An Efficient ID-based Proxy Signature Scheme from Pairings [⋆]

Chunxiang Gu and Yuefei Zhu

Network Engineering Department Information Engineering University,
Zhengzhou, 450002, P.R.China
gcxiang5209@yahoo.com.cn

Abstract. This paper proposes a new ID-based proxy signature scheme based on the bilinear pairings. The number of paring operation involved in the verification procedure of our scheme is only one, so our scheme is more efficient comparatively. The new scheme can be proved secure with the hardness assumption of the k-Bilinear Diffie-Hellman Inverse problem, in the random oracle model.
Keywords: ID-based cryptography, proxy signatures, bilinear pairings.

1 Introduction

In 1984, Shamir [1] first proposed the idea of ID-based public key cryptography (ID-PKC) to simplify key management procedure of traditional certificate-based PKI. In ID-PKC, an entity's public key is directly derived from certain aspects of its identity, such as an IP address belonging to a network host or an e-mail address associated with a user. Private keys are generated for entities by a trusted third party called a private key generator (PKG). The direct derivation of public keys in ID-PKC eliminates the need for certificates and some of the problems associated with them. Recently, due to the contribution of D.Boneh et.al [2], a rapid development of ID-PKC has taken place. Using bilinear pairings, people proposed many new ID-based signature schemes [3–5]. With these ID-based signature schemes, a lot of new extensions, such as ID-based proxy signature scheme, ID-based ring signature scheme, etc.[6, 7], have also been proposed.

A proxy signature scheme allows one entity, called *original signer*, to delegate her signing capability to one or more entities, called *proxy signers*. Then the proxy signer can generate *proxy signatures*, which are signatures of some messages on behalf of the original signer. Upon receiving a proxy signature, a verifier can validate its correctness by the given verification procedure, and then is convinced of the original signer's agreement on the signed message.

Since Mambo, Usuda and Okamoto [8] first introduced the proxy signature scheme, many new constructions have been proposed. Based on the delegation type, proxy signatures can be classified as *full delegation*, *partial delegation* and *delegation by warrant*. In [9], Kim et al provided a new type of delegation called partial delegation with warrant, which can be considered as the combination of

⋆ Research supported by Found 973 (No. G1999035804), NSFC (No. 60473021).

partial delegation and delegation by warrant. Depending on whether the original signer can generate the same proxy signatures as the proxy signers do, there are two kinds of proxy signature schemes: *proxy-unprotected* and *proxy-protected*. In practice, the *proxy-protected partial delegation by warrant* schemes have attracted much more investigations than others, because they clearly distinguish the rights and responsibilities between the original signer and the proxy signer. In this paper, we will also focus on this kind of schemes. In fact, for simplicity, this special kind of schemes is often called as proxy signature scheme.

In [6], Zhang and Kim provided an ID-based proxy signature scheme based on pairings. The scheme is similar to Kim et al.'s scheme [9] which is based on certificate-based public key setting. There are no security proof in their original work. Later, Gu and Zhu [10] gave a formal security model for ID-based proxy signature schemes and provided a security proof for the scheme of Zhang and Kim in the random oracle model.

In this paper, we provide a more efficient ID-based proxy signature scheme from pairings. The new scheme can be proved secure in the random oracle model. The rest of this paper is organized as follows: In Section 2, we recall some preliminary works. In Section 3, we present a new ID-based proxy signature scheme with a correctness and efficiency analysis. In Section 4, we offer a formal security proof in the random orale model. Finally, we conclude in Section 5.

2 Preliminaries

2.1 Bilinear Pairings

Let $(G_1, +)$ and $(G_2, \cdot)$ be two cyclic groups of prime order q. $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a map which satisfies the following properties.

1. Bilinear: $\forall P, Q \in G_1, \forall \alpha, \beta \in Z_q, \hat{e}(\alpha P, \beta Q) = \hat{e}(P, Q)^{\alpha\beta}$;
2. Non-degenerate: If P is a generator of G_1, then $\hat{e}(P, P)$ is a generator of G_2;
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

Such an bilinear map is called an *admissible bilinear pairing* [2]. The Weil pairings and the Tate pairings of elliptic curves can be used to construct efficient admissible bilinear pairings.

We review a complexity problem related to bilinear pairings: the Bilinear Diffie-Hellman Inverse (BDHI) problem [11]. Let P be a generator of G_1, and $a \in Z_q^*$.

– k-**BDHI problem**: given $(P, aP, a^2P, ...a^kP) \in (G_1^*)^{k+1}$, output $\hat{e}(P, P)^{a^{-1}}$. An algorithm $\mathcal{A}$ solves k-BDHI problem with the probability ε if

$$Pr[\mathcal{A}(P, aP, a^2P, ...a^kP) = \hat{e}(P, P)^{a^{-1}}] \geq \varepsilon,$$

where the probability is over the random choice of generator $P \in G_1^*$, the random choice of $a \in Z_q^*$ and random coins consumed by $\mathcal{A}$.

