Ann. Telecommun. (2011) 66:657–662
DOI 10.1007/s12243-011-0244-0

An ID-based proxy signature schemes without bilinear pairings

He Debiao · Chen Jianhua · Hu Jin

Received: 20 September 2010 / Accepted: 10 February 2011 / Published online: 27 February 2011

Abstract The proxy signature schemes allow proxy signers to sign messages on behalf of an original signer, a company, or an organization. Such schemes have been suggested for use in a number of applications, particularly in distributed computing, where delegation of rights is quite common. Many identity-based proxy signature schemes using bilinear pairings have been proposed. But the relative computation cost of the pairing is approximately twenty times higher than that of the scalar multiplication over elliptic curve group. In order to save the running time and the size of the signature, in this letter, we propose an identity-based signature scheme without bilinear pairings. With the running time being saved greatly, our scheme is more practical than the previous related schemes for practical application.

Keywords Digital signature · Identity-based proxy signature · Bilinear pairings · Elliptic curve

1 Introduction

In order to simplify the public-key authentication, Shamir [1] introduced the concept of identity-based (ID-based) cryptosystem problem. In this system, each user needs to register at a key generator center (KGC) with identity of himself before joining the network. Once a user is accepted, the KGC will generate a private key for the user and the user's identity (e.g., user's name or email address) becomes the corresponding public key. In this way, in order to verify a digital signature or send an encrypted message, a user only needs to know the "identity" of his communication partner and the public key of the KGC.

Mambo et al. [2] introduced the notion of proxy signature scheme. A proxy signature scheme allows an entity called original signer to delegate his signing capability to another entity, called proxy signer. Since it is proposed, the proxy signature schemes have been suggested for use in many applications, particularly in distributed computing where delegation of rights is quite common. In order to adapt different situations, many proxy signature variants are produced, such as one-time proxy signature, proxy blind signature, multi-proxy signature, and so on. Since the proxy signature appears, it attracts many researchers' great attention. Using bilinear pairings, people proposed many new ID-based signature schemes [3–5] and ID-based proxy signature (IBPS) scheme [6–10]. All the above IBPS schemes are very practical, but they are based on bilinear pairings and the pairing is regarded as the most expensive cryptography primitive. The relative computation cost of a pairing is approximately twenty times higher than that of the scalar multiplication over elliptic curve group [11]. Therefore, IBPS schemes without bilinear pairings would be more appealing in terms of efficiency.

In this paper, we present an IBPS scheme without pairings. The scheme rests on the elliptic curve discrete logarithm problem (ECDLP).With the pairing-free realization, the scheme's overhead is lower than that of previous schemes [6–10] in computation.

2 Preliminaries

2.1 Background of elliptic curve group

Let the symbol E/F_p denote an elliptic curve E over a prime finite field F_p, defined by an equation

H. Debiao (✉) · C. Jianhua · H. Jin
School of Mathematics and Statistics, Wuhan University,
Wuhan 430072, China
e-mail: hedebiao@163.com

$$y^2 = x^3 + ax + b, \quad a, b \in F_p \tag{1}$$

and with the discriminant

$$\Delta = 4a^3 + 27b^2 \neq 0. \tag{2}$$

The points on E/F_p together with an extra point O called the point at infinity form a group

$$\mathbb{G} = \{(x,y) : x, y \in F_p, E(x,y) = 0\} \cup \{O\}. \tag{3}$$

Let the order of $\mathbb{G}$ be n. $\mathbb{G}$ is a cyclic additive group under the point addition "+" defined as follows: Let P, $Q \in G$, l be the line containing P and Q (tangent line to E/F_p if $P = Q$), and R, the third point of intersection of l with E/F_p. Let l' be the line connecting R and O. Then P "+" Q is the point such that l' intersects E/F_p at R and O and P "+" Q. Scalar multiplication over E/F_p can be computed as follows:

$$tP = P + P + \cdots + P(t \text{ times}) \tag{4}$$

The following problems defined over $\mathbb{G}$ are assumed to be intractable within polynomial time.

