Cryptographic Pairings on Elliptic Curves

Colin D. Walter

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

Outline

- Motivation – some examples
- Bilinear Pairings
- Tate & Weil Pairings
- Miller's Algorithm
- Other Scalar Mult^n Algorithms
- Efficiency: Ground vs. Extension Field
- Divisions and Denominators
- Frobenius & Final Exponentiation
- Pairing Friendly Curves: super-singular & MNT
- Curve & Field Representations

Preliminary Remarks

- Only about half these slides will be covered. The remainder are for future study.
- The first half of the slides are introductory to give you a feel for the topic.
- The second half sketches some of the main implementation issues.
- There are additional notes which give much more detail. These are primarily for when you really have to learn the subject in order to implement pairings.
- There is a lot of deep mathematics behind pairings, but it can all (or mostly) be avoided.
- For those who have not seen pairings before, the key things to master are:
    1. The notation
    2. Miller's Algorithm
    3. The Final Exponentiation

Motivation – Some Applications

- Pairings provide new protocols and services not available in classical cryptography.

We would like identity-based encryption where all users share common public parameters – this would reduce key distribution problems.

Classical RSA doesn't work. The keys d_i, e_i satisfy

$$d_i e_i = 1 \mod N.$$

Then j can deduce i's private key from e_i since he knows N and inversion mod N is "easy".

- Are there systems where j cannot deduce i's private key from his own keys?

Ohgishi, Sakai & Kasahara (1999) and
Boneh & Franklin (2001)
solved this with ***pairings*** on elliptic curves.

The public identity-based encryption key of an identity (person) may contain, for example:

- the identity's email address
- the current date
- a subject or security level
- a department/group name.

An example would be a national health service.

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' 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 $A,B \in G_1$, $C,D \in G_2$.

- G_1 and G_2 will be subgroups or quotient groups of an elliptic curve (written additively),
- G' is a group of roots of unity in a field (the "embedding" field), with $|G_1| = |G_2| = |G'|$.
- Note that $e([r]A, [s]C) = e(A,C)^{rs}$.

Our pairings require two additional properties:

i) *non-degeneracy*: $e(A,C) \neq 1$ for some $A \in G_1$, $C \in G_2$

ii) *computability*: there is an efficient algorithm to determine e(A,C) from A and C.

Example $e(\underline{u},\underline{v}) = \underline{u}M\underline{v}^T$ for vectors $\underline{u},\underline{v}$ and matrix M.

The DLP – Discrete Log Problem

Notation:

- Elliptic curve group G_1 defined over $\mathbf{K_0}$
- Pairing $\boldsymbol{e}$ into the "embedding" field $\mathbf{K}$
  (the field containing the necessary roots of unity, i.e. of 1.)

Suppose $Q = \alpha P$ for $P \in G_1$.

- The discrete log problem (DLP) for G_1 is to find α.

  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 DLP in K yields α, so solves DLP in G_1.

- We want Q and P public and α private.
  So we need to the DLP to be difficult in K.

The MOV threshold from this attack means an embedding degree of at least 6 to 9 when the order q of K_0 has 192 bits,

i.e. K has an order q^k of at least 1536 bits, say.

- Computations over K are expensive, so the embedding degree $k = [K:K_0]$ must be kept down.

Joux's Three Party Key Agreement

In 2000, Joux presented a scheme for one-round, 3-party key agreement based on bilinear maps, the first constructive use of these maps in crypto.

(The usual Diffie-Helman scheme is between two parties and so takes more rounds for more parties.)

In the Joux scheme,

- P is a public generator of $G = G_1 = G_2$.
- The three parties A, B, C choose random integer secrets a, b, c.

The protocol is as follows:

- A publishes aP
  B publishes bP
  C publishes cP
- 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 possess the shared secret key

$$s = e(P,P)^{abc} \in G'.$$

An attacker cannot deduce a from P and aP as the DLP is too hard.

IBE – Identity-Based Encryption

There are four parts to this:

- SETUP chooses the public parameters such as the curve and a private master key.
- EXTRACT generates private decrypt keys from the public and private parameters determined by SETUP and a public encrypt key.
- ENCRYPT is a description of encryption.
- DECRYPT is a description of decryption.

