On compressible pairings and their computation

Michael Naehrig[1⋆], Paulo S. L. M. Barreto[2⋆⋆], and Peter Schwabe[1]

[1] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
{michael,peter}@cryptojedi.org
[2] Escola Politécnica, Universidade de São Paulo.
Av. Prof. Luciano Gualberto, tr. 3, n. 158.
BR 05508-900, São Paulo(SP), Brazil.
pbarreto@larc.usp.br

Abstract. In this paper we provide explicit formulæ to compute bilinear pairings in compressed form. We indicate families of curves where the proposed compressed computation method can be applied and where particularly generalized versions of the Eta and Ate pairings due to Zhao *et al.* are especially efficient. Our approach introduces more flexibility when trading off computation speed and memory requirement. Furthermore, compressed computation of reduced pairings can be done without any finite field inversions. We also give a performance evaluation and compare the new method with conventional pairing algorithms.

Keywords: pairing-based cryptography, compressible pairings, algebraic tori, Tate pairing, Eta pairing, Ate pairing, twists.

1 Introduction

Cryptographically relevant bilinear maps like the Tate and Weil pairing usually take values over an extension field $\mathbb{F}_{p^k}$ of the base field $\mathbb{F}_p$. Pairing inputs are typically points on an elliptic curve defined over $\mathbb{F}_p$. It has been known for a while (see the work of Scott and Barreto [13] and Granger, Page and Stam [7]) that pairing values can be efficiently represented in compressed form by using either traces over subfields or algebraic tori. The former approach leads to a small loss of functionality: the trace of an exponential, $\mathrm{Tr}(g^x)$, can be computed from the trace $\mathrm{Tr}(g)$ and the exponent x alone, but the trace of a product $\mathrm{Tr}(gh)$ cannot be easily computed from $\mathrm{Tr}(g)$ and $\mathrm{Tr}(h)$. The latter approach does not suffer from this drawback, since torus elements can implicitly be multiplied in the compressed representation. With either approach, pairing values can be

⋆ Most of the work presented in this paper was done, while the first author was visiting the Escola Politécnica, Universidade de São Paulo, Brazil. The visit was supported by the German Research Foundation (DFG).

⋆⋆ Supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under grant 312005/2006-7.

efficiently compressed to one half or one third of the original length, depending on the precise setting of the underlying fields and curves.

Our contribution in this paper is to provide explicit formulæ to compute pairings directly in compressed form. Although we do not claim any performance improvement over existing methods, we show that full implementation of arithmetic over $\mathbb{F}_{p^k}$ can be avoided altogether; only operations for manipulating pairing arguments and (compressed) pairing values are needed.

From an implementor's or hardware designer's perspective the contribution of this paper consists of mainly two aspects. Firstly, the explicit formulæ for multiplication and squaring of torus elements give more flexibility in trading off computation speed with memory requirement. The second aspect concerns field inversions during pairing computation. Using projective representation for curve points, inversions can be avoided in the Miller loop. However, a very efficient way to then compute the final exponentiation is to decompose the exponent into three factors and use the Frobenius automorphism to compute powers for two of these factors. This involves an inversion in $\mathbb{F}_{p^k}$, which can be avoided using the compressed representation of pairing values. Hence, we can entirely avoid field inversion during pairing computation and still use fast Frobenius actions in the final exponentiation. From a more theoretical perspective this approach can be seen as a first step to further enhancement of the resulting algorithms, and parallels the case of hyperelliptic curve arithmetic where the introduction of explicit formulæ paved the way to more efficient arithmetic.

We provide timing results for implementations of different pairing algorithms, comparing the newly proposed pairings in compressed form with their conventional counterparts. Additionally, we give examples of curve families amenable to pairing compression where generalized versions of the Eta and Ate pairings due to Zhao *et al.* are more efficient than the non-generalized versions. We provide examples for the three AES security levels 128, 192 and 256 bits. In this paper we use the notion Eta pairing instead of twisted Ate pairing, because it has originally been used in the non-supersingular case as well.

This paper is organized as follows. In Sections 2 and 3 we review mathematical concepts related to pairings and algebraic tori. In Section 4 we discuss torus-based pairing compression and provide explicit formulæ for pairing computation in compressed form. We describe how to avoid inversions in Section 5. In Section 6 implementation costs are given and we conclude in Section 7.

2 Preliminaries on pairings

Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$ of characteristic $p \geq 5$. Let r be a prime divisor of the group order $n = \#E(\mathbb{F}_p)$ and let k be the embedding degree of E with respect to r, i.e. k is the smallest integer such that $r \mid p^k - 1$. We assume that $k > 1$.

Let $\mathbb{F}_q$ be an extension of $\mathbb{F}_p$. An elliptic curve E' over $\mathbb{F}_q$ is called a *twist of degree* d if there exists an isomorphism $\psi_d : E' \to E$ defined over $\mathbb{F}_{q^d}$ and d is minimal with this property. There is a nice summary about twists of elliptic

curves regarding their existence and the possible group orders of $E'(\mathbb{F}_q)$ given by Hess, Smart and Vercauteren in [8].

We consider an r-torsion point $P \in E(\mathbb{F}_p)[r]$ and an independent r-torsion point $Q \in E(\mathbb{F}_{p^k})[r]$. We fix $G_1 = \langle P \rangle \subseteq E(\mathbb{F}_p)[r]$ and $G_2 = \langle Q \rangle \subseteq E(\mathbb{F}_{p^k})[r]$. If the curve has a twist of order d we may choose the point Q arising as $Q = \psi_d(Q')$, where Q' is an $\mathbb{F}_{p^{k/d}}$-rational point of order r on the twist E', see again [8]. Taking this into account we can represent points in $\langle Q \rangle$ by the points in $\langle Q' \rangle \subseteq E'(\mathbb{F}_{p^{k/d}})[r]$. Let t be the trace of Frobenius on $E/\mathbb{F}_p$ and $\lambda = (t-1)^{k/d} \bmod r$. Notice that λ is a primitive d-th root of unity modulo r.

The i-th Miller function $f_{i,P}$ for P is a function with divisor $(f_{i,P}) = i(P) - ([i]P) - (i-1)(\mathcal{O})$. We use Miller functions to compute pairings. Let the function e_s be defined by

$$e_s : G_1 \times G_2 \to \mu_r,\ (P, Q) \mapsto f_{s,P}(Q)^{(p^k-1)/r}.$$

For certain choices of s this function is a non-degenerate bilinear pairing. For $s = r$ we obtain the reduced Tate pairing τ, $s = \lambda$ yields the reduced Eta pairing η and $s = T = t - 1$ leads to the reduced Ate pairing α by switching the arguments. Altogether we have

- Tate pairing: $\tau(P, Q) = f_{r,P}(Q)^{(p^k-1)/r}$,
- Eta pairing: $\eta(P, Q) = f_{\lambda,P}(Q)^{(p^k-1)/r}$,
- Ate pairing: $\alpha(P, Q) = f_{T,Q}(P)^{(p^k-1)/r}$.

To obtain unique values, all pairings are reduced via the final exponentiation by $(p^k - 1)/r$. The Eta pairing was introduced in the supersingular context by Barreto, Galbraith, Ó' hÉigeartaigh and Scott in [1]. The Ate pairing was introduced by Hess, Smart and Vercauteren [8]. Actually the concept of the Eta pairing can be transferred to ordinary curves as well. Hess, Smart and Vercauteren [8] call it the twisted Ate pairing.

Recently much progress has been made in improving the performance of pairing computation. Main achievements have been made by suggesting variants of the above pairings which shorten the loop length in Miller's algorithm, for example so called generalized pairings [15], optimized pairings [11], the R-Ate pairing [10] as well as optimal pairings [14].

