An Identity-Based Proxy Signature Scheme from Pairings

Kyung-Ah Shim*

Department of Mathematics, Ewha Womans University,
Seoul, Korea
kashim@ewha.ac.kr

Abstract. A proxy signature enables an original signer to delegate her signing capability to a proxy signer and then the proxy signer can sign a message on behalf of the original signer. In this paper we propose an ID-based proxy signature scheme from bilinear pairings. We provide exact security proof of the proposed ID-based proxy signature scheme in the random oracle model under the Computational Diffie-Hellman assumption without using Forking Lemma.

1 Introduction

In 1984, Shamir [11] introduced the concept of ID-based cryptography. In traditional public key cryptosystems, Alice's public key is a random string. When Bob wishes to send a message to Alice, he must first obtain her authenticated public key in public directories. The central idea in ID-based cryptosystems is to eliminate the public key distribution problem by making Alice's public key derivable from some known aspect of her identity, such as her email address. Such cryptosystems alleviate the certificate overhead and solve the problems of Public Key Infrastructure (PKI) technology: certificate management, including storage and distribution, and the computational cost of certificate verification. Over the years a number of researchers tried to propose secure and efficient ID-based encryption schemes, but with little success. This state of affairs changed in 2001 when an ID-based encryption scheme based on Weil pairing was proposed by Boneh and Franklin [4]. The pairings of algebraic curves have initiated some completely new fields in cryptography, making it possible to realize cryptographic primitives that were previously unknown or impractical. In particular, the pairing-based cryptosystems provide solutions for construction of special-purposed signature schemes; short signature, aggregate signature, multisignature, verifiably encrypted signature, etc. [6,5,1,3,13,15].

The concept of proxy signature was first introduced by Mambo, Usuda, and Okamoto in 1996 [8]. The proxy signature schemes allow a proxy signer to sign messages on behalf of an original signer within a given context (the context and limitations on proxy signing capabilities are captured by a certain warrant

* This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD).(KRF-2005-217-C00002).

P. Ning, S. Qing, and N. Li (Eds.): ICICS 2006, LNCS 4307, pp. 60–71, 2006.

issued by the delegator which is associated with the delegation act). Proxy signatures have found numerous practical applications, particularly in distributed computing where delegation of rights is quite common; distributed systems, Grid computing, mobile agent applications, distributed shared object systems, global distribution networks, and mobile communications. Since Mambo *et al.*'s scheme, many proxy signature schemes have been proposed [14,7,9,13,14,15]. Almost all of these works only provide informal security analysis, i.e., there are no proven-secure proxy signature schemes. The first work to formally define the model of proxy signatures, is the work of Boldyreva, Palacio, and Warinschi [2]. However, the security of almost all of (ID-based) signature and proxy signature schemes including Boldyreva *et al.*'s triple Shnorr proxy signature scheme is proved by using Forking Lemma [10], i.e., they do not provide tight security reductions. Recently, Xu *et al.* [12] proposed an ID-based proxy signature scheme from pairings. They extended Boldyreva *et al.*'s security model for proxy signature schemes to the ID-based setting and proved its security in that model without using Forking Lemma. In this paper we define a new security model for ID-based proxy signature schemes and propose a more efficient ID-based proxy signature scheme from pairings with a tight security reduction to the intractability of the Computational Diffie-Hellman problem without using Forking Lemma.

The rest of this paper is organized as follows. In the following Section, we describe basic tools and new security notions for ID-based proxy signature schemes. In Section 3, we propose an ID-based proxy signature scheme from pairings. We then provide exact security proof of the proposed ID-based proxy signature scheme in the random oracle model under the Computational Diffie-Hellman assumption. A concluding remark is given in Section 4.

2 Preliminaries

2.1 Definition and Assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of a large prime order q. We write $\mathbb{G}_1$ additively and $\mathbb{G}_2$ multiplicatively. We assume that the discrete logarithm problems in both $\mathbb{G}_1$ and $\mathbb{G}_2$ are hard.

Admissible Pairing: We call e an *admissible pairing* if $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a map with the following properties:

1. Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and for all $a, b \in \mathbb{Z}$.
2. Non-degeneracy: There exists $P \in \mathbb{G}_1$ such that $e(P, P) \neq 1$.
3. Computability: There is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

The Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties can be modified to create such admissible pairings, as in [4].

We consider the following problem and assumption in $(\mathbb{G}_1, +)$.

Definition 2.1. [Computational Diffie-Hellman (CDH) Problem]. Given (P, xP, yP), to compute xyP, where $x, y \in_R \mathbb{Z}_q^*$, and P is a generator of $\mathbb{G}_1$.

