Provable Secure Generic Construction of Proxy Signature from Certificate-based Signature

Rufen Huang[1,*], Zhenjie Huang[1,2] and Qunshan Chen[1]

[1]College of Computer Science, Minnan Normal University, Zhangzhou, Fujian, 363000, P.R. China

[2]Zhangzhou City University, Zhangzhou, Fujian, 363000, P.R. China

Abstract: The certificate-based signature is an attractive cryptography primitive whose original motivation is to simplify certificate's management and to eliminate key escrow problem. The proxy signature is another cryptography paradigm which permits an entity to delegate his signing rights to another. In this paper, we first note that certificate-based signatures and proxy signatures have something in common, and analyze the relationship between the certificate-based signatures and the proxy signatures. Secondly, we introduce a generic construction of the proxy signature *CBS-to-PS* from a previous secure certificate-based signature, and prove that our *CBS-to-PS* scheme is secure if the underlying certificate-based signature scheme is secure. Finally, we give a concrete application for our *CBS-to-PS* as an example.

Keywords: Certificate-based signature, Proxy signature, Generic Construction, Conversion, Security mode.

1. INTRODUCTION

The concept of certificate-based cryptography (*CBC*) was first introduced by Gentry [1] to integrate the merits of identity-based cryptography (*IBC*) [2] into public key cryptography (*PKC*) in Eurocrypt 2003, whose original motivation is to simplify the certificate management procedures, including revocation, storage, distribution and verification of certificates in conventional *PKC*, and to overcome key escrow problem in *IBC*. In 2004, the notion of certificate-based signature (*CBS*) was first proposed by Kang *et al.* [3] following the idea of Gentry's *CBC*. A *CBS* scheme includes a Certificate Authority (*CA*) and a user, also called signer, the user generates his own private/public key and requests an up-to-date certificate from the *CA*, while the certificate in a *CBS* is implicitly used as a part of signing key. In this way, there is no inspection of genuineness about the certificates. The *CBS* achieves the same trust level (Level 3) [4] of the authority as that of the conventional *PKC*, and does not suffer to the Denial-of-Decryption (*DoD*) attack [5] if we use certificate as a part of signing key. Therefore, *CBS* has become a topic of active research in cryptography. Since Kang *et al.'s* seminal paper [3], the security model of *CBS* and the formally definition of the key replacement attack have been introduced by Li *et al.* in 2007 [6]. Then Au *et al.* constructed a certificate-based (linkable) ring signature scheme [7] in the same year, and Liu *et al.* [8] proposed two *CBS* schemes in 2008, one was a scheme without pairings, the other was proved for its security in the standard model. Unfortunately, Zhang [9] pointed out that Liu *et al.'s* scheme [8] was insecure and he improved Liu *et al.'s* scheme [8] in 2009. Subsequently, several new CBS schemes have been proposed, such as Liu *et al.'s* scheme [10], Li *et al.'s* scheme [11], etc., including many extensions of the basic certificate-based signature schemes, like Chen and Huang's certificate-based proxy signature scheme [12], Huang *et al.'s* certificate-based blind signature scheme [13], and so on.

Proxy signature is another cryptography paradigm used for delegating the signing rights. The seminal concept of proxy signature was invented by Mambo *et al.* in 1996 [14]. In a proxy signature scheme, there are two entities involved, namely, an original signer and a proxy signer. An original signer can delegate its signing power to a proxy signer, who can thus sign on behalf of the original signer. The proxy signature plays an important role in cases when a user wants to delegate his signing right to the other user [15-17], such as mobile agent, mobile communications, distributed networks, grid computing, and e-commerce etc., where delegation of signing rights is commonly required. In Mambo *et al.'s* seminal paper [14], the delegation types were categorized into three levels of delegation: full delegation, partial delegation, and delegation by warrant. So far, there are a number of proxy signature schemes proposed for each, including partial delegation [14, 18], delegation by warrant [15], and partial delegation with warrant [19, 20], etc.

Although they are different forms of signatures, but the certificate-based signature and the proxy signature have something in common. In this paper, we first discuss the relationship between the certificate-based signature and the proxy signature delegated by warrant (*PS*). Secondly, we convert the previous certificate-based signature scheme to the proxy signature scheme, namely, we propose a generic construction of the proxy signatures *CBS-to-PS* from the previous certificate-based signatures. Then, we give a security proof to prove our generic construction *CBS-to-PS* is secure only if the underlying *CBS* scheme is secure, in other words, our *CBS-to-PS* scheme is secure only if the underlying *CBS* scheme is secure. Finally, as an example, we construct a concrete application for our generic construction *CBS-to-PS*.

The rest of the paper is organized as follows: In Section 2, we review the related definitions of *CBS* and *PS* briefly, including the formal definition, adversary types and security model for *CBS* and *PS*. In Section 3, we first analyze the similarities, differences and relationships between the certificate-based signatures and the proxy signatures delegated by warrant, and propose a generic construction of *CBS-to-PS* and give a security proof for our *CBS-to-PS*. In Section 4, we give a concrete application example for our *CBS-to-PS*. Finally, we conclude the paper in Section 5.

2. PRELIMINARIES

In this section, we review the related preliminaries including the formal definition and the security model for *CBS* and *PS*.