Here is *BasicIdent* in the Boneh-Franklin scheme. The full scheme is a “minor” extension.

Given:

- A curve with subgroup G_1 of prime order r,
- A pairing e into G', also of order r.
- H a hash function from n-bit texts to G_1
- H' a hash function from G' to n-bit texts.

SETUP

- choose a master secret s,
- choose a random generator P of G_2,
- publish P and $P_{pub} = sP$.

EXTRACT for public identity **ID**

- compute the hash $Q_{ID} = H(ID) \in G_1$
- generate ID's private key $d_{ID} = sH(ID) = sQ_{ID}$

ENCRYPT message M for identity **ID**

- compute $Q_{ID} = H(ID) \in G_1$
- compute $g_{ID} = e(Q_{ID}, P_{pub}) \in G'$
- choose random $\rho \in \mathbb{Z}$
- output ciphertext pair ρP, $M \oplus H'(g_{ID}^{\rho})$.

DECRYPT (U, V):

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

The arithmetic requirements are:

- a random number generator
- four hashing functions (for the full scheme)
- determining suitable fields & curves
- computing the pairing
- exponentiation in G'

Next, we must fill in the missing arithmetic detail.

Notation *(for reference)*

- p the field characteristic. Usually a small prime (2 or 3), or a large prime of 160+ bits.
- q the order of the field K_0 containing the elliptic curve. Typically 160 or 192 bits.
- $K_0 = \mathbb{GF}(q)$ the field in which the curve lies.
- $E(K_0)$ the chosen elliptic curve over K_0.
- r the prime order of a large subgroup of $E(K_0)$. Ideally q/r is small, typically $q/r < 24$.
- $G_1 = E(K_0)[r]$ the group of points whose r^{th} multiple is the point $\mathcal{O}$ at infinity. This is cyclic (because r is so large).
- k the *embedding degree*, the smallest number such that r divides $q^k–1$. k divides $r–1$, and typically divides 12.
- $K = \mathbb{GF}(q^k)$ the *embedding field*, the smallest extension of K_0 containing the rth roots of 1.
- $G_2 = E(K) / rE(K)$, the quotient group of points defined over K modulo the r^{th} multiples.
- $G' = \mu_r$, the group of r^{th} roots of 1 in $\mathbb{GF}(q^k)$
- e: $G_1 \times G_2 \rightarrow G'$ the selected bilinear pairing.

Divisors

Divisors summarise a function's main properties using a formal representation of its zeros & poles.

Definition. A *divisors* is a formal sum over $\mathbb{Z}$ of points on the curve E(K), e.g.

$$D = \sum_{P \in \mathrm{E(K)}} n_P(P) \quad \text{where } n_P \in \mathbb{Z} \text{ for all } P \in \mathrm{E(K)}.$$

Terminology:

- The ***degree*** of D is $\deg(D) = \sum_P n_P$.
- Its ***support*** is $\sup(D) = \{P \mid n_P \neq 0\}$.
- **(*P*)** is the divisor of degree 1 and support $P \in \mathrm{E(K)}$.
- $\mathbf{ord_P(}f\mathbf{)}$ is the multiplicity of P as a zero of f where $f : \mathrm{E(K)} \to \mathrm{K}$.
- $(f) = \sum_P \mathrm{ord}_P(f)\ (P)$ is a ***principal*** divisor.

Properties

- $(cf) = (f)$ for non-zero $c \in \mathrm{K}^*$.
- $(c) = 0$ is the empty sum for $c \in \mathrm{K}^*$.
- $(fg) = (f) + (g)$, $\quad (f/g) = (f) - (g)$.
- $\deg((f)) = 0$ and $\sum_P [\mathrm{ord}_P(f)]\ P = \mathcal{O}$.

- $\deg((f)) = 0$ and $\sum_P [\mathrm{ord}_P(f)]\, P = \mathcal{O}$.

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

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

So $\mathcal{O}$ is a pole of order 3.

Definition For $f : \mathrm{E(K)} \to \mathrm{K}$ and $D = \sum_P n_P(P)$, define

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

This extends the definition of f from points on the curve to any divisor.

