TAFC: Time and Attribute Factors Combined Access Control for Time-Sensitive Data in Public Cloud

Jianan Hong, Kaiping Xue, *Senior Member, IEEE*, Yingjie Xue, Weikeng Chen, David S. L. Wei, *Senior Member, IEEE*, Nenghai Yu, and Peilin Hong

Abstract—The new paradigm of outsourcing data to the cloud is a double-edged sword. On the one hand, it frees data owners from the technical management, and is easier for data owners to share their data with intended users. On the other hand, it poses new challenges on privacy and security protection. To protect data confidentiality against the honest-but-curious cloud service provider, numerous works have been proposed to support fine-grained data access control. However, till now, no schemes can support both fine-grained access control and time-sensitive data publishing. In this paper, by embedding timed-release encryption into Ciphertext-Policy Attribute-based Encryption (CP-ABE), we propose a new time and attribute factors combined access control on time-sensitive data for public cloud storage (named TAFC). Based on the proposed scheme, we further propose an efficient approach to design access policies faced with diverse access requirements for time-sensitive data. Extensive security and performance analysis shows that our proposed scheme is highly efficient and satisfies the security requirements for time-sensitive data storage in public cloud.

Index Terms—Cloud storage, access control, time-sensitive data, fine granularity

1 Introduction

Cloud storage service has significant advantages on both convenient data sharing and cost reduction [1], [2]. Thus, more and more enterprises and individuals outsource their data to the cloud to be benefited from this service. However, this new paradigm of data storage poses new challenges on data confidentiality preservation [3], [4]. As cloud service separates the data from the cloud service client (individuals or entities), depriving their direct control over these data [5], the data owner cannot trust the cloud server to conduct secure data access control. Therefore, the secure access control problem has become a challenging issue in public cloud storage.

Ciphertext-policy attribute-based encryption (CP-ABE) [6] is a useful cryptographic method for data access control in cloud storage [7], [8], [9], [10], [11], [12]. All these CP-ABE based schemes enable data owners to realize fine-grained and flexible access control on their own data. However, CP-ABE determines users' access privilege based only on their inherent attributes without any other critical factors, such as the time factor. In reality, the time factor usually plays an important role in dealing with time-sensitive data [13], [14], [15] (e.g., to publish a latest electronic magazine, or to expose a company's future business plan). In these scenarios, both the mechanism of access privilege timed releasing and fine-grained access control should be together taken into account. Let us take the enterprise data exposure for instance: A company usually prepares some important files for different intended users, and these users can gain their access privilege at different time points. For example, the future plan of this company may contain some business secrets. Thus at an early time, the access privilege can be released to the CEO only. Then the managers of some relevant departments could get access privilege at a later time point, when they take responsibility for the plan execution. At last, other employees in some specific departments of the company can access the data to evaluate the completeness of this enterprise plan. When uploading time-sensitive data to the cloud, the data owner wants different users to access the content after different time points. To the outsourced data storage, CP-ABE can characterize different users and provide fine-grained access control. However, to our best knowledge, these schemes cannot support gradual access privilege releasing.

To realize the function of timed releasing, it is necessary to introduce an effective scheme, which will not release the data access privilege to intended users until reaching predefined time points. A trivial solution is to let data owners manually release the time-sensitive data: The owner uploads the encrypted data under different policies at each releasing time such that the intended users cannot access

- *J. Hong, Y. Xue, W. Chen, N. Yu, and P. Hong are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China. E-mail: {hongjn, xyj1108, cwk32}@mail.ustc.edu.cn, {ynh, plhong}@ustc.edu.cn.*
- *K. Xue is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China, and the State Key Laboratory of Information Security, (Institute of Information Engineering), Chinese Academy of Sciences, Beijing 100093, China. E-mail: kpxue@ustc.edu.cn.*
- *D.S.L. Wei is with the Computer and Information Science Department, Fordham University, New York, NY 10458. E-mail: wei@dsm.fordham.edu.*

Manuscript received 22 Aug. 2016; revised 11 Jan. 2017; accepted 6 Mar. 2017. Date of publication 14 Mar. 2017; date of current version 12 Feb. 2020.
Digital Object Identifier no. 10.1109/TSC.2017.2682090

the data until the corresponding time arrives. However, this solution forces the owner to repeatedly upload the different encryption versions of the same data, which puts unnecessary and heavy burden on the data owner.

From the perspective of cryptography, the function of timed access privilege releasing can be achieved by Timed-Release Encryption (TRE). Rivest et al. [16] proposed the first practical TRE algorithm, which has been subsequently introduced into different scenarios [17], [18], [19]. In a TRE-based system, a trust time agent, rather than data owner, can uniformly release the access privilege at a specific time. Some schemes, such as [20], [21], have been proposed to integrate TRE into remote data access control. However, these schemes either lack fine-grained access control or leave an unbearable burden.

How to achieve the capacity of both timed-release and fine-grained access control in cloud storage? A direct but naive method is to handle the time factor as an attribute [20]. However, unbearable number of time-related keys need to be issued to each user at each pre-defined time point, which introduces heavy overhead on both computation and communication. Qin et al. [18] made a preliminary attempt to integrate time with attributes, but it only addresses the issue that the attributes' life period of each user is limited by time. A more practical requirement is that: each user with different attribute set can have different releasing time points for the same file. Unfortunately, Qin's scheme cannot meet this requirement.

In this paper, we propose an efficient time and attribute factors combined access control scheme, named TAFC, for time-sensitive data in public cloud. Our scheme possesses two important capabilities: 1) It inherits the property of fine granularity from CP-ABE; 2) By introducing the trapdoor mechanism, it further retains the feature of timed release from TRE. Note that in TAFC, the introduced trapdoor mechanism is only related to the time factor, and only one corresponding secret needs to be published when exposing the related trapdoors. This makes our scheme highly efficient, which only brings about little overhead to the original CP-ABE based scheme. We should address how to design an efficient access structure for arbitrary access privilege construction with both time and attribute factors, especially when an access policy embeds multiple access privilege releasing time points. As an extension of the previous conference version [22], we give the potential sub-policies for time-sensitive data, and then present an efficient and practical method to construct relevant access structures.

The main contributions of this paper can be summarized as follows:

1) By integrating TRE and CP-ABE in public cloud storage, we propose an efficient scheme to realize secure fine-grained access control for time-sensitive data. In the proposed scheme, the data owner can autonomously designate intended users and their relevant access privilege releasing time points. Besides realizing the function, it is proved that the negligible burden is upon owners, users and the trusted *CA*.
2) We present how to design access structure for any potential timed release access policy, especially embedding multiple releasing time points for different intended users. To the best of our knowledge, we are the first to study the approach to design structures for general time-sensitive access requirements.
3) Furthermore, a rigorous security proof is given to validate that the proposed scheme is secure and effective.

The rest of this paper is organized as follows. We first review some existing work that are related to data access control for time-sensitive data in Section 2. In Section 3, we present the system architecture and state the security model. Section 4 describes main techniques. In Section 5, we give detailed algorithm of our proposed TAFC, and analyze the scheme in terms of its security and performance in Section 6. Section 7 provides an effective method to design access polices for any potential access requirement of time-sensitive data. Finally, we conclude this paper in Section 8.

2 Related Work

Based on various cryptographic primitives, there have been numerous works on secure data sharing in cloud storage. Among these schemes, some aimed at protecting the integrity of the shared data, e.g., [23], [24], [25], and some aimed at protecting the confidentiality and access control of the data, e.g., [8], [9], [10], [26], [27], [28], [29], [30]. In the area of data access control, attribute-based encryption (ABE) [31] is utilized as a basic cryptographic technique. These ABE-based access control schemes, in general, can be divided into two main categories: key-policy ABE (KP-ABE) based schemes [32], such as [33], [34]; and ciphertext-policy ABE based schemes [6], such as [8], [9], [10], [11]. The latter one is more suitable for achieving flexible and fine-grained access control for the public cloud, in which each file is labelled with an access structure, and each user owes a security key embedded with a set of attributes.

However, the existing ABE based schemes do not support the scenario where the access privilege of one file is required to be respectively released to different sets of users after different time points, but needs only one time of the ciphertext upload. A trivial solution is to let the data owner him/herself retrieve the file, re-encrypt it under the new policy, and upload it again when the releasing time arrives. However, such solution brings about heavy burden of both communication and computation overhead on the data owner. Goyal et al. [32] and Yang et al. [35], [36] have proposed policy update methods for KP-ABE based and CP-ABE based schemes respectively. In [32], [35], [36], if the data owner wants to release the access privilege to new sets of users, he/she does not need to re-encrypt and upload the whole file. Taking Yang's scheme [35] as an example, the data owner generates and sends a policy update key to the cloud, and the cloud can re-encrypt the stored file. With the modification of access policy, new sets of users are able to access the file. However, Yang's scheme have just discussed how to update the access structure, but not embedded the time factor into the access structure, which requires that the data owner must be online when implementing policy updating. Therefore, it is desperately needed to devise an efficient scheme, in which the data owner can designate all of the file's future access policies when it is first encrypted.

Towards this challenge, Timed-Release Encryption becomes a promising primitive, in which, a trusted time agent, instead of data owners, uniformly executes the timed-release function. Such notion has been widely intergrated to many scenarios. Yuan et al. [17] makes TRE be integrated to the searchable encryption scheme, in which the intended user is constrained to wait for a particular time to search the outsourced data. The combination of TRE and proxy-encryption were proposed in cloud environment [28], [37]. TRE also helps achieve a conditional oblivious transfer scheme such that the access pattern is exposed after a specific time [19], [38].

In the scenario of data access control for public cloud storage, some schemes that adopt the basic idea of TRE have been proposed [18], [20], [21]. Qin et al. [18] proposed a proxy-encryption scheme for data sharing, where the data access privilege can be accurately distributed to intended users who own a certain attribute set during a specific time period. The proposed scheme can well preserve data confidentiality. However, it cannot satisfy the requirement that users are constrained to access data after particular designated time. Androulaki et al. [20] designed an approach to realize time-sensitive data access control in cloud. However, this approach lacks fine granularity, which leaves the data owners an unbearable burden in a large-scale system. Fan et al. [21] proposed timed-release predicate encryption for cloud computing. However, each file can be labeled with only one time point, which cannot release the access privilege of one file to different intended users at different time.