2.1. Certificate-based Signature

We refer [6] to review the formal definition and the security model for the certificate-based signature in this section. We use the prefix *CB*- to denote a *CBS* system for convenience below. .

2.1.1. The Formal Definition

In a certificate-based signature scheme, there are two participants involved, including a *CA* and a user.

Definition 2.1. (*CBS*). A certificate-based signature scheme is defined by the following five algorithms:

- ***CB-Setup****(k)→(CB-params,mpk,msk)***:** The algorithm inputs a security parameter k, outputs the system public parameters $CB-params$ and the *CA*'s master key pair (mpk,msk), where $CB-params$ is the system public parameters except the system master public key mpk, such as the descriptions about the groups, hash functions etc.
- ***CB-UserKeyGen****(CB-params,ID)→(SK*$_{ID}$*,PK*$_{ID}$*)***:** The algorithm inputs the system public parameters $CB-params$ and a user's identity ID, outputs the user's key pair (SK_{ID}, PK_{ID}).
- ***CB-CertGen****(CB-params,msk,ID,PK*$_{ID}$*)→Cert*$_{ID}$**:** The algorithm inputs the system public parameters $CB-params$, the *CA*'s master secret key msk, the user's identity ID and his public key PK_{ID}, outputs a certificate $Cert_{ID}$ corresponding to the user ID and public key PK_{ID}.
- ***CB-Sign****(m,CB-params,ID,SK*$_{ID}$*,Cert*$_{ID}$*)→*σ**:** The algorithm inputs a message m to be signed, the system public parameters $CB-params$, the user's identity ID and the corresponding certificate $Cert_{ID}$, user's private key SK_{ID}, outputs a certificate-based signature σ on the message m.
- ***CB-Verify****(m,* σ *,CB-params,mpk,ID*$_{ID}$*,PK*$_{ID}$*)→(true, false)***:** The algorithm inputs a message/signature pair (m,σ), the system public parameters $CB-params$, the *CA*'s master public key mpk, the user's identity ID and public key PK_{ID}, returns *true* if σ is valid, otherwise returns *false*.

2.1.2. Adversarial Type

We want a *CBS* scheme to be secure against two types of adversary with different capabilities, and they are known as Type *I* adversary A_I and Type *II* adversary A_{II}.

- **A_I:** The type *I* adversary A_I simulates an uncertified user which holds the private key of the user, and A_I can replace any entity's public key with a value chosen by himself, but A_I knows nothing about the *CA*'s master secret key.
- **A_{II}:** The type *II* adversary A_{II} simulates a malicious *CA* which holds the *CA*'s master secret key, but A_{II} neither knows anything about the user's private key nor replaces any user's public key.

2.1.3. Attack Model

In this section, we will recall security model of certificate-based signatures, which is defined by two games between an adversary $A \in \{A_I, A_{II}\}$ and a challenger *C*.

Definition 2.2. (*CB-Game1*). The game between a Type *I* adversary A_I and a challenger *C* is defined as follows.

- ***CB-Setup*:** For a given security parameter *k,* the challenger *C* runs the algorithm $CB-Setup$ to obtain the system public parameters $CB-params$ and the system master key pair (mpk,msk), gives $CB-params$ and mpk to the adversary A_I, and msk is kept secretly by himself.
- ***CB-Query Oracls*:** The adversary A_I can adaptively issue the following queries in polynomial time:

  1)UserKeyQuery. The adversary A_I issues $UserKeyQuery(ID_i)$ for an identity ID_i, the challenger *C* returns a key pair (SK_{IDi}, PK_{IDi}) to A_I.

  2)CertQuery. The adversary A_I issues $CertQuery(ID_i, PK_{IDi})$ for an identity ID_i and the corresponding public key PK_{IDi}, the challenger *C* returns a certificate $Cert_{IDi}$ to A_I.

3)ReplacePublicKeyQuery. For a given identity ID_i, the adversary A_I chooses a new public key PK'_{ID_i} by himself, and replaces ID_i's public key with PK'_{ID_i}.

4)SignQuery. The adversary A_I issues $SignQuery(m, ID_i, PK_{IDi})$ for an identity ID_i and the public key PK_{IDi} on a message m, the challenger C returns a signature σ to A_I.

- ***CB-Output*:** Finally, A_I outputs a forged signature σ^* on the message m^* for a target ID^* and the public key PK_{ID^*}.

We say A_I wins $CB-Game1$ if σ^* is a valid CBS on the message m^* under the public key PK_{ID^*} with the identity ID^*, and (ID^*, PK_{ID^*}) has never been submitted to $CertGenQuery$, (m^*, ID^*, PK_{ID^*}) has never been submitted to $SignQuery$.

Definition 2.3. (*CB-Game2*). The game between a Type *II* adversary A_{II} and a challenger *C* is defined as follows.

- ***CB-Setup*:** For a given security parameter k, the challenger *C* runs the algorithm $CB-Setup$ to obtain the system public parameters $CB-params$ and the system master key pair (mpk, msk), returns $\{CB-params, mpk, msk\}$ to the adversary A_{II}.
- ***CB-Query Oracles*:** The adversary A_{II} can generate the user's certificate because A_{II} holds the *CA*'s master secret key msk, so A_{II} doesn't issue $CertQuery$. The adversary A_{II} can adaptively issue the following queries in polynomial time.