For a non-zero constant $c \in \mathrm{K}$,

$$(cf)(D) = f(D) \;\text{ if }\; \deg(D) = 0$$

because the constants c all cancel.
We are only interested in divisors with degree 0.

Definition Two divisors are *equivalent* if they differ by a principal divisor, i.e.

$$D_1 \sim D_2 \quad \text{if, and only if,} \quad D_1 = D_2 + (f)$$

for some function f.

(We need this for technical reasons but will ignore related theory.)

The Tate Pairing

The groups for the pairing $e : G_1 \times G_2 \to G'$ are

- $G_1 = E(K)[r] = \{P \in E(K) \mid [r]P = \mathcal{O}\}$,
  i.e. the subgroup of curve points defined over K whose order divides the prime r.
- $G_2 = E(K) / rE(K)$ where $rE(K) = \{[r]P \mid P \in E(K)\}$
  This has exponent r.
- $G' = K^* / K^{*r} \cong \mu_r$, the r^{th} roots of unity.

Pick $P \in E(K)[r]$ and $Q \in E(K)$. Let f be such that

$$(f) = r(P) - r(\mathcal{O}) .$$

Choose divisor $D \sim (Q) - (\mathcal{O})$ with support disjoint from (f). Then the *Tate Pairing* is defined by

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

As $|K^*| = q^k - 1$, we can raise elements of K to the power $(q^k - 1)/r$ to obtain an r^{th} root of unity and avoid the ambiguity of the quotient group. Hence we define the pairing

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

Note that $\langle P,Q \rangle = 1_K \bmod K^{*r}$ if $P,Q \in E(L)$ for any subfield $L \subset K$ not containing μ_r since $f(D) \in L \cap \mu_r = \{1\}$. So we usually pick $P \in E(K_0)[r]$ and Q properly in $E(K)$.

There are a number of other related pairings, e.g.

- The Weil pairing is a ratio $\langle P,Q\rangle / \langle Q,P\rangle \in \mu_r$ with no "final exponentiation", but is too expensive.
- The *eta* (η), *ate*, *R-ate*, *optimal ate,* etc. pairings are more efficient, but specific to particular contexts.

We won't cover these, but the arithmetic techniques are similar.

In particular, the *ate* pairing involves a smaller final exponent and the function f is evaluated at one divisor, namely Q, not the two implied by D.

Chords to the Curve

Two lines are used to describe the addition law $P+Q=S$ on an elliptic curve over K, namely

- The chord ℓ_{PQ} through P and Q, and
- The vertical line v_S through S, $-S$ and $\mathcal{O}$.

The corresponding divisors are

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

and

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

Consequently,

$$(\ell_{P,Q} / v_S) = (P) + (Q) - (S) - (\mathcal{O})$$

nicely represents the addition.

Computationally, in affine coords,

$$\ell_{P,Q}(x,y) \equiv rx+sy+t$$

for some $r,s,t \in K$ such that $rx_P+sy_P+t = 0$ and $rx_Q+sy_Q+t = 0$. If $P,Q \in E(K_0)$ then $r,s,t \in K_0$.

Similarly, if the curve has form $y^2 = f(x)$ then

$$v_S(x,y) \equiv x-x_S$$

which is satisfied by S and $-S = (x_S,-y_S)$.

Miller's Algorithm

We need to construct a function f with divisor

$$(f) = r(P) - r(\mathcal{O}) .$$

This is done by constructing $f_1, f_2, \ldots, f_r$ such that

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

Lemma. Let ℓ_{ij} be the chord through $[i]P, [j]P$ and v_{i+j} the vertical line through $[i+j]P, -[i+j]P$. Then:

- $f_1 = c_1$, and
- $f_{i+j} = c_{ij} f_i f_j \ell_{ij} v_{i+j}^{-1}$

for some constants c_1, c_{ij} for $1 \le i, j \le r$.

Proof. First, $(f_1) = 1(P) - ([1]P) - (1-1)(\mathcal{O}) = 0$ is the empty formal sum. So f_1 is a constant.

