Li *et al. Cybersecurity* (2024) 7:18
https://doi.org/10.1186/s42400-024-00211-1

Cybersecurity

RESEARCH

Open Access

Revocable and verifiable weighted attribute-based encryption with collaborative access for electronic health record in cloud

Ximing Li[1], Hao Wang[1], Sha Ma[1*], Meiyan Xiao[1] and Qiong Huang[1]

Abstract

The encryption of user data is crucial when employing electronic health record services to guarantee the security of the data stored on cloud servers. Attribute-based encryption (ABE) scheme is considered a powerful encryption technique that offers flexible and fine-grained access control capabilities. Further, the multi-user collaborative access ABE scheme additionally supports users to acquire access authorization through collaborative works. However, the existing multi-user collaborative access ABE schemes do not consider the different weights of collaboration users. Therefore, using these schemes for weighted multi-user collaborative access results in redundant attributes, which inevitably reduces the efficiency of the ABE scheme. This paper proposes a revocable and verifiable weighted attribute-based encryption with collaborative access scheme (RVWABE-CA), which can provide efficient weighted multi-user collaborative access, user revocation, and data integrity verification, as the fundamental cornerstone for establishing a robust framework to facilitate secure sharing of electronic health records in a public cloud environment. In detail, this scheme employs a novel weighted access tree to eliminate redundant attributes, utilizes encryption version information to control user revocation, and establishes Merkle Hash Tree for data integrity verification. We prove that our scheme is resistant against chosen plaintext attack. The experimental results demonstrate that our scheme has significant computational efficiency advantages compared to related works, without increasing storage or communication overhead. Therefore, the RVWABE-CA scheme can provide an efficient and flexible weighted collaborative access control and user revocation mechanism as well as data integrity verification for electronic health record systems.

Keywords Revocable attribute-based encryption, Data verification, Collaborative access, Electronic health record

Introduction

The electronic health record (EHR) refers to a digital version of a patient's medical record, encompassing comprehensive medical and treatment history. These records enable medical professionals to comprehensively evaluate the patient's physical condition and formulate precise and efficient treatment plans. Electronic health records not only simplify the workflow of medical professionals but also reduce the burden on patients to undergo repeated testing. With the swift advancement of cloud computing and machine learning technology, electronic health records are used by more and more hospitals. In an electronic health record system, medical institutions collect patients' treatment information through artificial or health Internet of Things devices and upload it to the cloud server. Medical personnel can download these records to assist them in their work. As a result of the significant difficulties presented by the storage and management of extensive EHR data, numerous EHR servers are commonly delegated to external cloud computing service

*Correspondence:
Sha Ma
martin_deng@163.com
[1] College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510000, Guangdong, China

providers like Oracle Health. However, these outsourced records contain a substantial amount of patient's private information. To ensure the security of this sensitive data, medical institutions typically employ encryption techniques prior to uploading them onto the cloud server.

Attribute-based Encryption (ABE), initially proposed by Sahai and Waters (2005), not only ensures data security but also offers users with fine-grained access control, thereby enhancing the overall protection and management of user information. Current ABE schemes can be classified into two distinct groups: key-policy ABE (KP-ABE) (Goyal et al. 2006) and ciphertext-policy ABE (CP-ABE) (Bethencourt et al. 2007). In the KP-ABE scheme, each user is assigned a private key that corresponds to an access policy, while every encrypted message is linked to a collection of attributes. It enables users to decrypt only ciphertext that matches the access policy incorporated in their private key. In the CP-ABE scheme, a user's private key is linked to a collection of attributes, while each encrypted message incorporates an access structure based on selected attributes. The decryption of the ciphertext is possible only when the attribute set of the user matches the access structure embedded within the ciphertext. By expressing access policies using attribute-based access structures, users can customize access policies according to their needs. Compared to the KP-ABE scheme, the CP-ABE scheme enables users to flexibly control their data stored on cloud servers, which is discussed in this paper. Consequently, integrating CP-ABE technology into EHR not only ensures data security effectively but also facilitates fine-grained access control.

Motivation and main ideas

So far, there have been several studies (Li et al. 2022b; Zhou et al. 2019) that concentrate on the utilization of different versions of ABE to safeguard the confidentiality of electronic health record (EHR) data when it is stored on cloud servers. However, considering the practical application of EHR, users' health records may require complex collaboration among multiple departments to access and the departmental weights are frequently heterogeneous. For example, *Cardiac Surgery EHR*: A patient's EHR requires collaboration from the cardiology department, surgery department, and archives department to access it to ensure that the record is not for private use. But it also allows the high-weighted dean's office to work with any department to access it to ensure that the record can be used in an emergency. There exist some CP-ABE schemes with multi-user collaborative access that could implement such access requirements. Xue et al. (2019)'s attribute-based controlled collaborative access control scheme (CC-ABE) allows multiple users with different attributes who do not meet the access conditions to get access permission through collaboration by using user grouping and secret value translation technologies. Chen et al. (2022)'s ciphertext-policy attribute-based encryption with shared decryption scheme (CP-ABE-SD) simplifies the computation of multi-user collaborative access by utilizing an integrated access tree (Wang et al. 2016b) as the access control structure, thereby improving the efficiency of multi-user collaborative access. These schemes do not take into account the different weights of collaboration users. Therefore, when implementing the access control condition of *Cardiac Surgery EHR*, it is inevitable to establish the access tree depicted in Fig. 1, which contains redundant attributes "Cardiology", "Surgery" and "Archives". Since the CP-ABE scheme needs to operate on all attributes on an access tree, these redundant attributes inevitably reduce the effectiveness of the CP-ABE scheme.

Extending the access control condition for *Cardiac Surgery EHR*, assuming that N represents the number of distinct attributes required for collaborative access and S denotes a special attribute without being included in the previous set of N, we derive the ensuing access control condition.

- Sub-policy I: When the special attribute S does not participate in collaborative access, the other N different attributes can collaborate to gain access permission. (For example, the access tree in the blue box in Fig. 1.)
- Sub-policy II: When the special attribute S participates in collaborative access, it can collaborate with a few (less than $N-1$) other attributes to gain access permission. (For example, the access tree in the red box in Fig. 1.)

The complex access policy mentioned above, which includes both Sub-policy I and Sub-policy II, is abstracted as the special attribute policy (SAP). Obviously, using existing schemes to solve the problem of implementing SAP inevitably produces redundant attributes, which

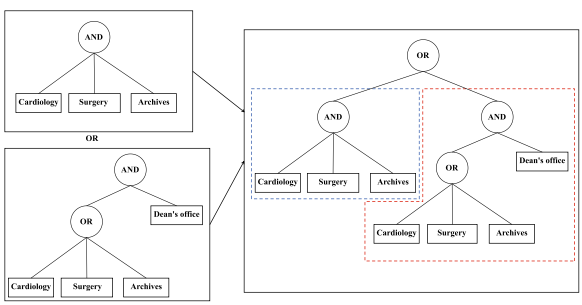


Fig. 1 Access tree for cardiac surgery EHR

would definitely result in a decrease in the effectiveness of the CP-ABE.

To efficiently and effectively address the SAP problem, we propose utilizing the attribute weighting method to improve the existing multi-user collaborative access scheme. We use a weighted threshold secret sharing scheme based on Lagrangian Interpolation (Shamir 1979) to achieve the attribute weighting for SAP. In detail, the number of secrets associated with an attribute is contingent upon the weight assigned to that attribute. For instance, if the weight of an attribute is denoted as n, it will be allocated n unique identifiers along with their corresponding secrets. Through this technique, we can build the *Weighted Multi-user Collaborative Access Tree* shown in Fig. 2 for the *Cardiac Surgery EHR*. As shown in the figure, the double wire frame node indicates that it is used for collaborative access, which is associated with the numbers of secret values s_i and transition values t_i according to the weight w_s. Different colored $\{s_i, t_i\}$ pairs represent these values held by different users. The secret value s_i can be converted into s_i^* that can be used for collaborative access by using the transition value t_i through the transformation function Tr. The value in the root node signifies the threshold, and the node can be successfully accessed when the sum of participating nodes' weights is greater than or equal to the threshold. As depicted in the figure, the weights assigned to "Cardiology", "Surgery", "Archives" and "Dean's office" are 1, 1, 1, and 2 respectively. Consequently, through collaboration, "Cardiology", "Surgery" and "Archives" can successfully access the root node, and by collaborating with any one of these three departments ("Cardiology", "Surgery" or "Archives"), the "Dean's office" can also gain access to the root node. Compared with the access tree in Fig. 1, our *Weighted Multi-user Collaborative Access Tree* provides an efficient and effective way to eliminate redundant attributes.

On the other hand, data access is not static as it is subject to dynamic changes in user permissions. Thus, supporting user revocation is essential to prevent unauthorized access to encrypted files. In Wei et al. (2021)'s revocable-storage and hierarchical attribute-based access scheme (RS-HABE), a binary tree is employed for the management of multiple time periods. In case of user revocation, an unused time period is re-selected from these time periods, and then this time period is used to update the ciphertext. In Cui et al. (2018)'s attribute-based keyword search with an efficient revocation scheme (AKSER), user revocation is achieved through the encryption version information. Specifically, in the event of a user being revoked, the system undergoes an update to its encryption version details and subsequently generates fresh trapdoors for users who have not been revoked. Only these newly generated trapdoors can successfully access the ciphertext. Inspired by AKSER's approach, our scheme also uses the encryption version information to achieve user revocation. The difference is that our scheme generates ciphertext update information and key update information instead of trapdoors after the user is revoked. These pieces of updated information are then sent to the cloud server and the non-revoked user to update the ciphertext and private key, respectively. Only the user with the updated key, that is, the non-revoked user, can access the updated ciphertext, so as to realize the revocation of the user. Compared with the time period method of RS-HABE, our implementation technology does not need to store and deal with all the time period information but only the current and the previous encryption version information, which gives it a significant efficiency advantage. However, our technology imposes stringent timeliness requirements for key update and ciphertext update. For example, there are three revocation events A, B, and C, and the cloud server is processing event A. In the event of continuous occurrence of events B and C, using our technology, it is imperative for the cloud server to promptly accomplish event A and execute event B in order to prevent the encrypted information of event B from being overwritten by that of event C. The time period method of RS-HABE, in contrast, simultaneously stores time period information for events A, B, and C, thereby allowing for a more relaxed requirement on timeliness. Therefore, we need the key and ciphertext updating algorithm of the proposed scheme to have high computational efficiency.

Considering that the authority of ciphertext updating is given to the cloud server, which may not perform ciphertext updating correctly, it is necessary to authenticate the data integrity of the updated ciphertext. To accomplish this, we introduce a verification server that maintains a Merkle Hash Tree (MHT) of the ciphertext with the help of the encryption user. The cloud server also maintains a same MHT. The auxiliary authentication information is transmitted by the cloud server to the verification server

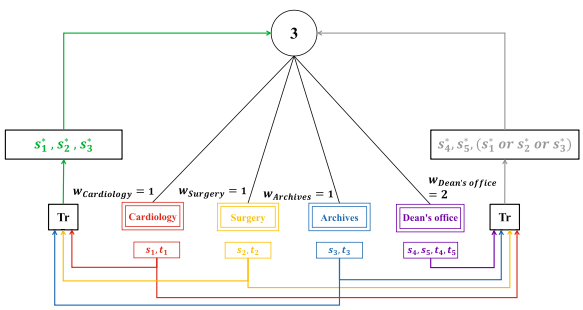


Fig. 2 Weighted multi-user collaborative access tree

for the purpose of verifying the integrity of the ciphertext. The verification server uses the MHT verification algorithm to verify whether the auxiliary authentication information is from the correct ciphertext. Hence, the scheme we propose guarantees the preservation of ciphertext integrity on the cloud server.

Main contribution

This paper introduces the special attribute policy (SAP) problem of weighted multi-user collaborative access, which addresses how to incorporate multiple sub-access policies in one access control structure by changing the necessary number of attributes required for collaboration through a special attribute. In order to effectively solve SAP problems and realize user revocation and data integrity verification in the EHR system, this paper presents the following contributions.