As an example we consider the generalized versions of the Eta and Ate pairings by Zhao, Zhang and Huang [15]:

- generalized Eta pairing: $\eta_c(P, Q) = f_{\lambda^c \bmod r,P}(Q)^{(p^k-1)/r}$, $0 < c < k$,
- generalized Ate pairing: $\alpha_c(P, Q) = f_{T^c \bmod r,Q}(P)^{(p^k-1)/r}$, $0 < c < k$.

For a certain choice of c the loop length of the generalized pairings may turn out shorter than the loop length of the original pairing. Notice that if $T^c \equiv -1 \pmod r$ or $\lambda^c \equiv -1 \pmod r$ the loop length is $r - 1$ which is the same as for the Tate pairing and does not give any advantage.

For each of the three AES security levels 128, 192 and 256 bits we give examples of elliptic curve families where generalized pairings lead to a shortening

of the loop length. The examples all have embedding degree divisible by 6 and a twist of degree 6 such that the compressed pairing computations of Sections 4 and 5 can be applied. We stress that for all example families the generalized Eta pairing is more efficient than the Tate pairing. We emphasize the Eta pairing since this goes along with our compression method, but we note that there are versions of the Ate pairing which have a much shorter loop length than the pairings suggested here. For example the curves in Example 1 can be used for an optimal Ate pairing with loop length $\log_2 r/4$ (see Vercauteren [14]).

Example 1. We consider the family of elliptic curves introduced by Barreto and Naehrig in [2]. Let E be an elliptic curve of the family parameterized by $p = 36u^4 + 36u^3 + 24u^2 + 6u + 1$ and $t = 6u^2 + 1$. From the construction it follows that the curve has prime order, i.e. $r = n$, complex multiplication discriminant $D = -3$ and embedding degree $k = 12$. As shown in [2] E admits a twist E' of degree $d = 6$. This also follows from Lemma 4 in Section 4.2. We consider

$$\lambda = (t-1)^{k/d} = (6u^2)^2 \equiv 36u^4 \pmod{n}.$$

Since $n = 36u^4 + 36u^3 + 18u^2 + 6u + 1$ for positive values of u the length of λ is about the same as n, which means that there is no point in using the eta pairing. But for negative u we obtain $\lambda \equiv -36u^3 - 18u^2 - 6u - 1 \pmod{n}$ which is only $3/4$ the size of n. Thus the Eta pairing gets faster than the Tate pairing.

For positive u the generalized version of the Eta pairing suggests to use a different power of λ. For example we could use $\lambda^4 = -\lambda$ since λ is a primitive sixth root of unity. We have $-\lambda \equiv -36u^4 \equiv 36u^3 + 18u^2 + 6u + 1 \pmod{n}$ and the length of $-\lambda$ is as well $3/4$ of that of n which yields a faster pairing than the Tate pairing.

Example 2. A family of curves with embedding degree $k = 18$ was found by Kachisa and is described in Example 6.14 of [6]. For those curves we have $r(u) = u^6+37u^3+343$ and $t(u) = \frac{1}{7}(u^4+16u+7)$. The generalized Ate pairing computing the loop over $T^{12} \equiv u^3+18 \pmod{r}$ for positive u and $T^3 \equiv -u^3-18 \pmod{r}$ for negative u is more efficient than the standard Ate pairing using $T \equiv \frac{1}{7}(u^4+16u)$.

The curves have a sextic twist and can be used for the Eta pairing with a loop over $\lambda = T^3$ which for negative u is as short as the generalized Ate pairing loop. For positive u take T^{12} for the generalized Eta pairing.

Example 3. Recently, Kachisa, Schaefer and Scott [9] found a family of pairing friendly curves with embedding degree $k = 36$. The curves have a sextic twist and lead to shorter loops in pairing computation. The group order is parametrized by a polynomial of degree 12 which we omit for space reasons. The trace of Frobenius is parametrized by the following polynomial of degree 7:

$$\begin{aligned} t = {} & 125339925335955356307102330222u^7 + 875884065532485689314301650 2u^6 \\ & +26231736075114618891078427 8u^5 + 436450441960757801531619 0u^4 \\ & +43570655272439907140970u^3 + 260978358826886222466u^2 \\ & +868445151522065613u + 1238521382045476. \end{aligned}$$

Both the generalized Ate and Eta pairings can be computed with a loop over $T_6 = T^6 \bmod r$ with

$$\begin{aligned} T_6 = {} & 1525930427756943709606797 3u^6 + 913997772652077313277994u^5 \\ & + 228109987087504077455555u^4 + 303628259738257192620u^3 \\ & + 2273330651802144795u^2 + 9077823883505034u + 15103919293237. \end{aligned}$$

For details see [9].

3 Preliminaries on tori

Let $\mathbb{F}_q$ be a finite field and $\mathbb{F}_{q^l} \supseteq \mathbb{F}_q$ a field extension. Then the norm of an element $\alpha \in \mathbb{F}_{q^l}$ with respect to $\mathbb{F}_q$ is defined as the product of all conjugates of α over $\mathbb{F}_q$, namely $N_{\mathbb{F}_{q^l}/\mathbb{F}_q}(\alpha) = \alpha\alpha^q \cdots \alpha^{q^{l-1}} = \alpha^{1+q+\cdots+q^{l-1}} = \alpha^{(q^l-1)/(q-1)}$.

Rubin and Silverberg describe in [12] how algebraic tori can be used in cryptography. We recall the definition of a torus. For a positive integer l define the torus

$$T_l(\mathbb{F}_q) = \bigcap_{\mathbb{F}_q \subseteq F \subsetneq \mathbb{F}_{q^l}} \ker(N_{\mathbb{F}_{q^l}/F}). \tag{1}$$

Thus we have $T_l(\mathbb{F}_q) = \{\alpha \in \mathbb{F}_{q^l} \mid N_{\mathbb{F}_{q^l}/F}(\alpha) = 1,\ \mathbb{F}_q \subseteq F \subsetneq \mathbb{F}_{q^l}\}$. If $\mathbb{F}_q \subseteq F \subsetneq \mathbb{F}_{q^l}$ then $F = \mathbb{F}_{q^d}$ where $d \mid l$ so the relative norm is given as $N_{\mathbb{F}_{q^l}/\mathbb{F}_{q^d}}(\alpha) = \alpha^{(q^l-1)/(q^d-1)}$. The number of elements in the torus is $|T_l(\mathbb{F}_q)| = \Phi_l(q)$, where Φ_l is the l-th cyclotomic polynomial. We know that

$$X^l - 1 = \prod_{d|l} \Phi_d(X) = \Phi_l(X) \prod_{d|l, d\neq l} \Phi_d(X).$$

Thus the torus $T_l(\mathbb{F}_q)$ is the unique subgroup of order $\Phi_l(q)$ of $\mathbb{F}_{q^l}^*$. Set $\Psi_l(X) = \prod_{d|l,d\neq l} \Phi_d(X) = (X^l - 1)/\Phi_l(X)$.