Some researches have also tried to combine the mechanisms of TRE and CP-ABE, such as [39], [40], to provide a flexible and fine-grained access control for time-sensitive data. Zhu et al. [39] proposed a temporal access control system for cloud storage, in which the cloud server manages the time as a universal clock service. Such construction cannot resist the collusion between cloud server and users. In [40], the authors proposed a time-domain access control system, in which access control takes both user's attribute set and the access time into consideration. Different from [35], [36], this work achieves data access privilege automatically releasing for users without data owner's online participation. However, it introduces heavy extra overhead: The authority needs to generate update keys for all potential attributes each time to implement the time-related function, and the computational complexity increases with the amount of involved attributes.

A more smart scheme is needed to realize fine-grained access control for time-sensitive data in cloud storage.

3 System and Security Model

3.1 System Model

Similar to most CP-ABE based schemes, the system in this paper consists of the following entities: a central authority (*CA*), several data owners (Owner), many data consumers (User), and a cloud service provider (Cloud).

- *The central authority* is responsible to manage the security protection of the whole system: It publishes system parameters and distributes security keys to each user. In addition, it acts as a time agent to maintain the timed-releasing function.
- *The data owner (Owner)* decides the access policy based on a specific attribute set and one or more releasing time points for each file, and then encrypts the file under the decided policy before uploading it.
- *The data consumer (User)* is assigned a security key from *CA*. He/she can query any ciphertext stored in the cloud, but is able to decrypt it only if both of the following constraints are satisfied: 1) His/her attribute set satisfies the access policy; 2) The current access time is later than the specific releasing time.
- *Cloud service provider (Cloud)* includes the administrator of the cloud and cloud servers. The cloud undertakes the storage task for other entities, and executes access privilege releasing algorithm under the control of *CA*.

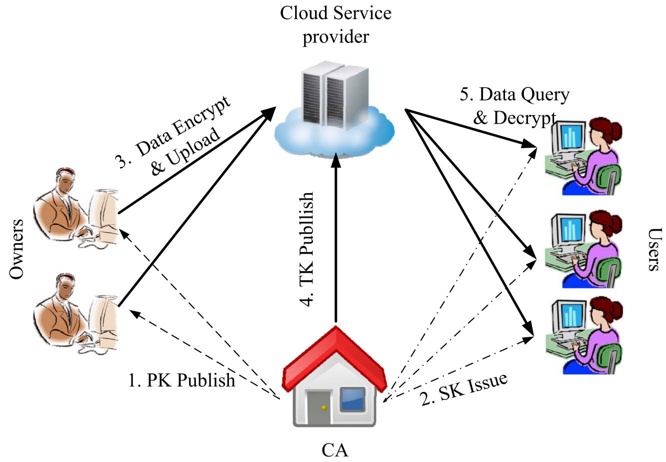


Fig. 1. TAFC architecture and operations.

As depicted in Fig. 1, the ciphertexts are transmitted from owners to the cloud, and users can query any ciphertexts. *CA* controls the system with the following two operations: 1) It issues security keys to each user, according to user's attribute set; 2) At each time point, it publishes a time token (*TK*), which is used to release access privilege of data to users.

3.2 Security Assumption

In our access control system, the cloud is assumed to be *honest-but-curious*, which is similar to that assumed in most of the related literatures on secure cloud storage [9], [10], [27], [28]: On the one hand, it offers reliable storage service and correctly executes every computation mission for other entities; On the other hand, it may try to gain unauthorized information for its own benefits.

Beyond the cloud, the whole system consists of one *CA*, some owners and users, in which *CA* is assumed to be fully-trusted, while users could be malicious. *CA* is responsible for key distribution and time token publishing. A malicious user will try to decrypt the ciphertexts to obtain unauthorized data by any possible means, including colluding with other mailicious users.

The proposed TAFC can realize a fine-grained and timed-releasing access control system: Only one user with a satisfied attribute set can access the data after the specific time. The proposed scheme is defined to be compromised if either of the following two types of users can successfully decrypt the ciphertext: 1) A user whose attribute set does not satisfy the access policy of a corresponding ciphertext; 2) A user who tries to access the data before the specified releasing time, even if he/she has satisfying attribute set.

4 Technical Preliminaries

4.1 Bilinear Pairings and Complexity Assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear map with the following properties:

1) *Computability.* There is an efficient algorithm to compute $e(u, v) \in \mathbb{G}_2$, for any $u, v \in \mathbb{G}_1$.
2) *Bilinearity.* For all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
3) *Non-degeneracy.* If g is a generator of $\mathbb{G}_1$, then $e(g, g)$ is also a generator of $\mathbb{G}_2$.

Definition 1 (Decisional BDH Assumption, DBDH). *The DBDH assumption is that no polynomial-time adversary is able to distinguish the tuple $(g^a, g^b, g^c, e(g, g)^{abc})$ from another tuple $(g^a, g^b, g^c, e(g, g)^z)$, if the adversary has no knowledge of the random elements $a, b, c, z \in \mathbb{Z}_p^*$.*

4.2 Ciphertext-Policy Attribute-Based Encryption

CP-ABE [6] is a cryptography prototype for one-to-many secure communication. In a CP-ABE based scheme, besides the storage platform, the system consists of three basic parties: the authority, the owner and the user. The authority is introduced to publish system parameters and issue secret keys for the users. The owner shares files to the intended users by designating an access policy and encrypting the file under the policy. In CP-ABE based approach, the access policy is expressed as a tree over a set of attributes and logic gates, which will be illustrated in detail later. Each user obtains his/her secret key from the authority based on his/her own attributes.

The functionality and security model of CP-ABE assumes that the storage platform (e.g., cloud server) does not conduct the access control management. This type of schemes allow the user to query any ciphertext, but he/she is able to decrypt the ciphertext if and only if his/her attribute set satisfies the access policy of the file. A CP-ABE scheme consists of the following four algorithms:

Setup. It takes a security parameter λ and the attribute universe description U as the input, and outputs a master key MK, and a public parameter PK.

Key Generation. It takes the master key MK and a set of attributes as the input, and outputs the security key SK associated with the input attribute set.

Encryption. It takes the public parameter PK, a message M, and an access policy $\mathcal{T}$ over some attributes as the input. It outputs the ciphertext CT.

Decryption. It takes the security key SK, and the ciphertext CT as the input, and outputs either a message M or the distinguished symbol $\perp$.

Please refer to [6] for more details about CP-ABE. The literatures, such as [8], [9], [41], have introduced CP-ABE to construct fine-grained access control frameworks.

4.3 Timed-Release Encryption

The concept of timed-release encryption is for scenarios that someone wants to securely send a message to another one in the future. In detail, the owner encrypts his/her message for the purpose that intended users can decrypt it after a designated time. From the security aspect, TRE satisfies that: 1) Except the intended users, no one is able to get any information of the message; 2) Even the intended user cannot get the plaintext of the message before the designated releasing time. In order to support an accurate timed-release mechanism, a trusted time agent is required to manage the clock of the system. At each time point T, the agent releases a time token TK_T, which is an important notion in TRE.

When encrypting the message, the ciphertext is generated with the public key of the intended user and the designated releasing time T. The ciphertext holds the feature that only with the corresponding user's secret key and time token TK_T, can a user correctly get the plaintext of the message; otherwise, without either of the two components, the user cannot successfully conduct the decryption.

The literatures, such as [16], [17], have introduced algorithms to realize a practical TRE. Please refer to them for more details.

TABLE 1
Some Notations

Notation	Description
MK	Master secret key of CA
PK	Public parameter of the system
M	Plaintext of the data
$\mathcal{T}$	Access policy over attributes and time
CT	Ciphertext of the data
S_j	Attribute set of user U_j
SK_j	Attribute-associated security key of user U_j
TS_x	Time trapdoor upon node x, in *unexposed* status
TS'_x	Time trapdoor upon x, in *exposed* status
TK_t	Time token of time t
$\mathbb{F}_T$	Unified format of time
H_1	Hash function that maps elements in $\{0, 1\}^*$ to elements in $\mathbb{G}_1^*$
H_2	Hash function that maps elements in $\mathbb{G}_2^*$ to elements in $\mathbb{Z}_p^*$

5 Main Construction of Our Scheme

We first give an overview of our proposed TAFC, mainly discussing how to achieve timed-release function in this paper. Then, we introduce the concepts of access policy, time trapdoor and token. Lastly, we describe our proposed TAFC in details.

Table 1 describes the basic notations in this paper.

5.1 Overview of TAFC

In order to build a scalable and fine-grained access control system for outsourced time-sensitive data, we combine two advanced cryptographic techniques, namely CP-ABE and TRE. The former one is to provide an expressive access control primitive with determined attribute sets; and the latter one is to realize timed-release function.

The general idea of our unique mechanism is to realize access structures in a new form. As shown in Fig. 3, apart from attributes and logic gates defined in existing CP-ABE, the access structure in our scheme contains one or more time trapdoors (TS), each of which represents a time point. The trapdoor is implemented for the timed release function in CP-ABE algorithm. It can be placed upon any node in the structure, arbitrarily defining access privilege releasing time

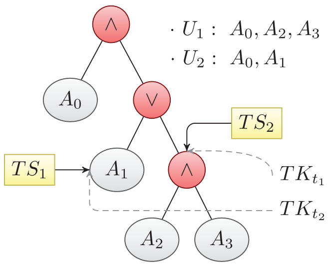


Fig. 2. Example of TAFC access structure.

for different users. The accessing time, together with user's attribute set, determines whether the user satisfies the policy.

For every shared file, the data owner him/herself determines the access policy to encrypt the file. Especially, the time trapdoors in the policy are generated according to a time point $t \in \mathbb{F}_T$. $\mathbb{F}_T$ is system's unified time format, such as "$dd/mm/yyyy$". The time format designates the granularity of timed-release function, e.g., monthly, daily, or hourly. Such mechanism removes the complicated interactions between *CA* and data owners. In the access policy, a node attached with a time trapdoor is said to be satisfied if it holds the following features: 1) Just like CP-ABE, if it is a leaf node, the relevant attribute is among the attribute set; otherwise, the number of its satisfied child nodes exceeds a threshold (will be discussed in detail later); 2) The current access time is later than the relevant releasing time point of the time trapdoor.

From the cryptographic perspective, such idea is realized since *CA* publishes time token TK_T in every time point, just like the time agent does in TRE. Our scheme works if the following feature holds: A user can decrypt a file if and only if his/her attribute set and the obtained time tokens satisfy the access policy. For the performance consideration, in our scheme, time related decryption can be outsourced to the cloud without losing confidentiality.

Moreover, in order to ensure an approximate time consistency, we could introduce a less tight time synchronization mechanism. For example, a third-party Internet Time Server can be introduced, or owners and users all synchronize with CA, who opens a time synchronization interface for the public.

