Cryptographic Pairings on Elliptic Curves

Colin D. Walter

Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom
Colin.Walter@rhul.ac.uk

Abstract. Pairings on elliptic curves provide many interesting new protocols and services that are not available within classical asymmetric cryptography. These notes provide a more in depth description of the main components necessary for building such systems for those wishing to take the lecture material further. It contains a more detailed discussion of the enhanced arithmetic required.

1 Motivation - Some Applications

A standard problem with asymmetric public key cryptography is the man-in-the-middle attack. He replaces public key parameters of others with his own, enabling him to read and modify messages which he intercepts. Users therefore need a guarantee that they have public key parameters correctly. One solution is for an organisation to use the same parameters throughout its systems. They can be distributed very securely, once for all, and they take little space. Traditionally a different key is generated and distributed for every new member, revoked for retiring members, and it takes time and space as well as being less secure.

In order to solve this, suppose the company were to pick a group G on which all employees are to perform the cryptography, with individual i having public encryption exponent e_i. With classical RSA or ECC, the encrypt/decrypt keys d_i, e_i satisfy $d_i e_i = 1 \bmod |G|$. Then j can deduce i's private key from e_i since he knows $|G|$. So this cannot solve the previous problem.

Pairings on elliptic curves enable one to circumvent this problem, the first solution (1999) being given by Ohgishi, Sakai & Kasahara [23]. Independently Boneh & Franklin [7] discovered an almost identical technique (2001) but also provided a security proof. (A *security proof* is a proof that breaking the system is equivalent to solving a mathematical problem which is generally regarded as being "hard".) With this solution there are still public parameters which must be distributed securely to everyone by the KDC (key distribution centre) and individuals need to obtain their decryption key(s) from the KDC, but it is no longer possible to deduce decrypt keys from other (d, e) pairs.

Such a scheme must have a sufficiently large number of possible encryption keys that they can be generated from a random string. This means that, in practice, the organisation can specify formats of e for which the KDC will provide decryption keys and senders just need to follow those formats. In particular, e may have to be the recipient's email address. However, it might include further attributes such as the current date. This would enable the KDC to deny addressees the decrypt key if they have left the company before that date. So a revocation list which all should access is replaced by recipients regularly renewing their decryption keys. Other attributes can be included to avoid a plethora of email addresses for different groups of members. This answered positively a question posed by Shamir [28] about using one's identity as a public key[1].

An example would be a national health service. A patient takes his health card to a pharmacy, which then signs a request to the patient's doctor for the required prescription and that is then sent securely from the surgery to the pharmacy. The three parties may never have met previously, but the process works (with the right protocols) without each party having to know parameters for everyone in the health service – which is so huge that the membership is continuously changing.

2 Definition of Bilinear Pairing

A bilinear pairing ("pairing" for short) is a map $e : G_1 \times G_2 {\rightarrow} G'$ for abelian groups G_1, G_2, G' (here written additively, additively & multiplicatively respectively) such that

i) $e(A{+}B, C) = e(A, C)e(B, C)$, and
ii) $e(A, C{+}D) = e(A, C)e(A, D)$

for all $A, B \in G_1, C, D \in G_2$. An easy example is to take the finite field $\mathbb{F}_p$ with, say $p = 7$ and generator 3. The multiplicative sub-group of non-zero elements has $p{-}1 = 6$ elements. This is G' for the example, and G_1 and G_2 are both the additive group of integers modulo 6, (which is the group of exponents of the generator 3 of $\mathbb{F}_7$.) Then the map $e(A, C) = 3^{AC} mod 7$ provides the pairing. Properties (i) and (ii) are easily checked. (An exercise for the reader.)

For us, G_1 and G_2 are subgroups or quotient groups of an elliptic curve, and G' is a group of roots of unity in a field (the "embedding" field), with $|G_1| = |G_2| = |G'|$. Note that

$$e(A, C) = e(A{+}0, C) = e(A, C)e(0, C) \ ,$$

so that $e(0, C) = 1$ for all $C \in G_2$. Similarly $e(A, 0) = 1$ for all $A \in G_1$. Also $e(nA, C) = e(A, C)^n = e(A, nC)$ for $n \in \mathbb{Z}$, including $n = 0$ (which we have just noted), and $e(aA, cC) = e(A, C)^{ac}$. We are only interested in pairings with two additional properties:

a) *non-degeneracy*: $e(A, C) \neq 1$ for some $A \in G_1, C \in G_2$.
b) *computability*: there is an efficient algorithm to determine $e(A, C)$ from A and C.

[1] In [28] Shamir used RSA to give an identity-based signature scheme.

We will not bother about property (a), which is for mathematicians to verify. One of the main goals here is to describe an algorithm satisfying (b) and cover a few special cases where the algorithm becomes more efficient.

For a curve with a pairing, we can translate discrete log problems from a curve subgroup G_1 defined over the field $K_0 = \mathbb{F}_q$ to the "embedding" field $K = \mathbb{F}_{q^k}$, i.e. that containing G'. Thus, suppose $Q = \alpha P$. Use the non-degeneracy of e to choose $R \in G_2$ with $h = e(Q, R) \neq 1$. Then $h = e(\alpha P, R) = g^\alpha$ where $g = e(P, R)$. Solving the discrete log problem (DLP) in G' yields α, which solves the DLP in G_1. This can be done using a (sub-exponential) index calculus method. So we need to know that the DLP is difficult in the embedding field. The MOV threshold arising from the complexity of this attack (Menezes, Okamoto, Vanstone [20,16]) means an embedding degree of at least 6 to 9 (depending on the required security) when the order q of the field over which the curve is defined has 192 bits. So the pairings are computed in the field $\mathbb{GF}(q^6)$ or $\mathbb{GF}(q^9)$ or an extension of it – at least 1536 bits, say.

However, the computations are increasingly expensive as the embedding degree rises. The need to keep it small severely restricts the curves over which the cryptography is practical because, on average, the embedding degree k has the same order as the group, i.e. $O(q)$, rather than 6, 9 or 12, say. The usable curves are called "*pairing-friendly*" curves (see [13,17]). The earliest to be constructed were the MNT curves. Given the difficulty of generating curves with large enough prime subgroups, small enough embedding degree and efficient arithmetic in the embedding field, as well as the work involved in determining the size of the elliptic curve and the potential for selecting an insecure curve, one should only consider choosing a recommended curve, i.e. a well-documented, well-studied curve with known properties and which is preferably accepted in some standard such as P1363.3 [17].

We won't cover the different families of such curves, but one must be chosen. Unfortunately, the associated finite fields tend not to have particularly efficient arithmetic known for them. There is essentially just one pairing, but it comes in various different guises: Weil, Tate, Eta, Ate, R-Ate pairings which may or may not be applicable to the chosen curve family (see [17], §A.13.2.) Here we concentrate on Tate pairings and MNT curves.

3 Joux's Three Party Key Agreement

This is another application of bilinear maps, and one that sparked the interest in them for constructive cryptography as opposed to their previous destructive use, e.g. in the DLP. In 2000, Joux presented a scheme for one-round, 3-party key agreement based on bilinear maps [18]. The usual Diffie-Helman scheme is between two parties and so takes further rounds to agree a secret key among more parties. In the Joux scheme, P is a public generator of $G = G_1 = G_2$. The three parties A, B, C have secrets $a, b, c \in \mathbb{Z}_r{}^*$ respectively where $r = |G'|$. The protocol is as follows:

1. A publishes aP, B publishes bP, C publishes cP.
2. A computes $e(bP,cP)^a = e(P,P)^{abc}$, B computes $e(aP,cP)^b = e(P,P)^{abc}$, C computes $e(aP,bP)^c = e(P,P)^{abc}$.

All parties now have the shared key $K = e(P,P)^{abc} \in G'$.

4 IBE – Identity-Based Encryption

There are many applications of pairings in modern cryptography. This motivational section provides a brief technical outline of the main one: an identity-based encryption system. It can be skipped, but it shows what arithmetic components we need to be able to implement.

There is a standard structure for defining such systems so that "proofs" of security (i.e. reductions to simpler, standard problems of equivalent complexity which are generally believed to be "hard") are clearer:

1. Setup
2. Extract
3. Encrypt
4. Decrypt

These are fairly self-explanatory. Setup chooses the public parameters such as the curve and a private master key. Extract generates private decrypt keys for different identities from the public and private parameters determined by Setup and an identity's public encrypt key. Encrypt and Decrypt are descriptions of the appropriate functions.

The encryption is best described in two stages, starting with a simplified one called BasicIdent. Suppose Setup has already generated a suitable curve with subgroup G_1 of prime order r, and given us a pairing e into G', also of order r. Let H, H' be two hash functions from $\mathbb{F}_2^n$ (i.e. n-bit texts) to G_1 and from G' back to $\mathbb{F}_2^n$ respectively. Setup also chooses a master secret $s \in \mathbb{Z}$ of order $|G_2|$, a random generator P of G_2, and computes the public parameter $P_{pub} = sP$.

Let ID $\in \mathbb{F}_2^n$ be the public encryption string associated with an identity. Extract first computes the hash

$$Q_{ID} = H(ID) \in G_1$$

and then generates ID's private key

$$d_{ID} = sH(ID) = sQ_{ID} \in G_1$$

where s is the master secret. Encryption starts with computing $Q_{ID} = H(ID)$ and $g_{ID} = e(Q_{ID}, P_{pub}) \in G'$ and choosing a random ρ. The ciphertext of message M is the pair

$$\rho P, M \oplus H'(g_{ID}^{\rho})$$

To DECRYPT pair (U, V), compute

$$\begin{aligned} V \oplus H'(e(d_{ID}, U)) &= (M \oplus H'(g_{ID}^{\rho})) \oplus H'(e(d_{ID}, \rho P)) \\ &= M \oplus H'(e(Q_{ID}, P_{pub})^{\rho}) \oplus H'(e(sQ_{ID}, \rho P)) \\ &= M \oplus H'(e(Q_{ID}, P)^{\rho s}) \oplus H'(e(Q_{ID}, P)^{\rho s}) = M \end{aligned}$$

Overall, a random number generator is required, computation of pairings (one per identity by the encrypter, one per message by the decrypter), two hashing functions, and an exponentiation in G' by the encrypter.

