A Comparison Between Hardware Accelerators for the Modified Tate Pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$

Jean-Luc Beuchat[1], Nicolas Brisebarre[2], Jérémie Detrey[3], Eiji Okamoto[1], and Francisco Rodríguez-Henríquez[4]

[1] Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
[2] LIP/Arénaire (CNRS – ENS Lyon – INRIA – UCBL), ENS Lyon, 46, allée d'Italie, F-69364 Lyon Cedex 07, France
[3] Cosec group, B-IT, Dahlmannstraße 2, D-53113 Bonn, Germany
[4] Computer Science Section, Electrical Engineering Department, Centro de Investigación y de Estudios Avanzados del IPN, Av. Instituto Politécnico Nacional No. 2508, 07300 México City, México

Abstract. In this article we propose a study of the modified Tate pairing in characteristics two and three. Starting from the η_T pairing introduced by Barreto *et al.* [1], we detail various algorithmic improvements in the case of characteristic two. As far as characteristic three is concerned, we refer to the survey by Beuchat *et al.* [4]. We then show how to get back to the modified Tate pairing at almost no extra cost. Finally, we explore the trade-offs involved in the hardware implementation of this pairing for both characteristics two and three. From our experiments, characteristic three appears to have a slight advantage over characteristic two.

Keywords: modified Tate pairing, reduced η_T pairing, finite field arithmetic, elliptic curve, hardware accelerator, FPGA.

1 Introduction

Over the past few years, bilinear pairings over elliptic and hyperelliptic curves have been the focus of an ever increasing attention in cryptology. Since their introduction to this domain by Menezes, Okamoto & Vanstone [23] and Frey & Rück [8], and the first discovery of their constructive properties by Mitsunari, Sakai & Kasahara [26], Sakai, Oghishi & Kasahara [31], and Joux [16], a large number of pairing-based cryptographic protocols have already been published. For those reasons, efficient computation of pairings is crucial and, according to the recommendations of [11, 21], the Tate pairing appears to be the most appropriate choice.

Miller [24, 25] proposed in 1986 the first algorithm for iteratively computing the Weil and Tate pairings. In the case of the Tate pairing, a further final exponentiation of the Miller's algorithm result is required to obtain a uniquely defined value. Various improvements were published in [2, 6, 9, 22] and we will consider in this paper the modified Tate pairing as defined in [2]. Generalizing some results by Duursma & Lee [6], Barreto *et al.* then introduced the η_T pairing [1], in which the number of iterations in Miller's algorithm is halved. This nondegenerate bilinear pairing can also be used as a tool for computing the modified Tate pairing, at the expense of an additional exponentiation.

General purpose microprocessors are intrinsically not suited for computations on finite fields of small characteristic, hence software implementations are bound to be quite slow and the need for special purpose hardware coprocessors is strong [4, 5, 10, 15, 17, 19, 20, 28–30, 33]. In this context, we extend here to the characteristic two the results by Beuchat *et al.* [4] in the case of the hardware implementation of the reduced η_T pairing in characteristic three.

In Section 2, we detail the algorithms required to compute the reduced η_T pairing in characteristic two. Some algorithmic improvements in both the pairing computation and the tower-field arithmetic are also presented, and an accurate cost analysis in terms of operations over the base field $\mathbb{F}_{2^m}$ is given. We then study in Section 3 the relation between the η_T and Tate pairings, and show that the modified Tate pairing can be computed from the reduced η_T pairing at almost no extra cost in characteristics two and three. Section 4 gives hardware implementation results of the modified Tate pairing in both characteristics and for various field extension degrees. Comparisons between $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$ are presented at equivalent levels of security and they show a slight advantage in favor of characteristic three. Finally, some comparisons with already published solutions are also given to attest the meaningfulness of our results.

2 Computation of the Reduced η_T Pairing in Characteristic Two

2.1 Preliminary Definitions

We consider the supersingular curve E over $\mathbb{F}_{2^m}$ defined by the equation

$$y^2 + y = x^3 + x + b, \tag{1}$$

where $b \in \{0, 1\}$ and m is odd. We define $\delta = b$ when $m \equiv 1,\ 7 \pmod 8$; in all other cases, $\delta = 1 - b$. The number of rational points of E over $\mathbb{F}_{2^m}$ is given by $N = \#E(\mathbb{F}_{2^m}) = 2^m + 1 + \nu 2^{(m+1)/2}$, where $\nu = (-1)^\delta$. The embedding degree of this curve, which is the least positive integer k such that N divides $2^{km} - 1$, is 4.

Choosing $T = 2^m - N$ and a prime ℓ dividing N, Barreto *et al.* [1] defined the η_T pairing of two points P and $Q \in E(\mathbb{F}_{2^m})[\ell]$ as:

$$\eta_T(P, Q) = f_{T', P'}(\psi(Q)),$$

where $T' = -\nu T$, $P' = [-\nu]P$, and $E(\mathbb{F}_{2^m})[\ell]$ denotes the ℓ-torsion subgroup of $E(\mathbb{F}_{2^m})$. ψ is a distortion map from $E(\mathbb{F}_{2^m})[\ell]$ to $E(\mathbb{F}_{2^{4m}})[\ell]$ defined as $\psi(x, y) = (x + s^2, y + sx + t)$, for all $(x, y) \in E(\mathbb{F}_{2^m})[\ell]$ [1]. s and t are elements of $\mathbb{F}_{2^{4m}}$