5.2 Access Policy and Time-Related Components

5.2.1 Access Policy Structure

In TAFC, an access policy is over some attributes and one or more releasing time points. Fig. 2 shows an example of the policy structure.

A structure $\mathcal{T}$ consists of a policy tree of several nodes, and some time trapdoors *TS*. A leaf node represents a certain attribute (In Fig. 2, $A_0, \ldots, A_3$ are the relevant attributes), and each non-leaf node represents a threshold gate ("AND", "OR", or others). Each non-leaf node x has two logic values n_x and k_x, where n_x is the number of its child node, and k_x is the threshold. Particularly, $k_x = 1$ if x is an OR gate, or $k_x = n_x$ if x is an AND gate.

In a structure $\mathcal{T}$, the number of included time trapdoors can be zero, one, or more than one. Each trapdoor TS_x is appended to a node x. From the perspective of algorithm, *TS* can be appended to arbitrary node of the structure (*leaf*, *non-leaf*, or even *root*). For instance, in Fig. 2, TS_1 is appended to a leaf node in order to restrict the attribute A_1, while TS_2 is upon a non-leaf node to restrict a sub-policy "$A_2 \wedge A_3$".

5.2.2 Time Trapdoors and Time Tokens

Time trapdoor (*TS*) can be embedded in an access structure, such that the corresponding user's access permission is restricted by the status of *TS*. In this paper, we define two statuses, namely *exposed* or *unexposed*, for the time trapdoor.

- *Unexposed.* A trapdoor (*TS*) is *unexposed* if the intended users cannot access the corresponding secret through the trapdoor with their security keys.
- *Exposed.* A trapdoor is *exposed* if the intended users can get the corresponding secret through this trapdoor. An *exposed* trapdoor is denoted as TS'.

The status of a trapdoor can be transferred from "Unexposed" to "Exposed" with a relevant time token (TK_t). After TK_t is published at time t, anyone, including the cloud and any users, can transfer the status of corresponding time trapdoors (In this paper, the cloud server performs the operation of status transferring, which will not bring about user's overhead or introduce other undesired factors).

In our proposed TAFC, a trapdoor *TS* is generated by a data owner when encrypting his/her data, and a time token *TK* is generated and published by *CA*. The cloud server can transfer one particular trapdoor's status from *unexposed* to *exposed* after obtaining the corresponding TK_t.

Taking Fig. 2 as an example: The trapdoor TS_1 is related to a time point t_1, and TS_2 is related to t_2. Users that satisfy "$A_0 \wedge A_2 \wedge A_3$" (such as U_1) cannot get access privilege until the token TK_{t_1} is published; And users satisfying "$A_0 \wedge A_1$" (such as U_2) should wait for *CA* to publish TK_{t_2}.

Note that, any time t_i in this paper represents a certain time point rather than a length of time interval. In the remaining of this paper, if $t_i < t_j$, it means that t_i is an earlier time point than t_j.

5.3 Construction

Our proposed TAFC consists of six procedures: *setup*, *key generation*, *encryption*, *token generation*, *trapdoor exposure* and *decryption*. Fig. 3 depicts a brief description of our scheme (*setup* and *key generation* are not included in the figure).

5.3.1 Setup

CA generates $I = [p, \mathbb{G}_1, \mathbb{G}_2, g, e, H_1, H_2, \mathbb{F}_T]$, where $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map, $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic multiplicative groups of a prime order p, g is a generator of $\mathbb{G}_1$, $H_1: \{0,1\}^* \rightarrow \mathbb{G}_1^*$, $H_2: \mathbb{G}_2^* \rightarrow \mathbb{Z}_p^*$. $\mathbb{F}_T$ is the time format.

CA randomly chooses $\alpha, \beta, \gamma \in \mathbb{Z}_p^*$. The public parameter is published as

$$PK = \left(I, h = g^{\beta}, f = g^{\gamma}, e(g,g)^{\alpha}\right),$$

and the master key MK is $(\beta, \gamma, g^{\alpha})$ implicitly exists in the system, and doesn't need to be obtained by any other entity. (Note that f and γ are used for timed-release function.)

5.3.2 Key Generation

For each user U_j with attribute set S_j, *CA* first chooses a random $u_j \in \mathbb{Z}_p^*$ as a unique identity for the user. Each attribute

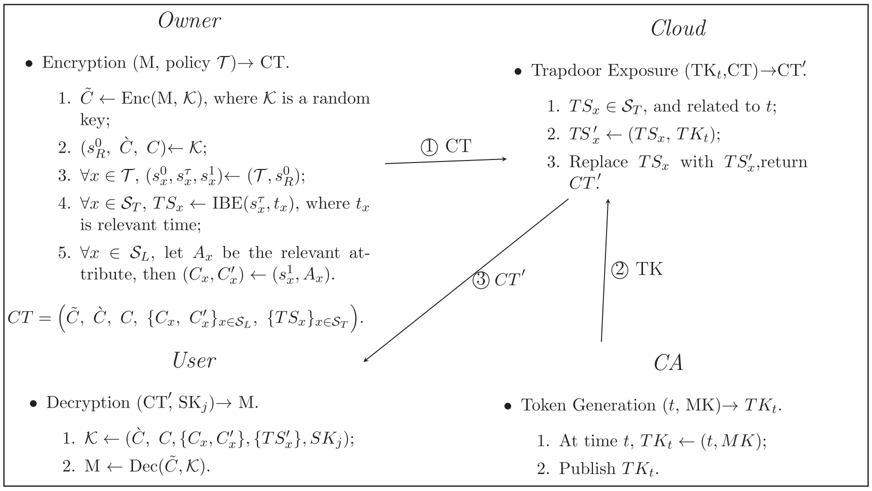


Fig. 3. Procedure description of TAFC construction ($\mathcal{S}_L$ is the set of leaf nodes in $\mathcal{T}$; $\mathcal{S}_T$ is the set of time trapdoors in $\mathcal{T}$, CT' is the notion of modified ciphertext whose time trapdoor has been exposed.).

$Att_i \in S_j$ is assigned a random r_i. Then, CA computes the user's security key as

$$SK_j = \{D = g^{(\alpha+u_j)/\beta},$$
$$\forall Att_i \in S_j : D_i = g^{u_j} \cdot H_1(Att_i)^{-r_i},\ D'_i = g^{r_i}\}.$$

At the end of this procedure, the security key SK_j is sent to U_j in a secure tunnel.

5.3.3 Encryption

The data owner uses a symmetric cryptography to encrypt the data M with a random chosen key $\mathcal{K} \in \mathbb{G}_2$.

In this procedure, each node x in the predefined access structure $\mathcal{T}$ will associate with three secret parameters, denoted as s_x^0, s_x^1 and s_x^τ. Here, s_x^0 is shared with its parent node, s_x^1 is shared with its child node (or dealt with the relevant attribute if x is a leaf node), and s_x^τ is a time-related parameter. Specifically, if x is the root R, s_R^0 is the base secret of $\mathcal{T}$. The parameter assigning is in a top-down manner, starting from the root R as follows:

If x is R, the owner randomly chooses a random parameter $s_R^0 \in \mathbb{Z}_p$. For each node x with s_x^0, the parameters s_x^1 and s_x^τ are chosen as

$$\begin{cases} s_x^\tau \in \mathbb{Z}_p^*, s_x^\tau \cdot s_x^1 = s_x^0 & x \text{ is linked to a time trapdoor} \\ s_x^\tau = 1, s_x^1 = s_x^0 & \text{otherwise} \end{cases}.$$

For each non-leaf node x with s_x^1, the data owner chooses a polynomial q_x, whose degree $d_x = k_x - 1$, and $q_x(0) = s_x^1$. For each of x's child nodes (y) with a unique index $index_y$, the data owner sets $s_y^0 = q_x(index_y)$.

For a trapdoor TS_x related to a releasing time $t \in \mathbb{F}_T$ and a secret parameter s_x^τ, the owner chooses a random r_t, and generates TS_x as follows:

$$TS_x = \Big(A_x = g^{r_t}, B_x = s_x^\tau + H_2(e(H_1(t), f)^{r_t})\Big). \quad (1)$$

For a leaf node x with s_x^1 and relevant attribute Att_x, the owner computes: $C_x = g^{s_x^1}$, $C'_x = H_1(Att_x)^{-s_x^1}$. The final ciphertext is uploaded as follows:

$$\begin{aligned} CT = \\ \big(\mathcal{T}, \quad & \tilde{C} = Enc(M, \mathcal{K}), \grave{C} = \mathcal{K}e(g,g)^{\alpha s_R^0}, C = h^{s_R^0}, \\ & \forall x (\in \mathcal{T}) \text{is a leaf node} : C_x, C'_x; \\ & \forall TS_x \in \mathcal{T} : TS_x = (A_x, B_x)\big). \end{aligned}$$

5.3.4 Token Generation

At each time point $t \in \mathbb{F}_T$, CA generates and publicly publishes a time token TK_t as follows:

$$TK_t = H_1(t)^\gamma.$$

5.3.5 Trapdoor Exposure

When arriving at the releasing time point t related to TS_x, the cloud can obtain a corresponding token TK_t, which is published by CA. Then, the cloud server implements this procedure to expose the trapdoor.

When the cloud gets TK_t, it queries all trapdoors associated with t in all access structures associated with the stored files on it. For each trapdoor $TS_x = (A_x, B_x)$, the cloud computes the *exposed* trapdoors as

$$TS'_x = B_x - H_2(e(TK_t, A_x)).$$

If the procedure is correctly implemented, we can get $TS'_x = s_x^\tau$. The cloud replaces TS_x with TS'_x in each relevant CT, in which the trapdoor can be removed, and the access privilege is transferred to be determined only by the attribute set.

5.3.6 Decryption

After querying CT from the cloud, a user U_j (with the attribute set S_j) conducts this procedure with the security key SK_j. As $TS'_x = s_x^\tau$, For each node x, we can have

$$\begin{cases} s_x^\tau = TS'_x & x \text{ is linked to an exposed trapdoor} \\ s_x^\tau = 1 & \text{no trapdoor is set upon } x \end{cases}.$$

The decryption procedure is performed in a bottom-up manner (from leaf nodes to the root R) as follows:

For a leaf node x with attribute Att_i, if $Att_i \in S_j$ and no unexposed trapdoor is set upon x, then the user computes

$$F_x = \left(\frac{e(D_i, C_x)}{e(D'_i, C'_x)}\right)^{s_x^\tau} = e(g,g)^{u_j s_x^1 s_x^\tau} = e(g,g)^{u_j s_x^0}.$$

If $Att_i \notin S_j$ or TS_x is *unexposed*, then $F_x = \perp$.

