Hindawi
Security and Communication Networks
Volume 2021, Article ID 8855341, 14 pages
https://doi.org/10.1155/2021/8855341

Research Article

CP-ABE-Based Secure and Verifiable Data Deletion in Cloud

Jun Ma,[1,2] **Minshen Wang**,[3] **Jinbo Xiong**,[3] **and Yongjin Hu**[2]

[1]*School of Telecommunications Engineering, Xidian University, Xi'an 710071, China*
[2]*Information Engineering University, Zhengzhou 450001, China*
[3]*Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China*

Correspondence should be addressed to Jinbo Xiong; jbxiong@fjnu.edu.cn

Received 30 August 2020; Revised 19 October 2020; Accepted 17 March 2021; Published 28 March 2021

Academic Editor: David Megias

Cloud data, the ownership of which is separated from their administration, usually contain users' private information, especially in the fifth-generation mobile communication (5G) environment, because of collecting data from various smart mobile devices inevitably containing personal information. If it is not securely deleted in time or the result of data deletion cannot be verified after their expiration, this will lead to serious issues, such as unauthorized access and data privacy disclosure. Therefore, this affects the security of cloud data and hinders the development of cloud computing services seriously. In this paper, we propose a novel secure data deletion and verification (SDVC) scheme based on CP-ABE to achieve fine-grained secure data deletion and deletion verification for cloud data. Based on the idea of access policy in CP-ABE, we construct an attribute association tree to implement fast revoking attribute and reencrypting key to achieve fine-grained control of secure key deletion. Furthermore, we build a rule transposition algorithm to generate random data blocks and combine the overwriting technology with the Merkle hash tree to implement secure ciphertext deletion and generate a validator, which is then used to verify the result of data deletion. We prove the security of the SDVC scheme under the standard model and verify the correctness and effectiveness of the SDVC scheme through theoretical analysis and ample simulation experiment results.

1. Introduction

The rapid developments of big data, Internet of Things (IoT), and the fifth-generation mobile communication (5G) technologies promote an explosion growth in data volumes generated by users' mobile devices, which also result in the widespread popularity and further upgrading of 5G cloud storage services in cloud service provider (CSP) [1–3]. CSP provides users with massive data storage services without requiring the users to store data in local devices [4], which not only saves users' a large amount of money for building their own storage, but also can search and retrieve the required data more quickly and share the data with other users more expediently, such as Dropbox, Baidu Cloud, and Alibaba Cloud [5–7]. Meanwhile, 5G technologies enable all kinds of intelligent devices to realize fast cloud connection, making it convenient for these devices to upload data to the CSP and providing convenient services for users [8].

As we know, once users upload their personal data to the CSP, the ownership of the data is separated from the administration of them, resulting in the users completely losing control over their uploaded data from various mobile devices, such as body sensor equipment, smart rings, and smart phones [9, 10]. However, the personal data inevitably contain users' private information; if it is not securely deleted from the CSP in time after their expiration, this will lead to serious problems, such as unauthorized access, resource abuse, side channel attack, data privacy disclosure, and other disastrous consequences [11–13]. Moreover, when users want to delete the cloud data, they need to completely trust CSP. After the users send request for deleting the expired data to the CSP, which generally returns "success" or "failure" as a response, the users cannot confirm the reliability of deletion results for their cloud data. Furthermore, driven by the interests, the "honest and curious" CSP does not delete or transfer users' data in time, but returns the

"success" message to deceive the users [14]. Therefore, this phenomenon leads to various types of users' privacy disclosure occurring frequently; for example, Facebook emails suggest that it considered selling users' data to third parties, and some cloud platforms have authorized their partners to get access to the sensitive information of users' data [15, 16].

Secure data deletion is a crucial part of protecting users' data security and privacy within their whole lifecycle [9, 17], and data deletion verification provides protection and assuring for secure data deletion [18]. Related researchers have studied secure data deletion and obtained certain research findings. Xiong et al. [19] introduced and analyzed related methods of secure data deletion based on cryptography [20]. These methods can be mainly divided into data assured deletion schemes based on trusted execution environment, key management, and access control policies. However, all of these methods lack verification technique of deletion results for cloud data. As a result, exploring how to implement secure data deletion and deletion verification for cloud data is of great significance to healthy development of the cloud computing. There are four main solutions for secure data deletion and verification in the cloud, such as overwrite-based, provable data possession (PDP)-based, blockchain-based, and attribute-based encryption (ABE)-based schemes. Among these methods, ABE has a flexible access policy that enables fine-grained access control for cloud data [21, 22]. In particular, ciphertext-policy ABE (CP-ABE) [23, 24] can help us achieve fine-grained policy management and flexible access control for constructing flexible scheme for fine-grained secure data deletion and verification in the cloud.

Therefore, this paper proposes a novel secure data deletion and verification (SDVC) scheme based on CP-ABE for cloud data in cloud, which includes a secure data deletion method and a data deletion verification method. The SDVC scheme constructs an attribute association tree (AAT) and a rule transposition algorithm (RTA) to realize attribute revocation [24] and reencrypting keys to quickly delete the cloud data and verify the data deletion results. The SDVC achieves rapid data deletion and results verification, and the main contributions of the SDVC scheme are as follows:

(i) A secure data deletion method based on CP-ABE algorithm implements fine-grained deletion control of cloud data, which constructs an AAT to achieve fast attribute revocation. Through updating node attribute within the AAT, we can reconstruct new access policies and reencrypt the data; thus, the ciphertext cannot be recovered and, finally, secure data deletion is implemented in a timely manner. Meanwhile, constructing the AAT actually reduces the attribute query overhead.

(ii) A data deletion verification method is built based on RTA and overwriting algorithm, which generates random data blocks to overwrite the expired cloud data blocks. When it overwrites data, the verification values of each data block are generated and the value of root node can be generated by MHT as a data deletion validator. This method not only makes the expired data unrecoverable, but also saves communication overhead compared to the methods uploading random data directly.

(iii) The security proof demonstrates that the SDVC scheme realizes secure deletion and verification for cloud data. The theory analysis and ample simulation results indicate that the SDVC scheme is effective and efficient in implementing secure data deletion and verification compared with the related methods.

The rest of the paper is organized as follows: Section 2 describes the related work; Section 3 gives preliminaries; and Section 4 describes the problem description, including system model, scheme overview, security model, and implementation goals. In Section 5, we construct the SDVC scheme. Section 6 and Section 7 give the security analysis, theoretical analysis, and performance evaluation. Section 8 concludes the whole paper.

2. Related Work

Various cloud services bring a lot of convenience to people who are increasingly inclined to store a large amount of data into CSP [25–27]. Once users upload their individual data to the cloud, the ownership of the data are separated from the administration of them, resulting in users losing complete control over their data. Therefore, in the process of secure data deletion and verification, there are problems such as unauthorized access, privacy leakage, and unverifiable deletion results [28]. Relevant scholars have obtained certain research findings; Xiong et al. [19] summarized three main types of secure data deletion methods for cloud data: trusted execution environment, key management, and access control policies. However, these methods do not consider how to verify the deletion results [29]. The existing research on cloud data deletion and verification can be mainly divided into the following four solutions, as described in Table 1.

2.1. Overwrite-Based Deletion and Verification. Regarding the overwrite-based deletion verification method, Paul and Saxena [30] proposed a provable, overwrite-based data deletion verification scheme, where the users first generate the same size of random data as the expired data and upload it to the CSP to overlay the expired data. After performing the overwriting operation, CSP will generate a data validator. If the returned validator is the same as the locally generated validator, the expired data is considered to be securely deleted. On the contrary, CSP may not delete data in time. Du et al. [31] proposed a deletion verification scheme for cloud data based on overwriting verification, which uses CP-ABE algorithm [36] to encrypt plaintext. When the cloud data expires, the ciphertext associated access policy is changed and used to reencrypt the expired ciphertext to achieve secure deletion. Furthermore, a searchable path of hash binary tree is generated according to the number of expired data blocks. Starting from the root node of the binary tree, this solution hierarchically traverses the binary tree, generates random binary data of the same size as the

Table 1: Comparative analysis of related data deletion and verification solutions.

Classifications	Solutions	Techniques	Advantages	Limitations
Overwrite-based	[30]	Random data overwriting	Simple data deletion and verification	Huge commu. overhead
	[31]	CP-ABE, random data overwriting	Flexible policy and easy verification	Comput., commu. overhead
	[32]	TPM, signature verification	Transparent, public verification	Small storage capacity
PDP-based	[33]	Dynamic PDP, skip-list structure	Controllability, confidentiality	Complexity, key management
Blockchain-based	[34]	MHT, timestamp, blockchain	Simple verification	Not dynamic and fine-grained
	[18]	Smart contract, blockchain	Simple verification in cloud	Not dynamic and fine-grained
ABE-based	[35]	KP-ABE, reencrypting	Flexible, fine-grained	Huge commu. overhead
	SDVC	CP-ABE, AAT, RTA, MHT	Flexible, fine-grained, effectiveness	Decryption overhead

expired data, obtains the shortest path between the root node and each leaf node, then records all the node sequence numbers of the shortest path, converts them into binary expired data, and performs an XOR bit by bit to generate new data, which is used to overwrite the expired data. Finally, the algorithm calculates the value of each leaf node data as a validator by hash operation and verifies the data deletion results with the validator. Hao et al. [32] proposed a data deletion verification scheme based on trusted platform module (TPM). The basic idea of this scheme is to store the key through additional TPM hardware module configured by the CSP and perform data encryption and decryption operations [37]. When the expired data needs to be cleared, the TPM module performs the key deletion operation. Besides, the TPM module verifies the signature of the key deletion operation and uses the signature verification as a basis to prove whether the CSP has not performed the key deletion operation or tamper with the information of storage key in TPM. The TPM makes the deletion process more transparent and deletion results supporting public authentication, but is limited by the small storage capacity of its module. Miao et al. [38] proposed a solution where the data owners first calculate the metainformation of the expired data and send it to the CSP; after the CSP deletes the expired data, the metainformation of the new data is returned to the users, and the users verify data metainformation through verified equation; if the validation passes, the data has already been deleted. In this solution, as the amount of updating and deleting data increases, this leads to linear growth of computation, communication, and storage overhead of the CSP.

