DA
PK_j	The authority public key of DA
GPK_u	The global public key of user

continued on following page

Table 1. Continued

Notation	Meaning
GSK_u	The global secret key of user
USK	The user secret key
M	The plaintext data
W	The access structure
RL	The user revocation list
TK	The decryption token
I_{A_j}	The attributes subset of DA
KUKS	The key update key
CUK	The ciphertext update key
KEK_u	The path node secret value

Setup

The system initialization stage consists of three steps: CA initialization, DA initialization, and user registration. CA manages this stage. First, it generates the public parameters of the system and its master key; then, it initializes multiple DAs to alleviate the burden of creating the key and removing the user; additionally, it provides registration function for data owners to generate their global unique private keys and public keys.

Key Delegation

At this stage, the authority generates the user's key based on the data user's attributes. When a data user possesses the attribute $att_i \in S$, they need to request the DA that supports this attribute to generate their key component. When DA receives a request to generate the key from a data user, it first checks whether the user has the attribute att_i; if not, it aborts.

Encrypt

First, the data owner use symmetric encryption method to encrypts the plaintext data with the content key. Then, the content key is encrypted by encrypt algorithm.

Decrypt

All legitimate users can query any data of interest to them from the cloud. However, only if the user's property meets the access structure in the ciphertext can a decryption token be generated that can be used for further decryption. The following two steps illustrate the decryption phase.

1. Generation decryption token of CSP: This stage is executed by the cloud server for semi-outsourced decryption. User u sends his/her key to the CSP and asks the CSP to run to generate token of the ciphertext CT. Only if the property owned by user u meets the access structure defined in the ciphertext CT and is not in the revocation list is the server then able to compute the correct decryption token TK successfully.
2. Decryption of user data: Once this TK is received, the user can use its global secret key to decrypt the ciphertext. The user then decrypts the encrypted data using the content key k.

Revocation

Suppose DA_j revokes the attribute att_i of user u. Attribute revocation involves three stages: the DA generates update components, unrevokes users update keys, and the cloud server updates ciphertext. Key renewal prevents revoked users from accessing data. When a newly added user's attribute meets an access policy associated with the ciphertext, ciphertext updates can ensure that they can still access previously published data before joining the system.

1. **Update key generation by DA**: The associated authority DA_j runs the UKeyGen algorithm. First, it generates a new attribute key x'_{att_i} and calculates the attribute update key, which it then uses to generate the user's key update key. Then, execute algorithm *Path(u)* and algorithm *cover(R)*, using the KEK key to encrypt the key to $KUKS = H_{KEK}(KUK)$. The ciphertext update key is CUK_{att_i}. Then, DA_j broadcasts a message to all of that attribute's owners, updating the public attribute key of att_i.
2. **Unrevoked user of key update**: For each unrevoked user possessing the attribute att_i, obtain the key update key KUKS from DA_j. Utilizing the algorithm *Path(u)* and algorithm *cover(R)*, compute *Path(u)*∩*cover(R)*, where there is a unique intersection between the two sets. Calculate the decryption KEKS using the KEK value of the leaf node. Users can get the new update key for revoked attributes from DA_j. There is no need for the authority to distribute to users one by one, which can significantly improve the attribute update's efficiency.
3. **Ciphertext update by cloud server**: There are two scenarios for updating ciphertext: One is when only user revocation is conducted, where the CSP updates only the RL in the ciphertext, resulting in an updated ciphertext CT' related to the latest RL. The other scenario involves attribute revocation: DA_j sends cloud the ciphertext update key CUK_{att_i}. Once CUK_{att_i} is received, the server runs the CTUpdate algorithm to update the ciphertext. It takes the current ciphertext CT and CUK_{att_i} as inputs. All it has to do is update the ciphertext components associated with the revocation of att_i.

In the scheme, it is necessary to update only a handful of components associated with the revoked attribute, while the rest of the components remain unchanged. This can significantly increase the efficiency of revocation of the attribute.

Trace

This algorithm is performed by authoritative authority. If the key fails the integrity check, the algorithm aborts and outputs ⊥. Otherwise, the algorithm is executed as follows:

1. Compute $u_u = Dec_\gamma(K')$ to recover the value u_u which is the associated with user u.
2. Find u_u from the associated user group G_x of attribute att_i and then output the user u associated with u_u. If not found, the algorithm aborts and outputs ⊥.
3. If $u \notin RL$, then add u to the revocation list RL and update the new revocation list to $RL' = RL \cup \{u\}$.

```
Algorithm 1: CASetup()
Input: S and 1λ
Output: PP and MSK
1: Select two cyclic groups G, and GT, where p is a prime
order, and g is the generator of G;
2: Define an asymmetric map e: G×G→GT,
3: a ← Zp,
4: while each atti in S
```