For a non-leaf node x, let S_x be an arbitrary k_x-size set of its child nodes, and for each $z \in S_x$, $F_z \neq \perp$. If such S_x exists, and x is not embedded with an *unexposed* trapdoor, then the user computes

$$F_x = \Big(\prod_{z \in S_x} F_z^{\prod_{y \in S_x, y \neq z} \frac{index_y}{index_y - index_z}} \Big)^{s_x^{\tau}} = e(g,g)^{u_j s_x^0}.$$

Otherwise, F_x returns $\perp$.

For the root node R, if $F_R \neq \perp$, then the user can get $F_R = e(g,g)^{u_j s_R^0}$. Finally, the the user can recover the content of M as follows:

$$\mathcal{K}' = \frac{\grave{C}}{e(C,D)/F_R} = \mathcal{K};$$

$$M' = Dec(\tilde{C}, \mathcal{K}') = Dec(Enc(M, \mathcal{K}), \mathcal{K}) = M.$$

6 Security and Performance Analysis

6.1 Security Analysis

We analyze the security properties of TAFC on some critical aspects as follows.

1) *Fine-Grained and Timed-Release Access Control:* Our proposed TAFC provides data owners with the capability to define access policies according to flexible association of attributes and releasing times. With the access policy embedded in the ciphertext, a user can decrypt the ciphertext to access the data, only if his/her attribute set satisfies the policy, and the access time is later than the predefined releasing time.
2) *Security against Collusion Attack:* In TAFC, each user's attribute set-associated security key SK_j is blinded based on a secure random number $u_j \in \mathbb{Z}_p^*$. This mechanism is implemented to resist the collusion attack: The adversary cannot combine different security keys (SK) to forge a new security key associated with a different attribute combination which comes from multiple attribute sets belong to different users. Therefore, the collusion will not bring more privileges to the adversary.
3) *Data Confidentiality:* The confidentiality property of TAFC can be analyzed from two aspects, the cryptography and the protocol as follows:

As a cryptography algorithm to take into account, the adversary model can be described as the following security game:

Setup. The challenger runs the Setup algorithm of TAFC and gives the public parameters to the adversary.

Phase 1. The adversary is allowed to issue queries for a security key for a set of attributes $\mathcal{S}_U$, declare an access time t_A, and a challenge access policy $\mathcal{T}$, where $\mathcal{S}_U$ does not satisfy $\mathcal{T}$ at the time point t_A. The challenger generates the security key associated with $\mathcal{S}_U$ and a series of time tokens that represent time points that are not later than t_A, and then gives the security key and time tokens to the adversary.

Challenge. The adversary submits two equal-length messages M_0 and M_1. The challenger flips a random coin $\nu \in \{0,1\}$, and encrypts M_ν with $\mathcal{T}$. The ciphertext is sent to the adversary.

Phase 2. Phase 1 is repeated to enhance the size of the attribute set of challenger's security key, and to declare a later access time t_B. But the new attribute set cannot satisfy $\mathcal{T}$ at t_B.

Guess. The adversary outputs a guess ν' of ν.

The advantage of adversary is defined as

$$Adv_{\mathcal{A}} = |Pr[\nu' = \nu] - \frac{1}{2}|.$$

Definition 2. *Our proposed TAFC is secure if all polynomial time adversaries have at most a negligible advantage in the above security game.*

Our further analysis classifies all adversaries into two categories:

1) An adversary without a satisfied attribute set for challenge access policy $\mathcal{T}$, although arriving at privilege releasing time;
2) An adversary with satisfied attribute set for $\mathcal{T}$, but the relevant privilege releasing time has not yet arrived.

Apart from the two categories, the remaining adversaries are those neither with satisfied attribute set, nor at the privilege releasing time. We can issue them either security keys for additional attributes, or more time tokens, such that the adversaries can belong to either of the two categories. As such appended information at least has not decreased the adversaries' advantage, the further analysis only focuses on the above two kinds. We conclude the confidentiality of TAFC as follows:

Theorem 1. *If DBDH assumption holds, no polynomial-time adversary belongs to the first category can selectively break TAFC with non-negligible advantage.*

PROOF 1. Suppose we have an adversary $\mathcal{A}$ with a non-negligible advantage $Adv_{\mathcal{A}}$ in the selective security game against TAFC. In such game, the adversary queries adequate time tokens and any secret key. However, the decryption cannot proceed due to the inadequate attributes that are embedded in his/her security key. With these constraints, we can build a simulator $\mathcal{B}$ that plays the DBDH game with a non-negligible advantage as follows.

Initialize. The challenger $\mathcal{C}$ of the DBDH game sets the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with the bilinear map e and generator $g \in \mathbb{G}_1$. $\mathcal{C}$ securely flips a random coin $\mu \in (0,1)$. If $\mu = 0$, $\mathcal{C}$ sets a tuple $(A,B,C,Z) = (g^a, g^b, g^c, e(g,g)^{abc})$; otherwise, the tuple is set as $(g^a, g^b, g^c, e(g,g)^z)$ for random a,b,c,z. Then, $\mathcal{C}$ sends (A,B,C,Z) to $\mathcal{B}$.

Setup. The simulator $\mathcal{B}$ reuses $\mathbb{G}_1$, $\mathbb{G}_2$, e and g from $\mathcal{C}$, randomly chooses $\alpha, \beta, \gamma \in \mathbb{Z}_p^*$, and defines the time format $\mathbb{F}_T$. There is a hash function $H_2 : \mathbb{G}_2^* \to \mathbb{Z}_p^*$. The other hash function H_1 is programmed as a random oracle by building a table, described as follows:

Considering a call to $H_1(A_i)$, if $H_1(A_i)$ was already defined in the table, the oracle returns the same answer in the table. Otherwise, $\mathcal{B}$ chooses a random value $d_i \in \mathbb{Z}_p^*$, and programs the oracle as $H_1(A_i) = g^{d_i}$, then $H_1(A_i) = g^{d_i}$ is inserted into the table. Note that the response from the oracle is distributed randomly due to the g^{d_i} value. Then the public parameter PK is given as

$$PK = \left(p, \mathbb{G}_1, \mathbb{G}_2, g, e, H_2, \mathbb{F}_T, h = g^\beta, f = g^\gamma, e(g,g)^\alpha\right).$$

The simulator $\mathcal{B}$ then sends PK to the adversary $\mathcal{A}$.

Phase 1. In this phase, $\mathcal{A}$ makes requests for a security key associated with an attribute set $\mathcal{S}_U = (A_1, A_2, \ldots, A_{l_1})$, and an access time point t_A. It also designs a challenge access policy $\mathcal{T}$ such that non subset of $\mathcal{S}_U$ satisfies $\mathcal{T}$ before or at t_A. Let $\mathcal{S}_T$ denote the attribute set in $\mathcal{T}$.

Upon receiving the request and $\mathcal{T}$, $\mathcal{B}$ finds a set Γ, which holds the following constraints:

- $\Gamma \bigcap \mathcal{S}_U = \emptyset$ and $\Gamma \subset \mathcal{S}_T$.
- The set $\mathcal{S}_T - \Gamma$ does not satisfy the policy $\mathcal{T}$ before or at t_A.
- If two sets Γ_1 and Γ_2 both hold the first two constraints, and $\Gamma_1 \subsetneq \Gamma_2$, then choose Γ_1.

Note that there may not be a unique Γ. For instance, against a (t,n) gate, $\mathcal{A}$ requests attributes that satisfies k child nodes of the gate, where $k \leq t-2$, then there will be at least C_{n-k}^{t-k-1} choices to design Γ. Such factor will lead to the withdrawal of the simulation, which will occurs in the 2nd phase.

The simulator $\mathcal{B}$ randomly chooses r_i for each element in $\mathcal{S}_U$, and generates $D = (C \cdot g^\alpha)^{1/\beta}$. For each $A_i \in \mathcal{S}_U$, it constructs (D_i, D_i') as

$$D_i = C \cdot H_1(A_i)^{r_i} = C \cdot (g^{d_i})^{r_i}, D_i' = g^{r_i}.$$

Then $\mathcal{B}$ returns $\left(D; \{D_i, D_i' | A_i \in \mathcal{S}_U\}\right)$ to $\mathcal{A}$ as the security keys.

Before the *Challenge* procedure, we first define two functions: *PolySat* and *PolyUnsat*.

PolySat($\mathcal{T}_x, s_x$). This procedure sets up the polynomials for the nodes of a sub-tree $\mathcal{T}_x$ with satisfied root node x, which means $\mathcal{S}_U$ satisfies the access policy of $\mathcal{T}_x$. If x links to an attached trapdoor, $s_x^\tau \in_R \mathbb{Z}_p^*$; otherwise $s_x^\tau = 1$. It first sets a polynomial q_x, with correct degree constraints, and $q_x(0) = s_x / s_x^\tau$. Each child node y obtains $s_y = q_x(index_y)$. Then it sets polynomials for each child node y by calling *PolySat*($\mathcal{T}_y, s_y$).

PolyUnsat($\mathcal{T}_x, g^{s_x}$). It sets up the polynomials for the nodes of $\mathcal{T}_x$ with unsatisfied root node, which means $\mathcal{S}_U$ does not satisfy $\mathcal{T}_x$. s_x^τ is defined to be similar to that in *PolySat*($\mathcal{T}_x, s_x$). It first defines a polynomial q_x with correct degree, and $g^{q_x(0)} = (g^{s_x})^{1/s_x^\tau}$. Due to the feature of unsatisfied node for x, no more than $t_x - 1$ child nodes are satisfied. The function first classifies the child nodes y into two categories: If there are successor nodes that belongs to the set Γ, then y is classified into unsatisfied node; otherwise, it belongs to satisfied one. For each satisfied y, it chooses a random $s_y \in \mathbb{Z}_p$. It then fixes the remaining unsatisfied points of q_x to complete the definition of the polynomial. The procedure recursively defines the polynomials for the child node y by calling:

- PolySat($\mathcal{T}_y, q_x(index_y)$) if y is a satisfied node. $\mathcal{B}$ knows the value $s_y = q_x(index_y)$ in this case.
- PolyUnsat($\mathcal{T}_y, g^{q_x(index_y)}$) if y is an unsatisfied node. Here, only g^{s_y} is known.

Against the challenge policy $\mathcal{T}$, $\mathcal{B}$ runs *PolyUnsat* ($\mathcal{T}, A$), where A is the element of DBDH tuple.