(1) We propose a revocable and verifiable weighted attribute-based encryption with collaborative access scheme (RVWABE-CA) to implement multi-user collaborative access for special attribute policy (SAP) while ensuring user revocation and data integrity verification. The proposed scheme effectively eliminates redundant attributes resulting from the merging of multiple access policies in SAP by assigning appropriate weights to each attribute. Additionally, user revocation is achieved through updating the encryption version information, while data integrity verification is ensured using Merkle Hash Trees.
(2) We prove that the RVWABE-CA scheme has the security of indistinguishability under chosen plaintext attack (IND-CPA) under the decisional bilinear Diffie-Hellman (DBDH) assumption in standard model.
(3) We conduct an experimental comparison of our scheme with related schemes. The results show that except for a slightly lower efficiency in the key update algorithm, our scheme has significant advantages in terms of the computational efficiency of encryption, decryption, ciphertext update, and collaborative decryption algorithms without sacrificing storage overhead.

Organization of this paper

The following sections of this paper are structured as follows. In Section , we provide a review of CP-ABE and the utilization of attribute-weighted and collaborative access control in CP-ABE. In Section , we introduce some fundamental concepts including the bilinear map, DBDH assumption, Lagrange Interpolation, and Merkle Hash Tree. The system model, algorithm, and security model are defined in Section , while the intricate details of our proposed RVWABE-CA scheme are presented in Section . In Section , we prove the security of our proposed scheme under the DBDH assumption. We provide theoretical analysis and experimental results in Section before concluding with a summary in Section .

Related work

The attribute-based encryption (ABE) scheme is widely acknowledged as a powerful cryptographic technique, capable of providing precise access management for information. Existing ABE schemes can be classified as key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE). In 2006, Goyal et al. (2006) proposed the first KP-ABE scheme, which enables the utilization of ciphertext attributes to match the access policy embedded within the private key. In 2007, Bethencourt et al. (2007) introduced the initial CP-ABE scheme, which associates a user's private key with a collection of attributes and incorporates the access policy into the ciphertext. The ciphertext can only be decrypted if the user's attribute set meets the access policy requirements. Subsequently, the works of Hoang et al. (2019a); Lai et al. (2022); Fan et al. (2021), and Chen et al. (2023) enhanced the security of ABE, enabling it to achieve stronger security properties such as forward security, adaptive security, and resistance against chosen-ciphertext attack. On the other hand, the works of Xu et al. (2020); Eltayieb et al. (2019), and Wang et al. (2022) enhanced the effectiveness of ABE encryption and decryption processes. Ciphertext-policy weighted attribute-based encryption (CP-WABE) has the potential to streamline the access structure by minimizing attribute count. In 2016, Wang et al. (2016a) proposed a CP-WABE scheme to reduce the storage and encryption overhead when sharing data within the cloud computing environment. In 2018, based on Wang et al. (2014); Li et al. (2018) used CP-WABE to unify access trees in a cloud computing data sharing scheme. Later Tian et al. (2019) studied hierarchical authority in CP-WABE. In 2020, Yan et al. (2020) implemented a combination of traitor tracing and CP-WABE, and proposed traceable CP-WABE. In 2022, Li et al. (2022a) utilized 0-1 encoding to support the contrast between attributes with assigned weights and the representation of attributes as boolean values in CP-WABE and proposed an efficient CP-WABE for the Internet of Health Things (IoHT). In 2022, Ionita (2022) extended weighted attribute nodes into a tree with binary characters as leaf nodes in order to compare the magnitude between attribute weights and thresholds in the access structure, achieving an efficient weighted ABE system.

To improve the applicability of traditional ABE schemes in real-life situations, extensive research has been conducted on implementing user revocation measures within the context of ABE. In 2009, a revocable KP-ABE scheme was proposed by Attrapadung and Imai (2009). In their research, the revocation of users is implemented through the inclusion of revocation information within ciphertext during the encryption. Later, Zhang et al. (2013) proposed a directly revocable CP-ABE scheme by incorporating a list of revoked entities into the encrypted data. Therefore, the encryptor must frequently request the trusted authority to maintain an updated revocation list. Qin et al. (2019) introduced an alternative revocable ABE scheme that differs from the work of Attrapadung and Imai (2009) and Zhang et al. (2013), in which the ciphertext can only be decrypted by the user who possesses the updated key and the information required for the key update is only sent by the authorization center to the non-revocable user. Cui et al. (2018) implemented an efficient and revocable ABE with the assistance of the server. In this scheme, the server keeps its own random number, which is used to generate the user's trapdoor and participate in matching operations. Only when the random number is the same can the search be successful, and the server directly updates the random number when a user is revoked. In 2019, Hoang et al. (2019b) implemented a forward-secure revocable ABE scheme using integrated re-encryption technology, which guarantees that the data previously shared cannot be decrypted by the revoked user. In 2022, in order to ensure that the ciphertext before and after the cancellation is encrypted by the same plaintext, Ge et al. (2022) introduced a revocable ABE scheme supporting data integrity protection. In 2023, Huang et al. (2023) for the first time combined the revocable ABE with the arithmetic span program to reduce the unnecessary cost of defining access strategies and achieve an efficient revocable ABE. In terms of applications, Huang (2021) implemented revocable ABE for the Internet of Things (IoT) by removing the subset-cover revocation framework, and Wei et al. (2021) used a revocable ABE scheme to ensure the security of e-Health record systems.

Collaborative access control constitutes another research domain that is closely aligned with our current investigation. Many achievements have been made in the research of collaborative access control, such as Alsharef et al. (2020) proposed a collaborative access control framework for online social networks and Edemacu et al. (2020) employed ABE to achieve privacy security in a collaborative e-Health system. In these studies, "collaborative" usually means that users with different permissions can work together within the same system. For example, a user who can only read a document can work with another user who can edit it. The concept of "collaboration" in Li et al. (2014)'s study is most similar to that in our work, and their group-oriented ABE (GO-ABE) solved the problem of collaboration between users with different attribute sets for the first time. To improve security, Xue et al. (2019) proposed an attribute-based controlled collaborative access control scheme (CC-ABE). Compared to GO-ABE, the CC-ABE scheme restricts unauthorized collaborative access between users, not only improving security but also making the scheme more versatile. In 2022, Chen et al. (2022) proposed an efficient CP-ABE scheme with shared decryption (CP-ABE-SD) which uses the integrated access tree proposed by Wang et al. (2016b) to implement multi-user collaborative access. Compared to GO-ABE and CC-ABE, the CP-ABE-SD scheme simplifies the calculation of decryption and achieves higher efficiency. To the best of our knowledge, none of these schemes consider collaborative users with different weights, which leads to redundant attributes in the access control structure, as we analyzed in Section .

Preliminaries

In this section, we give the basic background knowledge of bilinear map, security assumption, Lagrange Interpolation, and Merkle Hash Tree. The notations of the symbols used in this paper are provided in Table 1.

Bilinear map

Let $\mathbb{GP} = \{e, \mathbb{G}, \mathbb{G}_T, g, p\}$, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups with a same prime order p. g is the generator of $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ possesses the following three properties:

Table 1 Notations

Symbol	Description
λ	The security parameter
$\mathbb{G}, \mathbb{G}_T$	Bilinear multiplicative groups with prime order p
$\mathbb{Z}_p$	The group under multiplication modulo p
$\mathcal{T}$	An access tree of a certain access policy
H	A hash function that is resistant to collusion attacks
S	A set of attributes
L	A set of leaf nodes
V	A set of cooperation nodes
Δ	The Lagrange coefficient
PK	The public key of system
MSK	The master secret key of system
SK	The secret key of a user
CT	The ciphertext of a message
Aai_{data}	The auxiliary authentication information of *data*

(1) Bilinearity: There exist $\forall a, b \in Z_p$ and $g_1, g_2 \in \mathbb{G}$, such that we have $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$.
(2) Non-degeneracy: $\forall g_1, g_2 \in \mathbb{G}$ such that $e(g_1, g_2) \neq 1$, where the symbol 1 represents the identity element within the group $\mathbb{G}_T$.
(3) Computability: $\forall g_1, g_2 \in \mathbb{G}$, the computation of $e(g_1, g_2)$ is possible to accomplish within a polynomial time complexity.

DBDH assumption

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map and g be the generator of $\mathbb{G}$. There are four random elements $x_1, x_2, x_3, z \in \mathbb{Z}_p$. The adversary $\mathcal{A}$ aims to differentiate between $e(g,g)^{x_1x_2x_3}$ and $e(g,g)^z$. Given the DBDH tuple $\{e, \mathbb{G}, \mathbb{G}_T, g, p, g^{x_1}, g^{x_2}, g^{x_3}, R\}$, where $\mathcal{R}$ is equal to $e(g,g)^{x_1x_2x_3}$ or $e(g,g)^z$ with an equal probability, $\mathcal{A}$ judges which one the $\mathcal{R}$ is equal to. If $\mathcal{R} = e(g,g)^{x_1x_2x_3}$, $\mathcal{A}$ outputs 1; otherwise, it outputs 0. The advantage of $\mathcal{A}$ in solving the DBDH problem can be expressed as follows.

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{DBDH}} = \Big| \Pr[\mathcal{A}(g, g^{x_1}, g^{x_2}, g^{x_3}, e(g,g)^{x_1x_2x_3}) = 1] - \Pr[\mathcal{A}(g, g^{x_1}, g^{x_2}, g^{x_3}, e(g,g)^z) = 1] \Big| = \varepsilon$$

If no adversary $\mathcal{A}$ can achieve an advantage of at least ε in breaking the DBDH problem, it can be concluded that the validity of the DBDH assumption holds.

Lagrange interpolation

Given a set of points $\{(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)\}$ and $x_\alpha \neq x_\beta$, where $\alpha \neq \beta$ and $\alpha, \beta \in \{0, 1, \cdots, n\}$. These nodes uniquely decide an n-degree polynomial through the application of the Lagrange Interpolation algorithm. The polynomial can be explicitly represented in the following form.

$$P_n(x) = \sum_{\alpha=0}^{n} \Delta_\alpha(x) y_\alpha = \sum_{\alpha=0}^{n} \left(\prod_{\beta=0, \beta\neq\alpha}^{n} \left(\frac{x - x_\beta}{x_\alpha - x_\beta}\right) y_\alpha \right).$$

Let $\Delta_\alpha(x) = \prod_{\beta=0,\beta\neq\alpha}^{n} \left(\frac{x-x_\beta}{x_\alpha - x_\beta}\right)$ be the Lagrange coefficient for insertion point α in the set $\{0, 1, \cdots, n\}$.

Merkle hash tree

The Merkle Hash Tree (MHT) is a commonly utilized method for authentication purposes. It involves the creation of a binary tree structure, where the leaf nodes hold hash values representing genuine data, and each non-leaf node stores a hash value derived from its child nodes. The example of MHT is illustrated in Fig. 3, where $Data_1$, $Data_2$, $Data_3$, $Data_4$ are data blocks and $h_3 = h(Data_1)$, $h_4 = h(Data_2)$, $h_5 = h(Data_3)$, $h_6 = h(Data_4)$, where h represents a hash function. The value of each parent node is obtained by combining the values of its two child nodes. For example, $h_R = h(h_1||h_2)$, $h_1 = h(h_3||h_4)$, and $h_2 = h(h_5||h_6)$. Auxiliary authentication information Aai_{Data_i} consists of sibling nodes along the path from the i-th leaf node to the root node, and is used for root node calculation. For example, $Aai_{Data_1} = h_4||h_2$. When updating the Merkle Hash Tree, $Data_i$ is first replaced, and then the hash values of all nodes from this node to the root node are recalculated. For example, if $Data_1$ is updated to $Data_1^*$, the Merkle Hash Tree is updated as follows: first, replace $Data_1$ with $Data_1^*$ and then compute $h_3 = h(Data_1^*)$, $h_1 = h(h_3||h_4)$, $h_R = h(h_1||h_2)$.

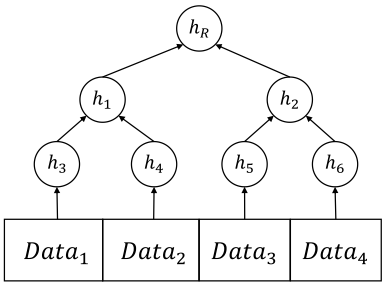


Fig. 3 Merkle hash tree

Formal definition

System model

The main components of the system model include EHR Owner, Verification Server, Cloud Server, Trusted Authority, and EHR User, as depicted in Fig. 4.

e-Health Record Owner (EHR Owner): The user desires to store and share information on cloud servers. The EHR Owner uses a symmetric encryption algorithm to encrypt the files and establishes a weighted multi-user collaborative access tree by weighting attributes. Then the EHR Owner uses the access tree to encrypt the symmetric encryption key. Finally, the EHR Owner sends the encrypted key and symmetric encryption ciphertext as its ciphertext to the verification server (VS). When the system performs user revocation, the EHR Owner updates its ciphertext and sends the hash value of the updated ciphertext to the verification server.