do Xi ← Zp, Yi←gXi;
5: Define a secure hash functions H: 0,1*→G;
6: PP ← e,G,GT,g,ga,Yii∈S,H, MSK← a,Xii∈S;
7: return PP and MSK.
Algorithm 2: DASetup()
Input: PP, MSK, idjj=1n, Sjj=1n and T
Output:SKj,PKjj=1n
1: while j in idjj=1n
do αj← Zp, βj← Zp,γj← Zp;
2: while j in idjj=1n
do PKj← (eg,g)αj,g1βj,gγjβj,T, SKj ←αj,βj,γj,Xii∈Sj;
3: return SKj,PKjj=1n.
Algorithm 3: RegisterUser()
Input: PP, u and MSK
Output: GPKu and GSKu
1: for user uu← Zp, zu← Zp;
2: GPKu←uu,guu,g1 zu, GSKu←zu;
3: return GPKu and GSKu.
Algorithm 4: SkeyGen()
Input: PP,SKj, GPKu and Su,j
Output: USKu,j
1: t←Encγuu,
2: Ku,j'←t, Ku,j←gαj zu, Ku,j″←gauu, Lu,j'←g a βjtu,j, Lu,j←g βj zu, Ru,j←gatu,j;
3: while each atti in S
do Ku,atti←gβjγjzutu,j·(gvatti·Hatti)βjγjuu,
4: SKu,j, Ku,j←(Ku,j', Ku,j, Ku,j″, Lu,j', Lu,j, Ru,j,∀atti∈Su,j:Ku,atti)
5: return USKu,j.
Algorithm 5: Encrypt()
Input: PP, PKjj∈IA, M, W and RL
Output: CT
1: s,y2,…,yn← Zp
2: v→←s,y2,…,yn
3: for each i∈q,l do λi←v→·Mi
4: r1,r2,…,rl←ZP
5: C←K·(e(g,g)αj)s, C′←gs, C″←gs βj;
6: for i←1 to l
do Ci←gaλi·((gvρiHρi)γj)−ri, D1,i←gri βj,D2,i←g−γj βjri;
7: CT←(C, C′, C″, ∀i=1to l: Ci, D1,i, D2,i, ρi∈DAj,RL),
8: return CT.
Algorithm 6: TokenGen()
Input: PP, CT, GPKu and USKu,jj∈IA
Output: TK
Let IAjj∈DA is all the attribute sets related to ciphertext, where IAj is the attributes subset of DAj
1: IAj←i:ρi∈SAj, I←|IAj|;
2: for each i in I
do wi← Zp;
3: if λi is valid in M

```
   do s←∑i∈Iwiλi
   4: for each i in IAj
   do TK1 ←(eCi,GPKu·eD1,i,Ku,ρi·eD2,i,Lu,j)wiNA
   5: for each k in IA
   do eC′,Ku,j·e(Ru,j,C″)−1/TK1→TK
   6: return TK.
   Algorithm 7: Decrypt()
   Input: CT, TK and GSKu
   Output: M
   1: k←C/TKzu
   2: use k do decrypt CT→M
   3: return M.
   Algorithm 8: UKGen()
   Input: PP, SKj, PKj and Gatt_i
   Output: KUKSatti,j and CUKatti,j
   1: AUKatti←λjxatti'−xatti
   2: KUKatti←gβj·AUKatti
   3: KUKS←HKEKKUK
   4:CUKatti←βjγjAUKatti
   5: return KUKSatti,j and CUKatti,j.
   Algorithm 9: Skupdate()
   Input: PP,USKu,j,atti,PKj,KUKSatti,j,KEKu
   Output: USKu,j'
   1: Ku,j'←Ku,j,Lu,j'←Lu,j,Ru,j'←Ru,j,Ku,atti'←Ku,atti·KUKattiu,
   2: for each att in Sj
   do if att≠atti
   do Ku,j'=Ku,j
   3: USKu,j'←(Ku,j', Lu,j', Ru,j', Ku,atti',
∀att∈Sj,att≠atti:Ku,j')
   4: return USKu,j'.
   Algorithm 10: CTUpdate()
   Input: CT,CUKatti,j
   Output: CT'
   1: C←k·(∏k∈IAe(g,g)αj)s, C′←gs, c″←gsβj
   2: for i←1 to l
   do if ρi≠atti
   do  Ci←gaλi·((gxattiHatti)λj)−ri, D1,i←griβj, D2,i←g−λjβjri
   else if ρi=atti
   do Ci'=Ci·D2,iCUKatti,D1,i=griβj,D2,i=g−λjβjri
   3: CT←(C, C′, c″, ∀i=1 to l: if ρi≠atti: Ci, D1,i,
D2,i;if ρi=atti:Ci', D1,i, D2,i)
   4: return CT'.
```

SECURITY ANALYSIS

In this section, we demonstrate the IND-CPA security of FUR-HABE based on the assumption of q-BDHE.

Theorem 1

If the q-BDHE hardness assumption holds, then there are no polynomial-time adversaries that can defeat the FUR-HABE scheme with the not inconsiderable advantage under the chosen plaintext attacks.

Proof: if there is an adversary A that can break the FUR-HABE scheme with a non-negligible advantage ε, then construct a challenger B that can solve the q-BDHE hardness assumption with the advantage $\frac{\varepsilon}{2}$.

Let G and G_T be two multiplication cyclic groups of prime order p, and g is a generator of G. A map e: $G \times G \rightarrow G_T$ is a bilinear map.

Init: A chooses a challenge access policy $W^* = (M^*, \rho^*)$ and a revocation list RL^*, where M^* is a $l^* n$ matrix, and ρ maps a row of M^* into an attribute.

Setup: B runs the *CASetup* and *DASetup* algorithm, and give to A. For each authority DA_j, B randomly chooses $\alpha_j^{'}, \beta_j, \gamma_j \in Z_P$ and implicitly sets $\alpha_j = \alpha_j^{'} + a^{q+1}$ by letting $e(g,g)^{\alpha_j} = e(g^a, g^{a^q})\, e(g,g)^{\alpha_j^{'}}$.

Then, the B programs the random oracle H by building a table. If $H(x)$ is already defined in the table, the oracle returns the same answer as before. Otherwise, begin by choosing a random value d_x. Let X represent the set of indices i, such that $\rho^*(i) = x$. Then, B programs the oracle as $H(x) = g^{d_x} \prod_{i \in X} g^{\frac{a^2 M_{i,1}^*}{b_i}} g^{\frac{a^2 2M_{i,2}^*}{b_i}} \cdots g^{\frac{a^{n^*+1} M_{i,n^*}^*}{b_i}}$.

If $X = \varnothing$, then we have $H(x) = g^{d_x}$. In addition, the response from the oracle is distributed randomly.

