Fine-Grained Two-Factor Access Control for Web-Based Cloud Computing Services

Joseph K. Liu, *Member, IEEE*, Man Ho Au, *Member, IEEE*, Xinyi Huang, Rongxing Lu, *Senior Member, IEEE*, and Jin Li

***Abstract*— In this paper, we introduce a new fine-grained two-factor authentication (2FA) access control system for web-based cloud computing services. Specifically, in our proposed 2FA access control system, an attribute-based access control mechanism is implemented with the necessity of both a user secret key and a lightweight security device. As a user cannot access the system if they do not hold both, the mechanism can enhance the security of the system, especially in those scenarios where many users share the same computer for web-based cloud services. In addition, attribute-based control in the system also enables the cloud server to restrict the access to those users with the same set of attributes while preserving user privacy, i.e., the cloud server only knows that the user fulfills the required predicate, but has no idea on the exact identity of the user. Finally, we also carry out a simulation to demonstrate the practicability of our proposed 2FA system.**

***Index Terms*— Fine-grained, two-factor, access control, Web services.**

I. Introduction

CLOUD COMPUTING is a virtual host computer system that enables enterprises to buy, lease, sell, or distribute software and other digital resources over the internet as an on-demand service. It no longer depends on a server or a number of machines that physically exist, as it is a *virtual* system. There are many applications of cloud computing, such as data sharing [22], [30], [31], [33], data storage [15], [25], [32], [45], big data management [4], medical information system [44] etc. End users access cloud-based applications through a web browser, thin client or mobile app while the business software and user's data are stored on servers at a remote location. The benefits of web-based cloud computing services are huge, which include the ease of accessibility, reduced costs and capital expenditures, increased operational efficiencies, scalability, flexibility and immediate time to market.

Though the new paradigm of cloud computing provides great advantages, there are meanwhile also concerns about security and privacy especially for web-based cloud services. As sensitive data may be stored in the cloud for sharing purpose or convenient access; and eligible users may also access the cloud system for various applications and services, user authentication has become a critical component for any cloud system. A user is required to login before using the cloud services or accessing the sensitive data stored in the cloud. There are two problems for the traditional account/password-based system. First, the traditional account/password-based authentication is not privacy-preserving. However, it is well acknowledged that privacy is an essential feature that must be considered in cloud computing systems. Second, it is common to share a computer among different people. It maybe easy for hackers to install some spyware to learn the login password from the web-browser. A recently proposed access control model called *attribute-based access control* is a good candidate to tackle the first problem. It not only provides anonymous authentication but also further defines access control policies based on different attributes of the requester, environment, or the data object. In an attribute-based access control system,[1] each user has a user secret key issued by the authority. In practice, the user secret key is stored inside the personal computer. When we consider the above mentioned second problem on web-based services, it is common that computers may be shared by many users especially in some large enterprises or organizations. For example, let us consider the following two scenarios:

- In a hospital, computers are shared by different staff. Dr. Alice uses the computer in room A when she is on duty in the daytime, while Dr. Bob uses the same computer in the same room when he is on duty at night.
- In a university, computers in the undergraduate lab are usually shared by different students.

Manuscript received March 10, 2015; revised July 31, 2015 and September 20, 2015; accepted September 29, 2015. Date of publication October 26, 2015; date of current version December 24, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61472083, Grant U1405255, and Grant 61402110, in part by the Fok Ying Tung Education Foundation under Grant 141065, in part by the Program for New Century Excellent Talents in Fujian University under Grant JA14067, in part by the Distinguished Young Scholars Fund of Fujian Province, China, in part by the State Key Laboratory of Cryptology Research Fund, China, and in part by the Natural Science Foundation of Guangdong Province for Distinguished Young Scholars under Grant 2014A030306020. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Shouhuai Xu. *(Corresponding author: Man Ho Au.)*

J. K. Liu is with the Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia (e-mail: joseph.liu@monash.edu).

M. H. Au is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csallen@comp.polyu.edu.hk).

X. Huang is with the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350108, China, and also with the State Key Laboratory of Cryptology, Beijing 100878, China (e-mail: xyhuang81@gmail.com).

R. Lu is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: rxlu@ntu.edu.sg).

J. Li is with the Department of Computer Science, Guangzhou University, Guangzhou 510006, China (e-mail: jinli71@gmail.com).

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/TIFS.2015.2493983

[1]More details will be given in Section II-A.

In these cases, user secret keys could be easily stolen or used by an unauthorized party. Even though the computer may be locked by a password, it can still be possibly guessed or stolen by undetected malwares.

A more secure way is to use two-factor authentication (2FA). 2FA is very common among web-based e-banking services. In addition to a username/password, the user is also required to have a device to display a one-time password. Some systems may require the user to have a mobile phone while the one-time password will be sent to the mobile phone through SMS during the login process. By using 2FA, users will have more confidence to use shared computers to login for web-based e-banking services. For the same reason, it will be better to have a 2FA system for users in the web-based cloud services in order to increase the security level in the system.

A. Our Contribution

In this paper, we propose a fine-grained two-factor access control protocol for web-based cloud computing services, using a lightweight security device. The device has the following properties: (1) it can compute some lightweight algorithms, e.g. hashing and exponentiation; and (2) it is tamper resistant, i.e., it is assumed that no one can break into it to get the secret information stored inside.

With this device, our protocol provides a 2FA security. First the user secret key (which is usually stored inside the computer) is required. In addition, the security device should be also connected to the computer (e.g. through USB) in order to authenticate the user for accessing the cloud. The user can be granted access only if he has both items. Furthermore, the user cannot use his secret key with another device belonging to others for the access.

Our protocol supports fine-grained attribute-based access which provides a great flexibility for the system to set different access policies according to different scenarios. At the same time, the privacy of the user is also preserved. The cloud system only knows that the user possesses some required attribute, but not the real identity of the user.

To show the practicality of our system, we simulate the prototype of the protocol.

In the next section, we will review some related works that are related to our concept.

II. Related Works

We review some related works including attribute-based cryptosystems and access control with security device in this section.

A. Attribute-Based Cryptosystem

Attribute-based encryption (ABE) [20], [39] is the cornerstone of attribute-based cryptosystem. ABE enables fine-grained access control over encrypted data using access policies and associates attributes with private keys and ciphertexts. Within this context, ciphertext-policy ABE (CP-ABE) [6] allows a scalable way of data encryption such that the encryptor defines the access policy that the decryptor (and his/her attributes set) needs to satisfy to decrypt the ciphertext. Thus, different users are allowed to decrypt different pieces of data with respect to the pre-defined policy. This can eliminate the trust on the storage server to prevent unauthorised data access.

Besides dealing with authenticated access on encrypted data in cloud storage service [21], [23], [24], [27]–[29], [36], [42], [43], ABE can also be used for access control to cloud computing service, in a similar way as an encryption scheme can be used for authentication purpose: The cloud server may encrypt a random message using the access policy and ask the user to decrypt. If the user can successfully decrypt the ciphertext (which means the user's attributes set satisfies the prescribed policy), then it is allowed to access the cloud computing service.

In addition to ABE, another cryptographic primitive in attribute-based cryptosystem is attribute-based signature (ABS) [35], [38], [41]. An ABS scheme enables a user to sign a message with fine-grained control over identifying information. Specifically, in an ABS scheme, users obtain their attribute private keys from an attribute authority. Then they can later sign messages for any predicate satisfied by their attributes. A verifier will be convinced of the fact that the signer's attributes satisfy the signing predicate if the signature is valid. At the same time, the identity of signer remains hidden. Thus it can achieve anonymous attribute-based access control efficiently. Recently, Yuen *et al.* [47] proposed an attribute-based access control mechanism which can be regarded as the interactive form of ABS.

B. Access Control With Security Device

1) Security Mediated Cryptosystem: Mediated cryptography was first introduced in [8] as a method to allow immediate revocation of public keys. The basic idea of mediated cryptography is to use an on-line mediator for every transaction. This on-line mediator is referred to a SEM (SEcurity Mediator) since it provides a control of security capabilities. If the SEM does not cooperate then no transactions with the public key are possible any longer. Recently, an attribute-based version of SEM was proposed in [13].

The notion of SEM cryptography was further modified as security mediated certificateless (SMC) cryptography [14], [46]. In a SMC system, a user has a secret key, public key and an identity. In the signing or decryption algorithm, it requires the secret key and the SEM together. In the signature verification or encryption algorithm, it requires the user public key and the corresponding identity. Since the SEM is controlled by an authority which is used to handle user revocation, the authority refuses to provide any cooperation for any revoked user. Thus revoked users cannot generate signature or decrypt ciphertext.

Note that SMC is different from our concept. The main purpose of SMC is to solve the revocation problem. Thus the SME is controlled by the authority. In other words, the authority needs to be *online* for every signature signing and ciphertext decryption. The user is not anonymous in SMC. While in our system, the security device is controlled by the user. Anonymity is also preserved.

2) Key-Insulated Cryptosystem: The paradigm of key-insulated cryptography was introduced in [17]. The general idea of key-insulated security was to store long-term keys in a physically-secure but computationally-limited device. Short-term secret keys are kept by users on a powerful but insecure device where cryptographic computations take place. Short term secrets are then refreshed at discrete time periods via interaction between the user and the base while the public key remains unchanged throughout the lifetime of the system. At the beginning of each time period, the user obtains a partial secret key from the device. By combining this partial secret key with the secret key for the previous period, the user renews the secret key for the current time period.

Different from our concept, key-insulated cryptosystem requires all users to update their keys in every time period. The key update process requires the security device. Once the key has been updated, the signing or decryption algorithm does *not* require the device anymore within the same time period. While our concept *does* require the security device every time the user tries to access the system. Furthermore, there is no key updating required in our system.

III. Preliminaries

In this section, we introduce the notations deployed in our scheme.

A. Pairings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of prime order p. A map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ 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.

B. Monotone Span Program

Our access control mechanism depends on expressing the attribute predicate as a monotonespan program. We review some notation about monotone span program using the notation in [35]. Let $\Upsilon : \{0, 1\}^n \rightarrow \{0, 1\}$ 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 $\rho : [1, \ell] \rightarrow [1, n]$ 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 does have compact monotone span programs. Given a monotone boolean function Υ, one can use the method given in [20] to obtain the matrix $\mathbf{M}$.

C. BBS+ Signatures

We briefly review a signature scheme called BBS+. It belongs to a class of signature schemes, commonly known as CL-signatures [11]. CL-signatures are useful in certifying credentials since their structures allows (1) a signer to create a signature on committed values; and (2) a signer holder to prove to any third party that he/ she is in possession of a signature from the signer in zero knowledge. BBS+ is proposed by Au et al. [3], which is based on the schemes of Camenisch and Lysyanskaya [12] and of Boneh et al. [7]. It is also referred to as credential signatures [2] as it is normally used to certify a set of credentials [1], [10], [34].

Let $(p, \mathbb{G}, \mathbb{G}, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$ be the public parameters as discussed. In addition, let $\hat{g}, h, h_0, h_1, \ldots, h_n \in \mathbb{G}$ be publicly known generators of $\mathbb{G}$.