We assume through this paper that the k-BDHI problem is intractable, which means that there is no polynomial time algorithm to solve k-BDHI problem with non-negligible probability.

2.2 ID-based Proxy Signatures

In this paper, if there is no special statement, let A be the original signer with identity ID_A and private key d_A. He delegates his signing rights to a proxy signer B with identity ID_B and private key d_B. A warrant is used to delegate signing right. In [10], Gu and Zhu gave a formal security model for ID-based proxy signature schemes.

Definition 1. *[10] An ID-based proxy signature scheme is specified by eight polynomial-time algorithms with the following functionalities.*

- **Setup:** *The parameters generation algorithm, takes as input a security parameter $k \in N$ (given as 1^k), and returns a master secret key s and system parameters Ω. This algorithm is performed by PKG.*
- **Extract:** *The private key generation algorithm, takes as input an identity $ID_U \in \{0,1\}^*$, and outputs the secret key d_U corresponding to ID_U. PKG uses this algorithm to extract the users' secret keys.*
- **Delegate:** *The proxy-designation algorithm, takes as input A's secret key d_A and a warrant m_ω, and outputs the delegation $W_{A \to B}$.*
- **DVerify:** *The designation-verification algorithm, takes as input ID_A, $W_{A \to B}$ and verifies whether $W_{A \to B}$ is a valid delegation come from A.*
- **PKgen:** *The proxy key generation algorithm, takes as input $W_{A \to B}$ and some other secret information z (for example, the secret key of the executor), and outputs a signing key d_p for proxy signature.*
- **PSign:** *The proxy signing algorithm, takes as input a proxy signing key d_p and a message $m \in \{0,1\}^*$, and outputs a proxy signature (m, δ).*
- **PVerify:** *The proxy verification algorithm, takes as input ID_A, ID_B and a proxy signature (m, δ), and outputs 0 or 1. In the later case, (m, δ) is a valid proxy signature of A.*
- **ID:** *The proxy identification algorithm, takes as input a valid proxy signature (m, δ), and outputs the identity ID_B of the proxy signer.*

An ID-based proxy signature scheme should first be correct. That is, $\forall\, m, \omega \in \{0,1\}^*$, it should have the following properties:

1. $DVerify(Delegate(\omega, D_A), ID_A) = 1$
2. For $W_{A \to B} = Delegate(\omega, D_A)$, let $D_P \leftarrow PKgen(W_{A \to B}, D_B)$, then $PVerify(PSign(m, D_P), ID_A, ID_B) = 1$, and $ID(PSign(m, D_P)) = ID_B$.

We consider an adversary $\mathcal{A}$ which is assumed to be a probabilistic Turing machine which takes as input the global scheme parameters and a random tape.

Definition 2. *[10] For an ID-based proxy signature scheme ID_PS. We define an experiment $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ of adversary $\mathcal{A}$ and security parameter k as follows:*

1. *A challenger $\mathcal{C}$ runs **Setup** and gives the system parameters Ω to $\mathcal{A}$.*
2. $C_{list} \leftarrow \phi$, $D_{list} \leftarrow \phi$, $G_{list} \leftarrow \phi$, $S_{list} \leftarrow \phi$. *(ϕ means NULL.)*
3. *Adversary $\mathcal{A}$ can make the following requests or queries adaptively.*
   - *Extract(.): This oracle takes as input a user's ID_i, and returns the corresponding private key d_i. If $\mathcal{A}$ gets $d_i \leftarrow Extract(ID_i)$, let $C_{list} \leftarrow C_{list} \cup \{(ID_i, d_i)\}$.*
   - *Delegate(.): This oracle takes as input the designator's identity ID and a warrant m_ω, and outputs a delegation W. If $\mathcal{A}$ gets $W \leftarrow Delegate(ID, m_\omega)$, let $D_{list} \leftarrow D_{list} \cup \{(ID, m_\omega, W)\}$.*
   - *PKgen(.): This oracle takes as input the proxy signer's ID and a delegation W, and outputs a proxy signing key d_p. If $\mathcal{A}$ gets $d_p \leftarrow PKgen(ID, W)$, let $G_{list} \leftarrow G_{list} \cup \{(ID, W, d_p)\}$.*
   - *PSign(.): This oracle takes as input the delegation W and message $m \in \{0,1\}^*$, and outputs a proxy signature created by the proxy signer. If $\mathcal{A}$ gets $(m, \tau) \leftarrow PSign(W, m)$, let $S_{list} \leftarrow S_{list} \cup \{(W, m, \tau)\}$.*
4. *$\mathcal{A}$ outputs (ID, m_ω, W) or (W, m, τ).*
5. *If $\mathcal{A}$'s output satisfies one of the following terms, $\mathcal{A}$'s attack is successful.*
   - *The output is (ID, m_ω, W), and satisfies: $DVerify(W, ID) = 1$, $(ID, .) \notin C_{list}$, $(ID, ., .) \notin G_{list}$ and $(ID, m_\omega, .) \notin D_{list}$. $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ returns 1.*
   - *The output is (W, m, τ), and satisfies $PVerify((m, \tau), ID_i) = 1$, $(W, m, .) \notin S_{list}$, and $(ID_j, .) \notin C_{list}$, $(ID_j, W, .) \notin G_{list}$, where ID_i and ID_j are the identities of the designator and the proxy singer defined by W, respectively. $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ returns 2.*

   Otherwise, $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ returns 0.

