Implementing the Tate pairing

Steven D. Galbraith[1], Keith Harrison, David Soldera
Trusted E-Services Laboratory
HP Laboratories Bristol
HPL-2002-23
March 8th , 2002*

E-mail: Steven.Galbraith@rhul.ac.uk, keith_harrison@hp.com, David_Soldera@hplb.hpl.hp.com

Tate pairing, bilinear pairing, implementing characteristic 3

The Weil and Tate pairings have found several new applications in cryptography. To efficiently implement these cryptosystems it is necessary to optimise the computation time for the Tate pairing. This paper provides methods to achieve fast computation of the Tate pairing. We also give division-free formulae for point tripling on a family of elliptic curves in characteristic three. Examples of the running time for these methods are given.

[1] Mathematics Department, Royal Holloway University of London, Egham, Surrey TW20 0EX, UK

Implementing the Tate pairing

Steven D. Galbraith*[1], Keith Harrison[2], and David Soldera[2]

[1] Mathematics Department, Royal Holloway University of London,
Egham, Surrey TW20 0EX, UK.
`Steven.Galbraith@rhul.ac.uk`
[2] Hewlett-Packard Laboratories, Bristol,
Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK.
`keith_harrison@hp.com, David_Soldera@hplb.hpl.hp.com`

Abstract. The Weil and Tate pairings have found several new applications in cryptography. To efficiently implement these cryptosystems it is necessary to optimise the computation time for the Tate pairing. This paper provides methods to achieve fast computation of the Tate pairing. We also give division-free formulae for point tripling on a family of elliptic curves in characteristic three. Examples of the running time for these methods are given.

1 Introduction

The Weil and Tate pairings have recently been used to construct cryptosystems, such as the identity-based key exchange and signature schemes of Sakai, Ohgishi and Kasahara [14], the tripartite Diffie-Hellman protocol of Joux [9], the escrow El Gamal system of Verheul [17], the identity-based encryption scheme of Boneh and Franklin [3], the credential scheme of Verheul [18], the short signature scheme of Boneh et al [4], the ID-based key exchange system of Smart [16] and the ID-based signature scheme of Paterson [13].

For most of these applications either the Weil pairing or Tate pairing may be used (these pairings both provide good functionality for use in cryptosystems). In practice, as has been observed in [7, 4], the Tate pairing is more efficient for computation. (We give some timings in Section 10.1 which show how much slower the Weil pairing is). If these cryptosystems are to be adopted for practical applications it is essential to provide methods which improve the performance of Tate pairing computations.

In this paper we give techniques which enable efficient computation of the Tate pairing for cryptographic applications. Some of these techniques are familiar from the literature on fast point exponentiation for elliptic curve cryptography, but most of them are specific to the cryptographic application of the Tate pairing.

We now summarise the paper. Sections 2 and 3 describe the basics of the Tate pairing and Miller's algorithm. Section 4 indicates how the Tate pairing is used in cryptosystems. Section 5 contains the core observations which dictate

* This author thanks Hewlett-Packard Laboratories, Bristol for support.

the development of our later techniques. Section 6 shows how properties of the group order (namely, the size of the large prime, and small Hamming weights) may be used to give improved performance. Section 7 introduces new formulae for elliptic curve point tripling in characteristic three, and shows how this leads to an efficient ternary Miller's algorithm. Section 8 and 9 discuss the implementation of the finite field arithmetic. Section 10 contains some of our timing results.

We must note that in the final stages of preparing this paper we became aware of the work of Barreto, Kim and Scott [1] which also provides an excellent solution to the problem of efficient computation of the Tate pairing.

2 The Tate pairing

The Weil pairing was first introduced into cryptography by Menezes, Okamoto and Vanstone [12] who used it to attack the elliptic curve discrete logarithm problem on certain elliptic curves. The Tate pairing was introduced to cryptography by Frey and Rück [5] in their extension of the work of Menezes, Okamoto and Vanstone.

Let E be an elliptic curve over a finite field $\mathbb{F}_q$. We write $\mathcal{O}_E$ for the point at infinity on E. Let l be a positive integer which is coprime to q. In most applications l is a prime and $l | \#E(\mathbb{F}_q)$. Let k be a positive integer such that the field $\mathbb{F}_{q^k}$ contains the lth roots of unity (in other words, $l | (q^k - 1)$). Let $G = E(\mathbb{F}_{q^k})$ and write $G[l]$ for the subgroup of points of order l and G/lG for the quotient group (which is also a group of exponent l). Then the Tate pairing is a mapping

$$\langle \cdot, \cdot \rangle : G[l] \times G/lG \rightarrow \mathbb{F}_{q^k}^* / (\mathbb{F}_{q^k}^*)^l. \tag{1}$$

The quotient group on the right hand side of (1) can be thought of as the set of equivalence classes of $\mathbb{F}_{q^k}^*$ under the equivalence relation $a \equiv b$ if and only if there exists $c \in \mathbb{F}_{q^k}^*$ such that $a = bc^l$. We call this relation 'equivalence modulo lth powers'.