B randomly chooses two numbers $\beta_j, \gamma_j \in Z_P$ and the user tree T. Then, it generates the public key of authority DA_j as $PK_j = \left(e(g,g)^{\alpha_j}, g^{\frac{1}{\beta_j}}, g^{\frac{\gamma_j}{\beta_j}}, T\right)$. B can simulate the public attribute keys PK_{att_i} by randomly selecting a version number $v_{att_i} \in Z_P$ as $PK_{att_i} = \left(g^{v_{att_i} + d_{att_i}} \prod_{i \in X} g^{\frac{a^2 M_{i,1}^*}{b_i}} g^{\frac{a^2 2M_{i,2}^*}{b_i}} \cdots g^{\frac{a^{n^*+1} M_{i,n^*}^*}{b_i}}\right)^{\gamma_k}$.

B assigns to A a user identity uid and randomly chooses two numbers, $u_{uid}^{'}, z_{uid} \in Z_p$. Then, it sets $GSK_{uid} = z_{uid}$ and implicitly sets $u_{uid} = u_{uid}^{'} - \left(\frac{a^q}{z_{uid}}\right)$ by setting $GPK_{uid} = g^{u_{uid}^{'}} (g^{a^q})^{-\frac{1}{z_{uid}}}$. B then sends the global key pairs $\left(GPK_{uid}, GSK_{uid}\right)$ to A.

Phase 1: In this phase, B answers secret key queries and update key queries from A. Suppose A makes secret key queries by submitting pairs $\left(uid, S_j\right)$ to B, where S_j is a set of attributes associated with authority DA_j.

B finds a vector $\vec{w} = (w_1, w_2, \ldots, w_{n^*}) \in Z_P^{n^*}$, such that $w_1 = -1$, and, for all i where $\rho^*(i) \in S_k$, we have that $\vec{w} \cdot M_i^* = 0$. By the definition, if an LSSS, such a vector must exist since S_k does not satisfy M^*. B then implicitly defines t by calculating $t = Enc_{\gamma}\left(u_u id\right)$ as $K_{u,j}^{'} = t$ and $t_{uid,j} = r + w_1 a^{q-1} + w_2 a^{q-2} + \cdot + w_{n^*} a^{q-n^*}$.

By setting $L_{uid,j} = (g^{\frac{\beta_j}{z_{uid}}})$, B then constructs $R_{uid,j}$ as $R_{uid,j} = g^{ar} \cdot \prod_{i=1}^{n^*} (g^{a^{q+1-i}})^{w_i}$. From the definition of $g^{u_{uid}}$, we find that $g^{au_{uid}}$ contains a term of $g^{a^{q+1}/z_{uid}}$, which will cancel out with the unknown term in $g^{a/z_{uid}}$ when creating $K_{uid,j}$. Then, B can calculate $K_{uid,j} = g^{\frac{\alpha_j}{z_{uid}}}, K_{uid,j}'' = g^{au_u id}, L_{uid,j}^{'} = g^{\frac{a}{\beta_j} r} \prod_{i=1,\ldots,n^*} (g^{a^{q-i}})^{w_i \frac{\beta_j}{z_{uid}}}$.

In FUR-HABE, a new piece is added to the component $K_{att_i,uid,j}$ in the secret key. Fortunately, the new piece can be easily simulated as $(PK_{att_i})^{\gamma_j}$. Thus, for the calculation of $K_{att_i,uidj}$, if att_i is used in the access structure, B computes as follows.

$$K_{uid,att_i} = (L_{uid,j})^{\gamma_j} \cdot (PK_{att_i})^{\beta_j u_{uid}^{'} + \gamma_j} \cdot (g^{a^q})^{\frac{-\beta_j (v_{att_i} + d_{att_i})}{z_{uid}}} \cdot \prod_{i \in X} \prod_{j=1,\ldots,n^*} (g^{\frac{a^{q+1} r_i}{b_i}})^{-\beta_j \gamma_j M_{i,j}^*}$$

If it has nothing to do with attribute att in the access structure, that is to say there is no i such that $\rho^*(i) = att$. For those attributes, we can let $K_{uid,att_i} = (L_{uid,i})^{\gamma_i} \cdot (GPK_{uid})^{\beta_j \gamma_j (v_{att_i} + d_{att_i})} \cdot g^{\gamma_j^2 (v_{att_i} + d_{att_i})}$. To update key queries, assume A submits pairs of $\left(uid, att_i\right)$. If att_i has a new version number $v_{att_i}^{'}$, and uid is an nonrevoked user, it returns an encrypted key by KEK key as $KUKS_{uid,att_i} = H_{KEK}\left(g^{\beta_j \gamma_j (v_{att_i}^{'} - v_{att_i})}\right)$. Otherwise, it responds "⊥".

Challenge: In this phase, B programs the challenge ciphertect. A gives two messages m_0, m_1 to B. B flips a coin b. It creates and encrypts the m_b with the content key K, $C = K \cdot (\prod_{j \in DA} e(g,g)^{\alpha_j})^s$ and $C' = g^s$, $C'' = g^{\frac{s}{\beta_j}}$

The hard part is simulating the C_i values, as it contains terms that have to be canceled. However, B can choose the secret splitting to cancel them. Intuitively, B will choose random $y_2^{'},\ldots,y_{n^*}^{'}$ and share s using the vector $\vec{v} = \left(s, sa + y_2^{'}, sa^2 + y_3^{'}, \ldots, sa^{n^*-1} + y_{n^*}^{'}\right) \in Z_p^{n^*}$. It also chooses random values $r_1^{'},\ldots,r_l^{'}$. For $i = 1,\ldots,n^*$, let R_i be the set of all $k \neq i$ such that $\rho^*(i) = \rho^*(k)$. The challenge ciphertext components can be generated as $D_{1,i} = (g^{r_i} g^{sb_i})^{\frac{1}{\beta_i}}, D_{2,i} = (g^{r_i} g^{sb_i})^{-\frac{\gamma_i}{\beta_i}}$. From the vector $\vec{v}$, we can construct the share of the secret as $\lambda_i = s \cdot M_{i,1} + \sum_{j=2,\ldots,n^*} \left(sa^{j-1} + y_j^{'}\right) M_{i,j}^*$, and we also have the revocation list RL^*. Then, we can simulate the C_i as:

$$C_i = (g^{v_{\rho^*(i)}} \cdot H(\rho^*(i)))^{\gamma_j r_i} \cdot \left(\prod_{j=1,\ldots,n^*} g^{aM_{i,j}y_j}\right) \cdot (g^{b,s})^{-\gamma_j\left(v_{\rho^*(i)} + d_{\rho^*(i)}\right)} \cdot \left(\prod_{k \in R_i} \prod_{j=1,\ldots,n^*} \left(g^{a^j s\left(\frac{b}{b_i}\right)}\right)^{\gamma_j M_{k,j}^*}, RL^*\right)$$

Phase 2: this is the same as Phase 1.