The signer's secret key is $\gamma \in_R \mathbb{Z}_p$ and the public key is $w = h^\gamma$.

To sign a message block $(x_0, x_1, \ldots, x_n) \in \mathbb{Z}_p^{n+1}$, the signer randomly picks $e, s \in_R \mathbb{Z}_p$, computes $A = (h h_0^{x_0} h_1^{x_1} \cdots h_n^{x_n} \hat{g}^s)^{\frac{1}{\gamma+e}}$. The signer outputs (A, e, s) as the signature on the block of messages $(x_0, \ldots, x_n)$.

To verify a BBS+ signature, one can test if the following equation holds.

$$\hat{e}(A, w h^e) = \hat{e}(h h_0^{x_0} h_1^{x_1} \cdots h_n^{x_n} \hat{g}^s, h)$$

BBS+ is existentially unforgeable against adaptive chosen message attack under the q-SDH assumption.

IV. Overview

A. Intuition

A naive thinking to achieve our goal is to use a normal ABS and simply split the user secret key into two parts. One part is kept by the user (stored in the computer) while another part is initialized into the security device. Special care must be taken in the process since normal ABS does not guarantee that the leakage of part of the secret key does not affect the security of the scheme while in two 2FA, the attacker could have compromised one of the factors. Besides, the splitting should be done in such a way that most of the computation load should be with the user's computer since the security device is not supposed to be powerful.

We specifically design our system in another manner. We do not split the secret key into two parts. Instead, we introduce some additional unique information stored in the security device. The authentication process requires this piece of information together with the user secret key. It is guaranteed that missing either part cannot let the authentication pass. There is also a linking relationship between the user's device and the secret key so that the user cannot use another user's device for the authentication. The communication overhead is minimal and the computation required in the device is just some lightweight algorithms such as hashing or exponentiation over group $\mathbb{G}_T$.[2] All the heavy computations such as pairing are done on the computer.

The idea of our system is illustrated in Figure 1.

[2]The exponentiation done in group $\mathbb{G}_T$ is much lighter than those in group $\mathbb{G}$. It can be seen from the simulation data in the benckmark.

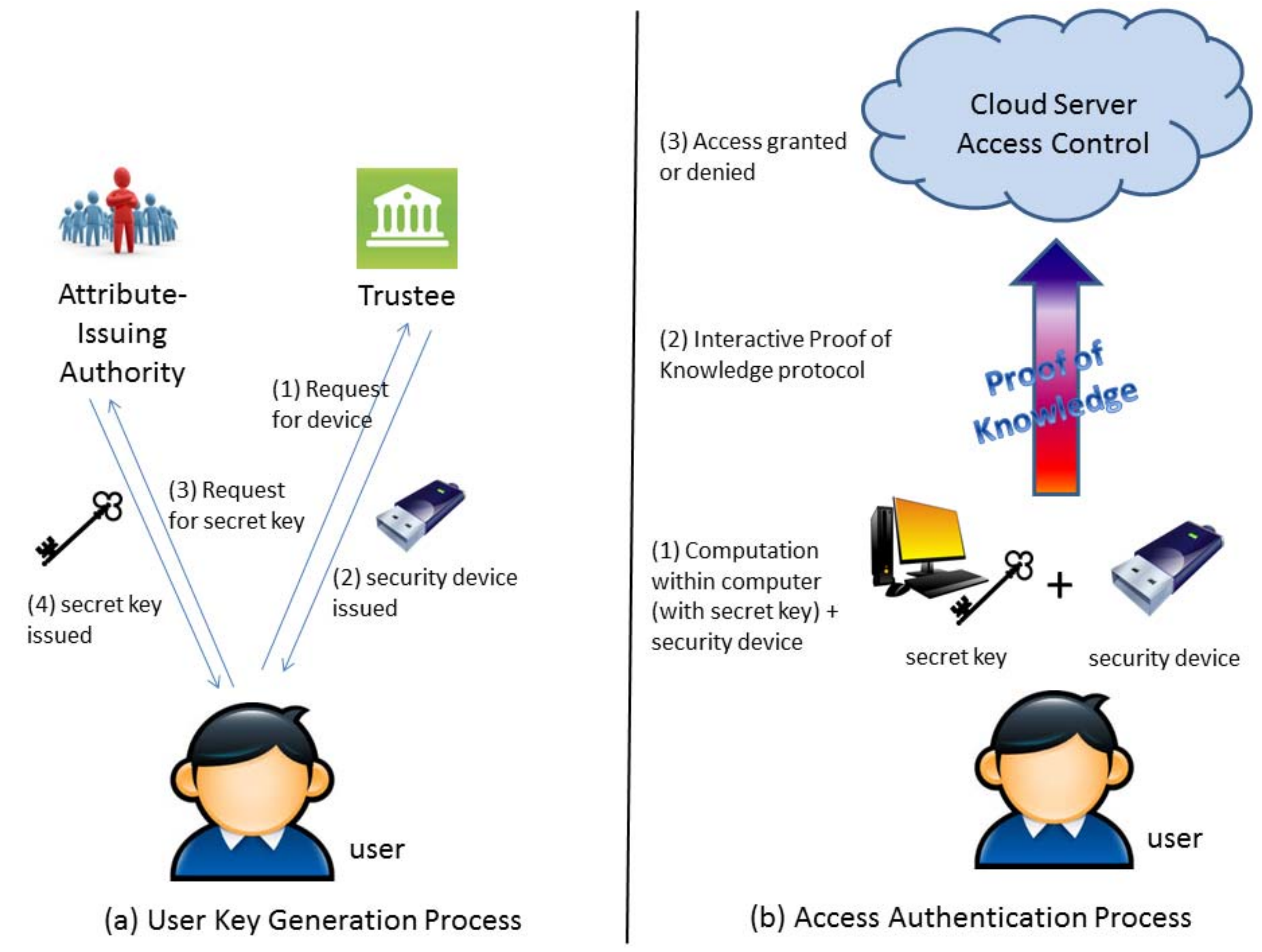

Attribute-Issuing Authority
Trustee
(1) Request for device
(2) security device issued
(3) Request for secret key
(4) secret key issued
user
(a) User Key Generation Process
Cloud Server Access Control
(3) Access granted or denied
(2) Interactive Proof of Knowledge protocol
Proof of Knowledge
(1) Computation within computer (with secret key) + security device
secret key
+
security device
user
(b) Access Authentication Process

Fig. 1. Overview idea of our system.

B. Entities

Our system consists of the following entities:

- Trustee: It is responsible for generating all system parameters and initialise the security device.
- Attribute-issuing Authority: It is responsible to generate user secret key for each user according to their attributes.
- User: It is the player that makes authentication with the cloud server. Each user has a secret key issued by the attribute-issuing authority and a security device initialized by the trustee.
- Cloud Service Provider: It provides services to anonymous authorised users. It interacts with the user during the authentication process.

C. Assumptions

The focus of this paper is on preventing private information leakage at the phase of access authentication. Thus we make some assumptions on system setup and communication channels. We assume each user communicates with the cloud service provider through an anonymous channel [26], [37] or uses IP-hiding technology. We also assume that trustee generates the security parameters according to the algorithm prescribed. Other potential attacks, such as IP hijacking, distributed denial-of-service attack, man-in-the-middle attack, etc., are out of the scope of this paper.

D. Threat Model

In this paper, we consider the following threats:

1) Authentication: The adversary tries to access the system beyond its privileges. For example, a user with attributes {`Student`, `Physics`} may try to access the system with policy "`Staff`" AND "`Physics`". To do so, he may collude with other users.
2) Access without Security Device: The adversary tries to access the system (within its privileges) without the security device, or using another security device belonging to others.
3) Access without Secret Key: The adversary tries to access the system (within its privileges) without any secret key. It can have its own security device.
4) Privacy: The adversary acts as the role of the cloud server and tries to find out the identity of the user it is interacting with.

TABLE I
FREQUENTLY USED NOTATIONS

TPK	public parameters of the trustee
APK	public key of the attribute-issuing authority
ASK	secret key of the attribute-issuing authority
UPK	public key of the user
USK	partial secret key of the user
$\mathsf{sk}_{\mathcal{A},\mathsf{UPK}}$	attribute secret key for the user with attribute $\mathcal{A}$ and public key UPK
Υ	claim-predicate for the access control

E. Notation

Frequently used notations in our system are summarized in Table I.

V. Our Proposed System

A. Specification of the Security Device

We assume the security device employed in our system satisfies the following requirements.

1) *Tamper-resistance.* The content stored inside the security device is not accessible nor modifiable once it is initialized. In addition, it will always follow the algorithm specification.
2) *Capability.* It is capable of evaluation of a hash function. In addition, it can generate random numbers and compute exponentiations of a cyclic group defined over a finite field.

B. Construction

Let $\mathbb{A}$ be the desired universe of attributes. For simplicity, we assume $\mathbb{A} = [1, n]$ for some natural number n. We will use a vector $\vec{x} \in \{0, 1\}^n$ to represent the user's attribute set. Let $\vec{x} = (x_1, \dots, x_n) \in \{0, 1\}^n$. If the user is in possession of attribute i, $x_i = 1$. Otherwise, $x_i = 0$.

1) System Setup: The system setup process consists of two parts. The first part TSetup is run by a trustee to generate public parameters. The second part ASetup is run by the attribute-issuing authority to generate its master secret key and public key.

<u>TSetup</u>: Let λ be a security parameter. The trustee runs $\mathcal{G}(1^\lambda)$ (described in Section III-A) to generate $\mathsf{param} = (\mathbb{G}, \mathbb{G}_T, p, \hat{e})$ and randomly picks generators $g, \hat{g}, h, h_0, h_1, \dots, h_n \in \mathbb{G}$. It also picks a collision resistant hash function $H : \{0, 1\}^* \to \mathbb{Z}_p$. Further, let $\mathsf{tpk} = \hat{e}(g, h_0)^{\mathsf{tsk}}$ for a randomly generated $\mathsf{tsk} \in_R \mathbb{Z}_p$.

It publishes $\mathsf{TPK} = (\mathsf{param}, g, \hat{g}, h, h_0, h_1, \dots, h_n, H, \mathsf{tpk})$.

<u>ASetup</u>: The attribute-issuing authority randomly picks $\gamma \in \mathbb{Z}_p$ and computes $w = h^\gamma$. It publishes $\mathsf{APK} = (w)$ and sets $\mathsf{ASK} = (\gamma)$.

2) User Key Generation: The user key generation process consists of three parts. First, the user generates his secret and public key in USetup. Then the security device is initialized by the trustee in Device Initialization. Finally the attribute-issuing authority generates the user attribute secret key according to the user's attribute in AttrGen.

<u>USetup</u>: The user randomly picks $y \in \mathbb{Z}_p$. It publishes $\mathsf{UPK} = Y = h_0^y$ and sets $\mathsf{USK} = y$.