Secondly, $(\ell_{ij} / v_{i+j}) = ([i]P) + ([j]P) - ([i+j]P) - (\mathcal{O})$ gives $(f_i f_j \ell_{ij} / v_{i+j})$

$= i(P) - ([i]P) - (i-1)(\mathcal{O}) + j(P) - ([j]P) - (j-1)(\mathcal{O}) + (\ell_{ij}/v_{i+j})$

$= i(P) - (i-1)(\mathcal{O}) + j(P) - (j-1)(\mathcal{O}) - ([i+j]P) - (\mathcal{O})$

$= (i+j)(P) - ([i+j]P) - (i+j-1)(\mathcal{O}) = (f_{i+j})$

which establishes the claim by induction. ∎

$\langle P,Q\rangle$ requires evaluating f at a divisor D of degree 0.

Corollary. For a divisor D of degree 0,

- $f_1(D) = 1$, and
- $f_{i+j}(D) = f_i(D)\ f_j(D)\ \ell_{ij}(D)\ v_{i+j}(D)^{-1}$.

Thus the constants cancel in the previous lemma, and so can be ignored.

Since $(f_r) = r(P)–([r]P)–(r–1)(\mathcal{O}) = r(P) – (r)(\mathcal{O}) = (f)$,

$$\langle P,Q\rangle = f_r(D)\ .$$

The construction of f_r via the formula for f_{i+j} in terms of f_i and f_j shows $f_r(D)$ can be computed by inserting some extra computation into an algorithm for determining $[r]P$.

Choose $D = (Q+S)–(S)$ for random point S.
Then $D \sim (P) – (\mathcal{O})$ as required for the Tate pairing.
In the following, T holds a multiple $[j]P$ of P, and
f contains the corresponding $f_j(D)$.

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 $\langle P,Q \rangle$ mod K^{*r}.

```
{   Choose random S∈ E(K)
    Q' ← Q+S
    T ← P
    f ← 1_K
    For i ← n–2 downto 0 do
    {   Determine lines ℓ and υ for doubling T.
        T ← [2]T
        f ← f^2 ℓ(Q') ℓ(S)^-1 υ(Q')^-1 υ(S)
        If r_i = 1 then
        {   Determine ℓ and υ for the addition T+P.
            T ← T+P
            f ← f ℓ(Q') ℓ(S)^-1 υ(Q')^-1 υ(S)
        }
    }
    Return f
}
```

- This is the binary left-to-right algorithm for $[r]P$.
- Pick S in $E(K_0)$ so $f(S)$ is cheaper to calculate.
- The support of D must not intersect any f_i which occurs.
  This will be evident by obtaining 0/0.
  If this happens, run it again with a different S.

Other Scalar Multiplication Algorithms

Other scalar multiplication algorithms for $[r]P$ can be used, e.g. NAF or m-ary, but most are too expensive. (See additional notes for details.)

Useful cases restrict allowable operations to only:

- doublings $T \leftarrow [p]T$ or
- tuplings $T \leftarrow [p]T$ for very small $p = \text{char(K)}$

and

- additions $T \leftarrow T \pm P$

The main drawback with alternative algorithms is the cost of an extra multiplication $f_d(D)$ for digits $d \neq \pm 1$ since the addition step becomes:

Determine ℓ and υ for the addition $T+P_d$.
$T \leftarrow T+P_d$
$f \leftarrow f\, f_d\, \ell(Q')\, \ell(S)^{-1} \upsilon(Q')^{-1} \upsilon(S)$

Recall $f_1(D) = 1$, so digit $d=1$ is not a problem. Otherwise there are extra multiplications in K.

The Main Efficiency Issues

- Work in K_0 when possible.

Values in K take k times more space than in K_0, and multiplications take k^2-fold more time.

- If $P \in E(K_0)$ then all the lines are over K_0.
- If $S \in E(K_0)$ then half the evaluations are in K_0.

About half the multiplication count is calculating $[r]P$, the rest determining and evaluating the lines. So, with lines and S in K_0,

- Evaluating the lines at Q' dominates the time.
- K_0 should be kept down to the size used in classical ECC.

Clearly,

- Divisions & inversions should be avoided.

So f is split into *numerator* and *denominator* values f_N and f_D. Then, e.g., the addition step becomes

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

which has the same number of multiplications.

Then there is a single final division $f = f_N/f_D$.

