2014

k-times attribute-based anonymous access control for cloud computing

Joseph K. Liu
University of Bristol

Tsz Hon Yuen
University of Hong Kong, thy738@uow.edu.au

Man Ho Au
University of Wollongong, aau@uow.edu.au

Xinyi Huang
Fujian Normal University, University of Wollongong, xh068@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

See next page for additional authors

Publication Details

Liu, J. K., Yuen, T., Au, M. Ho., Huang, X., Susilo, W. & Zhou, J. (2015). k-times attribute-based anonymous access control for cloud computing. IEEE Transactions on Computers, 64 (9), 2595-2608.

k-times attribute-based anonymous access control for cloud computing

Abstract
In this paper, we propose a new notion called k-times attribute-based anonymous access control, which is particularly designed for supporting cloud computing environment. In this new notion, a user can authenticate himself/herself to the cloud computing server anonymously. The server only knows the user acquires some required attributes, yet it does not know the identity of this user. In addition, we provide a k-times limit for anonymous access control. That is, the server may limit a particular set of users (i.e., those users with the same set of attribute) to access the system for a maximum k-times within a period or an event. Further additional access will be denied. We also prove the security of our instantiation. Our implementation result shows that our scheme is practical.

Keywords
times, attribute, k, anonymous, computing, access, control, cloud

Disciplines
Engineering | Science and Technology Studies

Publication Details
Liu, J. K., Yuen, T., Au, M. Ho., Huang, X., Susilo, W. & Zhou, J. (2015). k-times attribute-based anonymous access control for cloud computing. IEEE Transactions on Computers, 64 (9), 2595-2608.

Authors
Joseph K. Liu, Tsz Hon Yuen, Man Ho Au, Xinyi Huang, Willy Susilo, and Jianying Zhou

k-times Attribute-Based Anonymous Access Control for Cloud Computing

Tsz Hon Yuen, Joseph K. Liu*, Man Ho Au, Xinyi Huang, Willy Susilo, Jianying Zhou

***Abstract*—In this paper, we propose a new notion called *k-times attribute-based anonymous access control*, which is particularly designed for supporting cloud computing environment. In this new notion, a user can authenticate himself/herself to the cloud computing server anonymously. The server only knows the user acquires some required attributes, yet it does not know the identity of this user. In addition, we provide a k-times limit for anonymous access control. That is, the server may limit a particular set of users (i.e., those users with the same set of attribute) to access the system for a maximum k-times within a period or an event. Further additional access will be denied. We also prove the security of our instantiation. Our implementation result shows that our scheme is practical.**

Keywords: attribute-based, anonymous access, k-times, cloud computing

I. Introduction

Cloud computing offers various types of computing services to end users via computer networks and it is being widely adopted due to its many advantages. Cloud computing provides an extensible and powerful environment for growing amounts of services and data by means of on-demand self-service. It also relieves the client's burden from management and maintenance by providing a comparably low-cost, scalable, location-independent platform. The benefits of cloud computing include reduced costs and capital expenditures, increased operational efficiencies, scalability, flexibility and immediate time to market. Different service-oriented cloud computing models have been proposed, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [33]. Numerous commercial cloud computing systems have been built at different levels, e.g., EC2 [1] and S3 [2] from Amazon are IaaS systems, while Google App Engine [3] and Yahoo Pig [4] are representatives of PaaS systems, and Googles Apps [8] and Salesforces Customer Relation Management (CRM) System [5] belong to SaaS systems. With these cloud computing systems, on one hand, enterprise users are no longer required to invest in hardware/software systems nor hire IT professionals to maintain these IT systems, thus saving cost on IT infrastructure and human resources; on the other hand, computing utilities provided by cloud computing are being offered at a relatively low price in a pay-as-you-use style.

Joseph K. Liu is the corresponding author. Joseph K. Liu and Jianying Zhou are with Institute for Infocomm Research, Singapore; Tsz Hon Yuen is with Huawei, Singapore; Man Ho Au is with Department of Computing, Hong Kong Polytechnic University, Hong Kong; Willy Susilo is with University of Wollongong, Australia; Xinyi Huang is with School of Mathematics and Computer Science, Fujian Normal University, China.

Although the new paradigm of cloud computing provides great advantages, there are also concerns about security and privacy due to its Internet-based management. In the pay-as-you-use style, the cloud service providers may charge each enterprise or individual user for using the service. The traditional solution is to setup an account for each user. A user is required to login for using the cloud services. The service provider then charges the user based on his/her usage. This solution works perfectly, if privacy is not a concern. Nevertheless, it is well acknowledged that privacy is an essential feature that must be considered in several practical applications.

When we consider user privacy at the same time, account-based access control does not work since it is not anonymous. Recently proposed access control models, such as attribute-based access control, can provide anonymous authentication while it can further define access control policies based on different attributes of the requester, environment, or the data object. The concept of attribute-based encryption (ABE) [18], [27] is a promising approach that fulfills these requirements. ABE features a mechanism that enables an access control over encrypted data using access policies and ascribed attributes among private keys and ciphertexts. Especially, ciphertext-policy ABE (CP-ABE) [11] provides a scalable way of encrypting data such that the encryptor defines the attribute set that the decryptor needs to possess in order to decrypt the ciphertext. Thus, different users are allowed to decrypt different pieces of data with respect to the security policy. This effectively eliminates the need to rely on the storage server for preventing unauthorized data access.

Nevertheless, ABE only deals with authenticated access on encrypted data in cloud storage service. It is impractical to be deployed in the case of access control to cloud computing service: The cloud server may encrypt a random message using the access policy and asks the user to decrypt it. If the user can successfully decrypt the ciphertext, it is allowed to access the cloud computing service. Although this approach can fulfill the requirement, it is highly inefficient.

In addition to ABE, another similar cryptographic primitive is attribute-based signature (ABS) [22], [29], [26]. An ABS enables a party to sign a message with fine-grained access control over identifying information. Specifically, in an ABS system, users obtain their attribute private keys from an attribute authority, with which they can later sign messages for any predicate satisfied by their attributes. A verifier will be convinced of the fact that whether the signer's attributes satisfies the signing predicate while remaining completely ignorant of the identity of signer. Thus it can achieve anonymous attribute-based access control efficiently.

A. k-times Access Control

ABS is anonymous and unlinkable. It can be used to provide unlimited times anonymous authentication. However, in the cloud computing environment, unlimited times access control is sometimes *undesirable*. Let us consider the following scenario. Netflix hosts its service in the cloud by enabling its user to access the movies online. For trial users, each user is only permitted to access 2 movies in a month. For regular users, up to 30 movies a month is permitted and for VIP users, there will be unlimited access. For this particular scenario, it may be feasible to develop this system using a traditional username/password control. However, when user privacy is required, then the problem becomes more daunting. In this particular scenario, user's privacy is important as the user wants to be ensured that the movies that he/she is viewing will not be tracked by Netflix.

We shall note that a regular ABS is capable to provide unlinkable and anonymous attribute access control. Yet, given any access transcript, no one can find whether it is the user's first time access, second time and so forth. Thus, unfortunately it cannot provide *limited times* anonymous access control.

B. Related Works

In the area of k-times anonymous access control, some related research have been conducted in the literature. The notion of k-times anonymous authentication (k-TAA) (such as [30], [25], [31], [7], [14], [24], [32]) allows a user to authenticate himself/herself to a verifier anonymously. The verifier further knows that whether the user has been authenticated less than k-times. Different from an attribute-based access system, the authorized group of k-TAA has to be fixed a priori, which makes it less flexible in practice. In contrast, an attribute-based access system allows dynamicity in the control. Another primitive that allows spontaneous anonymous access control is the notion of ring signature [28], [15]. A ring signature allows a user to sign on behalf of the whole group by including their public keys or identities. The verifier only knows the fact that the signer is one of the users within the group, but the actual identity of the signer remains hidden. While normal ring signature provides unlinkability, linkable ring signature [20], [21], [6] allows the verifier to know whether the user has signed previously (i.e., to link with previous signatures). Intuitively, it is clear that linkable ring signature may be used as k-times anonymous access control. Nevertheless, it requires the signer to know the public keys or identities of *all* users within the group prior to generating the signature, which is impractical for many scenarios.

C. Our Contributions

In this paper, we propose a new notion called k-times attribute-based anonymous access control system for cloud computing. A normal anonymous attribute-based authentication system (e.g. ABS) does not allow the verifier to know it is the i-th time the prover has authenticated to the system. In contrast to this, our scheme provides a mechanism to detect whether the user has exceeded k-times for accessing the system using some defined claim-predicate (within a period or a single event). At the same time the user is anonymous to the system at any time. In Table II, we compare our new notion against k-TAA, unlinkable ring signatures (URS), linkable ring signatures (LRS) and attribute-based signatures. We also define the security model for this new notion. We also provide a concrete instantiation for our system and prove the security of our instantiation. Finally we implement our scheme to show that it is practical.

Primitive	Dynamicity	Anonymity	Unlinkability	Limit
k-TAA	×	✓	✓	✓
URS	×	✓	✓	×
LRS	×	✓	×	×
ABS	✓	✓	✓	×
Ours	✓	✓	✓or ×[a]	✓

[a] Our scheme provides an option for service provider to make it linkable or unlinkable. Yet even if it is unlinkable, the service provider knows whether the user has exceeded the k-times access limit.

Table 1. Comparison

II. Backgrounds

A. Pairings and Mathematical Assumption

Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of prime order p. A map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is bilinear if for any generators $g \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$. Let $\mathcal{G}$ be a pairing generation algorithm which takes as input a security parameter 1^λ and outputs $(p, \mathbb{G}, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$. The generators of the groups may also be given. All group operations as well as the bilinear map $\hat{e}$ are efficiently computable.

For our scheme, we assume the following problem is difficult to solve in the setting described above.

Definition 1: (q-Decisional Bilinear Diffie-Hellman Inversion (DBDHI)[12]) Given the tuple $(g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q}, T)$, where $g \in_R \mathbb{G}$, and $\alpha \in_R \mathbb{Z}_p$, decide if $T = \hat{e}(g, g)^{\frac{1}{\alpha}} \in \mathbb{G}_T$.

B. Monotone Span Program

We review some notation about monotone span program using the notation in [22]. Let $\Upsilon : \{0,1\}^n \to \{0,1\}$ be a monotone boolean function. A monotone span program for Υ over a field $\mathbb{F}$ is an $\ell \times m$ matrix $\mathbf{M}$ with entries in $\mathbb{F}$, along with a labeling function $\rho : [1, \ell] \to [1, n]$ that associates each row of $\mathbf{M}$ with an input variable of Υ, that, for every $(x_1, \ldots, x_n) \in \{0,1\}^n$, satisfies the following:

$$\Upsilon(x_1, \ldots, x_n) = 1 \Longleftrightarrow \exists \vec{v} \in \mathbb{F}^{1\times\ell} : \vec{v}\mathbf{M} = [1, 0, 0, \ldots, 0] \text{ and } (\forall i : x_{\rho(i)} = 0 \Rightarrow v_i = 0).$$

In other words, $\Upsilon(x_1, \ldots, x_n) = 1$ if and only if the rows of $\mathbf{M}$ indexed by $\{i | x_{\rho(i)} = 1\}$ span the vector $[1, 0, 0, \ldots, 0]$. We call ℓ the length and m the width of the span program, and $\ell + m$ the size of the span program. Every monotone boolean

function can be represented by some monotone span program, and a large class do have compact monotone span programs.