satisfying $s^2 = s + 1$ and $t^2 = t + s$. This allows for representing $\mathbb{F}_{2^{4m}}$ as an extension of $\mathbb{F}_{2^m}$ using the basis $(1, s, t, st)$: $\mathbb{F}_{2^{4m}} = \mathbb{F}_{2^m}[s, t] \cong \mathbb{F}_{2^m}[X, Y]/(X^2 + X + 1, Y^2 + Y + X)$. Finally, $f_{n,P}$, for $n \in \mathbb{N}$ and $P \in E(\mathbb{F}_{2^m})[\ell]$, is a rational function defined over $E(\mathbb{F}_{2^{4m}})[\ell]$ with divisor $(f_{n,P}) = n(P) - ([n]P) - (n-1)(\mathcal{O})$. In our case, we have

$$\begin{aligned} f_{T',P'} : E(\mathbb{F}_{2^{4m}})[\ell] &\longrightarrow \mathbb{F}^*_{2^{4m}} \\ \psi(Q) &\longmapsto \left(\prod_{i=0}^{\frac{m-1}{2}} g_{[2^i]P'}(\psi(Q))^{2^{\frac{m-1}{2}-i}} \right) l_{P'}(\psi(Q)), \end{aligned} \tag{2}$$

where:

- The point doubling formula is given by

  $$\left[2^i\right] P' = \left(x_{P'}^{2^{2i}} + i, y_{P'}^{2^{2i}} + i x_{P'}^{2^{2i}} + \tau(i) \right),$$

  with

  $$\tau(i) = \begin{cases} 0 & \text{if } i \equiv 0, 1 \pmod 4, \\ 1 & \text{otherwise.} \end{cases}$$

- g_V, for all $V = (x_V, y_V) \in E(\mathbb{F}_{2^m})[\ell]$, is the rational function defined over $E(\mathbb{F}_{2^{4m}})[\ell]$ corresponding to the doubling of V. For all $(x, y) \in E(\mathbb{F}_{2^{4m}})[\ell]$, we have $g_V(x, y) = (x_V^2 + 1)(x_V + x) + y_V + y$ [1]. According to the equation of the elliptic curve (Equation (1)), $x_V^3 + x_V + y_V = y_V^2 + b$ and we obtain [33]:

  $$g_V(x, y) = x(x_V^2 + 1) + y_V^2 + y + b. \tag{3}$$

  We considered both forms of $g_V(x, y)$ when studying η_T pairing algorithms over $\mathbb{F}_{2^m}$ and discovered that the second one always leads to the fastest algorithms.
- l_V, for all $V = (x_V, y_V) \in E(\mathbb{F}_{2^m})[\ell]$, is the equation of the line corresponding to the addition of $\left[2^{\frac{m+1}{2}}\right] V$ with $[\nu]V$, and defined for all $(x, y) \in E(\mathbb{F}_{2^{4m}})[\ell]$ as follows:

  $$\begin{aligned} l_V(x, y) = x_V^2 + (x_V + \alpha)(x + \alpha) + x + y_V + y + \delta + 1 + \\ (x_V + x + 1 - \alpha)s + t, \end{aligned} \tag{4}$$

  where

  $$\alpha = \begin{cases} 0 & \text{if } m \equiv 3, 7 \pmod 8, \\ 1 & \text{if } m \equiv 1, 5 \pmod 8. \end{cases}$$

2.2 Computation of the η_T Pairing in Characteristic Two

Barreto *et al.* suggested reversing the loop to compute the η_T pairing [1]. They introduced the new index $j = \frac{m-1}{2} - i$ and obtained

$$f_{T',P'}(\psi(Q)) = l_{P'}(\psi(Q)) \prod_{j=0}^{\frac{m-1}{2}} \left(g_{\left[2^{\frac{m-1}{2}-j}\right]P'}(\psi(Q)) \right)^{2^j}.$$

A tedious case-by-case analysis allows one to prove that:

$$\left(g_{\left[2^{\frac{m-1}{2}-j}\right]P'}(\psi(Q))\right)^{2^j} = (x_{P'}^{2^{-j}} + \alpha)\cdot(x_Q^{2^j} + \alpha) + y_{P'}^{2^{-j}} + y_Q^{2^j} + \beta + (x_{P'}^{2^{-j}} + x_Q^{2^j} + \alpha)s + t,$$

where

$$\beta = \begin{cases} b & \text{if } m \equiv 1,3 \pmod 8, \\ 1-b & \text{if } m \equiv 5,7 \pmod 8. \end{cases}$$

This equation differs from the one given by Barreto *et al.* [1]: taking advantage of the second form of g_V (Equation 3), we obtain a slight reduction in the number of additions over $\mathbb{F}_{2^m}$.

We suggest a second improvement to save a multiplication over $\mathbb{F}_{2^m}$. At first glance multiplying $l_{P'}(\psi(Q))$ by $g_{\left[2^{\frac{m-1}{2}}\right]P'}(\psi(Q))$ involves three multiplications over $\mathbb{F}_{2^m}$. However, when $j = 0$, we have:

$$g_{\left[2^{\frac{m-1}{2}}\right]P'}(\psi(Q)) = (x_{P'} + \alpha)(x_Q + \alpha) + y_{P'} + y_Q + \beta + (x_{P'} + x_Q + \alpha)s + t.$$

Seeing that $\alpha + \beta = \delta + 1$, we rewrite $l_{P'}(\psi(Q))$ as follows:

$$l_{P'}(\psi(Q)) = g_{\left[2^{\frac{m-1}{2}}\right]P'}(\psi(Q)) + x_{P'}^2 + x_Q + \alpha + s.$$

