Time-Domain Attribute-Based Access Control for Cloud-Based Video Content Sharing: A Cryptographic Approach

Kan Yang, Zhen Liu, Xiaohua Jia, *Fellow, IEEE*, and Xuemin Sherman Shen, *Fellow, IEEE*

***Abstract*—With the ever-increasing demands on multimedia applications, cloud computing, due to its economical but powerful resources, is becoming a natural platform to process, store, and share multimedia contents. However, the employment of cloud computing also brings new security and privacy issues as few public cloud servers can be fully trusted by users. In this paper, we focus on how to securely share video contents to a certain group of people during a particular time period in cloud-based multimedia systems, and propose a cryptographic approach, a provably secure time-domain attribute-based access control (TAAC) scheme, to secure the cloud-based video content sharing. Specifically, we first propose a provably secure time-domain attribute-based encryption scheme by embedding the time into both the ciphertexts and the keys, such that only users who hold sufficient attributes in a specific time slot can decrypt the video contents. We also propose an efficient attribute updating method to achieve the dynamic change of users' attributes, including granting new attributes, revoking previous attributes, and regranting previously revoked attributes. We further discuss on how to control those video contents that can be commonly accessed in multiple time slots and how to make special queries on video contents generated in previous time slots. The security analysis and performance evaluation show that TAAC is provably secure in generic group model and efficient in practice.**

***Index Terms*—ABAC, cloud computing, MA-CP-ABE, multimedia, time-domain, time-domain attribute-based access control (TAAC), video content sharing.**

I. Introduction

With the rapid development of communication technologies and mobile devices, video applications (e.g., video chat, video conference, movies, short sight, etc.) have become more and more popular in our daily life. Meanwhile, the demands on video quality and user experience have also been

Manuscript received September 15, 2015; revised December 21, 2015; accepted January 10, 2016. Date of publication February 29, 2016; date of current version April 15, 2016. The guest editor coordinating the review of this manuscript and approving it for publication was Prof. Honggang Wang.

K. Yang and X. S. Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: kan.yang@uwaterloo.ca; sshen@uwaterloo.ca).

Z. Liu is with Security and Data Sciences, Hong Kong Applied Science and Technology Research Institute Company Limited, Hong Kong, China (e-mail: liu.zhen@my.cityu.edu.hk).

X. Jia is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China (e-mail: csjia@cityu.edu.hk).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org provided by the authors. This includes formal definitions for bilinear pairing, access structures, and linear secret sharing schemes (LSSS); the state tree construction method in the proposed time-domain attribute-based access control (TAAC) scheme; and the detailed formal security proof of the scheme. This material is 201 KB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2016.2535728

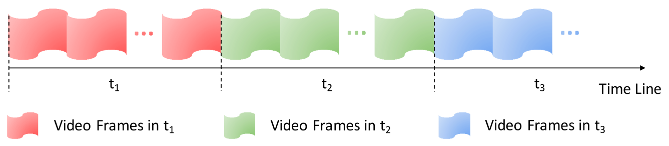


Fig. 1. Example of live video streaming in multiple time periods.

increasing significantly in many video applications, such as Ultra-high definition (UHD) live streaming, 3-D movies, instant high definition (HD) video messages, etc. The ever-increasing demands pose great challenges on video processing, coding, presentation as well as communication, especially when the resources of media devices (e.g., bandwidth, power and computation) are limited. Cloud computing, due to its flexible, scalable and economic resources [1], is a natural fit for storing, processing and sharing multimedia contents [2]–[4].

In cloud-based multimedia systems [5], some contents may be time-sensitive and can only be accessed by a certain group of people during a particular time period. For example, as shown in Fig. 1, if a user only purchases a live streaming service for time period t_2, this user may be granted to access the live streaming or videos recorded in time period t_2. However, this user does not have any permission to access the live streaming in time period t_3 or videos recorded in time periods t_1 and t_3 when he/she does not purchase the service. Moreover, during each time period, users may purchase different types of services, e.g., live streaming service, regular recorded video service, HD recorded video service, UHD recorded video service, etc. Therefore, it is necessary to achieve fine-grained time-domain access control for diverse video content sharing.

When outsourcing video contents into the cloud, it is not easy to achieve fine-grained access control especially in time-domain, as the owners of video contents are not able to control their data as on their own servers. The untrustworthy cloud servers further make this issue more challenging, because: 1) cloud servers may not be fully trusted by the owners to control the access of their video contents; and 2) cloud servers may also be curious about the stored video contents. Thus, existing server-based access control methods (e.g., access control lists) are not applicable for cloud-based video content sharing. A possible approach is to encrypt video contents and only authorized users are given decryption keys. In the above mentioned example in Fig. 1, all the video frames in t_1 are encrypted by one key, while all the video frames in t_2 are encrypted by another key.

However, due to the large volume of video contents and the performance requirements (e.g., speed, visual quality, compression friendliness, etc.), traditional encryption methods (e.g., AES, DES, IDEA, RSA, etc.) may not be suitable for data encryption. Some video encryption algorithms [6]–[9] are proposed to balance the privacy and the quality of video contents. When video contents are encrypted by using these video encryption algorithms with session keys, the access control of video contents becomes the control of session key distribution in time-domain, which is a challenging problem due to the large number of users. Hence, traditional public key encryption [10]–[12] may produce multiple copies of ciphertexts for each session key, the number of which is proportional to the number of users. To solve this problem, attribute-based encryption (ABE) methods [13]–[15] can be applied for session key encryption, where only one copy of encrypted session key is generated for all the users. However, these ABE methods do not take the time into consideration, and hence the main challenging issue to achieve time-domain access control is how to embed the time into the ABE.

Another challenging issue to control video content sharing in time-domain is the dynamic change of attributes. During different time periods, users may purchase different types of services and accordingly be assigned various sets of attributes. In each time period, a user may be *entitled* some new attributes, or *revoked* some attributes, or *re-granted* some previously revoked attributes. Existing attribute revocation methods [16]–[19] need to re-encrypt all the previous data so that all the new coming users may still be able to decrypt the previous data if their attributes satisfy data access policies. Hence, these methods may not be suitable for fine-grained video content sharing within a certain time period.

In this paper, we focus on how to securely share video contents to a certain group of people during a particular time period in cloud-based multimedia systems, and propose a cryptographic approach, a provably secure Time-domain Attribute-based Access Control (TAAC) scheme, to control the access of session keys that are used to encrypt video contents. Specifically, TAAC is constructed based on a typical multi-authority ciphertext policy attribute-based encryption (CP-ABE) scheme proposed by Lewko and Waters [15], so TAAC can support attributes issued from multiple attribute authorities as well. In order to support time-domain access control, we embed the time into both the ciphertexts and the keys in the multi-authority CP-ABE scheme in [15], such that only those users who hold sufficient attributes in a specific time period can decrypt the data. To deal with the attribute update problem, we follow the ideas in identity-based encryption revocation [20] and divide the time into slots. At the beginning of each time slot, each authority can revoke/re-grant any attribute in its domain from/to any user. Different from the revocation method in [20], our scheme does not allow any new users to decrypt the data by making use of the update keys published in the previous time intervals. Moreover, compared with existing revocation methods [16]–[19], our scheme does not need any ciphertext re-encryption/update, which can significantly improve the efficiency especially in large scale cloud-based multimedia systems.

The contributions of this paper are summarized as follows.

1) We formulate the time-domain video content sharing problem in cloud-based multimedia systems and propose a cryptographic approach—TAAC scheme.
2) We propose a provably secure time-domain ABE scheme by embedding the time into both the ciphertexts and the keys, such that only users who hold sufficient attributes in a specific time period can decrypt the data.
3) We propose an efficient attribute updating method to achieve the dynamic change of users' attributes, including granting new attributes, revoking previous attributes and re-granting previously revoked attributes.

The remainder of this paper is organized as follows. We first review some existing work that may be related to secure video sharing and time-domain access control in Section II. Then, we describe the system model of video content sharing and define a framework as well as a security model to achieve TAAC in Section III. Section IV describes the technique overview and the detailed construction of TAAC and Section V shows how to solve the attribute dynamic updating problem. Section VI provides security analysis and performance evaluation of TAAC. In Section VII, we also discuss on how to achieve access control of video contents that are commonly accessed in multiple time slots and how to make special queries on video contents generated in previous time slots. Finally, the conclusion of this paper is drawn in Section VIII.

II. Related Work

Cloud-based multimedia content sharing is one of the most significant services in cloud-based multimedia systems. In [21], some security and privacy issues of multimedia services are proposed by exploring the multimedia-oriented mobile social network. In [22], ABE is adopted to share scalable media based on the attributes rather than the names of the consumers. Some works focus on dealing with the security issues in wireless sensor networks [23], [24] and crowdsourcing networks [36], [37], which is important in multimedia data collection and transmission.