Definition 2.2. [Computational Diffie-Hellman (CDH) Assumption]. Let $\mathcal{G}$ be a CDH parameter generator. We say that an algorithm $\mathcal{A}$ has advantage $\epsilon(k)$ in solving the CDH problem for $\mathcal{G}$ if for a sufficiently large k,

$$Adv_{\mathcal{G},\mathcal{A}}(t) = Pr\left[\begin{array}{c}\mathcal{A}(q,\mathbb{G}_1,\ P,\ xP,yP) = xyP \\ |(q,\mathbb{G}_1) \leftarrow \mathcal{G}(1^k),\ P \leftarrow \mathbb{G}_1,\ x,y \leftarrow \mathbb{Z}_q^*\end{array}\right] \geq \varepsilon(k)$$

We say that $\mathcal{G}$ satisfies the CDH assumption if for any randomized polynomial-time in t algorithm $\mathcal{A}$ we have that $Adv_{\mathcal{G},\mathcal{A}}(t)$ is a negligible function. When $\mathcal{G}$ satisfies the CDH assumption we say that CDH is hard in $\mathbb{G}_1$ generated by $\mathcal{G}$.

2.2 Security Notions

We define the security notion for ID-based proxy signature schemes by simplifying Boldyeva *et al*'s and Xu *et al*'s ones [1,12]. An ID-based proxy signature scheme consists of three kinds of participants, an original signer, a proxy signer and a verifier, and the following five algorithms: `Setup, Extract, Proxy Key Extract, Sign`, and `Verify`. We remove the `Proxy Designation Protocol` and `Standard Signature Signing Algorithm` from Boldyeva *et al*'s and Xu *et al*'s ones by adding the `Proxy Key Extract` algorithm whose roles are designation of a proxy signer and to extract a proxy signing key for the designated proxy signer.

Component of ID-based proxy signature schemes. An ID-based proxy signature scheme IBPS=(`Setup`, `Extract`, `Proxy Key Extract`, `Sign`, `Verify`) is specified by five polynomial time algorithms with the following functionality;

1. The randomized parameter generation algorithm `Setup` takes input 1^k, where $k \in \mathbb{Z}$ is the security parameter and outputs some publicly known system parameters `Params`. These may contain a security parameter, the description of a cyclic group and a generator, and the description of a hash function.
2. The randomized private key extraction algorithm `Extract` takes input system parameters `Params` and an identity ID and outputs a pair (Q_{ID}, S_{ID}) consisting of a public key and the corresponding private key, respectively.
3. The randomized proxy signing key extraction algorithm `Proxy Key Extract` takes input system parameters `Params` and a pair of identities $\{ID_i, ID_j\}$ with a warrant w (it implies that an original signer ID_i designates ID_j as a proxy signer) and outputs a proxy signing key σ_P for ID_j. The order of $\{ID_i, ID_j\}$ is important, i.e, $\{ID_i, ID_j\}$ and $\{ID_j, ID_i\}$ are different inputs in the `Proxy Key Extract` algorithm.
4. The randomized proxy signing algorithm `Sign` takes input a proxy signing key corresponding to an identity ID_j, a message $m \in \{0,1\}^*$ and outputs a proxy signature $\sigma \leftarrow$`Sign`(σ_P, m).
5. The randomized verification algorithm `Verify` takes input a set of identities $\{ID_i, ID_j\}$ with a warrant w, a message $m \in \{0,1\}^*$ and a proxy signature σ of m for $\{ID_i, ID_j\}$, and outputs `True` if the signature is correct, or $\perp$ otherwise, i.e., $\{$`True`$, \perp\} \leftarrow$ `Verify`$(w, m, ID_i, ID_j, \sigma)$.

The most general security notion of a standard signature scheme is existential unforgeability under an adaptively chosen-message attack. We extend this notion to an ID-based proxy signature scheme, namely, existential unforgeability under an adaptively chosen-message and an adaptively chosen-ID attack, where an adversary can adaptively choose identities as well as messages. Informally, existential forgery here means that the adversary attempts to forge an ID-based proxy signature on identities and messages of his choice, i.e., adversary's goal is the existential forgery of a proxy signature. We give the adversary the power to choose identities on which it wishes to forge a proxy signature and the power to request private keys and proxy signing keys on all these identities. The adversary is also given access to a `Sign` oracle on any desired identity. All designation phases defined in Boldyeva *et al*'s and Xu *et al*'s ones are unified into the `Proxy Key Extract`. We formalize the ID-based proxy signature model as follows.