Lemma 1. *Let $\alpha \in \mathbb{F}_{q^l}^*$. Then $\alpha^{\Psi_l(q)} \in T_l(\mathbb{F}_q)$.*

Proof. Let $\beta = \alpha^{\Psi_l(q)}$, then $\beta^{\Phi_l(q)} = \alpha^{q^l-1} = 1$, thus β has order dividing $\Phi_l(q)$ and therefore lies in $T_l(\mathbb{F}_q)$. □

Lemma 2. *For each divisor $d \mid l$ of l it holds $T_l(\mathbb{F}_q) \subseteq T_{l/d}(\mathbb{F}_{q^d})$.*

Proof. Let $\beta \in T_l(\mathbb{F}_q)$. Then $N_{\mathbb{F}_{q^l}/F}(\beta) = 1$ for all fields $\mathbb{F}_q \subseteq F \subsetneq \mathbb{F}_{q^l}$. In particular the norm is 1 for all fields $\mathbb{F}_{q^d} \subseteq F \subsetneq \mathbb{F}_{q^l}$. And so $\beta \in T_{l/d}(\mathbb{F}_{q^d})$. □

Combining the above two Lemmas shows that the element α raised to the power $\Psi_l(q)$ is an element of each torus $T_{l/d}(\mathbb{F}_{q^d})$ for all divisors $d \mid l$, $d \neq k$.

Let E be an elliptic curve defined over $\mathbb{F}_p$ with embedding degree k as in the previous section. By the definition of the embedding degree we have $r \nmid \Phi_d(p)$

for all divisors $d \mid k$, $d \neq k$. From that we see that the final exponent can be split up as

$$\frac{p^k - 1}{r} = \Psi_k(p)\frac{\Phi_k(p)}{r}.$$

This means that pairing values lie in the torus $T_k(\mathbb{F}_p)$ und thus by the preceeding Lemmas in each torus $T_{k/d}(\mathbb{F}_{p^d})$ for $d \mid k$, $d \neq k$.

4 Compressed pairing computation

Scott and Barreto [13] show how to compress the pairing value before the final exponentiation and how to use traces to compute the result. Also the use of tori has been investigated for the final exponentiation and to save bandwidth.

It is already shown by Granger, Page and Stam [7] how a pairing value in a field extension $\mathbb{F}_{q^6}$ can be compressed to an element in $\mathbb{F}_{q^3}$ plus one bit. We note that the technique of compression that we use here has already been explained in [7] for supersingular curves in characteristic 3. Granger, Page and Stam [7] mention that the technique works as well for curves over large characteristic fields. We describe and use the compression in the case of large characteristic and additionally as a new contribution include the compression into the Miller loop to compress the computation itself. In the following section 4.1 we recapitulate the compression for even embedding degree and show how to use it during pairing computation.

To make the paper as self-contained as possible and to enhance better understanding we derive and prove certain facts which are already known in the literature.

4.1 Compression for even embedding degree

Let k be even and let $p \geq 5$ be a prime. In this section let $q = p^{k/2}$ and thus $\mathbb{F}_q = \mathbb{F}_{p^{k/2}}$ such that $\mathbb{F}_{q^2} = \mathbb{F}_{p^k}$. Choose $\xi \in \mathbb{F}_q$ to be a nonsquare. Then the polynomial $X^2 - \xi$ is irreducible and we may represent $\mathbb{F}_{q^2} = \mathbb{F}_q(\sigma)$ where σ is a root of $X^2 - \xi$.

Lemma 3. *Let $\alpha \in \mathbb{F}_{q^2}$. Then α^{q-1} is an element of $T_2(\mathbb{F}_q)$ and can be represented by a single element in $\mathbb{F}_q$ plus one additional bit. This element can be computed by one inversion in $\mathbb{F}_q$.*

Proof. We compute the q-Frobenius of σ which gives $\pi_q(\sigma) = \sigma^q = -\sigma$. The element α can be written as $\alpha = a_0 + a_1\sigma$ with coefficients $a_0, a_1 \in \mathbb{F}_q$. Raising α to the power of $q - 1$ we obtain

$$(a_0 + a_1\sigma)^{q-1} = \frac{(a_0 + a_1\sigma)^q}{a_0 + a_1\sigma} = \frac{a_0 - a_1\sigma}{a_0 + a_1\sigma}.$$

If $a_1 \neq 0$ we can proceed further by dividing in numerator and denominator by a_1 which gives

$$(a_0 + a_1\sigma)^{q-1} = \frac{a_0/a_1 - \sigma}{a_0/a_1 + \sigma} = \frac{a - \sigma}{a + \sigma}. \tag{2}$$

It is clear that the above fraction is an element of $T_2(\mathbb{F}_q)$. It can be represented by $a \in \mathbb{F}_q$ only. But we need an additional bit to represent 1 in the torus. If $a_1 = 0$ we started with an element of the base field and the exponentiation gives 1. In summary α^{q-1} can be represented by just one value in $\mathbb{F}_q$ plus one bit to describe the unit element 1. □

The final exponentiation in the reduced pairing algorithm has to be carried out in the large field $\mathbb{F}_{p^k}$. The idea is to do part of the final exponentiation right inside the Miller loop to move elements to the torus $T_2(\mathbb{F}_q)$. Using torus arithmetic we may compute the compressed pairing value by computations in the torus only using less memory than with full extension field arithmetic. The rest of the final exponentiation can be carried out in the end on the compressed pairing value by also using torus arithmetic only.

Now if we have an elliptic curve with embedding degree k, in the final exponentiation we raise the output of the Miller loop to $(p^k-1)/r = (q^2-1)/r$ where r is the order of the used subgroup. Since the embedding degree is k we have that $r \nmid q-1$. Therefore we may split up the final exponentiation and raise the elements to $q-1$ right away. This can be done in the above described manner by only one $\mathbb{F}_q$ inversion. Since the pairing value is computed multiplicatively we already exponentiate the line functions in the Miller loop by $q-1$ and then carry out multiplications in torus arithmetic.

There is no need to have a representation for 1 in the torus during the pairing computation. The remaining part of the final exponentiation $(q+1)/r$ is even, if q is the power of an odd prime and r is a large prime which thus is also odd. Therefore both values 1 and -1 are mapped to 1 when the final exponentiation is completed. We thus may take the representation for -1 whenever 1 occurs during computation. This will not alter the result of the pairing. Note that the torus element -1 has a regular representation with $a = 0$, since then the fraction (2) assumes the value -1. In this way we can save the bit which is usually needed to represent 1 when working in the torus.

For $\alpha = a_0 + a_1\sigma$ we denote by $\hat{\alpha} \in \mathbb{F}_q$ the torus representation of α^{q-1} for the pairing algorithm, i.e. $\hat{\alpha} = a_0/a_1$ if $a_1 \neq 0$ and $\hat{\alpha} = 0$ if $a_1 = 0$. The latter means we identify 1 and -1. Granger, Page and Stam [7] have demonstrated that arithmetic in the multiplicative group $T_2(\mathbb{F}_q)$ can now be done via

$$\frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma} \cdot \frac{\hat{\beta}-\sigma}{\hat{\beta}+\sigma} = \frac{\widehat{\alpha\beta}-\sigma}{\widehat{\alpha\beta}+\sigma},$$

where

$$\widehat{\alpha\beta} = (\hat{\alpha}\hat{\beta}+\xi)/(\hat{\alpha}+\hat{\beta}) \tag{3}$$

if $\hat{\alpha} \neq -\hat{\beta}$ and $\hat{\alpha} \neq 0$ and $\hat{\beta} \neq 0$. If $\hat{\alpha} = -\hat{\beta}$ the result is simply 1. If one of the values represents 1 we return the other value. For squaring a torus element with $\hat{\alpha} \neq 0$ we compute $\widehat{\alpha^2} = \hat{\alpha}/2 + \xi/(2\hat{\alpha})$.