1)UserKeyQuery. The adversary A_{II} issues $UserKeyQuery(ID_i)$ for an identity ID_i, the challenger C returns the public key PK_{IDi} to A_{II}.

2)CorruptionQuery. The adversary A_{II} issues $CorruptionQuery(ID_i)$ for an identity ID_i, the challenger C returns a private key SK_{IDi} to A_{II}.

3)SignQuery. The $SignQuery$ is similar to $CB-Game1$.

- ***CB-Output*:** Finally, A_{II} outputs a forged signature σ^* on the message m^* for a target ID^* and the public key PK_{ID^*}.

We say A_{II} wins $CB-Game2$ if σ^* is a valid CBS on the message m^* under the public key PK_{ID^*} with the identity ID^*, and ID^* has never been submitted to $CorruptionQuery$, (m^*, ID^*) has never been submitted to $SignQuery$.

Definition 2.4. (*Unforgability of CBS*) A certificate-based signature scheme is existentially unforgeable under adaptively chosen message attack if the probability of success that any polynomial bounded adversary A_I and A_{II} win $CB-Game1$ and $CB-Game2$ respectively is negligible.

2.2. Proxy Signature

We refer [21] to review the formal definition and the security model of the proxy signature in this section. We will use the prefix *PS-* to denote a *PS* system for convenience below.

2.2.1. The Formal Definition

In a proxy signature scheme, there are two participants involved, including an original signer *O* and a proxy signer *P*.

Definition 2.5. (*PS*). A proxy signature scheme is defined by the following five algorithms:

- ***PS-Setup****(k)→(PS-params):* The algorithm inputs a security parameter *k*, outputs the system public parameters $PS-params$.
- ***PS-KeyGen****(PS-params)→(SK_O,PK_O,SK_P,PK_P):* The algorithm consists of the following two sub-algorithms:
- ***PS-OKeyGen****(PS-params)→(SK_O,PK_O)*: The algorithm inputs the system public parameters $PS-params$, outputs the original signer *O*'s key pair (SK_O, PK_O).
- ***PS-PKeyGen****(PS-params)→(SK_P,PK_P):* The algorithm inputs the system public parameters $PS-params$, outputs the proxy signer *P*'s key pair (SK_P, PK_P).
- ***PS-DelGen****(PS-params,w,SK_O)→* D_w**:** The algorithm inputs the system public parameters $PS-params$, the original signer *O*'s private key SK_O and a warrant w, outputs a delegation D_w which corresponds to the warrant w.
- ***PS-PSign****(m,PS-params,w,D_w,SK_P)→* σ *:* The algorithm inputs a message m to be signed, the system public parameters $PS-params$, a warrant w and the corresponding delegation D_w, the proxy signer *P*'s private key SK_P, outputs a proxy signature σ on the message m.
- ***PS-Verify****(m,* σ *,PS-params,w,D_w,PK_O,PK_P)→(true, false):* The algorithm inputs a message/signature pair

(m,σ), the system public parameters $PS-params$, a warrant w and the corresponding delegation D_w, the original signer O's public key PK_O, proxy signer P's public key PK_P. If σ is valid, the algorithm outputs $true$, otherwise it outputs $false$.

2.2.2. Adversarial Types

In a proxy signature scheme, we are concerned with three different types of attackers, by an outside adversary, a malicious proxy signer and a malicious original signer, respectively. We want a proxy signature scheme to be secure against these three adversaries, namely Type 1 adversary A_1, Type 2 adversary A_2, and Type 3 adversary A_3.

- ***A_1*:** Type 1 adversary A_1 simulates a malicious proxy signer which holds the private key of the proxy signer, and the public keys of the original signer and the proxy signer.
- ***A_2*:** Type 2 adversary A_2 simulates a malicious original signer which holds the private key of the original signer, and the public keys of the original signer and the proxy signer.
- ***A_3*:** Type 3 adversary A_3 simulates an outside adversary which only holds the public keys of the original signer and the proxy signer.

2.2.3. Attack Model

We want a proxy signature scheme to be existentially unforgeable against each of the above three adversaries. But if a proxy signature scheme is existentially unforgeable against Types 1 and Type 2 adversaries, then it is also existentially unforgeable against a Type 3 adversary in evidence. So we only consider a proxy signature to be existentially unforgeable against Type 1 and Type 2 adversaries. The existential unforgeability of a proxy signature scheme is defined by the following games between adversaries and the challenger.

(1) **PS-Game1.** The game between a Type 1 adversary A_1 and a challenger C is defined as follows:

- ***PS-Setup*:** For a given security parameter k, the challenger C runs the algorithm $PS-Setup$ to obtain the system public parameters $PS-params$, and runs the algorithm *PS-KeyGen* to obtain the original signer O's key pair (SK_O,PK_O) and the proxy signer P's key pair (SK_P,PK_P), gives $PS-params$ and (SK_P,PK_P,PK_O) to the adversary A_1.
- ***PS-Query Oracles***: Type 1 adversary A_1 can adaptively issue the following queries in polynomial time.

1)*DelQuery*: The adversary A_1 issues $DelQuery(w_i)$ for a warrant w_i, the challenger C returns a delegation D_{wi} to A_1.

2)*PSignQuery*: The adversary A_1 issues $PSignQuery(m,w_i)$ on a message m under the warrant w_i, the challenger C returns a signature σ to A_1.