Multi-authority CP-ABE [15], [25]–[27] is regarded as one of the most appropriate techniques for access control of data stored in the cloud, because it allows the data owner to define and enforce the access policy over attributes from multiple attribute authorities. For example, data owners may share the data using access policy "Google.Engineer AND University X.Alumni" where the attribute "Engineer" is issued by Google and the attribute "Alumni" is issued by the University X. Among these existing multi-authority CP-ABE schemes, the scheme proposed by Lewko and Waters [15] is widely accepted due to its high scalability and security. However, we cannot directly apply this scheme to achieve time-domain access control, because the access policy and the encryption in [15] does not involve in any time parameters.

To achieve access control in time-domain, the main challenging issue is how to embed the time into the ABAC. A straightforward method is to take time as an attribute and embed it into the access policy by changing the access policy from P to $(P \text{ AND } Attr(t))$. Here comes the problem: Taking the most popular multiple authority ABE scheme proposed by Lewko and Waters [15] as an example, if the policy of a ciphertext CT is $(A \ \mathsf{AND}\ B \ \mathsf{AND}\ Attr(t))$, a user who holds A and $Attr(t)$ at time t cannot decrypt CT. Suppose the user obtains the attribute B at time $t+1$, the user then can decrypt CT with A, B and $Attr(t)$ at time $t+1$, because the attribute B is only associated with the user identity uid which is the same as the attributes A and $Attr(t)$. To solve this problem, one approach is to associate the time with each attribute as $(A, A(t)), (B, B(t+1))$, etc. That is, for each attribute, there is a corresponding time attribute issued by the same authority, which doubles the size of attribute universe. Moreover, for each time slot t_j, the authority needs to generate and distribute all the corresponding time-attributes at t_j to each user, which also eliminates the advantages of ABE.

Towards the attribute dynamic updating problem, some methods[16]–[19], [28]–[30] are proposed to deal with the attribute revocation in ABE systems. For example, in [16], the authors propose a revocable CP-ABE method, by using ciphertext delegation and piecewise property of secret keys. For each time slot, the authority will generate a set of update keys (the other piece of secret keys) according to the revocation list. Then, all the ciphertexts should be re-encrypted with a new access control policy, i.e., the principal access policy $(\mathbb{A}, \rho)$ is not changed but the additional access policy $(\mathbb{B}, \beta)$ will be changed during each time slot. They also proposed a ciphertext delegation method for changing/re-encrypting a ciphertext encrypted under a certain access policy to a more restrictive policy using only public information.

However, the method in [16] requires to re-encrypt the ciphertexts such that for those new users entitled with sufficient attributes in later time slots can still decrypt the ciphertext. However, in our problem, the time-domain access control does not allow any new users to decrypt the ciphertexts generated in the previously time periods. If simply removing the ciphertext update/re-encryption procedure in [16], it still cannot achieve the requirement of time-domain access control, because a user who is entitled x at a later time slot $t' > t$ can make use of the update keys for (x, t) to decrypt the previous published ciphertexts. Furthermore, their scheme is proposed for single authority ABE systems, and it is not straightforward to extend to multi-authority systems.

III. System Model and Framework Definitions

In this section, we first introduce the system model of cloud-based video content sharing and then define the framework as well as its security model for time-domain access control. For convenience, some notations are summarized in Table I.

A. System Model of Cloud-Based Video Content Sharing

We consider a cloud-based video content sharing system as follows. The time in the system is slotted and the time space is defined as $\mathcal{T} = \{1, 2, \dots\}$. Without loss of generality, the system initializes its time to 0, and then increases it by 1 for the next time slot (for any time slot $t \in \mathcal{T}$, $t-1$ is its last time slot and $t+1$ is its next time slot). As shown in Fig. 2, the system model consists of four types of entities: attribute authorities

TABLE I
Notations

Symbol	Meaning
$\mathcal{T}$	Slotted Time Space
$\mathbb{D}$	Index Set of all the Authorities
$\mathcal{U}_d$	Attribute Set managed by AA_d (the d^{th} AA)
$\mathcal{U}$	Universe Attribute Set ($\mathcal{U} = \bigcup_d \mathcal{U}_d$)
$\mathsf{CT}_{\mathbb{A},t_e}$	Ciphertext with access policy $\mathbb{A}$ at time slot t_e
gid	User Global Identity
S_{gid}	Set of all attributes user gid has EVER had
$S_{\mathrm{gid},t}$	Attribute Set user gid possesses at time slot t
$SK_{\mathrm{gid},x}$	Secret Key of attribute x for user gid
ST_x	State Tree of Attribute x
$\mathsf{UL}_{x,t}$	Update List for Attribute x at time slot t
$\mathsf{UK}_{x,t}$	Update Key of Attribute x at time slot t
$\mathsf{DK}_{\mathrm{gid},x,t}$	Decryption Key of attribute x at t for user gid

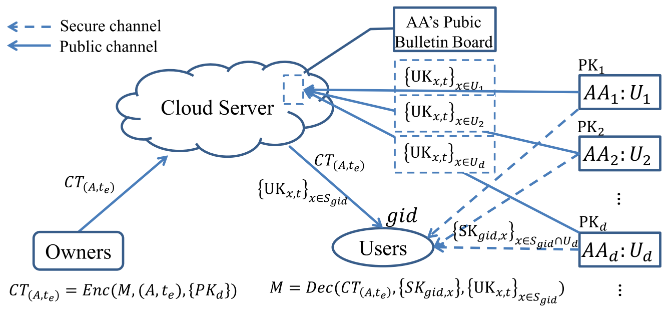

Secure channel
Public channel
AA's Pubic Bulletin Board
Cloud Server
$\{\mathrm{UK}_{x,t}\}_{x\in U_1}$
$\{\mathrm{UK}_{x,t}\}_{x\in U_2}$
$\{\mathrm{UK}_{x,t}\}_{x\in U_d}$
PK_1
$AA_1: U_1$
PK_2
$AA_2: U_2$
PK_d
$AA_d: U_d$
$CT_{(A,t_e)}$
$CT_{(A,t_e)}$
$\{\mathrm{UK}_{x,t}\}_{x\in S_{gid}}$
gid
$\{\mathrm{SK}_{gid,x}\}_{x\in S_{gid}\cap U_d}$
Owners
Users
$CT_{(A,t_e)} = Enc(M, (A, t_e), \{PK_d\})$
$M = Dec(CT_{(A,t_e)}, \{SK_{gid,x}\}, \{\mathrm{UK}_{x,t}\}_{x\in S_{gid}})$

Fig. 2. System model of cloud-based video content sharing.

(AAs), the cloud server (server), data owners (owners) and data consumers (users).

Each AA is an independent attribute authority that is responsible for entitling, revoking or re-granting attributes to/from/to users according to their roles or identities in its domain. Each attribute is associated with a single AA, but each AA can manage an arbitrary number of attributes. In practice, attributes belong to different authorities can be identified by encoding the attributes with different prefix. For simplicity, we use a mapping $\phi : \mathcal{U} \mapsto \mathbb{D}$ to map any attribute to the index of corresponding authority. Each AA has full control over the structures and semantics of its attributes, and maintains a state tree and an update list for each attribute in its domain. Each AA is also responsible for issuing secret keys to users when they are entitled attributes in its domain, and publishing update keys for each attribute at the beginning of each time slot to reflect the users' possessions of the attribute at this time slot.

Data owners define access policies $\mathbb{A}$ on attributes from multiple authorities and a time slot t_e, then encrypt the video content under the policies before outsourcing to cloud servers. They do not rely on the server to enforce the access policy. Instead, the *access control happens inside the cryptography*: only the users who possess eligible attributes (satisfying the access policy $\mathbb{A}$) at a particular time slot t_e can decrypt the ciphertext $\mathsf{CT}_{\mathbb{A},t_e}$ associated with $(\mathbb{A}, t_e)$.

Each user has a global identity gid in the system. Let S_{gid} denote all attributes that user gid has ever had. As the user's attributes may change dynamically from time slot to time slot,

$S_{\mathrm{gid},t}$ is used to denote the attribute set that *the user gid possesses at time slot* t. When a user gid is entitled an attribute x, he will be issued a corresponding *secret key* $\mathsf{SK}_{\mathrm{gid},x}$. However, the secret keys of a user gid are insufficient to decrypt a ciphertext encrypted under $(\mathbb{A}, t_e)$ even when the corresponding attributes satisfy $\mathbb{A}$. The user has to obtain a set of *update keys* at each time slot t (i.e., $\mathsf{UK}_{x,t}$) from the corresponding authorities. Based on the received update keys and his secret keys, the user can compute *decryption key*s for each time slot t, and further uses them to decrypt the ciphertext.

