ID-Based Proxy Signature Using Bilinear Pairings

Jing Xu[1,2], Zhenfeng Zhang[1,3], and Dengguo Feng[1,3]

[1] State Key Laboratory of Information Security, P.R. China
[2] Graduate School of Chinese Academy of Sciences, Beijing 100039, P.R. China
[3] Institute of Software, Chinese Academy of Sciences, Beijing 100080, P.R.China
{xujing, zfzhang, feng}@is.iscas.ac.cn

Abstract. Identity-based (ID-based) public key cryptosystem can be a good alternative for certificate-based public key setting, especially when efficient key management and moderate security are required. A proxy signature scheme permits an entity to delegate its signing rights to another entity. But to date, no ID-based proxy signature scheme with provable security has been proposed. In this paper, we formalize a notion of security for ID-based proxy signature schemes and propose a scheme based on the bilinear pairings. We show that the security of our scheme is tightly related to the computational Diffie-Hellman assumption in the random oracle model.

keywords: ID-based signatures, proxy signatures, bilinear pairings, provable security.

1 Introduction

The paradigm of proxy signature is a method for an entity to delegate signing capabilities to other participants so that they can sign on behalf of the entity within a given context (the context and limitations on proxy signing capabilities are captured by a certain warrant issued by the delegator which is associated with the delegation act). For example, Alice the executive might want to empower Bob the secretary to sign on her behalf for a given week when Alice is out of town. Such proxy capability transfer may be defined recursively to allow high flexibility in assigning limited entitlements.

Proxy signatures have found numerous practical applications, particularly in distributed computing where delegation of rights is quite common. Examples discussed in the literature include distributed systems, Grid computing, mobile agent applications, distributed shared object systems, global distribution networks, and mobile communications. The proxy signature primitive and the first efficient solution were introduced by Mambo, Usuda and Okamoto [1]. Since then proxy signature schemes have enjoyed a considerable amount of interest from the cryptographic research community. Furthermore, various extensions of the basic proxy signature primitive have been considered. These include threshold proxy

signatures [2], blind proxy signatures [3], proxy signatures with warrant recovery [4], nominative proxy signatures [5], one-time proxy signatures [6], and proxy-anonymous proxy signatures [7].

Unfortunately, the extensive cryptographic research on the topic has brought developers more confusion than guidance because almost every other paper breaks some previously proposed construction, and proposes a new one. Very few schemes were left unbroken, and none of them has provable-security guarantees. Typically, security of these schemes is argued by presenting attacks that fail, which provides only very weak guarantees. The first work to formally define the model of proxy signatures, is the work of Boldyreva, Palacio, and Warinschi [8]. Recently, Malkin, Obana and Yung develop the first formal model for fully hierarchical proxy signatures and prove that proxy signatures are equivalent to key-insulated signatures [9].

In a certificate-based public key system, before using the public key of a user, the participants must verify the certificate of the user at first. As a consequence, this system requires a large storage and computing time to store and verify each users public key and the corresponding certificate. In 1984 Shamir [10] proposed ID-based encryption and signature schemes to simplify key management procedures in certificate-based public key setting. Since then, many ID-based encryption and signature schemes have been proposed. The main idea of ID-based cryptosystems is that the identity information of each user works as his/her public key, in other words, the user's public key can be calculated directly from his/her identity rather than being extracted from a certificate issued by a certificate authority (CA). ID-based public key setting can be a good alternative for certificate-based public key setting, especially when efficient key management and moderate security are required.

The bilinear pairings, namely the weil-pairing and the tate-pairing of algebraic curves, are important tools for research on algebraic geometry. They have been found various applications in cryptography recently [11],[12],[13],[14]. More precisely, they can be used to construct ID-based cryptographic schemes.

In the area of provable security, the last couple of years saw the rise of a new trend consisting of providing tight security reductions for asymmetric cryptosystems : the security of a cryptographic protocol is said to be tightly related to a hard computational problem if an attacker against the scheme implies an efficient algorithm solving the problem with roughly the same advantage. But up to now, no one proposes an ID-based proxy signature scheme providing tight security reductions.

Our current work is aimed at filling this void. Based on the work of [8] and [9], we define a formal model for the security of ID-based proxy signature scheme. Then we propose an efficient ID-based proxy signature scheme whose security can be proved tightly related to computational Diffie-Hellman (CDH) problem in the random oracle model. Unlike [8], we do not rely on the forking lemma in our security reduction, hence the advantage relation can be shown to be linear, which is almost the best possible.

The rest of the paper is organized as follows. In Section 2 we give formal definitions of presumed hard computational problems from which our reductions are made. In Section 3 a formal security model of ID-based proxy signature scheme is given. In Sections 4 and 5, we present an ID-based proxy signature scheme and analyze its security , respectively. And we end with concluding remarks in Section 6.

2 Definitions

2.1 The Bilinear Pairing

Let G be a cyclic additive group generated by P, whose order is a prime q, and V be a cyclic multiplicative group of the same order. Let $\hat{e} : G \times G \to V$ be a pairing which satisfies the following conditions:

1. Bilinearity: For any $P, Q, R \in G$, we have $\hat{e}(P + Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$ and $\hat{e}(P, Q + R) = \hat{e}(P, Q)\hat{e}(P, R)$. In particular, for any $a, b \in \mathbf{Z}_q$,

$$\hat{e}(aP, bP) = \hat{e}(P, P)^{ab} = \hat{e}(P, abP) = \hat{e}(abP, P).$$

2. Non-degeneracy: There exists $P, Q \in G$, such that $\hat{e}(P, Q) \neq 1$.

3. Computability: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in G$.

The typical way of obtaining such pairings is by deriving them from the weil-pairing or the tate-pairing on an elliptic curve over a finite field.