Verification Server (VS): This server is used to receive and forward the ciphertext sent by the EHR Owner and to assign a unique identifier *fname* to the ciphertext. At the same time, VS keeps a Merkle Hash Tree that is composed of the hash value of ciphertext and its unique identifier to authenticate the data's integrity on the cloud server (CS) after revocation. When the system performs user revocation, VS updates Merkle Hash Tree using the hash value of the updated ciphertext sent by the EHR

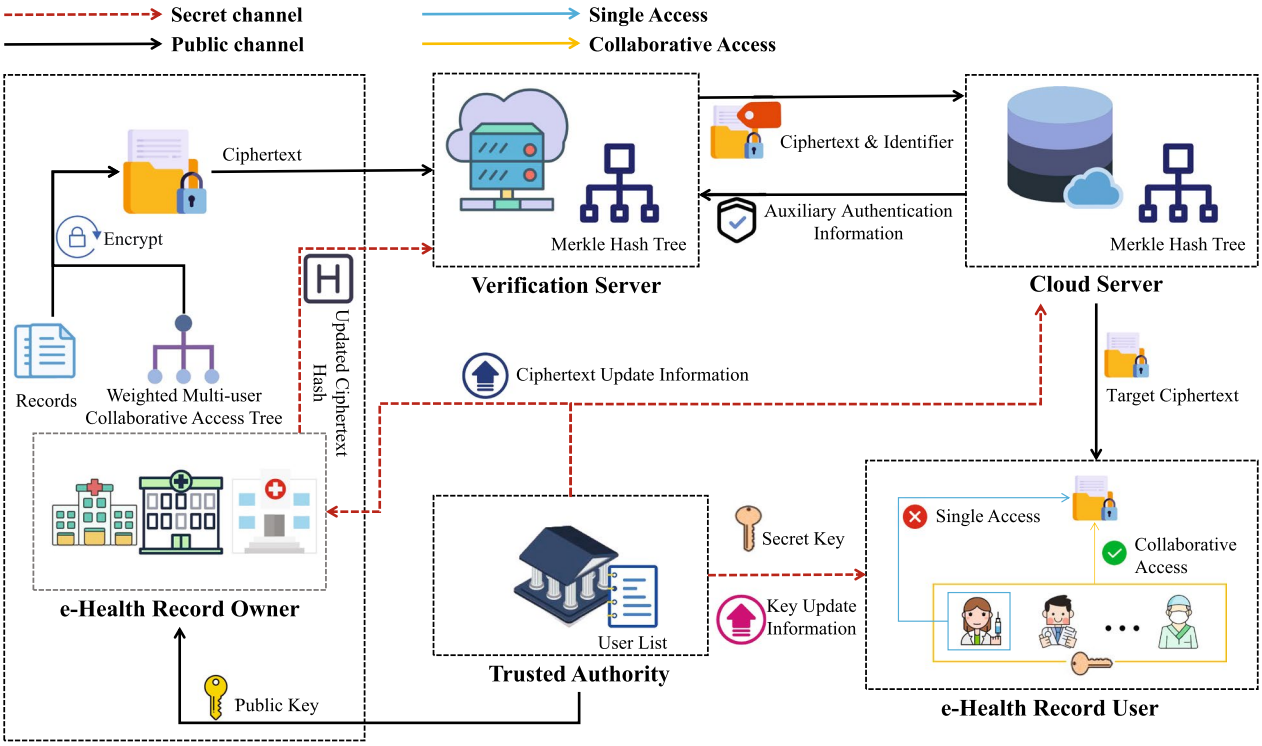


Fig. 4 System model of RVWABE-CA

Owner. VS can verify the integrity of data on CS using the auxiliary authentication information Aai_{fname} of ciphertext *fname* sent by CS and the tag value of VS's MHT Tag_R.

Cloud Server (CS): The platform provides users with the ability to store and share data. The EHR Owner can upload its ciphertext to CS. The EHR user can download its target ciphertext stored on cloud servers. The CS maintains a Merkle Hash Tree, just like the one in VS. When the system performs user revocation, CS updates the stored ciphertext and the Merkle Hash Tree. When the VS wants to authenticate the integrity of a ciphertext, CS generates the auxiliary authentication information Aai_{fname} of that ciphertext and sends it to VS.

Trusted Authority (TA): The authority generates the public key, master secret key, and secret keys for e-Health record users. It also generates version information for encryption and key generation, as well as update information for key and ciphertext updates. TA maintains a user list `LIST` composed of unique user identifiers for user revocation, and it is generally regarded as completely credible.

e-Health Record User (EHR User): The user seeks access to shared files provided by the EHR Owner. It can get the secret key generated from the TA according to its attributes. Then the EHR User employs its secret key to decrypt the target ciphertext received from CS. When they cannot satisfy the access policy on their own, they can cooperate with the users for collaborative access. When the system performs user revocation, the user who has not been revoked needs to update its secret key using the key update information sent by TA.

Definition of RVWABE-CA scheme

Definition 1 The RVWABE-CA scheme comprises eight algorithms: **Setup**, **Encrypt**, **KeyGen**, **Revoke**, **CTUpdate**, **KeyUpdate**, **Decrypt**, and **Verify**. The following definitions outline these algorithms.

(1) Setup $(1^\lambda) \rightarrow \{MSK, PK, S_{ver}, S_{up}\}$. The algorithm is run by TA. Given the input of a security parameter 1^λ, it generates the public key PK, the master secret key MSK, as well as data encryption version S_{ver} and key update version S_{up}.

(2) Encrypt $(PK, M, \mathcal{T}, S_{ver}) \rightarrow CT$. The algorithm is run by the EHR Owner. Taking the public key PK, a message M, the access policy $\mathcal{T}$, and encryption version S_{ver} as input, it encrypts M under $\mathcal{T}$ with S_{ver} and returns the ciphertext CT.

(3) KeyGen $(S, PK, MSK, S_{up}) \rightarrow SK$. This algorithm is executed by TA. Taking an EHR User's attributes S, the public key PK, the master secret key MSK, and key update version S_{up} as input, it outputs the secret key SK.

(4) Revoke $(PK, MSK, S_{ver}, S_{up}) \rightarrow (S_{ver_{new}}, S_{up_{new}})$. The algorithm is run by TA. Taking PK, MSK, S_{ver}, and S_{up} as input, it returns update information tuple $\{S_{ver_{new}}, S_{up_{new}}\}$.

(5) CTUpdate $(PK, CT, S_{ver_{new}}) \rightarrow CT_{up}$. This algorithm is executed by CS. Taking PK, CT, and $S_{ver_{new}}$ as input, it returns updated ciphertext CT_{up}.

(6) KeyUpdate $(SK, S_{up_{new}}) \rightarrow SK_{up}$. This algorithm is executed by all non-revoked users. Taking the SK and $S_{up_{new}}$ as input, it returns an updated key SK_{up}.

(7) Decrypt $(CT_{up}, PK, \{SK_{up}\}) \rightarrow M$. This algorithm is executed by the EHR User. It takes the ciphertext CT_{up}, the public key PK, and a set of secret keys $\{SK_{up}\}$ as input. The CT_{up} contains an access policy $\mathcal{T}$ and the $\{SK_{up}\}$ consists of the secret key SK_{up} for each user participating in collaborative decryption. Since each user's secret key SK_{up} is generated by a set of attributes, $\{SK_{up}\}$ can be viewed as generated by a set of attributes S, which contains the attributes of all the users involved in collaborative decryption. When the attribute set S of $\{SK_{up}\}$ meets the access policy $\mathcal{T}$ for CT_{up}, it is capable of successfully decrypting the given ciphertext and providing the message M as output.

(8) Verify $(Aai_{fname}, Tag_R) \rightarrow True/False$. The algorithm is run by VS. It takes the auxiliary authentication information Aai_{fname} of file $fname$ and the tag Tag_R as input, where $fname$ is the unique tag that VS assigns to the file, Aai_{fname} is generated by CS according to its MHT when verification is required, and Tag_R is the tag value of the root node of VS's MHT. Then it verifies whether the auxiliary authentication information is generated from the correct ciphertext.

Security model

The security of our scheme is mainly reflected in data security, revocation security and verification security.

Data and revocation security

The definitions of data security and revocation security, along with their corresponding security model, are presented as follows.

Data security: Data is confidential to all unauthorized authorities, collaborative users, and cloud servers.

Revocation security: After the revocation operation is performed, the revoked user cannot obtain the plaintext. In other words, the scheme still has date security after the key and ciphertext are updated.

Since the update operation in revocation is based on the generated secret key and ciphertext, the revocation security model can be integrated into the data security model. The security model is outlined as follows.

The data and revocation security model of the RVWABE-CA scheme can be described through an IND-CPA game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$. The adversary $\mathcal{A}$ queries the secret key and tries to tell whether the challenge ciphertext is encrypted by the plaintext m_0 or m_1. The challenger $\mathcal{C}$ generates the corresponding key for adversary $\mathcal{A}$ and generates challenge ciphertext by randomly selecting m_0 or m_1 according to the RVWABE-CA algorithm. The interactive processes between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ are as follows.

Init: $\mathcal{A}$ chooses an access policy $\mathcal{T}$ to contest and submits it to challenger $\mathcal{C}$.

Setup: By executing Setup algorithm, $\mathcal{C}$ gets the public key PK, the master secret key MSK, encryption version S_{ver} and key update version S_{up}. $\mathcal{C}$ executes Revoke algorithm to generate update information $S_{ver_{new}}$ and $S_{up_{new}}$, and sends key update information $S_{up_{new}}$ to the adversary $\mathcal{A}$.

Query phase I: $\mathcal{A}$ repeatedly queries the private key for a series of attribute sets $S_1, S_2, \cdots, S_n$. For each query, $\mathcal{C}$ executes KeyGen algorithm to generate the secret key SK for attribute set S_i, and sends SK to $\mathcal{A}$. $\mathcal{A}$ executes KeyUpdate algorithm to update secret key SK to SK_{up}. Each query of the adversary can be considered as a user participating in collaborative decryption, so that the adversary is equivalent to querying the attribute set $Atts = \{S_1, S_2, \cdots, S_n\}$, where $Atts$ should not satisfy the access conditions of T.

Challenge: The adversary $\mathcal{A}$ selects two messages m_0 and m_1 of equal length and sends them to $\mathcal{C}$. $\mathcal{C}$ randomly chooses a message m_ρ where $\rho \in \{0, 1\}$, and executes Encrypt algorithm to encrypt m_ρ under access policy

$\mathcal{T}$ to get the ciphertext CT. Then $\mathcal{C}$ executes CTUpdate algorithm to update CT to CT_{up} and returns it to $\mathcal{A}$.

Query phase II: $\mathcal{A}$ repeatedly makes the queries as the same as the *Query phase* I.

Guess: The adversary $\mathcal{A}$ outputs a guess $\rho' \in \{0, 1\}$ of ρ. If $\rho' = \rho$, $\mathcal{A}$ achieves success in the game. The description of the advantage that $\mathcal{A}$ possesses in achieving victory in the security game can be articulated as follows.

$$\mathrm{Adv}^{\mathrm{IND-CPA}}_{\mathrm{RVWABE-CA},\mathcal{A}}(1^{\lambda}) = \left| \Pr[\rho' = \rho] - \frac{1}{2} \right|.$$

Definition 2 The RVWABE-CA scheme achieves the security of IND-CPA if no polynomial-time adversary can win the IND-CPA game with a non-negligible advantage.

Verification security

Verification security: The data is not disclosed to unauthorized organizations, servers, or users during data verification.

The verification of our scheme requires the interaction of multiple entities in the system. As shown in Fig. 4, the verification process involves the EHR Owner, Cloud Server, Verification Server and Trusted Authority, where Trusted Authority is completely trusted. We assume that the EHR Owner has no incentive to disclose its ciphertext information. If the verification process ensures that Cloud Server and Verification Server cannot obtain additional information, then it does not disclose data information to unauthorized organizations, servers, or users, thus achieving verification security.