B. Framework of TAAC

Let $AA_1, AA_2, \ldots$ be attribute authorities and $\mathbb{D} = \{1, 2, \ldots\}$ be the index set of the AAs. Let $d(d \in \mathbb{D})$ denote the index of attribute authority AA_d and $\mathcal{U}_d$ be the set of attributes managed by AA_d, where $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset$ for all $i \neq j \in \mathbb{D}$. The attribute universe of the system is defined as $\mathcal{U} = \bigcup_{d\in\mathbb{D}} \mathcal{U}_d$. The AAs are independent from each other and do not need to know the existence of others. The framework of TAAC is defined as follows.

Definition 1: TAAC is a collection of the following algorithms: $\mathsf{Global\ Setup}$, $\mathsf{Authority\ Setup}$, $\mathsf{SKeyGen}$, $\mathsf{UKeyGen}$, $\mathsf{DKeyCom}$, $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$.

- $GlobalSetup(\lambda) \rightarrow \mathsf{GPP}$: The global setup algorithm takes the security parameter λ as input. It outputs the global public parameters GPP.
- $AuthoritySetup(\mathsf{GPP}, \mathcal{U}_d) \rightarrow (\mathsf{PK}_d, \mathsf{MSK}_d)$: The authority setup algorithm is run by each AA. It takes as inputs the global public parameters GPP, an attribute domain $\mathcal{U}_d$. It outputs the authority's public key PK_d and the master secret key MSK_d. In addition, for each attribute $x \in \mathcal{U}_d$, this algorithm initializes the state tree ST_x, and initializes the update list of x to empty.
- $SKeyGen(gid, x, \mathsf{ST}_x, \mathsf{GPP}, \mathsf{MSK}_{\phi(x)}) \rightarrow (\mathsf{SK}_{\mathrm{gid},x}, \mathsf{ST}_x)$: The secret key generation algorithm is run by each AA. It takes as inputs the user's global identity gid, an attribute x and its state tree ST_x, the global public parameters GPP and the master secret key $\mathsf{MSK}_{\phi(x)}$. The algorithm outputs a secret key $\mathsf{SK}_{\mathrm{gid},x}$ for the (attribute, identity) pair (x, gid), and an updated state tree ST_x.
- $UKeyGen(t, x, \mathsf{ST}_x, \mathsf{UL}_{\ x,t}, \mathsf{GPP}, \mathsf{MSK}_{\phi(x)}) \rightarrow (\mathsf{UK}_{x,t})$: At each time slot t, for each attribute $x \in \mathcal{U}_{\phi(x)}$, the authority $AA_{\phi(x)}$ runs the update key generation algorithm *once* by taking as inputs the state tree ST_x of x, the update list $\mathsf{UL}_{\ x,t}$ of x at time slot t, the global public parameters GPP, and its master secret key $\mathsf{MSK}_{\phi(x)}$. The algorithm outputs the *update key* $\mathsf{UK}_{x,t}$ for x at t, which will be published on the public bulletin board of the authority, which is stored on cloud servers.
- $DKeyCom(\mathsf{SK}_{\mathrm{gid},x}, \mathsf{UK}_{x,t}) \rightarrow (\mathsf{DK}_{\mathrm{gid},x,t})$ or $\perp$: For any time slot t and any attribute x, a user gid can run the decryption key computation algorithm with secret key $\mathsf{SK}_{\mathrm{gid},x}$ and update key $\mathsf{UK}_{x,t}$ as inputs. The algorithm outputs a *decryption key* $\mathsf{DK}_{\mathrm{gid},x,t}$ implying that gid possesses x at t, or a special symbol $\perp$ implying that gid does not possess x at t.
- $Encrypt(M, t_e, \mathbb{A}, \mathsf{GPP}, \{\mathsf{PK}_d\}) \rightarrow (\mathsf{CT})$: The encryption algorithm is run by data owners. It takes as inputs a message M, a time slot t_e, an access policy $\mathbb{A}$ over attributes from multiple authorities, the global public parameters GPP, and the public keys $\{\mathsf{PK}_d\}$ related to $\mathbb{A}$. It outputs a ciphertext CT which includes $\mathbb{A}$ and t_e.
- $Decrypt(\mathsf{CT}, \mathsf{GPP}, \{\mathsf{PK}_d\}, \{\mathsf{DK}_{\mathrm{gid},x,t}\}_{x\in S_{\mathrm{gid},t}}) \rightarrow (M)$ or $\perp$: The decryption algorithm is run by users. It takes as inputs a ciphertext CT which includes access policy $\mathbb{A}$ and time slot t_e, the global public parameters GPP, the public keys $\{\mathsf{PK}_d\}$ related to $\mathbb{A}$, and decryption keys $\{\mathsf{DK}_{\mathrm{gid},x,t}\}_{x\in S_{\mathrm{gid},t}}$ corresponding to a (global identity, time slot) pair (gid, t). The algorithm outputs a message M or a special symbol $\perp$ implying decryption failure.

Correctness: The system must satisfy the following correctness property: for all access policies $\mathbb{A}$ over the whole attribute universe $\mathcal{U}$, all time slots $t_e \in \mathcal{T}$, all messages M, all global identities gid, and all possible states trees $\{\mathsf{ST}_x\}_{x\in S_{\mathrm{gid}}}$ and update lists $\{\mathsf{UL}_{x,t_e}\}_{x\in S_{\mathrm{gid}}}$, if S_{gid,t_e} satisfies $\mathbb{A}$, the following experiment returns 1 with probability 1:

$$(\mathsf{GPP}) \leftarrow \mathsf{GlobalSetup}(\lambda),$$
$$\{(\mathsf{PK}_d, \mathsf{MSK}_d) \leftarrow \mathsf{Authority\ Setup}(\mathsf{GPP}, \mathcal{U}_d)\},$$
$$\{(\mathsf{SK}_{\mathrm{gid},x}, \mathsf{ST}_x) \leftarrow \mathsf{SKeyGen}(gid, x, \mathsf{ST}_x, \mathsf{GPP}, \mathsf{MSK}_{\phi(x)})\}_{x\in S_{\mathrm{gid}}},$$
$$\{\mathsf{UK}_{x,t_e} \leftarrow \mathsf{UKeyGen}(t_e, x, \mathsf{ST}_x, \mathsf{UL}_{x,t_e}, \mathsf{GPP}, \mathsf{MSK}_{\phi(x)})\}_{x\in S_{\mathrm{gid}}},$$
$$\{\mathsf{DK}_{\mathrm{gid},x,t_e} \leftarrow \mathsf{DKeyCom}(\mathsf{SK}_{\mathrm{gid},x}, \mathsf{UK}_{x,t_e})\}_{x\in S_{\mathrm{gid}}},$$
$$\mathsf{CT} \leftarrow \mathsf{Encrypt}(M, t_e, \mathbb{A}, \mathsf{GPP}, \{\mathsf{PK}_d\}).$$

If $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{GPP}, \{\mathsf{PK}_d\}, \{\mathsf{DK}_{\mathrm{gid},x,t}\}_{x\in S_{\mathrm{gid},t_e}}) = M$, then it returns 1; Otherwise, it returns 0.

C. Security Model of TAAC

In cloud-based video content sharing systems, we consider the case that the server may send data to the users who do not have access permission. The server is also curious about the video contents. The users, however, are dishonest and may collude to obtain unauthorized access to video contents. Some of the AAs can be corrupted or compromised by the attackers. The security of TAAC is defined by the following game run between a challenger and an adversary $\mathcal{A}$.

Setup.

1) The challenger runs $\mathsf{GlobalSetup}$ and gives the output GPP to $\mathcal{A}$.
2) $\mathcal{A}$ specifies index set $\mathbb{D}_c \subset \mathbb{D}$ as the corrupted AAs. For good (non-corrupted) AAs in $\mathbb{D} \setminus \mathbb{D}_c$, the challenger runs the $\mathsf{AuthoritySetup}$ algorithm and gives the output $\mathsf{PK}_d(d \in \mathbb{D} \setminus \mathbb{D}_c)$ to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ can obtain secret keys and update keys by querying the following oracles:

1) $\mathsf{SKQ}(gid, x)$: $\mathcal{A}$ makes secret key queries by submitting pairs (gid, x), where gid is a global identity and x is an attribute belonging to some good authority (i.e., $\phi(x) \in \mathbb{D} \setminus \mathbb{D}_c$). The challenger runs $\mathsf{SKeyGen}$ to return a secret key $\mathsf{SK}_{\mathrm{gid},x}$ to $\mathcal{A}$ and update the state tree ST_x.

2) $\mathsf{UKQ}(t, x, \mathsf{UL}_{x,t})$: $\mathcal{A}$ makes update key queries by submitting tuples $(t, x, \mathsf{UL}_{x,t})$, where $t \in \mathcal{T}$ is a time slot, x is an attribute belonging to some good authority (i.e., $\phi(x) \in \mathbb{D} \setminus \mathbb{D}_c$), and $\mathsf{UL}_{x,t}$ is a valid update list of x at t. The challenger runs $\mathsf{UKeyGen}$ to return an update key $\mathsf{UK}_{x,t}$ to $\mathcal{A}$.