2.2. PDP-Based Deletion and Verification. Liu et al. [33] proposed an improved data transfer and data deletion verification scheme, which uses the dynamic provable data possession (PDP) mechanism [39] to update the original data of the cloud with the purpose of data deletion. Furthermore, it makes the original data unrecoverable and finally verifies whether the cloud holds the expired data through the data proofing function of the dynamic PDP mechanism and the skip-list authentication structure. The program considers the requirements for controllability and data confidentiality in the deletion process. The data owner introduces custom destruction mode to realize the controllability of the data deletion process, while the data confidentiality is realized by encrypting the plaintext data and using the key control to partitioned storage.

2.3. Blockchain-Based Deletion and Verification. Yang et al. [34] proposed a data deletion public verification scheme based on blockchain technology [40]. The scenario assumes that the CSP is semitrusted [24]. Firstly, the user encrypts the data and uploads it to the CSP. When data deletion is required, the user constructs a hash chain by combining the Merkle hash tree (MHT) [41], the timestamp server, and the blockchain technique. The hash chain verifies whether the CSP has actually deleted the expired data by the value in the hash chain. Liu et al. [18] proposed a cloud data deletion verification protocol based on blockchain. Firstly, the personal identity is authenticated by the smart contract algorithm, and the deletion operation record is created at the same time. The CSP then deletes the expired data specified by the data owner and generates a hash value to join the blockchain by hashing operation. Finally, the user verifies the data deletion result through the blockchain.

2.4. ABE-Based Deletion and Verification. Xue et al. [35] proposed a secure deletion solution based on KP-ABE. The user encrypts data with the KP-ABE algorithm. When the expired data needs to be deleted, the users cannot decrypt and access the data by revoking the attribute corresponding to the expired data and reencrypting the data [42]. Before the cloud data storage, the data is encrypted, and a digital signature is generated for each data block in the ciphertext by a hash operation. This digital signature corresponds to the ciphertext attribute, and the validator is generated by the MHT and feeds it back to the user to verify the deletion result.

The deletion operation in the process of deleting data based on the overwriting-based deletion verification scheme is coarse-grained; the encryption algorithm is used to encrypt the data to ensure the security of the data, but the key is not handled securely; thus, there is a key leakage issue [43, 44]. The PDP-based deletion verification scheme does not delete the ciphertext stored in the CSP. When an unauthorized user gets access to ciphertext and cracks the cloud ciphertext through the ciphertext analysis and brute force cracking, privacy information is faced with threat of leakage [45–47]. Aiming at the issues of unauthorized access, large computational cost in the process of data deletion, and inability to implement fine-grained deletion and verification data, this work focuses on a secure and effective cloud data deletion and verification solution to achieve fine-grained data security deletion and verification through flexible and effective key management.

3. Preliminaries

In this section, we give two preliminaries, Merkle hash tree and CP-ABE.

3.1. Merkle Hash Tree. MHT [41] is a full binary tree authentication data structure that can be used to verify the correctness of data storage [48]. Usually, the value on each node of the MHT is a hash value of stored data, and the value of a parent node is obtained by hashing the value of its child nodes; thus, the root node can be sequentially derived out by the leaf node. Suppose that there is a data set $\{d_1, d_2, d_3, d_4\}$; $\{h_4, h_5, h_6, h_7\}$ are the hash values of data set $\{d_1, d_2, d_3, d_4\}$, respectively. The hash value of the parent node is calculated from the child node until the hash value of the root node is obtained: $h_2 = H(h_4 \| h_5)$, $h_3 = H(h_6 \| h_7)$, $h_1 = H(h_2 \| h_3)$. In order to verify completeness and correctness of data $\{d_1, d_2, d_3, d_4\}$, the verifier can implement the goals by constructing MHT and calculating the root node.

3.2. CP-ABE. Sahai and Waters first proposed the idea of attribute-based encryption (ABE) [49]. As a typical ABE, the ciphertext-based attribute encryption algorithm (CP-ABE) is widely used in cloud data access control [50–52]. The main principle of CP-ABE is to set the attribute set A, private key SK, ciphertext CT, and access policy $\mathbb{A}$. $\mathbb{A}$ is associated with CT, SK can be obtained through attribute set A, and access policy $\mathbb{A}$ is derived from attribute set A by the threshold function operation. The ciphertext can only be accessed if the value of threshold is met after the logical intersect operation between the user's A and $\mathbb{A}$ associated with CT [53]. The CP-ABE algorithm is mainly composed of the following four algorithms:

(i) Initialization, performed by a trusted authority: it takes as input a security parameter and attribute descriptions, generates a public key PK and a master key MK, and protects the confidentiality of MK.

(ii) Encryption, performed by the users: it takes as input a plaintext f, a system public key PK, and the access policy $\mathbb{A}$ and outputs a ciphertext CT. $\mathbb{A} = (\Lambda, \rho)$ through linear secret sharing scheme (LSSS), where Λ is a $l \times n$ matrix and function ρ is the line attribute tag function of Λ. $\mathbb{A}$ is implicitly included in the corresponding CT.

(iii) Key generation, performed by a trusted authority: it takes as input the master key MK and an attribute set A used to describe the key and then outputs the private key SK.

(iv) Decryption, executed by the user: it takes as input the ciphertext CT and the private key SK. Ciphertext CT can only be decrypted if the attribute set satisfies the access policy $\mathbb{A}$.

4. Problem Description

This section mainly describes the system model, scheme overview, security model, and implementation goals of the SDVC scheme. The main symbols and descriptions in the SDVC scheme are shown in Table 2.

4.1. System Model. The system model of SDVC scheme is shown in Figure 1. It consists of four entities: cloud service provider, trusted authority, data owner, and data user.

(i) Cloud service provider (CSP): it has powerful computing power and storage resources and provides various services such as data storage and data distribution for users. CSP has the property of being "honest and curious" and may retain expired data driven by interest.

(ii) Trusted authority (TA): it starts the system initiating process, generates master key and public parameter, and securely protects users' keys.

(iii) Data owner (DO): the use of various cloud applications in daily life makes the amount of data increasingly large. However, our smart devices have limited storage and computing resources, so the DO should lease CSP rich storage resources and computing resources for data storage and distribution.

(iv) Data user (DU): it requests the private key from the TA through the owned attribute set, and then DU retrieves data from the CSP. After the used data is expired, the plaintext, ciphertext, and key will be deleted in a timely manner.

Firstly, DO divides the original data F into a number of data blocks f with 64 MB; after that, f is encrypted by using an AES-256 algorithm, and then the DO uploads the ciphertext to CSP. The TA generates the public key of the CP-ABE algorithm and publishes it to the DO, who encrypts the symmetric key dk by the CP-ABE algorithm through the public key and sends it to the TA. If the attribute set owned by the DU satisfies the access policy, then TA allocates a private key for the DU, who decrypts and obtains the dk by the private key of CP-ABE, and then decrypts the ciphertext to get the original data. When part of the data expires, the DO sends a deleting request to TA to delete the ciphertext of dk. The DO updates the attribute through the AAT, reencrypts the data-associated dk, and sends it to the TA using the new access policy. At the same time, the DO sends a random data block and deletion parameters to the CSP. The deletion parameter triggers the rule transposition algorithm to generate random data of the same size as the expired data, overwrites the expired data, and generates a validation value every time during the overwriting process for generation of verifier by the MHT. After the CSP overwrites the expired data, it returns the validator to the DO within a reasonable time range, and the DO compares it with the local validator to complete the verification.

4.2. Scheme Overview. The SDVC scheme is mainly composed of eleven algorithms: DO initialization DODeup, TA initialization TASetup, f encryption FileEnc, dk encryption DKEnc, private key generation SKGen, dk decryption DKDec, f decryption FileDec, attribute revocation ARev,

Table 2: Symbols and descriptions.