<u>Device Initialization</u>: The trustee initializes the security device for user (whose public key is UPK) with values $\mathsf{TY} = \hat{e}(g, Y)$, $\mathsf{TG} = \hat{e}(g, h_0)$ and tsk.

<u>AttrGen</u>: The key generation algorithm takes as input TPK, APK, $\mathsf{UPK} = Y$ and an attribute set $\mathcal{A}$ represented as a by a vector $(x_1, \dots, x_n) \in \{0, 1\}^n$.

The user runs a zero-knowledge proof of knowledge protocol PK_0 with the attribute-issuing authority to prove the knowledge of his partial secret key y:

$$PK_0\{y : Y = h_0^y\}.$$

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 chooses random $e, s \in \mathbb{Z}_p$ and uses his secret key ASK to create the user attribute secret key $\mathsf{sk}_{\mathcal{A},Y} := (A, e, s)$ as

$$A = (hYh_1^{x_1} \cdots h_n^{x_n} \hat{g}^s)^{\frac{1}{\gamma+e}}$$

3) Access Authentication: The access authentication process is an interactive protocol between the user and the cloud service provider. It requires the user to have his partial secret key, attribute secret key[3] and the security device.

<u>Auth</u>: The interactive authentication protocol takes as input TPK, APK and a claim-predicate Υ. The user has some additional inputs including an attribute secret key $\mathsf{sk}_{\mathcal{A},Y}$ for attribute $\mathcal{A}$, $\mathsf{USK} = y$ and the security device. Assume $\Upsilon(\mathcal{A}) = 1$. Parse $\mathsf{sk}_{\mathcal{A},Y}$ as $(A, e, s, \vec{x})$.

1) The authentication server picks at random a challenge $R \in \mathbb{Z}_p$ and sends R to the user.
2) The user computes $C = \hat{e}(g, h_0)^{\frac{1}{y+R}}$ and submits (C, y, R) to his/her security device.
3) The security device validates $C^{(y+R)} = \mathsf{TG}$ and $\mathsf{TG}^y = \mathsf{TY}$.
4) Upon successful validation, the security device picks a random $r \in_R \mathbb{Z}_p$, computes $c_R = H(\mathsf{TG}^r || R || C)$ and $z_R = r - c_R\mathsf{tsk}$. It returns (c_R, z_R) to the user.
5) The user converts 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 $\vec{v} = (v_1, \dots, v_\ell) \in \mathbb{Z}_p^\ell$ that corresponds to the satisfying assignment $\mathcal{A}$. That is $\vec{v}\mathbf{M} = (1, 0, \dots, 0)$. Note that if $x_{\rho(i)} = 0$ (i.e., the user does not possess the attribute $\rho(i)$), v_i must be 0).
6) For $i = 1$ to ℓ, the user randomly picks $a_i, t_i \in_R \mathbb{Z}_p$ and computes $C_i = g^{v_i} h^{t_i}$, $D_i = g^{x_{\rho(i)}} h^{a_i}$. The user also computes $b_i = t_i - a_i v_i$.
7) For $j = 1$ to m, the user computes $f_j = \sum_{i=1}^{\ell} t_i M_{i,j}$. Then the user sends $(C, c_R, z_R, C_1, \dots, C_\ell, D_1, \dots, D_\ell)$ to the authentication server.

[3]We assume the user stores both the partial secret key and attribute secret key in his/her computer.

8) They then engage in the following zero-knowledge proof-of-knowledge protocol.

$$\mathcal{PK}_1 \left\{ \begin{array}{l} \begin{pmatrix} A, e, s, y, \{x_i\}_{i=1}^n, \{a_i\}_{i=1}^\ell, \{b_i\}_{i=1}^\ell, \\ \{v_i\}_{i=1}^\ell, \{t_i\}_{i=1}^\ell, \{f_i\}_{i=1}^m \end{pmatrix}: \\ \qquad \hat{e}(A, wh^e) = \hat{e}(hh_0^y h_1^{x_1} \ldots h_n^{x_n} \hat{g}^s, h) \wedge \\ \hat{e}(g, h_0)C^{-R} = C^y \wedge \\ \qquad \bigwedge_{i=1}^{\ell} \left(D_i = g^{x_{\rho(i)}} h^{a_i}\right) \wedge \\ \qquad \bigwedge_{i=1}^{\ell} \left(C_i = g^{v_i} h^{t_i}\right) \wedge \\ \qquad \bigwedge_{i=1}^{\ell} \left(C_i = D_i^{v_i} h^{b_i}\right) \wedge \\ (\prod_{i=1}^{\ell} C_i^{M_{i,1}})/g = h^{f_1} \wedge \\ \bigwedge_{j=2}^{m} (\prod_{i=1}^{\ell} C_i^{M_{i,j}} = h^{f_j}) \end{array} \right\}$$

9) The authentication server validates PK_1 and that $c_R = H(\mathsf{tpk}^{c_R} \hat{e}(g, h_0)^{z_R} || R || C)$.

To better illustrate the idea of or protocol, a running example is presented in the Appendix.

The details of the proof of knowledge PK_0 and PK_1 instantiation will be described in the following subsection.

C. Proof of Knowledge

We briefly introduce the proof of knowledge as defined in [5]. Intuitively, a two-party protocol constitutes a system for proofs of knowledge if one party (called the verifier) is convinced that 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.

1) Implementation of Protocol PK_0:

$$PK_0\{y : Y = h_0^y\}.$$

Suppose Alice wants to prove the knowledge of y to Bob. Alice picks a random number $r \in \mathbb{Z}_p$ and sends the commitment $R = h_0^r$ to Bob. Bob returns a random challenge $c \in \mathbb{Z}_p$. Alice computes the response $z = r + cy$. Bob verifies that $h_0^z = R \cdot Y^c$. Details of the protocol can be found in [9, Ch. 3]. For completeness, we briefly outline how PK_0 provides soundess and zero-knowledgeness here. *(Soundness)* Suppose the simulator is given a discrete logarithm instance (h_0, 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

$$h_0^z Y^{-c} = h_0^{z'} Y^{-c'}.$$

Hence the simulator can obtain $\frac{z-z'}{c-c'}$ as the solution of $\log_{h_0} Y$. *(Zero-knowledge)* Given the public key h_0, Y, the simulator can randomly picks $c, z \in \mathbb{Z}_p$ and computes $R = h_0^z Y^{-c}$. The transcript (R, c, z) has the same distribution as those coming from Alice and Bob.

2) Implementation of Protocol PK_1: Before discussing PK_1, it is useful to describe the goal of PK_1. The set $\{C_i\}_{i=1}^\ell$ is the commitment of the vector $\vec{v}$ such that $\vec{v}\mathbf{M} = (1, 0, \ldots, 0)$. In other words, the goal of PK_1 is to ensure the authenticating user is in possession of a set of attributes that satisfies the monotone boolean function. The first challenge is to ensure the user can only set v_i to be non-zero if he is in possession of attribute $\rho(i)$. This is done by having his attributes certified with the BBS+ signature. More formally, the user attribute key (A, e, s) is a BBS+ signature on the tuple $(y, x_1, \ldots, x_n)$. To ensure $v_i \neq 0$ if and only if $x_{\rho(i)} = 1$, PK_1 requires the user to demonstrate several relationships. First of all, the user has to commit the relevant x_i, which results in D_i. Next, the user proves that both C_i and D_i are correctly computed as in $D_i = g^{x_{\rho(i)}} h^{a_i}$ and $C_i = g^{v_i} h^{t_i}$. The final relation $C_i = D_i^{v_i} h^{b_i}$ is crucial, as it ensures that v_i equals $x_i v_i$. That is, if x_i is 0, v_i must be zero. Finally, the relation

$$\hat{e}(A, wh^e) = \hat{e}(hh_0^y h_1^{x_1} \ldots h_n^{x_n} \hat{g}^s, h)$$

ensures the set of x_i together with the user secret key y has been signed (the corresponding signature is (A, e, s)), which means the set of attributes used is properly certified. We shall elaborate how this could be conducted based on the signature verification protocol of BBS+.

The final two relations mean that $\vec{v}\mathbf{M}$ evaluates to $(1, 0, \ldots, 0)$.

Now we are ready to describe the implementation of PK_1. The prover first randomly generates $k_1, k_2 \in_R \mathbb{Z}_p$, computes $\mathfrak{A}_1 = g^{k_1} h^{k_2}$, $\mathfrak{A}_2 = Ah^{k_1}$, $\beta_1 = k_1 e$, $\beta_2 = k_2 e$, and conducts the following proof.

$$\mathcal{PK}'_1\left\{\begin{array}{l}\begin{pmatrix} e, s, y, \{x_i\}_{i=1}^{n}, \{a_i\}_{i=1}^{\ell}, \{b_i\}_{i=1}^{\ell}, \\ \{v_i\}_{i=1}^{\ell}, \{t_i\}_{i=1}^{\ell}, \{f_i\}_{i=1}^{m}, k_1, k_2, \beta_1, \beta_2 \end{pmatrix}: \\ \qquad \mathfrak{A}_1 = g^{k_1} h^{k_2} \wedge \\ \qquad 1 = \mathfrak{A}_1^{-e} g^{\beta_1} h^{\beta_2} \wedge \\ \qquad \dfrac{\hat{e}(\mathfrak{A}_2, w)}{\hat{e}(h,h)} = \hat{e}(h_0,h)^{y}\hat{e}(h_1,h)^{x_1}\cdots\hat{e}(h_n,h)^{x_n} \\ \qquad\qquad \hat{e}(\hat{g},h)^{s}\hat{e}(h,w)^{k_1} \\ \qquad\qquad \hat{e}(h,h)^{\beta_1}/\hat{e}(\mathfrak{A}_2,h)^{e} \wedge \\ \hat{e}(g,h_0)C^{-R} = C^{y} \wedge \\ \bigwedge_{i=1}^{\ell} \left(D_i = g^{x_{\rho(i)}} h^{a_i}\right) \wedge \\ \bigwedge_{i=1}^{\ell} \left(C_i = g^{v_i} h^{t_i}\right) \wedge \\ \bigwedge_{i=1}^{\ell} \left(C_i = D_i^{v_i} h^{b_i}\right) \wedge \\ (\prod_{i=1}^{\ell} C_i^{M_{i,1}})/g = h^{f_1} \wedge \\ \bigwedge_{j=2}^{m} (\prod_{i=1}^{\ell} C_i^{M_{i,j}} = h^{f_j}) \end{array}\right\}$$

$\mathcal{PK}'_1$ is a standard zero-knowledge proof-of-knowledge which instantiates the idea of $\mathcal{PK}_1$.

a) Complete specification of $\mathcal{PK}'_1$: For completeness, we describe the honest-verifier zero-knowledge version of $\mathcal{PK}'_1$ 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_e, ρ_s, ρ_y, ρ_{x_1}, …, ρ_{x_n}, ρ_{a_1}, …, ρ_{a_ℓ}, ρ_{b_1}, …, ρ_{b_ℓ}, ρ_{v_1}, …, ρ_{v_ℓ}, ρ_{t_1}, …, ρ_{t_ℓ}, ρ_{f_1}, …, ρ_{f_m}, ρ_{k_1}, ρ_{k_2}, ρ_{β_1}, ρ_{β_2} and computes the following $(3\ell + m + 4)$ values, usually referred to as commitments.