Separating Mult^s over K_0 and K

Field multiplications for updating f might be separated into

- those over K_0 using variables f_{0N} , f_{0D} and
- those over K using variables f_{1N} , f_{1D}, say.

By estimating each K×K mult^n as equivalent to k^2 mult^{ns} over K_0, it is clear that only the total number of K×K mult^{ns} is significant.

- The mult^n count is unchanged by the above split.
- Mult^{ns} should be done in the order which minimises the number of K×K operations.

The saving is O(k) whereas
the cost of all multiplications is O(k^2).
Hence this is significant only for small k.

Projective or Affine Coordinates?

- Use affine coordinates for Q'.

As $Q' \in \mathrm{E}(\mathrm{K})$, 3 coordinates for Q' increases the cost of each $\ell(Q')$ and $\omega(Q')$ in Miller's algorithm.

It is cheaper to convert $Q' = Q+S$ to affine coord$^{\mathrm{s}}$ once for all when the random $S \in \mathrm{E}(\mathrm{K}_0)$ is chosen.

To do this, the Extended Euclidean Algorithm gives the inverse at the cost of 10 or so multiplications – much less than the number of double & add steps.

- According to Galbraith, empirical evidence does not justify projective coord$^{\mathrm{s}}$ elsewhere.

The main extra cost from projective coords is that the vertical lines are now in the form $z_T x - x_T z = 0$.

Evaluating this at Q' and S costs another $\mathrm{K}_0 \times \mathrm{K}$ mult$^{\mathrm{n}}$ and another $\mathrm{K}_0 \times \mathrm{K}_0$ mult$^{\mathrm{n}}$, which is more expensive than the inversion in K_0 to put T in affine coordinates (unless k is exceptionally small).

Removing the Factors involving S

- Recall the "final exponentiation" to obtain $e(P,Q)$

Pick $S \in E(K_0)[r]$ in Miller's algorithm to make calculations of $\ell(S)$ and $\nu(S)$ entirely within K_0.

- Then $\ell(S)^{q-1} = 1 = \nu(S)^{q-1}$.

If $r \nmid q-1$ then $(q^k-1)/r$ is a multiple of $q-1$, so

$$\ell(S)^{(q^k-1)/r} = 1 = \nu(S)^{(q^k-1)/r}.$$

So $\ell(S)$ and $\nu(S)$ make no contribution to

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

Theorem. 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 $\nu(S)$ deleted from it.

Unfortunately, the calculations of $\ell(Q')$ and $\nu(Q')$ are much more expensive than $\ell(S)$ and $\nu(S)$, so the saving is not so significant.

Theorem. In Miller's Algorithm, if r is prime to $q-1$ and $k>1$ then all the factors of f, namely f_d, $\ell(Q')$, $\ell(S)$, $v(Q')$, $v(S)$ can be replaced by multiples $\alpha_1 f_d$, $\alpha_2 \ell(Q')$,... for any $\alpha_1, \alpha_2, \ldots \in K_0^*$.

The reasoning is the same here – each α_j is annihilated by the final exponentiation.

This is useful if denominators have appeared, e.g. through calculating a gradient and avoiding the inversion when adding two points.

The Frobenius Automorphism

Consider the final exponentiation

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

If p = char(K), the *Frobenius automorphism* is the map $\gamma \to \gamma^p$ for $\gamma \in$ K. Clearly,

$$\gamma^p = \gamma \text{ for } \gamma \in \mathbb{F}_p$$

by Lagrange's theorem on the $\times^{ve}$ group $\mathbb{F}_p^*$.

If $q = p^n$ then elements of $\mathbb{F}_q$ are stored using a representation of $\mathbb{F}_q$ as a vector space over $\mathbb{F}_p$. If $\gamma = \sum_{i=0}^{n-1} \gamma_i b_i$, $\gamma_i \in \mathbb{F}_p$, is the rep^n of $\gamma \in \mathbb{F}_q$ using basis $(b_0, b_1, \ldots, b_{n-1})$ over $\mathbb{F}_p$ then

$$\gamma^p = \left(\sum_{i=0}^{n-1} \gamma_I b_i\right)^p = \sum_{i=0}^{n-1} \gamma_i{}^p b_i{}^p = \sum_{i=0}^{n-1} \gamma_I b_i{}^p$$

because all absent terms in the product have binomial coefficients which are multiples of p.