The Tate pairing satisfies the following properties [5]:

1. (Well-defined) $\langle \mathcal{O}_E, Q \rangle \in (\mathbb{F}_{q^k}^*)^l$ for all $Q \in G$ and $\langle P, Q \rangle \in (\mathbb{F}_{q^k}^*)^l$ for all $P \in G[l]$ and all $Q \in lG$.
2. (Non-degeneracy) For each point $P \in G[l] - \{0\}$ there is some point $Q \in G$ such that $\langle P, Q \rangle \notin (\mathbb{F}_{q^k}^*)^l$.
3. (Bilinearity) For any integer n, $\langle [n]P, Q \rangle \equiv \langle P, [n]Q \rangle \equiv \langle P, Q \rangle^n$ modulo lth powers.

The Tate pairing is defined as follows. Given the point P compute a function g such that the divisor of g is equal to $l((P) - (\mathcal{O}_E))$ (see [15] for an introduction to divisors). Then compute a divisor D which is equivalent to $(Q) - (\mathcal{O}_E)$ such that the support of D is disjoint from the support of g. Then the value of the Tate pairing (up to lth powers) is

$$\langle P, Q \rangle = g(D)$$

where $g(D) = \prod_i g(P_i)^{n_i}$ if $D = \sum_i n_i P_i$.

We emphasise that the Tate pairing is only defined up to a multiple by an lth power in $\mathbb{F}_{q^k}^*$. For most applications in cryptography a unique value is required, and so it is necessary to exponentiate the value of the Tate pairing to the power $(q^k - 1)/l$ (since raising to this power eliminates all multiples of order l).

3 Miller's algorithm

The Tate pairing can be computed using an algorithm first proposed by Miller [11] which is also described in [5,6]. Miller's algorithm is basically the usual 'double and add' algorithm for elliptic curve point multiplication combined with an evaluation of certain intermediate functions which are the straight lines used in the addition process.

Before giving the details of this algorithm we recall the elliptic curve addition law (for more details see [2,15]).

Let P and Q be points on an elliptic curve E. Let l_1 be the line through P and Q (if $P = Q$ then l_1 is taken to be the tangent to the curve E at P, if one of P or Q is $\mathcal{O}_E$ then l_1 is a 'vertical line' through the affine point). Then l_1 intersects the cubic curve E at one further point, say R_1. Now let l_2 be the line between R_1 and $\mathcal{O}_E$ (which is a 'vertical line' when R_1 is not equal to $\mathcal{O}_E$). Then l_2 intersects E at a third point R_2 which is defined to be the sum of P and Q.

The lines l_1 and l_2 can be thought of as functions on the curve, and the corresponding principal divisors are

$$(l_1) = (P) + (Q) + (R_1) - 3(\mathcal{O}_E) \text{ and } (l_2) = (R_1) + (R_2) - 2(\mathcal{O}_E).$$

It follows that we have the following equality of divisors

$$(P) - (\mathcal{O}_E) + (Q) - (\mathcal{O}_E) = (R_2) - (\mathcal{O}_E) + (l_1/l_2).$$

Let E be an elliptic curve over $\mathbb{F}_q$ and let P and Q be given points of prime order l for which we want to compute $\langle P, Q \rangle$. Miller's algorithm is given in Figure 1.

To understand how this algorithm works, first note that the divisor $(Q') - (S)$ is equivalent to the divisor $(Q) - (\mathcal{O}_E)$ and, since S was chosen randomly, it is likely that the points Q' and S in the support of $(Q') - (S)$ do not appear in any intermediate computations in the algorithm. Secondly, note that at each stage in the algorithm T_1 is the point obtained by computing $[m]P$ where m is the integer whose binary expansion is the top n bits of the binary expansion of l. The value f_1 is the evaluation at the divisor $(Q') - (S)$ of the function f defined such that

$$m((P) - (\mathcal{O}_E)) = (T_1) - (\mathcal{O}_E) + (f).$$

Hence, at the end of the algorithm we have $T_1 = \mathcal{O}_E$ and f_1 is the evaluation at $(Q') - (S)$ of the function g such that $l((P) - (\mathcal{O}_E)) = (g)$, as required from the definition of the Tate pairing.

```
Choose a random point $S \in E(\mathbb{F}_{q^k})$ and compute $Q' = Q + S \in E(\mathbb{F}_{q^k})$.

Set $n = \lfloor \log_2(l) \rfloor - 1$, $T_1 = P$, $f_1 = 1$.

While $n \geq 1$ do

  – Calculate the equations of the straight lines $l_1$ and $l_2$ arising in a doubling of $T_1$.
    Set $T_1 = [2]T_1$ and $f_1 = f_1^2(l_1(Q')l_2(S))/((l_2(Q')l_1(S))$.
  – If the $n$th bit of $l$ is one then
      • Calculate the equations of the straight lines $l_1$ and $l_2$ arising in an addition
        of $T_1$ and $P$. Set $T_1 = T_1 + P$ and set $f_1 = f_1(l_1(Q')l_2(S))/((l_2(Q')l_1(S))$.
  – Decrement $n$.

Return $f_1$.
```

Fig. 1. Miller's Algorithm.

