A General Formal Framework of Analyzing Selective Disclosure Attribute-Based Credential Systems

Caimei Wang[1,2], Yan Xiong[1], Wenjuan Cheng[3], Wenchao Huang[1], Huihua Xia[1], Jianmeng Huang[1]

(Corresponding author: Wenjuan Cheng)

School of Computer Science, University of Science and Technology of China[1]
Elec-3 (Diansan) Building, West Campus of USTC, Huang Shan Road, Hefei, Anhui Province, China
Department of Computer Science, Hefei University[2]
Building 38, No.99, Jinxiu Avenue, Hefei, Anhui Province, China
School of Computer and Information, HeFei University of Technology[3]
193, Tunxi Road, Hefei, Anhui
(Email: cheng@ah.edu.cn)

(Received Nov. 16, 2016; revised and accepted Feb. 21 & Mar. 11, 2017)

Abstract

A selective disclosure attribute-based credential system (SDABCS) can provide a communication mechanism to protect both security and privacy in electronic communication, by issuing a kind of credential with attributes, which the user can disclose parts of attributes. We present a general framework for formally verification of SDABCS with applied Pi calculus, and provide three definitions of relevant security properties. The framework can implement secure communication among the user, service provider and trusted authority. Two important functions are implemented: the first allows the user to receive a credential encoded a list of attributes from a trusted authority; the second allows the user to convince a service provider with the credential. Particularly, the user can selectively reveal parts of the attributes according to the needs of service provider, while not revealing the rest of the attributes. In our experiments, we apply the framework to a concrete security protocol and successfully prove three security properties in the protocol using ProVerif.

Keywords: Attribute-based Credential; Formalize; General Framework; Selective Disclosure

1 Introduction

A selective disclosure attribute-based credential system [11, 17, 22] (SDABCS) allows a service provider to identify a user with a credential, which contains a list of attributes, from an authority. One of the most important characteristics in this system is that the user can selectively disclose parts of the attributes in the credential to different service providers according to practical requirements. For the other undisclosed attributes, the user can generate cryptographic commitments to encode attributes which can be verified without revealing the attribute information. Authentication is one of the potential applications of this system. For example, *Alice* gets a credential C_{cre} with several attributes from an authority and generates two different presentation proofs, P_1 and P_2, according to different attributes. Then she can access services provided by provider $S1$ under P_1, and by $S2$ under P_2, while no one can find that the two presentation proofs belong to the same user.

Although a lot of scholars study SDABCS [8, 14], so far there is no an effective formal verification framework to verify the correctness of the system and the related security properties. Actually, organizations and individuals are now paying more attention to the design of secure system. The formal analysis is a state-of-the-art method used to analyze security protocols and systems. There are many researches on the formal analysis and verification of security protocols [2, 4, 19, 23]. Li et al. propose a general symbolic model for anonymous credential protocols and make formal definitions of a few critical security properties [13]. Shao et al. conduct a formal analysis and verification of the enhanced authorization mechanism under TPM 2.0 API [20]. In [3], zero knowledge proof is applied to a simplified Direct Anonymous Attestation protocol, which enables remote authentication of *TPM* while preserving the user's privacy at the same time. They present security proof and report a novel attack. However, existing formal work considers neither the feature of selectively

disclosed attributes nor the credential protected by hardware.

In order to carry out an analysis of SDABCS, we propose a general framework of formally verifying the SDABCS. In our framework, a user can authenticate herself to a service provider by disclosing parts of her attributes, while preserving the privacy of the undisclosed ones. All attributes are encoded in the credential issued by an authority. In order to increase the security, we consider the situation that the credential is protected by the security hardware. The complexity of the system makes the formalized process involve many parameters. As a result, it is challenging to reasonably construct all the functions and equations with applied Pi calculus [9].

As a case study, we formally verify an innovative cryptographic protocol named U-Prove protocol [15, 18], whose SDK has been released by Microsoft and then integrated into a range of its own identity products. We first give the detailed description of the formal analysis, then prove the authenticity properties by setting correspondence assertions and prove the untraceability of U-Prove token by observational equivalence using ProVerif.

To the best of our knowledge, this is the first formal and automated verification of the SDABCS.

Contributions. The contributions of this paper are threefold: first, we put forward a general model framework of a SDABCS in applied Pi calculas, and provide three formal definition of security properties. It is meaningful to reduce the workload of verification of attribute-based credential system. Second, we apply our framework to the U-Prove protocol and give the formalized description of it. Finally, we prove that the protocol satisfies the relevant security properties by Proverif.

Outline of the Paper. In Section 2, applied Pi calculus is reviewed. In Section 3, we propose our general model framework of the SDABCS and the definitions of relevant security property. In Section 4, after detailed analysis of the U-Prove protocol, we apply the framework to verify the protocol, and successfully prove the three security properties using Proverif. In Section 5, conclusion and future work are presented.

2 Review of Applied Pi Calculus

2.1 Syntax and Semantics

The applied Pi calculus [3, 12, 21] is a language for describing and analyzing security protocols. Now we briefly review the syntax and operational semantics of the applied Pi calculus, and define the additional notation used in our paper. The syntax of the applied Pi calculus is given in Table 1.

The syntax of the calculus is composed of terms and processes. Terms are defined by a signature Σ which consists of a finite set of function symbols with arbitrary arity. Every function symbol means a primitive used by security protocols, and all symbols are divided into two finite sets of constructor and destructor symbols. Both two symbols

Table 1: Syntax of the applied Pi calculus

$M, N, F, Z ::=$	Terms
$s, k, ..., a, b$	names
x, y, z	vars
$f(M_1, ..., M_k)$ $f \in \Sigma$ and k is the arity of f	function
$P, Q ::=$	Processes
0	null process
$P \mid Q$	parallel composition
$vn.P$	name restriction
$!P$	replication
$u(x).P$	message input
$\overline{u}\langle N\rangle.P$	message output
if $M = N$ *then* P *else* Q	conditional

are built from an infinite set of names, and an infinite set of variables, e.g., encryption function, decryption function, digital signature function, etc. Usually, constructors are used to generate terms which model primitives used by protocols, while destructors are used to handle terms generated by constructors.