Challenge. $\mathcal{A}$ submits two challenge messages M_0 and M_1 to $\mathcal{B}$, and $\mathcal{B}$ flips a secure coin $\nu \in (0,1)$. For each attribute $A_i \in \mathcal{S}_T$: if $A_i \notin \Gamma$, then $C_i = B^{q_i(0)}, C_i' = (B^{t_i})^{q_i(0)}$; otherwise, $C_i = g^{q_i(0)}, C_i' = (g^{t_i})^{q_i(0)}$.

For each time trapdoor TS_x whose related time point satisfies $t \leq t_A$, simulator $\mathcal{B}$ can generate $TS_x = s_x^\tau$ to expose the trapdoor. Accordingly, for each trapdoor whose related time point holds $t > t_A$, the trapdoor keeps unexposed, simulator $\mathcal{B}$ can compute as in Eq. (1).

The ciphertext CT is constructed as

$$CT = \left(\mathcal{T}, M_\nu \cdot \frac{e(C \cdot g^\alpha, A)}{Z}, h^s = A^\beta, \{C_i, C_i'\}, \{TS_x\}\right).$$

Thus, user $\mathcal{B}$ is able to simulate the scheme. Furthermore, from the perspective of $\mathcal{A}$, the distribution of each component is identical to that in the original scheme.

If $\mu = 0$, the $Z = e(g,g)^{abc}$. We let the security key of unsatisfied attribute $A_i \in \Gamma$ be $D_i = g^{bc} \cdot (g^{d_i})^{r_i}, D_i' = g^{r_i}$. Suppose the Lagrange interpolation for secret s is

$$s = \sum_{A_i \in \mathcal{S}} \lambda_i \cdot q_i(0),$$

for any attribute set $\mathcal{S}$ that satisfies $\mathcal{T}$. Because the secret of root node is the logarithm of A, we then have reconstruction of F_R as

$$\begin{aligned} F_R &= \prod_{A_i \in \mathcal{S}} F_i^{\lambda_i} = \prod_{A_i \in \mathcal{S}} \left(\tfrac{e(D_i, C_x)}{e(D_i', C_x')}\right)^{\lambda_i} \\ &= (e(g,g)^{bc})^{\sum_{i \in \mathcal{S}} \lambda_i q_x(0)} = e(g,g)^{abc}. \end{aligned}$$

Therefore, CT is a valid random encryption of M_ν.

Otherwise, if $\mu = 1$, $Z = e(g,g)^z$ is only a random element from $\mathbb{G}_2$ from the view of $\mathcal{A}$, and such CT contains no information on M_ν.

Phase 2. Repeat Phase 1 to request security keys for a certain larger attribute set, which still does not satisfy $\mathcal{T}$. As this proof cares about the adversary without adequate attribute set, the change of access time t_A is not taken into account, which is discussed in the next proof.

Especially, $\mathcal{A}$ potentially requests a security key for attribute $A_i \in \mathcal{S}_T - \Gamma$, and this action may still be an aspect of the constraints of this game. If it occurs, $\mathcal{B}$ aborts the simulation. Otherwise, it continues the game. Let q denote the possibility that this event does not happen. This possibility differs the adopted attribute set $\mathcal{S}_U$ with the challenge policy $\mathcal{T}$. In general, smaller $\mathcal{S}_U$ and more complex $\mathcal{T}$ bring larger q. We constrain the complexity of the policy, as Yang, et al. [9] did in their proof, then we can have a positive constant q_D such that $q > q_D$. This proof does not analyze the value of q_D.

Guess. $\mathcal{A}$ submits a guess ν' of ν. If $\nu' = \nu$, $\mathcal{B}$ will output its own guess $\mu' = 0$ to indicate that the tuple of DBDH game is a valid BDH-tuple; otherwise, it outputs $\mu' = 1$ to indicate that it was given a random four-tuple.

We assume the distribution of μ and ν is independent. Let $\mathcal{X}$ be the event that the simulation is aborted. Consider the case $\mathcal{B}$ has not abort the simulation. When $\mu = 1$, $\mathcal{A}$ obtains no information on ν. We have $Pr[\nu \neq \nu' | \mu = 1, \bar{\mathcal{X}}] = \frac{1}{2}$. Since $\mu' = 1$ when $\nu \neq \nu'$, we have $Pr[\mu' =$

$\mu|\mu = 1, \bar{\mathcal{X}}] = \frac{1}{2}$. Otherwise $\mu = 0$, CT is a valid encryption of M_ν. The adversary has an advantage $Adv_{\mathcal{A}}$ by definition. We have $Pr[\nu = \nu'|\mu = 0, \bar{\mathcal{X}}] = \frac{1}{2} + Adv_{\mathcal{A}}$. Since $\mathcal{B}$ will guess $\mu' = 0$ when $\nu = \nu'$, we have $Pr[\mu' = \mu|\mu = 0, \bar{\mathcal{X}}] = \frac{1}{2} + Adv_{\mathcal{A}}$. The following formula is derived

$$\begin{aligned} Pr[\mu' = \mu|\bar{\mathcal{X}}] =& Pr[\mu' = \mu|\mu = 1, \bar{\mathcal{X}}] \cdot Pr[\mu = 1|\bar{\mathcal{X}}] \\ &+ Pr[\mu' = \mu|\mu = 0, \bar{\mathcal{X}}] \cdot Pr[\mu = 0|\bar{\mathcal{X}}] \\ =& \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times (\frac{1}{2} + Adv_{\mathcal{A}}) \\ =& \frac{1}{2} Adv_{\mathcal{A}} + \frac{1}{2}. \end{aligned}$$

Now we take into account the case when $\mathcal{B}$ aborts the simulation, it randomly chooses μ' of μ. In this case, the probability of correct guess is up to $\frac{1}{2}$.

The overall advantage of $\mathcal{B}$ in DBDH game is as

$$\begin{aligned} Adv_{\mathcal{B}} &= Pr[\mu' = \mu|\bar{\mathcal{X}}] \cdot Pr[\bar{\mathcal{X}}] + Pr[\mu' = \mu|\mathcal{X}] \cdot Pr[\mathcal{X}] - \frac{1}{2} \\ &= (\frac{1}{2} Adv_{\mathcal{A}} + \frac{1}{2}) \times q_D + \frac{1}{2} \times (1 - q_D) - \frac{1}{2} \\ &= \frac{q_D}{2} Adv_{\mathcal{A}}. \end{aligned}$$

As proved above, there exists a non-negligible polynomial-time adversary $\frac{q_D}{2} Adv_{\mathcal{A}}$ in DBDH game if the polynomial-time adversary in our scheme is $Adv_{\mathcal{A}}$. We can conclude that our scheme is semantically secure against *chosen plaintext attack*, for the adversary that lacks adequate attribute-related keys. □

Theorem 2. *If DBDH holds, no polynomial-time adversary belongs to the second category can selectively break TAFC with non-negligible advantage.*

PROOF 2. We still assume that an adversary $\mathcal{A}$ exists with a non-negligible advantage $Adv_{\mathcal{A}}$ against TAFC. Compared with the last proof, the difference in this game is that the decryption cannot be executed at a too-early access time. Then we can build a simulator $\mathcal{B}$ that plays the DBDH game with non-negligible advantage.

Initialize. It is the same as that in *PROOF 1*.

Setup. The simulator $\mathcal{B}$ does almost the same to generate public parameter PK like CA does in TAFC. The only modification in this simulation is the generation of f and $H_1(t), t \in \mathbb{F}_T$. The time-related parameter is from DBDH game that $f = B$. A random oracle is used to formulate H_1 like that in *PROOF 1*. The public parameter is sent to the adversary.

Phase 1. $\mathcal{A}$ makes a key associated with attribute set and access time requests with the same constraints like that in *PROOF 1*. Faced with the attribute set $\mathcal{S}_U$, access time t_A and challenge policy $\mathcal{T}$, the simulator $\mathcal{B}$ finds the earliest time t_B, at which time, $\mathcal{S}_U$ becomes a satisfied set for $\mathcal{T}$. Note that $t_B > t_A$ if $\mathcal{A}$ obeys the request constraints. For any time point t that is not later than t_A, $\mathcal{B}$ sets $H_1(t) = g^{d_t}$, and $TK_t = B^{d_t}$, with a random $d_t \in \mathbb{Z}_p^*$. The user's security key SK is generated like the original TAFC scheme.

Then $\mathcal{B}$ sends SK and tokens $\{TK_t | t \leq t_A\}$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ sends M_0 and M_1 to $\mathcal{B}$. After flipping a coin $\nu \in (0, 1)$, $\mathcal{B}$ encrypts M_ν as follows: the *non-leaf* nodes and *leaf* nodes are conducted like the original scheme; for each trapdoor TS_x with parameter s_x^τ, we consider two cases:

1) If the access time $t_x < t_B$, $\mathcal{B}$ selects a random r_t, and calculates $A_x = g^{r_t}$, $B_x = s_x^\tau + H_2(e(g^{d_{t_x}}, B)^{r_t})$.
2) Otherwise, the random oracle sets $H_1(t_x) = C \cdot g^{d_{t_x}}$, with random d_{t_x}. In the trapdoor, $A_x = A \cdot g^{r_t}$ with random r_{t_x}, and B_x is computed as

$$B_x = s_x^\tau + H_2\Big(Z \cdot e(B, C)^{r_{t_x}} \cdot e(A, B)^{d_{t_x}} \cdot e(g, B)^{r_{t_x} \cdot d_{t_x}}\Big). \tag{2}$$

Thus, $\mathcal{B}$ is able to simulate the scheme, where, the distribution of each component is identical to that in the original scheme from the perspective of $\mathcal{A}$.

We consider a trapdoor TS_x, whose relevant access time is $t_x \geq t_B$, and the secret parameter is s_x^τ. With $\mathcal{S}_U$, there is a Lagrange interpolation for secret s

$$s = s_x^\tau \cdot \Big(\sum_{A_i \in \mathcal{S}_1} \lambda_j \cdot q_j(0)\Big) + \sum_{A_i \in \mathcal{S}_2} \lambda_i \cdot q_i(0), \tag{3}$$

where, $\mathcal{S}_1 \subset S_U$ is a set of attributes that are controlled by the trapdoor TS_x, and $\mathcal{S}_2$ is the set of other attributes.

