Provably Secure Delegation-by-Certification Proxy Signature Schemes

Zuowen Tan, Zhuojun Liu *

Institute of Systems Science, AMSS, Chinese Academy of Sciences,
State Key Laboratory of Information Security,
Graduate School of Chinese Academy of Sciences

{ztan, zliu}@mmrc.iss.ac.cn

Abstract

In this paper, we first show that previous proxy signature schemes by delegation with certificate are not provably secure under adaptive-chosen message attacks and adaptive-chosen warrant attacks. The schemes do not provide the strong undeniability. Then we construct a proxy signature scheme by delegation with certificate based on Co-GDH group from bilinear map. Our proxy signature scheme is existentially unforgeable against adaptive-chosen message attacks and adaptive-chosen warrant attacks in random oracle model. We adopt a straight method of security reduction in which our scheme's security is reduced to hardness of the computational co-Diffie-Hellem problem. The proposed signature scheme is the first secure delegation-by-certificate proxy signature based on co-GDH groups from bilinear maps under the formal security model in random oracle model.

Key Words: digital signature, proxy signature, bilinear map, co-GDH groups, provably secure.

1 Introduction

Digital signature schemes can provide the authenticity and integrity of the digital documents. However, digital signature schemes cannot directly be applied to the proxy situation. Mambo et al. [11] introduced a new concept proxy signature to solve the problem. In a proxy signature scheme, an entity called original signer can delegate his signing power to another entity called proxy signer, and the proxy signer can sign message on behalf of the original. After the signature

*Partially Supported by National Science Foundation of China(10371127)

verifier receives the proxy signature, he/she can not only check the validity of the signature and identify the proxy signer, but also can be convinced of the original's agreement on the signed message. According to the delegation type, Mambo et al. classify proxy signature schemes into full delegation, partial delegation and delegation by warrant schemes [11]. In full delegation, the original signer sends its private key as the proxy signature key to the proxy via a secure channel. The original signer's standard signature is indistinguishable from the proxy signature. In partial delegation, the proxy signer has a proxy signature key obtained from the proxy signer's private key and a delegation key from the original. The delegation key is generated by the original through a trap-door permutation of the original signer's private key. The proxy signature is different from both the original's standard signature and the proxy's standard signature. In delegation by certificate, the original uses its standard signature algorithm to sign a warrant which includes the type of the information delegated, both the parties' identities and the period of delegation, etc. The signature of the warrant is called certificate, which prevents the transfer of proxy power to a third party. Verification of a proxy signature contains two phases. Since Mambo et al. introduced proxy signature, many proxy signature schemes have been proposed, such as threshold proxy signatures [13,16,20], nominative proxy signatures [12], one-time proxy signatures [6,19], multi-proxy signature [4], proxy multi-signature [3] and proxy blind signature [8,17], etc. Now the proxy signature schemes have been suggested for numerous applications including mobile agent [5,9,10], mobile communications [4], and electronic voting, etc.

Most of the research work on proxy signature focuses on constructing a more efficient scheme. One always thought that if the underlying signature scheme is secure, the proxy signature scheme must be secure. However, it is not true. Almost every other paper breaks some previously proposed proxy signature scheme [14,15,18] and proposes a new scheme. Since those schemes lack provably-security guarantee, to date very few schemes are left unbroken. In 2001, J. Lee, H. Kim and K. Kim [9] gave five strong security properties that a secure proxy signature scheme should hold: verifiability, strong unforgeability, strong non-repudiation, strong identifiability and prevention of misuse. But these informal requirements cannot give a precise meaning of security for proxy signature schemes. Boldyreva et al. formalized a notion of security for proxy signature schemes [2], specifying an adversary's capabilities and goal, and indicating the situation when an attacker is considered successful. In this model, the adversary is allowed to corrupt the original signer or the proxy signer and learns their secret keys, even the

proxy signature key. The adversary can see the transcripts of all the executions of delegation, i.e., the model does not require a secure channel. The adversary's goal is to generate a standard signature of the original signer, a standard signature of the proxy signer, or a proxy signature on a new message. Boldyreva et al. gave a slight modification to the delegation-by-certificate scheme (Henceforth BPW proxy scheme) [2] and claimed the resulting scheme is secure on the assumption the underlying standard signature is secure. This is the first work on proxy signatures in the provable-security direction.

In the paper, we will present an attack mounted by the original or the proxy on BPW proxy scheme [2]. We give a new notion of security for delegation-by-warrant proxy signature and show that our delegation-by-certificate scheme is secure on the assumption the underlying standard signature is secure. The security model and analysis are detailed in Section 2.

The other main contribution of the paper is construction of a delegation-by-warrant proxy signature from the co-GDH signature scheme based on bilinear maps [1]. The new proxy scheme is proved secure in the random oracle model under the computational co-Diffie-Hellem assumption. We reduce our scheme's security to hardness of computational co-Diffie-Hellem problem. If the scheme is not secure against existential forgery under adaptive chosen-message attacks and adaptive chosen-warrant attacks, we could construct an algorithm which can solve the computational co-Diffie-Hellem problem with a probability non-negligible in the security parameter. Unlike the proof for the security for Triple Schnorr scheme [2], we give the bound about the security parameter. The result will be given in Section 3 and Section 4.

2 Analysis of BPW's Proxy Signature Scheme

2.1 BPW's delegation-by-certificate proxy signature scheme

Let $DS = (G, K, S, V)$ be a signature scheme. The algorithms of the corresponding delegation-by-certificate proxy signature scheme $PS[DS]$=(G_1,K_1,S_1,V_1, (D, P), PS, PV, ID) are defined as follows.

- The randomized parameter-generation algorithm G takes input 1^k, where k is security parameter, and outputs some global parameters.
- The randomized key-generation algorithm K takes as input global parameters and outputs a public/secret key pair (pk, sk).
- The signing algorithm S takes as input a secret key sk and a message M, and outputs a signature σ.

- The deterministic verification algorithm V takes as input a public key pk, a message M and a signature σ for M, and outputs 0 if σ is invalid or 1 if σ is valid.
- The parameter-generation algorithm $G_1 = G$, the key-generation algorithm $K_1 = K$, the signing algorithm $S_1(sk, M) = S(sk, 11||M)$, and the verification $V_1(pk, M) = V(pk, 11||M, \sigma)$.
- (D, P) is a pair of interactive randomized algorithms forming the proxy-designation protocol. D takes as input the secret key of the original signer and the public keys of the two parties involved, but D has no local output. P takes as input the secret key of the proxy signer and the public keys of the two parties. The result of the interaction is $(\omega, cert)$.
- If User i designates user j as a proxy signer, i sends to j a warrant ω together with a signature **cert** for message $00||\omega$ by using the signature algorithm S under the secret key of user i.
- The user j produces the proxy signature which contains message M, the warrant ω, the certificate **cert** and a signature for $01||M$ under sk_j.

$$PS(sk_j, w, cert, M) = (w, cert, M, S(sk_j, 01||M)).$$

- Proxy signature verification is defined.

$$PV(pk, M, (\omega, pk', cert, \sigma)) = V(\omega, 00||\omega, cert) \wedge V(pk', 01||M, \sigma).$$

- The identification algorithm is defined as $ID(\omega, pk', \sigma) = pk'$.

Boldyreva et al. claim that if DS is a secure digital signature scheme, the delegation-by-certificate proxy signature scheme $PS[DS]$ specified above is a secure proxy signature scheme [2]. The result follows from the security of the aggregate-signature-based proxy signature schemes.

2.2 Analysis of BPW's delegation-by-certificate proxy signature

We will show that Boldyreva et al.'s statement [2] is improper. Let O be an original signer and P a proxy signer. (pk_O, sk_O) and (pk_P, sk_P) are the public/private key pairs. At time T_i, O produces warrant ω_i and corresponding certificate $cert_i$. By $\overline{M}_i$ we denote message space which consists of all the messages possibly delegated at time T_i. Given a message $m_i \in \overline{M}_i$, a warrant ω_i and a certificate $cert_i$, the proxy P generates a valid proxy signature $(\omega_i, cert_i, S(sk_P, 01||m_i))$, i.e. $V(pk_i, 00||\omega_i, cert_i) = 1$ and $V(pk', 01||M, S(sk_P, 01||m_i)) = 1$.