A process is a set of programs, which are connected via channels. That means two processes can communicate with each other by such channels. The executing program is known as the active program. An active program can execute operation and send messages to another program. A process or an extended process with a hole is called context, marked as $C[\_]$.

In Pi calculus, the grammar of processes is defined as follows: the null process 0 does nothing; $P \mid Q$ is the parallel composition of precess P and Q, used to express participants of a protocol running in parallel; the process $vn.P$ is used to produce a fresh name n and then behaves as P, in which the restricted name n is binded inside P; replication process is the infinite compositions $P|P|\ldots$, which means there are infinite copies of P running in parallel; in $u(x).P$ process, a message x can be received from the channel u, then the process behaves as P; on the contrary, in $\overline{u}\langle N\rangle.P$ process, a message N can be sent to the channel u, and then process behaves as P; the conditional process $if\ M = N\ then\ P\ else\ Q$ means that when equation holds process behaves as P, otherwise Q, when Q is $null$, we always abbreviate it as $if\ M = N\ then\ P$.

A protocol P consists of a set of agents and channels, agents can communicate over the channels. Every agent runs a set of programs, and a program is an honest program only if it follows the protocol.

We equip the terms with an equational theory E, that is a finite set of equations of the form $M = N$ where $M, N \in \Sigma$.

2.2 Security Properties

This section presents two security properties that will be used in our paper.

Correspondence property is usually used to prove the

authentication of the participants in many protocols. We proof it by setting the events in protocol at different stages, and then verify identity according to the events occur successively relationship. So we can annotate processes with a set of events $\{e_1, e_2, \ldots e_n\}$ in a running protocol.

Definition 1 (**Correspondence property**). *A correspondence property is used to express relationship between events with the form:*

$$e_i < e_j \quad where\ i < j, \quad i, j \in \{1, 2, \ldots, n\}$$

The property means that if an event e_i has happened then event e_j must have happened previously. Authentication can be captured as a correspondence property.

Zero-knowledge proof regarded as an important and most basic technology will be used in our system. Anyone can use zero-knowledge proof to communicate with other and convince the latter the given statement is true, without conveying any more information in addition to the statement. A formal definition of non-interactive zero-knowledge has been devised in [3].

Observational equivalence is the property that two or more underlying entities are indistinguishable on the basis of their observable implications. We define Observational Equivalence as the following description.

Definition 2 (**Observational Equivalence**). *If there are two terms M and N satisfy C[M] and C[N] are both valid terms with the same value in all contexts C[_], then it is not possible, within the system, to distinguish the two terms. We call the observation equivalence as the largest symmetric relation between M and N.*

3 General Model Framework of SDABCS

3.1 Modelling the Roles in Framework

A general model on the verification of a SDABCS allows a user to receive a credential with a list of attributes from a trusted party. This credential can be used to convince a service provider. The core feature of the system enables a user to select disclosable attributes, and generates the verifiable presentation proof for the undisclosed attributes, then the Verifier finishes the verification work.

There are three types of agents in our model: Issuer, Prover, and Verifier. The role of Issuer is an authority, who can generate and issue the credentials with a list of attributes containing an unforgeable digital signature by applying its private key I_{sk}. The role of Prover is a person in possession of a credential encoded a list of attributes, which can be optionally protected by a security hardware. Prover can control which attributes are revealed, and which attributes are generated presentation proof according to the needs of different service providers. The role of Verifier is a person who verifies the Issuer's signature in the credential, and the presentation proof of attributes generated by the Prover.

We set a process for each role in our model, and assume that the system is executed in a public network, where attackers can listen to, delete, forge and send all messages in the network, following the so-called Dolev-Yao model [10].

Prover process. When a Verifier relying on authentication or other identity-related attributes communicates with a Prover, the Prover must first demand a credential with attributes signed by an Issuer which is trusted by the Verifier. Then the Prover provides necessary attributes, and generates a cryptographic presented proof for the undisclosure attributes. Hence a Prover process can be defined as in Equation (1).

$$\begin{aligned} ProverP \stackrel{def}{=}\ & in\ (c, IP).\ (P^{Ini}(CI, C_{sk}, C_{pk}) \\ & \mid P^{getsig}(CI, I_{pk}) \mid P^{getp}(CI, AUp, ACp)) \end{aligned} \tag{1}$$

c models the public channel which is used to transmit all kinds of messages. IP models an unique identifier for the Issuer parameters including Issuer's public key. CI models an unique identifier of the credential. The process $P^{Ini}(CI, C_{sk}, C_{pk})$ models the Prover's behavior of generating the private key C_{sk} and the corresponding public key C_{pk} for the credential CI. The process $P^{getsig}(CI, I_{pk})$ models the Prover's behavior of getting the signature from Issuer I_{PK} using the blind operation to the credential. The process $P^{getp}(CI, AUp, ACp)$ models the Prover's behavior of generating presentation proof for the credential CI. AUp models the proof of undisclosed attributes, and ACp models the proof of needing to submit information commitment attributes. So presentation proof contained several parts as in Equation (2).

$$\begin{aligned} P^{getp}(CI, AUp, ACp) \stackrel{def}{=}\ & P^{undisp}(CI, \\ & AU)\ .\ P^{commitp}(CI, AC)\ .\ P^{TPMp} \end{aligned} \tag{2}$$

where $P^{undisp}(CI, AU)$ models the Prover's behavior of generating presentation proof for undisclosed attributes AU in credential CI, and $P^{commitp}(CI, AC)$ models the behavior of generating presentation proof. If the token is protected by TPM 2.0, then P^{TPMp} models the Prover's behavior of generating presentation proof for TPM 2.0.

