Compressed Pairings

Michael Scott[1]* and Paulo S. L. M. Barreto[2]**

[1] School of Computing
Dublin City University
Ballymun, Dublin 9, Ireland.
mike@computing.dcu.ie
[2] Universidade de São Paulo, Escola Politécnica.
Av. Prof. Luciano Gualberto, tr. 3, 158.
BR 05508-900, São Paulo(SP), Brazil.
pbarreto@larc.usp.br

Abstract. Pairing-based cryptosystems rely on bilinear non-degenerate maps called pairings, such as the Tate and Weil pairings defined over certain elliptic curve groups. In this paper we show how to compress pairing values, how to couple this technique with that of point compression, and how to benefit from the compressed representation to speed up exponentiations involving pairing values, as required in many pairing based protocols.
Keywords: pairing-based cryptosystem, efficient implementation.

1 Introduction

With the discovery of a viable identity-based encryption scheme based on the Weil pairing [4], pairing-based cryptography has become of great interest to cryptographers. Since then, pairing-based protocols – many with novel properties – have been proposed for key exchange [25], digital signature [5], encryption [4], and signcryption [23]. Although the Weil pairing was initially proposed as a suitable construct for the realisation of such protocols, it is now accepted that the Tate pairing is preferable for its greater efficiency. Supersingular elliptic curves were originally proposed as a suitable setting for pairing-based schemes; recent work has shown that certain ordinary curves are equally suitable, and offer greater flexibility in the choice of security parameters [3, 21]. Fast computer algorithms for the computation of the Tate pairing on both supersingular and ordinary curves have been suggested in [1, 3, 9].

The Tate pairing calculation involves an application of Miller's algorithm [19] coupled to a final exponentiation to get a unique value. A typical protocol step requires the calculation of a pairing value followed by a further exponentiation of the result.

In this paper we explore the concept of *compressed pairings*, their efficient computation, and the subsequent processing (typically exponentiation) of pairing

* Supported in part by Enterprise Ireland RIF grant IF/2002/0312/N
** Co-sponsored by Scopus Tecnologia S. A.

values. Our main contribution is to show that one can effectively reduce the bandwidth occupied by pairing values without impairing security nor processing time; in some cases, one even obtains a 30%–40% speed enhancement. As a by-product, our work gives further motivation for the approach Galbraith *et al.* [10], who propose taking the trace of the pairing value to avoid loss of security.

This paper is organized as follows. Section 2 introduces basic mathematical concepts. Section 3 discusses laddering exponentiation of pairing values, and introduces a laddering variant of the BKLS [1] algorithm to compute pairings. Section 4 describes how to compress pairing values to half length, and establishes a connection with the techniques of point compression and point reduction. Section 5 defines a ternary exponentiation ladder for finite fields in characteristic 3. Section 6 describes how to compress pairing values to one third of their length, presents a more efficient and slightly simpler version of the Duursma-Lee algorithm [8] that enables pairing computation in compressed form, and discusses improved variants of point compression and point reduction in characteristic 3. We summarise our work in section 7.

2 Mathematical Preliminaries

The theory behind elliptic curve cryptography is well documented in standard texts. The reader is referred to [18] for more background.

Let p be a prime number, m a positive integer and $\mathbb{F}_{p^m}$ the finite field with $p^m - 1$ elements; p is said to be the *characteristic* of $\mathbb{F}_p$, and m is its *extension degree*. Unless otherwise stated, we assume $p \neq 2$ throughout this paper.

Let $q = p^m$. An *elliptic curve* $E(\mathbb{F}_q)$ is the set of solutions (x, y) over $\mathbb{F}_q$ to an equation of form $E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, where $a_i \in \mathbb{F}_q$, together with an additional *point at infinity*, denoted O. The same equation defines curves over $\mathbb{F}_{q^k}$ for $k > 0$ (although note that the a_i remain in $\mathbb{F}_q$). The number of points of an elliptic curve $E(\mathbb{F}_{q^k})$, denoted $\#E(\mathbb{F}_{q^k})$, is called the *order* of the curve over the field $\mathbb{F}_{q^k}$.

An (additive) Abelian group structure is defined on E by the well known secant-and-tangent method [24]. Let $n = \#E(\mathbb{F}_{q^k})$. The order of a point $P \in E$ is the least nonzero integer r such that $rP = O$, where rP is the sum of r terms equal to P. The order of a point divides the curve order. For a given integer r, the set of all points $P \in E$ such that $rP = O$ is denoted $E[r]$. Commonly this set forms a single cyclic group. However, multiple subgroups of prime order r (where $r^2 \nmid n$) will exist with *embedding degree* k for some $k > 0$ if $r \mid q^k - 1$ and $r \nmid q^s - 1$ for any $0 < s < k$. It is in fact not difficult to find suitable curves with this property for relatively small values of k as described in [2, 6, 7]. We are interested here in curves where k is even, as this case facilitates fast calculation of the Tate pairing [3].

For our purposes, a *divisor* is a formal sum $\mathcal{A} = \sum_P a_P(P)$ of points on the curve $E(\mathbb{F}_{q^k})$. An Abelian group structure is defined on the set of divisors by the addition of corresponding coefficients in their formal sums; in particular, $n\mathcal{A} = \sum_P (n\, a_P)(P)$. The *degree* of a divisor $\mathcal{A}$ is the sum $\deg(\mathcal{A}) = \sum_P a_P$.

Let $f : E(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}$ be a function on the curve and let $\deg(\mathcal{A}) = 0$. We define $f(\mathcal{A}) \equiv \prod_P f(P)^{a_P}$. The divisor of a function f is $(f) \equiv \sum_P \mathrm{ord}_P(f)(P)$. A divisor $\mathcal{A}$ is called *principal* if $\mathcal{A} = (f)$ for some function (f). A divisor $\mathcal{A}$ is principal if and only if $\deg(\mathcal{A}) = 0$ and $\sum_P a_P P = O$ [18, theorem 2.25]. Two divisors $\mathcal{A}$ and $\mathcal{B}$ are *equivalent*, $\mathcal{A} \sim \mathcal{B}$, if their difference $\mathcal{A} - \mathcal{B}$ is a principal divisor. Let $P \in E(\mathbb{F}_q)[r]$ where r is coprime to q, and let $\mathcal{A}_P$ be a divisor equivalent to $(P) - (O)$; under these circumstances the divisor $r\mathcal{A}_P$ is principal, and hence there is a function f_P such that $(f_P) = r\mathcal{A}_P = r(P) - r(O)$. The (reduced) *Tate pairing* of order r is the map $e_r : E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k}) \to \mathbb{F}^*_{q^k}$ given by $e_r(P, Q) = f_P(\mathcal{D})^{(q^k-1)/r}$ for some divisor $\mathcal{D} \sim (Q) - (O)$. The Tate pairing is bilinear, and will be non-degenerate if Q is chosen from a coset containing a point of order r which is linearly independent from P. The computation of $f_P(\mathcal{D})$ is achieved by an application of Miller's algorithm [19], whose output is only defined up to an r-th power in $\mathbb{F}^*_{q^k}$. The final exponentiation to the power of $(q^k - 1)/r$ is needed to produce a unique result, and it also makes it possible to compute $f_P(Q)$ rather than $f_P(\mathcal{D})$ [1]. Sometimes we will drop the r subscript of the Tate pairing, writing simply $e(P, Q)$.