Eliptic curve discrete logarithm problem: For $x \in_R Z_n^*$ and G the generator of $\mathbb{G}$, given $P = x \cdot G$ compute x.

2.2 ID-based proxy signatures

In this paper, unless stated otherwise, let O be the original signer with identity ID_O and private key D_O. He delegates his signing rights to a proxy signer A with identity ID_A and private key D_A. A warrant is used to delegate signing right. In [6], Gu and Zhu gave a formal security model for ID-based proxy signature schemes.

Definition 1 An ID-based proxy signature scheme is specified by the following polynomial-time algorithms with the following functionalities [6].

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

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 For an ID-based proxy signature scheme IBPS, we define an experiment $\mathrm{Exp}_F^{\mathrm{IBPS}}(k)$ of adversary $\mathcal{A}$ and security parameter k as follows [6]:

1. A challenger C runs Setup and gives the system parameters Ω to $\mathcal{A}$.
2. $C_{\mathrm{list}} \leftarrow \phi$, $D_{\mathrm{list}} \leftarrow \phi$, $G_{\mathrm{list}} \leftarrow \phi$, $S_{\mathrm{list}} \leftarrow \phi$. ($\phi$ means null.)
3. Adversary $\mathcal{A}$ can make the following requests or queries adaptively.
   - Extract(.): This oracle takes a user's ID_i as input, and returns the corresponding private key D_i. If $\mathcal{A}$ gets $D_i \leftarrow \mathrm{Extract}(\mathrm{ID}_i)$, let $C_{\mathrm{list}} \leftarrow C_{\mathrm{list}} \cup \{(\mathrm{ID}_i, D_i)\}$.
   - Delegate(.): This oracle takes the designator's identity ID and a warrant m_w as input, and outputs a delegation W. If $\mathcal{A}$ gets $W \leftarrow \mathrm{Delegate}(\mathrm{ID}, m_w)$, let $D_{\mathrm{list}} \leftarrow D_{\mathrm{list}} \cup \{(\mathrm{ID}, m_w, W)\}$.
   - PKgen(.): This oracle takes the proxy signer's ID and a delegation W as input, and outputs a proxy signing key D_p. If $\mathcal{A}$ gets $D_p \leftarrow \mathrm{PKgen}(\mathrm{ID}, W)$, let $G_{\mathrm{list}} \leftarrow G_{\mathrm{list}} \cup \{(\mathrm{ID}, m_w, W)\}$.
   - PSign(.): This oracle takes the delegation W and message $m \in \{0, 1\}^*$as input, and outputs a proxy signature created by the proxy signer. If $\mathcal{A}$ gets $(m, \tau) \leftarrow \mathrm{PSign}(W, m)$, let $S_{\mathrm{list}} \leftarrow S_{\mathrm{list}} \cup \{(m, \tau)\}$.
4. $\mathcal{A}$ outputs (ID, m_w, 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_w, W), and satisfies: $\mathrm{DVerify}(W, \mathrm{ID}) = 1$, $(\mathrm{ID}, \cdot) \notin C_{\mathrm{list}}$, $(\mathrm{ID}, \cdot) \notin G_{\mathrm{list}}$ and $(\mathrm{ID}, m_w, \cdot) \notin D_{\mathrm{list}}$. $\mathrm{Exp}_F^{\mathrm{IBPS}}(k)$ returns 1.
   - The output is (W, m, τ), and satisfies $\mathrm{PVerify}((m, \tau), \mathrm{ID}_i, \mathrm{ID}_j) = 1$, $(W, m, \cdot) \notin S_{\mathrm{list}}$ and $(\mathrm{ID}_j, \cdot) \notin C_{\mathrm{list}}$ $(\mathrm{ID}_j, W, \cdot) \notin G_{\mathrm{list}}$, where ID_i and ID_j are the identities of the designator and the proxy signer defined by W, respectively. ExpID $\mathrm{Exp}_F^{\mathrm{IBPS}}(k)$ returns 2.