- ***PS-Output*:** Finally, A_1 outputs a forged signature σ^* on the message m^* under the warrant w^*.

We say A_1 wins $Game1$ if σ^* is a valid proxy signature on the message m^* under the warrant w^*, and w^* has never been submitted to $DelQuery$, (m^*,w^*) has never been submitted to $PSignQuery$.

(2) **PS-Game2.** The game between a Type 2 adversary A_2 and a challenger C is defined as follows:

- ***PS-Setup*:** For a given security parameter k, the challenger C runs the algorithm $PS-Setup$ to obtain the system public parameters $PS-params$, and runs the algorithm *PS-KeyGen* to obtain the original signer O's key pair (SK_O,PK_O) and the proxy signer P's key pair (SK_P,PK_P), gives $PS-params$ and (SK_O,PK_O,PK_P) to the adversary A_2.
- ***PS-Query Oracles***: Type 2 adversary A_2 can adaptively issue the *PSignQuery* in polynomial time.
- ***PSignQuery*:** The adversary A_2 issues $PSignQuery(m,w_i)$ on a message m under the warrant w_i, the challenger C returns a signature σ to A_2.
- ***PS-Output*:** Finally, A_2 outputs a forged signature σ^* on the message m^* under the warrant w^*.

We say A_2 wins $Game2$ if σ^* is a valid proxy signature σ^* on the message m^* under the warrant w^* and (m^*,w^*) has never been submitted to $PSignQuery$.

Definition 2.6. (*Unforgeability of PS*). A proxy signature scheme is existentially unforgeable under adaptively chosen message attacks if the probability of success that any polynomial bounded adversary A_1 and A_2 win the $PS-Game1$ and $PS-Game2$, respectively, is negligible.

3. THE GENERIC CONSTRUCTION OF *PS* FROM *CBS*

Note that there are some conditions common between the certificate-based signatures and the proxy signatures delegated by warrant, in this section, we firstly analyze the similarities and differences between *CBS* and *PS*, then propose a generic construction of the proxy signatures *CBS-to-PS* from the previous certificate-based signatures, and give a security proof for our *CBS-to-PS*.

3.1. Comparison of CBS and PS

Although at the first glance, *CBS* and the *PS* are completely different concepts, but we have been able to find some common grounds between a *PS* scheme and a *CBS* scheme.

- In a *CBS* scheme, the role of *CA* is similar to that of the original signer in a *PS* scheme, both of them should generate an authorization for another signer, that is, a certificate for the user in a *CBS* scheme or a delegation for the proxy signer in a *PS* scheme. The role of the user in a *CBS* scheme is similar to that of the proxy signer in a *PS* scheme, both of them should generate a valid signature by using their own private key and authorization information.
- The *CBS* scheme and the *PS* scheme both involve two entities, and one of them must generate an authorization (certificate or delegation) for the other one. In a certificated-based signature scheme, there are two entities involved, namely, a *CA* and a user, and the *CA* generates a certificate which includes a user's identity and public key, while in a proxy signature scheme, there are two entities involved as well, namely, an original signer *O* and a proxy signer *P* and the original signer *O* generates a delegation which includes the type of security policy for the proxy signer and the original signer.
- The *CBS* scheme and the *PS* scheme both require two pieces of secret information when they sign on a message. In a *PS* scheme, it will take both the proxy signer's private key and a delegation corresponding to the warrant information for signing a message, similarly, in a *CBS* scheme, it will take both user's private key and a certificate corresponding to user's public key for signing a message.

3.2. The Generic Construction of CBS-to-PS

We describe how to convert a previous *CBS* scheme to a *PS* scheme in this section. Namely, we now show a generic construction for *CBS-to-PS* as follows. For convenience, we let Π_{CB} denote a *CBS* scheme with five algorithms: *CB-Setup*, *CB-UserKeyGen*, *CB-CertGen*, *CB-Sign*, and *CB-Verify*, Π_{PS} denote a proxy signature scheme with five algorithms: *PS-Setup*, *PS-KeyGen*, *PS-DelGen*, *PS-PSign*, and *PS-Verify* throughout the paper.

