Fine-Grained Two-Factor Access Control For web-Based Cloud Computing Services Using OTP

R.V.Pandian[1], K.Narayana[2], B.Srinivasulu[3]
[1]Dept of CSE
[2]Head of the Department, Dept of CSE
[3]Assistant professor, Dept of CSE
[1,2,3]Seshachala Institute of Technology,Puttur.

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 user secret key and a lightweight security device. As a user cannot access the system if s/he does 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.*

Keywords- 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 pro-vides great advantages, there are meanwhile also con-cerns 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. How-ever, it is well acknowledged that privacy is an essential feature that must be considered in cloud computing sys-tems. 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. 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.

1.1 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 secu-rity. First the user secret key (which is usually stored inside the computer) is required. In addition, the secu-rity 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 ac-cess 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.

2.1 Attribute-Based Cryptosystem

Attribute-based encryption (ABE) is the corner-stone 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) 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 en-crypted 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 mes-sage 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 signa-ture (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. proposed an attribute-based access control mechanism which can be regarded as the interactive form of ABS.

2.2 Access Control with Security Device

2.2.1 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 me-diator 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.

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 algo-rithm, 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 prob-lem. 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.

2.2.2 Key-Insulated Cryptosystem

The paradigm of key-insulated cryptography was intro-duced in. 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 cryptosys-tem 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.

3.1 Pairings

Let G and G_T be cyclic groups of prime order p. A map e^ : G G ! G_T is bilinear if for any generators g 2 G and a; b 2 Z_p, e^(g^a; g ^{b}) = e^(g; g)^{ab}. Let G be a pairing generation algorithm which takes as input a security parameter 1 and outputs (p; G; G; G_T ; e^) G(1). The generators of the groups may also be given. All group operations as well as the bilinear map e^ are efficiently computable.

3.2 Monotone Span Program

Our access control mechanism depends on expressing the attribute predicate as a monotonespan program. Let : f0; $1g^n$! f0; 1g be a monotone boolean function. A monotone span program for over a field F is an ` m matrix **M** with entries in F, along with a labeling function : [1; `] ! [1; n] that associates each row of **M** with an input variable of , that, for every (x_1; : : : ; x_n) 2 f0; $1g^n$, satisfies the following:

(x_1; : : : ; x_n) = 1 ()9~v 2 $F^{1\ `}$: ~vM = [1; 0; 0; : : : ; 0] and (8i : $x_{(i)}$ = 0) v_i = 0):

In other words, (x_1; : : : ; x_n) = 1 if and only if the rows of **M** indexed by fijx $_{(i)}$ = 1g span the vector [1; 0; 0; : : : ; 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 , one can use the method given in to obtain the matrix **M**.

3.3 BBS+ Signatures

We briefly review a signature scheme called BBS+. It belongs to a class of signature schemes, commonly known as CL-signatures. 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, which is based on the schemes of Camenisch and Lysyanskaya and of Boneh et al. It is also referred to as credential signatures as it is normally used to certify a set of credentials.

IV. OVERVIEW

4.1 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

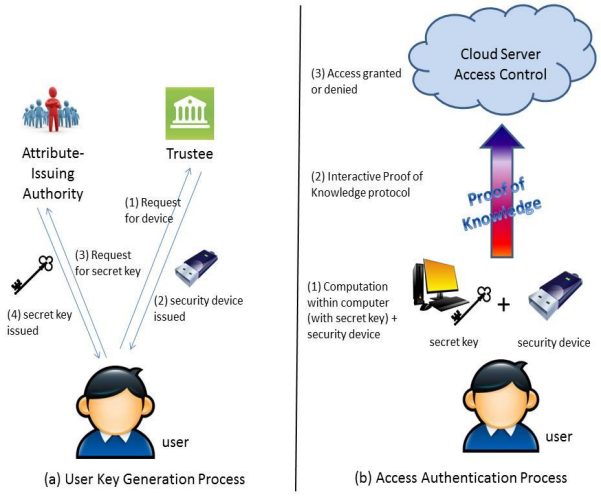

Fig. 1: Overview idea of our system

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 relation-ship 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 exponentia-tion over group G_T [2]. All the heavy computations such as pairing are done on the computer.

4.2 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 gen-erate 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.

4.3 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 com-municates with the cloud service provider through an anonymous channel [37], [26] or uses IP-hiding technol-ogy. 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.

4.4 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 fStudent; Physicsg 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 se-curity 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.

V. OUR PROPOSED SYSTEM

5.1 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 se-curity device is not accessible nor

modifiable once it is initialized. In addition, it will always follow the algorithm specification.

Capability. It is capable of evaluation of a hash function. In addition, it can generate random num-bers and compute exponentiations of a cyclic group defined over a finite field

5.2 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 2 f0; $1g^n$ to represent the user's attribute set. Let ~x = (x_1; : : : ; x_n) 2 f0; $1g^n$. If the user is in possession of attribute i, x_i = 1. Otherwise, x_i = 0.