In general, for different subscript i, $\bigcap \overline{M}_i \neq \phi$. We consider the simplest case $i = 1, 2$. Since $(\omega_1, cert_1, S(sk_P, 01||m_1))$ and $(\omega_2, cert_2, S(sk_P, 01||m_2))$ are valid

proxy signatures, then $(\omega_1, cert_1, S(sk_P, 01||m_2))$ and $(\omega_2, cert_2, S(sk_P, 01||m_1))$ are also P's valid proxy signatures. This is because

$$V(pk_O, 00||\omega_1, cert_1) = 1 \text{ and } V(pk_P, 01||M, S(sk_P, 01||m_2)) = 1.$$

$$V(pk_O, 00||\omega_2, cert_2) = 1 \text{ and } V(pk_P, 01||M, S(sk_P, 01||m_1)) = 1.$$

Moreover, BPW's proxy signature schemes suffer a type of attacks mounted by a malicious original signer. Given a valid proxy signature $(\omega_i, cert_i, S(sk_P, 01||m_i))$, an original O can substitute ω_j for ω_i, where $j \neq i$, and generates a certificate $cert_j$. Thus, the malicious original signer O obtains $(\omega_j, cert_j, S(sk_P, 01||m_i))$ which satisfies the verification equation:

$$V(pk_O, 00||\omega_j, cert_j) = 1 \text{ and } V(pk_P, 01||M, S(sk_P, 01||m_i)) = 1.$$

In fact, the proxy signer P does not receive and accept the delegation $(\omega_j, cert_j)$ of the original signer O. If the malicious original changes the period T_i in the warrant ω_i into the period T_j in the warrant ω_j, the forgery perhaps impairs the proxy signer. BPW's delegation-by-certificate proxy signature scheme can not provide any protection of the proxy signer.

3 The proposed delegation-by-certificate proxy signature

3.1 Construction of the proposed scheme

Suppose that an original signer O delegates a proxy signer P to sign message M on its behalf. O and P have public\private key pairs (pk_O, sk_O) and (pk_P, sk_P) respectively. The algorithms of the corresponding delegation-by-certificate proxy signature scheme are $PS[DS]$= (G_1,K_1,S_1,V_1,(D, P),PS,PV,ID). The randomized parameter-generation algorithm G_1, the randomized key-generation algorithm K_1, the signing algorithm S_1 and the deterministic verification algorithm V_1 are defined as in Section 2.1. Other algorithms are defines as follows.

- The pair (D, P) of interactive randomized algorithms outputs $(\omega, cert)$, where the warrant ω includes the type of the information delegated, two involved parties' public keys pk_O, pk_P and the period of delegation, and the certificate is $cert = S(sk_O, 00||\omega)$.
- User P produces a proxy signature, which contains message M, the warrant ω, the certificate **cert** and a signature σ for $01||M||\omega||cert$ under sk_P.

$$PS(sk_P, w, cert, M) = (w, cert, M, S(sk_P, 01||M||\omega||cert)).$$

- Proxy signature verification is defined as follows.
  $PV(pk, M, (\omega, pk', cert, \sigma)) = V(pk, 00||\omega, cert) \wedge V(pk', 01||M||\omega||cert, \sigma)$.
- The identification algorithm is defined as $ID(\omega, pk', \sigma) = pk'$.

3.2 Notion of Security for delegation-by-certificate proxy signature schemes

Now we consider the security model for delegation-by-certificate proxy signature schemes. An adversary aims at personating the original signer or the proxy signer and forging a standard signature, or forging a proxy signature. In our model, an adversary is working against a single honest user, say user 1, and can select and register keys for all other users i, where $i = 2, 3, \cdots, n$. The adversary can play the role of the user $i \neq 1$, as an original signer or a proxy signer. We allow the adversary request user 1 to delegate himself and see the transcript of the self-delegation. The security model does not assume the existence of a secure channel between the original signer and the proxy signer. We model chosen-message attack and chosen-warrant attack capabilities by equipping the adversary with a signing oracle. As a matter of convenience, we classify the adversaries into outside adversaries and inside adversaries. The inside adversaries refer to the malicious original signers or the malicious proxy signers. Since the insider adversaries own the original signer's or proxy signer's private key, they have more advantages over the outside adversaries during the signature forgery. We only consider inside attacks. Our notion of security for delegation-by-certificate proxy signature schemes is formally defined as follows.

Definition [security for proposed proxy signature schemes]

We consider two experiments $\mathbf{Exp}^{os}_{PS,A1}$ and $\mathbf{Exp}^{ps}_{PS,A2}$ related to the proposed proxy scheme PS, adversaries $A1, A2$ and security parameter k.

We first consider the experiment $\mathbf{Exp}^{os}_{PS,A1}$. In the experiment, only one target signer of the adversary $A1$ is the original signer. System parameters are generated by running G on 1^k. Then the key-generation algorithm K is run and generates a public\private key pair (pk_O, sk_O) of the original signer. Adversary $A1$ is given input the public key pk_O of the target user. In the sequel, we make the convention that the target signer is user 1. $A1$ can make the following requests or quests in any order and any number of times.

- (i register pk_i) $A1$ can request to register user i by outputting a public\private key pair (pk_i, sk_i) of the user i. These pairs are stored. We keep a list $\mathbf{WCERT}_i$ that are initially empty. The list $\mathbf{WCERT}_i$ will consist of all

the warrants and certificates by which user 1 delegates the user i as the proxy signer.

- (1 designates i) $A1$ can request user 1 to designate $A1$ as a proxy signer i. $A1$ plays the role of the proxy signer i by running the algorithm P. User 1 runs the algorithm D and produces a warrant ω_i and send it to the adversary $A1$. $A1$ queries the signing oracle $O_S(sk_1, \cdot)$ about the warrant ω_i. The oracle outputs $cert_i$. Add $(\omega_i, cert_i)$ to the list $\mathbf{WCERT}_i$.
- (1 designates 1) $A1$ can request user 1 to designate itself. User 1 runs the algorithms D and P, produces a warrant ω_1 and send it to the adversary $A1$. $A1$ queries the signing oracle $O_S(sk_1, \cdot)$ about the warrant ω_i and obtains a certificate $cert_1 = O_S(sk_1, 01||\omega_1)$. Add $(\omega_1, cert_1)$ to the list $\mathbf{WCERT}_1$.
- (standard signature by 1) $A1$ can request user 1 to sign message M of his choice by querying the signing oracle $O_S(sk_1)$. Let $\sigma = O_S(sk_1, 11||M)$. Add the message\signature (M, σ) to the list **STMS**.
- (l-self delegation signature of user 1) $A1$ can make a query $(1, l, M)$ to the signing oracle $O_S(sk_1, \cdot)$. If $\mathrm{WCERT}_1[\mathrm{l}]$ in the list $\mathbf{WCERT}_1$ has been defined, the oracle returns $\sigma = O_S(sk_1, 01||M||\omega_1[l]||cert_1[l])$. Add $(\omega_1[l], cert_1[l], M, \sigma)$ to the list **SDS**. Otherwise, the oracle returns $\perp$.

Finally, $A1$ outputs a forgery (M, σ) or $(\omega, cert, M, \sigma)$. The output of the experiment is defined as follows.

1. If the adversary $A1$'s forgery is (M, σ), where $V(pk_O, 11||M, \sigma) = 1$ and $(M, \sigma) \notin \mathbf{STMS}$, then the experiment $\mathbf{Exp}^{os}_{PS,A1}$ returns 1.
2. If the forgery is $(\omega_1[j], cert_1[j], M, \sigma)$, where $V(pk_O, 00||\omega_1[j], cert_1[j]) = 1$, $V(pk_O, 01||\omega_1[j]||cert_1[j]||M, \sigma) = 1$, and $(\omega_1[j], cert_1[j], M, \sigma) \notin \mathbf{SDS}$, then the experiment $\mathbf{Exp}^{os}_{PS,A1}$ returns 1.
3. If the forgery is $(\omega_i[j], cert_i[j], M, \sigma)$, $i = 2, 3, \cdots, n$, where $V(pk_O, 00||\omega_i[j], cert_i[j]) = 1$, $V(pk_i, 01||\omega_i[j]||cert_i[j]||M, \sigma) = 1$, and $(\omega_i[j], cert_i[j], \cdot, \cdot) \notin \mathbf{WCERT}_i$, then the experiment $\mathbf{Exp}^{os}_{PS,A1}$ returns 1.
4. Otherwise, $\mathbf{Exp}^{os}_{PS,A1}$ returns 0.

Here we in essence define the goal of the adversary $A1$: I. a standard signature by user 1 for a message\signature pair that did not appear in the query list **STMS**. II. a self delegation proxy signature $(\omega_1[j], cert_1[j], M, \sigma)$ of the target user 1 such that $(\omega_1[j], cert_1[j], M, \sigma) \notin \mathbf{SDS}$. III. a delegation warrant $\omega_i[j]$ and certificate $cert_i[j]$ which was never generated by the original user 1 and $(\omega_i[j], cert_i[j], \cdot, \cdot) \notin \mathbf{WCERT}_i$. In other words, user 1 never designated user i by warrant $\omega_i[j]$ and certificate $cert_i[j]$.

We define advantage of the adversary $A1$ in the experiment $\mathbf{Exp}^{os}_{PS,A1}$ related to scheme PS, an adversary $A1$ and security parameter k as

$$\mathbf{Adv}^{os}_{PS,A1}(k) = \Pr[\mathbf{Exp}^{os}_{PS,A1} = 1].$$

Now, we consider the experiment $\mathbf{Exp}^{ps}_{PS,A2}$. In this experiment, only one target signer of an adversary $A2$ is the proxy signer. System parameters are generated by running G on 1^k. Then the key-generation algorithm K is run and generates a public\private key pair (pk_P, sk_P) of the proxy signer. Adversary $A2$ is given input the public key pk_P of the target signer user 1. $A2$ can make the following requests or quests in any order and any number of times.