$$\begin{aligned} T_1 &= g^{\rho_{k_1}} h^{\rho_{k_2}}, \\ T_2 &= \mathfrak{A}_1^{-\rho_e} g^{\rho_{\beta_1}} h^{\rho_{\beta_2}}, \\ T_3 &= \hat{e}(h_0,h)^{\rho_y}\hat{e}(h_1,h)^{\rho_{x_1}}\cdots\hat{e}(h_n,h)^{\rho_{x_n}} \cdot \\ &\qquad \hat{e}(\hat{g},h)^{\rho_s}\hat{e}(h,w)^{\rho_{k_1}}\hat{e}(h,h)^{\rho_{\beta_1}}\hat{e}(\mathfrak{A}_2,h)^{-\rho_e}, \\ T_4 &= C^{\rho_y}, \\ T_{5,i} &= g^{\rho_{x_{\rho(i)}}} h^{\rho_{a_i}}, \quad \text{for } i = 1 \text{ to } \ell \\ T_{6,i} &= g^{\rho_{v_i}} h^{\rho_{t_i}}, \quad \text{for } i = 1 \text{ to } \ell \\ T_{7,i} &= D_i^{\rho_{v_i}} h^{\rho_{b_i}}, \quad \text{for } i = 1 \text{ to } \ell \\ T_8 &= h^{\rho_{f_1}}, \\ T_{9,j} &= h^{\rho_{f_j}} \quad \text{for } j = 2 \text{ to } m. \end{aligned}$$

Alice sends $(T_1, T_2, T_3, T_4, \{T_{5,i}\}_{i=1}^{\ell}, \{T_{6,i}\}_{i=1}^{\ell}, \{T_{7,i}\}_{i=1}^{\ell}, T_8, \{T_{9,j}\}_{j=2}^{m})$ to Bob. In practice, all pairing operations with known public parameters can be pre-computed. The only pairing operation that needs to be computed online is $\hat{e}(\mathfrak{A}_2, h)$.

- *Challenge Phase*: Bob randomly picks $c \in_R \mathbb{Z}_p$ and sends c to Alice.
- *Response Phase*: Alice computes the following $(n + 4\ell + m + 7)$ values, usually referred to as responses.

$$\begin{aligned} z_e &= \rho_e - ce, \\ z_s &= \rho_s - cs, \\ z_y &= \rho_y - cy, \\ z_{x_i} &= \rho_{x_i} - cx_i \quad \text{for } i = 1 \text{ to } n, \\ z_{a_i} &= \rho_{a_i} - ca_i \quad \text{for } i = 1 \text{ to } \ell, \\ z_{b_i} &= \rho_{b_i} - cb_i \quad \text{for } i = 1 \text{ to } \ell, \\ z_{v_i} &= \rho_{v_i} - cv_i \quad \text{for } i = 1 \text{ to } \ell, \\ z_{t_i} &= \rho_{t_i} - ct_i \quad \text{for } i = 1 \text{ to } \ell, \\ z_{f_j} &= \rho_{f_j} - cf_j \quad \text{for } j = 1 \text{ to } m, \\ z_{k_1} &= \rho_{k_1} - ck_1, \\ z_{k_2} &= \rho_{k_2} - ck_2, \\ z_{\beta_1} &= \rho_{\beta_1} - c\beta_1, \\ z_{\beta_2} &= \rho_{\beta_2} - c\beta_2. \end{aligned}$$

Alice sends these $n + 4\ell + m + 7$ values to Bob.

- *Verification Phase*: Bob validates the proof by evaluating the following $3\ell + m + 4$ equations.

$$\begin{aligned} T_1 &= \mathfrak{A}_1^{c} g^{z_{k_1}} h^{z_{k_2}}, \\ T_2 &= \mathfrak{A}_1^{-z_e} g^{z_{\beta_1}} h^{z_{\beta_2}}, \\ T_3 &= (\frac{\hat{e}(\mathfrak{A}_2, w)}{\hat{e}(h,h)})^{c}\hat{e}(h_0,h)^{z_y}\hat{e}(h_1,h)^{z_{x_1}}\cdots\hat{e}(h_n,h)^{z_{x_n}} \\ &\quad \cdot \hat{e}(\hat{g},h)^{z_s}\hat{e}(h,w)^{z_{k_1}}\hat{e}(h,h)^{z_{\beta_1}}\hat{e}(\mathfrak{A}_2,h)^{-z_e}, \\ T_4 &= (\hat{e}(g,h_0)C^{-R})^{c} C^{z_y}, \\ T_{5,i} &= D_i^{c} g^{z_{x_{\rho(i)}}} h^{z_{a_i}}, \quad \text{for } i = 1 \text{ to } \ell \\ T_{6,i} &= C_i^{c} g^{z_{v_i}} h^{z_{t_i}}, \quad \text{for } i = 1 \text{ to } \ell \\ T_{7,i} &= C_i^{c} D_i^{z_{v_i}} h^{z_{b_i}}, \quad \text{for } i = 1 \text{ to } \ell \\ T_8 &= (\frac{\prod_{i=1}^{\ell} C_i^{M_{i,1}}}{g})^{c} h^{z_{f_1}}, \\ T_{9,j} &= (\prod_{i=1}^{\ell} C_i^{M_{i,j}})^{c} h^{z_{f_j}} \quad \text{for } j = 2 \text{ to } m. \end{aligned}$$

Bob accepts the proof if and only if all the above equalities hold. In practice, most of the pairing operations can be pre-computed. Only $\hat{e}(\mathfrak{A}_2, w)$ and $\hat{e}(\mathfrak{A}_2, h)$ need to be computed online.
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 $\mathcal{PK}'_1$: Our instantiation of $\mathcal{PK}_1$ thus consists of the auxiliary values $\mathfrak{A}_1$ and $\mathfrak{A}_2$, in addition to $\mathcal{PK}'_1$. $\mathcal{PK}'_1$ consists of various statements related to the knowledge of discrete logarithm. Interested readers may refer to [9, Ch. 3] which discusses how complex statements (such as the one employed in $\mathcal{PK}'_1$) can be composed from simple statements. Furthermore, the resulting proof protocol 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 whose distribution is identical to those coming from Alice. Furthermore, $\mathcal{PK}_1'$ is sound, meaning that there exists another efficient algorithm, called extractor $\mathcal{E}'$, which is capable of outputting the underlying set $(e, s, y, \{x_i\}_{i=1}^n, \{a_i\}_{i=1}^\ell, \{b_i\}_{i=1}^\ell, \{v_i\}_{i=1}^\ell, \{t_i\}_{i=1}^\ell, \{f_i\}_{i=1}^m, k_1, k_2, \beta_1, \beta_2)$ when it is given blackbox access to a prover in $\mathcal{PK}_1'$.

c) Honest-verifier zero-knowledgeness of $\mathcal{PK}_1$: To show that our instantiation of $\mathcal{PK}_1$ 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 $\mathcal{PK}_1$ on input challenge c.

Our construction of $\mathcal{S}$ simply picks at random $\mathfrak{A}_1, \mathfrak{A}_2 \in_R \mathbb{G}$ and invokes the zero-knowledge simulator $\mathcal{S}'$ for $\mathcal{PK}_1'$. 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 value A, there exists a unique value k_2 such that $\mathfrak{A}_2 = Ah^{k_1}$. For this specific k_2, there exists a unique value k_1 such that $\mathfrak{A}_1 = g^{k_1}h^{k_2}$. In other words, the auxiliary values $\mathfrak{A}_1, \mathfrak{A}_2$ picked by $\mathcal{S}$ are correctly distributed.

d) Soundness of $\mathcal{PK}_1$: Soundness of $\mathcal{PK}_1'$ guarantees existence of a simulator $\mathcal{E}'$ which allows our construction of $\mathcal{E}$ to extract from the prover the set of values $(e ,s, y, \{x_i\}_{i=1}^n, \{a_i\}_{i=1}^\ell, \{b_i\}_{i=1}^\ell, \{v_i\}_{i=1}^\ell, \{t_i\}_{i=1}^\ell, \{f_i\}_{i=1}^m, k_1, k_2, \beta_1, \beta_2)$. It remains to show how $\mathcal{E}$ can obtain the value A which satisfies the equations for $\mathcal{PK}_1$.

$\mathcal{E}$ first invokes $\mathcal{E}'$. Due to the soundness of the first equation in $\mathcal{PK}_1'$, we have

$$\begin{aligned}\frac{\hat{e}(\mathfrak{A}_2, w)}{\hat{e}(h, h)} &= \hat{e}(h_0, h)^y \hat{e}(h_1, h)^{x_1} \cdots \hat{e}(h_n, h)^{x_n}\\ &\quad \hat{e}(\hat{g}, h)^s \hat{e}(h, w)^{k_1}\\ &\quad \hat{e}(h, h)^{\beta_1} / \hat{e}(\mathfrak{A}_2, h)^e\end{aligned}$$

Due to the soundness of $\mathcal{PK}_1'$, we have $\mathfrak{A}_1 = g^{k_1}h^{k_2}$ and $1 = \mathfrak{A}_1^e g^{\beta_1} h^{\beta_2}$. This implies $\beta_1 = k_1 e$. Putting $\beta_1 = k_1 e$ and rearranging the terms,

$$\hat{e}(\mathfrak{A}_2 h^{-k_1}, wh^e) = \hat{e}(hh_0^y h_1^{x_1} \cdots h_n^{x_n} \hat{g}^s, h)$$

Thus, $\mathcal{E}$ can compute A as $\mathfrak{A}_2 h^{-k_1}$. $\mathcal{E}$ outputs $(A, e, s, y, \{x_i\}_{i=1}^n, \{a_i\}_{i=1}^\ell, \{b_i\}_{i=1}^\ell, \{v_i\}_{i=1}^\ell, \{t_i\}_{i=1}^\ell, \{f_i\}_{i=1}^m)$ as the set of witnesses satisfying $\mathcal{PK}_1$. Therefore, our instantiation of PK_1 is sound.

D. Security Analysis

1) Authentication, Access Without Security Device, Access Without Secret Key: Let n be the total number of users of the system. Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of users. We use (Y_i, y_i), $\mathcal{A}_i$, $\mathsf{sk}_{\mathcal{A}_i, Y_i}^{(i)}$, token_i to denote the user key pair, attribute set, attribute key and the security device for U_i respectively.

If user U_i is controlled by the attacker, we assume y_i is chosen by the attacker and $(\mathsf{sk}_{\mathcal{A}_i, Y_i}^{(i)}, \mathsf{token}_i)$ is given to the attacker. In addition, the attacker may have $(Y_i, y_i, \mathsf{sk}_{\mathcal{A}_i, Y_i}^{(i)})$ or token_i, but not both, from some honest user U_i.

