

## A COMPREHENSIVE STUDY IN WEB-BASED CLOUD COMPUTING SERVICES TO FINE-GRAINED TWO-FACTOR ACCESS CONTROL

**HIMABINDHU MARAM REDDY**

Krishna Chaitanya Institute of Technology  
& Sciences,  
Markapur, Prakasam(dist),A.P.  
[himabindumaramreddy@gmail.com](mailto:himabindumaramreddy@gmail.com)

**SK. ALIMOON**

M. Tech, (Ph. D), Asst. professor  
krishna chaitanya institute of technology &  
sciences, markapur,  
[skalimoon@gmail.com](mailto:skalimoon@gmail.com)

### 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, data storage, big data management, medical information system 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 may be 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,<sup>1</sup> 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.

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) 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 cipher texts. Within this context, cipher text-policy ABE (CP-ABE) 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 cipher text. 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, 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

cipher text (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). 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 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.

The notion of SEM cryptography was further modified as security mediated certificate less (SMC) cryptography. 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 cipher text.

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 cipher text decryption. The user is not anonymous in SMC. While in our system, the security device is controlled by the user. Anonymity is also preserved.

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

**B. Monotone Span Program**

Our access control mechanism depends on expressing the attribute predicate as a monotone span program. We review some notation about monotone span program using the notation in [35]. Let  $Y : \{0, 1\}^n \rightarrow \{0, 1\}$  be a monotone boolean function. A monotone span program for  $Y$  over a field  $F$  is an  $\times m$  matrix  $M$  with entries in  $F$ , along with a labeling function  $\rho : [1, ] \rightarrow [1, n]$  that associates each row of  $M$  with an input variable of  $Y$ , that, for every  $(x_1, \dots, x_n) \in \{0, 1\}^n$ , satisfies the following:

$$(x_1, \dots, x_n) = 1 \iff \exists v \in F^{1 \times} : vM = [1, 0, 0, \dots, 0] \text{ and } (\forall i : x_{\rho(i)} = 0 \Rightarrow v_i = 0).$$

In other words,  $Y(x_1, \dots, x_n) = 1$  if and only if the rows of Indexed by  $\{i | x_{\rho(i)} = 1\}$  span the vector  $[1, 0, 0, \dots, 0]$ . We call the length and  $m$  the width of the span program, and  $+ m$  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  $Y$ , one can use the method given in [20] to obtain the matrix  $M$ .

**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  $(p, G, G, G_T, e^\wedge) \leftarrow G(1^\lambda)$  be the public parameters as discussed. In addition, let  $g^\wedge, h, h_0, h_1, \dots, h_n \in G$  be publicly

**IV. OVERVIEW**

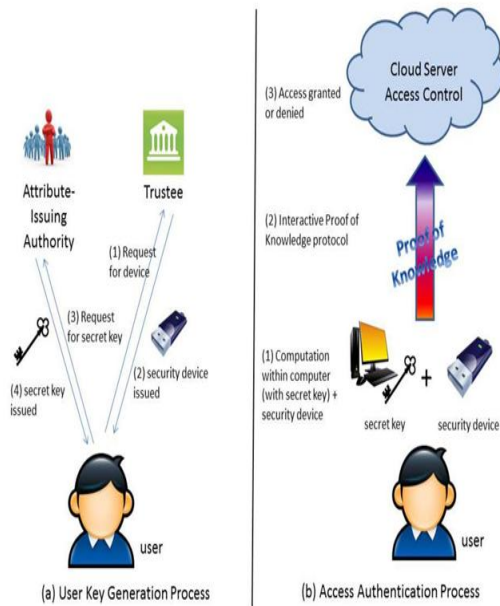
**A. Intuition**

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  $G_T$ .<sup>2</sup> All the heavy computations such as pairing are done on the computer.

The idea of our system is illustrated in Figure 1.

<sup>2</sup>The exponentiation done in group  $G_T$  is much lighter than those in group  $G$ . It can be seen from the simulation data in the benchmark.



**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 initialize 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 authorized 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 Us Ed 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
$sk_{A,UPK}$	attribute secret key for the user with attribute $A$ and public key UPK
$\Upsilon$	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  $A$  be the desired universe of attributes. For simplicity, we assume  $A = [1, n]$  for some natural number  $n$ . We will use a vector  $x \in \{0, 1\}^n$  to represent the user's attribute set. Let  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ . 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.

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

**Device Initialization:** The trustee initializes the security device for user (whose public key is UPK) with values  $TY = e^{\wedge}(g, Y)$ ,  $TG = e^{\wedge}(g, h_0)$  and  $tsk$ .

**AttrGen:** The key generation algorithm takes as input TPK, APK,  $UPK = Y$  and an attribute set  $A$  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  $P K_0$  with the attribute-issuing authority to prove the knowledge of his partial secret key  $y$ :

$$P K_0\{y : Y = h_0^y\}.$$

This proof of knowledge of discrete logarithm is straight-forward 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  $sk_{A,Y} := (A, e, s)$

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<sup>3</sup> and the security device.