Symbol	Description
F, f	Data and subdata blocks
CT, M	File ciphertext and key ciphertext
dk	Symmetric key
A, a	Attribute and attribute values
rule	Transposition rule
σ, sig	Index and digital signature
MK, SK, PK	Master key, private key, and public key
H, h	Hash function and hash value
$\mathbb{A}$	Access policy
λ, γ	Security parameter
rule	Transposition rule
v	Validator
Att	Attacker

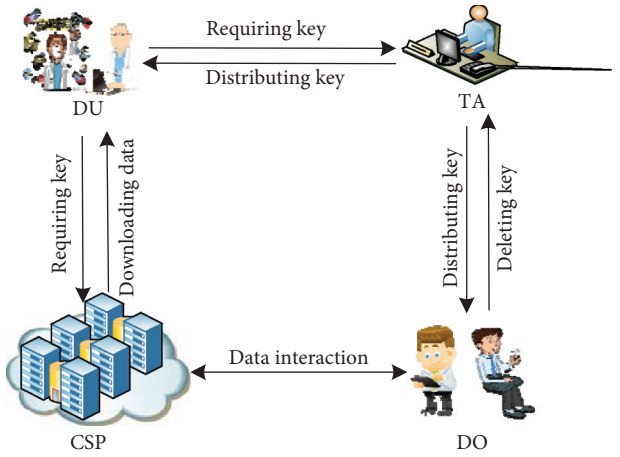

Figure 1: System model.

rule transpose algorithm Transpose, overwriting algorithm OverWri, and verification algorithm Verify.

DOSetup (1^{λ}) is executed by DO. It takes a security parameter λ as input, generates a symmetric key dk, and constructs access policy $\mathbb{A}$ through attribute set A.

TASetup (1^{γ}) is completed by TA. It takes a security parameter γ as input, generates the public key PK of CP-ABE algorithm and master key MK, and then distributes PK to DO and maintains MK confidentiality.

FileEnc (f, dk) is completed by DO. It takes plaintext data and symmetric key dk as input, generates ciphertext data CT by using AES-256 algorithm, and uploads CT to the CSP.

DKEnc $(\mathrm{dk}, \mathrm{PK}, \mathbb{A})$ is executed by DO. It uses public key PK and access strategy $\mathbb{A}$ to encrypt dk into ciphertext M_{dk} associated with $\mathbb{A}$ by CP-ABE algorithm and uploads M_{dk} to TA for secure storage.

SKGen (MK, ψ) is completed by TA. It takes the master key MK and the corresponding attribute set ψ as input and outputs the private key SK of the CP-ABE algorithm.

KeyDec $(M_{\mathrm{dk}}, \mathrm{SK})$ is completed by TA. It takes SK as input, and if the attribute set of DU satisfies the access policy $\mathbb{A}$, TA decrypts M_{dk} and returns dk to DU.

FileDec (CT, dk) is completed by DU. It takes dk and the ciphertext CT as input and outputs the plaintext F by AES-256 algorithm.

AttRev (A) is completed by DO. DO revokes the attributes, rebuilds the access structure $\mathbb{A}'$, and then uses PK and $\mathbb{A}'$ to reencrypt dk to get M'_{dk}. After that, DO requests TA to update M_{dk} to M'_{dk}.

Transpose (f_R, rule) is completed by CSP. It takes as input a random data block f_R and a transposition rule, rule, uploaded by DO. It outputs a random data block with the same size as the expired data by the RTA, and a hash value of each random data block by a hash function.

OverWri (f_R) is completed by CSP. It takes the generated random data block f_R as input, overwrites the expired data block multiple times using a random overwriting algorithm, and generates a verification value at the same time.

Verify $(v_{\mathrm{Root}}, v'_{\mathrm{Root}})$ is performed by DO. It generates the validator from verification value through MHT algorithm. DO compares the validator v'_{Root} fed back from CSP with the locally generated validator v_{Root}, and the result will verify whether the expired data is successfully overwritten or deleted.

4.3. Security Model. The security model of the SDVC scheme mainly considers the confidentiality of the ciphertext when it is attacked by an attacker Att after the data is deleted. It is well known that AES-256 cannot be compromised in probabilistic polynomial time (PPT), so this SDVC considers that Att will attack the key to decrypt data by allowing Att to query the private key of any access policy from TA except the target policy. The security model is formalized under indistinguishable encryption against chosen plaintext attack (IND-CPA) to the selected attribute set [51, 52].

(i) Initialization stage: the attacker Att sets a series of attribute cracking keys and sends the attribute set to the challenger. Firstly, the challenger generates public key PK and master key MK, sends the PK to the attacker, and protects the confidentiality of the MK. Then, Att requests the private key SK from the challenger:Att sends a randomly constructed attribute set to the challenger, and the challenger executes the algorithm SKGen to generate Att's private keys which are attribute-associated with Att except for satisfying the access structure $\mathbb{A}$.

(ii) Challenge stage: Att selects two equal lengths of plaintext, respectively, M_0, M_1, and sends them to the challenger to request encryption; the challenger randomly chooses $b \in \{0, 1\}$ and then returns ciphertext CT_b to Att. At this stage, Att initiates a series of encryption requests and gets the feedback from the challenger.

(iii) Guess stage: Att outputs guess $b' \in \{0, 1\}$; if $b' = b$, then Att wins in the challenge and regards Att as IND-CPA attacker. The advantage of attacking the SDVC scheme is

$$\mathrm{Adv}_{\mathrm{Att}} = \left|\Pr[b' - b] - \frac{1}{2}\right|. \quad (1)$$

In all games, if the advantage of being successful, that is, the attacker achieves as many queries as possible, is negligible within the PPT under the decisional bilinear Diffie–Hellman (DBDH) problem, this indicates that the SDVC scheme is security.

4.3.1. DBDH Definition. Consider that there are two multiplicative cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with big prime order p, g represents the generator of $\mathbb{G}_1$, and e: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ represents a bilinear map. Given (g, g^a, g^b, g^c) and $h \in \mathbb{G}_2$, we need to decide if $h = e(g,g)^{\mathrm{abc}}$ is true.

4.3.2. DBDH Hypothesis [54]. No PPT attacker Att can distinguish quintuple $(\beta, \beta_1, \beta_2, \beta_3, e(\beta,\beta)^z)$ and $(\beta, \beta_1', \beta_2', \beta_3', e(\beta,\beta)^{z'})$ with a probability greater than negligible, where z and z' are random values from $\mathbb{Z}_p$. The advantage of Att is

$$\left|pr[(\beta, \beta_1, \beta_2, \beta_3, e(\beta,\beta)^z) = 0] - pr\left[\left(\beta, \beta_1', \beta_2', \beta_3', e(\beta,\beta)^{z'}\right) = 0\right]\right|, \quad (2)$$

where the probability is taken over the selection of $\beta \in \mathbb{G}_1$.

4.4. Implementation Goals. The SDVC scheme mainly considers the following goals:

(1) Service availability: secure data deletion operations do not affect the use of other users' data and any other services of the CSP.

(2) Unrecoverability: the expired data cannot be accessed by anyone after it is securely deleted in order to prevent unauthorized visitors from obtaining private information and attempt to recover deleted data.

(3) Fine-grained deletion: CSP deletes the specified data according to the user's requirement, while other data cannot be affected.

(4) Timeliness of deletion: the data deletion operation needs to be timely and rapid. When the data is deleted, no one can access the deleted data.

(5) Deletion verification: the CSP deletes the data and returns the deletion certificate to the user to prove that the data has been securely deleted. Therefore, users do not have to worry about the risk of privacy leakage after deleting the expired data.

5. Construction of the SDVC Scheme

The SDVC scheme mainly consists of data encryption and decryption phase, secure data deletion phase, and data deletion verification phase.

5.1. Data Encryption and Decryption Stage

5.1.1. Data Encryption. Firstly, file F is divided into n subfile data blocks $\{f_1, f_2, \ldots, f_n\}$, and the hash value of each subfile is calculated as the index value $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, with each index value pointing to a subfile. To achieve fine-grained deletion of files, file encryption operations are performed in units of data blocks. In order to get block of ciphertext ct_i, DO generates a symmetric key dk_i through security parameters λ and separately encrypts f_i by AES-256 algorithm. After encryption, DU saves $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ and other metadata, and DO generates all possible attribute sets $\{A_1, A_2, \ldots, A_n\}$ and sets the attribute weight value. Attributes have different values $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,n}\}$; according to different sensitivity levels, the data blocks are arranged in different attribute sets. Attribute value set $\{a_1, a_2, \ldots, a_m\}$ is mapped to hash set $\{h_1, h_2, \ldots, h_m\}$ satisfying finite field by using anticollision hash function H: $\{0,1\}^* \leftarrow \mathbb{Z}_p$. Then, the access policy is constructed by the hash set and DO requests public key of CP-ABE from TA. TA selects two multiplicative cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with big prime order p, for which there exists a bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$, and the generator of $\mathbb{G}_1$ is g. Indexes α, β ($\alpha\beta \in \mathbb{Z}_p$, and $\mathbb{Z}_p$ is a finite field with prime order p) are selected randomly and g^α, g^β, and $e(g,g)^\alpha$ are calculated. Then, TA calculates the public key $\mathrm{PK} = \{e, g, g^\beta, e(g,g)^\alpha\}$ and the master key MK. After that, TA securely stores MK and sends PK to DO.