Otherwise, $\mathrm{Exp}_F^{\mathrm{IBPS}}(k)$ returns 0.

Definition 3 [6]. An ID-based proxy digital signature scheme IBPS 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 $\mathscr{A}$, $\Pr\left[\mathrm{Exp}_F^{\mathrm{IBPS}}(k)=1\right]$ and $\Pr\left[\mathrm{Exp}_F^{\mathrm{IBPS}}(k)=2\right]$ are negligible.

3 Our scheme

3.1 Scheme description

In this section, we present an ID-based proxy signature scheme without pairing. Our scheme rests on the ECDLP.

Setup Takes a security parameter k, returns system parameters and a master key. Given k, KGC does as follows.

1) Choose a k-bit prime p and determine the tuple $\{F_p, E/F_p, G, P\}$ as defined in Section 2.
2) Choose the master private key $x \in Z_n^*$ and compute the master public key $P_{\mathrm{pub}}=x \cdot P$.
3) Choose two cryptographic secure hash functions $H_1 : \{0,1\}^* \to Z_n{}^*$ and $H_2 : \{0,1\}^* \times G \to Z_p{}^*$.
4) Publish $\{F_p, E/F_p, G, P, P_{\mathrm{pub}}, H_1, H_2\}$ as system parameters and keep the master key x secretly.

Extract Takes system parameters, master key, and a user's identifier as input, returns the user's ID-based private key. With this algorithm, KGC works as follows for each user U with identifier ID_U.

1) Choose at random $r_U \in Z_n^*$, compute $R_U=r_U P$ and $h_U=H_1(\mathrm{ID}_U, R_U)$.
2) Compute $D_U = r_U + h_U x$.

U's private key is the tuple (D_U, R_U) and is transmitted to U via a secure out-of-band channel. U can validate her private key by checking whether the equation

$$D_U \cdot P = R_U + h_U \cdot P_{\mathrm{pub}} \tag{5}$$

holds. The private key is valid if the equation holds and vice versa.

Delegate Takes O's secret key D_O and a warrant m_ω as input, and outputs the delegation $W_{O\to A}$. As shown in Fig. 1, the user O does as the follows.

1) Generate a random a and compute $K=a \cdot P$.
2) Compute $e_1=H_2(m_w, K, \mathrm{ID}_A)$ and $\sigma = e_1 D_O + a \bmod n$.

The delegation is $W_{O\to A}=(\mathrm{ID}_O, R_O, \mathrm{ID}_A, m_w, K, \sigma)$.

DVerify As shown in Fig. 1, to verify the delegation $W_{O\to A}$ for message m_w, the user A first computes $e_1=H_2(m_w, K, \mathrm{ID}_A)$, $h_O=H_1(\mathrm{ID}_O, R_O)$ and then checks whether

$$\sigma \cdot P = e_1\left(R_O + h_O \cdot P_{\mathrm{pub}}\right) + K \tag{6}$$

Accept if it is equal. Otherwise reject.

PKgen If A accepts the delegation $W_{O\to A}$, as shown in Fig. 1, he computes the proxy signing key D_p as $D_p = \sigma + D_A e_2 \bmod n$, where $e_2=H_2(m_w, K, \mathrm{ID}_O)$.

Sign Takes system parameters, the proxy signing key D_p and a message m as inputs, returns a signature of the message m. The user A does as follows.

1) Choose at random $b \in Z_n{}^*$ to compute $R=b \cdot P$.
2) Compute $h=H_2(m, R)$.
3) Verify whether the equation gcd $(b+h, n)=1$ holds: Continue if it does and return to step 1 otherwise.
4) Compute $s = (l+h)^{-1} D_p \bmod n$.
5) The resulting signature is $(\mathrm{ID}_O, R_O, \mathrm{ID}_A, R_A, m_w, K, \sigma, R, s)$.