To obtain a scheme with proven security, this needs the application of a technique by Fujisaki & Okamoto (see [7] for details.) Let H_1, H_2 be two more hash functions on the message space, and $E_{ID}(M, r)$ the encryption of M under the basic scheme. Then the encryption of M under the full scheme requires a second random, say σ, and is the pair

$$E_{ID}(\sigma, H_1(\sigma, M)), H_2(\sigma) \oplus M$$

– essentially another blinding of M by a random σ. Decryption is as follows: Let U, V be the pair $E_{ID}(\sigma, H_1(\sigma, M))$ forming the first component and $W = H_2(\sigma) \oplus M$ be the second component in the full scheme. As before, the decrypter first decrypts the E_{ID} component by computing $V \oplus H'(e(d_{ID}, U))$ to recover σ. Then $H_2(\sigma) \oplus W = M$ reveals the message.

So there are no new arithmetic requirements in the full scheme, just some further hashing. Overall, from previous lectures in this course we know how to do everything with the possible exception of i) finding suitable curves, and ii) calculating the pairing. There are, of course, some small details regarding hashing to G_1 or G' instead of $\mathbb{B}^n$, but they are straightforward.

5 Elliptic Curve Pairings

5.1 Notation

In line with the introductory sections above, the following (standard) notation will be used:

- p is the characteristic of the finite fields of interest. Usually a small prime such as 2 or 3, or a large prime of at least 160 bits.
- $q = p^*$ is the power of p which is the order of the field over which the elliptic curve cryptography is performed. (Typically 160 or 192 bits but no more than 256 bits; normally prime or a power of 2 or may be 3.)
- $K_0 = \mathbb{F}_q$ is the finite field of order q in which the curve is normally defined to lie.
- $E(K_0)$ is the chosen elliptic curve defined over K_0.
- r the order of the largest prime subgroup of the elliptic curve $E(K_0)$ over K_0. By the Hasse bound, the curve has order close to q. We want q/r small, ideally $q = O(r)$. Typically $q/r < 24$.

- $G_1 = E(K_0)[r]$ is the elliptic curve subgroup of points defined over K_0 whose rth multiple is the point $\mathcal{O}$ at infinity. This is cyclic of order r because r is so large.
- k is the embedding degree, the smallest number such that r divides $q^k - 1$. Also called the *security multiplier*. It divides $r-1$, and, for practical purposes, is typically under 12 in applications, but greater than 1. Often $k = 6$ is chosen, but the recommended lower bound has risen slowly as theory has developed.
- $K = \mathbb{F}_{q^k}$ is the embedding field. It is the smallest extension of $K_0 = \mathbb{F}_q$ containing the rth roots of 1, i.e. K contains an element ζ such that $\zeta^r = 1$ but $\zeta^a \neq 1$ for all positive $a < r$.
- $G_2 = E(K)/rE(K)$ is the elliptic curve quotient group of points defined over K modulo the rth multiples.
- $G' = \mu_r$ is the group of rth roots of unity in $K = \mathbb{F}_{q^k}$, i.e. the set of elements $\zeta \in K$ such that $\zeta^r = 1$.
- $e : G_1 \times G_2 \rightarrow G'$ is the selected bilinear pairing.

5.2 Divisors

Pairing values are obtained by constructing a function from the elliptic curve into the embedding field which has a particular set of zeros and poles. (Zeros are where the function has the value 0, and poles are where it has the value ∞.) The concept of a *divisor* is straightforward, and simply summarises a function using just a formal representation of its set of zeros and poles. In order to write down the definitions correctly, we have to go up to the "algebraic closure" $\overline{K}$ of K but all calculations will be done in K or K_0, so that $\overline{K}$ can be ignored. The elliptic curves $E(K)$ and $E(K_0)$, defined over K and K_0 respectively, are just subgroups of the curve $E(\overline{K})$.

Definition 1. Divisors *are formal (finite) sums over* $\mathbb{Z}$ *of points on the curve* $E(\overline{K})$. *For example,* $D = \sum_{P\in E(\overline{K})} n_P(P)$ *where* $n_P\in\mathbb{Z}$ *for all* $P\in E(\overline{K})$. *The* degree *of* D *is* $\deg(D) = \sum_P n_P$ *and its* support *is* $\sup(D) = \{P \mid n_P \neq 0\}$.

Divisors form an additive group (in the obvious way) and (P) is the way of writing the divisor of degree 1 and support $P \in E(\overline{K})$. The obvious function f corresponding to divisor D in the definition is the "rational" function

$$f_D(x) = \frac{\Pi_{P\in E(\overline{K}),n_P>0}(x-P)^{n_P}}{\Pi_{P\in E(\overline{K}),n_P<0}(x-P)^{-n_P}}$$

This function has zeros of order n_P at P when $n_P > 0$ and poles of order $-n_P$ at P when $n_P < 0$.

If f is a function from $E(\overline{K})$ to $\overline{K}$, then $\mathrm{ord}_P(f)$ is the multiplicity of P as a zero of f, with negative numbers representing the multiplicity of P as a pole (i.e. the number of times $x-P$ is a factor in the numerator or denominator of $f(x)$). This yields a *principal* divisor $(f) = \sum_P \mathrm{ord}_P(f)(P)$. So the principal divisor

(f_D) for the example function f_D is the divisor D in the definition. Note that a non-zero constant multiple cf of f gives the same divisor: $(cf) = (f)$ because the two functions have the same zeros and poles. For $c \in \overline{K}^*$, the divisor (c) is the empty formal sum 0.

Definition 2. *Two divisors are equivalent if they differ by a principal divisor, i.e.* $D_1 \sim D_2$ *if, and only if,* $D_1 = D_2 + (f)$ *for some function* f.

The group of divisors of degree 0 modulo this equivalence is the *divisor class group* of E. Computation of the pairing will be done in this group.

Theorem 1. *Suppose* $D = \sum_P n_P(P)$ *has degree 0. Then* $D \sim 0$ *if and only if* $\sum_P [n_P]P = \mathcal{O}$, *i.e.* $D = (f)$ *for some function* f *if, and only if, the points in* D *(with multiplicity) add to* $\mathcal{O}$.

The first sum here is the formal sum defining a divisor, the second sum is a sum of points on the elliptic curve, and $\mathcal{O}$ is the point at infinity on the curve. For example, if $\ell_{PQ}(x, y)$ is the line through P and Q on $E(K)$, then $R = -P-Q$ is the third point at which the line meets $E(K)$ and the associated divisor is

$$(\ell) = (P)+(Q)+(R) - 3(\mathcal{O})$$

since the zero of the curve group, namely $\mathcal{O}$, is a pole of the line with order 3.[2]

Finally, for a function $f : E(\overline{K}) \to \overline{K}$ and divisor $D = \sum_P n_P(P)$, we can define $f(D)$ by

$$f(D) = \prod_P f(P)^{n_P}$$

where $f(P)$ is the value of f at P. This is a well-defined element of K^* if the support of (f) does not intersect that of D. Clearly, for a non-zero constant $c \in \overline{K}$, $(cf)(D) = f(D)$ if $\deg(D) = 0$ since all the instances of c cancel.

5.3 The Tate Pairing

Most of this section can be skipped by readers wishing to avoid any mathematical complexity. The key contents to note are the definitions of the Tate pairing in equation (1) and the pairing $e(*, *)$ in equation (2).

For r coprime[3] to $p = \mathrm{char}(K)$, the cyclic group of rth roots of unity in $\overline{K}$ is denoted μ_r. This is the set of elements $\zeta \in \overline{K}$ such that $\zeta^r = 1$. There are r of them and $\phi(r)$ have order r. This will be the group G' into which our pairing maps. The group from which the first argument of the pairing comes is

$$G_1 \;=\; E(K)[r] \;=\; \{P \in E(K) \mid [r]P = \mathcal{O}\}\ ,$$

[2] This is not obvious. It uses a theorem of algebraic geometry which says $\deg((f)) = 0$.

[3] Any $r > 0$ with this property is suitable. Only the applications need r to be a large prime factor of $E(K_0)$.

i.e. the subgroup of curve points defined over K whose order divides r. The group

$$rE(K) = \{[r]P \mid P \in E(K)\}$$

of rth multiples is used in defining the quotient group

$$G_2 \;=\; E(K)/rE(K)$$

from which the second argument of the pairing comes. This group has exponent r, i.e. the rth multiple of an element is the zero of the group. $G_1 = E(K)[r]$ and $G_2 = E(K)/rE(K)$ have the same order, but are not necessarily isomorphic although they are for the cases of interest to us.

Pick two points $P \in E(K)[r]$ and representative $Q \in E(K)$ of a class in $E(K)/rE(K)$. Since the divisor $r(P) - r(\mathcal{O})$ has degree 0, there is a function[4] f with that as divisor, i.e. an f such that $(f) = r(P) - r(\mathcal{O})$. Choose divisor $D \sim (Q) - (\mathcal{O})$ whose support is disjoint from that of (f)[5]. Usually, if $S \in E(K)$ is an arbitrary point, then it suffices to take $D = (Q{+}S) - (S)$. Here $f(D) \in K^*$ because of the definitions of f, D and the support property of D. The Tate Pairing is defined by

$$\langle P, Q\rangle = f(D) \in K^*/(K^*)^r \tag{1}$$

Note that $K^*/(K^*)^r$ is isomorphic to μ_r in a canonical way[6], so we can view the pairing as giving us a value in μ_r. The choice of Q and D from their classes means that this is only defined up to multiplication by an rth power. The main evaluation problems are therefore: constructing the function f of divisor $r(P) - r(\mathcal{O})$, and determining which rth root of unity is in the same class as $f(D)$.