Challenge Phase. $\mathcal{A}$ submits to the challenger two equal-length messages M_0, M_1, an access policy $\mathbb{A}^*$ over $\mathcal{U}$, and a time slot $t^* \in \mathcal{T}$. In addition, $\mathcal{A}$ must also give the challenger the public keys $\{\mathsf{PK}_d\}$ for any corrupted authorities whose attributes appear in $\mathbb{A}^*$. The challenger flips a random coin $\beta \in \{0, 1\}$ and sends to $\mathcal{A}$ an encryption of M_β under $(\mathbb{A}^*, t^*)$.

Phase 2. $\mathcal{A}$ makes further queries as in Phase 1.

Guess. $\mathcal{A}$ submits a guess β' for β.

$\mathcal{A}$ wins the game if $\beta' = \beta$ under the following restrictions.

1) $\mathsf{UKQ}(t, \cdot, \cdot)$ can be queried on time slot which is greater than or equal to the time slot of all previous queries, i.e., the adversary is allowed to query only in non-decreasing order of time slot.[1] Also, for any pair (t, x), $\mathsf{UKQ}(t, x, \cdot)$ can be queried only once (because at each time slot t, for any attribute x, the corresponding authority will publish the corresponding update key $\mathsf{UK}_{x,t}$ only once).
2) For any queried gid, S_{gid,t^*} does not satisfy $\mathbb{A}^*$.

The advantage of $\mathcal{A}$ is defined as $|\Pr[\beta = \beta'] - 1/2|$.

Definition 2: TAAC is secure in generic group model if for all polynomial-time adversary $\mathcal{A}$ in the above game, the advantage of $\mathcal{A}$ is negligible.

IV. TAAC: Time-Domain Attribute-based Access Control

In this section, we first go through the main technical ideas of secure cloud-based video content sharing, and then describe the detailed construction of TAAC.

A. Technique Overview

Due to the large volume of video contents and the performance requirements (e.g., speed, visual quality, compression friendliness, etc.), video contents are firstly encrypted by using video encryption methods [6]–[9] with session keys. To support time-domain access control of video contents, we control the distribution of session keys by proposing a new time-domain ABE method, which embeds the time into both the ciphertexts and the keys of the multi-authority CP-ABE in [15]. Specifically, the system first defines a slotted time space $\mathcal{T}$. For each attribute $x \in \mathcal{U}_d$, AA_d maintains a state tree ST_x (a binary tree that describes the users who have ever hold the attribute x, and the construction of a state tree is described in the Supplementary File) and an update list $\mathsf{UL}_{x,t}$. The state tree ST_x is updated when a user is entitled x, and the update list $\mathsf{UL}_{x,t}$ (initially empty) is updated at time slot t if any user is revoked or re-granted x at that time slot.

When a user gid is entitled an attribute $x \in \mathcal{U}_d$, AA_d issues a *secret key* $\mathsf{SK}_{\mathrm{gid},x}$ to gid and updates ST_x. When the owner encrypts a session key κ, besides specifying an access policy $\mathbb{A}$ over attribute universe $\mathcal{U}$, he also specifies a time slot $t_e \in \mathcal{T}$, which implies that only the users *who possess eligible attributes at time slot* t_e can decrypt the ciphertext. Specifically, a user gid who only has secret keys $\{\mathsf{SK}_{\mathrm{gid},x}\}_{x \in S_{\mathrm{gid}}}$ is unable to decrypt a ciphertext encrypted by $(\mathbb{A}, t_e)$ even when S_{gid} satisfies $\mathbb{A}$. Besides, he has to obtain the *update key* for (x, t_e) (i.e., $\{\mathsf{UK}_{x,t_e}\}_{x \in S_{\mathrm{gid},t_e}}$) from the corresponding AAs. By using $\mathsf{SK}_{\mathrm{gid},x}$ and UK_{x,t_e}, the user gid can compute a *decryption key* $\mathsf{DK}_{\mathrm{gid},x,t_e}$ and use it to decrypt the ciphertext encrypted under $(\mathbb{A}, t_e)$, if S_{gid,t_e} satisfies $\mathbb{A}$. Only those users who hold sufficient attributes in a specific time slot can decrypt session keys of video contents.

To deal with the attribute revocation problem, we follow the ideas in identity-based encryption revocation [20] and divide the time into slots. At each time slot $t \in \mathcal{T}$, for any $x \in \mathcal{U}_d$, AA_d generates the update keys $\mathsf{UK}_{x,t}$ according to ST_x and $\mathsf{UL}_{x,t}$ so that only the users who possess x at time slot t are able to obtain valid $\mathsf{UK}_{x,t}$. Thus, by setting the update list $\mathsf{UL}_{x,t}$ and publishing the corresponding $\mathsf{UK}_{x,t}$ the authority can achieve dynamic change of x. However, a critical issue is how to prevent a user who is entitled x at a later time slot $t' > t$ from making use of the update keys for (x, t). To solve this issue, we assign a random number to each node in the state tree, which is embedded into both secret keys and update keys. Moreover, our proposed minimum cover set selection algorithm MinCSS guarantees that the previous released update keys will not cover any nodes in the paths of new users. These two mechanisms together can guarantee that the new user in our scheme cannot apply the previous released update keys to cancel the random number in its secret keys.

B. Construction of TAAC

Based on the algorithms defined in Section III-B, TAAC contains the following phases: System Initialization, Key Generation by AAs, Data Encryption by Owners, and Data Decryption by Users.

Phase 1: System Initialization

The system initialization consists of two steps: *Global Setup* and *Authority Setup*.

1) Global Setup

The system is initialized by running the global setup algorithm $\mathsf{Global\ Setup}$. Let G and G_T be a bilinear group of order p, where p is a prime.[2] Let g be a generator of G. The global public parameter is published as $\mathsf{GPP} = (p, g, e, H)$, where e is a bilinear pairing, and H is a hash function that maps global identities to elements of G.

2) Authority Setup

The authority setup algorithm $\mathsf{Authority\ Setup}$ is run by each authority $AA_d (d \in \mathbb{D})$. For each attribute $x \in \mathcal{U}_d$, the algorithm chooses two random exponents $\alpha_x, \beta_x, \gamma_x \in \mathbb{Z}_p$. The algorithm chooses a random element $\tau_x \in \mathbb{Z}_p$ as the seed of a pseudo random function F. Then, for each node v_x in the state tree ST_x,

[1]This captures the practical scenarios, where the authority will always generate and publish update keys for current time slot, so that the update keys can only be obtained in the non-decreasing order of time slot.

[2]Note that we consider groups in prime order in this paper due to the efficiency of group operations. Our methods can also be applied to the CP-ABE methods constructed with composite order bilinear groups in [15].

it sets a random number $R_{v_x} = F(\tau_x, v_x) \in G$, which will be used to generate the user's secret key. Let $H_d : \mathcal{U}_d \times \mathcal{T} \mapsto G$ be a hash function that maps (attribute, time slot) pairs in $\mathcal{U}_d \times \mathcal{T}$ to elements of G. The public key is set to

$$\mathsf{PK}_d = (\{e(g,g)^{\alpha_x}, g^{\beta_x}, g^{\frac{1}{\gamma_x}}\}_{x \in \mathcal{U}_d}, H_d)$$

and the master secret key is set to $\mathsf{MSK}_d = \{\alpha_x, \beta_x, \gamma_x\}$.

Phase 2: Key Generation by AAs

The key generation contains both *Secret Key Generation* and *Update Key Generation*.

1) Secret Key Generation

When a user gid is entitled an attribute $x \in \mathcal{U}_{\phi(x)}$, the corresponding authority $AA_{\phi(x)}$ runs the secret key generation algorithm $\mathsf{SKeyGen}$ to update ST_x and generate a secret key $\mathsf{SK}_{\mathrm{gid},x}$ for this user gid as follows.

a) Sets $u_{x,gid} = 2^{h_x} + ctr_x$, adds $(gid, u_{x,gid})$ to $List_x$ (assigning the leaf node $u_{x,gid}$, i.e., the most left empty leaf node, to gid), and sets $ctr_x = ctr_x + 1$.

b) Sets

$$\begin{aligned}\mathsf{SK}_{\mathrm{gid},x} = \{K_{\mathrm{gid},x,v_x} &= g^{\alpha_x} H(gid)^{\beta_x} (R_{v_x})^{\alpha_x r_{\mathrm{gid},x,v_x} + r'_{\mathrm{gid},x,v_x}} \\ K'_{\mathrm{gid},x,v_x} &= r_{\mathrm{gid},x,v_x}, \\ K''_{\mathrm{gid},x,v_x} &= (R_{v_x})^{\gamma_x r'_{\mathrm{gid},x,v_x}}\}_{v_x \in \mathsf{Path}(u_{x,gid})}\end{aligned}$$

where $\mathsf{Path}(u_{x,gid})$ is the set of nodes on the path from v_x to the root node (including both v_x and the root node), $r_{\mathrm{gid},x,v_x} \in \mathbb{Z}_p^*$ is randomly chosen.

