Exponent Group Signature Schemes and Efficient Identity Based Signature Schemes Based on Pairings

F. Hess

Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB
florian@cs.bris.ac.uk

Abstract. We describe general exponent group signature schemes and show how these naturally give rise to identity based signature schemes if pairings are used. We prove these schemes to be secure in the random oracle model. Furthermore we describe a particular identity based signature scheme which is quite efficient in terms of bandwidth and computing time, and we develop a further scheme which is not derived from an exponent group signature scheme. The realization of these schemes uses supersingular elliptic curves and the Tate pairing, which is more efficient than the Weil pairing. Finally we show that these schemes have a more natural solution, than Shamir's original scheme, to the "escrow" property that all identity based signature schemes suffer from.

Keywords: Exponent group signatures, identity based signatures, Weil pairing, Tate pairing, key escrow.

1 Introduction

Digital signatures are one of the most important security services offered by cryptography. In traditional public key signature algorithms the public key of the signer is essentially a random bit string picked from a given set. This leads to a problem of how the public key is associated with the physical entity which is meant to be performing the signing. In these traditional systems the binding between the public key and the identity of the signer is obtained via a digital certificate. As noticed by Shamir [14] it would be more efficient if there was no need for such a binding, in that the users identity would be their public key, more accurately given the users identity the public key could be easily derived using some public deterministic algorithm.

An identity based signature scheme based on the difficulty of factoring integers is given in [14], and it remained an open problem to develop an identity based encryption scheme. In 2001 two such schemes were given, the first by Cocks

[6] was based on the quadratic residuosity problem, whilst the second given by Boneh and Franklin [3] was based on the Weil pairing.

Originally the existence of the Weil pairing was thought to be a bad thing in cryptography. For example in [9] it was shown that the discrete logarithm problem in supersingular curves was reducible to that in a finite field using the Weil pairing. This led supersingular elliptic curves to be dropped from cryptographic use. The situation changed with the work of Joux [8], who gave a simple tripartite Diffie-Hellman protocol based on the Weil pairing on supersingular curves. Since Joux's paper a number of other applications have arisen, including an identity based encryption scheme [3], a general signature algorithm [4]. The extension to higher genus curves has also recently been fully explored in [7]. This new work has resulted in a rekindling of cryptographic interest in supersingular elliptic curves. Although most of the literature discusses these schemes in terms of the Weil pairing, in turns out that it is far more efficient to use the Tate pairing as we shall explain. In [13] an identity based public key signature algorithm is given which uses the Weil pairing. This algorithm is less efficient than ours both in terms of bandwidth and computational cost, in addition no proof of security is given for the scheme in [13]. Very recently and independently of this work another identity based signature scheme [11] has been proposed, however also without a formal proof of security. In this paper we present identity based signature schemes based on the Weil or Tate pairing different from [11, 13] with proof of security. We describe two schemes in detail which have slightly better communications overhead, but are around fifty percent more efficient in terms of computing resources than the system of [13].

The paper is organized as follows. We first describe general exponent group signature schemes and prove them to be secure in the random oracle model. Using pairings with certain properties we then show that these exponent group signature schemes can naturally be transformed into identity based signature schemes, thereby obtaining a new class of such signature schemes. Among these we describe a particularly efficient scheme in more detail. We also develop a further scheme which does not belong to this class. We then discuss how these schemes can be realized using elliptic curves and the Weil or Tate pairings. We mention ways of computing these pairings. It turns out that we can use supersingular elliptic curves for our purpose. Finally, we address issues regarding key escrow and the distribution of keys to multiple trust authorities.

2 Signature Schemes for General Exponent Groups

In this section we describe a generalization of public key discrete logarithm based signature schemes like the modified ElGamal or Schnorr schemes to general exponent groups.

Let $(G, +)$ and $(V, \cdot)$ denote groups of prime order l and let

$$\exp : G \to V$$

be an isomorphism. The group G is referred to as exponent group. The main example is $G = \mathbb{F}_l^+$, V the subgroup of $\mathbb{F}_q^\times$ generated by $\zeta_l = \zeta^{(q-1)/l}$ for ζ a generator of $\mathbb{F}_q^\times$ and $l \mid q-1$, and $\exp : x \mapsto \zeta_l^x$. Thus exp can be viewed as an exponentiation and the discrete logarithm problem is replaced by computing preimages under exp, or in other words, inverting exp. Clearly we have in mind that images under exp be easily computed while preimages should be infeasible to compute.

Let $a \in G \backslash \{0\}$ and $k \in \mathbb{F}_l^\times$. We define a hash function

$$h : \{0,1\}^* \times V \to \mathbb{F}_l^\times \times \mathbb{F}_l^\times .$$

There will be various choices for h as described later.

