IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 27, 2020, accepted January 7, 2021, date of publication January 12, 2021, date of current version January 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051061

Tenant-Led Ciphertext Information Flow Control for Cloud Virtual Machines

ZHAO ZHANG[1], ZHI YANG[1], XUEHUI DU[1], WENFA LI[2], XINGYUAN CHEN[1,3], AND LEI SUN[1]

[1]Zhengzhou Information Science and Technology Institute, PLA Information Engineering University, Zhengzhou 450001, China
[2]School of Information, Beijing Union University, Beijing 100101, China
[3]State Key Laboratory of Cryptology, Beijing 100084, China

Corresponding authors: Zhi Yang (zynoah@163.com) and Xingyuan Chen (chxy302@vip.sina.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61972040, in part by the National Key Research and Development Program of China under Grant 2016YFB050190104, and in part by the Premium Funding Project for Academic Human Resources Development in Beijing Union University under Grant BPHR2020AZ03.

ABSTRACT When users upload their private data to the cloud, they lose control of the data stored in the cloud server. If the cloud system cannot provide an effective security mechanism to protect the data, the consequent data leakage issue will hinder the development of cloud computing. Conventional access control and encryption technologies cannot effectively control the propagation of tenant private data in the system. The mandatory one-way information flow control model is limited by the complexity of the cloud environment, and it is difficult to effectively protect private data stored in the cloud. To solve the above problems, this article proposes a tenant-led ciphertext information flow control method for cloud virtual machines. Through the design of a decentralized information flow control security policy, a secret-domain key management scheme, and a multi-ID-based threshold encryption scheme, the information flow control strategies of taint infection, secret-level reduction, and ability propagation are realized in a ciphertext form, which can effectively prevent malicious users inside and outside the system from illegally reading private data. The feasibility of this method is verified by a security proof and an experiment.

INDEX TERMS Cloud computing, information flow control, information flow encryption, data protection.

I. INTRODUCTION

With the widespread adoption of cloud computing, cloud security has emerged as a critical issue. When users store their private data in a cloud server, they lose control of their data. To ensure privacy, it is necessary to design security mechanisms and architectures to effectively protect the private data stored in the cloud [1]–[3].

In recent years, to counter the security threat in cloud computing environments, major cloud providers have adopted various conventional security protection mechanisms based on the characteristics of their cloud platforms. Identity authentication mechanisms, role-based access control models, and attribute-based access control models have been widely used and can effectively implement access control to the data stored in the cloud [4]. However, conventional access control mechanisms have a security problem, i.e., data cannot be subsequently protected after user access. When users are restricted by access control policies, they gain the ability to manipulate data. Users can copy or back up the data and then leak them without permission. In this scenario, the system cannot ensure data security.

Employing an information flow control technology can solve the above problems. The core idea of an information flow control mechanism is to assign tags to the data; the tags spread along with the data in the entire system and restrict the flow of data between programs through strategies to protect sensitive data from being read and destroyed by unauthorized users. The mandatory one-way information flow control model is often counterproductive in complex cloud computing environments because of the aggressive strategy employed, and users cannot formulate the strategy for their private data. In comparison, a decentralized information flow control model enables scattered subjects in a system to assign tags and secret-level reduction ability, and users can control their private data more flexibly. However, this model cannot supervise the data stored in the cloud servers outside the system; once the data leave the system, they cannot

The associate editor coordinating the review of this manuscript and approving it for publication was Shagufta Henna.

be protected. Untrusted cloud storage servers or attackers outside the system can leak user data.

To improve the data protection ability of the decentralized information flow model in an untrusted cloud environment, this study combines an encryption technology with decentralized information flow control. Conventional access control encryption implementation strategies heavily rely on users. Malicious users in the system or users controlled by hackers outside the system can arbitrarily reveal their secrets. The information flow control technology ensures the security of the information flowing in the system by formulating an appropriate security strategy. If the security strategy of the information flow control model is realized using an encryption algorithm with a control mechanism, user data can be better protected.

The contributions of our study are as follows: (1) We designed a tenant-led information flow control framework and policy rules. From the perspective of a full-cycle protection of the transmission, storage, and use of private data of cloud tenants, we realized data owner-led cloud private data access protection, where the readers of private data cannot distribute the data copies to other users who are not authorized by the data owner. (2) We designed a ciphertext information flow control mechanism, which is a first where decentralized information flow control and encryption technology are combined for a full-cycle protection of the private data of cloud tenants. The proposed mechanism can effectively realize tenant-led information flow control. A multi-ID-based threshold encryption scheme is designed to realize the security strategies of taint infection, secret-level reduction, and ability transmission in a cipher text form in the information flow control system. Through a security proof and experimental analysis, we confirmed that the scheme proposed in this article is secure and efficient.

The rest of this article is organized as follows: Section II discusses related work. Section III introduces the decentralized ciphertext information flow control system. Section IV presents a detailed design of the system. In Section V, formal and experimental verifications are presented. Finally, Section VI summarizes our study.

II. RELATED WORK

A. ACCESS CONTROL AND INFORMATION FLOW CONTROL

To ensure the security of private data in the user privacy domain, it is mainly implemented through access control. Access control helps prevent private data theft. Only authorized users can transfer and provide data [5]. Current information systems mainly use autonomous access control mandatory access control, role-based access control and attribute-based access control [1]. The traditional mandatory access control data flow is one-way, the flexibility is poor, the strategy tends to be too strict or loose, the practicability is poor, and it is difficult to achieve the expected effect. The autonomous access control model cannot ensure the security of the data once accessed. Therefore, studying the information flow control technology is important. The information flow control method was originally the lattice model proposed by Denning in 1976 [6]. Conventional information flow control research mainly focused on the information flow tracking control method. This method involves system-wide simulation, dynamic binary instrumentation, and compiler-based implementation. Based on the implementation level of the computer architecture, the information flow control mechanism is implemented in hardware layer, operating system layer, virtual machine layer, compiler layer, database, and network layer [7]. However, most studies dealt with single-machine environment objects.

With the popularity of cloud computing, studies have been conducted [8]–[15] on the combination of information flow control technology and cloud computing environment. In [8], a loadable Linux kernel module FlowK was designed to provide a mechanism to enforce information flow policies at runtime. At the application layer, applications with an information flow tracking library were provided to track the application data; at the operating system layer, the operating system kernel was modified, system calls that generate information flows were intercepted, and information flow control was performed on the basis of information flow control strategies. The authors of [9] designed a middleware that supports information flow control and implemented information flow control between applications and virtual machines. The management interface of a VM layer was used to isolate conflicting virtual domains, thereby realizing information flow control. The authors of [10] proposed a CamFlow model as a prototype implementation of a data-centric security model in PaaS cloud. CamFlow enforces the information flow control strategy at the kernel level and executes the information flow control strategy between machines while exchanging messages. Later, Khurshid *et al.* [11] improved the migration process of key protected data to cloud data on the basis of the CamFlow model. The authors of [12] proposed a two-layer information flow control model for the cloud environment, designed and implemented a prototype system, that can provide cloud tenants with information flow tracking and control as a service, and proposed a protection mechanism for common system attacks such as stack overflow and buffer overflow. To overcome the loss of control of sensitive data manipulated by applications in the SaaS cloud environment and the lack of security enforcement measures, the authors of [13] proposed an information flow as a service (IFCaaS) framework, which uses information flow control as a security verification method for SaaS layer applications. In [14], by analyzing the behavior of entities and linking them with trust levels and security classes, security attributes were dynamically formulated, thereby enhancing information flow control to ensure the security attributes of tenants and organizations. Due to the sharing of virtual machine resources, it is difficult to identify the information flow boundary of tenant virtual machines. The authors of [15] proposed a dynamic control method for tenant sensitive information flow based on virtual boundary recognition, which combines the concepts

of centralized and decentralized information flow control. Thus, a dynamic control method was established for sensitive information flow.

Most existing research on information flow control has been conducted to make the information flow control mechanism to adapt to the cloud environment and facilitate the control of information flow in the cloud. It can be seen as an extension of the information flow model in the conventional single-machine mode. In the cloud environment, cloud data storage breaks the conventional form of protecting private data by itself. In the single-machine mode, the information flow model can directly supervise the files. The use of an encryption technology has a greater impact on the performance and limited security improvement. In a cloud environment, data are not in a user's control after being stored in the cloud, and an encryption technology can improve the security of private data in an open cloud environment. In conventional information flow control tenants cannot manage their own information flow policies independently; moreover, it requires a dynamic adjustment of permissions and propagation of capabilities in the cloud. Therefore, it is necessary to study a ciphertext information flow control mechanism based on a tenant-led strategy.