An important observation is that $\langle P, Q\rangle \sim 1_K$ is trivial if $P, Q \in E(L)$ for any subfield $L \subset K$ which does not contain any non-trivial rth roots of unity μ_r. This is because f and D are defined over L and so $f(D) \in L \cap \mu_r = \{1\}$. In particular, for prime r and $P, Q \in E(K_0)[r]$ we have $\langle P, Q\rangle = 1$ if the embedding degree $k > 1$. This means that we normally choose one argument in $E(K_0)$ to simplify the arithmetic but must choose the other properly in $E(K)$ to avoid everything degenerating to 1.

As $|K^*| = q^k{-}1$, we can raise elements of K to the power $(q^k{-}1)/r$ to obtain an rth root of unity. This gives a canonical element in a coset of $K^*/(K^*)^r$ since any rth power is killed by raising it to the power $(q^k{-}1)/r$. It also avoids the problem of determining which root of unity is in the class of $\langle P, Q\rangle$. Hence we define the pairing

$$e(P, Q) = \langle P, Q\rangle^{(q^k-1)/r} \in \mu_r\ . \tag{2}$$

Computing the power on the right side is called the *final exponentiation.* The bilinearity properties of $\langle *, *\rangle$ (which we have assumed) are inherited by $e(*, *)$.

Finally, readers interested in the Weil pairing could look at Galbraith's account in [6], IX.6. There he defines that pairing in terms of evaluating two Tate

[4] In the next but one section there is a construction for this using Miller's algorithm.

[5] We can't take $D = (Q) - (\mathcal{O})$ since $\mathcal{O}$ is then in the support of both D and (f).

[6] The class of $\zeta \in \mu_r$ is mapped to ζ.

pairings $\langle P, Q\rangle$ and $\langle Q, P\rangle$. As the Weil pairing already maps into μ_r, there is no need for the "final exponentiation". So it appears to require less than twice the computational effort. However, in the Tate pairing we can usually pick P in $E(K_0)$ but not Q in $E(K_0)$. When we come to Miller's algorithm for the Tate pairing, $\langle Q, P\rangle$ is then much more expensive than $\langle P, Q\rangle$ to compute. So Tate's pairing is very much faster and so generally preferred.

6 Computing the Pairing

6.1 Example

Two lines are used to describe the addition law $P+Q = S$ on an elliptic curve over K. The first is that through P and Q, say ℓ_{PQ}. It intersects the curve again at a point R, say, such that $P+Q+R = \mathcal{O}$. The second, say v_R, is the vertical line through R and $\mathcal{O}$, which intersects the curve again at $S = -R$, making $R+S = \mathcal{O}$. Specifically, $\ell_{PQ}(x, y) \equiv rx+sy+t$ for some $r, s, t \in K$ such that $rx_P+sy_P+t = 0$ and $rx_Q+sy_Q+t = 0$. Similarly, if the curve has an equation of the form $y^2 = f(x)$ then $v_R(x, y) \equiv x-x_R$, which is satisfied by R and $S = -R = (x_R, -y_R)$[7].

If we are using Weierstraß coordinates then lines over K_0 have equations of the form $\ell(x, y) = 0$ where $\ell(x, y) = rx+sy+t$, say, for some $r, s, t \in K$. The equation of the line gives a formula for y in terms of x. When this is substituted into the curve equation it yields a cubic which has three roots, namely x_P, x_Q and x_R for ℓ_{PQ}. Hence ℓ_{PQ} determines the three points P, Q, R and the divisor (ℓ_{PQ}) includes $(P)+(Q)+(R)$. The full mathematical definition of divisors is more complex and some hard work is required to show that there is a balancing multiple of $(\mathcal{O})$. However, we could eventually show that

$$(\ell_{P,Q}) = (P)+(Q)+(R)-3(\mathcal{O})$$

and, similarly, that

$$(v_R) = (R)+(S)-2(\mathcal{O}) = (R)+(-R)-2(\mathcal{O})$$

Since the divisor provides the multiplicities of the zeros and poles of a function, we have $(fg) = (f)+(g)$. Hence

$$(\ell_{PQ}/v_R) \;=\; (P)+(Q)+(R)-3(\mathcal{O}) - (R)-(S)+2(\mathcal{O}) \;=\; (P)+(Q)-(S)-(\mathcal{O})$$

6.2 Miller's Algorithm

The main step in evaluating the pairing is to construct is a function f such that $(f) = r(P) - r(\mathcal{O})$. This is done using a sequence of functions $f_1, f_2, ..., f_r$ with the divisor property

$$(f_i) = i(P) - ([i]P) - (i-1)(\mathcal{O}) \;. \tag{3}$$

[7] We cannot always pick the equation to have this form, in which case we can expect $y_{-R} \neq -y_R$.

Using the construction in the example of §6.1, there are two lines ℓ, v which define a group addition. For a given point $P \in E(K)[r]$, let ℓ_{ij}, v_{i+j} be those associated with the addition $[i]P + [j]P$. Thus ℓ_{ij} is the line through $[i]P$ and $[j]P$ whereas v_{i+j} is the line through $[i{+}j]P$ and $-[i{+}j]P$. In the same way as the roots of a polynomial determine the polynomial up to a non-zero multiple, so a divisor of degree 0 determines a function which is unique up to multiplication by a non-zero constant. Hence the functions f_i are determined up to a constant. They can be generated iteratively using:

Lemma 1. *The functions f_i satisfy:*
$f_1 = c_1$, *and*
$f_{i+j} = c_{ij} f_i f_j \ell_{ij} {v_{i+j}}^{-1}$
for some constants c_1, c_{ij} for $1 \le i, j \le r$.

To see this, note first that the divisor $(f_1) = 1(P){-}([1]P){-}(1{-}1)(\mathcal{O}) = \phi$ is the empty formal sum by eqn. (3). Hence f_1 is just a non-zero constant function.

Next, we have

$$(\ell_{ij}/v_{i+j}) \;=\; ([i]P) + ([j]P) - ([i{+}j]P) - (\mathcal{O})$$

by the example in the previous sub-section. Thus, by the obvious induction hypothesis that eqn. (3) holds for all indices less than $i{+}j$,

$$\begin{aligned}(f_i f_j \ell_{ij}/v_{i+j}) &= i(P){-}([i]P){-}(i{-}1)(\mathcal{O}) + j(P){-}([j]P){-}(j{-}1)(\mathcal{O}) + (\ell_{ij}/v_{i+j}) \\ &= i(P) - (i{-}1)(\mathcal{O}) + j(P) - (j{-}1)(\mathcal{O}) - ([i{+}j]P) - (\mathcal{O}) \\ &= (i{+}j)(P) - ([i{+}j]P) - (i{+}j{-}1)(\mathcal{O}) \\ &= (f_{i+j})\end{aligned}$$

which establishes the claim by induction and the preceding remark. □

Observe that $P \in E(K)[r]$ means $[r]P = \mathcal{O}$, so that $(f_r) = r(P){-}r(\mathcal{O})$ and $f_r = f$ is the function we seek in order to evaluate the pairing. Note also that for any point S, $(Q{+}S){-}(S) \sim (Q){-}(\mathcal{O})$ by Theorem 1 because $\{Q{+}S\}{-}S = Q{-}\mathcal{O}$. Hence, by definition, $\langle P, Q\rangle = f(D)$ for a divisor $D = (Q{+}S){-}(S)$ where S is a random point on the curve. So $\langle P, Q\rangle = f(Q{+}S)/f(S)$. This means the constants c_{ij} cancel in the iterative computation of $f(D)$:

Corollary 1. *For a divisor D of degree 0, the values $f_i(D)$ satisfy:*
$f_1(D) = 1$, *and*
$f_{i+j}(D) = f_i(D) f_j(D) \ell_{ij}(D) v_{i+j}(D)^{-1}$.

Since f_r is calculated iteratively using lines which compute multiples of P up to $[r]P$, this corollary means f can be computed simply by inserting some extra computation into any scalar multiplication algorithm for computing $[r]P$. In the following algorithm due to Victor Miller, the variable T holds a multiple of P, say $[j]P$ where j is given by the leftmost i bits of r, and the variable f contains the corresponding value $f_j(D)$ for $D = (Q'){-}(S) = (Q{+}S){-}(S)$.

```
ALGORITHM: Miller
INPUTS: Points $P, Q \in E(K)$, P of order $r = \sum_{i=0}^{n-1} r_i 2^i, r_i \in \mathbb{B}, r_{n-1}=1$.
OUTPUT: The Tate pairing $\langle P, Q\rangle \bmod K^{*r}$.
      Choose a random point $S \in E(K)$
      $Q' \leftarrow Q+S$
      $T \leftarrow P$
      $f \leftarrow 1_K$
      For i $\leftarrow$ n-2 downto 0 do
      {   Determine lines $\ell$ and $v$ for doubling $T$.
          $T \leftarrow [2]T$
          $f \leftarrow f^2 \ell(Q')\ell(S)^{-1}v(Q')^{-1}v(S)$
          If $r_i = 1$ then
          {   Determine $\ell$ and $v$ for the addition $T+P$.
              $T \leftarrow T+P$
              $f \leftarrow f\ell(Q')\ell(S)^{-1}v(Q')^{-1}v(S)$
          }
      }
      Return $f$
```

Recall the assumption in the definition of the pairing that the support of f had to be disjoint from the divisor $Q'-S$ at which f is evaluated. This is to avoid f becoming 0 when it is evaluated at any point in the divisor. f has at most about $\frac{15}{2}\log_2 r$ zeros or poles besides $\mathcal{O}$. (Three for each line ℓ and two for each vertical v for, on average, $\frac{3}{2}\log_2 r$ cases.) As r is large and there are at least r choices for S, there is therefore only a negligible chance of a random S resulting in Q' or S being part of the support of f. This would be noticed in the algorithm by an attempt to multiply or divide f by 0. In this case the algorithm would have to be run again with a different S.