Below we give an example illustrating how a simple monotone boolean function $\Upsilon : (x_1, x_2, x_3) \mapsto x_1 \vee (x_2 \wedge x_3)$ can be represented by a span program.

One possible realization of matrix $\mathbf{M}$ is: $\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$ together with the labeling function $\rho : i \in [1,3] \mapsto i$. It is easy to verify that for any (x_1, x_2, x_3) such that $\Upsilon(x_1, x_2, x_3) = 1$, we can find a vector $\vec{v}$ such that $\vec{v} \cdot \mathbf{M} = [1, 0]$ and that $v_i = 0$ if $x_{\rho(i)} = 0$.

For example, $(x_1, x_2, x_3) := (0, 1, 1)$ satisfies Υ and we can find a corresponding vector $\vec{v}$ as $[0, 1, -1]$. Note that this vector satisfies the condition that $v_0 = 0$ (since $x_{\rho(0)} = 0$). We would like to remark that the vector is not necessarily unique. For instance, if $(x_1, x_2, x_3) := (1, 1, 1)$, the vector $\vec{v}$ could be $[1, 0, 0]$ or $[3, 2, -2]$.

On the other hand, no valid vector exists for (x_1, x_2, x_3) if $\Upsilon(x_1, x_2, x_3) = 0$.

III. Security Requirement

We give our security models of anonymous attributed-based access control scheme and define relevant security notions.

A. Syntax of Attribute-Based Access Control

Let $\mathbb{A}$ be the universe of possible attributes. A *claim-predicate* over $\mathbb{A}$ is a monotone boolean function, whose inputs are associated with attributes of $\mathbb{A}$. We say that an attribute set $\mathcal{A} \subseteq \mathbb{A}$ satisfies a claim-predicate Υ if $\Upsilon(\mathcal{A}) = 1$.

Definition 2: An attribute-based access control (ABAC) scheme is a tuple of five algorithms parameterized by a universe of possible attributes $\mathbb{A}$:

- TSetup (This is a global setup algorithm to be run by a trustee for the generation of public reference information.): On input the security parameter 1^λ, generates public reference information TPK.
- ASetup (This is a setup algorithm to be run by an attribute-issuing authority for the generation of the secret key and public key of the authority.): On input the security parameter 1^λ, generates a key pair $(\mathsf{APK}, \mathsf{ASK})$.
- USetup (This is a setup algorithm to be run by a user to generate the user secret key and public key.): On input the security parameter 1^λ, generates a key pair $(\mathsf{UPK}, \mathsf{USK})$.
- $\mathsf{AttrGen}$ (This is a key generation algorithm to generate the user attribute secret key. This is an interactive protocol between the user and the authority.): It is a two party protocol $\mathsf{AttrGen}_U$ and $\mathsf{AttrGen}_A$ run by the user and the attribute-issuing authority respectively. The common inputs are $(\mathcal{PK} = (\mathsf{TPK}, \mathsf{APK}), \mathcal{A} \subseteq \mathbb{A}, \mathsf{UPK})$. The secret input to $\mathsf{AttrGen}_U$ is USK and to $\mathsf{AttrGen}_A$ is ASK. $\mathsf{AttrGen}_A$ outputs a secret key $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$[1].
- Auth: (This is the authentication protocol between the user and the server.) It is a two party protocol Auth_P and Auth_V run by the user and the service provider respectively. The common inputs are $(\mathsf{PK} = (\mathsf{TPK}, \mathsf{APK}), \Upsilon, M, \mathsf{UPK})$, where Υ is the claim-predicate and M is the maximum number of authentication. The secret input to Auth_P is $(\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}, \mathsf{USK}, k)$, where k is the number of past authentication with the service provider. At the end, Auth_V outputs accept or reject.

[1]It is used to identify the case that different users may have identical attributes in the system.

Correctness. It must satisfy that authentication according to specification are accepted if the key used is correctly generated.

B. Notions of Security of Attributed-Based Access Control

We now describe the security properties of attributed-based access control. Informally speaking, the attributed-based access control scheme should have the basic property of *mis-authentication resistance*, which means that no collusion of unregistered users and users with attributes not satisfying the predicate can authenticate with an honest service provider. A valid user cannot authenticate for more than M times.

For the *anonymity* property of attributed-based access control, it means that the service provider and the attribute-issuing authority cannot identify the authenticating user.

We now give the formal definition for the above properties, which are captured in the security model of authentication, privacy and linkability.

1) AUTHENTICATION.
The security definition of authentication is divided into two parts. The first part guarantees no user with attributes not satisfying the predicate can authenticate. No collusion of users (that is, the merge of attributes within different users) is allowed. This is also the traditional definition of ABS. The second part guarantees a valid user (with respect to the part 1 definition) cannot authenticate for more than M times, where M is specified in the predicate.

Part 1: Authentication for a claim-predicate Υ^* in attributed-based access control schemes should prevent the Adversary $\mathfrak{A}$ to authenticate, where:

- $\mathfrak{A}$ has some secret keys $\mathsf{sk}_{\mathcal{A}^*,\mathsf{UPK}}$ for some UPK and $\Upsilon^*(\mathcal{A}^*) = 1$, but $\mathfrak{A}$ does not know the USK corresponding to UPK;
- $\mathfrak{A}$ has some secret keys $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}'}$ for some UPK' and has the corresponding USK', but $\Upsilon^*(\mathcal{A}) \neq 1$ for all $\mathcal{A}$.

Our model allows collusion of these two types of users. Authentication is defined in the following game between the Challenger $\mathfrak{C}$ and the Adversary $\mathfrak{A}$ in which $\mathfrak{A}$ is given access to some oracles.

a) $\mathfrak{C}$ generates $\mathsf{TPK} \leftarrow \mathsf{TSetup}(1^\lambda)$ and $(\mathsf{APK}, \mathsf{ASK}) \leftarrow \mathsf{ASetup}(1^\lambda)$. $\mathfrak{C}$ gives $\mathfrak{A}$ the public information $\mathsf{PK} = (\mathsf{TPK}, \mathsf{APK})$.

b) $\mathfrak{A}$ may query the following oracles according to any adaptive strategy.

- $\mathsf{UPK} \leftarrow \mathcal{JO}()$. The $\mathsf{Join\ Oracle}$ runs $(\mathsf{UPK}, \mathsf{USK}) \leftarrow \mathsf{USetup}(1^\lambda)$. It stores $(\mathsf{UPK}, \mathsf{USK})$ in a list $\mathcal{L}$ which is initially empty. It returns UPK.

- $\mathsf{USK} \leftarrow \mathcal{CO}(\mathsf{UPK})$. The Corrupt Oracle, on input UPK, searches $(\mathsf{UPK}, \mathsf{USK})$ in the list $\mathcal{L}$ and returns USK.
- $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}} \leftarrow \mathcal{GO}(\mathcal{A}, \mathsf{UPK})$. The AttrGen Oracle, on input an attribute set $\mathcal{A}$ and a UPK, it runs the $\mathsf{AttrGen}_A(\mathsf{PK}, \mathcal{A}, \mathsf{UPK}, \mathsf{ASK})$ with $\mathfrak{A}$ (who runs $\mathsf{AttrGen}_U$). $\mathfrak{A}$ obtains the corresponding secret key $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$. We require for the same $(\mathcal{A}, \mathsf{UPK})$ as input, the same $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$ is the output.
- $\sigma \leftarrow \mathcal{AO}(\mathcal{A}, \Upsilon, \mathsf{UPK}, k, M)$. The Auth Oracle, on input an attribute set $\mathcal{A}$, a UPK in $\mathcal{L}$, a past authentication number k, a maximum M and a claim-predicate Υ such that $\Upsilon(\mathcal{A}) = 1$, $k < M$, it searches USK where $(\mathsf{UPK}, \mathsf{USK}) \in \mathcal{L}$ and runs $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}} \leftarrow \mathsf{AttrGen}(\mathsf{ASK}, \mathcal{A}, \mathsf{UPK})$. It runs a valid Auth_P protocol with $\mathfrak{A}$ (who runs Auth_V), and accept $\leftarrow \mathsf{Auth}_P(\mathsf{PK}, \Upsilon, M, \mathsf{UPK}, \mathsf{sk}_{\mathcal{A},\mathsf{UPK}}, \mathsf{USK}, k)$.

c) $\mathfrak{A}$ gives $\mathfrak{C}$ a claim-predicate Υ^*, a UPK^*, a number M^* and runs Auth_P protocol with $\mathfrak{C}$.

$\mathfrak{A}$ wins the game if:

(1) $\mathsf{Auth}_V(\mathsf{PK}, \Upsilon^*, M^*, \mathsf{UPK}^*)$=accept for the final phase;

(2) $\Upsilon^*(\mathcal{A}) = 0$ for all $(\mathcal{A}, \mathsf{UPK}^*)$ queried to $\mathcal{GO}$ if UPK^* was asked to $\mathcal{CO}$ or UPK^* was not outputted by $\mathcal{JO}$.

We denote by

$$\mathbf{Adv}_{\mathfrak{A}}^{auth-1} = \Pr[\ \mathfrak{A} \text{ wins the game }].$$

Part 2: If the user is allowed to authenticate for at most M times, then he/she cannot run the Auth protocol for $M+1$ times without being detected. It implies that the service provider can detect such mis-usage and reject this authentication.

This part of authentication is defined in the following game between the Challenger $\mathfrak{C}$ and the Adversary $\mathfrak{A}$.

a) $\mathfrak{C}$ generates $\mathsf{TPK} \leftarrow \mathsf{TSetup}(1^\lambda)$ and $(\mathsf{APK}, \mathsf{ASK}) \leftarrow \mathsf{ASetup}(1^\lambda)$. $\mathfrak{C}$ gives $\mathfrak{A}$ the public information $\mathsf{PK} = (\mathsf{TPK}, \mathsf{APK})$. $\mathfrak{A}$ is given all oracles described in part 1.

b) $\mathfrak{A}$ gives $\mathfrak{C}$ a claim-predicate Υ^*, a UPK^*, a number M^* and runs Auth_P protocol with $\mathfrak{C}$ for M^*+1 times.

$\mathcal{B}$ wins the game if: $\mathsf{Auth}_V(\mathsf{PK}, \Upsilon^*, M^*, \mathsf{UPK}^*)$ =accept for all M^*+1 times and UPK^* is an output from $\mathcal{JO}()$.

