A secure data self-destructing scheme in cloud computing

Jinbo Xiong, *Student Member, IEEE,* Ximeng Liu, *Student Member, IEEE,* Zhiqiang Yao, Jianfeng Ma, Qi Li, Kui Geng, and Patrick S. Chen

Abstract—With the rapid development of versatile cloud services, it becomes increasingly susceptible to use cloud services to share data in a friend circle in the cloud computing environment. Since it is not feasible to implement full lifecycle privacy security, access control becomes a challenging task, especially when we share sensitive data on cloud servers. In order to tackle this problem, we propose a key-policy attribute-based encryption with time-specified attributes (KP-TSABE), a novel secure data self-destructing scheme in cloud computing. In the KP-TSABE scheme, every ciphertext is labeled with a time interval while private key is associated with a time instant. The ciphertext can only be decrypted if both the time instant is in the allowed time interval and the attributes associated with the ciphertext satisfy the key's access structure. The KP-TSABE is able to solve some important security problems by supporting user-defined authorization period and by providing fine-grained access control during the period. The sensitive data will be securely self-destructed after a user-specified expiration time. The KP-TSABE scheme is proved to be secure under the decision l-bilinear Diffie-Hellman inversion (l-Expanded BDHI) assumption. Comprehensive comparisons of the security properties indicate that the KP-TSABE scheme proposed by us satisfies the security requirements and is superior to other existing schemes.

Index Terms—Sensitive data, secure self-destructing, fine-grained access control, privacy-preserving, cloud computing

1 Introduction

Cloud computing is considered as the next step in the evolution of on-demand information technology which combines a set of existing and new techniques from research areas such as service-oriented architectures (SOA) and virtualization. With the rapid development of versatile cloud computing technology and services, it is routine for users to leverage cloud storage services to share data with others in a friend circle, e.g., Dropbox, Google Drive and AliCloud [1].

The shared data in cloud servers, however, usually contains users' sensitive information (e.g., personal profile, financial data, health records, etc.) and needs to be well protected [2]. As the ownership of the data is separated from the administration of them [3], the cloud servers may migrate users' data to other cloud servers in outsourcing or share them in cloud searching [4]. Therefore, it becomes a big challenge to protect the privacy of those shared data in cloud, especially in cross-cloud and big data environment [5]. In order to meet this challenge, it is necessary to design a comprehensive solution to support user-defined authorization period and to provide fine-grained access control during this period. The shared data should be self-destroyed after the user-defined expiration time.

One of the methods to alleviate the problems is to store data as a common encrypted form. The disadvantage of encrypting data is that the user cannot share his/her encrypted data at a fine-grained level. When a data owner wants to share someone his/her information, the owner must know exactly

- *J. Xiong and Z. Yao (corresponding author) are with the Faculty of Software, Fujian Normal University, Fuzhou, 350108, China. E-mail: jbxiongfj@gmail.com; yzq@fjnu.edu.cn.*
- *X. Liu and K. Geng are with the School of Telecommunications Engineering, Xidian University, Xi'an, 710071, China. E-mail: snbnix@gmail.com; gengkui@139.com.*
- *J. Ma and Q. Li are with the School of Computer Science and Technology, Xidian University, Xi'an, 710071, China. E-mail: jfma@mail.xidian.edu.cn, qilijs@gmail.com.*
- *P. Chen is with the Department of Information Management, Tatung University, 40, Sect.3, Zhongshan N. Road, 104 Taipei, Taiwan. E-mail: chenps@ttu.edu.tw.*

Manuscript received Jan. 14, 2014; revised xxx, 2014.

the one he/she wants to share with [6]. In many applications, the data owner wants to share information with several users according to the security policy based on the users' credentials. Attribute-based encryption (ABE) has significant advantages based on the tradition public key encryption instead of one-to-one encryption because it achieves flexible one-to-many encryption [7]. ABE scheme provides a powerful method to achieve both data security and fine-grained access control. In the key-policy ABE (KP-ABE) scheme to be elaborated in this paper, the ciphertext is labeled with set of descriptive attributes. Only when the set of descriptive attributes satisfies the access structure in the key, the user can get the plaintext [8].

In general, the owner has the right to specify that certain sensitive information is only valid for a limited period of time, or should not be released before a particular time. Timed-release encryption (TRE) provides an interesting encryption service where an encryption key is associated with a predefined release time, and a receiver can only construct the corresponding decryption key in this time instance [9]. On this basis, Paterson et al. proposed a time-specific encryption (TSE) [10] scheme, which is able to specify a suitable time interval such that the ciphertext can only be decrypted in this interval (decryption time interval, DTI). It can be used in many applications, e.g., Internet programming contest, electronic sealed-bid auction, etc. Electronic sealed-bid auction is a method to establish the price of goods through the Internet while keeping the bids secret during the bidding phase. That is, the bids (ciphertext) should be kept secret during the bidding phase (a specific time interval).

However, applying the ABE to the shared data will introduce several problems with regard to time-specific constraint and self-destruction, while applying the TSE will introduce problems with regard to fine-grained access control. Thus, in this paper, we attempt to solve these problems by using KP-ABE and adding a constraint of time interval to each attribute in the set of decryption attributes.

1.1 Related Works

1.1.1 Attribute-based encryption

Attribute-based encryption is one of the important applications of fuzzy identity-based encryption [7]. ABE comes in two flavors called KP-ABE [8][11] and ciphertext-policy ABE (CP-ABE) [12][13]. In CP-ABE, the ciphertext is associated with the access structure while the private key contains a set of attributes. Bethencourt et al. proposed the first CP-ABE scheme [12], the drawback of their scheme is that security proof was only constructed under the generic group model. To address this weakness, Cheung et al. presented another construction under a standard model [13]. Waters used a linear secret sharing scheme (LSSS) matrix as a general set of access structures over the attributes and proposed an efficient and provably secure CP-ABE scheme under the standard model [14].

In KP-ABE, the idea is reversed: the ciphertext contains a set of attributes and the private key is related to the access structure. The first construction of KP-ABE scheme was proposed in [8]. In their scheme, when a user made a secret request, the trusted authority determined which combination of attributes must appear in the ciphertext for the user to decrypt. Instead of using the Shamir secret key technique [15] in the private key, this scheme used a more generalized form of secret sharing to enforce a monotonic access tree. Ostrovsky et al. presented the first KP-ABE system which supports the non-monotone formulas in key policies [16]. Yu et al. used a combining technique of KP-ABE, proxy re-encryption, and lazy re-encryption which allows the data owner to delegate most of the computation tasks involved in fine-grained data access control to untrusted cloud servers without disclosing the underlying data contents [17]. Tysowski et al. modified the ABE and leveraged re-encryption algorithm to propose a novel scheme to protect mobile user's data in cloud computing environment [18]. Due to the lack of time constraints, the above-mentioned ABE schemes do not support user-defined authorization period and secure self-destruction after expiration for privacy-preserving of the data lifecycle in cloud computing.

1.1.2 Secure self-destruction scheme

A well-known method for addressing this problem is secure deletion of sensitive data after expiration when the data was used [19]. Recently, Cachin et al. employed a policy graph to describe the relationship between attributes and the protection class and proposed a policy-based secure data deletion scheme [20]. Reardon et al. leveraged the graph theory, B-tree structure and key wrapping and proposed a

novel approach to the design and analysis of secure deletion for persistent storage devices [21]. Because of the properties of physical storage media, the above-mentioned methods are not suitable for the cloud computing environment as the deleted data can be recovered easily in the cloud servers [22].

A data self-destructing scheme, first proposed by Geambasu et al. [23], is a promising approach which designs a Vanish system enables users to control over the lifecycle of the sensitive data. Wang et al. improved the Vanish system and proposed a secure self-destructing scheme for electronic data (SSDD) [24]. In the SSDD scheme, a data is encrypted into a ciphertext, which is then associated and extracted to make it incomplete to resist against the traditional cryptanalysis and the brute-force attack. Then, both the decryption key and the extracted ciphertext are distributed into a distributed hash table (DHT) network to implement self-destruction after the update period of the DHT network. However, Wolchok et al. made a lot of experiments and confirmed that the Vanish system is vulnerable to Sybil attacks by using the Vuze DHT network [25]. So the security of the SSDD scheme is also questionable.

