Efficient Identity Based Signature Schemes based on Pairings

Florian Hess

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

Abstract. We develop an efficient identity based signature scheme based on pairings whose security relies on the hardness of the Diffie-Hellman problem in the random oracle model. We describe how this scheme is obtained as a special version of a more general generic scheme which yields further new provably secure identity based signature schemes if pairings are used. The generic scheme also includes traditional public key signature schemes. We further discuss issues of key escrow and the distribution of keys to multiple trust authorities. The appendix contains a brief description of the relevant properties of supersingular elliptic curves and the Weil and Tate pairings.

Keywords: 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 [18] 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 [18], and it remained an open problem to develop an identity based encryption scheme. In 2001 two such schemes were given, the first by Cocks [7] was based on the quadratic residuosity problem, whilst the second given by Boneh and Franklin [3] was based on the bilinear Diffie-Hellman problem with respect to a pairing, e.g. the Weil pairing.

Originally the existence of the Weil pairing was thought to be a bad thing in cryptography. For example in [10] it was shown that the discrete logarithm problem in supersingular elliptic 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 [9], 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] and a general signature algorithm [4]. The extension to higher genus curves and abelian varieties has also recently been fully explored in [8, 16]. This new work has resulted in a rekindling of cryptographic interest in supersingular elliptic curves.

In [17] an identity based public key signature algorithm is given which uses the Weil pairing and other identity based signature schemes [5, 14] have recently been proposed.

In this paper we present an identity based signature scheme whose security is based on the Diffie-Hellman problem in the domain of the pairing. Furthermore, we describe a generic scheme in a more general underlying situation and relate its security to a computational problem. Instantiations of this scheme using pairings then yield further new identity based signature schemes with security again based on the above Diffie-Hellman problem. These schemes are different from [5, 14, 17] and we argue that they can offer advantages over those schemes. Furthermore it appears possible that there are other instantiations of the general scheme, using for example RSA one way functions instead of pairings similar to Shamir's scheme [18]. Finally we address issues regarding key escrow and the distribution of keys to multiple trust authorities and discuss in the appendix how our schemes can be realized using elliptic curves and the Weil or Tate pairings.

2 An Identity Based Signature Scheme

Let $(G, +)$ and $(V, \cdot)$ denote cyclic groups of prime order l, $P \in G$ a generator of G and 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. Furthermore we define the hash functions

$$h : \{0,1\}^* \times V \to (\mathbb{Z}/l\mathbb{Z})^\times, \quad H : \{0,1\}^* \to G^*$$

where $G^* := G \backslash \{0\}$. We also abbreviate $V^* := V \backslash \{1\}$.

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

Scheme 1.

Setup : The TA picks a random integer $t \in (\mathbb{Z}/l\mathbb{Z})^\times$, computes $Q_{TA} = tP$ and publishes Q_{TA} while t is kept secret.

Extract : This algorithm is performed by the TA when a signer requests the secret key corresponding to their identity. Suppose the signer's identity is given by the string ID. The secret key of the identity is then given by $S_{ID} = tH(ID)$, which is computed by the TA and given to the signer.

The extraction step is typically done once for every identity and uses the same setup data for many different identities.

Sign : To sign a message m the signer chooses an arbitrary $P_1 \in G^*$, picks a random integer $k \in (\mathbb{Z}/l\mathbb{Z})^\times$ and computes:

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

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

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

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

It is straightforward to check that the verification equation holds for a valid signature.

We discuss some general performance enhancements for Scheme 1. The signing operation can be further optimized by the signer precomputing $e(P_1, P)$ for a P_1 of his choice, for example $P_1 = S_{ID}$, and storing this value with the signing key. For $P_1 = S_{ID}$ this means that the signing operation involves one exponentiation in the group V, one hash function evaluation and one multiplication involving an element 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 $e(H(ID), -Q_{TA})$.

For the security of Scheme 1 we have the following theorem. The attack model is explained in section 5.