An exponent group signature scheme, based on an exponentiation and public keys, consists of the following three algorithms, **Setup**, **Sign** and **Verify**. There are two parties in such a system, the signer and verifier.

Setup : The signer creates a private-public key pair $(a, y) \in G\backslash\{0\} \times V\backslash\{1\}$ by choosing a random $a \in G\backslash\{0\}$. The he computes

$$y = \exp(a)$$

and publishes y.

Scheme 1.

Sign : To sign a message m the signer picks a random $k \in G\backslash\{0\}$ and then computes:

1. $r = \exp(k)$
2. $(v, w) = h(m, r)$
3. $u = av + kw$

The signature is the pair $(u, r) \in G \times V\backslash\{1\}$.

Verify : On receiving a message m and signature (u, r) the verifier computes:

1. $(v, w) = h(m, r)$
2. Accept the signature if and only if $\exp(u) = y^v r^w$.

That this verification equation holds for a valid signature follows from the following algebra:

$$\begin{aligned} \exp(u) &= \exp(av + kw) \\ &= \exp(a)^v \exp(k)^w \\ &= y^v r^w . \end{aligned}$$

There is a number of variations of this basic scheme. First of all, the signer and verifier could compute $(w, v) = h(m, r)$ instead of $(v, w) = h(m, r)$. If $G = \mathbb{F}_l$ they could additionally assign $h(m, r)$ to any of the six combinations $(u, v), (u, w), \ldots$ and the signer would solve for the remaining variable in $u = av + kw$. The signature consists then of the value of this remaining variable together with r. These variations are equally secure (in the random oracle model) since the respective tuples (u, v, w) are computationally indistinguishable, because of $a, k \neq 0$. The variations correspond to the well known six variations of the modified ElGamal scheme.

There is a number of possibilities for the definition of h. If $h_1 : \{0,1\}^* \to \mathbb{F}_l^\times$, $h_2 : V \to \mathbb{F}_l^\times$ and $h_3 : \{0,1\}^* \times V \to \mathbb{F}_l^\times$ are hash functions we can for example consider $h(m,r) := (h_1(m), h_2(r))$ or $h(m,r) := (h_3(m,r), 1)$. The choice $h(m,r) := (h(m), r)$ would for example not be admissible. The case

$$h(m,r) := (h_3(m,r), 1)$$

is of particular interest. Namely, in scheme 1 we then have $w = 1$ and the verification equation is $\exp(u) = y^v r$. This means we can solve $r = \exp(u) y^{-v}$. It is hence equivalent giving (u, v) as a signature instead of (u, r), which might take less memory. We obtain the following modified signing and verification steps, yielding a generalized Schnorr signature scheme.

Scheme 2.

Sign : To sign a message m the signer picks a random $k \in G \backslash \{0\}$ and then computes:

1. $r = \exp(k)$
2. $v = h_3(m, r)$
3. $u = av + k$

The signature is the pair $(u, v) \in G \times \mathbb{F}_l^\times$.

Verify : On receiving a message m and signature (u, v) the verifier computes:

1. $r = \exp(u) y^{-v}$
2. Accept the signature if and only if $v = h_3(m, r)$.

If we are given $\exp : G \to V$ we can derive other exponentiations in the following way. Let $g \in G \backslash \{0\}$. We define $\exp_g : \mathbb{F}_l \to V$ by $\exp_g(x) := \exp(xg)$. If we can compute preimages under $\exp_g$ we can compute preimages under exp. This implies that $\exp_g$ is at least as "secure" as exp. The security of the corresponding signature schemes is discussed later.

3 Identity based Signature Schemes from Pairings

In this section we describe how non-degenerate pairings can be used to obtain identity based signature schemes. There are two constructions. The first is to transform any exponent group signature scheme, based on public keys, into an identity based scheme. The second appears to be "conceptually new".

Let $e : G \times G \to V$ be a pairing which satisfies the following conditions

1. Bilinear: $e(x_1 + x_2, y) = e(x_1, y)e(x_2, y)$ and $e(x, y_1 + y_2) = e(x, y_1)e(x, y_2)$.
2. Non-degenerate: There exists $x \in G$ and $y \in G$ such that $e(x, y) \neq 1$.

We also assume that $e(x, y)$ can be easily computed while, for any given random $b \in G$ and $c \in V$, it should be infeasible to compute $x \in G$ such that $e(x, b) = c$. We remark that the pairing e is not required to be symmetric or antisymmetric.