Guess: finally, A output a guess b' of b. If $b' = b$, B then output 0, indicating that $T = e(g,g)^{a^{q+1}s}$; otherwise, it outputs 1 to show that it considers T to be an element of the random group of G_T.

If T is a tuple, B gives a perfectly simulating, so we have that $Pr[(\vec{y}, T = e(g,g)^{a^{q-1}s} = 0] = \frac{1}{2} + \varepsilon$. If T is a random group element, the message m_b is totally hidden from the adversary, and we have $Pr[(\vec{y}, T = e(g,g)^{a^{q+1}s}) = 0] = \frac{1}{2}$.

Thus, B is able to play a non-negligible advantage in deciding the q-BDHE game.

PERFORMANCE ANALYSIS

In this section, an evaluation and comparison of the functionality and efficiency of the proposed solution and schemes (Wang et al., 2018; Han et al., 2020; Deng et al., 2023) are conducted. Table 2 discusses the functional comparison, where scheme (Deng et al., 2023) introduces two revocation schemes, namely user revocation and attribute revocation. However, this scheme uses indirect revocation performed at fixed intervals. It fails to respond promptly when revocation occurs, leading to certain security vulnerabilities. Even without any attributes to revoke, it still requires updates at fixed intervals, resulting in additional overhead. The scheme from Han et al. (2020) only achieves user-level revocation with ciphertext updates but cannot provide fine-grained attribute revocation. The solution from Wang et al. (2018) offers outsourced partial decryption but can only achieve user revocation by revoking all the user's attributes. Our solution provides flexible key delegation and enables lightweight decryption. Furthermore, our solution can be flexibly revoked, allowing different revocation methods, namely attribute revocation and user revocation, to be chosen based on actual application requirements.

Table 2. Functionality comparison

Scheme	Revocation Type	Revocation Method	Update	Lightweight Decryption	Key Delegation
Wang et al. (2018)	Attribute	Direct	Ciphertext&Key	√	×
Han et al. (2020)	User	Direct	Ciphertext	×	×
Deng et al. (2023)	User&Attribute	Indirect	Ciphertext&Key	×	×
Ours	User&Attribute	Direct	Ciphertext&Key	√	√

E represents the exponential operation in G, G_T . P represents the bilinear pairing operation. M represents the multiplication operation, and l represents attributes number in the access policy. s represents user attributes number. N represents attributes number in the decryption key that satisfies the access policy. r represents the coverage set length. j represents the path length.

Table 3 shows the effectiveness of the proposed solution in comparison with other solutions. Comparing the schemes, it is found that FUR-HABE is more efficient. In the KeyGen and decrypt algorithms of our scheme, the complexity of exponentiation and multiplication operations is lower, leading to higher efficiency. In the encryption algorithm, the decryption efficiency is a little better than that of the proposed scheme because the document (Deng et al., 2023) has only the indirect attribute revocation. In the scheme proposed in the literature (Wang et al., 2018), it is possible for a user to decrypt plaintext by using a combination of ciphertext components and decryption keys, which sacrifices the pairing operation of user attributes and access policies, reducing the security of encryption. In the revocation update algorithm, as the scheme from Han et al. (2020) only implements user-level revocation, the update efficiency is somewhat superior to that of the proposed scheme. Attribute revocation in FUR-HABE is controlled and executed by the CA and DA related to the attributes. The DA encrypts user key update components with KEK and broadcasts them through a public channel, while the ciphertext is updated by the CSP, greatly reducing the workload of the authority. In the comparison between FUR-HABE and the literature (Han et al., 2020; Deng et al., 2023), our scheme is more efficient in user decryption. Due to the additional exponentiation operations in the literature (Deng et al., 2023) and more multiplication operations in the literature (Wang et al., 2018), our scheme is significantly more efficient in the revocation update algorithm. Our scheme does not require generating a one-to-one update key for each user, reducing the burden on the authority. Compared to Deng et al.'s scheme (2023), our scheme is less efficient in terms of encryption and decryption algorithms.

Table 3. Efficiency comparison

Scheme	KeyGen	Encrypt	Decrypt	Update
Wang et al. (2018)	(4+4s)E+M	(3+4l+lr) E+(l+1)M	(1+4n)P+(3+n) E+(3+3n)M	(4+s+lr)E+(3+2s)M
Han et al. (2020)	(2s+j+4) E+(3s+j+3)M	(3+4l+r) E+(l+2)lM	(2+3n)P+(3+n) E+(5+4n)M	$\left(\sum_{j' \in cover(R')}(dept(j') - 1 - dept(j))\right)E$
Deng et al. (2023)	(2sj+1)E+sjM	(2l+3)E	(3n+2)P+(n+2) E+3nM	(2r+6)E+(5+r)M
Ours	(2s+5)E+(4s+4) M	(4l+3)E+(l+1) M	(3n+2)P+(n+1) E+(4n+2)M	3E+8M

Figure 4 and Figure 5, respectively, describe the update time of different revocation mechanisms between the existing schemes (Wang et al., 2018; Han et al., 2020; Deng et al., 2023) and the proposed scheme, as well as comparisons of KeyGen, encryption, decryption, and update. There is also performance and time cost comparison of the algorithms. To analyze the performance of FUR-HABE, we used the Charm framework implemented in Python, which features super-singular elliptic curve groups ("SS512"), an elliptic curve bilinear group with a 160-bit group order and 1024-bit security. All of the tests were done on the VMware Workstation (Version 17.0.0 build-20800274), with a 2.90 GHz Intel Core i5 CPU and running 64-bit Linux Ubuntu 22.04.3 long-term support. All tests are done with Python (Version 3.7.9) and Charm (Version 0.50). In the simulation experiments, the number of user attributes ranged from 10 to 50, and the final results were an average of 50 trials.