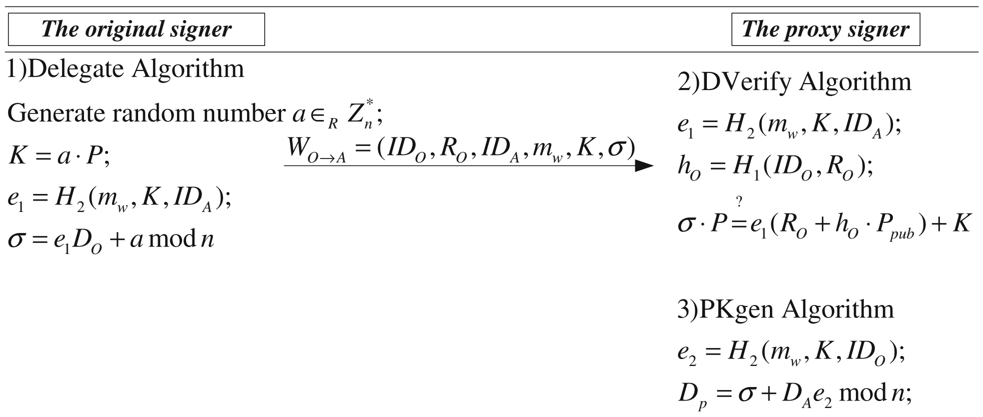
The original signer
The proxy signer
1)Delegate Algorithm
Generate random number $a \in_R Z_n^*$;
$K = a \cdot P$;
$e_1 = H_2(m_w, K, ID_A)$;
$\sigma = e_1 D_O + a \bmod n$
$W_{O\to A} = (ID_O, R_O, ID_A, m_w, K, \sigma)$
2)DVerify Algorithm
$e_1 = H_2(m_w, K, ID_A)$;
$h_O = H_1(ID_O, R_O)$;
$\sigma \cdot P \overset{?}{=} e_1(R_O + h_O \cdot P_{pub}) + K$
3)PKgen Algorithm
$e_2 = H_2(m_w, K, ID_O)$;
$D_p = \sigma + D_A e_2 \bmod n$;

Fig. 1 The process of Delegate, DVerify, and PKgen

Verify To verify the signature (ID$_O$, R_O, ID$_A$, R_A, m_w, K, σ, R, s) for message m, a verifier first checks if the proxy signer and the message conform to m_w, then he computes, $h_O = H_1(\mathrm{ID}_O, R_O)$,$h_A$=$H_1$(ID$_A$, R_A), e_1=H_2(m_w, K, ID$_A$), e_2=H_2(m_w, K, ID$_O$), $h = H_2(m, R)$, and then checks whether

$$s(R+h\cdot P)=e_1\left(R_O+h_O\cdot P_{\text{pub}}\right)+e_2\left(R_A+h_A\cdot P_{\text{pub}}\right)+K \tag{7}$$

Accept if it is equal. Otherwise reject.

Since R=b·P and $s=(b+h)^{-1}D_p \bmod n$, we have

$$\begin{aligned} s\cdot(R+h\cdot P)&=(b+h)^{-1}\cdot D_p\cdot(b\cdot P+h\cdot P)\\ &=(b+h)^{-1}\cdot D_p\cdot(b+h)\cdot P\\ &=D_p\cdot P=(\sigma+D_Ae_2)P=\sigma\cdot P+e_2D_A\cdot P\\ &=e_1\left(R_O+h_O\cdot P_{\text{pub}}\right)+K+e_2\left(R_A+h_A\cdot P_{\text{pub}}\right) \end{aligned} \tag{8}$$

Then the correctness of our scheme is proved.

3.2 Security analysis

Assume there is an adversary $\mathscr{A}$ who can break our ID-based proxy signature scheme $\sum$. We will construct a polynomial-time algorithm $\mathcal{F}$ that, by simulating the challenger and interacting with $\mathscr{A}$, solves the ECDLP.