There is, of course, still the final exponentiation by $(q^k-1)/r$ to perform to obtain $e(P, Q)$. We know how to do this, but it is a main topic in section 7.

6.3 Other Scalar Multiplication Algorithms

In fact, alternative scalar multiplication algorithms for $[r]P$ can be adapted to speed up the version of Miller's Algorithm presented in the previous section. For example, by using NAF (non-adjacent form) for which there are non-zero digits ± 1 which are always separated by at least one zero digit, the number of addition operations is reduced from $\frac{1}{2}\log_2 r$ to $\frac{1}{3}\log_2 r$ on average. Almost all efficient scalar multiplication algorithms use an iterative step which involves a number of doublings $T \leftarrow [2]T$ and an addition $T \leftarrow T+[d]P$ of a small, signed digit multiple dP. The latter uses f_d, so $[d]P$ and $f_d(D)$ should be pre-computed and stored in variables P_d and f_d, say. Then, for positive digits d, the addition step becomes:

```
If $d>0$ then
{   Determine $\ell$ and $v$ for the addition $T{+}P_d$.
    $T \leftarrow T{+}P_d$
    $f \leftarrow f\ f_d\ \ell(Q')\ell(S)^{-1}v(Q')^{-1}v(S)$
}
```

However, unless $d = 1$ the insertion of the factor f_d increases the cost of the addition step so much that alternatives to double and add are usually more expensive! This is confirmed in more detail below.

For negative digits d, the function f_d has not been constructed and so f_{i+d} is obtained from f_i and f_{-d} by re-ordering the property

$$f_i(D) = f_{i+d}(D)f_{-d}(D)\ell_{i+d,-d}(D)v_i(D)^{-1}$$

from Corollary 1 into ([8], eqn.(1))

$$f_{i+d}(D) = f_i(D)f_{-d}(D)^{-1}\ell_{i+d,-d}(D)^{-1}v_i(D)$$

Thus, the two lines of interest are associated with the addition of $[-d]P$ and $[i{+}d]P$. The line $\ell_{i+d,-d}$ goes through $[i{+}d]P, -[d]P$ and so also $-[i]P$. Thus it is the line $\overline{\ell}_{i,d} = \ell_{-i,-d}$ through the points $-[d]P$ and $-[i]P$ which is complementary to the line $\ell_{i,d}$ through $[d]P, [i]P$ and so essentially already known as $\pm[d]P$ was pre-computed and $T = [i]P$ is the current point. If the curve equation has the form $y^2 = F(x)$, then $\overline{\ell}_{i,d}$ is just the mirror image in the x-axis of the line $\ell_{-i,-d}$. We can refer to $\overline{\ell}_{i,d}$ as the line associated with the subtraction $T{-}P_{-d}$. It is, indeed, one of the two possible lines that can be used to determine the difference $T{-}P_{-d}$ and so causes no extra work beyond what is necessary to determine $T{+}P_d$. The required vertical line is v_i through $[i]P$ and $-[i]P$, i.e. T and $-T$. So the full addition step becomes:

```
If $d>0$ then
{   Determine $\ell$ and $v$ for the addition $T{+}P_d$.
    $T \leftarrow T{+}P_d$
    $f \leftarrow f\ f_d\ \ell(Q')\ell(S)^{-1}v(Q')^{-1}v(S)$
}
If $d<0$ then
{   Determine $\overline{\ell}$ for the subtraction $T{-}P_{-d}$ and $v$ through $T$.
    $T \leftarrow T{-}P_{-d}$
    $f \leftarrow f\ {f_{-d}}^{-1}\overline{\ell}(Q')^{-1}\overline{\ell}(S)v(Q')v(S)^{-1}$
}
```

In practice, only the digits ±1 are of interest. For these the factor $f_{\pm d} = 1$ disappears. So a NAF representation of r or a 2-bit sliding window method with digits $0, \pm1$ would provide some speed-up over the usual double-and-add methods.

7 Efficiency Aspects

7.1 The Main Efficiency Issues

The largest efficiency saving is from working in $K_0 = \mathbb{F}_q$ rather than $K = \mathbb{F}_{q^k}$ whenever possible. Representations of elements in K are k times the length of representations of elements in K_0 so that a multiplication over K_0 is $k{\times}k$ times faster than one over K. Although small values of k such as $k = 6$ are typical, the saving is nevertheless considerable if available choices enable many multiplications to be made over K_0 instead of K.

The focus of attention in this sub-section is the evaluation of the line

$$\ell(x,y) \equiv rx{+}sy{+}t \tag{4}$$

at S and Q'. In Miller's algorithm, if $P \in E(K_0)$ then all the lines are over K_0 and point additions are performed over K_0. Thus the coefficients of (4) are in K_0 and calculating the coordinates of multiples of P only requires arithmetic in K_0. Moreover, S can be chosen so that either S or Q' is defined over K_0. Then half the line evaluations $\ell(S)$ and $\ell(Q')$ in the algorithm are at points defined over K_0, and once again all the arithemtic is in K_0. In fact, if K_0 is not a prime field (i.e. $\mathrm{char}(K) < q$) then S might be chosen in a much smaller subfield than K_0, making multiplication using its coordinates still cheaper.

The cost of one point doubling or point addition step in Miller's algorithm is roughly the following, when field additions and multiplications by small constants are ignored. There are typically 12 to 17 field multiplications per point addition, depending on the coordinate system (see [6], Table V.1), and this should yield both lines with little extra work since ℓ is essentially determined when calculating a point sum. It is comparable but a little less for point doubling – typically 80% of the cost of a point addition. All these should be over K_0. Another 12 multiplications are required to evaluate the two lines at the two points when projective coordinates are used. Six of these will be over K_0 if S or Q' is chosen to be defined over K_0. However, the other six will be mixed multiplications with one argument in K_0 and the other in K, making them k times the cost of a multiplication over K_0. Finally, 4 or 5 multiplications or divisions (for the point addition and doubling respectively) are required to determine f from those evaluations. Since $f \in K$, most of these operations are properly in K.

Temporarily counting a division to have the same cost as a multiplication[8], over half of the multiplications of an addition or doubling step are therefore over K_0 instead of K if $P, S \in E(K_0)$, namely those for the point addition or doubling and two of the line evaluations, and most of the rest have at least one argument in K_0. Only 3 or 4 are fully over K. Overall, choosing $S \in E(K_0)$ is useful but much less significant than choosing $P \in E(K_0)$. Consequently, although one has to keep K large enough to avoid a DLP attack via K, K_0 should be kept down to the size used in classical ECC applications and P chosen in $E(K_0)$.

[8] Division is treated properly in the following paragraphs.

Clearly, field divisions and inversions should be avoided. Any constant multiple of ℓ or v is acceptable because their contributions to f are via $\ell(D) = \ell(Q')/\ell(S)$ and $v(D) = v(Q')/v(S)$, so that any extra constant is cancelled. Hence computing the coefficients of lines ℓ and v, and determining the values $\ell(Q')$, $\ell(S)$, $v(Q')$ and $v(S)$ should not involve any division.

f is computed using a similar approach to that of replacing affine with projective coordinates, *viz.* adding another variable to store values as ratios: f is split into numerator and denominator values f_N and f_D for the main loop. It requires twice the storage for f, which is little overall increase in memory. Naturally, the table elements $f_d(D)$ would have any divisions performed before storage so their denominators would also be 1. Then each division in the computation of f can be replaced by a single multiplication. Thus, for example, in the addition step the updating of f becomes

$$f_N \leftarrow f_N \ell(Q')v(S) \; ; \quad f_D \leftarrow f_D \ell(S)v(Q') \tag{5}$$

which involves two multiplications of elements in K_0 by elements in K and two multiplications properly over K. (The same number of multiplications as we counted earlier.) Eventually there is a single field division $f = f_N/f_D$ after the loop terminates.

There are, of course, a number of papers describing how to save one or two multiplications per double and add step in scalar multiplication algorithms, e.g. [8,9].

7.2 Alternative Scalar Multiplication Algorithms

This section can be skipped. It is mostly an interesting exercise in understanding the relative merits of other likely sources of improved efficiency and should be helpful in developing general skills for tackling this issue in other contexts.

Our aim here is to reduce the total number of multiplications properly over K since this is the most expensive arithmetic operation aside from the single final division after the loop termination. These multiplications dominate Miller's algorithm. Alternative point multiplication algorithms which involve digits other than $0, \pm 1$ require an extra multiplication by $f_{\pm d} \in K$ in the point addition step. For example, for $d > 1$, f is updated using

$$f_N \leftarrow f_N f_d \ell(Q')v(S) \; ; \quad f_D \leftarrow f_D \ell(S)v(Q')$$

This increases the number of multiplications with both arguments properly in K from 2 to 3 compared with when $d = 1$. By comparison, the doubling step

$$f_N \leftarrow {f_N}^2 \ell(Q')v(S) \; ; \quad f_D \leftarrow {f_D}^2 \ell(S)v(Q')$$

requires two squarings and two multiplications properly over K, which is typically equivalent to about 3.6 such multiplications.

With a prime r of 160 bits, there are on average 159/2 addition steps in the binary double and add algorithm, requiring $2 \times \frac{1}{2} \times 159 = 159$ multiplications over

K. With a NAF representation, one in three digits is non-zero and so the addition steps require on average only $2\times\frac{1}{3}\times 159 = 106$ multiplications. For a 3-bit sliding window algorithm, the digits are $\pm1, \pm3$ and one in four of the doubling steps is associated with a non-zero digit. So one in eight involves ±3 and requires an extra multiplication. The total is $(\frac{2}{8}+\frac{3}{8})\times 159 \approx 99$ multiplications for the addition steps. The saving of under 7 multiplications over NAF is insufficient to perform the extra division necessary when computing $f_3(D)$ for the table. Of course, the saving is proportional to the bit length of r, and will be more for larger r. One concludes that only scalar multiplication methods involving digits $0, \pm1$ are worthwhile for the usual size of r.