Definition 3. *[10] An ID-based proxy digital signature scheme ID_PS is said to be existential delegation and signature unforgeable under adaptive chosen message and ID attacks (**DS-EUF-ACMIA**), if for any polynomial time adversary $\mathcal{A}$, any polynomial $p(.)$ and big enough k,*

$$Pr[Exp_{\mathcal{A}}^{ID\text{-}PS}(k) = 1] < \frac{1}{p(k)} \quad and \quad Pr[Exp_{\mathcal{A}}^{ID\text{-}PS}(k) = 2] < \frac{1}{p(k)}$$

3 A New Efficient ID-based Proxy Signature Scheme

In this section, we present a new efficient ID-based proxy signature scheme. Our scheme is based on a variation of the ID-based signature scheme proposed by Barreto et.al [5] in Asiacrypt'05. The method for obtaining private keys from identities is a simplification of a method suggested by Sakai and Kasahara [12]. This leads to a more efficient performance.

3.1 Description of the Scheme

The new scheme can be described as follows:

- **Setup**: Takes as input a security parameter k, and returns a master key s and system parameters $\Omega = (G_1, G_2, q, \hat{e}, P, P_s, P_{ss}, g, g_s, H_1, H_2)$, where $(G_1, +)$ and $(G_2, \cdot)$ are two cyclic groups of order q, $\hat{e} : G_1 \times G_1 \to G_2$ is an admissible bilinear map, $P_s = sP$, $P_{ss} = s^2P$, $g = \hat{e}(P, P)$, $g_s = \hat{e}(P_s, P)$, $H_1 : \{0,1\}^* \to Z_q^*$ and $H_2 : \{0,1\}^* \times G_1 \to Z_q$ are hash functions.
- **Extract**: Takes as input an identity $ID_X \in \{0,1\}^*$, computers $D_X = (H_1(ID_X) + s)^{-1}P$, and lets D_X be the user's secret key.
- **Delegate**: Takes as input the secret key D_A, the proxy signer's identity ID_B and a warrant m_ω, selects a random $x \in Z_q^*$, computes $q_B = H_1(ID_B)$, $r_A = g_s^x \cdot g^{q_B x}$, $h_A = H_2(m_\omega, r_A)$, $V_A = (x + h_A)D_A$, and outputs the delegation $W_{A \to B} = (m_\omega, r_A, V_A)$.
- **DVerify**: Once B receives $W_{A \to B} = (m_\omega, r_A, V_A)$, he computes $h_A = H_2(m_\omega, r_A)$, $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$, and accepts the delegation only if
$$\hat{e}((q_A + q_B)P_s + q_A q_B P + P_{ss}, V_A) = r_A \cdot g_s^{h_A} \cdot g^{q_B h_A}.$$
- **PKgen**: If B accepts the delegation $W_{A \to B} = (m_\omega, r_A, V_A)$, he computes the proxy signing key D_P as $D_P = h_A \cdot D_B - V_A$, where $h_A = H_2(m_\omega, r_A)$.
- **PSign**: The proxy signer can pre-computing $\xi = g^{h_A(q_A - q_B)}/r_A$, where $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$ and r_A is from $W_{A \to B}$. Let D_P be the proxy signing key, for a message m, the proxy signer chooses $y \in Z_q^*$ at random and computes $r_P = \xi^y$, $h_P = H_2(m, r_P)$, $V_P = (y + h_P)D_P$, and lets $(m, \tau) = (m, r_P, V_P, m_\omega, r_A)$ be the proxy signature for m.
- **PVerify**: For a proxy signature $(m, r_P, V_P, m_\omega, r_A)$, a recipient first checks if the proxy signer and the message conform to m_ω. Then he computes $h_P = H_2(m, r_P)$, $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$ and verifies whether
$$\hat{e}((q_A + q_B)P_s + q_A q_B P + P_{ss}, V_P) = r_P \cdot g^{h_A h_P (q_A - q_B)} \cdot r_A^{-h_P}.$$
If both steps succeed, the proxy signature on behalf of A is valid.
- **ID**: The proxy signer's identity ID_B can be revealed by m_ω.

