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

Jianan Hong, Kaiping Xue*, Wei Li, Yingjie Xue
The Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027 China
*kpxue@ustc.edu.cn

***Abstract*—The new paradigm of outsourcing data to the cloud is a double-edged sword. On one side, it frees up data owners from the technical management, and is easier for the data owners to share their data with intended recipients when data are stored in the cloud. On the other side, it brings about new challenges about 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 efficient schemes can provide the scenario of fine-grained access control together with the capacity of time-sensitive data publishing. In this paper, by embedding the mechanism of timed-release encryption into CP-ABE (Ciphertext-Policy Attribute-based Encryption), we propose TAFC: a new time and attribute factors combined access control on time-sensitive data stored in cloud. 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.**

Keywords*—*Cloud Storage, Access control, Time-sensitive data, Fine granularity.

I. Introduction

Cloud storage service has significant advantages on both convenient data sharing and cost reduction. However, this new paradigm of data storage brings about new challenges about data confidentiality protection. Data are no longer in data owner's trusted domain, and he/she cannot trust the cloud server to conduct secure data access control. Therefore, the secure access control problem has become a challenging issue in cloud storage.

There have been numerous works [1–5] on privacy preserving data sharing in cloud based on various cryptographic primitives, in which the schemes [1–3] based on CP-ABE [6] attract extensive attentions, since they can guarantee data owner fine-grained and flexible access control of his/her own data. However, these schemes determine user's access privilege only based on his/her inherent attributes without any other critical aspects, such as the time factor. In reality, the time factor usually plays an important role in dealing with time-sensitive data [7] (e.g. to publish a latest electronic magazine, to expose a company's future business plan). When uploading time-sensitive data to the cloud, the data owner may want different users to access the content after different time. However, to the best of our knowledge, existing CP-ABE based schemes cannot meet such requirement.

To tackle the above issue of timed release, it is necessary to introduce an effective scheme, which will not release the data access privilege to intended user until corresponding predefined time. A trivial solution is to leave data owners to manually release the time-sensitive data: The owner uploads the encrypted data under different policies at each release time, thus intended users cannot access the data until the corresponding time arrives. However, such solution restricts the owner to be online to repeatedly upload the different encryption versions of the same data, which makes the data owner in a heavy trouble.

From the perspective of cryptography, the goal of timed release can be achieved by Timed-Release Encryption (TRE). Rivest et al. [8] have proposed an effective TRE scheme, and it has been subsequently introduced into different aspects, such as searchable encryption [9], proxy re-encryption [10], conditional oblivious transfer [11]. In a TRE-based system, a trust time agent, rather than data owner, can uniformly release the access privilege at each predefined time. Androulaki et al. [12] have designed an approach to realize time-sensitive data access control in cloud. Whereas, this approach lacks fine granularity, which may leave the data owners an unbearable burden in a large-scale system. Fan et al. [13] have proposed timed-release predicate encryption for cloud computing. In their scheme, each data file can be labelled with only one release time point, which cannot release the access privilege of one file to different intended users at different time.

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 time as an attribute [12]. However, unbearable number of time-related keys will be issued to each user at each corresponding time, and this will bring about heavy overhead on both computation and communication. In existing literatures, Qin et al. [10] have made a preliminary attempt to integrate time with attributes. It only addresses the issue that the attributes' life period of each user may be limited by time. However, a more practical scheme is that: each user with different attribute sets will have different release time for the same data file. Thus, the scheme in [10] cannot meet this important requirement.

In this paper, we propose an efficient time and attribute factors combined access control scheme for time-sensitive data in public cloud, named TAFC. Our scheme has two important capacities: on one side, it inherits the property of fine granularity from CP-ABE; on the other side, by introducing the trapdoor mechanism, it further has the feature of timed release from TRE. In our scheme, the introduced trapdoor mechanism is only related to the time factor, in which only one corresponding secret should be published at each time to expose the related trapdoors. This makes our scheme highly efficient, with only little extra overhead added to the original CP-ABE based scheme.