To model the temper-resistant nature of the security device, we model token_i as an oracle $\mathcal{O}_i$ with the following behaviour:

Oracle $\mathcal{O}_i$	(internal state: TG, TY, tsk)
Input	(C, y, R)
Output	(c_R, z_R) if $\mathsf{TG} = C^{y+R} \wedge \mathsf{TY} = \mathsf{TG}^y$ s.t. $c_R = H(\mathsf{tpk}^{c_R} \hat{e}(g, h_0)^{z_R} \| R \| C)$; $\perp$ otherwise.

We allow the attacker to specify the security device for revocation. If a security device token_i is revoked, oracle $\mathcal{O}_i$ will no longer be available.

We further assume the claim-predicate Υ is chosen by the attacker. An attacker is said to breach the security requirement of authentication, access without security device or access without secret key if it can authenticate successfully for the predicate Υ if for all i such that U_i is controlled by the attacker, $\Upsilon(\mathcal{A}_i) \neq 1$ unless the token_i has been revoked.

The last condition is to capture the situation that the security device is used as a mechanism to revoke a user. A user who is in possession of a security device should not be able to authenticate anymore after it has been revoked.

Regarding the security of our scheme, we have the following lemma.

Lemma 1: If there exists an attacker $\mathcal{F}$ against our scheme, there exists a simulator $\mathcal{S}$, having blackbox access to $\mathcal{F}$, that can existentially forge a BBS+ signature or the Schnorr signature under the adaptive chosen message attack or solving the discrete logarithm problem.

Proof: In the following we prove Lemma 1 by constructing the simulator $\mathcal{S}$ under the assumption that attacker $\mathcal{F}$ exists. We utilise the fact that PK_0 and PK_1 are zero-knowledge proof-of-knowledge protocols. In other words, there exist knowledge extractors E_0 and E_1 that can extract the underlying witnesses of the corresponding protocols.

- *Common Parameters.* Let $p, \mathbb{G}, \mathbb{G}_T, \hat{e}$ be a bilinear group with $g \in \mathbb{G}$ being a generator. Let $\hat{g}, h, h_0, h_1, \ldots, h_n$ be additional generators of $\mathbb{G}$. We use TG to denote $\hat{e}(g, h_0)$. We further assume full domain hash $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ which is modelled as a random oracle.
- *Problem Instances.* $\mathcal{S}$ is given a discrete logarithm problem instance Y^*, h_0. $\mathcal{S}$ is also given the public key of an instance of BBS+ signature $\mathsf{B.PK}$ and the public key of an instance of Schnorr signature $\mathsf{S.PK}$ defined over the common parameters. Specifically,

  $$\mathsf{B.PK} = w = h^\gamma, \quad \mathsf{S.PK} = \mathsf{tpk} = \mathsf{TG}^{\mathsf{tsk}}.$$

  The corresponding signing keys, $\mathsf{B.SK} = \gamma$ and $\mathsf{S.SK} = \mathsf{tsk}$, are kept secret from $\mathcal{S}$. As an adaptive chosen message attacker, $\mathcal{S}$ can issue signature queries to the following two oracles:

  - $\mathcal{O}_{BBS+}$. On input $(m_0, m_1, \ldots, m_n)$, this oracle returns (A, e, s) such that

    $$\hat{e}(A, wh^e) = \hat{e}(hh_0^{x_0} h_1^{x_1} \cdots h_n^{x_n} \hat{g}^s, h).$$

  - $\mathcal{O}_{Schnorr}$. On input (m), this oracle returns (c, z) such that

    $$c = H(\mathsf{tpk}^c \mathsf{TG}^z \| m).$$

- *Goal of $\mathcal{S}$.* The goal of $\mathcal{S}$ is to output y^* such that $Y^* = h_0^{y^*}$ or to output a BBS+ signature (resp. Schnorr signature) on a message M that has never been input to $\mathcal{O}_{BBS+}$ (resp. $\mathcal{O}_{Schnorr}$).

Now we are ready to construct $\mathcal{S}$ who has blackbox access to attacker $\mathcal{F}$ against our scheme and show that whenever $\mathcal{F}$ is successful, $\mathcal{S}$ can output a forgery.

- *System Setup.* Using the common parameters, $\mathsf{B.PK}$ and tpk, $\mathcal{S}$ sets

$$\mathsf{TPK} = (\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g, \hat{g}, h, h_0, h_1, \ldots, h_n, H, \mathsf{tpk})$$

and

$$\mathsf{APK} = w.$$

TPK and APK are given to $\mathcal{F}$.

- *User Key Generation.* $\mathcal{S}$ maintains four index sets, namely, $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4$, all of which are empty initially. For each user U_i, $\mathcal{F}$ could choose one of the following three options.
  1) *Full Control.* $\mathcal{F}$ presents a value Y_i, which will be treated as UPK of the user U_i. $\mathcal{F}$ will also present a set $\mathcal{A}_i = (x_1, \ldots, x_n)$, which represents the attribute of user U_i. $\mathcal{F}$ and $\mathcal{S}$ engage in protocol PK_0. $\mathcal{S}$ invokes the extractor E_0 to obtain y_i such that $Y_i = h_0^{y_i}$. Next, $\mathcal{S}$ queries its oracle $\mathcal{O}_{BBS+}$ on input $(y_i, x_1, \ldots, x_n)$ and obtains $\mathsf{sk}_{\mathcal{A}_i, Y_i} = (A_i, e_i, s_i)$. $\mathcal{S}$ returns to $\mathcal{F}$ $(\mathsf{sk}_{\mathcal{A}_i, Y_i})$ as the user's attribute key. $\mathcal{F}$ will also have access to oracle $\mathcal{O}_i$ which simulates token_i. How $\mathcal{S}$ simulates $\mathcal{O}_i$ will be discussed in the next item. $\mathcal{S}$ sets $\mathcal{I}_1 = \mathcal{I}_1 \cup \{i\}$.
  2) *Stolen Key.* $\mathcal{F}$ presents an attribute set $(\mathcal{A}_i = (x_1, \ldots, x_n))$ for user U_i. $\mathcal{S}$ randomly picks y_i, computes $Y_i = h_0^{y_i}$ and sets UPK for user U_i to be Y_i. Next, $\mathcal{S}$ queries its oracle $\mathcal{O}_{BBS+}$ on input $(y_i, x_1, \ldots, x_n)$ and obtains $\mathsf{sk}_{\mathcal{A}_i, Y_i} = (A_i, e_i, s_i)$. $\mathcal{S}$ returns to $\mathcal{F}$ $(\mathsf{sk}_{\mathcal{A}_i, Y_i})$ as the user's attribute key and y_i as USK. $\mathcal{S}$ sets $\mathcal{I}_2 = \mathcal{I}_2 \cup \{i\}$.
  3) *Stolen Device.* $\mathcal{F}$ presents an attribute set $(\mathcal{A}_i = (x_1, \ldots, x_n))$ for user U_i. $\mathcal{S}$ randomly picks β_i, computes $Y_i = Y^{*\beta_i}$ and sets UPK for user U_i to be Y_i. $\mathcal{S}$ returns to $\mathcal{F}$ the public key UPK and $\mathcal{F}$ is allowed to query oracle $\mathcal{O}_i$. $\mathcal{S}$ sets $\mathcal{I}_3 = \mathcal{I}_3 \cup \{i\}$.
- *Oracle $\mathcal{O}_i$.* Note that $\mathcal{F}$ is only allowed to access $\mathcal{O}_i$ for $i \in \mathcal{I}_1 \cup \mathcal{I}_3$. $\mathcal{F}$ inputs (C, y, R) to oracle $\mathcal{O}_i$. $\mathcal{S}$ first checks if $\hat{e}(g, \mathsf{UPK}_i) = \mathsf{TG}^y$ and $C^{y+R} = \mathsf{TG}$. If any of these checks fail, return $\perp$. If $i \in \mathcal{I}_3$ and the checks succeed, $\mathcal{S}$ returns y/β_i as the solution to the discrete logarithm problem and aborts.[4] Otherwise, $\mathcal{S}$ issues a query to $\mathcal{O}_{Schnorr}$ on input $(R||C)$ and obtains (c, z). $\mathcal{S}$ returns (c, z) to $\mathcal{F}$.
- *Device revocation.* $\mathcal{F}$ instructs $\mathcal{S}$ to revoke security device token_i for some $i \in \mathcal{I}_1$. $\mathcal{S}$ sets $\mathcal{I}_1 = \mathcal{I}_1 \setminus \{i\}$ and $\mathcal{I}_4 = \mathcal{I}_4 \cup \{i\}$. $\mathcal{F}$ will no longer be able to access oracle $\mathcal{O}_i$.

[4]Recall that for $i \in \mathcal{I}_3$, $\mathsf{UPK}_i = Y^{*\beta_i}$. Thus, $\hat{e}(g, \mathsf{UPK}_i) = \mathsf{TG}^y$ implies $\hat{e}(g, Y^*)^{\beta_i} = \hat{e}(g, h_0)^y$ which in turns implies $Y^* = h_0^{y/\beta_i}$.

Finally, $\mathcal{F}$ chooses a claim-predicate Υ. This is subject to the condition that for all $i \in \mathcal{I}_1$, $\Upsilon(\mathcal{A}_i) \neq 1$.

- $\mathcal{S}$ picks a random R and sends it to $\mathcal{F}$.
- $\mathcal{F}$ submits $(C, c_R, z_R, C_1, \ldots, C_\ell, D_1, \ldots, D_\ell)$ and engages with $\mathcal{S}$ in protocol PK_1. $\mathcal{S}$ uses extractor E_1 to extract from $\mathcal{F}$ the set of witnesses

$$\begin{pmatrix} A, e, s, y, \{x_i\}_{i=1}^n, \{a_i\}_{i=1}^\ell, \{b_i\}_{i=1}^\ell, \\ \{v_i\}_{i=1}^\ell, \{t_i\}_{i=1}^\ell, \{f_i\}_{i=1}^m \end{pmatrix}.$$