We denote by

$$\mathbf{Adv}_{\mathfrak{A}}^{auth-2} = \Pr[\ \mathfrak{A} \text{ wins the game }].$$

Definition 3 (Authentication): An Attribute-Based Access Control scheme is authenticate if for all PPT adversary $\mathfrak{A}$, both $\mathbf{Adv}_{\mathfrak{A}}^{auth-1}$ and $\mathbf{Adv}_{\mathfrak{A}}^{auth-2}$ are negligible.

2) PRIVACY.

In order to protect privacy, ABAC must hide the attributes used during authentication against the attribute-generating authority. If an honest user only authenticates within a permitted number of times, then all his past authentication must be anonymous and unlinkable. We model two cases:

- If an honest user only authenticates within a permitted number of times, his/her authentication should not be linked with his/her past authentications.
- Even if two users have the same attributes, no one, including the service provider and the authority, can find out who is responsible for the authentication.

This is defined in the following game between the Challenger $\mathfrak{C}$ and the Adversary $\mathfrak{A}$ in which $\mathfrak{A}$ is given the ASK.

a) $\mathfrak{C}$ generates $\mathsf{TPK} \leftarrow \mathsf{TSetup}(1^\lambda)$ and $(\mathsf{APK}, \mathsf{ASK}) \leftarrow \mathsf{TSetup}(1^\lambda)$. $\mathfrak{C}$ gives $\mathfrak{A}$ the public information $\mathsf{PK} = (\mathsf{TPK}, \mathsf{APK})$ and also ASK.

b) $\mathfrak{A}$ may query the oracles according to any adaptive strategy, including $\mathcal{JO}$ and $\mathcal{CO}$. For $\mathcal{JO}$, $\mathfrak{C}$ stores $(\mathsf{UPK}, \mathsf{USK}, 0)$ in the list $\mathcal{L}$, where the last number denotes the number of past authentication is zero. $\mathfrak{A}$ may also query the following oracle:

- $\sigma \leftarrow \mathcal{AO}(\mathcal{A}, \Upsilon, \mathsf{UPK}, M)$. The Auth Oracle, on input an attribute set $\mathcal{A}$, a UPK in $\mathcal{L}$, a maximum M and a claim-predicate Υ such that $\Upsilon(\mathcal{A}) = 1$, it searches USK where $(\mathsf{UPK}, \mathsf{USK}, k) \in \mathcal{L}$. It $k > M$, it halts and return $\perp$. Otherwise, it runs $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}} \leftarrow \mathsf{AttrGen}(\mathsf{PK}, \mathcal{A}, \mathsf{UPK}, \mathsf{USK}, \mathsf{ASK})$ with $\mathfrak{A}$ acts as the authority[2]. It runs a valid Auth_P protocol with $\mathfrak{A}$ (who runs Auth_V), and accept $\leftarrow \mathsf{Auth}_P(\mathsf{PK}, \Upsilon, M, \mathsf{UPK}, \mathsf{sk}_{\mathcal{A},\mathsf{UPK}}, \mathsf{USK}, k)$.

c) $\mathfrak{A}$ chooses two users $\mathsf{UPK}_0^*, \mathsf{UPK}_1^*$, two attribute sets $\mathcal{A}_0^*$, $\mathcal{A}_1^*$, a number M^* and a claim-predicate Υ^* such that $\Upsilon^*(\mathcal{A}_0^*) = \Upsilon(\mathcal{A}_1^*) = 1$, $(\mathsf{UPK}_0^*, \mathsf{USK}_0^*, k^*), (\mathsf{UPK}_1^*, \mathsf{USK}_1^*, k^*) \in \mathcal{L}$[3] and $k^* < M^*$. It sends $(\mathsf{UPK}_0^*, \mathsf{UPK}_1^*, \mathcal{A}_0^*, \mathcal{A}_1^*, M^*, \Upsilon^*)$ to $\mathfrak{C}$.

$\mathfrak{C}$ chooses a random bit $b \in \{0,1\}$ and generates $\mathsf{sk}_{\mathcal{A}_b^*,\mathsf{UPK}_b^*} \leftarrow \mathsf{AttrGen}(\mathsf{PK}, \mathcal{A}_b^*, \mathsf{UPK}_b^*, \mathsf{USK}_b^*, \mathsf{ASK})$. It runs $\mathsf{Auth}_P(\mathsf{PK}, \Upsilon^*, M^*, \mathsf{UPK}_b^*, \mathsf{sk}_{\mathcal{A}_b^*,\mathsf{UPK}_b^*}, \mathsf{USK}_b^*, k^*)$ with $\mathfrak{A}$. $\mathfrak{C}$ also updates $(\mathsf{UPK}_{1-b}^*, \mathsf{USK}_{1-b}^*, k^*+1) \in \mathcal{L}$ for consistency of the two entries in $\mathcal{L}$.

d) $\mathfrak{A}$ continues to query oracles with arbitrary interleaving.

e) $\mathfrak{A}$ outputs a bit b'.

$\mathfrak{A}$ wins the game if $b' = b$. We denote by

$$\mathbf{Adv}_{\mathfrak{A}}^{anon} = \Big| \Pr[\ \mathfrak{A} \text{ wins the game }] - \frac{1}{2} \Big|.$$

[2]This step can be omitted if the same key was generated in previous query.

[3]It means that $\mathsf{UPK}_0^*, \mathsf{UPK}_1^*$ are from the output of $\mathcal{JO}$ and both UPK_0^* and UPK_1^* are asked to $\mathcal{AO}$ for k^* times. Note that UPK_0^* may be the same as UPK_1^*, in order to capture unlinkability for different authentications from the same user. For different UPK_0 and UPK_1, they may have the same attributes, in order to capture the anonymity property.

Definition 4 (Privacy): An Attribute-Based Access Control scheme is private if for all PPT adversary $\mathfrak{A}$, $\mathbf{Adv}_{\mathfrak{A}}^{anon}$ is negligible.

IV. Our Proposed Scheme

In this section, we first describe our basic construction which does not contain any event. Later we will explain how to extend the basic construction into event-oriented in Section V. That is, a user is allowed to access the system for a maximum k times for a particular event. There is a counter for each event.

A. Notation

We first give the notations used in our scheme in Table I.

TABLE I: Frequently Used Notations

$\mathbb{A}$	universe of attributes
TPK	public key of the trustee
APK	public key of the attribute-issuing authority
ASK	secret key of the attribute-issuing authority
UPK	user public key
USK	user secret key
$\mathcal{A}$	an attribute set of a user
$\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$	attribute secret key for a user with a set of attributes $\mathcal{A}$ and user public key UPK
k	the number of past authentication
M	the maximum number of authentication allowed
Υ	the claim-predicate used for authentication
$\mathbf{M}$	the monotone span program matrix (with size $\ell \times m$) converted from Υ

B. Intuition

We extend the key generating system for group signatures [13] into the attribute-based setting. The original key structure in [13] is

$$\mathsf{sk} = (g^{\frac{y}{\alpha+s}}, \quad g^s, \quad h^s),$$

where (y, α) is the secret key of the group manager and s is the user identity.

We have two technical difficulties when adapting this key structure. Firstly, we have to add the user secret key to the system to avoid the possible attacks from malicious attribute-issuing authority. Secondly, we have to replace the simple identity-based construction into more complicated attributed-based setting.

For the first problem, we consider y as the secret key of the user and $Y = \hat{g}^y$ as the corresponding public key. In order to avoid the user changing his/her public and secret key after obtaining his/her attributed-based secret key, the first part of the key is changed to $(Yg)^{\frac{1}{\alpha+s}}$. Hence the user public key is bound with the attributed-based secret key if the discrete logarithm $\log_g \hat{g}$ is hard.

For the second problem, we attempt to merge the attribute-based key system of the instantiation 3 in [22] into the above construction. Therefore, we change $s = at$ where a is also the secret key of the attribute-generating authority. Then for each attribute x the user possesses, the user also obtains h_x^t, where h_x is the public parameters. Therefore, the final key structure for the attribute set $\mathcal{A}$ is:

$$\mathsf{sk} = ((Yg)^{\frac{1}{\alpha+at}}, \quad g^{at}, \quad h^{at}, \quad h_x^t \text{ for all } x \in \mathcal{A}).$$

The authentication protocol works as follows. The user runs the zero-knowledge proof of knowledge protocol to prove that he knows his attribute secret key $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$ and user secret key $\mathsf{USK} = y$ (both correspond to the user public key UPK) such that

$$\Upsilon(\mathcal{A}) = 1 \quad \wedge \quad C = \hat{e}(g,g)^{\frac{1}{y+H(k)}}$$

Note that the tuples $\{\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}, y, Y\}$ will not be sent to the service provider. The user will only prove (in zero-knowledge) that he has the knowledge of these tuples. If the proof goes through, that means the user has the required attributes and it is his k-th time authentication. The service provider stores C in its database and checks whether the newly received C exists in its database or not. If it is already in the database, that means the user has re-used k and will reject his authentication.

C. Construction

Let $\mathbb{A}$ be the desired universe of attributes. This construction supports all claim-predicates whose monotone span programs have width at most n, where n is an arbitrary parameter. We treat $\mathbb{A} = \mathbb{Z}_p^*$ as the universe of attributes, where p is the size of the cyclic group used in the scheme.

Our attribute-based access control construction is as follows.

$\underline{\mathsf{TSetup}}$: Let λ be a security parameter. The trustee runs $\mathsf{param} = (\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$, randomly picks generators $g, \hat{g}, h \in \mathbb{G}$. For $i \in [1,n]$, it picks $\delta_i, \in \mathbb{Z}_p$. It also picks a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. For $i \in [1,n]$, it sets

$$h_i = h^{\delta_i}, \quad \hat{h}_i = h^{\frac{1}{\delta_i}}.$$

It publishes $\mathsf{TPK} = (\mathsf{param}, g, \hat{g}, h, h_1, \ldots, h_n, \hat{h}_1, \ldots, \hat{h}_n, H)$.

$\underline{\mathsf{ASetup}}$: The attribute-issuing authority randomly picks $\alpha, a \in \mathbb{Z}_p$. Suppose $\phi : \mathbb{A} \rightarrow [1,n]$ is a mapping function. It publishes authority public key $\mathsf{APK} = (g^\alpha, \hat{h}_1^a, \ldots, \hat{h}_n^a, \phi)$ and sets authority secret key $\mathsf{ASK} = (\alpha, a)$.

$\underline{\mathsf{USetup}}$: The user randomly picks $y \in \mathbb{Z}_p$. It publishes user public key $\mathsf{UPK} = Y = \hat{g}^y$ and sets user secret key $\mathsf{USK} = y$.