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

1) To the best of our knowledge, this paper is the first

978-1-4799-5952-5/15/$31.00 ©2015 IEEE

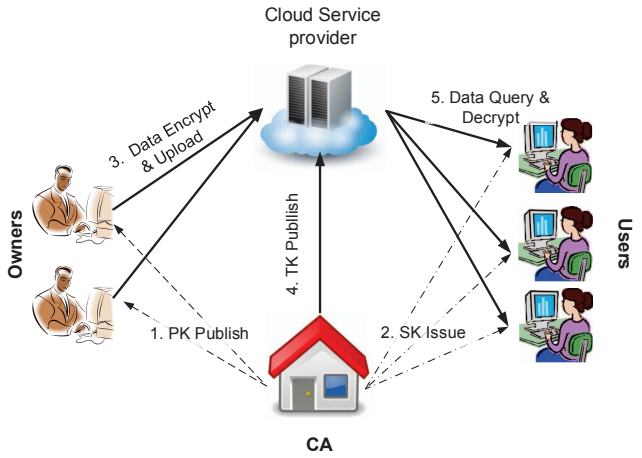

Fig. 1. TAFC Architecture and Operations

that proposes two factors (time and attributes) combination based access control scheme in cloud storage, which can simultaneously achieve the features of fine granularity and timed release.

2) We design an effective architecture to realize our scheme, in which we redesign an entity (the central authority, CA) to be responsible for the timed-release function. Besides distributing attribute-associated private keys, CA only needs to periodically publish universal time-related tokens to release access privileges. Such architecture occupies only a small amount of cost to provide our required access control scheme, which is reasonable and worthy.
3) For the function of timed release, there is no use of a secure tunnel between CA and the data owner. Thus, the additional overhead is lightweight.

The rest of this paper is organized as follows. In section II, we present the system architecture and state the security model. Section III reviews the related technical preliminaries. Section IV gives detailed construction of our proposed TAFC. In Section V we analyze our proposed scheme in terms of its security and performance. Finally, we conclude this paper in Section VI.

II. System and Security Model

A. System Model

As depicted in Fig. 1, the system consists of the following entities: cloud service provider (*cloud*), a central authority (*CA*), several data owners (*owner*), and many data consumers (*user*).

- ***Cloud service provider (cloud)*** includes the administrator of the cloud and cloud servers. The cloud stores a collection of data from owners, accepts download request from any users, as well as helps owners and users conduct onerous computations.
- ***The central authority (CA)*** is responsible to manage the security protection of the whole system: It publishes system parameters and distributes private keys related to specific attributes for each user. In addition, it acts as a time agent to publish the time-related secret (denoted as *time token, TK)* at each pre-defined time (This can be simplified as a periodic operation).
- ***The data owner (owner)*** decides the access policy based on attributes and release time, then encrypts the data under the policy before uploading it.
- ***The data consumer (user)*** is assigned a private key from CA. He/she can query any ciphertext stored in the cloud, but is able to decrypt it only if he/she satisfies both of the following constraints: 1) His/her attribute set satisfies the access policy; 2) The current access time is later than the corresponding release time.

B. Security Model

In our access control system, the cloud is assumed to be "honest-but-curious", which is similar to most of the related literatures in the topic of cloud secure storage [2–5]: On 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, many owners and users, in which CA is assumed to be fully-trust, while users can be malicious. CA is responsible for key distribution and time token publishing. We assume that a malicious user may try to decrypt the ciphertext to obtain unauthorized data by all means, including colluding with other users.

The proposed TAFC can realize a fine-grained and timed-release access control system: Only a user with satisfied attribute set can access the data after the designate 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 corresponding ciphertext; 2) A user who tries to access the data before the specified release time, even if he/she has satisfied attribute set.

III. Preliminaries

A. 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 $a, b, c, z \in \mathbb{Z}_p^*$.

B. Ciphertext-Policy Attribute-based Encryption

CP-ABE [6] is a cryptography prototype for one-to-many communication. In CP-ABE, each data file is encrypted under an access policy over some attributes, and each user is assigned a private key associated with a set of attributes. A user can access the data only if his/her attribute set satisfies the access policy. A CP-ABE scheme consists of the following four algorithms:

Setup. It takes a security parameter 1^λ as 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 input, outputs the private key SK associated to the inputting attribute set.