Issuer process. Correspondingly, an Issuer process consists of initializing issuer parameter modelled by the process $I^{Ini}(I_{pk}, IP)$, and signing the credential with specific attributes modelled by the process $I^{sign}(IP, CI)$. The process $I^{Ini}(I_{pk}, IP)$ models the behavior of generating issuer parameters by Issuer I_{pk}, an application-specific unique identifier for the issuer parameters is denoted as IP. The process $I^{sign}(I_{pk}, IP, CI)$ models Issuer I_{pk} signing the credential CI under IP. An Issuer process can be defined as in Equation (3).

$$IssuerP \stackrel{def}{=}\ I^{Ini}(I_{pk}, IP) \mid I^{sign}(IP, CI) \tag{3}$$

Verifier process. Verification of a credential by a Verifier is made up of four parts. So the verifier needs to do: firstly, verify the signature of credential CI from

the Issuer I_{pk}, which is modelled by $V^{Isig}(CI, I_{pk})$. Secondly, verify the proof of undisclosed attribute, which is modelled by $V^{undisp}(AUp, C_{pk})$. Thirdly, verify the proof of the committed attributes, which is modelled by $V^{commitp}(ACp, C_{pk})$. Lastly, if token is protected by the TPM 2.0, verify the proof of the TPM 2.0, which is modelled by V^{TPMp}. An Verifier process can be defined as in Equation (4).

$$verifierP \stackrel{def}{=} V^{Isig}(CI, I_{pk}).\ V^{undisp}(AUp, C_{pk}).\ V^{commitp}(ACp, C_{pk}).\ V^{TPMp} \quad (4)$$

3.2 Modules

According to the different functions implemented by the roles, the system needs to complete two modules. One is denoted as issuance module, the other is denoted as presentation proof module. A Prover can retrieve a credential encoded any kinds of attributes from an Issuer in an issuance module, and generate a presented proof for undisclosed attributes in the credential to a Verifier in a presentation proof module.

We model the issuance module process as an unbounded number of Prover processes and trusted Issuer processes running in parallel. The presentation proof module process is modelled as an unbounded number of Prover processes and Verifier processes running in parallel.

Definition 3 (Issuance module)**.** *An Issuance Process is made up of four sub-processes, including the initialization process run by the Prover, the initialization process run by the Issuer, the process of producing and issuing credential run by the Issuer, the process of getting and blinding credential run by the Prover. Due to the process of blind in this protocol the Issuer never sees his own digital signature on the credential. The composition of the processes in applied pi shows as Equation (5).*

$$IP \stackrel{def}{=} v\widetilde{n}.\ (P^{Ini}(CI, C_{sk}, C_{pk}) \mid P^{getsig}(CI, I_{pk}) \mid I^{Ini}(I_{pk}, IP) \mid I^{sign}(IP, CI)) \quad (5)$$

The restricted name $\widetilde{n}$ models the secrets shared between the Provers and the Issuer. The rest of the parameters have the same meaning as described elsewhere.

Definition 4 (Presentation Proof module)**.** *A Presentation Proof Process is made up of two sub-processes, including the presentation proof generation run by the Prover and the presentation proof verification run by the Verifier. The composition of the processes in applied pi shows as Equation (6).*

$$showP \stackrel{def}{=} in(c, sig_t).P^{getp}(CI, AUp, ACp) \mid in(c, < AUp, ACp, TPMp > P^{verifierP} \quad (6)$$

The restricted c models the public channel. The rest of the parameters means the same above.

3.3 Events and Properties

As mentioned in a lot of papers [1, 5], correspondence assertions can be used to prove many trace-based security properties among events. Here we summarize a set of events which can be used later.

- **PCreInf**(CI, Cre_{sk})**:** where CI is the value of the credential information field. It is used to encode Credential-specific information that is always disclosed to Verifier, such as a validity period, credential usage restrictions and so on. Cre_{sk} is the private key of the credential, which is generated in the issuance protocol and should be kept secret. This event is executed after the Prover to generate the private key of the credential with the value of CI in the P^{Ini} process.

- **IissueCre**(CI, I_{sk})**:** where CI is the same as above. I_{sk} is the private key of the Issuer. This event is executed before the Issuer sends credential with the value of CI signed by himself.

- **PgetCre**(Cre_{pk}, I_{pk})**:** where Cre_{pk} is the public key of a credential corresponding to its private key Cre_{sk}. I_{pk} is the public key of the Issuer corresponding to its private key I_{sk}. This event is executed after the Prover to generates presentation proof for the credential with Cre_{pk}.

- **VerifiedCre**$(Cre_{pk}, proof, I_{pk})$**:** where Cre_{pk}, $proof$ and I_{pk} are the same as above. $proof$ is one of the parameters needs to be proved. This event is executed after successful validation by the Verifier.

According to the definition of Section 2.2, we provide three definitions of basic security properties which the general model framework needs to meet.

Definition 5 (Authenticity of the Issuance Protocol)**.** *Given processes as follows.*

$$< P^{getsig}(CI, I\_pk) \mid I^{sig}(CI, I\_sk) >$$

authenticity of the Issuance Protocol is satisfied if the process satisfies the following property:

$$\forall (CI, I\_sk, I\_pk), \exists \{PgetCre(CI, I\_pk) \Longrightarrow IissueCre(CI, I\_sk)\}$$

This property shows that when a Prover gets a token signed by an Issuer with public key I_pk, the Issuer has actually signed with the corresponding private key I_sk and issued that credential.