To address this problem, Zeng et al. proposed a SeDas system, which is a novel integration of cryptographic techniques with active storage techniques [26]. Xiong et al. leveraged the DHT network and identity-based encryption (IBE) [27] and proposed an IBE-based secure self-destruction (ISS) scheme [22]. In order to protect the confidentiality and privacy security of the composite documents within the whole lifecycle in cloud computing, Xiong et al. applied the ABE algorithm to propose a secure self-destruction scheme for composite documents (SelfDoc) [28][29]. Recently, Xiong et al. employed identity-based timed-release encryption (ID-TRE) algorithm [9] and the DHT network and proposed a full lifecycle privacy protection scheme for sensitive data (FullPP), which is able to provide full lifecycle privacy protection for users' sensitive data by making it unreadable before a predefined time and automatically destructed after expiration [3]. The main idea of the above-mentioned schemes is that they respectively combine different cryptographic techniques with the DHT network to provide fine-grained data access control during the lifecycle of the protected data and to implement data self-destruction after expiration. However, using of the DHT network will result in the fact that the lifecycle of the protected data is limited by the update periods of the nodes in the network. How to implement user-defined data lifecycle is a worth exploring problem.

1.1.3 Time-specific encryption

The time-specific encryption scheme TSE, proposed by Peterson et al. [10], was introduced as an extension of TRE [9]. In TRE, a protected data can be encrypted in such a way that it cannot be decrypted (even by a legitimate receiver who owns the decryption key for the ciphertext) until the time (called the release-time) that was specified by the encryptor. Most of the previous TRE schemes that adopt a time-sever model are in fact public-key TRE schemes. They do not consider the sensitive data privacy after expiration [30][31][32].

In the TSE scheme, a time sever broadcasts a time instant key (TIK), a data owner encrypts a message into a ciphertext during a time interval, and a receiver can decrypt the ciphertext if the TIK is valid in that interval. Kasamatsu designed an efficient TSE scheme by using forward-secure encryption (FSE) in which the size of the ciphertext is greatly small than that generated by the previous schemes [33]. The time interval may be considered as the authorization period of the protected data, and TSE schemes are able to meet this requirement. However, it is a tricky problem when the traditional TSE is used in the cloud computing environment: cloud computing environment needs a fine-grained access control [17], which cannot be provided by the traditional TSE schemes. How to achieve the time-specified ciphertext into a fine-grained access control level is a problem to be explored.

1.2 Motivation

As the-state-of-the-art of the secure self-destruction scheme, both SSDD and FullPP have some limitations. First, SSDD does not consider the issue of the desired release time of the sensitive data, the expiration time of both SSDD and FullPP schemes is limited by the DHT network and cannot be determined by the user. Second, SSDD and many other schemes are dependent on the ideal assumption of "*No attacks on VDO (vanishing data object) before it expires*" [23]. Third, it is demonstrated that the Vanish scheme [23] is vulnerable to the Sybil attacks from the DHT network, the SSDD scheme and other schemes are similar. As a result,

unauthorized users can freely access to the sensitive data and this flaw would lead to a serious privacy disclosure [25].

To address these problems, in this paper, we propose a novel solution called key-policy attribute based encryption with time-specified attributes (KP-TSABE) scheme, which is based on our observation that, in practical cloud application scenarios, each data item can be associated with a set of attributes and every attribute is associated with a specification of time interval (decryption attribute time interval, DATI), e.g., [09:00,17:00], denoting that the encrypted data item can only be decrypted between 09:00 to 17:00 on a specified date and it will not be recoverable before 09:00 and after 17:00 that day. The data owner encrypts his/her data to share with users in the system, in which every users key is associated with an access tree and each leaf node is associated with a time instant, e.g., 14:30. The access tree of each user can be defined as a unique logical expression over these DATI attributes to reflect the data item authorized to the user. In order to decrypt the ciphertext successfully, the valid attributes should satisfy the access tree where the time instant of each leaf in the users key should belong to the DATI (e.g.,14:30$\in$[09:00,17:00]) in the corresponding attribute in the ciphertext. As the logical expression of the access tree can represent any desired data set with any time interval, it can achieve fine-grained access control. If the time instant is not in the specified time interval, the ciphertext cannot be decrypted, i.e., this ciphertext will be self-destructed and no one can decrypt it because of the expiration of the secure key. Therefore, secure data self-destruction with fine-grained access control is achieved.

1.3 Contributions

In this paper, we propose a KP-TSABE scheme, which is a novel secure self-destructing scheme for data sharing in cloud computing. We first introduce the notion of KP-TSABE, formalize the model of KP-TSABE and give the security model of it. Then, we give a specific construction method about the scheme. Finally, we prove that the KP-TSABE scheme is secure.

Especially, KP-TSABE has the following advantages with regard to security and fine-grained access control compared to other secure self-destructing schemes.

① KP-TSABE supports the function of user-defined authorization period and ensures that the sensitive data cannot be read both before its desired release time and after its expiration.

② KP-TSABE does not require the ideal assumption of "*No attacks on VDO before it expires*".

③ KP-TSABE is able to implement fine-grained access control during the authorization period and to make the sensitive data self-destruction after expiration without any human intervention.

④ KP-TSABE is proven to be secure under the standard model by using the l-bilinear Diffie-Hellman inversion assumption.

Organization. The rest of this paper is organized as follows: Section 2 describes the preliminaries, and the concepts, system model, formal model and security model of the KP-TSABE scheme are presented in Section 3. We construct the KP-TSABE scheme in Section 4. Security proof, comparison and analysis are presented in Section 5 and Section 6 respectively. Finally, we concluded this paper in section 7.

2 Preliminaries

In this section, some preliminaries related to bilinear maps, complexity assumptions and access structure are presented.

2.1 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}'$ be two multiplicative cyclic groups with big prime order p. Let g be a generator of $\mathbb{G}$. Let e be a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}'$ with the following properties [34]:

1) Bilinearity: For all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, the equation $e(u^a, v^b) = e(u, v)^{ab}$ holds.
2) Non-degeneracy: $e(g, g) \neq 1$.
3) Computability: There exists an efficient algorithm to compute bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}'$.

2.2 Bilinear Diffie-Hellman Inversion (BDHI) Assumption

In order to prove the security of the KP-TSABE scheme, we introduce l-BDHI assumption used in [34]. The l-BDHI problem in $\mathbb{G}$ is as follows: Given g, h and g^{y^i} in $\mathbb{G}$ for $i = 1, 2, \cdots, l$ as input for some unknown random $y \in \mathbb{Z}_p^*$, output $W \in \mathbb{G}'$ to decide whether $W = e(g, h)^{y^{l+1}}$. We say that

a polynomial-time adversary $\mathcal{A}$ has advantage ϵ in solving the decisional l-BDHI problem $(\mathbb{G}, \mathbb{G}')$ if:

$$|\Pr[\mathcal{A}(g, h, \mathbf{y}, e(g,h)^{y^{l+1}}) = 0] - \Pr[\mathcal{A}(g, h, \mathbf{y}, e(g,h)^{z}) = 0]| \geq \epsilon$$

where the probability is taken over random y, z and the random bits consumed by $\mathcal{A}$.