Unforgeability of ID-based Proxy Signature Schemes. Adversary's advantage $Adv_{\mathcal{IBPS},\mathcal{A}}$ is defined as its probability of success in the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$;

1. $\mathcal{C}$ runs `Setup` algorithms and its resulting system parameters are given to $\mathcal{A}$.
2. $\mathcal{A}$ issues the following queries;
   - `Hash Query`: $\mathcal{C}$ computes the hash value of the requested input and sends the value to $\mathcal{A}$.
   - `Extract Query`: Given an identity ID, $\mathcal{C}$ computes its private key S_{ID} corresponding to Q_{ID} derived from ID.
   - `Proxy Key Extract Query`: Proceeding adaptively, for a given pair of identities $\{ID_i, ID_j\}$ with a warrant w, i.e., it implies that an original signer ID_i designates ID_j as a proxy signer, $\mathcal{C}$ computes ID_j's proxy signing key.
   - `Sign Query`: Given a message m for $\{ID_i, ID_j\}$ with a warrant w, $\mathcal{C}$ returns a proxy signature σ.
3. $\mathcal{A}$ outputs σ on a message m for $\{ID_i, ID_j\}$ with a warrant w such that
   i) m is not equal to the inputs of any query to `Sign` under ID_j,
   ii) $\{ID_i, ID_j\}$ with a warrant w is not requested to `Proxy Key Extract` query, i.e., ID_j was not designated by ID_i as a proxy signer.

   $\mathcal{A}$ wins the game if σ is a valid proxy signature.

Definition 2.3. A forger $\mathcal{A}(t, q_E, q_{PE}, q_S, q_H, \varepsilon)$-breaks an ID-based proxy signature scheme $\mathcal{IBPS}$ if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_E `Extract` queries, q_{PE} `Proxy Key Extract` queries, q_S `Sign` queries and at most q_H queries to the hash function, and $Adv_{\mathcal{IBPS},\mathcal{A}}$ is at least ε. An ID-based proxy signature scheme is $Adv_{\mathcal{IBPS},\mathcal{A}}(t, q_E, q_{PE}, q_S, q_H, \varepsilon)$-existentially unforgeable under an adaptively chosen-message and an adaptively chosen-ID attack if no forger $Adv_{\mathcal{IBPS},\mathcal{A}}(t, q_E, q_{PE}, q_S, q_H, \varepsilon)$-breaks it.

Security Requirements of ID-based Proxy Signature Schemes. Like the general proxy signature, an ID-based proxy signature scheme should satisfy the following requirements [7,8];

1. **Distinguishability:** Proxy signatures are distinguishable from normal signatures by everyone.
2. **Verifiability:** From the proxy signature, the verifier can be convinced of the original signers agreement on the signed message.
3. **Strong Non-Forgeability:** A designated proxy signer can create a valid proxy signature for the original signer. But the original signer and other third parties who are not designated as a proxy signer cannot create a valid proxy signature.
4. **Strong Identifiability:** Anyone can determine the identity of the corresponding proxy signer from the proxy signature.
5. **Strong Non-Deniability:** Once a proxy signer creates a valid proxy signature of an original signer, he/she cannot repudiate the signature creation.
6. **Prevention of Misuse:** The proxy signer cannot use the proxy key for other purposes than generating a valid proxy signature. That is, it cannot sign messages that have not been authorized by the original signer.

3 New ID-Based Proxy Signature Scheme from Pairings

3.1 Proposed ID-Based Proxy Signature Scheme: $\mathcal{NIBPS}$

We propose a new ID-based proxy signature scheme $\mathcal{NIBPS}$ from pairings. Let A and B be an original signer and a proxy signer with identities ID_A and ID_B, respectively. The scheme consists of five algorithms; `Setup`, `Extract`, `Proxy Key Extract`, `Sign`, and `Verify`. We let k be the security parameter given to the `Setup` algorithm. The proposed ID-based proxy signature scheme $\mathcal{NIBPS}$ runs as follows;

Setup. Given a security parameter $k \in \mathbb{Z}$, the algorithm works as follows;

1. Run the parameter generator $\mathcal{G}$ on input k to generate a prime q, two groups $\mathbb{G}_1, \mathbb{G}_2$ of order q, two different generators P and Q in $\mathbb{G}_1$ and an admissible pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
2. Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{Pub} = sP$.
3. Choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_i : \{0,1\}^* \rightarrow \mathbb{Z}_q$, $i = 2, 3$. The security analysis will view H_1, H_2 and H_3 as random oracles. The system parameters is `Params`$=< q, \mathbb{G}_1, \mathbb{G}_2, e, P, Q, P_{Pub}, H_1, H_2, H_3 >$.