Figure 4. Comparison of update times of different undo mechanisms (a)

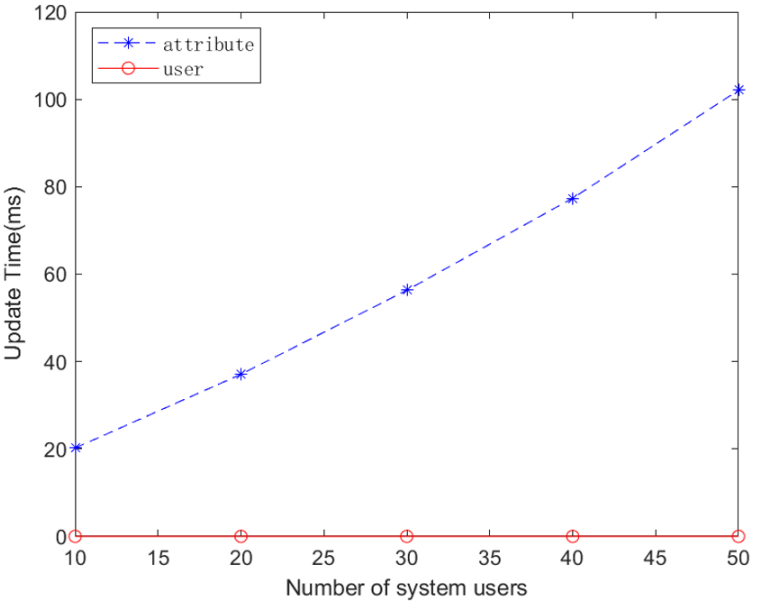
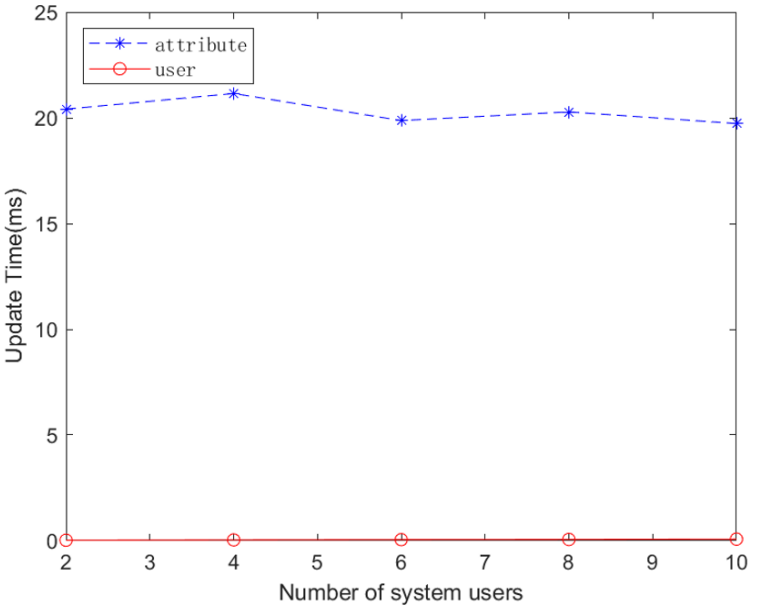


Figure 5. Comparison of update times of different undo mechanisms (b)



Although attribute-level revocation can achieve fine-grained access control and generally outperforms user-level revocation, we stress that it is important in user-level revocation in specific revocation situations. We conducted the experiments to illustrate how user-level undo demonstrates advantages over attribute-level undo in certain scenarios. In Figure 3, we show the average update

time costs for the two revocation types, with the number of attributes revoked ranging from 10 to 50. Additionally, we fixed the number of system users at N=16, with only one user being revoked. In order to achieve simulation attribute-level revocation, we updated each attribute user set involved in the revoked user and set the user revocation list as RL=$\{u_i\}$. In Figure 4, the independent variable was change from 2 to 1, system users count was N=16, and the number of attributes for each revoked user was 10. We assume that every user that has been revoked has identical attributes. Likewise, we update the attribute set for revocation at the attribute level. For user-level revocation, for $RL = u_{i\,|i\in}$ $_{|U|}$, from Figure 3 and Figure 4, we can see that, when handling a revoked user, the algorithm directly revokes it instead of revoking all its attributes, costing much less time. Therefore, it is necessary and practical to combine attribute level revocation with user level revocation.

Compared to existing solutions, FUR-HABE is most effective in achieving key delegation, revocation, and burden alleviation. In Figure 5, we demonstrate the time costs of the KeyGen algorithm for this scheme compared to the aforementioned scheme. Our scheme has the shortest time and highest efficiency while associating the decryption key with the KEK. Next, in Figure 6, the time costs of the encryption algorithm are compared. We can get that, although the encryption time of our scheme is higher than the one proposed by Deng et al. (2023), it is also more efficient and achieves direct revocation and hierarchical key delegation. In Figure 7, we compare the time costs of the decryption algorithm. Both our scheme and the one proposed in Wang et al. (2018) use lightweight decryption methods, resulting in significantly lower decryption overhead for users compared to other solutions, with clear advantages. Furthermore, in terms of update performance, the proposed scheme exhibits the least attribute update time compared to the schemes in Figure 8 (Wang et al., 2018; Deng et al., 2023). It consumes more time than the solution in Han et al. (2020), but the solution proposed there only achieves user-level revocation and unable to implement fine-grained attribute control. In conclusion, it can be concluded that the overhead of our scheme is acceptable for the effectiveness and practical implementation of its features.