The representation of the inverse of a torus element given by $\hat{\alpha}$ can be seen to be $-\hat{\alpha}$, since

$$\alpha^{-1} = \left(\frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma}\right)^{-1} = \frac{\hat{\alpha}+\sigma}{\hat{\alpha}-\sigma} = \frac{-\hat{\alpha}-\sigma}{-\hat{\alpha}+\sigma}. \tag{4}$$

We point out that doing inversions in torus representation does not need inversions in a finite field. Instead computation of an inverse only requires negation of a finite field element.

As seen above, we need to compute the result of the Miller loop only up to sign since -1 will be mapped to 1 in the final exponentiation. If we take the negative of a torus element, we obtain

$$-\frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma} = \frac{\sigma^2-\hat{\alpha}\sigma}{\sigma^2+\hat{\alpha}\sigma} = \frac{\xi-\hat{\alpha}\sigma}{\xi+\hat{\alpha}\sigma} = \frac{\xi/\hat{\alpha}-\sigma}{\xi/\hat{\alpha}+\sigma},$$

as long as $\hat{\alpha} \neq 0$. If $\hat{\alpha} = 0$ we are dealing with the element -1 and the negative of it is 1. This computation shows that the negative of a torus element $\alpha \neq \pm 1$ represented by $\hat{\alpha}$ is represented by $\xi/\hat{\alpha}$.

There may be potential to even further compress the computation inside the Miller loop. If it is possible to raise elements to $\Psi_k(p)$ in an efficient way, one may use the norm conditions in other tori to deduce equations which allow to achieve even more compact representations for the field elements used in the pairing computation. We will see in section 4.2 how this works in the special case $k \equiv 0 \pmod 6$.

4.2 Curves with a sextic twist and $6 \mid k$

From now on we assume that $6 \mid k$, i.e. $k = 6m$, where m is an arbitrary positive integer. In this section we fix $q = p^m$. Then $\mathbb{F}_q = \mathbb{F}_{p^m}$ and $\mathbb{F}_{q^6} = \mathbb{F}_{p^k}$. We have a look at the case where we are dealing with an elliptic curve which has complex multiplication discriminant $D = -3$. Under the above assumptions we give the details of our new method to include compression into the Miller loop. The existence of twists of degree 6 leads to compressed values of line functions which can easily be computed by only a few field operations in $\mathbb{F}_q$.

The description of twists and their orders given by Hess, Smart and Vercauteren in [8] yields the following lemma.

Lemma 4. *Let E be an ordinary elliptic curve with CM discriminant $D = -3$. Let E be defined over $\mathbb{F}_q$ where $q \equiv 1 \pmod 6$ and let r be a divisor of the group order $\#E(\mathbb{F}_q)$. The curve E can be represented as $E : y^2 = x^3 + B$, $B \in \mathbb{F}_q$.*

Then there exists a twist E' of degree $d = 6$ which is defined over $\mathbb{F}_q$ and $E'(\mathbb{F}_q)$ has order divisible by r.

The twist is given by $E' : y^2 = x^3 + B/\xi$, where $\xi \in \mathbb{F}_q^*$ is not a square or a third power. A $\mathbb{F}_{q^6}$-isomorphism is given by

$$\psi_d : E' \to E,\ (x, y) \mapsto (\xi^{1/3}x, \xi^{1/2}y). \tag{5}$$

We can represent the field extensions of $\mathbb{F}_q$ contained in $\mathbb{F}_{q^6}$ as $\mathbb{F}_{q^2} = \mathbb{F}_q(\xi^{1/2})$ and $\mathbb{F}_{q^3} = \mathbb{F}_q(\xi^{1/3})$ respectively. We use the twist to compactly represent the second argument of the pairing. This also implies that elliptic curve arithmetic in the group G_2 can be replaced by arithmetic in $E'(\mathbb{F}_q)$.

The twist also gives rise to further improvements for the compressed pairing computation. We consider terms which arise from line functions inside the Miller loop. Let $l_{U,V}(Q)$ be the line function of the line through the points U and V evaluated at Q. In the Miller loop U and V are points in $E(\mathbb{F}_p)$ and $Q = \psi_d(Q')$ for a point $Q' \in E'(\mathbb{F}_q)$ on the twist. These assumptions can not be made when computing the Ate pairing. Let $U = (x_U, y_U)$, $V = (x_V, y_V)$ and $Q' = (x_{Q'}, y_{Q'})$, and thus $Q = (x_Q, y_Q) = (\tau x_{Q'}, \sigma y_{Q'})$ where $\sigma = \xi^{1/2} \in \mathbb{F}_{q^2}$ and $\tau = \xi^{1/3} \in \mathbb{F}_{q^3}$. Notice that $\sigma^q = -\sigma$ and that $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}(\sigma)$. For $U \neq -V$ the line function then yields

$$l_{U,V}(Q) = \lambda(x_Q - x_U) + (y_U - y_Q),$$

where λ is the slope of the line through U and V, i.e. $\lambda = (y_V - y_U)/(x_V - x_U)$ if $U \neq \pm V$ and $\lambda = (3x_U^2)/(2y_U)$ if $U = V$ respectively. In the case $U = -V$ the line function is $l_{U,-U}(Q) = x_Q - x_U$.

We take advantage of the fact that Q arises as $Q = \psi_d(Q')$ for some point $Q' \in E'(\mathbb{F}_q)$ and obtain

$$\begin{aligned} l_{U,V}(Q) &= \lambda(\tau x_{Q'} - x_U) + (y_U - \sigma y_{Q'}) \\ &= (y_U - \lambda x_U + \lambda x_{Q'}\tau) - y_{Q'}\sigma. \end{aligned}$$

For $U = -V$ we have $l_{U,-U}(Q) \in \mathbb{F}_{q^3}$. We thus proved the following lemma.

Lemma 5. *For $U \neq -V$ the torus representation of $(l_{U,V}(Q))^{q^3-1}$ can be computed as $(\lambda x_U - y_U - \lambda x_{Q'}\tau)/y_{Q'} \in \mathbb{F}_{q^3}$.*

Although $(\lambda x_U - y_U - \lambda x_{Q'}\tau)/y_{Q'}$ is an element of $\mathbb{F}_{q^3}$ it is possible to compute it with just a few $\mathbb{F}_q$ computations since λ as well as the coordinates of all involved points are elements of $\mathbb{F}_q$. Note that no exponentiation in $\mathbb{F}_{q^3}$ is required.

Inside the Miller loop we must carry out multiplications and squarings in torus representation. Squarings have to be done with elements represented by full $\mathbb{F}_{q^3}$ elements. But multiplications always include a line function as one factor. Let $\mu = -(y_U - \lambda x_U + \lambda x_{Q'}\tau)$ be the numerator of the representative for the exponentiated line function. If we compute the torus product with $\hat{\alpha}$ an arbitrary $\mathbb{F}_{q^3}$ element and $\hat{\beta} = \mu/y_{Q'}$ we get the following.

$$\widehat{\alpha\beta} = \frac{\hat{\alpha}\hat{\beta} + \xi}{\hat{\alpha} + \hat{\beta}} = \frac{\hat{\alpha}\mu + \xi y_{Q'}}{\hat{\alpha} y_{Q'} + \mu}.$$

There is no need to invert $y_{Q'}$ to compute the corresponding torus representation for $(l_{U,V}(Q))^{q^3-1}$. Instead we directly compute the product representative. Thus there is only one inversion in $\mathbb{F}_{q^3}$ needed to exponentiate the line function and compute the product in the Miller loop.