Defining $g_0 = (x_{P'} + \alpha)(x_Q + \alpha) + y_{P'} + y_Q + \beta$, $g_1 = x_{P'} + x_Q + \alpha$, and $g_2 = x_{P'}^2 + x_Q + \alpha$, we eventually obtain:

$$g_{\left[2^{\frac{m-1}{2}}\right]P'}(\psi(Q)) = g_0 + g_1 s + t \quad \text{and} \quad l_{P'}(\psi(Q)) = (g_0 + g_2) + (g_1 + 1)s + t.$$

The product $l_{P'}(\psi(Q)) \cdot g_{\left[2^{\frac{m-1}{2}}\right]P'}(\psi(Q))$ can be computed by means of two multiplications over $\mathbb{F}_{2^m}$ (see Appendix D.2). Algorithm 1 describes the computation of the η_T pairing according to this construction. Addition over $\mathbb{F}_{2^m}$ involves m bitwise exclusive-OR operations that can be implemented in parallel. We refer to this operation as addition (A) when we give the cost of an algorithm. However, the addition of an element of $\mathbb{F}_2$ requires a single exclusive-OR operation, denoted by XOR. Additionally, M denotes multiplications, S squarings and R square roots. We also introduce $\bar{\delta} = 1 - \delta$.

The first step consists in computing $P' = [-\nu]P$ (line 1). Multiplication over $\mathbb{F}_{2^{4m}}$ usually requires nine multiplications and twenty additions over $\mathbb{F}_{2^m}$. However, the sparsity of G allows one to compute the product $F \cdot G$ (line 14) by means of only six multiplications and fourteen additions over $\mathbb{F}_{2^m}$ (see Appendix D.2 for further details). Contrary to what was suggested by Ronan *et al.* [29], the loop unrolling technique introduced by Granger *et al.* [12] in the context of the Tate pairing in characteristic three turns out to be useless in our case. Let G_j and G_{j+1} denote the values of G at iterations j and $j+1$, respectively. Algorithm 1 computes $(F \cdot G_j) \cdot G_{j+1}$ by means of twelve multiplications and some additions over $\mathbb{F}_{2^m}$. The loop unrolling trick consists in taking advantage of the sparsity of G_j and G_{j+1}: only three multiplications over $\mathbb{F}_{2^m}$ are required to compute the product $G_j \cdot G_{j+1}$. Unfortunately, the result is not a sparse polynomial, and the multiplication by F involves nine multiplications over $\mathbb{F}_{2^m}$. Thus, computing $(G_j \cdot G_{j+1}) \cdot F$ instead of $(F \cdot G_j) \cdot G_{j+1}$ does not decrease the number of multiplications over the underlying field.

Algorithm 1 Computation of the η_T pairing in characteristic two: reversed-loop approach with square roots.

Input: $P, Q \in \mathbb{F}_{2^m}[\ell]$.
Output: $\eta_T(P,Q) \in \mathbb{F}^*_{2^{4m}}$.

$y_P \leftarrow y_P + \bar{\delta}$;	($\bar{\delta}$ XOR)
$u \leftarrow x_P + \alpha$; $v \leftarrow x_Q + \alpha$	(2α XOR)
$g_0 \leftarrow u \cdot v + y_P + y_Q + \beta$;	(1 M, 2 A, β XOR)
$g_1 \leftarrow u + x_Q$; $g_2 \leftarrow v + x_P^2$;	(1 S, 2 A)
$G \leftarrow g_0 + g_1 s + t$;	
$L \leftarrow (g_0 + g_2) + (g_1 + 1)s + t$;	(1 A, 1 XOR)
$F \leftarrow L \cdot G$;	(2 M, 1 S, 5 A, 2 XOR)
for $j = 1$ to $\frac{m-1}{2}$ **do**	
$x_P \leftarrow \sqrt{x_P}$; $y_P \leftarrow \sqrt{y_P}$; $x_Q \leftarrow x_Q^2$; $y_Q \leftarrow y_Q^2$;	(2 R, 2 S)
$u \leftarrow x_P + \alpha$; $v \leftarrow x_Q + \alpha$	(2α XOR)
$g_0 \leftarrow u \cdot v + y_P + y_Q + \beta$;	(1 M, 2 A, β XOR)
$g_1 \leftarrow u + x_Q$;	(1 A)
$G \leftarrow g_0 + g_1 s + t$;	
$F \leftarrow F \cdot G$;	(6 M, 14 A)
end for	
return F^M;	

The square roots in Algorithm 1 could be computed according to the technique described by Fong *et al.* [7]. However, this approach would require dedicated hardware and could potentially slow down a pairing coprocessor. Thus, it is attractive to study square-root-free algorithms which allow one to design simpler arithmetic and logic units. Another argument preventing the usage of square roots is that the complexity of their computation heavily depends on the particular irreducible polynomial selected for representing the field $\mathbb{F}_{2^m}$. On the other hand, the complexity of squarings is somehow more independent of the irreducible polynomial [27, 32]. To get rid of the square roots, we remark that

$$\eta_T(P,Q) = \eta_T\left(\left[2^{-\frac{m-1}{2}}\right] P, Q\right)^{2^{\frac{m-1}{2}}}.$$