2.2 Gap Diffie-Hellman (GDH) Groups

Let G be a cyclic group of prime order q and P be a generator of G.

1. The decisional Diffie-Hellman (DDH) problem is to decide whether $c = ab$ in Z/qZ for given $P, aP, bP, cP \in G$. If so, (P, aP, bP, cP) is called a valid Diffie-Hellman (DH) tuple.

2. The computational Diffie-Hellman (CDH) problem is to compute abP for given $P, aP, bP \in G$.

Definition 2.1 The advantage of an algorithm $\mathcal{F}$ in solving the computational Diffie-Hellman problem on group G is

$$AdvCDH_{\mathcal{F}} = Pr[\mathcal{F}(P, aP, bP) = abP : \forall a, b \in Z_q]$$

The probability is taken over the choice of a, b and $\mathcal{F}'$s coin tosses. An algorithm $\mathcal{F}$ is said (t, ε)-breaks the computational Diffie-Hellman problem on G if $\mathcal{F}$ runs in time at most t, and $AdvCDH_{\mathcal{F}}$ is at least ε.

Now we present a definition for a gap Diffie-Hellman (GDH) group.

Definition 2.2 A group G is a (t, ε)-gap Diffie-Hellman (GDH) group if the decisional Diffie-Hellman problem in G can be efficiently computable and there exists no algorithm (t, ε)-breaks computational Diffie-Hellman on G.

If we have an admissible bilinear pairing $\hat{e}$ in G, we can solve the DDH problem in G efficiently as follows:

(P, aP, bP, cP) is a valid DH tuple $\Leftrightarrow \hat{e}(aP, bP) = \hat{e}(P, cP)$
Hence an elliptic curve becomes an instance of a GDH group if the Weil (or the Tate) pairing is efficiently computable and the CDH is sufficiently hard on the curve.

2.3 ID-Based Setting from Bilinear Pairings

The ID-based public key systems allow some public information of the user such as name, address and email *etc.*, rather than an arbitrary string to be used as his public key. The private key of the user is calculated by a trusted party, called PKG and sent to the user via a secure channel.

ID-based public key setting from bilinear pairings can be implemented as follows:

Let G be a cyclic additive group generated by P, whose order is a prime q, and V be a cyclic multiplicative group of the same order.A bilinear pairing is the map $\hat{e} : G \times G \rightarrow V$. Define cryptographic hash function $H : \{0,1\}^* \rightarrow G$.

– $\mathcal{G}$**:** PKG chooses a random number $s \in Z_q^*$ and sets $P_{pub} = sP$. He publishes system parameters $params = \{G, V, \hat{e}, q, P, P_{pub}, H\}$; and keeps s secretly as the *master-key*.
– $\mathcal{K}$**:** A user submits his/her identity information ID and authenticates him to PKG. PKG computes the user′s private key $d_{ID} = sQ_{ID} = sH(ID)$ and sends it to the user via a secure channel.

3 ID-Based Proxy Signature

Based on the work of [8] and [9], we give formal definition for ID-based proxy signature schemes.

3.1 Syntax of ID-Based Proxy Signature Schemes

Definition 3.1 An ID-based proxy signature scheme is a tuple $(\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$, where algorithms $\mathcal{G}$ and $\mathcal{K}$ are the same as in section 2.3, and the others are defined as follows.
– $\mathcal{S}$**:** The signing algorithm, which takes a signing key d_{ID} of original designator and a message m_ω as input, outputs a signature ω called warrant on m_ω. The message m_ω contains the identity(ID) of the designated proxy signer and, possibly, restrictions on the message the proxy signer is allowed to sign.
– $\mathcal{V}$**:** The verification algorithm, which takes ID of original designator, a message m_ω, and a warrant ω as input, outputs "*accept*" if the signature is valid, or "*reject*" otherwise.
– $(\mathcal{D}, \mathcal{P})$**:** (interactive) Proxy-designation algorithms (where $\mathcal{D}$ and $\mathcal{P}$ are owned by the designator ID_i and the proxy signer ID_j, respectively). The input to each algorithm includes ID_i, ID_j. $\mathcal{D}$ also takes as input the secret key d_i of the designator, a message m_ω and a warrant ω. $\mathcal{P}$ also takes as input the secret key d_j

of the proxy signer. As result of the interaction, the expected local output of $\mathcal{P}$ is the warrant ω and skp, a proxy signing key that user ID_j uses to produce proxy signatures on behalf of user ID_i. $\mathcal{D}$ has no local output.
– $\mathcal{PS}$**:** The proxy signing algorithm, which takes a proxy signing key skp, a message m and a warrant ω as input, outputs a proxy signature $psig$.
– $\mathcal{PV}$**:** The proxy verification algorithm, which takes the identity of the original designator(ID), a message m, a warrant ω and a proxy signature $psig$ as input, outputs "*accept*" if the proxy signature is valid, or "*reject*" otherwise.
– $\mathcal{ID}$**:** The proxy identification algorithm, which takes a warrant ω and a proxy signature $psig$ as input, outputs an identity of the designated proxy signer.

Correctness. We require that for all message m and all users $i, j \in \mathbb{N}$, if the proxy signing key skp and the warrant ω are the output of consecutive executions of $(skp, \omega) \leftarrow [\mathcal{D}(ID_i, ID_j, d_i, m_\omega, \omega), \mathcal{P}(ID_i, ID_j, d_j)]$, then $\mathcal{PV}(ID_i, m, \omega, \mathcal{PS}(skp, m, \omega)) = 1$, $\mathcal{ID}(\omega, \mathcal{PS}(skp, m, \omega)) = ID_j$ and the message m does not violate the warrant ω.