7.3 Another Less Efficient Method

This section may also be skipped – it reveals another slight efficiency gain with little extra expense in memory use.

Similarly to the splitting of f into numerator and denominator, the field multiplications for updating f might be separated into those over K_0 and those over K. Suppose variables $f_0 = f_{0N}/f_{0D}$ and $f_1 = f_{1N}/f_{1D}$ are used for these, and they are multiplied together at the end to obtain $f = f_N/f_D$. Again ignoring the inversions, each iterate of the doubling step

$$f_N \leftarrow {f_N}^2 \ell(Q')v(S) \ ; \quad f_D \leftarrow {f_D}^2 \ell(S)v(Q')$$

is split into

$$f_{0N} \leftarrow {f_{0N}}^2 v(S) \ ; \quad f_{0D} \leftarrow {f_{0D}}^2 \ell(S) \ ; \quad f_{1N} \leftarrow {f_{1N}}^2 \ell(Q') \ ; \quad f_{1D} \leftarrow {f_{1D}}^2 v(Q')$$

Both require two multiplications and two squarings over K. In addition, the first has two mixed multiplications $K\times K_0 \rightarrow K$ but the second only two multiplications and two squarings over K_0, which is cheaper for $k \geq 2$. However, the saving is only $O(k)$ per loop iteration in Miller's algorithm, whereas the total cost is $O(k^2)$ per loop iteration. Notice the importance of performing multiplications in an order which keeps successive products in the smallest possible field.

7.4 Affine or Projective Coordinates?

Next to consider is the choice of coordinates. Only S and Q' appear in the computation of f, and, being random, S can easily be chosen over K_0 with affine coordinates. As neither is changed during the computation, it is best also to put Q' into affine coordinates initially and stay with affine coordinates for the calculation of f, thereby making the line evaluations at S and Q' cheaper.

For the iterative calculation of $[r]P$ it is tempting to use projective coordinates. One of their main uses is to speed up exactly this computation. Observe, however, that the line v through the point T (which is the variable in Miller's algorithm holding a scalar multiple of P) then has x-coordinate given by the ratio x_T/z_T, so that the evaluation of $v \equiv z_T x - x_T z$ at S or Q' requires a multiplication which was not required before, the latter case with one argument in K. These multiplications cause an extra computational effort of $O(k)$ per loop iteration. However, the additional cost of affine coordinates for T is a division in K_0, which, as a function of k, is an $O(1)$ operation. Consequently, according to Galbraith [6] chap. IX, affine coordinates are empirically faster. Indeed, the analysis here shows that this is inevitable as k grows larger.

8 The Final Exponentiation

Recall that our aim is to compute $e(P,Q) = \langle P,Q\rangle^{(q^k-1)/r}$, and Miller's algorithm only outputs $\langle P,Q\rangle$.

8.1 Removing the Factors involving S

In many cases, part of the work in calculating the pairing may be saved using the following theorem.

Theorem 2. *[2] For the evaluation of* $e(P,Q)$*, if* r *is prime to* $q{-}1$ *and* $k > 1$ *then* $S = \mathcal{O}$ *can be chosen in Miller's Algorithm and the factors* $\ell(S)$ *and* $v(S)$ *deleted from it.*

In practice, as we have noted, S is generally chosen in $E(K_0)[r]$ to ensure the calculations of $\ell(S)$ and $v(S)$ are entirely within K_0. However, the calculations of $\ell(Q')$ and $v(Q')$ are in K and so much more expensive. Hence the saving from this theorem is not as great as the half that might be expected at first sight, but it is one of the improvements that makes other pairings, such as the *ate* pairing, attractive.

The main point here is that, in order to calculate $e(P,Q)$, the final exponentiation by $(q^k{-}1)/r$ is by a multiple of $q{-}1$. Ignoring any problems regarding the support of f, by picking $S \in E(K_0)$ we would have $\ell(S) \in K_0$ and $v(S) \in K_0$ and so $\ell(S)^{q-1} = 1$ and $v(S)^{q-1} = 1$ unless $\ell(S) = 0$ or $v(S) = 0$. So their contribution after the final exponentiation is trivial, and the factors $\ell(S)$ and $v(S)$ can be safely deleted from the algorithm. Of course, in our case K_0 is chosen small, so $k > 1$. By the definition of r, r will therefore generally be prime to $q{-}1$, so that normally this theorem can be applied.

The restriction on the choice of S is that it is not in the support of any function f that is used. However, only $O(\log(r))$ points of $E(K_0)[r]$) occur in such supports as only that number of functions f are used in the calculation. So there are points S defined over K_0 which could be used legitimately and which avoid the support of the function f. Unfortunately, we need to be a bit more clever since we want to take $S = \mathcal{O}$ in order to have $Q' = Q$, and this is one of the points in the support of f. As the course is not about proofs, so we will skip this tiresome detail.

Theorem 3. *[2] When computing* $e(P,Q)$*, if* r *is prime to* $q{-}1$ *and* $k > 1$ *then in Miller's Algorithm all the factors of* f*, namely* f_d*,* $\ell(Q')$*,* $\ell(S)$*,* $v(Q')$ *and* $v(S)$*, can be replaced by multiples* $\alpha_1 f_d$*,* $\alpha_2\ell(Q')$*,... for any* $\alpha_1, \alpha_2, ... \in K_0^*$*.*

The reasoning is the same here – each α_j is killed by the final exponentiation. Since the line ℓ will normally have a denominator in its gradient, a consequence of this theorem, in particular, is that the line ℓ can have any denominators safely removed to avoid divisions. Generally, the multiples $[i]P$ of P will end up having "rational" coordinates (i.e. a ratio of a numerator and a denominator) so that the denominators infect all the calculations. This theorem enables all of them to be avoided.

8.2 The Frobenius Automorphism

The final exponentiation by $r' = (q^k - 1)/r$ can sometimes benefit from the Frobenius automorphism on K. If $p = \mathrm{char}(K)$ is the characteristic of K, then the *Frobenius automorphism* is the map $\gamma \to \gamma^p$ for $\gamma \in K$. This is used for fields of very small characteristic or those for which the exponent has few non-zero digits modulo the characteristic.

If $\mathbb{F}_m$ is a subfield of K with order $m = p^j$ for some j, then elements of K can be stored using a representation of K as a vector space over $\mathbb{F}_m$. If $\gamma = \sum_{i=0}^{n-1} \gamma_i b_i, \gamma_i \in \mathbb{F}_m$, is the representation of $\gamma \in K$ using basis $(b_0, b_1, ..., b_{n-1})$ over $\mathbb{F}_m$ then $\gamma^m = (\sum_{i=0}^{n-1} \gamma_i b_i)^m = \sum_{i=0}^{n-1} \gamma_i^m b_i^m = \sum_{i=0}^{n-1} \gamma_i b_i^m$ because, for the second equality, all the binomial coefficients of the omitted terms are multiples of the characteristic p, and, for the third equality, the order $m-1$ of $\mathbb{F}_m^*$ means $\gamma_i^m = \gamma_i$ by Lagrange's theorem. Thus,

- Raising to the power of the characteristic requires no general field multiplications, only constant multiplications and additions.

The values of b_i^m are pre-computed and used for this – in any application of pairing-based cryptography the field K is usually fixed and so this is generally done once in the lifetime of the application. The b_i^m are linear combinations of the original basis elements, hence the constant multiplications and additions. With careful choice of the field representation, such as a *normal basis*, the constants are small or even trivial.

To take advantage of the cheap Frobenius automorphism, there is a base m analogue of square-and-multiply, namely m-ary exponentiation. Each iteration raises the value so far to the power m, and multiplies in the pre-computed value corresponding to the coefficient:

```
ALGORITHM: $m$-ary Exponentiation
INPUTS: $\gamma \in K$, $r' = \sum_{i=0}^{n-1} r_i m^i$.
OUTPUT: $\gamma^{r'}$.
      Pre-compute and store the occurring values $\gamma^{r_i}$
      $\Gamma \leftarrow \gamma^{r_{n-1}}$
      For i $\leftarrow$ n-2 downto 0 do
        { $\Gamma \leftarrow \Gamma^m$; if $r_i \neq 0$ then $\Gamma \leftarrow \Gamma \times \gamma^{r_i}$ }
      Return $\Gamma$
```

For small characteristic and $m = \mathrm{char}(K) > 2$ this is normally cheaper than a windowing method using the base 2 representation of r' because raising to the power m is so cheap and squaring relatively expensive. Note also, of course, that squaring is cheap when $\mathrm{char}(K) = 2$.

The same cheapness is true for larger characteristics providing that only a few different small digits appear in the base m representation of the exponent $r' = (q^k{-}1)/r$. This turns out to be almost the case for many constructions, which tend to concentrate pairing-friendly curve sizes at the limits $q{\pm}2\sqrt{q}{+}1$ or at natural division points of Hasse's bounds $q{-}2\sqrt{q} \le |E(K_0){-}1| \le q{+}2\sqrt{q}$ on the size of the elliptic curve. (All the super-singular curves have this property, for example – see [6], Thm. IX.20.) Thus a representation of $n = |E(K_0)|$ to base q or $\sqrt{q}$ is often possible with just two or three tiny digits. (Recall that r is usually a very large prime factor of n.) We can convert the exponentiation to the power r' into one to the power $n' = (q^k{-}1)/n$ if n divides $q^k{-}1$: for $P{\in}E(K_0)[r]$, $n = |E(K_0)|$ dividing $q^k{-}1$ and the obvious notation, the property

$$e(P,Q) = \langle P,Q\rangle_r^{(q^k-1)/r} = \langle P,Q\rangle_n^{(q^k-1)/n}$$

holds. (The two Tate pairings are those related to the subgroups of points with order dividing r and n respectively.) Since the co-factor n/r is preferably very small, there is a good chance that $n = |E(K_0)|$ will divide $q^k{-}1$, so that we can compute $\langle P,Q\rangle_n^{(q^k-1)/n}$ instead. When n/r is small this requires only a few more iterations in Miller's algorithm. Moreover, the form of n means $n' = (q^k{-}1)/n$ has a good chance of having a sparse representation in base $p = \mathrm{char}(K)$, which is what is required to obtain the greatest benefit from the Frobenius automorphism.