3.2 Correctness and Efficiency

Set $Q = (q_A + q_B)P_s + q_A q_B P + P_{ss}$. Consistency of the scheme is easily proved as follows: For any $m_\omega \in \{0,1\}^*$, $Delegate(m_\omega, D_A) = (m_\omega, r_A, V_A)$, $h_A = H_2(m_\omega, r_A)$. Then,

$$\begin{aligned}\hat{e}(Q, V_A) &= \hat{e}((q_A + s)(P_s + q_B P), (x + h_A)(q_A + s)^{-1}P) \\ &= \hat{e}(P_s + q_B P, (x + h_A)P) \\ &= r_A \cdot (g_s \cdot g^{q_B})^{h_A}\end{aligned}$$

That is, $DVerify(Delegate(m_\omega, D_A), ID_A) = 1$. On the other hand,

$$D_P = PKgen((m_\omega, r_A, V_A), D_B) = h_A \cdot D_B - V_A = \frac{h_A(q_A - q_B) - x(s + q_B)}{(s + q_A)(s + q_B)}P.$$

For any $m \in \{0,1\}^*$, $PSign(m, D_P) = (m, r_P, V_P, m_\omega, r_A)$, $h_P = H_2(m_\omega, r_P)$. Then,

$$
\begin{aligned}
\hat{e}(Q, V_P) &= \hat{e}((q_A + s)(s + q_B)P, (y + h_P)\frac{h_A(q_A - q_B) - x(s + q_B)}{(s + q_A)(s + q_B)}P) \\
&= \hat{e}(P, (h_A(q_A - q_B) - x(s + q_B))P)^{(y+h_P)} \\
&= (g^{h_A(q_A-q_B)}/g^{x(s+q_B)})^{(y+h_P)} \\
&= (g^{h_A(q_A-q_B)}/r_A)^{(y+h_P)} \\
&= r_P \cdot (g^{h_A(q_A-q_B)}/r_A)^{h_P}
\end{aligned}
$$

Hence, $PVerify(PSign(m, D_P), ID_A, ID_B) = 1$. m_ω designates the identity of the proxy signer, and it is a part of the signature. So it is easy to see that $ID(PSign(m, D_P)) = ID_B$.

Denote by M an ordinary scalar multiplication in $(G_1, +)$, by E an Exp. operation in $(G_2, .)$, and by $\hat{e}$ a computation of the pairing. The hash function maps an identity to an element in G_1 used by the scheme in [6] usually requires a "Maptopoint operation" [2]. As discussed in [2], Maptopoint operation (denoted by H) is so inefficient that we can't neglect it. Do not take other operations into account. We compare our new scheme to the ID-based proxy signature scheme of Zhang and Kim [6] in the following table.

schemes	Delgate	DVerify	PKgen	PSign	PVerify
Zhang-Kim [6]	$2M + 1E$	$2\hat{e} + 1E + 1H$	$1M$	$2M + 1E$	$2\hat{e} + 2E + 2H$
proposed	$1M + 2E$	$1\hat{e} + 2M + 2E$	$1M$	$1M + 1E$	$1\hat{e} + 2M + 2E$

Note: The hash function used in our scheme which maps an identity to an element in Z_q^* is so efficient that we usually can neglect it.

Some general performance enhancements can be applied to our schemes. For pre-selected $P \in G_1$ and $\mu \in G_2$, there are efficient algorithms [13] to compute kP and μ^l for random $k, l \in Z_q$ by pre-computing and storing. In our scheme, P, P_s and g, g_s are fixed system parameters. Secret keyes of the signer and the proxy signer are also fixed for themselves.

4 Security Proof

In this section, we reduce the security of our scheme to the hardness assumption of k-BDHI problem in the random oracle model.

Assume there is an adversary $\mathcal{F}_0$ who can breaks the ID-based proxy signature scheme. We will construct a polynomial time algorithm $\mathcal{F}_1$ that, by simulating the challenger and interacting with $\mathcal{F}_1$, solves $(n_1 + 1)$-BDHI problem, where n_1 is the number of queries that $\mathcal{F}_0$ can ask to the random oracle $H_1(.)$.

Lemma 1. *Given system parameters $\Omega = (G_1, G_2, q, \hat{e}, P, P_s, P_{ss}, g, g_s, H_1, H_2)$ and identities $ID_A, ID_B \in \{0,1\}^*$, let $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$, $D_A =$*

$(s+q_A)^{-1}P$, and $T=(q_A+q_B)P_s+q_Aq_BP+P_{ss}$, the following distributions are the same.

$$\delta=\left\{(r,h,V)\;\middle|\;\begin{array}{c}x\in_R Z_q^*\\ h\in_R Z_q\\ r=g_s^x\cdot g^{xq_B}\\ V=(x+h)D_A\end{array}\right\}\text{ and }\delta'=\left\{(r,h,V)\;\middle|\;\begin{array}{c}V\in_R G_1\\ h\in_R Z_q\\ r=\hat{e}(T,V)\cdot g_s^{-h}\cdot g^{-q_Bh}\\ r\neq 1\end{array}\right\}$$

