A Lightweight Identity Based Signature Scheme

David Galindo[1]* and Flavio D. Garcia[2]**

[1] University of Luxembourg
david.galindo@uni.lu
[2] Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands.
flaviog@cs.ru.nl

Abstract. We present a new identity based signature scheme that is secure against existential forgery on adaptively chosen message and identity attacks. The security is proven in the Random Oracle Model under the discrete logarithm assumption. The new scheme does not use pairings and is arguably the most efficient identity-based signature scheme known to date.

Keywords: identity-based signature, efficiency, provable security, Schnorr, random oracle model.

1 Introduction

Digital signatures are fundamental primitives in public key cryptography. They ensure the authenticity of the originator of a digital document and the integrity of that document, while they also prevent the originator from repudiating that very same document later on. A signature on a given bit-string is valid if it passes the associated verification test, that takes as inputs a verification key, a purported signature, and a document. In traditional signature schemes the verification key is a mathematical object taken at random from some set. A external binding between the verification key and the signing entity is therefore needed. This binding takes the form of a certificate, which is created by a Trusted Third Party (TTP).

The concept of identity-based signature (IBS) was introduced by Shamir [Sha85]. The idea is that the the identity of the signer is used as the verification key. This dispenses with the need of a external binding. The identity can be any string that singles out the entity, for instance, a social security number or an IP address. IBS systems have a drawback, namely, the entity cannot build the

* Work done while at University of Malaga. Partially funded by the Spanish Ministry of Science and Education through the projects ARES (CSD2007-00004) and CRISIS (TIN2006-09242).

** partially supported by the research program Sentinels (www.sentinels.nl), project PEARL (7639). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

signing key by itself. Instead, the TTP is in charge of assigning and delivering, via a confidential channel, the secret key to each user. Such a system is called *key-escrowed*, since the TTP knows the private keys of the users. This in sharp contrast to the traditional setting.

Several IBS schemes based on factoring or RSA were proposed in the late eighties and early nineties, for instance [Sha85,FS87,GQ90,Oka93] to name a few. After the revival of research in identity-based cryptography due to the use of pairings (cf. [BSS05] Chapter 5 for an introduction to pairings), many other IBS schemes have been proposed, for instance [SOK00,Hes03,CC02].

We present a new identity-based signature scheme. Our construction uses Schnorr signatures [Sch91] and does not need pairings. We show that the resulting scheme is arguably the most efficient provably-secure IBS scheme known to date, be it based of factoring, discrete logarithm or pairings. This makes it attractive for application in resource-constrained environments.

Organization of this paper. Section 2 introduces standard definitions from the literature. Section 3 describes our new identity based signature scheme. In Section 4 we prove the security of the scheme. Section 5 compares the computational and length complexity of our scheme to other schemes from the literature. Finally Section 6 concludes the paper.

2 Preliminaries

This section introduces the syntaxis and security definitions of identity-based signatures and the discrete logarithm assumption. Most of it is standard, we refer the reader to [BNN04] for a thorough explanation. We introduce some basic notation. If S is a set then $s_1, \ldots, s_n \overset{\$}{\leftarrow} S$ denotes the operation of picking n elements s_i of S independently and uniformly at random. We write $\mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \leftarrow \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and letting z be the output.

2.1 Identity Based Signatures

An *identity based signature scheme* (IBS) is a quadruple $(\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V})$ of probabilistic polynomial-time algorithms, where

- $\mathcal{G}$ is the *parameter-generation algorithm.* It takes as input the security parameter 1^η and outputs the system public parameters mpk and the master secret key msk.
- $\mathcal{E}$ is the *key-extraction algorithm* that takes as input parameters mpk, a master key msk and an identity id and outputs a private key $\mathsf{sk}_{\mathsf{id}}$ corresponding to the user with this identity.

- $\mathcal{S}$ is a *signing algorithm* that takes as input parameters mpk, a private key $\mathsf{sk}_{\mathsf{id}}$ and a message m and outputs a signature σ.
- $\mathcal{V}$ is a deterministic *signature verification algorithm* that takes as input parameters mpk, a signature σ, a message m and an identity id and outputs whether or not σ is a valid signature of m relative to $(\mathsf{mpk}, \mathsf{id})$.