If $\mu = 0$, the $Z = e(g, g)^{abc}$, the argument of H_2 in Eq. (2) (denoted as ξ) can be derived as

$$\begin{aligned} \xi =& Z \cdot e(B, C)^{r_{t_x}} \cdot e(A, B)^{d_{t_x}} \cdot e(g, B)^{r_{t_x} \cdot d_{t_x}} \\ =& e(B, C)^a \cdot e(B, C)^{r_{t_x}} \cdot e(g^{d_{t_x}}, B)^a \cdot e(g^{d_{t_x}}, B)^{r_{t_x}} \\ =& \Big(e(B, C) \cdot e(g^{d_{t_x}}, B)\Big)^{a + r_{t_x}} \\ =& e(C \cdot g^{d_{t_x}}, B)^{a + r_{t_x}}, \end{aligned}$$

where $C \cdot g^{d_{t_x}}$ is the output of $H_1(t_x)$ of the random oracle, B is used for the public parameter f, and $g^{a + r_{t_x}} = A \cdot g^{r_{t_x}}$ is the other component of TS_x. Thus, it is a valid trapdoor of s_x^τ. Furthermore, the interpolation of Eq. (3) can reconstruct secret s, and the decryption will ultimately recover the plaintext.

Otherwise, if $\mu = 1$, the $Z = e(g, g)^z$ is only a random element from $\mathbb{G}_2$, and the Trapdoor Exposure procedure will generate a random element of s_x^τ. It will lead to a random $B_x \in \mathbb{Z}_p$ with Eq. (2), then a random $s_x^{tau} \in \mathbb{Z}_p$ with Trapdoor Exposure, further a random secret $s \in \mathbb{Z}_p$ with Eq. (3). Finally, the CT contains no information on M_ν.

Phase 2. Repeat *Phase 1* to request later access time, which still does not satisfy $\mathcal{T}$ with $\mathcal{S}_U$.

Guess. $\mathcal{A}$ submits a guess ν' of ν. If $\nu' = \nu$, $\mathcal{B}$ will outputs its guess $\mu = 0$; otherwise, it outputs $\mu' = 1$.

When $\mu = 1$, $\mathcal{A}$ obtains no information on ν. We have $Pr[\nu \neq \nu'|\mu - 1] = \frac{1}{2}$. Due to the tactics of $\mathcal{B}$, $Pr[\mu = \mu'|\mu = 1] = \frac{1}{2}$. Otherwise $\mu = 0$, the CT is valid because of valid trapdoors. The adversary has an advantage $Adv_{\mathcal{A}}$. We have $Pr[\nu \neq \nu'|\mu - 1] = \frac{1}{2} + Adv_{\mathcal{A}}$. And $\mathcal{B}'$ tactics leads to $Pr[\mu = \mu'|\mu = 1] = \frac{1}{2} + Adv_{\mathcal{A}}$. The following formula is derived

$$\begin{aligned} Adv_{\mathcal{B}} &= Pr[\mu' = \mu|\mu = 1] + Pr[\mu' = \mu|\mu = 0] - \frac{1}{2} \\ &= \frac{1}{2} Adv_{\mathcal{A}}. \end{aligned}$$

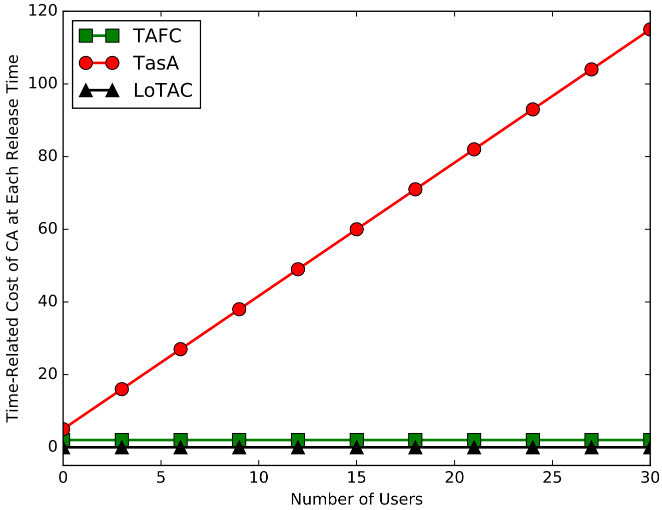


Fig. 4. Cost of *CA* versus number of users.

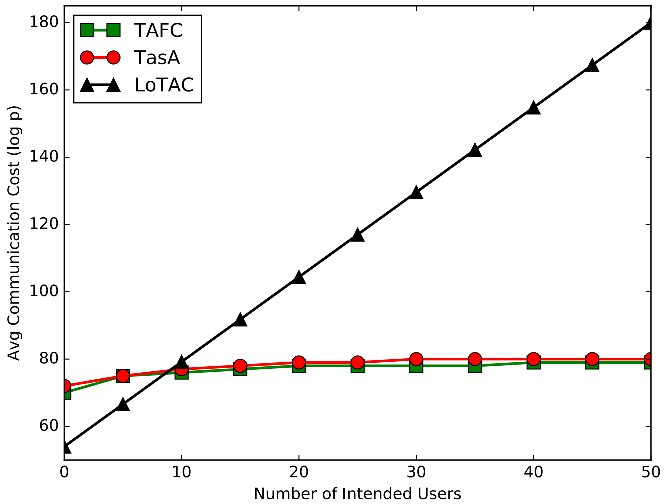


Fig. 6. Cost of owner versus number of intended users.

The proof shows the existence of a non-negligible adversary $\frac{1}{2}Adv_{\mathcal{A}}$ in DBDH game. We can conclude that our scheme is semantically secure against *chosen plaintext attack*, when the attack takes place before the specific access time. □

From the protocol perspective, the s_x^τ is exposed with a relevant time token TK_t, which is generated and published by *CA* at each release time. As the token can be published to the system, rather than securely distributed to other entities, the security feature of this mechanism, therefore, does not rely on an extra secure tunnel.

6.2 Performance Analysis

In order to give an intuitive evaluation of the performance of TAFC, we make a comparison with other related schemes, such as Androulaki et al. [20] (denoted as **LoTAC**), and an approach based on CP-ABE, where time is handled as attribute (denoted as *TasA*). Since the performance differences among these three schemes are mainly on communication and computation cost of *CA* and the data owner, we analyze these two aspects as follows.

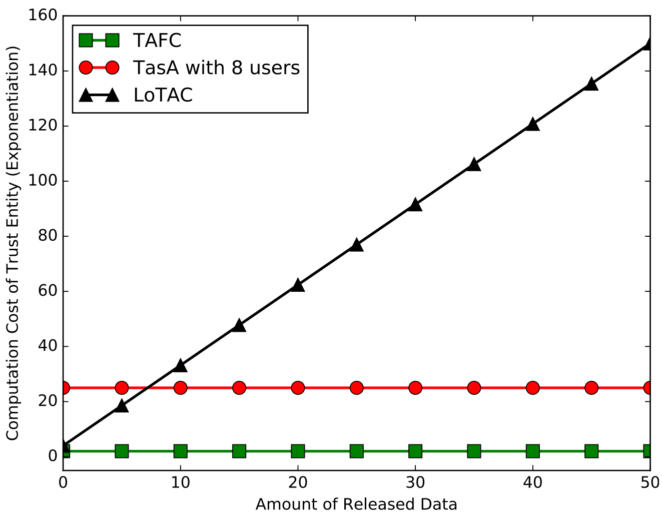


Fig. 5. Computation overhead of trust entity versus amount of released data.

6.2.1 CA's Cost for Timed-Release Function

Figs. 4 and 5 show the overhead evaluation of trust entities (including *CA*), with increasing number of users and released data respectively.

In TAFC, a time token TK_t is a universal parameter among all users for one time point t. *CA*, therefore, only needs to calculate and publish one token at each time. On the contrary, if time is handled as an attribute (as in TasA), *CA* should distribute time-associated security key to each user at each time, meaning that the extra cost is linear to the number of users.

In LoTAC, although *CA* does not need to do anything for timed-release function, another trust entity, should implement the timed-release decryption algorithm for each file at each release time. The overhead of this trust entity for this job is linear to the amount of relevant data, as shown in Fig. 5. On the contrary, in TAFC, the timed-release computation for every file can be outsourced to the *honest-but-curious* cloud, without leaking any unauthorized secret.

Thus, our proposed TAFC shows its superiority on *CA*'s cost reduction, when the access control system includes large amount of users and shared data.

6.2.2 Owner's Cost versus Number of Intended Users

When the owner uploads his/her file, his/her communication cost depends on the package size of the corresponding ciphertext. If we only consider the number of intended users, the cost of owner in LoTAC is $O(|U|)$, where $|U|$ is the number of intended users; while the cost in TAFC and TasA is $O(N_{att})$, where N_{att} is the number of attributes in an access policy. In reality, when the number of intended users increases, N_{att} will increase much more slowly than $|U|$, in quite a high probability. With this assumption, Fig. 6 gives the overhead evaluation of data owner with increasing intended users, when encrypting one data file. Because of fine granularity inherited from CP-ABE, TAFC and TasA significantly reduce the communication complexity of data owner when the access privilege should be released to quite a number of users.

Based on the performance analysis on various aspects, we can conclude that TAFC well tolerates the increasing number of users and shared data. Thus, TAFC can provide a lightweight, flexible, and fine-grained access control system for time-sensitive data in cloud storage.

7 Access Policy Design for General Time-Sensitive Data with Multiple Releasing Time Points

The main construction in Section 5 provides the basic algorithm and cryptography techniques to embed both time and attribute factors into access control for public cloud. However, it lacks a general method for data owners to make an efficient access structure for arbitrary access privilege construction with both time and attribute factors, especially, when a policy is embedded with multiple releasing time points, there exist many cases as described later in this section. These cases cannot be defined by a tree-based structure with existing mechanisms. In this section, we first list the potential sub-policies for time-sensitive data, and then gives an efficient and practical method to construct relevant access structures.

In this paper, all access policies hold the constraints of *Monotonous Access Capability* defined as follows.

Definition 3 (Monotonous Access Capability). *The Access capability should hold the monotonic property that can be formulated with both of the two constraints:*

- *For any user U_j, and any file M_i, if $t_1 < t_2$ and U_j can access M_i at t_1, then he/she can also access it at t_2.*
- *If two attribute sets $\mathcal{S}_1$ and $\mathcal{S}_2$ have $\mathcal{S}_1 \subset \mathcal{S}_2$ for a file, and the releasing times for these two sets are t_1 and t_2, respectively, then we have $t_1 \geq t_2$.*

With the above defined constraints, we can summarize the sub-policy design mechanism faced with boolean formulas and (t, n) threshold, which have also been mentioned in [35].

For boolean formulas, there are two types that hold the above definition: 1) Converting an attribute to an OR gate; and 2) Removing an attribute from an AND gate. The first type can be realized with the example structures in Fig. 2, where we denote P as $A_2 \wedge A_3$. At earlier time t_1 (assume $t_1 < t_2$), the sub-policy is an attribute A_1, and after t_2, the sub-policy is automatically updated to $A_1 \vee P$. As structures and algorithms for this type can be ideally achieved in the main constructions, we will discuss how to achieve the second type in Section 7.2.