Definition 6 (Authenticity of the Presentation Protocol)**.** *Given processes as follows.*

$$P^{getsig}(CI, I_{pk}) \mid P^{verifierP}$$

authenticity of the Presentation Protocol is satisfied if the process satisfies the following correspondence property.

$$\forall(CI, Cre\_pk, Cre\_sk),$$
$$\exists\{VerifedCrek(Cre\_pk, proof, I\_pk) \Longrightarrow$$
$$PCreInf(CI, Cre\_sk)PgetCre(Cre\_pk, I\_pk)\}$$

This property shows that if there is a Verifier who can complete the validation of *proof* signed by the Issuer and Cre_sk, we can say not only an Issuer with public key I_pk has actually signed and issued that credential with the corresponding private key I_sk, but also the Prover has got that credential.

Definition 7 (Untraceability of Credential)**.** *Assume the same attributes CI encoded into two different credentials (CI_1, CI_2) by the Prover, if the issuer cannot identify two credential from the same user, we call this feature as untraceability.*

This property shows that when a user provides two credentials issued by an issuer, the issuer cannot judge whether the two certificates are from the same user.

4 Case Study: U-Prove Protocol

4.1 Detail of the U-Prove Protocol

U-Prove is an innovative cryptographic technology, its core is a U-Prove token [18] with any type of application-specific attributes. There are three types of angents in the protocol: *Prover* (P), *Issuer* (I), *Verifier* (V). Each U-Prove token can be seen as a credential of a *Prover* (P). It can be optionally protected by a trusted hardware such as security chip TPM 2.0. Its prototype has been first introduced in [7] by Liqun Chen. The trusted hardware in this paper refers to TPM 2.0. In the rest of the paper, we use the following notation: Let $G_q =< g >$ be a cyclic group of prime order q and g be a generator.

There are two sub-protocols in U-Prove protocol: issuance protocol and presentation proof protocol. The former mainly generates and issues a U-Prove token. The latter mainly produces and verifies a presentation proof about the undisclosed attributes in the U-Prove token.

The Issuance Protocol.

Assume I and P agree on the application-specific attributes $(A_1, \ldots, A_n)$, the value of the token information field TI is used to encode token-specific information. P wants to get a U-Prove token with attributes A_i, $i \in (1, \ldots, n)$ from I, both parties must communicate with each other under the issuance protocol.

The protocol consists of the following steps:

1) I generates the issuer parameters denoted as IP.

$$IP = \{(g_0, g_1, \ldots, g_n, g_t), (e_0, e_1, \ldots, e_n)\}$$

where $(g_0, g_1, \ldots, g_n, g_t)$ represents I's public key and satisfies the equation $g_0 = g^{y_0}$, which y_0 is I's private key. The rest of g_i values must be random generators of G_q.

2) I and P complete the precomputation respectively: $\gamma = (g_0 g_1^{x_1} \ldots g_n^{x_n} g_t^{x_t}\ h_d)$, where $(x_1, \ldots, x_n)$ are the operation results, according to the values in the list of $(e_0, e_1, \ldots, e_n)$. The value of e_i is 0 or 1. x_i=$H(A_i)$ when $e_i = 1$, otherwise x_i=A_i. H is a collision-resistent hash function. The parameter h_d means that the credential needs to be protected by TPM 2.0 with private key x_d and the corresponding public key $h_d = g_d^{x_d}$. The parameter g_d is one of the random generators of G_q.

3) Signature process of I includes two steps: commit computation and signature computation. The former contains parameters: $\sigma_a = g^w$, $\sigma_b = \gamma^w$, w is a random number. The latter contains parameter: $\sigma_z = \gamma^{y_0}$ and sends $(\sigma_a, \sigma_b, \sigma_z)$ to P.

4) The main job of P consists of two parts: one is generating a pair of keys for the U-Prove token, the other is masking the parameters come from I which prevents I from seeing the value of its signature. P generates a random α and computes h: $h = (g_0 g_1^{x_1} \ldots g_n^{x_n} g_t^{x_t} h_d)^{\alpha}$. The value of α^{-1} is regarded as the U-Prove token's private key and the public key is h. After receiving the message $(\sigma_a, \sigma_b, \sigma_z)$, P produces two blind factors β_1 and β_2 to mask the three parameters respectively, then gets $(\sigma'_a, \sigma'_b, \sigma'_z)$ and σ'_c, using the following formula respectively:

$$\begin{aligned}
\sigma'_a &= g_0^{\beta_1} g^{\beta_2} \sigma_a, \\
\sigma'_z &= \sigma_z^{\alpha}, \\
\sigma'_b &= = \sigma_z'^{\beta_1} h^{\beta_2} \sigma_b^{\alpha} \\
\sigma'_c &= H(h, PI, \sigma'_z, \sigma'_a, \sigma'_b) \rightarrow Z_q, \\
\sigma_c &= \sigma_c + \beta_1 \bmod\ q
\end{aligned}$$

where PI is the Prover information field produced by P, which is always revealed during presentation protocol. Then P sends σ_c to I.

5) After I got the parameter σ_c , I generates and conveys the signature σ_c with private key y_0 to P by the Schnorr signature scheme: $\sigma_r = \sigma_c y_0 + w\ mod\ q$.

6) After P received the parameter σ_r, P masks σ_r with β_2 as follow: $\sigma'_r = \sigma_r + \beta_2\ mod\ q$. Then P verifies the validity of the signature, as in Equation (7).

$$\sigma'_a \sigma'_b = (gh)^{\sigma'_r} (g_0 \sigma'_z)^{-\sigma'_c} \tag{7}$$

P gets a valid U-Prove token denoted as follows if result is valid.

$$\mathcal{T} = h, TI, PI, \sigma'_z, \sigma'_c, \sigma'_r$$

The Presentation Proof Protocol.