Definition 1 (IBSEC). An identity based signature scheme $\Sigma = (\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V})$ is said to be secure against *existential forgery on adaptively chosen message and identity attacks* if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the probability of the experiment $\mathbf{IBSEC}_{\Sigma}(\mathcal{A}) = 1$ defined below is a negligible function of η. During this experiment $\mathcal{A}$ has access to two oracles: a key-extraction oracle $\mathcal{O}_{\mathcal{E}}$ that takes as input an identity id and outputs $\mathcal{E}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$; and a signature oracle $\mathcal{O}_{\mathcal{S}}$ that takes as input an identity id and a message m and returns a signature $\mathcal{S}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{id}}, m)$.

```
$\mathbf{IBSEC}_{\Sigma}(\mathcal{A})$:
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathcal{G}(1^{\eta})$
$(\mathsf{id}^{\star}, m^{\star}, \sigma^{\star}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{E}}(\cdot), \mathcal{O}_{\mathcal{S}}(\cdot,\cdot)}(\mathsf{mpk})$
return $\mathcal{V}(\mathsf{mpk}, \sigma^{\star}, m^{\star}, \mathsf{id}^{\star})$
```

Some restrictions apply. In particular, it is required that $\mathsf{id}^{\star}$ and $(\mathsf{id}^{\star}, m^{\star})$ are not equal to any query made to the oracles $\mathcal{O}_{\mathcal{E}}(\cdot)$ and $\mathcal{O}_{\mathcal{S}}(\cdot, \cdot)$ respectively, and that the same id cannot be queried to $\mathcal{O}_{\mathcal{E}}(\cdot)$ twice.

Definition 2 (Discrete Logarithm Assumption). We say that a group generation function $(\mathcal{G}, g, q) \leftarrow \mathsf{Gen}(1^{\eta})$ generates DL-secure groups if for all probabilistic polynomial-time algorithms $\mathcal{A}$, the probability

$$\mathbb{P}[(\mathcal{G}, g, q) \leftarrow \mathsf{Gen}(1^{\eta}); a \xleftarrow{\$} \mathbb{Z}_q : a \leftarrow \mathcal{A}(\mathcal{G}, q, g, g^a)]$$

is a negligible function of η.

3 The Construction

The idea behind our construction is to use two concatenated Schnorr signatures [Sch91]. Firstly, the TTP produces a Schnorr signature on the identity of the user by using the master secret key. This signature implicitly defines a unique Schnorr-like public key for which only the user knows the corresponding private key. Next, the user builds a second Schnorr-like signature, this time on the message, by using its private key. The verification of a signature on message m under identity id implicitly checks whether the two concatenated Schnorr signatures are correct. The idea of using a Schnorr signature as a private/public key pair is not original of ours, and it can already be found in [CJT04,BSNS05,BFPW07].

The details of the new scheme are given below.

- The parameter generation algorithm $\mathcal{G}$ on input 1^η outputs public parameters mpk and a master secret key msk where: $(\mathcal{G}, g, q)$ is generated by calling the group generation algorithm Gen on input 1^η. $\mathcal{G}$ is the description of a group of prime order $q = q(\eta)$ with $2^\eta \leq q < 2^{\eta+1}$ and g is a generator of $\mathcal{G}$. $G\colon \{0,1\}^* \to \mathbb{Z}_q, H\colon \{0,1\}^* \to \mathbb{Z}_q$ are descriptions of hash functions. Let z be a random element from $\mathbb{Z}_q$ and set $(\mathsf{mpk}, \mathsf{msk}) = \big((\mathcal{G}, g, q, G, H, g^z), z\big)$.

- The key-extraction algorithm $\mathcal{E}$ on input public parameters mpk, a master key $\mathsf{msk} = z$ and an identity id, picks $r \overset{\$}{\leftarrow} \mathbb{Z}_q$ and sets $y = r + z \cdot H(g^r, \mathsf{id}) \bmod q$. Then it outputs the secret key $\mathsf{sk}_{\mathsf{id}} = (y, g^r)$.

- The signing algorithm $\mathcal{S}$ on input parameters mpk, a private key $\mathsf{sk}_{\mathsf{id}} = (y, g^r)$ and a message m proceeds as follows. It selects $a \overset{\$}{\leftarrow} \mathbb{Z}_q$ and computes $b = a + y \cdot G(\mathsf{id}, g^a, m)$. Then it outputs the signature $\sigma = (g^a, b, g^r)$.

- The verification algorithm $\mathcal{V}$ on input parameters $\mathsf{mpk} = (\mathcal{G}, g, q, G, H, g^z)$, a signature $\sigma = (g^a, b, g^r)$, a message m and an identity id proceeds as follows. It outputs whether or not the equation $g^b = g^a(g^r g^{zc})^d$ holds, where $c = H(g^r, \mathsf{id})$ and $d = G(\mathsf{id}, g^a, m)$.