- ***PS-Setup***: The algorithm inputs a security parameter k, runs $CB-Setup(k)$ of Π_{CB} to get $CB-params$, mpk and msk. Sets $PS-params = CB-params$, $SK_O = msk$, $PK_O = mpk$. The algorithm outputs $PS-params$ as the system public parameters of Π_{PS}.
- ***PS-KeyGen***: The algorithm consists of the following two sub-algorithms:
- ***PS-OKeyGen***: The algorithm outputs the original signer O's key pair (SK_O, PK_O), which is already obtained in $PS-Setup$ phase.
- ***PS-PKeyGen:*** We denote *P* the proxy signer's identity. The algorithm inputs the system public parameters $PS-params$. Sets $CB-params = PS-params$, $ID = P$, first runs $CB-UserKeyGen$ $(CB-params, ID)$ of Π_{CB} to get (SK_{ID}, PK_{ID}), then sets $SK_P = SK_{ID}$, $PK_P = PK_{ID}$. The algorithm outputs the proxy signer *P*'s key pair (SK_P, PK_P).
- ***PS-DelGen***: The algorithm inputs the system public parameters $PS-params$, the original signer *O*'s private key SK_O, the warrant w and the proxy signer's identity *P*. The algorithm sets $CB-params = PS-params$, $msk = SK_O$, $ID = P \| w$, $PK_{ID} = PK_P$, first runs $CertGen(CB-params, msk, ID, PK_{ID})$ of Π_{CB} to get $Cert_{ID}$, then sets $D_w = Cert_{ID}$, outputs the delegation D_w which corresponds to the warrant w.
- ***PS-PSign***: The algorithm inputs a message m to be signed, the system public parameters $PS-params$, a warrant w and the corresponding delegation D_w, the proxy signer *P*'s private key SK_P. The algorithm sets $CB-params = PS-params$, $ID = P \| w$, $Cert_{ID} = D_w$, $SK_{ID} = SK_P$, runs $CB-Sign$ $(m, CB-params, ID, Cert_{ID}, SK_{ID})$ of Π_{CB} to get a signature σ, outputs σ as a proxy signature on the message m.
- ***PS-Verify:*** The algorithm inputs the message/signature pair (m,σ), the system public parameters $PS-params$, a warrant w and the corresponding delegation D_w, the original signer *O*'s public key PK_O, the proxy signer *P*'s public key PK_P. The algorithm sets $CB-params = PS-params$, $mpk = PK_O$, $ID = P \| w$, $PK_{ID} = PK_P$, outputs $CB-Verify(m, \sigma, CB-params, mpk, ID, PK_{ID})$.

3.3. Security Proof for CBS-to-PS

Theorem 1 (*Unforgeability of CBS-to-PS).* The constructed *CBS-to-PS* scheme is existentially unforgeable against adaptively chosen-message attack only if the underlying *CBS* scheme is secure.

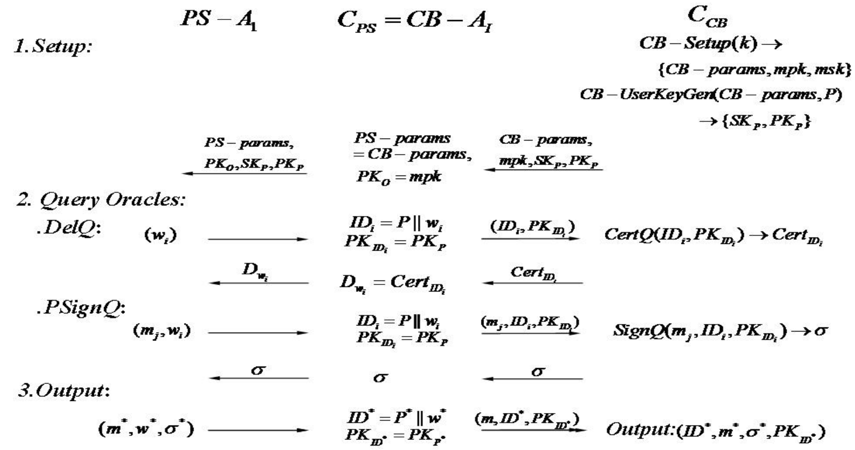

Fig. (1). The process of proof for Game1 of *CBS-to-PS*.

Lemma 1. The constructed *CBS-to-PS* scheme is existentially unforgeable against type 1 adversary $PS-A_1$ only if the underlying *CBS* scheme is existentially unforgeable against Type I adversary $CB-A_I$ under adaptively chosen-message attack.

Proof: Let $PS-A_1$ be a Type 1 adversary of *PS*, which can win the $PS-Game1$ for *CBS-to-PS* scheme above, we can construct a Type *I* adversary $CB-A_I$, which also plays as the role of challenger C_{PS} simultaneously, to win the $CB-Game1$ for underlying *CBS* scheme as follows. Let C_{CB} be a challenger of the *CBS* scheme.

- ***PS-Setup***: Lets *P* denote the identity of a proxy signer. The challenger C_{CB} runs $CB-Setup(k)$ of Π_{CB} to obtain $(CB-params, mpk, msk)$, runs $CB-UserKeyGen(CB-params, P)$ of Π_{CB} to obtain the proxy signer *P*'s key pair (SK_P, PK_P), returns $\{CB-params, mpk, SK_P, PK_P\}$ to $CB-A_I$, $CB-A_I$ sets $PS-params = CB-params$, $PK_O = mpk$, then returns $\{PS-params, PK_O, PK_P, SK_P\}$ to $PS-A_I$.
- ***PS-Query Oracles***: Type 1 adversary $PS-A_1$ can adaptively submit the following query oracles:

1)DelQuery:

1) On a new query w_i, the adversary $PS-A_1$ sends w_i to $CB-A_I$;

2) $CB-A_I$ sets $ID_i = P \| w_i$, $PK_{IDi} = PK_P$ and issues $CertQuery(ID_i, PK_{IDi})$. The challenger C_{CB} returns $Cert_{IDi}$ to $CB-A_I$;

3) $CB-A_I$ sets $D_{wi} = Cert_{IDi}$, and returns D_{wi} to $PS-A_I$.

2)PSign Query:

1) On a new query (m_j, w_i), the adversary $PS-A_I$ sends (m_j, w_i) to $CB-A_I$;

2) $CB-A_I$ sets $mpk = PK_O$, $ID_i = P \| w_i$, $PK_{IDi} = PK_P$, issues $CertQuery(ID_i, PK_{IDi})$ to obtain $Cert_{IDi}$, and issues $SignQuery(m_j, ID_i, PK_{IDi})$ to obtain a signature forgery σ_j. The challenger C_{CB} returns σ_j to $CB-A_I$;

1) $CB-A_I$ returns σ_j to $PS-A_I$.