Theorem 1. *In the random oracle model, suppose that an adaptive adversary A exists which makes at most $n_1 \geq 1$ queries of an identity hash and extraction oracle, at most $n_2 \geq 1$ queries of a message hash and signature oracle and which succeeds within time T_A of making an existential forgery with probability*

$$\varepsilon_A \geq \frac{a\, n_1 n_2^2}{l}$$

for some constant $a \in \mathbb{Z}^{\geq 1}$*. Then there is another probabilistic algorithm* C *and a constant* $c \in \mathbb{Z}^{\geq 1}$ *such that* C *solves the Diffie-Hellman problem with respect to*

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

on input of any given $R \in G^*$*, in expected time*

$$T_C \leq \frac{c\, n_1 n_2 T_A}{\varepsilon_A}.$$

Proof. Follows from Theorem 2 and Example 1 in section 5. See also the remarks after Theorem 2.

If G is represented as a subgroup of another group $\hat{G}$ the definition of H might be difficult in practice. Just as in section 4.3 and Theorem 4.7 of the full version of [3] one can relax this requirement on the hash function H that it produces elements in G by using admissible encodings, without affecting the above security theorem.

3 Comparison to other Identity Based Schemes

We compare Scheme 1 to the three most recent schemes [5, 14, 17] which are also based on pairings.

First, the setup and extraction steps are virtually identical for all four schemes and compatible with the identity based encryption scheme in [3]. Next we compare the efficiency. In the following table we denote by E an exponentiation in V, by M a scalar multiplication in G, by SM a simultaneous scalar multiplication of the form $\lambda P + \mu Q$ in G and by P a computation of the pairing. We do not take hash evaluations into account.

	Scheme 1	[5]	[14]	[17]
Signing	1E + 1M	2M	1M + 1SM	2M
Verifying	1E + 2P / 1E + 1P	1M + 2P	2E + 2P / 2E + 1P	3P / 2P
Signature	$G \times (\mathbb{Z}/l\mathbb{Z})^{\times}$	$G \times G$	$G \times G$	$G \times G$

The pairing computation is the operation which by far takes the most running time. The verifying step in Scheme 1 and the scheme of [14] can be optimized to 1E + 1P and 2E + 1P respectively if the same identities occur frequently, thus roughly taking half the runtime of [5]. The scheme in [17] requires 3P (2P in the optimized version) and is hence the slowest. In terms of communication requirements Scheme 1 is at least as efficient as the other two since an element in G requires at least as much memory as an element in $(\mathbb{Z}/l\mathbb{Z})^{\times}$ (remember $\#G = l$). Often one would actually need to apply special compression techniques in G.

In terms of provable security Scheme 1 and the scheme of [5] rely on the hardness of the Diffie-Hellman problem in G. There is no formal reduction to an underlying hard problem in [14, 17].

We conclude that Scheme 1 can offer advantages in runtime, communication requirements and provable security over the schemes [5, 14, 17].

4 Key Escrow

In this section we pause to discuss the issue of key escrow. The generalization of Scheme 1 will then be given in the next section.

One criticism against identity based signature schemes, as opposed to identity based encryption schemes, is that the trust authority has access to the signer's private key and hence can sign messages as if they came from the signer. 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 built in escrow property. By using multiple trust authorities however the threat from escrowing the private key can be reduced. For Shamir's original scheme [18] one would for example need to produce an RSA modulus N and a public exponent e such that no individual trust authority knows the factors of N and each trust authority has a share d_i of the private exponent d. Protocols exist for this problem, see for example [1], [2] and [6]. In order to distribute shares of the private key, a secret sharing scheme has also been applied in [3] in the identity based encryption setting. In general such a strategy would require a third party which computes the shares of the private key and distributes them to the trust authorities, together with some other data (e.g. appropriate Lagrange coefficients) allowing a signer to reconstruct his private key given the private partial information he obtains from sufficiently many trust authorities. In the case of an (n, n)-threshold secret sharing scheme the third party can however be dropped resulting in a similar, more efficient technique as follows.