4 Security of the Construction

This section analyzes the security of the proposed scheme in the random oracle model. For the sake of self-containment we include the Multiple-Forking Lemma of Boldyreva, Palacio and Warinschi [BPW03] as it is strongly used in our security proof. The lemma is a further generalization of the General Forking Lemma proposed by Bellare and Neven [BN06] to multiple random oracles and signatures. The General Forking Lemma itself is a generalization of the Forking Lemma, originally proposed in [PS00]. Intuitively, this lemma states that, in the random oracle model, if there is an algorithm (a forger) that can produce a forgery, then it is possible to get a different forgery on the same message (by changing the answer of the random oracle). Finally, Theorem 3 shows the security of our scheme.

Lemma 1 (Multiple-Forking Lemma [BPW03]). *Fix $\alpha \in \mathbb{Z}^+$ and a set S such that $|S| \geq 2$. Let Y be a randomized algorithm that on input x and a sequence of elements $s_1 \dots s_\alpha \in S$, returns a triple (I, J, σ) consisting of two integers associated to Y and a bitstring σ. Let $n \geq 1$ be an odd integer and x a bitstring. The multiple-forking algorithm $\mathsf{MF}_{Y,n}$ associated to Y and n is defined as*

```
**algorithm** $\mathsf{MF}_{Y,n}(x)$:
  *Pick random coins* $\rho$ *for* $Y$
  $s_1 \dots s_\alpha \xleftarrow{\$} S$
  $(I, J, \sigma_0) \leftarrow Y(x, s_1 \dots s_\alpha; \rho)$
  *if* $(I = 0 \vee J = 0)$ *then return* $\perp$
  $s_I^1 \dots s_\alpha^1 \xleftarrow{\$} S$
  $(I_1, J_1, \sigma_1) \leftarrow Y(x, s_1, \dots s_{I-1}, s_I^1 \dots s_\alpha^1; \rho)$
  *if* $((I, J) \neq (I_1, J_1) \vee s_I = s_I^1)$ *then return* $\perp$
  *for* $i = 2\,; i < n\,; i = i + 2$ *do*
    $s_1^i \dots s_\alpha^i \xleftarrow{\$} S$
    $(I_i, J_i, \sigma_i) \leftarrow Y(x, s_1 \dots s_{J-1}, s_J^i, \dots, s_\alpha^i; \rho)$
    *if* $((I_i, J_i) \neq (I, J) \vee s_J^i = s_J^{i-1})$ *then return* $\perp$
    $s_I^{i+1} \dots s_\alpha^{i+1} \xleftarrow{\$} S$
    $(I_{i+1}, J_{i+1}, \sigma_{i+1}) \leftarrow Y(x, s_1 \dots s_{J-1}, s_J^i, \dots, s_{I-1}^i, s_I^{i+1}, \dots, s_\alpha^{i+1}; \rho)$
    *if* $((I_{i+1}, J_{i+1}) \neq (I, J) \vee s_J^{i+1} = s_J^i)$ *then return* $\perp$
  *endfor*
  *return* $\sigma_0 \dots \sigma_n$
```

Let

$$\mathsf{acc} = \mathbb{P}[x \xleftarrow{\$} PG; s_1 \dots s_\alpha \xleftarrow{\$} S; (I, J, \sigma) \leftarrow Y(x, s_1 \dots s_\alpha) : I \geq 1 \wedge J \geq 1]$$
$$\mathsf{frk} = \mathbb{P}[x \xleftarrow{\$} PG : \mathsf{MF}_{Y,n}(x) \neq \perp].$$

Then

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}^n}{\alpha^{2n}} - \frac{n}{|S|}\right)$$

and therefore

$$\mathsf{acc} \leq \sqrt[n+1]{\alpha^{2n}\,\mathsf{frk}} + \sqrt[n+1]{\frac{n\alpha^{2n}}{|S|}}$$

Theorem 3. *If the group generation function* Gen *generates discrete logarithm secure groups then the construction above is secure with respect to Definition 1, in the random oracle model.*

Proof. Assume there is an adversary $\mathcal{A}$ that wins the game IBSEC with non-negligible probability. Eventually, $\mathcal{A}$ outputs an attempted forgery of the form $\sigma = (A, B, R)$. Let E be the event that σ is a valid signature and R was contained in an answer of the signing oracle $\mathcal{O}_\mathcal{S}$. Let NE be the event that σ is a valid signature and R was never part of an answer of $\mathcal{O}_\mathcal{S}$. Clearly, $\mathsf{Adv}_\mathcal{A}^{\text{IBSEC}}(\eta) = \mathbb{P}[E] + \mathbb{P}[NE]$.