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

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

In order to protect data confidentiality against collusion attack, the attribute-associated key of each user is blinded by a secret random number.

Please refer to [6] for more details about CP-ABE. The literatures such as [1, 2, 14] have introduced CP-ABE to construct fine-grained access control frameworks.

IV. Our Scheme

At the beginning of this section, we firstly present an overview of our proposed TAFC, mainly discussing the proposed mechanisms to achieve timed-release function. In the following, we introduce the concepts of access policy, time trapdoor and token. After that, we describe TAFC in details.

Table I describes our basic notations, which will be used in the following paragraphs.

TABLE I. 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 private key of user U_j
TS_x	Time trapdoor set 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, such as "$dd/mm/yyyy$"

A. Overview of TAFC

To realize access control for time-sensitive data stored in cloud, we introduce timed-release encryption (TRE) into basic CP-ABE, to propose a relevant access control scheme for cloud storage. One challenging issue is how to combine CP-ABE and TRE. As Androulaki et al. [12] have mentioned, there lack effective approaches to associate these two techniques, except handling time as attribute, which will bring about unbearable number of keys. Our goal is to make a time-related key (or *time token* in TAFC) become a universal parameter for all users, so that the trust agent CA only needs to publish one token at each time, rather than issue time-related keys for each of the users. To achieve our goal, we propose a suit of mechanisms as follows:

1) Besides implementing the attribute-associated key management in basic CP-ABE, CA will define a timeline including a sequence of time points (t_1, t_2, $\cdots$, t_n), as shown in Fig. 2. In the following procedures, these time points can be selected by each data owner to release access privileges. The timeline is a virtual concept, and is realized as a unified time format $\mathbb{F}_T$ as illustrated in Table I.

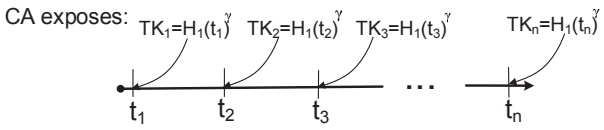

Fig. 2. Model of Timeline

2) A ciphertext is encrypted under an access policy $\mathcal{T}$ according to a pre-defined attribute set and release time, as shown in Fig. 3. To realize timed-release function, a time-related trapdoor TS_x is set in a specified node x in $\mathcal{T}$. The trapdoor is used to blind the secret of x, so that only when the trapdoor is exposed (TS'_x as in Table I) can the relevant users access the data with their attribute-associated keys. A trapdoor is set with release time $t \in \mathbb{F}_T$. TAFC implements an identity-based encryption (IBE) algorithm [15] to set time trapdoors, in which the time t is handled as an identity. Such mechanism realizes timed-release function without complex interaction between CA and owner.
3) The cloud implements the computations to expose corresponding trapdoors at each time t, without any possibilities of leaking unauthorized secrets. In order to realize this feature, CA will publish one time-related token TK_t at each time t, as shown in Fig. 2. The cloud obtains TK_t and transfers the status of each relevant time trapdoors. In this procedure, the cloud or other malicious user cannot obtain the relevant data, unless his/her attribute set satisfies the access policy of the data.
4) The decryption procedure is similar to original CP-ABE, while the major difference is: The secret should be re-constructed with the *exposed* time trapdoors, as well as attribute-associated private keys.

B. Access Policy and Time-Related Components

Access policy structure, and time-related components (time trapdoor TS and time token TK) are two of the important modifications over original CP-ABE based schemes. Therefore, we describe the concepts of these modification here.

1) Access Policy Structure: In TAFC, an access policy is over some attributes and release time. Fig. 3 shows an example of policy structure.

A structure $\mathcal{T}$ consists of several nodes of a policy tree, and some time trapdoors TS. A leaf node represents a certain attribute (In Fig. 3, A_0, $\cdots$, A_3 are the relevant attributes), and each non-leaf node represents a threshold gate ("AND", "OR", or the 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. Especially, $k_x = 1$ 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. 3, TS_2 is appended to a leaf node in order to restrict the attribute A_1, while TS_1 is upon a non-leaf node to restrict a sub-policy "$A_2 \wedge A_3$".