Proof: First we choose a triple (α,β,γ) such that $\alpha\in G_2^*$,$\beta\in Z_q$,$\gamma\in G_1$ and satisfying $\alpha=\hat{e}(T,\gamma)\cdot g_s^{-\beta}\cdot g^{-\beta q_B}$. We then compute the probability of appearance of this triple following each distribution of probabilities:

$$Pr_\delta[(r,h,V)=(\alpha,\beta,\gamma)]=\Pr_{x\neq 0}\begin{bmatrix}\alpha=g_s^x\cdot g^{xq_B}\\ h=\beta\\ (x+h)D_A=\gamma\end{bmatrix}=\frac{1}{q(q-1)}.$$

$$Pr_{\delta'}[(r,h,V)=(\alpha,\beta,\gamma)]=\Pr_{r\neq 1}\begin{bmatrix}h=\beta\\ V=\gamma\\ \alpha=r=\hat{e}(T,V)\cdot g_s^{-h}\cdot g^{-q_Bh}\end{bmatrix}=\frac{1}{q(q-1)}.$$

Hence, we can simulate the *Delegate*(.) oracle for input (ID_A,ID_B,m_ω) without the secret key D_A indistinguishably from the real one as following:

- $\mathcal{S}_\mathcal{D}(ID_A,ID_B,m_\omega)$:
  - Pick randomly $V_A\in G_1$, $h_A\in Z_q$.
  - Compute $r_A=\hat{e}((q_A+q_B)P_s+q_Aq_BP+P_{ss},V_A)\cdot g_s^{-h_A}\cdot g^{-q_Bh_A}$, where $q_A=H_1(ID_A)$, $q_B=H_1(ID_B)$,
  - If $H_2(m_\omega,r_A)$ has been defined, then abort (a collision appears). Otherwise, set $H_2(m_\omega,r_A)=h_A$.
  - Set $W=(m_\omega,r_A,V_A)$.

Lemma 2. *Given system parameters $\Omega=(G_1,G_2,q,\hat{e},P,P_s,P_{ss},g,g_s,H_1,H_2)$, identities $ID_A,ID_B\in\{0,1\}^*$ and $W_{A\to B}=(m_\omega,r_A,V_A)$, let $q_A=H_1(ID_A)$, $q_B=H_1(ID_B)$, $h_A=H_2(m_\omega,r_A)$, $D_B=(s+q_B)^{-1}P$, $D_P=h_AD_B-V_A$, $\xi=g^{h_A(q_A-q_B)}/r_A$ and $T=(q_A+q_B)P_s+q_Aq_BP+P_{ss}$, the following distributions are the same.*

$$\delta=\left\{(r_P,h_P,V_P)\;\middle|\;\begin{array}{c}y\in_R Z_q^*\\ h_P\in_R Z_q\\ r_P=\xi^y\\ V_P=(y+h_P)D_P\end{array}\right\}\quad\text{and}$$

$$\delta'=\left\{(r_P,h_P,V_P)\;\middle|\;\begin{array}{c}V_P\in_R G_1\\ h_P\in_R Z_q\\ r_P=\hat{e}(T,V_P)\cdot(g^{-h_A(q_A-q_B)}\cdot r_A)^{h_P}\\ r\neq 1\end{array}\right\}$$

Proof: Readers can see that the proof is almost the same as that of Lemma 1. We omit it in this paper.

That is, we can simulate the *PSign*(.) oracle for input $(W_{A\to B}=(m_\omega,r_A,V_A),m)$ without the secret proxy key D_P indistinguishably from the real one as following:

– $\mathcal{S}_{\mathcal{PS}}(W_{A\to B}, m)$:
  - Pick randomly $V_P \in G_1$, $h_P \in Z_q$.
  - Check whether $H_2(m_\omega, r_A)$ is defined. If not, request oracle $H_2(.)$ with (m_ω, r_A). Let $H_2(m_\omega, r_A) = e$.
  - Compute $r_P = \hat{e}((q_A + q_B)P_s + q_A q_B P + P_{ss}, V_P) \cdot (g^{-e(q_A - q_B)} \cdot r_A)^{h_P}$, where $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$.
  - If $H_2(m, r_P)$ has been defined, then abort(a collision appears). Otherwise, set $H_2(m, r_P) = h_P$.
  - Let $(m, \tau) = (m, r_P, V_P, m_\omega, r_A)$ be the reply.

Theorem 1. *In the random oracle mode, let $\mathcal{F}_0$ be a polynomial-time adversary who manages an $Exp_A^{ID\text{-}PS}(k)$ within a time bound $T(k)$, and gets return 1 or 2 by un-negligible probability $\varepsilon(k)$. We denote respectively by n_1,n_2 and n_3 the number of queries that $\mathcal{F}_0$ can ask to the random oracle $H_1(.)$, $H_2(.)$ and the proxy singing oracle $PSign(.)$. Assume that $\varepsilon(k) \geq 10(n_3+1)(n_2+n_3)n_1/q$, then there is an adversary $\mathcal{F}_1$ who can solve (n_1+1)-BDHI problem within expected time less than $120686 \cdot n_2 \cdot n_1 \cdot T(k)/\varepsilon(k)$.*

Proof: Without any loss of generality, we may assume that for any ID, $\mathcal{F}_0$ queries $H_1(.)$ with ID before ID is used as (part of) an input of any query to $Extract(.)$, $Delegate(.)$, $PKgen(.)$ and $PSign(.)$, by using a simple wrapper of $\mathcal{F}_0$.