Existing encrypted access control schemes are mostly used to solve the cloud storage access problem by controlling data access through a key mechanism. The basic idea is that a subject encrypts the data, and only the object that can be decrypted can access the data [2]. In the following, we mainly introduce research on existing encryption technologies for cloud data confidentiality protection from three aspects: Conventional public key encryption, access control encryption, and attribute-based encryption [16].

B. PUBLIC KEY ENCRYPTION

The conventional public key encryption system uses a public key to encrypt data, and only those with a private key can decrypt the data. Belguith *et al.* [17] proposed a method where symmetric encryption is used to encrypt files and asymmetric encryption to distribute the keys. This scheme has certain drawbacks, such as key management problems and difficulty in fine-grained access to files, and it is not flexible enough to be extended. To solve the public key authentication problem of the conventional public key encryption system, Shamir [18] proposed an identity-based encryption (IBE) algorithm. In this approach, the data owner uses his/her own unique ID to generate a public key to encrypt the data, and the private key is decrypted. Although the problem of public key authentication is solved, a fine-grained access control to the files cannot be realized, and the scalability is poor. When the conventional public key encryption system shares data in the cloud, there are many problems that make this method inapplicable to a cloud environment.

C. ACCESS CONTROL ENCRYPTION

Access control encryption is a technology that combines encryption and access control. Damgård *et al.* were the first to propose an access control encryption (ACE) [19]. This helped strengthen the information flow by providing different users with different sender and receiver key sets. In this approach, n types of senders and receivers are defined in the form of an access control matrix P, where P[i][j] = 1 implies that sending is allowed, and 0 implies that sending is not allowed. First, a 1-ACE scheme is constructed for a single identity, and then an ACE scheme plan with multiple identities is constructed using the repetition of 1-ACE. In the ACE of the multi-identity scheme, data should be encrypted n times with different keys. To reduce the ACE complexity of the multi-identity scheme, Garg *et al.* [20] proposed a scheme based on indistinguishability obfuscation (IO). Tan *et al.* [21] divided the general ACE scheme into a digital signature scheme, a predicate encryption scheme, and a function encryption scheme that supports random functions. In addition, they proposed a scheme that supports dynamic and more fine-grained access control strategies. Although the access control encryption technology solves the problems of the conventional public key encryption system, such as the difficulty in fine-grained access to files and insufficient flexibility, it is essentially the same as the conventional access control technology, and it remains challenging to solve the problem of data transmission after control.

D. ATTRIBUTE-BASED ENCRYPTION

Attribute-based encryption (ABE) is a more effective access control encryption method. In [22], attribute-based encryption was first proposed as an encryption technique based on fuzzy identity. The basic idea is that ciphertext and key are associated with a set of attributes. When the attributes of a user's private key and ciphertext match, only then can the ciphertext be solved. It is a technology that combines encryption and access control, so it has been widely extensive research in deal with cloud data protection issues. In [23], CP-ABE was proposed. In this solution, the access structure responsible for specifying the encryption strategy is associated with a ciphertext. A private key is created for the user based on his/her attributes. If the attributes in the private key satisfy the access structure in the ciphertext, the user can decrypt the ciphertext. To protect user privacy and reduce trust and dependency on the center, the authors of [24] proposed a privacy-protected distributed CP-ABE. In this solution, multiple independent working authorization agencies are set up without any collaboration. Recently, Han *et al.* [25] combined ACE with ABE. To implement IFC and realize flexible strategies, they proposed a property-based, fine-grained, efficient information flow control scheme. However, the author only realized a conventional information flow strategy with no read-up rule or write-down rule through an attribute-based encryption algorithm. To deal with the multiple sharing of private ciphertext, the authors of [26] combined the advantages of proxy re-encryption and anonymity technology to realize a secure and conditional sharing of ciphertext multiple times. To solve the problem of cloud data privacy leakage, a new attribute-based original

ciphertext strategy signcryption (CP-ABSC) was proposed in [27]. CP-ABSC combines the advantages of digital signature and encryption to ensure confidentiality, authenticity, unforgeability, anonymity, and resistance to collusion. When attribute-based encryption is faced with the revoking of user access rights, the algorithm requires the user to re-encrypt the data. Since the access control structure of the ABE is based on a secret sharing method, the cost of re-encryption is relatively high, and effectively supporting the dynamic policies has become a major problem in this method. ABE can only solve the problem of data storage access control. Only users who meet the conditions can decrypt the data. However, after decryption, the subsequent propagation of the data cannot be controlled. Malicious internal users can arbitrarily leak their private data.

E. DIFFERENTIAL PRIVACY DATA PROTECTION

In addition to research on cloud data protection, access control, and encryption technologies, differential privacy data protection [28] is a new privacy definition proposed by Dwork to prevent private data leakage from statistical databases. An attacker cannot obtain accurate individual information by observing the calculation results. Since its proposal, it has gradually become a research hotspot in the field of private data protection. Roth *et al.* proposed an interactive differential privacy protection mechanism for data publishing, namely the median mechanism [29]. The differential privacy protection technology mainly solves the problem of private data leakage in data computing; however, it remains challenging to solve the problem of data leakage during communication and storage in the cloud environment. The implicit data segmentation mechanism involves segmenting private data into different pieces and storing them in different locations. This makes it difficult for hackers to reconstruct the original data. Parakh and Kak [30] studied an implicit security mechanism based on solving polynomials in a finite field to achieve file segmentation. This mechanism avoids complex encryption operations to a certain extent while ensuring the data confidentiality; however, it cannot be easily applied to the cloud environment because the data storage location in the cloud is determined by the server to be transparent to users. Current research on data protection shows that access control and encryption technologies are more suitable for cloud environments. Other privacy protection technologies cannot fully protect the private data stored in the cloud throughout their lifecycle.

Overall, existing encryption technologies, whether it is access control encryption or attribute-based encryption, are essentially a combination of conventional access control and encryption technology. Although these approaches can perform storage and access control for private data, they cannot solve the problem of data transmission control. Once the access policy of private data is passed, the users of private data can operate the data at will. Moreover, despite being able to solve the problem of cloud data transmission control, existing information flow control technologies do not involve outsourcing data storage. The protection of private data stored in the cloud should involve every stage of the data life cycle. It is necessary to solve both the access control problem and the storage security problem, and prevent internal information leakage and counter attacks from outside the system on private data stored in untrusted cloud storage servers. To the best of our knowledge, there is no existing scheme that uses encryption technology to implement a dynamic strategy for information flow control to provide a comprehensive protection for cloud private data.

III. DECENTRALIZED CIPHERTEXT INFORMATION FLOW CONTROL FRAMEWORK AND STRATEGY

A. SYSTEM DESIGN

The core design idea of the ciphertext information flow control system is to implement a decentralized information flow control security strategy through a secret-domain key management mechanism and multi-ID-based threshold encryption scheme to ensure information flow between tenants in a cloud environment, and the private data stored in the cloud storage server are protected in accordance with the tenants. To realize the decentralized information flow rules, this study designed a multi-ID-based threshold encryption scheme. The file uploaded to a cloud server is not in control of a user's virtual machine, so the file must be encrypted on the basis of the information flow strategy designed in this scheme. The tenant with this file's security label can decrypt independently using the secret-domain key corresponding to this file's security label. Thus, ciphertext-based information flow control is performed on private data unsupervised by the user's virtual machine, thereby protecting the private data.

The system is mainly composed of four types of entities, as shown in Figure 1. (1) The central authority (CA), which is the label management center and key management center, is responsible for label management and key distribution in the system. (2) The encryption proxy (EP), which exists in the virtual machine hypervisor, is responsible for executing security policies and performing storage encryption proxy operations for legitimate cloud tenants. (3) Cloud server CS, which is a cloud server with a large amount of storage space and computing power. In this study, we mainly refer to the cloud storage server, which stores user data and provides users with data access services. (4) Cloud tenant virtual machines: all the tenant virtual machines interact with the CS through EP to dynamically access (read, write, modify, etc.) the data, and EP enforces encryption based on the security policies. All the tenants can apply to the CA to create a new secret-domain label and grant abilities to the virtual machines of trusted tenants.