Theorem 1 *Consider an adaptively chosen message attack in the random oracle model against* $\sum$. *If there is an attacker* $\mathscr{A}$ *that can break* $\sum$ *with at most* q_{H_2} H_2*-queries and* q_S *signature queries within time bound t and non-negligible probability* ε. *Then we can solve the ECDLP with non-negligible probability.*

Proof Suppose that there is an attacker $\mathscr{A}$ for an adaptively chosen message attack against Σ. Then, $\Pr\left[\mathrm{Exp}_F^{\mathrm{IBPS}}(k)=1\right]$ or $\Pr\left[\mathrm{Exp}_F^{\mathrm{IBPS}}(k)=2\right]$ are non-negligible. We show how to use the ability of $\mathscr{A}$ to construct an algorithm $\mathcal{F}$ solving the ECDLP.

Suppose $\mathcal{F}$ is challenged with a ECDLP instance (P, Q) and is tasked to compute $x\in Z_n^*$ satisfying $Q = x\cdot P$. To do so, $\mathcal{F}$ sets $\{F_p, E/F_p, G, P, P_{\text{pub}}=Q, H_1, H_2\}$ as the system parameter and answers $\mathscr{A}$'s queries (described in definition 2) as follows.

Extract-query: $\mathscr{A}$ is allowed to query the extraction oracle for an identity ID$_U$. S simulates the oracle as follows. It chooses a_U, $b_U\in Z_n^*$ at random and sets

$$R_U=a_U\cdot P_{\text{pub}}+b_U\cdot P,\ D_U=b_U, \tag{9}$$

$$h_U=H_1(ID_U,R_U)\leftarrow -a_U \bmod n.$$

Note that (D_U, R_U) generated in this way satisfies the equation $D_U\cdot P=R_U+h_U\cdot P_{\text{pub}}$ in the extract algorithm. It is a valid secret key. F outputs (D_U, R_U, h_U) as the secret key of ID$_U$ and stores the value of (D_U, R_U, h_U) in the C_{list}-table(we modify the content of C_{list}-table).

Delegate-query: $\mathscr{A}$ queries the delegate oracle for a warrant m_w, ID$_O$ and ID$_A$, $\mathcal{F}$ first checks that whether ID$_O$ and ID$_A$ have been queried for the extraction oracle before. If yes, it just retrieves (D_O, R_O, h_O) from the table and uses these values to delegate a warrant m_w, according to the delegate algorithm described in the scheme. It outputs the delegation $W_{O\to A}$=(ID$_O$, R_O, ID$_A$, m_w, K, σ) for m_w, ID$_O$ and ID$_A$ and stores the value $W_{O\to A}$ in the hash table D$_{\text{list}}$ for consistency. If ID$_O$ or ID$_A$ has not been queried to the extraction oracle, $\mathcal{F}$ executes the simulation of the extraction oracle and uses the corresponding secret key to sign the message.

Since $\mathcal{F}$ knows every user's private key(described in Extract-query), he can simulate Delegate-query, DVerify-query, PKgen-query, PSign-query, and PVerify-query as he simulates Delegate-query.

1. If $\mathscr{A}$ can forge a valid delegation on warrant m_w with the probability $\varepsilon\geq 10(q_{H_2}+1)(q_{H_2}+q_S)/2^k$, i.e. $\Pr\left[\mathrm{Exp}_F^{\mathrm{IBPS}}(k)=1\right]\geq 10(q_{H_2}+1)(q_{H_2}+q_S)/2^k$, where m_w has not been queried to the signature oracle, then a replay of $\mathcal{F}$ with the same random tape but different choice of H_2 will output two valid delegation (ID$_O$, R_O, ID$_A$, m_w, K, σ, e_1) and $\left(ID_O,R_O,ID_A,m_w,K,\sigma',e_1'\right)$. Then we have

$$\sigma\cdot P=e_1\left(R_O+h_O\cdot P_{\text{pub}}\right)+K, \tag{10}$$

and

$$\sigma'\cdot P=e_1'\left(R_O+h_O\cdot P_{\text{pub}}\right)+K. \tag{11}$$

Let K=$a\cdot P$, $R_O=a_O\cdot P_{\text{pub}}+b_O\cdot P$, $P_{\text{pub}}=Q=x\cdot P$, then we have