$\mathcal{F}_1$ is given input parameters of pairing $(q, G_1, G_2, \hat{e})$ and a random instance $(P, aP, a^2P, ..., a^{n_1}P, a^{n_1+1}P)$ of the (n_1+1)-BDHI problem, where P is random in G_1^* and a is a random in Z_q^*. $\mathcal{F}_1$ simulates the challenger and interacts with $\mathcal{F}_0$ as follows:

1. $\mathcal{F}_1$ randomly chooses different $h_0, h_1, ...h_{n_1-1} \in Z_q^*$, and computes $f(x) = \prod_{i=1}^{n_1-1}(x + h_i) = \sum_{i=0}^{n_1-1} c_i x^i$.
2. $\mathcal{F}_1$ computes $Q = \sum_{i=0}^{n_1-1} c_i a^i P = f(a)P$, $aQ = \sum_{i=0}^{n_1-1} c_i a^{i+1} P$, $a^2Q = \sum_{i=0}^{n_1-1} c_i a^{i+2} P$, and $Q' = \sum_{i=1}^{n_1-1} c_i a^{i-1} P$. In the (unlikely) situation where $Q = 1_{G_1}$, there exists an $h_i = -a$, hence, $\mathcal{F}_1$ can solve the (n_1+1)-BDHI problem directly and abort.
3. $\mathcal{F}_1$ computes $f_i(x) = f(x)/(x+h_i) = \sum_{j=0}^{n_1-2} d_j x^j$. Obviously, $(a+h_i)^{-1}Q = (a+h_i)^{-1}f(a)P = f_i(a)P = \sum_{j=0}^{n_1-2} d_j a^j P$ for $1 \leq i \leq n_1$.
4. $\mathcal{F}_1$ randomly chooses an index t with $1 \leq t \leq n_1$, sets $v = 0$.
5. $\mathcal{F}_1$ computes $g = \hat{e}(Q, Q)$, $g_s = \hat{e}(aQ, Q)$, sets the system parameters $\Omega = (G_1, G_2, q, \hat{e}, Q, aQ, a^2Q, g, g_s, H_1, H_2)$, where H_1, H_2 are random oracles controlled by $\mathcal{F}_1$.
6. $\mathcal{F}_1$ sets $C_{list} = \phi$, $D_{list} = \phi$, $G_{list} = \phi$, $S_{list} = \phi$, and starts $Exp_{\mathcal{F}_0}^{ID\text{-}PS}(k)$ by giving $\mathcal{F}_0$ the system parameters Ω. During the execution, $\mathcal{F}_1$ emulates $\mathcal{F}_0$'s oracles as follows:
   – $H_1(.)$: $\mathcal{F}_1$ maintains a H_1_list, initially empty. For a query ID, if ID already appears on the H_1_list in a tuple (ID, l, D), $\mathcal{F}_1$ responds with l. Otherwise, sets $v = v + 1$, $ID_v = ID$, if $v = t$, $\mathcal{F}_1$ sets $l_v = h_0$, $D_v = \perp$; otherwise, $\mathcal{F}_1$ selects a random $n_1 \geq \vartheta > 0$ which has not been chosen and sets $l_v = h_\vartheta + h_0$, $D_v = (a + h_\vartheta)^{-1}Q$. In both case, adds the tuple (ID_v, l_v, D_v) to H_1_list and responds with l_v.