3.2 ID-Based Proxy Signature Security

We first informally describe some of the features of our adversarial model.

We model a seemingly extreme case in which the adversary is working against a single honest user, say ID_1, and can extract the private keys of all other users. Since any attack can be carried out in the presence of more honest users, our assumption is without loss of generality. The adversary can play the role of user $ID_i (i \neq 1)$ in executions of the $(\mathcal{D}, \mathcal{P})$ protocol with ID_1, as designator or as proxy signer. In both cases, the adversary may behave dishonestly in an attempt to obtain information from ID_1. The adversary also can request ID_1 to run the $(\mathcal{D}, \mathcal{P})$ protocol with himself, and see the transcript of the execution.We emphasize that we do not assume the existence of a secure channel between a designator and a proxy signer.

We model chosen-message attack capabilities by providing the adversary access to two oracles: a standard signing oracle and a proxy signing oracle. The first oracle takes input a message m, and returns a standard signature for m by user ID_1. The second oracle takes input a tuple (i, l, m), and, if user ID_1 was designated by user ID_i at least l times, returns a proxy signature for m created by user ID_1 on behalf of user ID_i, using the l-th proxy signing key.

The goal of the adversary is to produce one of the following forgeries: (1) a standard signature by user ID_1 for a message that was not submitted to the standard signing oracle, (2) a proxy signature for a message m, such that no query (i, l, m) was made to the proxy signing oracle, or (3) a proxy signature for a message m by some user ID_i on behalf of user ID_1, such that user ID_i was never designated by user ID_1.

ID-based proxy signature security is formally defined as follows.

Definition 3.2 Let **PS**$= (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ be an ID-based proxy signature scheme. Consider an experiment $\mathbf{Exp}^{ps-uf}_{PS,\mathcal{A}}(k)$ related to scheme **PS**, adversary $\mathcal{A}$, and security parameter k. First, system parameters **params** are

generated by running $\mathcal{G}$ on input 1^k. Then the private key d_1 of user ID_1 is generated by $\mathcal{K}$. The empty arrays $\mathbf{skp_i}$ of each user and an empty set D are created. Adversary $\mathcal{A}$ can make the following requests or queries,in any order and any number of times.

- (**Extraction Queries**) Given an identity $ID_i (i \neq 1)$, the challenger returns the private key d_i corresponding to ID_i.
- (ID_1 **Designates** ID_i **Requests**) $\mathcal{A}$ can request to interact with user ID_1 running $\mathcal{D}(ID_1, ID_i, d_1)$, for some $i \neq 1$, and play the role of userID_i running $\mathcal{P}(ID_1, ID_i, d_i)$; after a successful run, D is set to $D \bigcup \{ID_i\}$.
- (ID_i **Designates** ID_1 **Requests**) $\mathcal{A}$ can request to interact with user ID_1 running $\mathcal{P}(ID_i, ID_1, d_1)$, for some $i \neq 1$, and play the role of userID_i running $\mathcal{D}(ID_i, ID_1, d_i)$. The private output skp of $\mathcal{P}$ is stored in the last unoccupied position of $\mathbf{skp_i}$. We emphasize that $\mathcal{A}$ does not have access to the elements of $\mathbf{skp_i}$.
- (ID_1 **Designates** ID_1 **Requests**) $\mathcal{A}$ can request that user ID_1 run $(\mathcal{D}, \mathcal{P})$ protocol with itself, and see the transcript of the interaction. The private output skp of ID_1 is stored in the next available position of $\mathbf{skp_1}$. $\mathcal{A}$ does not have access to the elements of $\mathbf{skp_1}$.
- (**Standard Signature Queries**) $\mathcal{A}$ can query signatures with respect to identity ID_1 on messages of his choice.
- (**Proxy Signature Queries**) $\mathcal{A}$ can query proxy signatures by ID_1 on behalf of ID_i using the l-th proxy signing key, i.e. query (i, l, m). If key $\mathbf{skp_i}[l]$ has already been defined, we say the query is valid and the challenger returns $\mathcal{PS}(\mathbf{skp_i}[l], m)$; if $\mathbf{skp_i}[l]$ has not been defined, the query is said to be invalid and the challenger returns $\perp$.

Eventually, $\mathcal{A}$ outputs a forgery (m, sig) or $(m, psig, ID)$. The output of the experiments is determined as follows:

1. If the forgery is of the form (m, sig), where $\mathcal{V}(ID_1, m, sig) = 1$, and m was not queried to standard signature oracle, then return 1.[forgery of a standard signature]

2. If the forgery is of the form $(m, psig, ID)$, where $ID = ID_i$ for some $i \neq 1$, $\mathcal{PV}(ID_i, m, psig) = 1$, $\mathcal{ID}(psig) = ID_1$, and no valid query (i, l, m) was made to proxy signature oracle, then return 1. [forgery of a proxy signature by user ID_1 on behalf of user ID_i]

3. If the forgery is of the form $(m, psig, ID_1)$, where $\mathcal{PV}(ID_1, m, psig) = 1$ and $\mathcal{ID}(psig) \notin D \cup \{ID_1\} \cup \{\perp\}$, then return 1. [forgery of a proxy signature by user ID_i on behalf of user ID_1; user ID_i was not designated by user ID_1]

4.Otherwise, return 0.