Let $[2^j]Q = \left(x_{[2^j]Q}, y_{[2^j]Q}\right)$. Since

$$g_{\left[2^{\frac{m-1}{2}-j}\right]\left(\left[2^{-\frac{m-1}{2}}\right]P'\right)}(\psi(Q)) = g_{[2^{-j}]P'}(\psi(Q)),$$

the η_T pairing is defined by

$$f_{T',P'}(\psi(Q)) = l_{\left[2^{-\frac{m-1}{2}}\right]P'}(\psi(Q))^{2^{\frac{m-1}{2}}} \prod_{j=0}^{\frac{m-1}{2}} \left(\left(g_{[2^{-j}]P'}(\psi(Q))\right)^{2^{2j}}\right)^{2^{\frac{m-1}{2}-j}},$$

where

$$\begin{aligned} l_{\left[2^{-\frac{m-1}{2}}\right]P'}(\psi(Q)) &= x_{P'}^2(x_{P'}^2 + x_Q + \alpha) + (\alpha+1)x_{P'}^2 + y_{P'}^2 + y_Q + \\ &\quad \gamma + \delta + (x_{P'}^2 + x_Q)s + t, \\ \left(g_{[2^{-j}]P'}(\psi(Q))\right)^{2^{2j}} &= \left(x_{P'}^2 + 1\right) \cdot \left(x_{[2^j]Q} + 1\right) + \\ &\quad y_{P'}^2 + y_{[2^j]Q} + b + \left(x_P^2 + x_{[2^j]Q} + 1\right)s + t, \end{aligned}$$

and

$$\gamma = \begin{cases} 0 & \text{if } m \equiv 1, 7 \pmod 8, \\ 1 & \text{if } m \equiv 3, 5 \pmod 8. \end{cases}$$

Again, one can simplify the computation of the product $l_{\left[2^{-\frac{m-1}{2}}\right]P'}(\psi(Q)) \cdot g_{P'}(\psi(Q))$. Noting that $\gamma + \delta = b$ and defining $g_0' = x_{P'}^2 x_Q + x_{P'}^2 + x_Q + y_{P'}^2 + y_q + b + 1$, $g_1' = x_{P'}^2 + x_Q + 1$, and $g_2' = x_{P'}^4 + x_Q + 1$, we obtain

$$l_{\left[2^{-\frac{m-1}{2}}\right]P'}(\psi(Q)) \cdot g_{P'}(\psi(Q)) = ((g_0' + g_2') + (g_1' + 1)s + t) \cdot (g_0' + g_1's + t).$$

An implementation of the η_T pairing following this construction is given in Algorithm 2.

We also studied direct approaches based on Equation (2). However, they turned out to be slower and we will not consider such algorithms in this paper (see Appendix A for details).

2.3 Final Exponentiation

The η_T pairing has to be reduced in order to be uniquely defined. We have to raise $\eta_T(P, Q)$ to the Mth power, where

$$M = \frac{2^{4m} - 1}{N} = (2^{2m} - 1)(2^m + 1 - \nu 2^{\frac{m+1}{2}}).$$

Ronan *et al.* [29] unrolled the different powerings and proposed an algorithm involving a single inversion over $\mathbb{F}_{2^{4m}}$ when $\nu = 1$. Shu *et al.* [33] discovered that the final exponentiation only requires an inversion over $\mathbb{F}_{2^{2m}}$ when $\nu = -1$.

Algorithm 2 Computation of the η_T pairing in characteristic two: reversed-loop approach without square roots.

Input: $P, Q \in \mathbb{F}_{2^m}[\ell]$.
Output: $\eta_T(P,Q) \in \mathbb{F}^*_{2^{4m}}$.

$y_P \leftarrow y_P + \bar{\delta}$; ($\bar{\delta}$ XOR)

$x_P \leftarrow x_P^2$; $y_P \leftarrow y_P^2$; (2 S)

$y_P \leftarrow y_P + b$; $u \leftarrow x_P + 1$; ($b+1$ XOR)

$g_1 \leftarrow u + x_Q$; (1 A)

$g_0 \leftarrow x_P \cdot x_Q + y_P + y_Q + g_1$; (1 M, 3 A)

$x_Q \leftarrow x_Q + 1$; (1 XOR)

$g_2 \leftarrow x_P^2 + x_Q$; (1 S, 1 A)

$G \leftarrow g_0 + g_1 s + t$;

$L \leftarrow (g_0 + g_2) + (g_1 + 1)s + t$; (1 A, 1 XOR)

$F \leftarrow L \cdot G$; (2 M, 1 S, 5 A, 2 XOR)

for $j \leftarrow 1$ to $\frac{m-1}{2}$ **do**

  $F \leftarrow F^2$; (4 S, 4 A)

  $x_Q \leftarrow x_Q^4$; $y_Q \leftarrow y_Q^4$; (4 S)

  $x_Q \leftarrow x_Q + 1$; $y_Q \leftarrow y_Q + x_Q$; (1 A, 1 XOR)

  $g_0 \leftarrow u \cdot x_Q + y_P + y_Q$; (1 M, 2 A)

  $g_1 \leftarrow x_P + x_Q$; (1 A)

  $G \leftarrow g_0 + g_1 s + t$;

  $F \leftarrow F \cdot G$; (6 M, 14 A)

end for

Return F^M;

Let us show that this result can be extended to the case where $\nu = 1$. Since $M = (2^{2m} - 1)(2^m + 1) + \nu(1 - 2^{2m})2^{\frac{m+1}{2}}$, we compute

$$\eta_T(P,Q)^M = \left(\eta_T(P,Q)^{2^{2m}-1}\right)^{2^m+1} \cdot \left(\eta_T(P,Q)^{\nu(1-2^{2m})}\right)^{2^{\frac{m+1}{2}}},$$

and we remark that the final exponentiation requires a single inversion over $\mathbb{F}^{2^{2m}}$. Let $U = \eta_T(P,Q) \in \mathbb{F}^*_{2^{4m}}$. Writing $U = U_0 + U_1 t$, where U_0 and $U_1 \in \mathbb{F}_{2^{2m}}$ and noting that $t^{2^{2m}} = t + 1$, we obtain $U^{2^{2m}} = U_0 + U_1 + U_1 t$. Therefore, we have:

$$\begin{aligned} U^{2^{2m}-1} &= \frac{U_0 + U_1 + U_1 t}{U_0 + U_1 t} = \frac{(U_0 + U_1 + U_1 t)^2}{(U_0 + U_1 t) \cdot (U_0 + U_1 + U_1 t)} \\ &= \frac{U_0^2 + U_1^2 + U_1^2 s + U_1^2 t}{U_0^2 + U_0 U_1 + U_1^2 s}, \text{ and} \\ U^{1-2^{2m}} &= \frac{U_0 + U_1 t}{U_0 + U_1 + U_1 t} = \frac{U_0^2 + U_1^2 s + U_1^2 t}{U_0^2 + U_0 U_1 + U_1^2 s}, \end{aligned}$$

where $U_0^2 + U_0 U_1 + U_1^2 s \in \mathbb{F}_{2^{2m}}$. Algorithm 3 summarizes the computation of the $\eta_T(P,Q)^M$:

– According to our notation, we have $U = U_0 + U_1 t$, where $U_0 = u_0 + u_1 s$ and $U_1 = u_2 + u_3 s$. Since $s^2 = s + 1$, we remark that:

$$U_0^2 = (u_0^2 + u_1^2) + u_1^2 s,$$
$$U_1^2 = (u_2^2 + u_3^2) + u_3^2 s, \quad U_1^2 s = u_3^2 + u_2^2 s.$$

Therefore, 4 squarings and 2 additions over $\mathbb{F}_{2^m}$ allow us to get $T_0 = U_0^2$, $T_1 = U_1^2$, and $T_2 = U_1^2 s$.

– Multiplication over $\mathbb{F}_{2^{2m}}$ on line 3 is performed according to the Karatsuba-Ofman's scheme and involves three multiplications and four additions over $\mathbb{F}_{2^m}$:

$$T_3 = U_0 U_1 = u_0 u_2 + u_1 u_3 + ((u_0 + u_1)(u_2 + u_3) + u_0 u_2)s.$$

– Thanks to the tower field, inversion of $D = U_0^2 + U_0 U_1 + U_1^2 s \in \mathbb{F}_{2^{2m}}$ is replaced by an inversion (denoted by I), a squaring, three multiplications, and two additions over $\mathbb{F}_{2^m}$ (see Appendix C for details).

– The next step consists in computing $V = V_0 + V_1 t = U^{2^{2m}-1}$ and $W = W_0 + W_1 t = U^{\nu(1-2^{2m})}$, where V_0, V_1, W_0, and $W_1 \in \mathbb{F}_{2^{2m}}$. Defining $T_5 = \frac{U_0^2 + U_1^2 s}{U_0^2 + U_0 U_1 + U_1^2 s}$ and $T_6 = \frac{U_1^2}{U_0^2 + U_0 U_1 + U_1^2 s}$ (line 6), we easily check that $U^{2^{2m}-1} = (T_5 + T_6) + T_6 t$ and $U^{1-2^{2m}} = T_5 + T_6 t$. Thus,

$$V_0 = T_5 + T_6, \qquad W_0 = \begin{cases} T_5 + T_6 & \text{if } \nu = -1, \\ T_6 & \text{if } \nu = 1, \end{cases} \qquad V_1 = W_1 = T_6.$$

– Raising $V = V_0 + V_1 t \in \mathbb{F}_{2^{4m}}^*$ to the (2^m+1)th power over $\mathbb{F}_{2^{4m}}$ (line 15) consists in multiplying V^{2m} by V. This operation turns out to be less expensive than the usual multiplication over $\mathbb{F}_{2^{4m}}$ (see Appendix D.3 for details).

2.4 Overall Cost Evaluations

Table 1 summarizes the costs of the algorithms studied in this section in terms of arithmetic operations over $\mathbb{F}_{2^m}$. Software implementations benefit from the Extended Euclidean Algorithm (EEA) to perform the inversion over $\mathbb{F}_{2^m}$. However, supplementing a pairing coprocessor with dedicated hardware for the EEA is not the most appropriate solution. Computing the inverse of $a \in \mathbb{F}_{2^m}$ by means of multiplications and squarings over $\mathbb{F}_{2^m}$ according to Fermat's little theorem and Itoh and Tsujii's work [14] allows one to keep the circuit area as small as possible without impacting too severely on the performances [3]. Since $a^{-1} = \left(a^{2^{m-1}-1}\right)^2$, we first raise a to the power of $2^{m-1} - 1$ using a Brauer-type addition chain for $m - 1$. Then, a squaring over $\mathbb{F}_{2^m}$ suffices to obtain a^{-1}. We reported the cost of this inversion scheme for typical values of m in Table 2.

Algorithm 3 Final exponentiation of the reduced η_T pairing.

Input: $U = u_0 + u_1 s + u_2 t + u_3 st \in \mathbb{F}^*_{2^{4m}}$.
The intermediate variables m_i belong to $\mathbb{F}_{2^m}$. The T_i's, V_i's, W_i's, and D belong to $\mathbb{F}_{2^{2m}}$. V and $W \in \mathbb{F}_{2^{4m}}$.
Output: $V = U^M \in \mathbb{F}^*_{2^{4m}}$, with $M = (2^{2m}+1)(2^m - \nu 2^{\frac{m+1}{2}} + 1)$.