2.1 Lucas sequences

Lucas sequences provide a relatively cheap way of implementing $\mathbb{F}_{q^2}$ exponentiation in a subgroup whose order divides $q+1$. They have been extensively studied in the literature, and a fast "laddering" algorithm for their computation has been developed [14, 15, 28], using ideas originally develop by Montgomery to speed up scalar multiplication on elliptic curves [22]. Lucas sequences have been suggested as a suitable vehicle for certain public-key schemes [26]. The laddering algorithm can in fact be used as an alternative to the standard square-and-multiply approach to exponentiation in any Abelian group, but it is particularly well-suited for Lucas sequences and certain parameterisations of elliptic curves [15]. The authors of [15] go on to emphasise that the laddering algorithm requires very little memory, facilitates parallel computing, and has a natural resistance to side-channel attacks when used in a cryptographic context.

The Lucas sequence consists of a pair of functions $U_k(P, Q)$ and $V_k(P, Q)$, evaluating as elements of $\mathbb{F}_q$. Commonly $Q = 1$, in which case we write simply $U_k(P)$ and $V_k(P)$ or omit the arguments altogether. For this distinguished case the sequences are defined as

$$U_0 = 0,\ U_1 = 1,\ U_{k+1} = PU_k - U_{k-1}$$
$$V_0 = 2,\ V_1 = P,\ V_{k+1} = PV_k - V_{k-1}$$

Only the V_k sequence needs to be explicitly evaluated, as we also have the relationship

$$U_k = (PV_k - 2V_{k-1})/(P^2 - 4)$$

The fast laddering algorithm is described in Appendix A.

3 Exponentiating pairing values

We consider first the case of embedding degree $k = 2$ (although the following discussion also covers the case $k = 2d$ with the substitution $q \rightarrow q^d$).

We represent an element of the field $\mathbb{F}_{q^2}$ as $x + iy$, where $x, y \in \mathbb{F}_q$, and $i^2 = \delta$ for some quadratic non-residue $\delta \in \mathbb{F}_q$. Assume in what follows that all arithmetic is in the field $\mathbb{F}_q$.

The final exponentiation in this case consists of a raising to the power of $(q-1)(q+1)/r$. This can be considered in two parts – exponentiation to the power of $q-1$ followed by exponentiation to the power of $(q+1)/r$. Now if the output of Miller's algorithm is $x + iy \in \mathbb{F}_{q^2}$, then

$$(x+iy)^{q-1} = (x+iy)^q/(x+iy) = (x-iy)/(x+iy)$$

which is obviously much quicker than the standard square-and-multiply algorithm. The element $a + ib \equiv (x+iy)^{q-1}$ calculated in this fashion has the property:

$$a^2 - \delta b^2 = 1 \tag{1}$$

where $a^2 - \delta b^2$ is called the *norm* of $a + ib$; this property, easily verified by simple substitution, is maintained under any subsequent exponentiation. An element of this form in $\mathbb{F}_{q^2}$ is called *unitary* [12]. Also observe that $(a+ib)^{-1} = (a-ib)$ for a unitary element. In fact, any element of $\mathbb{F}_{q^2}$ whose order divides $q+1$ will have this property.

A unitary element can obviously be determined up to the sign of b from a alone, using equation 1. And this is our first observation - the output of the Tate algorithm contains some considerable redundancy. It could be represented by a single element of $\mathbb{F}_q$ and a single bit to represent the sign of b, rather than as a full element of $\mathbb{F}_{q^2}$.

One can efficiently raise a unitary element of $\mathbb{F}_{q^2}$ to a power m by means of Lucas sequences. This is a consequence of the observation that

$$(a+bi)^m = V_m(2a)/2 + U_m(2a)bi,$$

as one can verify by induction. As pointed out above, only $V_m(2a)$ needs to be explicitly calculated.

If M is a multiplication and S a squaring in $\mathbb{F}_q$, then the computational cost of this method to compute $(a+bi)^m$ is therefore $1M + 1S$ per step, where a step involves the processing associated with a single bit of m. The conventional binary exponentiation algorithm for non-special numbers in $\mathbb{F}_{q^2}$ takes 1 squaring and about 1/2 multiplication in $\mathbb{F}_{q^2}$ for an overall cost of $2S + 1M$ and roughly $3M/2$ multiplications in $\mathbb{F}_q$ per step. If $\delta = -1$, then this can be reduced to $2M$ plus $3M/2$ multiplications in $\mathbb{F}_q$ per step[3]. Thus the improved algorithm costs about 60% as much as the basic binary square-and-multiply method. When memory is not an issue the binary algorithm can be implemented by using windowing

[3] If $a+bi$ is unitary and $\delta = -1$, one can compute $(a+bi)^2$ as $(2a^2-1)+[(a+b)^2-1]i$, and $(a+bi)(c+di)$ as $(u-v)+(w-u-v)i$ where $u = ac$, $v = bd$, $w = (a+b)(c+d)$.

techniques, as described in [11]. However the laddering algorithm proposed here for unitary elements will always be faster than a conventional binary algorithm for a general element in $\mathbb{F}_{q^2}$.

Note that this improvement is relevant not only for the second part of the final exponentiation of the Tate pairing, but for any exponentiation directly involving pairing values, as happens in many pairing-based protocols [4, 13, 23].

3.1 A laddering pairing algorithm

For $U, V \in E(\mathbb{F}_q)$, define $g_{U,V}$ to the line through U and V. For all $a, b \in \mathbb{Z}$, the line function satisfies $(g_{aP,bP}) = (aP) + (bP) + (-[a+b]P) - 3(O)$.

Let $P \in E(\mathbb{F}_q)$, and for $c \in \mathbb{Z}$ let f_c be a function with divisor $(f_c) = c(P) - (cP) - (c-1)(O)$. One can show that $f_{a+b}(\mathcal{D}) = f_a(\mathcal{D}) \cdot f_b(\mathcal{D}) \cdot g_{aP,bP}(\mathcal{D})/g_{[a+b]P,-[a+b]P}(\mathcal{D})$ up to a constant nonzero factor. This is called *Miller's formula.* In the computation of the Tate pairing $e_r(P, Q)$, this formula can be simplified to $f_{a+b}(Q) = f_a(Q) \cdot f_b(Q) \cdot g_{aP,bP}(Q)$.