Suppose we have the trust authorities TA_i for $1 \leq i \leq n$. We can trivially "distribute" the master secret t among the n trusted authorities in the following way. Each TA_i generates their own private key t_i independently of each other and publishes $Q_{TA_i} = t_i P$. The master private key is defined to be $t = \sum_{i=1}^{n} t_i$, a quantity completely unknown to any of the participants in the scheme. The master public key is $Q_{TA} = \sum_{i=1}^{n} Q_{TA_i}$. A signer obtains a share of its private key from each TA_i via $S_{ID}^{(i)} = t_i Q_{ID}$. The signer's secret key is then computed by the signer via $S_{ID} = \sum_{i=1}^{n} S_{ID}^{(i)}$. For the trust authorities to determine the signer's private key they would all need to collude, providing the (n, n)-threshold secret sharing scheme.

There is the possibility of one or several trust authorities cheating and not responding with the correct value of $S_{ID}^{(i)}$, with respect to Q_{TA_i}, for a given signer's key extraction request. This would have the effect of producing an invalid private key for the signer. A signer can check whether some trust authorities have responded with incorrect values, and which trust authorities have been involved, in one of the two following ways.

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

5 A Generic Signature Scheme

In this section we explain the general principle behind Scheme 1 thereby obtaining the generic scheme and further identity based signature schemes. We relate the security of these schemes to a computational problem which specializes to the Diffie-Hellman problem in G if pairings are used. Finally the key escrow problem is discussed briefly and some examples are given.

Let $(G, +)$, $(G_1, +)$ and $(V, \cdot)$ denote cyclic groups of prime order l and $p : G \to V$ an efficiently computable monomorphism. We define a hash function

$$h : \{0,1\}^* \times V \to (\mathbb{Z}/l\mathbb{Z})^\times \times (\mathbb{Z}/l\mathbb{Z})^\times$$

with image size greater than or equal to l. There will be various choices for h as described later. We further define a full hash function

$$H : \{0,1\}^* \to G_1^*$$

where $G_1^* := G_1 \backslash \{0\}$. We also abbreviate again $G^* := G \backslash \{0\}$ and $V^* := V \backslash \{1\}$.

The general signature scheme consists of 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.

Scheme 2.

Setup : The TA chooses efficiently computable monomorphisms $s : G_1 \to G$ and $q : G_1 \to V$ with $p(s(x)) = q(x)$ for all $x \in G_1$ and publishes q, while s is kept secret.

Extract : This algorithm is performed by the TA when a signer requests the secret key corresponding to their identity. Suppose the signer's identity is given by the string ID. The secret key of the identity is then given by $a = s(H(ID))$, which is computed by the TA and given to the signer.

The extraction step is typically done once for every identity and uses the same setup data for many different identities. The public key of the signer is

$y = q(H(ID))$ which can be computed by the verifier using the identity of the signer.

Sign : To sign a message m the signer picks a random $k \in G^*$ and then computes:

1. $r = p(k)$
2. $(v, w) = h(m, r)$
3. $u = va + wk$

The signature is the pair $(u, r) \in G \times V^*$.

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

1. $(v, w) = h(m, r)$
2. $y = q(H(ID))$
3. Accept the signature if and only if $p(u) = y^v r^w$.

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

$$\begin{aligned} p(u) &= p(va + wk) \\ &= p(a)^v p(k)^w \\ &= p(s(H(ID)))^v p(k)^w \\ &= q(H(ID))^v p(k)^w \\ &= y^v r^w. \end{aligned}$$

ElGamal Variations and Schnorr Version

There are a number of variations of Scheme 2. First of all, the signer and verifier could compute $(w, v) = h(m, r)$ instead of $(v, w) = h(m, r)$. If $G = (\mathbb{Z}/l\mathbb{Z})^+$ 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 the equation $u = va+wk$. The signature consists then of the value of this remaining variable together with r. We remark that the case $G = (\mathbb{Z}/l\mathbb{Z})^+$ does however not lead to true identity based schemes. These variations are equally secure (in the random oracle model) since the respective tuples (u, v, w) are computationally indistinguishable, because of $a, k \neq 0$. These variations correspond to the well known six variations of the modified ElGamal scheme, see [11, 13].