We define a hash function

$$H : \{0,1\}^* \to G \backslash \{0\}.$$

An identity based signature scheme consists of the following four algorithms, **Setup**, **Extract**, **Sign** and **Verify**. There are three parties in such a system, the trust authority (or TA), the signer and the verifier.

Setup : The TA picks a random element $P \in G\backslash\{0\}$ and a secret integer $t \in \mathbb{F}_l^\times$. The TA then computes

$$Q_{TA} = tP$$

and publishes (P, Q_{TA}). The value t is stored by the TA.

Extract : This algorithm is performed by the TA when a user requests the secret key corresponding to their identity. Suppose the user's identity is given by the string ID. The public key of the user is then given by

$$Q_{ID} = H(ID),$$

whilst the private key is computed by the TA as

$$S_{ID} = tQ_{ID},$$

and this value is given to the user.

3.1 Transforming Exponent Group Signature Schemes

Given a pairing as above we obtain an isomorphism $\exp : G \to V$ by letting $\exp(x) := e(x, P)$. The idea is to use this exponentiation in the schemes of section 2. We can indeed proceed as in the algorithms **Sign** and **Verify** given there, using this exponentiation exp and $a = S_{ID}$. The verifier has to check $e(u, P) = y^v r^w$, where y is the personal public key $y = \exp(a)$ of the signer ID. The key point is that

$$\begin{aligned} y &= e(a, P) \\ &= e(tQ_{ID}, P) \\ &= e(Q_{ID}, tP) \\ &= e(Q_{ID}, Q_{TA}). \end{aligned}$$

The verifier thus only needs the identity of the signer and the public key of the TA in order to verify the signature. There is no need at all for a personal public key. Thus

Theorem 1. *The exponent group signature schemes can be transformed into identity based signature schemes by using a suitable pairing.*

To be more precise we formulate the algorithms **Sign** and **Verify** of scheme 2 in the identity based setting.

Scheme 3.

Sign : To sign a message m the signer chooses an arbitrary $P_1 \in G \backslash \{0\}$, picks a random integer $k \in \mathbb{F}_l^{\times}$ and computes:

1. $r = e(P_1, P)^k$.
2. $v = h_3(m, r)$.
3. $u = vS_{ID} + kP_1$.

The signature is then the pair $(u, v) \in (G, \mathbb{F}_l^{\times})$.

Verify : On receiving a message m and signature (u, v) the verifier computes:

1. $r = e(u, P) \cdot e(Q_{ID}, -Q_{TA})^v$
2. Accept the signature if and only if $v = h_3(m, r)$.

We discuss some general performance enhancements for the scheme 3, similar remarks apply to scheme 1 in the identity based setting as well.

The signing operation can be optimized by the signer precomputing $e(P_1, P)$ for the P_1 of his choice, for example $P_1 = P$, and storing this value with the signing key. This means that the signing operation involves one exponentiation in the group V, one hash function evaluation and one simultaneous multiplication in the group G.

The verification operation requires one exponentiation in V, one hash function evaluation and two evaluations of the pairing. One of the pairing evaluations can be eliminated, if a large number of verifications are to be performed for the same identity, by precomputing $e(Q_{ID}, -Q_{TA})$.

The signature scheme 3 is also very efficient in terms of communication requirements. One needs to transmit one element of the group G and one element of $\mathbb{F}_l$.

3.2 Another Identity based Signature Scheme

The previous schemes used the bilinearity to express the personal public key via the identity and the trust authority's public key. Except for this the bilinearity had not been applied further. The following scheme now makes full use of the bilinearity of the pairing. Let

$$h' : \{0,1\}^* \times G \to \mathbb{F}_l^{\times}$$

be a hash function.

Scheme 4.

Sign : To sign a message m the signer picks a random integer $k \in \mathbb{F}_l^\times$ and then computes:

1. $r = kP$.
2. $v = h'(m, r)$.
3. $u = (v/k)S_{ID}$.

The signature is then the pair $(u, r) \in (G \backslash \{0\}, G \backslash \{0\})$.

Verify : On receiving a message m and signature (u, r) the verifier computes:

1. $v = h'(m, r)$
2. Accept the signature if and only if $e(u, r) = e(Q_{ID}, Q_{TA})^v$.

That this verification equation holds for a valid signature follows from the following algebra:

$$\begin{aligned} e(u, r) &= e((v/k)S_{ID}, kP) \\ &= e(S_{ID}, P)^v \\ &= e(Q_{ID}, Q_{TA})^v. \end{aligned}$$

Depending on G and V this scheme appears to be slightly more efficient than scheme 3.