Extract. For a given string $ID \in \{0,1\}^*$, the algorithm does;

1. Compute $Q_{ID} = H_1(ID) \in \mathbb{G}_1$.
2. Set the private key S_{ID} to be $s \cdot Q_{ID}$, where s is a master secret.

Proxy Key Extract

1. The original signer, A prepares a warrant w which is explicit description of the delegation relation.

2. A chooses $r_A \in_R \mathbb{Z}_q^*$ and computes

$$U = r_A P \in \mathbb{G}_1,\ h_A = H_2(w, U_A) \in \mathbb{Z}_q,\ V_A = h_A S_A + r_A Q\ \in \mathbb{G}_1.$$

Then A sends (w, U_A, V_A) to the proxy signer B.

3. The proxy signer verifies whether

$$e(V_A,\ P) = e(h_A Q_A,\ P_{Pub}) \cdot e(U_A, Q)$$

holds or not. If it holds, B computes $h_B = H_3(w, U_A)$ and

$$\sigma_P = V_A + h_B S_B \in \mathbb{G}_2$$

and keeps it as a proxy signing key.

Sign. Given its proxy signing key V_P, and a message $M \in \{0,1\}^*$, B does;

1. Choose a random $r \in \mathbb{Z}_q^*$ and compute $U = rP \in \mathbb{G}_1$, $h = H_3(w, M, U) \in \mathbb{Z}_q$.
2. Compute $V = h \cdot \sigma_P + rQ \in \mathbb{G}_1$.
3. Output the proxy signature (w, M, U_A, U, V).

Verify. Given a proxy signature (w, M, U_A, U, V) for the original signer A and the proxy signer B, a verifier does;

1. Compute $Q_A = H_1(ID_A),\ Q_B = H_1(ID_B)$ and $h_A = H_2(w, U_A),\ h_B = H_3(w, U_A),\ h = H_3(w, M, U) \in \mathbb{Z}_q$.
2. Verify whether

$$e(V,\ P) = e(h[h_A Q_A + h_B Q_B],\ P_{Pub}) \cdot e(hU_A + U,\ Q)$$

holds or not. If it holds, accept the signature.

Correctness. By bilinearity of the pairing e, the consistency of the signature scheme is easy to verify;

$$e(V,\ P) = e(h \cdot \sigma_P + rQ,\ P) = e(h[h_A S_A + r_A Q + h_B S_B] + rQ,\ P)$$

$$= e(h[h_A Q_A + h_B Q_B],\ P_{Pub}) \cdot e(hU_A + U,\ Q).$$

3.2 Security Proof

Now, we prove the security of the proposed ID-based proxy signature scheme $\mathcal{NIBPS}$. Let an adversary $\mathcal{A}$ be a probabilistic polynomial time algorithm whose input is `Params`$=< q, \mathbb{G}_1, \mathbb{G}_2, e, P, Q, H_1, H_2, H_3 >$, where $q \geq 2^k$. $\mathcal{A}$ can make q_S queries to the `Sign`, q_{H_1} queries to the H_1-hash, q_{H_2} queries to the H_2-hash, q_{H_3} queries to the H_3-hash, q_E queries to the `Extract` and q_{PE} queries to the `Proxy Key Extract`.

Theorem 3.1. *If the CDH problem is (t', ε')-hard, the proposed proxy signature scheme $\mathcal{NIBPS}$ is $(t, q_{H_1}, q_{H_2}, q_{H_2}, q_E, q_{PE}, q_S, \varepsilon)$-secure against existential forgery under an adaptively chosen-message and an adaptively chosen-ID attack, for any t and ε satisfying*

$$\varepsilon \geq e \cdot (q_E + 1) \cdot \varepsilon',$$

$$t \leq t' - c_{\mathbb{G}_1}(q_{H_1} + q_E + 4q_{PE} + 6q_S + 5),$$

where e is the base of natural logarithms, and $c_{\mathbb{G}_1}$ is the time of computing a scalar multiplication $\mathbb{G}_1$ and an inversion in $\mathbb{Z}_q^*$.

Proof. Suppose that $\mathcal{A}$ is a forger who breaks the proposed ID-based proxy signature scheme $\mathcal{NIBPS}$. A CDH instance (P, xP, yP) is given for $x, y \in_R \mathbb{Z}_q^*$. By using the forgery algorithm $\mathcal{A}$, we will construct an algorithm $\mathcal{B}$ which outputs the CDH solution xyP in $\mathbb{G}_1$. Algorithm $\mathcal{B}$ performs the following simulation by interacting with forger $\mathcal{A}$.

