A Certificate-based Signature Scheme

Bo Gyeong Kang, Je Hong Park, and Sang Geun Hahn

Department of Mathematics, Korea Advanced Institute of Science and Technology,
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea
{snubogus,Jehong.Park,sghahn}@kaist.ac.kr

January 14, 2004

Abstract. In this paper, we propose the security notion of certificate-based signature that uses the same parameters and certificate revocation strategy as the encryption scheme presented at Eurocrypt 2003 by Gentry. Certificate-based signature preserves advantages of certificate-based encryption, such as implicit certification and no private key escrow. We present concrete certificate-based signature schemes derived from pairings on elliptic curves and prove their security in the random oracle model assuming that the underlying group is GDH. Additionally, we propose a concrete delegation-by-certificate proxy signature scheme which is derived from a certificate-based signature scheme after simple modifications. Our proxy scheme is provably secure in the random oracle model under the security notion defined by Boldyreva, Palacio and Warinschi.

1 Introduction

1.1 Certificate-based cryptosystem

In traditional public key signatures (PKS), the public key of the signer is essentially a random bit string picked from a given set. So, the signature does not provide the authorization of the signer by itself. This problem can be solved via a certificate which provides an unforgeable and trusted link between the public key and the identity of the signer by the CA's signature. And there is a hierarchical framework that is called a *public key infrastructure* (PKI) to issue and manage certificates. In general, the signer registers its own public key with its identity in certificate server and anyone wishing to obtain the signer's public key requests it by sending the server the identity of the signer and gets it. Before verifying a signature using the signer's public key, however, a verifier must obtain the signer's certification status, hence in general make a query on the signer's certificate status to the CA. It is called a *third-party query*. As mentioned in [11], even though the third-party query has some problems in public key encryptions, those problems can be surmounted in signature schemes simply by transmitting the certificate for its valid public key with its signature. Despite of this settlement, a verifier must verify the certificate first and if authorization of the CA about the signer's public key is valid then verifies the signed message with given public key from the signer. In the point of a verifier, two verification steps for independent signatures are needed. As a consequence, this system requires a large amount of computing time and storage when the number of users increases rapidly.

To simplify key management procedures of conventional PKIs, Shamir asked for ID-based cryptography (IBC) in 1984 [18], but recently Boneh and Franklin [2] proposed a practical ID-based encryption (IBE) scheme based on bilinear maps. Subsequently, several ID-based signature (IBS) schemes which share system parameters with the IBE scheme of Boneh and Franklin are proposed [16, 12, 8]. The main practical benefit of IBC is in greatly reducing the

need for, and reliance on, the public key certificates. But IBC uses a trusted third party called a *Private Key Generator* (PKG). The PKG generates the secret keys of all of its users, so a user can decrypt only if the PKG has given a secret key to it (so, certification is implicit), hence reduces the amount of storage and computation. On the other hand, private key *escrow* is inherent and secret keys must be sent over *secure channels*, making private key distribution difficult [11].

To import several merits of IBC into conventional PKIs, the concept of *certificate-based encryption* (CBE) was introduced by Gentry [11]. A CBE scheme which is created by combining a public key encryption (PKE) scheme and an IBE scheme consists of a certifier and users. Each user generates its own secret and public key pair and requests a certificate from the CA, then the CA uses the user private key generation algorithm in the Boneh-Franklin IBE scheme to generate certificates. That gives us implicit certification by virtue of the fact that a certificate can be used as a signing key, and so allows to eliminate third-party queries on certificate status. But this ordinary CBE scheme is inefficient when the CA has a large number of users and performs frequent certificate updates, so Gentry suggests to use subset covers to overcome inefficiency [11].

1.2 Our contributions

We give the first construction of a certificate-based signature (CBS) scheme that can use the same parameters and certificate revocation strategy as the CBE scheme of [11]. In Section 2, we define a formal security model of CBS, and describe two similar pairing-based CBS schemes which are secure in the random oracle model in Section 3. It is obvious that our schemes maintain most of the advantages of CBE over PKE and IBE. Both of these schemes use an IBS scheme for signing phase and the BLS signature scheme [4] for certificate issuing phase, but one scheme uses multisignatures, while the other does aggregate signatures as a temporary signing key, which provide implicit certification. Since the CA does not know user's personal secret key, CBS does not suffer from the key escrow property which is inherent in IBC and since the CA's certificate need not be kept secret, there is no secret key distribution problem.

We show in Section 4 that a delegation-by-certificate proxy signature scheme immediately follows. A proxy signature permits an entity to delegate its signing rights to another entity. The basic model of proxy signature schemes is that the original signer creates a signature on delegation information and gives it to the proxy signer, and then the proxy signer uses it to generate a proxy key pair. That is analogous to the certificate issuing and temporary signing key generation phases in CBS. Based on this fact, we make slight modifications to our CBS scheme, and prove that the resulting proxy signature scheme is secure in the random oracle model, assuming that the underlying group is GDH.

1.3 Related works

The concept of CBS is analogous to self-certified signatures [15], certificateless signatures [1] and IBS without TTP [9].

The general notion of self-certified signatures (SCS) proposed by Lee and Kim [15] is that a signer computes a temporary signing key with its secret key and certificate information together, and generates a signature on a message and certificate information using the temporary signing key. Then a verifier verifies both signer's signature on the message and

related certification information together. We can easily see that both SCS and CBS provide the authenticity of a digital signature and the authorization of a public key simultaneously. But there are some different aspects between SCS and CBS. The former does not concern the certificate revocation problem which is the main contribution of the latter. It only specifies how to sign a message and verify a signature using a long-lived key pair and the corresponding certificate together, and provides explicit authentication of a public key.

The notion of certificateless public key signature (CL-PKS) presented by Al-Riyami and Paterson [1] does not require the use of certificates. In CL-PKS, the Key Generation Center (KGC) supplies an user with a partial secret key which the KGC computes from the user's identity and a master key, and then the user combines its partial secret key and the KGC's public parameters with some secret information to generate its actual secret key and public key respectively. In this way, an user's secret key is not available to the KGC, whereas the KGC must send the partial secret keys over secure channels. In this case, it is assumed that the KGC is trusted not to replace users' public keys because a new public key could have been created by the KGC and it cannot be easily decided which is the case. This rather strong security assumption can be reduced by a slight modification that an user must first generate its public key and then bind it with its identity as the new identity of the user. The user sends it to the KGC to generate a partial secret key. This technique makes the trust level of the KGC apparent and equivalent to that of the CA in conventional PKIs. Independently, Chen, Zhang and Kim [9] apply the same idea of above modification to the IBS scheme of Cha and Cheon [8]. Although these schemes remove the key escrow property, they still require secure channels and are less efficient than our CBS scheme.

2 Preliminaries

In this section, we review some definitions and provide a formal security model necessary to build our signature scheme. We refer the reader to [2, 4, 10, 13] for a discussion of how to build a concrete instance using supersingular curves and compute the bilinear map.

2.1 Cryptographic assumptions

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of some large prime order q. We view $\mathbb{G}_1$ as an additive group and $\mathbb{G}_2$ as a multiplicative group. A bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ between these two groups which is called *admissible pairing* must satisfy the following properties [2, 11]:

1. Bilinear: $\hat{e}(aQ, bR) = \hat{e}(Q, R)^{ab}$ for all Q, $R \in \mathbb{G}_1$ and all a, $b \in (\mathbb{Z}/q\mathbb{Z})^*$.
2. Non-degenerate: $\hat{e}(Q, R) \neq 1$ for some Q, $R \in \mathbb{G}_1$.
3. Computable: There is an efficient algorithm to compute $\hat{e}(Q, R)$ for any Q, $R \in \mathbb{G}_1$.

From an admissible pairing $\hat{e}$, the decisional Diffie-Hellman (DDH) problem in $\mathbb{G}_1$ can be easily solved, since $\hat{e}(aP, bP) = \hat{e}(P, abP)$ implies that (P, aP, bP, cP) is a valid Diffie-Hellman tuple.

Definition 1. A prime order group $\mathbb{G}$ is a *GDH group* if there exists an efficient algorithm which solves the DDH problem in $\mathbb{G}$ and there is no polynomial time algorithm which solves the CDH problem.

A *GDH parameter generator* $\mathcal{IG}$ is a randomized algorithm that takes a security parameter $k \in \mathbb{N}$, runs in time polynomial in k, and outputs the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of

the same prime order q and the description of an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. We say that $\mathcal{IG}$ satisfies the *GDH assumption* if the following probability is negligible (in k) for all PPT algorithm $\mathcal{A}$:

$$\Pr[\mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP) = abP \,|\, (\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k), P \leftarrow \mathbb{G}_1^*, a, b \leftarrow (\mathbb{Z}/q\mathbb{Z})^*].$$

As noted in [8], a BDH parameter generator $\mathcal{IG}_{\mathsf{BDH}}$ [2] satisfying the BDH assumption can also be viewed as a GDH parameter generator $\mathcal{IG}_{\mathsf{GDH}}$ satisfying the GDH assumption because the BDH assumption is stronger than the GDH assumption.