Next we build the following adversaries $\mathcal{B}_1, \mathcal{B}_2$ against the discrete logarithm assumption. Intuitively, $\mathcal{B}_1$ breaks the DL-assumption when the event E happens and $\mathcal{B}_2$ in case of NE.

$\mathcal{B}_1$ takes as argument the description of a group $(\mathcal{G}, q, g)$ and a challenge g^r with $r \xleftarrow{\$} \mathbb{Z}_q$ and tries to extract the discrete logarithm r. To do so it will run the adversary $\mathcal{A}$, for which simulates its environment as follows:

- $\mathcal{B}_1$ picks a random $i \leftarrow [Q_G]$, where Q_G is the maximal number of queries that the adversary $\mathcal{A}$ performs to the random oracle G. Let $\mathsf{id}^\star$ (the target identity) be the identity in the i-th query to the random oracle G. Next, $\mathcal{B}_1$ chooses $z \xleftarrow{\$} \mathbb{Z}_q$ and sets public parameters $\mathsf{mpk} = (\mathcal{G}, q, g, G, H, \mathsf{mpk} = g^z)$ where G, H are description of hash functions modeled as random oracles. As usual, $\mathcal{B}_1$ simulates these oracles by keeping two lists L_G, L_H containing the queried values together with the answers given to $\mathcal{A}$.
- Every time $\mathcal{A}$ queries the key extraction oracle $\mathcal{O}_\mathcal{E}$, for user id, $\mathcal{B}_1$ chooses $c, y \xleftarrow{\$} \mathbb{Z}_q$, sets $R = g^{-zc}g^y$ and adds $((R, \mathsf{id}), c)$ to the list L_H. Then it returns the key (y, R) to $\mathcal{A}$.
- When $\mathcal{A}$ makes a call to the signature oracle (id, m) with $\mathsf{id} \neq \mathsf{id}^\star$, $\mathcal{B}_1$ simply computes id's private key as described in the previous bullet. Then calls it runs the signing algorithm $\mathcal{S}$ and returns the produced signature to $\mathcal{A}$.
- When $\mathcal{A}$ makes a call to the signature oracle (id, m) with $\mathsf{id} = \mathsf{id}^\star$, $\mathcal{B}_1$ chooses $t \xleftarrow{\$} \mathbb{Z}_q, B \xleftarrow{\$} \mathcal{G}$, sets $R = g^{-zc}(g^r)^t$, $c = H(\mathsf{id}, g^r)$ and $A = B(g^r g^{zc})^{-d}$. Then it returns the signature (A, B, R) to the adversary $\mathcal{A}$.
- $\mathcal{B}_1$ runs the algorithm $\mathsf{MF}_{Y,1}(\mathsf{mpk})$ as described in Lemma 1. Here algorithm Y is simply a wrapper that takes as explicit input the answers from the random oracles. Then it calls $\mathcal{A}$ and returns its output together with two integers I, J. These integers are the indexes of $\mathcal{A}$'s calls to the random oracles G, H with the target identity $\mathsf{id}^\star$.

```
**algorithm** $\mathsf{MF}_{Y,1}(\mathsf{mpk})$:
  Pick random coins $\rho$ for $Y$
  $s_1 \ldots s_{Q_G} \xleftarrow{\$} \mathbb{Z}_q$
  $(I, J, \sigma_0) \leftarrow Y(\mathsf{mpk}, s_1 \ldots s_{Q_G}; \rho)$
  **if** $(I = 0 \vee J = 0)$ **then return** $\perp$
  $s^1_I \ldots s^1_{Q_G} \xleftarrow{\$} \mathbb{Z}_q$
  $(I_1, J_1, \sigma_1) \leftarrow Y(\mathsf{mpk}, s_1, \ldots s_{I-1}, s^1_I \ldots s^1_{Q_G}; \rho)$
  **if** $((I, J) \neq (I_1, J_1) \vee s_I = s^1_I)$ **then return** $\perp$
  **else return** $\sigma_0, \sigma_1$
```

In this way we get two forgeries of the form $\sigma_0 = (\mathsf{id}, m, (A, B_1, R))$ and $\sigma_1 = (\mathsf{id}, m, (A, B_2, R))$. Let d_1 be the answer from the random oracle G given to $\mathcal{A}$ in the first execution, i.e., s_I in $\mathsf{MF}_{Y,1}(\mathsf{mpk})$, and let d_2 be the second answer s^1_I. If the identity id is not equal to the target identity $\mathsf{id}^\star$ then $\mathcal{B}_1$ aborts. Otherwise it terminates and outputs the attempted discrete logarithm $(B_1 - B_2)(td_1 - td_2)^{-1}$.