- ***PS-Output***: Finally, $PS-A_I$ outputs a forged proxy signature (m^*, σ^*, w^*) on the message m^* under the warrant w^*. $CB-A_{II}$ sets $ID^* = P \| w^*$, $PK_{ID^*} = PK_P$, outputs $(m^*, \sigma^*, ID^*, PK_{ID^*})$ as a *CBS* forgery. If σ^* is a valid *PS* under the warrant w^* on the message m^*, then σ^* must be a valid *CBS* under the identity ID^* and the public key PK_{ID^*}. That is, if we forge a *PS* signature σ^* successfully, then the signature σ^* must be a valid forgery for *CBS*. So the constructed *CBS-to-PS* scheme is secure if underlying *CBS* scheme is secure. The process is shown in Fig. (**1**).

Lemma 2. The constructed *CBS-to-PS* scheme is existentially unforgeable against type 2 adversary $PS-A_2$ if the underlying *CBS* scheme is existentially unforgeable against Type II adversary $CB-A_{II}$ under adaptively chosen-message attack.

Proof: Let $PS-A_2$ be a Type 2 adversary of *PS*, which can win the $PS-Game2$ for *CBS-to-PS* scheme above, then we can construct a Type *II* adversary $CB-A_{II}$, which also plays as the role of the challenger C_{PS} simultaneously, to win the $CB-Game2$ for underlying *CBS* scheme. Let C_{CB} be a challenger of the *CBS* scheme.

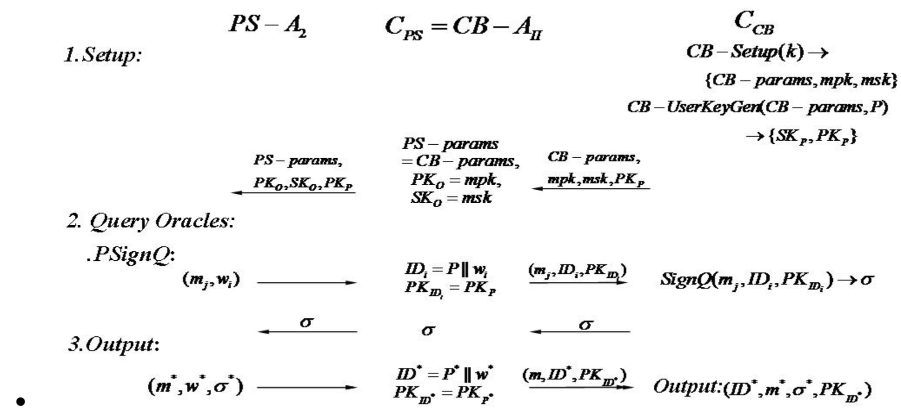

Fig. (2). The process of proof for Game2 of *CBS-to-PS*.

- ***PS-Setup***: Let P denote the identity of a proxy signer. The challenger C_{CB} runs $CB-Setup(k)$ of Π_{CB} to obtain $(CB-params, mpk, msk)$, runs $CB-UserKeyGen(CB-params, P)$ of Π_{CB} to obtain the proxy signer P's key pair (SK_P, PK_P), returns $\{CB-params, msk, mpk, PK_P\}$ to $CB-A_{II}$, $CB-A_{II}$ sets $PS-params = CB-params$, $PK_O = mpk$, $SK_O = msk$, then returns $\{PS-params, PK_P, PK_O, SK_O\}$ to $PS-A_2$
- ***PS-Query Oracles***: Type 2 adversary $PS-A_2$ which can adaptively submit the *PSignQuery* oracle as follows:
- On a new query (m_j, w_i), $PS-A_2$ sends (m_j, w_i) to $CB-A_{II}$;
- $CB-A_{II}$ sets $mpk = PK_O$, $ID_i = P \| w_i$, $PK_{ID_i} = PK_P$, computes $Cert_{IDi}$ by using (ID_i, PK_{IDi}, msk), and issues $SignQuery(m_j, ID_i, PK_{IDi})$ to obtain a signature forgery σ_j. The challenger C_{CB} returns σ_j to $CB-A_{II}$;
- $CB-A_{II}$ returns σ_j to $PS-A_2$.
- ***PS-Output***: Finally, $PS-A_2$ outputs a forged proxy signature (m^*, σ^*, w^*) on the message m^* under the warrant w^*. $CB-A_{II}$ sets $ID^* = P \| w^*$, $PK_{ID^*} = PK_P$, outputs $(m^*, \sigma^*, ID^*, PK_{ID^*})$ as a *CBS* forgery. If σ^* is a valid *PS* forgery on the message m^* under the warrant w^*, then σ^* must be a valid *CBS* under the identity ID^* and the public key PK_{ID^*}. That is, if we forge a *PS* signature σ^* successfully, then the signature σ^* must be a valid forgery for *CBS*. So the constructed *CBS-to-PS* scheme is secure if underlying *CBS* scheme is secure. The process is shown in Fig. (**2**).