After getting an issued U-Prove token ($\mathcal{T}$) with a pair of keys (α^{-1}, h), where α^{-1} is secret as private key, h is disclosed to V without revealing to I in the issuance

Protocol as public key. When using $\mathcal{T}$, P can selectively disclose parts of attributes to V, and create a presentation proof for undisclosed attributes. $\mathcal{T}$ can be protected by the TPM 2.0 or not. The usage of $\mathcal{T}$ based on the TPM 2.0 protection will be discussed in this section.

Generate Presentation.

P has the right to determine the following parameters: the indices of disclosed attributes is denoted as $D \subset \{1,\ldots,n\}$, the indices of undisclosed attributes is denoted as $U \subset \{1,\ldots,n\} - D$, the indices of committed attributes is denoted as $C \subset U$. P can not only generate a pseudonym for $\mathcal{T}$, but also specify the pseudonym derived from a specific scope s.

P can generate the presentation proof with the TPM 2.0 as follows:

1) P sends s to the TPM 2.0.
2) After receiving s, TPM 2.0 generates w'_d at random, then computes: $a_d = g_d^{w'_d}$, $g_s = GenEle(s))$, $a'_p = g_s^{w'_d}$, $P'_s = g_s^{x_d}$ and sends (a_d, a'_p, P_s) to P.
3) P computes each x_i, and generates w_0, w_d, for $i \in U$. P generates w_i at random, for $i \in C$. P generates $\widetilde{o}_i, \widetilde{w}_i$ at random. Then computes equations as follow.
$a = H(h^{h_0}(\Pi_{i\in U} g_i^{w_i}) g_d^{w_d} a_d)$,
$g_s = GenEle(s)$, $a_p = H(g_s^{w_p}) a'_p)$,
$\widetilde{c}_i = g^{x_i} g_1^{\widetilde{o}_i})$, $\widetilde{a}_i = H(g^{w_i} g_1^{\widetilde{w}_i})$,
$c_p = H(a, D, x_{i,i\in D}, C, \{\widetilde{c}\}_{i,i\in C}, \{\widetilde{a}\}_{i,i\in C}, a_p, P_s, m)$,
$c = H(c_p, m_d), r_0 = c\alpha^{-1} + w_0$,
$r_i = -cx_i + w_{i,i\in U}$
sends (c_p, m_d) to the TPM 2.0.
4) TPM 2.0 computes and sends the response r'_d :
$$c = H(c_p, m_d), r'_d = -cx_d + w'_d.$$
5) P completes the following operation and eventually gets the presentation proof about $\mathcal{T}$:
$PP =< A_{i,i\in D}, a, (a_p, P_s), r_0, r_{i,i\in U}, r_d, \{\widetilde{c}_i, \widetilde{a}_i, \widetilde{r}_i\}_{i\in C} >$.
where: $r_d = r'_d + w_d$, $\widetilde{r}_i = -c\widetilde{o}_i + \widetilde{w}_i$ $(i \in C)$

Verify Presentation. Given a presented $\mathcal{T}$, V can check it without any secret information. V first gets the input parameters:

$$\begin{aligned} x_i &= cxi(IP, A_i)\ for\ each\ i \in D \\ c_p &= H(a, D, \{x_{i,i\in D}, C, \{\widetilde{c}_{i,i\in C}, \{\widetilde{a}_{i,i\in C}, a_p, P_s, m) \\ c &= H(c_p, m_d). \end{aligned}$$

Then V computes the following equations:

$$\begin{aligned} a &= H((g_0 g_t^{x_t} \Pi_{i\in D} g_i^{x_i})^{-c} h^{r_0} (\Pi_{i\in U} g_i^{r_i}) g_d^{r_d}) \\ g_s &= GenEle(s),\ a_p = H(P_s^c g_s^{r_d}), \\ \widetilde{a}_i &= H(\widetilde{c}_i^c g^{r_i} g_1^{\widetilde{r}_i})(i \in C). \end{aligned}$$

If the above equations are proved, the authentication is successful.

4.2 Modelling the U-Prove Protocol

According to the previous framework, we model the U-Prove Protocol with the applied Pi calculus. This calculus is an extension of the Pi calculus with function symbols which satisfy particular equations. We design function symbols and an equational theory for modelling the U-Prove protocol. All the parameters of this section have the same meaning as Subsection 4.2.

According to our model in Section 3, we define three processes according to different roles:

- *Prover* process (written as $\mathcal{P}$) with three sub-processes $Pini$, $Pgetsig$, $Pgetp$.
- *Issuer* process (written as $\mathcal{I}$) with two sub-processes $Iini$ and $Isig$.
- *Verifier* process (written as $\mathcal{V}$) with one sub-process $verifier$, TPM 2.0 process (written as $\widetilde{\mathcal{T}}$) with one sub-process TPM 2.0.

We define a public channel c to represent a public network through which all messages are transmitted. How I and P agree on the contents of the issued U-Prove token, and how TPM 2.0 provides its public key are outside the scope of this paper.

Functions. We define the relevant functions in this paper with respect to the signature

$$\begin{aligned} \Sigma = \{&c/2,\ sign/2,\ Schsign/2,\ getgt/1,\ getgU/1, \\ &getgD/1,\ getgc/1,\ getAD/1, \&getAU/1, \\ &getAC/1,\ cxt/3,\ cxi/2,\ b/2,\ bc/2,\ b1/2,\ b2/2\} \end{aligned}$$

, where,

- $\{c/2\}$ models generating a commitment value.
- $\{sign/2\}$ models signing messages.
- $\{Schsign/2\}$ models getting a Schnorr signature.
- $\{getgt/1,\ getgU/1,\ getgD/1,\ getgc/1\}$ models derivation functions to derive Issuer's public key.
- $\{getAD/1,\ getAU/1,\ getAC/1\}$ models derivation functions to derive all attributes by Prover and Issuer.
- $\{cxt/3,\ cxi/2\}$ models getting the value of parameters used to compute the γ.
- $\{b/2,\ bc/2,\ b1/2,\ b2/2\}$ models the blind processes of commitments in different situations.