Example
For $q = 3^{163}$ there is a super-singular curve of order $n = q{-}\sqrt{3q}{+}1 = 3^{163}{-}3^{82}{+}1 = 7r$ for a prime r and its embedding degree is 6 since n is a factor of $q^2{-}q{+}1$ which divides the factor $q^3{+}1$ of $q^6{-}1$. So we can evaluate the pairing using n rather than r. The final exponentiation is then by $n' = (q^6{-}1)/n = (q^3{-}1)(q{+}1)\overline{n}$ for $\overline{n} = q{+}\sqrt{3q}{+}1 = 3^{163}{+}3^{82}{+}1$. Hence there is a base 3 representation of n' with at most $2{\times}2{\times}3 = 12$ small non-zero digits. This makes exponentiation by n' extremely fast.

Note further that m-ary exponentiation could be used advantageously in Miller's algorithm. For example, in characteristic 3, direct point tripling is cheaper than forming $[3]P$ from $P{+}[2]P$ (*see* §9.2). Moreover, the scalar r can be written in base 3 using only the digits $0, \pm1$. As illustrated in the example, the chosen number of iterations n may also have almost no non-zero digits in its base 3 representation. So there could be considerable savings.

9 Pairing-Friendly Curves

The need for "pairing-friendly" curves has already been noted. They have a small embedding degree, and usually a large prime factor in the curve order. There are now a number of different families of such curves. A good reference is the "Taxonomy" paper of Freeman, Scott and Teske [13]. Two representative cases are studied here, namely the super-singular curves and the MNT curves. There are a number of individual curves that have been studied in the research literature and for which there are speed details and, hopefully, also the necessary implementation details for achieving the claimed speed. The IACR ePrint library is a good source for these, e.g. [5]. There is no obvious reason not to choose such curves when they belong to families for which there are no security reservations.

9.1 Super-Singular Elliptic Curves

Definition 3. *An elliptic curve E defined over a field $\mathbb{F}_q$ of characteristic p is* super-singular *if $p \mid t$, where $t = q+1-|E(\mathbb{F}_q)|$. If $p \nmid t$ then E is* ordinary.

So the number of finite points is a multiple of the characteristic on super-singular curves. A list of popular super-singular curves is given in [6] Table IX.1. Recall that the *trace* t is bounded by $|t| \leq 2\sqrt{q}$ according to Hasse's bound. However, the only choices for super-singular curves satisfy:

- $t = 0$ with $k = 2$,
- $t = \pm 2\sqrt{q}$ with $k = 1$,
- $t = \pm\sqrt{q}$ if $\mathrm{char}(\mathbb{F}) \equiv 0, 2 \pmod 3$ with $k = 3$,
- $t = \pm\sqrt{2q}$ if $\mathrm{char}(\mathbb{F}) = 2$ with $k = 4$, or
- $t = \pm\sqrt{3q}$ if $\mathrm{char}(\mathbb{F}) = 3$ with $k = 6$.

If $r > 4\sqrt{q}$ is a large prime divisor of the order of a super-singular curve over K_0 then r^2 exactly divides $|E(K)|$, $E(K)[r]$ is the direct product of two subgroups of order r and $E(K)[r] \cap rE(K) = \{\mathcal{O}\}$. This means that there are no difficulties between the two arguments $E(K)[r]$ and $E(K)/rE(K)$ of the pairing: by restricting the latter to classes of $E(K)[r]$ we just get $E(K)[r]$ for both arguments. There is also a nice "distortion" map which maps $E(K_0)[r]$ to another subgroup of $E(K)$ with order r (corresponding to the image of the curve over a conjugate of K_0) so that the two generate the full group $E(K)[r]$ on which the pairing is of interest. The advantage of this is that half the work of a pairing calculation is avoided: all the denominators can be ignored – see §9.3 below and [2].

The most frequently used constructions for these curves are the following:

- $E : y^2 = x^3+a$ over $\mathbb{F}_p$, where $p \equiv 2 \pmod 3$.
  Here $|E(\mathbb{F}_p)| = p+1$, $t = 0$, $k = 2$, and the distortion map is $(x, y) \mapsto (\zeta_3 x, y)$ where ζ_3 is a primitive cube root of 1.
- $E : y^2 = x^3+x$ over $\mathbb{F}_p$, where $p \equiv 3 \pmod 4$.
  Here $|E(\mathbb{F}_p)| = p+1$, $t = 0$, $k = 2$, and the distortion map is $(x, y) \mapsto (-x, iy)$ where i is a primitive fourth root of 1.
- $E : y^2 = x^3+a$ over $\mathbb{F}_{p^2}$, where $p \equiv 5 \pmod 6$ and $a \in \mathbb{F}_{p^2}$ is a square but not a cube. Here $|E(\mathbb{F}_{p^2})| = p^2-p+1$, $t = -p$, $k = 3$, and the distortion map is $(x, y) \mapsto (\gamma b x^p, b y^p)$ where $b = a^{-(p-1)/2}$ and $\gamma^3 = a^{-1}$ for $\gamma \in \mathbb{F}_{p^6}$.

- $E : y^2{+}y = x^3{+}x{+}a$ where $\mathbb{F}_q, q = 2^l$ for odd l and $a \in \mathbb{F}_2$.
  Here $|E(\mathbb{F}_q)| = q \pm \sqrt{2q}{+}1$ according to the choice of a, $k = 4$, and the distortion map is $(x, y) \mapsto (u^2x{+}s^2, y{+}u^2sx{+}s)$ where $u^2{+}u{+}1 = 0$ for $u \in \mathbb{F}_4$ and $s^2{+}(u{+}1)s{+}1 = 0$ for $s \in \mathbb{F}_{16}$.
- $E : y^2 = x^3{-}x{+}a$ where $\mathbb{F}_q, q = 3^l$ for odd l and $a \in \mathbb{F}_3^*$.
  Here $|E(\mathbb{F}_q)| = q \pm \sqrt{3q}{+}1$ according to the choice of a, $k = 6$, and the distortion map is $(x, y) \mapsto (\alpha{-}x, iy)$ where $i \in \mathbb{F}_9$ is a fourth root of 1 and $\alpha \in \mathbb{F}_{27}$ satisfies $\alpha^3{-}\alpha{-}a = 0$.

Suitable fields and curves with these parameters are found as follows. First, systematically choose random numbers p of the required magnitude for the characteristic, applying the Rabin-Miller primality testing algorithm to check primality, and then running it again on r where r is obtained from $n = |E(K_0)|$ using the above formula, and removing all small factors. Note that Rabin-Miller need only be run initially enough times to give a reasonable chance that p is prime. If r turns out to have a high probability of being prime, then p can be checked further. Typically a sieve of Eratosthenes is used on a large interval of numbers p with a given residue mod 210, say, in which all elements with a prime divisor less than 2^{16}, say, have been tagged. Rabin Miller is run a few times on the surviving elements. The probability of both p and n being prime is $O(\log(n)^{-1}\log(p)^{-1})$, and so suitable primes p are not difficult to find. Assuming you can already construct the field K_0, the embedding field here is given for cases with $\mathrm{char}(K_0) \neq 2, 3$ by adjoining the elements α, i, γ and ζ_3, as appropriate.

For $\mathrm{char}(K_0) = 2, 3$ the choice of fields is very limited: for $q = 2^l$ or $q = 3^l$ there are very few reasonable cases where $|E(\mathbb{F}_q)|$ has a large enough prime factor; $q = 3^{163}$ is a case mentioned above. There are tables of irreducible polynomials for generating finite fields from which $\mathbb{F}_{2^{251}} = \mathbb{F}_2[t]/(t^{251}{+}t^7{+}t^4{+}t^2{+}1)$, for example (e.g. [27]). Note that fast multiplication methods, such as Karatsuba-Ofman and Toom, are useful for typical fields in elliptic curve cryptography.

9.2 Characteristic 2 and 3 fields

This is a brief aside on the connection between $P \mapsto [p]P$ on a super-singular curve with characteristic p and the Frobenius map $\alpha \mapsto \alpha^p$ when $p = 2, 3$.

Take char(K)=2 first, with $E : y^2{+}y = x^3{+}x{+}a$ so that $k = 4$. Take $P = (x_1, y_1) \in E$. The tangent at P is $\ell : y = \lambda(x{-}x_1){+}y_1$ where $\lambda = x_1^2{+}1$. This intersects the curve again at $x_2 = \lambda^2 = x_1^4{+}1$. The vertical line through this is $v : x{-}x_2 = 0$. So $[2]P = (x_2, y_2)$ where $y_2 = 1{+}\lambda(x_2{+}x_1){+}y_1$. Then, easily,

$$[2]P = (x_1^4{+}1, x_1^4{+}y_1^4)$$

if $a \in \mathbb{F}_2$, so that doubling requires just four squarings (the Frobenius) and some additions, and the lines for Miller's algorithm are a by-product. Obviously also

$$[4]P = (x_1^{16}, y_1^{16}{+}1) \ .$$

For char(K)=3 take the curve $E : y^2 = x^3{+}a_4x{+}a_6$ and point $P = (x_1, y_1)$ on E. The tangent at P has slope $\lambda = 1/y_1$ and so is $\ell : x = y_1y{-}y_1^2{+}x_1$. This intersects the curve again at $(\lambda^2{+}x_1, \lambda^3{+}y_1)$, giving $[2]P = (\lambda^2{+}x_1, -\lambda^3{-}y_1)$. The line between P and $[2]P$ has slope $\lambda' = y_1^3{-}\lambda$, from which one can deduce

$$[3]P = (x_1^9{+}a_6(1{-}a_4), -y_1^9) \ .$$

Again, there are no divisions and the tripling is given by several applications of the Frobenius, which is very efficient. So pairing calculations for fields of characteristic 3 should be performed with 3-ary exponentiation.