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

The b_i^p are pre-computed if p is small.

With careful choice of the field representation, such as a *normal* basis, the expressions for the b_i^p are very simple linear combinations of the b_j.

To take advantage of a cheap Frobenius automorphism, use m-ary exponentiation with $m=p$:

ALGORITHM: p-ary Exponentiation
INPUTS: $\gamma \in K$, $r' = \sum_{i=0}^{n-1} r_i p^i$.
OUTPUT: $\Gamma = \gamma^{r'}$.

Pre-compute & store the occurring γ^{r_i}
$\Gamma \leftarrow \gamma^{r_{n-1}}$
For $i \leftarrow n-2$ downto 0 do
    $\Gamma \leftarrow \Gamma^p$;
    if $r_i \neq 0$ then $\Gamma \leftarrow \Gamma \times \gamma^{r_i}$
Return Γ

For small $p \neq 2$, this should be cheaper than exponentiation using a base 2 representation of the final exponent $r' = (q^k - 1)/r$.

For large p we want only a few non-zero digits r_i, which is almost the case for many pairing-friendly curves after this adjustment:

Theorem Suppose r divides n, n divides both q^k-1 and $|E(K_0)|$, and $P \in E(K_0)[r]$. Then

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

Example

For $q = 3^{163}$ take the super-singular curve of order

$$n = q-\sqrt{(3q)}+1 = 3^{163}-3^{82}+1$$

$n = 7r$ for a prime r. The embedding degree is k=6 since n is a factor of q^2-q+1 which divides q^3+1 and hence q^6-1.

So we can evaluate the pairing using n rather than r. The final exponentiation is then by

$$n' = (q^6-1)/n = (q^3-1)(q+1)n''$$

for $n'' = q+\sqrt{3q}+1 = 3^{163}+3^{82}+1$.

Multiplying out, there is a base 3 representation of n' with at most 2×2×3 = 12 small non-zero digits. This makes exponentiation by n' extremely fast.

Pairing-Friendly Curves

These curves have:

- a small embedding degree k,
- a large prime factor r in the curve order,
- a small co-factor $|E(K_0)|/r$.

There are a number of different families of curves. P1363 is a bit out of date, but a good reference is the “Taxonomy” paper of Freeman, *et al.*

We look at two cases:

- Super-singular curves,
- MNT curves.

Super-Singular Elliptic Curves

Definition. 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.

By Hasse's bound, $|t| \leq 2\sqrt{q}$ for *trace t*.
The only choices for super-singular curves are:

- $t = 0$ with k=2,
- $t = \pm 2\sqrt{q}$ with k=1,
- $t = \pm\sqrt{q}$ if char($\mathbb{F}$) ≡ 0,2 (mod 3) with k=3,
- $t = \pm\sqrt{(2q)}$ if char($\mathbb{F}$) = 2 with k=4, and
- $t = \pm\sqrt{(3q)}$ if char($\mathbb{F}$) = 3 with k=6.

If $r > 4\sqrt{q}$ is prime divisor of $|E(K_0)|$, then r^2 exactly divides $|E(K)|$, $E(K)[r]$ is the direct product of two subgroups of order r and $E(K)[r] \cap rE(K) = \{\mathcal{O}\}$.

So $E(K)/rE(K) \cong E(K)[r]/rE(K)$ and we can effectively take both arguments from $E(K)[r]$.

There is also a nice "distortion" map which maps $E(K_0)[r]$ to another subgroup of $E(K)$ with order r. This is defined over a conjugate of K_0 and the two generate the full group $E(K)[r]$.

The advantage is that half the work of a pairing calculation is avoided: all the denominators can be ignored as above (page 24) because they are annihilated when raising to the power $q-1$.