$$\sigma\cdot P=e_1\left(a_O\cdot P_{\text{pub}}+b_O\cdot P+h_O\cdot P_{\text{pub}}\right)+a\cdot P, \tag{12}$$

and

$$\sigma'\cdot P=e_1'\left(a_O\cdot P_{\text{pub}}+b_O\cdot P+h_O\cdot P_{\text{pub}}\right)+a\cdot P. \tag{13}$$

then we have

$$\sigma\cdot P=e_1\cdot a_O\cdot x\cdot P+e_1\cdot b_O\cdot P+e_1\cdot h_O\cdot x\cdot P+a\cdot P, \tag{14}$$

and

$$\sigma'\cdot P=e_1'\cdot a_O\cdot x\cdot P+e_1'\cdot b_O\cdot P+e_1'\cdot h_O\cdot x\cdot P+a\cdot P. \tag{15}$$

Table 1 Cryptographic operation time (in milliseconds)

Modular exponentiation	Pairing	Pairing-based sca.mul	ECC-based sca.mul.	Map-to-point hash	General hash
5.31	20.04	6.38	2.21	3.04	<0.001

Hence, we have

$$(e_1 \cdot a_O + e_1 \cdot h_O - e'_1 \cdot a_O - e'_1 \cdot h_O) \cdot x \cdot P = (\sigma - \sigma' - e_1 \cdot b_O + e'_1 \cdot b_O) \cdot P \quad (16)$$

Let $u = (e_1 \cdot a_O + e_1 \cdot h_O - e'_1 \cdot a_O - e'_1 \cdot h_O)^{-1} \bmod n$ and $v = (\sigma - \sigma' - e_1 \cdot b_O + e'_1 \cdot b_O) \bmod n$, then, we get $x = uv \bmod n$. According to [12, **Lemma 4**], the ECDLP can be solved with probability $\varepsilon' \geq 1/9$ and time $t' \leq 23 q_{H_2} t/\varepsilon$.

2. From case 1, we know the adversary $\mathscr{A}$ cannot generate a valid delegation. In this case we will prove, if $\mathscr{A}$ can forge a valid signature on message m under the delegation $W_{O\to A}$=(ID$_O$, R_O, ID$_A$, m_w, K, σ) with the probability $\varepsilon \geq 10(q_{H_2}+1)(q_{H_2}+q_S)/2^k$, i.e. $\Pr\left[\mathrm{Exp}_F^{\mathrm{IBPS}}(k) = 2\right] \geq 10(q_{H_2}+1)(q_{H_2}+q_S)/2^k$, where m has not been queried to the signature oracle, then a replay of $\mathfrak{F}$ with the same random tape but different choice of H_2 will output two valid signatures (ID$_O$, R_O, ID$_A$, R_A, m_w, K, σ, R, s, e_1, e_2, h), and (ID$_O$, R_O, ID$_A$, R_A, m_w, K, σ', R, s, e'_1, e'_2, h'). Then we have

$$s(R + h \cdot P) = e_1\left(R_O + h_O \cdot P_{\mathrm{pub}}\right) + e_2\left(R_A + h_A \cdot P_{\mathrm{pub}}\right) + K, \quad (17)$$

and

$$s'(R + h' \cdot P) = e'_1\left(R_O + h_O \cdot P_{\mathrm{pub}}\right) + e'_2\left(R_A + h_A \cdot P_{\mathrm{pub}}\right) + K. \quad (18)$$

Let $K = a \cdot P$, $R = b \cdot P$, $R_O = a_O \cdot P_{\mathrm{pub}} + b_O \cdot P$, $P_{\mathrm{pub}} = Q = x \cdot P$, then we have

$$s(b + h) = e_1((a_O x + b_O) + h_O x) + e_2((a_A x + b_A) + h_A x) + a \bmod n, \quad (19)$$

and

$$s'(b + h') = e'_1((a_O x + b_O) + h_O x) + e'_2((a_A x + b_A) + h_A x) + a \bmod n. \quad (20)$$

In these equations, only b and x are unknown to $\mathfrak{F}$. $\mathfrak{F}$ solves for these values from the above like he does in case 1, and outputs x as the solution of the discrete logarithm problem.