- (i register pk_i) $A2$ can request to register user i by outputting a public\private key pair (pk_i, sk_i) of the user i. These pairs are stored. We keep a list $\mathbf{WCERT}_i$ that are initially empty. The list $\mathbf{WCERT}_i$ will consist of all the warrants and certificates by which user i delegate the user 1 as the proxy signer.
- (i designates 1) $A2$ can designate user 1 as a proxy signer. $A2$ chooses a warrant ω_i, produces a signature $cert_i$ on the warrant ω_i and sends them to the user 1. Because $A2$ has the private key sk_i of user i, $A2$ runs signing algorithm: $S(sk_i, 01||\omega_i) = cert_i$. User 1 checks validity of the delegation by running the verification algorithm $V(pk_i, \omega_i, cert_i)$. If the algorithm outputs 1, add $(\omega_i, cert_i)$ to the list $\mathbf{WCERT}_i$. Otherwise abandons.
- (1 designates 1) $A2$ can request user 1 to designate itself. User 1 runs the algorithm D and P and produces a warrant ω_1 and sends it to the adversary $A2$. $A2$ queries the signing oracle $O_S(sk_1, \cdot)$ about the warrant ω_i and obtains a certificate $cert_1 = O_S(sk_1, 01||\omega_1)$. Add $\omega_1, cert_1$ to the list $\mathbf{WCERT}_1$.
- (standard signature by 1) $A2$ can request user 1 to sign message M of his choice by querying the signing oracle $O_S(sk_1)$. Let $\sigma = O_S(sk_1, 11||M)$. Add the message\signature (M, σ) to the list **STMS**.
- (l-self delegation signature of user 1) $A2$ can make a query $(1, l, M)$ to the signing oracle $O_S(sk_1, \cdot)$. If $\mathrm{WCERT}_1[\mathrm{l}]$ in the list $\mathbf{WCERT}_1$ have been defined, the oracle returns $\sigma = O_S(sk_1, 01||M||\omega_1[l]||cert_1[l])$. Add $(\omega_1[l], cert_1[l], M, \sigma)$ to the list **SDS**. Otherwise, the oracle returns $\perp$.
- (proxy signature by user 1 on behalf of i with the l-th delegation-by-certificate) $A2$ can make a query (i, l, M) to the signing oracle $O_S(sk_1, \cdot)$, $i = 2, 3, \cdots, n$. If $\mathrm{WCERT}_i[\mathrm{l}]$ in the list $\mathbf{WCERT}_i$ has not been defined, adversary $A2$ can produce $(\omega_i[l], cert_i[l])$ and add them to the list. Hence

we always assume that $\text{WCERT}_i[\text{l}]$ in the list $\mathbf{WCERT}_i$. The signing oracle $O_S(sk_1, \cdot)$ returns σ, where $\sigma = O_S(sk_1, 01||M||\omega_i[l]||cert_i[l])$. Add $(\omega_i[l], cert_i[l], M, \sigma)$ to the list $\mathbf{PDS}_i$.

Finally, $A2$ outputs a forgery (M, σ) or $(\omega, cert, M, \sigma)$. The experiment produces the output according to the following.

1. If the adversary $A2$'s forgery is (M, σ), where $V(pk_P, 11||M, \sigma) = 1$ and $(M, \sigma) \notin \mathbf{STMS}$, then the experiment $\mathbf{Exp}^{ps}_{PS,A2}$ returns 1.
2. If the forgery is $(\omega_1[i], cert_1[i], M, \sigma)$, where $V(pk_P, 00||\omega_1[i], cert_1[i]) = 1$, $V(pk_P, 01||\omega_1[i]||cert_1[i]||M, \sigma) = 1$, and $(\omega_1[i], cert_1[i], M, \sigma) \notin \mathbf{SDS}$, then the experiment $\mathbf{Exp}^{ps}_{PS,A2}$ returns 1.
3. If the forgery is $(\omega_i[j], cert_i[j], M, \sigma)$, where $V(pk_O, 00||\omega_i[j], cert_i[j]) = 1$, $V(pk_i, 01||\omega_i[j]||cert_i[j]||M, \sigma) = 1$, and $(\omega_i[j], cert_i[j], M, \sigma) \notin \mathbf{PDS}_i$, then the experiment $\mathbf{Exp}^{ps}_{PS,A2}$ returns 1.
4. Otherwise, $\mathbf{Exp}^{ps}_{PS,A2}$ returns 0.

Here we define the goal of the adversary $A2$: I. a standard signature by user 1 for a message\signature pair that did not appear the query list **STMS**. II. a self delegation proxy signature $(\omega_1[j], cert_1[j], M, \sigma)$ by the target user 1 such that $(\omega_1[j], cert_1[j], M, \sigma) \notin \mathbf{SDS}$. III. the proxy signature $(\omega_i[j], cert_i[j], M, \sigma) \notin \mathbf{PDS}_i$, i.e, the query $01||M||\omega_i[j]||cert_i[j]$ to the signing oracle $O_S(sk_1, \cdot)$ was not made.

We define the advantage of adversary $A2$ in the experiment $\mathbf{Exp}^{ps}_{PS,A2}$ as

$$\mathbf{Adv}^{ps}_{PS,A2}(k) = \Pr[\mathbf{Exp}^{ps}_{PS,A2} = 1].$$

Finally we define the advantage of PS. For any t, q, the advantage of PS is defined as

$$\mathbf{Adv}^{ps,os}_{PS}(t, q, k) = \max_{\mathcal{A}}\{\mathbf{Adv}^{ps}_{PS,A}(k), \mathbf{Adv}^{os}_{PS,A}(k)\}.$$

where the maximum is over all A having time-complexity t and making q oracle queries. The time t of the adversary A is the total time of two experiments which includes the time taken in parameter and key generation, and response to the requests and the queries.

We say that a delegation-by-certificate proxy signature scheme PS is secure if $\mathbf{Adv}^{ps,os}_{PS}(t, q, k)$ is negligible for all the adversaries A of time polynomial t in the security parameter k.

Theorem 1 Assume the underlying signature scheme $DS = (G, K, S, V)$ is secure against adaptive chosen-message attacks, then our delegation-by-certificate proxy signature scheme $PS = (G,\ K,\ S,\ V,\ (D, P),\ PS,\ PV,\ ID)$ constructed as above is secure against adaptive chosen-message attacks and chosen-certificate attacks. ∎

We give proof sketch of the theorem in Appendix B.

4 A Concrete Delegation-by-certificate Proxy Signature Scheme

4.1 Construction of our proposed scheme

We first present a delegation-by-certificate proxy signature scheme based on short signature scheme from the weil pairing (See Appendix A.2). The proxy signature scheme PS comprises of the following phases.

- *System Setup*

  The parameter-generation algorithm G generates (t, ε) co-Gap Diffie-Hellman pair (G_1, G_2) with $|G_1| = |G_2| = p$ and a generator pair (g_1, g_2), an efficiently computable isomorphism $\psi : G_2 \longrightarrow G_1$, a bilinear map $e : G_1 \times G_2 \longrightarrow G_T$ (See Appendix A.1) where G_T is a order-p group, and a public full-domain hash function $H : \{0,1\}^* \longrightarrow G_1$. On input these parameters, the key-generation algorithm K outputs an original signer O's public\private key pair (x_O, y_O) and a proxy signer P's public\private key pair (x_P, y_P), $y_P = g_2^{x_P}, y_O = g_2^{x_O}$.
- *Delegation-by-Certificate Phase*

  1. (Certificate Generation Phase) The original signer O generates a warrant $\omega \in \{0,1\}^*$, which records the delegation limits of authority, valid period of delegation, and the identities of the original signer and proxy signer. O computes $cert = H(00||\omega)^{x_O}$ and sends $(\omega, cert)$ to the proxy signer P.
  2. (Delegation Verification Phase) After the proxy signer receives the delegation warrant and certification $(\omega, cert)$, P checks if $e(g_2, cert) = e(y_O, H(00||\omega))$. If so, P begins to execute the proxy signature generation algorithm. Otherwise, P refuses this delegation.

- *Proxy Signature Generation Phase*

  P computes $\sigma = H(01||M||\omega||cert)^{x_P}$. Then, the proxy signature on message M is $(M, \omega, cert, \sigma)$.

- *Proxy Signature Verification Phase*

  To verify a delegation-by-certificate signature $(M, \omega, cert, \sigma)$, the proxy verification algorithm PV is run. If $e(g_2, cert) = e(y_O, H(00||\omega))$ and $e(g_2, \sigma) = e(y_P, H(01||M||\omega||cert))$, the verification algorithm outputs 1. Otherwise, it outputs 0. The first verification equation checks the validity of the certificate and the other equation checks the validity of the remaining part of proxy signature.

4.2 Security Analysis of our Proposed Scheme

Now we consider the security of our proposed scheme in the security model for delegation-by-certificate. An adversary aims at personating the original signer or the proxy signer and forging a standard signature, or forging a proxy signature. We model chosen-message attack and chosen-warrant attack capabilities by equipping the adversary with a signing oracle.