Let $(r_t, \ldots, r_0)_2$ be the binary representation of r. By coupling Miller's simplified formula with Montgomery's scalar multiplication ladder, we get a laddering version of the BKLS algorithm [1] to compute $e_r(P, Q)$:

Laddering BKLS algorithm to compute $e_r(P, Q)$:

```
$v_0 \leftarrow 1, \ v_1 \leftarrow 1$
$R_0 \leftarrow O, \ R_1 \leftarrow P$
for $i \leftarrow t-1$ downto $0$ do
    if $r_i = 0$ then
        $v_0 \leftarrow v_0^2 \cdot g_{R_0,R_0}(Q), \ R_1 \leftarrow R_0 + R_1$
        $R_0 \leftarrow 2R_0, \ v_1 \leftarrow v_0 \cdot g_{R_0,P}(Q)$
    else
        $v_1 \leftarrow v_1^2 \cdot g_{R_1,R_1}(Q), \ R_0 \leftarrow R_0 + R_1$
        $R_1 \leftarrow 2R_1, \ v_0 \leftarrow v_1 \cdot g_{R_1,-P}(Q)$
    end if
end for
return $v_0^{(q^k-1)/r}$
```

4 Compressing pairings to half length

Instead of keeping the full $a + bi$ value of the Tate Pairing, it may be possible for cryptographic purposes to discard b altogether, leaving the values defined only up to complex conjugation, which means one of the pairing arguments will only be defined up to a sign:

$$e(P, Q) = a + bi \Rightarrow a - bi = (a + bi)^{-1} = e(P, Q)^{-1} = e(P, -Q).$$

This is similar to the point reduction technique, whereby instead of keeping $Q = (x, y)$ one only keeps the abscissa x.

Definition 1. *The* $\mathbb{F}_q$-trace *of an element* $u \in \mathbb{F}_{q^2}$ *is the sum of the conjugates of* u, $\mathrm{tr}(u) = u + u^q$.

Notice that $\mathrm{tr}(a + ib) = (a + ib) + (a - ib) = 2a$, in effect discarding the imaginary part. We define the *compressed Tate pairing* $\varepsilon(P, Q)$ as $\mathrm{tr}(e(P, Q))$.

4.1 Point reduction

Point reduction is an optimization technique introduced by V. Miller in 1985 [20]. It consists of basing cryptographic protocols solely on the x coordinate of the points involved rather than using both coordinates. This setting is possible because the x coordinate of any multiple of a given point P depends only on the x coordinate of P. A related but less efficient technique is that of point compression, which consists of keeping not only the x coordinate but also a single bit β from the y coordinate to choose between the two roots $y_\pm = \pm\sqrt{x^3 + ax + b}$.

Some pairing-based cryptosystems have been originally defined to take profit from point reduction. An example is the BLS signature scheme [5], where the signature of a message represented by a curve point M under the signing key s is the x coordinate σ of the point $S = sM$. This means that, implicitly, the actual signature is $\pm S$ rather than S alone. To verify a BLS signature, the verifier checks whether $e(M, V) = e(\pm S, Q)$, where the verification key is $V = sQ$. Incidentally, the verification key itself can be reduced to its x coordinate (say, ξ), even though this possibility does not seem to have been considered by the authors of BLS.

4.2 Coupling point reduction with compressed pairings

Verifying a BLS signature involves computing a point $V \in \{V, -V\}$ from ξ, a point $S' \in \{S, -S\}$ from σ and checking whether $e_r(M, V') = e(S', Q)$ or $e(M, V') = e(S', Q)^{-1}$. Using the property that any pairing value z is unitary (and hence $z^{-1} = \overline{z}$), one can simply check whether $\mathrm{tr}(e(M, V')) = \mathrm{tr}(e(S', Q))$. This is especially interesting, since a compressed pairing $\varepsilon(P, Q)$ is precisely $\mathrm{tr}(e(\pm P, \pm Q))$.

An important aside is that exponentiation of compressed pairings must take into account the fact that they are actually traces of full pairings. This means one cannot exponentiate a pairing as if it were a simple $\mathbb{F}_{q^{k/2}}$ value; rather, one must always handle it as a Lucas sequence element.

5 A ternary exponentiation ladder

Supersingular curves in characteristic 3 are a popular choice of underlying algebraic structure for pairing-based cryptosystems, since many optimisations are possible in such a setting [1, 8, 9]. Pairing compression is possible for those systems, and we now propose a ternary ladder for Lucas sequences in characteristic 3 that keeps the exponentiation cost in $\mathbb{F}_{q^k}$ within about 33% of the exponentiation cost in $\mathbb{F}_{q^{k/2}}$.

Assume the sequence element index is written in *signed* ternary notation, $K = (d_{t-1}, \ldots, d_0)_{\bar{3}}$, with $d_{t-1} = 1$. At step j (counting downwards from $t-1$ to 0), we want to compute V_{K_j} where $K_j = \sum_{i=j}^{t-1} d_i 3^i$. Thus, by definition, $K_j = 3K_{j+1} + d_i$.

For $d_j = -1$, we write down the formulas to compute $V_{3K_{j+1}-2}$, $V_{3K_{j+1}-1}$, and $V_{3K_{j+1}}$:

$$\begin{aligned} V_{3K_{j+1}} &= V_{K_{j+1}}^3 \\ V_{3K_{j+1}-1} &= PV_{3K_{j+1}-2} - V_{K_{j+1}-1}^3 \\ V_{3K_{j+1}-2} &= PV_{3K_{j+1}-1} - V_{K_{j+1}}^3 \end{aligned}$$

Similarly, for $d_j = 1$ we write down the formulas to compute $V_{3K_{j+1}}$, $V_{3K_{j+1}+1}$, and $V_{3K_{j+1}+2}$:

$$\begin{aligned} V_{3K_{j+1}} &= V_{K_{j+1}}^3 \\ V_{3K_{j+1}+1} &= PV_{3K_{j+1}+2} - V_{K_{j+1}+1}^3 \\ V_{3K_{j+1}+2} &= PV_{3K_{j+1}+1} - V_{K_{j+1}}^3 \end{aligned}$$