5.1.2. Key Encryption. When encrypting symmetric key dk, the attribute set $\{h_1, h_2, \ldots, h_m\}$ is used to build access policies $\mathbb{A} = (\Lambda, \rho)$. To get ciphertext $\mathrm{CT}_{\mathrm{dk}}$, dk is encrypted by using public key PK and $\mathbb{A} = (\Lambda, \rho)$ through CP-ABE algorithm. The equations for the encryption process of dk are as follows:

$$\begin{cases} \overrightarrow{v} = (s, y_1, y_2, \ldots, y_n) \in \mathbb{Z}_p, \\ \lambda_i = \overrightarrow{v} \cdot \Lambda_i (i = 1, \ldots, l), (r_1, \ldots, r_l) \in \mathbb{Z}_p, \\ \mathrm{CT}_{\mathrm{dk}} = \left(ct = \mathrm{dk} \cdot e(g,g)^{\alpha s}, c' = g^s, \left(c_1 = g^{\alpha\lambda_1} h_{\rho(1)}^{-r_1}, \theta_1 = g^{r_1}\right), \ldots, \left(c_n = g^{\alpha\lambda_l} h_{\rho(l)}^{-r_l}, \theta_l = g^{r_l}\right)\right). \end{cases} \quad (3)$$

In fact, the $\overrightarrow{v} = (s, y_1, y_2, \ldots, y_n)$ is used to split the encryption index s and then distribute a random index $s \in \mathbb{Z}_p$ according to Λ, $\lambda_i = \overrightarrow{v} \cdot \Lambda_i (i = 1, \ldots, l)$ is the i-th share of the splits, and $c_i (i = 1, 2, \ldots, l)$ associates λ_i with the $\rho(i)$ property value. Symmetric keys $\{\mathrm{dk}_1, \mathrm{dk}_2, \ldots, \mathrm{dk}_n\}$ are encrypted into ciphertext $\{\mathrm{CT}_{\mathrm{dk}1}, \mathrm{CT}_{\mathrm{dk}2}, \ldots, \mathrm{CT}_{\mathrm{dk}n}\}$ by the

above encryption process and generate the corresponding digital signature SIG = $\{\text{sig}_1, \text{sig}_2, \ldots, \text{sig}_n\}$. DO encapsulates the file ciphertext, index value, and digital signature $(\text{CT}_1, \sigma_1, \text{sig}_1), (\text{CT}_2, \sigma_2, \text{sig}_2), \ldots, (\text{CT}_n, \sigma_n, \text{sig}_n)$ and uploads them to CSP for storage. Meanwhile, the ciphertext of the symmetric key $\{\text{CT}_{\text{dk}1}, \text{CT}_{\text{dk}2}, \ldots, \text{CT}_{\text{dk}n}\}$ is uploaded to TA.

5.1.3. Data Decryption. DU sends its own attribute set to the TA to request the private key of CP-AES. TA takes as input the master key MK, public key PK, and attribute set ψ; selects index $\varphi \in \mathbb{Z}_P$ randomly; calculates $\delta = g^{\alpha} g^{\beta\varphi}$, $L = g^{\varphi}$, $\delta_x = h_x^{\varphi}$ $(\forall x \in \psi)$; and then generates a private key SK = $(\delta, L, \delta_x$ $(x \in \psi))$. If the attribute set of DU is satisfied with the access policy corresponding to the CT_{dk}, the ciphertext CT_{dk} can be decrypted by the SK. Define $I = \{i: \rho(i) \in \psi\} \subset \{1, 2, \ldots, l\}$; let $\{\omega_i \in Z_q | i \in I\}$; if $\{\lambda_i\}$ is the effective share in the matrix Λ corresponding to random indexes, then $\sum_{i \in I} \omega_i \lambda_i = s$. If the attribute set ψ of DU does not meet the access policy, you cannot get the real dk. The decryption process is represented as follows:

$$\text{Des}\left(\text{CT}_{\text{dk}}\right) = \text{ct} \cdot \frac{\prod_{i \in I} e\left(c_i, \gamma\right) \cdot e\left(\theta_i, \delta_{\rho(i)}\right)^{\omega_i}}{e\left(c', \delta\right),} \tag{4}$$

$$\begin{aligned} \frac{\prod_{i \in I} e\left(c_i, \gamma\right) \cdot e\left(\theta_i, \delta_{\rho(i)}\right)^{\omega_i}}{e\left(c', \delta\right)} &= \frac{\prod_{i \in I} e\left(g, g\right)^{\varphi\beta\lambda_i\omega_i}}{e\left(g, g\right)^{\alpha s} \cdot e\left(g, g\right)^{\beta s \varphi}} \\ &= \frac{1}{e\left(g, g\right)^{\alpha s}}. \end{aligned} \tag{5}$$

5.1.4. The Decryption of File Ciphertext. The DU is authorized to request the file ciphertext within the range of permission from CSP. After downloading the ciphertext, DU uses the symmetric key dk to get the file f obtained by decrypting the ciphertext through AES-256 algorithm, and finally gets the original file F.

5.2. Secure Data Deletion Phase. The secure data deletion phase is completed by the attribute revocation of the CP-ABE algorithm and random overwriting algorithm.

Attribute revocation implements the fine-grained, timely, and logical secure deletion for the expired data in CSP. SDVC scheme constructs an AAT and assumes that n DUs $\{\text{uid}_1, \text{uid}_2, \ldots, \text{uid}_n\}$ have different attribute sets $\{\psi_1, \psi_2, \ldots, \psi_n\}$, and each attribute set has multiple attribute values $\psi_i = \{h_1, h_2, \ldots, h_l\}$. The amount of data in CSP is relatively large, and, usually, different file data blocks will be accessed by different DUs. To achieve efficient attribute revocation, the association form of each attribute value of DU is formally described as an AAT based on multibranch tree in the data structure. The highest level of attribute value in the attribute set is selected as a root node of the AAT. The main attributes usually have a high level; for example, H ("Group 1") or H ("Group 2")... is the root node; H ("Subsidiary 1") or H ("Subsidiary 2")..., H ("Department 1") or H ("Department 2")..., and H ("Group 1") or H ("Group 2")... are child nodes. The AAT is established through the relationship among the attributes, assuming that the data block is $\{d_1, d_2, \ldots, d_9\}$, as shown in Figure 2. The attribute revocation is divided into two cases: the leaf nodes of the attribute set share the same parent node; the leaf nodes of the attribute set are associated with different parent nodes, respectively.

(1) The leaf nodes of AAT share the same attribute of parent node, and there exists the expired data $\{d_1, d_2, d_3\}$. The attribute set of the access policy related to data block d_1 is $\{h_1, h_2, h_5\}$, and the attribute set corresponding to d_2 is $\{h_1, h_2, h_6\}$, while the attribute set of d_3 is $\{h_1, h_2, h_7\}$. Because the leaf nodes of the attributes of $\{d_1, d_2, d_3\}$ share the same parent node, we update the attribute of parent node h_2 to implement the attribute revocation of the expired data $\{d_1, d_2, d_3\}$.

(2) The leaf nodes of AAT do not share the same attribute of parent node, and there exists the expired data $\{d_1, d_4, d_7\}$. The attribute set of the access policy related to data block d_1 is $\{h_1, h_2, h_5\}$, the attribute set of d_4 is $\{h_1, h_3, h_8\}$, and the attribute set of d_7 is $\{h_1, h_4, h_{11}\}$. Because the leaf nodes of the attributes of $\{d_1, d_4, d_7\}$ are associated with different parent node, respectively, the attribute values of the three leaf nodes need to be updated separately to implement the attribute revocation of the expired data $\{d_1, d_4, d_7\}$.

DO reencrypts the symmetric key dk to obtain the ciphertext CT_{dk}' and the digital signature sig′ using new access policy constructed by the updated attribute, uploads CT_{dk}' to TA, and deletes CT_{dk}. Therefore, the original attributes of the DU cannot be satisfied with the access structure of CT_{dk}', so DU is unable to decrypt CT_{dk}' and cannot decrypt the file ciphertext.

Random overwriting implements assured deletion for the expired data in CSP. In order to completely delete cloud data and prevent the privacy leakage caused by key leakage or brute force attack [55], the expired data is randomly overwritten by random data of the same size as the expired data, thereby achieving complete data deletion. In order to reduce the computational cost of the DO to generate random data blocks and the communication overhead of uploading them, a rule transposition algorithm (RTA) is designed based on transposition algorithm [56, 57]. The implementation process of RTA is described as follows. Firstly, the DO generates a random data block f_R and a transposing rule which is called "rule." Rule contains parameters $\{l_1, S, l_2, m\}$, among which, l_1 represents a subblock with l_1-bit sizes divided from f_R, S indicates the initial value of the interval between subblocks transposed, l_2 represents a fixed length value, and m is the number of

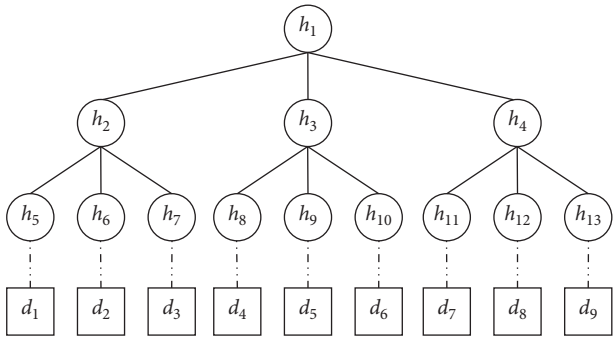

Figure 2: Attribute association tree (AAT).

overwriting data blocks. $(\sigma_1, \sigma_2, \ldots, \sigma_m)$ is the index value of the expired data block. Let f_R and $\{l_1, S, l_2, m\}$ be the input of the RTA, and let data blocks $\{f_{r1}, f_{r2}, \ldots, f_{rm}\}$ equaling to expired data and generated by f_R in sequence be the output.