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

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

Adversary $\mathcal{A}$ is said $(t, q_H, q_E, q_{\mathcal{S}}, q_{\mathcal{PS}}, \varepsilon)$-breaks a proxy signature scheme if: $\mathcal{A}$ runs in time at most t; $\mathcal{A}$ makes at most q_H queries to the hash function H, at most q_E queries to the key extraction oracle, at most $q_{\mathcal{S}}$ queries

to the standard signing oracle and at most $q_{\mathcal{PS}}$ queries to the proxy signing oracle; and $\mathbf{Adv}^{ps-uf}_{PS,\mathcal{A}}(k)$ is at least ε. We say a proxy signature scheme is $(t, q_H, q_E, q_{\mathcal{S}}, q_{\mathcal{PS}}, \varepsilon)$-secure if no adversary $(t, q_H, q_E, q_{\mathcal{S}}, q_{\mathcal{PS}}, \varepsilon)$-breaks it.

4 Our Proxy Signature Scheme

Our proxy signature scheme is based on SOK-IBS (Sakai-Ogishi-Kasahara Identity Based Signature)[15].The constituent algorithms of our proxy signature scheme **PS**= $(\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ are defined as follows.

- $\mathcal{G}$**:** Assume k is a security parameter.G is a GDH group of prime order $q > 2^k$ generated by P, and $\hat{e} : G \times G \to V$ is a bilinear map. Pick a random master key $s \in Z_q^*$ and set $P_{pub} = sP$. Choose hash functions $H_1, H_2, H_3 : \{0,1\}^* \to G$, and hash function $H_4 : \{0,1\}^* \to Z_q^*$
- $\mathcal{K}$**:** Given a users identity ID, compute $Q_{ID} = H_1(ID) \in G$ and the associated private key $d_{ID} = sQ_{ID} \in G$.
- $\mathcal{S}$**:** Given the private key d_i of original designator ID_i, in order to sign a message m_ω,

  1. Randomly pick $r_\omega \in Z_q^*$ and compute $U_\omega = r_\omega P \in G$ and then put $H_\omega = H_2(ID_i, m_\omega, U_\omega) \in G$.
  2.Compute $V_\omega = d_i + r_\omega H_\omega \in G$.

  The signature on m_ω is the warrant $\omega = \langle U_\omega, V_\omega \rangle$
- $\mathcal{V}$**:** To verify a signature $\omega = \langle U_\omega, V_\omega \rangle$ on a message m_ω for an identity ID_i, the verifier first takes $Q_i = H_1(ID_i) \in G$ and $H_\omega = H_2(ID_i, m_\omega, U_\omega) \in G$. He then accepts the signature if $\hat{e}(P, V_\omega) = \hat{e}(P_{pub}, Q_i)\hat{e}(U_\omega, H_\omega)$ and rejects it otherwise.
- $(\mathcal{D}, \mathcal{P})$**:** In order to designate user ID_j as a proxy signer, user ID_i sends user ID_j a message m_ω and an appropriate warrant ω. The user ID_j verifies this signature ω,and if it is valid, he computes a proxy signing key as

$$skp = H_4(ID_i, ID_j, m_\omega, U_\omega)d_j + V_\omega.$$

- $\mathcal{PS}$**:** Given proxy signing key skp, in order to sign a message m on behalf of user ID_i,

  1. Randomly pick $r_p \in Z_q^*$ and compute $U_p = r_p P \in G$ and then put $H_p = H_3(ID_j, m, U_p) \in G$.
  2.Compute $V_p = skp + r_p H_p \in G$.

  The proxy signature for message m on behalf of user ID_i produced by user ID_j is $psig = (m_\omega, ID_j, U_\omega, U_p, V_p)$
- $\mathcal{PV}$**:** To verify a proxy signature $psig = (m_\omega, ID_j, U_\omega, U_p, V_p)$ for message m with the original designator's identity ID_i, the verifier first takes $Q_i = H_1(ID_i) \in G$, $Q_j = H_1(ID_j) \in G$, $H_\omega = H_2(ID_i, m_\omega, U_\omega) \in G$ and $H_p = H_3(ID_j, m, U_p) \in G$. He then accepts the signature if

$$\hat{e}(P, V_p) = \hat{e}(P_{pub}, Q_j)^{H_4(ID_i, ID_j, m_\omega, U_\omega)}\hat{e}(P_{pub}, Q_i)\hat{e}(U_p, H_p)\hat{e}(U_\omega, H_\omega)$$

and rejects it otherwise.
- $\mathcal{ID}$**:** Given a proxy signature $psig = (m_\omega, ID_j, U_\omega, U_p, V_p)$ for message m, the proxy identification algorithm is defined as $\mathcal{ID}(psig) = ID_j$.

5 Analysis of Our Scheme

5.1 Correctness

Correctness. The proxy signature scheme is correct because of the following.

$$\begin{aligned}
&\hat{e}(P, V_p) \\
&= \hat{e}(P, skp + r_p H_p) \\
&= \hat{e}(P, H_4(ID_i, ID_j, m_\omega, U_\omega) d_j + V_\omega + r_p H_p) \\
&= \hat{e}(P, H_4(ID_i, ID_j, m_\omega, U_\omega) d_j + d_i + r_\omega H_\omega + r_p H_p) \\
&= \hat{e}(P_{pub}, Q_j)^{H_3(ID_i, ID_j, m_\omega, U_\omega)} \hat{e}(P_{pub}, Q_i) \hat{e}(U_p, H_p) \hat{e}(U_\omega, H_\omega).
\end{aligned}$$

5.2 Security

The following theorem formally relates the security of our scheme to computational Diffie-Hellman assumption in the random oracle model.