There are various possibilities for the definition of h. If $h_1 : \{0,1\}^* \to (\mathbb{Z}/l\mathbb{Z})^\times$, $h_2 : V \to (\mathbb{Z}/l\mathbb{Z})^\times$ and $h_3 : \{0,1\}^* \times V \to (\mathbb{Z}/l\mathbb{Z})^\times$ 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 however not be admissible. The case $h(m, r) := (h_3(m, r), 1)$ is of particular interest. Namely, in Scheme 2 we then have $w = 1$ and the verification equation is $p(u) = y^v r$. This means we can solve $r = p(u)y^{-v}$. It is hence equivalent giving (u, v) as a signature instead of (u, r) and depending on V this might take less memory. The modification of the signing and verification steps is straightforward and yields a general version of Schnorr signatures (compare also with Scheme 1).

Security

We proceed with the security discussion. We consider an adversary A which is assumed to be a (polynomial time) probabilistic Turing machine which besides the global scheme parameters takes as input q and a random tape. The adversary's goal is to produce an existential forgery of a signature by a signer ID of its choice. To aid the adversary we allow it to query four oracles:

Identity Hash Oracle : For any given identity ID this oracle will produce the corresponding hash value $H(ID)$.

Extraction Oracle : For any given identity ID this oracle will produce the corresponding secret key $s(H(ID))$.

Message Hash Oracle : For any given message m and $r \in V^*$ this oracle will produce the corresponding hash value $h(m, r)$.

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 not be a signature such that the secret key of the corresponding identity or the signature itself have been asked of the oracles.

Theorem 2. *In the random oracle model, suppose that an adaptive adversary A exists which makes at most $n_1 \geq 1$ queries of the identity hash and extraction oracle, at most $n_2 \geq 1$ queries of the message hash and signature oracle and which succeeds within time T_A of making an existential forgery with probability*

$$\varepsilon_A \geq \frac{a\, n_1 n_2^2}{l}$$

for some constant $a \in \mathbb{Z}^{\geq 1}$. Then there is another probabilistic algorithm C and a constant $c \in \mathbb{Z}^{\geq 1}$ such that C computes

$$s(R)$$

on input of any given $P, R \in G_1^*$ and $Q \in G^*$ with $Q = s(P)$, in expected time

$$T_C \leq \frac{c\, n_1 n_2 T_A}{\varepsilon_A}.$$

Proof. We assume familiarity with [15] and their proof technique. Let $P, R \in G_1^*$ and $Q \in G^*$ be generators such that $Q = s(P)$ and hence $p(Q) = q(P)$. We first explain how the oracle queries of A are simulated. For the i-th identity hash or extraction query for ID_i we return $H(ID_i) := \lambda_i P$ and, if requested, $s(H(ID_i)) = \lambda_i Q$ where $\lambda_i \in (\mathbb{Z}/l\mathbb{Z})^\times$ and $\lambda = (\lambda_i)_{i=1,2,\ldots}$ constitutes a random tape. A query for the hash value $h(m, r)$, if not yet defined, is answered by a random $(v, w) = \delta_i \in (\mathbb{Z}/l\mathbb{Z})^\times \times (\mathbb{Z}/l\mathbb{Z})^\times$ which is successively taken from a random tape $\delta = (\delta_i)_{i=1,2,\ldots}$. To sign a message m for ID we issue the hash query $H(ID)$ ourselves and generate a random $u \in G^*$ and a random vector $(v, w) \in (\mathbb{Z}/l\mathbb{Z})^\times \times (\mathbb{Z}/l\mathbb{Z})^\times$ where $u = \gamma_\nu$ and $(v, w) = \delta_j$ are successively taken from the random tapes $\gamma = (\gamma_\nu)$ and $\delta = (\delta_j)$ respectively. Let $y := q(H(ID))$. We then compute $r := (p(u)/y^v)^{1/w}$. We remark that r is a random element in V since γ and δ are random and independent. We define the hash value $h(m, r) := (v, w)$ and return the signature (u, r). This procedure fails if $r = 1$ or $h(m, r)$ is already defined (the adversary could for example make a large number of queries $h(m, r)$ without us computing and recording the associated signatures). Since r is random the probability for a simulation failure during the n_2 message hash and signing queries is less than $2n_2^2/l$. The simulation is indistinguishable since (u, r) is random.