Setup. Algorithm $\mathcal{B}$ choose a random $t \in \mathbb{Z}_q^*$, computes tP and sets $P_{Pub} = xP$ and $Q = tP$. $\mathcal{B}$ starts by giving $\mathcal{A}$ the system parameters including $\langle Q, P_{pub} \rangle$.

At any time, algorithm $\mathcal{A}$ can query the random oracles H_1, H_2 and H_3 and `Extract`, `Proxy Key Extract` and `Sign` queries. To answer these queries, $\mathcal{B}$ does the following;

H_1-queries. To respond to H_1-queries, algorithm $\mathcal{B}$ maintains a list of tuples (ID, W, b, c) as explained below. We refer to this list as the H_1-`list`. The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point $ID \in \{0,1\}^*$, algorithm $\mathcal{B}$ responds as follows;

1. If the query ID already appears on the H_1-`list` in a tuple (ID, W, b, c) then algorithm $\mathcal{B}$ responds with $H_1(ID) = W \in \mathbb{G}_1$.
2. Otherwise, $\mathcal{B}$ picks a random coin $c \in \{0,1\}$ with $Pr[c = 0] = \frac{1}{q_E+1}$.
   - If $c = 0$ then $\mathcal{B}$ computes $W = b(yP)$ for a random $b \in \mathbb{Z}_q^*$.
   - If $c = 1$ then $\mathcal{B}$ computes $W = bP$ for a random $b \in \mathbb{Z}_q^*$.

   $\mathcal{B}$ adds the tuple (ID, W, b, c) to the H_1-`list` and responds to $\mathcal{A}$ with $H_1(ID) = W$.

H_2-queries. To respond to H_2-queries, algorithm $\mathcal{B}$ maintains a list of tuples (m, U, h) as explained below. We refer to this list as the H_2-`list`. The list is initially empty. When $\mathcal{A}$ queries the oracle H_2 at a point $m \in \{0,1\}^*$, algorithm $\mathcal{B}$ responds as follows;

1. If the query m already appears on the `H2-list` in a tuple (m, U, h) then algorithm $\mathcal{B}$ responds with $H_2(m, U) = h \in \mathbb{Z}_q$.
2. Otherwise, $\mathcal{B}$ picks a random $h \in \mathbb{Z}_q$ and adds the tuple (m, U, h) to the H_2-`list` and responds to $\mathcal{A}$ with $H_2(m, U) = h$.

H_3-queries. To respond to H_3-queries, algorithm $\mathcal{B}$ maintains a list of tuples (m, U, h') as explained below. We refer to this list as the H_3-`list`. The list is

initially empty. When $\mathcal{A}$ queries the oracle H_3 at a point $m \in \{0,1\}^*$, algorithm $\mathcal{B}$ responds as follows;

1. If the query m already appears on the H_3-`list` in a tuple (m, U, h') then algorithm $\mathcal{B}$ responds with $H_3(m, U) = h' \in \mathbb{Z}_q$.
2. Otherwise, $\mathcal{B}$ picks a random $h' \in \mathbb{Z}_q$ and adds the tuple (m, U, h') to the H_3-`list` and responds to $\mathcal{A}$ with $H_3(m, U) = h'$.

Extract Queries. When $\mathcal{A}$ queries the private key corresponding to ID, $\mathcal{B}$ first finds the corresponding tuple (ID, W, b, c) from the H_1-`list`;

- If $c = 0$ then $\mathcal{B}$ fails and halts.
- Otherwise, it means that $H_1(ID) = bP$ was previously determined. Then algorithm $\mathcal{B}$ computes $S_{ID} = bP_{pub} = b(xP)$ and responds to $\mathcal{A}$ with S_{ID} as a private key of ID.

Proxy Key Extract Queries. When $\mathcal{A}$ queries a proxy signing key with inputs $\{ID_i, ID_j, w\}$ (it means that an original signer ID_i designates ID_j as a proxy signer), $\mathcal{B}$ first finds (ID_i, W_i, b_i, c_i) and (ID_j, W_j, b_j, c_j) from the H_1-`list`;

- If $c_i = 0$ or $c_j = 0$ then $\mathcal{B}$ fails and halts.
- Otherwise, it means that $H_1(ID_i) = b_iP$ and $H_1(ID_j) = b_jP$ were previously determined. Then algorithm $\mathcal{B}$ chooses $r_i \in_R \mathbb{Z}_q^*$, computes $U_i = r_iP$ and chooses $h_i, h_i' \in_R \mathbb{Z}_q$. If the tuples containing h_i and h_i' already appear in H_2-`list` and H_3-`list`, respectively then $\mathcal{B}$ chooses another r_i and tries again. Then $\mathcal{B}$ computes

$$\sigma_P = h_i(b_iP_{Pub}) + r_iQ + h_i'(b_jP_{Pub})$$

  and stores (w, U_i, h_i) and (w, U_i, h_i') in H_2-`list` and H_3-`list`, respectively. Finally, $\mathcal{B}$ responds to $\mathcal{A}$ with (U_i, σ_P) as ID_j's proxy signing key.