In each case, the second and third relations constitute a simple linear system. Solving them, we get these expressions for $V_{3K_{j+1}-1}$, $V_{3K_{j+1}}$, and $V_{3K_{j+1}+1}$:

$$\begin{aligned} V_{3K_{j+1}-1} &= (P^2-1)^{-1}(PV_{K_{j+1}}^3 + V_{K_{j+1}-1}^3) \\ &= (P^2-1)^{-1}[PV_{K_{j+1}}^3 + (PV_{K_{j+1}} - V_{K_{j+1}+1})^3] \\ &= (P^2-1)^{-1}[(P+P^3)V_{K_{j+1}}^3 - V_{K_{j+1}+1}^3] \\ V_{3K_{j+1}} &= V_{K_{j+1}}^3 \\ V_{3K_{j+1}+1} &= (P^2-1)^{-1}(PV_{K_{j+1}}^3 + V_{K_{j+1}+1}^3) \\ &= (P^2-1)^{-1}[PV_{K_{j+1}}^3 + (PV_{K_{j+1}} - V_{K_{j+1}-1})^3] \\ &= (P^2-1)^{-1}[(P+P^3)V_{K_{j+1}}^3 - V_{K_{j+1}-1}^3] \end{aligned}$$

If $(P^2-1)^{-1}$ and $P+P^3$ are precomputed, computing $V_{3K_{j+1}}$ and *one* of $V_{3K_{j+1}-1}$ or $V_{3K_{j+1}+1}$ involves two products and two cubes, and the computation can be carried out using only $V_{K_{j+1}}$ and *one* of $V_{K_{j+1}-1}$ or $V_{K_{j+1}+1}$. We can therefore keep track of which value between these two actually accompanies $V_{K_{j+1}}$, and compute V_{K_j} and V_{K_j+1} at the cost of only 2 products and two cubes per step. Besides, since we are working in characteristic 3, the cost of cubing is negligible compared to the cost of multiplying.

The binary ladder computes V_{K_j} and V_{K_j+1} at the cost of one square and one product, or about 1.8 product, per step. However, the step count of the ternary ladder is only about $1/\lg(3)$ of its binary counterpart, and hence its total cost is about 70% of the binary ladder. We point out that the ternary ladder can be used for plain exponentiation in characteristic 3 as an independent technique, even in contexts where compressed pairings are not desired or not an option.

A detailed ternary ladder algorithm is described in Appendix A.

6 Compressing pairings to a third of their length

Definition 2. *The* $\mathbb{F}_{q^2}$-trace *of an element* $f \in \mathbb{F}_{q^6}$ *is the value* $\mathrm{tr}(f) = f + f^{q^2} + f^{q^4} \in \mathbb{F}_{q^2}$.

When the elliptic curve has an embedding degree $k = 6$, the Tate pairing algorithm outputs an element of $\mathbb{F}_{q^6}$ of order r, where r divides $q^6 - 1$, but not $q^i - 1$ for $0 < i < 6$. Now $q^6 - 1 = \Phi_1(q)\Phi_2(q)\Phi_3(q)\Phi_6(q)$. Therefore the output of the Tate pairing is an element of order r which divides $\Phi_6(q) = q^2 - q + 1$. For $q \equiv 2 \pmod 3$, these are precisely the type of points considered in the XTR public key scheme [16], and all of the time/space optimizations that have been developed for this scheme [16, 27] apply here as well. In particular, we note that laddering algorithms again appear to be optimal [27], and the Tate pairing output can be represented by its $\mathbb{F}_{q^2}$-trace, and hence compressed by a factor of 3. Observe that the compressed value, being a trace, must be implicitly exponentiated using the Lenstra-Verheul algorithm [16, Algorithm 2.3.7] – the trace value *per se* is not even a point of order r.

For supersingular curves in characteristic 3 we can do better than merely take the trace – rather, it is possible to do all computations without resorting to arithmetic any more complex than that on $\mathbb{F}_{q^2}$ and implicit trace exponentiation.

6.1 Simpler arithmetic for pairing computation in characteristic 3

Let $q = 3^m$ for some odd m, let $b = \pm 1$, and let $\sigma, \rho \in \mathbb{F}_{q^6}$ be elements satisfying $\sigma^2 + 1 = 0$ and $\rho^3 - \rho - b = 0$. The *modified Tate pairing* on the supersingular curve $E(\mathbb{F}_{3^m}) : y^2 = x^3 - x + b$ is the mapping $f_P(\phi(Q))^{(q^6-1)/r}$ where $\phi : E(\mathbb{F}_q) \to E(\mathbb{F}_{q^6})$ is the distortion map $\phi(x, y) = (\rho - x, \sigma y)$.

Duursma and Lee showed [8, Theorem 5] that the modified Tate pairing for points $P = (\alpha, \beta)$ and $Q = (x, y)$ can be written as a product of factors of form $g = \beta y \bar{\sigma} - (\alpha + x - \rho + b)^2$. This expression can be rewritten as $g = \lambda - \mu\rho - \rho^2$, where $\mu \equiv \alpha + x + b \in \mathbb{F}_q$ and $\lambda \equiv \beta y \bar{\sigma} - \mu^2 \in \mathbb{F}_{q^2}$. Specifically, the Duursma-Lee algorithm to compute $f_P(\phi(Q))$ is as follows (cf. [8, Algorithm 4]):

Duursma-Lee algorithm to compute $f_P(\phi(Q))$:

```
$f \leftarrow 1$
for $i \leftarrow 1$ to $m$ do
    $\alpha \leftarrow \alpha^3, \quad \beta \leftarrow \beta^3$
    $\mu \leftarrow \alpha + x + b, \quad \lambda \leftarrow \beta y \bar{\sigma} - \mu^2$
    $g \leftarrow \lambda - \mu\rho - \rho^2, \quad f \leftarrow f \cdot g$
    $x \leftarrow x^{1/3}, \quad y \leftarrow y^{1/3}$
end for
return $f$
```

The output is an element $f \in \mathbb{F}_{q^6}$. We now show that this algorithm can be modified to compute $\mathrm{tr}(f)$ instead, by maintaining a ladder of three values

$[\mathrm{tr}(f), \mathrm{tr}(f\rho), \mathrm{tr}(f\rho^2)]$. Since f is initialized to 1, the initial ladder can be computed from ρ alone, namely, $[\mathrm{tr}(1), \mathrm{tr}(\rho), \mathrm{tr}(\rho^2)] = [0, 0, (2m^2) \bmod 3]$, as one readily deduces from the definition of ρ:

Theorem 1. *Let $q = 3^m$ for some m, and let $\rho \in \mathbb{F}_{q^6}$ satisfy $\rho^3 - \rho - b = 0$. Then $\mathrm{tr}(\rho) = 0$ and $\mathrm{tr}(\rho^2) = (2m^2) \bmod 3$.*