Definition 1. *We say that the (t, ϵ)-l-BDHI assumption holds in $(\mathbb{G}, \mathbb{G}')$ if no t-time algorithm has the probability at least ϵ in solving the l-BDHI problem for non-negligible ϵ.*

2.3 Access structure and access tree

2.3.1 Access structure

Definition 2 (Access structure [35]). *Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, \cdots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotonic access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2 \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, \cdots, P_n\}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

2.3.2 Access tree with time-specific attributes

We denote Υ as an access tree. Each non-leaf node of the tree represents a threshold gate, described by a threshold value and its children [8]. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x < num_x$ holds. The threshold gate is an OR gate when threshold value $k_x = 1$. If threshold value of node x satisfied $k_x = num_x$, it is an AND gate [34]. Each leaf node x of the tree is associated with a time instant t_x. If the t_x belongs to a time interval $[t_{L,x}, t_{R,x}]$, which is associated with the corresponding attribute x in the ciphertext, we let value $k_x = 1$.

Some functions are defined in order to facilitate dealing with Υ. In Υ, the function $parent(x)$ is represented as the parent of the node x. The component of attributes is associated with the leaf node x in Υ. Υ also defines an ordering between the children of a node which are numbered from 1 to num. The function $index(x)$ returns such a number associated with the node x, where the index values are uniquely allocated to nodes in Υ for a given key [34].

In the following we will describe how to satisfy an access tree with attributes and time constraints. Let Γ be a Υ with root r. Γ_x is represented as the subtree of Γ with the root node at x. For the root r of Γ, we denote Γ_r. If a set of attributes S satisfies Γ_x, we denote it as $\Gamma_x(S) = 1$. $\Gamma_x(S)$ is calculated recursively as follows: If x is a non-leaf node, evaluate $\Gamma_{x'}(S)$ for all children x' of the node x. $\Gamma_x(S)$ returns 1 if and only if at least k_x children return 1. If x is a node belongs to the last layer from bottom, then $\Gamma_x(S)$ returns 1 if and only if the current time instant t_x associated with leaf node (attribute) in the access tree belongs to time interval $[t_{L,x}, t_{R,x}]$ associated with the corresponding attribute x in the ciphertext, that is $t_x \in [t_{L,x}, t_{R,x}]$.

3 Concepts and Models

In this section, some concepts are first described, and then the system model, formal model and security model of the KP-TSABE scheme are presented.

3.1 Concepts

To form a basis for the KP-TSABE scheme, we introduce the following concepts [3].

(1) *Authorization period*. It is a time interval predefined by a data owner starting from the desired release time and ending at the expiration time. The ciphertext is associated with this interval; the user can construct the decryption key only when the time instant is within this interval.

(2) *Expiration time*. It is a threshold time instant predefined by the owner. The shared data can only be accessed by the user before this time instant, because the shared data will be self-destructed after expiration.

(3) *Full lifecycle*. It is a time interval from the creation of the shared data, authorization period to expiration time. This paper provides full lifecycle privacy protection for shared data in cloud computing.

3.2 System model of KP-TSABE

In our system, we mainly focus on how to achieve fine-grained access control during the authorization period of the shared data in cloud and how to implement self-destruction after expiration. Specifically, we define the system model by dividing the KP-TSABE scheme into the following six entities as shown in Fig.1.

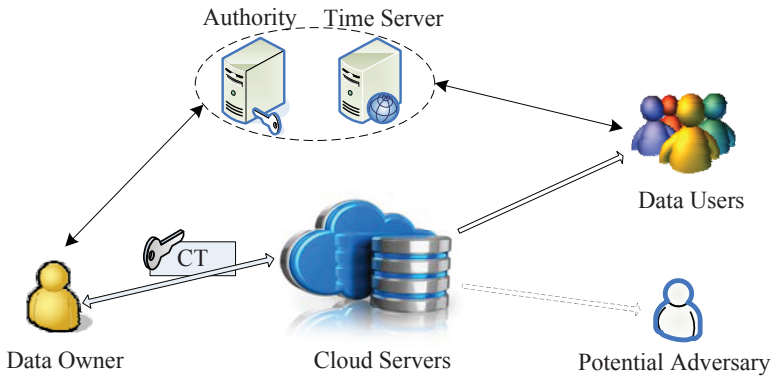

Fig. 1. System model of the KP-TSABE scheme

(1) *Data Owner*. Data owner can provide data or files that contain some sensitive information, which are used for sharing with his/her friends (data users). All these shared data are outsourced to the cloud servers to store.

(2) *Authority*. It is an indispensable entity which is responsible for generating, distributing and managing all the private keys, and is trusted by all the other entities involved in the system.

(3) *Time Server*. It is a time reference server without any interaction with other entities involved in the system. It is responsible for a precise release time specification.

(4) *Data Users*. Data users are some peoples who passed the identity authentication and access to the data outsourced by the data owner. Notice that, the shared data can only be accessed by the authorized users during its authorization period.

(5) *Cloud Servers*. It contains almost unlimited storage space which is able to store and manage all the data or files in the system. Other entities with limited storage space can store their data to the cloud servers.

(6) *Potential Adversary*. It is a polynomial time adversary and described in the security model of the KP-TSABE scheme in section 3.4.

3.3 Formal Model of KP-TSABE

The KP-TSABE scheme can be described as a collection of the following four algorithms: *Setup*, *Encrypt*, *KeyGen*, and *Decrypt*.

Setup$(1^\kappa, \mathcal{U})$: This algorithm is run by the Authority and takes as input the security parameter 1^κ and attribute universe $\mathcal{U}$, generates system public parameters $params$ and the master key MSK. The Authority publishes $params$ and keeps MSK secret to itself.

Encrypt$(M,\ params,\ S,\ T_S)$: Given the public parameters $params$, the shared message M which the owner wants to encrypt, the attribute set S and the set of time intervals T_S in which every element in T_S is associated with a corresponding attribute in S. This algorithm generates the ciphertext CT which is associated with the fuzzy attribute set S.

KeyGen$(\mathrm{MSK}, \Upsilon, T')$: This algorithm takes as input the master key MSK, the access tree Υ and the time set T'. Every attribute x in Υ is associated with a time instant $t_x \in T'$. It outputs a private key SK which contains Υ.

Decrypt$(\mathrm{CT}, \mathrm{SK})$: This algorithm takes as input the ciphertext CT and the private key SK. When a set of time-specific attributes satisfies Υ, it is able to decrypt the ciphertext and return the plaintext M.

3.4 Security model for KP-TSABE

KP-TSABE security is defined by the following games between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

Init. The adversary $\mathcal{A}$ declares the attribute set γ^* that he wishes to be challenged upon.

Setup. The challenger $\mathcal{B}$ runs the *Setup* algorithm to generate $params$ and MSK. The $params$ is given to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ generates repeated private keys corresponding to many access structures $\mathbb{A}_j$ and time instants in which none of these attribute structures satisfies that $\gamma^* \in \mathbb{A}_j$.

Challenge. $\mathcal{A}$ submits two equal-length messages M_0, M_1, and a challenge attribute set γ^*. $\mathcal{B}$ flips a random coin b, and encrypts M_b under γ^*. The ciphertext CT^* is given to $\mathcal{A}$.

Phase 2. Same as in phase 1.

Guess. $\mathcal{A}$ outputs a guess b' of b.

The advantage of $\mathcal{A}$ in this game is defined as $\mathrm{Adv}_{\mathcal{A}} = \Pr[b' = b] - \frac{1}{2}$.

Definition 3. *The KP-TSABE scheme is indistinguishable secure against selective attribute chosen plaintext attack if all polynomial time adversaries have at most a negligible advantage in the above game.*

4 Construction of the KP-TSABE Scheme

In this section, we will present the KP-TSABE scheme in two levels: *System Level* and *Algorithm Level*. System level describes the implementations of the upper operations, while algorithm level mainly focuses on the concrete details of the underlying algorithms which are invoked by system level operations [3]. The details of these two levels are described as follows.

4.1 System descriptions of the KP-TSABE

1) System setup

In the system initialization phase, a data owner chooses a large security parameter κ and attribute universe $\mathcal{U}$, and invokes the algorithm **Setup**$(1^{\kappa}, \mathcal{U})$ belonging to the algorithm level to generate system parameters *params* and master key MSK.