- $H_2(.)$: For a query (m, r), $\mathcal{F}_1$ checks if $H_2(m, r)$ is defined. If not, $\mathcal{F}_1$ picks a random $c \in Z_q^*$ and defines $H_2(m, r) = c$. $\mathcal{F}_1$ returns $H_2(m, r)$ to $\mathcal{F}_0$.
- $Extract(.)$: For input ID_i, $\mathcal{F}_1$ searches in H_1_list for (ID_i, l_i, D_i). If $D_i = \perp$ then $\mathcal{F}_1$ aborts. Otherwise, $\mathcal{F}_1$ responds with D_i. Set $C_{list} \leftarrow C_{list} \cup \{(ID_i, D_i)\}$.
- $Delegate(.)$: For input ID_i and warrant m_ω (we assume the identity of the proxy signer's is ID_j), if $i \neq t$, $\mathcal{F}_1$ computes $W = Delegate(D_i, m_\omega)$. Otherwise, $\mathcal{F}_1$ runs the simulator $\mathcal{S}_{\mathcal{D}}(ID_t, ID_j, m_\omega)$ and gets the reply W. Let W be the reply to $\mathcal{F}_0$, and set $D_{list} \leftarrow D_{list} \cup \{(ID_i, m_\omega, W)\}$.
- $PKgen(.)$: For input proxy signer's ID_j and delegation $W = (m_\omega, r_0, V_0)$, if $j = t$, then abort. Otherwise, $\mathcal{F}_1$ computes $D_P = H_2(m_\omega, r_0)D_j - V_0$ as the reply to $\mathcal{F}_0$. Let $G_{list} \leftarrow G_{list} \cup \{(W, ID_j, D_P)\}$.
- $PSign(.)$: For input $W = (m_\omega, r_0, V_0)$ and message m, designator's identity be ID_i and proxy signer's identity be ID_j. If $j \neq t$, $\mathcal{F}_1$ computes the proxy signature $\tau = (r_P, V_P, m_\omega, r_0)$ on m with secret signing key $D_P = H_2(m_\omega, r_0)D_j - V_0$, and return (m, τ) as the reply. Otherwise, $\mathcal{F}_1$ simulates ID_t's proxy signature on behalf of ID_i with the simulator $\mathcal{S}_{\mathcal{PS}}(W, m)$ and lets the output (m, τ) of $\mathcal{S}_{\mathcal{PS}}$ as the reply. Let $S_{list} \leftarrow S_{list} \cup \{(W, m, \tau)\}$.

7. $\mathcal{F}_1$ keeps interacting with $\mathcal{F}_0$ until $\mathcal{F}_0$ halts or aborts.
   - Case 0: If $\mathcal{F}_0$'s output is (ID^*, m_ω^*, W^*), where $W^* = (m_\omega^*, r_0^*, V_0^*)$, and satisfies: $DVerify(W^*, ID^*) = 1$, $(ID^*, .) \notin C_{list}$, $(ID^*, ., .) \notin G_{list}$ and $(ID^*, m_\omega^*, .) \notin D_{list}$, and $ID^* = ID_t$, $\mathcal{F}_1$ can get a delegation forgery $(m_\omega^*, r_0^*, h_0^*, V_0^*)$ corresponding to identity ID_t (whose secret key is $a^{-1}Q$), where $h_0^* = H_2(m_\omega^*, r_0^*)$. By replays of Step 6 with the same random tape but different choices of $H_2(.)$, as done in the Forking Lemma [14], $\mathcal{F}_1$ can get another valid forgery $(m_\omega^*, r_0^*, h_1^*, V_1^*)$ such that $h_1^* \neq h_0^*$. Set $statue = 0$.
   - Case 1: If $\mathcal{F}_0$'s output is $(W^*, m^*, \tau^*) = ((m_\omega^*, r_0^*, V_0^*), m^*, (r_P^*, V_P^*, m_\omega^*, r_0^*))$ with designator's identity ID_t and proxy signer's identity ID_j, and satisfies $PVerify((m^*, \tau^*), ID_t) = 1$, $(W^*, m^*, .) \notin S_{list}$, and $(ID_j, .) \notin C_{list}$, $(ID_j, W^*, .) \notin G_{list}$, $\mathcal{F}_1$ can get a forgery $(W^*, m^*, (r_P^*, h_P^*, V_P^*, m_\omega^*, r_0^*))$ corresponding to proxy signing key $D_P = \mu a^{-1}Q - V_0^*$, where $\mu = H_2(m_\omega^*, r_0^*)$ and $h_P^* = H_2(m, r_P^*)$. Define $H_2(m_\omega^*, r_0^*) = \mu$. By replays of Step 4 with the same random tape but different choices of $H_2(.)$, as done in the Forking Lemma [14], $\mathcal{F}_1$ can get another valid forgery $(W^*, m^*, (r_P^*, h_{P1}^*, V_{P1}^*, m_\omega^*, r_0^*))$ such that $h_{P1}^* \neq h_P^*$. Set $statue = 1$.
8. $\mathcal{F}_1$ can computes $a^{-1}Q$ as follows:
   - If $statue = 0$,
   $$a^{-1}Q = (h_1^* - h_0^*)^{-1}(V_1^* - V_0^*).$$
   - If $statue = 1$,
   $$a^{-1}Q = H_2(m_\omega^*, r_0^*)^{-1}((h_{P1}^* - h_P^*)^{-1}(V_{P1}^* - V_P^*) + V_0^*).$$

9. $\mathcal{F}_1$ computes $\hat{e}(Q, a^{-1}Q) = \hat{e}(Q,Q)^{a^{-1}}$. Then, $\mathcal{F}_1$ computes and outputs $\hat{e}(P,P)^{a^{-1}} = \hat{e}(Q,Q)^{a^{-1}}/\hat{e}(Q', Q + c_0P))^{c_0^{-2}}$ as the solution to the given instance of $(n_1 + 1)$-BDHI problem.