4 The cryptographic applications

We do not discuss the cryptographic applications of the Tate pairing in great detail since we are interested in implementation issues which are common to all schemes. We simply note that:

1. Cryptosystems based on the Weil pairing may be modified to use the Tate pairing, and this will improve their computational performance.
2. In many of these schemes the calculation of the Tate pairing is the dominant computational task.

In most applications of the Weil and Tate pairing to cryptography we consider an elliptic curve E over $\mathbb{F}_q$ with number of points divisible by some prime l. It is necessary that l have at least 160 bits for security, and for efficiency it is desired that l and q not be too large. Also important for these applications is the finite field $\mathbb{F}_{q^k}$ where k is defined to be the smallest integer such that $l|(q^k - 1)$. It is necessary that q^k have at least 1000 bits for security, and for good efficiency it is desired that q^k not be too large. Further discussion about these matters may be found in [7], but the conclusion is that there are three cases most relevant for cryptography:

1. Supersingular elliptic curves such as $y^2 = x^3 + 1$ over certain prime fields $\mathbb{F}_p$ where p has 512 bits (in this case $k = 2$).
2. Supersingular elliptic curves of the form $y^2 + y = x^3 + x + b$ ($b \in \{0, 1\}$) over $\mathbb{F}_2$ considered as a group over $\mathbb{F}_{2^m}$ where m is prime of size around 250 (in this case $k = 4$).
3. Supersingular elliptic curves of the form $y^2 = x^3 - x \pm 1$ over $\mathbb{F}_3$ considered as a group over $\mathbb{F}_{3^m}$ where m is prime of size around 110 (in this case $k = 6$).

For the cryptographic applications the basic operation is to compute the value of the Tate pairing $\langle P, Q \rangle$ where $P \in E(\mathbb{F}_q)$ and where $Q \in E(\mathbb{F}_{q^k})$ (usually Q is the image of some multiple of P under a non-rational endomorphism or "distortion map"). We stress that since a unique value is required for the cryptographic applications we must also raise the value of the Tate pairing to the power $(q^k - 1)/l$.

5 Efficient computation of the Tate pairing

Our analysis begins in this section, where we make three general comments about efficient computation of the Tate pairing in the specific application we have in mind.

The most important observation is that we are computing $\langle P, Q\rangle$ where $P \in E(\mathbb{F}_q)$ and where $Q \in E(\mathbb{F}_{q^k})$. In practice, this means that the coefficients of the lines l_i in Miller's algorithm (Figure 1) are all elements of the smaller field $\mathbb{F}_q$ while the large field $\mathbb{F}_{q^k}$ is only used for computing the value f_1.

This observation is the most fundamental observation in the paper and most of the implementation details arise from trying to make the most it. In particular, to benefit from this observation, one should work with an efficient representation of $\mathbb{F}_q$ for all operations involving the elliptic curve E, the points T_1 and T_2, and the straight lines l_i. One should then implement efficient operations for $\mathbb{F}_{q^k}$ which allow fast scalar multiplication by elements in $\mathbb{F}_q$. The natural way to proceed is to represent $\mathbb{F}_{q^k}$ as a degree k extension of $\mathbb{F}_q$. We give many more details in Section 9. We comment that this is different to the approach proposed by Boneh, Lynn and Shacham [4].

A further example of working in subfields whenever possible is to consider the choice of the random point S in Miller's algorithm (Figure 1). As stated, $S \in E(\mathbb{F}_{q^k})$ but in fact we may take $S \in E(\mathbb{F}_q)$ and this reduces the number of operations in $\mathbb{F}_{q^k}$.

It is interesting at this point to consider the relationship between the Weil pairing and the Tate pairing. We write $e_l(P, Q)$ for the Weil pairing. In most situations the Weil pairing is related to the Tate pairing by the equation $e_l(P, Q) = \langle P, Q\rangle / \langle Q, P\rangle$ (and no exponentiation is required to get a unique value) and this is the way the Weil pairing is usually computed. Other methods to compute the Weil pairing (such as Section III.8 of [**?**]) seem to be even less efficient. This leads to the often quoted statement "the Weil pairing is just two applications of the Tate pairing". However, in the case that $P \in E(\mathbb{F}_q)$ but $Q \in E(\mathbb{F}_{q^k})$ then these two Tate pairing operations require very different computation times. Hence, the Weil pairing seems to require much more than twice the running time of the Tate pairing in the cryptographic applications.