According to [12, **Lemma 4**], the ECDLP can be solved with probability $\varepsilon' \geq 1/9$ and time $t' \leq 23 q_{H_2} t/\varepsilon$.

4 Comparison with previous scheme

In this section, we will compare the efficiency of our new scheme with Gu et al.'s scheme [6], Zhang's scheme [7], Wu et al.'s scheme [8], Gu et al.'s [9], and Ji et al.'s scheme [10]. In the computation efficiency comparison, we obtain the running time for cryptographic operations using MIRACAL [13], a standard cryptographic library.

The hardware platform is a PIV 3-GHZ processor with 512-MB memory and a Windows XP operation system. For the pairing-based scheme, to achieve the 1,024-bit RSA level security, we use the Tate pairing defined over the supersingular elliptic curve $E/F_p : y^2 = x^3 + x$ with embedding degree 2, where q is a 160-bit Solinas prime $q = 2^{159} + 2^{17} + 1$ and p a 512-bit prime satisfying $p + 1 = 12qr$. For the ECC-based schemes, to achieve the

Table 2 Performance comparison of different schemes

	Running time					
	Extract	Delegate	Dverify	PKgen	Psign	PVerify
Gu et al.'s scheme [6]	9.42	11.69	45.39	6.38	11.69	57.09
Zhang et al.'s scheme [7]	9.42	32.78	51.71	20.04	84.51	52.8
Wu et al.'s scheme [8]	9.42	6.38	60.06	–	31.6	112.26
Gu et al.'s scheme [9]	6.38	17.02	43.4	6.38	11.69	43.4
Ji et al.'s scheme [10]	6.38	12.76	46.42	6.38	19.14	46.42
Our scheme	2.21	2.21	4.42	–	2.21	8.84

same security level, we employed the parameter secp160r1 [14], recommended by the Certicom Corporation, where $p = 2^{160} - 2^{31} - 1$. The running times are listed in Table 1, where sca.mul. stands for scalar multiplication.

To evaluate the computation efficiency of different schemes, we use the simple method from [15, 16]. For example, the Extract algorithm of our scheme requires a KGC to carry out one ECC-based scale multiplication; thus, the computation time of the sign algorithm is 2.21×1=2.21 ms; the Delegate algorithm has to carry out one ECC-based scalar multiplications, and the resulting running time is 2.21×1=2.21 ms; the Dverify algorithm has to carry out two ECC-based scalar multiplications, and the resulting running time is 2.21×2=4.42 ms; the PKgen algorithm has to carry out one modular multiplications, we will ignore running time; the Psign algorithm has to carry out one ECC-based scalar multiplications, and the resulting running time is 2.21×1=2.21 ms; the PVerify algorithm has to carry out four ECC-based scalar multiplications, and the resulting running time is 2.21×4=8.84 ms. As another example, in Gu et al.'s scheme [6], the Extract algorithm of requires a KGC to carry out one pairing-based scale multiplication and a map-to-point hash function; thus, the computation time of the sign algorithm is 6.38×1+3.04=9.42 ms; the Delegate algorithm has to carry out one pairing -based scalar multiplications and a modular exponentiation, and the resulting running time is 6.38×1+5.31=11.69 ms; the Dverify algorithm has to carry out two pairing operations and a modular exponentiation, and the resulting running time is 20.04×2+5.31=45.39 ms; the PKgen algorithm has to carry out one pairing -based scalar multiplications, and the resulting running time is 6.38×1=6.38 ms; the Psign algorithm has to carry out one pairing -based scalar multiplications and a modular multiplication, then the resulting running time is 6.38+5.31=11.69 ms; the PVerify algorithm has to carry out two pairing, one pairing-based scalar multiplications and two modular exponentiations, then the resulting running time is 20.04×2+6.38+5.31×2=57.09 ms. Table 2 shows the results of the performance comparison.