The process of generation of data block f_{r1} is described as follows. f_R is divided into n subblocks $\{d_{r1}, d_{r2}, \ldots, d_{rn}\}$ with the equal size of l_1 bits. The n subblocks $\{d_{r1}, d_{r2}, \ldots, d_{rn}\}$ are transposed successively to construct a data block f_{r1} according to the value interval $S + l_2$. For example, when $S_0 = 0$ and $l_2 = 1$, let $S_1 = S_0 + l_2$; we exchange (d_i, d_{i+S1}) in terms of subblocks to get $\{d_2, d_1, d_4, d_3, \ldots, d_n, d_{n-1}\}$ and reconstitute the data block f_{r1}. When we transpose f_{r2}, let $S_2 = S_1 + l_2$, and transpose the subblocks of f_R to generate f_{r2} according to the value interval S_2. In a similar way, data block f_{ri} is generated from the subblocks $\{d_{r1}, d_{r2}, \ldots, d_{rn}\}$ according to the value interval $S_i = S_{i-1} + l_2$. Therefore, we get $\text{Transpose}(f_R, \text{rule}) \longrightarrow \{f_{r1}, f_{r2}, \ldots, f_{rm}\}$. The RTA generates the same amount of data blocks with the equal size to the expired ciphertext and obtains an index $(\sigma_1', \sigma_2', \ldots, \sigma_m')$ of the data block by hash algorithm.

DO uploads f_R $(\sigma_1, \sigma_2, \ldots, \sigma_m)$ and rule to CSP to request deleting expired data and save the f_R and rule. After receiving the request, CSP generates a series of random data blocks using RTA triggered by f_R and rule. The expired data blocks with an index of $(\sigma_1, \sigma_2, \ldots, \sigma_m)$ are randomly overwritten multiple times by means of bitstream overwriting using new random data blocks with $(\sigma_1', \sigma_2', \ldots, \sigma_m')$, thereby implementing the secure deletion of the expired ciphertext. The brief workflow is shown in Algorithm 1.

5.3. Data Deletion Verification Phase. The data deletion verification method is used to verify whether the "honest but curious" CSP actually deleted the expired ciphertext. When CSP overwrites the expired ciphertext, there is a risk of forging an index or not generating corresponding random data according to a given *rule* due to the fact that the index $\{\sigma_1', \sigma_2', \ldots, \sigma_m'\}$ is generated by CSP. After the expired ciphertext $\{\text{CT}_1, \text{CT}_2, \ldots, \text{CT}_m\}$ is overwritten by $\{f_{r1}, f_{r2}, \ldots, f_{rm}\}$ successively, DO promptly gets the random data from CSP based on $\{\sigma_1', \sigma_2', \ldots, \sigma_m'\}$; for example, (σ_2', σ_5') corresponds to $\{f_{r2}, f_{r5}\}$; then, DO locally generates the random data blocks $\{f_{r2}', f_{r5}'\}$ by f_R and rule via RTA and uses the hash algorithm to calculate the hash value $(H(f_{r2}'), H(f_{r5}'))$ of $\{f_{r2}', f_{r5}'\}$, respectively. If CSP does not generate data according to the specified transposition rule, $\sigma_2' = H(f_{r2}')$, $\sigma_5' = H(f_{r5}')$, $f_{r2} = f_{r2}'$, and $f_{r5} = f_{r5}'$ cannot be satisfied at the same time. Otherwise, the random data blocks and indexes $\{\sigma_1', \sigma_2', \ldots, \sigma_m'\}$ generated by CSP are real.

Furthermore, for the next step of verification, CSP successively concatenates the index σ_i of the expired ciphertext block, index σ_i' of f_{ri}, and digital signature sig_i' of $\text{CT}_{\text{dk}i}$ to generate deletion verification value v_i, $v_i = (\sigma_i \| \sigma_i' \| \text{sig}_i')$, which is used to generate the root node value v_R as a validator by MHT, as shown in Figure 3. Assuming that the expired ciphertext is $\{\text{CT}_1, \text{CT}_2, \ldots, \text{CT}_8\}$ and the generated verification value is $\{v_1, v_2, \ldots, v_8\}$, CSP makes hash to the verification values and concatenates them separately to generate the verification values of the parent nodes, $H(v_{i,j}) = H(v_i) \| H(v_j)$. Finally, the root node verifier v_R is generated by working up layer by layer. CSP will generate a validator v_R and return to DU.

DO generates local verification value $v_i' = (\sigma_i \| \sigma_i' \| \text{sig}_i')$ by concatenating locally reserved index of $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, digital signature $\{\text{sig}_1, \text{sig}_2, \ldots, \text{sig}_n\}$, and index $(\sigma_1', \sigma_2', \ldots, \sigma_m')$ returned by CSP. Furthermore, DO calculates a local validator v_R' by MHT algorithm and compares it with the returned v_R from CSP. If the two validators are inconsistent, this indicates that the verification value of the leaf node is incorrect, so CSP does not overwrite the partially expired ciphertext. Otherwise, it indicates that CSP completely overwrites the expired ciphertext. This process is shown in Algorithm 2.

6. Security Analysis

In SDVC scheme, the AES-256 algorithm is used to encrypt the plaintext, and then the symmetric key is encrypted by the CP-ABE algorithm. The fine-grained security deletion is implemented by the attribute revocation of CP-ABE, and the overwriting algorithm is used to completely delete the expired data and verify the deletion result. Currently, the AES-256 algorithm is recognized as unable to attack in PPT. Therefore, based on the security model described in Section 4.3, the security of the SDVC is reduced to the attacker Att trying to orchestrate the attribute set to challenge TA for obtaining the symmetric key. An attribute in the CP-ABE is associated with multiple lines in access policy; that is, an attribute may exist on multiple access policies.

Theorem 1. *Under the security model, if the advantage of being successful, that is, the attacker achieves as many queries as possible in all games, is negligible within the PPT, the SDVC scheme is secure.*

Proof. Under the selected access policy model, if there is an attacker Att_1 takes the advantage Adv_{att} to attack the ciphertext in PPT, the security of the SDVC scheme can be guaranteed by the challenger with the advantage Adv_{att}' to solve the deterministic DBDH problem. The proof process consists of the following four phases.

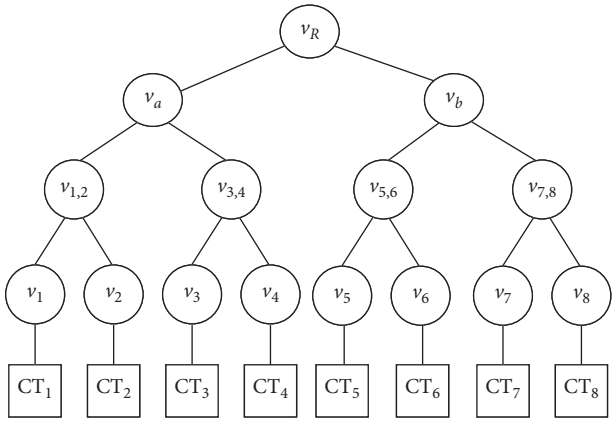

Figure 3: Process of validator generation.

Initialization phase: challenger selects multiplication cycle group $\mathbb{G}_1$ with $\mathbb{G}_2$, chooses a bilinear mapping e: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$, and chooses randomly $\{\alpha, s, \beta_1, \beta_2, \ldots, \beta_q\} \leftarrow \mathbb{Z}_p$ and $\eta \leftarrow \{0, 1\}$. Challenger gets access to strategy $\mathbb{A} = (\Lambda, \rho)$ which Att_1 wants to challenge.

Inquiry phase: Att_1 provides an attribute set A that does not satisfy matrix Λ and asks for the private key. Challenger takes the vector $\overrightarrow{\omega} = (\omega_1, \omega_2, \ldots, \omega_n) \in \mathbb{Z}_p^{n^*}$; when $\omega_1 = -1$ and all i satisfied $\rho(i) \in A$, we have $\overrightarrow{\omega} \cdot \Lambda_i = 0$.

Challenge phase: Att_1 submits two challenge ciphertexts M_0 and M_1 to the challenger, who randomly selects $b \leftarrow \{0, 1\}$ and calculates the ciphertext M_b.

Guess phase: Att_1's guess on b is b'. If $b' = b$, challenger outputs $\eta' = 0$; if $b' \neq b$, challenger outputs $\eta' = 1$. When $\eta = 1$, Att_1 cannot obtains any of b's information, so that $\Pr[\beta' \neq \beta | \eta = 1] = 1/2$. When $b' \neq b$, the challenger guesses $\eta' = 1$, so $\Pr[\eta' \neq \eta | \eta = 1] = 1/2$. When $\eta = 0$, Att_1 obtains ciphertext M_b, due to the advantage of Att_1 being ε, $\Pr[\beta' \neq \beta | \eta = 1] = 1/2 + \varepsilon$. When $\beta' \neq \beta$, the challenger guesses $\eta' = 0$, so $\Pr[\eta' \neq \eta | \eta = 1] = 1/2 + \varepsilon$.