2) Time Trapdoors and Time Tokens: A time trapdoor (TS) is embedded in an access structure, such that the corresponding

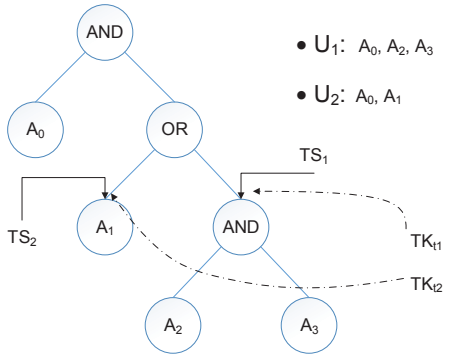

Fig. 3. Example of TAFC Access Structure

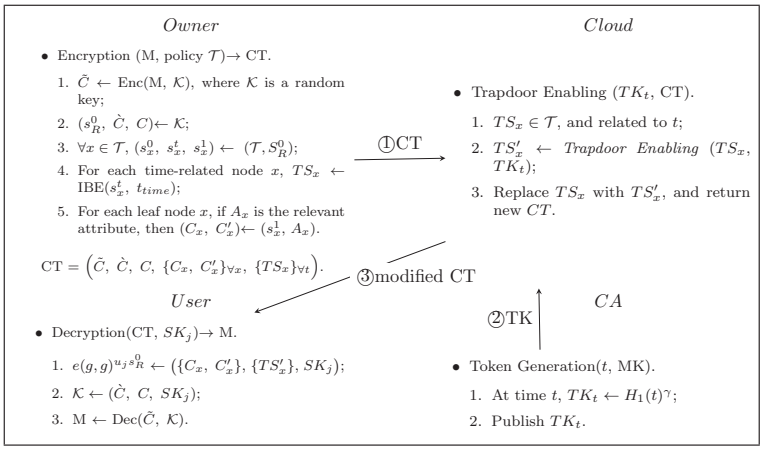

Fig. 4. Procedures Description of TAFC Construction

user's access permission is restricted by the status of TS. In this paper, we define two statuses for the time trapdoor: *exposed* or *unexposed*.

Unexposed. A trapdoor (TS) is *unexposed* if the intended users cannot access the corresponding secret through the trapdoor with their private 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'.

A trapdoor's status can be transferred (or *exposed*) with a relevant time token (TK_t). After TK_t is published at time t, anyone can transfer the status of corresponding time trapdoors.

In TAFC scheme, 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 service provider transfers each trapdoor's status from *unexposed* to *exposed* after obtaining the corresponding TK_t.

We can turn to Fig. 3 for instance: The trapdoor TS_1 is related to a time 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}.

C. Construction

Our scheme consists of six procedures: *setup*, *key generation*, *encryption*, *token generation*, *trapdoor exposure* and *decryption*. Fig. 4 depicts a brief description of our scheme (*setup* and *key generation* are not included in the figure).

1) Setup: The 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.

The 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 $\left(\beta, \gamma, g^\alpha\right)$. (Note that f, γ are used for timed-release function.)

2) Key Generation: For each user U_j with attribute set S_j, CA firstly chooses a random $u_j \in \mathbb{Z}_p^*$ as a unique identity for the user. Each attribute $Att_i \in S_j$ is assigned a random r_i. Then, CA computes the user's private key as:

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

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

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

Each node x in the policy $\mathcal{T}$ will obtain three secret parameters in this procedure as s_x^0, s_x^1 and s_x^τ. Here, s_x^0 is a parameter shared from its parent node, and s_x^1 is the one shared with its child node (or dealt with the relevant attribute, if x is a leaf node). The s_x^τ is a time-related parameter. Specially, 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 $s_R^0 \in \mathbb{Z}_p$. For each node x with s_x^0, the s_x^1 and s_x^τ are chosen as:

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

For each non-leaf node x with s_x^1, the 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 node y with a unique index $index_y$, the owner sets $s_y^0 = q_x(index_y)$.

For a trapdoor TS_x related to a release 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:

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

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:

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

4) Token Generation: At each time $t \in \mathbb{F}_T$, CA generates and publishes a time token TK_t as follows:

$$TK_t = H_1(t)^{\gamma} \tag{2}$$

5) Trapdoor Exposure: If the current time t is the same as the release time related to TS_x, the cloud will receive a corresponding token TK_t from CA. Then, the cloud runs this procedure to expose the trapdoor.