The final exponentiation is raising to $(q^6-1)/r$ in terms of q. We may write this as

$$\frac{q^6-1}{r} = (q^3-1)(q+1)\frac{q^2-q+1}{r}.$$

What we did up to now is to raise line functions to q^3-1 in order to already move the elements to $T_2(\mathbb{F}_{q^3})$. But when we now do the exponentiation to $q+1$ we have raised the element to $\Psi_6(q)$ and therefore end up with an element in $T_6(\mathbb{F}_q)$. This in particular means that our element lies in the kernel of $N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}$. If we use this property we may compress the element $\hat{\alpha}$ to two $\mathbb{F}_q$ elements which also has been demonstrated similarly by Granger, Page and Stam [7], and compute the pairing using this compact representation.

Proposition 1. *Let $p \equiv 1 \pmod 3$ and $\alpha \in \mathbb{F}_{q^6}^*$. Then $\alpha^{\Psi_6(q)}$ can be uniquely represented by a pair (a_0, a_1) of $\mathbb{F}_q$ elements.*

Proof. As seen before we can represent α^{q^3-1} by $\hat{\alpha}$ as $\alpha^{q^3-1} = \frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma}$. Let

$$\beta = \alpha^{\Psi_6(q)} = \left(\frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma}\right)^{q+1}.$$

We represent β by its torus representative $\hat{\beta}$, which can be computed as follows:

$$\beta = \left(\frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma}\right)^{q} \cdot \frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma} = \frac{\hat{\alpha}^q+\sigma}{\hat{\alpha}^q-\sigma} \cdot \frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma} = \frac{-\hat{\alpha}^q-\sigma}{-\hat{\alpha}^q+\sigma} \cdot \frac{\hat{\alpha}-\sigma}{\hat{\alpha}+\sigma}.$$

If $\hat{\alpha}^q = \hat{\alpha}$ we get $\beta = 1$. Otherwise, using (3) we get $\hat{\beta} = (-\hat{\alpha}^{q+1}+\xi)/(-\hat{\alpha}^q+\hat{\alpha})$. We now make use of the property that α has been raised to $\Psi_6(q)$ and thus lies in the torus $T_6(\mathbb{F}_q)$. We have $N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}(\beta) = 1$, i. e.

$$\left(\frac{\hat{\beta}-\sigma}{\hat{\beta}+\sigma}\right)^{1+q^2+q^4} = 1,$$

which is equivalent to $(\hat{\beta}-\sigma)^{1+q^2+q^4} = (\hat{\beta}+\sigma)^{1+q^2+q^4}$. We write $\hat{\beta} = b_0 + b_1\tau + b_2\tau^2$ with $b_i \in \mathbb{F}_q$ and use the fact that $\tau^q = \zeta^2\tau$ for ζ a primitive third root of unity which lies in $\mathbb{F}_q$ since $q \equiv 1 \pmod 3$. An explicit computation of $(\hat{\beta}\pm\sigma)^{1+q^2+q^4}$ and simplification of the equation $(\hat{\beta}-\sigma)^{1+q^2+q^4} = (\hat{\beta}+\sigma)^{1+q^2+q^4}$ gives the following relation:

$$-3b_1b_2\xi + \xi + 3b_0^2 = 0.$$

This equation can be used to recover b_2 from b_0 and b_1 if $b_1 \neq 0$ as

$$b_2 = \frac{3b_0^2+\xi}{3b_1\xi}. \tag{6}$$

If $b_1 = 0$ we have $\xi = -3b_0^2$. Since $p \equiv 1 \pmod 3$ then -3 is a square modulo p thus ξ is a square which is not true. Therefore b_1 can not be 0 in this case. Summarizing we see that we can represent the element β by b_0 and b_1 only which concludes the proof. □

We now turn our attention again to the line functions $l_{U,V}(Q)$ used in Miller's algorithm.

Proposition 2. *Let $\zeta \in \mathbb{F}_q$ be a primitive third root of unity such that $\tau^q = \zeta^2\tau$. Let $\beta = (l_{U,V}(Q))^{\Psi_6(q)}$ where $Q = \psi_d(Q')$. If $\beta \neq 1$ then β can be uniquely represented by*

$$c_0 = \left(\frac{-\zeta}{1-\zeta^2}y_{Q'}^{-1}\right)(y_U - \lambda x_U),\ c_1 = \left(\frac{\zeta^2}{1-\zeta^2}y_{Q'}^{-1}\right)\lambda x_{Q'}. \tag{7}$$

Proof. In the proof of Proposition 1 we have seen how to compute $\hat{\beta} = (-\hat{\alpha}^{q+1} + \xi)/(-\hat{\alpha}^q + \hat{\alpha})$. For the line function we take $\hat{\alpha} = (\lambda x_U - y_U - \lambda x_{Q'}\tau)/y_{Q'}$ from Lemma 5. We thus obtain

$$-\hat{\alpha}^q = \frac{y_U - \lambda x_U + \lambda x_{Q'}\zeta^2\tau}{y_{Q'}}.$$

Multiplying with $\hat{\alpha}$ yields

$$-\hat{\alpha}^{q+1} = -\frac{1}{y_{Q'}^2}\Big((y_U - \lambda x_U)^2 + (1+\zeta^2)\lambda x_{Q'}(y_U - \lambda x_U)\tau + \lambda^2 x_{Q'}^2\zeta^2\tau^2\Big).$$

We further have

$$-\hat{\alpha}^q + \hat{\alpha} = \frac{\lambda x_{Q'}(\zeta^2 - 1)\tau}{y_{Q'}}$$

and compute

$$\begin{aligned}\hat{\beta} &= \frac{(1+\zeta^2)\lambda x_{Q'}(y_U - \lambda x_U)\xi + \lambda^2 x_{Q'}^2\zeta^2\xi\tau + ((y_U - \lambda x_U)^2 - \xi y_{Q'}^2)\tau^2}{\lambda(1-\zeta^2)x_{Q'}y_{Q'}\xi} \\ &= \frac{1+\zeta^2}{1-\zeta^2}\cdot\frac{y_U - \lambda x_U}{y_{Q'}} + \frac{\zeta^2}{1-\zeta^2}\cdot\frac{\lambda x_{Q'}}{y_{Q'}}\tau + \frac{(y_U - \lambda x_U)^2 - \xi y_{Q'}^2}{\lambda(1-\zeta^2)x_{Q'}y_{Q'}\xi}\tau^2.\end{aligned}$$

Recall that $\tau^3 = \xi$. Taking c_i the coefficient of τ^i we have the property $c_2 = \frac{3c_0^2+\xi}{3c_1\xi}$ and thus c_2 can be computed from c_0 and c_1. □

The input Q is not changed during one pairing computation. Hence, $y_{Q'}^{-1}$ can be computed at the beginning of the pairing computation and we do not need inversions to compute the values of the exponentiated line functions inside the Miller loop.