In summary, the challenger's advantage is

$$\frac{1}{2}\Pr[\eta' \neq \eta | \eta = 0] - \frac{1}{2}\Pr[\eta' \neq \eta | \eta = 1] = \frac{1}{2}\left(\frac{1}{2} + \varepsilon\right) - \frac{1}{2} \times \frac{1}{2} = \frac{\varepsilon}{2}. \tag{6}$$

Therefore, the challenger's advantage in the DBDH hypothesis is $(\varepsilon/2)$. The advantage of the attacker is ε. The advantage of attacking success is negligible in PPT. □

7. Theoretical Analysis and Performance Evaluation

Firstly, the theoretical complexity of the SDVC scheme is analyzed, and then the effectiveness of the SDVC scheme is verified by simulation experiments.

7.1. Theoretical Analysis. The theoretical analysis in this section mainly analyzes the computational cost, communication overhead, and storage overhead of each step of the SDVC scheme, as shown in Table 3.

The computational cost will be analyzed from the following three phases. Data encryption and decryption phase: We firstly encrypt the plaintext through AES-256 algorithm and encrypt the symmetric key dk using the CP-ABE algorithm. Furthermore, we calculate the digital signature of the ciphertext of dk by SHA-256. When the key needs to be updated, we should reencrypt the encryption key using CP-ABE and make a new signature of the new ciphertext of symmetric key. When the data needs to be decrypted, we get the symmetric key decrypted by CP-ABE and use this key to decrypt the file ciphertext by AES-256. Secure data deletion phase: The data owner revokes the attributes and reconstructs the access policy to encrypt the symmetric key and request the trusted authority to delete the original key. Data owner generates a random data block f_R, and CSP performs transposition operations m times to generate random data and overwrite the expired ciphertext. Data deletion verification phase: The validator is calculated by MHT from m verification values. Finally, the data owner compares the validator returned by the CSP with the locally generated validator to determine whether the ciphertext in the CSP is actually deleted.

Communication overhead mainly occurs in the DO's acquisition keys, verification, and uploading data. The acquisition key is mainly for the DO to request the key from the TA, who returns the generated public key, and the communication overhead is $m \cdot O(l_{\mathrm{PK}})$. The communication overhead required to upload ciphertext to the CSP is $m \cdot O(l_f)$. The communication of the deletion verification is mainly reflected in the CSP sending the validator to the DO. In this paper, we employ SHA-256 to calculate the validator, so the length of validator is 256 bits and the communication overhead is $O(1)$.

Storage overhead mainly considers the storage of data and keys. Firstly, the DO stores a random block of data f_R and replacement rules. The TA needs to store the ciphertext of m symmetric keys, so the storage overhead is $m \cdot O(l_{\mathrm{dk}} + l_{\mathrm{MK}})$. The CSP only stores the ciphertext, the file is divided into m subfiles of l_f in length, the storage overhead is $m \cdot O(l_f)$.

7.2. Performance Evaluation. We conducted a number of simulation experiments with Ubuntu 14.04 test computer with the following configuration: Intel Core (TM) i5-4539 @3.30 GHz CPU, 8 GB RAM, and 1024 GB hard disk. In order to demonstrate the effectiveness and efficiency of the SDVC scheme, we organize the following simulation experiments, the time cost of encrypting and decrypting different size of data blocks by AES-256 algorithm, the time cost of encrypting symmetric key by CP-ABE setting different number of attributes, the time cost of overwriting data by the rule transposition algorithm during secure data deletion phase, and the time cost of MHT generating the root node.

input: *random data block* f_R *transpose rule*
output: *a series of random data blocks*

(1) **Begin**;
(2) **for** $j \leftarrow 0$ to m **do**
(3) $DivideFile(f_R, l_1) \longrightarrow \{d_{r1}, d_{r2}, \ldots, d_{rn}\}$;
(4) **for** $i \leftarrow 1$ to $\lceil n/2 \rceil$ **do**
(5) $S_i = S_{i-1} + l_2$;
(6) $d_i = d_i \oplus d_{i+Si}$;
(7) $d_{i+S1} = d_{i+Si} \oplus d_i$;
(8) $d_i = d_i \oplus d_{i+Si}$;
(9) **end for**
(10) $Merger(d_{1+Si}, d_{2+Si}, \ldots, d_{n-Si}) \longrightarrow \{f_{rj}\}$;
(11) $\sigma_j' = Hash(f_{rj})$;
(12) **end for**

Algorithm 1: Rule transposition process.

input: random data block and its index, expired ciphertext and its index, sig_i
output: the verification result

(1) **Cloud service provider:**
(2) **for** $i \leftarrow 0$ to m **do**
(3) $v_i = \sigma_i' \| \sigma_i \| \text{sig}_i$;
(4) $Overwri(f_{ri}, \text{CT}_i)$;
(5) **end for**
(6) $v_R = MHT(v_1, v_2, \ldots, v_m)$;
(7) Sent v_R and $(\sigma_1', \sigma_2', \ldots, \sigma_m')$ to data owners;
(8) **Data owner:**
(9) $Transpose(f_R, \text{rule}) \longrightarrow \{f_{ri}'\}$;
(10) $\sigma_i = H(f_{ri}')$;
(11) **if** $\sigma_i = \sigma_i'$ **then**
(12) flag = 1;
(13) **end if**
(14) **if** flag = 1 **then**
(15) **for** $i \leftarrow 0$ to num **do**
(16) $v_i' = \sigma_i' \| \sigma_i \| \text{sig}_i$;
(17) **end for**
(18) $v_R' = MHT(v_1', v_2', \ldots, v_m')$;
(19) **end if**
(20) **if** $v_R' = v_R$ **then**
(21) *return TRUE*;
(22) **else**
(23) *return FALSE*;
(24) **end if**

Algorithm 2: Data deletion and verification process.

From the data set MSR-Cambridge, we select 9 groups of experimental data with different sizes: 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, and 512 MB. We test the running time in the phases of encryption and decryption when encrypting data blocks with different sizes and compare the SDVC scheme with Du et al. [31] scheme and Xue et al. [35] scheme. Each experiment runs 100 times to obtain an average value as the final result. The time cost of encryption (Encrypt) with different sizes of data block is shown in Figure 4, and that of decryption (Decrypt) with different sizes of data block is shown in Figure 5. As the size of the data increases, the time cost of encryption and decryption also increases in a nearly linear relationship. In terms of these two figures, it can be found that, compared with Du et al. [31] scheme and Xue et al. [35] scheme, the time cost for encryption and decryption of our scheme is significantly less than that of the other two schemes. The main reason is that our scheme employs AES-256 to encrypt and decrypt data; hence, it is very efficient. On the other hand, Du et al. [31] scheme uses CP-ABE algorithm to encrypt data, and the ciphertext needs to be associated with the access policy; Xue et al. [35] scheme uses KP-ABE algorithm to encrypt data, and the ciphertext is related to the attribute set. As the data size increases, the number of

Table 3: Complexity analysis of the SDVC.

Computational cost	
Encryption and decryption phase	
Key encryption	$m \cdot O(l_f)$AES + $m \cdot O(dk)$ABE + $m \cdot O(l_{dk})$Hash
Key updating	$O(l_{dk})$ABE + $O(l_{\mathrm{dk}})$Hash
Data decryption	$m \cdot O(l_{dk})$ABE + $m \cdot O(l_f)$AES
Secure data deletion phase	
Request deletion	$O(f_R)$Random
Secure data deletion	$m \cdot O(l_{dk})$ABE + $m \cdot O(l_{fr})$Trans + $m \cdot O(l_{fr})$OW
Proof deletion phase	
Validator generating	$m \cdot O(l_{fr})$Hash + $2O(v)$MHT
Deletion verification	$O(1)$
Communication overhead	
Getting the key	$m \cdot O(l_{\mathrm{PK}})$
Uploading the file	$m \cdot O(z)$
Verifying the result	$O(1)$
Storage overhead	
Data owner	$m \cdot O(l_{dk} + \mathbb{A}) + O(f_R + \mathrm{rule})$
Key management center	$m \cdot O(l_{dk})$
Cloud storage server	$m \cdot O(l_f)$

Note. m represents the number of data blocks, l_f represents the length of the data block, l_{fr} represents random block length, l_{dk} represents key length, AES represents AES-256 encryption function, hash represents SHA-256, ABE represents CP-ABE encryption function, MHT represents Merkle hash tree, OW represents overwrite function, Random represents random number generation function, and Trans represents replacement function.