Let us first consider an adversary $A1$ with knowledge of the original signer's private key x_O (here x_i, $i = 2, 3, \cdots, n$). Let qS, (resp. qH) be the number of signature queries (resp. hash queries) the adversary $A1$ made. The succedent theorem shows that our scheme can resist the chosen-message and chosen-warrant attacks in random oracle model. During the proof of the theorem, hash function is treated as a random function. For clearness, we divide hash queries into three types: H_1, H_2 and H_3. Supposed that $A1$ makes at most qH_1^i H_1-queries about $(i, 00, \omega)$ hash queries, at most qH_2^i H_2-queries about $(i, 01, \omega, d, cert, M)$ hash queries, and at most qH_3 H_3-queries about $(11, M)$ hash queries. The total of hash queries is still at most $qH = \sum_{i=1}^{n}(qH_1 + qH_2) + qH_3$, where user 1 is the target user of the adversary $A1$ and A registers at most $n - 1$ users. The signature queries contain at most qS_1 queries of the standard signature and at most qS_2^i queries of the proxy signature with the original signer user i, where $qS_1 + \sum_{i=1}^{n} qS_2^i = qS$.

Theorem 2. Let (G_1, G_2) be a (t, ε)-co-GDH group pair of order p as defined above. Then the proposed delegation-by-certificate proxy signature scheme is $(t', qS, qH, \varepsilon')$-secure against existential forgery under an adaptive chosen-message and chosen-warrant attack (in the random oracle model) for all t' and ε' for the adversary $A1$, where

$$\varepsilon \geq (1 - 1/(qS + q + 1))^{qS+q+1} \cdot 1/(qS + q + 1) \cdot \varepsilon'. \quad (1)$$

$$t \leq t' + \sum_{i=1}^{n}(qH_1^i + qH_2^i + 2qS_2^i + q_i) + q_1 + qH_3 + 2qS_1. \quad (2)$$

We provide the proof of Theorem 2 in Appendix C. ∎

Now, let us consider an adversary $A2$ with knowledge of the proxy signer's private key x_P (here x_i, $i = 2, 3, \cdots, n$). Let qS, (resp. qH) be the number of signature queries (resp. hash queries) the adversary $A2$ made The following theorem shows that our scheme can resist the chosen-message and chosen-warrant attacks of the adversary $A2$ in random oracle model. For clearness, we still divide hash queries into three types: H_1, H_2 and H_3. When user O (here user 1) delegates user i, $A2$ makes at most qH_1^i hash queries. During the generation of the proxy signature, $A2$ makes at most qH_2^i hash queries. During the generation of the standard signature, $A2$ makes at most qH_3 hash queries. The total of these hash queries is still at most qH, i.e. $\sum_{i=1}^{n}(qH_1 + qH_2) + qH_3 = qH$. User 1 is the target user of the adversary $A2$ and $A2$ registers at most $n-1$ users. The signature queries contain at most qS_1 queries of the standard signature and at most qS_2^i queries of the proxy signature with the proxy signer user i, where $qS_1 + \sum_{i=1}^{n} qS_2^i = qS$.

Theorem 3. Let (G_1, G_2) be a (t, ε)-co-GDH group pair of order p. Then the proposed delegation-by-certificate proxy signature scheme is $(t', qS, qH, \varepsilon')$-secure against existential forgery under an adaptive chosen-message and chosen-warrant attack in the random oracle model for all t' and ε' for the adversary $A2$.

$$\varepsilon \geq (1 - 1/(qS + q + 1))^{qS_1 + qS_2^1 + q + 1} \cdot 1/(qS + q + 1) \cdot \varepsilon'. \tag{3}$$

$$t \leq t' + \sum_{i=1}^{n}(qH_1^i + qH_2^i + qS_2^i + 2q_i) + qH_3 + qS_1. \tag{4}$$

We provide the proof of Theorem 3 in Appendix C. ∎

5 Conclusion

In this paper, we have presented an attack on that the delegation-by-certificate proxy signature scheme [2]. We have proposed delegation-by-certificate proxy signature schemes and constructed a concrete one based on Co-Gap group from bilinear map. Our proxy signature scheme is existentially unforgeable against adaptive-chosen message attacks and adaptive-chosen warrant attacks in random oracle model. A method of reducing the scheme's security to hardness of the computational co-Diffie-Hellman problem is proposed. In addition, it is possible to construct secure hierarchy proxy signatures by the proposed delegation-by-certificate proxy signature schemes.

References

[1] D.Boneh, B. Lynn and H. Shacham. Short Signatures from the Weil Pairing. In *Proceedings of Asiacrypt 2001*, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2248, pp. 514-532, 2001.

[2] A.Boldyreva, A. Palacio, B. Warinschi. Secure Proxy Signature Schemes for Delegation of Signing Rights. At:http://eprint.iacr.org/2003/096.

[3] S. J. Hwang and C. C. Chen. A new proxy multi-signature scheme. In *International workshop on cryptology and network security*, Tamkang University, Taipei, Taiwan, Sep. 26-28, 2001.

[4] S. J. Hwang and C. H. Shi. A simple multi-proxy signature scheme. In *Proceedings of the Tenth National Conference on Information Security*, pp. 134-138, 2000.

[5] P. Kotzanikolaous, M.Burmester, and V. Chrisskopoulos. Secure transactions with mobile agent in hostile environments. In *Proc. ACISP 2000*, Lecture Notes in Computer Science 1841, Springer-Verlag, pp. 289-297, 2000.

[6] H. Kim, J. Baek, B. Lee, and K. Kim. Secrets for mobile agent using one-time proxy signature. *Cryptography and Information Security 2001*, Vol 2/2, pp. 845-850, 2001.

[7] S. J. Kim, S. J. Park, D. H. Won. Proxy Signatures, revisited. *ICICS'97*, Lecture Notes in Computer Science 1334, pp. 223-232, Springer-Verlag.

[8] W.D. Lin and J.K. Jan. A security personal learning tools using a proxy blind signature scheme. In *Pro. of International Conference on Chinese Language Computing*, Illinois, USA, July 2000, pp. 273-277, 2000.

[9] B. Lee, H. Kim, and K. Kim. Strong proxy signgture and its applications. In *Proceedings of SCIS*, 2001, pp. 603-608.

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

[11] M. Mambo, K. Usuda and E. Okamoto. Proxy signatures for delegating signing operation. In *Proc. 3rd ACM Conference on Computer and Communications Security*, ACM Press, 1996, pp. 48-57.

[12] H.-U. Park and L.-Y. Lee. A digital nominative proxy signature scheme for mobile communications. *ICICS 2001*, Lecture Notes in Computer Science 2229, Springer-Verlag, pp. 451-455, 2001.

[13] H. M. Sun. An efficient nonrepudiable threshold proxy signatures with known signers. *Computer Communications* 22(8),1999, pp. 717-722.

[14] H.-M Sun and B.-T Hsieh. On the security of some proxy blind signature schemes. In *AISW2004*, Dunedin, New Zealand. Conferences in Research and Practice in Information Technology, Vol. 32.

[15] H. M. Sun, and B.-T. Hsieh. On the security of some proxy signature scheme. At:http://eprint.iacr.org/2003/068.

[16] H. Sun, N. -Y. Lee, and T. Hwang. Threshold proxy signatures. In *IEEE Proceedings-Computes and Digital Technique*, Vol. 146, IEEE Press, 1999, pp. 259-263.

[17] Z.-W.Tan, Z.-J. Liu. Proxy blind signature scheme based on DLP. *Journal of Software*, 2003/14, pp. 1931-1935, 2003.

[18] Guilin Wang, Feng Bao, Jianying Zhou, and Robert H. Deng. Security Analysis of Some Proxy Signatures. In I*nformation Security and Cryptology - ICISC 2003*, Springer-Verlag, 2004.

[19] Huaxion Wang and Josef Pieprzyk. Efficient One-time proxy signatures. In *ASIACRYPT 2003*, pp. 507-522, 2004, Springer-Verlag.

[20] K. Zhang. Threshold proxy signature schemes. In *1997 Information Security Workshop*, Japan, pp. 191-197, 1997.

A Preliminaries

Here we review a few concepts related to co-GDH Diffie-Hellem groups and the short signature based on co-GDH Diffie-Hellem groups from bilinear maps. We use the following notations:

1. G_1 and G_2 are two multiplicative cyclic groups of prime order p;
2. G_T is an additional group of prime order p;
3. g_1 is a generator of G_1 and g_2 is a generator of G_2;
4. ψ is an isomorphism from G_2 to G_1 such that $\psi(g_2) = g_1$;

A.1 co-GDH Diffie-Hellem groups from bilinear maps

Definition 1. Let G_1, G_2 and G_T be groups as mentioned above. A map $e : G_1 \times G_2 \longrightarrow G_T$ is a **bilinear map** if the map e satisfies the following properties:

1. Bilinear: $e(au, bv) = e(u, v)^{ab}$ for all $u \in G_1, v \in G_2$ and $a, b \in Z$.
2. Non-degenerate: $e(g_1, g_2) \neq 1$.
3. Efficient Computable: There is a polynomial algorithm to compute $e(u, v)$ for all $u \in G_1, v \in G_2$.