For squaring and multiplication in the Miller loop we need formulæ to compute with compressed values. Squaring of an element (a_0, a_1) can be done with the following formulæ which can be derived by computing the square of the

corresponding torus elements explicitly and compressing again. Compute

$$\begin{aligned}
r_0 &= a_0^5 + \xi(a_0^3 - 2a_0^2a_1^3) + \xi^2(\tfrac{1}{3}a_0 - a_1^3),\\
r_1 &= a_0^5 + \xi(2a_0^3 - 2a_0^2a_1^3) + \xi^2(a_0 - 2a_1^3),\\
s_0 &= a_0(a_0r_0 + a_1^6\xi^2 + \tfrac{1}{27}\xi^3) - \tfrac{1}{3}a_1^3\xi^3,\\
s_1 &= a_1(a_0r_1 + a_1^6\xi^2 + \tfrac{4}{27}\xi^3),\\
s &= 2(a_0r_0 + a_1^6\xi^2 + \tfrac{1}{27}\xi^3),\\
c_0 &= \frac{s_0}{s},\\
c_1 &= \frac{s_1}{s}.
\end{aligned}$$

Then the square of the $\mathbb{F}_{q^6}$ element represented by (a_0, a_1) is represented by (c_0, c_1). Multiplication can be derived in a similar way. We give formulæ for the computation of the product of two elements given by (a_0, a_1) and (b_0, b_1) in compressed form.

$$\begin{aligned}
r_0 &= a_0^2 + \tfrac{1}{3}\xi,\\
r_1 &= b_0^2 + \tfrac{1}{3}\xi,\\
s_0 &= \xi(a_1b_1(a_0b_0 + \xi) + a_1^2r_1 + b_1^2r_0),\\
s_1 &= a_1b_1\xi(a_0b_1 + a_1b_0) + r_0r_1,\\
s_2 &= a_1^2b_1^2\xi + a_0a_1r_1 + b_0b_1r_0,\\
t_0 &= a_1b_1\xi(a_0 + b_0),\\
t_1 &= a_1b_1\xi(a_1 + b_1),\\
t_2 &= b_1r_0 + a_1r_1,\\
u &= t_0^3 + t_1^3\xi + t_2^3\xi^2 - 3\xi t_0t_1t_2,\\
u_0 &= t_0^2 - t_1t_2\xi,\\
u_1 &= t_2^2\xi - t_0t_1,\\
u_2 &= t_1^2 - t_0t_2,\\
v_0 &= s_0u_0 + s_1u_2\xi + s_2u_1\xi,\\
v_1 &= s_0u_1 + s_1u_0 + s_2u_2\xi,\\
c_0 &= \frac{v_0}{u},\\
c_1 &= \frac{v_1}{u}.
\end{aligned}$$

The product is then represented by (c_0, c_1).

5 Dealing with field inversions

In this section we use the assumptions from section 4.2, i. e. $q = p^m$. For compressed computation we need to do inversions during our computations. This

is usually unpleasant, because inversions are very expensive. First of all, one can replace inversion of an element a in $\mathbb{F}_{p^m}$ by an inversion in $\mathbb{F}_p$ and at most $\lfloor \lg m \rfloor + 1$ multiplications in $\mathbb{F}_{p^m}$ by

$$\frac{1}{a} = \frac{a^{p+p^2+\cdots+p^{m-1}}}{N_{\mathbb{F}_{p^m}/\mathbb{F}_p}(a)}.$$

The term in the numerator can be computed by addition chain like methods. For a description of this method see section 11.3.4 in [5].

5.1 Avoid inversions by storing one more $\mathbb{F}_q$ element

The above squaring and multiplication formulæ for compressed computation include an inversion in $\mathbb{F}_q$. We may avoid to do the inversions in each step by additionally storing the denominator and homogenizing the formulae. This means we represent compressed elements in a projective space. At the cost of providing memory space for one more $\mathbb{F}_q$ element and some additional multiplications we get rid of all inversions during the Miller loop. For the compressed line functions computed in Proposition 2 this means that we do not store (c_0, c_1) given by equations (7) but instead we store (C_0, C_1, C), where

$$C_0 = \left(\frac{-\zeta}{1-\zeta^2}\right)(\nu y_U - \mu x_U),\ C_1 = \left(\frac{\zeta^2}{1-\zeta^2}\right)\mu x_{Q'},\ C = \nu y_{Q'}. \tag{8}$$

Here $\mu, \nu \in \mathbb{F}_p$ are the numerator and denominator of the slope λ of the line function, i.e. $\lambda = \mu/\nu$. Notice that μ and ν are elements of $\mathbb{F}_p$ since they arise from points in $E(\mathbb{F}_p)$ (when the pairing we compute is the Tate or Eta pairing).

5.2 Storing only one more $\mathbb{F}_p$ element

When $m > 1$ we are able to compress further, by using the method described at the beginning of Section 5. The denominator C which has to be stored in a third coordinate can be replaced by a denominator which is an element in $\mathbb{F}_p$, namely the norm $N_{\mathbb{F}_{p^m}/\mathbb{F}_p}(C)$ of the previous denominator in $\mathbb{F}_q$. We only need to multiply the other two coordinates by $C^{p+p^2+\cdots+p^{m-1}}$.

In this way it is possible to avoid inversions during pairing computation. Taking into account that inversion of torus elements can be done by negating the representative, we also do not need finite field inversions for the final exponentiation. Normally an inversion is needed to efficiently implement the exponentiation by using the Frobenius automorphism. Furthermore, the cheap inversion of torus elements makes it possible to use windowing methods for Miller loop computations without any field inversions. This is particularly interesting if the loop scalar can not be chosen to be sparse.

We give an example of the compressed squaring and multiplication formulæ for embedding degree $k = 12$.

Example 4. For embedding degree 12 we have $q = p^2$. Let $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$ and $i^2 = -z$ for some element $z \in \mathbb{F}_p$. Let (A_0, A_1, A) be an element in compressed form, i.e. $A_0, A_1 \in \mathbb{F}_{p^2}$ and $A \in \mathbb{F}_p$. Squarings and Multiplications can be computed using the following formulæ.

Squaring: We can compute the square (C_0, C_1, C) as follows.

$$\begin{aligned}
R_0 &= A_0^5 + \xi(A_0^3A^2 - 2A_0^2A_1^3) + \xi^2(\tfrac{1}{3}A_0A^4 - A_1^3A^2),\\
R_1 &= A_0^5 + 2\xi(A_0^3A^2 - A_0^2A_1^3) + \xi^2(A_0A^4 - 2A_1^3A^2),\\
S_0 &= A_0(A_0R_0 + A_1^6\xi^2 + \tfrac{1}{27}A^6\xi^3) - \tfrac{1}{3}A_1^3A^4\xi^3,\\
S_1 &= A_1(A_0R_1 + A_1^6\xi^2 + \tfrac{4}{27}A^6\xi^3),\\
S &= 2A(A_0R_0 + A_1^6\xi^2 + \tfrac{1}{27}A^6\xi^3).
\end{aligned}$$

Write $S = s_0 + is_1$ with $s_0, s_1 \in \mathbb{F}_p$. Then the square is given by

$$\begin{aligned}
C_0 &= S_0(s_0 - is_1),\\
C_1 &= S_1(s_0 - is_1),\\
C &= s_0^2 + zs_1^2.
\end{aligned}$$

Multiplication: To multiply two compressed elements (A_0, A_1, A) and (B_0, B_1, B) we have to use the following formulæ.

$$\begin{aligned}
R_0 &= A_0^2 + \tfrac{1}{3}A^2\xi,\\
R_1 &= B_0^2 + \tfrac{1}{3}B^2\xi,\\
S_0 &= \xi(A_1B_1(A_0B_0 + \xi AB) + A_1^2R_1 + B_1^2R_0),\\
S_1 &= A_1B_1\xi(A_0B_1 + A_1B_0) + R_0R_1,\\
S_2 &= A_1^2B_1^2\xi + A_0A_1R_1 + B_0B_1R_0,\\
T_0 &= A_1B_1\xi(A_0B + B_0A),\\
T_1 &= A_1B_1\xi(A_1B + B_1A),\\
T_2 &= B_1BR_0 + A_1AR_1,\\
T &= T_0^3 + T_1^3\xi + T_2^3\xi^2 - 3\xi T_0T_1T_2,\\
U_0 &= T_0^2 - T_1T_2\xi,\\
U_1 &= T_2^2\xi - T_0T_1,\\
U_2 &= T_1^2 - T_0T_2,\\
V_0 &= S_0U_0 + S_1U_2\xi + S_2U_1\xi,\\
V_1 &= S_0U_1 + S_1U_0 + S_2U_2\xi.
\end{aligned}$$

Write $T = t_0 + it_1$ where $t_0, t_1 \in \mathbb{F}_p$. Then the product (C_0, C_1, C) of the two elements is given by

$$\begin{aligned} C_0 &= V_0(t_0 - it_1), \\ C_1 &= V_1(t_0 - it_1), \\ C &= t_0^2 + zt_1^2. \end{aligned}$$

For an implementation of a pairing algorithm in compressed form without inversions one can use (8) to compute the evaluated compressed line functions and then use the above formulæ for squaring and multiplication in Miller's algorithm. The remaining part of the exponent for the final exponentiation is $(p^4 - p^2 + 1)/n$. The final pairing value can be computed by use of the Frobenius and a square and multiply algorithm with the above squaring and multiplication formulæ (see Devigili, Scott and Dahab [4]). Pseudocode of the above squaring and multiplication algorithms is given in Appendix A.