2.2 ID-based signature scheme of Cha and Cheon

Recently, Cha and Cheon [8] proposed an IBS scheme which is not only efficient but also provably secure in the random oracle model assuming that the underlying group is GDH. This scheme consists of the following algorithms:

IBS.Setup: Choose a generator P of $\mathbb{G}_1$, pick a random $s \in \mathbb{Z}/q\mathbb{Z}$ and set $P_{\text{pub}} = sP$. Choose hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_3 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}/q\mathbb{Z}$. The system parameter is $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{\text{pub}}, H_1, H_3)$ and the master secret key is s.

IBS.Extr: Given an identity ID, compute the public key $Q_{\text{ID}} = H_1(\text{ID})$ and output the secret key $D_{\text{ID}} = sQ_{\text{ID}}$ associated to ID.

IBS.Sign: Given a secret key D_{ID} and a message m, pick a random number $r \in \mathbb{Z}/q\mathbb{Z}$ and output a signature $\sigma = (U, V)$ where $U = rQ_{\text{ID}}$, $h = H_3(m, U)$ and $V = (r + h)D_{\text{ID}}$.

IBS.Vrfy: To verify a signature $\sigma = (U, V)$ of a message m for an identity ID, check whether $(P, P_{\text{pub}}, U + hQ_{\text{ID}}, V)$, where $h = H_3(m, U)$ is a valid Diffie-Hellman tuple.

2.3 BLS signature and multisignature schemes

Here, we introduce a pairing-based signature scheme of Boneh, Lynn and Shacham [4], and a multisignature scheme of Boldyreva [5]. Let $\mathbb{G}_1$ be a GDH group of prime order q and let P be a generator of $\mathbb{G}_1$. The global information PARAMS contains P, q and a description of a hash function H mapping arbitrary strings to the elements of $\mathbb{G}_1^*$. A *BLS signature scheme* consists of the following algorithms:

BLS.Key: Given a security parameter PARAMS, choose a random element $x \in (\mathbb{Z}/q\mathbb{Z})^*$ and return a pair $(SK, PK) = (x, xP)$.

BLS.Sign: Given a secret key SK and a message $m \in \{0,1\}^*$, compute $H(m)$ and return a signature $\sigma = xH(m)$.

BLS.Vrfy: To verify a signature σ of a message m, check whether $(P, PK, H(m), \sigma)$ is a valid Diffie-Hellman tuple. If valid then return 1, else return 0.

Suppose n users each has a secret and public key pair (SK_i, PK_i) via running BLS.Key algorithm. For simplicity we are assigning the members consecutive integer identities $1, 2, \ldots, n$. Suppose user i signs a message $m \in \{0,1\}^*$ to obtain the signature σ_i via BLS.Sign algorithm. The multisignature of an arbitrary subset of $L \subseteq [n]$ is computed simply as $\sigma = \prod_{i \in L} \sigma_i \in \mathbb{G}_1^*$. To verify the multisignature σ on condition that public keys of all users in L are given, compute $PK_L = \prod_{i \in L} PK_i$ and check whether $(P, PK_L, H(m), \sigma)$ is a valid Diffie-Hellman tuple. If valid, then return 1 else return 0.

Both the BLS and induced multisignature scheme are proven to be secure in the random oracle model assuming that the underlying group $\mathbb{G}_1$ is GDH.

2.4 The model

We now provide a formal definition of certificate-based signature schemes and their security. Our definition parallels the definition of a CBE scheme of Gentry. As stated in [11], it does not necessarily have to be "certificate updating". Two main entities involved in CBS are a certifier and a user. This model does not require a secure channel between the two entities.

Definition 2. A *certificate-updating certificate-based signature scheme* consists of the following algorithms:

- $\mathsf{CBS.Gen}_{\mathsf{IBS}}$, the IBS *key generation* algorithm, takes as input a security parameter 1^{k_1} and (optionally) the total number of time periods t. It returns SK_C (the certifier's master secret) and public parameters PARAMS that include a public key PK_C, and the description of a string space $\mathcal{S}$.
- $\mathsf{CBS.Gen}_{\mathsf{PKS}}$, the PKS *key generation* algorithm, takes as input a security parameter 1^{k_2} and (optionally) the total number of time periods t. It returns a secret key SK_U and public key PK_U (the user's secret and public keys).
- CBS.Upd1, the *certifier update* algorithm, takes as input SK_C, PARAMS, i, string $s \in \mathcal{S}$ and PK_U at the start of time period i. It returns CERT'_i, which is sent to the user.
- CBS.Upd2, the *user update* algorithm, takes as input PARAMS, i, CERT'_i and (optionally) CERT_{i-1} at the start of time period i. It returns CERT_i.
- CBS.Sign, the *signature generation* algorithm, takes $(m, \text{PARAMS}, \text{CERT}_i, SK_U)$ as input in time period i. It computes the temporary signing key $SK = f(SK_U, \text{CERT}'_i)$ where f is public algorithm, and outputs a signature σ.
- CBS.Vrfy, the *verification* algorithm, takes $(\sigma, m, i, PK_C, PK_U)$ as input and outputs a binary value 0 (invalid) or 1 (valid).

As the formal model for CBE, CBS is designed as a combination of PKS and IBS, where the signer need both its personal secret key and a certificate from the CA to sign. The string s includes a message that the certifier signs and may be changed depending on the scheme.

Security Roughly speaking, we are concerned with two different types of attacks by an uncertified user and by the certifier, as considered in CBE. We want CBS to be secure against each of these entities, even though each basically has *half* of the secret information needed to sign. Accordingly, we define different two games and the adversary chooses one game to play. In Game 1, the adversary essentially assumes the role of an uncertified user. After proving knowledge of the secret key corresponding to its claimed public key, it can make CBS.Sign and CBS.Upd1 queries. In Game 2, the adversary essentially assumes the role of the certifier. After proving knowledge of the master secret corresponding to its claimed PARAMS, it can make CBS.Sign queries. Let $\text{PID} = (i, PK_C, PK_U, \text{USINFO})$ be a match for a user U's ID in IBC and call it by *pseudo ID*.

Game 1: The challenger runs $\mathsf{IBS.Gen}(1^{k_1}, t)$, and gives PARAMS to the adversary. The adversary then issues CBS.Cert and CBS.Sign queries. These queries are answered as follows:

- On certification query (PID, SK_U), the challenger checks that SK_U is the secret key corresponding to PK_U in PID. If so, it runs CBS.Upd1 and returns CERT'_i; else returns $\perp$.

- On sign query (PID, SK_U, m), the challenger checks that SK_U is the secret key corresponding to PK_U in PID. If so, it generates CERT_i and outputs a valid signature $\mathsf{CBS.Sign}(m, \text{PARAMS}, \text{CERT}_i, SK_U)$; else it returns $\perp$.

The adversary outputs (PID, m, σ), where $\text{PID} = (i, PK_C, PK_A, \text{ASINFO})$, m is a message and σ is a signature, such that PID and (PID, m) are not equal to the inputs of any query to CBS.Cert and CBS.Sign, respectively. The adversary wins the game if σ is a valid signature of m for i.

Game 2: The challenger runs $\mathsf{CBS.Gen}_{\mathsf{PKS}}(1^{k_2}, t)$, and gives PK_U to the adversary. The adversary then issues CBS.Sign query.

- On CBS.Sign query $(\text{PID}, SK_C, \text{PARAMS}, m)$, the challenger checks that SK_C is the secret key corresponding to PK_C in PARAMS. If so, it generates CERT_i and outputs a valid signature $\mathsf{CBS.Sign}(m, \text{PARAMS}, \text{CERT}_i, SK_U)$; else returns $\perp$.

The adversary outputs (PID, m, σ), such that (PID, m) is not equal to the inputs of any query to CBS.Sign. The adversary wins the game if σ is a valid signature of m for i.

Definition 3. A certificate-updating certificate-based signature scheme is *secure against existential forgery under adaptively chosen message and pseudo ID attacks* if no PPT adversary has non-negligible advantage in either Game1 or Game2.

3 Concrete certificate-based signature schemes

We describe two concrete certificate-based signature schemes called CBSm and CBSa. They use an IBS scheme in common, but as a temporary signing key, the former uses multisignatures and the latter does aggregate signatures[1]. Let k be the security parameter given to the setup algorithm, and let $\mathcal{IG}$ be a GDH parameter generator. Both of them have the same setup and certificate update algorithm.

CBS.Setup: The CA runs $\mathcal{IG}$ on input k to generate groups $\mathbb{G}_1$, $\mathbb{G}_2$ of some prime order q and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Then picks an arbitrary generator $P \in \mathbb{G}_1$ and a random secret $s_C \in \mathbb{Z}/q\mathbb{Z}$, and sets $PK_C = s_C P$. Chooses cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$, and $H_3 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}/q\mathbb{Z}$.

The system parameters are $\text{PARAMS} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, PK_C, H_1, H_3)$ and the CA's master secret key is $SK_C = s_C \in \mathbb{Z}/q\mathbb{Z}$. The CA uses its parameters and its secret to issue certificates. And Alice computes a secret and public key pair as $(SK_A, PK_A) = (s_A, s_A P)$ according to the parameters issued by the CA.

CBS.Cert: Alice obtains a certificate from his CA as follows.

1. Alice sends ALICESINFO to the CA, which includes her public key $s_A P$ and any necessary additional identifying information, such as her name.
2. The CA verifies Alice's information;
3. If satisfied, the CA computes $P_A = H_1(i, PK_C, PK_A, \text{ALICESINFO}) \in \mathbb{G}_1$ in period i.
4. The CA then computes $\text{CERT}_A = s_C P_A$ and sends this certificate to Alice.