In the system, it is assumed that the CA and EP are credible, i.e., the CA will faithfully manage the security labels and keys of all the tenants in the system, and EP will faithfully implement the ciphertext information flow security strategy. System threats come from accidental or malicious information leakage by the tenants, and possible Trojan horse viruses

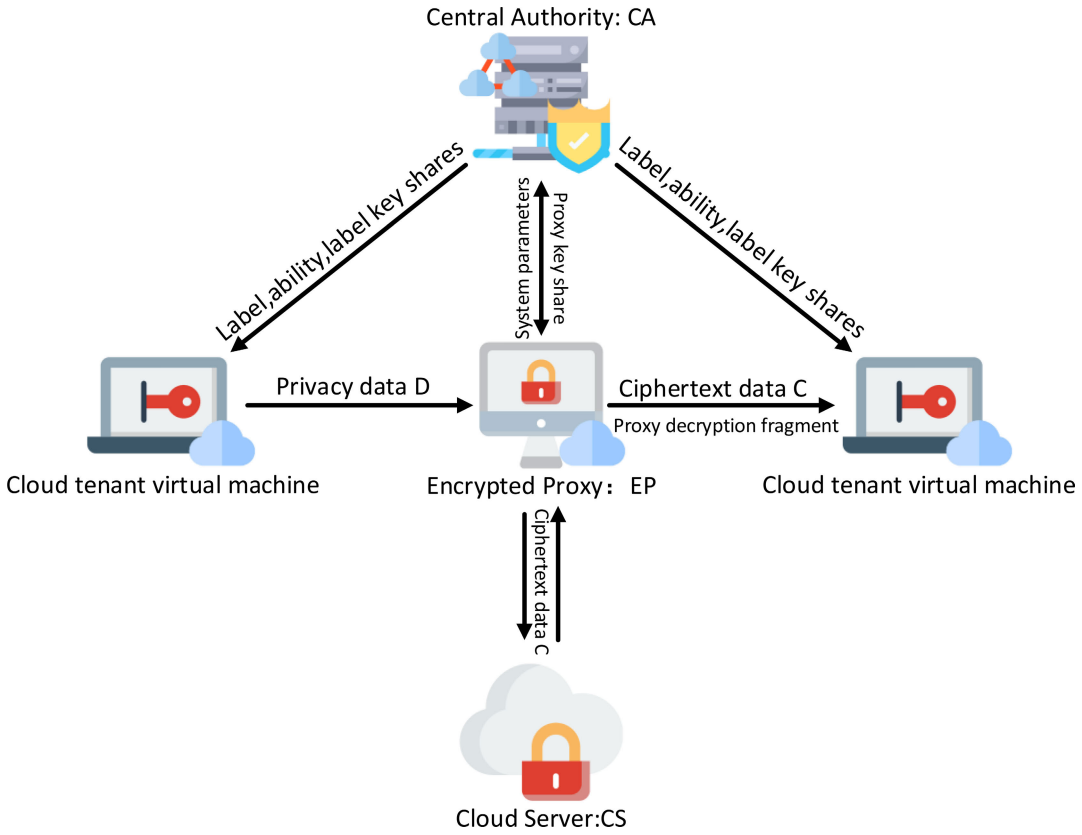

FIGURE 1. Tenant-led ciphertext information flow control system framework.

in tenant virtual machines and CS can steal the private data of other tenants. Through our method, it is possible for tenants to formulate the communication strategy of their data. The strategy established by the tenant is realized through the tenant's key distribution, and the information flow strategy established by it will not affect other tenants' data security.

Before introducing the security strategy, we first give a system example for a better understanding of the system composition.

1) SYSTEM EXAMPLE

As shown in Figure 2, there are two tenant virtual machines, namely Alice and Bob, in the system. Alice's security label L_a is $\{(t_a, p), (t_c, s)\}$ and ability set A_a is $\{t_a^{\pm}, t_{c-s}^{-}\}$. Bob's security label L_b is $\{(t_b, p)\}$ and ability set A_b is $\{t_{a-p}^{+}, t_b^{\pm}\}$. Now, there are two files A_1 and A_2 in Alice's secret domain a. Alice stores the A_1 file in the cloud storage system. After deleting the C secret-domain label, the A_2 file is sent to Bob.

Alice writes file A_1 to the cloud storage server, and Alice's encryption proxy in the virtual machine hypervisor uses the current label $\{(t_a, p), (t_c, s)\}$ to call the encryption algorithm for encryption, and obtains the ciphertext A_1 file. Bob wants to read the A_1 file in the cloud storage system. Because he does not meet the network file reading rules, he is rejected by the virtual machine hypervisor. If Bob is a malicious tenant or if a malicious user outside the system illegally obtains file A_1, because the file is encrypted, the file stored in the cloud server is still protected by the ciphertext information flow control system.

In the virtual machine hypervisor, the file A_2 sent by the information flow control module from Alice to Bob meets the information flow security rules, and Bob can receive the A_2 file, but Bob's label is infected $\{(t_a, p), (t_b, p)\}$. When Bob wants to write file B in the secret domain of his private data to the cloud storage server, because he does not have the ability t_{a-p}^{-}, the written data are encrypted by EP using the label $\{(t_a, p), (t_b, p)\}$ call encryption algorithm.

B. INFORMATION FLOW SECURITY STRATEGY

1) SYSTEM ELEMENTS

a: ENTITY

The entity set O includes the set of all active objects in the system and the set of all protected information in the system. In this study, the entity set includes all the cloud tenant virtual machines and private data files.

b: TAG

The label t_k represents a private data protection secret domain and is used to identify a type of private data. The secret-domain set S represents the set of secret-domain label symbols to which the subject belongs, e.g., $t_A \in S$, where t_A represents the private data of the tenant Alice, and the subject who can read the data has at least tag t_A. To protect private

data more effectively, this study introduces a secret-level label set SL; SC_k represents the secret level of the subject in the secret domain; based on the privacy level from low to high, it is divided into open, secret, confidential, and top secret. SC_b means that the tenant Alice sets the secret level for file B in a certain private data protection secret domain, and the subject who can read the file must have the authority of the secret level or above.

c: SECURITY LABEL

The security label represents the set of all tags contained in the subject $L \subseteq S \times SL$. The set of labels for each subject $oL_o = \{l_a, l_b, l_c \ldots l_k\}$. l_k is the two-tuple (t_k, SC_k), meaning that the subject o has SC_k secret levels in the t_k secret domain.

A private data file is an entity with a permanent label. The security label is persisted through file extension attributes. When the file is created, the security label is assigned to the private data file based on the information flow rules. When a tenant virtual machine modifies its security label based on its abilities, it may no longer be able to read or write the files it creates. A tenant virtual machine that does not change its security flag will never lose access to the files that it could access before.

d: ABILITY

Each subject has an ability set A to indicate the ability to adjust its own label. $t^+_{k-SC_k}$ means it can add label (t_k, SC_k) to its security label. $t^-_{k-SC_k}$ means that it has the ability to delete label (t_k, SC_k) from the security label, $t^{\pm}_k$ means that it fully controls the data in the secret domain, has the ability to grant t^+_{k-SL} or t^-_{k-SL} to the subject it trusts, and can revoke the label (t_k, SC_k) and ability-level labels t^+_{k-SL} and t^-_{k-SL} of a subject.

For subject O and label t_k, $t^+_{k-SC_k} \in A_O$ and $t^-_{k-SC_k} \notin A_O$ indicate that the subject O can raise its own security label to $L_O \cup (t_k, SC_k)$, so that O can read the data of the secret level below SC_k in the secret domain t_k, but cannot disclose the information to other entities O ($(t_k, SC_k) \notin L_O$). $t^+_{k-SC_k} \in A_O$ and $t^-_{k-SC_k} \in A_O$ indicate that O can delete (t_k, SC_k) from its own security label, and then the data of the secret level SC_k or less in the propagation secret domain t_k can be obtained.

The subject's ability is initially taken from the tenant's own attributes, set and managed by the authorized administrator, and is obtained and loaded from the CA when the virtual machine is created. After the virtual machine starts running, it can be changed when the access permission of a certain secret domain can be granted by other subjects.

2) SAFETY RULES

a: INFORMATION FLOW RULES BETWEEN VIRTUAL MACHINES

The tenant virtual machine a sends the data file d to the tenant virtual machine b. The flow of information from entity a to entity b is secure if and only if the following rules are met:

$$a \to b if L_a - A^-_a \subseteq L_b + A^+_b$$

When the tenant virtual machine b receives data, the security mark of b is infected, and the following changes occur:

$$L'_b \leftarrow L_b \cup (L_a - A^-_a)$$

A safe information flow must ensure that the low-security-level information flows to the high-security-level in the secret domain while considering the sending ability of the information outflow entity a and the receiving ability of the information inflow entity b, and the implementation of label taint propagation.

b: CLOUD STORAGE OPERATION INFORMATION FLOW RULES

1. Network file reading

The tenant virtual machine a reads the data file d from the cloud storage server. The information flows from the entity d to entity a, represented as $d \xrightarrow{R} a$, and must meet the network file reading rules:

$$d \xrightarrow{R} a if L_d \subseteq L_a + A^+_a then L'_a \leftarrow L_a \cup L_d$$

To read data files in the cloud storage server, the confidentiality level of the entity a must be higher than that of data d to prevent the inflow of private information from the high-security level to the low-security level. After tenant virtual machine a reads data d, the security label of a is infected and it changes.