The proposed scheme

We introduce the weighted multi-user collaborative access tree and then provide a detailed RVWABE-CA construction.

Weighted multi-user collaborative access tree

Let $\mathcal{T}$ be a weighted multi-user collaborative access tree. To simplify the explanation of this access control framework, we have established the subsequent functions and terminologies.

- (x, y) denotes a node in $\mathcal{T}$. If (x, y) represents a leaf node, it signifies an attribute. The symbol x represents the node situated at the level x of $\mathcal{T}$ (from top to bottom, starting from 1), and y represents the node located at the y-th position on a certain level of $\mathcal{T}$ (from left to right, starting from 1). For example, (1, 1) denotes the root node and (2, 2) denotes the node with attribute A in Fig. 5.

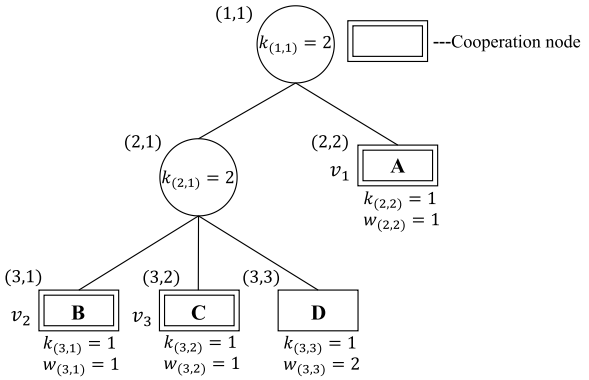


Fig. 5 Weighted multi-user collaborative access tree

- $w_{(x,y)}$ denotes the weight of node (x, y). For example, in Fig. 5, $w_{(3,3)} = 2$ means that the weight of (3, 3) in this access tree is 2. The weight is determined by the user's access policy, which is tailored to their specific needs. Note that the weight of non-leaf nodes can only be 1.
- v_i denotes a cooperation node for multi-user cooperative access. Users can only have collaborative access through these cooperation nodes.
- $child_{(x,y)}$ denotes the set of child nodes of (x, y). For example, $child_{(2,1)} = \{(3, 1), (3, 2), (3, 3)\}$ in Fig. 5.
- $parent(x, y)$ is used to obtain the parent node of (x, y). For example, in Fig. 5, $parent(3, 1) = parent(B) = (2, 1)$.
- $k_{(x,y)}$ denotes the threshold value of node (x, y), where $1 \leq k_{(x,y)} \leq \sum_{i \in child_{(x,y)}} w_{(x,y)}$. When the value of $k_{(x,y)}$ is equal to 1, the threshold gate functions as an "OR" gate. Conversely, when $k_{(x,y)}$ equals $\sum_{i \in child_{(x,y)}} w_{(x,y)}$, it operates as an "AND" gate. Specifically, when (x, y) is a leaf node, the value of $k_{(x,y)}$ is equal to 1.
- $index(x, y)_i$ returns the i-th unique identifier of the node (x, y) in $\mathcal{T}$. i ranges from 1 to $w_{(x,y)}$.
- $att(x, y)$ returns an attribute associated with the leaf node (x, y) in $\mathcal{T}$. For example $att(2, 2) = A$ in Fig. 5.
- $share(x, y)_i$ represents the i-th secret value of attribute (x, y), where the value of i ranges from 1 to $w_{(x,y)}$. For example, $share(D)_i$ has two values $share(D)_1$ and $share(D)_2$, while $share(2, 1)_i$ has one value $share(2, 1)_1$ in Fig. 5.

A RVWABE-CA construction

(1) Setup $(1^{\lambda}) \rightarrow \{MSK, PK\, S_{ver}, S_{up}\}$. This algorithm first generates $\mathbb{GP} = \{e, \mathbb{G}, \mathbb{G}_T, g, p\}$ for bilinear map based on the security parameter λ. Then it chooses

random elements $\alpha, \beta, k \in \mathbb{Z}_p$ and three hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \{0,1\}^* \to \mathbb{Z}_p$, $H_3 : \mathbb{G}_T \to \{0,1\}^*$ and computes $f = g^{\beta}$, $S_{ver} = g^{-k}$, $S_{up} = g^{\frac{k}{\beta}}$, and $e(g,g)^{\alpha}$. Finally, it generates the public key PK, the master secret key MSK, the initial encryption version S_{ver}, and the initial key update version S_{up} as outputs.

$$\begin{aligned} PK &= \left\{e, \mathbb{G}, \mathbb{G}_T, g, p, f, e(g,g)^{\alpha}, H_1, H_2, H_3\right\}, \\ MSK &= \left\{\beta, \alpha, k\right\}, \\ S_{ver} &= g^{-k}, \\ S_{up} &= g^{\frac{k}{\beta}}. \end{aligned}$$

When performing initialization, TA also initializes a user list `LIST` without any user information.

(2) Encrypt $(PK, M, \mathcal{T}, S_{ver}) \to CT$. It chooses a random element $s \in \mathbb{Z}_p$ as the secret value. For each node (x, y) in $\mathcal{T}$, it constructs a polynomial $q_{(x,y)}$. The degree of $q_{(x,y)}$ is $k_{(x,y)} - 1$. The algorithm sets $q_{(1,1)}(0) = s$ and constructs the weighted multi-user collaborative access tree from the root node (1, 1) according to the following rules.

- For each node (x, y) in $\mathcal{T}$, the algorithm assigns $w_{(x,y)}$ identifiers to it, calculates $\{share(x,y)_i = q_{parent(x,y)}(index(x,y)_i), (i = 1, \cdots, w_{(x,y)})\}$, and stores the result.
- Then, it sets $q_{(x,y)}(0) = share(x,y)_1$ for all (x, y).

The weighted multi-user collaborative access tree can be established from top to bottom through the above steps. Then this algorithm uses this access tree to encrypt the message M by performing the following operations.

- The encryption algorithm calculates C and $\widetilde{C}$ as follows.

$$\begin{aligned} C &= M \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{-ks}, \\ \widetilde{C} &= g^{s}. \end{aligned}$$

- The encryption algorithm computes the translation value C_{v_i} for each cooperation node v_i.

$$C_{v_i} = g^{share(v_i)_n}, (n \in \{1, \cdots, w_{v_i}\}).$$

The translation value C_{v_i} is a set, where $C_{v_i}(n)$ is the n-th element in the set.

- For each leaf node (x, y) in the access tree $\mathcal{T}$, the encrypt algorithm calculates $C_{(x,y)_i}$ and $C'_{(x,y)_i}$ as follows.

$$\begin{aligned} C_{(x,y)_i} &= f^{share(x,y)_i}, \\ C'_{(x,y)_i} &= H_1(att(x,y))^{share(x,y)_i}, (i = 1, \cdots, w_{(x,y)}). \end{aligned}$$

It should be noted that the attribute of each leaf node is unchanged, so the hash operation of each leaf node only needs to be performed once.

Let L denote the collection of leaf nodes in the access tree $\mathcal{T}$, and V represent the set of all cooperation nodes. It organizes the ciphertext CT as follows.

$$\begin{aligned} CT = \Big\{ &\mathcal{T}, C, \widetilde{C}, \left\{C_{v_i}\right\}_{v_i \in V}, \\ &\left\{C_{(x,y)_i}, {C_{(x,y)_i}}'\right\}_{(\forall (x,y) \in L, i = 1, \cdots, w_{(x,y)})} \Big\}. \end{aligned}$$

After executing this algorithm, EHR Owner calculates $H_3(C)$ as the tag of this message and sends the CT with $H_3(C)$ to VS. When VS receives these data, it assigns a unique identifier *fname* to this ciphertext and sends it to CS along with the CT. Then VS and CS construct Merkle Hash Tree using $Tag_{fname} = H_2(fname||H_3(C))$ as data respectively.

(3) KeyGen $(S, PK, MSK, S_{up}) \to SK$. This algorithm procedures the user's secret key as follows. Firstly, it selects a random element $r \in \mathbb{Z}_p$, and calculates D. Then for each attribute $j \in S$, it selects a random element $r_j \in \mathbb{Z}_p$, and computes D_j and ${D_j}'$.

$$\begin{aligned} D &= g^{\alpha + r}, \\ D_j &= g^{\frac{r+k}{\beta}} \cdot H_1(j)^{\frac{r_j}{\beta}}, (j \in S), \\ {D_j}' &= g^{r_j}, (j \in S). \end{aligned}$$

Finally, the algorithm outputs SK.

$$SK = \left\{D, \left\{D_j, {D_j}'\right\}_{\forall j \in S}\right\}.$$

When TA finishes executing this algorithm, TA assigns a unique identification to each newly registered user and appends it to the list of users `LIST`.

(4) Revoke $(PK, MSK, S_{ver}, S_{up})$ It selects a new value of k marked as $k_{new} \in \mathbb{Z}_p$. Subsequently, it computes $S_{ver_{new}}$ and $S_{up_{new}}$ using the following procedure.

$$\begin{aligned} S_{ver_{new}} &= g^{k - k_{new}}, \\ S_{up_{new}} &= g^{\frac{-k + k_{new}}{\beta}}. \end{aligned}$$

It returns the update information tuple $\{S_{ver_{new}}, S_{up_{new}}\}$ and sets $MSK = \{\beta, \alpha, k_{new}\}$

When TA finishes executing this algorithm, TA removes the revoked user from `LIST`, and sends $S_{up_{new}}$ to all non-revoked users in `LIST`, while transmitting $S_{ver_{new}}$ to CS and EHR Owner.

(5) CTUpdate $(PK, CT, S_{ver_{new}})$ It takes the new version key $S_{ver_{new}}$ to update the ciphertext CT as

$$\begin{aligned} C_{up} &= C \cdot e(\widetilde{C}, S_{ver_{new}}) \\ &= M \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{-ks} \cdot e(g,g)^{ks+k_{new}s} \\ &= M \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{-k_{new}s}. \end{aligned}$$

It reorganizes the updated ciphertext CT_{up} as follows.

$$\begin{aligned} CT_{up} = \Big\{ &\mathcal{T}, C_{up}, \widetilde{C}, \{C_{v_i}\}_{v_i \in V}, \\ &\{C_{(x,y)_i}, C_{(x,y)_i}{}'\}_{(\forall (x,y) \in L, i=1,\cdots,w_{(x,y)})} \Big\}. \end{aligned}$$

After executing this algorithm, CS computes $Tag_{fname} = H_2(fname||H_3(C_{up}))$ for file *fname* and uses it as data to update the Merkle Hash Tree. When EHR Owner finishes executing this algorithm, EHR Owner computes $H_3(C_{up})$ and sends it to VS, which computes $Tag_{fname} = H_2(fname||H_3(C_{up}))$ and updates the Merkle Hash Tree.

(6) KeyUpdate $(SK, S_{up_{new}})$ It takes the new version key $S_{up_{new}}$ to update the user's *SK*. For each D_j in *SK*, it calculates

$$\begin{aligned} D_{j_{up}} &= D_j \cdot S_{up_{new}} \\ &= g^{\frac{r+k}{\beta}} \cdot H_1(j)^{\frac{r_j}{\beta}} \cdot g^{\frac{-k+k_{new}}{\beta}} \\ &= g^{\frac{r+k_{new}}{\beta}} \cdot H_1(j)^{\frac{r_j}{\beta}}. \end{aligned}$$

Finally, it reorganizes the updated key as

$$SK_{up} = \Big\{ D, \{D_{j_{up}}, D_j{}'\}_{\forall j \in S} \Big\}.$$

The Decrypt algorithm is subsequently presented. This algorithm consists of two parts: Step 1) and Step 2). In Step 1), a single key is utilized for calculation, which can be interpreted as each user calculating their own secret value using their secret key *SK*. The calculations of step 1) are performed on all the keys in the key set $\{SK\}$, resulting in a set of values. Subsequently, in Step 2), this set of values is employed to compute the plaintext, representing collaborative decryption among users corresponding to the key set $\{SK\}$.

(7) Decrypt $(CT_{up}, PK, \{SK_{up}\}) \rightarrow M$. This algorithm performs the following works.