In CBSm, before signing a message $m \in \{0,1\}^*$, Alice signs ALICESINFO, producing $s_A P_A$ and then computes $S_A = s_C P_A + s_A P_A = \text{CERT}_A + s_A P_A$, which is a two person multisignature. Alice will use this multisignature as his temporary signing key.

[1] We refer the reader to [3] for a discussion of multisignatures and aggregate signatures

CBSm.Sign: To sign $m \in \{0,1\}^*$ using ALICESINFO, picks a random $r \in \mathbb{Z}/q\mathbb{Z}$ and outputs a signature $\sigma = (U, V)$ where $U = rP_A$, $h = H_3(m, U)$ and $V = (r + h)S_A = (r + h)(s_C + s_A)P_A$.

CBSm.Vrfy: To verify a signature $\sigma = (U, V)$ of a message m, checks whether $e(s_C P + s_A P, U + hP_A) = e(P, V)$, where $h = H_3(m, U)$.

In CBSa, before signing a message $m \in \{0,1\}^*$, Alice also signs ALICESINFO, producing $s_A P'_A$ where $P'_A = H_1(\text{ALICESINFO})$. And she computes her temporary signing key $S_A = s_C P_A + s_A P'_A = \text{CERT}_A + s_A P'_A$, which is a two person aggregate signature.

CBSa.Sign: To sign $m \in \{0,1\}^*$ using ALICESINFO, Alice does the following:

1. Computes $P_A = H_1(i, PK_C, s_A P, \text{ALICESINFO}) \in \mathbb{G}_1$.
2. Picks a random $r \in \mathbb{Z}/q\mathbb{Z}$ and outputs a signature $\sigma = (U_1, U_2, V)$ where $U_1 = rP_A$, $U_2 = rP'_A$, $h = H_3(m, U_1, U_2)$ and $V = (r + h)S_A = (r + h)(s_C P_A + s_A P'_A)$.

CBSa.Vrfy: To verify a signature $\sigma = (U_1, U_2, V)$ of a message m, checks $e(PK_C, U_1 + hP_A) \cdot e(PK_A, U_2 + hP'_A) = e(P, V)$, where $h = H_3(m, U_1, U_2)$.

As a note, CBSm can be vulnerable to the following "chosen-key" attack [3] induced by multisignatures. If a malicious signer $\mathcal{A}$ given a secret and public key pair $(s_\mathcal{A}, s_\mathcal{A} P)$ sets $PK'_\mathcal{A} = s_\mathcal{A} P - s_C P$ as its public key whose corresponding secret key it does not know, then $\mathcal{A}$ can generate a valid temporary signing key $S_\mathcal{A} = s_\mathcal{A} P_\mathcal{A} (= \text{CERT}_\mathcal{A} + (s_\mathcal{A} - s_C)P_\mathcal{A})$ by himself, without a valid certificate of period i. To prevent this attack, CBSm is required to have one assumption that the signer must provide a separate proof that it knows the secret key corresponding to its claimed public key and then the verifier must check this separate proof, but the verifier only has to do this once over the lifetime of the public key of the signer, so this extra verification cost may be amortized. For example, the notion of "long-lived certificate" in [11] can be used directly as a role of the separate proof. This auxiliary assumption prevents some malicious users from doing chosen-key attacks and, what is more, not a burden to the implementation of CBSm.

On the other hand, CBSa uses the temporary signing keys obtained by aggregate signatures instead of multisignatures. It prevents the chosen-key attack even if we do not consider above assumption and is provably secure under the security notion in subsection 2.4, naturally. But the verification time is slower than that of CBSm (3 pairing computations instead of 2 are needed).

Remark 1. As stated above, even though CBSm has an auxiliary assumption, it is quite acceptable since extra cost may be ignored. Furthermore, CBSm induces proxy signatures immediately and efficiently in conventional PKIs. So, we intend to focus on multisignature based CBS (i.e. CBSm) for the rest of this paper.

Remark 2. The IBS scheme of Hess [12] can be used in place of the scheme of Cha and Cheon, but the latter is rather efficient than the former in general case [7].

Security Proof A pseudo ID is the input value of H_1 to derive a certificate from the CA in period i. We modify the notion of security in subsection 2.4, which is acceptable for CBSm schemes. An adversary in Game 2 is allowed to make BLS.Sign queries. Independently, we need to prohibit a signature forged by the chosen-key attack from being accepted as a valid one. Thus it is required that a forged signature (PID, m, σ) is accepted as a *valid* one only when it

comes with the secret key corresponding to the public key in PID. Without loss of generality, we say that a certificate-based signature scheme is *secure against existential forgery under adaptively chosen message and pseudo ID attacks* if no PPT adversary has a non-negligible advantage in the following one of games:

Game 1: As the Game 1 in subsection 2.4 except the notion of forged signature validity.

Game 2: Addition to the queries of the Game 2 in subsection 2.4, the adversary is allowed to issue BLS.Sign query. This query is answered as follows:

- On BLS sign query (PID, SK_C), the challenger checks that SK_C is the secret key corresponding to the public key PK_C in PARAMS. If so, it returns $\mathsf{BLS.Sign}(SK_U, \text{PID})$; else returns $\perp$.

The adversary outputs (PID, m, σ) and SK_C. It is valid when PID and (PID, m) are not equal to the inputs of any query to BLS.Sign and CBS.Sign respectively, and SK_C is the secret key corresponding to the public key in PID. The adversary wins the game if σ is a valid signature of m for PID.

For notational purposes, the result of the CBS.Sign query will be denoted by (PID, m, U, h, V) where (U, V) is the output of the signing algorithm of our scheme and $h = H_1(m, U)$.

Theorem 1. *Our certificate-based signature scheme is secure against existential forgery under adaptively chosen message and pseudo ID attacks assuming the underlying group is GDH.*

The proof of the above theorem is given in Appendix A.

Theorem 2. *An aggregate signature based CBS scheme (CBSa) is also secure against existential forgery under adaptively chosen message and pseudo ID attacks.*

As stated above, this theorem can be proved exactly under the security notion in subsection 2.4. The proof of the above theorem is given in Appendix B.

Remark 3. As a certificate-based encryption scheme, adapting CBS to a hierarchy of CAs is fairly obvious. In this case, we need to use aggregate signatures instead of multisignatures because of certification of CAs.

4 Proxy signature schemes

Next, we show an application of CBS to proxy signatures. The concept of proxy signatures was first introduced by Mambo, Usuda and Okamoto in 1996. A proxy signature scheme which consists of an original signer, a proxy signer and verifiers, allows the original signer to delegate its signing capability to the proxy signer, to sign messages on its behalf. From a proxy signature, anyone can check both the original signer's delegation and the proxy signer's digital signature.

Recently, Boldyreva, Palacio and Warinschi [6] formalize a notion of security for proxy signatures and show that secure proxy signature schemes can be derived from secure standard signature schemes. But they focus on the case that one digital signature scheme is used for standard signing, proxy designation and proxy signing, simultaneously.

4.1 Definition and security notion of proxy signature schemes

The basic idea to implement a secure proxy signature scheme is that the original signer creates a signature on the delegation information (warrant[2]) and then the proxy signer uses it to generate a proxy secret key and signs on the delegated message. Since the proxy key pair is generated using the original signer's signature on delegation information, any verifier can check the original signer's agreement from the proxy signature. For simplicity, let users be identified by natural numbers, PK_i denote the public key of user $i \in \mathbb{N}$, and SK_i denote the corresponding secret key. Then a *proxy signature scheme* consists of eight algorithms. Three algorithms $\mathsf{PS.Key}, \mathsf{S.Sign}$ and $\mathsf{S.Vrfy}$ are as in ordinary signature schemes. The other five algorithms provide the proxy signature capability.

- $(\mathsf{PS.Del}, \mathsf{PS.Pro})$, a pair of interactive algorithms forming the *proxy designation protocol*, takes as input (PK_i, PK_j) for the original signer i and the proxy signer j in common. Each $\mathsf{PS.Del}$ and $\mathsf{PS.Pro}$ also takes as input SK_i and SK_j, respectively. As result of the interaction, $\mathsf{PS.Pro}$ outputs a proxy signing key SKP.
- $\mathsf{PS.Sign}$, the *proxy signature generation* algorithm, takes as input (m, SKP), and outputs a proxy signature $p\sigma$.
- $\mathsf{PS.Vrfy}$, the *proxy verification* algorithm, takes as input $(p\sigma, m, PK_i)$, and outputs 0 (invalid) or 1 (valid).
- $\mathsf{PS.Iden}$, the *proxy identification* algorithm, takes as input $p\sigma$, and outputs PK_j.

For all messages m and all users $i, j \in \mathbb{N}$, if SKP is a proxy signing key for user j on behalf of user i, then $\mathsf{PS.Vrfy}(\mathsf{PS.Sign}(m, SKP), m, PK_i) = 1$ and $\mathsf{PS.Iden}(\mathsf{PS.Sign}(m, SKP)) = PK_j$.