**Auth:** The interactive authentication protocol takes as input TPK, APK and a claim-predicate  $Y$ . The user has some additional inputs including an attribute secret key  $sk_{A,Y}$  for attribute  $A$ ,  $USK = y$  and the security device. Assume

$$Y(A) = 1. \text{ Parse } sk_{A,Y} \text{ as } (A, e, s, x).$$

- 1) The authentication server picks at random a challenge  $R \in \mathbb{Z}_p$  and sends  $R$  to the user.
  - 2) The user computes  $C = e^{\wedge}(g, h_0) y^+ R$  and submits  $(C, y, R)$  to his/her security device.
  - 3) The security device validates  $C^{(y+R)} = TG$  and  $TG^y = TY$ .
  - 4) Upon successful validation, the security device picks a random  $r \in \mathbb{Z}_p$ , computes  $c_R = H(TG^r || R || C)$  and  $z_R = r - c_R$  tsk. It returns  $(c_R, z_R)$  to the user.
  - 5) The user converts  $Y$  to its corresponding monotone span program  $\mathbf{M} = (M_{i,j}) \in (\mathbb{Z}_p)^{m \times n}$ , with row labeling  $\rho : [1, m] \rightarrow A$ . Also compute the vector  $v = (v_1, \dots, v_m) \in \mathbb{Z}_p$  that corresponds to the satisfying assignment  $A$ . That is  $v \mathbf{M} = (1, 0, \dots, 0)$ . 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  $m$ , the user randomly picks  $a_i, t_i \in \mathbb{Z}_p$  and computes  $C_i = g^{v_i} h^{t_i}$ ,  $D_i = g^{x_{\rho(i)}} h^{a_i}$ .
- We assume the user stores both the partial secret key and attribute secret key in his/her computer.
- 7) They then engage in the following zero-knowledge proof-of-knowledge protocol.

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

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

The details of the proof of knowledge P K0 and P K1 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 any- thing 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 inter- action between the honest prover and the cheating (or honest) verifier.

#### 1) Implementation of Protocol

P K0:  $P K0\{y : Y = hy\ 0\}$ .

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  $R = hr\ 0$  to Bob. Bob returns a random challenge  $c \in \mathbb{Z}_p$ . Alice computes the response  $z = r + cy$ . Bob verifies that  $hz\ 0 = R \cdot Y\ c$ . Details of the protocol can be found in [9, Ch. 3]. For completeness, we briefly outline how P K0 provides soundness 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  $hz\ 0Y - c = hz\ 0Y - c$ . Hence the simulator can obtain  $z - z\ c - c$  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  $R = hz\ 0Y - c$ . The transcript  $(R, c, z)$  has the same distribution as those coming from Alice and Bob.

#### 2) Implementation of Protocol P K1:

Before discussing P K1, it is useful to describe the goal of P K1. The set  $\{C_i\}_{i=1}$  is the commitment of the vector v such that  $vM = (1, 0, \dots, 0)$ . In other words, the goal of P K1 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  $(y, x_1, \dots, x_n)$ . To ensure  $v_i = 0$  if and only if  $x_{\rho(i)} = 1$ , P K1 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  $C_i = D^{v_i} h^{t_i}$  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  $e^{\hat{A}}(A, \text{whe}) = e^{\hat{A}}(h^{h_0} h^{x_1} \dots h^{x_n} n^g s, h)$  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+.

#### 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  $U = \{U_1, \dots, U_n\}$  be a set of users. We use  $(Y_i, y_i)$ ,  $A_i$ ,  $sk(i)$ ,  $A_i, Y_i$ ,  $token_i$  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  $(sk(i), A_i, Y_i, token_i)$  is given to the attacker. In addition, the attacker may have  $(Y_i, y_i, sk(i), A_i, Y_i)$  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  $O_i$  with the following behaviour:

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

We allow the attacker to specify the security device for revocation. If a security device  $token_i$  is revoked, oracle  $O_i$  will no longer be available. We further assume the claim-predicate  $Y$  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  $Y$  if for all  $i$  such that  $U_i$  is controlled by the attacker,  $Y(A_i) = 1$  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  $F$  against our scheme, there exists a simulator  $S$ , having blackbox access to  $F$ , 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  $S$  under the assumption that attacker  $F$  exists. We utilise the fact that  $P_{K0}$  and  $P_{K1}$  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  $p, G, GT, e^{\hat{}}$  be a bilinear group with  $g \in G$  being a generator. Let  $g^{\hat{}}$ ,  $h, h_0, h_1, \dots, h_n$  be additional generators of  $G$ . We use  $TG$  to denote  $e^{\hat{}}(g, h_0)$ . We further assume full domain hash  $H : \{0, 1\}^* \rightarrow Z_p$  which is modelled as a random oracle.