2) Encryption with time constraint

The data owner chooses an attribute set S for the shared message M and defines a time interval set T_S for S. Then, the data owner invokes the algorithm **Encrypt**(M, $params$, S, T_S) to encrypt M to its ciphertext CT, which is associated with the set S and T_S. Finally, CT is sent to cloud servers.

3) Fine-grained access control during the authorization period

When a user wants to access the shared data M during its authorization period, he must pass the identity authentication and should perform the following processes:

Firstly, the current time instant t_x is provided by the time server with $t_x \in T'$, which is associated with each attribute x. If $T' \subseteq T_S$ and the attribute set of the user matches the access tree Υ. Then, the Authority runs the algorithm **KeyGen**$(\mathrm{MSK}, \Upsilon, T')$ to generate the private key SK and sends it to the user. Once the user received the SK, he will get the CT from the cloud servers and invokes the algorithm **Decrypt**$(\mathrm{CT}, \mathrm{SK})$ to decrypt CT to obtain the shared data M.

Because each attribute x is associated with a current time instant t_x, if and only if $t_x \in T_S$ and attribute set matches Υ, the user can obtain the correct private key SK to decrypt CT. Therefore, the KP-TSABE scheme allows for extremely flexible implementation of fine-grained access control through combining different attributes with corresponding time intervals.

4) Data self-destruction after expiration

Once the current time instant t_x becomes after the threshold value (expiration time) of the valid time interval $t_{R,x}$, the user cannot obtain the true private key SK. Therefore, the cyphertext CT is not able to be decrypted in polynomial time, facilitating the self-destruction of the shared data after expiration.

4.2 Algorithm constructions of the KP-TSABE

In this section, we will give the concrete constructions of the KP-TSABE scheme. In order to implement fine-grained access control, we associate every attribute in the attribute set with a time interval (authorization period). The attribute is valid if and only if the current time instant is in this time interval. Only if the valid attribute in the ciphertext satisfies the access tree in the key, the algorithm can decrypt the message correctly. The algorithm level of the KP-TSABE scheme includes four algorithms: *Setup*, *Encrypt*, *KeyGen*, and *Decrypt*.

Setup$(1^{\kappa}, \mathcal{U})$: Let $\mathbb{G}$ be a bilinear group of prime order p, let g be a generator of $\mathbb{G}$, and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}'$ be a bilinear map. In addition, let T be the maximum time in the system that is provided by the time server which satisfies $|T| = n'$. We choose a big security parameter κ, and define the universe of all attributes $\mathcal{U} = \{1, \cdots, n\}$. Then, we choose y from $\mathbb{Z}_p$ randomly and set $g_1 = g^y$. Meanwhile, we choose $g_2, u'_{1,1}, \cdots, u'_{n,1}, u'_{1,2}, \cdots, u'_{n,2}, u_1, \cdots, u_T \in \mathbb{G}$ randomly.

The public parameters is published as:

$$params = \{g, g_1, g_2, \{\forall i = 1 : n, u'_{i,1}, u'_{i,2}\}, \{\forall j = 1 : T, u_j\}\}.$$

The master key MSK is: $MSK = g_2^y$.

Encrypt(M, S_{att}, $params$, T'): To encrypt a message M under a set of attributes S_{att} with every

attribute $i \in S_{att}$, where i is constrained by a time interval $T_i' \in [t_{m_{L,i}}, t_{m_{R,i}}]$, choose a random value $s \in \mathbb{Z}_p$, define $c_{L,i}$ as index, let $c_{L,i} = n' - m_{L,i}$ and publish the ciphertext as:

$$CT = \{C_M = M \cdot e(g, g_2)^{sy}, g^s, S_{att}, \{E = (u'_{i,1} \prod_{j=1}^{m_{R,i}+1} u_j^{t_j})^s, E' = (u'_{i,2} \prod_{j=1}^{c_{L,i}} u_j^{T-t_j})^s, T_i'\}_{i \in S_{att}}\}.$$

KeyGen$(params, MSK, \Upsilon, T_k)$: This algorithm inputs the public parameters $params$, the master key $MSK \in \mathbb{G}'$, the access tree Υ and the time instant set T_k in which the element of T_k is associated with the leaf node of Υ. It outputs a private key SK also associated with Υ.

The algorithm proceeds as follows. First, it chooses a polynomial q_x for each node x except the leaves nodes in Υ. These polynomials are chosen in a top-down manner, starting from the root node r, which is described as follows.

For non-leaf node x in Υ, set the degree d_x of the polynomial q_x and its threshold value k_x satisfying $d_x = k_x - 1$. For the root node r, set $q_r(0) = y$ and choose other d_r points randomly to completely define the polynomial q_r. For any other node x, set $q_x(0) = q_{parent(x)}(index(x))$ and pick d_x other points of q_x randomly to define it completely [8].

We define a leaf node $x \in S_Y$ in the tree as an attribute which is constrained by a time instant t'_{n_x}. S_Y denotes as the leaf node set of Υ. The algorithm randomly chooses $r_x, r'_x \in \mathbb{Z}_p$, defines n_x be the index which lets $c_x = n' - n_x$, computes and gives the following secret value d to the user:

$$d = \{D_{x,1}, D_{x,2}, g^{r_x}, g^{r'_x}, u_{n_x+2}^{r_x} \cdots, u_T^{r_x}, u_{c_x+1}^{r'_x} \cdots, u_T^{r'_x}, t_{n_x}\}_{x \in S_Y},$$

where

$$D_{x,1} = g_2^{q_x(0)+\tau_x} (u'_{i,1} \prod_{j=1}^{n_x+1} u_j^{t_j})^{r_x},$$
$$D_{x,2} = g_2^{-\tau_x} (u'_{i,2} \prod_{j=1}^{c_x} u_j^{T-t_j})^{r'_x}.$$

Decrypt(CT, SK): The decryption procedure is a recursive algorithm which is from bottom to up. In order to decrypt the ciphertext successfully, the valid attribute set (for $x \in S_{att}$, $t_{n_x} \in [t_{m_{L,x}}, t_{m_{R,x}}]$) should satisfy Υ, where the node x is a leaf node, $[t_{m_{L,x}}, t_{m_{R,x}}]$ is the time interval associated with x belongs to the ciphertext, while t_{n_x} associated with x belonging to the private key.

For the leaf node x: If $t_{n_x} \notin [t_{m_{L,x}}, t_{m_{R,x}}]$, the decryption algorithm simply outputs $\perp$. Otherwise, the algorithm chooses random $r''_x, r'''_x \in \mathbb{Z}_p$ and calculates:

$$d_{upp1} = \{a_0, g^{r_{R,x}} \cdot g^{r''_x}, u_{m_{R,x}+2}^{r_{R,x}} \cdot u_{m_{R,x}+2}^{r''_x} \cdots, u_T^{r_{R,x}} \cdot u_T^{r''_x}\},$$

$$d_{upp2} = \{b_0, g^{r_{L,x}} \cdot g^{r'''_x}, u_{c_{L,x}+1}^{r_{L,x}} \cdot u_{c_{L,x}+1}^{r'''_x} \cdots, u_T^{r_{L,x}} \cdot u_T^{r'''_x}\},$$

where

$$a_0 = D_{x,1}(u'_{i,1} \prod_{j=n_x+1}^{m_{R,x}+1} u_j^{t_j})^{r_{R,x}} (u'_{i,1} \prod_{j=1}^{m_{R,x}+1} u_j^{t_j})^{r''_x}$$
$$= g_2^{q_x(0)+\tau_x} (u'_{i,1} \prod_{j=1}^{m_{R,x}+1} u_j^{t_j})^{r_{R,x}+r''_x}.$$

$$b_0 = D_{x,2}(u'_{i,2} \prod_{j=c_x}^{c_{L,x}} u_j^{T-t_j})^{r_{L,x}} (u'_{i,2} \prod_{j=1}^{c_{L,x}} u_j^{T-t_j})^{r'''_x}$$
$$= g_2^{-\tau_x} (u'_{i,2} \prod_{j=1}^{c_{L,x}} u_j^{T-t_j})^{r_{L,x}+r'''_x}.$$

Then, the algorithm calculates as follows:

$$DN = \frac{e(g^s, a_0) \cdot e(b_0, g^s)}{e(E, g^{r_{R,x}+r''_x}) \cdot e(g^{r_{L,x}+r'''_x}, E')} = e(g, g_2)^{sq_x(0)}.$$

We now consider a non-leaf node x with all nodes z that are the children of x. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_x \neq \perp$. If no such set exists, we say that the node is not satisfied and the function returns $\perp$.