4 Proofs of Security

4.1 Exponent Group Signature Schemes

Security for the exponent group signature schemes will be defined by adapting the standard definition of security against existential forgery under an adaptively chosen message attack, see for example [12]. The adversary A is assumed to be a polynomial time probabilistic Turing machine which takes as input the public key y of a user. The adversary's goal is to produce an existential forgery of a signature by the given user. To aid the adversary we allow her to query an oracle:

Signature Oracle : For any given message m and public key y this oracle will produce a signature from the user with public key y on the message m.

Of course the output of the adversary A should not be a signature which has been asked of its signature oracle. We shall call such an adversary against our signature scheme an adaptive adversary. One could extend the attack model by assuming that the adversary can at some point in the attack choose the identity on which they aim to obtain a forged signature. Extending the proof of the following theorem to this attack model is trivial, but results in a significant reduction in the tightness of the resulting security.

Theorem 2. *In the random oracle model, suppose that an adaptive adversary A exists against an exponent group signature scheme which makes q_S queries of its signature oracle, q_h queries of its random oracle h and which succeeds within time T of making an existential forgery with probability*

$$\epsilon > \frac{10(q_S+1)(q_S+q_h)}{l}.$$

Then there is another probabilistic algorithm which solves

$$\exp(a) = y$$

in a which succeeds in expected time

$$T' < \frac{120686 q_h T}{\epsilon}.$$

Proof. Let A be our adversary, we shall use A to construct another algorithm B_A which inverts exp. We shall model the hash function h as random oracle, and so we will need to keep lists of the oracle queries made.

Let the input to algorithm B_A be the public key y. Algorithm B_A chooses a random message m and, applying the Forking Lemma of Pointcheval and Stern [12], runs algorithm A with the same random tape, but with different outputs of the hash function h, until two forgeries of the same message are obtained:

$$(r, (v, w) = h(m, r), u)\,,$$
$$(r, (v', w') = h'(m, r), u')\,.$$

Algorithm B_A will succeed in producing such a pair of forgeries in expected time T' as follows from Theorem 13 of [12].

Since the value of r is the same in both forgeries we have the equation

$$\exp(u - (w/w')u') = y^{v-(w/w')v'}.$$

Furthermore, as the hash values are random we expect the vectors (v, w) and (v', w') to be $\mathbb{F}_l$-linearly independent, and thus $v - (w/w')v' \neq 0$. Hence, algorithm B_A can compute

$$y = \exp((v - (w/w')v')^{-1}(u - (w/w')u'))$$

thereby solving $\exp(a) = y$ in a in expected time T'.

All that remains is to explain how algorithm B_A will answer the signature oracle queries of algorithm A. Recall that such queries need to be simulated since the secrets needed to perform the queries in a real attack are not available to algorithm B_A.

Signature Oracle Queries : On input of a message m to sign we generate random $u, u' \in G\backslash\{0\}$ and random linearly independent vectors (v, w), (v', w')

in $\mathbb{F}_l^\times \times \mathbb{F}_l^\times$. We then compute $\lambda = (w-1)/w'$ and $r = \exp(u - \lambda u')/y^{v-\lambda v'}$. We remark that r is a random element in V. If $r = 1$ a new vector (v', w') is chosen until $r \in V\backslash\{1\}$. We define the hash value $h(m, r) := (v, w)$ and return the signature (u, r).

Let $\exp_g$ be the exponentiation derived from exp and $g \in G\backslash\{0\}$. Inverting $\exp_g$ is at least as hard as inverting exp. By Theorem 2 we can conclude that the signature schemes for $\exp_g$ are at least as secure as the corresponding signature schemes for exp.

4.2 Identity based Signature Schemes

The security of the identity based signature schemes obtained from the exponent group signature schemes is derived from their security. For the security of scheme 4 we obtain a weaker result which follows quite similarly.

The adversary A is assumed to be a polynomial time probabilistic Turing machine which takes as input the data

$$(P, Q_{TA}, Q_{ID})$$

where P is the "base point", Q_{TA} is the trust authority's public key and Q_{ID} is the public key of a user. The adversary's goal is to produce an existential forgery of a signature by the given user. To aid the adversary we allow her to query two oracles:

Extraction Oracle : For any given identity $ID' \neq ID$ this oracle will produce the corresponding secret key $S_{ID'}$.

Signature Oracle : For any given message m and identity ID this oracle will produce a signature from the user with identity ID on the message m.

Of course the output of the adversary A should again not be a signature which has been asked of its signature oracle.