9.3 Distortion Maps and Vertical Lines

Super-singular curves have "distortion" maps. Suppose $E(K)$ has no points of order r^2, ϕ is a non-rational endomorphism of E, and $Q \in E(K_0)[r]$. If $\phi(Q) \notin E(K_0)$ then $e(Q, \phi(Q)) \neq 1$. Such a ϕ is called a *distortion map*, and some were listed in §9.1. They only exist for super-singular curves, but calculating the Tate pairing at $\phi(Q)$ involves only numbers in K_0 and its conjugate $\phi(K_0)$. In many circumstances and the right presentation of the curve equation, these numbers can be kept sufficiently separate for there to computational savings, particularly as a result of the final exponentiation killing off elements of K_0 in the same way as noted above. So, for the second parameter in our pairing, using $\phi(Q)$ with $Q \in E(K_0)[r]$ is cheaper than a more arbitrary choice from $E(K)[r] \setminus E(K_0)[r]$. In particular, for the vertical lines v,

- The contributions $v(\phi(Q)), Q{\in}E(K_0)[r]$, can be ignored providing the curve parameters are chosen suitably (see [2],Table 2).

This happens mostly because the final exponentiation annihilates $v(\phi(Q))$ in the same way that it does elements from K_0. This observation saves half the work before the final exponentiation.

Proof for char 3 (i.e. the case $k = 6$ with parameters as given above in §9.1). For the equation $y^2{=}x^3{-}x{+}a$, the distortion map is $\phi(x, y){=}(\alpha{-}x, iy)$ where $\alpha, i{\in}K$ satisfy $\alpha^3{-}\alpha{-}a{=}0, i^2{=}{-}1$. By easy induction, $\alpha^{3^t}{=}\alpha{+}a{\times}(t \bmod 3)$ for $t \geq 0$. The lines v have the form $x{=}c$ for some $c{\in}K_0$, so that $v(\phi(Q)){=}\alpha{-}x_Q{-}c$. As $x_Q, c \in K_0$, we have $x_Q^{q^t} = x_Q$ and $c^{q^t} = c$ for $t \geq 0$. So

$$v(\phi(Q))^{q^3} \ = \ \alpha^{q^3}{-}x_Q^{q^3}{-}c^{q^3} \ = \ \alpha{-}x_Q{-}c \ = \ v(\phi(Q)) \ .$$

Thus $v(\phi(Q))^{q^3-1} = 1$. However, the exponent in the final exponentiation is $(q^6{-}1)/r$ where r divides $q{\pm}\sqrt{3q}{+}1$, which is a factor of $q^3{+}1$, not $q^3{-}1$. So the exponent is a multiple of $q^3 - 1$. Hence the contribution of $v(\phi(Q))$ to $e(P, \phi(q))$ is 1, and it can be ignored. □

9.4 MNT Elliptic Curves

One of the most widely used and oldest of the non-singular pairing-friendly curves are the MNT curves. "MNT" refers to the authors Miyaji, Nakabayashi and Takano of these curves [21]. Our main interest is implementing pairings efficiently on some device, assuming the curve is given, and that has been covered in previous sections. This section, which may be skipped, outlines the work involved if one's own MNT curve has to be generated.

MNT curves are ordinary curves (i.e. not super-singular) with "complex multiplication" and require some effort to generate. In particular, the construction requires the Weber class polynomial [1,16] which typically has degree up to 10^3 and coefficients in $\mathbb{C}$ of up to 10^3 bits for the cases here, for which the relevant "discriminant" has a value up to 2^{32} (a single word) in absolute value. To keep the time complexity reasonable, the use of Karatsuba-Ofman or similar is essential for multiplying both complex numbers and polynomials in these calculations. Space complexity is also at the limit of reasonable computing because of the size of the numbers and polynomials. There are recent improvements by Enge and others which construct these polynomials modulo a variety of primes and then apply the Chinese Remainder Theorem to deduce their coefficients modulo the characteristic of the chosen K_0, see e.g. [10,11]. The details of both the classical and the modern construction methods are beyond the scope of this course – generally one should use dedicated computation packages, such as Mike Scott's Miracl software [26], to find the desired curve. Then the detail of the calculations can be avoided. Those interested in the basics can consult section §A.12 of the number-theoretic appendix to the IEEE's P-1363 standard [16,17][9].

The construction begins with a factorisation of the polynomial $x^k-1, (k = 3, 4, 6)$ to obtain necessary values for the trace $t = q+1-|E(\mathbb{F}_q)|$. These are parameterised by an integer l:

k	q	t
3	$12l^2-1$	$-1 \pm 6l$
4	l^2+l+1	$-l$ or $l+1$
6	$4l^2+1$	$1 \pm 2l$

For example, the curve order for $k = 6$ is $q \mp \sqrt{q-1}$, which is a factor of q^2-q+1, which in turn divides q^3+1 and q^6-1, establishing that $k = 6$. As in the case of the super-singular curves, l and k are chosen to give fields K_0, K of the desired size, with the parameter l being varied until q is prime and the curve order $q+1-t$ has a sufficiently large prime factor r.

However, there is an added complication in selecting l, namely that the discriminant $\Delta = t^2-4q$ must consist of a large square times a small square-free factor (typically under 2^{32}) or else the numbers become too large[10] and the likelihood of finding solutions of the required size becomes too small. As an example, consider the case when $k = 6$. Then $\Delta = t^2 - 4q = -12l^2 \pm 4l - 3$. Writing $\Delta = y^2\delta$, one has to solve $y^2\delta = -12l^2 \pm 4l - 3$, i.e. $-3y^2\delta = (6l \mp 1)^2 + 8$. So we require solutions (x, y) to the Diophantine equation $x^2 + 3\delta y^2 = -8$. These are given in a similar way to those for Pell's Equation $x^2 + 3\delta y^2 = 1$ which come

[9] The class polynomial section has a hard-to-find sign error at least in the early editions of this standard.

[10] Specifically, the degree of the Weber polynomial and its coefficients become too large.

from the continued fraction approximation to $\sqrt{-3\delta}$. Of course, this means computing a very accurate value for $\sqrt{-3\delta}$. One solution to $x^2 + 3\delta y^2 = -8$ can be combined with the fundamental solution of $x^2 + 3\delta y^2 = 1$ to obtain a series of increasingly large solutions, among which might be one that yields a value for l in the required range. Typically, many values of Δ must be tried before a prime l of the right size is found.

The next problem is to construct the Weber class polynomial $w_\Delta(t)$. (The Hilbert class polynomial is too large for the applications here.) Its roots are known complex numbers related to $\sqrt{-\Delta}$. Details are given in [1,17] and involve the sine and cosine functions. The roots need to be calculated sufficiently accurately and then the linear factors multiplied together to obtain the polynomial with sufficient accuracy for its coefficients to be known with an accuracy better than $\frac{1}{2}$. Since the coefficients are (huge) integers, the polynomial is then known exactly. This is now reduced modulo the field characteristic p. A linear or cubic factor $g(t)$ of the polynomial must now be extracted – there are algorithms for this. Then the coefficients $a, b \in \mathbb{F}_p$ of the curve equation $y^2 = x^3+ax+b$ are derived from $g(t)$ with some straight-forward manipulation.

Finally, the curve has only a 50-50 chance of having the right order. The ambiguity in the curve order is reflected in the $\pm$ sign. Only one of the two signs gives a size divisible by r. The next stage is to check which curve has been obtained. Choose a random (non-zero) point on the curve. Multiply it by the intended curve size over r to make sure it is not killed by a small factor of the group order. Then multiply it by the group order. If the result is $\mathcal{O}$, then the right curve has been found. Otherwise, return to the factor $g(t)$ to make a different choice of coefficients a, b.

10 Elliptic Curves Coordinates

This section just contains a few remarks on the representation of the curve. The choice of coordinate system can make a substantial difference to the efficiency of any calculations on it. The total time for the elliptic curve operations in a pairing calculation is proportional to the time taken for a single elliptic curve operation. However, the difference in speeds between the fastest and the slowest curve additions is within a factor of about 2. So at best the pairing will be done at twice the speed with another choice. A useful list of the various main addition and doubling formulae is given by Bernstein & Lange [4]. They are mostly just variations on a theme, with the Weierstraß affine coordinates as a standard example – see Prof. Koç's lecture.

The usual equation over $\mathbb{F}_p$, p odd, is the short Weierstraß form[11]

$$y^2 \ = \ x^3 + ax + b$$

for which the addition and doubling formulae for $(x_1, x_2)+(x_2, y_2) = (x_3, y_3)$ and $[2](x_1, y_1) = (x_2, y_2)$ are given by

$$x_3 = \lambda^2 - x_1 - x_2 \qquad \text{where } \lambda = (y_2 - y_1)/(x_2 - x_1)$$

[11] $E : y^2 + xy = x^3 + a_2x^2 + a_6$ for $\mathbb{F}_{2^n}$.

$$y_3 = \lambda(2x_1+x_2) - \lambda^3 - y_1$$

and

$$x_2 = (3x_1^2+a)^2/(2y_1)^2 - x_1 - x_1$$
$$y_2 = (2x_1+x_1)(3x_1^2+a)/(2y_1) - (3x_1^2+a)^3/(2y_1)^3 - y_1$$

Note that the addition formula cannot work for doubling since it gives the value 0/0. Note also the cost is made up of general field multiplications, field additions and subtractions, multiplications by small constants, and an inversion. This is the cost that is normally used to compare most alternatives, but be aware that communication costs between processor and memory may also be significant, as may be the costs of moving data between registers.