Chosen message attack capabilities are formed by providing the adversary access to two oracles: a standard signing oracle and a proxy signing oracle. The first oracle takes input a message m, and returns a standard signature for m by user 1. The second oracle takes input a tuple (i, l, m), and if user 1 was designated by user i at least l times, returns a proxy signature for m created by user 1 on behalf of user i, using the l-th proxy signing key. The goal of the adversary is to produce one of the following forgeries:

1. a standard signature by user 1 for a message that was not submitted to the standard signing oracle.
2. a proxy signature for a message m by user 1 on behalf of some user i such that either user i never designated user 1 or m was not in a query (i, l, m) made to the proxy signing oracle, or
3. a proxy signature for a message m by some user i on behalf of user 1, such that user i was never designated by user 1.

Formal notion of security for proxy signatures defined in [6] is arranged in Appendix C due to space limitation.

4.2 A concrete proxy signature scheme

We construct a delegation-by-certificate proxy signature scheme which is derived from CBS. Contrary to the examples in [6], we use the BLS signature scheme for standard signing different

[2] A warrant is a message containing the public key of the designated proxy signer and possibly restrictions on the messages the proxy signer is allowed to sign.

from the proxy signing. After all, our proxy signature scheme employs the BLS signature scheme for standard signing and for delegation, and allows an IBS scheme for proxy signing.

Assume that there are two participants, Charlie and Alice with secret and public key pairs $(s_C, s_C P)$ and $(s_A, s_A P)$ respectively, and that they have the common system parameters $\text{PARAMS} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, H_1, H_3)$.

- S.Sign: A standard signature for message m is obtained by signing the result using BLS.Sign.
- S.Vrfy: The verification of a signature σ for a message m is done by computing BLS.Vrfy.
- (PS.Del, PS.Pro): In order to designate Alice as a proxy signer, Charlie simply sends to Alice an appropriate warrant w together with a signature $\text{CERT}_A = s_C P_A$, where $P_A = H_1(PK_C, PK_A, w)$. The corresponding proxy signing key of Alice is $SKP_A = \text{CERT}_A + s_A P_A$.
- PS.Sign: A proxy signature for message m produced by Alice on behalf of Chalie, contains a warrant w, the public key of the proxy signer PK_A, and signature $\sigma = (U, V)$ where $U = rP_A$, $h = H_3(m, U)$ and $V = (r + h)SKP_A = (r + h)(s_C + s_A)P_A$.
- PS.Vrfy: To verify a signature $(PK_C, m, (PK_A, w, \sigma))$, checks whether $e(PK_C + PK_A, U + hP_A) = e(P, V)$, where $h = H_3(m, U)$.
- PS.Iden: The identification algorithm is defined as $\mathsf{PS.Iden}(PK_A, w, \sigma) = PK_A$.

In our proxy signature scheme, the role of the CA in CBS schemes is transformed to the original signer, so trust of certificate provider may be removed. And the signer's information to be signed by the CA for certification in CBS schemes is issued conversely from the original signer as a warrant for delegation. Due to a merit of CBS, our proxy scheme does not need to include a signature for the warrant under the secret key of the original signer in the proxy signature. And it does not require a secure channel for proxy designation [14]. The following theorem shows that our proxy scheme is secure under the security notion of [6].

Theorem 3. *The scheme defined above is a secure proxy signature scheme in the random oracle model assuming that the underlying group is GDH.*

The proof of the above theorem is in Appendix D.

Remark 4. Recently, the proxy signature scheme using the same idea is proposed by Zhang, Safavi-Naini and Lin [19]. It is based on the IBS scheme of Hess [12], and uses the BLS signature scheme for standard signature and for certification of warrant. Though their scheme also holds desirable and implicit security conditions, it does not guarantee a provable security. We make sure that our work bridges this gap.

5 Conclusion

In this paper, we defined the security notion of certificate-based signature using the same parameters and certificate revocation strategy as the encryption scheme by Gentry. We presented and compared two concrete CBS schemes, and provided proofs of security in the random oracle model assuming that the underlying group is GDH. Our scheme may be useful to construct an efficient PKI combining the CBE scheme of Gentry. Additionally, we derived a concrete delegation-by-certificate proxy signature scheme from a certificate-based signature scheme through simple modifications and proved its security under the security notion defined by Boldyreva, Palacio and Warinschi.

Acknowledgement

The authors would like to thank Craig Gentry, Jung Hee Cheon, Adriana Palacio and the anonymous referees of CT-RSA 2004 for many helpful discussions and comments. This work was supported by grant no. R01-2002-000-00151-0 from the Basic Research Program of the KOSEF.

References

1. S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. *Advances in Cryptology - ASIACRYPT 2003*, C.S. Laih (Ed.), Lecture Notes in Comput. Sci. **2894**, Springer-Verlag, pp. 452–473 (2003).
2. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, **32**(3), pp. 586–615 (2003). A preliminary version appeared in *Advances in Cryptology - CRYPTO 2001*, J. Kilian (Ed.), Lecture Notes in Comput. Sci. **2139**, Springer-Verlag, pp. 231–229 (2001).
3. D. Boneh, C. Gentry, B. Lynn and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *Advances in Cryptology - EUROCRYPT 2003*, E. Biham (Ed.), Lecture Notes in Comput. Sci. **2656**, Springer-Verlag, pp. 416–432 (2003).
4. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. *Advances in Cryptology - ASIACRYPT 2001*, C. Boyd (Ed.), Lecture Notes in Comput. Sci. **2248**, Springer-Verlag, pp. 514-532 (2001).
5. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. *Public Key Cryptography - PKC 2003*, Y.G. Desmedt (Ed.), Lecture Notes in Comput. Sci. **2567**, pp. 31–46 (2003).
6. A. Boldyreva, A. Palacio and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. *Cryptology ePrint Archive*, Report **2003/096**.
7. X. Boyen. Multipurpose identity-based signcryption - A Swiss army knife for identity-based cryptography. *Advances in Cryptology - CRYPTO 2003*, D. Boneh (Ed.), Lecture Notes in Comput. Sci. **2729**, Springer-Verlag, pp. 382–398 (2003).
8. J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. *Public Key Cryptography - PKC 2003*. Lecture Notes in Comput. Sci. **2567**, Y.G. Desmedt (Ed.), Springer-Verlag, pp. 18–30 (2003).
9. X. Chen, F. Zhang and K. Kim. A new ID-based group signature scheme from bilinear pairings. *Cryptology ePrint Archive*, Report **2003/116**.
10. S.D. Galbraith. Supersingular curves in cryptography. *Advances in Cryptology - ASIACRYPT 2001*, Lecture Notes in Comput. Sci. **2248**, C. Boyd (Ed.), Springer-Verlag, pp. 495–513 (2001).
11. C. Gentry. Certificate-based encryption and the certificate revocation problem. *Advances in Cryptology - EUROCRYPT 2003*, Lecture Notes in Comput. Sci. **2656**, E. Biham (Ed.), Springer-Verlag, pp. 272–293 (2003).
12. F. Hess. Efficient identity based signature scheme based on pairings. *Selected Areas in Cryptography - SAC 2002*, K. Nyber and H. Heys (Eds.), Lecture Notes in Comput. Sci. **2595**, Springer-Verlag, pp. 310–324 (2003).
13. A. Joux. The Weil and Tate pairings as building blocks for public key cryptosystems. *Algorithmic Number Theory - ANTS V*, C. Fieker and D.R. Kohel (Eds.), Lecture Notes in Comput. Sci. **2369**, Springer-Verlag, pp. 20–32 (2002).
14. J.-Y. Lee, J.H. Cheon and S. Kim. An analysis of proxy signatures: is a secure channel necessary? *Topics in Cryptology - CT-RSA 2003*, M. Joye (Ed.), Lecture Notes in Comput. Sci. **2612**, Springer-Verlag, pp. 68–79 (2003).
15. B. Lee and K. Kim. Self-certified signatures. *Progress in Cryptology - INDOCRYPT 2002*, A. Menezes and P. Sarkar (Eds.), Lecture Notes in Comput. Sci. **2551**, Springer-Verlag, pp. 199–214 (2002).
16. K.G. Paterson. ID-based signatures from pairings on elliptic curves. *Electron. Lett.*, **38**(18), 1025–1026 (2001).
17. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology* **13**(4), 361–396 (2000).
18. A. Shamir. Identity-based cryptosystems and signature schemes. *Advances in Cryptology - CRYPTO'84*, Lecture Notes in Comput. Sci. **196**, G.R. Blakley and D. Chaum (Eds.), Springer-Verlag, pp. 47–53 (1985).
19. F. Zhang, R. Safavi-Naini and C.-Y. Lin. New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairings. *Cryptology ePrint Archive*, Report **2003/104**.

A Proof of Theorem 1

Our proof uses the same technique in Cha and Cheon's proof of [8, Theorem 1]. Fix $k \in \mathbb{N}$ and let $\mathcal{A}$ be a PPT adversary against our CBS scheme in the adaptively chosen message and pseudo ID attacks scenario. We will construct either a new PPT adversary $\mathcal{B}$ which attempts to solve the CDH problem on $\mathbb{G}_1$ or $\mathcal{C}$ which makes an existential forgery on the IBS scheme of Cha and Cheon [8]. Whenever $\mathcal{A}$ makes a query to $\mathcal{B}$ or $\mathcal{C}$, they must check that the secret key which comes with PID or (PID, m), corresponds to the public key in PID. As noted in section 2.4, this makes challengers $\mathcal{B}, \mathcal{C}$ be able to sign and also prevents adversaries to choose public key without knowing corresponding secret key. We skip this procedure in the following proof because the formal security assumes that. Without loss of generality, we assume that the adversary does not repeat any random oracle queries since it can just store the responses in a table. The adversary $\mathcal{A}$ chooses one of games to play.