Figure 6. KeyGen time analysis

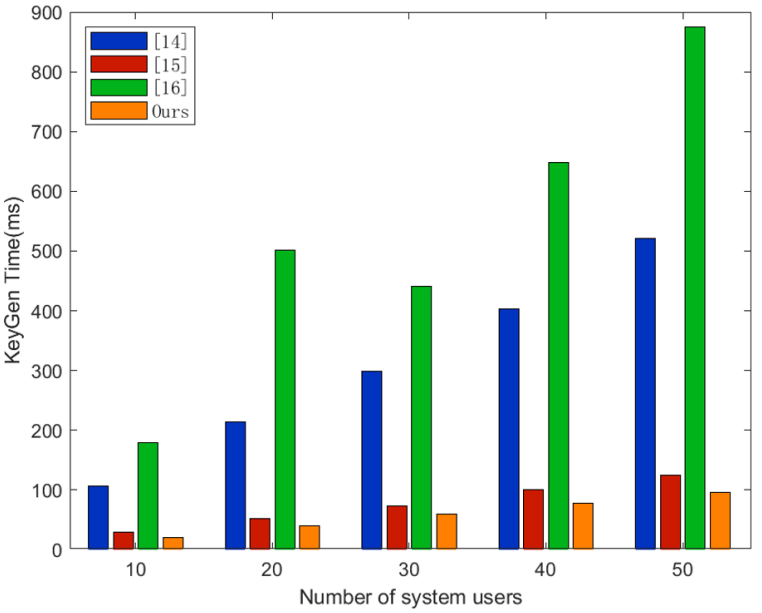


Figure 7. Encrypt time analysis

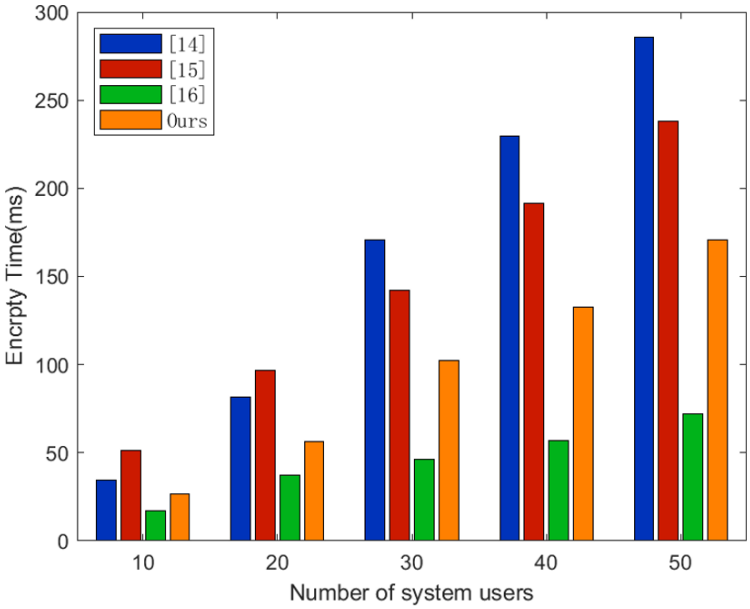


Figure 8. Decrypt time analysis

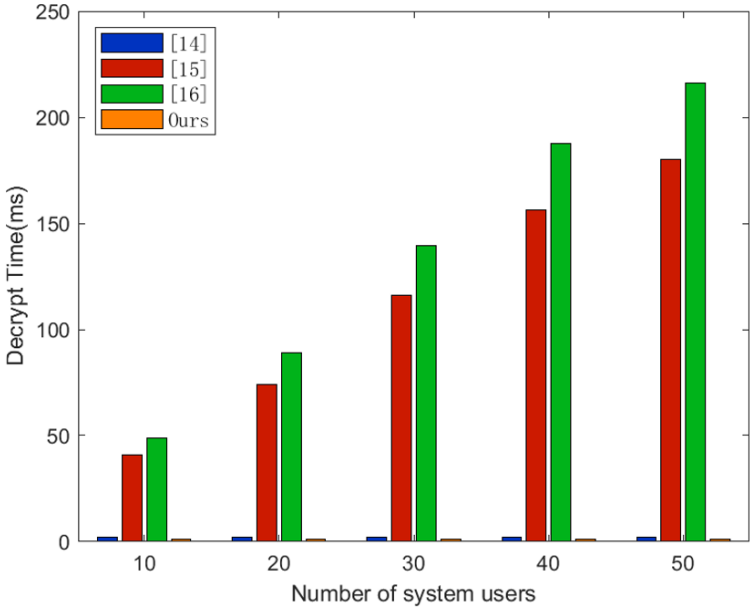
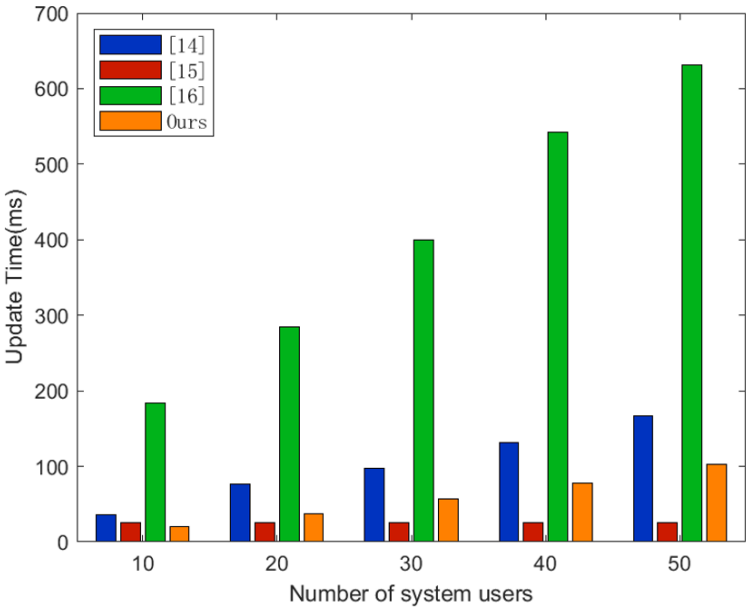