$\underline{\mathsf{AttrGen}}$: The key generation algorithm takes as input $\mathsf{TPK}, \mathsf{APK}, \mathsf{UPK} = Y$ and a subset of attributes $\mathcal{A} \subset \mathbb{A}$. The user runs a zero-knowledge proof of knowledge protocol PK_0 with the attribute-issuing authority to prove the knowledge of his user secret key y:

$$PK_0\{y : Y = \hat{g}^y\}.$$

This proof of knowledge of discrete logarithm is straightforward and is shown in the next subsection. If the proof is correct, the attribute-issuing authority first chooses a random

$t \in \mathbb{Z}_p$ and uses the authority secret key ASK to create the attribute secret key $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$ for the user as

$$K = (Yg)^{\frac{1}{\alpha+at}}, \quad L_1 = g^{at}, \quad L_2 = h^{at},$$
$$K_x = h^t_{\phi(x)} \qquad \forall x \in \mathcal{A}.$$

Auth: The authentication algorithm takes as input the public keys TPK, APK and a claim-predicate Υ. The user has some additional inputs including an attribute secret key $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$ for attribute $\mathcal{A}$, a user secret key $\mathsf{USK} = y$ and the number of past authentication k. Assume $\Upsilon(\mathcal{A}) = 1$.

More concretely, first convert Υ to its corresponding monotone span program $\mathbf{M} = (M_{i,j}) \in (\mathbb{Z}_p)^{\ell \times m}$, with row labeling $\rho : [1, \ell] \to \mathbb{A}$. The maximum number of authentication M is determined by Υ. (We refer the reader to the PhD thesis of [9] and the appendix of [19] for a discussion of how to perform this conversion.) Also compute the vector $\vec{v} = (v_1, \dots v_\ell) \in \mathbb{Z}_p^\ell$ that corresponds to the satisfying assignment $\mathcal{A}$. Then the user parses $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$ as $(K, L_1, L_2, \{K_x | x \in \mathcal{A}\})$. Then he computes $C = \hat{e}(g,g)^{\frac{1}{y+H(k)}}$ and sends (C, k) to the service provider and runs the following proof of knowledge protocol with the service provider:

$$\begin{aligned}
&PK_1\{(K, L_1, L_2, K^{v_1}_{\rho(1)}, \dots, K^{v_\ell}_{\rho(\ell)}, y) : \\
&\quad \hat{e}(K, g^\alpha L_1) = \hat{e}(\hat{g}^y g, g) \\
&\wedge \quad \hat{e}(h, L_1) = \hat{e}(L_2, g) \\
&\wedge \quad C = \hat{e}(g,g)^{\frac{1}{y+H(k)}} \\
&\wedge \quad \prod_{i=1}^{\ell} \hat{e}(K^{v_i}_{\rho(i)}, \hat{h}^{aM_{i,j}}_{\phi(\rho(i))}) = \begin{cases} \hat{e}(L_2, h), & \text{if } j = 1, \\ 1, & \text{if } 1 < j \le m. \end{cases} \}
\end{aligned}$$

If the above proof is valid and $k < M$, the service provider checks if C is used in past authentications. If true, the service provider rejects. If false, the service provider accepts the authentication and stores C for future checking. The user updates $k \leftarrow k + 1$.

Remarks:

1) The user may not have $K_{\rho(i)}$ for every attribute $\rho(i)$ mentioned in the claim-predicate. But when this is the case, $v_i = 0$, and so the value is not needed. Detailed implementation of PK_1 is given in the next subsection.
2) If the user does *not* increment k, his/her next round authentication will be rejected. It can be easily seen from the fact that as $C = \hat{e}(g,g)^{\frac{1}{y+H(k)}}$, y is fixed (it is user secret key) and if k is unchanged, then C is also unchanged and thus will be detected by the service provider.
3) If the user has been authenticated for more than M times (that is, $k \ge M$), although the service provider will reject its further access, it still *cannot* revoke its identity. The privacy of the user is still preserved.

D. Implementations of Protocol PK_0 and PK_1

We first briefly introduce the proof of knowledge as defined in [10]. Intuitively, a two-party protocol constitutes a system for proofs of knowledge if one party (called the verifier) is convinced then the other party (called the prover) indeed knows some "knowledge".

If R is a binary relation, we let $R(x) = \{y : (x, y) \in R\}$ and the language $L_R = \{x : \exists y \text{ such that } (x, y) \in R\}$. If $(x, y) \in R$, we call y the witness of x.

A proof of knowledge is a two-party protocol with the following properties:

1) **Completeness**: If $(x, y) \in R$, the honest prover who knows witness y for x succeeds in convincing the honest verifier of his knowledge.
2) **Soundness**: If $(x, y) \notin R$, no cheating prover can convince the honest verifier that $(x, y) \in R$, except with some small probability. It can be captured by the existence of a *knowledge extractor* E to extract the witness y: given oracle access to a cheating prover P, the probability that E outputs y must be at least as high as the success probability of P in convincing the verifier.

For a zero-knowledge proof of knowledge, it has the extra property of **Zero-knowledge**: no cheating verifier learns anything other than $(x, y) \in R$. It is formalized by showing that every cheating verifier has some simulator that can produce a transcript that is indistinguishable with an interaction between the honest prover and the cheating (or honest) verifier.

Implementation of Protocol PK_0:

$$PK_0\{y : Y = \hat{g}^y\}.$$

Suppose Alice wants to prove the knowledge of y to Bob. Alice picks a random number $r \in \mathbb{Z}_p$ and send the commitment $R = \hat{g}^r$ to Bob. Bob returns a random challenge $c \in \mathbb{Z}_p$. Alice computes the response $z = r + cy$. Bob verifies that $\hat{g}^z = R \cdot Y^c$. The soundness and the zero-knowledgeness are straightforward.

(Soundness) Suppose the simulator is given a discrete logarithm instance $(\hat{g}, Y)$, it uses Y as the public key. Given a transcript (R, c, z), it rewinds to obtain another transcript (R, c', z'). Since both of them are valid, it means that

$$\hat{g}^z Y^{-c} = \hat{g}^{z'} Y^{-c'}.$$

Hence the simulator can obtain $\frac{z-z'}{c-c'}$ as the solution of $\log_{\hat{g}} Y$.

(Zero-knowledge) Given the public key $\hat{g}, Y$, the simulator can randomly picks $c, z \in \mathbb{Z}_p$ and computes $R = \hat{g}^z Y^{-c}$. The transcript (R, c, z) has the same distribution as those coming from Alice and Bob.

Implementation of Protocol PK_1:

$$\begin{aligned}
&PK_1\{(K, L_1, L_2, K^{v_1}_{\rho(1)}, \dots, K^{v_\ell}_{\rho(\ell)}, y) : \\
&\quad \hat{e}(K, g^\alpha L_1) = \hat{e}(\hat{g}^y g, g) \\
&\wedge \quad \hat{e}(h, L_1) = \hat{e}(L_2, g) \\
&\wedge \quad C = \hat{e}(g,g)^{\frac{1}{y+H(k)}} \\
&\quad \wedge \quad \prod_{i=1}^{\ell} \hat{e}(K^{v_i}_{\rho(i)}, \hat{h}^{aM_{i,j}}_{\phi(\rho(i))}) = \begin{cases} \hat{e}(L_2, h), & \text{if } j = 1, \\ 1, & \text{if } 1 < j \le m. \end{cases} \}
\end{aligned}$$

$\mathcal{PK}_1$ is a zero-knowledge proof-of-knowledge protocol which allows the prover Alice to convince a verifier Bob that she knows a set of elements

$$
PK_1' \left\{ \begin{array}{cccrcl}
 & & & \mathfrak{B}_K & = & \mathfrak{g}^{\nu_K}\mathfrak{h}^{\mu_K} \\
 & \wedge & & 1_{\mathbb{G}} & = & \mathfrak{B}_K^{-\mu_{L_1}}\mathfrak{g}^{\beta_2}\mathfrak{h}^{\beta_1} \\
 & \wedge & & \hat{e}(\mathfrak{A}_K, g^{\alpha}\mathfrak{A}_{L_1}) & = & \hat{e}(\mathfrak{g}, g^{\alpha}\mathfrak{A}_{L_1})^{\mu_K}\hat{e}(\mathfrak{A}_K, \mathfrak{g})^{\mu_{L_1}}\hat{e}(\mathfrak{g}, \mathfrak{g}^{-1})^{\beta_1}\hat{e}(\hat{g}, g)^{y} \\
 & \wedge & & \hat{e}(h, \mathfrak{A}_{L_1})\hat{e}(\mathfrak{A}_{L_2}, g^{-1}) & = & \hat{e}(h, \mathfrak{g})^{\mu_{L_1}}\hat{e}(\mathfrak{g}, g^{-1})^{\mu_{L_2}} \\
\begin{pmatrix} \mu_K, \nu_K, \mu_{L_1}, \\ \mu_{L_2}, \beta_1, \beta_2, \\ \mu_1, \ldots, \mu_\ell, \\ y \end{pmatrix} & : \wedge & & C^{-H(k)}\hat{e}(g, g) & = & C^{y} \\
 & \wedge & \hat{e}(\mathfrak{A}_{L_2}, h^{-1})\prod_{i=1}^{\ell}\hat{e}(\mathfrak{A}_i, \hat{H}_{i,1}) & = & \hat{e}(\mathfrak{g}^{-1}, h)^{\mu_{L_2}}\prod_{i=1}^{\ell}\hat{e}(\mathfrak{g}, \hat{H}_{i,1})^{\mu_i} \\
 & \wedge & \prod_{i=1}^{\ell}\hat{e}(\mathfrak{A}_i, \hat{H}_{i,2}) & = & \prod_{i=1}^{\ell}\hat{e}(\mathfrak{g}, \hat{H}_{i,2})^{\mu_i} \\
 & & & \vdots & & \\
 & \wedge & & & & \\
 & \wedge & \prod_{i=1}^{\ell}\hat{e}(\mathfrak{A}_i, \hat{H}_{i,m}) & = & \prod_{i=1}^{\ell}\hat{e}(\mathfrak{g}, \hat{H}_{i,m})^{\mu_i}
\end{array} \right\}
$$

Fig. 1: PK_1 representation

$(K, L_1, L_2, K_{\rho(1)}^{v_1}, \ldots, K_{\rho(\ell)}^{v_\ell}, y) \in \mathbb{G}^{\ell+2} \times \mathbb{Z}_p^*$ satisfying the following $m+3$ equations simultaneously.

$$
\begin{aligned}
\hat{e}(K, g^{\alpha}L_1) &= \hat{e}(\hat{g}^{y}g, g) \\
\hat{e}(h, L_1) &= \hat{e}(L_2, g) \\
C &= \hat{e}(g, g)^{\frac{1}{y+H(k)}} \\
\prod_{i=1}^{\ell}\hat{e}(K_{\rho(i)}^{v_i}, \hat{h}_{\phi(\rho(i))}^{aM_{i,1}}) &= \hat{e}(L_2, h) \\
\prod_{i=1}^{\ell}\hat{e}(K_{\rho(i)}^{v_i}, \hat{h}_{\phi(\rho(i))}^{aM_{i,j}}) &= 1 \qquad \text{for } j \in [2, m].
\end{aligned}
$$

For $i = 1$ to ℓ we use κ_i to denote the value $K_{\rho(i)}^{v_i}$. The reason is that Alice is proving to Bob that she knows $K_{\rho(i)}^{v_i}$, which does not means she know $K_{\rho(i)}$ or v_i. Likewise, we use $\hat{H}_{i,j}$ to denote the value $\hat{h}_{\phi(\rho(i))}^{aM_{i,j}}$ for $i = 1$ to ℓ, $j = 1$ to m. The value of $\hat{H}_{i,j}$ can be computed locally by both Alice and Bob.