In the case of Game 1, let $\mathcal{A}$ be a PPT adversary against our CBS scheme assuming the role of an uncertified user. $\mathcal{A}$ is given fixed PK_C and attacks our CBS scheme in the adaptively chosen message and pseudo ID. Using the same method in the proof of [8, Lemma 1], we can construct a PPT adversary $\mathcal{A}_1$ of Game 1 in the adaptively chosen message and *given* pseudo ID, which is derived from $\mathcal{A}$. That is, $\mathcal{A}_1$ outputs a CBS forgery for a given pseudo ID as its final result. So it is sufficient to show that $\mathcal{A}_1$ yields a PPT algorithm $\mathcal{B}$ solving the CDH problem on $\mathbb{G}_1$ as follows:

The new algorithm $\mathcal{B}$ is given a challenge (P, aP, bP) and the goal of $\mathcal{B}$ is to compute abP. For that purpose, $\mathcal{B}$ gives $PK_C = aP$ with fixed $\mathrm{PID}^* = (i^*, PK_C, PK_A, \textsc{Asinfo})$ and s_A to $\mathcal{A}_1$ where $PK_A = s_A P$ for randomly chosen $s_A \in \mathbb{Z}/q\mathbb{Z}$. Then $\mathcal{B}$ simulates H_1', $\mathsf{CBS.Cert}'$ and $\mathsf{CBS.Sign}'$ to answer queries given by $\mathcal{A}_1$ as follows. For randomly chosen $x_j \in \mathbb{Z}/q\mathbb{Z}$,

$$P_j = H_1'(\mathrm{PID}_j) = \begin{cases} bP & \text{if } \mathrm{PID}_j = \mathrm{PID}^* \\ x_j P & \text{otherwise,} \end{cases}$$
$$\mathsf{CBS.Cert}'(\mathrm{PID}_j) = x_j PK_C = aP_j,$$
$$\mathsf{CBS.Sign}'(\mathrm{PID}_j, m_{j_k}) = (m_{j_k}, U_{j_k}, h_{j_k}, V_{j_k}).$$

where $\mathrm{PID}_j = (i_j, PK_C, PK_{A_j}, \textsc{Ajsinfo})$, and $U_{j_k} = y_{j_k} P - h_{j_k} P_j$ and $V_{j_k} = y_{j_k}(PK_C + PK_j)$ for randomly chosen $y_{j_k}, h_{j_k} \in \mathbb{Z}/q\mathbb{Z}$. Then H_1', $\mathsf{CBS.Cert}'$ and $\mathsf{CBS.Sign}'$ generate indistinguishable distributions in the view of $\mathcal{A}_1$ since H_1' has a random distribution, $\mathsf{CBS.Cert}'$ generate $aP_j = x_j PK_C$ which is valid signature of a certifier with master secret key a, and signatures computed by $\mathsf{CBS.Sign}'$ can be verified with respect to PK_C, PK_{A_j}, H_1' and $\mathsf{CBS.Cert}'$. But we have to consider a problem of collisions of the query result of $\mathsf{CBS.Sign}'$ and H_3. $\mathcal{B}$ stores h_{j_k} as the value of $H_3(m_{j_k}, U_{j_k})$ whenever $\mathcal{B}$ returns $\mathsf{CBS.Sign}$ query of $\mathcal{A}_1$ on $(\mathrm{PID}_j, m_{j_k})$. Since U_{j_k} is randomly chosen, the probability of collision that $H_3(m_{j_k}, U_{j_k})$ is already defined is negligible. If no collisions have appeared, the collusion of the attacker $\mathcal{A}_1$ and above simulation performs a no-message attack for a given pseudo ID, PID^*. We apply the forking lemma [17, Theorem 13] to our scheme, after replaying $\mathcal{A}_1$ with the same random tape, $\mathcal{B}$ obtains two signatures $(\mathrm{PID}^*, m, U, h, V)$ and $(\mathrm{PID}^*, m, U, h', V')$ which are expected to be valid ones with respect to hash functions H_3 and H_3' having different values $h \neq h'$ on (m, U), respectively. Then $V = (a + s_A)(U + h(bP))$ and $V' = (a + s_A)(U + h'(bP))$, so by subtracting the equations, $V - V' = (h - h')(a + s_A)(bP)$, thus $abP = (h - h')^{-1}(V - V') - s_A(bP)$ is obtained.

In the case of Game 2, let $\mathcal{A}$ be a PPT adversary against our CBS scheme assuming the role of the certifier. $\mathcal{A}$ attacks our CBS scheme in the adaptively chosen message and pseudo ID. In this setting, we only concern the one honest user, namely Alice, and construct a PPT algorithm $\mathcal{C}$ attacking the based IBS scheme in the adaptively chosen message and ID attacks. The new algorithm $\mathcal{C}$ is given an IBS system parameter $\text{PARAMS} = (P, aP, \mathbb{G}_1, \mathbb{G}_2, H_1, H_3)$ and access four oracles $H_1, H_3, \mathsf{IBS.Extr}^{H_1}(a, \cdot)$ and $\mathsf{IBS.Sign}^{H_1,H_3}(a, \cdot, \cdot)$. The goal of $\mathcal{C}$ is to output a valid signature forgery of an $\text{ID} \in \{0,1\}^*$. For the purpose, $\mathcal{C}$ gives $PK_A = aP$ and ALICEINFO to $\mathcal{A}$ and then allows $\mathcal{A}$ to run. $\mathcal{C}$ answers about queries made by $\mathcal{A}$ using its own four oracles. If $\mathcal{A}$ makes hash queries on $\text{PID}_j = (i_j, PK_{C_j}, PK_A, \text{ALICESINFO})$ and (m_k, U_k), then $\mathcal{C}$ submits them to random oracle H_1 and H_3 respectively, and returns the responses to $\mathcal{A}$. If $\mathcal{A}$ makes a sign query on (PID_j, m_{j_k}), then $\mathcal{C}$ submits it to signing oracle $\mathsf{IBS.Sign}^{H_1,H_3}(a, \cdot, \cdot)$ and gets a valid signature (U, V). And $\mathcal{C}$ returns $V + s_j(U + hP_j)$ to $\mathcal{A}$ where s_j is a secret key corresponding to PK_{C_j} in PID_j. Additionally, if $\mathcal{A}$ makes a BLS signature query on PID_j, then $\mathcal{C}$ submits it to the IBS extraction oracle $\mathsf{IBS.Extr}^{H_1}(a, \cdot)$ and returns the response to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a valid signature forgery $(\text{PID}^*, m, \sigma)$ and s_C where $\sigma = (U, V)$, $\text{PID}^* = (i, PK_C, PK_A, \text{ALICESINFO})$, and s_C is a secret key corresponding to PK_C in PID^*. As a result, $\mathcal{C}$ returns $(U, \bar{V})$ where $\bar{V} = V - s_C(U + hP_A) = a(U + hP_A)$ and $P_A = H_1(\text{PID}^*)$, as an IBS forgery of (PID^*, m). Since $(P, aP, U + hP_A, \bar{V})$ is a valid Diffie-Hellman tuple and queried to neither $\mathsf{IBS.Extr}^{H_1}(a, \cdot)$ nor $\mathsf{IBS.Sign}^{H_1,H_3}(a, \cdot, \cdot)$ oracle. So, $(U, \bar{V})$ is a valid IBS forgery.

B Proof of Corollary 2

Let $\mathcal{A}$ be a PPT adversary against CBSa in the formal security model in section 2.4. We will construct a new PPT adversaries $\mathcal{B}$ and $\mathcal{C}$ which attempt to solve the CDH problem in $\mathbb{G}_1$. As noted in the formal security model, $\mathcal{A}$ makes queries of the form PID or (PID, m) revealing the secret key corresponds to the public key in PID. We skip this procedure in the following proof. Without loss of generality, we assume that the adversary does not repeat any random oracle queries since it can just store the responses in a table. The adversary $\mathcal{A}$ chooses one of games to play.