Sign Queries. When $\mathcal{A}$ makes a `Sign`-query on M with $\{ID_i, ID_j, w\}$, $\mathcal{B}$ first finds the corresponding tuples (ID_i, W_i, b_i, c_i) and (ID_j, W_j, b_j, c_j) from the H_1-`list`;

- If $c_i = 0$ or $c_j = 0$ then $\mathcal{B}$ fails and halts.
- Otherwise, it means that $H_1(ID_i) = b_iP$ and $H_1(ID_j) = b_jP$ were previously determined. Then algorithm $\mathcal{B}$ chooses $r_i, r_j \in_R \mathbb{Z}_q^*$ and computes $U_i = r_iP$, $U_j = r_jP$.
  - If the queries w already appear on the H_2-`list` and H_3-`list` in a tuple (w, U_i, h_i), (w, U_i, h_i') and (w, m, U_j, h_j'), respectively, then $\mathcal{B}$ uses such h_i, h_i' and h_j'.
  - Otherwise, $\mathcal{B}$ chooses random $h_i, h_i', h_j' \in_R \mathbb{Z}_q$ and stores (w, U_i, h_i), (w, U_i, h_i'), and (w, m, U_j, h_j') in the H_2-`list` and H_3-`list`, respectively. Then, $\mathcal{B}$ computes

$$V = h_j'[h_i(b_iP_{Pub}) + r_iQ + h_i'(b_jP_{Pub})] + r_jQ$$

    and responds to $\mathcal{A}$ with $\sigma = (U_i, U_j, V)$.

All responses to `Sign` queries are valid; indeed, the output (w, m, U_i, U_j, V) of `Sign` query is a valid proxy signature on m for $\{ID_i, ID_j, w\}$, to see this,

$$e(V,\ P) = e(h'_j[h_i(b_i P_{Pub}) + r_i Q + h'_i(b_j P_{Pub})] + r_j Q,\ P)$$

$$= e(h'_j V_P,\ P_{Pub}) \cdot e(h'_j U_i + U_j,\ Q).$$

If $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s `Sign` queries, `Extract` queries and `Proxy Key Extract` queries then $\mathcal{A}$'s view is identical to its view in the real attack.

Output. Eventually $\mathcal{A}$ outputs a forgery σ on a message M for $\{ID_i, ID_j, w\}$. Again by assumption, $\mathcal{A}$ has previously issued hash-queries for $\{ID_i, ID_j\}$. If the coins flipped by $\mathcal{B}$ for the query to ID_k, where k is one of i and j, did not show 0 then $\mathcal{B}$ fails (in fact, $c_i = 0$ and $c_j = 0$ do not appear simultaneously because $Pr[c = 0] = \frac{1}{q_E+1}$). Otherwise, $(c_i = 0, c_j = 1)$ or $(c_i = 1, c_j = 0)$.

- If $(c_i = 0, c_j = 1)$ then $H_1(ID_i) = b_i(yP)$ and $H_1(ID_j) = b_j P$ and $\mathcal{B}$ outputs

$$(h'_j \cdot h_i \cdot b_i)^{-1}[V - h'_j t U_i - h'_j h'_i (b_j P_{Pub}) - t U_j]$$

$$= (h'_j \cdot h_i \cdot b_i)^{-1}[h'_j h_i S_{ID_i}] = (h'_j \cdot h_i \cdot b_i)^{-1} h'_j h_i x(b_i yP) = xyP.$$

- If $(c_i = 1, c_j = 0)$ then $H_1(ID_i) = b_i P$ and $H_1(ID_j) = b_j(yP)$ and $\mathcal{B}$ outputs

$$(h'_j \cdot h'_i \cdot b_j)^{-1}[V - h'_j h_i (b_i P_{Pub}) - h'_j t U_i - t U_j]$$

$$= (h'_j \cdot h'_i \cdot b_j)^{-1}[h'_j h'_i S_{ID_j}] = (h'_j \cdot h'_i \cdot b_j)^{-1} h'_j h'_i x(b_j yP) = xyP.$$

This completes the description of algorithm $\mathcal{B}$. It remains to show that $\mathcal{B}$ solves the given instance of the CDH problem with probability at least ε'. To do so, we analyze five events needed for $\mathcal{B}$ to succeed;

- E_1: $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s `Extract` queries.
- E_2: $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s `Proxy Key Extract` queries.
- E_3: $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s `Sign` queries.
- E_4: $\mathcal{A}$ generates a valid signature forgery $(w, M, ID_i, ID_j, U_i, U_j, V)$.
- E_5: Event E_4 occurs and $c_k = 0$, where k is one of i and j such that $c_k = 0$ for the tuple containing ID_k on the H_1-`list`.