Figure 9. Update time analysis



CONCLUSION

This paper presents an encryption scheme, FUR-HABE, based on privacy protection, traceable hierarchical key delegation, and effective KEK revocation. In FUR-HABE, multiple authorities are involved in the key delegation, allowing the users to outsource the bulk of the decryption computations to the cloud. It also supports both user revocation and attribute revocation. The outsourcing of decryption, attribute revocation, and KEK update revocation reduces the computational burden on users and authoritative entities. Based on the q-BDHE assumption, we provide security proof for this scheme. Both theoretical analysis and experimental results demonstrate its effectiveness. In the future, we can focus on how to combine this system with single sign-on functions to improve user experience and further enhance data security, such as researching how to achieve seamless user access while maintaining data security; and user experience optimization, analyzing user feedback on the existing system and designing a user-friendly authentication process to reduce the user's operational burden.

FUNDING STATEMENT

This research was supported by the Natural Science Foundation of Hebei Province of China [F2021201052].

COMPETING INTERESTS

The author of this publication declares that there are no competing interests.

PROCESS DATES

Received: June 30, 2024, Revision: October 24, 2024, Accepted: November 30, 2024

CORRESPONDING AUTHOR

Correspondence should be addressed to Ya'nan Tao; taoy88115@gmail.com

REFERENCES

Akinyele, J. A., Garman, C., Miers, I., Pagano, M. W., Rushanan, M., Green, M., & Rubin, A. D. (2013). Charm: A framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, *3*(2), 111–128. DOI: 10.1007/s13389-013-0057-3

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, *54*(15), 2787–2805. DOI: 10.1016/j.comnet.2010.05.010

Bethencourt, J., Sahai, A., & Waters, B. (2007, May). Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP)* (pp. 321–334). IEEE. DOI: 10.1109/SP.2007.11

Blog, S. How Backblaze's Fireball B2 Moves Big Datasets to the Cloud Easily, Safely and Swiftly.https://blog.seagate.com/enterprises/how-backblazes-fireball-b2-moves-big-datasets-to-the-cloud-easily-safely-and-swiftly/ (2021)

Boneh, D., & Franklin, M. (2001, August). Identity-based encryption from the Weil pairing. In Kilian, J. (Ed.), Lecture Notes in Computer Science: *Vol. 2139. Advances in Cryptology — CRYPTO 2001. CRYPTO 2001*. Springer., DOI: 10.1007/3-540-44647-8_13

Deng, S., Yang, G., Dong, W., & Xia, M. (2022). Flexible revocation in ciphertext-policy attribute-based encryption with verifiable ciphertext delegation. *Multimedia Tools and Applications*, *82*(14), 22251–22274. DOI: 10.1007/s11042-022-13537-0

Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006, October). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06* (pp. 89–98). Association for Computing Machinery, New York, NY, USA. DOI: 10.1145/1180405.1180418

Han, D., Pan, N., & Li, K. C. (2020). A traceable and revocable ciphertext-policy attribute-based encryption scheme based on privacy protection. *IEEE Transactions on Dependable and Secure Computing*, *19*(1), 316–327. DOI: 10.1109/TDSC.2020.2977646

Han, F., Hu, J., & Xi, K. (2010, June). Highly efficient one-time pad key generation for large volume medical data protection. In *2010 5th IEEE Conference on Industrial Electronics and Applications, 54* (pp. 330–335). DOI: 10.1109/ICIEA.2010.5516860

Huang, X., Xiong, H., Chen, J., & Yang, M. (2021). Efficient revocable storage attribute-based encryption with arithmetic span programs in cloud-assisted internet of things. *IEEE Transactions on Cloud Computing*, *11*(2), 1273–1285. DOI: 10.1109/TCC.2021.3131686

Hur, J., Koo, D., Hwang, S. O., & Kang, K. (2013). Removing escrow from ciphertext policy attribute-based encryption. *Computers & Mathematics with Applications (Oxford, England)*, *65*(9), 1310–1317. DOI: 10.1016/j.camwa.2012.02.005

Hur, J., & Noh, D. K. (2010). Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, *22*(7), 1214–1221. DOI: 10.1109/TPDS.2010.203

Jiang, Y., Xu, X., & Xiao, F. (2022). Attribute-based encryption with blockchain protection scheme for electronic health records. *IEEE Transactions on Network and Service Management*, *19*(4), 3884–3895. DOI: 10.1109/TNSM.2022.3193707

Li, J., Chen, N., & Zhang, Y. (2019). Extended file hierarchy access control scheme with attribute-based encryption in cloud computing. *IEEE Transactions on Emerging Topics in Computing*, *9*(2), 983–993. DOI: 10.1109/TETC.2019.2904637

Li, J., Yao, W., Zhang, Y., Qian, H., & Han, J. (2016). Flexible and fine-grained attribute-based data storage in cloud computing. *IEEE Transactions on Services Computing*, *10*(5), 785–796. DOI: 10.1109/TSC.2016.2520932

Li, M., Yu, S., Zheng, Y., Ren, K., & Lou, W. (2012). Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems*, *24*(1), 131–143. DOI: 10.1109/TPDS.2012.97

Liu, Z., Liu, Y., Xu, J., & Wang, B. (2020). Verifiable attribute-based keyword search encryption with attribute revocation for electronic health record system. *International Journal of Network Security*, *22*(5), 845–856. DOI: 10.6633/ijns.202009_22(5).15