2. Network file writing The tenant virtual machine a writes the data file d to the cloud storage server, which is a process in which the subject a creates the subject d. The network file creation operation is represented as $a \xrightarrow{W} \mathrm{d}$. The following network file writing rules must be met:

$$a \xrightarrow{W} \mathrm{d}\ then L_d := L_a$$

The security labels of the data files written in the cloud storage server are inherited from the security label of the data provider.

c: ABILITY ADJUSTMENT AND DISSEMINATION RULES

1. Label adjustment strategy

The tenant virtual machine a adds/deletes label t_k to its own label L. L' is the new label of a, and the label L changes to L' and must meet the following rules:

$$L_a \to L'_a if \{L'_a - L_a\}^+ \subseteq A^+_a or \{L_a - L'_a\}^- \subseteq A^-_a$$

Subjects can actively change their own security label within the scope of their ability, actively improve their own security label to produce files with higher security, and then send the files to specific subjects. Combined with the explicit change in the subject security label, the security information flow conditions are more flexible, and a flexible tenant-led information flow control can be realized.

2. Ability adjustment strategy

When the subject a grants ability privilege t^+_k, t^-_k, or $t^{\pm}_k$ to subject b, the generated flow is represented by $a \overset{t^+_k or t^-_k or t^{\pm}_k}{\Rightarrow} b$.

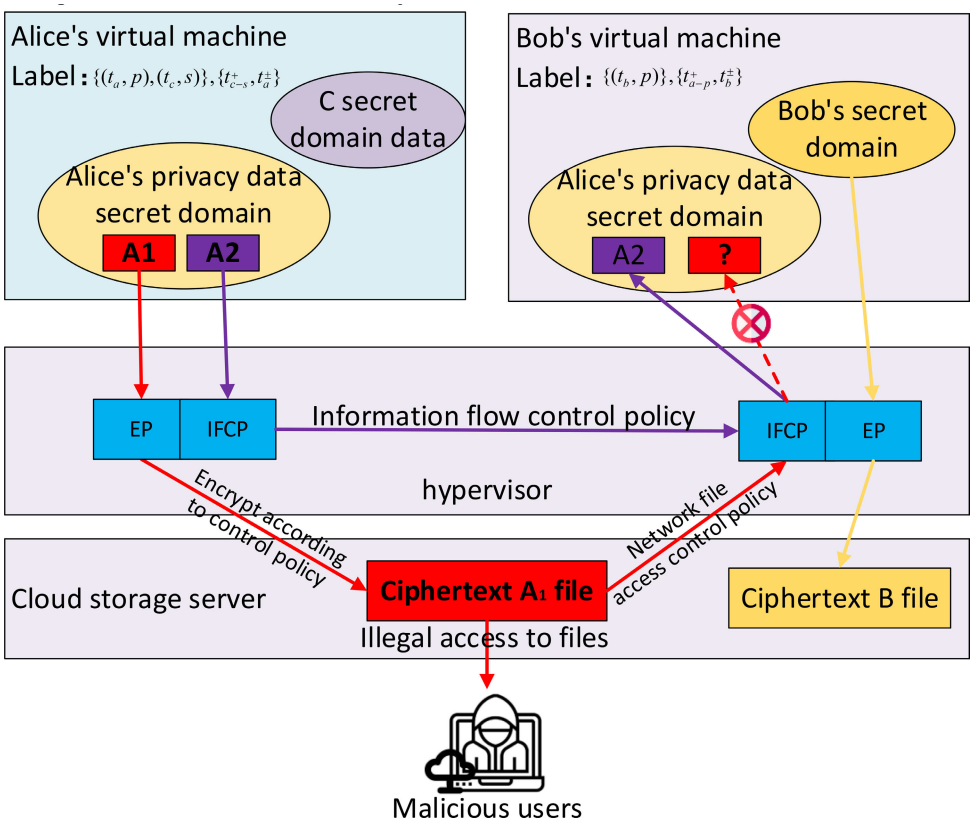

FIGURE 2. Ciphertext information flow control process.

When subject a grants subject ability privilege t_k^+, t_k^-, or $t_k^\pm$, it must have $t_k^\pm$. In other words,

$$a \overset{t_k^+ ort_k^- ort_k^\pm}{\Rightarrow} b \ only \ if \ t_k^\pm \in A_a$$

We define that subject a records the generation flow as $a \overset{t_k^+ ort_k^-}{\mapsto} b$ when the ability privilege t_k^+ or t_k^- of subject B is withdrawn. When subject a has the ability $t_k^\pm$, a can withdraw b ability privilege t_k^+ or t_k^-.

$$a \overset{t_k^+ ort_k^-}{\mapsto} b \ only \ if \ t_k^\pm \in A_a \ and \ (t_k^+ ort_k^-) \in A_b$$

$t_k^\pm$ represents the ownership of a type of secret-domain information, and the entity with ability $t_k^\pm$ has complete control over the data in the secret domain. Once an entity has been given privilege $t_k^\pm$, it will completely trust all operations of the entity on such private data, and this privilege will not be withdrawn.

So far, we formulated detailed information flow control rules for the program. Before we carry out a detailed design of the ciphertext information flow control mechanism, which is presented in Section IV, we first introduce the working process of the system.

C. SYSTEM WORKFLOW

1) SYSTEM INITIALIZATION PROCESS

1. CA generates a public key PK and a system private key MK.
2. The CA assigns identification *tid* to each label based on the initial strategy.
3. Secret-domain label key initialization: taking label t as an example, CA executes a key generation algorithm $Keygen(PK, tid, MK)$ to generate a tenant key share $sk_{tid-user}$ and a proxy key share sk_{tid-ep}. The proxy key is sent to the EA through a secure channel, and all the secret-domain tags initialize the key in turn.
4. The CA initializes the secret-domain secret-level label key for each private data secret domain. SCK_{k1} represents the secret-domain k top secret key, SCK_{k2} represents the secret-domain k confidential key, SCK_{k3} represents the secret-domain k secret key, and SCK_{k4} represents the secret-domain k public key.
5. When the virtual machine is created, the CA will load the security label and ability for the virtual machine and assign the corresponding label's tenant key shares $sk_{tid-user}$ to it.

2) LABEL MANAGEMENT

1. The data owner tenant virtual machine requests the CA to create a private data secret-domain label t, and the CA accepts the request to execute the key initialization process of label t, and distributes the tenant key private key share $sk_{tid-user}$ and proxy key share sk_{tid-ep} to the tenant virtual machine. t^+, t^-, $t^\pm$ are added to the tenant ability set A.

2. The tenant virtual machine a requests CA to grant tenant virtual machine b the t_{k-p}^{+} ability, i.e., tenant virtual machine b has the ability of adding label (t_k, p) to its own security label. The CA verifies whether the ability set of the tenant virtual machine a has label $t_k^{\pm}$. If yes, the CA adds t_{k-p}^{+} ability set of the tenant virtual machine b.
3. The tenant virtual machine applies to the CA to add a label (t_k, SC_k) based on its ability. The CA verifies whether the tenant virtual machine's ability set has the label $t_{k-SC_k}^{+}$. If yes, the tenant key share $sk_{tid-user}$ is issued to the tenant virtual machine, and the tenant virtual machine independently adds the label (t_k, SC_k) based on the ability $t_{k-SC_k}^{+}$.
4. The tenant virtual machine a requests the CA to withdraw the label (t_k, SC_k) or the ability t_k^{+} or t_k^{-} of the tenant virtual machine b. The CA verifies whether the tenant virtual machine a has the ability set of $t_k^{\pm}$, and if so, the label (t_k, SC_k) or the ability t_k^{+} or t_k^{-} of the tenant virtual machine b is deleted.
5. The tenant virtual machine a requests the CA to grant the tenant virtual machine b with the ability $t_k^{\pm}$, and the CA verifies whether the ability set of the tenant virtual machine a has label $t_k^{\pm}$, and if so, it adds $t_k^{\pm}$ to the ability set of the tenant virtual machine b.

3) VIRTUAL MACHINE COMMUNICATION PROCESS

The tenant virtual machine a sends file A_1 to the tenant virtual machine b. The decentralized information flow control module in the virtual machine hypervisor performs an information flow policy rule check on the security label of the tenant virtual machines a and b. If the rule is met, the security label of tenant virtual machine b is infected, and the tenant virtual machine b can read file A_1.

4) CLOUD NETWORK FILE READ AND WRITE ENCRYPTION AND DECRYPTION PROCESSES

a: TENANT VM A WRITES FILE A_1 TO THE CLOUD STORAGE SERVER

The security secret proxy module in the virtual machine hypervisor executes the information flow strategy network file writing rules, and the security encryption module executes algorithm $Encrypt(PK, tids, M)$ to encrypt the file A_1 based on the confidentiality label contained in the user's security label. The security label for the file is updated on the basis of the writing rule and is then transferred to the cloud storage server.

b: TENANT VM B READS FILE A_1 FROM THE CLOUD STORAGE SERVER

The decentralized information flow control module in the virtual machine hypervisor performs an information flow policy network file reading rule check on the file mark and the tenant virtual machine security mark. If the rules are met, the algorithm $Decrypt(PK, \{sk_{tid-ep}\}_{tid\in tids}, C)$ is used to decrypt the decrypted fragment $\{m_{tid-ep}\}_{tid\in tids}$ using the proxy key share, distribute the ciphertext of the file and the decrypted fragments to the tenant virtual machine b; otherwise, it replies that tenant virtual machine b exceeds the authority.