For a (k, n) threshold gate, two potential cases should be considered: 1) Delaying the time point t and reducing the threshold k at the same time can hold the defined constraints. We will discuss how to efficiently achieve it in Section 7.3.2) Also, the constraints allow the scenario where later access brings in larger n. The achievement of this will be presented in Section 7.4.

With the above considerations, we will first introduce a modified algorithm for the time trapdoor construction. Then we will further design structures for the sub-policies of time-related data into two cases, and any potential access policy for time-sensitive data can be expressed as the combination of these proposed structures.

7.1 Unattached Time Trapdoor: A Trapdoor as a Single Leaf Node

In the main construction of TAFC, a time trapdoor should be attached to a node of the policy tree. Here we further give another scheme to support a time trapdoor without being attached to any node, which will be utilized to realize time-related sub-policies in the following sections. From the perspective of the structure construction, such time trapdoor is a leaf node, which can be regarded as a special attribute. In this section, we use *attached time trapdoor* to indicate that it's attached to a certain internal node, and *unattached time trapdoor* to indicate that it is not attached to any node.

In the Encryption procedure, we can obtain a secret share $s_x^0 \in \mathbb{Z}_p^*$ and an unattached time trapdoor TS_x from its parent node. Then, we can get $s_x^\tau = s_x^0$ (different from that for an attached time trapdoor), and an unexposed trapdoor $TS_x = (A_x, B_x)$ is generated with s_x^τ and the predetermined releasing time $t \in \mathbb{F}_T$ as shown in Eq. (1).

A trapdoor can be exposed by the cloud server with the same mechanism as that in Section 5.3.5. When a user U_j has got a ciphertext with an exposed trapdoor TS'_x, he/she can further compute F_x as follows:

$$\begin{aligned} F_x &= \Big(\frac{e(h, g^{(\alpha+u_j)/\beta})}{e(g,g)^\alpha} \Big)^{TS'_x} \\ &= \Big(\frac{e(g^\beta, g^{(\alpha+u_j)/\beta})}{e(g,g)^\alpha} \Big)^{s_x^\tau} = e(g,g)^{u_j s_x^\tau}. \end{aligned} \tag{4}$$

For an unattached time trapdoor, $s_x^0 = s_x^\tau$, we can further get $F_x = e(g,g)^{u_j s_x^0}$, which can be utilized to reconstruct its parent's secret F_y as shown in Eq. (2).

In the following sections, we will mainly focus on the placement of time trapdoors. For clarity, we use time t instead of the node TS_x to indicate the time trapdoor in this section.

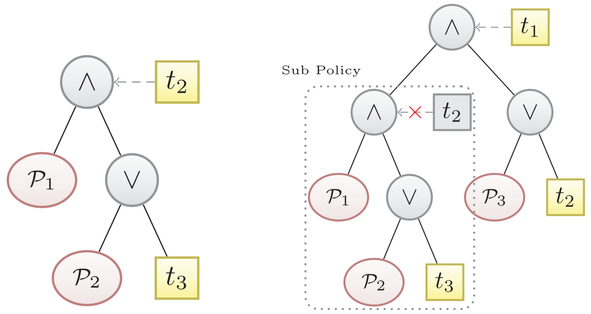
7.2 An Additional Satisfied Sub-Policy Wins an Earlier Access (Case 1)

This case is used to satisfy the scenario: For example, a user whose attribute set satisfies a sub-policy $\mathcal{P}_1$ can access a file at time t_3. If the user can *additionally* satisfy a sub-policy $\mathcal{P}_2$, the access privilege will be granted at earlier time $t_2 < t_3$.[1] The $\mathcal{P}_1$ and $\mathcal{P}_2$ can be either a single attribute or a sub-structure with multiple nodes.

Fig. 7a depicts our proposed access structure to realize the above access policy: An OR gate is set over the additional policy $\mathcal{P}_2$ and the trapdoor t_3; and an AND gate is over this OR gate and the sub-policy $\mathcal{P}_1$. Finally, the trapdoor t_2 is linked to the AND gate. With this kind of structure, a user whose attributes satisfy both $\mathcal{P}_1$ and $\mathcal{P}_2$ will get access privilege when it reaches the time point t_2, while a user whose attributes only satisfy $\mathcal{P}_1$ cannot satisfy the whole policy until it reaches time point t_3, since neither $\mathcal{P}_2$ nor t_2 under the OR gate can be satisfied.

We can extend this case to a multiple-hierarchy scenario: If a user's attribute set satisfies $\mathcal{P}_1$, he/she can access the file at time point t_3; If his/her attribute set additionally satisfies $\mathcal{P}_2$, he/she can access the file at time point t_2 (earlier than t_3); While his/her attribute set satisfies $\mathcal{P}_3$ in addition to $\mathcal{P}_1$ and $\mathcal{P}_2$, the access privilege will be granted to him/her further earlier, say at time t_1.

1. In order to easily introduce the multiple-hierarchy scenario with no puzzle based on this type of one hierarchy scenario, we first use t_2 and t_3 leaving t_1 to be introduced in the multiple-hierarchy scenario later on.



(a) One Hierarchy (b) Multiple Hierarchies

Fig. 7. Structure for case 1.

Fig. 7b shows the structure to meet the above requirement. The structure in Fig. 7a can be treated as a sub-policy of the policy indicated in Fig. 7b, which has only a small modification: the time trapdoor t_2 is no longer linked to the original AND gate. Instead, it is under an OR gate along with $\mathcal{P}_3$. If the modified sub policy is treated as one basic symbol, the two structures in Figs. 7a and 7b are similar from the perspective of the policy structure construction. From this viewpoint, our proposed mechanism can be utilized to construct a recursive structure, which can be extended to satisfy the scenarios with more hierarchies.

7.3 More Satisfied Sub-Policies Wins an Earlier Access (Case 2)

In this case, we consider the scenario: There are a collection of sub-policies ($\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_n$). If at the time point t_i (We assume $t_{i+1} > t_i$), the access policy is a (k_i, n) threshold gate over the above sub-policies, in which, we have

$$t_1 \leq t_i < t_j \leq t_m \Leftrightarrow k_i > k_j. \tag{5}$$

Then, we have relevant access structure to realize this kind of policy requirement as shown in Fig. 8. A non-leaf node whose threshold is k_1 (the relevant access time is t_1) is set as the root of this structure, and it has $n + k_1 - k_m$ child nodes: The child nodes include the sub-policies $\mathcal{P}_1$, $\mathcal{P}_2$, $\ldots$, $\mathcal{P}_n$ and a series of unattached time trapdoors. The number of unattached time trapdoors for each predefined time t_i $(2 \leq i \leq m)$ equals to $k_{i-1} - k_i$. Each of the child nodes, whether it's the candidate sub-policy $\mathcal{P}_n$, or the unattached time trapdoor, will get its unique secret share from the root node, with a $(k_1, n + k_1 - k_m)$ secret sharing method. Finally, the time trapdoor t_1 is an attached time trapdoor and is linked to the root.

When it reaches time point t_i $(1 \leq i \leq m)$, apart from t_1, the trapdoors associated with the time point $\{t_2, t_3, \ldots, t_i\}$ have been exposed, whose total number is $\sum_{j=2}^{i}(k_{j-1} - k_j) = k_1 - k_i$. If a user wants to access the data at that time,

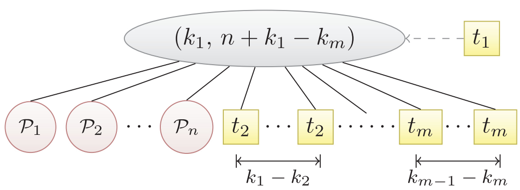


Fig. 8. Structure for case 2.

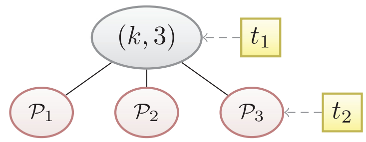


Fig. 9. Structure for case 3.

he/she can compute F_x for each exposed trapdoor as Eq. (4), where the number of all trapdoors is $k_1 - k_i$. Thus, if and only if his/her attribute set satisfies at least k_i of the n candidate sub-policies $\mathcal{P}_j$, the total number of satisfied child nodes equals to k_1, which is just the threshold. Furthermore, F_x of the root node can be reconstructed as Eq. (2).

If we only consider the access structure at this moment, the user only needs to concern that whether his/her attribute set can satisfy k_i of the n sub-policies. This is just the access requirement of the access policy depicted in the first paragraph of this case.

In addition, if the access time is earlier than t_1, the attached trapdoor t_1 can prevent such unauthorized access behaviour. Therefore, the structure in Fig. 8 supports a (k_i, n) threshold gate for the n sub-policies at each required time point t_i, and is able to meet the access policy requirement.

7.4 Later Access has Larger n of (k, n) Gate (Case 3)

In this case, we consider a scenario, where the (k, n) has such requirement: The threshold k is constant, where more candidate sub-policies will enlarge the n. In this scenario, later access means more choice to constitute one's attributes to satisfy the access policy. We use a simple access control requirement as an example: At time point t_1, the threshold of the access policy is $(k, 2)$ gate, where the candidate sub-polices are $\mathcal{P}_1$ and $\mathcal{P}_2$. While at time point t_2 $(t_2 > t_1)$, a user whose attribute set can satisfy k of three sub-policies can also satisfy the policy, where $\mathcal{P}_3$ is the additional candidate sub-policy, and k is not changed. Fig. 9 shows the access structure of this example.

In this structure, the access control at different time is as follows:

1) An access before t_1 will fail because of the trapdoor t_1.
2) If the time is between t_1 and t_2, the sub-policy $\mathcal{P}_3$ is linked to an unexposed trapdoor t_2. Therefore, candidate sub-policies that can be used are only $\mathcal{P}_1$ and $\mathcal{P}_2$. Whether or not a user's attribute set satisfies $\mathcal{P}_3$ does not affect the access judgement.
3) At time point t_2, the remaining time trapdoor is exposed, which means $\mathcal{P}_3$ becomes a candidate sub-policy of the $(k, 3)$ gate. Therefore, a user whose attribute set satisfies k of the three sub-polices will satisfy the entire policy.

Note that, if $k = 1$, the structure is similar to that in Fig. 2. The above analysis shows that access structures like Fig. 9 can achieve time-sensitive data access control requirement of Case 3. What is more, if we add unattached time trapdoors $t_i > t_1$ to Fig. 2, as child nodes of the root, we can achieve increasing threshold k and decreasing candidate n in one structure, which is the combination of Cases 2 and 3.