1) The algorithm initially establishes a recursive function $DecryNode(CT, SK_o, (x,y))$, where $SK_o \in \{SK_{up}\}$. For a leaf node (x, y), if $att(x,y) \notin S$, $DecryNode(CT_{up}, SK_o, (x,y)) = \perp$; otherwise $DecryNode(CT_{up}, SK_o, (x,y))$ calculates as follows.

$$\begin{aligned} &DecryNode(CT_{up}, SK_o, (x,y)) \\ &= \frac{e(C_{(x,y)_i}, D_{j_{up}})}{e(C_{(x,y)_i}{}', D_j{}')} \\ &= \frac{e(g^{\beta \cdot share(x,y)_i}, g^{\frac{r+k_{new}}{\beta}} \cdot H_1(j)^{\frac{r_j}{\beta}})}{e(H_1(att(x,y))^{share(x,y)_i}, g^{r_j})} \\ &= e(g,g)^{(r+k_{new}) \cdot share(x,y)_i}. \end{aligned}$$

The algorithm recursively executes $DecryNode(CT_{up}, SK_o, (x,y))$ if (x, y) is a non-leaf node. For node (x, y)'s child node Λ, this algorithm runs $DecryNode(CT_{up}, SK_o, \Lambda)$ and stores the output as F_Λ. Let $S(x, y)$ be a node set composed of arbitrary child node Λ of (x, y) such that $F_\Lambda \neq \perp$. If no such set exists, then $F_{(x,y)} = \perp$; otherwise it is computed as follows.

$$\begin{aligned} F_{(x,y)} &= \prod_{\Lambda \in S(x,y)} F_\Lambda{}^{\Delta_{i,S(x,y)'}(0)} \\ &= \prod_{\Lambda \in S(x,y)} (e(g,g)^{(r+k_{new}) \cdot share(\Lambda)_{w(\Lambda)}})^{\Delta_{i,S(x,y)'}(0)} \\ &= e(g,g)^{(r+k_{new}) \cdot share(x,y)_1}, \end{aligned}$$

where $\Delta_{i,S(x,y)'}$ is the Lagrange coefficient, $w(\Lambda)$ is the weight of node Λ form set $\{1, \cdots, w_\Lambda\}$, i is the identifier of node Λ from set $\{index(\Lambda)_j\}_{j \in [1,w_\Lambda]}$, and $S(x,y)'$ consists of the identifier set of each child node Λ of (x, y), i.e. $S(x,y)' = \Big\{ \{index(\Lambda)_j\}_{j \in [1,w_\Lambda]} : \Lambda \in S(x,y) \Big\}$.

For each cooperation node v_i participated in collaborative decryption, the algorithm performs $DecryNode(CT_{up}, SK_o, v_i)$ to get $F' = e(g,g)^{r \cdot share(v_i)_n}$. The algorithm then performs the following operations to obtain a secret value that can participate in collaborative decryption.

$$\begin{aligned} F_{v_i}(n) &= \frac{e(C_{v_i}(n), D)}{F'} \\ &= \frac{e(g^{share(v_i)_n}, g^{\alpha+r})}{e(g,g)^{(r+k_{new}) \cdot share(v_i)_n}} \\ &= e(g,g)^{(\alpha - k_{new}) \cdot share(v_i)_n}. \end{aligned}$$

where $n \in \{1, \cdots, w_{v_i}\}$ and $\Delta_{i,S(x,y)'}$ is the Lagrange coefficient as described in step 1).

2) After acquiring the secret value of all required cooperation nodes, it performs the following calculation to obtain the plaintext. Let U represent a set of cooperation nodes involved in collaborative decryption that satisfies their parent node *parent*'s access policy. If there is no such set, it outputs $\perp$. We have

$$\begin{aligned} F_{parent} &= \prod_{u\in U, n\in[1,w_u]} (F_u(n))^{\Delta_{i,U'}(0)} \\ &= \prod_{u\in U, n\in[1,w_u]} (e(g,g)^{(\alpha-k_{new})\cdot share(u)_n})^{\Delta_{i,U'}(0)} \\ &= e(g,g)^{(\alpha-k_{new})\cdot share(parent)_1}. \end{aligned}$$

The algorithm performs the above operations for all cooperation nodes involved in collaborative decryption and obtains a set of values $\{F_{parent}\}$. The algorithm recursively performs the above operation with F_{parent} as $F_u(n)$, that is, Lagrange Interpolation is used to iterate from the cooperative node to the root node, and gets

$$F_{(1,1)} = e(g,g)^{\alpha s} \cdot e(g,g)^{-k_{new}s}.$$

Finally, the user μ who is selected as the representative gets M by performing

$$\frac{C_{up}}{F_{(1,1)}} = \frac{M \cdot e(g,g)^{\alpha s} e(g,g)^{-k_{new}s}}{e(g,g)^{\alpha s} e(g,g)^{-k_{new}s}} = M.$$

The Verify algorithm is described next. This algorithm aims to validate whether the root node tag Tag_R^* calculated from the auxiliary authentication information Aai_{fname} aligns with the root node tag Tag_R of the verification server's Merkle hash tree, where Aai_{fname} is generated by the cloud server and Tag_R is computed by the verification server based on the provided Tag_{fname} from the EHR owner.

(8) Verify (Aai_{fname}, Tag_R) For file *fname*, this algorithm uses Tag_{fname}, provided by EHR Owner, and Aai_{fname} to calculate the hash value Tag_R^* of the root node of VS's Merkle hash tree. It returns $Tag_R^* \stackrel{?}{=} Tag_R$.

Note that since the verification of our scheme is based on the Merkle hash tree, the auxiliary authentication information Aai_{fname} as well as the root node tags Tag_R and Tag_R^* in this algorithm are generated according to the construction rules of Merkle hash tree.

Correctness

In step 2) of the decryption process, after obtaining the necessary secret values, the correctness of F_{parent} is calculated using Lagrange Interpolation as follows. It is assumed that these secret values satisfy *parent*'s access control conditions.

$$\begin{aligned} F_{parent} &= \prod_{u\in U, n\in[1,w_u]} (F_u(n))^{\Delta_{i,U'}(0)} \\ &= \prod_{u\in U, n\in[1,w_u]} (e(g,g)^{(\alpha-k_{new})\cdot share(u)_n})^{\Delta_{i,U'}(0)} \\ &= \prod_{u\in U, n\in[1,w_u]} e(g,g)^{\alpha\cdot share(u)_n\cdot\Delta_{i,U'}(0)} \\ &\quad \cdot \prod_{u\in U, n\in[1,w_u]} e(g,g)^{-k_{new}\cdot share(u)_n\cdot\Delta_{i,U'}(0)} \\ &= e(g,g)^{\alpha\cdot\sum_{u\in U, n\in[1,w_u]} (share(u)_n\cdot\Delta_{i,U'}(0))} \\ &\quad \cdot e(g,g)^{-k_{new}\cdot\sum_{u\in U, n\in[1,w_u]} (share(u)_n\cdot\Delta_{i,U'}(0))} \\ &= e(g,g)^{\alpha\cdot q_{parent}(0)} \cdot e(g,g)^{-k_{new}\cdot q_{parent}(0)} \\ &= e(g,g)^{(\alpha-k_{new})\cdot share(parent)_1}. \end{aligned}$$

q_{parent} represents the polynomial of the *parent* node when building the access tree and $\Delta_{i,U'}(0)$ is the Lagrange coefficient as described in Decrypt algorithm. According to Lagrange Interpolation, we can get $q_{parent}(0)$ from $\sum_{u\in U, n\in[1,w_u]} \left(share(u)_n \cdot \Delta_{i,U'}(0)\right)$, which is $share(parent)_1$. The above process calculates the secret value of the *parent* node for partially cooperation nodes in collaborative decryption. By performing this calculation for all cooperation nodes, we obtain a set of secret values F_{parent}. We bring F_{parent} as $F_u(n)$ into the above formula for iterative calculation and finally get

$$\begin{aligned} F_{(1,1)} &= e(g,g)^{(\alpha-k_{new})q_{(1,1)}(0)} \\ &= e(g,g)^{(\alpha-k_{new})s} \\ &= e(g,g)^{\alpha s} \cdot e(g,g)^{-k_{new}s}. \end{aligned}$$

Security proof

Data and revocation security

Theorem 1 *If there exists an adversary $\mathcal{A}$ that can achieve a non-negligible advantage ε in polynomial time in breaking the IND-CPA security of RVWABE-CA scheme, we can construct a simulator $\mathcal{C}$ to solve the DBDH problem with a non-negligible advantage of at least $\frac{\varepsilon}{2}$ in polynomial time.*

Proof: Let $\mathcal{A}$ be the adversary in IND-CPA game, $\mathcal{B}$ be a DBDH challenger, and $\mathcal{C}$ be the simulator to solve the DBDH problem as well as a challenger in

IND-CPA game. $\mathcal{B}$ generates two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with generator g, as well as a bilinear map $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Then it selects a random element $\mathcal{R} \in \mathbb{G}_T$, three random elements $x_1, x_2, x_3 \in \mathbb{Z}_p$, and sets $X_1 = g^{x_1}$, $X_2 = g^{x_2}$, $X_3 = g^{x_3}$. Y is an element in group $\mathbb{G}_T$. $\mathcal{C}$ flips a fair coin $\zeta \in \{0,1\}$. If $\zeta = 0$, $\mathcal{C}$ sets $Y = \mathcal{R}$ otherwise $Y = e(g,g)^{x_1 x_2 x_3}$. $\mathcal{B}$ sends the DBDH tuple $\{e, \mathbb{G}, \mathbb{G}_T, g, p, X_1, X_2, X_3, Y\}$ to $\mathcal{C}$. $\mathcal{C}$ outputs its guess ζ' of ζ.

Init: $\mathcal{A}$ selects an access policy $\mathcal{T}$ to challenge and delivers it to $\mathcal{C}$.

Setup: $\mathcal{C}$ randomly selects $\alpha^* \in \mathbb{Z}_p$, $k^* \in \mathbb{Z}_p$, and three hash functions $H_1: \{0,1\}^* \to \mathbb{G}$, $H_2: \{0,1\}^* \to \mathbb{Z}_p$, $H_3: \mathbb{G}_T \to \{0,1\}^*$. Then it implicitly sets $\beta = x_2$, $\alpha = x_2(\alpha^* + x_1)$, $k = x_2 k^*$, and computes $S_{ver} = g^{-k} = g^{-x_2 k^*} = X_2^{-k^*}$, $S_{up} = g^{\frac{k}{\beta}} = g^{k^*}$, $f = g^{\beta} = g^{x_2} = X_2$, $e(g,g)^{\alpha} = e(g,g)^{x_2(\alpha^* + x_1)} = e(g,g)^{\alpha^* x_2} e(g,g)^{x_2 x_1} = e(g^{\alpha^*}, X_2) e(X_1, X_2)$. $\mathcal{C}$ randomly chooses $k^*_{new} \in \mathbb{Z}_p$, and implicitly sets $k_{new} = x_2 k^*_{new}$. $\mathcal{C}$ computes $S_{ver_{new}} = g^{k - k_{new}} = g^{x_2 k^* - x_2 k^*_{new}} = X_2^{k^* - k^*_{new}}$ and $S_{up_{new}} = g^{\frac{-k + k_{new}}{\beta}} = g^{\frac{-x_2 k^* + x_2 k^*_{new}}{x_2}} = g^{-k^* + k^*_{new}}$. $\mathcal{C}$ returns $PK = \{e, \mathbb{G}, \mathbb{G}_T, g, p, f, e(g,g)^{\alpha}, H_1, H_2, H_3\}$ and $S_{up_{new}}$ to $\mathcal{A}$.