Then, the authority $AA_{\phi(x)}$ sends the secret key $\mathsf{SK}_{\mathrm{gid},x}$ to the user gid. Note that only when a new attribute $x \in \mathcal{U}_{\phi(x)}$ is assigned to the user gid, a corresponding secret key $\mathsf{SK}_{\mathrm{gid},x}$ is issued to this user gid by the authority $AA_{\phi(x)}$. This secret key assignment happens at most *once* for each attribute on each user. When the attribute x is revoked or re-granted some time, the secret key $\mathsf{SK}_{\mathrm{gid},x}$ will not be removed or re-assigned to the user gid anymore.

2) Update Key Generation

At the beginning of each time slot t, for each attribute $x \in \mathcal{U}_{\phi(x)}$, the corresponding authority $AA_{\phi(x)}$ determines the update list $\mathsf{UL}_{x,t}$ and runs the update key generation algorithm $\mathsf{UKeyGen}$ *once* to generate and publish the update key $\mathsf{UK}_{x,t}$. The algorithm computes the minimum set of nodes $N_{x,t}$ that covers all the non-revoked users who possess x at t by running the Minimum Cover Set Selection algorithm MinCSS (which will be described in Section V), then for each $v_x \in N_{x,t}$ it chooses a random exponent $\xi_{v_x,t} \in \mathbb{Z}_p$. The update key for (x,t) is set to

$$\mathsf{UK}_{x,t} = \left\{\left(E_{v_x} = (R_{v_x})^{\alpha_x \gamma_x} H_{\phi(x)}(x,t)^{\xi_{v_x,t}}, \ E'_{v_x} = g^{\frac{\xi_{v_x,t}}{\gamma_x}}\right)\right\}_{v_x \in N_{x,t}}.$$

Then, all the update keys $\{\mathsf{UK}_{x,t}\}$ are published on the public bulletin board of the corresponding authority $AA_{\phi(x)}$ *at the beginning of the time slot* t. All the users in the system can access these update keys from the public bulletin board of each authority, which is also stored in the cloud. When all the users have downloaded the update keys or after a time period, the old update keys on the cloud server can be deleted in order to reduce storage overhead.

Phase 3: Data Encryption by Owners

The owner first encrypts the video content with a session key by using video encryption algorithms. It then runs the encryption algorithm $\mathsf{Encrypt}$ to encrypt the session key κ. The encryption algorithm is defined as:

$\mathsf{Encrypt}(\kappa, t_e, \mathbb{A} = (A,\rho), \mathsf{GPP}, \{\mathsf{PK}_d\}) \rightarrow (\mathsf{CT})$. κ is the session key, t_e is a time slot, $\mathbb{A}$ is the access policy which is expressed by an LSSS matrix (A,ρ), where A is an $m \times n$ matrix and ρ maps each row A_i of A to an attribute $\rho(i)$, and $\{\mathsf{PK}_d\}$ are the public keys related to (A,ρ), where $d \in \{\phi(\rho(i)) | 1 \le i \le m\}$. The algorithm chooses a random number $s \in \mathbb{Z}_p$ as the encryption secret.

Then, it chooses two random vectors $\vec{v}, \vec{u} \in \mathbb{Z}_p^n$, with s and 0 as the first entry respectively, and for each $i \in \{1, 2, \ldots, m\}$, it randomly picks $r_i \in \mathbb{Z}_p$. Let $\lambda_i = A_i \cdot \vec{v}$ and $\mu_i = A_i \cdot \vec{u}$. The ciphertext of the session key is

$$\begin{aligned}\mathsf{CT} = \langle\, (A,\rho), t_e, \ & C = \kappa \cdot e(g,g)^s \\ & \{C_{i,1} = e(g,g)^{\lambda_i} e(g,g)^{\alpha_{\rho(i)} r_i} \\ & C_{i,2} = g^{\mu_i} g^{\beta_{\rho(i)} r_i} \\ & C_{i,3} = g^{r_i}, \ C'_{i,3} = \left(g^{\frac{1}{\gamma_x}}\right)^{r_i} \\ & C_{i,4} = H_{\phi(\rho(i))}(\rho(i), t_e)^{r_i} \}_{i=1}^m \rangle.\end{aligned}$$

Then, data owners send both the ciphertext CT of κ and the video content encrypted under κ to cloud server.

Phase 4: Data Decryption by Users

All legal users can download any ciphertexts they are interested in. But only the users who possess eligible attributes (satisfying the access policy $\mathbb{A}$) at a particular time slot t_e can decrypt the ciphertext associated with $(\mathbb{A}, t_e)$. The decryption phase consists of two steps: *Decryption Key Computation* and *Ciphertext Decryption*.

1) Decryption Key Computation

At each time slot t, each user can get update keys for each attribute it possesses at this time slot from the public bulletin boards of the authorities. For each attribute x it possesses at time slot t, the user gid computes a decryption key $\mathsf{DK}_{\mathrm{gid},x,t}$ for this attribute by running the algorithm $\mathsf{DKeyCom}$ constructed as follows.

$\mathsf{DKeyCom}(\mathsf{SK}_{\mathrm{gid},x}, \mathsf{UK}_{x,t}) \rightarrow (\mathsf{DK}_{\mathrm{gid},x,t})$ or $\perp$. If $x \in S_{\mathrm{gid},t}$, i.e., gid possesses x at t, then there exists a unique v_x such that $v_x \in \mathsf{Path}(u_{x,gid}) \wedge v_x \in N_{x,t}$ (this is guaranteed by the Minimum Cover Set Selection algorithm MinCSS), the algorithm sets the decryption key for attribute x of gid at t to

$$\begin{aligned}\mathsf{DK}_{\mathrm{gid},x,t} = (D_{\mathrm{gid},x,t} &= K_{\mathrm{gid},x,v_x}, \\ D'_{\mathrm{gid},x,t} &= (E_{v_x})^{K'_{\mathrm{gid},x,v_x}} \cdot K''_{\mathrm{gid},x,v_x}, \\ D''_{\mathrm{gid},x,t} &= (E'_{v_x})^{K'_{\mathrm{gid},x,v_x}}).\end{aligned}$$

If $x \notin S_{\mathrm{gid},t}$, as there does not exist such a v_x (this is guaranteed by the MinCSS algorithm as well), the algorithm outputs $\perp$.

Only the users who possess x at time t can compute a valid decryption key, i.e., a user gid who is entitled x at a later time slot $t' > t$ is unable to compute a valid $\mathsf{DK}_{\mathrm{gid},x,t}$ even he has $\mathsf{SK}_{\mathrm{gid},x}$ (issued at t') and $\mathsf{UK}_{x,t}$ (generated at t). However, in practice, sometimes such a user is allowed to obtain a valid $\mathsf{DK}_{\text{gid},x,t}$ as a special case. Later we will also give a discussion how to efficiently handle such special cases.

2) Ciphertext Decryption

Each user can freely get the ciphertext from the server, but he can decrypt the ciphertext only when the attributes he possesses at time slot t_e satisfy the access policy defined in the ciphertext. Suppose the user gid has sufficient attributes at time slot t_e, it can generate sufficient valid decryption keys $\{\mathsf{DK}_{\text{gid},x,t_e}\}_{x\in S_{\text{gid},t_e}}$ to decrypt the ciphertext by running the following decryption algorithm.