- $\mathcal{S}$ checks the history of query to $\mathcal{O}_i$. If there is no query with input $(C, \cdot, R)$, (c_R, z_R) forms a new Schnorr signature on $C||R$. In this case, $\mathcal{S}$ simply outputs (c_R, z_R) as the forged signature on $R||C$ and aborts.
- If $\mathcal{S}$ does not abort, the query to $\mathcal{O}_i$ must be from an index i in $\mathcal{I}_1$ or $\mathcal{I}_3$. The probability that it is in $\mathcal{I}_4$ is negligible since R is chosen at random. We have already shown a successful query for $\mathcal{O}_i$ such that $i \in \mathcal{I}_3$ implies a solution to the discrete logarithm problem. Thus, it is safe to conclude that $i \in \mathcal{I}_1$. We denote the index to be i^*.
- Due to the soundness of PK_1, the value y extracted satisfies the relationship

$$\hat{e}(g, h_0)C^{-R} = C^y.$$

This implies $\mathsf{TG} = C^{y+R}$. Since each UPK is unique, and there is only one unique y that allows the use of oracle $\mathcal{O}_i$, the index i^* is unique and there must have been an input (C, y, R) to $\mathcal{O}_i^*$. This value y satisfies $\mathsf{UPK}_i^* = h_0^y$.
- $\mathcal{S}$ has only issue one oracle query to $\mathcal{O}_{BBS+}$ with input $(y, \ldots)$ (this is issued when it is creating user i^*). Denote the query as $(y, x_1', \ldots, x_n')$. On the other hand, due to the soundness of PK_1, the extracted values $(A, e, s, y, x_1, \ldots, x_n)$ satisfy the relationship

$$\hat{e}(A, wh^e) = \hat{e}(hh_0^y h_1^{x_1} \ldots h_n^{x_n} \hat{g}^s, h).$$

In other words, (A, e, s) is a valid BBS+ signature on $(y, x_1, \ldots, x_n)$. In the following we show $(y, x_1, \ldots, x_n) \neq (y, x_1', \ldots, x_n')$ and thus $\mathcal{S}$ can output the former as the forged BBS+ signature. Recall that $\mathcal{A}' := (x_1', \ldots, x_n')$ represents the attribute set for user U_i^*. Thus, $\Upsilon(\mathcal{A}') \neq 1$ since $i^* \in \mathcal{I}_1$.
Due to the soundness of PK_1, $\{x_i\}_{i=1}^n, \{a_i\}_{i=1}^\ell, \{b_i\}_{i=1}^\ell, \{v_i\}_{i=1}^\ell, \{t_i\}_{i=1}^\ell)$ satisfy

$$\bigwedge_{i=1}^{\ell} \left(D_i = g^{x_{\rho(i)}} h^{a_i}\right) \ \wedge$$
$$\bigwedge_{i=1}^{\ell} \left(C_i = g^{v_i} h^{t_i}\right) \ \wedge$$
$$\bigwedge_{i=1}^{\ell} \left(C_i = D_i^{v_i} h^{b_i}\right).$$

Since all x_i' are from $\{0, 1\}$, we have completed the proof if there exists $x_i \neq 0$ and $x_i \neq 1$. It means that D_i can only be gh^{a_i} or h^{a_i}. If $x_{\rho(i)} = 0$, v_i must be 0. Otherwise, $\mathcal{S}$ has found a way to represent C_i as $g^{v_i} h^{t_i}$ and also $h^{a_i v_i + b_i}$ and can thus solve the discrete logarithm problem of h to base g. Following this idea, $g^a h^b = g^c h^d$

TABLE II
TIME AND SPACE COST ON DIFFERENT PLATFORMS

	Computer	Smartphone	Smartcard [40]
P	6.4	32	-
$E1$	2.5	13	-
ET	0.4	2.2	200
Zp	160		
G	512		
GT	1024		

TABLE III
TEST PLATFORMS

	Computer	Smartphone
Config	Intel Core i5-4210U@1.70Ghz	Samsung Galaxy S5 Quad-core 2.45Ghz
Ram	4GB	2GB
OS	Windows 7 Pro	Andriod 4.4.2

implies $a = c$ and $b = d$ under the discrete logarithm assumption.

Consider the last two relations from PK_1:

$$(\prod_{i=1}^{\ell} C_i^{M_{i,1}})/g = h^{f_1} \wedge \bigwedge_{j=2}^{m} (\prod_{i=1}^{\ell} C_i^{M_{i,j}} = h^{f_j}).$$

Equating the exponent of g on both sides, we have $\sum_{i=1}^{\ell} v_i M_{i,1} = 1$. Likewise, we have $\sum_{i=1}^{\ell} v_i M_{i,j} = 0$ for $j = 2$ to m. That is, $\vec{v}\mathbf{M} = (1, 0, \ldots, 0)$.

In other words, there exists $\vec{v}$ such that $\vec{v}\mathbf{M} = (1, 0, \ldots, 0)$ and for all i, $x_{\rho(i)} = 0$ implies $v_i = 0$. It means that $\Upsilon(x_1, \ldots, x_n) = 1$. Recall that $\Upsilon(x'_1, \ldots, x'_n) \neq 1$, we have shown that $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_n)$. Thus, $\mathcal{S}$ can submit (A, e, s) as a forged BBS+ signature on message $(y, x_1, \ldots, x_n)$. □

2) Privacy: In every authentication, the information obtained by the server consists of two parts, namely, $(C, c_R, z_R, C_1, \ldots, C_\ell, D_1, \ldots, D_\ell)$ and the verifier's view of PK_1. Note that PK_1 is zero-knowledge and is simulated without having the actual witnesses. Note that this implies we have to employ the zero-knowledge version instead of the honest-verifier zero-knowledge version since the server is playing the role of the verifier. c_R, z_R leaks no information since they are distributed identically for all legitimate users. $C_1, \ldots, C_\ell, D_1, \ldots, D_\ell$ are information-theoretic secure commitment and again leak no information. The only component containing information about the user is $C = \hat{e}(g, h_0)^{\frac{1}{y+R}}$. This is the verifiable random function with seed y on input R and is computationally indistinguishable from a random value in $\mathbb{G}_T$ [18].

E. Efficiency Analysis

We analyze the efficiency of our protocol in two parts. In the first part, we identify the major operations for the authentication protocol in Table IV. The symbols P, $E1$, ET represent the time cost (in ms) of a pairing operation, an exponentiation in group $\mathbb{G}$ and group $\mathbb{G}_T$ respectively. The symbol Zp, G, GT represents the size of an element (in bits) in $\mathbb{Z}_p$, $\mathbb{G}$ and $\mathbb{G}_T$ respectively.

TABLE IV
AUTHENTICATION COMPLEXITY

	Time and Space Cost
User's Computer	$(9\ell + m + n + 6)$ E1 + 2 ET+1 P
Security Device	3 ET
Server	$(9\ell + 2m + n + 12)$ E1 + 1 ET + 2 P
Transmission	3GT + $(5\ell + 2)$G+$(4\ell + m + n + 12)$Zp

We consider three different platforms, namely, a computer, a smart phone and a smart card.

For the time cost on a smartcard, we use the benchmark result from [40]. The configuration of our test platforms, namely, Computer and Smartphone, are shown in Table III. The time and space cost on the three platforms are listed in Table II. Details of the experiment settings are discussed below.

We use Miracl library version 5.2. The base field is a prime field F_q, where q is a 512-bit prime whose value is:

$$8BA2A5229BD9C57CFC8ACEC76DFDBF3E$$
$$3E1952C6B3193ECF5C571FB502FC5DF4$$
$$10F9267E9F2A605BB0F76F52A79E8043$$
$$BF4AF0EF2E9FA78B0F1E2CDFC4E8549B$$

The elliptic curve is defined by the equation $y^2 = x^3 + 1 \bmod q$. The group $\mathbb{G}$ (as well as $\mathbb{G}_T$) is of order $p = 8000000000000000000000000000000000020001$, where p is a 160-bit prime. The pairing is Tate pairing. Table IV listed the number of operations and communication for an authentication transaction. Recall that n is the size of the attribute universe, ℓ and m are the length and width of the span program representing the access policy.

1) Simulation: Assume the total number of attributes in the system is 100. In other words, the attribute universe $\mathbb{A} = \{1, \ldots, 100\}$. In the following we estimate the efficiency of our system using policy of the following format:

$$\bigvee_{i=1}^{a} \left(\bigwedge_{j=1}^{b} (\mathsf{attr}_{i,j}) \right),$$

where $\mathsf{attr}_{i,j}$ maybe re-used in different clauses. In general, this kind of policy can be represented by a span program of length $\ell = a * b$ and width $m = a * (b - 1) + 1$. The following graphs shows the bandwidth requirement, computational cost at server and user of our system for policy of various size.

Fig. 2 shows the time cost of the server to authenticate a single user. For a relatively simple policy, say, consisting of 2 clauses with 2 attributes per clause for a total of 4 attributes, the time is less than 0.3 seconds. For a policy of 10 clauses with 10 attributes per clause, the time is around 3 seconds. While the asymptotic complexity at the user is similar to that of the server, the time cost for a user is

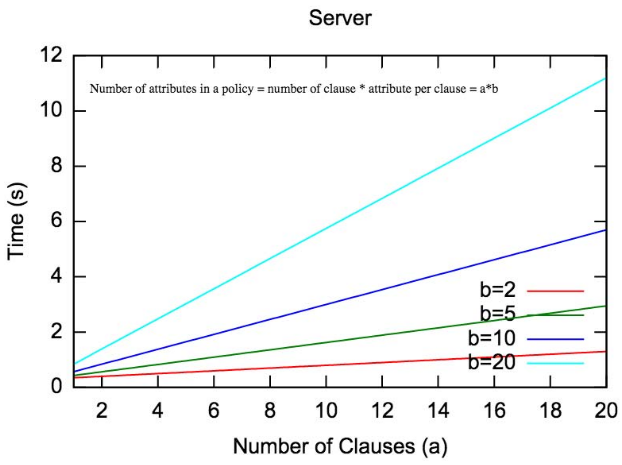


Fig. 2. Running time of the Auth protocol (Server side) (s).

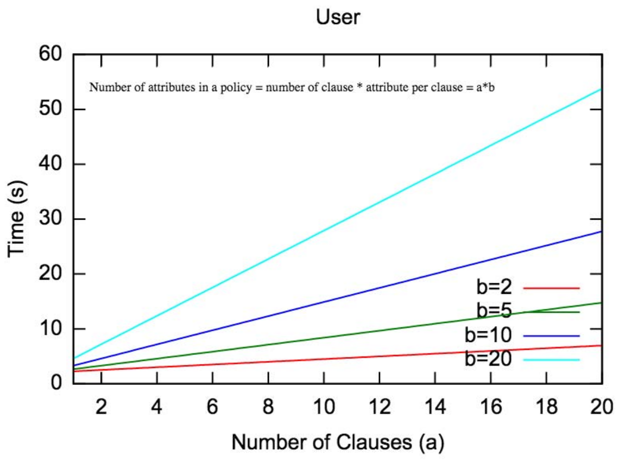


Fig. 3. Running time of the Auth protocol (User side) (s).

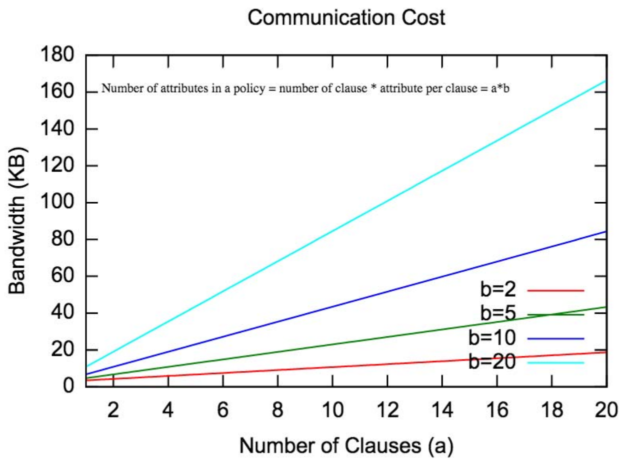


Fig. 4. Communication cost of the Auth protocol (KB).

about five times slower due to the use of a less powerful computing device (a smartphone). One should note that the security device is not the bottleneck as it only accounts for a constant time cost of 0.6 seconds. Please refer to Fig. 3 for the time complexity at the user side. The total authentication time for a policy with 100 attributes, arranged as 10 clauses with 10 attributes each, is about 18 seconds. The communication cost of our protocol is depicted in Fig. 4. In particular, for a policy of 100 attributes, the total bandwidth requirement is around 45 KB, which is acceptable for today's network. One could conclude that our protocol is plausible for very simple policy and is still not practical yet for policy of medium size.