Our second observation relates to the well-known fact that divisions are more expensive than multiplications. This statement is particularly true for divisions in the large field $\mathbb{F}_{q^k}$ since we are representing it as a degree k extension of the field $\mathbb{F}_q$. Hence it is desirable to reduce the number of divisions in $\mathbb{F}_{q^k}$ in Miller's algorithm. Consider the divisions which are required in the large field $\mathbb{F}_{q^k}$ when computing the value f_1. It is obvious that these divisions can all be gathered into a single division at the conclusion of the algorithm by representing the value f_1 as a quotient f_1/f_2 and using multiplications to update the f_i.

Our third general observation is that, as with elliptic curve point exponentiation algorithms, there is a significant improvement by using window methods (see [2], [8]). These methods employ a precomputation stage which computes the values $[n]P$ for all values n in a 'window' of 3 or 4 bits. Miller's algorithm then proceeds by performing addition operations according to windows in the binary expansion of the exponent l instead of bit by bit. This does not change the number of doubling operations, but it does reduce the number of addition operations. The methods are completely standard (see [2, 8]) and it is not necessary to repeat them here.

Note that in Section 6 we describe a class of groups which are particularly efficient for the Tate pairing computation, and the window methods are no longer useful for these groups.

Finally we mention the use of projective coordinates and homogenizing Miller's algorithm to remove divisions. We have implemented such a method and found that in practice it gives worse performance. The number of multiplications involved in such a method grows so much that it outweighs the benefit gained from avoiding divisions in $\mathbb{F}_q$ (we find that an inversion in $\mathbb{F}_{3^m}$ represented with a polynomial basis takes only about 5 times the time of a multiplication, see Section 9.3).

6 Choice of groups

As noted by Boneh and Franklin [3] it is not necessary that the prime order l be of the same size as the field q. For instance, when working with supersingular elliptic curves over $\mathbb{F}_p$ where $p > 3$ it is necessary that p have at least 512 bits, but l may be chosen to have 160 bits.

This technique of working in a smaller subgroup has a huge impact on the complexity of Miller's algorithm, since the number of iterations depends on $\log_2(l)$. This technique may be used in characteristic 2 and 3 as well, whenever the group order of $E(\mathbb{F}_q)$ has factors of a suitable size.

A further method which speeds up the Tate pairing very significantly is to choose the prime l such that it has very low hamming weight (or, more generally, so that it has low hamming weight in a signed binary representation, or in a ternary representation in characteristic three). This greatly reduces the number of addition operations in Miller's algorithm. Note that this technique means that window methods are no longer required, and so there is no precomputation step in this case.

The system of Boneh and Franklin [3] for large prime characteristic can be trivially modified to employ primes l of low Hamming weight. In characteristic 2 an example of such a group order is the following: Let E be the elliptic curve $y^2 + y = x^3 + x + 1$ over $\mathbb{F}_{2^{283}}$. Then $\#E(\mathbb{F}_{2^{283}}) = l$ where l is the prime $l = 2^{283} + 2^{142} + 1$, which has Hamming weight 3. There are other cases in characteristic 2 with prime number of points which have the same property of their (signed) binary expansion. Similarly, supersingular curves with a prime number of points in characteristic 3 will have low Hamming weight of the signed ternary expansion of l.

For several examples in characteristic two and three the group order N has small Hamming weight, but the large prime factor l is a quotient of N by a small cofactor and so it does not have small Hamming weight. In practice one can compute the Tate pairing of the points P and Q of order l with respect to the group order N (and it is recommended to then raise to the exponent $(q^k - 1)/N$ which also has low Hamming weight). In this case the small Hamming weight of N provides computational savings in Miller's algorithm, while the result is still a unique element of order l. An important point to note is that the value of the Tate pairing when computed with respect to the group order N is not necessarily the same as the value of the Tate pairing when computed with respect to the large prime l (hence the use of this time-saving technique must be specified in the system parameters to enable interoperability). This technique is used for the implementation results in Section 10 and it reduces the running time by at least 30%.

7 Specific advantages in characteristic 2 and 3

In this section we discuss certain features of elliptic curves in small characteristic. In particular, we discuss certain arithmetic operations which are particularly efficient, such as point tripling in characteristic three.

7.1 Doubling in characteristic two

It is well-known in elliptic curve cryptography that there are performance advantages available in characteristic two, particularly when implementing elliptic curve exponentiation directly in hardware. For a survey of point exponentiation methods in characteristic two see Hankerson, Hernandez and Menezes [8]. These methods can all be used to improve Miller's algorithm in characteristic two, and it follows that cryptosystems based on the Tate pairing on supersingular curves in characteristic two have good performance. Note that, for the field sizes we are considering, Karatsuba multiplication does not provide any benefit. All the relevant methods from [8] were used to obtain the timings in Section 10.

7.2 Tripling in characteristic three

In characteristic three for our supersingular elliptic curves (and, more generally, for curves over $\mathbb{F}_{3^m}$ with equations of the form $y^2 = x^3 + Ax + B$) it happens that the tripling operation can be performed extremely efficiently.