8 Conclusion

This paper aims at fine-grained access control for time-sensitive data in cloud storage. One challenge is to simultaneously achieve both flexible timed release and fine granularity with lightweight overhead, which was not explored in existing works. In this paper, we proposed a scheme to achieve this goal. Our scheme seamlessly incorporates the concept of timed-release encryption to the architecture of ciphertext-policy attribute-based encryption. With a suit of proposed mechanisms, this scheme provides data owners with the capability to flexibly release the access privilege to different users at different time, according to a well-defined access policy over attributes and release time. We further studied access policy design for all potential access requirements of time-sensitive, through suitable placement of time trapdoors. The analysis shows that our scheme can preserve the confidentiality of time-sensitive data, with a lightweight overhead on both *CA* and data owners. It thus well suits the practical large-scale access control system for cloud storage.

Acknowledgments

The authors sincerely thank the anonymous referees for their invaluable suggestions that have led to the present improved version of the original manuscript. This work is supported by the National Natural Science Foundation of China under Grant No. 61379129 and No. 61671420, Youth Innovation Promotion Association CAS, and the Fundamental Research Funds for the Central Universities. K. Xue is the corresponding author.

References

[1] Z. Qin, H. Xiong, S. Wu, and J. Batamuliza, "A survey of proxy re-encryption for secure data sharing in cloud computing," *IEEE Trans. Services Comput.*, (2016). [Online]. Available: https://doi.org/10.1109/TSC.2016.2551238

[2] F. Armknecht, J.-M. Bohli, G. O. Karame, and F. Youssef, "Transparent data deduplication in the cloud," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 886–900.

[3] R. Masood, M. A. Shibli, Y. Ghazi, A. Kanwal, and A. Ali, "Cloud authorization: Exploring techniques and approach towards effective access control framework," *Frontiers Comput. Sci.*, vol. 9, no. 2, pp. 297–321, 2015.

[4] K. Xue and P. Hong, "A dynamic secure group sharing framework in public cloud computing," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 459–470, Oct.–Dec. 2014.

[5] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.

[6] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. 28th IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.

[7] Y. Zhu, D. Huang, C.-J. Hu, and X. Wang, "From RBAC to ABAC: Constructing flexible data access control for cloud storage services," *IEEE Trans. Services Comput.*, vol. 8, no. 4, pp. 601–616, Jul./Aug. 2015.

[8] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 743–754, Apr. 2012.

[9] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.

[10] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[11] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1484–1496, May 2016.

[12] K. Xue, et al., "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 4, pp. 953–967, Apr. 2017.

[13] E. Bertino, P. A. Bonatti, and E. Ferrari, "TRBAC: A temporal role-based access control model," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 191–233, 2001.

[14] I. Ray and M. Toahchoodee, "A spatio-temporal role-based access control model," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2007, pp. 211–226.

[15] D. Kulkarni and A. Tripathi, "Context-aware role-based access control in pervasive computing systems," in *Proc. 13th ACM Symp. Access Control Models Technol.*, 2008, pp. 113–122.

[16] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. MIT-LCS-TR-684, 1996.

[17] K. Yuan, Z. Liu, C. Jia, J. Yang, and S. Lv, "Public key timed-release searchable encryption," in *Proc. 4th Int. Emerging Intell. Data Web Technol.*, 2013, pp. 241–248.

[18] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Inf. Sci.*, vol. 258, no. 3, pp. 355–370, 2014.

[19] L. Xu, F. Zhang, and S. Tang, "Timed-release oblivious transfer," *Secur. Commun. Netw.*, vol. 7, no. 7, pp. 1138–1149, 2014.

[20] E. Androulaki, C. Soriente, L. Malisa, and S. Capkun, "Enforcing location and time-based access control on cloud-stored data," in *Proc. 34th IEEE Int. Distrib. Comput. Syst.*, 2014, pp. 637–648.

[21] C.-I. Fan and S.-Y. Huang, "Timed-release predicate encryption and its extensions in cloud computing," *J. Internet Technol.*, vol. 15, no. 3, pp. 413–426, 2014.

[22] J. Hong, K. Xue, W. Li, and Y. Xue, "TAFC: Time and attribute factors combined access control on time-sensitive data in public cloud," in *Proc. IEEE Global Commun. Conf.*, 2015, pp. 1–6.

[23] H. Tian, et al., "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Services Comput.*, (2016). [Online]. Available: https://doi.org/10.1109/TSC.2015.2512589

[24] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 220–232, Apr.–Jun. 2012.

[25] H. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 328–340, Mar./Apr. 2015.

[26] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[27] Z. Zhou, H. Zhang, Q. Zhang, Y. Xu, and P. Li, "Privacy-preserving granular data retrieval indexes for outsourced cloud data," in *Proc. IEEE Global Commun. Conf.*, 2014, pp. 601–606.

[28] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Reliable re-encryption in unreliable clouds," in *Proc. IEEE Global Commun. Conf.*, 2011, pp. 1–5.

[29] J. Li, W. Yao, Y. Zhang, and H. Qian, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Services Comput.*, (2016), [Online]. Available: https://doi.org/10.1109/TSC.2016.2520932

[30] Y. Xue, J. Hong, W. Li, K. Xue, and P. Hong, "LABAC: A location-aware attribute-based access control scheme for cloud storage," in *Proc. IEEE Global Commun. Conf.*, 2016, pp. 1–6.

[31] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 457–473.

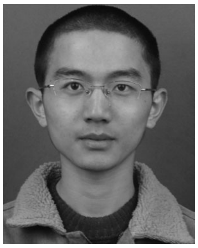
[32] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.

[33] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. 29th IEEE Int. Conf. Comput. Commun.*, 2010, pp. 1–9.

[34] L. Touati and Y. Challal, "Collaborative KP-ABE for cloud-based internet of things applications," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–7.

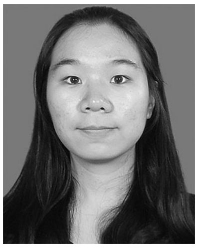
[35] K. Yang, X. Jia, K. Ren, R. Xie, and L. Huang, "Enabling efficient access control with dynamic policy updating for big data in the cloud," in *Proc. 33rd IEEE Int. Conf. Comput. Commun.*, 2014, pp. 2013–2021.

[36] K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3461–3470, Dec. 2015.
[37] K. Liang, Q. Huang, R. Schlegel, D. S. Wong, and C. Tang, "A conditional proxy broadcast re-encryption scheme supporting timed-release," in *Information Security Practice and Experience*. Berlin, Germany: Springer, 2013, pp. 132–146.
[38] X. Ma, L. Xu, and F. Zhang, "Oblivious transfer with timed-release receiver's privacy," *J. Syst. Softw.*, vol. 84, no. 3, pp. 460–464, 2011.
[39] Y. Zhu, H. Hu, G.-J. Ahn, D. Huang, and S. Wang, "Towards temporal access control in cloud computing," in *Proc. 31st IEEE Int. Conf. Comput. Commun.*, 2012, pp. 2576–2580.
[40] K. Yang, Z. Liu, X. Jia, and X. Shen, "Time-domain attribute-based access control for cloud-based video content sharing: A cryptographic approach," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 940–950, May 2016.
[41] X. Zhu, S. Shi, J. Sun, and S. Jiang, "Privacy-preserving attribute-based ring signcryption for health social network," in *Proc. IEEE Global Commun. Conf.*, 2014, pp. 3032–3036.



Jianan Hong received the BS degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2012. He is currently working toward the PhD degree in information security from the Department of Electronic Engineering and Information Science(EEIS), USTC. His research interests include secure cloud computing and mobile network security.

Kaiping Xue (M'09-SM'15) received the BS degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and the PhD degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is an associate professor in the Department of Information Security and Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks, and network security. He is a senior member of the IEEE.



Yingjie Xue received the BS degree from the Department of Information Security, University of Science and Technology of China (USTC), in July, 2015. She is currently working toward the master degree in communication and information system from the Department of Electronic Engineering and Information Science(EEIS), USTC. Her research interests include network security and cryptography.

Weikeng Chen is currently working toward the BS degree from the Department of Information Security, University of Science and Technology of China(USTC). He will be a doctoral student at the University of California, Berkeley, in fall 2017. His research interests include network security protocol design and analysis.

David S. L. Wei (SM'07) received the PhD degree in computer and information science from the University of Pennsylvania, in 1991. He is currently a professor of Computer and Information Science Department, Fordham University. From May 1993 to August 1997, he was on the faculty of computer science and engineering with the University of Aizu, Japan, (as an associate professor and then a professor). He has authored and co-authored more than 100 technical papers in various archival journals and conference proceedings. He served on the program committee and was a session chair for several reputed international conferences. He was a lead guest editor of the *IEEE Journal on Selected Areas in Communications* for the special issue on Mobile Computing and Networking, a lead guest editor of the *IEEE Journal on Selected Areas in Communications* for the special issue on Networking Challenges in Cloud Computing Systems and Applications, a guest editor of the *IEEE Journal on Selected Areas in Communications* for the special issue on Peer-to-Peer Communications and Applications, and a lead guest editor of the *IEEE Transactions on Cloud Computing* for the special issue on Cloud Security. He was the chair of Intelligent Transportation Forum of Globecom 2010, the general chair of Intelligent Transportation Workshop of ICC 2011, and the chair of Cloud Security Forum and Intelligent Transportation Forum of Globecom 2011. He is currently an associate editor of the *IEEE Transactions on Cloud Computing*, an associate editor of the *Journal of Circuits, Systems and Computers*, and a guest editor of the *IEEE Transactions on Big Data* for the special issue on Trustworthiness in Big Data and Cloud Computing Systems. Currently, His research interests include cloud computing, big data, IoT, and cognitive radio networks. He is a senior member of the IEEE.

Nenghai Yu received the BS degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1987, the ME degree from Tsinghua University, Beijing, China, in 1992, and the PhD degree from the University of Science and Technology of China, Hefei, China, in 2004. Since 1992, he has been a faculty in the Department of Electronic Engineering and Information Science, USTC, where he is currently a professor. He is the executive director of the Department of Electronic Engineering and Information Science, USTC, and the director of the Information Processing Center, USTC. He has authored or co-authored more than 130 papers in journals and international conferences. His research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.

Peilin Hong received the BS and MS degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986. Currently, she is a professor and advisor for PhD candidates in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security. She has published two books and more than 150 academic papers in several journals and conference proceedings.