Query phase I: The private key is queried by $\mathcal{A}$ for a series of attribute sets $S_1, S_2, \cdots, S_n$. $\mathcal{C}$ randomly chooses $r^* \in \mathbb{Z}_p$ and implicitly sets $r = x_2(r^* - x_1)$. $\mathcal{C}$ calculates $D = g^{\alpha + r} = g^{x_2(\alpha^* + x_1) + x_2(r^* - x_1)} = g^{x_2 \alpha^* + x_2 r^*} = X_2^{\alpha^* + r^*}$. Then, for every attribute s in S_i, $\mathcal{C}$ chooses $r_j^* \in Z_p$, implicitly setting $r_j = x_2 r_j^*$, and computes $D_j = g^{\frac{r+k}{\beta}} \cdot H(j)^{\frac{r_j}{\beta}} = g^{\frac{x_2(r^* - x_1 + k^*)}{x_2}} \cdot H(j)^{\frac{x_2 r_j^*}{x_2}} = g^{r^* + k^* - x_1} \cdot H(j)^{r_j^*} = \frac{g^{r^* + k^*} \cdot H(j)^{r_j^*}}{X_1}$ and $D_j' = g^{r_j} = g^{x_2 r_j^*} = X_2^{r_j^*}$. For each query of $\mathcal{A}$, $\mathcal{C}$ generates the corresponding $SK = \left\{D, \{D_j, D_j'\}_{\forall j \in S_i}\right\}$ and returns it to $\mathcal{A}$. $\mathcal{A}$ computes $D_{j_{up}} = D_j \cdot S_{up_{new}} = \frac{g^{r^* + k^*} \cdot H(j)^{r_j^*}}{X_1} \cdot g^{-k^* + k^*_{new}} = \frac{g^{r^* + k^*_{new}} \cdot H(j)^{r_j^*}}{X_1}$ and gets $SK_{up} = \left\{D, \{D_{j_{up}}, D_j'\}_{\forall j \in S_i}\right\}$. Each query of the adversary can be regarded as a new user participating in collaborative decryption so that the adversary is equivalent to querying the collaborative attribute set $Atts = \{S_1, S_2, \cdots, S_n\}$, where $Atts$ should not satisfy the access conditions of T.

Challenge: The adversary $\mathcal{A}$ selects two messages m_0 and m_1 of equal length, then sends them to $\mathcal{C}$. $\mathcal{C}$ randomly chooses a message m_ρ, where $\rho \in \{0,1\}$, to encrypt m_ρ under $\mathcal{T}$. $\mathcal{C}$ implicitly sets $s = x_3$ and computes $C = m_\rho \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{-ks} = m_\rho \cdot e(g,g)^{x_2(\alpha^* + x_1)x_3}$. $e(g,g)^{-x_2 k^* x_3} = m_\rho \cdot e(g,g)^{(\alpha^* - k*)x_2 x_3} e(g,g)^{x_1 x_2 x_3} = m_\rho \cdot e(X_2, X_3)^{\alpha^* - k^*} \cdot Y$, $\widetilde{C} = g^s = g^{x_3} = X_3$. For each node, $\mathcal{C}$ constructs a polynomial of order d as: $a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + s$, where $a_d, a_{d-1}, \cdots, a_1$ are randomly chosen by $\mathcal{C}$. For $x = i$, $\mathcal{C}$ computes $g^{a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + s} = (g^{a_d})^{i^n} \cdots (g^{a_1})^i g^s = (g^{a_d})^{i^n} \cdots (g^{a_1})^i X_3$. Then $\mathcal{C}$ can construct the weighted multi-user collaborative access tree for access policy $\mathcal{T}$ by using Lagrange Interpolation. While for every cooperation node v_i, $\mathcal{C}$ computes $C_{v_i} = \{g^{share(v_i)_1}, \cdots, g^{share(v_i)_{w_{v_i}}}\}$, where $\{g^{share(v_i)_1}, \cdots, g^{share(v_i)_{w_{v_i}}}\}$ can be calculated during the construction of the weighted multi-user collaborative access tree. Finally, $\mathcal{C}$ computes $C_{up} = C \cdot e(\widetilde{C}, S_{ver_{new}}) = m_\rho \cdot e(X_2, X_3)^{\alpha^* - k^*} \cdot Y \cdot e(X_3, X_2)^{k^* - k^*_{new}} = m_\rho \cdot e(X_2, X_3)^{\alpha^* - k^*_{new}} \cdot Y$ and returns $CT = \left\{C_{up}, \widetilde{C}, \{C_{v_i}\}_{v_i \in V}\right\}$ to $\mathcal{A}$.

Query phase II: $\mathcal{A}$ performs the same queries as in *Query phase* I repeatedly.

Guess: The adversary $\mathcal{A}$ presents its estimate ρ' for ρ. If $\rho' = \rho$, $\mathcal{C}$ outputs $\zeta' = 1$ to indicate $Y = e(g,g)^{x_1 x_2 x_3}$, otherwise $\mathcal{C}$ outputs $\zeta' = 0$. If $\zeta = 0$, $\mathcal{A}$ gets the ciphertext $m_\rho \cdot e(X_2, X_3)^{\alpha^*} \cdot \mathcal{R}$, which is a random value that completely hides the information about m_ρ, thus $\mathcal{A}$ has no advantage to break the IND-CPA security of RVWABE-CA. We have $\Pr[\rho' = \rho | \zeta = 0] = \frac{1}{2}$. Similarly, if $\zeta = 1$, $\mathcal{A}$ gets a right ciphertext $m_\rho \cdot e(X_2, X_3)^{\alpha^*} \cdot e(g,g)^{x_1 x_2 x_3}$, thus it has a non-negligible advantage ε in breaking RVWABE-CA's IND-CPA security. Then, we have $\Pr[\rho' = \rho | \zeta = 1] = \frac{1}{2} + \varepsilon$. The probability that $\mathcal{C}$ successfully guesses $\zeta' = \zeta$ is

$$\begin{aligned}\Pr[\zeta' = \zeta] &= \Pr[\zeta = 0] \Pr[\zeta' = \zeta | \zeta = 0] \\ &\quad + \Pr[\zeta = 1] \Pr[\zeta' = \zeta | \zeta = 1] - \frac{1}{2} \\ &= \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times (\frac{1}{2} + \varepsilon) - \frac{1}{2} \\ &= \frac{\varepsilon}{2}.\end{aligned}$$

This is also the advantage of $\mathcal{C}$ in solving DBDH problems. Throughout the IND-CPA game, $\mathcal{C}$ needs to generate *PK*, *SK* and *CT*. Since all the elements in *PK*, *SK*, and *CT* either are known to $\mathcal{C}$ or can be calculated by $\mathcal{C}$, the IND-CPA game can be simulated successfully. Therefore, Theorem 1 is proven.

In the IND-CPA game that proves Theorem 1, the secret key is updated in *Query phase* I and the ciphertext is updated in the *Challenge* phase, which means that the above IND-CPA game includes revocation operation, thereby leading to Corollary 1.

Corollary 1 *If Theorem* 1 *holds, then the RVWABE-CA scheme has revocation security.*

Table 2 Comparisons of property with related works

Schemes	Access control structure	Revocation level	Security	Collaborative access	Verifiability
RS-HABE	LSSS	User	Selective	×	×
RS-CPABE-ASP	ASP	User	Adaptive	×	×
RABE-DI	LSSS	User & attribute	Selective	×	✓
CC-ABE	Access tree	×	Selective	✓	×
Ours	Access tree	User	Selective	✓	✓

Table 3 Comparisons of Computational Overhead

Schemes	Encryption cost	Decryption cost	Key update cost	Ciphertext update cost	Collaborative decryption cost
RS-HABE	$(2+4n_a+n_p)$E	$(2+3n_c)$P+$2n_c$E	$(4+n_u)$E	$(2+4n_a+n_p)$E	N/A
RS-CPABE-ASP	$(4+5n_a+2n_p)$E	$(3+4n_c)$P+$3n_c$E	5E	P+$(1+n_a)$E	N/A
RABE-DI	$(6+6n_a)$E	$(2+4n_c)$P+$2n_c$E	N/A	$(2+3n_a)$E	N/A
CC-ABE	$(3+4n_a+2n_{cop})$E	$(2+4n_c)$P+$(n_c+\lfloor \log n_c \rfloor)$E	N/A	N/A	$(2+4n_c+n_{cop})$P+$(n_c+\lfloor \log n_c \rfloor)$E
Ours	P+$(1+2n_a+3n_{cop})$E	$(1+2n_c)$P+$(n_c+\lfloor \log n_c \rfloor)$E	n_uE	P+2E	$(1+2n_c+n_{cop})$P+$(n_c+\lfloor \log n_c \rfloor)$E

[1] The absence of the algorithm in the related scheme is denoted by N/A

Verification security

It is worth noting that the EHR Owner has no incentive to expose its data when executing the Verify algorithm. Instead, the EHR Owner utilizes the hash function H_3 to process the ciphertext and shares $H_3(M \cdot e(g,g)^{(\alpha-k_{new})s})$ to the Verification Server (VS), where $M \cdot e(g,g)^{(\alpha-k_{new})s}$ is a part of the updated ciphertext. To obtain this information, VS needs to crack the hash function H_3, which means that the security of this information is equivalent to the security of the H_3 hash function used in the scheme. Then it can be concluded that if the H_3 hash function is security, VS cannot obtain the updated ciphertext information. In addition, it can be seen from the system model that VS can get the original ciphertext, but according to Theorem 1, this ciphertext is IND-CPA secure. Consequently, VS can only know that the hash value of the original ciphertext is different from that of the updated ciphertext (which is natural), without receiving any additional information.

On the other hand, before executing the Verify algorithm, the Cloud Server (CS) executes CTUpdate to get the updated ciphertext, which is part of the revocation process. According to Theorem 1 and Corollary 1, the ciphertext on the cloud service is IND-CPA secure. At the time of verification, CS sends VS the auxiliary authentication information generated by the Merkle hash tree. According to Merkle (1980), the auxiliary authentication information of the Merkle hash tree does not reveal the ciphertext to VS.

Table 4 Comparisons of storage overhead

Schemes	Ciphertext size	Key size	Update key size
RS-HABE	$\lvert\mathbb{G}_T\rvert$+$(1+3n_a+\frac{1}{2}n_p)\lvert\mathbb{G}\rvert$	$(3+(2+n_p)n_u)\lvert\mathbb{G}\rvert$	$(3+n_p)\lvert\mathbb{G}\rvert$
RS-CPABE-ASP	$\lvert\mathbb{G}_T\rvert$+$(3+5n_a+n_p)\lvert\mathbb{G}\rvert$	$(1+3n_u)\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert$
RABE-DI	$2\lvert\mathbb{G}_T\rvert$+$(2+4n_a)\lvert\mathbb{G}\rvert$	$(2+n_u)\lvert\mathbb{G}\rvert$	N/A
CC-ABE	$\lvert\mathbb{G}_T\rvert$+$(2+4n_a+2n_{cop})\lvert\mathbb{G}\rvert$	$(2+2n_u)\lvert\mathbb{G}\rvert$	N/A
Ours	$\lvert\mathbb{G}_T\rvert$+$(1+4n_a+n_{cop})\lvert\mathbb{G}\rvert$	$(1+2n_u)\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert$

[1] The absence of the algorithm in the related scheme is denoted by N/A

Performance analysis

We showcase the exceptional performance of our RVWABE-CA scheme by conducting a comprehensive comparative analysis with existing studies. Considering the three fundamental properties of our proposed scheme, namely user revocation, data integrity verification, and collaborative access, we have selected RS-HABE (Wei et al. 2021), RS-CPABE-ASP (Huang et al. 2023), RABE-DI (Ge et al. 2022), and CC-ABE (Xue et al. 2019) schemes for comparative analysis. Among them, RS-HABE and RS-CPABE-ASP support user revocation efficiently, while RABE-DI supports both user revocation and data integrity verification. However, to the best of our knowledge, there is currently no scheme that simultaneously supports user revocation, data integrity verification and collaborative access. Therefore, we select scheme CC-ABE for comparison in order to evaluate the efficiency of collaborative access in our proposed scheme.

To provide a comprehensive evaluation, both theoretical and experimental analyses are conducted.

Theoretical analysis

The essential properties of our RVWABE-CA scheme and related work are summarized in Table 2. Table 3 displays the computational expenses associated with our approach and other relevant studies at each stage. The comparison of the storage overhead of our scheme and related works is presented in Table 4 for further analysis. The meanings of the symbols used in these tables are as follows.

- E: The computational overhead of an exponentiation operation.
- P: The computational overhead of pairing operation.
- $|\mathbb{G}|$: The size of an element in the group $\mathbb{G}$ with the prime order p.
- $|\mathbb{G}_T|$: The size of an element in the group $\mathbb{G}_T$ with the prime order p.
- n_a: The number of attributes included in the access policy.
- n_{cop}: The number of cooperation nodes used for collaborative access.
- n_c: The number of attributes required to decrypt a ciphertext.
- n_p: The number of levels of the binary tree used to manage the encrypted version.
- n_u: The number of attributes associated with a user.