$m_0 \leftarrow u_0^2$; $m_1 \leftarrow u_1^2$; $m_2 \leftarrow u_2^2$; $m_3 \leftarrow u_3^2$; (4 S)
$T_0 \leftarrow (m_0 + m_1) + m_1 s$; $T_1 \leftarrow (m_2 + m_3) + m_3 s$; (2 A)
$T_2 \leftarrow m_3 + m_2 s$; $T_3 \leftarrow (u_0 + u_1 s) \cdot (u_2 + u_3 s)$; (3 M, 4 A)
$T_4 \leftarrow T_0 + T_2$; $D \leftarrow T_3 + T_4$; (4 A)
$D \leftarrow D^{-1}$; (1 I, 3 M, 1 S, 2 A)
$T_5 \leftarrow T_1 \cdot D$; $T_6 \leftarrow T_4 \cdot D$; (6 M, 8 A)
$V_0 \leftarrow T_5 + T_6$; (2 A)
V_1, $W_1 \leftarrow T_5$;
if $\nu = -1$ **then**
    $W_0 \leftarrow V_0$;
else
    $W_0 \leftarrow T_6$;
end if
$V \leftarrow V_0 + V_1 t$; $W \leftarrow W_0 + W_1 t$;
$V \leftarrow V^{2^m+1}$ (5 M, 2 S, 9 A)
for $i \leftarrow 1$ to $\frac{m+1}{2}$ **do**
    $W \leftarrow W^2$; (4 S, 4 A)
end for
Return $V \cdot W$; (9 M, 20 A)

3 Computation of the Modified Tate Pairing

Several researchers designed hardware accelerators over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$ for the modified Tate pairing. According to Barreto *et al.* [1], a second exponentiation allows one to compute the modified Tate pairing from the reduced η_T pairing. Thus, the modified Tate pairing is believed to be slower and a comparison between architectures for the modified Tate and η_T pairings would be unfair. Here, we take advantage of the bilinearity of the reduced η_T pairing and show how to get the modified Tate pairing almost for free.

3.1 Modified Tate Pairing in Characteristic Two

The modified Tate pairing in characteristic two is given by $\hat{e}(P,Q)^M = \eta_T(P,Q)^{MT}$, where $M = \frac{2^{4m}-1}{N}$ and $T = 2^m - N$ [1]. Let $V = \eta_T(P,Q)^M$. We have $V^N = \eta_T(P,Q)^{2^{4m}-1} = 1$. Since $\eta_T(P,Q)^M$ is a bilinear pairing, we obtain:

$$\hat{e}(P,Q)^M = V^T = V^{2^m - N} = V^{2^m} = \eta_T(P,Q)^{M \cdot 2^m} = \eta_T([2^m]P,Q)^M,$$

where $[2^m]P = (x_P + 1, x_P + y_P + \alpha + 1)$. Thus, it suffices to provide a hardware accelerator for the reduced η_T pairing with $[2^m]P$ and Q to get the modified

Table 1. Cost of the presented algorithms for computing the reduced η_T pairing in characteristic two in terms of operations over the underlying field $\mathbb{F}_{2^m}$.

	η_T pairing with square roots (Algorithm 1)	η_T pairing without square root (Algorithm 2)	Final Exponentiation (Algorithm 3)
Additions	$10 + 17 \cdot \frac{m-1}{2}$	$11m$	$2m + 53$
XORs	$3 + \bar{\delta} + (2\alpha + \beta) \cdot \frac{m+1}{2}$	$5 + \bar{\delta} + b + \frac{m-1}{2}$	–
Multiplications	$3 + 7 \cdot \frac{m-1}{2}$	$3 + 7 \cdot \frac{m-1}{2}$	26
Squarings	$m + 1$	$4m$	$2m + 9$
Square roots	$m - 1$	–	–
Inversions	–	–	1

Table 2. Cost of inversion over $\mathbb{F}_{2^m}$ according to Itoh and Tsujii's algorithm in terms of multiplications and squarings.

Field	$\mathbb{F}_{2^{239}}$	$\mathbb{F}_{2^{251}}$	$\mathbb{F}_{2^{283}}$	$\mathbb{F}_{2^{313}}$
Cost	10 M, 238 S	10 M, 250 S	11 M, 282 S	10 M, 312 S

Tate pairing. Since this preprocessing step involves an XOR operation and an addition over $\mathbb{F}_{2^m}$, it can be computed in software. Conversely, a processor for the modified Tate pairing computes the η_T pairing if its inputs are $[2^{-m}]P$ and Q:

$$\eta_T(P, Q)^M = \hat{e}([2^{-m}]P, Q)^M,$$

where $[2^{-m}]P = (x_P + 1, x_P + y_P + \alpha)$.

3.2 Modified Tate Pairing in Characteristic Three

The same approach allows one to compute the modified Tate pairing in characteristic three. Let m be a positive integer coprime to 6 and E be the supersingular elliptic curve defined by $E : y^2 = x^3 - x + b$, where $b \in \{-1, 1\}$. The number of rational points of E over $\mathbb{F}_{3^m}$ is given by $N = \#E(\mathbb{F}_{3^m}) = 3^m + 1 + \mu b 3^{\frac{m+1}{2}}$, with

$$\mu = \begin{cases} 1 & \text{if } m \equiv 1, 11 \pmod{12}, \\ -1 & \text{if } m \equiv 5, 7 \pmod{12}. \end{cases}$$