Having said that, we would like to remark that the protocol might be optimised. Two possible approaches could be adopted. Firstly, notice that many of the exponentiations are of the form $g^x h^y$ for some fixed bases g and h. This kind of operation is known as multi-base exponentiation and can be computed at about the cost of 110% of a single base exponentiation. It is also worth noting that for fixed base, there are a number of pre-processing techniques available. It is quite likely to reduce the time by half.

VI. Conclusion

In this paper, we have presented a new 2FA (including both user secret key and a lightweight security device) access control system for web-based cloud computing services. Based on the attribute-based access control mechanism, the proposed 2FA access control system has been identified to not only enable the cloud server to restrict the access to those users with the same set of attributes but also preserve user privacy. Detailed security analysis shows that the proposed 2FA access control system achieves the desired security requirements. Through performance evaluation, we demonstrated that the construction is "feasible". We leave as future work to further improve the efficiency while keeping all nice features of the system.

Appendix
A Running Example

We will use the access structure $(\mathsf{attr}_1 \wedge \mathsf{attr}_2) \vee \mathsf{attr}_3$ on attributes $\mathsf{attr}_1, \mathsf{attr}_2, \mathsf{attr}_3$ for our running example. This simple policy could represent, for example, access to the computer laboratory is allowed for students from the CS department or any staff member. The attribute universe is $\mathbb{A} = \{1, 2, 3\}$. This policy can be represented by the span program $(\mathbf{M}, \rho)$ where $\mathbf{M}$ is a 3×2 matrix:

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

with the labelling function $\rho : [1, 3] \to \mathbb{A}$ defined by $\rho(1) = 3$, $\rho(2) = 2$ and $\rho(3) = 1$. In this example, a user with attributes $(\mathsf{attr}_1, \mathsf{attr}_2)$ can compute the vector $\vec{v} = (0, 1, -1)$ which illustrate his satisfaction of the policy since:

$$(0, 1, -1) \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} = (1, 0);$$

and for all i, $v_{\rho(i)} = 0$ if the user is not in possession of attribute attr_i.

Likewise, the vector for a user with attribute (attr_3) can compute the vector $(1, 0, 0)$. Note that $v_{\rho(1)}$ and $v_{\rho(2)}$ for this vector are both 0 and

$$(1,0,0)\begin{bmatrix}1 & 0\\ 1 & 1\\ 0 & 1\end{bmatrix} = (1,0).$$

A. Key Generation of User With Attributes $(\mathsf{attr}_1, \mathsf{attr}_2)$

Let $\mathsf{USK} = y$ and $\mathsf{UPK} = Y = h_0^y$. The security device of this user is initialized with $\mathsf{TY} = \hat{e}(g, Y)$, $\mathsf{TG} = \hat{e}(g, h_0)$ and tsk. Upon completion of the $\mathsf{AttrGen}$ protocol, the user obtains a $\mathsf{sk}_{\{\mathsf{attr}_1,\mathsf{attr}_2\},Y} := (A, e, s)$ such that

$$A = (hYh_1h_2\hat{g}^s)^{\frac{1}{y+e}}.$$

B. Auth

Step 1 to 4 of the authentication protocol is quite intuitive. Below we show in detail what this user does from step 5 of the protocol.

5) Since the user is in possession of attributes $(\mathsf{attr}_1, \mathsf{attr}_2)$, his vector $\vec{v} = (v_1, v_2, v_3)$ will be $(0,\ 1,\ -1)$ such that $\vec{v}\mathbf{M} = (1, 0)$. Recall that $\mathbf{M}$ is a 3×2 Matrix of the form:

$$\begin{bmatrix}1 & 0\\ 1 & 1\\ 0 & 1\end{bmatrix}.$$

6) For $i = 1$ to 3, the user randomly picks $a_i, t_i \in_R \mathbb{Z}_p$ and computes $C_i = g^{v_i}h^{t_i}$, $D_i = g^{x_{\rho(i)}}h^{a_i}$. The user also computes $b_i = t_i - a_iv_i$. That is,

$$\begin{aligned}C_1 &= h^{t_1}, \quad D_1 = gh^{a_1}, \quad b_1 = t_1\\ C_2 &= gh^{t_2}, \quad D_2 = gh^{a_2}, \quad b_2 = t_2 - a_2\\ C_3 &= g^{-1}h^{t_3}, \quad D_3 = h^{a_3}, \quad b_3 = t_3 + a_3\end{aligned}$$

7) For $j = 1$ to 2, the user computes $f_j = \sum_{i=1}^{3} t_iM_{i,j}$. That is,

$$\begin{aligned}f_1 &= t_1 + t_2\\ f_2 &= t_2 + t_3\end{aligned}$$

Then the user sends $(C,\ c_R,\ z_R,\ C_1,\ C_2,\ C_3,\ D_1,\ D_2,\ D_3,\ f_1,\ f_2)$ to the authentication server. Note that $(b_1,\ b_2,\ b_3,\ f_1,\ f_2)$ are not sent. They are used as witnesses in the zero-knowledge proof.

8) We explain the intuition of $\mathcal{PK}_1$ for this access policy in this particular scenario.

- The first relation

$$\hat{e}(A, wh^e) = \hat{e}(hh_0^yh_1^{x_1}h_2^{x_2}h_3^{x_3}\hat{g}^s, h)$$

convinces the server that the tuple (y, x_1, x_2, x_3) has been properly certified by the attribute-issuing authority. For this user, the tuple is $(y, 1, 1, 0)$ where y is his secret key. Soundness of the zero-knowledge proof (and the unforgeability of BBS+ signature) guarantees that the user can only use his secret key y, $x_1 = 1$, $x_2 = 1$ and $x_3 = 0$ in this proof.

- The second relation

$$\hat{e}(g, h_0)C^{-R} = C^y$$

convinces the server that the second step of the protocol is correctly computed since it implies $C = \hat{e}(g, h_0)^{\frac{1}{y+R}}$.

- We look at the next three relations together:

$$\begin{aligned}&D_1 = g^{x_3}h^{a_1} \wedge \quad D_2 = g^{x_2}h^{a_2} \wedge \quad D_3 = g^{x_1}h^{a_3} \wedge\\ &C_1 = g^{v_1}h^{t_1} \wedge \quad C_2 = g^{v_2}h^{t_2} \wedge \quad C_3 = g^{v_3}h^{t_3} \wedge\\ &C_1 = {D_1}^{v_1}h^{b_1} \wedge \quad C_2 = {D_2}^{v_2}h^{b_2} \wedge \quad C_3 = {D_3}^{v_3}h^{b_3}\end{aligned}$$

The goal of these three relations is to convince the verifier that $v_{\rho(i)} = 0$ if $x_i = 0$. To see this, we plug in the value of x_i for this authenticating user which is in possession of $\mathsf{attr}_1, \mathsf{attr}_2$. That is, $x_1 = 1$, $x_2 = 1$ and $x_3 = 0$.

$$\begin{aligned}&D_1 = h^{a_1} \wedge \quad D_2 = gh^{a_2} \wedge \quad D_3 = gh^{a_3} \wedge\\ &C_1 = g^{v_1}h^{t_1} \wedge \quad C_2 = g^{v_2}h^{t_2} \wedge \quad C_3 = g^{v_3}h^{t_3} \wedge\\ &C_1 = {D_1}^{v_1}h^{b_1} \wedge \quad C_2 = {D_2}^{v_2}h^{b_2} \wedge \quad C_3 = {D_3}^{v_3}h^{b_3}\end{aligned}$$

Looking at the first column, since $D_1 = h^{a_1}$ and $C_1 = {D_1}^{v_1}h^{b_1}$, we have $C_1 = h^{a_1v_1+b_1}$. Now we also have $C_1 = g^{v_1}h^{t_1}$. Thus, we have $v_1 = 0$ and $t_1 = a_1v_1 + b_1 = b_1$. This deduction is correct under the assumption that the discrete logarithm of h to base g is not known to the prover. On the other hand, if $x_{\rho(i)} = 1$, the value of v_i can be arbitrary. For example, since $D_2 = gh^{a_2}$, $C_2 = {D_2}^{v_2}h^{b_2}$ implies $C_2 = g^{v_2}h^{a_2v_2+b_2}$. And we have $C_2 = g^{v_2}h^{t_2}$. In this case, we have $t_2 = a_2v_2 + b_2$ and there is no constraint on v_2.

- Finally, we can look at the last two relations:

$$C_1C_2/g = h^{f_1} \wedge C_2C_3 = h^{f_2}.$$

From the discussions above discussions, we know that $C_i = g^{v_i}h^{t_i}$ for $i = 1$ to 3. Thus, $C_1C_2/g = h^{f_1}$ implies $g^{v_1+v_2}h^{t_1+t_2} = gh^{f_2}$. It in turn implies that $v_1 + v_2 = 1$. Likewise, we have $v_2 + v_3 = 0$. From this, the prover is convinced that:

$$(v_1, v_2, v_3)\begin{bmatrix}1 & 0\\ 1 & 1\\ 0 & 1\end{bmatrix} = (v_1 + v_2, v_2 + v_3) = (1, 0).$$

- In other words, the authenticating user has a vector $\vec{v} = (v_1, v_2, v_3)$ such that $\vec{v}\mathbf{M} = (1, 0)$ and that $v_i = 0$ if $x_{\rho(i)} = 0$. Thus, the server can be convinced that the user satisfies the access policy.

9) Finally, the authentication server validates that $c_R = H(\mathsf{tpk}^{c_R}\hat{e}(g, h_0)^{z_R}||R||C)$. It means that the user is in possession of the security device that endorse this specific transaction (through endorsing the nonce R) and the use of the user secret key y.

This completes the descriptions of an authentication example.

C. An Intuition of Anonymity

To give an intuition of the proof of privacy, we consider the values used by another user who is in possession of attribute attr_3. Assume his secret key is (A', e', s', y'). The vector he use will be $\vec{v}' = (v'_1, v'_2, v'_3) = (1, 0, 0)$. Below we show the values used in step 6 for this user.