6 Performance evaluation

In order to evaluate the performance of the compressed pairing computation, we implemented several pairing algorithms in C. For all these implementations[3] we used the curve $E : y^2 = x^3 + b$ over $\mathbb{F}_p$ with parameters described in Table 1 which belongs to the family in Example 1. It has been constructed using the method of Barreto and Naehrig described in [2]. This curve has also been used for the performance evaluation of pairing algorithms by Devegili, Scott and Dahab in [4]. For a fair comparison we implemented pairing algorithms with $\mathbb{F}_{p^{12}}$ constructed as a quadratic extension on top of a cubic extension which is again built on top of a quadratic extension, as described in [4] and by Devigili, Scott, Ó' hÉigeartaigh and Dahab in [3]. For Ate, generalized Eta and Tate pairings we thus achieve similar timings as [4]. We do not use windowing methods since the curve parameters are chosen to be sparse. The final exponentiation for the non-compressed pairings uses the decomposition of the exponent $(p^k - 1)/n$ into the factors $(p^6 - 1)$, $(p^2 + 1)$ and $(p^4 - p^2 + 1)/n$.
In the Miller loop we entirely avoided to do field inversions, by computing the elliptic curve operations in Jacobian coordinates and by using the compressed representation and storing denominators separately as described in Subsection 5.2. For multiplication and squaring of torus elements we use the algorithms given in Appendix A. The figures in table 2 indicate that, depending on the machine architecture, compressed pairing computation is about 20-45% slower than standard pairing computation, if both computations are optimized for computation speed rather than memory usage.
Performance was measured on a 2.2 GHz Intel Core 2 Duo (T7500), a 2.4 GHz Intel Pentium IV (Northwood) and an AMD Athlon XP 2600+ running on 1.9 GHz. The CPU cycles required for Miller loop and final exponentiation respectively are given in Table 2.

[3] The code for our implementation can be found at `http://www.cryptojedi.org/downloads/`

p	824340166543006797212173535031900388365717818113862289211673224128190294931 83
n	824340166543006797212173535031900388362846685642966864301145100525564013737 69
bitsize	256
t	287113247089542491052812360262628119415
k	12
λ^c	$(t-1)^8 \mod n$

Table 1. Parameters of the curve used in our implementation

	Core 2 Duo	**Pentium IV**	**Athlon XP**
Ate	16,750,000	50,400,000	38,000,000
	13,000,000	38,600,000	29,300,000
Generalized Eta	22,370,000	67,400,000	51,700,000
	13,000,000	38,600,000	29,300,000
Tate	30,300,000	90,500,000	69,500,000
	13,000,000	38,600,000	29,300,000
Compressed generalized Eta	31,000,000	107,000,000	84,900,000
	11,700,000	40,300,000	30,900,000
Compressed Tate	41,400,000	146,000,000	115,000,000
	11,700,000	40,300,000	30,900,000

Table 2. Rounded average results of measurements on various CPUs. The upper number describes cycles needed for the Miller loop, the lower number cycles needed for final exponentiation

7 Conclusion

We have described explicit formulæ for pairing computation in compressed form for the Tate and Eta pairings. For different AES security levels we have also indicated families of curves amenable to pairing compression where generalized versions of the Eta and Ate pairings are very efficient. Our implementations and cost measurements show that the pairing algorithms in compressed form are on certain platforms only about 20% slower than the conventional algorithms. The algorithms in compressed form have the advantage that they can be implemented without finite field inversions. This is not only an advantage for pairing computations on restricted devices, but also favors implementation of inversion-free windowing methods for the Miller loop. Furthermore compressed pairing computation gives more flexibility in trading off computation speed versus memory requirement.

Neither the algorithms nor the curve families considered herein are exhaustive; we thus hope that these are the first steps toward further algorithmic enhancements for compressed pairings and towards new, efficient constructions of compressible pairing-friendly curves.

Acknowledgments

We thank the anonymous referees and are grateful to Tanja Lange, Mike Scott, Steven Galbraith and Rob Granger for their valuable comments on earlier versions of this work. The first and third authors did portions of this work at the Institute for Theoretical Information Technology, RWTH Aachen University.

References

1. P. S. L. M. Barreto, S. D. Galbraith, C. Ó' hÉigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Designs, Codes and Cryptography*, 42(3):239–271, 2007.
2. P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography – SAC'2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer-Verlag, 2006.
3. A. J. Devegili, C. Ó hÉigeartaigh, M. Scott, and R. Dahab. Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report 2006/471, 2006. `http://eprint.iacr.org/`.
4. A. J. Devegili, M. Scott, and R. Dahab. Implementing cryptographic pairings over barreto-naehrig curves. Cryptology ePrint Archive, Report 2007/390, 2007. Available from `http://eprint.iacr.org/2007/390`.
5. C. Doche. Finite field arithmetic. In H. Cohen and G. Frey, editors, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, chapter 11, pages 201–238. CRC Press, 2005.
6. D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372, 2006. Available from `http://eprint.iacr.org/2006/372`.
7. R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing based cryptography. *LMS Journal of Computation and Mathematics*, 9:64–85, March 2006.
8. F. Hess, N. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, October 2006.
9. E. J. Kachisa, E. F. Schaefer, and M. Scott. Constructing brezing-weng pairing friendly elliptic curves using elements in the cyclotomic field. Cryptology ePrint Archive, Report 2007/452, 2007. `http://eprint.iacr.org/`.
10. E. Lee, H. Lee, and C. Park. Efficient and generalized pairing computation on abelian varieties. Cryptology ePrint Archive, Report 2008/040, 2008. `http://eprint.iacr.org/`.
11. S. Matsuda, N. Kanayama, F. Hess, and E. Okamoto. Optimised versions of the ate and twisted ate pairings. In Steven D. Galbraith, editor, *IMA Int. Conf.*, volume 4887 of *Lecture Notes in Computer Science*, pages 302–312. Springer, 2007.
12. K. Rubin and A. Silverberg. Torus-based cryptography. In *Advances in Cryptology – Crypto'2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 349–365. Springer-Verlag, 2003.
13. M. Scott and P. S. L. M. Barreto. Compressed pairings. In *Advances in Cryptology – Crypto'2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156, Santa Barbara, USA, 2004. Springer-Verlag.
14. F. Vercauteren. Optimal pairings. Cryptology ePrint Archive, Report 2008/096, 2008. `http://eprint.iacr.org/`.