**Table II: Time And Space Cost On Different Platforms**

	Computer	Smartphone	Smartcard [40]
$P$	6.4	32	-
$E_1$	2.5	13	-
$ET$	0.4	2.2	200
$Z_p$		160	
$G$		512	
$GT$		1024	

**Table III: Test Platforms**

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

implies  $a = c$  and  $b = d$  under the discrete logarithm assumption. Consider the last two relations from  $P_{K1}$ :

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

**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	$3GT + (5\ell + 2)G + (4\ell + m + n + 12)Zp$

2) Privacy: In every authentication, the information obtained by the server consists of two parts, namely, (C, cR, zR, C1,...,C, D1,..., D) and the verifier's view of P K1. Note that P K1 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. cR, zR leaks no information since they are distributed identically for all legitimate users. C1,...,C, D1,..., D are information-theoretic secure commitment and again leak no information. The only component containing information about the user is  $C = \hat{e}(g, h_0) 1 y + R$ . This is the verifiable random function with seed y on input R and is computationally indistinguishable from a random value in GT.

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 G and group GT respectively. The symbol Z p, G, GT represents the size of an element (in bits) in Zp, G and GT respectively. 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:

```

8B          A2A5229B
D9C57CFC8ACEC76DFDBF3E
3E1952C6B3193ECF5C571FB502FC5DF
4
10F9267E9F2A605B
B0F76F52A79E8043
B          F4AF0E          F2E9F
A78B0F1E2CDFC4E8549B
  
```

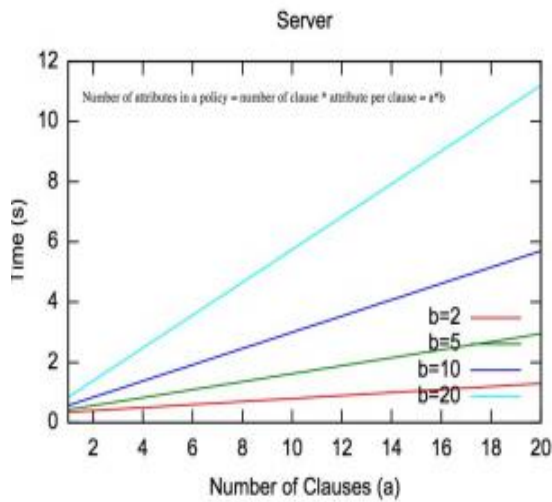
The elliptic curve is defined by the equation  $y^2 = x^3 + 1 \pmod q$ . The group G (as well as GT) is of order  $p = 80000000000000000000000000000000020001$ , where p is a 160-bit prime. The pairing is Tate pairing. Table IV listed the number of operations and communication for an authentication transaction. Recall that n is the size of the attribute universe, 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  $A = \{1, \dots, 100\}$ . In the following we estimate the efficiency of our system using policy of the following format:

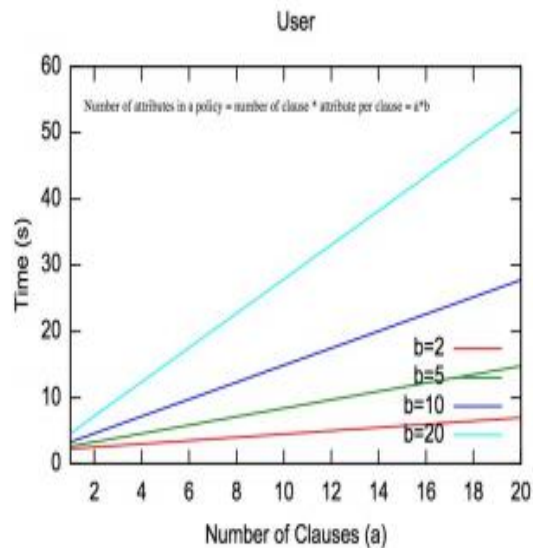
$$\bigvee_{i=1}^a \left( \bigwedge_{j=1}^b (\text{attr}_{i,j}) \right),$$

where attr<sub>i,j</sub> maybe re-used in different clauses. In general, this kind of policy can be represented by a span program of length = 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 about five times slower due to the use of a less powerful computing device (a smartphone). One should note that the

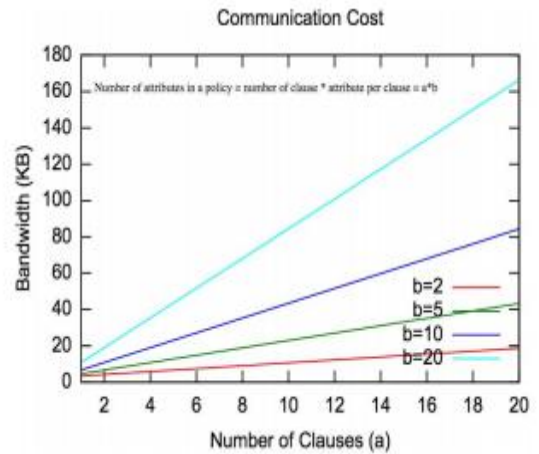
security device is not the bottleneck as it only accounts for a constant time cost of 0.6 seconds. Please refer to Fig. 3 for the time complexity at the user side. The total authentication time for a policy with 100 attributes, arranged as 10 clauses with 10 attributes each, is about 18 seconds. The 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.



**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).**

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  $g^{xh}$  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.

## 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.