When there are points of order three, an elliptic curve has points (x, y) satisfying the following Hessian form equation:

$$x^3 + y^3 + 1 \;=\; 3dxy$$

The addition and doubling formulae are, respectively,

$$x_3 = (y_1^2x_2 - y_2^2x_1)/(x_2y_2 - x_1y_1)$$
$$y_3 = (x_1^2y_2 - x_2^2y_1)/(x_2y_2 - x_1y_1)$$

and

$$x_2 = y_1(1 - x_1^3)/(x_1^3 - y_1^3)$$
$$y_2 = x_1(y_1^3 - 1)/(x_1^3 - y_1^3)$$

The (additive) inverse is given by $-(x_1, y_1) = (y_1, x_1)$. There is a strong symmetry between the x and y coordinates here, as can be seen immediately from the equation. Again, just one inversion appears, being the result of computing the gradient of the chord or tangent on the curve. Counting the other field operations shows this to have about two thirds of the cost of the Weierstraß form. Most other representations fall between these two in terms of efficiency.

As noted in §7.4, Galbraith has compared affine and projective coordinates and found the latter not to provide a noticeable advantage despite claims in the literature. Although the requirement for speed is perhaps most acute in embedded systems, side-channel leakage may be a problem there, primarily enabling adds and doubles to be distinguished. Of course, the curve parameters, including the scalar r and final exponent r', are public, but P is the private key in decryption for the Boneh-Franklin scheme. So its value must not be leaked during the execution of Miller's algorithm. There are many different solutions to such leakage. We could, for example, note that the projective representation of the Hessian form uses the same formula[12] for both addition and doubling, and so choose that. These issues are covered elsewhere in this course, often by ensuring

[12] The formula is $[2](X_1, Y_1, Z_1) = (Z_1, X_1, Y_1) + (Y_1, Z_1, X_1)$.

that adding and doubling formulae execute the same sequence of field operations. This means, in particular, that the above formulae do not have quite enough detail for us. Specifically, the order of executing the various field operations and read and write operations needs to be given also... This is another story.

11 Finite Field Construction

Constructing finite fields is an important topic. This has been covered by Prof. Koç in his presentations. One of the problems is that curves chosen for the efficiency of their arithmetic, such as those in the standards produced by NIST and Certicom [12], are not generally suitable for pairing-based cryptography as the embedding degree k is far too large. Therefore one has to generate one's own field. There are two main cases: very small or very large characteristic.

For small characteristics, there is a choice between polynomial and normal bases. The frequent use of the Frobenius in the final exponentiation and, indeed, its appearance in point multiplication by the characteristic should be borne in mind when deciding between the two. This suggests that a normal basis would be better. There are tables of irreducible polynomials and techniques to obtain normal bases for such cases [27]. The polynomials are generally sparse (typically no worse than pentanomials), so that a polynomial basis is also not so expensive to implement.

For larger characteristics p, the standard classical technique is to choose p to have low Hamming weight (in a suitable representation such as binary of NAF) so that field multiplications are cheap. This is the exact analogy of choosing a sparse polynomial in the low characteristic case. Unfortunately, in pairing-based cryptography, we do not generally have the luxury of a choice of p as it is essentially random. The extension degree of K over $\mathbb{F}_p$ is generally small – usually a divisor of 24. Hence the generating polynomial for $K/\mathbb{F}_p$ has small degree and it is feasible to search for a suitable irreducible polynomial with very small coefficients (one computer word), hopefully with many being zero.

12 Appendix: Other Essential Algorithms

There are a number of other basic but necessary algorithms which there has been no time to cover. They are generally mostly straightforward. Many are found in the very readable IEEE P1363 [17][13]. Here's a list of them:

– *The Rabin-Miller Primality Test.* This is needed in checking for prime characteristic when generating MNT curves, and for finding large prime factors in the order of any curve.

[13] P1363.3 begins with a brief overview of the main definitions, terminology and algorithms. It is a useful proof-free summary which is easily read by those with a mathematical background. It is comprehensive, starting with modular arithmetic and finite fields.

– *Square Root Algorithm.* This is used when generating random or other points on a curve when only the x-coordinate is present, as in point compression. Using the curve equation, the y-coordinate is the square root of a polynomial in x. (A point can be represented by its x-coordinate and a bit to determine which of the two values of the y-coordinate to select.)
– *The SEA Point Counting Algorithm.* Named after the originator Schoof and subsequent developers Elkies and Atkin, this is for finding the order of an elliptic curve. It is a complex and time-consuming algorithm. By picking curves with complex multiplication, it can be avoided. Usually one only has to check the order against a claim, and this can be done by verifying the order of a random point, as observed in §9.4.

Generally speaking, there is no need to implement algorithms such as SEA or Rabin-Miller. These are available in many software packages, such as Scott's Miracl [26].

References

1. A.O.L. Atkin & F. Morain, *Elliptic Curves and Primality Proving*, Math. Comput. **61**(203), July 1993, pp. 29–68.
2. P.S.L.M. Barreto, H.Y. Kim, B. Lynn & M. Scott, *Efficient Algorithms for Pairing-Based Cryptosystems*, Crypto 2002, LNCS **2442**, pp. 354–368.
3. P. Barreto, *Pairing Based Crypto Lounge*, at http://www.larc.usp.br/∼pbarreto/pblounge.html
4. D. Bernstein & T. Lange, *Explicit-Formulas Database*, at http://hyperelliptic.org/EFD/index.html
5. J.-L. Beuchat, J.E.G. Daz, S. Mitsunari, E. Okamoto, F. Rodrguez-Henrquez & T. Teruya, *High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves*, ePrint 2010/354, IACR.
6. I.F. Blake, G. Seroussi & N.P. Smart, *Advances in Elliptic Curve Cryptography: Further Topics*, LMS Lecture Note Series **317**, Cambridge Univ. Press, 2005.
7. D. Boneh & M. Franklin, *Identity-Based Encryption from the Weil Pairing*, SIAM J. Computing **32**(3), pp. 586–615, 2003. (See also Crypto 2001, LNCS **2139**, pp. 213–229.)
8. K. Eisenträger, K. Lauter & P.L. Montgomery, *Fast Elliptic Curve Arithmetic and Improved Weil Pairing Evaluation*, CT-RSA 2003, LNCS **2612**, Springer-Verlag 2003, pp. 343–354.
9. K. Eisenträger, K. Lauter & P.L. Montgomery, *Improved Weil and Tate Pairings for Elliptic and Hyperelliptic Curves*, ANTS VI, LNCS **3076**, Springer-Verlag 2004, pp. 169–183.
10. A. Enge, *The Complexity of Class Polynomial Computation via Floating Point Approximations*, Math. Comp. **78**(2009), pp. 1089–1107.
11. A. Enge & A.V. Sutherland, *Class Invariants by the CRT Method*, ANTS IX, LNCS **6197**, Springer-Verlag 2010, pp. 142–156.
12. *Digital Signature Standard (DSS)*, FIPS Pub 186-3, National Institute of Standards and Technology (NIST), June 2009.
13. D. Freeman, M. Scott & E. Teske, *A Taxonomy of Pairing-Friendly Elliptic Curves*, IACR ePrint 2006/372.

14. S. Galbraith, *Elliptic Curve Cryptography*, http://www.isg.rhul.ac.uk/∼sdg/ecc.html (for a good overview of the best books in the area.)
15. S. Galbraith, K. Harrison & D. Soldera, *Implementing the Tate Pairing*, HPL-2002-23, HP Labs, 2002. *Also* ANTS 2002, LNCS **2369**, Springer-Verlag, 2002, pp. 324–337.
16. IEEE P1363-2000 *IEEE Standard Specifications for Public-Key Cryptography*, Annex A: Number-Theoretic Background, §A.12.1, IEEE 2000. http://grouper.ieee.org/groups/1363/P1363/index.html
17. IEEE P1363.3 *Standard for Identity-Based Cryptographic Techniques Using Pairings*, http://grouper.ieee.org/groups/1363/IBC/
18. A. Joux, *A One Round Protocol for Tripartite Diffie-Hellman*, Algorithmic Number Theory, LNCS **1838**, Springer-Verlag 2000, pp. 385-394. *and* J. Cryptology **17**(4), Sept 2004, pp. 263–276.
19. M. Joye, *The Montgomery Powering Ladder*, CHES 2002, LNCS **2523**, Springer-Verlag 2003, pp. 291-302.
20. A. Menezes, T. Okamoto & S. Vanstone, *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field*, IEEE Trans. Inform. Thy. **39**, 1993, pp. 1639–1646.
21. A. Miyaji, M. Nakabayashi & S. Takano, *New explicit conditions of elliptic curve traces for FR-reductions*, IEICE Trans on Fundamentals, **E84-A**(5), pp. 1234–1243, 2001.
22. P.L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Mathematics of Computation, **48**(177) Jan 1987, pp. 243-264.
23. K. Ohgishi, R. Sakai & M. Kasahara, *Notes on ID-based Key Sharing Systems over Elliptic Curve*, IEICE Technical Report, ISEC 99(414) Nov. 1999, pp. 37–42. (Available at http://ci.nii.ac.jp/naid/110003298699/.)
24. R. Sakai, K. Ohgishi & M. Kasahara, *Cryptosystems based on Pairing*, SCIS '2000, Okinawa, Japan, Jan. 2000, pp. 26–28.
25. M. Scott, *The Tate Pairing*, http://www.computing.dcu.ie/∼mike/tate.html
26. M. Scott, *Multiprecision Integer and Rational Arithmetic C/C++ Library (Miracl)*, Shamus Software, Ireland.
27. G. Seroussi, *Table of Low-Weight Binary Irreducible Polynomials*, HPL-98-135, HP Labs, 1998.
28. A. Shamir, *Identity-based Cryptosystems and Signature Schemes*, Crypto '84, LNCS **196**, pp. 47–53.
29. J.H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics **106**, Springer-Verlag, 2nd ed., 2009.