Definition 2. Given two order-p multiplicative groups G_1, G_2, two elements $g_2, g_2^a \in G_2$ and $h \in G_1$, compute $h^a \in G_1$. The problem is called **Computational co-Diffie-Hellman (co-CDH)** Problem on (G_1, G_2).

Definition 3. Given two order-p multiplicative groups G_1, G_2, two elements $g_2, g_2^a \in G_2$ and two elements $h, h^b \in G_1$, determine whether $a = b \mod p$ holds.

The problem is called **Decisional co-Diffie-Hellman (co-DDH)** Problem on (G_1, G_2).

Before we define the co-GDH gap group pair, we recall the definition of the advantage of an algorithm A in solving the co-CDH problem on (G_1, G_2):

$$\textbf{Adv co-CDH}_A \stackrel{def}{=} Pr[A(g_2, g_2^a, h) = h^a : a \stackrel{R}{\leftarrow} Z_p, h \stackrel{R}{\leftarrow} G_1], \tag{5}$$

where the probability is taken over the random choice of a from Z_p and h from G_1, over the coin tosses of the algorithm A. If an algorithm A solve the co-DDH problem in time at most t, and **Adv co-CDH**$_A$ at least ε, we say that A (t, ε)-breaks Computational co-Diffie-Hellman on (G_1, G_2).

Definition 4. Two order-p groups (G_1, G_2) are a (t, ε)-**co-GDH groups** if the following properties are satisfied:

1. The group action on G_1 and G_2 and the isomorphism ψ from G_2 to G_1 can be computed in one time unit.
2. The Decisional co-Diffie-Hellman on (G_1, G_2) can be solved in one time unit.
3. No algorithm (t, ε)-breaks Computational co-Diffie-Hellman on (G_1, G_2).

Definition 5. Suppose the order-p groups (G_1, G_2) and G_T, e be given as in Definition 1. If no algorithm (t, ε)-breaks Computational co-Diffie-Hellman on (G_1, G_2), (G_1, G_2) are called a (t, ε)-**bilinear group pair**.

For a (t, ε)-bilinear group pair (G_1, G_2), decisional co-Diffie-Hellman (co-DDH) problem on can be solved by using the efficiently-computable bilinear map e as follows: For a tuple (g_2, g_2^a, h, h^b), $h \in G_1$, we have

$$a = b \mod p \Leftrightarrow e(h, g_2^a) = e(h^b, g_2).$$

A.2 Short signature scheme from the weil pairing

We recall the short signature from co-GDH groups from bilinear maps. Let (G_1, G_2) be (t, ε) co-GDH group pair from bilinear maps. The signature scheme is comprised of three algorithms, $KeyGen, Sign$ and $Verify$. Let $H : \{0,1\}^* \longrightarrow G_1$ be a full-domain hash function.

$KeyGen$ take a random $x \in_R Z_p$ as input and outputs $y = g_2^x$. Then (y, x) is a public/secret pair.

$Sign$ takes a message $M \in \{0,1\}^*$ and a secret key $x \in Z_p$ as input and outputs a signature $\sigma = h^x$, where $h = H(M) \in G_1$.

$Verify$ is given the public $y \in G_2$, a message $M \in \{0,1\}^*$, and a signature $\sigma \in G_1$, and outputs **1** if (g_2, y, h, σ) is a valid co-Diffie-Hellman tuple, otherwise outputs **0**.

The signature scheme has been proved secure against existential forgery under a chosen-message attack in the random oracle model (for details see [1]).

B The proof of Theorem 1

Theorem 1 Assume the underlying signature scheme $DS = (G, K, S, V)$ is secure against adaptive chosen-message attacks, then our delegation-by-certificate proxy signature scheme PS = (G, K,S,V,(D, P),PS,PV,ID) constructed as above is secure against adaptive chosen-message attacks and chosen-certificate attacks.

Proof Sketch : We prove it by reduction. If an adversary can forge a valid standard signature or a self delegation proxy signature by the target signer user 1, it will contradict that the underlying signature scheme is secure. If the proxy signer is an adversary $A1$ and $A1$ generates a valid proxy signature $(\omega, cert, M, \sigma)$, then $(\omega, cert, \cdot, \cdot)$ is not in $\mathbf{WCERT}_i$, i.e, $A1$ forges a valid warrant and certificate pair. If the original signer is an adversary $A2$ and $A2$ generates a valid proxy signature $(\omega, cert, M, \sigma)$, then $(\omega, cert, M, \sigma)$ is not in $\mathbf{PDS}_i$. i.e, $A2$ forges a valid underlying signature on $01||M||\omega||cert$. These contradict that the underlying signature scheme is secure. ∎

C The proof of Theorem 2 and 3

Theorem 2. Let (G_1, G_2) be a (t, ε)-co-GDH group pair of order p as defined above. Then the proposed delegation-by-certificate proxy signature scheme is $(t', qS, qH, \varepsilon')$-secure against existential forgery under an adaptive chosen-message and chosen-warrant attack (in the random oracle model) for all t' and ε' for the adversary $A1$, where

$$\varepsilon \geq (1 - 1/(qS + q + 1))^{qS+q+1} \cdot 1/(qS + q + 1) \cdot \varepsilon'. \tag{6}$$

$$t \leq t' + \sum_{i=1}^{n}(qH_1^i + qH_2^i + 2qS_2^i + q_i) + q_1 + qH_3 + 2qS_1 \tag{7}$$

Proof: Suppose $A1$ is a forgery algorithm that $(t', qS, qH, \varepsilon')$-breaks the signature scheme. We will construct an algorithm B to solve co-CDH in (G_1, G_2) with probability at least ε in time at most t by using the adversary $A1$.

Algorithm B is given a generator g_2 of G_2, $u \in G_2$ and $h \in G_1$, where $u = g_2^a$. B attempts to output $h^a \in G_1$. Note that a is unknown to B. B must simulate the challenger and respond to $A1$'s queries and requests.

- **Setup** B gives $A1$ the generator g_2 of G_2 and $u \cdot g_2^r, r \in_R Z_p$ as the public key of the target user 1. Here user 1 is the proxy signer.

- **Simulation** B responds to the queries of the adversary $A1$.

  1. (i registers pk_i) When $A1$ outputs a public\private key pair (pk_i, sk_i) of the user i, B stores these key pairs, where $i = 2, 3, \cdots, n$, n is the number of the registered users. B keeps a list $\mathbf{WCERT}_i$ that are initially empty.
  2. (H_1-queries) When $A1$ queries $(i, 00, \omega)$ to H_1-oracle, where $i = 1, 2, \cdots, n$, we keep B's response to these queries in the list LH_1^i. If $(i, 00, \omega, d, \cdot, \cdot) \in LH_1^i$, B returns d . Otherwise, B randomly chooses a bit c_j so that $\Pr[c_j = 0]=1/(qS + q + 1)$, where $q = \sum_{i=1}^{n} q_i$, $b_j \in Z_p$. We assumes $A1$ makes at most qH_1^i H_1-queries about $(i, 00, \omega)$. The subscript j shows that this is the j-th queries to H_1-oracle. If $c_j = 0$, $d_j = h \cdot \psi(g_2)^{b_j} \in G_1$, otherwise $c_j = 1$, $d_j = \psi(g_2)^{b_j} \in G_1$. B returns d_j and adds $(i, oo, \omega, d_j, c_j, b_j)$ to the list LH_1^i.
  3. (H_2-queries) When $A1$ queries $(i, 01, \omega, d, cert, M)$ to H_2-oracle, where $i = 1, 2, \cdots, n$, we keep B's response to the queries in the list LH_2^i. We assume $A1$ makes at most qH_2^i H_2-queries about $(i, 01, \omega, d, cert, M)$ and this is the j-th queries to H_1-oracle, where $j = 1, 2, \cdots, qH_2^i$. If $(i, 01, \omega, d, cert, M, d_M, \cdot, \cdot) \in LH_2^i$, B returns d_M. Otherwise, B randomly chooses a bit c_j so that $\Pr[c_j = 0]=1/(qS + q + 1)$, $b_j \in Z_p$, if $c_j = 0$, $d_M = h \cdot \psi(g_2)^{b_j} \in G_1$; if $c_j = 1$, $d_M = \psi(g_2)^{b_j} \in G_1$. B returns d_M and add $(i, 01, \omega, d, cert, M, d_M, c_j, b_j)$ to the list LH_2^i.
  4. (H_3-queries) When $A1$ queries $(11, M)$ to H_3-oracle, we keep B's response to these queries in the list LH_3. We assumes $A1$ makes at most qH_3 H_3-queries about $(11, M)$ and this is the j-th queries to H_3-oracle. If $(11, M, d, \cdot, \cdot) \in LH_3$, B returns d. Otherwise, B randomly chooses a bit c_j so that $\Pr[c_j = 0]=1/(qS + q + 1)$, $b_j \in Z_p$, where $j = 1, 2, \cdots, qH_3$, if $c_j = 0$, $d_j = h \cdot \psi(g_2)^{b_j} \in G_1$, otherwise $c_j = 1$, $d_j = \psi(g_2)^{b_j} \in G_1$. Return d_j. Add $(11, M, d_j, c_j, b_j)$ to the list LH_3.
  5. (i designates 1) $A1$ can play the role of the original signer i ($i = 2, 3, \cdots, n$) to designate user 1 as a proxy signer. We assumes that $A1$ makes at most q_i such designations from user i to user 1. $A1$ outputs ω, then queries $(i, 00, \omega)$ to H_1-oracle. B makes the same response as in H_1-oracle. After that, $A1$ outputs $(i, \omega, cert)$ and sends the tuple to B.