Proof. From $\rho^3 = \rho + b$ it follows by induction that $\rho^{3^n} = \rho + (n \bmod 3)b$, and hence $\rho^{q^2} = \rho^{3^{2m}} = \rho + (2m \bmod 3)b$ and $\rho^{q^4} = \rho^{3^{4m}} = \rho + (4m \bmod 3)b$, so that $\mathrm{tr}(\rho) = \rho + \rho^{q^2} + \rho^{q^4} = \rho + \rho + (2m \bmod 3)b + \rho + (4m \bmod 3)b = (6m \bmod 3)b = 0$. Moreover, $(\rho^2)^{3^n} = (\rho^{3^n})^2 = (\rho + (n \bmod 3)b)^2 = \rho^2 - (n \bmod 3)b\rho + (n \bmod 3)^2$, so that $\mathrm{tr}(\rho^2) = \rho^2 + (\rho^2)^{q^2} + (\rho^2)^{q^4} = \rho^2 + \rho^2 - (2m \bmod 3)b\rho + (2m \bmod 3)^2 + \rho^2 - (4m \bmod 3)b\rho + (4m \bmod 3)^2 = -(6m \bmod 3)b\rho + (2m \bmod 3)^2 + (4m \bmod 3)^2 = (2m \bmod 3)^2 + (4m \bmod 3)^2 = (2m^2) \bmod 3$. □

Notice that, if $3 \nmid m$ (which happens most of the time, since m is usually prime), this simplifies to $[\mathrm{tr}(1), \mathrm{tr}(\rho), \mathrm{tr}(\rho^2)] = [0, 0, 2]$.

At each step of the loop, we compute $[\mathrm{tr}(fg), \mathrm{tr}(fg\rho), \mathrm{tr}(fg\rho^2)]$ according to the following theorem:

Theorem 2.

$$\begin{bmatrix} \mathrm{tr}(fg) \\ \mathrm{tr}(fg\rho) \\ \mathrm{tr}(fg\rho^2) \end{bmatrix} = A \cdot \begin{bmatrix} \mathrm{tr}(f) \\ \mathrm{tr}(f\rho) \\ \mathrm{tr}(f\rho^2) \end{bmatrix}, \textit{ where } A \equiv \begin{bmatrix} \lambda & -\mu & -1 \\ -b & (\lambda - 1) & -\mu \\ -b\mu & -(\mu + b) & (\lambda - 1) \end{bmatrix}.$$

Proof. Using the $\mathbb{F}_{q^2}$-linearity of the trace and the defining property $\rho^3 = \rho + b$, we have $fg = f(\lambda - \mu\rho - \rho^2) \implies \mathrm{tr}(fg) = \lambda\,\mathrm{tr}(f) - \mu\,\mathrm{tr}(f\rho) - \mathrm{tr}(f\rho^2)$. Similarly, $fg\rho = f(\lambda - \mu\rho - \rho^2)\rho = \lambda f\rho - \mu f\rho^2 - f\rho - bf \implies \mathrm{tr}(fg\rho) = -b\,\mathrm{tr}(f) + (\lambda - 1)\,\mathrm{tr}(f\rho) - \mu\,\mathrm{tr}(f\rho^2)$. Finally, $fg\rho^2 = -bf\rho + (\lambda - 1)f\rho^2 - \mu f\rho - \mu bf \implies \mathrm{tr}(fg\rho^2) = -\mu b\,\mathrm{tr}(f) - (\mu + b)\,\mathrm{tr}(f\rho) + (\lambda - 1)\,\mathrm{tr}(f\rho^2)$. □

Therefore, defining $L \equiv [L_0, L_1, L_2]^T = [\mathrm{tr}(f), \mathrm{tr}(f\rho), \mathrm{tr}(f\rho^2)]^T$ and using the matrix A defined above, the modified algorithm to compute implicit pairings reads:

A laddering algorithm to compute $\mathrm{tr}(f_P(\phi(Q)))$**:**

```
$L \leftarrow [0, 0, (2m^2) \bmod 3]$   // this is simply $[\mathrm{tr}(1), \mathrm{tr}(\rho), \mathrm{tr}(\rho^2)]$
for $i \leftarrow 1$ to $m$ do
    $\alpha \leftarrow \alpha^3, \quad \beta \leftarrow \beta^3$
    $\mu \leftarrow \alpha + x + b, \quad \lambda \leftarrow \beta y \bar{\sigma} - \mu^2$
    $L \leftarrow A \cdot L$
    $x \leftarrow x^{1/3}, \quad y \leftarrow y^{1/3}$
end for
return $L_0$
```

Each step of this algorithm takes 17 $\mathbb{F}_q$ multiplications. This compares well with the original Duursma-Lee algorithm where each step takes 20 $\mathbb{F}_q$ multiplications, and completely avoids $\mathbb{F}_{q^6}$ arithmetic.

6.2 Implicit exponentiation in characteristic 3

To obtain a unique pairing value from the output of the implicit pairing algorithm, we must replace the final exponentiation $f_P(\phi(Q))^{(q^6-1)/r}$ by an implicit exponentiation $\mathrm{tr}(f_P(\phi(Q))^{(q^6-1)/r})$. It is also quite commonplace that the pairing value undergoes further exponentiation as dictated by the underlying cryptographic protocol. We are thus confronted with the task of computing $\mathrm{tr}(g^m)$ given the value of $\mathrm{tr}(g)$. The Lenstra-Verheul algorithm [16, Algorithm 2.3.7] performs this task, but demands that the characteristic be $p \equiv 2 \pmod 3$. We now describe a variant that works in characteristic 3.

Let $c \in \mathbb{F}_{q^2}$, and let $F(c, X) \equiv X^3 - cX^2 + c^q X - 1 \in \mathbb{F}_{q^2}[X]$ with roots $h_0, h_1, h_2 \in \mathbb{F}_{q^6}$. One can show[4] [16, Lemma 2.2.1] that, if $g \in \mathbb{F}_{q^6}$ is an element of order dividing $\Phi_6(q) = q^2 - q + 1$, then the roots of $F(\mathrm{tr}(g), X)$ are the $\mathbb{F}_{q^2}$-conjugates of g.