Running A and answering its oracle queries in the described way depends deterministically on λ, δ, γ and the random tape ω the adversary A is provided with. Because of the assumption on A, the indistinguishability and the low failure probability of the simulation, providing random values and running A results in a signature in time T_A with probability at least $\varepsilon_A - 2n_2^2/l \geq \varepsilon_A/2$ (true for $a = 14$), using at most n_1 identity and extraction queries, n_2 message hash and signature queries. This reduces the discussion to the case of a "passive" adversary, depending deterministically on λ, δ, γ and ω.

We now want to apply the forking lemma technique of [15] to control the identity hash and message hash values for which the adversary computes forged signatures.

We can proceed almost verbatim as in [15, Lemma 8] and its proof: Since the probability of guessing the hash value of an identity is $1/l$ and $\varepsilon_A/2 \geq 7n_1/l$ (true for $a = 14$), running A repeatedly with random input yields a signature for the hash value $H(ID_\beta) = \lambda_\beta P$ and (finite) index β after (expected) $4/\varepsilon_A$ executions with probability at least $4/5$. Next, with probability $1/4$ we have that a replay of A with the same δ, γ, ω, and $\lambda = (\lambda_i)$ unchanged for $i < \beta$ and randomly chosen for $i \geq \beta$, yields a signature with probability at least $(\varepsilon_A/2)/(14n_1)$, for the now different $H(ID_\beta) = \lambda_\beta P$ but same β. The point is that we may as well answer the β-th identity hash query of A by values μR for random $\mu \in (\mathbb{Z}/l\mathbb{Z})^\times$ since this is as random as $H(ID_\beta)$ and A does not issue an extraction query for this identity, which we could not answer. As an aside, note that if we have an identity hash collision $H(ID_{\beta'}) = H(ID_\beta) = \mu R$ we have solved the discrete logarithm problem $R = \lambda P$, namely $\lambda = \lambda_{\beta'}/\mu$.

We combine these steps similar to [15, Lemma 8]. Replaying A an at most $(14n_1)/(\varepsilon_A/2)$ (expected) times and answering the β-th identity hash query with random values μR results in a signature for some μR with probability at least $3/5$. We thus obtain a machine B which on input of δ, γ and the random tape ω' returns a signature for an identity hash μR in time $T_B \leq (4/\varepsilon_A + (14n_1)/(\varepsilon_A/2))T_A \leq 32n_1T_A/\varepsilon_A$ with probability $\varepsilon_B \geq 1/9$, where μ depends on δ, γ, ω' and is drawn from a set of $(14n_1)/(\varepsilon_A/2)$ candidates which depends only on ω'. Furthermore at most $32n_1n_2/\varepsilon_A$ values are taken from δ

and the message hash $h(m,r)$ of the signature equals one of the last n_2 queried values of δ.