The procedure is run as: When the cloud receives TK_t, it queries all trapdoors associated to t. For all such $TS_x = \big(A_x,\ B_x\big)$, the cloud computes the *exposed* trapdoors as:

$$TS'_x = B_x - H_2(e(TK_t, A_x)) \tag{3}$$

If the procedure is run in right manner, we have $TS'_x = s_x^{\tau}$. The cloud replaces the trapdoor with TS'_x in each relevant CT.

6) Decryption: After querying CT from the cloud, a user U_j (with attribute set S_j) conducts this procedure with SK_j. As $TS'_x = s_x^{\tau}$, we can have:

$$\begin{cases} s_x^{\tau} = TS'_x & x \text{ is linked to exposed trapdoor} \\ s_x^{\tau} = 1 & \text{no trapdoor is set upon } x \end{cases} \tag{4}$$

The decryption procedure is run 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 as:

$$F_x = \Big(\frac{e(D_i, C_x)}{e(D'_i, C'_x)}\Big)^{s_x^{\tau}} = e(g,g)^{u_j s_x^1 s_x^{\tau}} = e(g,g)^{u_j s_x^0} \tag{5}$$

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 z such that $F_z \neq \perp$. If such S_x exists, and x is not embedded with an *unexposed* trapdoor, then the user computes:

$$F_x = \Big(\prod_{z \in S_x} F_z^{\prod_{y \in S_x, y \neq z} \frac{index_y}{index_y - index_z}}\Big)^{s_x^{\tau}} = e(g,g)^{u_j s_x^0} \tag{6}$$

Otherwise, F_x returns $\perp$.

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

$$\begin{aligned} \mathcal{K}' =& \frac{\grave{C}}{e(C,D)/F_R} = \mathcal{K}; \\ M' =& Dec(\tilde{C}, \mathcal{K}') = Dec(Enc(M, \mathcal{K}), \mathcal{K}) = M. \end{aligned} \tag{7}$$

V. Analysis

A. Security Analysis

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

1) Fine-Grained and Timed-Release Access Control: TAFC provides data owner with the capability to define an access policy according to flexible association of attributes and release time. 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 corresponding release time.

2) Security against Collusion Attack: In TAFC, each user's attribute-associated private key SK_j is blinded by a secret number $u_j \in \mathbb{Z}_p^*$. This mechanism is implemented to resist the collusion attack: The adversary cannot combine different SK to forge a new private key associated with a larger set of attributes, therefore, the collusion will not bring more privileges to the adversary.

3) Data Confidentiality: From the cryptography perspective, the data confidentiality of TAFC relies on the security of CP-ABE [6] and IBE [15], both of which are proved secure under DBDH assumption in these literatures. In decryption procedure of our scheme, the F_x is computed as $\big(e(g,g)^{u_j s_x^1}\big)^{s_x^{\tau}}$, in which $e(g,g)^{u_j s_x^1}$ is protected by CP-ABE algorithm, and s_x^{τ} is protected by IBE. An adversary cannot successfully guess F_x in an innegligible probability, if he/she has no information of either $e(g,g)^{u_j s_x^1}$ or s_x^{τ}. Thus, TAFC well preserves the data confidentiality property of CP-ABE and IBE.

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.

B. 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. [12] (denoted as ***LoTAC***), and an approach based on CP-ABE, where time is handled as attribute (denoted as ***TasA***). Since the performance difference is mainly on communication and computation cost of CA and data owner, we analyze these two aspects as follows.

1) CA's Cost for Timed-Release Function: Fig. 5 and Fig. 6 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 private key to each user at each time, meaning 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. 6. 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, TAFC shows its advantage on CA's cost reduction, when the access control system includes large amount of users and shared data.