Next we show that $\mathcal{B}_1$'s output is indeed the discrete logarithm r of the challenge G^r. Since both signatures are valid we get that

$$g^{B_1} = A(Rg^{zc})^{d_1} \qquad \text{and} \qquad g^{B_2} = A(Rg^{zc})^{d_2}$$

where $c = H(g^r, \mathsf{id})$ and $R = g^{-zc}g^{rt}$. Hence, $B_1 = \log A + rtd_1$ and $B_2 = \log A + rtd_2$ and therefore $r = (B_1 - B_2)(td_1 - td_2)^{-1}$.

Next we bound success probability of $\mathcal{B}_1$ against de discrete logarithm assumption. With probability $1/Q_G$ the target identity $\mathsf{id}^\star$ equals the identity id output by the adversary. Then, it follows from Lemma 1 that

$$\mathsf{Adv}^{DL}_{\mathcal{B}_1} \geq \frac{\mathsf{frk}}{Q_G} \geq \frac{\mathsf{Adv}^{\text{IBSEC}}_{\mathcal{A}}(\eta)}{Q_G}\left(\frac{\mathsf{Adv}^{\text{IBSEC}}_{\mathcal{A}}(\eta)}{Q_G^2} - \frac{1}{2^\eta}\right)$$

Next we bound the success probability of NE. This case is slightly more involved as it uses nested rewindings of the adversary. In this case $\mathcal{B}_2$ attacks the public key of the trusted authority g^z. It takes as argument the description of a group $(\mathcal{G}, q, g)$ and a challenge g^z with $z \xleftarrow{\$} \mathbb{Z}_q$ and outputs its discrete logarithm z. To do so it will run the adversary $\mathcal{A}$ simulating its environment as follows:

- At the beginning of the experiment, $\mathcal{B}_2$ sets public parameters $\mathsf{mpk} = (\mathcal{G}, q, g, G, H, \mathsf{mpk} = g^z)$ where G, H are description of hash functions modeled as random oracles and g^z is the challenge. As usual, $\mathcal{B}_2$ simulates these oracles by keeping two lists L_G, L_H containing the queried values together with the answers given to $\mathcal{A}$.
- Every time $\mathcal{A}$ queries the key extraction oracle $\mathcal{O}_{\mathcal{E}}$, for user id, $\mathcal{B}_2$ chooses $c, y \xleftarrow{\$} \mathbb{Z}_q$, sets $R = g^{-zc}g^y$ and adds $((R, \mathsf{id}), c)$ to the list L_H. Then it returns the key (y, R) to $\mathcal{A}$.
- When $\mathcal{A}$ makes a call to the signature oracle (id, m), $\mathcal{B}_2$ simply computes id's secret key as described in the previous step. Then computes a signature by calling $\mathcal{S}$, adding the respective call to the oracle G, $((\mathsf{id}, g^a, m), d)$ to the list L_G. Then it gives the resulting signature to the adversary.
- $\mathcal{B}_2$ runs the algorithm $\mathsf{MF}_{\mathcal{A},3}(\mathsf{mpk})$ as described in Lemma 1. In this way, either $\mathcal{B}_2$ aborts prematurely or we get, for some identity id, some message m and some R, four forgeries $(\mathsf{id}, m, (A_k, B_k, R))$, $k = 1 \ldots 4$ with $A_1 = A_2$ and $A_3 = A_4$. As all these signatures are valid, the following equations hold

  $$\begin{aligned} B_1 &= \log A_1 + (\log R + c_1 z)d_1 \qquad & B_2 &= \log A_2 + (\log R + c_1 z)d_2 \\ B_3 &= \log A_3 + (\log R + c_2 z)d_3 \qquad & B_4 &= \log A_4 + (\log R + c_2 z)d_4 \end{aligned}$$

  with $c_1 \neq c_2$, $d_1 \neq d_2$ and $d_3 \neq d_4$. Since we know $c_1, c_2, d_1, \ldots, d_4$, a simple computation yields