Since the number of levels n_p of the binary tree utilized for managing the encrypted version is a fixed value, and the number of cooperation nodes n_{cop}, which also represents the number of users participating in collaborative access, is usually small and fixed, we temporarily ignore these two parameters in the theoretical analysis stage.

According to Table 2, the RS-HABE, RS-CPABE-ASP, RABE-DI, and our RVWABE-CA schemes all incorporate user-level revocation, except for CC-ABE which primarily focuses on multi-user collaborative access. In the context of security, with the exception of the RS-CPABE-ASP scheme that attains enhanced adaptive security by employing the arithmetic span program (ASP) as its access control structure, our RVWABE-CA scheme and most other schemes achieve selective security. Compared with RS-HABE, RS-CPABE-ASP, and RABE-DI which implement user revocation, our scheme additionally achieves collaborative access. As Ge et al. (2022) consider in their paper, there is no guarantee that the cloud server updates the ciphertext as required, so it is necessary to authenticate the integrity of the updated ciphertext, while our RVWABE-CA scheme implements this verification function.

The encryption overhead of our scheme, as shown in Table 3, includes a pairing operation to embed the encrypted version in the ciphertext. But other schemes perform more exponentiation operations than our scheme due to management encryption periods (RS-HABE and RS-CPABE-ASP), data integrity verification (RABE-DI) or redundant attributes (CC-ABE). According to the data from Wei et al. (2021), a pairing operation can be perceived as a fixed number of exponential operations. Therefore, compared to other schemes, our RVWABE-CA scheme reduces the number of exponentiation operations from $2n_a$ to $3n_a$ in the encryption algorithm. The ciphertext update algorithm of the proposed scheme only needs a constant number of exponential operations, which is significantly more efficient than other schemes whose computational overhead increases linearly with n_a. This is because we refrain from modifying the access control-related attributes during ciphertext updates, opting instead to update them during key updates. However, It results in a computational cost for the key update algorithm that varies linearly with the number of user attributes n_u, resembling the RS-HABE scheme but surpassing the RS-CPABE-ASP scheme in terms of complexity. In terms of decryption overhead, compared with our RVWABE-CA scheme, the RS-HABE, RS-CPABE-ASP, RABE-DI, and CC-ABE schemes increase n_c to $2n_c$ pairing operations due to similar reasons as explained for encryption computational cost analysis. Finally, thanks to the elimination of redundant attributes, our RVWABE-CA scheme reduces $2n_c$ pairing operations compared with the CC-ABE scheme in collaborative decryption.

As indicated in Table 4, the ciphertext storage overhead of our scheme is similar to CC-ABE, slightly higher than RS-HABE, and lower than RS-CPABE-ASP and RABE-DI. This is due to the impact of the attribute's weight in our system, which leads to a rise in the ciphertext's length. However, this increase is comparatively smaller than that induced by the ASP access structure of RS-CPABE-ASP by n_a. On the other hand, RABE-DI introduces an additional element from group $\mathbb{G}_T$ and $2n_a$ additional elements from group $\mathbb{G}$ to enable verification of data integrity. In terms of key storage overhead, the RS-HABE scheme needs to manage time periods and RS-CPABE-ASP uses ASP as the access control structure, which leads to their high key storage overhead. The key storage overhead of our scheme is similar to that of CC-ABE scheme. The RABE-DI scheme exhibits the lowest storage overhead among these schemes due to its implementation of a limitation on the overall quantity of attributes. Similar to the RS-CPABE-ASP scheme, our RVWABE-CA scheme maintains a constant update key

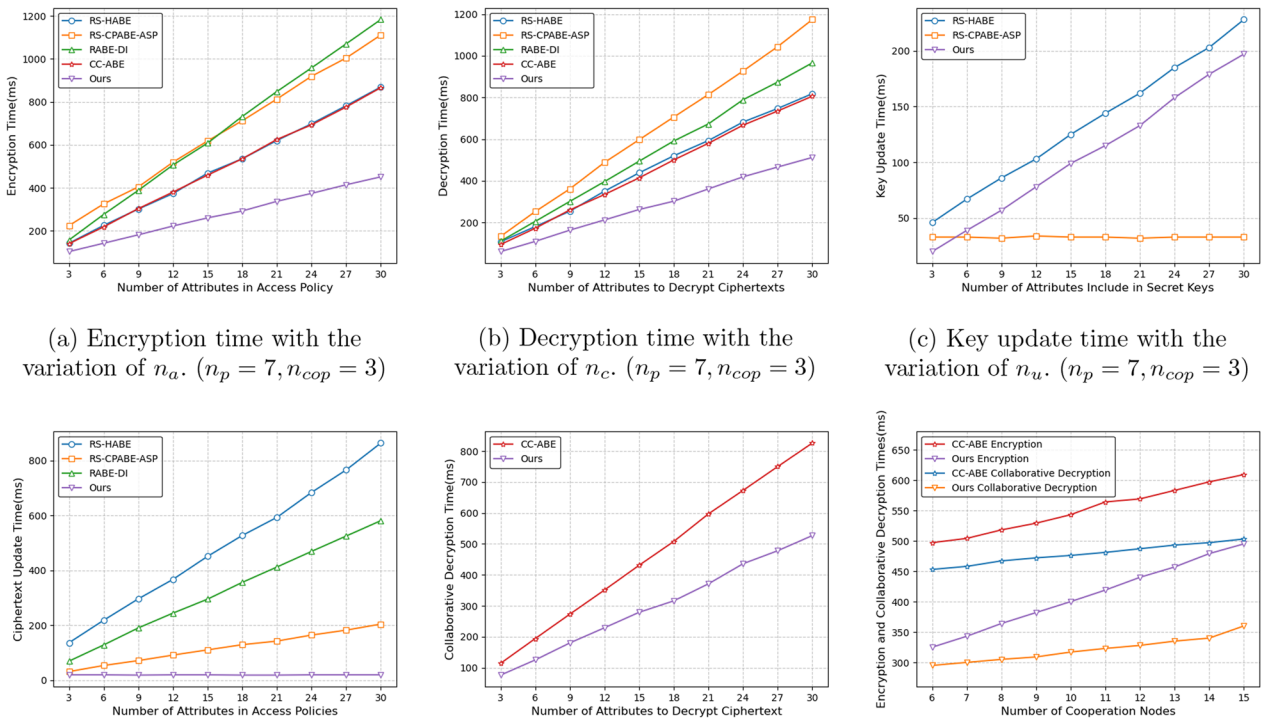
size of $2|\mathbb{G}|$, whereas RS-HABE experiences linear growth in its update key size due to the necessity of managing time periods.

In conclusion, compared with related revocable ABE schemes, our scheme additionally realizes multi-user collaborative access and realizes data verification function by considering the integrity of updated ciphertext as RABE-DI scheme. In terms of computational cost, except for higher key update cost than the constant level computation cost of the RS-CPABE-ASP scheme, our RVWABE-CA scheme significantly reduces the computational cost of encryption, decryption, key update, ciphertext update, and collaborative decryption by at least $2n_a$ exponential or $2n_c$ pairing operations compared to related schemes. Finally, in terms of storage cost, except the storage overhead of the ciphertext in our scheme is marginally greater compared to that of the RS-HABE scheme, the storage overhead of our RVWABE-CA scheme is similar to or slightly better than related schemes.

Experimental analysis

We implement RS-HABE, RS-CPABE-ASP, RABE-DI, CC-ABE, and our RVWABE-CA schemes in Java using the Java Pairing Based Cryptography (JPBC) Library version 2.0.0 and type A pairing parameters. The host computer for the experiments has 64-bit Windows 11 OS with 11th Gen Intel(R) Core(TM) i5-11260 H @ 2.60GHz and 16.0 GB RAM. In the experiments, type A pairing is constructed on the curve $y^2 = x^3 + x$ over the field $\mathbb{F}_q$, and the group $\mathbb{G}$ and group $\mathbb{G}_T$ of order p are subgroups of $E_{(F_q)}$, where the p is 160 bits and the q is 512 bits. The hash functions in these schemes are SHA-256.

Considering practical applications, we choose $n_p = 7$ to accommodate 127 time periods in the scheme, and $n_{cop} = 3$ to ensure three-party collaborative access. We take the values of n_a, n_c and n_u in set $\{3, 6, 9, 12, 15, 18, 21, 24, 27, 30\}$ when conducting experiments to compare the efficiency of these schemes. In particular, when comparing the encryption and decryption efficiency of the CC-ABE scheme and the RVWABE-CA scheme, we take the value of n_{cop} in set $\{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ to compare the efficiency under different n_{cop} values, where $n_a = 15$. We compare the efficiency of encryption, decryption, key update, ciphertext update, and collaborative decryption of RS-HABE, RS-CPABE-ASP, RABE-DI, CC-ABE, and our RVWABE-CA schemes. The experiments are



(a) Encryption time with the variation of n_a. $(n_p = 7, n_{cop} = 3)$

(b) Decryption time with the variation of n_c. $(n_p = 7, n_{cop} = 3)$

(c) Key update time with the variation of n_u. $(n_p = 7, n_{cop} = 3)$

(d) Ciphertext update time with the variation of n_a. $(n_p = 7, n_{cop} = 3)$

(e) Collaborative Decryption time with the variation of n_c. $(n_p = 7, n_{cop} = 3)$

(f) Encryption and Collaborative Decryption time with the variation of n_{cop}. $(n_p = 7, n_a = 15)$

Fig. 6 The average running time of the proposed RVWABE-CA scheme under the different parameter settings

independently conducted 100 times for each comparison, and the final results are obtained by averaging the outcomes.

As depicted in Fig. 6a, the encryption time of these schemes exhibits a positive linear correlation with the number of attributes n_a. This correlation can also be observed in the algorithm's decryption performance, key updating, ciphertext updating, and collaborative decryption. The cause for this result can be attributed to the fact that the ABE algorithm primarily focuses on attribute computation, which remains unaffected by the access control structure. On the other hand, compared with related schemes, our RVWABE-CA scheme has an obvious advantage in encryption efficiency, which becomes increasingly noticeable with the growth in the number of attributes. It is evident that when the count of attributes reaches 30, Our RVWABE-CA scheme demonstrates an average improvement in encryption efficiency of 55.24% compared to the other two schemes. The main cause behind this phenomenon is that the related schemes impose a higher computational burden on each attribute in comparison to our RVWABE-CA scheme. Specifically, our scheme performs two exponentiation computations for each attribute. In contrast, RS-HABE calculates four exponentiation computations for each attribute as it employs time periods to implement user revocation and utilizes a linear secret sharing scheme (LSSS) as an access control structure. In addition to utilizing time periods for user revocation, the RS-CPABE-ASP scheme further leverages ASP as an access control structure, resulting in each attribute requiring five exponentiation computations. The RABE-DI scheme necessitates multiple computations of the same attribute during data integrity verification, similar to the computational burden induced by redundant attributes in the CC-ABE scheme. In the former, it is necessary to calculate six exponentiation computations for each attribute, while the latter requires four.

According to the decryption time depicted in Fig. 6b, the time required for decryption in these schemes shows a proportional growth as the number of attributes increases. Our scheme demonstrates significant advantages in terms of decryption efficiency, which further amplifies as the number of attributes increases. The proposed scheme demonstrates an average decryption efficiency that is 45.49% higher compared to related schemes when the number of attributes reaches 30. The reason for this result is the same as the one for the encryption time in Fig. 6a. To be specific, the proposed scheme performs two pairing operations for each attribute, whereas the RS-HABE scheme requires three, and the RS-CPABE-SAP, RABE-DI, and CC-ABE schemes all necessitate four.