Equations. After setting all functions, we specify an equational theory in terms of a convergent rewriting system. This theory is suitable for ProVerif.

1) $b2(sign(\gamma, y0), \alpha) = blindsign(\gamma, \alpha, y0)$
2) $bc(c(\gamma, w), \alpha) = bc2(\gamma, \alpha, w)$
3) $bc(c(\gamma, \alpha), \beta_1, \beta_2) = bc3(\gamma, \alpha, \beta_1, \beta_2)$

4) $bc(bc(\gamma, \alpha, w), \beta_1, \beta_2) = bc4(\gamma, \alpha, w, \beta_1, \beta_2)$

5) $verif(bc3(\gamma, \alpha, \beta_1, \beta_2), b2(schsign(b(H(h, bc3(\gamma, \alpha, \beta_1, \beta_2), \sigma_{b1}, \sigma_{z1}), \beta_1), y0), \beta_2), pk(g, y0)) = true$

The four equations are convergence equations and form a convergent rewriting system when oriented from left to right. The last equation can be used to check parameters.

Modelling of the Issuance Protocol.

We denote IP as the unique identifier of I parameters under which a $\mathcal{T}$ is issued, all *Issuer* parameters are derived from IP using the derivation functions such as $getgt(IP)$, $getgU(IP)$, $getgD(IP)$, $getgC(IP)$. We denote TI as the unique identifier of the $\mathcal{T}$, all attributes are derived from TI, including disclosed attributes, undisclosed attributes and committed attributes, which are modelled by $getAD(TI)$, $getAU(TI)$, $getAC(TI)$ respectively.

The signature is generated by I under the Schnorr type of signature scheme [7]. The generation of the signature by I is divided into two parts: generation of commitment using a random number, and generation of signature using the private key. The function symbol $c/2$ represents the commitment constructor, and $sign/2$ represents the signature constructor.

$\mathcal{I}$ interacts with $\mathcal{P}$ as the following.

- To initiate a new session, $\mathcal{I}$ sends the messages $\bar{c}\langle sign(\gamma, y_0), c(g, w), c(\ \gamma, w)\rangle$ with a public parameter $sign(\gamma, y_0)$ and two commitments $c(g, w), c(\gamma, w)$.
- After receiving $\{sign(\ \gamma, y_0), c(g, w), c(\ \gamma, w)\}$, $\mathcal{P}$ calculates and sends out parameter σ_c.
- After receiving $\{\sigma_c\}$, $\mathcal{I}$ begins to generate the Schnorr signature by the operation of $\sigma_z = schsign(\sigma_c, y0)$, then sends out $\{\sigma_z\}$.
- After receiving $\{\sigma_z\}$, $\mathcal{P}$ begins to verify and blind the signature, and finally get the blind signature token by $\mathcal{I}$.

Modelling of the Presentation Protocol.

In this paragraph, we assume that the U-Prove token is protected by TPM 2.0 and V needs commitments about some undisclosed attributes. P can only complete the presentation proof generation under the help of TPM 2.0.

P needs to provide the disclosed attributes, I's signature, the token's public key and all presentation proofs. Presentation proofs include presentation proof for undisclosed attributes, presentation proof for committed attributes and presentation proof for information protected by TPM 2.0, denoted as a, $\widetilde{a}_i$, a_p respectively. We use $getAD(TI)$ to get the disclosed attributes (denoted as AD) and $b(\ \gamma, \alpha)$ to get the token's public key (denoted as h) in our codes.

$\mathcal{P}$ interacts with $\widetilde{\mathcal{T}}$ as the following.

- $\widetilde{\mathcal{T}}$ generates parameters and sends out $\{c(gd, wd1), c(gs, wd1), getgs(descG)\}$.
- $\mathcal{P}$ generates the related presentation proofs and sends out $\{h, a, ap, ai, r0, rU, ri, rd\}$.

where h models $\mathcal{T}$'s public key, a models presentation proof of the undisclosed attributes, ap models presentation proof of the protection of the token by TPM 2.0, ai models presentation proof of the commitments which is needed by V, $r0$ models blind signature signed with token's private key, rU models blind signature of each undisclosed attributes, ri models blind signature of each committed attributes, rd models blind signature signed with TPM's private key.

4.3 Security Analysis of the U-Prove Protocol

4.3.1 Authenticity of the Issuance Protocol

The first property we would like to model is the authenticity of the issuance protocol: if P reaches the end of the protocol and she believes she has got a U-Prove token issued by I, then I has really issued the token. To prove this property we annotate processes with some events, marking important stages reached by the protocol which do not have effect on the execution of the processes.

There are a lot of test events in our experiments, here we list three events, which are denoted as follows.

1) **Issuetoken** (TI, g, y_0)**.** I executes this event before signing the token with the private key y_0 and issuing a token with *identifier* TI.

2) **Gettoken** (TI, σ_{r1}, g_0)**.** P executes this event after receiving the token signed by I with the public key g_0.

3) **Bisign** (σ_{r1})**.** P executes this event after blinding the token issued by I.

Given processes as follows.

$$< I^{Ini}(IP, y_0) \mid P^{Ini}(TI, IP, g_0) \mid I^{sig}(TI, IP, y_0) \mid P^{getproof}(h, \alpha) >$$

According to the definition 5. We have verified the authenticity of the issuance protocol by the following two trace properties with ProVerif.