The tenant virtual machine b uses the tenant key share execution algorithm $Decrypt(PK, \{sk_{tid-ea}\}_{tid\in tids}, C)$ with the security label contained in the file to decrypt and obtain the decrypted fragment $\{m_{tid-user}\}_{tid\in tids}$. It uses all the decrypted fragments to execute the combination algorithm $combin(PK, \{m_{tid-w}\}_{tid\in tids, w\in\{user,ea\}}, C)$ to obtain the plaintext of the file. Finally, it completes the read operation.

IV. CIPHERTEXT INFORMATION FLOW CONTROL MECHANISM

A. CIPHERTEXT INFORMATION FLOW SYSTEM KEY MANAGEMENT

1) SECRET-DOMAIN LABEL KEY MANAGEMENT

The secret-domain owner $Owner_k$ applies for the secret domain k from the CA, and the CA performs a label key initialization of the secret domain. The secret-domain key distribution process is as follows:

$$CA \overset{send}{\rightarrow} sk_{tid} : \{sk_{tid-user}, sk_{tid-ea}\} \rightarrow^{to} Owner_k$$
$$CA \overset{send}{\rightarrow} message : \{sk_{tid-ea}\} \rightarrow^{to} EA$$

The secret-domain owner $Owner_k$ owns the two key shares of the secret domain, which means that he can decrypt it independently and can decrypt the data in the secret domain at will. $Owner_k$ holding the two key fragments of the secret domain has the right to read and spread the secret domain.

2) KEY MANAGEMENT MECHANISM BASED ON INFORMATION FLOW STRATEGY

a: SECRET-DOMAIN ABILITY t_k^{+} OR t_k^{-} GRANTED

The secret-domain owner $Owner_k$ shares the tenant key to the tenant he trusts, so that this tenant can decrypt the secret-domain data with the assistance of an encryption agent, and the secret-domain owner communicates with the CA to increase the corresponding ability for the tenant he trusts. The specific communication process is as follows:

$$Owner_k \overset{send}{\rightarrow} message : \{t_k^{+} or\ t_k^{-} \Rightarrow User_i\} \rightarrow^{to} CA$$
$$Owner_k \overset{send}{\rightarrow} message : \{skt_{User}\} \rightarrow^{to} User$$

b: SECRET-DOMAIN ABILITY t_k^{+} OR t_k^{-} REVOCATION

For the revocation of the label and ability, as long as the EP refuses to decrypt the segments for the tenant virtual machine computing proxy whose ability and label have been revoked, even if he has the tenant key share, the decryption cannot be completed. Therefore, only the label or ability needs to be changed for label and ability revocation. For label and ability revocation, information flow label revocation can be completed simply by changing its security label or ability set.

The specific process is as follows:

$$Owner_k \overset{send}{\rightarrow} message : \{t_k^+ or\ t_k^- or t_k \mapsto User_i\} \overset{to}{\rightarrow} CA$$

c: SECRET-DOMAIN ABILITY $t_k^{\pm}$ GRANT

The secret-domain $t_k^{\pm}$ ability represents the ownership of the secret-domain information, and an entity with ability $t_k^{\pm}$ fully controls the data in the secret domain. He/she can add/delete label t_k freely, must have all the key shares in the key management strategy, and can complete data decryption independently. Granting it to other tenant virtual machines only requires handing over the two pieces of keys to complete the granting of their secret-domain $t_k^{\pm}$ ability. The specific communication process is as follows:

$$Owner_k \overset{send}{\rightarrow} message : \left\{t_k^{\pm} \Rightarrow User_i\right\} \overset{to}{\rightarrow} CA$$
$$Owner_k \overset{send}{\rightarrow} message : \{skt_{User}, skt_{SEP}\} \overset{to}{\rightarrow} User$$

3) SECRET-DOMAIN SECRET-LEVEL KEY MANAGEMENT

The secret-domain owner $owner_k$ can apply to the CA for the secret-level key of the secret-domain k to realize the multi-level protection mechanism in the secret domain. The CA generates: $SCK_{k0} = Hash(MK||SK_k)$ and $SCK_{ki} = Hash(SCK_{ki-1})(0 < i \leq 3)$ and then generates $SCK_{kEP} = Hash(MK||tid_k)$ as the proxy key share of the secret-domain secret-level key. The CA distributes the secret-level keys of the secret domain $kSCK_{k0}$ and SCK_{kEP} to $Owner_k$ and EP. $SCK_{ki}(0 \leq i \leq 3)$ represents the secret-level keys corresponding to top secret, confidential, secret, and public.

The management mechanism of the secret-level key is the same as that of the label key. SCK_{kEP} is equivalent to skt_{EP}, and $SCK_{ki}(0 \leq i \leq 3)$ is equivalent to skt_{User}. At the same time, granting SCK_{kEP} and skt_{EP} means that the tenant has the ownership under the multi-level protection mechanism of this secret domain.

B. SPECIFIC IMPLEMENTATION OF ENCRYPTION ALGORITHM

The ciphertext information flow control encryption scheme: multi-ID-based threshold encryption scheme consists of four algorithms: system initialization, key generation algorithm, encryption algorithm, decryption algorithm, and combination algorithm.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be a multiplicative cyclic group with the order of a large prime number p, where the generator of $\mathbb{G}_1$ is g, and there is a bilinear pair $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The tag ID tid as the public key is an element in Z_p^*.

1) SYSTEM INITIALIZATION ALGORITHM

The system initialization is mainly to generate the system public parameter PK and the system master key MK by CA.

The CA randomly selects $x, y, z \in_R Z_p$ and calculates $X = g^x, Y = g^y, Z = g^z$. The system public parameters PK and system master key MK are: $PK = (g, X, Y, Z)$, $MK = (x, y, z)$.

2) KEY GENERATION ALGORITHM

In the *Keygen( )* algorithm, CA generates tid corresponding private key tenant key share and proxy key share for each secret domain, represented by:

$$Keygen(PK, tid, MK) \rightarrow sk_{tid-user}, sk_{tid-ep}$$

Randomly select polynomial $F(w) = z + aw(a \in_R Z_p)$ on Z_p^* and random number $r_u, r_p \in_R Z_p$, and calculate $K_{tid-u} = g^{F(1)/(tid+x+r_u y)}$ and $K_{tid-ep} = g^{F(2)/(tid+x+r_p y)}$. The tenant private key share $sk_{tid-user} = (r_u, K_{tid-u})$ and proxy private key share $sk_{tid-ep} = (r_p, K_{tid-p})$ are outputted.

3) ENCRYPTION ALGORITHM

The encryption algorithm *Encrypt( )* is an algorithm that uses all tid in the secret-domain identification set $tids$ as the public key to encrypt the data M to obtain the ciphertext C, represented by:

$$Encrypt(PK, tids, M) \rightarrow C$$

Select the random number $S_i \in Z_p^+, (i = 1, 2 \ldots, len(tids))$, $tids$ set as the set of marker tid that needs to be encrypted, and calculate the cipher text as follows:

$$C = (\{g^{S_i \cdot tid} X^{S_i}, Y^{S_i}\}_{tid \in tids, S_i \in S}, e(g, Z)^{\sum_{Si \in S} S_i} \cdot M)$$

The ciphertext C is divided into two parts, represented by $C = (\{A, B\}_{id \in tids}, D)$ for convenience; each tid corresponds to a group of A and B, $A = g^{S_i \cdot tid} X^{S_i}, B = Y^{S_i}$.

4) DECRYPTION ALGORITHM

The decryption algorithm *Decrypt( )*, i.e., the algorithm for decrypting the fragments, is obtained by the relevant parameter and the private key share, represented by:

$$Decrypt(PK, \{sk_{tid-w}\}_{tid \in tids}, C) \rightarrow \{m_{tid-w}\}_{tid \in tids}$$

To calculate the decryption segment $m_{tid}(tid \in tids)$ of $C = (\{A, B\}_{id \in tids}, D)$, the decryption party uses the key share $\{sk_{tid}\}_{tid \in tids}$ to decrypt the calculation to obtain: $m_{tid} = e(A_{tid} B_{tid}^{r_{tid}}, K_{tid})$.