Theorem 5.1. *Given a security parameter k, let G be a (t', ε')-GDH group of prime order $q > 2^k$. P be a generator of G, and $\hat{e} : G \times G \to V$ be a bilinear map. Then the ID-based proxy signature scheme on G is $(t, q_H, q_E, q_{\mathcal{S}}, q_{\mathcal{PS}}, \varepsilon)$-secure against forgery for any t and ε satisfying*

$$\varepsilon \geq 4e(q_E + 1)\left(1 - q_{\mathcal{S}}(q_{H_2} + q_{\mathcal{S}})2^{-k}\right)^{-1}\left(1 - q_{\mathcal{PS}}(q_{H_3} + q_{\mathcal{PS}})2^{-k}\right)^{-1}\varepsilon'$$

$$t \leq t' - C_G(q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 5q_{\mathcal{S}} + 7q_{\mathcal{PS}} + 4)$$

where e is the base of natural logarithms, and C_G is the time of computing a scalar multiplication and inversion on G.

Proof. Suppose adversary $\mathcal{A}(t, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_E, q_{\mathcal{S}}, q_{\mathcal{PS}}, \varepsilon)$-breaks the proxy signature scheme. We show how to construct a t'-time algorithm $\mathcal{C}$ that solves CDH in G with probability at least ε'. This will contradict the fact that G is a (t', ε')-GDH group.

Algorithm $\mathcal{C}$ is given $X = xP \in G$ and $Y = yP \in G$. Its goal is to output $xY = xyP \in G$. Algorithm $\mathcal{C}$ simulates the challenger and interacts with adversary $\mathcal{A}$ as follows.

***Setup*:** Algorithm $\mathcal{C}$ initializes $\mathcal{A}$ with $P_{pub} = X$ as a systems overall public key, provides $\mathcal{A}$ with a randomly generated identity ID_1 and creates an empty array $\mathbf{wskp_1}$.

Queries on oracle H_1**:** At any time adversary $\mathcal{A}$ can query the random oracle H_1. To respond to these queries, $\mathcal{C}$ maintains a list L_1 of tuples $\langle ID_i, b_i, c_i \rangle$ as

explained below. The list is initially empty. When an identity ID is submitted to the H_1 oracle, algorithm $\mathcal{C}$ responds as follows:

1. If the query ID already appears on the list L_1 in some tuple $\langle ID, b, c\rangle$ then algorithm $\mathcal{C}$ with $H_1(ID) = cbP + (1-c)bY$.

2. Otherwise, $\mathcal{C}$ generates a random coin $c \in \{0,1\}$ such that $Pr[c=0] = \frac{1}{q_E+1}$.

3. Algorithm $\mathcal{C}$ picks a random $b \in Z_q^*$, adds the tuple $\langle ID, b, c\rangle$ to the list L_1. If $c = 0$ holds, $\mathcal{C}$ responds to $\mathcal{A}$ with bY, with bP otherwise.

Queries on oracle H_2**:** To respond to queries to H_2 oracle, $\mathcal{C}$ maintains a list L_2 of tuples $\langle ID_i, m_i, U_i, v_i, s_i\rangle$ as explained below. When a tuple$\langle ID, m, U\rangle$ is submitted to the H_2 oracle, algorithm $\mathcal{C}$ responds as follows:

1. If the query tuple already appears on the list L_2 in some tuple $\langle ID, m, U, v, s\rangle$, then algorithm $\mathcal{C}$ responds with $H_2(ID, m, U) = svP + (1-s)vP_{pub} \in G$.

2. Otherwise, $\mathcal{C}$ generates a random coin $s \in \{0,1\}$ such that $Pr[s=0] = 1/2$

3. Algorithm $\mathcal{C}$ picks $v \in Z_q^*$ at random, stores the tuple $\langle ID, m, U, v, s\rangle$ in the list L_2.If $s = 0$ holds, $\mathcal{C}$ responds to $\mathcal{A}$ with vP_{pub}, with vP otherwise.

Queries on oracle H_3**:** To respond to queries to H_3 oracle, $\mathcal{C}$ maintains a list L_3 of tuples $\langle ID_i, m_i, U_i, \eta_i\rangle$ as explained below. When a tuple$\langle ID, m, U\rangle$ is submitted to the H_3 oracle, algorithm $\mathcal{C}$ responds as follows:

1. If the query tuple already appears on the list L_3 in some tuple $\langle ID, m, U, \eta\rangle$, then algorithm $\mathcal{C}$ responds with $H_3(ID, m, U) = \eta P \in G$.

2. Otherwise,algorithm $\mathcal{C}$ picks $\eta \in Z_q^*$ at random, stores the tuple $\langle ID, m, U, \eta\rangle$ in the list L_3 and returns ηP as a hash value to $\mathcal{A}$.

Queries on oracle H_4**:** To respond to queries to H_4 oracle, $\mathcal{C}$ maintains a list L_4 of tuples $\langle ID_i, ID_i', m_i, U_i, \mu_i\rangle$ as explained below. When a tuple$\langle ID, ID', m, U\rangle$ is submitted to the H_4 oracle, algorithm $\mathcal{C}$ responds as follows:

1. If the query tuple already appears on the list L_4 in some tuple $\langle ID, ID', m, U, \mu\rangle$, then algorithm $\mathcal{C}$ responds with $H_4(ID, ID', m, U) = \mu \in Z_q^*$.

2. Otherwise, algorithm $\mathcal{C}$ picks $\mu \in Z_q^*$ at random, stores the tuple $\langle ID, ID', m, U, \mu\rangle$ in the list L_4 and returns μ as a hash value to $\mathcal{A}$.

***Extraction Queries*:** When $\mathcal{A}$ requests the private key associated to an identity $ID_i (i \neq 1)$, $\mathcal{C}$ recovers the corresponding $\langle ID_i, b_i, c_i\rangle$ from L_1. If $c_i = 0$, then output "*failure*" and halts. Otherwise, it means that $H_1(ID_i)$ was previously defined to be b_iP and $b_iP_{pub} = b_iX \in G$ is then returned to $\mathcal{A}$ as a private key associated to ID_i.