While there are known and efficient techniques for proving discrete logarithm relationship as well as general pairing-product equation, the above set of equations does not fall into those categories. Thus, additional tricks are applied. Specifically, Bob chooses additional generators $\mathfrak{g}, \mathfrak{h}$ of group $\mathbb{G}$ so that discrete logarithm of $\mathfrak{h}$ to base $\mathfrak{g}$ is unknown to Alice. Next, Alice chooses a set of random values in $\mathbb{Z}_p^*$: μ_K, ν_K, μ_{L_1}, μ_{L_2}, μ_1, …, μ_ℓ and computes the following $\ell+4$ auxiliary values.

$$
\begin{array}{rclcrcl}
\mathfrak{A}_K &=& K\mathfrak{g}^{\mu_K}, & & \mathfrak{B}_K &=& \mathfrak{g}^{\nu_K}\mathfrak{h}^{\mu_K}, \\
\mathfrak{A}_{L_1} &=& L_1\mathfrak{g}^{\mu_{L_1}}, & & \mathfrak{A}_{L_2} &=& L_2\mathfrak{g}^{\mu_{L_2}}, \\
\mathfrak{A}_1 &=& \kappa_1\mathfrak{g}^{\mu_1}, & \cdots & \mathfrak{A}_\ell &=& \kappa_\ell\mathfrak{g}^{\mu_\ell}
\end{array}
$$

Alice sends these $\ell+4$ auxiliary values to Bob. Then, PK_1 can be represented as a zero-knowledge proof-of-knowledge of the set of $\ell+7$ values $(\mu_K, \nu_K, \mu_{L_1}, \mu_{L_2}, \mu_1 \ldots, \mu_\ell, y, \beta_1, \beta_2)$, where $\beta_1 = \mu_K\mu_{L_1}$ and $\beta_2 = \nu_K\mu_{L_1}$. Using the Camenisch-Stadler notations, the representation of PK_1 is shown in Figure 1.

a) Complete Specification of PK_1': : For completeness, we describe the honest-verifier zero-knowledge version of PK_1' assume the auxiliary elements are computed by Alice and has been sent to Bob. Note that the honest-verifier zero-knowledge protocol described below can be turned into full zero-knowledge using the technique described in [16].

- *Commitment Phase*: Alice randomly picks ρ_{μ_K}, ρ_{ν_K}, $\rho_{\mu_{L_1}}$, $\rho_{\mu_{L_2}}$, ρ_{β_1}, ρ_{β_2}, ρ_{μ_1}, …, ρ_{μ_ℓ}, $\rho_y \in_R \mathbb{Z}_p^*$ and computes the following $m+5$ values, usually referred to as commitments.

$$
\begin{aligned}
T_1 &= \mathfrak{g}^{\rho_{\nu_K}}\mathfrak{h}^{\rho_{\mu_K}}, \\
T_2 &= \mathfrak{B}_K^{-\rho_{\mu_{L_1}}}\mathfrak{g}^{\rho_{\beta_2}}\mathfrak{h}^{\rho_{\beta_1}}, \\
T_3 &= \hat{e}(\mathfrak{g}, g^{\alpha}\mathfrak{A}_{L_1})^{\rho_{\mu_K}}\hat{e}(\mathfrak{A}_K, \mathfrak{g})^{\rho_{\mu_{L_1}}}\hat{e}(\mathfrak{g}, \mathfrak{g}^{-1})^{\rho_{\beta_1}}\hat{e}(\hat{g}, g)^{\rho_y}, \\
T_4 &= \hat{e}(h, \mathfrak{g})^{\rho_{\mu_{L_1}}}\hat{e}(\mathfrak{g}, g^{-1})^{\rho_{\mu_{L_2}}}, \\
T_5 &= C^{\rho_y}, \\
S_1 &= \hat{e}(\mathfrak{g}^{-1}, h)^{\rho_{\mu_{L_2}}}\prod_{i=1}^{\ell}\hat{e}(\mathfrak{g}, \hat{H}_{i,1})^{\rho_{\mu_i}}, \\
S_2 &= \prod_{i=1}^{\ell}\hat{e}(\mathfrak{g}, \hat{H}_{i,2})^{\rho_{\mu_i}}, \\
&\vdots \\
S_m &= \prod_{i=1}^{\ell}\hat{e}(\mathfrak{g}, \hat{H}_{i,m})^{\rho_{\mu_i}}.
\end{aligned}
$$

Alice sends $(T_1, \ldots, T_5, S_1, \ldots S_m)$ to Bob. In practice, $\hat{e}(\mathfrak{g}, \mathfrak{g})$, $\hat{e}(h, \mathfrak{g})$ and $\hat{e}(\mathfrak{g}, g)$ can be pre-computed and $S_1, \ldots, S_m$ can be computed as follow, so that only a constant number of pairing operation is needed:

$$
\begin{aligned}
S_1 &= \hat{e}(\mathfrak{g}^{-1}, h)^{\rho_{\mu_{L_2}}}\hat{e}\Big(\mathfrak{g}, \textstyle\prod_{i=1}^{\ell}\hat{H}_{i,1}^{\rho_{\mu_i}}\Big), \\
S_2 &= \hat{e}\Big(\mathfrak{g}, \textstyle\prod_{i=1}^{\ell}\hat{H}_{i,2}^{\rho_{\mu_i}}\Big), \\
&\vdots \\
S_m &= \hat{e}\Big(\mathfrak{g}, \textstyle\prod_{i=1}^{\ell}\hat{H}_{i,m}^{\rho_{\mu_i}}\Big)
\end{aligned}
$$

- *Challenge Phase*: Bob randomly picks $c \in_R \mathbb{Z}_p$ and sends c to Alice.
- *Response Phase*: Alice computes the following $\ell+7$ values, usually referred to as responses.

$$
\begin{aligned}
z_{\mu_K} &= \rho_{\mu_K} - c\mu_K, \\
z_{\nu_K} &= \rho_{\nu_K} - c\nu_K, \\
z_{\mu_{L_1}} &= \rho_{\mu_{L_1}} - c\mu_{L_1}, \\
z_{\mu_{L_2}} &= \rho_{\mu_{L_2}} - c\mu_{L_2}, \\
z_{\beta_1} &= \rho_{\beta_1} - c\mu_K\mu_{L_1}, \\
z_{\beta_2} &= \rho_{\beta_2} - c\nu_K\mu_{L_1}, \\
z_{\mu_1} &= \rho_{\mu_1} - c\mu_1, \ \ldots \ z_{\mu_\ell} = \rho_{\mu_\ell} - c\mu_\ell, \\
z_y &= \rho_y - cy
\end{aligned}
$$

Alice sends $(z_{\mu_K}, z_{\nu_K}, z_{\mu_{L_1}}, z_{\mu_{L_2}}, z_{\beta_1}, z_{\beta_2}, z_{\mu_1}, \ldots, z_{\mu_\ell}, z_y)$ to Bob.

- *Verification Phase*: Bob validates the proof by evaluating

the following $m+5$ equations.

$$\begin{aligned}
T_1 &\stackrel{?}{=} \mathfrak{B}_K^c \mathfrak{g}^{z_{\nu_K}} \mathfrak{h}^{z_{\mu_K}},\\
T_2 &\stackrel{?}{=} \mathfrak{B}_K^{-z_{\mu_{L_1}}} \mathfrak{g}^{z_{\beta_2}} \mathfrak{h}^{z_{\beta_1}},\\
T_3 &\stackrel{?}{=} \hat{e}(\mathfrak{A}_K, g^{\alpha}\mathfrak{A}_{L_1})^c \hat{e}(\mathfrak{g}, g^{\alpha}\mathfrak{A}_{L_1})^{z_{\mu_K}} \hat{e}(\mathfrak{A}_K, \mathfrak{g})^{z_{\mu_{L_1}}}\\
&\quad \cdot \hat{e}(\mathfrak{g}, \mathfrak{g}^{-1})^{z_{\beta_1}} \hat{e}(\hat{g}, g)^{z_y},\\
T_4 &\stackrel{?}{=} \big(\hat{e}(h, \mathfrak{A}_{L_1})\hat{e}(\mathfrak{A}_{L_2}, g^{-1})\big)^c \hat{e}(h, \mathfrak{g})^{z_{\mu_{L_1}}}\\
&\quad \cdot \hat{e}(\mathfrak{g}, g^{-1})^{z_{\mu_{L_2}}},\\
T_5 &\stackrel{?}{=} \big(C^{-H(k)}\hat{e}(g,g)\big)^c C^{z_y},\\
S_1 &\stackrel{?}{=} \big(\hat{e}(\mathfrak{A}_{L_2}, h^{-1}) \textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,1})\big)^c\\
&\quad \cdot \hat{e}(\mathfrak{g}^{-1}, h)^{z_{\mu_{L_2}}} \textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{g}, \hat{H}_{i,1})^{z_{\mu_i}},\\
S_2 &\stackrel{?}{=} \big(\textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,2})\big)^c \textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{g}, \hat{H}_{i,2})^{z_{\mu_i}},\\
&\vdots\\
S_m &\stackrel{?}{=} \big(\textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,m})\big)^c \textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{g}, \hat{H}_{i,m})^{z_{\mu_i}}
\end{aligned}$$

Bob accepts the proof if and only if all the above equalities holds. In practice, $\hat{e}(\hat{g}, g)$, $\hat{e}(h, \mathfrak{g})$, $\hat{e}(\mathfrak{g}, \mathfrak{g})$, $\hat{e}(\mathfrak{g}, \mathfrak{g}^{-1})$, $\hat{e}(\mathfrak{g}, g)$ and $\hat{e}(g, g)$ can be pre-computed and $S_1, \ldots, S_m$ can be computed as follow, so that a number of pairing operation can be saved:

$$\begin{aligned}
S_1 &\stackrel{?}{=} \Big(\hat{e}(\mathfrak{A}_{L_2}, h^{-1}) \textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,1})\Big)^c\\
&\quad \cdot \hat{e}(\mathfrak{g}^{-1}, h)^{z_{\mu_{L_2}}} \hat{e}\Big(\mathfrak{g}, \textstyle\prod_{i=1}^{\ell} \hat{H}_{i,1}^{z_{\mu_i}}\Big),\\
S_2 &\stackrel{?}{=} \big(\textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,2})\big)^c \hat{e}\Big(\mathfrak{g}, \textstyle\prod_{i=1}^{\ell} \hat{H}_{i,2}^{z_{\mu_i}}\Big),\\
&\vdots\\
S_m &\stackrel{?}{=} \big(\textstyle\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,m})\big)^c \hat{e}\Big(\mathfrak{g}, \textstyle\prod_{i=1}^{\ell} \hat{H}_{i,m}^{z_{\mu_i}}\Big)
\end{aligned}$$