5) COMBINATION ALGORITHM

The combination algorithm *Combin( )* is an algorithm that uses the decryption fragments of the tenant and proxy keys corresponding to all tid in the secret-domain identification set $tids$ for decryption, represented by:

$$\text{combin}\left(PK, \{m_{tid-w}\}_{tid \in tids, w \in \{\text{user}, ep\}}, C\right) \rightarrow M$$

After the tenant virtual machine collects all the decrypted fragments, it calculates M:

$$M = D / \prod_{tid \in tids} (m_{tid-ep}^{-1} \cdot m_{tid-user}^{2})$$

Proof of correctness:

Since the decryption key can be calculated as $e(g, g)^{S_i F(x)}$,

$$\begin{aligned} m_i &= e(g^{S_i \cdot tid} \cdot g^{xS_i} \cdot g^{yr_i}, g^{F(x)/(tid+x+r_i y)}) \\ &= e(g^{S_i(tid+x+yr_i)}, g^{F(x)/(tid+x+r_i y)}) \\ &= e(g, g)^{S_i F(x)} \end{aligned}$$

TABLE 1. Performance analysis of encryption algorithm.

Pairing	Add.	Mul.	Exp.	Bit length
1(or 0)	n	n	3n	$2nl\mathrm{G}_1 + l\mathrm{G}_2$

we have $m_{tid-ep}^{-1} \cdot m_{tid-user}^{2} = e(g,g)^{S_i F(2)\times(-1)+S_i F(1)\times 2} = e(g,g)^{S_i(F(2)\times(-1)+F(1)\times 2)}$

Using Lagrange interpolation method, we have:

$$F(x) = \sum_{i=1}^{n} (F(x_i) \prod_{j\neq i}^{1\leq j\leq n} \frac{(x-x_j)}{(x_i-x_j)})$$

This study divides the secret-domain label key into two, in the formula n = 2, x_1 =1, x_2 =2, so $F(0) = F(2) \times (-1) + F(1) \times 2$.

$$m_{tid-ep}^{-1} \cdot m_{tid-user}^{2} = e(g,g)^{S_i F(0)} = e(g,Z)^{S_i}$$

$$\prod_{tid\in tids} (m_{tid-ep}^{-1} \cdot m_{tid-user}^{2}) = e(g,Z)^{\sum_{Si\in S} S_i}$$

Performance analysis:

As shown in Table 1, in our scheme, n represents the number of tags used in the encryption, which requires approximately n addition operations, n multiplication operations, 3n power operations, and at most one pairing operation (if $e(g, Z)$ is calculated in advance, then the pairing operation is not required). In terms of the spatial efficiency, let $l\mathbb{G}_1$ and $l\mathbb{G}_2$ be the bit lengths of $\mathbb{G}_1$ and $\mathbb{G}_2$ elements, and only $2nl\mathbb{G}_1 + l\mathbb{G}_2$ is required.

V. SYSTEM IMPLEMENTATION AND ANALYSIS

A. SYSTEM IMPLEMENTATION

The running and development environment is as follows: The tenant virtual machine operating system used is Ubuntu 20.04, and the kernel version of the Linux system is 5.4.0. The EP module is a security control module compiled into the client Linux kernel. It is implemented on the existing LSM framework under Linux. The GCC version is 9.3.0. The CA management system is a web-based control system developed using Java EE 8.0. The system is developed using MyEclipse 10.0. MySQL version is 8.0.22. Tomcat server version is apache-tomcat-7.0.19. The client virtual machine can run on various cloud platforms such as OpenStack and Xen. Our experiment was conducted on the VMware Workstation 15.5.0. The host operating system was Windows 10 Professional Edition, the CPU was Inter(R) Core(TM) i7-8700K CPU@ 3.70 GHz, and the memory was 16 GB. The encryption scheme code was implemented in C language based on the pairing-based cryptography (PBC) library [32].

The current mainstream open-source cloud platforms include OpenStack [33], CloudStack [34], and Eucalyptus [35]. In terms of private data protection, they all have high security performance to ensure that the virtual machine can access strictly according to the policy. OpenStack keystone component uses a unified token authentication, and only operations that pass the audit can continue to be executed. CloudStack has no dedicated access control components or resource domains, and the security groups are completely physically isolated. Eucalyptus also has no dedicated access control components. The cloud controller (CLC) handles identity authentication and user management, and the centralized controller (CC) manages the security groups, virtual machines, and network access. Currently, conventional cloud platforms use the access control technology to ensure the security of storage data access. They are all based on RBAC to achieve access control to data in the cloud. Although it can realize the basic control of data, it is difficult to meet the challenges of cloud computing because it does not provide a higher guarantee for securing private data. Compared with our scheme, the existing open-source cloud platform lacks protection for the subsequent propagation of data and a mechanism for users to make their own private data access policies.

In [36], an outsourcing data protection framework in cloud was proposed. The data owner encrypts the data locally and then stores them in the cloud. The user needs to make a request to the data visitor to access the data. The data owner responds to the user in the form of an access token and access certificate. The user extracts cloud data through the token and decrypts the data using the access credentials. The access control scheme in [37] is similar to that in [36] in terms of access control. The authorization certificate and decryption key are used to control the access to cloud data, and the user key is managed through a key tree. Compared with our scheme, the strategies proposed in [36] and [37] lack protection after data decryption. Moreover, the strategy in [36] does not support dynamic policy adjustment, and the strategy in [37] is inefficient when it comes to data update and policy adjustment.

B. SECURITY PROOF

1) ALGORITHM SECURITY MODEL

Multi-ID-based threshold encryption scheme is IND-IDs-CPA safe, if and only if the adversary A wins the challenger–adversary interactive game first in polynomial time under non-negligible conditions.

Initialization: The challenger runs the system initialization algorithm to obtain the PK and MK, then sends the PK to A, and MK saves it.

Query 1: Adversary A can initiate a private key query to the challenger. Adversary A enters the tag id tid. The challenger runs the *Keygen( )* algorithm to generate the tenant private key share $sk_{tid-user}$ corresponding to the tid, and then sends it to the adversary A.

Challenge: Adversary A chooses two equal-length plaintexts M_0 and M_1, and a set of tag IDs for encryption $\{tid\}_{tid\in tids}$, and the challenger randomly chooses $\alpha \in \{0, 1\}$. The challenge ciphertext $C_\alpha = Encrypt(PK, \{tid\}_{tid\in tids}, M_\alpha)$ is calculated, and then C_α is sent to adversary A.

Query 2: Adversary A can continue to query the private key as required; however, the input tid cannot be in set $\{tid\}_{tid\in tids}$, and the challenger responds to adversary A as before.

Guess: In the end, adversary A must answer 0 or 1 (denoted by α') as a guess for ciphertext C_α. If $\alpha = \alpha'$, adversary A will win the game.

The above-mentioned adversary A is called an IND-IDs-CPA attacker, and A's advantage in winning the game given the security parameter k is:

$$Adv_{hibte}^{IND-IDs-CPA}(k) = |2\Pr[\alpha' = \alpha] - 1|$$

2) ALGORITHM IND-CPA SECURITY PROOF

Theorem 1: If it is difficult to determine the BDHI problem, then the multi-ID-based threshold scheme proposed in this article is said to be IND-IDs-CPA safe.

Assumption BDHI [31]: For a positive integer q and $\chi \in_R Z_r^*$, given $(g, g^\chi, g^{\chi^2}, \ldots, g^{\chi^q}, e(g,g)^\chi)$, computing $e(g,g)^{1/\chi}$ is difficult.

If the scheme is not safe, there is an adversary A who can break this scheme with a non-negligible probability advantage $\varepsilon(k)$, and then adversary A has an algorithm B to solve the deterministic BDHI problem with an advantage $\varepsilon(k)$. B now plays the role of a challenger. Before interacting with the adversary A, B needs to generate a generator $g \in \mathbb{G}^*$ and q-1 pairs of two-tuple $(s_i, g_i)(g_i = g^{1/(\chi+s_i)})$, and B is unaware of the specific value of χ. These parameters are calculated in the following manner [31]:

1. Randomly select $s_1, s_2, \ldots, s_{q-1} \in Z_p^*$ and let $f(\theta) = \prod_{i=1}^{q-1}(\theta + s_i) = \prod_{i=0}^{q-1} c_i\theta^i (c_0 \neq 0)$.
2. Calculate $g = \prod_{i=0}^{q-1}(h^{\chi^i})^{c_i} = h^{f(\chi)}$, $v = \prod_{i=1}^{q}(h^{\chi^i})^{c_{i-1}} = h^{\chi f(\chi)}$, $v = g^\chi$, and $g \neq 1$.
3. Let $f_i(\chi) = f(\chi)/(\chi + s_i) = \sum_{i=0}^{q-2} e_i\chi^i$, so
$$g^{1/(\chi+s_i)} = g^{f_i(\chi)} = \prod_{i=0}^{q-2}(g^{\chi^i})^{e_i}$$
4. Calculate
$$T_0 = \prod_{i=0}^{q-2} e(h^{(\chi^k)}, h)^{c_0 c_{k+1}} \prod_{i=1}^{q-1} \prod_{j=0}^{1-2} e(h^{(\chi^i)}, h^{(\chi^j)})^{c_i c_{j+1}},$$
$$T_g = T^{(c_0^2)} \cdot T_0. If\ T = e(h,h)^{1/\chi}, T_g = e(g^{f(\chi)}, g^{f(\chi)})^{1/\chi} = e(g,g)^{1/\chi}.$$

Adversary A and B start adversary games as follows:

Initialization B simulates the challenger and executes the following steps:

B randomly selects $a, b, z \in Z_p^*$, satisfies ab= $tid_1(tid_1 \in \{tid\}_{tid\in tids})$, and calculates $X = v^{-a}g^{-tid} = g^{-a(\chi+b)}$, $Y = g^\chi$, $Z = g^z$. The public key is $PK = (g, X, Y, Z)$, and the system private key is $MK = (-a(\chi + b), \chi, z)$.