We apply the forking lemma a second time, now to B and with respect to δ (and regarding γ, ω' as one random tape). The conditions of [15, Lemma 8] are satisfied because we may assume that $\varepsilon_B \geq 1/9 \geq 7n_2/l$ (this is true for $a = 63$). Feeding δ, γ, ω', the input to B, from a random tape ω'' we obtain a machine C which on input of ω'' replays B at most $2/\varepsilon_B + 14n_2/\varepsilon_B$ times and returns two signatures (u_1, r) and (u_2, r) with message hash values $h_1(m,r)$ and $h_2(m,r)$ for the same m, r and identity hash values $\mu_1 R$ and $\mu_2 R$, in time $T_C \leq 16n_2 T_B/\varepsilon_B$ and with probability $\varepsilon_C \geq 1/9$. The value of m, r is the same since up to the query of $h_1(m,r)$ and $h_2(m,r)$ identical operations are carried out by the replays of B, as $\delta = (\delta_j)$ for $j < \beta$ for some fixed index β and γ, ω' remain unchanged. Furthermore, the $h_i(m,r)$ are randomly chosen from the set of successful hash values δ_β for the given fixed γ, ω' which has cardinality $\geq \varepsilon_B/(14n_2)(l-1)^2$ and can be partitioned into $\geq \varepsilon_B/(14n_2)(l-1)$ classes of linearly dependent vectors. Also, the μ_i are chosen from a set of $(14n_1)/(\varepsilon_A/2)$ candidates. Let $(v_1, w_1) = h_1(m,r)$ and $(v_2, w_2) = h_2(m,r)$. The pairs $(\mu_1 v_1, w_1)$, $(\mu_2 v_2, w_2)$ are hence linearly dependent with probability $\leq (14n_1)/(\varepsilon_A/2)/(\varepsilon_B/(14n_2)(l-1)) \leq 3528 n_1 n_2/(\varepsilon_A(l-1)) \leq 2/3$ (true for $a = 7056$ and $l \geq 5$). Thus C returns linearly independent $(\mu_1 v_1, w_1)$, $(\mu_2 v_2, w_2)$ in time $T_C \leq 16n_2 T_B/\varepsilon_B \leq 4608 n_1 n_2 T_A/\varepsilon_A$ with probability $\geq 1/27$.

By dividing the two signing equations $p(u_i) = y_i^{v_i} r^{w_i}$ for $y_i = q(\mu_i R)$ we obtain the equation

$$p(u_1 - (w_1/w_2)u_2) = q(R)^{\mu_1 v_1 - (w_1/w_2)\mu_2 v_2}.$$

Since the $(\mu_i v_i, w_i)$ are linearly independent we have $\mu_1 v_1 - (w_1/w_2)\mu_2 v_2 \neq 0$. Hence

$$p((\mu_1 v_1 - (w_1/w_2)\mu_2 v_2)^{-1}(u_1 - (w_1/w_2)u_2)) = q(R)$$

so that we have solved $p(x) = q(R)$ and thus computed $x = s(R)$ in time $\leq 4608 n_1 n_2 T_A/\varepsilon_A$ with probability $\geq 1/27$. This yields the constants $a = 7056$ and $c = 27 \cdot 4608 = 124416$.

Finally, we observe that the expected number of replays of A and B might not be explicitly known so that B and C cannot be given in the form above. In this case we can apply the same technique as in [15, Thm. 10] for B and C. Here B would have expected running time $T_B \leq 84480 n_1 T_A/(\varepsilon_A/2)$ and thus succeeds with probability $\geq 1/2$ in time $2T_B$. The expected running time of C is then $T_C \leq 84480 n_2 2T_B/(1/2) \leq 8 \cdot 84480^2 n_1 n_2 T_A/\varepsilon_A$, resulting in $c = 2 \cdot 168960^2$.

The constants in the proof can be optimized when l is chosen large with respect to n_1 and n_2. Also, the assumption $\varepsilon_A \geq a\, n_1 n_2^2/l$ of the theorem can be replaced by weaker assumptions as used in the proof.

Theorem 2 says in other words that, given p, q and one or several equations $Q = s(P)$, for the signature scheme to be secure it should be infeasible to compute other values $s(R)$. If this is the case then the TA's secret knowledge of s thus provides a generalized trap door with respect to p and q.