Defining $c_n \equiv h_0^n + h_1^n + h_2^n$, one can further show [16, Lemmas 2.3.2 and 2.3.4] that $c_{-n} = c_n^q$ and $c_{u+v} = c_u c_v - c_v^q c_{u-v} + c_{u-2v}$. From these properties, one easily deduces the following relations in characteristic 3:

$$\begin{aligned}
c_{2n} &= c_n^2 + c_n^q \\
c_{3n} &= c_n^3 \\
c_{3n-1} &= c_{2n} \cdot c_{n-1} - c_{n-1}^q \cdot c_{n+1} + c_2 \\
c_{3n-2} &= c^{-q} \cdot (c_{n-1} - c_n)^3 + c^{1-q} \cdot c_{3n-1} \\
c_{3n+1} &= c_{2n} \cdot c_{n+1} - c_{n+1}^q \cdot c_{n-1} + c_2^q \\
c_{3n+2} &= c^{-1} \cdot (c_{n+1} - c_n)^3 + c^{q-1} \cdot c_{3n+1}
\end{aligned}$$

Computing c_{2n} takes two $\mathbb{F}_q$ multiplications, $c_{3n\pm1}$ takes four $\mathbb{F}_q$ multiplications, and $c_{3n\pm2}$ takes six $\mathbb{F}_q$ multiplications.

Define $L_n(c) \equiv \langle c_{3n}, c_{3n+1}, c_{3n+2}, c_{3n+3}\rangle \in (\mathbb{F}_{q^2})^3$. Using the above formulas, one can compute any one of $L_{3n}(c)$, $L_{3n+1}(c)$, or $L_{3n+2}(c)$ from $L_n(c)$ at the cost of 12 $\mathbb{F}_q$ multiplications:

$$\begin{aligned}
L_{3n} &= \langle c_{9n}, \quad c_{9n+1}, c_{9n+2}, c_{9n+3}\rangle = \langle c_{3(3n)}, \quad c_{3(3n+1)-2}, c_{3(3n+1)-1}, c_{3(3n+1)}\rangle \\
L_{3n+1} &= \langle c_{9n+3}, c_{9n+4}, c_{9n+5}, c_{9n+6}\rangle = \langle c_{3(3n+1)}, c_{3(3n+1)+1}, c_{3(3n+2)-1}, c_{3(3n+2)}\rangle \\
L_{3n+2} &= \langle c_{9n+6}, c_{9n+7}, c_{9n+8}, c_{9n+9}\rangle = \langle c_{3(3n+2)}, c_{3(3n+2)+1}, c_{3(3n+2)+2}, c_{3(3n+3)}\rangle
\end{aligned}$$

From the definition of c_n, it is clear that $c_n = \mathrm{tr}(g^n)$ if $c = \mathrm{tr}(g)$. Hence, if $L_{\lfloor n/3 \rfloor}(\mathrm{tr}(g)) = \langle S_0, S_1, S_2, S_3\rangle$, then $\mathrm{tr}(g^n) = S_{n \bmod 3}$. The total cost of this algorithm, about $7.6 \lg n$ $\mathbb{F}_q$ multiplications, matches the complexity of the ternary ladder introduced in section 5 for $\mathbb{F}_{q^3}$-trace exponentiation. Appendix B lists this algorithm in detail. We point out that this ternary ladder can also be the basis of a characteristic 3 variant of the XTR cryptosystem.

[4] The proofs of [16, lemmas 2.2.1, 2.3.2, and 2.3.4] are independent of the field characteristic.

6.3 Coupling pairing compression with point reduction

A nice feature of this algorithm is that it is compatible with a variant of the point reduction technique.

The conventional approach to compress a point $R = (u, v)$ is to keep only u and a single bit of v; point reduction discards v altogether. In characteristic 3, it is more advantageous to discard u instead, keeping v and a trit of u to distinguish among the solutions of the curve equation $u^3 - u + (b - v^2) = 0$; alternatively, one can reduce R by keeping only v and modifying the cryptographic protocols to allow for any of the three points R_0, R_1, and R_2 that share the same v. Thus, we will show that the input to the laddering algorithm of section 6.1 can be only y (or β); the corresponding x (or α) can be easily recovered except for a trit, and the actual choice of this trit does not affect the compressed pairing value.

Let $z \in \mathbb{F}_{q^6}$ where $q = 3^m$ for odd m, and assume the order r of z divides $\Phi_6(q)$, i.e. $r \mid q^2 - q + 1$. The conjugates of z are z, z^{q^2}, and z^{q^4}, or equivalently z, z^{q-1}, and z^{-q}, since $q^2 \equiv q - 1 \pmod{r}$ and $q^4 \equiv -q \pmod{r}$. The trace of z is the sum of the conjugates, $\mathrm{tr}(z) = z + z^{q-1} + z^{-q}$ [16]. Consider the supersingular elliptic curve $E : y^2 = x^3 - x + b$, $b \in \{-1, 1\}$, whose order is [18, section 5.2.2] $n = q + 1 - t = 3^m + 1 \pm 3^{(m+1)/2}$, where $t = \pm 3^{(m+1)/2}$ is the trace of the Frobenius.

Let $P = (x, y) \in E(\mathbb{F}_q)$, and let $Q \in E(\mathbb{F}_{q^6})$ be a linearly independent point. The conjugates of $e(P, Q)$ are $e(P, Q)$, $e(P, Q)^{q-1} = e([q - 1]P, Q)$, and $e(P, Q)^{-q} = e(-qP, Q)$. The following property holds:

Lemma 1. *If $P \in E[r]$, points P, $[q - 1]P$, and $-qP$ share precisely the same y coordinate.*

Proof. Let $P = (x, y)$. A simple inspection of the group law for characteristic 3 [1] reveals that $3P = (x^9 - b,\ -y^9)$, and hence $3^j P = (x^{9^j} - (j \bmod 3)b,\ (-1)^j y^{9^j})$, where we take the $(j \bmod 3)$ factor to be an element of $\mathbb{F}_3$. Thus $[q-1]P = q^2 P = 3^{2m} P = (x^{9^{2m}} - (2m \bmod 3)b,\ (-1)^{2m} y^{9^{2m}}) = (x^{3^{4m}} + (m \bmod 3)b,\ y^{3^{4m}}) = (x + (m \bmod 3)b,\ y)$, where we used the fact that $u^{3^m} = u$ for any $u \in \mathbb{F}_{3^m}$. Similarly, $-qP = q^2(q^2 P) = q^2(x + (m \bmod 3)b,\ y) = (x - (m \bmod 3)b,\ y)$. □

We see that, for $m \not\equiv 0 \pmod{3}$, the x coordinates of P, $[q - 1]P$, and $-qP$ are the three solutions to $x^3 - x + (1 - y^2) = 0$, which are exactly $\{x, x+1, x+2\}$. Obviously, the traces of the pairings computed from the conjugates of P are all equal, since $\mathrm{tr}(e(P, Q))$ is simply the sum of the conjugates of $e(P, Q)$. Thus, the actual solution x to the curve equation above used to compute $\mathrm{tr}(e(P, Q))$ is irrelevant. Also, computing x from y is very efficient, since it amounts to solving a linear system (see appendix C).