Otherwise, the algorithm calculates:

$$F_x = \prod_{c \in S_x} F_c^{\Delta_{i,S'_x}(0)} = \prod_{c \in S_x} \left(e(g, g_2)^{s \cdot q_c(0)}\right)^{\Delta_{i,S'_x}(0)}$$
$$= \prod_{c \in S_x} \left(e(g, g_2)^{s \cdot q_{\text{paraent}(c)}(0)}\right)^{\Delta_{i,S'_x}(0)}$$
$$= \prod_{c \in S_x} e(g, g_2)^{s \cdot q_x(i) \cdot \Delta_{i,S'_x}(0)} = e(g, g_2)^{s \cdot q_x(0)}.$$

If we invoke the function on the root node r of the tree Υ, and if the tree is satisfied by S_Y, we set:

$$\Omega = e(g, g_2)^{s \cdot q_r(0)} = e(g, g_2)^{s \cdot y}.$$

Therefore, we can use Ω to decrypt C_M to obtain the shared message M.

5 Security of the KP-TSABE

Theorem 1. *If the decisional l-BDHI assumption holds in $(\mathbb{G}, \mathbb{G}', e)$, then the KP-TSABE scheme is indistinguishable secure against selective attribute set chosen plaintext attack.*

Proof. Suppose $\mathcal{A}$ has advantage δ in attacking the KP-TSABE system. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the l-BDHI assumption in $\mathbb{G}$.

For a random generator $g \in \mathbb{G}$ and $y \in \mathbb{Z}_p$, we define $g_i = g^{y^i} \in \mathbb{G}$. $\mathcal{B}$ is given a random tuple

$C' = (g, h, y_1, \cdots, y_l, K)$ as input. The C' is either sampled from $\mathcal{P}_{BDHI}$ (where $K = e(g,h)^{y^{l+1}}$) or from $\mathcal{R}_{BDHI}$ (where K is uniform and independent in $\mathbb{G}'$). $\mathcal{B}$ outputs 1 when the input tuple is sampled from $\mathcal{P}_{BDHI}$ and 0 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}$ as follows:

Init: The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses the set of attributes S^*_{att} it wishes to be challenged upon. For $att^*(j) \in S^*_{att}$, $att^*(j)$ is constrained by the time interval $[T^*_n, T^*_m]$, where $T^*_n = (t^*_1, \cdots, t^*_n) \in (\mathbb{Z}^*_p)^j$ of depth $n \leq l$, $T^*_m = (t^*_1, \cdots, t^*_m) \in (\mathbb{Z}^*_p)^j$ of depth $m \leq l$ that it intends to attack. If $n < l (m < l)$, then $\mathcal{B}$ pads $T^*_n (T^*_m)$ with $l - n (l - m)$ zeros on the right to make $att^*(j)$ a vector of length l.

Setup: To generate the system parameters, $\mathcal{B}$ picks a random $\gamma_j \in \mathbb{Z}_p$ and sets $g_1 = y_1 = g^y$ and $g_2 = y_l \cdot g^\gamma = g^{\gamma + y^l}$. Next, $\mathcal{B}$ picks random $\gamma_1, \cdots, \gamma_l \in \mathbb{Z}_p$ and sets $u_j = g^{\gamma_j}/y_{l-j+1}$ for $j = 1, \cdots, l$. $\mathcal{B}$ also picks random $\delta_x, \eta_x \in \mathbb{Z}_p$ and sets $u'_{x,1} = g^{\delta_x} \prod_{i=1}^{l} y_{l-i+1}^{t^*_i}$ and $u'_{x,2} = g^{\eta_x} \prod_{i=1}^{l} y_{l-i+1}^{l+t^*_i}$.

Finally, $\mathcal{B}$ gives $\mathcal{A}$ the public parameters $params = (g, g_1, g_2, u'_{x,1}, u'_{x,2}, u_1, \cdots, u_l)$. The master key corresponding to the public parameters $g_2^y = g^{y(y^l+\gamma)}$ which is unknown to $\mathcal{B}$ since $\mathcal{B}$ does not have y_{l+1}. It then gives the public parameters to $\mathcal{A}$.

Phase1: $\mathcal{A}$ adaptively makes requests for the private keys. We first define the following two procedures: PolySat and PolyUnsat.

PolySat(Γ_x, S, λ_x): This procedure sets up the polynomials for the nodes of an access sub-tree with satisfied root node, that is, $\Gamma_x(S) = 1$. The procedure takes an access tree Γ_x (with root node x) as input along with a set of attributes S and an integer $\lambda_x \in \mathbb{Z}_p$. It first sets up a polynomial q_x of degree d_x for the root node x. It sets $q_x(0) = \lambda_x$ and then sets rest of the points randomly to completely fix q_x. If the algorithm reaches the leaf node, it sets $q_k(0) = a_k^{\omega^*_k + 1} b$. Now it sets polynomials for each child node x' of x by invoking the procedure PolySat$(\Gamma_{x'}, S, q_x(\text{index}(x'))$. Notice that in this way, $q_{x'} = q_x(\text{index}(x'))$ for each child node x' of x.

PolyUnsat$(\Gamma_x, S, g^{\lambda_x})$: This procedure sets up the polynomials for the nodes of an access tree with unsatisfied root node, that is, $\Gamma_x(S) = 0$. The procedure takes an access tree Γ_x (with root node x) as input along with a set of attributes S and an element $g^{\lambda_x} \in \mathbb{G}$ (where $\lambda_x \in \mathbb{Z}_p$). It first defines a polynomial q_x of degree d_x for the root node such that $q_x(0) = \lambda_x$. Because $\Gamma_x(S) = 0$ no more than d_x children of x are satisfied. Let $h_x \leq d_x$ be the number of satisfied children of x. For each satisfied child x' of x, the procedure chooses a random point $\lambda_{x'} \in \mathbb{Z}_p$ and sets $q_x(\text{index}(x')) = \lambda_{x'}$. It then fixes the remaining $d_x - h_x$ points of q_x randomly to completely define q_x. Now the algorithm recursively defines polynomials for the rest of the nodes in the tree as follows. For each child node x' of x, the algorithm invokes:

-PolySat$(\Gamma_{x'}, S, q_x(\text{index}(x'))$. If x' is a satisfied node. Notice that $q_x(0)$ is known in this case.

-PolyUnsat$(\Gamma_{x'}, S, g^{q_x(\text{index}(x'))})$, if x' is not a satisfied node. Notice that only $g^{q_x(\text{index}(x'))}$ can be obtained by interpolation as only $g^{q_x(0)}$ is known in this case.

To give keys for access structure Γ, simulator $\mathcal{B}$ first runs PolyUnsat(Γ, S, A) to define a polynomial q_x for each node x of Γ. Notice that for each leaf node x of Γ, we know q_x completely if it is satisfied; If x is not satisfied, then at least $g^{q_x(0)}$ is known (in some cases q_x might be known completely). Furthermore, $q_r(0) = y$.