In the case of Game 1, let $\mathcal{A}$ be a PPT adversary against CBSa assuming the role of an uncertified user. $\mathcal{A}$ is given fixed PK_C and attacks CBSa in the adaptively chosen message and pseudo ID. Using the same method in the proof of [8, Lemma 1], we can construct a PPT adversary $\mathcal{A}_1$ of Game 1 in the adaptively chosen message and *given* pseudo ID, which is derived from $\mathcal{A}$. That is, $\mathcal{A}_1$ outputs CBSa forgery for a given pseudo ID as its final result. So it is sufficient to show that $\mathcal{A}_1$ yields a PPT algorithm $\mathcal{B}$ solving the CDH problem on $\mathbb{G}_1$ as follows:

The new algorithm $\mathcal{B}$ is given a challenge (P, aP, bP) and the goal of $\mathcal{B}$ is to compute abP. For that purpose, $\mathcal{B}$ gives $PK_C = aP$ with fixed $\text{PID}^* = (i^*, PK_C, PK_A, \text{ASINFO})$ and s_A to $\mathcal{A}_1$ where $PK_A = s_AP$ for randomly chosen $s_A \in \mathbb{Z}/q\mathbb{Z}$. Then $\mathcal{B}$ simulates H_1', $\mathsf{CBS.Cert}'$ and $\mathsf{CBS.Sign}'$ to answer queries given by $\mathcal{A}_1$ as follows. Note that $\mathcal{A}_1$ can make two types of H_1'-queries. One is of the form $\text{PID}_j = (i_j, PK_C, PK_{A_j}, \text{AJSINFO})$ where i_j, PK_{A_j} are a time period and the public key that it wants to be certified respectively, and AJSINFO is other

information contained in a certificate. The other is of the form AJSINFO. Then

$$P_j = H_1'(\mathrm{PID}_j) = \begin{cases} bP & \text{if } \mathrm{PID}_j = \mathrm{PID}^* \\ x_j P & \text{otherwise,} \end{cases}$$
$$P_j' = H_1'(\textsc{Ajsinfo}) = \begin{cases} z(bP) & \text{if } \textsc{Ajsinfo} = \textsc{Asinfo} \\ z_j P & \text{otherwise,} \end{cases}$$
$$\mathsf{CBS.Cert}'(\mathrm{PID}_j) = x_j PK_C = aP_j,$$

where $\mathrm{PID}_j = (i_j, PK_C, PK_{A_j}, \textsc{Ajsinfo})$ and randomly chosen $x_j, z, z_j \in \mathbb{Z}/q\mathbb{Z}$. Furthermore, $\mathcal{B}$ can simulate $\mathsf{CBS.Sign}$ query as follows: On input $(\mathrm{PID}_j, m_{j_k})$ given by $\mathcal{A}_1$, there are three cases to answer the query. If $\mathrm{PID}_j = \mathrm{PID}^*$, then computes $U_1 = y_1 P - h(bP), U_2 = y_2 P - hz(bP)$ and $V = y_1 PK_C + y_2 PK_A$, for random chosen y_1, h in $\mathbb{Z}/q\mathbb{Z}$ with $y_2 = zy_1$. Else if $\mathrm{PID}_j = (i', PK_C, PK_A, \textsc{Asinfo})$ where $i' \neq i$, then computes $U_1 = y_1 P - hP_j, U_2 = y_2(bP) - hP_A'$, and $V = y_1 PK_C + y_2(s_A(bP))$ for randomly chosen y_1, h in $\mathbb{Z}/q\mathbb{Z}$ with $y_1 z = x_j y_2$ and $P_A' = H_1'(\textsc{Asinfo})$. Else, computes $U_1 = y_1 P - hP_j, U_2 = y_2 P - hP_j'$, and $V = y_1 PK_C + y_2 PK_{A_j}$, for randomly chosen y_1, h in $\mathbb{Z}/q\mathbb{Z}$ with $y_1 z_j = x_j y_2$. Then returns $(m_{j_k}, U_1, U_2, h, V)$ as an answer. Then H_1', $\mathsf{CBS.Cert}'$ and $\mathsf{CBS.Sign}'$ generate indistinguishable distributions in the view of $\mathcal{A}_1$ since H_1' has a random distribution, $\mathsf{CBS.Cert}'$ generate $aP_j = x_j PK_C$ which is valid signature of a certifier with master secret key a, and signatures computed by $\mathsf{CBS.Sign}'$ can be verified with respect to PK_C, PK_{A_j}, H_1' and $\mathsf{CBS.Cert}'$. But we have to consider a problem of collisions of the query result of $\mathsf{CBS.Sign}'$ and H_3. $\mathcal{B}$ stores h as the value of $H_3(m_{j_k}, U_1, U_2)$ whenever $\mathcal{B}$ returns $\mathsf{CBS.Sign}$ query of $\mathcal{A}_1$ on $(\mathrm{PID}_j, m_{j_k})$. Since (U_1, U_2) is randomly chosen, the probability of collision that $H_3(m_{j_k}, U_1, U_2)$ is already defined is negligible. If no collisions have appeared, the collusion of the attacker $\mathcal{A}_1$ and above simulation performs a no-message attack for a given pseudo ID, PID^*. We apply the forking lemma to CBSa, after replaying $\mathcal{A}_1$ with the same random tape, $\mathcal{B}$ obtains two signatures $(\mathrm{PID}^*, m, (U_1', U_2'), h, V)$ and $(\mathrm{PID}^*, m, (U_1', U_2'), h', V')$ which are expected to be valid ones with respect to hash functions H_3 and H_3' having different values $h \neq h'$ on (m, U_1', U_2'), respectively. Then $V = a(U_1' + h(bP)) + s_A(U_2' + hP_A')$ and $V' = a(U_1' + h'(bP)) + s_A(U_2' + h'P_A')$, so by subtracting the equations, $V - V' = (h - h')(abP + s_A P_A')$, thus $abP = (h - h')^{-1}(V - V') - s_A P_A'$ is obtained.

In the case of $\mathsf{Game\ 2}$, let $\mathcal{A}$ be a PPT adversary against CBSa assuming the role of the certifier. $\mathcal{A}$ attacks CBSa in the adaptively chosen message and pseudo ID. In this setting, we only concern the one honest user, namely Alice. $\mathcal{A}$ is given fixed PK_A and ALICESINFO and outputs CBSa forgery for a pseudo ID. But using the same method in the proof of [8, Lemma 1], we can construct a PPT adversary $\mathcal{A}_1$ of $\mathsf{Game\ 2}$ in the adaptively chosen message and *given* pseudo ID, which is derived from $\mathcal{A}$. That is, $\mathcal{A}_1$ outputs CBSa forgery for a given pseudo ID as its final result. So it is sufficient to show that $\mathcal{A}_1$ yields a PPT algorithm $\mathcal{C}$ solving the CDH problem on $\mathbb{G}_1$ as follows:

The new algorithm $\mathcal{C}$ is given challenge (P, aP, bP) and the goal of $\mathcal{C}$ is to compute abP. For that purpose, $\mathcal{C}$ gives $PK_A = aP$, ALICEINFO with fixed $\mathrm{PID}^* = (i^*, PK_C, PK_A, \textsc{Alicesinfo})$ and s_C to $\mathcal{A}_1$ where $PK_C = s_C P$ for randomly chosen $s_C \in \mathbb{Z}/q\mathbb{Z}$. Then allows $\mathcal{A}_1$ to run. $\mathcal{C}$ simulates H_1' and $\mathsf{CBS.Sign}'$ to answer queries given by $\mathcal{A}_1$ as follows. Note that $\mathcal{A}_1$ can make two types of H_1'-queries. One is of the form $\mathrm{PID}_j = (i_j, PK_{C_j}, PK_A, \textsc{Alicesinfo})$ where i_j, PK_{C_j} are a time period and a public key of the CA chosen by $\mathcal{A}_1$ respectively. The other is ALICESINFO. For randomly chosen $x_j \in \mathbb{Z}/q\mathbb{Z}$, $P_j = H_1'(\mathrm{PID}_j) = x_j(bP)$ and $P_A' =$

$H_1'(\text{ALICESINFO}) = bP$ where $\text{PID}_j = (i_j, PK_{C_j}, PK_A, \text{ALICESINFO})$. On input (PID_j, m_{j_k}) given by $\mathcal{A}_1$, $\mathcal{C}$ simulates $\mathsf{CBS.Sign}$ query as follows: $\mathcal{C}$ computes $U_1 = y_1P - hP_j$, $U_2 = y_2P - hP_A'$ and $V = y_1PK_{C_j} + y_2PK_A$ for randomly chosen y_1, h in $\mathbb{Z}/q\mathbb{Z}$ with $y_1 = x_jy_2$ and returns $(m_{j_k}, U_1, U_2, h, V)$ as the response. Then H_1' and $\mathsf{CBS.Sign}'$ generate indistinguishable distributions in the view of $\mathcal{A}_1$ since H_1' has a random distribution and signatures computed by $\mathsf{CBS.Sign}'$ can be verified with respect to PK_{C_j}, PK_A and H_1'. But we have to consider a problem of collisions of the query result of $\mathsf{CBS.Sign}'$ and H_3. $\mathcal{C}$ stores h as the value of $H_3(m_{j_k}, U_1, U_2)$ whenever $\mathcal{C}$ returns $\mathsf{CBS.Sign}$ query of $\mathcal{A}_1$ on (PID_j, m_{j_k}). Since (U_1, U_2) is randomly chosen, the probability of collision that $H_3(m_{j_k}, U_1, U_2)$ is already defined is negligible. If no collisions have appeared, the collusion of the attacker $\mathcal{A}_1$ and above simulation performs a no-message attack for a given pseudo ID, PID^*. We apply the forking lemma to our scheme, after replaying $\mathcal{A}_1$ with the same random tape, $\mathcal{C}$ obtains two signatures $(\text{PID}^*, m, (U_1', U_2'), h, V)$ and $(\text{PID}^*, m, (U_1', U_2'), h', V')$ which are expected to be valid ones with respect to hash functions H_3 and H_3' having different values $h \neq h'$ on (m, U_1', U_2'), respectively. Then $V = s_C(U_1' + hP_A) + a(U_2' + h(bP))$ and $V' = s_C(U_1' + h'P_A) + s_A(U_2' + h'(bP))$ where $P_A = H_1'(\text{PID}^*)$, so by subtracting the equations, $V - V' = (h - h')(s_CP_A + a(bP))$, thus $abP = (h - h')^{-1}(V - V') - s_CP_A$ is obtained.