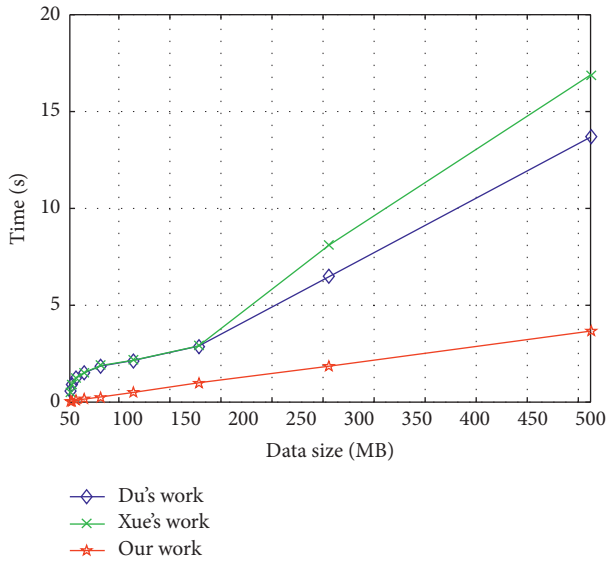

Figure 4: Time cost of encryption with different sizes of data blocks.

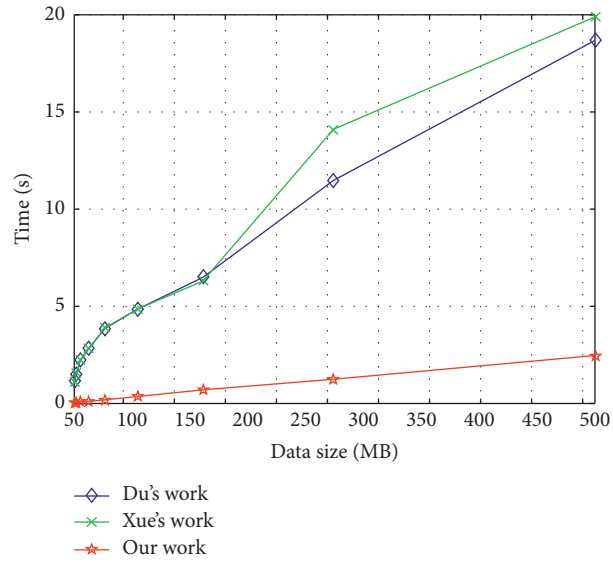

Figure 5: Time cost of decryption with different sizes of data blocks.

attributes associated with data also increases, so the time cost of encryption and decryption increases significantly.

In the SDVC scheme, the CP-ABE algorithm is used to encrypt the symmetric key with 256 bits. Therefore, for the fixed data size, we test the time cost of data encryption and decryption by setting the number of different attributes in the access policy. Literature [31] shows that when the number of attributes reaches 15, it can meet the security requirements of the scheme; accordingly, the number of attributes is selected from 5 to 15, and one attribute is added in turn, as shown in Figure 6. Experimental results show that the encryption time cost increases as the number of attributes increases. During the decryption process, the number of attributes increases, and the time cost of decryption also increases. In the case where only the symmetric key is encrypted, when the number of attributes associated with ciphertext is set to 15, the time costs of encryption and decryption are 226 ms and 255 ms, respectively.

We test the efficiency of overwriting by the time cost of overwriting the ciphertext blocks with different sizes of random data blocks, and we set 8 groups of data blocks with different sizes from 1 MB to 128 MB as experimental data: 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, and 128 MB. The time costs of all-zero overwriting and random

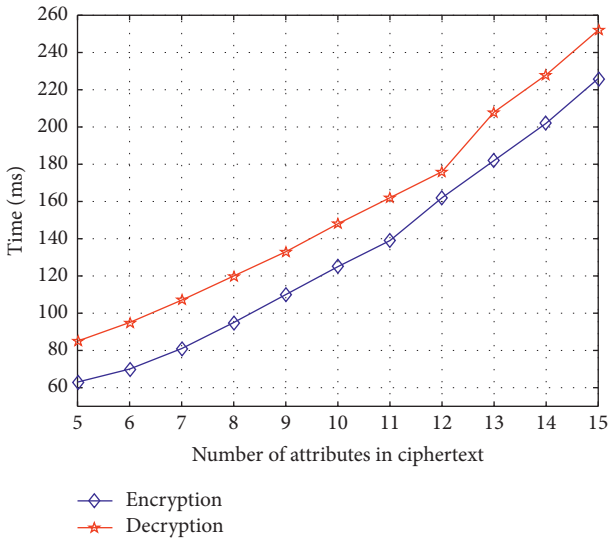

Figure 6: Time cost of encryption and decryption with different numbers of attributes.

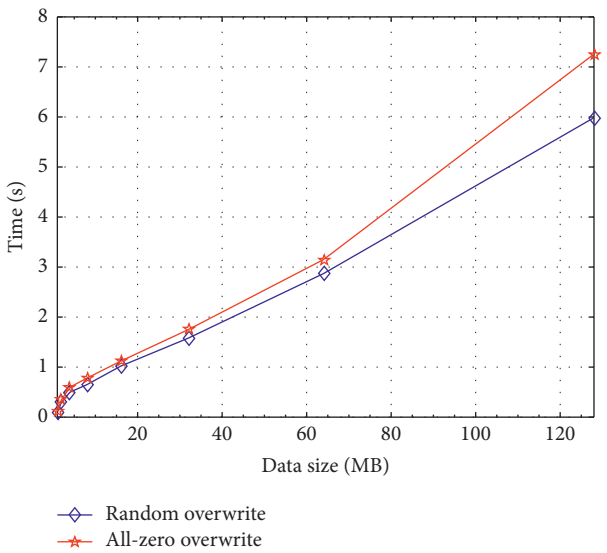

Figure 7: Time cost of overwriting with different sizes of data blocks.

overwriting for data blocks with different sizes are tested, respectively, as shown in Figure 7. As the size of data block increases, the time cost of data block overwriting also increases. The time cost of all-zero overwriting increases with the increasing of the data size, and the time cost increases faster than that of random overwriting time. The random overwriting method selected in our scheme has better efficiency.

The MHT experiment mainly tests the time cost of calculating the root node of MHT with different heights. The file sizes are different in the phase of secure data deletion, which causes the number of data blocks to be different too, and consequently the constructed MHT height is different. We test the running time of data deletion verification with different numbers of data blocks, assuming that each data block is 4 MB in size, and the data owner deletes a maximum file size of 16384 MB (16 GB) each time; accordingly, the maximum leaf nodes of MHT is 256 and their height is 14. Therefore, in our experiment, we set the height of the MHT increasing from 2 to 14 to test the time cost of generating the root node (validator). As can be seen from Figure 8, as the height of the MHT increases, the initial time cost increases slowly, and when the height is 5, the time cost no longer has a linear relationship.

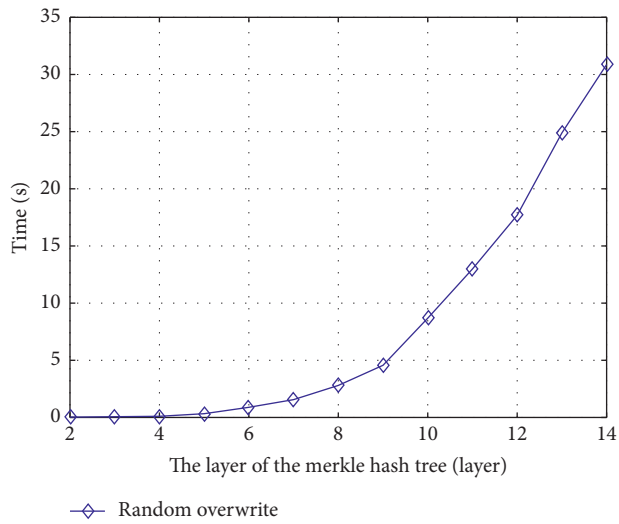

Figure 8: Time cost of generating validator.

8. Conclusion

With the rapid development of mobile Internet and cloud technologies, we proposed a secure data deletion and verification (SDVC) scheme based on CP-ABE to effectively address the issues of unauthorized access, privacy leakage, and verifying deletion result in cloud computing. We constructed an AAT to implement fast attribute revocation and reencryption of keys, employed CP-ABE algorithm to achieve fine-grained and secure data deletion for cloud data, and verified the data deletion result by constructing a random overwriting algorithm and a validator generated by MHT. The security of the SDVC scheme is proved under the standard model. The complexity analysis and ample simulation experiments are carried out, and the results indicate that the SDVC scheme is practical and effective. The future work is to design effective method to implement secure data deletion and verification for mobile devices under 5G environment.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (61872088, U1905211, and 61872090); the Foundation of Science and Technology on Information Assurance Laboratory (KJ-15-108); the Natural Science Foundation of Fujian Province (2019J01276); and the Guizhou Provincial Key Laboratory of Public Big Data Research Fund (2019BDKFJJ004).

References

[1] C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5G wireless communications: a deep learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 227–236, 2020.

[2] J. Cao, Z. Yan, R. Ma, Y. Zhang, Y. Fu, and H. Li, "LSAA: a lightweight and secure access authentication scheme for both UE and mMTC devices in 5G networks," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5329–5344, 2020.

[3] D. Wu, X. Han, Z. Yang et al., "Exploiting transfer learning for emotion recognition under cloud-edge-client collaborations," *IEEE Journal on Selected Areas in Communications*, vol. 99, p. 1, 2020.

[4] Y. Yang, X. Liu, X. Zheng et al., "Efficient traceable authorization search system for secure cloud storage," *IEEE Transactions on Cloud Computing*, vol. 8, no. 3, pp. 819–832, 2020.