ID_1 ***Designates*** ID_i ***Requests*:** If $\mathcal{A}$ requests to interact with $\mathcal{D}(ID_1, ID_i, d_1)$, for some $i \neq 1$, playing the role of $\mathcal{P}(ID_1, ID_i, d_i)$, $\mathcal{C}$ creates an appropriate message m_ω and makes query to signing oracle $\mathcal{O}_{\mathcal{S}}(d_1, m_\omega)$. Upon receiving an answer ω, it forwards m_ω, ω to $\mathcal{A}$.

ID_i ***Designates*** ID_1 ***Requests*:** If $\mathcal{A}$ requests to interact with $\mathcal{P}(ID_i, ID_1, d_1)$, for some $i \neq 1$, playing the role of $\mathcal{D}(ID_i, ID_1, d_i)$, when $\mathcal{A}$ outputs $m_\omega, \omega = \langle U_\omega, V_\omega\rangle$, challenger $\mathcal{C}$ verifies that $\langle U_\omega, V_\omega\rangle$ is a valid signature for message m_ω. If so, $\mathcal{C}$ stores $m_\omega, \langle U_\omega, V_\omega\rangle$ in the last unoccupied position of $\mathbf{wskp_i}$.

ID_1 ***Designates*** ID_1 ***Requests*:** If $\mathcal{A}$ requests user ID_1 run $(\mathcal{D}, \mathcal{P})$ protocol with itself, $\mathcal{C}$ creates an appropriate message m_ω and makes query to signing

oracle $\mathcal{O}_{\mathcal{S}}(d_1, m_\omega)$. Upon receiving an answer $\langle U_\omega, V_\omega \rangle$, it stores $m_\omega, \langle U_\omega, V_\omega \rangle$ in the last unoccupied position of $\mathbf{wskp_1}$.

***Standard Signature Queries*:** If $\mathcal{A}$ queries signing oracle $\mathcal{O}_{\mathcal{S}}(d_1, m)$, Algorithm $\mathcal{C}$ responds to this query as follows: algorithm $\mathcal{C}$ first recovers the previously defined value $Q_1 = H_1(ID_1) \in G$ from the list L_1. It then chooses $r_1, r_2 \in Z_q^*$ at random, sets $V = r_1 P_{pub} = r_1 X \in G$, $U = r_2 P_{pub} = r_2 X \in G$ and defines the hash value $H_2(ID_1, m, U)$ as $r_2^{-1}(r_1 P - Q_1) \in G$ ($\mathcal{C}$ output "*failure*" and halts if H_2 turns out to be already defined for the input $\langle ID_1, m, U \rangle$). The pair (U, V) is a valid signature on message m under ID_1. Algorithm $\mathcal{C}$ gives (U, V) to $\mathcal{A}$.

***Proxy Signature Queries*:** If $\mathcal{A}$ queries proxy signatures by ID_1 on behalf of ID_i using the l-th proxy signing key, i.e. query (i, l, m), Algorithm $\mathcal{C}$ responds to this query as follows. If $\mathbf{wskp_i[l]}$ is not defined, it returns $\perp$ to $\mathcal{A}$. Otherwise, it parses $\mathbf{wskp_i[l]}$ as $m_\omega, \langle U_\omega, V_\omega \rangle$, and performs the following operations.

- Recover the previously defined value $Q_1 = H_1(ID_1) \in G$ and $Q_i = H_1(ID_i) \in G$ from the list L_1.
- Make query $(ID_i, m_\omega, U_\omega)$ to H_2 oracle, $\mathcal{C}$ recovers the corresponding $\langle ID_i, m_\omega, U_\omega, v, s \rangle$ from L_2. If $s = 1$, then output "*failure*" and halts. Otherwise, it means that $H_2(ID_i, m_\omega, U_\omega)$ was previously defined to be vP_{pub}.
- Make query $(ID_i, ID_1, m_\omega, U_\omega)$ to H_4 oracle and return $H_4(ID_i, ID_1, m_\omega, U_\omega) = \mu$.
- Pick $r_1, r_2 \in Z_q^*$ at random and define $V_p = r_1 P_{pub} \in G$, $U_p = r_2 P_{pub} \in G$.
- Define the hash value $H_3(ID_1, m, U_p)$ as $r_2^{-1}(r_1 P - Q_i - \mu Q_1 - v U_\omega) \in G$ ($\mathcal{C}$ output "*failure*" and halts if H_3 turns out to be already defined for the input $\langle ID_1, m, U_p \rangle$).
- $(m_\omega, ID_1, U_\omega, U_p, V_p)$ is a valid proxy signature by ID_1 on behalf of ID_i using the l-th proxy signing key.

***Output*:** Finally, $\mathcal{A}$ halts and at least one of the following cases occurs.

1. $\mathcal{A}$ concedes failure, in which case so does $\mathcal{C}$.

2. $\mathcal{A}$ outputs a standard signature forgery (m, sig) and no query $\mathcal{O}_{\mathcal{S}}(d_1, m)$ was made.

3. $\mathcal{A}$ outputs a forgery of a proxy signature $(m, psig, ID_i)$ by user ID_1 on behalf of user ID_i and no valid query (i, l, m) was made.

4. $\mathcal{A}$ outputs a forgery of a proxy signature $(m, psig, ID_1)$ by user ID_i on behalf of user ID_1, and user ID_i was not designated by user ID_1.