Theorem 3. *In the random oracle model, suppose that an adaptive adversary A exists against scheme 3 (or any other identity based scheme derived from an exponent group signature scheme) which makes q_S queries of its signature oracle, q_h queries of its random oracle h and which succeeds within time T of making an existential forgery with probability*

$$\epsilon > \frac{10(q_S+1)(q_S+q_h)}{l}.$$

Then there is another probabilistic algorithm which solves

$$e(a, P) = e(Q_{ID}, Q_{TA})$$

in a which succeeds in expected time

$$T' < \frac{120686 q_h T}{\epsilon}.$$

Proof. The signature scheme is nothing else as an exponent group signature scheme with private key $a = S_{ID}$ and public key $y = e(S_{ID}, P)$. The theorem follows from Theorem 2 once we have explained how the oracles are simulated.

Extraction Oracle Queries : Given an identity ID' the extraction oracle computes a random $\lambda \in \mathbb{F}_l^\times$, $Q_{ID'} = \lambda P$ and $S_{ID'} = \lambda Q_{TA}$. Then it defines $H(ID') = Q_{ID'}$ and returns $S_{ID'}$. As usual these values are stored so that the extraction oracle returns the same value when queried for the same ID' again.

Signature Oracle Queries : The signature oracle can be simulated using the signature oracle of the previous section queried for m and $y = e(Q_{ID}, Q_{TA})$ where $Q_{ID} = H(ID)$. We could in fact allow queries for any Q and $y = e(Q, Q_{TA})$.

For scheme 4 we do not give a formal proof of security. The extraction and signature oracles can be simulated in the random oracle model as follows. For the extraction oracle we can use the above simulation. The signature oracle computes random $\lambda, \mu \in \mathbb{F}_l^\times$, sets $r = \lambda Q_{TA}$, $u = \mu Q_{ID}$ and defines $h'(m, r) = \mu\lambda$. Again these values are stored appropriately. If Q_{ID} is not known it is computed using the extraction oracle. These considerations show that an active adversary does not have better chances to forge a signature than a passive adversary.

By the Forking Lemma an algorithm B_A using the adversary A may obtain two forgeries of the same message m

$$(r, v = h'(m, r), u)\,,$$
$$(r, v' = h''(m, r), u')\,,$$

by running it with the same random tape but different outputs of the hash function h. Since the value of r is the same in both forgeries we then have the equation

$$e(u - u', r) = e(Q_{ID}, Q_{TA})^{v - v'}.$$

Now r must have been computed by B_A or A before the oracle query $h'(m, r)$ was made. But these oracle queries returned random results. Thus B_A is able to solve the following problem. After a finite computation data including an $r \in G$ is determined. Then

$$e(a, r) = c$$

is solved in a for a random $c \in V$.

One would naturally expect that solving $e(a, r) = c$ is uniformly as hard as solving $e(a, P) = c$ for a prescribed P, since G is cyclic.

Given any $r \in G \backslash \{0\}$ the difficulty of inverting the pairing $e(\cdot, r)$ can be related to problems in G and V as follows. Assume we have an efficiently computable isomorphism $i : V \to G$ which inverts the pairing e, that is $x = e(i(x), P)$. Let f be a generator of V. Then $g := e(i(f), i(f))$ is also a generator of V. Furthermore, $e(i(f^\lambda), i(f^\mu)) = g^{\lambda\mu}$. That is, given f^λ and f^μ we have computed $g^{\lambda\mu}$ and have hence solved an instance of the weak Diffie-Hellman problem in V. By [16] we can now conclude that inverting the pairing $e(\cdot, r)$ is at least as hard as solving the Diffie-Hellman problem in both V and G. We remark that the reduction to solving an instance of the Diffie-Hellman problem in V can also be obtained by a direct reasoning in the proof of Theorem 2 and Theorem 3.

5 Realization of the Identity based Signature Schemes

In order use the identity based schemes described so far we need to find suitable groups G, V and pairings $e : G \times G \to V$. These groups will be provided by finite fields and elliptic curves over finite fields, and the pairings will be derived from the Weil or Tate pairing.

More precisely G will be a point subgroup on an elliptic curve over a finite field and V a subgroup of the cyclic group of a larger finite field. We remark that elements in G can be represented in compressed form. Also, in scheme 3 the signature consists of $v \in \mathbb{F}_l$ instead of $r \in V$, resulting in a more bandwidth efficient scheme.

5.1 The Weil and Tate Pairings on Elliptic Curves

We shall summarize the properties we require of the Weil and Tate pairings, much of the details can be found in [3], [9] and [15]. We present both pairings since the Tate pairing is more efficient to compute than the Weil pairing.