In addition, the auxiliary values can be sent in conjunction with the commitment in the commitment phase. Thus, the resulting protocol still consists of three message flows.

b) Soundness and Zero-Knowledgeness of PK_1: Our instantiation of PK_1 thus consists of the auxiliary values $\mathfrak{A}_K$, $\mathfrak{B}_K$, $\mathfrak{A}_{L_1}$ $\mathfrak{A}_{L_2}$, $\mathfrak{A}_1$, $\cdots$, $\mathfrak{A}_\ell$, in addition to PK_1'. It is straightforward to show that PK_1' is honest-verifier zero-knowledge. That is, there exists an efficient algorithm, called simulator $\mathcal{S}'$, which, on input a random challenge c, can output a communication transcript $(T_1, \ldots, T_5, S_1, \ldots, S_m)$ and $(z_{\mu_K}, z_{\nu_K}, z_{\mu_{L_1}}, z_{\mu_{L_2}}, z_{\beta_1}, z_{\beta_2}, z_{\mu_1}, \ldots, z_{\mu_\ell}, z_y)$ whose distribution is identical to those coming from Alice. Furthermore, PK_1' is sound, meaning that there exists another efficient algorithm, called extractor $\mathcal{E}'$, which is capable of outputting the underlying set (μ_K, ν_K, μ_{L_1}, μ_{L_2}, β_1, β_2, μ_1, …, μ_ℓ, y) of knowledge when it is given blackbox access to a prover in PK_1'.

Honest-Verifier Zero-Knowledgeness of PK_1 Thus, to show that our instantiation of PK_1 is honest-verifier zero-knowledge, we only need to demonstrate to construct another simulator $\mathcal{S}$, which is capable of outputting the transcript of the whole PK_1 on input challenge c.

Our construction of $\mathcal{S}$ simply picks at random $\mathfrak{A}_K$, $\mathfrak{B}_K$, $\mathfrak{A}_{L_1}$ $\mathfrak{A}_{L_2}$, $\mathfrak{A}_1, \cdots, \mathfrak{A}_\ell \in_R \mathbb{G}$ and invokes the zero-knowledge simulator $\mathcal{S}'$ for PK_1'. The transcript outputted by $\mathcal{S}'$ will be correct, and it remains to show that the auxiliary values picked by $\mathcal{S}$ is also correctly distributed. The argument is as follows: For any set of witnesses $K, L_1, L_2, \kappa_1, \ldots, \kappa_\ell$, there exists a unique set of randomness μ_K, ν_K, μ_{L_1}, μ_{L_2}, μ_1, …, μ_ℓ such that the auxiliary values are computed by from the set of witnesses using the particular set of randomness. Thus, the auxiliary values picked by $\mathcal{S}$ is correctly distributed.

Soundness of PK_1. Soundness of PK_1' guarantees existence of a simulator $\mathcal{E}'$ which allows our construction of $\mathcal{E}$ to extract from the prover the set of values (μ_K, ν_K, μ_{L_1}, μ_{L_2}, β_1, β_2, μ_1, …, μ_ℓ, y). It remains to show how $\mathcal{E}$ can obtain the set of witnesses $K, L_1, L_2, \kappa_1, \ldots, \kappa_\ell, y$ which satisfies the equations for PK_1.

$\mathcal{E}$ first invokes $\mathcal{E}'$. The, it computes $\hat{\kappa_i}$ as $\mathfrak{A}_i \mathfrak{g}^{-\mu_i}$ for $i = 1$ to ℓ. Furthermore, $\mathcal{E}$ computes $\hat{K} = \mathfrak{A}_K \mathfrak{g}^{-\mu_K}$, $\hat{L}_1 = \mathfrak{A}_{L_1} \mathfrak{g}^{-\mu_{L_1}}$ and $\hat{L}_2 = \mathfrak{A}_{L_2} \mathfrak{g}^{-L_2}$.

Due to soundness of PK_1', for $j = 2$ to m:

$$\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,j}) = \prod_{i=1}^{\ell} \hat{e}(\mathfrak{g}, \hat{H}_{i,j})^{\mu_i}.$$

Thus, $\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,j}) = \prod_{i=1}^{\ell} \hat{e}(\mathfrak{g}, \hat{H}_{i,j})^{\mu_i}$. It implies $\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i \mathfrak{g}^{-\mu_i}, \hat{H}_{i,j}) = 1_{\mathbb{G}}$. That is,

$$\prod_{i=1}^{\ell} \hat{e}(\hat{\kappa}_i, \hat{H}_{i,j}) = 1_{\mathbb{G}}.$$

Similarly, soundness of PK_1' assures:

$$\begin{aligned}
&\hat{e}(\mathfrak{A}_{L_2}, h^{-1}) \prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i, \hat{H}_{i,1})\\
&= \hat{e}(\mathfrak{g}^{-1}, h)^{\mu_{L_2}} \prod_{i=1}^{\ell} \hat{e}(\mathfrak{g}, \hat{H}_{i,1})^{\mu_i}.
\end{aligned}$$

Rearranging the terms:

$$\prod_{i=1}^{\ell} \hat{e}(\mathfrak{A}_i \mathfrak{g}^{-\mu_i}, \hat{H}_{i,1}) = \hat{e}(\mathfrak{A}_{L_2} \mathfrak{g}^{-\mu_{L_2}}, h).$$

Thus,

$$\prod_{i=1}^{\ell} \hat{e}(\hat{\kappa}_i, \hat{H}_{i,1}) = \hat{e}(\hat{L}_2, h).$$

In addition, soundness of PK_1' guarantees:

$$C^{-H(k)} \hat{e}(g,g) = C^y.$$

It means $C^{y+H(k)} = \hat{e}(g,g)$. That is:

$$C = \hat{e}(g,g)^{\frac{1}{y+H(k)}}.$$

Furthermore, soundness of PK_1' guarantees:

$$\hat{e}(h, \mathfrak{A}_{L_1})\hat{e}(\mathfrak{A}_{L_2}, g^{-1}) = \hat{e}(h, \mathfrak{g})^{\mu_{L_1}} \hat{e}(\mathfrak{g}, g^{-1})^{\mu_{L_2}}.$$

Rearranging the terms:

$$\hat{e}(h, \mathfrak{A}_{L_1} \mathfrak{g}^{-\mu_{L_1}}) = \hat{e}(\mathfrak{A}_{L_2} \mathfrak{g}^{-\mu_{L_2}}, g).$$

This is,

$$\hat{e}(h, \hat{L}_1) = \hat{e}(\hat{L}_2, g).$$

Finally, soundness of PK_1' guarantees:

$$\mathfrak{B}_K = \mathfrak{g}^{\nu_K} \mathfrak{h}^{\mu_K} \ \wedge \ 1_{\mathbb{G}} = \mathfrak{B}_K^{-\mu_{L_1}} \mathfrak{g}^{\beta_2} \mathfrak{h}^{\beta_1}.$$

Rearranging the terms:

$$\mathfrak{B}_K = \mathfrak{g}^{\nu_K}\mathfrak{h}^{\mu_K} \ \wedge \ \mathfrak{B}_K^{\mu_{L_1}} = \mathfrak{g}^{\beta_2}\mathfrak{h}^{\beta_1}.$$

Due to the discrete logarithm assumption, $\beta_1 = \mu_K\mu_{L_1}$. Recall that soundness of PK_1' guarantees:

$$\begin{aligned} & \hat{e}(\mathfrak{A}_K, g^{\alpha}\mathfrak{A}_{L_1}) \\ = \ & \hat{e}(\mathfrak{g}, g^{\alpha}\mathfrak{A}_{L_1})^{\mu_K}\hat{e}(\mathfrak{A}_K, \mathfrak{g})^{\mu_{L_1}}\hat{e}(\mathfrak{g}, \mathfrak{g}^{-1})^{\beta_1}\hat{e}(\hat{g}, g)^{y}. \end{aligned}$$

Putting $\beta_1 = \mu_K\mu_{L_1}$ and rearranging the terms:

$$\hat{e}(\mathfrak{A}_K\mathfrak{g}^{-\mu_K}, g^{\alpha}\mathfrak{A}_{L_1}\mathfrak{g}^{-\mu_{L_1}}) = \hat{e}(\hat{g}, g)^{y}.$$

Thus,

$$\hat{e}(\hat{K}, g^{\alpha}\hat{L}_1) = \hat{e}(\hat{g}, g)^{y}.$$

$\mathcal{E}$ outputs $(\hat{K}, \hat{L}_1, \hat{L}_2, \hat{\kappa}_1, \ldots, \hat{\kappa}_\ell, y)$ as the set of witnesses satisfying PK_1. Therefore, our instantiation of PK_1 is sound.

E. Security Analysis

Theorem 1: The above scheme is authenticate in the generic group model.

Proof: We divide the proof into two parts according to our definition in Section III-B.

Part 1: The proof is given in the generic group model following [22]. Denote the public information $h = g^{\delta_0}, \hat{g} = g^{\xi}$, for some $\delta_0, \xi \in \mathbb{Z}_p$. The public information includes:

$$(g, g^{\delta_0}, g^{\alpha}, g^{\xi}, \{g^{\delta_0\delta_i}, g^{\frac{\delta_0}{\delta_i}}, g^{\frac{a\delta_0}{\delta_i}}\}_{i\in[1,n]}).$$

We first observe that the AttrGen Oracle and the Auth Oracle can be sampled by generating the k-th secret key for $\mathcal{A}_k$ of the form

$$(g^{\frac{1+\xi y_k}{\alpha+at_k}}, g^{at_k}, g^{at_k\delta_0}, \{g^{\delta_0 t_k \delta_{\phi(x)}}\}_{x\in\mathcal{A}_k}).$$

We now proceed with the proof, following the standard approach for generic groups. Let $\mathsf{Lin(S)}$ be the set of functions that are linear in the terms in the set S. Let $\mathsf{Hom(S)}$ be the subset of $\mathsf{Lin(S)}$ of homogeneous functions whose constant coefficient is zero. In the generic group model, the forgery generated by the adversary is a fixed function of the inputs given in the security experiment.

Suppose the adversary returns a claim-predicate Υ^*, a UPK^*, a number M^* and runs the Auth_P protocol correctly. Since the PK_1 protocols is sound [4], the adversary has a secret key for some attribute set $\mathcal{A}^*$ and UPK^*, such that $\Upsilon^*(\mathcal{A}^*) = 1$.