This completes the description of $\mathcal{F}_1$.

During $\mathcal{F}_1$'s execution, if $\mathcal{F}_0$ manages an $Exp_{\mathcal{F}_0}^{ID\text{-}PS}(k)$ and gets return 1 or 2, collisions appear with negligible probability, as mentioned in [14]. So $\mathcal{F}_1$'s simulations are indistinguishable from $\mathcal{F}_0$'s oracles. Because t is chosen randomly, $\mathcal{F}_1$ can get a forgery of $(m_\omega^*, r_0^*, h_0^*, V_0^*)$ corresponding to identity ID_t, or $(W^*, m^*, (r_P^*, h_P^*, V_P^*, m_\omega^*, r_0^*))$ corresponding to proxy signing key $D_P = \mu a^{-1}Q - V_0^*$, within expected time $T(k)$ with probability $\varepsilon(k)/n_1$.

In fact, the delegation and proxy signing are both schemes producing signatures of the form (m, r, h, V), where each of r, h, V corresponds to one of the three moves of a honest-verifier zero-knowledge protocol. By applying the **Forking lemma**[14], $\mathcal{F}_1$ can produce two valid forgery $(m_\omega^*, r_0^*, h_0^*, V_0^*)$ and $(m_\omega^*, r_0^*, h_1^*, V_1^*)$ such that $h_0^* \neq h_1^*$ within expected time less than $120686 \cdot n_2 \cdot n_1 \cdot \frac{T(k)}{\varepsilon(k)}$. So $\mathcal{F}_1$ can output $\hat{e}(P,P)^{a^{-1}}$. Thus we prove the theorem.

5 Conclusion

This paper presents an efficient and provably secure ID-based proxy signature scheme based on the bilinear pairings. Although fruitful achievements [15, 16] have been made in enhancing the computation of pairings, the computation of pairings are still a heavy burden for schemes from pairings. The number of paring operation involved in the verification procedure of our schemes is only one, so our scheme is more efficient comparetively. The scheme can be proved secure with the hardness assumption of the k-BDHI problem, in the random oracle model.

References

1. A. Shamir. Identity-based cryptosystems and signature schemes. In Advances in Cryptology - CRYPTO'84, LNCS 0196, pages 47-53. Springer-Verlag, 1984.
2. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, Advances in Cryptology- CRYPTO 2001, LNCS 2139, pages 213-229. Springer-Verlag, 2001.
3. J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In Y. Desmedt, editor, Public Key Cryptography - PKC 2003, volume 2567 of LNCS, pages 18-30. Springer-Verlag, 2002.
4. F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. Heys, editors, Selected Areas in Cryptography 9th Annual International Workshop, SAC 2002, volume 2595 of LNCS, pages 310-324. Springer-Verlag, 2003.
5. P. S. L. M. Barreto, B. Libert, N. McCullagh, J. Quisquater, Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In B. Roy, editor(s), Asiacrypt 2005, LNCS 3788, pages 515-532, Springer-Verlag, 2005.
6. F. Zhang and K. Kim, Efficient ID-based blind signature and proxy signature from bilinear pairings, ACISP 03, LNCS 2727, pages 312-323, Springer-Verlag, 2003.

7. F. Zhang, K. Kim, ID-based blind signature and ring signature from pairings. Asiacrypt'2002, LNCS 2501, pages 533–547, Springer-Verlag, 2002.
8. M. Mambo, K. Usuda, E. Okamoto. Proxy signatures for delegating signing operation. In: 3rd ACM Conference on Computer and Communications Security (CCS'96), pp. 48-57. New York: ACM Press, 1996.
9. S. Kim, S. Park, and D. Won, Proxy signatures, revisited, In Pro. of ICICS'97, LNCS 1334, pages 223-232, Springer-Verlag, 1997.
10. C. Gu, Y. Zhu. Provable Security of ID-based Proxy Signature Schemes. In Proc. ICCNMC'05, LNCS 3619, pages 1277-1286, Springer-Verlag, 2005.
11. D. Boneh, X. Boyen. Efficient Selective ID Secure Identity Based Encryption without Random Oracles. Advances In Cryptology-Eurocrypt 2004, LNCS 3027, pp. 223-238, Springer-Verlag, 2004.
12. R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054.
13. Y. Sakai, K. Sakurai. Efficient Scalar Multiplications on Elliptic Curves without Repeated Doublings and Their Practical Performance. ACISP 2000, LNCS 1841, pp. 59-73. Springer-Verlag 2000.
14. D.Pointcheval and J.Stern. Security arguments for digital signatures and blind signatures. Journal of Cryptology, 13(3):361-369,2000.
15. P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. Advances in Cryptology-Crypto'2002, LNCS 2442, pp. 354-368. Springer-Verlag, 2002.
16. I. Duursma and H. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p + x + d$. Advances in Cryptology-Asiacrypt'2003, LNCS 2894, pp. 111-123. Springer-Verlag, 2003.