5.2.1 System Setup

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

TSetup: Let be a security parameter. The trustee

runs G(1) (described in Section 3.1) to generate param = (G; G_T ; p; e^) and randomly picks generators g; g;^ h; h_0; h_1; : : : ; h_n 2 G. It also picks a collision re-sistant hash function H : f0; 1g ! Z_p. Further, let tpk = e^(g; h_0)^{tsk} for a randomly generated tsk 2_R Z_p.
It publishes TPK = (param; g; g;^ h; h_0; h_1; : : : ; h_n; H; tpk).

ASetup: The attribute-issuing authority randomly picks

2 Z_p and computes w = h . It publishes APK = (w) and sets ASK = ( ).

5.2.2 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.

USetup: The user randomly picks y 2 Z_p. It publishes UPK = Y = $h^y{}_0$ and sets USK = y.

Device Initialization: The trustee initializes the security device for user (whose public key is UPK) with values
TY = e^(g; Y), TG = e^(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; : : : ; x_n) 2 f0; $1g^n$.

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_0fy : Y = $h^y{}_0$g:

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 2 Z_p and uses his secret key ASK to create the user attribute secret key $sk_{A;Y}$:= (A; e; s) as
A = (hY h^{x1} h^{xn} g^^{s}) ^{1} —
+e
1 n

5.2.3 Access Authentication

The access authentication process is an interactive pro-tocol 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.
Auth: The interactive authentication protocol takes as input TPK, APK and a claim-predicate . The user has some additional inputs including an attribute secret key $sk_{A;Y}$ for attribute A, USK = y and the security device. Assume (A) = 1. Parse $sk_{A;Y}$ as (A; e; s; ~x).

1) The authentication server picks at random a chal-lenge R 2 Z_p and sends R to the user.

2) The user computes C = e^(g; h_0) $^{1/(y+R)}$ 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 2_R Z_p, computes c_R = H(TG^{r}jjRjjC) and z_R = r $c_R$tsk. It returns (c_R; z_R) to the user.

5) The user converts to its corresponding monotone span program M = ($M_{i;j}$) 2 $(Z_p)^{`\times m}$, with row labeling : [1; `] ! A. Also compute the vector ~v = (v_1; : : : ; $v_`$) 2 $Z^`_p$ that corresponds to the satisfying assignment A. That is ~vM = (1; 0; : : : ; 0). Note that if $x_{(i)}$ = 0 (i.e., the user does not possess the attribute (i)), v_i must be 0).

6) For i = 1 to `, the user randomly picks a_i; t_i 2_R Z_p and computes C_i = g^{vi} h^{ti} , D_i = $g^{x_{(i)}}$ h^{ai} . The user also computes b_i = t_i $a_i$$v_i$.

P`

7) For j = 1 to m, the user computes $f_j = \sum_{i=1} t_i M_{i;j}$. Then the user sends (C, c_R, z_R, C_1, : : :, $C_{\cdot}$, D_1, : : :, $D_{\cdot}$) to the authentication server.

They then engage in the following zero-knowledge

5.3 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) = fy : (x; y) 2 Rg and the language L_R = fx : 9y such that (x; y) 2 Rg. If (x; y) 2 R, we call y the witness of x.

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

Completeness: If (x; y) 2 R, the honest prover who knows witness y for x succeeds in

1) convincing the honest verifier of his knowledge.

Soundness: If (x; y) 2= R, no cheating prover can convince the honest verifier that (x; y) 2 R, except with some small probability. It can be captured by the existence of a knowledge extractor E to extract the witness y: given

1) 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) 2 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.

5.3.1 Implementation of Protocol PK_0

PK_0fy : Y = $h^y{}_0$g:

Suppose Alice wants to prove the knowledge of y to Bob. Alice picks a random number r 2 Z_p and sends the commitment R = $h^r{}_0$ to Bob. Bob returns a random challenge c 2 Z_p. Alice computes the response z = r+cy. Bob verifies that $h^z{}_0$ = R Y ^{c}. Details of the protocol can be found in Chapter 3 of [9]. 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^0; z^0). Since both of them are valid, it means that

$_hZ_{0Y}$ $_c$ = $_hZ_0{}^0{}_Y$ $_c{}^0$:

Hence the simulator can obtain $\frac{z}{c} \frac{z}{c0}{}^{0}$ as the solution of $\log_{h0}$ Y . (Zero-knowledge) Given the public key h_0; Y , the simulator can randomly picks c; z 2 Z_p and computes R = $h^z{}_0$Y ^{c}. The transcript (R; c; z) has the same distri-bution as those coming from Alice and Bob.