$\mathcal{B}$ succeeds if all of these events happen. The probability $Pr[E_1 \wedge E_1 \wedge E_3 \wedge E_4 \wedge E_5]$ is decomposed as

$$Pr[E_1 \wedge E_1 \wedge E_3 \wedge E_4 \wedge E_5] = Pr[E_1] \cdot Pr[E_2|E_1] \cdot Pr[E_3|E_1 \wedge E_2]$$

$$\cdot Pr[E_4|E_1 \wedge E_2 \wedge E_3] \cdot Pr[E_5|E_1 \wedge E_2 \wedge E_3 \wedge E_4] \quad \cdots\cdots \quad (1).$$

The following claims give a lower bound for each of these terms.

Claim 1. The probability that algorithm $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s `Extract` queries is at least $(1 - \frac{1}{q_E+1})^{q_E}$. Hence, $Pr[E_1] \geq (1 - \frac{1}{q_E+1})^{q_E}$.

Claim 2. The probability that algorithm $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s `Proxy Key Extract` queries is 1 since it is simulated so that $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s `Proxy Key Extract` queries under the aborted result of any of $\mathcal{A}$'s `Extract` queries. Hence, $Pr[E_2|E_1] = 1$.

Claim 3. The probability that algorithm $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s `Sign` queries is 1.

Proof. It is simulated so that $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s `Sign` queries under the aborted result of any of $\mathcal{A}$'s `Extract` and `Proxy Key Extract` queries. Hence, $Pr[E_2|E_1 \wedge E_2] = 1$.

Claim 4. If $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s `Sign` queries, `Extract` queries and `Proxy Key Extract` queries then $\mathcal{A}$'s view is identical to its view in the attack. Hence, $Pr[E_4|E_1 \wedge E_2 \wedge E_3] \geq \varepsilon$.

Claim 5. The probability that algorithm $\mathcal{B}$ does not abort after $\mathcal{A}$ outputs a valid and nontrivial forgery is at least $(1 - \frac{1}{q_E+1}) \cdot \frac{1}{q_E+1}$. Hence, $Pr[E_5|E_1 \wedge E_2 \wedge E_3 \wedge E_4] \geq (1 - \frac{1}{q_E+1}) \cdot \frac{1}{q_E+1}$.

Proof. Algorithm $\mathcal{B}$ succeeds only if $\mathcal{A}$ generates a forgery such that $c_k = 0$, where k is one of i and j for $\{ID_i, ID_j\}$. Hence, $Pr[E_5|E_1 \wedge E_2 \wedge E_3 \wedge E_4] \geq (1 - \frac{1}{q_E+1}) \cdot \frac{1}{q_E+1}$.

Algorithm $\mathcal{A}$ produces the correct forgery with probability at least

$$(1 - \frac{1}{q_E+1})^{q_E} \cdot \varepsilon \cdot \frac{1}{q_E+1}(1 - \frac{1}{q_E+1})$$

$$\geq (1 - \frac{1}{q_E+1})^{q_E+1} \cdot \varepsilon \cdot \frac{1}{q_E+1} \geq \frac{1}{e} \cdot \frac{\varepsilon}{(q_E+1)} \geq \varepsilon'$$

as required.

Algorithm $\mathcal{B}$'s running time is the same as $\mathcal{A}$'s running time plus the time is takes to respond to $(q_{H_1} + q_{H_2} + q_{H_2} + q_S)$ hash queries, q_E `Extract` queries, q_{PE} `Proxy Key Extract` queries and q_S `Sign` queries, and the time to transform $\mathcal{A}$'s final forgery into the CDH solution. The H_1, `Extract`, `Proxy Key Extract` and `Sign` queries requires 1, 1, 4, and 6 scalar multiplications. The output phase requires 4 scalar multiplications and an inversion. We assume that a scalar multiplication in $\mathbb{G}_1$ and an inversion in $\mathbb{Z}_q^*$ take time $c_{\mathbb{G}_1}$. Hence, the total running time is at most $t + c_{\mathbb{G}_1}(q_{H_1} + q_E + 4q_{PE} + 6q_S + 5) \leq t'$ as required. □

3.3 Further Security Analysis

Now, we show that our ID-based proxy signature scheme satisfies all the requirements described in the section 2.