Multiple Trust Authorities

The general signature scheme can be extended to multiple trust authorities TA_i in the following way. Suppose $s_i : G_1 \to G$ and $q_i : G_1 \to V$ are the secret and public monomorphisms of TA_i for $1 \leq i \leq n$. We can then form a virtual trust authority TA defined by the monomorphisms $s(x) := \sum_{i=1}^{n} s_i(x)$ and $q(x) := \prod_{i=1}^{n} q_i(x)$ such that $p(s(x)) = q(x)$ for all $x \in G_1$. Note that since the s_i are random and independent, the probability of s and q not to be injective is $1/l$. The signer computes his private key via $s(H(ID)) = \sum_i s_i(H(ID))$ where he obtains the $s_i(H(ID))$ from each TA_i in turn. The verifier determines the signer's public key via $q(H(ID)) = \prod_i q_i(H(ID))$ where he computes the $q_i(H(ID))$ using the public monomorphisms q_i of each TA_i.

Let $I \cup J = \{1, \ldots, n\}$ and $I \cap J = \{\}$. An existential forgery with respect to the virtual trust authority TA under collusion of the TA_i for $i \in I$ is equivalent to an existential forgery with respect to the virtual trust authority obtained by combining the TA_i for $i \in J$, for which we can apply Theorem 2.

Examples

Example 1. The pairing based signature scheme Scheme 1 is obtained from the general Scheme 2 in the following way. We let $(G_1, +) = (G, +)$ and define $p(x) := e(x, P)$, $q(x) := e(x, Q_{TA})$ and $s(x) := tx$. Since P and Q_{TA} are public, $p(x)$ and $q(x)$ can be computed by all parties. As t is known to the TA it can compute $s(x)$. The steps of Scheme 1 are obtained from the steps of Scheme 2 together with the slight modification outlined in the remarks about Schnorr signatures. Other schemes can be obtained from the variations of hash functions and the parameter order as described above. With respect to the security we have $p(Q_{TA}) = q(P)$ which we need for the assumptions of Theorem 2. It follows that if there is an adversary, we can compute $s(x)$ without knowing t. Now, if we are given (P, Q_{TA}, R) we can thus compute $s(R) = tR$ where $Q_{TA} = tP$, thereby solving the Diffie-Hellman problem in G. This proves Theorem 1.

Example 2. The usual public key signature schemes are obtained as follows. Let $(G_1, +) = (V, \cdot)$ and q be the identity. The trust authority is now identical to the signer and there are no other signers associated. The signer chooses $a \in G^*$ at random and computes $y = p(a)$. The monomorphism s is the inverse of p. A typical example would be $p(x) = g^x$ such that s is the discrete logarithm map with respect to the element g. In the attack model we would artificially require $q(H(ID)) = H(ID) = y$ and that precisely one H-query is made. As a consequence, $n_1 = 1$ and Theorem 2 implies the standard security of the schemes in the single user setting, relating it to the inversion of p.

Example 3. If we take $(G, +) = (G_1, +) = (V, \cdot) = (\mathbb{Z}/n\mathbb{Z})^\times$ for n an RSA-modulus we essentially recover versions of the identity based signature scheme in [18]. Note that we have defined our general scheme only for cyclic groups of prime order whereas $(\mathbb{Z}/n\mathbb{Z})^\times$ is in general not cyclic of prime order. Thus

Theorem 2 cannot readily be applied. Since $(\mathbb{Z}/n\mathbb{Z})^\times$ is not too far from being cyclic it might be possible to adapt Scheme 2 and Theorem 2 towards this case.

6 Conclusion

We have described a generic signature scheme which gives rise to efficient identity based signature schemes if pairings are used. The security of these schemes follows from the security of the generic scheme in the random oracle model and is based on the Diffie-Hellman problem in the domain of the used pairing. Other instantiations of the generic scheme include traditional public key signature schemes in cyclic groups. In addition we have discussed key escrow and the distribution of keys to multiple trust authorities.