The query for the private key for the leaf node x which is associated with T_n, where $T_n = (t_1, \cdots, t_n)$. We define $T'_n = (t_1, \cdots, T - t_n)$. To respond to the query, $\mathcal{B}$ first generates a private key for the time $T_j = (t_1, \cdots, t_j)$ (the restriction is that $t_j \neq t^*_j$ for $t_j \in \{t_1, \cdots, t_m\}$) and $T'_j = (t_1, \cdots, T - t_j)$ where $t_j \leq T$(the restriction is that $t_j \neq t^*_j$ for $t_j \in \{t_1, \cdots, T - t_n\}$). Then it constructs for attribute x for the time $T_n = (t_1, \cdots, t_j, \cdots, t_n)$ and $T'_n = (t_1, \cdots, T - t_j, \cdots, T - t_n)$

To generate the private key for the leaf node x which is constrained by $(t_1, \cdots, t_j)$, $\mathcal{B}$ picks a random r_x in $\mathbb{Z}_p$. We pose $r_x = \tilde{r}_x + \frac{y^j}{(t_{j+1} - t^*_{j+1})} \in \mathbb{Z}_p$. We show that $\mathcal{B}$ can calculate all elements of the private key given the values at its disposal. To generate the first component of the private key for the attribute x, we observe that:

$$(u'_{x,1} \prod_{i=1}^{j+1} u_i^{t_i})^{r_x} = \left(g^{\delta_x + \sum_{i=1}^{j+1} t_i \gamma_i} \cdot \prod_{i=1}^{j} y_{l-i+1}^{(t^*_i - t_i)} \cdot y_{l-j}^{t^*_{j+1} - t_{j+1}} \prod_{i=j+2}^{l} y_{l-i+1}^{t^*_i} \right)^{r_x}.$$

Let Z_x represent the product of the first, second and fourth terms. We have

$$Z_x = \left(g^{\delta_x + \sum_{i=1}^{j+1} t_i \gamma_i} \cdot \prod_{i=1}^{j} y_{l-i+1}^{(t^*_i - t_i)} \cdot \prod_{i=j+2}^{l} y_{l-i+1}^{t^*_i} \right)^{r_x}.$$

Next, we observe the third term is:

$$y_{l-j}^{(t^*_{j+1}-t_{j+1})r_x} = y_{l-j}^{(t^*_{j+1}-t_{j+1})\tilde{r}_x} \cdot y_{l-j}^{(t^*_{j+1}-t_{j+1})\frac{y^j}{(t_{j+1}-t^*_{j+1})}} = y_{l-j}^{(t^*_{j+1}-t_{j+1})\tilde{r}_x}/y_l.$$

Hence, the first component of the private key for the attribute x is:

$$g_2^{q_x(0)+\tau_x}(u'_{x,1}\prod_{i=1}^{j+1}u_i^{t_i})^{r_x} = g^{(q_x(0)+\tau_x)(y^l+\gamma)}(u'_{x,1}\prod_{i=1}^{j+1}u_i^{t_i})^{r_x} = y_{l-j}^{(t^*_{j+1}-t_{j+1})\tilde{r}_x} g^{(q_x(0)+\tau_x)(1+\gamma/y_l)} Z_x.$$

Similarly, we construct the second component of the private key for the attribute x. We pose $r'_x = \tilde{r}'_x + \frac{y^{k-1}}{(t_k - t^*_k)}$, and we observe that:

$$(u'_{x,2}\prod_{i=1}^{k}u_i^{T-t_i})^{r'_x} = \left(g^{\eta_x+\sum_{i=1}^{k}(T-t_i)\gamma_i}\cdot\prod_{i=1}^{k-1}y_{l-i+1}^{(t^*_i-t_i)}\right)^{r'_x}\cdot\left(y_{l-k+1}^{t^*_k-t_k}\cdot\prod_{i=k+1}^{l}y_{l-i+1}^{T-t^*_i}\right)^{r'_x}.$$

Let Z'_x represent the product of the first, second and fourth terms. That is

$$Z'_x = \left(g^{\eta_x+\sum_{i=1}^{k}(T-t_i)\gamma_i}\cdot\prod_{i=1}^{k-1}y_{l-i+1}^{(t^*_i-t_i)}\cdot\prod_{i=k+1}^{l}y_{l-i+1}^{T+t^*_i}\right)^{r'_x}.$$

Next, we observe the third term is:

$$y_{l-k+1}^{(t^*_k-t_k)r'_x} = y_{l-k+1}^{(t^*_k-t_k)\tilde{r}'_x}\cdot y_{l-k+1}^{(t^*_k-t_k)\frac{y^{k-1}}{(t_k-t^*_k)}} = y_{l-k+1}^{(t^*_k-t_k)\tilde{r}_x}/y_l.$$

Hence, the second component of the private key for the attribute x is constructed as:

$$g_2^{-\tau_x}(u'_{i,2}\prod_{i=1}^{k}u_i^{T-t_i})^s = y_{l-k+1}^{(t^*_k-t_k)\tilde{r}_x} g^{(-\tau_x)(1+\gamma/y_l)} Z'_x.$$

For all the leaf nodes, y_l can be canceled and all terms in the expression are known to $\mathcal{B}$. Also, $\mathcal{B}$ can also calculate the component $g^{r_x}, g^{r'_x}, u_{j+2}^{r_x}\cdots, u_l^{r_x}, u_{k+1}^{r'_x}\cdots, u_l^{r'_x}$. $\mathcal{B}$ uses this leaf node component of the private key to derive a private key for the T_n, T'_n and gives the result to $\mathcal{A}$.

Challenge: When $\mathcal{A}$ assures that Phase 1 is over, $\mathcal{A}$ submits two challenge messages $M_0, M_1 \in \mathbb{G}$ to the simulator $\mathcal{B}$. $\mathcal{B}$ then flips a fair binary coin, τ, and returns an encryption of M_τ. The ciphertext is outputted as:

$$CT = (M_\tau \cdot K \cdot e(y_1, h^\gamma), h, \{h^{\delta_i+\sum_{j=1}^{l}t^*_j\gamma_j}, h^{\eta_i+\sum_{j=1}^{l}(T-t^*_j)\gamma_j}, T_i^*\}_{i\in S_{att}}).$$

First note that if $h = g^c$ (for some unknown c in $\mathbb{Z}_p$) then,

$$h^{\delta_i+\sum_{j=1}^{l}t^*_j\gamma_j} = (g^{\delta_i}\prod_{j=1}^{l}y_{l-j+1}^{t^*_j}\cdot\prod_{j=1}^{l}(g^{\gamma_j}/y_{l-j+1})^{t^*_j})^c = (u'_{i,1}\prod_{j=1}^{l}u_j^{t^*_j})^c,$$

$$h^{\eta_i+\sum_{j=1}^{l}(T-t^*_j)\gamma_j} = (g^{\eta_i}\prod_{j=1}^{l}y_{l-j+1}^{T-t^*_j}\cdot\prod_{j=1}^{l}(g^{\gamma_j}/y_{l-j+1})^{T-t^*_j})^c = (u'_{i,2}\prod_{j=1}^{l}u_j^{T-t^*_j})^c,$$

$$e(g,h)^{y^{l+1}}\cdot e(y_1,h^\gamma) = (e(y_1,y_l)e(y_1,g^\gamma))^c = e(y_1,y_l g^\gamma)^c = e(g_1,g_2)^c.$$

If $K = e(g,h)^{y^{l+1}}$ is a valid l-BDHI tuple, then the challenge CT is a valid encryption of M_τ. Otherwise, the K is uniform and independent in $\mathbb{G}'$ (the input tuple is sampled from $\mathcal{R}_{BDHI}$), the challenge CT is independent of τ in $\mathcal{A}$'s view.

Guess. $\mathcal{A}$ will eventually outputs a guess τ' of τ. If $\tau = \tau'$, then $\mathcal{B}$ outputs 1 to indicate that $K = e(g,h)^{y^{l+1}}$. Otherwise, it outputs 0 to indicate that it believes K is a random group element in $\mathbb{Z}_p$.