Indeed, one can give tripling formulae which do not require divisions! For the Tate pairing computation it is necessary to obtain the equations of the straight lines used for the addition rule, and so one division is unavoidable.

We give all the details of the tripling operations and the straight lines below.

Let $P = (x_1, y_1)$ be a point on $E : y^2 = x^3 - x + b$ over $\mathbb{F}_{3^m}$. The tangent to E at P has slope $\lambda_2 = 1/y_1$ and the equation of the tangent line is

$$l_1 : y - \lambda_2 x + (\lambda_2 x_1 - y_1) = 0.$$

The point $(x_2, y_2) = [2]P$ has coordinates $x_2 = \lambda_2^2 + x_1$ and $y_2 = -\lambda_2^3 - y_1$. The equation of the vertical line is $l_2 : x - x_2 = 0$.

The line between (x_1, y_1) and (x_2, y_2) has slope $\lambda_3 = y_1^3 - \lambda_2$ and its equation is

$$l_1' : y + (\lambda_2 - y_1^3)x + (y_1^3 x_1 - \lambda_2 x_1 - y_1) = 0.$$

The point $(x_3, y_3) = [3](x_1, y_1)$ has coordinates $x_3 = x_1 + y_1^2 + y_1^6$ and $y_3 = -y_1^9$. The equation of the vertical line is $l_2' : x - x_3 = 0$. Note that these formulae provide a division-free algorithm for tripling on these elliptic curves in characteristic three.

Also note that cubing is very fast in characteristic three (especially in hardware, or if a normal basis representation is used) and so computing y_1^3, y_1^6 and y_1^9 is cheap from y_1 and y_1^2.

These formulae for point tripling are very efficient and so it is prudent to rewrite Miller's algorithm to utilise a signed base-3 representation of the exponent l. Recall that a signed base-3 representation is an expression

$$l = \sum_{n=0}^{m} l_i 3^i$$

where $l_i \in \{-1, 0, 1\}$. We call each l_i a 'trit'. We sketch the details in Figure 2. We stress that, in practice, care must be taken to implement the formulae for l_1 and l_1' above so that the number of multiplications in the large field $\mathbb{F}_{q^k}$ is minimised.

Note that the efficient tripling formulae are valuable for efficient implementation of the application of Koblitz in [10].

8 Efficient implementation of characteristic three fields

It is essential to have an efficient implementation of arithmetic in the finite field $\mathbb{F}_{3^m}$. A lot of research has been done into efficient implementation of characteristic two finite fields, and also for large characteristic p, but characteristic three does not seem to have been studied in detail. Either polynomial bases or normal bases may be used [2], and we use polynomial bases.

The conventional wisdom for representing values in characteristic two is to represent each coefficient by a single bit and to pack 32 coefficients into a single computer word. In this way, the addition of two values can be performed efficiently by using an exclusive-or machine instruction to add 32 coefficients at a time. Most finite field packages treat characteristic two as a special case and then degenerate to using a bignum implementation for odd characteristic. This can be improved upon.

```
1. Choose a random point $S \in E(\mathbb{F}_q)$ and compute $Q' = Q + S$.
2. Precompute $P_2 = [2]P$ and the value $f_2$ of the function $f$ such that $2((P) - (\mathcal{O}_E)) = (P_2) - (\mathcal{O}_E) + (f)$ evaluated at the divisor $(Q') - (S)$.
3. Let $n$ be the number of trits in a signed base-3 representation for $l$. Set $T_1 = P$ or $P_2$ and $f_1 = 1$ or $f_2$ according to whether the $n$th bit of the ternary expansion of $l$ is 1 or 2.
4. Decrement $n$.
5. While $n \geq 1$ do
   – Perform a tripling of $T_1$, i.e., compute the equations for the lines $l_1, l_2, l_1', l_2'$ above, set $T_1 = [3]T_1$, and update the value of $f_1$ via
         $f_1 = f_1^3(l_1/l_2 \cdot l_1'/l_2')((Q') - (S))$.
   – If the $n$th trit in the signed base-3 expansion of $l$ is 1 then set $T_1 = T_1 + P$ and update $f_1$ by $f_1 = f_1(l_1/l_2)((Q') - (S))$ where $l_1$ and $l_2$ are the lines appearing in the point addition.
   – If the $n$th trit in the signed base-3 expansion of $l$ is $-1$ then set $T_1 = T_1 - P_2$ and set $f_1 = f_1 f_2(l_1/l_2)((Q') - (S))$ where $l_1$ and $l_2$ are the lines appearing in the point addition.
   – Decrement $n$.
6. Return $f_1$.
```

Fig. 2. Miller's Algorithm in base three.

We note that a coefficient in characteristic 3 has the values 0, 1, or 2. That is, we need two bits to represent such a value. Rather than pack 16 2-bit coefficients into a 32 bit word, we pack the high order bits into one word array and the low order bits into a separate word array.