Let E be an elliptic curve defined over $\mathbb{F}_q$ and let G be a subgroup of $E(\mathbb{F}_q)$ of prime order l. For simplicity we will assume that $l^2 \nmid \#E(\mathbb{F}_q)$ so $G = E(\mathbb{F}_q)[l]$. We define α to be the smallest integer such that

$$l \mid (q^\alpha - 1).$$

The full l-torsion group $E[l]$ is defined over a unique minimal extension field $\mathbb{F}_{q^k}$,

$$E[l] \subseteq E(\mathbb{F}_{q^k}).$$

In practical implementations we will require k and α to be small. Let $\hat{G}$ be a subgroup of $E[l]$ such that $E[l] = G \oplus \hat{G}$.

The Weil pairing is a non-degenerate, bilinear and antisymmetric pairing

$$e_l : E[l] \times E[l] \to \mathbb{F}_{q^k}^\times .$$

We cannot use it immediately for our purpose since $e_l(P, Q) = 1$ for any $P, Q \in G$. One possibility is to consider an (injective) non $\mathbb{F}_q$-rational endomorphism $\phi : G \to \hat{G}$. We define

$$e : G \times G \to \mathbb{F}_{q^k}, \quad e(P, Q) := e_l(P, \phi(Q)).$$

This yields a pairing of the required properties since P and $\phi(Q)$ are linearly independent. Such non rational endomorphisms are known to exist for supersingular elliptic curves. They do not exist for ordinary elliptic curves since the

Frobenius acts trivially on G. From the non degeneracy of the Weil pairing we have $k \geq \alpha$.

The Tate pairing is a non-degenerate, bilinear pairing

$$t_l : E(\mathbb{F}_{q^\alpha})[l] \times E(\mathbb{F}_{q^\alpha})/lE(\mathbb{F}_{q^\alpha}) \to \mathbb{F}_{q^\alpha}^\times/(\mathbb{F}_{q^\alpha}^\times)^l.$$

For $\alpha = 1$ we have $E(\mathbb{F}_q)[l] = G$ and $E(\mathbb{F}_q)/lE(\mathbb{F}_q) \cong G$. Using this isomorphism we can define

$$e : G \times G \to \mathbb{F}_{q^\alpha}^\times, \quad e(P,Q) := t_l(P,Q)^{(q-1)/l}.$$

For $\alpha > 1$ we have $E(\mathbb{F}_{q^\alpha})[l] = E[l]$, $E(\mathbb{F}_{q^\alpha})/lE(\mathbb{F}_{q^\alpha}) \cong E[l]$ and $k = \alpha$ from the non-degeneracy of t_l. Here we again need a non rational endomorphism $\phi : G \to \hat{G}$ since t_l is trivial on G. Using the isomorphism we define

$$e : G \times G \to \mathbb{F}_{q^\alpha}^\times, \quad e(P,Q) := t_l(P,\phi(Q))^{(q^\alpha-1)/l}.$$

Both cases yield pairings with the required properties.

If there are no non rational endomorphisms we could use any group homomorphism $\phi : G \to \hat{G}$ defined by $P \mapsto \hat{P}$ for an arbitrary $\hat{P} \in \hat{G}\backslash\{0\}$. The computation of the pairing in the signature schemes only requires the evaluation of ϕ at P and Q_{TA} which means that the two additional points $\hat{P} = \phi(P)$ and $\hat{Q}_{TA} = \phi(Q_{TA})$, to be computed by the trust authority, have to be publicly known.

The Weil and Tate pairings are efficiently computable by an unpublished, but much referenced, algorithm of Miller [10].

Suppose given $P, Q \in G$ we wish to compute $e_l(P,\phi(Q))$ or $t_l(P,\phi(Q))$. We first compute, via Miller's algorithm, the functions f_P and $f_{\phi(Q)}$ whose divisors are given by

$$(f_P) = l(P+X) - l(X)$$

and

$$(f_{\phi(Q)}) = l(\phi(Q)) - l(\mathcal{O}),$$

where $X \in G$ is randomly chosen such that $\#\{\mathcal{O}, \phi(Q), X, P+X\} = 4$. Note, that since $P \in G \subseteq E(\mathbb{F}_q)$ we have

$$f_P \in \mathbb{F}_q(x,y)$$

whilst since $\phi(Q) \in \hat{G}$ we have

$$f_{\phi(Q)} \in \mathbb{F}_{q^r}(x,y)$$

where $r = k$ or $r = \alpha$ respectively. This means that computing f_P is easier than computing $f_{\phi(Q)}$.