B checks the validity of the tuple $(i, \omega, cert)$. Finally, add $(\omega, d, cert)$ to $\mathbf{WCERT}_i$.

6. (1 designates 1) $A1$ can request user 1 to designate itself. Assume that $A1$ requests at most q_1 self designations of user 1 and this is the j-th request. User 1 produces a warrant ω_1. The adversary $A1$ makes H_1-queries about $(1, 00, \omega)$. B responses as in H_1-queries and returns d. If $c_j = 0$, B returns FAILURE, otherwise $c_j = 1$, B lets $cert_j = \psi(u)^{b_j} \cdot \psi(g_2)^{rb_j}, r \in Z_p$. B returns $cert_j$. Add $(\omega_j, d_j, cert_j, \cdot, \cdot)$ to the list $\mathbf{WCERT}_1$.
7. (standard signature by 1) When $A1$ requests user 1 to sign message M of its choice by querying the signing oracle $O_S(sk_1)$, B first answers H_3-query about $(11, M)$. If there exists s, such that $(M, d, s) \in \mathbf{LS}$, B returns s; otherwise for $(11, M, d, c, b)$ in the list LH_3, if $c = 0$, B returns FAILURE; otherwise $c = 1$, B lets $s = \psi(u)^b \cdot \psi(g_2)^{rb}, r \in Z_p$ and returns s. Add (M, d, s) to the list **STMS**. Assume $A1$ makes at most qS_1 queries of standard signature by user 1.
8. (proxy signature by user 1 on behalf of i with the l-th delegation-by-certificate) Assume that $A1$ makes a query (i, l, M) to the signing oracle $O_S(sk_1, \cdot)$, where $i = 1, 2, \cdots, n$. If $\mathrm{WCERT}_i[\mathrm{l}]$ in the list $\mathbf{WCERT}_i$ has not been defined, B returns INVALID. If $\mathrm{WCERT}_i[\mathrm{l}] \notin \mathbf{WCERT}_i$, but $(i, \omega_l, d_l, cert_l, M, d_M, s) \in \mathbf{PDS}_i$ (when $i \neq 1$) or **SDS** (when $i = 1$), B returns s. If $\mathrm{WCERT}_i[\mathrm{l}] \notin \mathbf{WCERT}_i$ and $(i, \omega_l, d_l, cert_l, M, d_M, s) \notin \mathbf{PDS}_i$ (when $i \neq 1$) or **SDS** (when $i = 1$), B first answers H_2-query about $(i,\ 01,\ \omega_l, d_l, cert_l, M)$ and returns d_M, then B responses to the signing oracle $O_S(sk_1, \cdot)$ about $01||M||\omega_l||cert_l$: if $c_j = 0$, B returns FAILURE, otherwise $c_j = 1$, B lets $s_j = \psi(u)^{b_j} \cdot \psi(g_2)^{r_j b_j}, r_j \in Z_p$, and returns s_j. Add $(\omega_l, d_l,\ cert_l,\ M,\ d_M,\ s_j)$ to the list $\mathbf{PDS}_i$ when $i \neq 1$ or **SDS** when $i = 1$.

- **Forgery** $A1$ produces its forgery (M, s) or $(\omega, cert, M, s)$.
- **Output** Algorithm B generates its output according to the type of the forgery.

[F1] If the forgery is the standard signature (M, s) of the target user, where $V(pk_1, 11||M, s) = 1$ and $(M, \cdot, s) \notin$**STMS**, B finds $(11, M, \cdot, \cdot, \cdot)$ in LH_3. If $c = 1$, B returns FAILURE; otherwise $c = 0$, B computes $h^a = s/(h^r \cdot \psi(u)^b \cdot \psi(g_2)^{rb})$. The result follows from the fact $d = h \cdot \psi(g_2)^b$, $s = [h \cdot \psi(g_2)^b]^{a+r}$.

[F2] The forgery is the self-proxy signature $(\omega_1, cert_1, M, s)$ of the target user, where $V(pk_1, 00||\omega_1, cert_1) = 1$, $V(pk_1, 01||M||\omega_1||cert_1, s) = 1$ and $(\omega_1, cert_1, M, s) \notin \mathbf{SDS}$. If $(\omega_1, \cdot, cert_1) \in \mathbf{WCERT}_1$, B finds $(1, 01, \omega_1, d, cert_1, M, d_M, \cdot, \cdot)$ in the list LH_2^1. Then B outputs h^a as in the event **F1**. If $(\omega_1, \cdot, cert_1) \notin \mathbf{WCERT}_1$, then according to $V(pk_1, 00||\omega_1, cert_1) = 1$, B can find $(1, 00, \omega_1, d, \cdot, \cdot)$ in the list LH_1. B outputs h^a as above.

[F3] The forgery is the proxy signature $(\omega_i, cert_i, M, s)$ of the target user with the original signer i, $i = 2, 3, \cdots, n$, where $V(pk_i, 00||\omega_i, cert_i) = 1$, $V(pk_1, 01||M||\omega_i||cert_i, s) = 1$ and $(\omega_i, cert_i, M, s) \notin \mathbf{PDS}_i$. Whether $(\omega_i, \cdot, cert_i) \in \mathbf{WCERT}_i$ or $(\omega_i, \cdot, cert_i) \notin \mathbf{WCERT}_i$, we always assume that $(i, 01, \omega_i, cert_i, M, \cdot, \cdot, \cdot) \in LH_2^i$. In fact, when the tuple is not in the list LH_2^i, B responses to the H_2-query and add $(i, 01, \omega_i, cert_i, M, d_M, \cdot, \cdot)$ to LH_2^i. Then B can output h^a as above.

We complete the description of algorithm B. Next, we analyze the success probability ε of B in solving the given instance of the co-CDH problem in (G_1, G_2). For the convenience, we make further assumptions that $A1$'s forgery **F1**, **F2** and **F3** are independent from each other and happen with probability at most ε_1', ε_2' and ε_3', respectively, where $\varepsilon' = \varepsilon_1' + \varepsilon_2' + \varepsilon_3'$. B succeeds in **F1** if all the following events happen.

E11: B does not abort as a result of any of $A1$'s signature queries, hash queries and designation requests. $A1$'s signature queries includes standard signature queries, self-proxy signature queries and the proxy signature queries with the original signer user i for all $i = 2, 3, \cdots, n$. $A1$'s designation requests refer to all the self delegation of user 1 and all the delegation by user i.

E12: $A1$ produces a valid standard signature forgery (M, s).

E13: Event **E12** happens and $c = 0$ for $(11, M, \cdot, \cdot, \cdot)$ in the list LH_3.

By $B1$ we denote the event that B succeeds in **F1**.

$$\begin{aligned} \Pr(\mathrm{B1}) &= \Pr[\mathrm{E11} \wedge \mathrm{E12} \wedge \mathrm{E13}] \\ &= \Pr[\mathrm{E11}] \cdot \Pr[\mathrm{E12}|\mathrm{E11}] \cdot \Pr[\mathrm{E13}|\mathrm{E11} \wedge \mathrm{E12}]. \end{aligned} \tag{8}$$

We will give a lower bound for these terms in the following claims.

Claims 1.1: $\Pr[\mathrm{E11}] \geq (1 - 1/(qS + q + 1))^{qS+q+1}$.

Proof. When $A1$ requests user 1 to sign i-th message M of its choice, B finds $(11, M, d, c, b)$ in the list LH_3 or adds the tuple to LH_3. Prior to the query, $A1$'s view is indepent of the bit c. Moreover, d is uniformly distributed in G_1 for any

bit c. Therefore the query $(11, M)$ causes B to abort with the probability at most $1/(qS + q + 1)$. For qS_1 queries of user 1's standard signature, the probability that B does not abort is at least $(1 - 1/(qS + q + 1))^{qS_1}$. If we make a similar analysis to the proxy signature queries (i, l, M) with the original signer user i $(i = 1, 2, \cdots, n)$. And the probability that B does not abort during making responses to the queries is at least $(1 - 1/(qS + q + 1))^{\sum_{i=1}^{n} qS_2^i}$. Similarly, the probability that B does not abort during answering requests of all the delegations is at least $(1 - 1/(qS + q + 1))^q$.