$\mathsf{Decrypt}(\mathsf{CT}, \mathsf{GPP}, \{\mathsf{PK}_d\}, \{\mathsf{DK}_{\text{gid},x,t}\}_{x\in S_{\text{gid},t}}) \rightarrow (M)$ or $\perp$. Assume CT is associated with access policy and time $\langle(A,\rho), t_e\rangle$ and the set of decryption keys $\{\mathsf{DK}_{\text{gid},x,t}\}_{x\in S_{\text{gid},t}}$ is associated with the pair (gid, t). If $t \neq t_e$ or $S_{\text{gid},t}$ does not satisfy (A,ρ), the algorithm outputs $\perp$, otherwise, it decrypts the ciphertext as follows:

a) finds a set $I = \{i|\rho(i) \in S_{\text{gid},t_e}\}$ and computes corresponding constants $\{w_i|i \in I\}$ such that $\sum_{i\in I} w_i A_i = (1, 0, \ldots, 0)$;

b) for each $i \in I$, computes

$$\begin{aligned}
\bar{C}_i &= \frac{C_{i,1}\cdot e(H(gid), C_{i,2})}{e(D_{\text{gid},\rho(i),t_e}, C_{i,3})} \cdot \frac{e(D'_{\text{gid},\rho(i),t_e}, C'_{i,3})}{e(D''_{\text{gid},\rho(i),t_e}, C_{i,4})} \\
&= \frac{e(g,g)^{\lambda_i} e(g,g)^{\alpha_{\rho(i)} r_i} \cdot e(H(gid), g^{\mu_i} g^{\beta_{\rho(i)} r_i})}{e(g^{\alpha_{\rho(i)}} H(gid)^{\beta_{\rho(i)}}, g^{r_i})} \\
&\quad \cdot \frac{e((R_{v_{\rho(i)}})^{\alpha_{\rho(i)}\gamma_{\rho(i)} r_{\text{gid},\rho(i),v_{\rho(i)}}}, g^{r_i/\gamma_{\rho(i)}})}{e((R_{v_{\rho(i)}})^{\alpha_{\rho(i)} r_{\text{gid},\rho(i),v_{\rho(i)}}}, g^{r_i})} \\
&\quad \cdot \frac{e(H_{\phi(\rho(i))}(\rho(i), t)^{\xi_{v_{\rho(i)},t} r_{\text{gid},\rho(i),v_{\rho(i)}}}, g^{r_i/\gamma_{\rho(i)}})}{e(g^{\xi_{v_{\rho(i)},t} r_{\text{gid},\rho(i),v_{\rho(i)}}/\gamma_{\rho(i)}}, H_{\phi(\rho(i))}(\rho(i), t_e)^{r_i})} \\
&\quad \cdot \frac{e((R_{v_{\rho(i)}})^{\gamma_{\rho(i)} r'_{\text{gid},\rho(i),v_{\rho(i)}}}, g^{r_i/\gamma_{\rho(i)}})}{e((R_{v_{\rho(i)}})^{r'_{\text{gid},\rho(i),v_{\rho(i)}}}, g^{r_i})} \\
&= e(g,g)^{\lambda_i} \cdot e(H(gid), g)^{\mu_i}
\end{aligned}$$

c) computes

$$\begin{aligned}
\prod_{i\in I} \bar{C}_i^{w_i} &= e(g,g)^{\sum_{i\in I} w_i\lambda_i} \cdot e(H(gid), g)^{\sum_{i\in I} w_i\mu_i} \\
&= e(g,g)^s
\end{aligned}$$

and recovers the session key κ by $\kappa = C/e(g,g)^s$.

Then, the user can use this session key to further decrypt the encrypted video contents.

V. Attribute Dynamic Updating in TAAC

Let $\mathsf{UL}_{x,t}$ be the update list of attribute x at time slot t. In particular, $\mathsf{UL}_{x,t}$ is a set of global identities, and $gid \in \mathsf{UL}_{x,t}$ means that gid possesses the attribute x at time slot t. The attribute dynamic updating in TAAC contains the following three phases:

Phase 1: Update List Determination

At the beginning of each time slot t, AA_d sets the elements of $\mathsf{UL}_{x,t}$ for any attribute $x \in \mathcal{U}_d$. Attributes can be easily revoked or re-granted from or to users by using the update list. Note that it is not required that $\mathsf{UL}_{x,t} \subseteq \mathsf{UL}_{x,t+1}$, i.e., the system can support access control on *time slot level*. For example, a user gid is revoked the decryption privilege of attribute x only at time slot t (i.e., $gid \in \mathsf{UL}_{x,t}$), but at time slot $t+1$ he needs to be re-granted the decryption privilege of x. Such a case can be easily supported by removing gid from $\mathsf{UL}_{x,t+1}$ (i.e., $gid \notin \mathsf{UL}_{x,t+1}$).

Algorithm 1: $\mathsf{MinCSS}(\mathsf{ST}_x, \mathsf{UL}_{x,t})$

```
1:  $X_e, X_r \leftarrow \emptyset$
2:  $N_{x,t} \leftarrow \emptyset$
3:  $u_e \leftarrow$ the most left empty leaf node
4:  $X_e \leftarrow \mathsf{Path}(u_e)$
5:  for each $u_r \in \mathsf{UL}_{x,t}$ do
6:      add $\mathsf{Path}(u_r)$ to $X_r$
7:  $X_r \leftarrow X_r \setminus X_e$
8:  for each $v_e \in X_e$ do
9:      if $v_e$ is not a leaf node then
10:         $v_{lc} \leftarrow$ left child of $v_e$
11:         if $v_{lc} \notin X_r \cup X_e$ then
12:             add $v_{lc}$ to $N_{x,t}$
13: for each $v_r \in X_r$ do
14:     if $v_r$ is not a leaf node then
15:         $v_{lc} \leftarrow$ left child of $v_r$
16:         if $v_{lc} \notin X_r$ then
17:             add $v_{lc}$ to $N_{x,t}$
18:         $v_{rc} \leftarrow$ right child of $v_r$
19:         if $v_{rc} \notin X_r$ then
20:             add $v_{rc}$ to $N_{x,t}$
21: if $N_{x,t} = \emptyset$ then
22:     add the root node $x$ to $N_{x,t}$
23: Return $N_{x,t}$
```

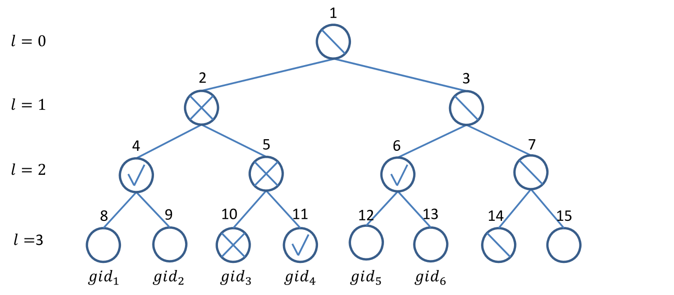
Phase 2: Minimum Cover Set Selection

After determining the update list $\mathsf{UL}_{x,t}$ for each attribute x at time slot t, the corresponding authority $AA_{\phi(x)}$ finds the minimal set of nodes for which it publishes update keys so that only the users who possess x at t can make use of the published update keys to compute corresponding decryption keys. The minimum cover set selection algorithm $\mathsf{MinCSS}(\mathsf{ST}_x, \mathsf{UL}_{x,t})$ is defined in Algorithm 1.

Moreover, different from the algorithm in [20], our algorithm prevents a user who is entitled x at a later time slot $t' > t$ from making use of the update keys for (x, t). Fig. 3 shows an example where $\mathsf{UL}_{x,t} = \{gid_3\}$. The AA only needs to compute the update key $\mathsf{UK}_{x,t,4}$ for $\{gid_1, gid_2\}$, $\mathsf{UK}_{x,t,6}$ for $\{gid_5, gid_6\}$, and $\mathsf{UK}_{x,t,11}$ for gid_4.

Phase 3: Update Key Generation

In this phase, $AA_{\phi(x)}$ generates the $\mathsf{UK}_{x,t}$ according to ST_x and $\mathsf{UL}_{x,t}$ so that only the users who possess x at time slot t are able to obtain valid $\mathsf{UK}_{x,t}$. For any attribute x, the corresponding authority runs $\mathsf{UKeyGen}$ only once at the beginning of each time slot t, taking the current values of ST_x and $\mathsf{UL}_{x,t}$. Note that for any time slot t and attribute x, once $\mathsf{UKeyGen}(t, x, \mathsf{ST}_x, \mathsf{UL}_{x,t}, \mathsf{GPP}, \mathsf{MSK}_d) \rightarrow (\mathsf{UK}_{x,t})$ is run and $\mathsf{UK}_{x,t}$ is published, the later changes of ST_x and $\mathsf{UL}_{x,t}$



$h_x = 3; ctr_x = 6;$
$List_x = \{(gid_1, 8), (gid_2, 9), (gid_3, 10), (gid_4, 11), (gid_5, 12), (gid_6, 13)\}.$
$RL_{x,t} = \{gid_3\} \Rightarrow N_{x,t} = \{4,6,11\}.$

Fig. 3. Example of minimum cover set selection where gid_3 was revoked.

TABLE II
SIZE COMPARISON OF COMPONENTS

Scheme	PK	SK	UK	CT
[16]	n_a	$2n_{u,a} \log n_u$	$n_{a,t} n_u \log n_u$	$2n_u(n_c + 1)$
TAAC	$3n_a$	$3n_{u,a} \log n_u$	$2n_{a,t} \log n_u$	$5n_c + 2$

will not affect $\mathsf{UK}_{x,t}$, and will be reflected at next time slot, i.e., $\mathsf{UK}_{x,t+1}$. Thus, by setting the update list $\mathsf{UL}_{x,t}$ and publishing the corresponding $\mathsf{UK}_{x,t}$, the authority achieves access control of x on time slot level.