According to Table 2, the running time of the Psign algorithm of our scheme is 18.9% of Gu et al.'s schemes [6], 2.61% of Zhang et al.'s scheme [7], 6.99%of Wu et al.'s scheme [8], 18.9% of Gu et al.'s scheme [9], and 11.55% of Ji et al.'s scheme [10], the running time of the Pverify algorithm of our scheme is 15.27% of Gu et al.'s schemes [6], 16.74% of Zhang et al.'s scheme [7], 7.87% of Wu et al.'s scheme [8], 20.37% of Gu et al.'s scheme [9], and 19.04% of Ji et al.'s scheme [10]. Thus our scheme is more efficient than the previous schemes [6–10].

5 Conclusion

In this paper, we have proposed an efficient identity-based proxy signature scheme. We also prove the security of the scheme under random oracle. Compared with previous scheme, the new scheme reduces the running time heavily. Therefore, our scheme is more practical than the previous related schemes for practical application.

Acknowledgments The authors thank the anonymous reviewers and Dr. Guy Pujolle for their valuable comments. This research was supported by the Fundamental Research Funds for the Central Universities under Grants 201275786.

References

1. Shamir A (1984) Identity-based cryptosystems and signature schemes. In: Proceedings of CRYPTO 1984. Lecture Notes in Computer Science. vol 196, pp. 47–53
2. Mambo M, Usuda K, Okamoto E (1996) Proxy signature: delegation of the power to sign messages. IEICE Trans Fundamentals E79-A(9):1338–1353
3. Cha JC, Cheon JH (2002) An identity-based signature from gap Diffie-Hellman groups. In: Desmedt YG (ed) PKC 2003. LNCS. vol 2567. Springer, Heidelberg, pp 18–30
4. Hess F (2003) Efficient identity based signature schemes based on pairings. In: Nyberg K, Heys HM (eds) SAC 2002. LNCS, vol 2595. Springer, Heidelberg, pp 310–324
5. Barreto PSLM, Libert B, McCullagh N, Quisquater J (2005) Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In: Roy B (ed) ASIACRYPT 2005. LNCS, vol 3788. Springer, Heidelberg, pp 515–532
6. Gu C, Zhu Y (2005) Provable security of ID-based proxy signature schemes. In: Lu X, Zhao W (eds) ICCNMC 2005. LNCS, vol 3619. Springer, Heidelberg, pp 1277–1286
7. Zhang J, Zou W (2007) Another ID-based proxy signature scheme and its extension. Wuhan Univ J Nat Sci 12:133–136
8. Wu W, Mu Y, Susilo W et al. (2007) Identity-based proxy signature from pairings, ATC 2007, LNCS 4610, pp. 22–31.
9. Gu C, Zhu Y (2008) An efficient ID-based proxy signature scheme from pairings. Inscrypt 2007, LNCS 4990, pp. 40–50
10. Ji H, Han W, Zhao L et al (2009) An identity-based proxy signature from bilinear pairings, 2009 WASE International Conference on Information Engineering, pp 14–17
11. Chen L, Cheng Z, Smart NP (2007) Identity-based key agreement protocols from pairings. Int J Inf Secur (6):213–241.
12. David P, Jacque S (2000) Security arguments for digital signatures and blind signatures. J Cryptol 13(3):361–396
13. Shamus Software Ltd., Miracl library, http://www.shamus.ie/index.php?page=home
14. The Certicom Corporation, SEC 2: Recommended Elliptic Curve Domain Parameters, www.secg.org/collateral/sec2_final.pdf
15. Cao X, Kou W, Du X (2010) A pairing-free identity-based authenticated key agreement protocol with, minimal message exchanges. Inf Sci 180:2895–2903
16. He D, Chen J, Hu J. An ID-based client authentication with key agreement protocol for mobile client-server environment on ECC with provable security, Information Fussion, doi:10.1016/j.inffus.2011.01.001.