7 Conclusions

We have introduced the notion of *compressed pairings*, and suggested how they can be realised as traces of ordinary Tate pairings. We also described how compressed pairings can be computed and implicitly exponentiated by means of laddering algorithms, with a compression ratio of 1/2 in characteristic $p > 3$ and 1/3 in characteristic 3; our algorithms thus reduce bandwidth requirements without impairing performance. Finally, we showed how to couple compressed pairings with the technique of point compression or point reduction. As a side result, we proposed an efficient laddering algorithm for plain exponentitation in characteristic 3, which can be used even in contexts where compressed pairings are not desired.

Our work constitutes evidence that the security of pairing-based cryptosystems is linked to the security of the Lucas/XTR schemes, and gives further motivation for the approach of Galbraith *et al.* regarding the use of traces to prevent security losses.

We leave it as an open problem to find a method to compute pairings directly in compressed form when the compression ratio is 1/3 or better on ordinary (non-supersingular) curves in characteristic $p > 3$.

8 Acknowledgements

We are grateful to Steven Galbraith and Waldyr Benits Jr. for their valuable comments during the preparation of this work.

References

1. P. S. L. M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – Crypto'2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag, 2002.
2. P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks – SCN'2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 263–273. Springer-Verlag, 2002.
3. P. S. L. M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In *Selected Areas in Cryptography – SAC 2003*, 2003. to appear.
4. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
5. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology – Asiacrypt'2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2002.
6. F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. Cryptology ePrint Archive, Report 2003/143, 2003. Available from `http://eprint.iacr.org/2003/143`.
7. R. Dupont, A. Enge, and F. Morain. Building curves with arbitrary small MOV degree over finite prime fields. Cryptology ePrint Archive, Report 2002/094, 2002. Available from `http://eprint.iacr.org/2002/094`.
8. I. Duursma and H.-S. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In *Advances in Cryptology – Asiacrypt'2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 111–123. Springer-Verlag, 2003.
9. S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithm Number Theory Symposium – ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.
10. S. Galbraith, H. Hopkins, and I. Shparlinski. Secure bilinear diffie-hellman bits. Cryptology ePrint Archive, Report 2002/155, 2002. Available from `http://eprint.iacr.org/2002/155`.
11. D. Gordon. A survey of fast exponentiation methods. *Journal of Algorithms*, 27:129–146, 2002.
12. K. Hoffman and R. Kunze. *Linear Algebra*. Prentice Hall, New Jersey, USA, 2nd edition, 1971.
13. A. Joux. A one-round protocol for tripartite Diffie-Hellman. In *Algorithm Number Theory Symposium – ANTS IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394, Berlin, Germany, 2000. Springer-Verlag.
14. M. Joye and J. J. Quisquater. Efficient computation of full Lucas sequences. *Electronics Letters*, 32(6):537–538, 1996.
15. M. Joye and S. Yen. The montgomery powering ladder. In *Cryptographic Hardware and Embedded Systems - CHES'2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302, Berlin, Germany, 2003. Springer-Verlag.
16. A. K. Lenstra and E. R. Verheul. The xtr public key system. In *Advances in Cryptology – Crypto'2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 1–19. Springer-Verlag, 2000.
17. R. Lidl and H. Niederreiter. *Finite Fields*. Number 20 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, UK, 2nd edition, 1997.

18. A. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
19. V. S. Miller. Short programs for functions on curves. Unpublished manuscript, 1986. Available from `http://crypto.stanford.edu/miller/miller.pdf`.
20. V. S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology – Crypto'85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426, Santa Barbara, USA, 1986. Springer-Verlag.
21. A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.
22. P. L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987.
23. D. Nalla and K. C. Reddy. Signcryption scheme for identity-based cryptosystems. Cryptology ePrint Archive, Report 2003/066, 2002. Available from `http://eprint.iacr.org/2003/066`.
24. J.H. Silverman. *The Arithmetic of Elliptic Curves*. Number 106 in Graduate Texts in Mathematics. Springer-Verlag, Berlin, 1986.
25. N. P. Smart. An identity based authenticated key agreement protocol based on the weil pairing. *Electronics Letters*, 38:630–632, 2002.
26. P. J. Smith. LUC public-key encryption: A secure alternative to RSA. *Dr. Dobbs Journal*, 18(1):44–49,90–92, 1993.
27. M. Stam and A. K. Lenstra. Speeding up XTR. In *Advances in Cryptology – Asiacrypt'2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 125–143. Springer-Verlag, 2001.
28. S. M. Yen and C. S. Laih. Fast algorithms for LUC digital signature computation. *IEE Proceedings on Computers and Digital Techniques*, 142(2):165–169, 1995.

A Computation of Lucas sequence elements

The Lucas sequence $V_n(P, 1)$ is defined by the following recurrence relations:

$$V_0 = 2,\ V_1 = P,\ V_{n+1} = PV_n - V_{n-1}.$$

Let $n = (n_t \dots n_0)_2$ be an integer in binary representation, with $n_t = 1$. The Lucas sequence element $V_n(P, 1)$ can be computed as:

```
$v_0 \leftarrow 2, \quad v_1 \leftarrow P$
for $j \leftarrow t$ downto $0$ do
    if $n_j = 1$ then
        $v_0 \leftarrow v_0 v_1 - P, \quad v_1 \leftarrow v_1^2 - 2$
    else
        $v_1 \leftarrow v_0 v_1 - P, \quad v_0 \leftarrow v_0^2 - 2$
    end if
end for
return $v_0$
```

Let $n = (n_t \dots n_0)_{\bar{3}}$ be the *signed* ternary representation of $n \geqslant 0$. The Lucas sequence element $V_n(P, 1)$ in characteristic 3 (as needed for the implicit exponentiation of $\mathbb{F}_{q^3}$-traces of $\mathbb{F}_{q^6}$ values) can be computed using the following algorithm:

```
$\mu \leftarrow (P^2 - 1)^{-1}, \quad T \leftarrow P + P^3$
$v_0 \leftarrow 2, \quad v_1 \leftarrow P, \quad up \leftarrow$ true
for $j \leftarrow t$ downto $0$ do
    $w \leftarrow v_0^3$
    if $n_j = -1$ then
        $v_0 \leftarrow$ if $up$ then $\mu(Tw - v_1^3)$ else $\mu(Pw + v_1^3)$
        $v_1 \leftarrow w$
        $up \leftarrow$ true
    else if $n_j = 1$ then
        $v_0 \leftarrow$ if $up$ then $\mu(Pw + v_1^3)$ else $\mu(Tw - v_1^3)$
        $v_1 \leftarrow w$
        $up \leftarrow$ false
    else /* $n_j = 0$ */
        $v_1 \leftarrow$ if $up$ then $\mu(Pw + v_1^3)$ else $\mu(Tw - v_1^3)$
        $v_0 \leftarrow w$
        $up \leftarrow$ true
    end if
end for
return $v_0$
```