Now we only consider case 3, as case 2 and case 4 are similar to case 3. If $\mathcal{A}$ outputs a forgery of a proxy signature $(m^*, psig^*, ID_i)$ by user ID_1 on behalf of user ID_i where $psig^* = (m_\omega^*, ID_1, U_\omega^*, U_p^*, V_p^*)$, then algorithm $\mathcal{C}$ recovers $\langle ID_1, b_1, c_1 \rangle$ and $\langle ID_i, b_i, c_i \rangle$ on the list L_1.

Algorithm $\mathcal{C}$ proceeds only if $c_1 = 0$ and $c_i = 1$. Otherwise, $\mathcal{C}$ declares failure and halts. It follows $Q_1 = b_1 Y$ and $Q_i = b_i P$. The proxy signature $(m^*, psig^*, ID_i)$ must satisfy verification equation

$$\hat{e}(P, V_p^*) = \hat{e}(P_{pub}, Q_1)^{H_4(ID_i, ID_1, m_\omega^*, U_\omega^*)} \hat{e}(P_{pub}, Q_i) \hat{e}(U_p^*, H_p^*) \hat{e}(U_\omega^*, H_\omega^*).$$

Next, algorithm $\mathcal{C}$ recovers $\langle ID_i, ID_1, m^*_\omega, U^*_\omega, \mu^* \rangle$ on the list L_4 ($H_4(ID_i, ID_1, m^*_\omega, U^*_\omega) = \mu^*$), $\langle ID_1, m^*, U^*_p, \eta^* \rangle$ on the list L_3 ($H^*_p = H_3(ID_1, m^*, U^*_p) = \eta^* P$) and $\langle ID_i, m^*_\omega, U^*_\omega, v^*, s^* \rangle$ on the list L_2. $\mathcal{C}$ declares failure and halts if $s^* = 0$. Otherwise, $H^*_\omega = H_2(ID_i, m^*_\omega, U^*_\omega) = v^* P$. Then we can deduce

$$\hat{e}(P, V^*_p) = \hat{e}(\mu^* P_{pub}, b_1 Y)\hat{e}(P_{pub}, b_i P)\hat{e}(U^*_p, \eta^* P)\hat{e}(U^*_\omega, v^* P).$$

$\mathcal{C}$ calculates and outputs the required xY as $b_1^{-1}\mu^{*-1}(V^*_p - b_i P_{pub} - \eta^* U^*_p - v^* U^*_\omega)$.

This completes the description of algorithm $\mathcal{C}$. To complete the proof, we shall show that $\mathcal{C}$ solves the given instance of CDH problem in with probability at least ε'. First, we analyze the five events needed for $\mathcal{C}$ to succeed:

- Σ_1: $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}$' key extraction queries.
- Σ_2: $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}$' standard signature queries.
- Σ_3: $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}$' proxy signature queries.
- Σ_4: $\mathcal{A}$ generates a valid and nontrivial proxy signature forgery.
- Σ_5: Event Σ_4 occurs, and, in addition, $c_1 = 0$, $c_i = 1$ and $s^* = 0$. Here c_1 and c_i are the c-component of the tuple on the list L_1. s^* is the s-component of the tuple on the list L_2.

Algorithm $\mathcal{C}$ succeeds if all of these events happen. The probability $Pr[\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3 \wedge \Sigma_4 \wedge \Sigma_5]$ can be decomposed as

$$Pr[\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3 \wedge \Sigma_4 \wedge \Sigma_5] = Pr[\Sigma_1]Pr[\Sigma_2|\Sigma_1]Pr[\Sigma_3|\Sigma_1 \wedge \Sigma_2]Pr[\Sigma_4|\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3]Pr[\Sigma_5|\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3 \wedge \Sigma_4] \quad (1)$$

Claim 1. The probability that algorithm $\mathcal{C}$ does not abort as a result of $\mathcal{A}$'s key extraction queries is at least $(1 - 1/(q_E + 1))^{q_E}$. Hence we have $Pr[\Sigma_1] \geq (1 - 1/(q_E + 1))^{q_E}$.

Proof. As $Pr[c = 0] = 1/(q_E + 1)$, for a key extraction query, the probability that $\mathcal{C}$ does not abort is $1 - 1/(q_E + 1)$. Since $\mathcal{A}$ makes at most q_E queries to the key extraction oracle, the probability that algorithm $\mathcal{C}$ does not abort as a result of's key extraction queries is at least $(1 - 1/(q_E + 1))^{q_E}$. □

Claim 2. The probability that algorithm $\mathcal{C}$ does not abort as a result of $\mathcal{A}$'s standard signature queries is at least $1 - q_S(q_{H_2} + q_S)2^{-k}$. Thus there hold $Pr[\Sigma_2|\Sigma_1] \geq 1 - q_S(q_{H_2} + q_S)2^{-k}$.

Proof. As the list L_2 never contains more than $q_{H_2} + q_S$ entries, the probability of $\mathcal{C}$ to fail in handling a signing query because of a conflict on is at most $q_S(q_{H_2} + q_S)2^{-k}$. And events Σ_1 and Σ_2 are independent, so $Pr[\Sigma_2|\Sigma_1] \geq 1 - q_s(q_{H_2} + q_s)2^{-k}$. □

Claim 3. The probability that algorithm $\mathcal{C}$ does not abort as a result of $\mathcal{A}$'s proxy signature queries is at least $1/2\left(1 - q_{\mathcal{PS}}(q_{H_3} + q_{\mathcal{PS}})2^{-k}\right)$. Thus there hold $Pr[\Sigma_3|\Sigma_1 \wedge \Sigma_2] \geq 1/2\left(1 - q_{\mathcal{PS}}(q_{H_3} + q_{\mathcal{PS}})2^{-k}\right)$.

Proof is similar to Claim 2.