In characteristic three, we have the following relation between the reduced η_T and modified Tate pairings [1]:

$$\left(\eta_T(P, Q)^M\right)^{3T^2} = \left(\hat{e}(P, Q)^M\right)^L,$$

with $M = \frac{3^{6m}-1}{N}$, $T = 3^m - N$, and $L = -\mu b 3^{\frac{m+3}{2}}$. Defining $V = \eta_T(P, Q)^M \in \mathbb{F}^*_{3^{6m}}$ and seeing that $V^N = 1$, we obtain

$$V^{3T^2} = V^{3^{2m+1} - 2 \cdot 3^{m+1} \cdot N + 3N^2} = V^{3^{2m+1}}.$$

Dividing by L at the exponent level, we finally get the following relation between the reduced η_T and modified Tate pairings:

$$\begin{aligned}\hat{e}(P,Q)^M &= V^{\frac{3^{2m+1}}{L}} \\ &= V^{-\mu b 3^{\frac{3m-1}{2}}} = \eta_T\left(\left[-\mu b 3^{\frac{3m-1}{2}}\right] P, Q\right)^M,\end{aligned}$$

where $\left[-\mu b 3^{\frac{3m-1}{2}}\right] P = (\sqrt[3]{x_P} - b, -\mu b \lambda \sqrt[3]{y_P})$ and

$$\lambda = (-1)^{\frac{m+1}{2}} = \begin{cases} 1 & \text{if } m \equiv 7,\, 11 \pmod{12}, \\ -1 & \text{if } m \equiv 1,\, 5 \pmod{12}. \end{cases}$$

Again, the overhead introduced is negligible compared to the calculation time of the reduced η_T pairing. Consider now the cube-root-free reversed-loop algorithm proposed by Beuchat *et al.* (Algorithm 4 in [4]). In this case, we suggest to compute $\eta_T\left([-\mu b]P, \left[3^{\frac{3m-1}{2}}\right] Q\right)^M$. Surprisingly, the modified Tate pairing in characteristic three turns out to be slightly less expensive than the η_T pairing: we save two cubings and one addition over $\mathbb{F}_{3^m}$ (see Appendix B for details). Conversely, a processor for the modified Tate pairing provided with $[-\mu b]P$ and $\left[3^{\frac{-3m+1}{2}}\right] Q$ will return the reduced η_T pairing.

4 Implementation Results and Comparisons

4.1 A Unified Operator for the Arithmetic over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$

In [3], Beuchat *et al.* presented an FPGA-based accelerator for the computation of the η_T pairing in characteristic three. The coprocessor was based on a unified operator capable of handling all the necessary arithmetic operations over the base field $\mathbb{F}_{3^m}$. This streamlined design led to smaller circuits while retaining competitive performances with respect to the other published architectures. For these reasons, we chose to use such a unified operator for our own implementations in characteristic three. We also adapted the operator for supporting finite-field arithmetic in characteristic two.

The core of this unified operator is an array multiplier [34] for computing the product of two elements of $\mathbb{F}_{p^m}$ (where $p = 2$ or 3), represented in a polynomial basis using a degree-m polynomial $f(x)$ irreducible over $\mathbb{F}_p$: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$. D coefficients of the multiplicand are processed at each clock cycle. The D corresponding partial products are then shifted and reduced modulo $f(x)$ according to their respective weight, and finally summed into a register thanks to a tree of adders over $\mathbb{F}_{p^m}$. A feedback loop allows the accumulation of the previous partial products. A product over $\mathbb{F}_{p^m}$ is therefore computed in $\lceil m/D \rceil$ clock cycles.

With only slight modifications, it is possible for this multiplier to also support the other operations required by the computation of the modified Tate pairing. For instance, bypassing the shift/modulo-$f(x)$ reduction stage allows for additions, subtractions and accumulations. Similarly, the Frobenius endomorphism (*i.e.* squaring in characteristic two or cubing in characteristic three) only amounts to a linear combination of the coefficients of the polynomial. This linear combination can be computed at design time and then directly hard-wired as an alternative datapath during the shift/modulo stage.

4.2 Characteristic Two versus Characteristic Three

It is common knowledge that arithmetic over $\mathbb{F}_{2^m}$ is more compact and efficient than over $\mathbb{F}_{3^m}$. However, due to the different embedding degrees enjoyed by the elliptic curves of interest, competitive levels of security for pairing implementations in characteristic two are only achieved at the price of working over extension degrees much larger than what their counterparts in characteristic three require.

For a better understanding of this trade-off, we present here FPGA implementation results of a coprocessor for the modified Tate pairing in both characteristics two and three. The coprocessor is based on the previously described unified operator and implements the square- and cube-root-free reversed-loop algorithms (Algorithm 2, and Algorithm 4 in [4]) along with the corresponding final exponentiation. We also experimented with several values for D, aiming at a more exhaustive study of the trade-off between cost and performances.

Tables 3 and 4 present the post-place-and-route results for characteristic two and three respectively. These results were obtained for a Xilinx Virtex-II Pro 20 FPGA with average speedgrade, using the Xilinx ISE 9.2i tool suite. The two tables are also summarized in Figure 1.

The given results show a slight advantage of characteristic three over characteristic two, for all the studied levels of security. This goes against the performances obtained by Barreto *et al.* in the case of software implementation [1], but also against the hardware results published by Shu *et al.* in [33].

Moreover, the optimal number D of coefficients processed per clock cycle for the array multiplier appears to be 15 in characteristic two and 7 in characteristic three. However, modifying the value of this parameter does not change but slightly the overall area-time product. According to each application's requirements in terms of area and speed, one can then select the most appropriate value for D.

4.3 Comparisons

Tables 5, 6 and 7 present the cost and performances of other coprocessors for the computation of the modified Tate and reduced η_T pairings in characteristics two and three as published in the open literature. The results are summarized in Figure 2 as a comparison of these solutions against our proposed architecture in terms of their area-time product.

Despite its inherent lack of parallelism between operations, our unified operator greatly benefits from its compact design in order to reach higher frequencies. Combined with the algorithmic improvements described in this paper and in [4],

Table 3. Implementation results of the modified Tate pairing in characteristic two using our unified operator (on a Xilinx Virtex-II Pro xc2vp20, speedgrade -6).