More concretely, first convert Υ^* to its corresponding monotone span program $\mathbf{M}^* = (M^*_{i,j}) \in (\mathbb{Z}_p)^{\ell\times m}$, with row labeling $\rho^* : [1,\ell] \to \mathbb{A}$. Also compute the vector $\vec{v}^* = (v_1^*, \ldots v_\ell^*) \in \mathbb{Z}_p^\ell$ that corresponds to the satisfying assignment $\mathcal{A}^*$. Suppose the adversary's secret extracted from PK_1 has the form

$$(g^{k^*}, g^{l_1^*}, g^{l_2^*}, \{g^{k_j^*}\}_{j=\rho^*(1),\ldots,\rho^*(\ell)}, y^*)$$

To be a forgery, we require that $k^*(\alpha+l_1^*) = 1+\xi y^*$, $\delta_0 l_1^* = l_2^*$ and

$$\sum_{i=1}^{\ell} k^*_{\rho^*(i)} \cdot \frac{aM^*_{i,j}}{\delta_{\phi(\rho^*(i))}} = l_1^*\delta_0 z_j \quad \forall j \in [1,m],$$

where $(z_1, z_2, \ldots, z_m) = (1, 0, \ldots, 0)$. The rest of our proof proceeds by assuming these constraints are functionally equivalent, and eventually obtaining a contradiction: that there exists a $k_0 \in [n]$ such that $\Upsilon^*(\mathcal{A}_{k_0}) = 1$. In other words, the adversary could have authenticated legitimately with the signing key for $\mathcal{A}_{k_0}$, and thus the output is not a forgery. We know that $k^*, l_1^*, l_2^*, \{k_j^*\}_{j=\rho^*(1),\ldots,\rho^*(\ell)} \in \mathsf{Lin}(\Gamma)$, where

$$\begin{aligned} \Gamma = (\{1, \delta_0, \alpha, \xi\} \cup \{\delta_0\delta_j, \frac{a\delta_0}{\delta_j}, \frac{\delta_0}{\delta_j}\}_{j\in[1,n]} \\ \cup \{\frac{1+\xi y_k}{\alpha+at_k}, at_k, at_k\delta_0, \{\delta_0\delta_{\phi(x)}t_k\}_{\forall x\in\mathcal{A}_k}\}_{k\in[1,q]}), \end{aligned}$$

where q is the number of AttrGen Oracle and Auth Oracle. The rest of our analysis proceeds by comparing terms in these constraints. We can show that the multi-linear functions given by the adversary's forgery cannot contain terms of certain kinds. Since $\delta_0 l_1^* = l_2^*$, we get that:

$$l_1^* \in \mathsf{Hom}(\{\delta_j, \frac{1}{\delta_j}, \frac{a}{\delta_j}\}_{j\in[1,n]} \cup \{at_k, \{\delta_{\phi(x)}t_k\}_{x\in\mathcal{A}_k}\}_{k\in[1,q]}).$$

Next, observe that $k^*(\alpha + l_1^*) = 1 + \xi y^*$. Therefore

$$y^* \in \mathsf{Hom}(y_1, \ldots, y_q).$$

As a result,

$$l_1^* \in \mathsf{Hom}(\frac{a}{\delta_1}, \ldots, \frac{a}{\delta_n}, at_1, \ldots, at_q).$$

Denote $w_k^* = \frac{1+\xi y_k}{\alpha+at_k}$. Then

$$k^*(\alpha + l_1^*) = \sum_k w_k^* A_k(\alpha + at_k) + C,$$

for some constant $C \in \mathbb{Z}_p$ and coefficients A_k. The above equation has degree 1 random variables α and a. Therefore, $l_1^* \in \mathsf{Hom}(at_1, \ldots, at_q)$. Recall that

$$\sum_{i=1}^{\ell} \cdot \frac{ak^*_{\rho^*(i)}M^*_{i,j}}{\delta_{\phi(\rho^*(i))}} = l_1^*\delta_0 z_j \quad \forall j \in [1,m],$$

Consider any t_{k_0} such that it has a non-zero coefficient in l_1^*. For $i \in [1,\ell]$, construct v_i^* such that

$$v_i^* = \frac{k^*_{\rho^*(i)}}{\delta_0\delta_{\phi(\rho^*(i))} \cdot [t_{k_0}]l_1^*},$$

where the $[x]\pi$ notation denotes the coefficient of the term x in π. We see that v^* is a vector of constant coefficients which satisfies the equation $v^*M^* = [z_1, \ldots, z_m] = [1, 0, \ldots, 0]$. Further, in every position where $v_i^* \neq 0$, the set $\mathcal{A}_{k_0}$ surely contained the attribute $\rho^*(i)$. By the properties of the monotone span program, it must be the case that $\Gamma^*(\mathcal{A}_{k_0}) = 1$, and thus the adversary did not win the game.

Part 2: The simulator $\mathfrak{S}$ generates the public parameters for the zero-knowledge proof of knowledge PK_1, together with a knowledge extractor E. $\mathfrak{S}$ generates the rest of TPK honestly and runs $(\mathsf{APK}, \mathsf{ASK}) \leftarrow \mathsf{ASetup}(1^\lambda)$. $\mathfrak{S}$ gives $\mathfrak{A}$ the public

[4]The soundness of PK_1 is proven in Subsection IV-D.

information $\mathsf{PK} = (\mathsf{TPK}, \mathsf{APK})$.

The adversary $\mathfrak{A}$ wins by successfully authenticating for $M^* + 1$ times while using the value M^* as the upper bound during each authentication. By the soundness property of PK_1, $\mathfrak{S}$ can use the knowledge extractor E to extract

$$C_i = \hat{e}(g, g)^{\frac{1}{y+H(k_i)}},$$

for all $i \in [1, M^* + 1]$, where C_i, k_i is used for the i-th authentication. Observe that the values $k_i \in [1, M^*]$ for all i. It implies that at least two values of k_i are identical by the pigeonhole principle. It implies that their corresponding C_i are identical (by the soundness of PK_1). However, it contradicts that the service provider will reject the authentication for repeating C_i value. Therefore no polynomial time adversary can win this game if the PK_1 protocol is sound.

By combining part 1 and part 2, we have proved the theorem. ∎

Theorem 2: The above scheme is private if the PK_0 and PK_1 protocols are zero knowledge [5] and the decisional q_a-BDHI assumption holds, where q_a is the number of Auth Oracle query.

Proof: Observe that C is actually the output of the pseudo-random function (PRF) in [17] with input $H(k)$ and the key y. Suppose the adversary $\mathfrak{A}$ wins the privacy security game with the simulator $\mathfrak{S}$. Then we show that $\mathfrak{S}$ can break the pseudo-randomness property of C in [17] when interacting with its challenger $\mathfrak{B}$.

$\mathfrak{S}$ receives some public parameters (g, Y) of the PRF from $\mathfrak{B}$. Then $\mathfrak{S}$ honestly generates the rest of $\mathsf{TPK}, \mathsf{APK}$ and ASK; and then sends them to $\mathfrak{A}$.

For the oracle query $\mathcal{JO}$, $\mathfrak{S}$ honestly generates the key pair $(\mathsf{UPK}, \mathsf{USK})$, except for one time, it sets $\mathsf{UPK} = Y$ and $\mathsf{USK} = \perp$, which means that it does not know the corresponding secret key. If $\mathfrak{A}$ asks for the $\mathcal{CO}$ with input Y, $\mathfrak{S}$ declares failure and exits [6]. If $\mathfrak{A}$ asks for the $\mathcal{AO}$ with $\mathsf{UPK} = Y$, $\mathfrak{S}$ first finds the corresponding k value stored. $\mathfrak{S}$ asks the oracle from its PRF challenger $\mathfrak{B}$ with input $H(k)$. Then $\mathfrak{S}$ receives the value $C = \hat{e}(g, g)^{\frac{1}{y+H(k)}}$ from $\mathfrak{B}$. If $\mathfrak{S}$ has not obtained $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$ for the attribute set $\mathcal{A}$, then $\mathfrak{S}$ uses the simulator of the zero-knowledge proof PK_0 to generate a valid proof for AttrGen; and obtains the key from $\mathfrak{A}$. $\mathfrak{S}$ uses the value $C, \mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$ and the simulator of the zero-knowledge proof PK_1 to generate a valid proof for the authentication.

In the challenge phase, $\mathfrak{A}$ chooses two users UPK_0^* and UPK_1^*. If none of them equals to Y, then $\mathfrak{S}$ declares failure and exits. Otherwise, without loss of generality, suppose $\mathsf{UPK}_0^* = Y$ and the corresponding counter value as k^*. $\mathfrak{S}$ submits $H(k^*)$ to the PRF challenger $\mathfrak{B}$ and receives C^*, which may be equal to $\hat{e}(g, g)^{\frac{1}{y+H(k^*)}}$ or a random element in $\mathbb{G}_T$. After that, $\mathfrak{S}$ uses this C^* and the simulator of the zero-knowledge proof PK_1 to generate a valid proof.

Further oracle queries can be simulated as before. Finally, $\mathfrak{A}$ submits its guess b'. If $\mathfrak{A}$ wins, then $\mathfrak{S}$ guesses $C^* = \hat{e}(g, g)^{\frac{1}{y+H(k^*)}}$. Otherwise, $\mathfrak{S}$ guesses C^* is a random group element.

[5]The zero knowledge property of PK_0 and PK_1 is proven in Section IV-D.

[6]Note that $\mathfrak{A}$ cannot corrupt all user public keys.

By the zero knowledge property of the PK_0 protocol, the authority does not know about the key y during the AttrGen algorithm. By the zero knowledge property of the PK_1 protocol, it is easy to see that the protocol does not leak any information about $\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$. The only possible information that can be gathered by $\mathfrak{A}$ is from the linkability tag C^*. Observe that C^* is actually the output of the pseudo-random function in [17] with input $H(k)$ and the key y. Since the pseudo-randomness property of C in [17] holds if the decisional q_a-BDHI assumption holds, the privacy of our scheme also holds under the same assumption.

∎

F. Efficiency Analysis

We analyze the efficiency of our scheme for the Auth protocol, in terms of the monotone span program which is actually an $\ell \times m$ matrix. We count the number of exponentiation, multi-exponentiation ($\approx$ 1.25 exponentiation [23]) and pairing operation on both the user side and the server side based on the implementation of PK_1 in Section IV-D. The result is summarized in Table II.

TABLE II: Summary of computation requirements

	User	Server
exponentiation	8	m
multi-exponentiation	$6 + m$	$6 + m$
pairing	$1 + m$	$5 + (\ell + 1) \times m$

Finally, we consider the storage requirement for the service provider for checking purpose. Recall that the service provider only needs to check the value C during authentication. Therefore, it only needs to store a $\mathbb{G}_T$ element after each authentication, whose size depends on the security parameter. As an example, to achieve a 80-bit of security, the size of a $\mathbb{G}_T$ element is around 1024 bits. Hence, the extra storage is only about 1Gb about one million authentications. Similarly, the comparison of this 1Gb data should also be efficient on the cloud server.