In other words, we write the 16 coefficients modulo 3 as $a = a_{\text{lo}} + 2a_{\text{hi}}$. This gives the following advantages:

1. Doubling a value can be performed by swapping the high and low order bit arrays. Note: negation is identical to doubling in characteristic 3.
2. Adding two values $r = a + b$ leads to

   $$(r_{\text{hi}}, r_{\text{lo}}) = (a_{\text{hi}}, a_{\text{lo}}) + (b_{\text{hi}}, b_{\text{lo}})$$

   where

   $$\begin{aligned} r_{\text{lo}} &= ((a_{\text{lo}} \wedge b_{\text{lo}}) \& (\sim (a_{\text{hi}} | b_{\text{hi}}))) | (a_{\text{hi}} \& b_{\text{hi}}) \\ r_{\text{hi}} &= ((a_{\text{hi}} \wedge b_{\text{hi}}) \& (\sim (a_{\text{lo}} | b_{\text{lo}}))) | (a_{\text{lo}} \& b_{\text{lo}}). \end{aligned}$$

   Here, as usual, $\sim$ means bitwise complement, & means bitwise and, | means bitwise or, and $\wedge$ means bitwise exclusive-or. In other words, we can add 32 coefficients with 12 boolean operations.
3. Cubing is performed analogously to squaring in characteristic 2 by using a "thinning" algorithm with a reduction operation (this is just a shift if a normal basis is used).

4. Subtracting two values is performed using addition:

$$(r_{\text{hi}}, r_{\text{lo}}) = (a_{\text{hi}}, a_{\text{lo}}) - (b_{\text{hi}}, b_{\text{lo}}) = (a_{\text{hi}}, a_{\text{lo}}) + (b_{\text{lo}}, b_{\text{hi}}).$$

9 Efficient computation in extension fields

We now describe some implementation details for finite field extensions. These issues arise because of our choice of field representation, which in turn is motivated by the benefit of working in subfields wherever possible.

The two most important cases are the elliptic curves $y^2 = x^3 - x \pm 1$ over extensions of $\mathbb{F}_3$ and $y^2 + y = x^3 + x + b$ over extensions of $\mathbb{F}_2$. In practice it is necessary to be able to work efficiently with finite fields $\mathbb{F}_{3^{6m}}$ and $\mathbb{F}_{2^{4m}}$ where m is prime. We give further details about how to achieve this.

9.1 Characteristic two

We represent $\mathbb{F}_{2^{4m}}$ by a tower of two quadratic extensions of $\mathbb{F}_{2^m}$. To be precise, let $F = \mathbb{F}_{2^m}$ and denote

$$F_1 = F[x]/(x^2 + x + 1) \cong \mathbb{F}_{2^{2m}}$$

and

$$F_2 = F_1[y]/(y^2 + (x+1)y + 1) \cong \mathbb{F}_{2^{4m}}.$$

A general element of F_2 can be written as $a + bx + cy + dxy$ with $a, b, c, d \in F$.

The naive way to perform multiplication of two elements $(u_1 + yv_1)$ and $(u_2 + yv_2)$ of F_2 (where $u_i, v_i \in F_1$) to obtain the product

$$(u_1u_2 + v_1v_2) + y(u_1v_2 + u_2v_1 + (x+1)v_1v_2)$$

would require 4 multiplications in F_1 (plus the 'special' multiplication by the term $(x+1)$). A more efficient multiplication process is to compute the three products $t_1 = u_1u_2, t_2 = v_1v_2$ and $t_3 = (u_1 + v_1)(u_2 + v_2)$. The desired product is then recovered as $(t_1 + t_2) + y(t_3 - t_1 + xt_2)$ which requires 3 multiplications in F_1 plus the 'special' multiplication xt_2 (which is shown below to be just a single addition).

Similarly, multiplication of general elements $(u_1 + xv_1)(u_2 + xv_2)$ in F_1 can be performed with just 3 multiplications in F, plus one 'special' multiplication.

Finally, note that the result of the special multiplication $x(u + xv)$ is equal to $v + x(u + v)$, which is computed by a single addition.

In conclusion, the cost of a general multiplication in F_2 is reduced from 16 (or more) multiplications in F to only 9 multiplications in F.

Division in F_2 can be reduced to a single division in F by using conjugates. The details are straightforward, and since there is only one division in F_2 in our algorithm this is not worth discussing in depth here.