4. A CONCRETE EXAMPLE FOR *CBS-TO-PS*

We give an example for our *CBS-to-PS* in this section.

4.1. Review of CBS Scheme

Now, we review a previous certificate-based signature scheme [6] briefly, it consists of the following five algorithms.

- ***Setup:*** Given a security parameter k, let G_1, G_2 be groups of prime order q. $e: G_1 \times G_1 \to G_2$ be a bilinear map, $P \in G_1$ be an arbitrary generator of G_1. $H_0: \{0,1\}^* \times G_1 \to G_1$, $H_1: \{0,1\}^* \times G_1 \times G_1 \to G_1$, and $H_2: \{0,1\}^* \times \{0,1\}^* \times G_1 \times G_1 \to G_1$ are three distinct cryptographic hash functions. The algorithm selects a random number $s \in Z_q^*$, sets $msk = s$ and computes $mpk = sP$. Then the system master key pair is (mpk, msk), and the system public parameters are $params = \{G_1, G_2, e, q, P, H_0, H_1, H_2\}$.
- ***UserKeyGen:*** Given the system public parameters $params$, user's identity $ID \in \{0,1\}^*$, the algorithm picks $s_{ID} \in Z_q^*$ at random and computes $PK_{ID} = s_{ID}P$, then the user's key pair is (s_{ID}, PK_{ID}).
- ***CertGen*:** Given the system public parameters $params$, the system master secret key s, user's identity ID and his public key PK_{ID}, the algorithm computes $Q_{ID} = H_0(ID \| PK_{ID})$ and outputs a certificate $Cert_{ID} = sQ_{ID}$, which can be verified by checking

whether the equation $e(Cert_{ID}, P) = e(mpk, Q_{ID})$ holds.

- ***Sign*:** Given a message m, the system public parameters $params$, user's identity ID and his private key s_{ID}, certificate $Cert_{ID}$, the algorithm performs as follows:

a) Picks $r \in Z_q^*$ at random and computes $U = rP$;

b) Computes $W_1 = H_1(m, U, PK_{ID})$,

$W_2 = H_2(m, ID, U, PK_{ID})$;

c) Computes $V = Cert_{ID} + s_{ID}W_1 + rW_2$;

d) Outputs the signature $\sigma = (U, V)$ on the message m.

- ***Verify:*** Given a message/signature pair (m, σ), the system public parameters $params$, the system master public key mpk, and user's public key PK_{ID}, the algorithm performs as follows:

a)Computes $Q_{ID} = H_0(ID \| PK_{ID})$,

$W_1 = H_1(m, U, PK_{ID})$,

$W_2 = H_2(m, ID, U, PK_{ID})$;

b)Checks whether the following equation holds, if so, outputs $true$, otherwise, outputs $false$.

$$e(V, P) = e(mpk, Q_{ID})e(W_1, PK_{ID})e(W_2, U)$$

4.2. The PS Scheme from CBS-to-PS

We construct a *PS* scheme from the previous *CBS* scheme which is described in section 4.1 by using our generic construction of *CBS-to-PS*. The constructed proxy signature scheme consists of the following algorithms.

- ***Setup:*** The system parameters generated are as same as the Section 4.1. The algorithm outputs the system public parameters :

$$params = \{G_1, G_2, e, q, P, H_0, H_1, H_2\}.$$

- ***KeyGen:*** The algorithm consists of the following two sub- algorithms:
    - ***OKeyGen:*** Given the system public parameters $params$, the original signer O picks a random number $s_O \in Z_q^*$, sets $SK_O = s_O$ and computes $PK_O = s_O P$. Returns (SK_O, PK_O) as the original signer O's key pair.
    - ***PKeyGen***: Given the system public parameters $params$, the proxy signer P picks a random number $s_P \in Z_q^*$, sets $SK_P = s_P$ and computes $PK_P = s_P P$. Returns (SK_P, PK_P) as the proxy signer P's key pair.

- ***DelGen*:** Given the system public parameters $params$, a warrant w, the original signer O's private key s_O and the proxy signer P's public key PK_P, the algorithm computes $Q_w = H_0(P \| w \| PK_P)$, $D_w = s_O Q_w$, outputs delegation D_w, which can be verified by checking whether the equation $e(D_w, P) = e(PK_O, Q_w)$ holds.

- ***PSign*:** Given a message $m \in \{0,1\}^*$, the system public parameters $params$, a warrant w and the corresponding delegation D_w, the proxy signer P's key pair (SK_P, PK_P), the algorithm performs as follows:

a) Picks $r \in Z_q^*$ at random and computes $U = rP$;

b) Computes $W_1 = H_1(m, U, PK_P)$,

$W_2 = H_2(m, P \| w, U, PK_P)$;

c) Computes $V = D_w + s_P W_1 + rW_2$;

d) Outputs the proxy signature $\sigma = (U, V)$ on the message m.