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 the real world, 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. 7 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 com-

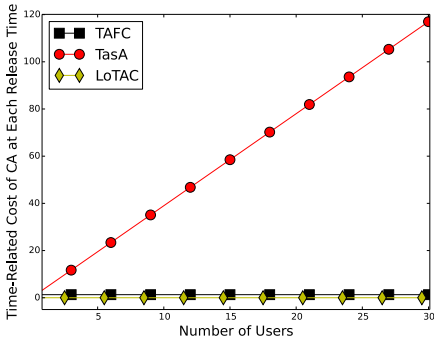

Fig. 5. Cost of CA versus Number of Users

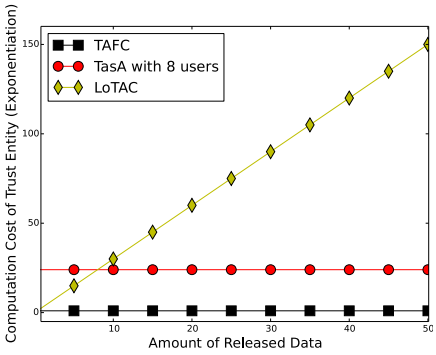

Fig. 6. Computation Overhead of Trust Entity versus Amount of Released Data

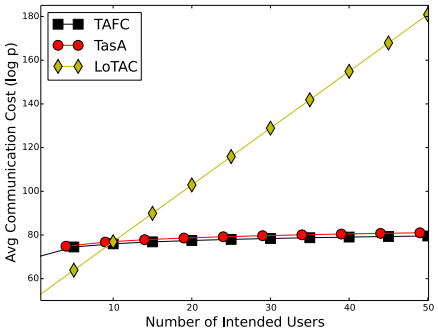

Fig. 7. Cost of Owner versus Number of Intended Users

munication complexity of data owner when the access privilege should be released to quite a number of users.

From the performance analysis on various aspects, we can find that TAFC well tolerate 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.

VI. Conclusion

This paper aims at fine-grained access control for time-sensitive data in cloud storage. One challenge is to simultaneously achieve flexible timed release and fine granularity with lightweight overhead, which is not provided in related work. In this paper, we propose 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. The analysis shows that our scheme can protect the confidentiality of time-sensitive data, with a lightweight overhead on both CA and data owners, thus well suits the practical large-scale access control system for cloud storage.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No. 61379129 and No. 61170231.

References

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

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

[3] 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 Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131–143, 2013.

[4] Z. Zhou, H. Zhang, Q. Zhang, Y. Xu, and P. Li, "Privacy-preserving granular data retrieval indexes for outsourced cloud data," in *Proceedings of the 2014 IEEE Global Communications Conference (GLOBECOM2014)*, pp. 601–606, IEEE, 2014.

[5] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Reliable re-encryption in unreliable clouds," in *Proceedings of the 2011 IEEE Global Communications Conference (GLOBECOM2011)*, pp. 1–5, IEEE, 2011.

[6] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 28th IEEE Symposium on Security and Privacy (S&P2007)*, pp. 321–334, IEEE, 2007.

[7] E. Bertino, P. A. Bonatti, and E. Ferrari, "TRBAC: A temporal role-based access control model," *ACM Transactions on Information and System Security*, vol. 4, no. 3, pp. 191–233, 2001.

[8] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," tech. rep., Massachusetts Institute of Technology, 1996.

[9] K. Yuan, Z. Liu, C. Jia, J. Yang, and S. Lv, "Public key timed-release searchable encryption," in *Proceedings of the 2013 Fourth International Emerging Intelligent Data and Web Technologies (EIDWT2013)*, pp. 241–248, IEEE, 2013.

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

[11] L. Xu, F. Zhang, and S. Tang, "Timed-release oblivious transfer," *Security and Communication Networks*, vol. 7, no. 7, pp. 1138–1149, 2014.

[12] E. Androulaki, C. Soriente, L. Malisa, and S. Capkun, "Enforcing location and time-based access control on cloud-stored data," in *Proceedings of the 2014 IEEE 34th International Distributed Computing Systems (ICDCS2014)*, pp. 637–648, IEEE, 2014.

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

[14] X. Zhu, S. Shi, J. Sun, and S. Jiang, "Privacy-preserving attribute-based ring signcryption for health social network," in *Proceedings of the 2014 IEEE Global Communications Conference (GLOBECOM2014)*, pp. 3032–3036, IEEE, 2014.

[15] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology (CRYPTO2001)*, pp. 213–229, Springer, 2001.