When K is a tuple, the simulator $\mathcal{B}$ gives a perfect simulation so we have that

$$\Pr\left[\mathcal{B}\left(\mathbf{y}, K = e(g,h)^{y^{l+1}}\right) = 0\right] = \frac{1}{2} + \mathrm{Adv}_{\mathcal{A}}.$$

When K is a random group element, the message M_τ is completely hidden from $\mathcal{A}$ and we have $\Pr[\mathcal{B}(\mathbf{y}, K = R) = 0] = \frac{1}{2}$.

Therefore, $\mathcal{B}$ is able to play l-BDHI game with non-negligible advantage. □

6 Comparison and Analysis

In this section, the comprehensive comparison is first described, and then the theoretical analysis of the KP-TSABE scheme is presented.

6.1 Comprehensive comparison

The KP-TSABE scheme is proved to be secure under the standard model. Therefore, we systematically compare this scheme with the existing self-destruction solutions (e.g., Vanish [23], SSDD [24], ISS [22], and FullPP [3]) from the following aspects, e.g., prerequisite condition, algorithm, resistance on attacks, fine-grained access control, user-defined authorization period, etc. The result of the comprehensive comparison is shown in Table 1.

TABLE 1
Comprehensive comparisons of the security properties

Security properties	Vanish[23]	SSDD[24]	ISS[22]	FullPP[3]	KP-TSABE
Need "no attacks on VDO before it expires"?	YES	YES	YES	NO	No need
Leveraging what kind of algorithm?	Symmetric	Symmetric	IBE	ID-TRE	KP-TSABE
Whether ciphertext is destructed or not?	NO	YES	YES	YES	No need
Whether the key is destructed or not?	YES	YES	YES	YES	No need
Resistance on the traditional cryptanalysis?	NO	YES	YES	YES	YES
Resistance on the Sybil attacks?	NO	NO	YES	YES	–
Resistance on the collusion attack?	–	–	–	–	YES
Supporting fine-grained access control?	NO	NO	YES	YES	YES
Providing full lifecycle privacy protection?	NO	NO	NO	YES	YES
Supporting user-defined time intervals?	NO	NO	NO	Half	YES
Security proof under standard model?	NO	NO	NO	YES	YES

Prerequisite condition. All the schemes of Vanish [23], SSDD [24] and ISS [22] need the ideal assumption "no attacks on VDO before it expires". Since a Sybil adversary is able to crawl sufficient key shares from the DHT network to reconstruct the decryption key. Once the adversary gets the VDO from the cloud servers before it expires, he/she will decrypt it with the reconstructed decryption key to obtain the plaintext. FullPP [3] does not need this ideal assumption because the decryption key is encrypted by the ID-TRE algorithm. Even if the adversary crawls sufficient key shares from the DHT network, he cannnot reconstruct the decryption key since he does not have the ID-TRE private key. KP-TSABE also does not need the ideal assumption because it does not require the DHT network.

Algorithm and resistance on attacks. Since both Vanish [23] and SSDD [24] only use symmetric encryption to encrypt the sensitive message, they bring complex key management and cannot achieve fine-grained access control for different users with different attributes. Vanish sends the entire ciphertext to the cloud server, so it cannot resist against the traditional cryptanalysis. Since the SSDD scheme distributes a part of the ciphertext and the decryption key to the DHT network, both of which will be self-destructed after expiration, so the cloud server stores incomplete ciphertext. Therefore, SSDD can resist against the traditional cryptanalysis. However, Vanish and SSDD cannot resist against the Sybil attackers who can continually crawl the key shares from the DHT network to recover the decryption key [25]. In contrast, both ISS [22] and FullPP [3] can not only resist against the traditional cryptanalysis and the Sybil attacks but also implement flexible access control because of the IBE and ID-TRE algorithms. KP-TSABE does not have the problem of the Sybil attacks because there is no use of the DHT network. Furthermore, it can provide fine-grained access control through combining different attributes with variance time intervals.

User-defined authorization period. Vanish [23], SSDD [24], ISS [22] and FullPP [3] leverage the DHT network to store the key shares or the hybrid ciphertext shares, which are self-discarded by the DHT nodes after a period of time. So the expiration time is limited by the update period of the DHT network and it cannot be controlled by the sensitive data owner. Better than those schemes, in the KP-TSABE scheme, every attribute in the attribute set associated with the ciphertext is matched with a time interval, which is the authorization period of the sensitive data and is predefined by the data owner. Therefore, the authorization period and the expiration time are not limited by the system constraint, but flexibly to be defined by the owner.

Security proof. Vanish [23], SSDD [24], and ISS [22] do not provide the security proof. The ID-TRE in the FullPP scheme is proved to be secure under the Bilinear Diffie-Hellman (BDH) assumption. Furthermore, the KP-TSABE scheme is proved to be secure under the standard model with the decision l-Expanded BDHI assumption to resist against the traditional cryptanalysis and the collusion attack.

In conclusion, the KP-TSABE scheme is superior to the existing self-destruction solutions from several security properties.

6.2 Theoretical analysis

In this section, we make the theoretical analysis of the proposed KP-TSABE scheme from the aspects of the computational cost and communication overhead.

1) Computational cost

In this analysis, we focus on the most time-consuming operations, paring and exponentiation conducted in groups $\mathbb{G}$ and $\mathbb{G}'$. Let ψ, ω, and ϖ respectively denote their computation times and let $O(TREE)$ be the computation complexity of the decryption TREE. We let ϑ be the attribute set used in decryption. The computational cost of each procedure in KP-TSABE is summarized in Table 2.

Since the computational cost seems to be expensive, we can alleviate the computational cost by pre-computation. For instance, in the **Encrypt** phase, $e(g,g_2)^y$ can be calculated by $e(g_1,g_2)$. The terms $(u'_{i,1}\prod_{j=1}^{m_{R,i}+1}u_j^{t_j})$ and $(u'_{i,2}\prod_{j=1}^{c_{L,i}}u_j^{T-t_j})$ can also be precalculated by choosing proper t_j, $m_{R,i}$ and $c_{L,i}$. Thus, the total computational cost can be reduced to $1\omega+1\varpi+2|S_{att}|\omega$. Similarly, in **KeyGen** phase, the terms $(u'_{i,1}\prod_{j=1}^{n_x+1}u_j^{t_j})$ and $(u'_{i,2}\prod_{j=1}^{c_x}u_j^{T-t_j})$ can also be pre-computed. Then, the computational cost in key generation is $|S_Y|(2|T|+5-n_x-c_x)\omega$. In **Decrypt** stage, $(u'_{i,1}\prod_{j=n_x+1}^{m_{R,x}+1}u_j^{t_j})$ and $(u'_{i,1}\prod_{j=1}^{m_{R,x}+1}u_j^{t_j})$ in a_0 can be preprocessed by chosen proper n_x and $m_{R,x}$. The computation of b_0 can be made similarly. The final computation cost is $O(TREE)\varpi+|\vartheta|[4\psi+(4T+6-2m_{R,x}-2c_{L,x})\omega]$.

2) Communication overhead

Table 3 analyzes the communication overhead of the KP-TSABE scheme through calculating the bit-length of the group elements included in the ciphertext and the key generation. It does not include the set of the attributes S_{att}, tree access policy, and the definition of the time instant, because their costs are negligible if compared with the key generation and ciphertext. We let $|\mathbb{G}|$ and $|\mathbb{G}'|$ be the bit-length of the element in $\mathbb{G}$ and $\mathbb{G}'$, respectively. In the key generation phase, the authority has to generate the keys for each user. The size of the private key is $|S_Y|(2|T|+3-n_x-m_x)|\mathbb{G}|$. After encrypting, the data owner needs to upload the encrypted message to the cloud server. The size of ciphertext is $1|\mathbb{G}'|+(1+2|S_{att}|)|\mathbb{G}|$.