- ***Verify*:** Given a message/signature pair (m, σ), the system public parameters $params$, a warrant w and the corresponding delegation D_w, the original signer O's public key PK_O and proxy signer P's public key PK_P, the algorithm performs as follows:

a) Computes $Q_w = H_0(P \| w \| PK_P)$,

$W_1 = H_1(m, U, PK_P)$,

$W_2 = H_2(m, P \| w, U, PK_P)$;

b)Checks whether the following equation holds, if so, outputs $true$, otherwise, outputs $false$.

$$e(V, P) = e(PK_O, Q_w)e(W_1, PK_P)e(W_2, U)$$

5. CONCLUSION

In this paper, we firstly analyze the similarities and differences between the certificate-based signatures and the proxy signatures delegated by warrant. Secondly, we propose a generic construction of the proxy signatures *CBS-to-PS* from the certificate-based signatures, then, we prove that our *CBS-to-PS* scheme is secure against all types of adversaries of *PS* if the underlying *CBS* scheme is existentially unforgeable under adaptively chosen-message attack. Finally, we give a concrete *CBS-to-PS* scheme as an example to demonstrate the application of our generic construction. Further, we can also research the generic construction of the certificate-based signatures from the proxy signatures.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

The authors would like to thank anonymous referees and reviewers for their suggestions to improve this work. Besides, this work is supported by the Natural Science Foundation of China (No. 61170246 and No. 61373140) and the Natural Science Foundation of Fujian Province (No. 2012J01295).

REFERENCES

[1] C. Gentry, "Certificate-based Encryption and the Certificate Revocation Problem," In: *Proc. Eurocrypt*, Warsaw: Poland, 2003, pp. 272-293.

[2] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," In: *Proc. Cryptology-Crypto*, Barbara, California: USA, 1985, pp. 47-53.

[3] A.G. Kang, J.H. Park, and S.G. Hahn, "A Certificate-based Signature Scheme," In: *Proc. CT-RSA*, San Francisco, CA: USA, 2003, pp. 99-111.

[4] M. Girault, "Self-certificated public keys," In: *Proc. Eurocrypt*, Brighton: UK, 1991, pp. 490-497.

[5] J. Liu, M. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model," In: *Proc. 2nd ACM symposium on Information, computer and communications security*, Kyoto: Japan, 2007, pp. 273-283.

[6] J. Li, X.Y. Huang, Y. Mu, W. Susilo, and Q.H. Wu, "Certificate-based Signature: Security Model and Efficient Construction," In: *Proc. EuroPKI*, Palma de Mallorca: Spain, 2007, pp. 110-125.

[7] M.H. Au, J.K. Liu, W. Susilo, and T.H. Yuen, "Certificate Based (Linkable) Ring Signature," In: *Proc. Ispec*, Hong Kong: China, 2007, pp. 79-92.

[8] J. K. Liu, J. Baek, W. Susilo, and J. Zhou, "Cettificate-based Signature Scheme without Pairings or Random oracles" In: *Proc. ISC*, Taipei: Taiwan, 2008, pp. 285-297.

[9] J. Zhang, "On the security of a certificate-based signature scheme and its improvement with pairings," In: *Proc. ISPEC*, Xi'an: China, 2009, pp. 47-58.

[10] J.K. Liu, F. Bao, and J. Zhou, "Short and Efficient Certificate-Based Signature," In: *Proc. Networking Workshops*, Valencia: Spain, 2011, pp. 167-168

[11] J.G. Li, X.Y. Huang and Y.C. Zhang, "An efficient short certificate-based signature scheme," *J. Syst. Softw.*, vol. 85, no. 2, pp. 314-322, Feb. 2012.

[12] J.N. Chen, and Z.J. Huang, "Certificate-based Proxy Signatures," In: *Proc. IEEE Int. Conf.*, Chengdu: China, 2010, pp. 465-468.

[13] R.F. Huang, and Q. Nong, "Efficient certificate-based blind signature scheme without bilinear pairings," *Appl. Mech. Mat.*, vol. 220, pp. 2735-2739, 2012.

[14] M. Mambo, K. Usuda, and E. Okamoo, "Proxy signatures: Delegation of he power to sign messages," In: *Proc. IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 79, no. 9, pp. 1338-1354, September, 1996.

[15] B.C. Neuman, "Proxy based authorization and accounting for distributed systems," In: *Proc. 13th International Conference on Distributed Computing Systems*, Pittsburgh, PA: USA, 1993, pp. 283-291.

[16] B. Lee, H. Kim, and K. Kim, "Secure mobile agent using strong non-designated proxy signature," In: *Proc. ACISP,* Sydney: Australia, 2001, pp. 474-486.

[17] J. Dai, X. Yang, and J. Dong, "Designated receiver proxy signature scheme for electronic commerce," In: *Proc. International Conference on Systems, Man and Cybernetics, IEEE*, Washington: DC, 2003, pp. 384-389.

[18] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegaion signing operation," In: *Proc. 3rd ACM Conference computer and communications security*, New Delhi: India, 1996, pp. 48-57.

[19] V. Varadharajan, P. Allen, and S. Black, "An analysis of the proxy problem in distributed systems," In: *Proc. IEEE computer society symposium on research in security and privacy*, Oakland, CA: 1991, pp. 255-275.

[20] S. Kim, S. Park, and D. Won, "Proxy Signatures, Revisited," In: *Proc. 1st International Information and Communications Security*, Beijing: China, 1997, pp. 223-232.

[21] Y. Yu, Y. Mu, W. Susilo, Y. Sun, and Y.F. Ji, "Provably secure proxy signature scheme from factorization," *J. Math. Comput. Model.*, vol. 55, no. 3, pp. 1160-1168, 2012.

Received: September 22, 2014 Revised: November 03, 2014 Accepted: November 06, 2014