  $$z = \frac{B_3 + B_2 - B_1 - B_4}{c_2(d_3 - d_4) - c_1(d_1 - d_2)} .$$

It follows that the success probability of $\mathcal{B}_2$ is bounded by

$$\mathsf{Adv}_{\mathcal{B}_2}^{DL}(\eta) \geq \mathsf{frk} \geq \mathsf{Adv}_{\mathcal{A}}^{\text{IBSEC}}(\eta) \left(\frac{(\mathsf{Adv}_{\mathcal{A}}^{\text{IBSEC}}(\eta))^3}{(Q_G Q_H)^6} - \frac{3}{2^\eta} \right)$$

$\square$

5 Comparison to Previous Schemes

This section compares the efficiency of our scheme with previous provably secure IBS schemes in terms of computational complexity and signature size. Given the considerable number of IBS schemes in the literature, we choose not to list all the existing schemes one by one. Instead we classify previous schemes in three categories, depending on whether those schemes are based on factoring, elliptic curve discrete logarithm (ECDL) or pairing. Then we show that our scheme can be considered as a benchmark in efficiency in each of these categories. The reader interested in the state of the art of IBS schemes is referred to [BNN04].

We need to introduce some terminology. In what follows, an exponentiation refers to computing g^r for randomly taken $g \xleftarrow{\$} G$ and $r \xleftarrow{\$} \mathbb{Z}_t$, where G denotes a finite group and r is an integer. A multi-exponentiation mexp(l) refers to computing $g_1^{r_1} \cdots g_l^{r_l}$, where $g_1, \ldots, g_l \xleftarrow{\$} G$ and $r_1, \ldots, r_l \xleftarrow{\$} \mathbb{Z}_t$. This operation can be computed more efficiently than just computing l single exponentiations due to an algorithm by Strauss [Str64], which is sometimes referred as Shamir's trick in the literature. Finally, we write $|G|$ to denote the number of bits needed to represent an element in G.

We start the comparison by stating the efficiency properties of our scheme. For reasons of efficiency we consider that our scheme should be built upon a group of points $\mathcal{G}$ of prime order q of a suitable elliptic curve. A signature on our scheme consists of two elements in $\mathcal{G}$ and one element in $\mathbb{Z}_q$. To sign, one needs to compute one exponentiation (a small number of multiplications in $\mathbb{Z}_q$ can be ignored). To verify, one needs to compute one multi-exponentiaton mexp(3). For the sake of concreteness, we fix $|\mathcal{G}| \approx |\mathbb{Z}_q| \approx 256$ bits for a security level equivalent to a 128-bit symmetric key for AES (cf. [ECR]). Following estimations by Brumley [Bru06], computing 1 mexp(3) has a cost of about 1.5 times that of a single exponentiation.

In the first category of our classification we find factoring-based schemes. As it is well-known, key sizes for factoring-based schemes are much larger than key sizes for ECDL-based schemes for an equivalent security level. This implies in particular that performing a group exponentiation in a factoring-based scheme, where $G = \mathbb{Z}_n^*$ for an RSA-modulus n, is more expensive than an exponentiation in an ECDL-based scheme. For instance, for the 128-bit security level, $|\mathbb{Z}_n| \approx 3072$ bits. The shortest signature size for existing factoring-based IBS schemes is equivalent to representing two elements in $|\mathbb{Z}_n|$ (cf. [BNN04]), and thus signature size is considerably bigger than in our scheme. They do present a higher computational cost in signing and verifying, since the most efficient algorithms require at least two exponentiations in $\mathbb{Z}_n$ for signing, and one mexp(2) in verifying. Computing one mexp(2) in $\mathbb{Z}_n$ is more costly than computing one mexp(3) in $\mathcal{G}$ due to the larger key size in factoring-based schemes.

In the second place, let us consider previous provably secure ECDL-based schemes. A comparison to previous schemes yields that our scheme is the more efficient in all three features, namely, in signature size, signing cost and verifying cost. Indeed, our scheme enjoys the same efficiency as the scheme named as Beth IBS scheme in [BNN04], which to our knowledge was the most efficient ECDL-based IBS scheme to date. In sharp contrast to our case, the security of the Beth IBS scheme is still unproven (cf. [BNN04]).