Claims 1.2: Pr[E12|E11]$\geq \varepsilon_1'$.

Proof: If B does not abort as a result of $A1$'s signature queries and delegation requests, then the view of $A1$ in the simulation is identical to the view of $A1$ in the actual attack. The claim follows from it.

Claims 1.3: Pr[E13|E11$\wedge$E12]$\geq 1/(qS + q + 1)$.

Proof: Conditioned on the events **E11** and **E12**, B will abort only if $A1$ produces a forgery (M, s) for which $(11, M, d, c, \cdot)$ in LH_3 has $c = 1$. Since $A1$ could not have made a standard signature query for M, c is independent of $A1$'s view and therefore we obtain Pr[E13|(E11$\wedge$ E12)]$\geq 1/(qS + q + 1)$.

From equation (8) and the above-mentioned claims, we have

$$\Pr(\text{B1}) \geq (1 - 1/(qS + q + 1))^{qS+q+1} \cdot 1/(qS + q + 1) \cdot \varepsilon_1'. \tag{9}$$

By $B2$ we denote the event that B succeeds in **F2**. We define the events needed for $B2$.

E21: It is the same as **E11**.

E22: $A1$ produces a valid self-proxy signature $(\omega, cert, M, s)$ of the target user.

E23: Event **E22** happens and the bit $c = 0$ for $(i, 01, \omega, d, cert, M, d_M, c_j, b_j)$ in the list LH_2^1.

Claims 2.1: Pr[E21]$\geq (1 - 1/(qS + q + 1))^{qS+q+1}$.

Claims 2.2: Pr[E22|E21]$\geq \varepsilon_2'$.

Claims 2.3: Pr[E23|(E21$\wedge$E22)]$\geq 1/(qS + q + 1)$.

Proof: Conditioned on the events **E21** and **E22**, B will abort only if $A1$ produces a forgery (M, s) for which $(i, 01, \omega, d, cert, M, d_M, c_j, b_j)$ in the list LH_2^1 has $c = 1$. Since $A1$ could not have made a self proxy signature query for $(1, l, M)$, c is independent of $A1$'s view and therefore we obtain Pr[E23|(E21$\wedge$ E22)]$\geq 1/(qS + q + 1)$. Thus, we have

$$\Pr(\text{B2}) \geq (1 - 1/(qS + q + 1))^{qS+q+1} \cdot 1/(qS + q + 1) \cdot \varepsilon_2'. \tag{10}$$

By $B3$ we denote the event that B succeeds in **F3**. We define the events

needed for $B3$.

E31: It is the same as **E11**.

E32: Adversary $A1$ produces a valid proxy signature $(\omega, cert, M, s)$ in stead of the target user 1.

E33: Event **E32** happens and $c = 0$ for $(i, 01, \omega, d, cert, M, d_M, c_j, b_j)$ in the list LH_2^i, where $i = 2, 3, \cdots, n$.

Claims 3.1: $\Pr[\text{E31}] \geq (1 - 1/(qS + q + 1))^{qS+q+1}$.

Claims 3.2: $\Pr[\text{E32}|\text{E31}] \geq \varepsilon_2'$.

Claims 3.3: $\Pr[\text{E33}|(\text{E31} \wedge \text{E32})] \geq 1/(qS + q + 1)$.

Proof: Conditioned on the events **E21** and **E22**, B will abort only if $A1$ produces a forgery (M, s) for which $(i, 01, \omega, d, cert, M, d_M, c_j, b_j)$ in the list LH_2^i has $c = 1$. Since $A1$ could not have made a self proxy signature query for $(1, l, M)$, c is independent of $A1$'s view and therefore we obtain $\Pr[\text{E23}|(\text{E21} \wedge \text{E22})] \geq 1/(qS + q + 1)$. Thus, we have

$$\Pr(\text{B1}) \geq \varepsilon_2' \cdot (1 - 1/(qS + q + 1))^{qS+q+1} \cdot 1/(qS + q + 1). \tag{11}$$

Therefore, the success probability ε of B in solving the given instance of the co-CDH problem in (G_1, G_2) is

$$\begin{aligned} \Pr(\text{B}) &= \Pr(\text{B1}) + \Pr(\text{B2}) + \Pr(\text{B3}) && (12)\\ &\geq (\varepsilon_1' + \varepsilon_2' + \varepsilon_3') \cdot (1 - 1/(qS + q + 1))^{qS+q+1} \cdot 1/(qS + q + 1) && (13)\\ &= (1 - 1/(qS + q + 1))^{qS+q+1} \cdot 1/(qS + q + 1) \cdot \varepsilon'. && (14) \end{aligned}$$

B's running time is $A1$'s running time plus the time taken to responds to all the queries and requests from $A1$. B responds to $\sum_{i=1}^{n} qH_1^i$ H_1-queries, $\sum_{i=1}^{n} qH_2^i$ H_2-queries and qH_3 H_3-queries. B's responses to at most q_1 self delegation queries includes at most q_1 H_1-answers and at most q_1 certification generations. For $\sum_{i=2}^{n} q_i$ delegation queries with the original (user i), B needs only at most $\sum_{i=2}^{n} q_i$ H_1-responses. B's responses to qS_1 standard signature queries include at most qS_1 H_3-answers and at most qS_1 signature generations. B's responses to $\sum_{i=1}^{n} qS_2^i$ proxy signature queries with the original user i includes at most $\sum_{i=1}^{n} qS_2^i$ H_2-answers and at most $\sum_{i=1}^{n} qS_2^i$ signature generations. Each response to Hash-oracle and each signature generation require an exponentiation in G_1, respectively. Hence, the total running time t of B satisfies

$$t \leq t' + \sum_{i=1}^{n} (qH_1^i + qH_2^i + 2qS_2^i + q_i) + q_1 + qH_3 + 2qS_1. \tag{15}$$

Thus, we have completed the proof of the theorem. ∎

Theorem 3. Let (G_1, G_2) be a (t, ε)-co-GDH group pair of order p. Then the proposed delegation-by-certificate proxy signature scheme is $(t', qS, qH, \varepsilon')$-secure against existential forgery under an adaptive chosen-message and chosen-warrant attack in the random oracle model for all t' and ε' for the adversary $A2$.

$$\varepsilon \geq (1 - 1/(qS + q + 1))^{qS_1 + qS_2^1 + q + 1} \cdot 1/(qS + q + 1) \cdot \varepsilon'. \tag{16}$$

$$t \leq t' + \sum_{i=1}^{n} (qH_1^i + qH_2^i + qS_2^i + 2q_i) + qH_3 + qS_1. \tag{17}$$

Proof: Suppose $A2$ is a forgery algorithm that $(t', qS, qH, \varepsilon')$-breaks the signature scheme. We will construct an algorithm C to solve co-CDH in (G_1, G_2) with probability at least ε in time at most t by using the adversary $A2$.

Algorithm C is given a generator g_2 of G_2, $u \in G_2$ and $h \in G_1$, where $u = g_2^a$. C attempts to output $h^a \in G_1$. C simulates the challenger and responds to $A2$'s queries.

- **Setup** C gives $A2$ the generator g_2 of G_2 and $u \cdot g_2^r$ as the public key of the target user 1. Here user 1 is the original signer.

- **Simulation** C responds to the queries of the adversary $A2$ as B responds to the queries of the adversary $A1$ with these exceptions that C responds to the requests of user 1 designating user i instead of that requests of user i designating user 1. In addition, C responds to a (i, l, M) proxy signature query in a different way from B.

  1. (1 designates i) $A2$ can play the role of the proxy signer i ($i = 2, 3, \cdots, n$). We assume that $A2$ makes at most q_i designation requests from user 1. C outputs a warrant ω, and then makes a response to H_1-oracle queries about $(i, 00, \omega)$. If $(i, 00, \omega, d, \cdot, \cdot)$ is in LH_1^i, C returns d. Otherwise, C randomly chooses a bit c_j so that $\Pr[c_j = 0] = 1/(qS + q + 1)$, where $q = \sum_{i=1}^{n} q_i$, $b_j \in Z_p$. If $c_j = 0$, $d_j = h \cdot \psi(g_2)^{b_j} \in G_1$; otherwise $c_j = 1$, $d_j = \psi(g_2)^{b_j} \in G_1$. Add $(i, 00, \omega_j, d_j, c_j, b_j)$ to the list LH_1^i. If $c_j = 0$, C returns FAILURE; otherwise $c_j = 1$, C lets $cert_j = \psi(u)^{b_j} \cdot \psi(g_2)^{rb_j}, r \in Z_p$. C returns $(\omega_j, cert_j)$ and adds $(\omega_j, d_j, cert_j, \cdot, \cdot)$ to the list $\mathbf{WCERT}_i$.
  2. (proxy signature by user i on behalf of user 1 with the l-th delegation-by-certificate) Assume that $A2$ makes a query (i, l, M). In the case $i = 1$, C responses to self delegation signature on message M by using the l-th self delegation certificate. If $\mathrm{WCERT}_1[l]$ in the list