For $i = 1$ to 3, the user randomly picks $a'_i, t'_i \in_R \mathbb{Z}_p$ and computes $C'_i = g^{v'_i} h^{t'_i}$, $D'_i = g^{x'_{\rho(i)}} h^{a'_i}$. The user also computes $b'_i = t'_i - a'_i v'_i$.

$$C'_1 = g h^{t'_1}, \quad D'_1 = h^{a'_1}, \quad b'_1 = t'_1 - a'_1$$
$$C'_2 = h^{t'_2}, \quad D'_2 = h^{a'_2}, \quad b'_2 = t'_2$$
$$C'_3 = h^{t'_3}, \quad D'_3 = h^{a'_3}, \quad b'_3 = t'_3$$

The random numbers a'_i, t'_i are picked uniformly at random and we are working in a cyclic group where g and h are generators. Thus, the distribution of $\{C'_i, D'_i\}$ is the same as that of $\{C_i, D_i\}$. This is the intuition behind the proof that the server would not be able to distinguish the different attributes used by the users in an authentication.

References

[1] M. H. Au and A. Kapadia, "PERM: Practical reputation-based blacklisting without TTPS," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, Raleigh, NC, USA, Oct. 2012, pp. 929–940.

[2] M. H. Au, A. Kapadia, and W. Susilo, "BLACR: TTP-free blacklistable anonymous credentials with reputation," in *Proc. 19th NDSS*, 2012, pp. 1–17.

[3] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-TAA," in *Proc. 5th Int. Conf. SCN*, 2006, pp. 111–125.

[4] J. Baek, Q. H. Vu, J. K. Liu, X. Huang, and Y. Xiang, "A secure cloud computing based framework for big data information management of smart grid," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 233–244, Apr./Jun. 2015.

[5] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Proc. 12th Annu. Int. CRYPTO*, 1992, pp. 390–420.

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

[7] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 2004, pp. 41–55.

[8] D. Boneh, X. Ding, and G. Tsudik, "Fine-grained control of security capabilities," *ACM Trans. Internet Technol.*, vol. 4, no. 1, pp. 60–82, 2004.

[9] J. Camenisch, "Group signature schemes and payment systems based on the discrete logarithm problem," Ph.D. dissertation, ETH Zurich, Zürich, Switzerland, 1998.

[10] J. Camenisch, M. Dubovitskaya, and G. Neven, "Oblivious transfer with access control," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, Chicago, IL, USA, Nov. 2009, pp. 131–140.

[11] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Proc. 3rd Int. Conf. Secur. Commun. Netw. (SCN)*, Amalfi, Italy, Sep. 2002, pp. 268–289.

[12] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 2004, pp. 56–72.

[13] Y. Chen, Z. L. Jiang, S. M. Yiu, J. K. Liu, M. H. Au, and X. Wang, "Fully secure ciphertext-policy attribute based encryption with security mediator," in *Proc. ICICS*, 2014, pp. 274–289.

[14] S. S. M. Chow, C. Boyd, and J. M. G. Nieto, "Security-mediated certificateless cryptography," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 3958. Berlin, Germany: Springer-Verlag, 2006, pp. 508–524.

[15] C.-K. Chu, W.-T. Zhu, J. Han, J.-K. Liu, J. Xu, and J. Zhou, "Security concerns in popular cloud storage services," *IEEE Pervasive Comput.*, vol. 12, no. 4, pp. 50–57, Oct./Dec. 2013.

[16] R. Cramer, I. Damgård, and P. D. MacKenzie, "Efficient zero-knowledge proofs of knowledge without intractability assumptions," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 1751, H. Imai and Y. Zheng, Eds. Berlin, Germany: Springer-Verlag, 2000, pp. 354–373.

[17] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Proc. EUROCRYPT*, 2002, pp. 65–82.

[18] Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 3386, S. Vaudenay, Ed. Berlin, Germany: Springer-Verlag, 2005, pp. 416–431.

[19] M. K. Franklin, in *Proc. 24th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, Aug. 2004.

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

[21] J. Han, W. Susilo, Y. Mu, and J. Yan, "Privacy-preserving decentralized key-policy attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 11, pp. 2150–2162, Nov. 2012.

[22] X. Huang *et al.*, "Cost-effective authentic and anonymous data sharing with forward security," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 971–983, Apr. 2015.

[23] J. Hur, "Attribute-based secure data sharing with hidden policies in smart grid," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2171–2180, Nov. 2013.

[24] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2271–2282, Oct. 2013.

[25] T. Jiang, X. Chen, J. Li, D. S. Wong, J. Ma, and J. Liu, "TIMER: Secure and reliable cloud storage against data re-outsourcing," in *Proc. 10th Int. Conf. ISPEC*, 2014, pp. 346–358.

[26] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proc. WPES*, 2005, pp. 61–70.

[27] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.

[28] M. Li, X. Huang, J. K. Liu, and L. Xu, "GO-ABE: Group-oriented attribute-based encryption," in *Proc. 8th Int. Conf. NSS*, 2014, pp. 260–270.

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

[30] K. Liang *et al.*, "A DFA-based functional proxy re-encryption scheme for secure public cloud data sharing," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 10, pp. 1667–1680, Oct. 2014.

[31] K. Liang, J. K. Liu, D. S. Wong, and W. Susilo, "An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing," in *Proc. 19th ESORICS*, 2014, pp. 257–272.

[32] K. Liang, W. Susilo, and J. K. Liu, "Privacy-preserving ciphertext multi-sharing control for big data storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1578–1589, Aug. 2015.

[33] J. K. Liu, M. H. Au, W. Susilo, K. Liang, R. Lu, and B. Srinivasan, "Secure sharing and searching for real-time video data in mobile cloud," *IEEE Netw.*, vol. 29, no. 2, pp. 46–50, Mar./Apr. 2015.

[34] J. K. Liu, M. H. Au, W. Susilo, and J. Zhou, "Enhancing location privacy for electric vehicles (at the right time)," in *Proc. 17th Eur. Symp. Res. Comput. Secur.*, Pisa, Italy, Sep. 2012, pp. 397–414.

[35] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Topics in Cryptology*, vol. 6558. Berlin, Germany: Springer-Verlag, 2011, pp. 376–392.

[36] M. Nabeel, N. Shang, and E. Bertino, "Privacy preserving policy-based content sharing in public clouds," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 11, pp. 2602–2614, Nov. 2013.

[37] T. Okamoto, "Receipt-free electronic voting schemes for large scale elections," in *Proc. 5th Int. Workshop Secur. Protocols*, 1997, pp. 25–35.

[38] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," in *Public Key Cryptography* (Lecture Notes in Computer Science), vol. 6571. Berlin, Germany: Springer-Verlag, 2011, pp. 35–52.

[39] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 195–203.

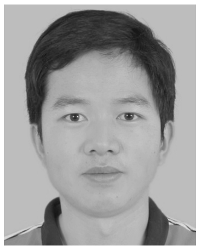
[40] M. Scott, N. Costigan, and W. Abdulwahab, “Implementing cryptographic pairings on smartcards,” in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 4249. Berlin, Germany: Springer-Verlag, 2006, pp. 134–147.
[41] S. F. Shahandashti and R. Safavi-Naini, “Threshold attribute-based signatures and their application to anonymous credential systems,” in *Progress in Cryptology* (Lecture Notes in Computer Science), vol. 5580. Berlin, Germany: Springer-Verlag, 2009, pp. 198–216.
[42] Z. Wan, J. Liu, and R. H. Deng, “HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.
[43] Y. Wu, Z. Wei, and R. H. 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.
[44] F. Xhafa, J. Wang, X. Chen, J. K. Liu, J. Li, and P. Krause, “An efficient PHR service system supporting fuzzy keyword search and fine-grained access control,” *Soft Comput.*, vol. 18, no. 9, pp. 1795–1802, 2014.
[45] C. Yao, L. Xu, X. Huang, and J. K. Liu, “A secure remote data integrity checking cloud storage system from threshold encryption,” *J. Ambient Intell. Humanized Comput.*, vol. 5, no. 6, pp. 857–865, 2014.
[46] W.-S. Yap, S. S. M. Chow, S.-H. Heng, and B.-M. Goi, “Security mediated certificateless signatures,” in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science), vol. 4521. Berlin, Germany: Springer-Verlag, 2007, pp. 459–477.
[47] T. H. Yuen, J. K. Liu, M. H. Au, X. Huang, W. Susilo, and J. Zhou, “k-times attribute-based anonymous access control for cloud computing,” *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2595–2608, Sep. 2015.

Joseph K. Liu (M’15) received the Ph.D. degree in information engineering from The Chinese University of Hong Kong, in 2004, specializing in cyber security, protocols for securing wireless networks, privacy, authentication, and provable security. He is currently a Senior Lecturer with the Faculty of Information Technology, Monash University, Australia. His current technical focus is particularly cyber security in the cloud-computing paradigm, smart city, lightweight security, and privacy enhanced technology. He has authored over 90 referred journal and conference papers and received the best paper award from ESORICS 2014 and ESORICS 2015. He has served as the Program Chair of ProvSec 2007, ProvSec 2014, ACISP 2016, and on the Program Committee of more than 35 international conferences.

Man Ho Au (M’12) received the bachelor’s and master’s degrees from the Department of Information Engineering, The Chinese University of Hong Kong, in 2003 and 2005, respectively, and the Ph.D. degree from the University of Wollongong, Australia, in 2009. He has been a Lecturer with the School of Computer Science and Software Engineering, University of Wollongong, Australia. He is currently an Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University. He works in the area of information security and applied cryptography. He has authored over 80 refereed journal and conference papers, including two papers in the ACM CCS conference that were named as Runners-up for PET Award 2009: Outstanding Research in Privacy Enhancing Technologies. His research interests include applying public-key cryptographic techniques to systems with security and privacy concerns. He has served as the Program Chair of NSS 2014 and ProvSec 2015, and as a Program Committee Member of more than 30 international conferences.

Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia. He is currently a Professor with the School of Mathematics and Computer Science, Fujian Normal University, China, and the Codirector of the Fujian Provincial Key Laboratory of Network Security and Cryptology. He has authored over 100 research papers in refereed international conferences and journals. His research interests include applied cryptography and network security. His work has been cited more than 2000 times at Google Scholar (H-index: 25). He is an Associate Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, and is on the Editorial Board of the *International Journal of Information Security* (Springer), and he has served as the program/general chair or program committee member in over 80 international conferences.

Rongxing Lu (S’09–M’11–SM’15) received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2006, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2012. From 2012 to 2013, he was a Postdoctoral Fellow with the University of Waterloo. Since 2013, he has been an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include computer network security, mobile and wireless communication security, and applied cryptography. He was a recipient of the Canada Governor General Gold Metal.



Jin Li received the B.S. degree in mathematics from Southwest University, in 2002, and the Ph.D. degree in information security from Sun Yat-sen University, in 2007. He is currently a Professor with Guangzhou University. He has been selected as one of the Outstanding Young Researchers in Guangdong Province. He has authored over 70 research papers in refereed international conferences and journals. His research interests include cloud-computing security and applied cryptography. His work has been cited more than 3500 times at Google Scholar. He has also served as the program chair or program committee member in many international conferences.