1) $\forall(TI, g, y_0), \exists\{Gettoken(TI, \sigma_{r1}, pk(g, y_0)) \Longrightarrow Issuetoken(TI, g, y_0)\}$,

2) $\forall(TI, \sigma_{r1}, g, y_0), \exists\{Gettoken(TI, \sigma_{r1}, pk(g, y_0)) \Longrightarrow Issuetoken(TI, g, y_0) \vee Bisign(\sigma_{r1})\}$

The first result shows that when P gets a token signed by I with public key $pk(g,\ y_0)$, then I has actually signed with the corresponding private key y_0 and issued that token. The second results shows that when P gets a token signed by I with public key $pk(g,\ y_0)$, not only the I has actually signed with the corresponding private key y_0 and issued that token, but also P has blinded the token.

In our experiment, we set up the event $Issuetoken$ in the process $I^{sig}(TI,\ IP,\ y_0)$, event $Gettoken$ in the process $P^{Ini}(TI,\ IP,\ g_0)$, event $Bisign$ in the process $P^{getproof}(h,\ \alpha)$ respectively. The following results are obtained after running the program.

1) RESULT event($Gettoken(h\_162, pk(g\_163, y0\_164))$) $\Longrightarrow$ event($Issuetoken(TI\_161, g\_163, y0\_164)$) is true.

2) RESULT event($Gettoken(h\_162, pk(g\_163, y0\_164))$) $\Longrightarrow$ event($Bisign(TI\_161, h\_162)$) && event($Issuetoken(TI\_161, g\_163, y0\_164)$)) is true.

4.3.2 Authenticity of the Presentation Protocol

We model the authenticity of the presentation protocol: if V reaches the end of the protocol with the presented U-Prove token, then P has really sent out the presented token signed by I.

As above, we annotate processes with three events and mark important stages reached by the protocol. Three events are denoted as follows.

1) **Gettokenpk** $(h,\ \alpha, gU, xU)$**.** P executes this event after the initialization of parameters, and gets the U-Prove token public key as token's identification.

2) **Evpproof** (h, a)**.** P executes this event before she sends out the generation proof of the token.

3) **Verifok** $(h, a, multpk2(pk(h, inverse(\ \alpha))$**.** V executes this event after she verifies the validity of the token's presentation proof.

Given processes as follows.

$$P^{getsig}(g_0, g, gU, gd) \mid P^{getproof}(h, \alpha) \mid verifier(g, g_0)$$

According to the definition 6. We have verified the authenticity of the Presentation Protocol with ProVerif by the following trace property.

$$\begin{gathered}\forall(h, a, \alpha, gU, xU):\\ \exists\{Verifok(h, a, multpk2(pk(h, inverse(\alpha)),\\ pk(gU, xU)))) \Longrightarrow Evpproof(h, a)\\ \vee Gettokenpk(h, \alpha, gU, xU)\}.\end{gathered}$$

In our experiment, We set up the event $Gettokenpk$ in the process $P^{getsig}(g_0, g, gU, gd)$, event $Evpproof$ in the process $P^{getproof}(h, \alpha)$, event $VerifOK$ in the process $verifier(g, g_0)$ respectively. The result shows that once validation is completed by V, then not only a valid I has actually signed and issued that token, but also P has blinded the token issued by I.

The following result is obtained after running the program.

RESULT $event(VerifOK(h\_213, a\_214, multpk2(pk(h\_213, inverse(alpha\_215)), pk(gU\_216, xU\_217))))$ $\Longrightarrow$ (event($Evpproof(h\_213, a\_214)$)&&$event(Gettokenpk(h\_213, alpha\_215, gU\_216, xU\_217))$) is true.

4.3.3 Untraceability of U-Prove Token

In order to protect the identity information about individuals. During the Issuance Protocol, the Issuer uses blind signature rather than conventional RSA or DSA signature, and issuance is a three-leg interactive protocol enabling the Prover to hide certain token elements from the Issuer. This makes the Issuer never see its own digital signature on an issued U-Prove token, and never see the public key of the U-Prove token.

The blind signature scheme provides a strong privacy guarantee for the Issuance Protocol by untraceability property: the Issuer and all Verifiers cannot learn even a single bit of information beyond what can be inferred from the disclosed attributes in presented U-Prove tokens, even if they would collude from the outset.

According to Definition 7. We have verified the untraceability of U-Prove token by the observational equivalence between two processes $P1$, $P2$, where in Pi the Prover gets and provides the token $\mathcal{T}_i$. In our design we define a natural formulation $P1$ and $P2$ as follows:

$$\begin{aligned}&Pi :=\\ &\quad let(TI, gd) = (TI1, gd1)\ in\ PIni(TI, gd)|\\ &\quad let(TI, gd) = (TI2, gd1)\ in\ PIni(TI, gd)|\\ &\quad in(sc2, (hi, \alpha i));\\ &\quad let(TI, h, \alpha) = (TI1, hi,\ \alpha i)\ in\ Pgetsig(TI, h, \alpha).\end{aligned}$$

We get the following result in our experimental.

RESULT Observational equivalence is true (bad not derivable).

5 Conclusion and Future Work

In this paper, we provide a general model framework on the SDABCS, and as a case study, provide the first formal analysis of U-Prove protocol. We give the detailed definitions about authenticity in each sub-protocols and untraceability of U-Prove token. Authenticities are expressed as a correspondence property and untraceability is proved by observational equivalence. All of the security properties are suitable for ProVerif [6].

As an innovative cryptographic technology, in addition to the properties proven in our paper, there are a lot of contents worth studying in SDABCS, such as revocable [16], reusable and so on. In particular, the properties of accountability has attract the attention of experts in recent years, we think studying the accountability in the U-Prove protocol is necessary.