VI. Security Analysis and Performance Evaluation

A. *Security Analysis*

Similar to the underlying multi-authority CP-ABE scheme in [15, Appendix D], we will prove TAAC is secure in the generic bilinear group model previously used in [31]–[33], modeling H and $\{H_d\}_{d \in \mathbb{D}}$ as random oracles. Security in this model assures us that an adversary cannot break TAAC with only black-box access to the group operations as well as H and $\{H_d\}_{d \in \mathbb{D}}$.

Theorem 1: In the generic bilinear group model and random oracle model, no polynomial time adversary can break TAAC with non-negligible advantage in the security game of Section III-C.

Proof: The formal proof is given in the Supplementary File. ■

B. *Performance Analysis*

We analyze the performance of TAAC by comparing with Sahai, Seyalioglu and Waters's scheme (SSW's scheme) [16] under the metrics of *Storage Overhead*, *Communication Cost* and *Computation Cost*. Because SSW's scheme does not support multi-authority scenario, the comparison here is in single scenario. We first compare each component in TAAC with SSW's Scheme, as shown in Table II, where n_a denotes the total number of attributes; n_u denotes the total number of users; $n_{u,a}$ denotes number of attributes user u holds; $n_{a,t}$ denotes the number of revoked attributes at time t; and n_c denotes the number of attributes associated with the ciphertext.

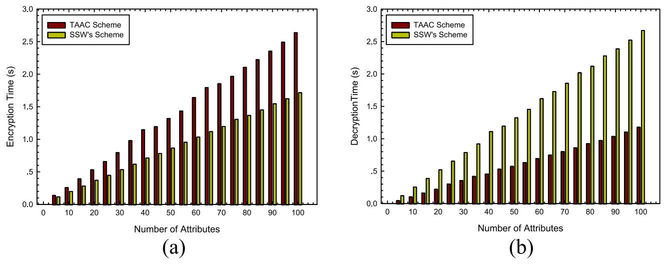


Fig. 4. Comparison of encryption/decryption time. (a) Encryption. (b) Decryption.

TABLE III
ENCRYPTION (DECRYPTION) TIME ON VIDEO CONTENTS [35]

Name	Size	Frame Pattern (I:P:B)	Encryption (Decryption) Time of Video
bus.mpeg	352×240	10:40:98	3.9779 s
space.mpeg	160×128	647:0:0	5.0372 s
twister.mpeg	320×240	206:206:824	38.3258 s

1) Storage Overhead: The storage overhead on the server mainly comes from the ciphertexts CT. Note that here we do not consider the size of encrypted data. The public key PK and the secret key SK also contribute to the overhead on data owners and users respectively. The overhead on each AA comes from both the overhead of MSK and the overhead of user management. From Table II, we can easily find that TAAC incurs less storage overhead on the server, because of its smaller size of ciphertext compared to SSW's scheme.

2) Communication Cost: The communication cost mainly comes from the delivery of update keys. Thus, the communication cost in TAAC is $O(n_{a,t} n_u)$, where $n_{a,t}$ is the number of attributes that need to be updated at time slot t. However, in SSW's scheme, the communication cost is $O(n_{a,t} n_u \log n_u)$, which is much larger than the one of TAAC.

In TAAC, the minimum cover set selection algorithm MinCSS is designed for reducing the communication cost. If no one is revoked in time slot t, the overhead of update keys will be reduced from $O(n_{a,t} n_u)$ to $O(n_{a,t})$ (i.e., only one update key is generated for each attribute). In general, by using the NNL technique [34], the number of nodes in the minimum cover set $|N_{x,t}|$ is approximate to $r \log \frac{n_{u,x}}{r}$, where $n_{u,x}$ is the number of users who possess x and r is the number of revoked users. Thus, the communication overhead caused by update keys can be reduced from $O(n_{a,t} n_u)$ to $O(n_{a,t} r \log \frac{n_u}{r})$.

3) Computation Cost: In TAAC, all the ciphertexts do not need to be updated during each time slot, while in SSW's scheme, the ciphertexts should be updated. We also simulate the computation time of encryption and decryption on a Linux system with an Intel Core 2 Duo CPU at 3.16 GHz and 4.00 GB RAM. The code uses the Pairing-Based Cryptography library version 0.5.14 to simulate the access control schemes. We use a symmetric elliptic curve α-curve, where the base field size is 512-bit and the embedding degree is 2. The α-curve has a 160-bit group order, which means p is a 160-bit length prime. The session key

TABLE IV
ENCRYPTION AND DECRYPTION TIME ON SESSION KEYS

	Number of Involved Attributes in Encryption/Decryption								
Time	1	2	4	6	8	10	14	16	20
Encryption (s)	0.034489	0.059939	0.113293	0.166348	0.218932	0.273655	0.408224	0.448147	0.550210
Decryption (s)	0.012035	0.023120	0.045970	0.068165	0.090815	0.126357	0.160122	0.182653	0.231622

size is set to 128 bits. All the simulation results are the mean of 20 trials.

As shown in Fig. 4(a) and (b), the decryption in TAAC is more efficient than the one in SSW's scheme, while the encryption in SSW's scheme is more efficient than the one in TAAC. In cloud-based multimedia systems, users usually access the data with their mobile devices, such as smart phones, tablets, etc., so TAAC is very suitable for end users to efficiently decrypt multimedia data with their mobile devices.

Due to the large volume of video contents, our proposed TAAC cannot be directly applied to encrypt video contents, especially for live video streaming. Instead, video contents are firstly encrypted by using video encryption methods with session keys. Then, the session keys are encrypted by TAAC for access control purpose. Authorized users who can decrypt session keys can further decrypt video contents. Because TAAC is only applied to control the access of session keys associated with the encrypted video contents, the quality of video contents are not affected by TAAC except the delay caused by encryption and decryption of session keys. To illustrate the encryption/decryption delay caused by TAAC, we first give encryption/decryption[3] time for video contents using video encryption algorithm [35] in Table III.

Then, we simulate the session key encryption and decryption by setting the length of session key to be 128-bit. The delay time for session key encryption and decryption is shown in Table IV, which is closely related to the number of attributes involved in the encryption and decryption. As shown in Tables III and IV, it is not difficult to find that the delay caused by the access control of session keys is not significant compared to the video content encryption/decryption. Moreover, in practical cloud-based multimedia systems, the number of attributes involved in the decryption is usually less than ten.

Besides, by using the minimum cover set selection algorithm MinCSS the computation cost on the update key generation can also be reduced from $O(n_u)$ to $O(r \log \frac{n_u}{r})$. The computation complexity of MinCSS is $O(\log n_{u,x}) + O(n'_{u,x})$, where $n_{u,x}$ is the total number of users who possess x, and $n'_{u,x}$ is the number of users in update list $\mathsf{UL}_{x,t}$. Moreover, the update keys can also be generated in the previous time slot (when the revocation list has been determined in the next time slot) and released at the beginning of the current time slot. As a result, the renew of update keys does not cause significant delay for live video streaming compared with the video decryption and decoding operations. Therefore, we can conclude that TAAC achieves time-domain access control for cloud-based video content sharing without significant delay.

[3]The video encryption algorithm is symmetric, such that the time of encryption and decryption is the same.

VII. DISCUSSION

In the above construction of TAAC, the data access is limited to one time slot. That is, video contents are encrypted under access policies in a time slot, and only users who hold sufficient attributes in that time slot can access them. However, some video contents may be commonly accessed during multiple time slots. In this section, we first discuss on how to modify TAAC to support the commonly accessed video contents during multiple time slots. Then, we further discuss on how to enable users to make special queries on video contents generated in previous time slot.

A. *Commonly Accessed Video Contents During Multiple Time Slots*

In cloud-based video content sharing systems, some video contents may be accessed by any users who hold sufficient attributes in any of these time slots. TAAC can be extended to support those commonly accessed video contents during multiple time slots by modifying the encryption algorithm $\mathsf{Encrypt}$ to a general one which encrypts video contents using $((A, \rho), T_e \subseteq \mathcal{T})$. In particular, $C_{i,4}$ can be modified to

$$C_{i,4} = \{H_{\phi(\rho(i))}(\rho(i), t_e)^{r_i}\}_{t_e \in T_e}.$$

During the time period T_e, anyone who obtain sufficient attributes in any time slot $t \in T_e$ can access the video contents.

B. *Special Queries on Video Contents Generated in Previous Time Slots*

In TAAC, a user gid who does not have sufficient attributes at time slot t cannot decrypt the ciphertext $\mathsf{CT}_{\mathbb{A},t}$, even when the user is entitled sufficient attributes at a later time slot $t'(t' > t)$. However, in some scenarios, this user may be authorized to query the ciphertext $\mathsf{CT}_{\mathbb{A},t}$. To achieve this, the authority will assign a special update key to the queried user. Specifically, for each attribute x required for decryption at time slot t, the authority $AA_{\phi(x)}$ generates a special update key as

$$\begin{aligned}\mathsf{UK}_{x,t,\mathrm{gid}} = \ & (E_{u_{x,\mathrm{gid}}} = R_{u_{x,\mathrm{gid}}} H_{\phi(x)}(x,t)^{\xi_{u_{x,\mathrm{gid}}}}, \\ & E'_{u_{x,\mathrm{gid}}} = g^{\xi_{u_{x,\mathrm{gid}}}})\end{aligned}$$

where $u_{x,\mathrm{gid}}$ is the leaf node assigned to gid and $\xi_{u_{x,\mathrm{gid}}} \in \mathbb{Z}_p$ is randomly chosen. Note that such an update key is useless to other users, and the user gid can only use this update key to obtain the decryption privilege of x at time slot t.