It remains to compare our scheme with existing pairing-based schemes. To this end, we need to recall some facts about pairings (see [BSS05] for a comprehensive introduction). Let $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{G}_T$ be finite Abelian groups in which the discrete logarithm is believed to be hard. A *pairing* is a bilinear function $e : \mathcal{G}_1 \times \mathcal{G}_2 \rightarrow \mathcal{G}_T$. Let $\mathcal{G}_1 = E(\mathbb{F}_p)$ denote an elliptic curve over the finite field $\mathbb{F}_p$. Let the order of $\mathcal{G}_1$ be divisible by a prime r such that r also divides $p^\alpha - 1$, where α is the order of p in $\mathbb{Z}_r^*$ and is called the MOV embedding degree. The modified Tate pairing $e(\cdot, \cdot)$, which is the bilinear map usually recommended, takes values in the subgroup $\mathcal{G}_T$ of $\mathbb{F}_{p^\alpha}^*$ of order r and is defined in two ways, depending on whether E is a supersingular or ordinary curve.

In the supersingular case $\mathcal{G}_1 = \mathcal{G}_2 = E(\mathbb{F}_p)$. For supersingular curves the best parameter choices are in characteristic 3, and in this case $\alpha = 6$ and $|\mathbb{F}_{p^\alpha}^*| \approx 3072$. As a consequence, $|\mathcal{G}_1| \geq 3072/6 = 512$, since in groups equipped with a pairing an algorithm solving the (finite-field) discrete logarithm in $\mathcal{G}_T$ can be used to compute the ECDL in $\mathcal{G}_1$.

Ordinary curves can also be used to implement pairings, but in this case $\mathcal{G}_2$ is set to be a subgroup of $E(\mathbb{F}_{p^\alpha})$. Several techniques have been presented [BKLS02,SB06,GPS06] in order to improve efficiency and bandwidth. For instance, the twist of the curve $E(\mathbb{F}_{p^{\alpha/2}})$ can be used to generate the group $\mathcal{G}_2$, and therefore there is a 50% save in bandwidth. Here, typical parameters would be $|\mathcal{G}_1| \geq 256$, $\alpha = 12$ and $|\mathcal{G}_2| \geq 3072/2 = 1536$.

For pairings over supersingular curves the shortest length signature schemes consist of two elements in $\mathcal{G}_1$, which amounts to 1024 bits. This is in favor of our scheme, since our signatures are 768 bits long. Regarding computational time, the fact that groups defined from supersingular curves have a longer representation immediately implies that our scheme surpasses in efficiency any previous IBS scheme in this category.

In the ordinary curves category, a more detailed comparison is needed. Here, the shortest signatures consist of two elements in $\mathcal{G}_1$, as in [Her06], or alternatively one element in $\mathcal{G}_1$ and one element in $\mathbb{Z}_q$, where $q = |\mathcal{G}_1|$, as in the scheme [BLMQ05] by Barreto et al. In both schemes, the signature size is smaller than ours in about 256 bits, which is in our disadvantage. However, our scheme performs better in computational time. We discuss that below.

Herranz's scheme signing requires 1 exponentiation in $\mathcal{G}_1$ plus 1 hashing onto $\mathcal{G}_1$, which is generally a computationally expensive operation. In contrast, our scheme only needs 1 exponentiation in $\mathcal{G}_1$, and therefore our signing algorithm is more efficient than Herranz's scheme. As for private key extraction, Herranz's algorithm computes 1 exponentiation in $\mathcal{G}_2$. Granger, Page and Smart [GPS06] estimate that in our concrete case exponentiating in $\mathcal{G}_2$ takes 3 times as much resources as exponentiating in $\mathcal{G}_1$. As for verifying, Herranz's scheme needs to compute 1 exponentiation in $\mathcal{G}_1$, 1 hash-to-group operation and 2 pairings. The latter operation can be replaced by the product of two pairings, which by a trick by Granger and Smart [GS06], has a cost of about 1.46 times that of a single pairing. On the other hand, our scheme needs to compute 1 mexp(3) for verification, which following estimations by Brumley [Bru06], has a cost of about 1.5 times that of a single exponentiation. With regards to pairings, [GPS06] estimates that for the 128-bit security level computing a pairing is 17 times more expensive than exponentiating in $\mathcal{G}_1$. Therefore, Herranz's scheme verification algorithm is 17 times slower than ours for the 128-bit security level.