Acknowledgments

The research is supported by National Natural Science Foundation of China under Grant No.61572453, No.6120-2404, No.61520106007, No.61170233, No.61232018,No.61-572454,Natural Science in Colleges and Universities in

Anhui Province under Grant No.KJ2015A257, and Anhui Provincial Natural Science Foundation under Grant No.1508085SQF215. We gratefully acknowledge the anonymous reviewers for our valuable comments.

References

[1] M. Abadi, B. Blanchet, and C. Fournet, "Just fast keying in the pi calculus," *ACM Transactions on Information and System Security*, vol. 10, no. 3, pp. 9, 2007.

[2] R. Amin, "Cryptanalysis and efficient dynamic id based remote user authentication scheme in multi-server environment using smart card.," *International Journal of Network Security*, vol. 18, no. 1, pp. 172–181, 2016.

[3] M. Backes, M. Maffei, and D. Unruh, "Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol," in *IEEE Symposium on Security and Privacy (SP'08)*, pp. 202–215, 2008.

[4] G. Barthe, B. Grégoire, and S. Zanella B., "Formal certification of code-based cryptographic proofs," *ACM SIGPLAN Notices*, vol. 44, no. 1, pp. 90–101, 2009.

[5] B. Blanchet and A. Chaudhuri, "Automated formal analysis of a protocol for secure file sharing on untrusted storage," in *IEEE Symposium on Security and Privacy (SP'08)*, pp. 417–431, 2008.

[6] B. Blanchet, B. Smyth, and V. Cheval, *Proverif 1.90: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2014.

[7] L. Chen and J. Li, "Flexible and scalable digital signatures in tpm 2.0," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security*, pp. 37–48, 2013.

[8] L. Chen and R. Urian, "Daa-a: Direct anonymous attestation with attributes," in *International Conference on Trust and Trustworthy Computing*, pp. 228–245, 2015.

[9] S. Delaune, M. Ryan, and B. Smyth, "Automatic verification of privacy properties in the applied pi calculus," in *IFIP International Conference on Trust Management*, pp. 263–278, 2008.

[10] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[11] R. Gay, I. Kerenidis, and H. Wee, "Communication complexity of conditional disclosure of secrets and attribute-based encryption," in *Annual Cryptology Conference*, pp. 485–502, 2015.

[12] J. Goubault-Larrecq, C. Palamidessi, and A. Troina, "A probabilistic applied pi–calculus," in *Asian Symposium on Programming Languages and Systems*, pp. 175–190, 2007.

[13] X. Li, Y. Zhang, and Y. Deng, "Verifying anonymous credential systems in applied pi calculus," in *Cryptology and Network Security*, pp. 209–225, 2009.

[14] W. Lueks, G. Alpár, J. Hoepman, and P. Vullers, "Fast revocation of attribute-based credentials for both users and verifiers," in *IFIP International Information Security Conference*, pp. 463–478, 2015.

[15] W. Mostowski and P. Vullers, "Efficient u-prove implementation for anonymous credentials on smart cards," in *Security and Privacy in Communication Networks*, pp. 243–260, 2012.

[16] L. Nguyen and C. Paquin, "U-prove designated-verifier accumulator revocation extension," Technical Report MSR-TR-2014-85, Microsoft Research, 2014.

[17] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 409–421, 2014.

[18] C. Paquin, *U-prove Technology Overview v1*, 2013.

[19] H. Patel, D. Jinwala, M. Highway, M. Bhandu, and S. Ichchhanath, "Automated analysis of internet key exchange protocol v2 for denial of service attacks," *International Journal of Network Security*, vol. 17, no. 1, pp. 66–71, 2015.

[20] J. Shao, Y. Qin, D. Feng, and W. Wang, "Formal analysis of enhanced authorization in the tpm 2.0," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pp. 273–284, 2015.

[21] F. Tiezzi and N. Yoshida, "Reversible session-based pi-calculus," *Journal of Logical and Algebraic Methods in Programming*, vol. 84, no. 5, pp. 684–707, 2015.

[22] P. Vullers and G. Alpár. "Efficient selective disclosure on smart cards using idemix,". in *Policies and Research in Identity Management*, pp. 53–67, 2013.

[23] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pp. 261–270, 2010.

Biography

Caimei Wang was born in 1978. She is a lecturer in Department of Computer Science and Technology, HeFei University. She also is a Ph.D. candidate of University of Science and Technology of China. Her main research interests include computer network, information security, and mobile computation. (Email: wangcmo@mail.ustc.edu.cn)

Yan Xiong was born in 1960. He is a professor in School of Computer Science and Technology, University of Science and Technology of China. His main research interests include distributed processing, mobile computation, computer network and information security. (Email: yxiong@ustc.edu.cn)

Wenjuan Cheng (corresponding author) was born in 1970. She is a professor in School of Computer and

Information, Hefei University of Technology. Her main research interests include computer network and information security, computer application technology.(Email: cheng@ah.edu.cn)

Wenchao Huang was born in 1982. He received both the B.S. and Ph.D. degrees in computer science from University of Science and Technology of China. He is associate professor in School of Computer Science and Technology, University of Science and Technology of China. His current research interests include mobile computing, information security, trusted computing and formal methods. (Email: huangwc@ustc.edu.cn)

Huihua Xia was born in 1994. He received the B.S. degree in computer science from University of Science and Technology of China. He is currently a PhD student in School of Computer Science and Technology, University of Science and Technology of China. His current research interests include data publishing and data privacy. (Email: download@mail.ustc.edu.cn)

Jianmeng Huang was born in 1991. He received the B.S. degree in computer science from University of Science and Technology of China in 2013. He is currently working towards the Ph.D. degree at the Department of Computer Science and Technology, University of Science and Technology of China. His current research interests include information security and mobile computing. (Email: mengh@mail.ustc.edu.cn).