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing (National Institute of Standards and Technology, Gaithersburg, MD)*. NIST Special Publication., DOI: 10.6028/NIST.SP.800-145

Miao, Y., Deng, R. H., Liu, X., Choo, K. R., Wu, H., & Li, H. (2019). Multi-authority attribute-based keyword search over encrypted cloud data. *IEEE Transactions on Dependable and Secure Computing*, *18*(4), 1667–1680. DOI: 10.1109/TDSC.2019.2935044

Naor, D., Naor, M., & Lotspiech, J. (2001). Revocation and tracing schemes for stateless receivers. In J. Kilian (Ed.), *Advances in Cryptology, CRYPTO 2001 - 21st Annual International Cryptology Conference, Proceedings* (pp. 41–62). Springer Verlag. DOI: 10.1007/3-540-44647-8_3

Peñuelas-Angulo, A., Feregrino-Uribe, C., & Morales-Sandoval, M. (2023). Revocation in attribute-based encryption for fog-enabled internet of things: A systematic survey. *Internet of Things : Engineering Cyber Physical Human Systems*, *23*, 100827. DOI: 10.1016/j.iot.2023.100827

Pirretti, M., Traynor, P., McDaniel, P., & Waters, B. (2006, October). Secure attribute-based systems. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (pp. 99–112). DOI: 10.1145/1180405.1180419

Praveen Kumar, P., Syam Kumar, P., & Alphonse, P. J. A. (2018). Attribute based encryption in cloud computing: A survey, gap analysis, and future directions. *Journal of Network and Computer Applications*, *108*, 37–52. DOI: 10.1016/j.jnca.2018.02.009

Rouselakis, Y., & Waters, B. (2015). Efficient statically-secure large-universe multi-authority attribute-based encryption. In Böhme, R., & Okamoto, T. (Eds.), *Vol. 8975*, pp. 315–332). Lecture Notes in Computer Science. Springer., DOI: 10.1007/978-3-662-47854-7_19

Sahai, A., & Waters, B. (2005). Fuzzy identity-based encryption. In Cramer, R. (Ed.), *Lecture Notes in Computer Science, 3494* (pp. 457–473). Springer., DOI: 10.1007/11426639_27

Sun, Y., Mu, Y., Susilo, W., Zhang, F., & Fu, A. (2020). Revocable identity-based encryption with server-aided ciphertext evolution. *Theoretical Computer Science*, *815*, 11–24. DOI: 10.1016/j.tcs.2020.02.031

Tysowski, P. K., & Hasan, M. A. (2013). Hybrid attribute-and re-encryption-based key management for secure and scalable mobile applications in clouds. *IEEE Transactions on Cloud Computing*, *1*(2), 172–186. DOI: 10.1109/TCC.2013.11

Wang, G., Liu, Q., Wu, J., & Guo, M. (2011). Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers & Security*, *30*(5), 320–331. DOI: 10.1016/j.cose.2011.05.006

Wang, S., Guo, K., & Zhang, Y. (2018). Traceable ciphertext-policy attribute-based encryption scheme with attribute level user revocation for cloud storage. *PLoS One*, *13*(9), e0203225. DOI: 10.1371/journal.pone.0203225 PMID: 30212473

Wang, X. K., & Sun, X. (2021, July). CP-ABE with efficient revocation based on the KEK tree in data outsourcing system. In *2021 40th Chinese Control Conference (CCC)*, Shanghai, China, 2021 (pp. 8610–8615). *IEEE*. DOI: 10.23919/CCC52363.2021.9549836

Wei, J., Chen, X., Huang, X., Hu, X., & Susilo, W. (2019). RS-HABE: Revocable-storage and hierarchical attribute-based access scheme for secure sharing of e-health records in public cloud. *IEEE Transactions on Dependable and Secure Computing*, *18*(5), 2301–2315. DOI: 10.1109/TDSC.2019.2947920

Xiao, M., Li, H., Huang, Q., Yu, S., & Susilo, W. (2022). Attribute-based hierarchical access control with extendable policy. *IEEE Transactions on Information Forensics and Security*, *17*, 1868–1883. DOI: 10.1109/TIFS.2022.3173412

Yang, K., Jia, X., & Ren, K. (2013, May). Attribute-based fine-grained access control with efficient revocation in cloud storage systems. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, Hangzhou, China (pp. 523–528). DOI: 10.1145/2484313.2484383

Yang, K., Jia, X., Ren, K., Zhang, B., & Xie, R. (2013). DAC-MACS: Effective data access control for multiauthority cloud storage systems. *IEEE Transactions on Information Forensics and Security*, *8*(11), 1790–1801. DOI: 10.1109/TIFS.2013.2279531

Yu, S., Wang, C., Ren, K., & Lou, W. (2010). Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security* ASIACCS '10. New York, NY, USA: Association for Computing Machinery (pp. 261–270). DOI: 10.1145/1755688.1755720

Yu, S., Wang, C., Ren, K., & Lou, W. (2010). Achieving secure, scalable, and fine-grained data access control in cloud computing. In *2010 Proceedings IEEE INFOCOM*, San Diego, CA, USA (pp. 1–9). IEEE. DOI: 10.1109/INFCOM.2010.5462174

Zhang, Y., Zheng, D., & Deng, R. H. (2018). Security and privacy in smart health: Efficient policy-hiding attribute-based access control. *IEEE Internet of Things Journal*, *5*(3), 2130–2145. DOI: 10.1109/JIOT.2018.2825289

Xiaohui Yang is a Professor at the School of Cyber Security and Computer, Hebei University, China. He received his PhD degree from the University of Science and Technology of China in 2010. He has published several research papers in reputed international journals and conferences. His primary research interests include distributed computing, information security and trusted computing.

Ya'nan Tao is a master's student majoring in Cyberspace Security at the School of Cybersecurity and Computer Science, Hebei University, China. Her main research interests include distributed computing, information security and privacy protection.