$\mathbf{WCERT}_1$ has not been defined, C returns INVALID; otherwise if (1, ω_l,d_l,$cert_l$,M,d_M,s)$\in$ **SDS**, C returns s; otherwise C first answers H_2-query about (1,01, ω_l,d_l,$cert_l$,M) and returns d_M, then C responses to the signing oracle $O_S(sk_1, \cdot)$ about $01||M||\omega_1[l]||cert_1[l]$: if $c_j = 0$, C returns FAILURE, otherwise $c_j = 1$, C lets $s_j = \psi(u)^{b_j} \cdot \psi(g_2)^{r_j b_j}, r_j \in Z_p$ and returns s_j. Add $(\omega_l, d_l, cert_l, M, d_M, s_j)$ to the list **SDS**. In the case $i = 2, 3, \cdots, n$, if WCERT$_i$[l] in the list $\mathbf{WCERT}_i$ has not been defined, C returns INVALID; otherwise if $(i, 01, \omega_l, d_l, cert_l, M, d_M, \cdot, \cdot)$ is in LH_2^i, C returns d_M. Otherwise C answers H_2-query about $(i, 01, \omega_l, d_l, cert_l, M)$ and returns d_M. $A2$ uses the private key sk_i and $(\omega_l, cert_l, d_l)$ to generate a proxy signature s on message M. C checks the validity of the signature and adds the valid signature $(i, \omega_l, d_l, cert_l, M, d_M, s)$ to $\mathbf{PDS}_i$.

- **Forgery** $A2$ produces its forgery (M, s) or $(\omega, cert, M, s)$.
- **Output** Algorithm C generates its output according to the type of the forgery.

[**F1′**] If the forgery is the standard signature (M, s) of the target user, where $V(pk_P, 11||M, s) = 1$ and $(M, \cdot, s) \notin$**STMS**, C finds $(11, M, \cdot, \cdot, \cdot) \in LH_3$. If $c = 1$, C returns FAILURE; otherwise $c = 0$, C computes $h^a = s/(h^r \cdot \psi(u)^b \cdot \psi(g_2)^{rb})$.

[**F2′**] The forgery is the self-proxy signature $(\omega_1, cert_1, M, s)$ of the target user, where $V(pk_1, 00||\omega_1, cert_1) = 1$, $V(pk_1, 01||M||\omega_1||cert_1, s) = 1$ and $(\omega_1, cert_1, M, s) \notin$ **SDS**. If $(\omega_1, \cdot, cert_1) \in$**WCERT**$_1$, C finds $(1, 01, \omega_1, d_1, cert_1, M, d_M, \cdot, \cdot)$ in the list LH_2^1. Then C outputs h^a as in the event **F1′**. If $(\omega_1, \cdot, cert_1) \notin$**WCERT**$_1$, then according to $V(pk_1, 00||\omega_1, cert_1) = 1$, C can find $(1, 00, \omega_1, d_1, \cdot, \cdot)$ in the list LH_1. C outputs h^a as above.

[**F3′**] The forgery is the proxy signature $(\omega_j, cert_j, M, s)$ of the target user with the proxy signer i, where $V(pk_1, 00||\omega_j, cert_j) = 1$, $V(pk_i, 01||M||$ $\omega_j||$ $cert_j$,s)$= 1$ and $(\omega_j, cert_j, M, s) \notin$ **PDS**$_i$. In fact, since $A2$ owns the private key sk_i, it is enough to only require that $(\omega_j, cert_j, M, s)$ has $(\omega_j, \cdot, cert_j) \notin$**WCERT**$_i$. C finds $(i, 00, \omega, d_j, c_j, b_j)$ in the list LH_1^i, then C outputs h^a as in the event **F2′**.

Next, we analyze the success probability ε of C in solving the given instance of the co-CDH problem in (G_1, G_2). We make the assumption that all $A2$'s forgeries

are independent from each other, and **F1′**, **F2′** and **F3′** happen with probability at most ε'_1, ε'_2 and ε'_3, respectively, where $\varepsilon' = \varepsilon'_1 + \varepsilon'_2 + \varepsilon'_3$.

C succeeds in **F1′** if the following three events all happen.

E11′: C does not abort as a result of any of $A2$'s signature queries, hash queries and designation requests. $A2$'s signature queries includes standard signature queries and self-proxy signature queries. Noted that C can response to all the proxy signature queries with the original signer user i for all $i = 2, 3, \cdots, n$ with probability 1. $A2$'s designation requests refer to all the self delegation of user 1 and all the delegations to user i.

E12′: $A2$ produces a valid standard signature forgery (M, s).

E13′: Event **E12′** happens and $c = 0$ for $(11, M, \cdot, \cdot, \cdot)$ in the list LH_3.

By $C1$ we denote the event that C succeeds in **F1′**.

$$\begin{aligned} \Pr(\mathrm{C1}) &= \Pr[\mathrm{E11}' \wedge \mathrm{E12}' \wedge \mathrm{E13}'] \\ &= \Pr[\mathrm{E11}'] \cdot \Pr[\mathrm{E12}'|\mathrm{E11}'] \cdot \Pr[\mathrm{E13}'|\mathrm{E11}' \wedge \mathrm{E12}']. \end{aligned} \tag{18}$$

We will give the lower bound for these terms in the following claims.

Claims 1.1: $\Pr[\mathrm{E11}'] \geq (1 - 1/(qS + q + 1))^{qS_1 + qS_2^1 + q + 1}$.

Proof. If $A2$ requests user 1 to sign i-th message M of its choice, C finds $(11, M, d, c, b)$ in the list LH_3 or adds the tuple to LH_3. Prior to the query, $A2$'s view is independent of c. Moreover, d is uniformly distributed in G_1 for any bit c. Therefore the query $(11, M)$ causes C to abort with the probability at most $1/(qS+q+1)$. For qS_1 queries of user 1's standard signature, the probability that C does not abort is at least $(1-1/(qS+q+1))^{qS_1}$. If we make a similar analysis to a self proxy signature queries $(1, l, M)$, we can obtain that the probability C does not abort during making responses to the queries is at least $(1-1/(qS+q+1))^{qS_2^1}$. Similarly, the probability that C does not abort during answering requests of all the delegations is at least $(1 - 1/(qS + q + 1))^q$.

Claims 1.2: $\Pr[\mathrm{E12}'|\mathrm{E11}'] \geq \varepsilon'_1$.

Claims 1.3: $\Pr[\mathrm{E13}'|\mathrm{E11}' \wedge \mathrm{E12}'] \geq 1/(qS + q + 1)$.

From equation (18) and the above-mentioned claims, we have

$$\Pr(\mathrm{C1}) \geq (1 - 1/(qS + q + 1))^{qS_1 + qS_2^1 + q + 1} \cdot 1/(qS + q + 1) \cdot \varepsilon'_1. \tag{19}$$

We define $C2$,$C3$ similarly as we define $B2$ and $B3$. Thus, we can obtain the following result.

$$\Pr(\mathrm{C2}) \geq \varepsilon'_2 \cdot (1 - 1/(qS + q + 1))^{qS_1 + qS_2^1 + q + 1} \cdot 1/(qS + q + 1). \tag{20}$$

$$\Pr(\mathrm{C3}) \geq \varepsilon'_3 \cdot (1 - 1/(qS + q + 1))^{qS_1 + qS_2^1 + q + 1} \cdot 1/(qS + q + 1). \tag{21}$$

Therefore, the success probability ε of C in solving the given instance of the co-CDH problem in (G_1, G_2) is

$$\begin{aligned} \Pr(\mathrm{C}) &= \Pr(\mathrm{C1}) + \Pr(\mathrm{C2}) + \Pr(\mathrm{C3}) \\ &\geq (1 - 1/(qS + q + 1))^{qS_1 + qS_2^1 + q + 1} \cdot 1/(qS + q + 1) \cdot \varepsilon'. \end{aligned} \tag{22}$$

C's running time is $A2$'s running time plus the time taken to respond to the queries and requests. C responds to $\sum_{i=1}^{n} qH_1^i$ H_1-queries, $\sum_{i=1}^{n} qH_2^i$ H_2-queries and qH_3 H_3-queries. C's responses to at most q_1 self delegations queries includes at most q_1 H_1-answers and at most q_1 certification generations. For $\sum_{i=2}^{n} q_i$ delegation queries with the proxy user i, C needs at most $\sum_{i=2}^{n} q_i$ H_1-responses and at most $\sum_{i=2}^{n} q_i$ certificate generations. C's responses to qS_1 standard signature queries includes at most qS_1 H_3-answers and at most qS_1 signature generations. C's responses to $\sum_{i=1}^{n} qS_2^i$ proxy signature queries with the proxy user i only needs at most $\sum_{i=1}^{n} qS_2^i$ H_2-answers. Each response to Hash-oracle and each signature generation require an exponentiation in G_1, respectively. Hence, the total running time t of C satisfies

$$t \leq t' + \sum_{i=1}^{n} (qH_1^i + qH_2^i + qS_2^i + 2q_i) + qH_3 + qS_1. \tag{23}$$

Thus, we have completed the proof of Theorem 3. ∎