[5] J. Xiong, Y. Zhang, S. Tang et al., "Secure encrypted data with authorized deduplication in cloud," *IEEE Access*, vol. 7, pp. 75090–75104, 2019.

[6] Z. Ying, W. Jang, S. Cao et al., "A lightweight cloud sharing PHR system with access policy updating," *IEEE Access*, vol. 6, pp. 64611–64621, 2018.

[7] Y. Miao, R. Deng, X. Liu et al., "Multi-authority attribute-based keyword search over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 99, p. 1, 2019.

[8] J. Cao, M. Ma, H. Li et al., "A survey on security aspects for 3GPP 5G networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 170–195, 2020.

[9] J. Xiong, F. Li, J. Ma, X. Liu, Z. Yao, and P. S. Chen, "A full lifecycle privacy protection scheme for sensitive data in cloud computing," *Peer-to-Peer Networking and Applications*, vol. 8, no. 6, pp. 1025–1037, 2015.

[10] H. Wang, D. Peng, W. Wang, H. Sharif, H.-h. Chen, and A. Khoynezhad, "Resource-aware secure ECG healthcare monitoring through body sensor networks," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 12–19, 2010.

[11] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.

[12] Y. Miao, X. Liu, K.-K. R. Choo et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2019.

[13] J. Xiong, X. Liu, Z. Yao et al., "A secure data self-destructing scheme in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 448–458, 2014.

[14] K. Ramokapane, A. Rashid, and J. Such, "Assured deletion in the cloud: requirements, challenges and future directions," in *Proceedings of the 2016 ACM on Cloud Computing Security Workshop*, Vienna, Austria, 2016.

[15] J. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in IOT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2020.

[16] S. Xu, G. Yang, Y. Mu, and X. Liu, "A secure IOT cloud storage system with fine-grained access control and decryption key exposure resistance," *Future Generation Computer Systems*, vol. 97, pp. 284–294, 2019.

[17] J. Xiong, L. Chen, M. Z. A. Bhuiyan et al., "A secure data deletion scheme for iot devices through key derivation encryption and data analysis," *Future Generation Computer Systems*, vol. 111, no. 10, pp. 741–753, 2020.

[18] Y. Liu, Y. Zhou, R. Lan et al., "Blockchain-based verification scheme for deletion operation in cloud," *Journal of Computer Research and Development*, vol. 55, no. 10, pp. 2199–2207, 2018.

[19] J. Xiong, F. Li, Y. Wang et al., "Research progress on cloud data assured deletion based on cryptography," *Journal of Communications*, vol. 37, no. 8, pp. 168–184, 2016.

[20] Y. Tian, Q. Li, J. Hu, and H. Lin, "Secure limitation analysis of public-key cryptography for smart card settings," *World Wide Web*, vol. 23, no. 2, pp. 1423–1440, 2020.

[21] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: efficient policy-hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130–2145, 2018.

[22] S. Xu, Y. Li, R. Deng et al., "Lightweight and expressive fine-grained access control for healthcare internet-of-things," *IEEE Transactions on Cloud Computing*, p. 1, 2019.

[23] J. Li, Y. Zhang, J. Ning et al., "Attribute based encryption with privacy protection and accountability for cloud IOT," *IEEE Transactions on Cloud Computing*, p. 1, 2019.

[24] Q. Li, J. Ma, R. Li, X. Liu, J. Xiong, and D. Chen, "Secure, efficient and revocable multi-authority access control system in cloud storage," *Computers & Security*, vol. 59, pp. 45–59, 2016.

[25] Y. Zhang, R. H. Deng, D. Zheng, J. Li, P. Wu, and J. Cao, "Efficient and robust certificateless signature for data crowdsensing in cloud-assisted industrial IOT," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 5099–5108, 2019.

[26] Q. Jiang, N. Zhang, J. Ni et al., "Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 99, p. 1, 2020.

[27] J. Xiong, Y. Zhang, L. Lin et al., "MS-POSW: a multi-server aided proof of shared ownership scheme for secure deduplication in cloud," *Concurrency and Computation: Practice and Experience*, vol. 32, p. e4252, 2020.

[28] Y. He, J. Ni, B. Niu, F. Li, and X. Shen, "Privbus: a privacy-enhanced crowdsourced bus service via fog computing," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 156–168, 2020.

[29] J. Li, H. Yan, and Y. Zhang, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 478–487, 2017.

[30] M. Paul and A. Saxena, "Proof of erasability for ensuring comprehensive data deletion in cloud computing," in *Proceedings of the 2010 International Conference on Network Security and Applications*, Chennai, India, 2010.

[31] R. Du, P. Shi, and X. He, "Cloud data deterministic deletion scheme based on overwrite verification," *Journal of Communications*, vol. 40, no. 1, pp. 130–140, 2019.

[32] F. Hao, D. Clarke, and F. Zorzo, "Deleting secret data with public verifiability," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, pp. 617–629, 2015.

[33] Y. Liu, S. Xiao, H. Wang et al., "New provable data transfer from provable data possession and deletion for secure cloud storage," *International Journal of Distributed Sensor Networks*, vol. 15, no. 4, pp. 1–12, 2019.

[34] C. Yang, X. Chen, and Y. Xiang, "Blockchain-based publicly verifiable data deletion scheme for cloud storage," *Journal of Network and Computer Applications*, vol. 103, pp. 185–193, 2018.

[35] L. Xue, Y. Yu, Y. Li, M. H. Au, X. Du, and B. Yang, "Efficient attribute-based encryption with attribute revocation for assured data deletion," *Information Sciences*, vol. 479, pp. 640–650, 2019.

[36] Q. Li, J. Ma, R. Li, J. Xiong, and X. Liu, "Provably secure unbounded multi-authority ciphertext-policy attribute-based encryption," *Security and Communication Networks*, vol. 8, no. 18, pp. 4098–4109, 2015.

[37] H. Wang, M. Hempel, D. Peng et al., "Index-based selective audio encryption for wireless multimedia sensor networks," *IEEE Transactions on Multimedia*, vol. 12, no. 3, pp. 215–223, 2010.

[38] M. Miao, J. Wang, J. Ma, and W. Susilo, "Publicly verifiable databases with efficient insertion/deletion operations," *Journal of Computer and System Sciences*, vol. 86, pp. 49–58, 2017.

[39] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Transactions on Cloud Computing*, p. 1, 2019.

[40] Y. Tian, Z. Wang, J. Xiong et al., "A blockchain-based secure key management scheme with trustworthiness in DWSNS," *IEEE Transactions on Industrial Informatics (Early Access)*, pp. 1–11, 2020.

[41] R. C. Merkle, "Protocols for public key cryptosystems," in *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1980.

[42] Y. Yu, L. Xue, Y. Li, X. Du, M. Guizani, and B. Yang, "Assured data deletion with fine-grained access control for fog-based industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4538–4547, 2018.

[43] S. Jia, L. Xia, B. Chen et al., "NFPS: adding undetectable secure deletion to flash translation layer," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, Xi'an, China, 2016.

[44] M. Wang, J. Xiong, R. Ma et al., "A novel data secure deletion scheme for mobile devices," in *Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN)*, Hangzhou, China, 2018.

[45] F. Li, H. Li, B. Niu, and J. Chen, "Privacy computing: concept, computing framework, and future development trends," *Engineering*, vol. 5, no. 6, pp. 1179–1192, 2019.

[46] X. Li, J. Li, S. Yiu et al., "Privacy-preserving edge-assisted image retrieval and classification in iot," *Frontiers of Computer Science*, vol. 13, no. 4, pp. 1136–1147, 2019.

[47] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[48] J. Ma, Y. Guo, J. Ma et al., "A hierarchical access control scheme for perceptual layer of iot," *Journal of Computer Research and Development*, vol. 50, no. 6, pp. 1267–1275, 2013.

[49] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the 2005 Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Aarhus, Denmark, 2005.

[50] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07)*, Berkeley, CA, USA, 2007.

[51] Q. Li, H. Zhu, J. Xiong et al., "Fine-grained multi-authority access control in IOT-enabled mhealth," *Annals of Telecommunications*, vol. 74, no. 7-8, pp. 389–400, 2019.

[52] K. Zhang, H. Li, J. Ma et al., "Efficient large-universe multi-authority ciphertext-policy attribute-based encryption with white-box traceability," *Science China Information Sciences*, vol. 61, no. 3, p. 32102, 2018.

[53] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *Proceedings of the 2011 International Workshop on Public Key Cryptography*, Taormina, Italy, 2011.

[54] M. Ali, M. Sadeghi, and X. Liu, "Lightweight revocable hierarchical attribute-based encryption for internet of things," *IEEE Access*, vol. 8, pp. 23951–23964, 2020.

[55] S. Ma, E. Bertino, R. Deng et al., "Finding flaws from password authentication code in android apps," in *Proceedings of the ESORICS*, Luxembourg, Luxembourg, 2019.

[56] M. Van Dijk, A. Juels, A. Oprea et al., "Hourglass schemes: how to prove that cloud files are encrypted," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, Raleigh, NC, USA, 2012.

[57] Y. Luo, M. Xu, S. Fu et al., "Enabling assured deletion in the cloud storage by overwriting," in *Proceedings of the 4th ACM International Workshop on Security in Cloud Computing*, Xi'an, China, 2016.