7 Acknowledgements

I would like to thank D. Kohel, J. Malone-Lee, K. Paterson 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. University of Waterloo technical report, CORR 98-19 (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. J. Cha and J. Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups *IACR preprint server*, submission 2002/018, 2002.
6. C. Cocks. Split knowledge generation of RSA parameters. In *Cryptography and Coding*, Springer-Verlag LNCS 1355, 89–95, 1997.
7. C. Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding*, Springer-Verlag LNCS 2260, 360–363. 2001.
8. S. D. Galbraith. Supersingular curves in cryptography. In *Advances in Cryptology - ASIACRYPT 2001*, Springer-Verlag LNCS 2248, 495-513, 2001.
9. A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Algorithmic Number Theory Symposium, ANTS-IV*, Springer-Verlag LNCS 1838, 385–394, 2000.
10. 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.
11. A. J. Menezes, P. C. Oorschot and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.
12. V. Miller. Short programs for functions on curves. Unpublished manuscript, 1986.

13. K. Nyberg and R. A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. *Designs, Codes and Cryptography*, **7(1/2)**, 61-81, 1996.
14. K. G. Paterson. ID-based signatures from pairings on elliptic curves *IACR preprint server*, submission 2002/003, 2002.
15. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13**, 361–396, 2000.
16. K. Rubin and A. Silverberg. The best and worst of supersingular abelian varieties in cryptology. *IACR preprint server*, submission 2002/006, 2002.
17. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, 2000.
18. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO '84*, Springer-Verlag LNCS 196, 47–53, 1984.
19. J. H. Silverman. *The Arithmetic of Elliptic Curves.* GTM 106, Springer-Verlag, 1986.

Appendix

Realization of the Identity Based Signature Schemes

In order use the identity based schemes described in the paper we need to find suitable groups G, V and pairings $e : G \times G \to V$. These groups are provided by finite fields and elliptic curves over finite fields, and the pairings are derived from the Weil or Tate pairing, as for example in [3].

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 1 the signature consists of $v \in (\mathbb{Z}/l\mathbb{Z})^\times$ instead of $r \in V$, resulting in a more bandwidth efficient scheme.

The Weil and Tate Pairings on Elliptic Curves

We shall summarize the properties we require of the Weil pairing, much of the details can be found in [3], [10] and [19]. We also present the Tate pairing since it 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

$$l \,|\, (q^\alpha - 1).$$

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 G' be a subgroup of $E[l]$ such that $E[l] = G \oplus G'$.

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 $P, Q \in G$. To overcome this problem one possibility is to consider an (injective) non $\mathbb{F}_q$-rational endomorphism $\phi : G \to G'$. 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 $\phi : G \to G'$ 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 $\phi : G \to G'$ defined by $P \mapsto P'$ for an arbitrary $P' \in G' \backslash \{0\}$. 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 $P' = \phi(P)$ and $Q'_{TA} = \phi(Q_{TA})$, 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 [12]. 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 $\phi(Q) \in G'$ 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

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

and 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 $f_{\phi(Q)}$ since one can work over $\mathbb{F}_q$ instead of $\mathbb{F}_{q^k}$. 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 $\mathbb{F}_{q^k}$.

Supersingular Elliptic Curves

Only supersingular elliptic curves provide non $\mathbb{F}_q$-rational endomorphisms. The following table lists the essential examples, the parameter α and a non $\mathbb{F}_q$-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

$$t_2(p) = \begin{cases} 2^{(p+1)/2} & \text{for } p \equiv \pm 1, \pm 7 \bmod 24 \equiv \pm 1 \bmod 8, \\ -2^{(p+1)/2} & \text{for } p \equiv \pm 5, \pm 11 \bmod 24 \equiv \pm 3 \bmod 8, \end{cases}$$

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

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