Barreto et al scheme needs to compute one exponentiation in $\mathcal{G}_1$ and one exponentiation in $\mathcal{G}_T$ for signing. For verification, it computes 1 exponentiation in $\mathcal{G}_2$, 1 exponentiation in $\mathcal{G}_T$ plus 1 pairing. Exponentiating in $\mathcal{G}_2$ and $\mathcal{G}_T$ requires a higher computational cost than computing an exponentiation in $\mathcal{G}_1$. Granger et al [GPS06] estimate that in our concrete case exponentiating G_T takes 3 times as much resources as exponentiating in $\mathcal{G}_1$. This implies that our scheme is at least 4 times more efficient for signing, and at least 23 times more efficient for verifying than Barreto et al scheme.

Figure 1 summarizes the comparison between our scheme and [BLMQ05,Her06]. The time unit is set to be the time needed to compute 1 exponentiation in $\mathcal{G}_1$. As a result, our scheme outperforms previous schemes in the pairing-based category, except for its signature size, where the benchmark is still [BLMQ05,Her06]. The new scheme drastically outperforms the other schemes in verification time. It is also worth noting our scheme relies on the discrete logarithm problem, while Herranz and Barreto et al. schemes rely on variations of the Computational Diffie-Hellman problem over pairing groups.

Scheme	Key derivation	Signature size	Sign	Verification
Ours (Section 3)	1	768 bits	1	1.5
Barreto et al ([BLMQ05])	1	512 bits	4	23
Herranz ([Her06])	3	512 bits	> 1	17

Fig. 1. Efficiency comparison among [BLMQ05,Her06] schemes and our proposal for the 128-bit security level.

6 Conclusion

In this work we have presented a new identity-based signature scheme. We have proven it secure under the discrete logarithm assumption using the random oracle methodology. Our new scheme outperforms in computational cost and underlying security assumption every previously proposed provably secure identity-based signature scheme. Moreover, our signatures have also smaller bit complexity than any other provably secure scheme in the literature, with the exception of [BLMQ05,Her06]. The properties enjoyed by our new scheme make it specially suited for deployment in resource constrained devices, e.g. wireless sensor networks.

References

BFPW07. Alexandra Boldyreva, Marc Fischlin, Adriana Palacio, and Bogdan Warinschi. A closer look at PKI: Security and efficiency. In *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 458–475, 2007.

BKLS02. P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 354–368, 2002.

BLMQ05. Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532, 2005.

BN06. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security (CCS'06)*, pages 390–399, New York, NY, USA, 2006. ACM.

BNN04. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286, 2004. The full version appears in Cryptology ePrint Archive: Report 2004/252.

BPW03. Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. Cryptology ePrint Archive, Report 2003/096, 2003. `http://eprint.iacr.org/`.

Bru06. Billy Bob Brumley. Efficient three-term simultaneous elliptic scalar multiplication with applications. In Viiveke Fåk, editor, *Proceedings of the 11th Nordic Workshop on Secure IT Systems—NordSec '06*, pages 105–116, Linköping, Sweden, October 2006.

BSNS05. Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Certificateless public key encryption without pairing. In *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2005.

BSS05. I.F. Blake, G. Seroussi, and N. Smart. *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Note Series.* Cambridge University Press, 2005.

CC02. Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In *Public Key Cryptography 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30, 2002.

CJT04. Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik. Secret handshakes from ca-oblivious encryption. In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 293–307. Springer, 2004.

ECR. ECRYPT. Ecrypt yearly report on algorithms and key lengths (2006). `http://www.ecrypt.eu.org/documents/D.SPA.21-1.1.pdf`. revision 1.1, 29 January 2007.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, 1987.

GPS06. Robert Granger, Dan Page, and Nigel P. Smart. High security pairing-based cryptography revisited. In *ANTS*, volume 4076 of *Lecture Notes in Computer Science*, pages 480–494, 2006.

GQ90. Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *CRYPTO 1988*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, 1990.

GS06. R Granger and N.P. Smart. On computing products of pairings. Cryptology ePrint Archive, Report 2006/172, 2006. `http://eprint.iacr.org/`.

Her06. Javier Herranz. Deterministic identity-based signatures for partial aggregation. *Comput. J.*, 49(3):322–330, 2006.

Hes03. Florian Hess. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2003.

Oka93. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, 1993.

PS00. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

SB06. Michael Scott and Paulo S. L. M. Barreto. Generating more mnt elliptic curves. *Des. Codes Cryptography*, 38(2):209–217, 2006.

Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

Sha85. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 1984*, volume 196 of *LNCS*, pages 47–53, 1985.

SOK00. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, 2000. Oiso, Japan.

Str64. Strauss. Addition chains of vectors. *American Mathematical Monthly*, 71(7):806–808, Aug.-Sept. 1964.