The time consumption of key update and ciphertext update algorithms as a function of attributes is illustrated in Fig. 6c and d. Our scheme's key update algorithm exhibits a slightly higher efficiency compared to RS-HABE but falls short of the efficiency achieved by the RS-CPABE-ASP scheme. On the other hand, our scheme's ciphertext update algorithm significantly outperforms other schemes in terms of efficiency. The efficiency of key update of the RS-HABE scheme is comparatively lower than that of other related schemes due to the necessity of regenerating all key and ciphertext contents during the processes of updating key and ciphertext. Similarly, the lower ciphertext update efficiency of the RABE-DI scheme is due to the necessity of recalculating all content-related attributes in the ciphertext when ciphertext updating. The RS-CPABE-ASP scheme recalculates the attribute-related content in the ciphertext during updates, thereby avoiding re-computation of the attribute-related content in the key during key updating. Contrary to the RS-CPABE-ASP scheme, our scheme avoids recalculating attribute-related content in ciphertext during ciphertext updating, instead focusing on updating attribute-related components of the key. Hence, it is evident that the cost of updating the key in our scheme escalates as the number of attributes increases, whereas the cost of updating the ciphertext remains constant. In contrast, the RS-CPABE-ASP scheme exhibits an opposite trend. However, our scheme exhibits an increase of approximately 0.17s in key update time compared to the RS-CPABE-ASP scheme, even when the number of attributes reaches 30.

The comparison regarding the efficiency of the collaborative decryption algorithm is illustrated in Fig. 6e and f. It is evident that our scheme exhibits superior efficiency in collaborative decryption compared to CC-ABE, and this advantage becomes more pronounced as the number of attributes increases. Specifically, when there are 30 attributes, our scheme achieves a remarkable improvement of 36.19%. The reason for this result is the same as the analysis of the decryption overhead for Fig. 6b. We primarily examine the impact of the number of cooperative nodes on encryption time and collaborative decryption time, as depicted in Fig. 6f. It is evident that the efficiency advantage of our scheme in collaborative decryption remains unaffected regardless of the quantity of cooperating nodes. This result arises from two factors: firstly, the advantage of our scheme is not related to the cooperation nodes, but rather the elimination of redundant attributes; secondly, CC-ABE and our scheme employ the same calculations for cooperation nodes. The efficiency advantage of our scheme's encryption decreases as the number of cooperation nodes increases. This is attributed to the fact that our scheme requires

three exponentiation operations for each cooperation node, while the CC-ABE scheme only requires two. However, in the worst-case scenario, where the number of cooperation nodes equals the number of attributes at 15, our scheme still maintains an advantage of approximately 100ms.

In conclusion, the experimental results demonstrate that except for a slightly lower key update efficiency than the RS-CPABE-ASP scheme, our RVWABE-CA scheme has obvious efficiency advantages in encryption, decryption, ciphertext update, and collaborative decryption algorithms. Specifically, our RVWABE-CA scheme outperforms related schemes by an average of 55% in terms of encryption efficiency, 45% in terms of decryption efficiency, and 36% in terms of collaborative decryption efficiency. Besides, the advantages of our RVWABE-CA scheme become increasingly evident with the growing number of attributes.

Conclusion

In this paper, to improve the capability of expressing in ciphertext-policy attribute-based encryption, we first introduced the special attribute policy (SAP), which can provide users with flexible weighted multi-user collaborative access control, and consider how to implement it along with user revocation and data integrity verification. In order to solve these problems in an efficient and feasible way, we proposed a revocable and verifiable weighted attribute-based encryption with collaborative access scheme (RVWABE-CA) by constructing the weighted multi-user collaborative access tree as the access control structure, utilizing encryption version information for user revocation, and constructing Merkle Hash Trees (MHT) for data integrity verification. Under the DBDH assumption, we proved that our scheme has the security of indistinguishability under chosen plaintext attack (IND-CPA). For comparison with related works, we conducted exhaustive experiments to show that except for a lower efficiency in the key update algorithm, our scheme has a significant advantage in computational efficiency of encryption, decryption, ciphertext update, and collaborative decryption without sacrificing storage and communication overhead. Thus, our RVWABE-CA scheme is highly promising to provide efficient and flexible weighted collaborative access control supporting user revocation as well as data integrity verification for electronic health record (EHR) systems.

Acknowledgements
The authors would like to thank the reviewers for their constructive comments that have greatly improved the paper.

Author contributions
XL: Conceptualization, Resources, Supervision, Methodology HW: Software, Investigation, Writing-Original draft preparation SM: Writing-Reviewing and Editing, Formal analysis, Project administration MX and QH: Assist in revision

Funding
This work is supported in part by the National Natural Science Foundation of China under Grant 61872409, Grant 61872152 and Grant 62272174, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515010751, in part by the Guangdong Major Project of Basic and Applied Basic Research under Grant 2019B030302008, in part by the Science and Technology Program of Guangzhou under Grant 201902010081, and in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515011194.

Availability of data and materials
The datasets used during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors have no conflicts of interest to declare that are relevant to the content of this article.

Received: 24 September 2023 Accepted: 18 January 2024
Published online: 03 March 2024

References

Alshareef H, Pardo R, Schneider G et al (2020) A collaborative access control framework for online social networks. J Log Algebr Methods Program 114:100562. https://doi.org/10.1016/j.jlamp.2020.100562

Attrapadung N, Imai H (2009) Conjunctive broadcast and attribute-based encryption. In: Shacham H, Waters B (eds) Pairing-Based Cryptography - Pairing 2009. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 248–265

Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP '07), pp 321–334, https://doi.org/10.1109/SP.2007.11

Chen J, Niu J, Lei H et al (2023) Adaptively secure multi-authority attribute-based broadcast encryption in fog computing. Comput Netw 232:109844. https://doi.org/10.1016/j.comnet.2023.109844

Chen N, Li J, Zhang Y et al (2022) Efficient cp-abe scheme with shared decryption in cloud storage. IEEE Trans Comput 71(1):175–184. https://doi.org/10.1109/TC.2020.3043950

Cui J, Zhou H, Zhong H et al (2018) Akser: attribute-based keyword search with efficient revocation in cloud computing. Inf Sci 423:343–352. https://doi.org/10.1016/j.ins.2017.09.029

Edemacu K, Jang B, Kim JW (2020) Collaborative ehealth privacy and security: an access control with attribute revocation based on OBDD access structure. IEEE J Biomed Health Inform 24(10):2960–2972. https://doi.org/10.1109/JBHI.2020.2973713

Eltayieb N, Elhabob R, Hassan A et al (2019) An efficient attribute-based online/offline searchable encryption and its application in cloud-based reliable smart grid. J Syst Archit 98:165–172. https://doi.org/10.1016/j.sysarc.2019.07.005

Fan CI, Tseng YF, Feng CC (2021) Cca-secure attribute-based encryption supporting dynamic membership in the standard model. In: 2021 IEEE Conference on Dependable and Secure Computing (DSC), pp 1–8, https://doi.org/10.1109/DSC49826.2021.9346247

Ge C, Susilo W, Baek J et al (2022) Revocable attribute-based encryption with data integrity in clouds. IEEE Trans Depend Secure Comput 19(5):2864–2872. https://doi.org/10.1109/TDSC.2021.3065999

Goyal V, Pandey O, Sahai A, et al (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. Association for Computing Machinery, New York, NY, USA, CCS '06, p 89-98, https://doi.org/10.1145/1180405.1180418,

Hoang VH, Lehtihet E, Ghamri-Doudane Y (2019a) Forward-secure data outsourcing based on revocable attribute-based encryption. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp 1839–1846, https://doi.org/10.1109/IWCMC.2019.8766674

Hoang VH, Lehtihet E, Ghamri-Doudane Y (2019b) Forward-secure data outsourcing based on revocable attribute-based encryption. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp 1839–1846, https://doi.org/10.1109/IWCMC.2019.8766674

Huang K (2021) Secure efficient revocable large universe multi-authority attribute-based encryption for cloud-aided iot. IEEE Access 9:53576–53588. https://doi.org/10.1109/ACCESS.2021.3070907

Huang X, Xiong H, Chen J et al (2023) Efficient revocable storage attribute-based encryption with arithmetic span programs in cloud-assisted internet of things. IEEE Trans Cloud Comput 11(2):1273–1285. https://doi.org/10.1109/TCC.2021.3131686

Ionita A (2022) Weighted attribute-based encryption with parallelized decryption. Cryptology ePrint Archive, Paper 2022/605, https://eprint.iacr.org/2022/605

Lai J, Guo F, Susilo W et al (2022) Generic conversions from CPA to CCA without ciphertext expansion for threshold abe with constant-size ciphertexts. Inf Sci 613:966–981. https://doi.org/10.1016/j.ins.2022.08.069

Li H, Yu K, Liu B et al (2022) An efficient ciphertext-policy weighted attribute-based encryption for the internet of health things. IEEE J Biomed Health Inform 26(5):1949–1960. https://doi.org/10.1109/JBHI.2021.3075995

Li M, Huang X, Liu JK et al (2014) Go-abe: Group-oriented attribute-based encryption. In: Au MH, Carminati B, Kuo CCJ (eds) Network and System Security. Springer International Publishing, Cham, pp 260–270

Li W, Ni W, Liu D et al (2018) Unified ciphertext-policy weighted attribute-based encryption for sharing data in cloud computing. Appl Sci. https://doi.org/10.3390/app8122519

Li W, Xu L, Wen Y et al (2022) Conjunctive multi-key searchable encryption with attribute-based access control for ehr systems. Comput Stand Interfaces 82:103606. https://doi.org/10.1016/j.csi.2021.103606

Merkle RC (1980) Protocols for public key cryptosystems. In: 1980 IEEE Symposium on Security and Privacy, pp 122–122, https://doi.org/10.1109/SP.1980.10006

Qin B, Zhao Q, Zheng D et al (2019) (Dual) server-aided revocable attribute-based encryption with decryption key exposure resistance. Inf Sci 490:74–92. https://doi.org/10.1016/j.ins.2019.03.053

Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Cramer R (ed) Advances in Cryptology - EUROCRYPT 2005. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 457–473

Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613. https://doi.org/10.1145/359168.359176

Tian Q, Han D, Jiang Y (2019) Hierarchical authority based weighted attribute encryption scheme. Comput Sci Inf Syst 16:797–813

Wang H, Li Y, Susilo W et al (2022) A fast and flexible attribute-based searchable encryption scheme supporting multi-search mechanism in cloud computing. Comput Stand Interfaces 82:103635. https://doi.org/10.1016/j.csi.2022.103635

Wang S, Liang K, Liu JK et al (2016) Attribute-based data sharing scheme revisited in cloud computing. IEEE Trans Inf Foren Secur 11(8):1661–1673. https://doi.org/10.1109/TIFS.2016.2549004

Wang S, Zhou J, Liu JK et al (2016) An efficient file hierarchy attribute-based encryption scheme in cloud computing. IEEE Trans Inf Foren Secur 11(6):1265–1277. https://doi.org/10.1109/TIFS.2016.2523941

Wang Y, Zhang D, Zhong H (2014) Multi-authority based weighted attribute encryption scheme in cloud computing. In: 2014 10th International Conference on Natural Computation (ICNC), pp 1033–1038, https://doi.org/10.1109/ICNC.2014.6975982

Wei J, Chen X, Huang X et al (2021) Rs-habe: revocable-storage and hierarchical attribute-based access scheme for secure sharing of e-health records in public cloud. IEEE Trans Depend Secure Comput 18(5):2301–2315. https://doi.org/10.1109/TDSC.2019.2947920

Xu S, Yuan J, Xu G et al (2020) Efficient ciphertext-policy attribute-based encryption with blackbox traceability. Inf Sci 538:19–38. https://doi.org/10.1016/j.ins.2020.05.115

Xue Y, Xue K, Gai N et al (2019) An attribute-based controlled collaborative access control scheme for public cloud storage. IEEE Trans Inf Foren Secur 14(11):2927–2942. https://doi.org/10.1109/TIFS.2019.2911166

Yan X, Yuan X, Zhang Q et al (2020) Traceable and weighted attribute-based encryption scheme in the cloud environment. IEEE Access 8:38285–38295. https://doi.org/10.1109/ACCESS.2020.2975813

Zhang Y, Chen X, Li J, et al (2013) Fdr-abe: Attribute-based encryption with flexible and direct revocation. In: 2013 5th International Conference on Intelligent Networking and Collaborative Systems, pp 38–45, https://doi.org/10.1109/INCoS.2013.16

Zhou Y, Zhao X, Liu S et al (2019) A time-aware searchable encryption scheme for EHRS. Digit Commun Netw 5(3):170–175. https://doi.org/10.1016/j.dcan.2018.09.003

Publisher's Note