Although the computational cost seems to be expensive, optimizations are made to alleviate the computational cost. After the optimization, the final computational cost is located in a reasonable range. In compensation to this, the proposed KP-TSABE scheme provides a big advantage by supporting user-defined time-specific authorization, fine-grained access control and data secure self-destruction, which are not well satisfied by the existing schemes.

7 Conclusions

With the rapid development of versatile cloud services, a lot of new challenges have emerged. One of the most important problems is how to securely delete the outsourced data stored in the cloud severs. In this paper, we proposed a novel KP-TSABE scheme which is able to achieve the time-specified ciphertext in order to solve these problems by implementing flexible fine-grained access control during the authorization period and time-controllable self-destruction after expiration to the shared and outsourced data in cloud computing. We also gave a system model and a security model for the KP-TSABE scheme. Furthermore, we proved that KP-TSABE is secure under the standard model with the decision l-Expanded BDHI assumption. The comprehensive analysis indicates that the proposed KP-TSABE scheme is superior to other existing schemes.

Acknowledgments

This work is supported by the Key Program of NSFC-Guangdong Union Foundation under grant No.U1135002, the National Natural Science Foundation of China under grant No.61402109 and No.61370078, the Changjiang Scholars and Innovative Research Team in University under grant No.IRT1078, the Fundamental Research Funds for the Central Universities under grant No.JB142001-12. We thank the reviewers for helpful comments.

References

[1] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditingfor shared data in the cloud," *Cloud Computing, IEEE Transactions on*, vol. 2, no. 1, pp. 43–56, 2014.

[2] J. Xiong, Z. Yao, J. Ma, X. Liu, Q. Li, and J. Ma, "Priam: Privacy preserving identity and access management scheme in cloud," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 8, no. 1, pp. 282–304, 2014.

TABLE 2
Computational cost

	KP-TSABE
Setup	2ω
Encryption	$1\psi + 1\omega + 1\varpi + \lvert S_{att}\rvert(m_{R,i} + 3 + c_{L,i})\,\omega$
Key generation	$\lvert S_Y\rvert(2\lvert T\rvert + 6)\omega$
Decryption	$O\,(TREE)\,\varpi + \lvert\vartheta\rvert[4\psi + (4T + 11 - n_x - c_x)\,\omega]$

TABLE 3
Communication overhead

Communication overhead	KP-TSABE
Data owner to cloud server	$1\lvert\mathbb{G}'\rvert + (1 + 2\lvert S_{att}\rvert)\lvert\mathbb{G}\rvert$
Authority to user	$\lvert S_Y\rvert(2\lvert T\rvert + 3 - n_x - m_x)\lvert\mathbb{G}\rvert$

[3] 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*. [Online]. Available: http://dx.doi.org/10.1007/s12083-014-0295-x

[4] P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: A systematic review," *Cloud Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 142–157, 2013.

[5] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," *Network, IEEE*, vol. 28, no. 4, pp. 46–50, 2014.

[6] X. Liu, J. Ma, J. Xiong, and G. Liu, "Ciphertext-policy hierarchical attribute-based encryption for fine-grained access control of encryption data," *International Journal of Network Security*, vol. 16, no. 4, pp. 351–357, 2014.

[7] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005*, ser. LNCS, vol. 7371. Springer, 2005, pp. 457–473.

[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and Communications Security*. ACM, 2006, pp. 89–98.

[9] A. F. Chan and I. F. Blake, "Scalable, server-passive, user-anonymous timed release cryptography," in *Proceedings of the International Conference on Distributed Computing Systems*. IEEE, 2005, pp. 504–513.

[10] K. G. Paterson and E. A. Quaglia, "Time-specific encryption," in *Security and Cryptography for Networks*. Springer, 2010, pp. 1–16.

[11] Q. Li, J. Ma, R. Li, J. Xiong, and X. Liu, "Large universe decentralized key-policy attribute-based encryption," *Security and Communication Networks*, 2014. [Online]. Available: http://dx.doi.org/10.1002/sec.997

[12] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 28th IEEE Symposium on Security and Privacy*. IEEE, 2007, pp. 321–334.

[13] L. Cheung and C. C. Newport, "Provably secure ciphertext policy abe," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 456–465.

[14] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," *Public Key Cryptography–PKC 2011*, pp. 53–70, 2011.

[15] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[16] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*. ACM, 2007, pp. 195–203.

[17] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of the 29th IEEE International Conference on Computer Communications*. IEEE, 2010, pp. 1–9.

[18] P. Tysowski and M. Hasan, "Hybrid attribute- and re-encryption-based key management for secure and scalable mobile applications in clouds," *Cloud Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 172–186, 2013.

[19] J. Reardon, D. Basin, and S. Capkun, "Sok: Secure data deletion," in *Proceedings of the 34th IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 1–15.

[20] C. Cachin, K. Haralambiev, H.-C. Hsiao, and A. Sorniotti, "Policy-based secure deletion," in *Proceedings of the ACM Conference Computer and Communications Security*. ACM, 2013, pp. 152–167.

[21] J. Reardon, H. Ritzdorf, D. Basin, and S. Capkun, "Secure data deletion from persistent media," in *Proceedings of the 2013 ACM Conference on Computer and Communications Security*. ACM, 2013, pp. 271–284.

[22] J. Xiong, Z. Yao, J. Ma, F. Li, and X. Liu, "A secure self-destruction scheme with ibe for the internet content privacy," *Chinese Journal of Computers*, vol. 37, no. 1, pp. 139–150, 2014.

[23] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy, "Vanish: Increasing data privacy with self-destructing data," in *Proceedings of the 18th USENIX Security Symposium*, 2009, pp. 299–315.

[24] G. Wang, F. Yue, and Q. Liu, "A secure self-destructing scheme for electronic data," *Journal of Computer and System Sciences*, vol. 79, no. 2, pp. 279–290, 2013.

[25] S. Wolchok, O. S. Hofmann, N. Heninger, E. W. Felten, J. A. Halderman, C. J. Rossbach, B. Waters, and E. Witchel, "Defeating vanish with low-cost sybil attacks against large dhts," in *Proceedings of the 17th Annual Network and Distributed System Security Conference, NDSS*. ISOC, 2010, pp. 1–15.

[26] L. Zeng, S. Chen, Q. Wei, and D. Feng, "Sedas: A self-destructing data system based on active storage framework,"

IEEE Transactions on Magnetics, vol. 49, no. 6, pp. 2548–2554, 2013.

[27] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[28] J. Xiong, Z. Yao, J. Ma, X. Liu, and Q. Li, "A secure document self-destruction scheme: An abe approach," in *Proceedings of the 15th IEEE International Conference on High Performance Computing and Communications*. IEEE, 2013, pp. 59–64.

[29] J. Xiong, Z. Yao, J. Ma, F. Li, X. Liu, and Q. Li, "A secure self-destruction scheme for composite documents with attribute based encryption," *Acta Electronica Sinica*, vol. 42, no. 2, pp. 366–376, 2014.

[30] J. H. Cheon, N. Hopper, Y. Kim, and I. Osipkov, "Provably secure timed-release public key encryption," *ACM Transactions on Information and System Security (TISSEC)*, vol. 11, no. 2, p. 4, 2008.

[31] A. W. Dent and Q. Tang, "Revisiting the security model for timed-release encryption with pre-open capability," in *Proceedings of the Information Security*. Springer, 2007, pp. 158–174.

[32] R. Kikuchi, A. Fujioka, Y. Okamoto, and T. Saito, "Strong security notions for timed-release public-key encryption revisited," in *Proceedings of the Information Security and Cryptology*. Springer, 2012, pp. 88–108.

[33] K. Kasamatsu, T. Matsuda, K. Emura, N. Attrapadung, G. Hanaoka, and H. Imai, "Time-specific encryption from forward-secure encryption," in *Security and Cryptography for Networks*. Springer, 2012, pp. 184–204.

[34] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Advances in Cryptology–EUROCRYPT 2005*. Springer, 2005, pp. 440–456.

[35] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.