VIII. Conclusion

In this paper, we have proposed a cryptographic approach, TAAC, to achieve time-domain attribute-based access control for cloud-based video content sharing. Specifically, we have proposed a provably secure time-domain ABE scheme by embedding the time into both the ciphertexts and the keys, such that only users who hold sufficient attributes in a specific time period can decrypt the data. To achieve the dynamic change of users' attributes, we have also proposed an efficient attribute updating method which enables attribute authorities to grant new attributes, revoke previous attributes and re-grant previously revoked attributes to users at the beginning of each time slot. We have further discussed on how to achieve access control of video contents that are commonly accessed in multiple time slots and how to make special queries on video contents generated in previous time slots. We have provided the security proof for the proposed TAAC scheme in generic bilinear group model and random oracle model.

In our future work, we will implement TAAC on real cloud-based multimedia systems and explore the time-domain access control scheme in standard model.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST SP 800-145, 2009.

[2] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 59–69, May 2011.

[3] D. Díaz-Sánchez, F. Almenarez, A. Marín, D. Proserpio, and P. A. Cabarcos, "Media cloud: An open cloud computing middleware for content management," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 970–978, May 2011.

[4] X. Wang, M. Chen, T. T. Kwon, L. Yang, and V. Leung, "AMES-Cloud: A framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 811–820, Jun. 2013.

[5] M. Hefeeda, T. ElGamal, K. Calagari, and A. Abdelsadek, "Cloud-based multimedia content protection system," *IEEE Trans. Multimedia*, vol. 17, no. 3, pp. 420–433, Mar. 2015.

[6] H. Shen, L. Zhuo, and Y. Zhao, "An efficient motion reference structure based selective encryption algorithm for H. 264 videos," *IET Inform. Security*, vol. 8, no. 3, pp. 199–206, 2014.

[7] Z. Shahid and W. Puech, "Visual protection of hevc video by selective encryption of cabac binstrings," *IEEE Trans. Multimedia*, vol. 16, no. 1, pp. 24–36, Jan. 2014.

[8] B. Zeng, S.-K. A. Yeung, S. Zhu, and M. Gabbouj, "Perceptual encryption of h.264 videos: Embedding sign-flips into the integer-based transforms," *IEEE Trans. Inform. Forensics Security*, vol. 9, no. 2, pp. 309–320, Feb. 2014.

[9] T. Stütz and A. Uhl, "A survey of H.264 AVC/SVC encryption," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 3, pp. 325–339, Mar. 2012.

[10] *RSA Cryptography Standard*, Public Key Cryptography Standard 1 ver. 2.2, 2012.

[11] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*. New York, NY, USA: Springer-Verlag, 1985, pp. 10–18.

[12] G. B. Algin and E. T. Tunali, "Scalable video encryption of h. 264 svc codec," *J. Vis. Commun. Image Represent.*, vol. 22, no. 4, pp. 353–364, 2011.

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

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

[15] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. EUROCRYPT*, 2011, pp. 568–588.

[16] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Proc. CRYPTO*, 2012, pp. 199–217.

[17] K. Yang, X. Jia, and K. Ren, "Attribute-based fine-grained access control with efficient revocation in cloud storage systems," in *Proc. 8th ACM SIGSAC Symp. Inform., Comput. Commun. Security*, 2013, pp. 523–528.

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

[19] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2014.

[20] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. 15th ACM Conf. Comput. Commun. Security*, 2008, pp. 417–426.

[21] K. Zhang, X. Liang, X. Shen, and R. Lu, "Exploiting multimedia services in mobile social networks from security and privacy perspectives," *IEEE Commun. Mag.*, vol. 52, no. 3, pp. 58–65, Mar. 2014.

[22] Y. Wu, Z. Wei, and R. Deng, "Attribute-based access to scalable media in cloud-assisted content sharing networks," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 778–788, Jun. 2013.

[23] Y. Zhang, S. He, and J. Chen, "Data gathering optimization by dynamic sensing and routing in rechargeable sensor networks," *IEEE/ACM Trans. Netw.*, to be published.

[24] H. Zhang, P. Cheng, L. Shi, and J. Chen, "Optimal dos attack scheduling in wireless networked control system," *IEEE Trans. Control Syst. Technol.*, to be published.

[25] M. Chase, "Multi-authority attribute based encryption," in *Proc. 4th Conf. Theory Cryptography*, 2007, pp. 515–534.

[26] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM Conf. Comput. Commun. Security*, 2009, pp. 121–130.

[27] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," in *Proc. 6th ACM Symp. Inform., Comput. Commun. Security*, 2011, pp. 386–390.

[28] K. Yang and X. Jia, "Attributed-based access control for multi-authority systems in cloud storage," in *Proc. 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 536–545.

[29] 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.

[30] K. Yang, K. Zhang, J. Ren, and X. S. Shen, "Security and privacy in mobile crowdsourcing networks," *IEEE Commun. Mag.*, vol. 53, no. 8, pp. 75–81, Aug. 2015.

[31] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Proc. EUROCRYPT*, 1997, pp. 256–266.

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

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

[34] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Proc. CRYPTO*, 2001, pp. 41–62.

[35] L. Qiao and K. Nahrstedt, "A new algorithm for mpeg video encryption," in *Proc. 1st Int. Conf. Imag. Sci., Syst. Technol.*, 1997, pp. 21–29.

[36] J. Ren, K. Zhang, and X. Shen, "Exploiting mobile crowdsourcing for pervasive cloud services: Challenges and solutions," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 98–105, Aug. 2015.

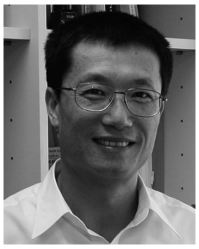
[37] J. Ren, K. Zhang, and X. Shen, "Social aware crowdsourcing with reputation management in mobile sensing," *Comput. Commun.*, vol. 65, no. 1, pp. 55–65, Aug. 2015.

Kan Yang received the B.Eng. degree in information security from the University of Science and Technology of China, Hefei, China, in 2008, and the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, China, in August 2013.

He is currently a Postdoctoral Fellow of the Broadband Communications Research Group with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include cloud security and privacy, big data security and privacy, mobile security and privacy, big data mining, applied cryptography, wireless communication and networks, and distributed systems.

Zhen Liu received the B.S. degree in applied mathematics and the M.S. degree in mathematics from Shanghai Jiao Tong University, Shanghai, China, and the Ph.D. degrees in computer science from the City University of Hong Kong, Hong Kong, China, and Shanghai Jiao Tong University, Shanghai, China.

His research interests include applied cryptography, in particular encryption and signature schemes, provable security and designing cryptographic primitives, lattice-based cryptography (mainly on lattice-based functional encryption), and fully homomorphic encryption.



Xiaohua Jia (A'00–SM'01–F'13) received the B.Sc. and M.Eng. degrees from the University of Science and Technology of China, Hefei, China, in 1984 and 1987, respectively, and the D.Sc. degree in information science from the University of Tokyo, Tokyo, Japan, in 1991.

He is currently the Chair Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong, China. His research interests include cloud computing and distributed systems, computer networks, wireless sensor networks, and mobile wireless networks.

Prof. Jia has been an Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (2006–2009), *Wireless Networks*, the *Journal of World Wide Web*, and the *Journal of Combinatorial Optimization*. He is the General Chair of ACM MobiHoc 2008, the TPC Co-Chair of IEEE MASS 2009, the Area Chair of the IEEE INFOCOM 2010, the TPC Co-Chair of the IEEE GlobeCom 2010, Ad Hoc, and Sensor Networking Symp, and the Panel Co-Chair of the IEEE INFOCOM 2011.

Xuemin Sherman Shen (M'97–SM'02–F'09) is currently a Professor and the University Research Chair with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks.

Prof. Shen is a Registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology and Communications Societies. He served as the Technical Program Committee Chair/Co-Chair for the IEEE GLOBECOM'16, IEEE INFOCOM'14, IEEE VTC'10 Fall, Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring and IEEE ICC'08, and the Technical Program Committee Chair for the IEEE GLOBECOM'07. He also serves/served as the Editor-in-Chief for the IEEE NETWORK, *Peer-to-Peer Networking and Application*, and *IET Communications*.