Claim 4. If algorithm $\mathcal{C}$ does not abort as a result of's signature queries and key extraction queries then algorithm's view is identical to its view in the real attack. Hence,

$$Pr[\Sigma_4|\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3] \geq \varepsilon.$$

Claim 5. The probability that algorithm $\mathcal{C}$ does not abort after $\mathcal{A}$ outputting a valid and nontrivial forgery is at least $\frac{1}{2}(1-\frac{1}{q_E+1})\frac{1}{q_E+1}$. Hence

$$Pr[\Sigma_5|\Sigma_1 \wedge \Sigma_2 \wedge \Sigma_3 \wedge \Sigma_4] \geq \frac{1}{2}(1-\frac{1}{q_E+1})\frac{1}{q_E+1}.$$

Proof. After $\mathcal{A}$ outputs a valid and nontrivial forgery, algorithm $\mathcal{C}$ does not abort if only if $c_1 = 0$, $c_i = 1$ and $s^* = 0$. So the conclusion is correct. □

According to the equation (1), algorithm $\mathcal{C}$ produces the correct answer with probability at least

$$\begin{aligned}
&\frac{1}{4}\left(1-q_{\mathcal{S}}(q_{H_2}+q_{\mathcal{S}})2^{-k}\right)\left(1-q_{\mathcal{PS}}(q_{H_3}+q_{\mathcal{PS}})2^{-k}\right)(1-\frac{1}{q_E+1})^{q_E+1}\frac{\varepsilon}{q_E+1}\\
\geq\ &\frac{1}{4}\left(1-q_{\mathcal{S}}(q_{H_2}+q_{\mathcal{S}})2^{-k}\right)\left(1-q_{\mathcal{PS}}(q_{H_3}+q_{\mathcal{PS}})2^{-k}\right)\frac{\varepsilon}{e(q_E+1)}\\
\geq\ &\varepsilon'
\end{aligned}$$

as required.

Algorithm $\mathcal{C}$'s running time is the same as $\mathcal{A}$'s running time plus the time to respond to $q_{H_1}+q_{H_2}+q_{H_3}+q_{H_4}+q_{\mathcal{S}}+q_{\mathcal{PS}}$ hash queries, q_E key extraction queries and $q_{\mathcal{S}}+q_{\mathcal{PS}}$ signature queries, and the time to transform's final forgery into the CDH solution. Hence, the total running time is at most $t + C_G(q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 5q_{\mathcal{S}} + 7q_{\mathcal{PS}} + 4) \leq t'$ as required. This completes the proof of Theorem. □

6 Conclusion

ID-based public key cryptosystem can be an alternative for certificate-based public key infrastructures. In this paper we formalized a notion of security for ID-based proxy signature scheme and proposed an scheme from bilinear pairings. The security of our scheme is tightly related to Computational Diffie-Hellman (CDH) problem in the Random Oracle model. Furthermore, we showed that optimal security reductions are also achievable for ID-based proxy signatures.

References

1. M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS), 48C57. ACM, 1996.

2. J. Herranz and G. Sez. Verifiable secret sharing for general access structures, with application to fully distributed proxy signatures. In Proceedings of Financial Cryptography 2003, LNCS. Springer-Verlag, 2003.
3. S. Lal and A. K. Awasthi. Proxy blind signature scheme. Cryptology ePrint Archive, Report 2003/072. Available at http://eprint.iacr.org/, 2003.
4. S. Lal and A. K. Awasthi. A scheme for obtaining a warrant message from the digital proxy signatures. Cryptology ePrint Archive, Report 2003/073. Available at http://eprint.iacr.org/, 2003.
5. H.-U. Park and L.-Y. Lee. A digital nominative proxy signature scheme for mobile communications. In ICICS 2001, volume 2229 of LNCS, 451C455. Springer-Verlag, 2001.
6. H. Kim, J. Baek, B. Lee, and K. Kim. Secret computation with secrets for mobile agent using one-time proxy signature. In Cryptography and Information Security 2001, 2001.
7. K. Shum and V.-K. Wei. A strong proxy signature scheme with proxy signer privacy protection. In Eleventh IEEE International Workshop on Enabling Technologies: Infrastucture for Collaborative Enterprises , 2002.
8. 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.
9. T.Malkin, S.Obana and M.Yung. The Hierarchy of Key Evolving Signatures and a Characterization of Proxy Signatures. Eurocrypt 2004, LNCS 3027, 306-322, Springer-Verlag, 2004.
10. A. Shamir, Identity-based cryptosystems and signature schemes, Advances in Cryptology-Crypto 1984, LNCS 196, 47-53, Springer-Verlag, 1984.
11. D. Boneh and M. Franklin, Identity-based encryption from the Weil pairing, Advances in Cryptology-Crypto 2001, LNCS 2139, 213-229, Springer-Verlag, 2001.
12. D. Boneh, B. Lynn, and H. Shacham, Short signatures from the Weil pairing, Advances in Cryptology-Asiacrypt 2001, LNCS 2248, 514-532, Springer-Verlag, 2001.
13. A. Joux, The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems, ANTS 2002, LNCS 2369, 20-32, Springer-Verlag, 2002.
14. D.Boneh and X.Boyen, Short Signatures Without Random Oracles. Eurocrypt 2004, LNCS 3027, 56-73, Springer-Verlag, 2004.
15. M.Bellare, C.Namprempre and G.Neven. Security Proofs for Identity-Based Identification and Signature Schemes, Advances in Cryptology-Eurocrypt 2004,LNCS 3027,268-286,Springer-Verlag, 2004.
16. B.Libert, and J.J.Quisquater. The Exact Security of an Identity Based Signature and Its Applications. Available from http://eprint.iacr.org/2004/102.