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

1. $E : y^2 = x^3+a$ over $\mathbb{F}_p$, where $p \equiv 2 \pmod 3$. Here $|E(\mathbb{F}_p)| = p+1$ and $k=2$.
2. $E : y^2 = x^3+x$ over $\mathbb{F}_p$, where $p \equiv 3 \pmod 4$. Here $|E(\mathbb{F}_p)| = p+1$ and $k=2$.
3. $E : y^2 = x^3+a$ 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 $|E(\mathbb{F}_{p^2})| = p^2-p+1$ and $k=3$.
4. $E : y^2+y = x^3+x+a$ where $\mathbb{F}_q$, $q=2^\ell$ for odd ℓ and $a \in \mathbb{F}_2$. Here $|E(\mathbb{F}_q)| = q \pm \sqrt{(2q)}+1$ and $k=4$.
5. $E : y^2 = x^3-x+a$ where $\mathbb{F}_q$, $q=3^\ell$ for odd ℓ and $a \in \mathbb{F}_3^*$. Here $|E(\mathbb{F}_q)| = q \pm \sqrt{(3q)}+1$ and $k=6$.

Suitable fields and curves with these parameters are found as follows:

1. First, systematically choose random p of the required magnitude: applying the Rabin-Miller primality testing algorithm to check primality.
2. Run Rabin-Miller again on r where r is obtained from $n = |E(K_0)|$ using the above formula, and removing all small factors.

Typically a sieve of Eratosthenes is used for p, tagging all elements in the table having a prime divisor less than 2^{16}, say.

- The probability of both p and n being prime is $O(1/\log(np))$, and so not difficult.

For $\mathrm{char}(K_0) = 2, 3$ the choice of fields is very limited: $|E(\mathbb{F}_q)|$ rarely has a large prime factor for suitable q. $q = 3^{163}$ is one example.

There are simple equations provided with these constructions which define how to build K/K_0. Also there are tables of irreducible polynomials for generating K_0 , e.g. $\mathbb{F}_{2^{251}} = \mathbb{F}_2[t]/(t^{251} + t^7 + t^4 + t + 1)$.

- Fast multiplication methods, such as Toom or Karatsuba-Ofman, are advisable for K.

Characteristic 2 – a worked example.

Take super-singular $E : y^2+y = x^3+x+a$ over $\mathbb{F}_2$, with $a \in \mathbb{F}_2$ so that $k = 4$.

The tangent at $P = (x_1,y_1)$ 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,

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

So doubling requires just four squarings
(the Frobenius) and some additions.
The lines for Miller's algorithm are a by-product.

Characteristic 3 – a worked example.

Take super-singular $E : y^2=x^3+a_4x+a_6$ over $\mathbb{F}_3$. The tangent at $P = (x_1,y_1)$ is

$$\ell : x = y_1 y - y_1^{\,2} + x_1$$

where $\lambda = 1/y_1$. 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

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

Again, no divisions and the tripling is given by four applications of the Frobenius, which is very efficient.

So pairing calculations for fields of characteristic 3 should be performed with 3-ary exponentiation for both scalar point multiplication and final exponent^n.

Distortion Maps and Vertical Lines

Super-singular curves have “distortion” maps:

Definition 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.

They only exist for super-singular curves, but calculating the Tate pairing at $\phi(Q)$ now 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.

In particular,

- The contributions $\boldsymbol{\omega}(\phi(Q))$, $Q \in E(K_0)[r]$, can be ignored, i.e. removed, providing the curve parameters are chosen suitably.
- This observation saves half the work before the final exponentiation.

(See the notes for a proof.)

MNT Elliptic Curves

(Miyaji, Nakabayashi & Takano)

One of the most widely used and oldest of the non-singular pairing-friendly curves are the *MNT* curves.

They are ordinary curves (i.e. not super-singular) with “complex multiplication” and require some effort to generate. One should use dedicated computation packages, such as Mike Scott's Miracl software, for this. More construction detail is in the notes and the IEEE P1363 standard.

There are recent improvements by Enge and others using only local methods & the CRT.

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

k	q	t
3	$12\ell^2-1$	$-1\pm6\ell$
4	$\ell^2+\ell+1$	$-\ell$ or $\ell+1$
6	$4\ell^2+1$	$1\pm2\ell$

E.g. the curve order for k=6 is $q\pm\sqrt{q}-1$,
which is a factor of q^2-q+1, which divides q^3+1
and so also q^6-1, establishing that k=6.