The Weil pairing is then given by

$$e_l(P,\phi(Q)) = \frac{f_P((\phi(Q)) - (\mathcal{O}))}{f_{\phi(Q)}((P+X) - (X))}.$$

The Tate pairing is computed via

$$t_l(P, \phi(Q)) = f_P((\phi(Q)) - (\mathcal{O})).$$

We see that not only is the Tate pairing easier to compute, since we do not need to compute $f_{\phi(Q)}$, but the single function we need to compute, namely f_P, is easier to compute than $f_{\phi(Q)}$. These facts together make computing the Tate pairing around fifty percent more efficient than the Weil pairing. On the other hand the value of the Tate pairing has to be raised to the power of $(q^\alpha - 1)/l$ to obtain the final result, and $f_{\phi(Q)}$ can be computed as $\phi(f_Q)$ using much more operations over $\mathbb{F}_q$ than over $\mathbb{F}_{q^k}$.

5.2 Supersingular Elliptic Curves

As we have seen we can use supersingular elliptic curves in order to obtain the required pairings for the identity based signature schemes. The following table lists a number of examples, the parameter α and a non rational endomorphism ϕ.

Field	Curve	$\#E$	α	ϕ
$\mathbb{F}_{2^p}$	$y^2 + y = x^3$	$2^p + 1$	2	$(x, y) \to (x + 1, y + x + \xi)$
$\mathbb{F}_{2^p}$	$y^2 + y = x^3 + x$	$2^p + 1 + t_2(p)$	4	$(x, y) \to (\xi^2 x + \zeta^2, y + \xi^2 \zeta x + \mu)$
$\mathbb{F}_{2^p}$	$y^2 + y = x^3 + x + 1$	$2^p + 1 - t_2(p)$	4	$(x, y) \to (\xi^2 x + \zeta^2, y + \xi^2 \zeta x + \mu)$
$\mathbb{F}_{3^p}$	$y^2 = x^3 + x$	$3^p + 1$	2	$(x, y) \to (-x, iy)$
$\mathbb{F}_{3^p}$	$y^2 = x^3 - x + 1$	$3^p + 1 + t_3(p)$	6	$(x, y) \to (-x + \tau_1, iy)$
$\mathbb{F}_{3^p}$	$y^2 = x^3 - x - 1$	$3^p + 1 - t_3(p)$	6	$(x, y) \to (-x + \tau_{-1}, iy)$
$\mathbb{F}_p$	$y^2 = x^3 + b$	$p + 1$	2	$(x, y) \to (\xi x, y)$
$\mathbb{F}_p$	$y^2 = x^3 + ax$	$p + 1$	2	$(x, y) \to (-x, iy)$

Here p denotes a prime ≥ 5 and

$$t_2(p) = \begin{cases} 2^{(p+1)/2} & \text{for } p \equiv \pm 1, \pm 7 \bmod 24, \\ -2^{(p+1)/2} & \text{for } p \equiv \pm 5, \pm 11 \bmod 24, \end{cases}$$

$$t_3(p) = \begin{cases} 3^{(p+1)/2} & \text{for } p \equiv \pm 1 \bmod 12, \\ -3^{(p+1)/2} & \text{for } p \equiv \pm 5 \bmod 12. \end{cases}$$

Furthermore, ϕ is a non rational endomorphism with

$$\begin{array}{ll} \xi^2 + \xi + 1 = 0, & \zeta^4 + \zeta + \xi + 1 = 0, \\ \mu^2 + \mu + \zeta^6 + \zeta^2 = 0, & \tau_s^3 - \tau_s - s = 0, \end{array}$$

and $i^2 + 1 = 0$. For $\xi \notin \mathbb{F}_p$ we need $p \equiv 2 \bmod 3$ and for $i \notin \mathbb{F}_p$ we need $p \equiv 3 \bmod 4$.

6 Key Escrow

One criticism against identity based signature schemes, as opposed to identity based encryption schemes, is that the TA has access to the users private key and hence can sign messages as if they came from the user. This escrow facility is deemed a great draw back for signature schemes, whereas one could debate that such an escrow facility for encryption may be desirable.

All previous identity based signature schemes have this in built escrow property. However, we shall show in a moment that by using multiple TAs we can reduce this threat from escrowing the private key. First we shall explain why using multiple TAs in Shamir's original scheme [14] is not very efficient.