C Security of a proxy signature scheme

Let $\mathsf{PS} = (\mathsf{PS.Key}, \mathsf{S.Sign}, \mathsf{S.Vrfy}, (\mathsf{PS.Del}, \mathsf{PS.Pro}), \mathsf{PS.Sign}, \mathsf{PS.Vrfy}, \mathsf{PS.Iden})$ be a proxy signature scheme. First, a secret and public key pair (SK_1, PK_1) for user 1 is generated via $\mathsf{PS.Key}(1^k)$. A counter n for the number of users is initialized to 1, and an empty array $\mathbf{skp}$ and an empty set D are created. Adversary $\mathcal{A}$ is given input PK_1 and it can make the following requests or queries, in any order and any number of times.

- i registers PK_i: $\mathcal{A}$ can request to register a public key PK_i for user $i = n+1$ by outputting a pair (SK_i, PK_i) of matching secret and public keys. These keys are stored, counter n is incremented, and an empty array $\mathbf{skp}_i$ is created.
- 1 designates i: $\mathcal{A}$ can request to interact with user 1 running $\mathsf{PS.Del}(SK_1, PK_1, PK_i)$ for some $i \in \{2, \ldots, n\}$, and play the role of user i running $\mathsf{PS.Pro}(PK_1, SK_i, PK_i,)$; after a successful run, D is set to $D \cup \{PK_i\}$.
- i designates 1: $\mathcal{A}$ can request to interact with user 1 running $\mathsf{PS.Pro}(PK_i, SK_1, PK_1)$ for some $i \in \{2, \ldots, n\}$, and play the role of user i running $\mathsf{PS.Del}(SK_i, PK_i, PK_1)$. The private output SKP of $\mathsf{PS.Pro}$ is stored in the last unoccupied position of $\mathbf{skp}_i$. We emphasize that $\mathcal{A}$ does not have access to the elements of $\mathbf{skp}_i$.
- standard signature by 1: $\mathcal{A}$ can query oracle $\mathsf{S.Sign}(SK_1, \cdot)$ with a message m and obtain a standard signature σ for m by user 1.
- proxy signature by 1 on behalf of i using the l-th proxy signing key: $\mathcal{A}$ can make a query (i, l, m), where $i \in [n]$, $l \in \mathbb{N}$ and $m \in \{0,1\}^*$, to oracle $\mathsf{PS.Sign}((\mathbf{skp}_u)_{u \in [n]}, \cdot, \cdot, \cdot)$. If key $\mathbf{skp}_i[l]$ has already been defined, we say the query is *valid* and the oracle returns $\mathsf{PS.Sign}(\mathbf{skp}_i[l], m)$; if $\mathbf{skp}_i[l]$ has not been defined, the query is said to be *invalid* and the oracle returns $\perp$.

Eventually, $\mathcal{A}$ outputs a forgery (m, σ) or $(PK, m, p\sigma)$. The output of the experiment is determined as follows:

1. If the forgery is of the form (m, σ), where $\mathsf{S.Vrfy}(PK_1, m, \sigma) = 1$, and m was not queried to oracle $\mathsf{S.Sign}(SK_1, \cdot)$, then return 1.

2. If the forgery is of the form $(PK, m, p\sigma)$, where PK is equal to PK_i for some $i \in [n]$, $\mathsf{PS.Vrfy}(PK_i, m, p\sigma) = 1$, $\mathsf{PS.Iden}(p\sigma) = PK_1$, and no valid query (i, l, m), for $l \in \mathbb{N}$, was made to the proxy signing oracle $\mathsf{PS.Sign}((\mathbf{skp}_u)_{u\in[n]}, \cdot, \cdot)$, then return 1.
3. If the forgery is of the form $(PK_1, m, p\sigma)$, where $\mathsf{PS.Vrfy}(PK_1, m, p\sigma) = 1$, $\mathsf{PS.Iden}(p\sigma) \notin D \cup \{PK_1\} \cup \{\perp\}$, and $\mathsf{PS.Iden}(p\sigma)$ is equal to PK_i for some $i \in [n]$, then return 1.
4. Otherwise, return 0.

We define the *advantage* of adversary $\mathcal{A}$ as

$$\mathrm{Adv}^{\mathsf{ps-uf}}_{\mathsf{PS},\mathcal{A}}(k) = \Pr[\mathbf{Exp}_{\mathsf{PS},\mathcal{A}}(k) = 1].$$

We adopt the convention that the *time complexity* of adversary $\mathcal{A}$ is the execution time of the entire experiment, including the time taken for parameter and key generation, and computation of answers to oracle queries. We say that PS is a *secure proxy signature scheme* if the function $\mathrm{Adv}^{\mathsf{ps-uf}}_{\mathsf{PS},\mathcal{A}}(\cdot)$ is negligible for all adversaries $\mathcal{A}$ of time complexity polynomial in the security parameter k.

Remark 5. We do not consider the case that the signer designates himself for simplicity.

D Proof of Theorem 3

Let $\mathcal{A}$ be a PPT adversary for our proxy signature scheme. We will construct either a PPT adversary $\mathcal{B}$ which makes an existential forgery on the BLS signature scheme or adversary $\mathcal{C}$ which makes an existential forgery on the IBS scheme of Cha and Cheon. Denote $\mathrm{PID}_{i,j}[l]$ be a tuple $(PK_i, PK_j, w_{i,j}[l])$ where user i designates user j as proxy signer at l-th time and warrant $w_{i,j}[l]$ contains information of user j and so on. As is usual in the random oracle model, hash functions G, H_1 and H_3 used in the scheme are assumed to behave as random oracles. Without loss of generality, we assumed that the adversary do not repeat any random oracle queries and there will be no restriction on the number of executions in our proof.

At first, we will construct a PPT adversary $\mathcal{B}$ that breaks the BLS signature scheme. $\mathcal{B}$ is given PK_1, and access to a random oracle G and a BLS signing oracle $\mathsf{BLS.Sign}^G(s_1, \cdot)$, where s_1 is a secret key corresponding to PK_1. $\mathcal{B}$ uses its own oracles and simulates random oracles H_1, H_3 and proxy signing oracle to answer $\mathcal{A}$'s requests and queries as described in detail below.

$\mathcal{B}$ initializes a counter $n = 1$ for the number of users, creates empty arrays $\mathbf{wskp}_1$ and hash tables for G, H_1 and H_3 respectively, chooses some randomness for $\mathcal{A}$, and runs $\mathcal{A}$ on input PK_1 with this randomness handling all of $\mathcal{A}$'s requests and answering all $\mathcal{A}$'s queries as follows:

- (i registers PK_i:) If $\mathcal{A}$ requests to register a new user $i = n + 1$ by outputting a secret and public key pair $(SK_i, PK_i) = (s_i, s_iP)$, then $\mathcal{B}$ stores these keys, increments n and creates an empty array $\mathbf{wskp}_i$.
- (1 designates i:) If $\mathcal{A}$ requests to interact user 1 with $\mathsf{PS.Del}(SK_1, PK_1, PK_i)$, where $i \in \{2, \ldots, n\}$, playing the role of $\mathsf{PS.Pro}(PK_1, SK_i, PK_i)$, $\mathcal{B}$ creates an appropriate warrant $w_{1,i}[l]$ and defines $H_1(\mathrm{PID}_{1,i}[l]) = x_{i_l}P$ and $\textsc{Cert}_{1,i}[l] = x_{i_l}PK_1 (= s_1(x_{i_l}P))$ for random $x_{i_l} \in \mathbb{Z}/q\mathbb{Z}$. Then $\mathcal{B}$ forwards $(w_{1,i}[l], \textsc{Cert}_{1,i}[l])$ to $\mathcal{A}$. After a successful run, D is set to $D \cup \{PK_i\}$.

- (i designates 1:) If $\mathcal{A}$ requests to interact with $\mathsf{PS.Pro}(PK_i, PK_1, SK_1)$, where $i \in \{2, \ldots, n\}$, playing the role of $\mathsf{PS.Del}(PK_i, PK_1, SK_i)$, when $\mathcal{A}$ output $(w_{i,1}[l], \textsc{Cert}_{i,1}[l])$ upon receiving an answer of hash query H_1 on $\mathrm{PID}_{i,1}[l]$, $\mathcal{B}$ verifies that $\textsc{Cert}_{i,1}[l]$ is valid certificate of user i for $\mathrm{PID}_{i,1}[l]$. Note that queries to H_1 in this case are answered with a freshly chosen random value. If so, stores $(w_{i,1}[l], \textsc{Cert}_{i,1}[l])$ in the last unoccupied position of $\mathbf{wskp}_i$.
- (standard signature by 1:) If $\mathcal{A}$ queries a standard signature by 1 with a message m, $\mathcal{B}$ makes query m to its own oracle $\mathsf{BLS.Sign}^G(s_1, \cdot)$ and forwards the response to $\mathcal{A}$.
- (proxy signature by 1 on behalf of i using the l-th proxy signing key:) If $\mathcal{A}$ makes a query (i, l, m), where $i \in [n]$, $l \in [T]$ and $m \in \{0,1\}^*$, to its oracle $\mathsf{PS.Sign}((\mathbf{wskp}_u)_{u\in[n]}, \cdot, \cdot)$, $\mathcal{B}$ responds as follows. If $\mathbf{wskp}_i[l]$ is not defined, it returns $\perp$ to $\mathcal{A}$. Otherwise, let $(w_{i,1}[l], \textsc{Cert}_{i,1}[l])$ be the content of this position. Define $U = yP - hH_1(\mathrm{PID}_{i,1}[l])$ and $V = y(PK_1 + PK_i)$ for random $y, h \in \mathbb{Z}/q\mathbb{Z}$. The fact that simulated signature (U, V) has the same distribution as a real signature can be induced from the proof of security of CBS (see Appendix A) or IBS [8]. Then $\mathcal{B}$ stores $H_3(m, U) = h$ and returns $(PK_i, m, p\sigma)$ where $p\sigma = (PK_1, w_{i,1}[l], (U, V))$ as a proxy signature.