B Implicit exponentiation of $\mathbb{F}_{q^{k/3}}$-traces

Let $n = (n_t \dots n_0)_3$ be the plain ternary representation of $n \geqslant 0$. The following algorithm computes the $\mathbb{F}_{q^2}$-trace $c_n \equiv \mathrm{tr}(g^n)$ of an element $g \in \mathbb{F}_{q^6}$ from its $\mathbb{F}_{q^2}$-trace $c \equiv \mathrm{tr}(g)$.

```
$c^{-1} \leftarrow c^q \cdot (c^q \cdot c)^{-1}$ // N.B. $(c^q \cdot c) \in \mathbb{F}_q$
$c^{q-1} \leftarrow c^q \cdot c^{-1}$, $c^{-q} \leftarrow (c^{-1})^q$, $c^{1-q} \leftarrow (c^{q-1})^q$, $c_2 \leftarrow c^2 + c^q$
$S_0 \leftarrow 0$, $S_1 \leftarrow c$, $S_2 \leftarrow c_2$, $S_3 \leftarrow c^3$
for $j \leftarrow t$ downto $0$ do
    if $n_j = 0$ then
        $S_3' \leftarrow S_1^3$
        $S_2' \leftarrow (S_1^2 + S_1^q) \cdot S_0 - S_0^q \cdot S_2 + c_2$
        $S_1' \leftarrow c^{-q} \cdot (S_0 - S_1)^3 + c^{1-q} \cdot S_2'$
        $S_0' \leftarrow S_0^3$
    else if $n_j = 1$ then
        $s_1 \leftarrow S_1$
        $s_2 \leftarrow S_2$
        $S_1' \leftarrow (s_1^2 + s_1^q) \cdot s_2 - s_2^q \cdot S_0 + c_2^q$
        $S_0' \leftarrow s_1^3$
        $S_2' \leftarrow (s_2^2 + s_2^q) \cdot s_1 - s_1^q \cdot S_3 + c_2$
        $S_3' \leftarrow s_2^3$
    else /* $n_j = 2$ */
        $S_0' \leftarrow S_2^3$
        $S_1' \leftarrow (S_2^2 + S_2^q) \cdot S_3 - S_3^q \cdot S_1 + c_2^q$
        $S_2' \leftarrow c^{-1} \cdot (S_3 - S_2)^3 + c^{1-q} \cdot S_1'$
        $S_3' \leftarrow S_3^3$
    end if
end for
return $S_{n \bmod 3}$
```

C Solving the curve equation in characteristic 3

Definition 3. *The* absolute trace *of a field element* $a \in \mathbb{F}_{3^m}$ *is the linear form:*

$$\mathrm{tr}(a) = a + a^3 + a^9 + \cdots + a^{3^{m-1}}.$$

The absolute trace will always be in $\mathbb{F}_3$ as one can easily check by noticing from the above definition that $\mathrm{tr}(a)^3 = \mathrm{tr}(a)$, for all $a \in \mathbb{F}_{3^m}$. Being surjective and linear over $\mathbb{F}_3$, it can always be represented as a (usually sparse) dual vector $T \in \mathbb{F}_{3^m}$ in a given basis, so that one can compute $\mathrm{tr}(u) = T \cdot u$ in no more than $O(m)$ time. In a normal basis $\{\theta^{3^i}\}$ with $\mathrm{tr}(\theta) = 1$, computing $\mathrm{tr}(u)$ amounts to summing up all coefficients of u.

The coordinates of a curve point $P = (x, y)$ are constrained by the curve equation to satisfy $y^2 = x^3 + ax + b$. Thus one can represent a point as either (x, β)

where $\beta \in \mathbb{F}_2$ indicates which of the two roots correspond to $y = \pm\sqrt{x^3 + ax + b}$, or else by (τ, y) where $\tau \in \mathbb{F}_3$ indicates which of the three solutions one has to take of the equation $x^3 + ax + (b - y^2) = 0$. In characteristic 3, cubing is a linear operation, which makes the second possibility more advantageous.

Consider the special equation $x^3 - x - u = 0$ for a given $u \in \mathbb{F}_{3^m}$, which is relevant for supersingular curves in characteristic 3. This equation has a solution if, and only if, $\mathrm{tr}(u) = 0$ [17, theorem 2.25]. This is the case for 1/3 of the elements in $\mathbb{F}_{3^m}$, since the trace function is linear and surjective. The complexity of solving the cubic equation is only $O(m^2)$, as we show now.

Let $\mathcal{C} : \mathbb{F}_{3^m} \to \mathbb{F}_{3^m}$ be defined by $\mathcal{C}(x) = x^3 - x$. The kernel of $\mathcal{C}$ is $\mathbb{F}_3$ [17, chapter 2,section 1], hence the rank of $\mathcal{C}$ is $m - 1$ [12, section 3.1, theorem 2].

Theorem 3. *The equation $x^3 - x - u = 0$ over $\mathbb{F}_{3^m}$ can be solved in $O(m^2)$ steps.*

Proof. If $\mathbb{F}_{3^m}$ is represented in standard polynomial basis, the cubic equation reduces to a system of linear equations with coefficients in $\mathbb{F}_3$, and can be solved in no more than $O(m^2)$ steps. This is achieved by first checking whether the system has solutions, i.e. whether $\mathrm{tr}(u) = 0$. If so, since the rank of $\mathcal{C}$ is $m - 1$ one obtains an invertible $(m-1) \times (m-1)$ matrix A by leaving out the one row and correspondingly one column of the matrix representation of $\mathcal{C}$ on the given basis. A solution of the cubic equation is then given by an arbitrary element $x_0 \in \mathbb{F}_3$ and by the solution of system $A\tilde{x} = \tilde{u}$, which is obtained as $\tilde{x} = A^{-1}\tilde{u}$ in $O(m^2)$ time.

Using a normal basis to represent field elements, it is not difficult to see that the cubic equation can be efficiently solved in $O(m)$ time by the following algorithm (the proof is straightforward and left as an exercise):

Cubic equation solving in normal basis:

```
$x_0 \leftarrow$ root selector (an arbitrary element from $\mathbb{F}_3$)
for $i \leftarrow 1$ to $m - 1$ do {
    $x_i \leftarrow x_{i-1} - u_i$
}
$x$ is a solution if, and only if, $x_{m-1} = x_0 + u_0$.
```

□