Query 1: Adversary A initiates a series of queries. B receives A's private key query about tid' and simulates the challenger to generate the key. First, it is checked whether tid' is equal to tid_1; if not, go to step a; otherwise, proceed to step b.

TABLE 2. Time consumptions under different number of tags.

$tidnum$	t_{KG} (ms)	t_E (ms)	t_{EPDe} (ms)	t_{UDe} (ms)	t_{UC} (ms)
1	1.708	42.38	1.426	1.337	42.08
5	8.038	53.12	6.610	6.587	51.31
10	17.42	62.22	13.94	13.402	63.87
15	24.22	75.38	20.27	20.09	75.76
20	32.42	86.48	27.16	26.70	88.90

a. B randomly selects polynomial $F(w) = z + aw(a \in_R Z_p)$ on Z_p^*, randomly selects an unused two-tuple (s_i, g_i), calculates $r_u = a + \frac{tid' - tid^*}{w_u}$, and returns $SK_{tid'-user} = (r_u, g^{F(1)/(r_u-\beta)})$ to adversary A.

b. B randomly selects $r_i \in Z_p^*$ and returns $SK_{tid'-user} = (r_u, g^{F(1)/(r_u-\beta)})$ to adversary A.

Challenge: Adversary A thinks that the query for $\{tid\}_{tid\in tids}$ can be ended, and A chooses two equal-length plaintexts M_0 and M_1 and a set of tag IDs set $\{tid\}_{tid\in tids}$ for encryption. B imitates the challenger to randomly choose $\alpha \in \{0, 1\}$. The challenge ciphertext is calculated:

$$C_\alpha = Encrypt(PK, \{g^{-al_{tid}}, g^{l_{tid}}\}_{tid\in tids}, \prod_{tid\in tids} T_g^{zl_{tid}} M_\alpha)$$

and then C_a is sent to adversary A.

Let $u = l/\chi (l \in Z_p^*)$, then

$$T_g^{zl} = e(g,g)^{z\cdot l/\chi} = e(g,Z)^s, g^l = Y^{l/\chi} = Y^s$$
$$g^{-al} = g^{-a\chi(l/\chi)} = g^{(x+tid^*)(l/\chi)} = g^{s\cdot tid^*\cdot X^s}$$

If $T_g = e(h,h)^{1/\chi}$, B simulates the encryption algorithm correctly.

Query 2: Adversary A can continue to Query 1 as required, and the private key query is q times at most. B returns to adversary A as it did in Query 1.

Guess: Adversary A thinks that Query 2 is over, and it must answer $\alpha' \in \{0, 1\}$ as a guess for ciphertext C_α. If $\alpha' = \alpha$, then B will output 1, which means $T = e(g,g)^{1/\chi}$; otherwise, B will output 0, which means $T \neq e(g,g)^{1/\chi}$.

In the above challenger–adversary game, if the input T is $T = e(g,g)^{1/\chi}$, then the adversary A's probability advantage is $Adv_{hibte}^{IND-IDs-CPA}(k) = |2\Pr[\alpha' = \alpha] - 1| > \varepsilon(k)$, so that B meets $\Pr[T = e(g,g)^{1/\chi}] > 1/2 + \varepsilon(k)$, when solving the BDHI problem. In summary, the probability advantage of algorithm B to solve the BDHI problem is:

$$Adv_B(k) = |(1/2 + \varepsilon(k)) - 1/2| = \varepsilon(k)$$

Hence, Theorem 1 is proven.

C. PERFORMANCE TESTING

In the experiment, we analyzed the effects of the number of secret-domain tags and file size on the efficiency of encryption and decryption, and performed an efficiency comparison.

TABLE 3. Time consumptions under different file sizes.

$file-size$	t_{EPDe} (ms)	t_{UDe} (ms)
1 KB	12.64	19.22
10 KB	12.94	19.55
100 KB	16.21	22.84
1 MB	49.78	56.36
5 MB	199.9	204.8
10 MB	386.7	389.6

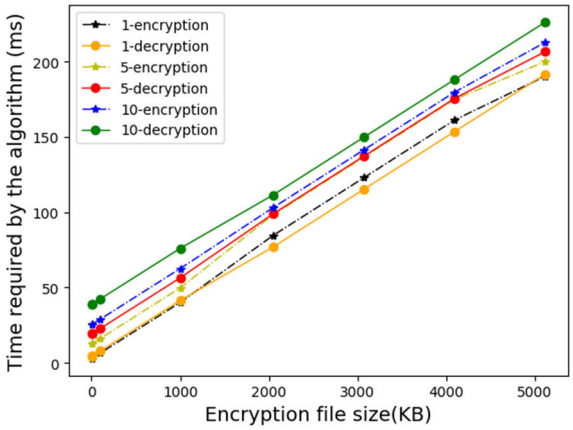

FIGURE 3. Comparison of time consumption under different file size and number of labels.

a: EFFICIENCY OF THE ALGORITHM UNDER DIFFERENT NUMBERS OF SECRET-DOMAIN TAGS

First, the time consumption of the algorithm is tested under different tag number, and the encrypted file is 1 MB in size. The time consumed for the CA generation key generation, encryption time consumption, proxy decryption time consumption, tenant decryption time consumption, and tenant synthesis decryption time is tested. As listed in Table 2, t_{KG} is the key generation time consumption, t_E is the encryption time consumption, t_{EPDe} is the time consumption of the decryption fragment decryption by the encryption agent, t_{UDe} is the time consumed for the decryption fragment decryption by tenant, and t_{UC} is the time consumed by the tenant to decrypt plaintext.

b: EFFICIENCY OF THE ALGORITHM UNDER DIFFERENT ENCRYPTED FILE SIZES

Table 3 lists the encryption and decryption times corresponding to different file sizes with five tags. The encryption time, decryption time, and file size are linear. t_E is the encryption time consumption, and t_{De} is the decryption time consumption.

By testing the time consumption of one tag, five tags, and ten tags under different file sizes, the encryption and decryption operations are performed on files with sizes ranging from 1 kB to 10 MB. Figure 3 shows the results, and the time consumption increases linearly with the file size; however, the growth rate is lower. The number of tags has little effect on the efficiency of the algorithm.

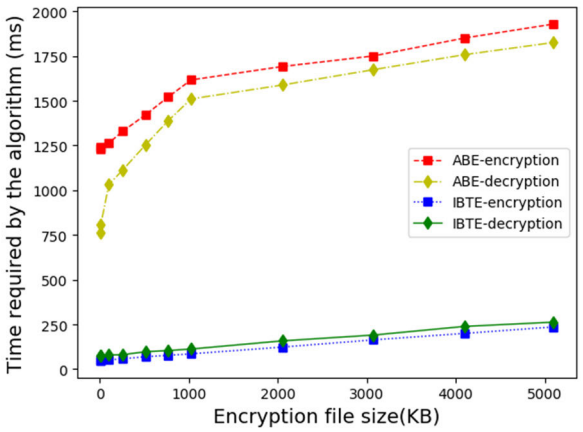

FIGURE 4. Efficiency comparison between ABE and IBTE under different file size.

c: EFFICIENCY COMPARISON

In the research on access control encryption, the attribute-based encryption technology has received widespread attention; however, attribute-based encryption is often inefficient. Therefore, this study uses multi-ID-based threshold encryption scheme for encryption design. Figure 4 shows a comparison between the multi-ID-based threshold encryption scheme and the attribute-based encryption in terms of the time consumption. In the ciphertext information flow control system proposed in this article, our encryption algorithm is faster than the attribute-based encryption algorithm. Therefore, the encryption mechanism proposed in this article is said to be feasible.