ℓ must be chosen to give fields of the required size, and varied until q is prime and the curve order $q+1-t$ has a sufficiently large prime factor r.

The *discriminant* $\Delta = t^2 - 4q$ must also consist of a large square times a small square-free factor (typically under 2^{32}) or else the numbers become too large and the likelihood of finding solutions of the required size becomes too small.

Elliptic Curves Coordinates

The choice of coordinate system affects 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.

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. They are mostly just variations on a theme, with the Weierstraß affine coordinates as a standard example – see Prof. Koç's lecture.

e(P,Q) will usually have P in E(K_0) and Q in E(K). So the computation of rP will be over K_0, making it relatively cheap, as are the values $\ell(S)$ and $\nu(S)$. The expensive part is $\ell(Q)$, $\nu(Q)$ and the final exponentiation. So Q should be put in affine coordinates.

Affine Coordinates.

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

$$E : y^2 = x^3 + ax + b$$

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

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

$$y_3 = (2x_1+x_2)(y_2-y_1)/(x_2-x_1)-(y_2-y_1)^3/(x_2-x_1)^3-y_1$$

and

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

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

The addition formula cannot work for doubling since it gives the value 0/0.

The cost is usually given as a count of general field mult^{ns}, field additions and subtractions, mult^{ns} by small constants, and an inversion.

However, be aware that communication costs between processor and memory may also be significant, as may be the costs of moving data between registers.

Hessian Form

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

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

The addition and doubling formulae are, resp[y],

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

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 inverse is $-(x_1,y_1) = (y_1,x_1)$.

Again, just one inversion appears, a result of computing the chord gradient. This has about two thirds the multiplication count of the Weierstraß form. Most other representations fall between these two in terms of efficiency.

Steven Galbraith has compared affine and projective coordinates and found the latter not to provide a noticeable advantage.

Although the requirement for speed is perhaps most acute in embedded systems, *side-channel leakage* may be a problem there, mainly from distinguishing adds from doubles.

There are many different solutions to such leakage. For example, the projective representation of the Hessian form uses the same formula for both addition and doubling, viz.

$$[2](X_1,Y_1,Z_1) = (Z_1,X_1,Y_1) + (Y_1,Z_1,X_1)$$

and one can choose that.

We don't need to hide *r* which is public, but in the Boneh-Franklin IBE scheme, *P* is the decryption key, and it must not be leaked.

These issues are covered elsewhere in this course, and solved by adding more detail to the above formulae, such as specifying the order of execution.

Finite Field Construction

Efficient field arithmetic has been covered by Prof. Koç. Standard curves are chosen using these principles. However, they are not generally suitable for pairing-based cryptography as the embedding degree is too large.

So one has to generate one's own field. There are two main cases: small or large characteristic.

For small characteristics, the frequent use of the Frobenius in the final exponentiation suggests that a normal basis would be better than a polynomial basis.

There are tables of irreducible polynomials (trinomials and pentanomials for p=2) and techniques to obtain normal bases for such them. Being sparse, a polynomial basis is also possible.

For large p, the classical technique is to choose p with low Hamming weight so that multiplications are cheap. Unfortunately, here p is essentially random and so it is difficult to obtain this property.

Consequently, pairings tend to be computed faster when the characteristic is very small.

For large p the extension degree of K over $\mathbb{F}_p$ is generally small – typically 6. Hence the generating polynomial of K/$\mathbb{F}_p$ has small degree. It is feasible to search for a nice irreducible polynomial with very small coefficients, with many being zero.

Appendix: Other Essential Algorithms

Other basic, necessary algorithms include:

- *The Rabin-Miller Primality Test*.

Needed in curve generation to check char(K) and finding large prime subgroups of $E(K_0)$.

- *Square Root Algorithm*.

Used to find the *y*-coord of a point from its *x*-coord.

- *The SEA Point Counting Algorithm*.

Finds the order of an elliptic curve. Avoid this by picking curves with known order.

Some attacks modify public parameters. However, the curve order can be checked easily by verifying the order of a random point.

Most algorithms are detailed in the number theory appendix A to IEEE P1363.

- Generally, there is no need to implement algorithms such as SEA or Rabin-Miller.
- They are available in many software packages, such as Mike Scott's Miracl.