Eventually $\mathcal{A}$ outputs a forgery. If it is not of the form (m, σ), $\mathcal{B}$ aborts; otherwise, $\mathcal{B}$ outputs (m, σ) as a valid BLS signature forgery.

Next, we will construct a PPT adversary $\mathcal{C}$ against the IBS scheme of Cha and Cheon. The new forger $\mathcal{C}$ is given PARAMS and $PK_C = PK_1 = s_1P$ as a public key of PKG and access to random oracles H_1, H_3 and its own oracles $\mathsf{IBS.Extr}^{H_1}(s_1, \cdot)$ and $\mathsf{IBS.Sign}^{H_1,H_3}(s_1, \cdot, \cdot)$ to answer $\mathcal{A}$'s requests and queries as describes in detail below. $\mathcal{C}$ initializes a counter $n = 1$ for the number of users, creates empty arrays $\mathbf{wskp}_1$ and hash tables for G, H_1 and H_3, chooses some randomness for $\mathcal{A}$, and runs $\mathcal{A}$ on input PK_1 with this randomness handling all of $\mathcal{A}$'s requests and answering all $\mathcal{A}$'s queries as follows:

- (i registers PK_i:) If $\mathcal{A}$ requests to register a new user $i = n + 1$ by outputting a secret and public key pair $(SK_i, PK_i) = (s_i, s_iP)$, then $\mathcal{C}$ stores these keys, increments n and creates an empty array $\mathbf{wskp}_i$.
- (1 designates i:) If $\mathcal{A}$ requests to interact user 1 with $\mathsf{PS.Del}(SK_1, PK_1, PK_i)$, where $i \in \{2, \ldots, n\}$, playing the role of $\mathsf{PS.Pro}(PK_1, SK_i, PK_i)$, $\mathcal{C}$ creates an appropriate warrant $w_{1,i}[l]$ and makes a query $\mathrm{PID}_{1,i}[l]$ to the oracle $\mathsf{IBS.Extr}^{H_1}(s_1, \cdot)$. Then sets the response $\textsc{Cert}_{1,i}[l]$ and forwards $(w_{1,i}[l], \textsc{Cert}_{1,i}[l])$ to $\mathcal{A}$. After a successful run, D is set to $D \cup \{PK_i\}$.
- (i designates 1:) If $\mathcal{A}$ requests to interact with $\mathsf{PS.Pro}(PK_i, PK_1, SK_1)$, where $i \in \{2, \ldots, n\}$, playing the role of $\mathsf{PS.Del}(PK_i, PK_1, SK_i)$, when $\mathcal{A}$ outputs $(w_{i,1}[l], \textsc{Cert}_{i,1}[l])$ upon receiving an answer of hash query H_1 of $\mathrm{PID}_{i,1}[l]$, $\mathcal{C}$ verifies that $\textsc{Cert}_{i,1}[l]$ is a valid for message $\mathrm{PID}_{i,1}[l]$. If so, stores $(w_{i,1}[l], \textsc{Cert}_{i,1}[l])$ in the last unoccupied position of $\mathbf{wskp}_i$.
- (standard signature by 1:) If $\mathcal{A}$ queries a standard signature by 1 with a message m, $\mathcal{C}$ chooses a random $x \in \mathbb{Z}/q\mathbb{Z}$, put $G(m) = xP$ and forwards $xPK_1 (= s_1(xP))$. Clearly, the distribution of simulated BLS signatures is indistinguishable from answers of the BLS signing oracle $\mathsf{BLS.Sign}^G(s_1, \cdot)$.
- (proxy signature by 1 on behalf of i using the l-th proxy signing key:) If $\mathcal{A}$ makes a query (i, l, m), where $i \in [n]$, $l \in [T]$ and $m \in \{0,1\}^*$, to its oracle $\mathsf{PS.Sign}^{H_1,H_3}((\mathbf{wskp}_u)_{u\in[n]}, \cdot, \cdot)$, $\mathcal{C}$ responds as follows. If $\mathbf{wskp}_i[l]$ is not defined, it returns $\perp$ to $\mathcal{A}$. Otherwise, let $(w_{i,1}[l], \textsc{Cert}_{i,1}[l])$ be the content of this position. $\mathcal{C}$ makes query $(\mathrm{PID}_{i,1}[l], m)$ to its signing oracle $\mathsf{IBS.Sign}^{H_1,H_3}(s_1, \cdot, \cdot)$. Upon receiving an answer (U, V), $\mathcal{C}$ computes $V' =$

$s_i(U + hH_1(\mathrm{PID}_{i,1}[l]))$ where $h = H_3(m, U)$. Then C returns $(PK_1, w_{i,1}[l], p\sigma)$ to $\mathcal{A}$ where $p\sigma = (PK_i, w_{i,1}[l], (U, V + V'))$.

Eventually $\mathcal{A}$ outputs a forgery. If it is of the form (m, σ), $\mathcal{C}$ aborts; otherwise, we can divide two cases.

One is that $\mathcal{A}$ outputs a forgery$(PK_i, m, p\sigma)$ for some $i \in [n]$ where $\mathsf{PS.Vrfy}(PK_i, m, p\sigma) = 1$, $\mathsf{PS.Iden}(p\sigma) = PK_1$ and no query (i, l, m) was made to $\mathsf{PS.Sign}^{H_1,H_3}((\mathbf{wskp}_u)_{u\in[n]}, \cdot, \cdot)$. So $\mathcal{C}$ can transform $\sigma = (U, V)$ included in $p\sigma$ into an IBS forgery $(U, \bar{V})$ of $(\mathrm{PID}_{i,1}[l], m)$, where $\bar{V} = V - s_i(U + hH_1(\mathrm{PID}_{i,1}[l]))$ and $h = H_3(m, U)$. Additionally, what no query (i, l, m) was made to the proxy signing oracle $\mathsf{PS.Sign}^{H_1,H_3}((\mathbf{wskp}_u)_{u\in[n]}, \cdot, \cdot)$ implies that $\mathrm{PID}_{i,1}[l]$ and $(\mathrm{PID}_{i,1}[l], m)$ are not queried to $\mathsf{IBS.Extr}^{H_1}(s_1, \cdot)$ and $\mathsf{IBS.Sign}^{H_1,H_3}(s_1, \cdot, \cdot)$, respectively. So $(U, \bar{V})$ is a valid IBS forgery of $(\mathrm{PID}_{i,1}[l], m)$.

The other is that $\mathcal{A}$ outputs a forgery $(PK, m, p\sigma)$ where $\mathsf{PS.Vrfy}(PK_1, m, p\sigma) = 1$, $\mathsf{PS.Iden}(p\sigma) = PK \notin D \cup \{PK_1\} \cup \{\perp\}$, and $\mathsf{PS.Iden}(p\sigma)$ is equal to PK_i for some $i \in [n]$. In this case, $\mathcal{C}$ can transform $\sigma = (U, V)$ in $p\sigma$ into an IBS forgery $(U, \bar{V})$ of $(\mathrm{PID}_{1,i}[l], m)$, where $\bar{V} = V - s_i(U + hH_1(\mathrm{PID}_{1,i}[l]))$. Furthermore, $\mathsf{PS.Iden}(p\sigma) = PK \notin D \cup \{PK_1\} \cup \{\perp\}$ implies that user i was never designated by 1, so $\mathcal{B}$ did not make queries $\mathrm{PID}_{1,i}[l]$ and $(\mathrm{PID}_{1,i}[l], m)$ to its oracles $\mathsf{IBS.Extr}^{H_1}(s_1, \cdot)$ and $\mathsf{IBS.Sign}^{H_1,H_3}(s_1, \cdot, \cdot)$, respectively. That is, $(U, \bar{V})$ is a valid IBS forgery of $(\mathrm{PID}_{1,i}[l], m)$.

Each adversary $\mathcal{B}$ and $\mathcal{C}$ perfectly simulate the environment provided to $\mathcal{A}$ in experiment $\mathbf{Exp}_{\mathcal{PS},\mathcal{A}}(k) = 1$. After the above experiment simulated by $\mathcal{B}$ and $\mathcal{C}$, if $\mathcal{A}$ wins with non-negligible probability then at least one of the forgeries 1, 2 or 3 must occur with non-negligible probability. If the event 1 occurs, then $\mathcal{B}$ can transform the forgery into a BLS signature forgery. And If the events 2 or 3 occur, then $\mathcal{C}$ can transform the forgery into an IBS forgery. This implies that the advantage of either $\mathcal{B}$ or $\mathcal{C}$ is non-negligible, as desired.