VI. CONCLUSION

To provide a comprehensive protection for the private data stored in a cloud environment, we developed a decentralized ciphertext information flow control scheme for cloud virtual machines. A decentralized information flow control mechanism with tenant-led data access control policies was implemented between the cloud virtual machines, and the private data stored in the cloud storage server were encrypted on the basis of the information flow control policy to prevent the system from controlling the reading of private data by illegal entities inside and outside the system. We designed an information flow security strategy, a secret-domain label key management scheme, and a multi-ID-based threshold encryption scheme to achieve taint infection, secret-level reduction, and ability propagation information flow control strategy in the ciphertext form in the system. As for the limitation of the study, the system proposed in this article works under a security assumption, i.e., CA and EP are safe and always online. However, in actual operation, the centralized EP and CA are prone to the single-point bottleneck problem, and when offline, the availability of the system will be affected. The privacy and integrity of the private key were not considered in this study. In the future work, we hope a prototype system of the decentralized ciphertext information flow control will be designed and implemented on the basis of the OpenStack

open source cloud platform, and the security and performance of the designed scheme will be evaluated and verified more comprehensively from an implementation perspective. We will study the scheme in the case of a distributed CA and consider the privacy integrity of the private key, which will make the protection of private data stored in the cloud more perfect and reliable.

REFERENCES

[1] C.-S. Feng, Z.-G. Qin, D. Yuan, and Y. Qing, "Key techniques of access control for cloud computing," *Chin. J. Electron.*, vol. 43, no. 2, pp. 312–319, 2015.
[2] Y. D. Wang, J. H. Yang, C. Xu, X. Ling, and Y. Yang, "Survey on access control technologies for cloud computing," *Ruan Jian Xue Bao/J. Softw.*, vol. 26, no. 5, pp. 1129–1150, 2015.
[3] R. El Sibai, N. Gemayel, J. Bou Abdo, and J. Demerjian, "A survey on access control mechanisms for cloud computing," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, Feb. 2020, Art. no. e3720.
[4] F. Cai, N. Zhu, J. He, P. Mu, W. Li, and Y. Yu, "Survey of access control models and technologies for cloud computing," *Cluster Comput.*, vol. 22, no. S3, pp. 6111–6122, May 2019.
[5] A. Almtrf, Y. Alagrash, and M. Zohdy, "Framework modeling for user privacy in cloud computing," in *Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2019, pp. 0819–0826.
[6] D. E. Denning, "A lattice model of secure information flow," *Commun. ACM*, vol. 19, no. 5, pp. 236–243, May 1976.
[7] Z.-Z. Wu, X.-Y. Chen, Z. Yang, and X.-H. and Du, "Survey on information flow control," *Ruan Jian Xue Bao/J. Softw.*, vol. 28, no. 1, pp. 135–159, 2017.
[8] T. F. J. M. Pasquier, J. Bacon, and D. Eyers, "FlowK: Information flow control for the cloud," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2014, pp. 70–77.
[9] J. Singh, T. F. J.-M. Pasquier, J. Bacon, and D. Eyers, "Integrating messaging middleware and information flow control," in *Proc. IEEE Int. Conf. Cloud Eng.*, Mar. 2015, pp. 54–59.
[10] T. F. J.-M. Pasquier, J. Singh, D. Eyers, and J. Bacon, "Camflow: Managed data-sharing for cloud services," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 472–484, Jul. 2017.
[11] A. Khurshid, A. N. Khan, F. G. Khan, M. Ali, J. Shuja, and A. U. R. Khan, "Secure-CamFlow: A device-oriented security model to assist information flow control systems in cloud environments for IoTs," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 8, Apr. 2019, Art. no. e4729.
[12] Z.-Z. Wu, X.-Y. Chen, and X.-H. Du, "Enhancing sensitive data security based-on double-layer information flow controlling in the cloud," *Acta Electron. Sinica*, vol. 46, no. 9, pp. 2245–2250, Sep. 2018.
[13] M. Elsayed and M. Zulkernine, "IFCaaS: Information flow control as a service for cloud security," in *Proc. 11th Int. Conf. Availability, Rel. Secur. (ARES)*, Aug. 2016, pp. 211–216.
[14] N. E. Moussaid and M. E. Azhari, "Enhance the security properties and information flow control," *Int. J. Electron. Bus.*, vol. 15, no. 3, pp. 249–274, 2020.
[15] X. Lu, L. Cao, and X. Du, "Dynamic control method for tenants' sensitive information flow based on virtual boundary recognition," *IEEE Access*, vol. 8, pp. 162548–162568, 2020.
[16] S. Aldossary and W. Allen, "Data security, privacy, availability and integrity in cloud computing: Issues and current solutions," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 4, pp. 485–498, 2016.
[17] S. Belguith, A. Jemai, and R. Attia, "Enhancing data security in cloud computing using a lightweight cryptographic algorithm," in *Proc. 11th Int. Conf. Autonomic Auton. Syst.*, May 2015, pp. 1–6.
[18] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, Berlin, Germany: Springer, 1984, pp. 47–53.
[19] I. Damgård, H. Haagh, and C. Orlandi, "Access control encryption: Enforcing information flow with cryptography," in *Proc. Theory Cryptogr. Conf.*, Berlin, Germany: Springer, 2016, pp. 547–576.
[20] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 40–49.
[21] G. Tan, R. Zhang, H. Ma, and Y. Tao, "Access control encryption based on LWE," in *Proc. 4th ACM Int. Workshop ASIA Public-Key Cryptogr. APKC*, 2017, pp. 43–50.
[22] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Berlin, Germany: Springer, 2005, pp. 457–473.
[23] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.
[24] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. Ho Allen Au, "Improving privacy and security in decentralized ciphertext-policy attribute-based encryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 665–678, Mar. 2015.
[25] J. Han, L. Chen, W. Susilo, X. Huang, A. Castiglione, and K. Liang, "Fine-grained information flow control using attributes," *Inf. Sci.*, vol. 484, pp. 167–182, May 2019.
[26] K. Liang, W. Susilo, and J. K. Liu, "Privacy-preserving ciphertext multi-sharing control for big data storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1578–1589, Aug. 2015.
[27] J. Liu, X. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption," *Future Gener. Comput. Syst.*, vol. 52, pp. 67–76, Nov. 2015.
[28] C. Dwork, "Differential privacy: A survey of results," in *Proc. Int. Conf. Theory Appl. Models Comput.*, Berlin, Germany: Springer, 2008, pp. 1–19.
[29] A. Roth and T. Roughgarden, "Interactive privacy via the median mechanism," in *Proc. 42nd ACM Symp. Theory Comput. - STOC*, 2010, pp. 765–774.
[30] A. Parakh and S. Kak, "Online data storage using implicit security," *Inf. Sci.*, vol. 179, no. 19, pp. 3323–3331, Sep. 2009.
[31] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Berlin, Germany: Springer, 2004, pp. 223–238.
[32] *The Pairing-Based Cryptography Library*. Accessed: Dec. 27, 2020. [Online]. Available: https://crypto.stanford.edu/pbc/
[33] *The Most Widely Deployed Open Source Cloud Software in the World*. Accessed: Dec. 27, 2020. [Online]. Available: http://www.openstack.org/
[34] *Apache CloudStack: About*. Accessed: Dec. 27, 2020. [Online]. Available: http://cloudstack.apache.org/
[35] *Eucalyptus Documentation*. Accessed: Dec. 27, 2020. [Online]. Available: https://www.eucalyptus.com/
[36] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, Berlin, Germany: Springer, 2010, pp. 136–149.
[37] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proc. ACM Workshop Cloud Comput. Secur. - CCSW*, 2009, pp. 55–66.

ZHAO ZHANG is currently pursuing the master's degree with PLA Information Engineering University, Zhengzhou, China. His research interests include information flow control, cryptographic technology, and cloud computing security.

ZHI YANG received the Ph.D. degree in computer science and technology from the Chinese Academy of Sciences, Beijing, China, in 2012. He is currently an Associate Professor with PLA Information Engineering University, Zhengzhou, China. His research interests include cryptographic technology, access control, and cloud computing security.

XUEHUI DU received the Ph.D. degree in computer science and technology from PLA Information Engineering University, Zhengzhou, China, in 2011. She is currently a Professor with PLA Information Engineering University. Her research interests include cyberspace security and access control.

WENFA LI received the Ph.D. degree in computer software and theory from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2009. He is currently a Professor with the Department of Software Engineering, College of Robotics, Beijing Union University, China. His research interests include intrusion detection, network security, data mining, and big data.

XINGYUAN CHEN received the Ph.D. degree in communication and information system from PLA Information Engineering University, Zhengzhou, China, in 2003. He is currently a Professor with PLA Information Engineering University. He is also a Ph.D. Supervisor with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. His research interests are generally in the areas of cyberspace security, cloud computing security, and OS security.

LEI SUN received the Ph.D. degree in computer and communication engineering from PLA Information Engineering University, Zhengzhou, China, in 2016. He is currently a Professor with PLA Information Engineering University. His research interests are generally in the areas of cyberspace security and OS security.