5.3.2 Implementation of Protocol PK_1

Before discussing PK_1, it is useful to describe the goal of PK_1. The set $\{C_i\}_{i=1}$ is the commitment of the vector ~v such that ~vM = (1; 0; : : : ; 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 (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; : : : ; x_n). To ensure v_i 6= 0 if and only if $x_{(i)}$ = 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 $D_i = g^{x_{(i)}} h^{a_i}$ 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

$e\hat{}(A; wh^e) = e\hat{}(hh^{y}{}_0 h^{x}{}_1{}^1 : : : h^{x}{}_n{}^n g\hat{}^s; h)$

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 cer-tified. We shall elaborate how this could be conducted based on the signature verification protocol of BBS+.

The final two relations mean that ~vM evaluates to

(1; 0; : : : ; 0).

Now we are ready to describe the implementation of PK_1. The prover first randomly generates k_1; k_2 2_R Z_p, computes $A_1 = g^{k1} h^{k2}$, $A_2 = Ah^{k1}$, $_1 = k_1 e$, $_2 = k_2 e$, and conducts the following proof.

5.5 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 4. The symbols P , E1, ET represent the time cost (in ms) of a pairing operation, an exponentiation in group G and group G_T respectively. The symbol Zp, G, GT represents the size of an element (in bits) in Z_p, G and G_T respectively.

We consider three different platforms, namely, a com-puter, a smart phone and a smart card.

TABLE 2: Time and space cost of 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 3: 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

For the time cost on a smartcard, we use the bench-mark result from [40]. The configuration of our platforms are as follows.

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:

8BA2A5229BD9C57CFC8ACEC76DFDBF 3E

3E1952C6B3193ECF 5C571FB502FC5DF 4

10F 9267E9F 2A605BB0F 76F 52A79E8043

BF 4AF 0EF 2E9FA78B0F 1E2CDFC4E8549B

The elliptic curve is defined by the equation $y^2 = x^3 + 1 \bmod q$. The group G (as well as G_T) is of order p=8000000000000000000000000000000000020001, where p is a 160-bit prime. The pairing is Tate pairing. Table 4 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.

5.5.1 Simulation

Assume the total number of attributes in the system is 100. In other words, the attribute universe A = f1; : : : ; 100g. In the following we estimate the efficiency of our system using policy of the following format:

$$\underline{@}_{i=1}^{0} \wedge_{j=1}^{1} (attr_{i;j})^{A};$$

where $attr_{i;j}$ maybe re-used in different clauses. In gen-eral, this kind of policy can be represented by a

span program of length ` = a b and width m = a (b 1)+1.

TABLE 4: Authentication Complexity

TABLE 4: Authentication Complexity

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

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, consist-ing 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.

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 expo-nentiations 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)

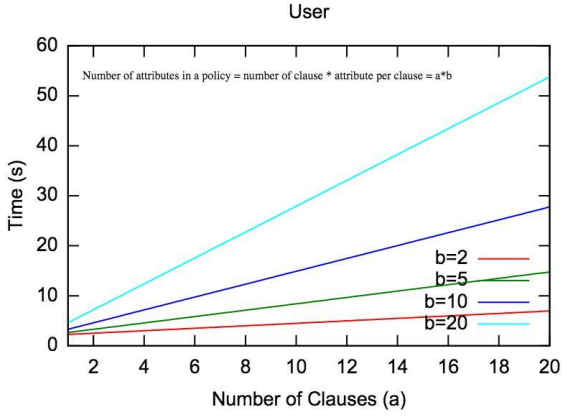

Fig. 3: Running time of the Auth protocol (User side) (s)

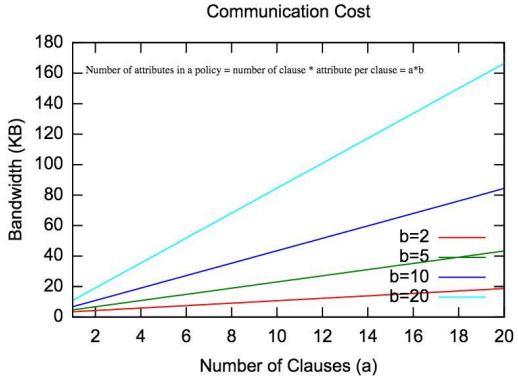

Fig. 4: Communication cost of the Auth protocol (KB)

Communication cost of the Auth protocol (KB) 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 secu-rity analysis shows that the proposed 2FA access con-trol 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 T. Yu, G. Danezis, and V. D. Gligor, editors, the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012, pages 929– 940. ACM, 2012.

[2] M. H. Au, A. Kapadia, and W. Susilo. Blacr: Ttp-free blacklistable anonymous credentials with reputation. In NDSS. The Internet Society, 2012.

[3] M. H. Au, W. Susilo, and Y. Mu. Constant-Size Dynamic k-TAA. In SCN, volume 4116 of Lecture Notes in Computer Science, pages 111–125. Springer, 2006.

[4] J. Baek, Q. H. Vu, J. K. Liu, X. Huang, and Y. Xiang. A secure cloud computing based framework for big data information man-agement of smart grid. IEEE T. Cloud Computing, 3(2):233–244, 2015.

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