Recall, in Shamir's scheme the TA has a RSA modulus N for which they know the corresponding factors, but no user is allowed to know the factors. The TA publishes N and an associated public exponent e. In the key extraction phase the TA computes

$$g = ID^d \pmod{N}$$

for the user, where

$$e \cdot d = 1 \pmod{\phi(N)}.$$

The value g is the users private key. To produce a signature (t, s) on a message m, the user computes

$$t = r^e \pmod{N},$$
$$s = g \cdot r^{H(m,t)} \pmod{N},$$

where r is a random element of $(\mathbb{Z}/N\mathbb{Z})^*$ and H a hash function. To verify a message one checks whether the equation

$$s^e = ID \cdot t^{H(m,t)} \pmod{N}$$

holds. Clearly with this scheme, to split the TA's key amongst a set of TAs we need to produce an RSA modulus N and a public exponent e such that no individual TA knows the factors of N and each TA has a share d_i of the private exponent d. Protocols exist for this problem, see for example [1], [2] and [5], but they are usually relatively inefficient.

For the identity based signature schemes described in this paper the situation is different as there is a natural, simple and elegant way to split the TAs master key into a set of shares. This is because our security is based on discrete logarithms rather than some factoring assumption. Suppose we have n TAs, denoted TA_i. We can now trivially distribute the master secret t among the n trusted authorities. Each TA now generates their own private key t_i and publishes

$$Q_{TA_i} = [t_i]P.$$

A user then obtains a share of the its private key from each TA via

$$S_{ID}^{(i)} = [t_i]Q_{ID}.$$

The user's secret key is then computed via

$$S_{ID} = \sum_{i=1}^{n} S_{ID}^{(i)}$$

with the corresponding value of Q_{TA} computed from

$$Q_{TA} = \sum_{i=1}^{n} Q_{TA_i}.$$

For the trusted authorities to determine the users private key they would then all need to collude, providing a so-called (n, n)-threshold secret sharing scheme.

There is the possibility of one TA cheating and not responding with the correct value of $S_{ID}^{(i)}$ for a given users key extraction request. This would have the effect of producing an invalid private key for the end user. The end user would like to determine which of the TAs has supplied the incorrect value. This can be done in one of two ways

1. The user tries to sign and verify a message using only the data provided by each TA in turn. This method clearly requires $O(n)$ signing operations.
2. The user could detect the incorrect value by forming and checking the various subkeys using a binary search method. This is a technique which will require $O(\log n)$ signing operations to determine the incorrect value of $S_{ID}^{(i)}$.

7 Conclusion

We have described exponent group signature schemes and how these naturally give rise to identity based signature schemes when pairings are used. These schemes are provably secure in the random oracle model. Furthermore we have described a particular identity based signature scheme which is quite efficient in terms of bandwidth and computing time, and have developed a further scheme which is not derived from an exponent group signature scheme. The realization of this scheme uses supersingular elliptic curves and the Tate pairing, which is more efficient than the Weil pairing. Finally we have discussed key escrow and the distribution of keys to multiple trust authorities.

8 Acknowledgements

I would like to thank D. Kohel, J. Malone-Lee and especially N. P. Smart for helpful discussions.

References

1. S. Blackburn, S. Blake-Wilson, M. Burmester and S. D. Galbraith. Shared Generation of Shared RSA Keys. Preprint, 1998.

2. D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. In *Advances in Cryptology - CRYPTO '97*, Springer-Verlag LNCS 1294, 425–439, 1997.
3. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology - CRYPTO 2001*, Springer-Verlag LNCS 2139, 213–229, 2001.
4. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology - ASIACRYPT 2001*, Springer-Verlag LNCS 2248, 514–532, 2001.
5. C. Cocks. Split knowledge generation of RSA parameters. In *Cryptography and Coding*, Springer-Verlag LNCS 1355, 89–95, 1997.
6. C. Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding*, Springer-Verlag LNCS 2260, 360–363. 2001.
7. S. D. Galbraith. Supersingular curves in cryptography. In *Advances in Cryptology - ASIACRYPT 2001*, Springer-Verlag LNCS 2248, 495-513, 2001.
8. A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Algorithmic Number Theory Symposium, ANTS-IV*, Springer-Verlag LNCS 1838, 385–394, 2000.
9. A. J. Menezes, T. Okamoto and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Info. Th.*, **39**, 1639–1646, 1993.
10. V. Miller. Short programs for functions on curves. Unpublished manuscript, 1986.
11. K. G. Paterson. ID-based signatures from pairings on elliptic curves *IACR preprint server*, 2002.
12. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13**, 361–396, 2000.
13. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, 2000.
14. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO '84*, Springer-Verlag LNCS 196, 47–53, 1984.
15. J. H. Silverman. *The Arithmetic of Elliptic Curves.* GTM 106, Springer-Verlag, 1986.
16. E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In *Advances in Cryptology - EUROCRYPT 2001*, Springer-Verlag LNCS 2045, 195-210, 2001.