1. **Distinguishability:** This is obvious, because there is a warrant w in a valid proxy signature, at the same time, this warrant w and the public keys of the

original signer and the proxy signer must occur in the verification equations of proxy signatures.
2. **Verifiability:** It derived from correctness of the proposed ID-based proxy signature scheme. In general, the warrant contains the identity information and the limitation of the delegated signing capacity and so satisfies the verifiability.
3. **Strong Non-Forgeability:** It derived from correctness of the Theorem 3.1.
4. **Strong Identifiability:** It contains the warrant w in a valid proxy signature, so anyone can determine the identity of the corresponding proxy signer from the warrant w.
5. **Strong Non-Deniability:** As the identifiability, the valid proxy signature contains the warrant w, which must be verified in the verification phase, it cannot be modified by the proxy signer. Thus once a proxy signer creates a valid proxy signature of an original signer, he cannot repudiate the signature creation.
6. **Prevention of Misuse:** In our proxy signature scheme, using the warrant w, we had determined the limit of the delegated signing capacity in the warrant w, so the proxy signer cannot sign some messages that have not been authorized by the original signer.

4 Conclusion

We have proposed a new ID-based proxy signature scheme from bilinear pairings with a tight security reduction without using the Forking Lemma [10]. The proposed scheme requires a scalar multiplication (two scalar multiplications can be precomputed) in signing and 3 scalar multiplications and 3 pairing computation in verification.

References

1. A. Boldyreva, Efficient threshold signature, multisignature and blind signature schemes based on the Gap-Diffie-Hellman-group signature scheme, PKC'03, LNCS 2567, Springer-Verlag (2003), pp. 31-46. 8.
2. A. Boldyreva, A. Palacio and B. Warinschi, Secure Proxy Signature Scheme for Delegation of Signing Rights, IACR ePrint Archive, available at http://eprint.iacr.org/2003/096/, 2003.
3. D. Boneh, X. Boyen, and H. Shacham, Short group signatures, Advances in Cryptology: Crypto'04, LNCS 3152, Springer-Verlag (2004), pp. 41-55.
4. D. Boneh and M. Franklin, Identity-based encryption from the Weil pairing, Advances in Cryptology: Crypto'01, LNCS 2139, Springer-Verlag (2001), pp. 213-229.
5. D. Boneh, C. Gentry, B. Lynn, and H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, Advances in Cryptology: Eurocrypt'03, LNCS 2656, Springer-Verlag (2003), pp. 416-432.
6. D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing, Advances in Cryptology: Asiacrypt'01, LNCS 2248, Springer-Verlag (2002), pp. 514-532.

7. B. Lee, H. Kim and K. Kim, Secure mobile agent using strong non-designated proxy signature, ACISP'01, LNCS 2119, Springer Verlag (2001), pp. 474-486.
8. M. Mambo, K. Usuda, and E. Okamoto, Proxy signature: Delegation of the power to sign messages, In IEICE Trans. Fundamentals, E79-A(9), pp. 1338-1353, 1996.
9. T. Okamoto, M. Tada and E. Okamoto, Extended proxy signatures for smart cards, ISW'99, LNCS 1729, Springer-Verlag (1999), pp. 247-258.
10. D. Pointcheval, and J. Stern, Security proofs for signature schemes, Advances in Cryptology: Eurocrypt'96, LNCS 1070, Springer-Verlag (1996), pp. 387-398.
11. A. Shamir, Identity-based cryptosystems and signature schemes, Advances in cryptology: Crypto'84, LNCS 196, Springer-Verlag (1884), pp. 47-53.
12. Jing Xu, Zhenfeng Zhang, Dengguo Feng, ID-Based Proxy Signature Using Bilinear Pairings, ISPA Workshops, LNCS 3559, Springer-Verlag (2005) pp. 359-367
13. F. Zhang and K. Kim, Efficient ID-based blind signature and proxy signature from pairings, ACISP'03, LNCS 2727, Springer-Verlag (2003), pp. 312-323.
14. F. Zhang, R. S. Naini and C. Y. Lin, New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairings, Cryptology ePrint Archive, Report 2003/117.
15. F. Zhang, R. Safavi-Naini, and W. Susilo, An efficient signature scheme from bilinear pairings and its application, PKC'04, LNCS 2947, Springer-Verlag (2004), pp. 277-290.