9.2 Characteristic three

We represent $\mathbb{F}_{3^{6m}}$ by a tower of extensions of $\mathbb{F}_{3^m}$. To be precise, let $F = \mathbb{F}_{3^m}$ and denote

$$F_1 = F[a]/(a^3 - a + 1) \cong \mathbb{F}_{3^{3m}}$$

and

$$F_2 = F_1[b]/(b^2 + 1) \cong \mathbb{F}_{3^{6m}}.$$

(Note that $i = b$ and $\alpha = a$ and $\beta = -a$ in the notation of Section 3.9 of [7])

As in the previous subsection, a multiplication of general elements in F_2 can be performed with fewer multiplications than the naive method. The details are as follows:

To multiply $(u_1 + bv_1)(u_2 + bv_2)$ where $u_1, u_2, v_1, v_2 \in F_1$ we compute $t_1 = u_1u_2$, $t_2 = v_1v_2$ and $t_3 = (u_1 + v_1)(u_2 + v_2)$. The product is then recovered as

$$(t_1 - t_2) + b(t_3 - t_1 - t_2).$$

The product of $(u_1+av_1+a^2w_1)$ and $(u_2+av_2+a^2w_2)$ with $u_1, u_2, v_1, v_2, w_1, w_2 \in F$ is

$$(u_1u_2+v_1w_2+w_1v_2)+a(u_1v_2+v_1u_2+v_1w_2+w_1v_2+w_1w_2)+a^2(u_1w_2+v_1v_2+w_1u_2+w_1w_2).$$

To compute this in fewer than 9 multiplications compute $t_1 = u_1u_2$, $t_2 = u_1w_2$, $t_3 = v_1v_2$, $t_4 = v_1w_2$, $t_5 = w_1u_2$, $t_6 = w_1v_2$, $t_7 = w_1w_2$ and $t_8 = (u_1 + v_1 + w_1)(u_2 + v_2 + w_2)$. The product is recovered as

$$(t_1 + t_4 + t_6) + a(t_8 - t_1 - t_2 - t_3 - t_5) + a^2(t_2 + t_3 + t_5 + t_7).$$

Hence we have reduced the cost of multiplication in F_2 from 36 to 24 multiplications in F.

Again, inversion can be reduced to a single inversion in F by using conjugates. The details are straightforward (the conjugates of $(u + av + a^2w)$ are simply $(u + (a + 1)v + (a + 1)^2w)$ and $(u + (a + 2)v + (a + 2)^2w)$).

Finally, the exponentiation operation in the finite field F_2 is performed using the signed base-3 expansion of the exponent (which has low Hamming weight in most of our examples and so window methods are not necessary).

9.3 Timing results

In summary, we have the following timing results for field operations. We record the cost in terms of the number of multiplications in the ground field. Let F be $\mathbb{F}_{2^m}$ or $\mathbb{F}_{3^m}$ respectively and F_2 be $\mathbb{F}_{2^{4m}}$ or $\mathbb{F}_{3^{6m}}$. Here, for instance, $F * F_2$ means the cost of multiplying a general element of F_2 by an element of F and $1/F$ means the cost of inverting an element of F.

	Characteristic two		Characteristic three	
m	241	283	97	163
$F * F$	1M	1M	1M	1M
$F * F_2$	4M	4M	6M	6M
$F_2 * F_2$	9M	9M	24M	24M
$1/F$	13.85M	9.25M	5.36M	5.05M
$1/F_2$	44.85M	40.25M	107.36M	107.05M

Notes:

1. The field extension inversion was not heavily optimised because it is only invoked once in the computation of a Tate or Weil pairing.
2. In characteristic three it is cheaper to perform a field inversion than to compute a field by field extension multiplication. We attribute this to the inefficiency of multiplication, rather than to any special benefit of inversion in characteristic three (it is an open problem to provide more efficient multiplication algorithms in characteristic three).
3. It is always worth tracking whether a value is in the field or in the field extension - and performing the appropriate operation.
4. The costs of performing the field inversion were established by timing 100,000 field inversions and 100,000 field multiplications. The other costs were established by examination of the code.

10 Timing results

We have implemented the Tate pairing using the methods given above. All timings were performed on a 1 GHz Pentium III with 256Mb RAM (an HP VISUALISE NT workstation). The language used was C. The compiler was Microsoft Visual C++ V6.0 with speed optimisations on.

10.1 Characteristic two timings

We give a few timings for characteristic two. Due to the numerous techniques available for efficient characteristic two arithmetic and elliptic curve operations it follows that characteristic two is the best choice for fast implementations of the Tate pairing.

Example 1:
Consider the elliptic curve $E : y^2 + y = x^3 + x + 1$ over

$$\mathbb{F}_{2^{241}} = \mathbb{F}_2[x]/\langle x^{241} + x^{70} + 1\rangle.$$

The large prime order is $l = 2^{241} - 2^{121} + 1$
Consider points $P \in E(\mathbb{F}_{2^{241}})$ and $Q \in E(\mathbb{F}_{2^{964}})$ of order l.
Weil Pairing $e_l(P, Q)$ time: 140.9 ms.

Tate Pairing $\langle P, Q\rangle^{(2^{964}-1)/l}$ time: 32.50 ms (including the finite field exponentiation).