15. C. Zhao, F. Zhang, and J. Huang. A note on the ate pairing. Cryptology ePrint Archive, Report 2007/247, 2007. Available from `http://eprint.iacr.org/2007/247`.

A Compressed multiplication and squaring algorithms

Algorithm 1 Squaring of the element (A_0, A_1, A)

Require: $(A_0, A_1, A) \in \mathbb{F}_{p^2} \times \mathbb{F}_{p^2} \setminus \{0\} \times \mathbb{F}_p$
Ensure: $(C_0, C_1, C) = (A_0, A_1, A)^2$

$r_1 \leftarrow A_0^2$, $r_2 \leftarrow A_0 r_1$, $S_0 \leftarrow r_1 r_2$, $t_0 \leftarrow A^2$, $r_4 \leftarrow r_2 t_0$, $r_5 \leftarrow A_1^2$, $r_5 \leftarrow A_1 r_5$,
$r_3 \leftarrow r_1 r_5$, $r_4 \leftarrow r_4 - r_3$, $r_0 \leftarrow r_4 \xi$, $r_0 \leftarrow 2r_0$, $S_1 \leftarrow S_0 + r_0$, $r_4 \leftarrow r_4 - r_3$, $r_4 \leftarrow r_4 \xi$,
$S_0 \leftarrow S_0 + r_4$, $t_1 \leftarrow t_0^2$, $r_4 \leftarrow t_1 A_0$, $r_0 \leftarrow \frac{1}{3} r_4$, $r_1 \leftarrow r_5 t_0$, $r_0 \leftarrow r_0 - r_1$, $r_1 \leftarrow 2r_1$,
$r_4 \leftarrow r_4 - r_1$, $r_0 \leftarrow \xi^2 r_0$, $r_4 \leftarrow \xi^2 r_4$, $S_0 \leftarrow S_0 + r_0$, $S_0 \leftarrow S_0 A_0$, $S_1 \leftarrow S_1 + r_4$,
$S_1 \leftarrow S_1 A_0$, $r_2 \leftarrow r_5^2$, $r_2 \leftarrow r_2 \xi^2$, $r_4 \leftarrow t_1 t_0$, $r_4 \leftarrow \frac{1}{27} \xi^3 r_4$, $r_1 \leftarrow r_2 + r_4$, $S_0 \leftarrow S_0 + r_1$,
$S \leftarrow S_0 A$, $S_0 \leftarrow S_0 A_0$, $S \leftarrow 2S$, $r_4 \leftarrow 4r_4$, $r_1 \leftarrow r_2 + r_4$, $S_1 \leftarrow S_1 + r_1$, $S_1 \leftarrow S_1 A_1$,
$r_1 \leftarrow r_5 t_1$, $r_1 \leftarrow \frac{1}{3} \xi^3 r_1$, $S_0 \leftarrow S_0 - r_1$
Write $S = s_0 + i s_1$
$r_1 \leftarrow (s_0 - i s_1)$, $C_0 \leftarrow S_0 r_1$, $C_1 \leftarrow S_1 r_1$, $C \leftarrow S r_1 = s_0^2 + c s_1^2$
return (C_0, C_1, C)

Algorithm 2 Multiplication of elements (A_0, A_1, A) and (B_0, B_1, B)

Require: $(A_0, A_1, A), (B_0, B_1, B) \in \mathbb{F}_{p^2} \times \mathbb{F}_{p^2} \setminus \{0\} \times \mathbb{F}_p$
Ensure: $(C_0, C_1, C) = (A_0, A_1, A) \cdot (B_0, B_1, B)$

$R_0 \leftarrow A_0^2$, $t_1 \leftarrow A^2$, $r_3 \leftarrow \frac{1}{3} \xi t_1$, $R_0 \leftarrow R_0 + r_3$,
$R_1 \leftarrow B_0^2$, $t_1 \leftarrow B^2$, $r_3 \leftarrow \frac{1}{3} \xi t_1$, $R_1 \leftarrow R_1 + r_3$

$r_3 \leftarrow A_1 B_1$, $r_4 \leftarrow A_0 B_0$, $t_1 \leftarrow AB$, $r_5 \leftarrow t_1 \xi$, $r_4 \leftarrow r_4 + r_5$, $S_0 \leftarrow r_3 r_4$, $S_2 \leftarrow r_3^2$
$S_2 \leftarrow S_2 \xi$, $r_4 \leftarrow A_0 B_1$, $r_5 \leftarrow A_1 B_0$, $r_4 \leftarrow r_4 + r_5$, $r_6 \leftarrow r_3 \xi$, $S_1 \leftarrow r_6 r_4$, $r_4 \leftarrow R_0 R_1$
$S_1 \leftarrow S_1 + r_4$, $r_4 \leftarrow A_1 R_1$, $r_5 \leftarrow r_4 A_0$, $S_2 \leftarrow S_2 + r_5$, $T_2 \leftarrow r_4 A$, $r_4 \leftarrow r_4 A_1$,
$S_0 \leftarrow S_0 + r_4$, $r_4 \leftarrow B_1 R_0$, $r_5 \leftarrow r_4 B$, $T_2 \leftarrow T_2 + r_5$, $r_5 \leftarrow r_4 B_0$, $S_2 \leftarrow S_2 + r_5$,
$r_4 \leftarrow r_4 B_1$, $S_0 \leftarrow S_0 + r_4$, $S_0 \leftarrow S_0 \xi$, $T_0 \leftarrow A_0 B$, $r_4 \leftarrow B_0 A$, $T_0 \leftarrow T_0 + r_4$, $T_0 \leftarrow r_6 T_0$,
$T_1 \leftarrow A_1 B$, $r_4 \leftarrow B_1 A$, $T_1 \leftarrow T_1 + r_4$, $T_1 \leftarrow T_1 r_6$

$r_0 \leftarrow T_0^2$, $r_1 \leftarrow T_1^2$, $r_2 \leftarrow T_2^2$, $T \leftarrow r_0 T_0$, $r_3 \leftarrow r_1 T_1$, $r_3 \leftarrow r_3 \xi$, $T \leftarrow T + r_3$
$r_3 \leftarrow r_2 T_2$, $r_3 \leftarrow r_3 \xi^2$, $T \leftarrow T + r_3$, $r_3 \leftarrow T_1 T_2$, $r_3 \leftarrow r_3 \xi$, $U_0 \leftarrow r_0 - r_3$, $r_3 \leftarrow r_3 T_0$
$r_3 \leftarrow 3r_3$, $T \leftarrow T - r_3$, $r_3 \leftarrow T_0 T_1$, $U_1 \leftarrow r_2 \xi$, $U_1 \leftarrow U_1 - r_3$, $r_3 \leftarrow T_0 T_2$, $U_2 \leftarrow r_1 - r_3$
$V_0 \leftarrow S_0 U_0$, $r_0 \leftarrow S_1 U_2$, $r_1 \leftarrow S_2 U_1$, $r_0 \leftarrow r_0 + r_1$, $r_0 \leftarrow r_0 \xi$, $V_0 \leftarrow V_0 + r_0$
$V_1 \leftarrow S_0 U_1$, $r_0 \leftarrow S_1 U_0$, $V_1 \leftarrow V_1 + r_0$, $r_0 \leftarrow S_2 U_2$, $r_0 \leftarrow r_0 \xi$, $V_1 \leftarrow V_1 r_0$
Write $T = t_0 + i t_1$
$r_1 \leftarrow (t_0 - i t_1)$, $C_0 \leftarrow V_0 r_1$, $C_1 \leftarrow V_1 r_1$, $C \leftarrow S r_1 = t_0^2 + c t_1^2$
return (C_0, C_1, C)