G. Implementation

We implemented our scheme for the following cases: First, we assume there are N different groups of users who can access the system. For group i, where $i \in [1, N]$, there are P_i policies. For example, group 1 = ``Student'' AND ```Computer Science Dept''. That means it has 2 policies. That is, $P_1 = 2$. In addition, each user in group 1 can access the system for k_1 times. We simulate the system for different cases. In each case, for simplicity, we assume $P_1 = \ldots = P_N$. We also assume $k_1 = 1, \ldots, k_N = N$.

The general access policy can be expressed as

$$\begin{aligned}
&\Big((A_{1,1} \texttt{ AND } A_{1,2} \texttt{ AND } \ldots \texttt{ AND } A_{1,P_1}) \texttt{ AND } k \le k_1\Big)\\
\texttt{OR}\quad &\Big((A_{2,1} \texttt{ AND } A_{2,2} \texttt{ AND } \ldots \texttt{ AND } A_{2,P_2}) \texttt{ AND } k \le k_2\Big)\\
\texttt{OR}\quad &\quad \ldots\ldots\ldots\\
\texttt{OR}\quad &\Big((A_{N,1} \texttt{ AND } A_{N,2} \texttt{ AND } \ldots \texttt{ AND } A_{N,P_N}) \texttt{ AND } k \le k_N\Big)
\end{aligned}$$

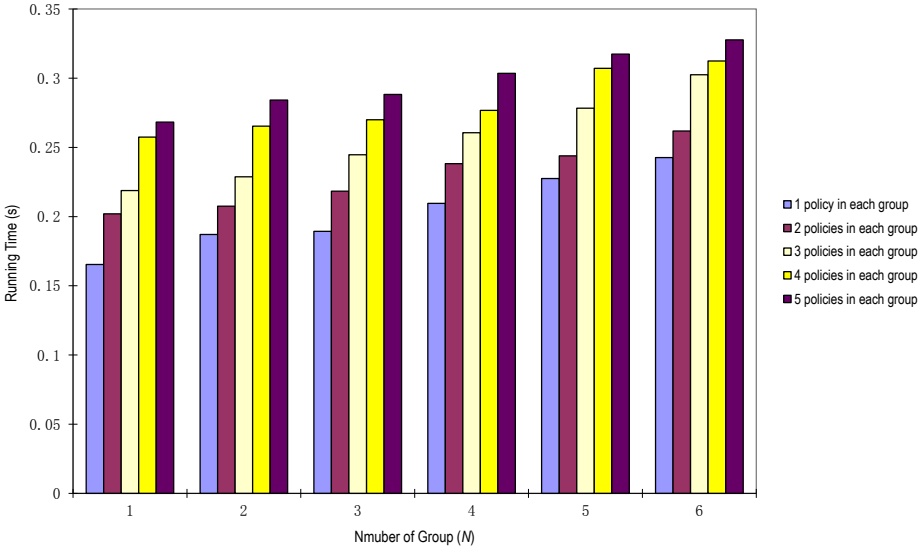

Fig. 2: Running time of the Auth protocol

where $A_{i,j}$ represents the j-th policy of group i; and k is the current number of access times for the user.

The running time result of Auth is plotted in Figure 2. The testing makes use of the Type-a pairing on the elliptic curve $Y^2 = X^3 + X$ over a 512-bit finite field with embedding degree 2 from the pairing-based library (version 0.5.12)[7].

The specification of the testing platform is summarized in Table III.

TABLE III: Testing Platform

OS	Ubuntu 10.10
CPU	Pentium(R) Dual-Core
Memory	2GB RAM
Hard disk	500G/5400rpm
Programming language	C

V. Extensions

In the last section, we have described the basic version of our system. In this section, we discuss some possible extensions to make it more practical.

A. Event Oriented Access Control

In the basic version, the user is linkable (and may be denied for access) if there is only one event or one claim-predicate. For example, suppose a student is allowed to access Matlab server two times a day. If a student who has accessed Matlab server two times yesterday (but none for today) should be allowed to access again today. However, in the basic version this cannot be achieved. If the student sets $k = 3$, the system will deny his/her authentication directly since $k > 2$. If he/she sets $k = 1$ (or 2), the system will find the same C in its database and thus deny his/her authentication. This is because the basic version *does not* support event oriented access control.

A simple solution to make it event oriented is to put the event description into the hash function H. For example, we can add

$$event = \texttt{``A student can access Matlab server 2 times on 1st Feb 2013.''}$$

into the hash. That is, we can set

$$C = \hat{e}(g, g)^{\frac{1}{y+H(k||event)}}$$

In this way, if the student wants to access Matlab on 2nd Feb, the hash output will be different even if $k = 1$ or $k = 2$.

The same logic can be applied to a different claim-predicate. For example, Alice is a student (who can access to Matlab 2 times a day) and she is also a part time staff (who can access to Matlab 3 times a day). Then she should be able to access Matlab 5 times a day. By using this event oriented solution, if Alice gains access using her student role, the event should be ``A student can access Matlab server 2 times on 1st Feb 2013.''. If she gains access using her staff role, then event should be ``A part time staff can access Matlab server 3

[7]http://crypto.stanford.edu/pbc/

`times on 1st Feb 2013.''`. Obviously the hash outputs of these two events are different. Therefore she will not be linked if she has not exceeded the maximum number of access times from the appropriate role.

B. An Option for Unlinkability

In the basic scheme, it allows the server to know the number of times a user has accessed (yet does not know the identity) as k is known to the server. Some servers may need this data for statistical purpose. For example, *today 10 users have accessed 8 times, 5 users have accessed 5 times and 13 users have accessed just once.* This is good in some scenarios or applications.

On the other side, sometimes unlinkability may be preferred as more privacy will be provided to users. In the basic scheme, if the number of authenticators is very few, it becomes linkable easily. Consider the following example. Alice and Bob are engineers in the university. They are allowed to access Matlab on the server 10 times a day. Assume there are only two engineers in the university. Today Alice has accessed once while Bob has not yet accessed the server. When Alice accesses the server on her 2nd time, although the server does not know it is Alice, it must know that it is from the same person who has previously accessed. It is because during her second access, $k = 2$ is sent to the server while for Bob's first access, $k = 1$. Even if there are many engineers, the server may also link a particular user easily. For example, Alice has accessed the server 8 times a day and Bob has accessed the server 5 times a day, while other users have accessed the server at most twice. If the next authenticator presents the information $k = 9$ to the server, it knows that this is the one who has accessed 8 times although it does not know it is Alice.

If unlinkability is preferred, we can require the user to choose a random fresh $k \in [1, M]$ (which has not been used before) instead of a sequential one. We use the above example to illustrate why by choosing a random k can make the system unlinkable. Since a random k is chosen every time, in Alice's 2nd time access, k can be any integer from 1 to 10 (instead of $k = 2$). Similarly, in Bob's 1st access, k can be any integer from 1 to 10 (instead of $k = 1$). Thus the server has no idea whether the access is from Bob or Alice. Note that even if Alice and Bob both choose the same k, the server only knows that they are from different users but still cannot link to any previous access.

There is also a trade-off for unlinkability. T he user needs to store every k he has used in previous authentication processes in this event while he only needs to store the previous k in the sequential version.

VI. Conclusion

In this paper, we coined a new cryptographic notion called *k-times attribute-based anonymous access control*, which is particularly designed for supporting cloud computing environment. This notion allows a user to authenticate himself/herself to the cloud computing server anonymously. Further, a k-times limit for anonymous access control is also provided. This means that the server may restrict a particular set of users to access the systems for a maximum k-times within a period or an event, meanwhile further access will be denied. We provided a security model as well as a concrete instantiation of this new notion. Our implementation result shows that our scheme is practical. Further, we provide the security proof for our instantiation. We also extended this notion to allow event oriented access control, as well as an option for unlinkability. These extensions are of independent interest.

References

[1] Amazon Elastic Compute Cloud (Amazon EC2) [online]. Available http://aws.amazon.com/ec2/.

[2] Amazon Web Services (AWS) [online]. Available https://s3.amazonaws.com/.

[3] Google App Engine [online]. Available http://code.google.com/appengine/.

[4] Yahoo Pig [online]. Available http://research.yahoo.com/node/90.

[5] Salesforce.com [online]. Available http://www.salesforce.com/.

[6] M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen. Secure id-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theor. Comput. Sci.*, 469:1–14, 2013.

[7] M. H. Au, W. Susilo, and Y. Mu. Constant-size dynamic *k*-taa. In *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006.

[8] K. Barlow and J. Lane. Like technology from an advanced alien culture: Google apps for education at ASU. In *SIGUCCS*, pages 8–10. ACM, 2007.

[9] A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[10] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.

[11] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[12] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

[13] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In T. Okamoto and X. Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer, 2007.

[14] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *ACM Conference on Computer and Communications Security*, pages 201–210. ACM, 2006.

[15] S. S. Chow, J. K. Liu, V. K. Wei, and T. H. Yuen. Ring Signatures without Random Oracles. In *ASIACCS 06*, pages 297–302. ACM Press, 2006.

[16] R. Cramer, I. Damgård, and P. D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 354–373. Springer, 2000.

[17] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, 2005.

[18] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[19] A. Lewko and B. Waters. Decentralizing attribute-based encryption, 2010. Cryptology ePrint Archive, Report 2010/351, 2010. http://eprint.iacr.org/.

[20] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *ACISP*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335. Springer, 2004.

[21] J. K. Liu and D. S. Wong. Enhanced security models and a generic construction approach for linkable ring signature. *Int. J. Found. Comput. Sci.*, 17(6):1403–1422, 2006.

[22] H. K. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. In *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2011.

[23] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. Handbook of applied cryptography. CRC Press LLC, 1997.

[24] L. Nguyen. Efficient dynamic *k*-times anonymous authentication. In *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 81–98. Springer, 2006.

[25] L. Nguyen and R. Safavi-Naini. Dynamic k-times anonymous authentication. In *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 318–333. Springer, 2005.

[26] T. Okamoto and K. Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 35–52. Springer, 2011.

[27] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203. ACM, 2007.

[28] R. L. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.

[29] S. F. Shahandashti and R. Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. In *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 198–216. Springer, 2009.

[30] I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication (extended abstract). In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2004.

[31] I. Teranishi and K. Sako. *k*-times anonymous authentication with a constant proving cost. In *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2006.

[32] C. Wachsmann, L. Chen, K. Dietrich, H. Löhr, A.-R. Sadeghi, and J. Winter. Lightweight anonymous authentication with tls and daa for embedded mobile devices. In *ISC*, volume 6531 of *Lecture Notes in Computer Science*, pages 84–98. Springer, 2010.

[33] 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*, 7(2):743–754, 2012.