Example 2:

Consider the elliptic curve $E : y^2 + y = x^3 + x + 1$ over

$$\mathbb{F}_{2^{283}} = \mathbb{F}_2[x]/\langle x^{283} + x^{194} + x^{129} + x^{65} + 1\rangle.$$

The large prime l is $2^{283} + 2^{142} + 1$

Consider points $P \in E(\mathbb{F}_{2^{283}})$ and $Q \in E(\mathbb{F}_{2^{1132}})$ of order l.

Weil Pairing $e_l(P, Q)$ time: 175.8 ms.

Tate Pairing $\langle P, Q\rangle^{(2^{1132}-1)/l}$ time : 57.19 ms (including the finite field exponentiation).

Notes:

1. These times show that cryptosystems based on the Tate pairing are completely practical for PC-based applications.
2. As explained in Section 5, the Weil pairing takes longer than twice the running time of the Tate pairing for the cryptographic applications.

10.2 Characteristic three timings

We now give timings for characteristic three.

Example 3:

Consider the elliptic curve $E : y^2 = x^3 - x + 1$ over

$$\mathbb{F}_{3^{97}} = \mathbb{F}_3[x]/\langle x^{97} + x^{16} + 1\rangle.$$

The group order is $N = 7l = 3^{97} + 3^{49} + 1$.

We took points $P \in E(\mathbb{F}_{3^{97}})$ and $Q \in E(\mathbb{F}_{3^{582}})$ of order l and computed the Tate pairing of order $7l$.

Tate Pairing: 168 ms (including finite field exponentiation)

Example 4:

Consider the elliptic curve $E : y^2 = x^3 - x + 1$ over

$$\mathbb{F}_{3^{163}} = \mathbb{F}_3[x]/\langle x^{163} + x^{80} + 2\rangle.$$

The group order is $N = 7l = 3^{163} - 3^{82} + 1$.

We took points $P \in E(\mathbb{F}_{3^{163}})$ and $Q \in E(\mathbb{F}_{3^{978}})$ of order l and computed the Tate pairing with respect to the order $7l$.

Tate Pairing: 581 ms (including the finite field exponentiation)

References

1. P. S. L. M. Barreto, H. Y. Kim and M. Scott, Efficient algorithms for pairing-based cryptosystems, Cryptology ePrint archive: Report 2002/008, 10 January 2002.
2. I.F. Blake, G. Seroussi and N.P. Smart. *Elliptic Curves in Cryptography.* Cambridge University Press, 1999.
3. D. Boneh and M. Franklin, Identity-based encryption from the Weil pairing, in J. Kilian (ed.) CRYPTO 2001, Springer LNCS 2139 (2001).
4. D. Boneh, B. Lynn and H. Shacham, Short signatures from the Weil pairing, in C. Boyd (ed.) ASIACRYPT 2001, Springer LNCS 2248, (2001) 514–532.
5. G. Frey and H.-G. Rück, A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves, *Math. Comp.*, **62**, No.206 (1994) 865–874.
6. G. Frey, M. Müller and H.-G. Rück, The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems, *IEEE Trans. Inform. Theory*, **45**, no. 5 (1999) 1717–1719.
7. S. D. Galbraith, Supersingular curves in cryptography, in C. Boyd (ed.) ASIACRYPT 2001, Springer LNCS 2248 (2001) 495–513.
8. D. Hankerson, J. Hernandez and A. J. Menezes, Software implementation of elliptic curve cryptography over binary fields, Proceedings of CHES 2000, Springer LNCS 1965 (2000), 1-24.
9. A. Joux, A one round protocol for tripartite Diffie-Hellman, in W. Bosma (ed.), ANTS-IV, Springer LNCS 1838 (2000) 385–393.
10. N. Koblitz, An elliptic curve implementation of the finite field digital signature algorithm, in H. Krawczyk (ed.), CRYPTO '98, Springer LNCS 1462 (1998) 327–337.
11. V. Miller, Short programs for functions on curves, unpublished manuscript 1986.
12. A. J. Menezes, T. Okamoto and S. A. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Trans. Inf. Theory*, **39**, No. 5 (1993) 1639–1646.
13. K. G. Paterson, ID-based signatures from pairings on elliptic curves, Cryptology ePrint archive: Report 2002/004.
14. R. Sakai, K. Ohgishi and M. Kasahara, Cryptosystems based on pairing, in SCIS 2000, Okinawa, Japan, January 2000.
15. J. H. Silverman, The arithmetic of elliptic curves, Springer GTM 106, 1986.
16. N. P. Smart, An identity based authenticated key agreement protocol based on the Weil pairing, Cryptology ePrint archive: Report 2001/111.
17. E. R. Verheul, Evidence that XTR is more secure than supersingular elliptic curve cryptosystems, in B. Pfitzmann (ed.), EUROCRYPT 2001, Springer LNCS 2045 (2001), 195–210.
18. E. R. Verheul, Self-blindable credential certificates from the Weil pairing, in C. Boyd (ed.), ASIACRYPT 2001 Springer LNCS 2248 (2001) 533–551.