Field	**Security [bits]**	D	**Area [slices]**	**Frequency [MHz]**	**#cycles**	**Estimated calc. time [μs]**
$\mathbb{F}_{2^{239}}$	956	7	2366	199	39075	196
		15	2736	165	20830	127
		31	4557	123	13147	107
$\mathbb{F}_{2^{251}}$	1004	7	2270	185	41969	227
		15	3140	145	22846	157
		31	4861	126	14794	117
$\mathbb{F}_{2^{283}}$	1132	7	2517	169	52820	313
		15	3481	140	27942	200
		31	5350	127	17765	140
$\mathbb{F}_{2^{313}}$	1252	7	2661	182	63167	347
		15	3731	156	33283	213
		31	6310	111	20831	186
$\mathbb{F}_{2^{459}}$	1836	7	3809	168	129780	771
		15	5297	135	66589	492
		31	8153	115	37601	327

Table 4. Implementation results of the modified Tate pairing in characteristic three using our unified operator (on a Xilinx Virtex-II Pro xc2vp20, speedgrade -6).

Field	**Security [bits]**	D	**Area [slices]**	**Frequency [MHz]**	**#cycles**	**Estimated calc. time [μs]**
$\mathbb{F}_{3^{97}}$	922	3	1896	156	27800	178
		7	2711	128	14954	117
		15	4455	105	9657	92
$\mathbb{F}_{3^{103}}$	980	3	2003	151	32649	217
		7	2841	126	16633	132
		15	4695	103	10227	99
$\mathbb{F}_{3^{119}}$	1132	3	2223	140	41788	299
		7	3225	125	20814	166
		15	5293	99	12607	127
$\mathbb{F}_{3^{127}}$	1208	3	2320	149	47234	317
		7	3379	129	24028	186
		15	5596	99	14349	145
$\mathbb{F}_{3^{193}}$	1835	3	3266	147	100668	682
		7	4905	111	48205	433
		15	8266	90	26937	298

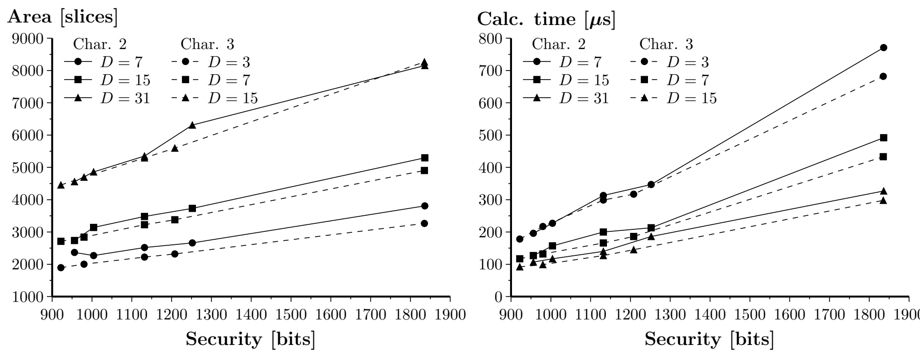

Fig. 1. Area (left) and calculation time (right) for the modified Tate pairing on our unified operator, in both characteristics two and three, for various extension degrees and different values for the parameter D.

this leads to competitive calculation times. Additionally, the streamlined design allows for reaching higher extension degrees and levels of security without risking to exhaust the FPGA resources: the slow increase of the area-time product with the security level of the system hints at the high scalability of the coprocessor.

Finally, the good performances of our solution against the previously published works vouches for a strong confidence in the outcome of our comparison between characteristics two and three for the hardware implementation of the modified Tate pairing.

5 Conclusion

We discussed several algorithms to compute the η_T pairing and its final exponentiation in characteristic two. We then showed how to get back to the modified Tate pairing at almost no extra cost. Finally, we explored the trade-offs involved in the hardware implementation of the modified Tate pairing for both characteristic two and three. Our architectures are based on the unified arithmetic operator introduced in [3], and achieve a better area-time trade-off compared to previously published solutions [10, 15, 17, 19, 20, 28–30, 33].

Our modified Tate pairing coprocessors embed a single multiplier. A challenge consists in designing parallel architectures with the same (or even a smaller) area-time product. Future work should also include a study of the η_T pairing over genus-2 curves. The Ate pairing [13] would also be of interest, for it supports non-supersingular curves.

Acknowledgment

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

Table 5. FPGA-based accelerators for the modified Tate pairing over $\mathbb{F}_{2^m}$ in the literature. The parameter D refers to the number of coefficients processed at each clock cycle by a multiplier. The architectures by Shu *et al.* [33] include four kinds of multipliers.

	Curve	**FPGA**	**#mult.**	D	**Area [slices]**	**Freq. [MHz]**	**Calculation time [μs]**
Shu *et al.* [33]	$E(\mathbb{F}_{2^{239}})$	xc2vp100	6	16	25287	84	41
			1	4			
			1	1			
			1	2			
Keller *et al.* [17]	$E(\mathbb{F}_{2^{251}})$	xc2v6000	13	1	16621	50	6440
				6	21955	43	2580
				10	27725	40	2370
Keller *et al.* [19]	$E(\mathbb{F}_{2^{251}})$	xc2v6000	1	6	3788	40	4900
			3	6	6181	40	3200
			9	6	13387	40	2600
Keller *et al.* [17]	$E(\mathbb{F}_{2^{283}})$	xc2v6000	13	1	18599	50	7980
				4	22636	49	3230
				6	24655	47	2810
Keller *et al.* [19]	$E(\mathbb{F}_{2^{283}})$	xc2v6000	1	6	4273	40	6000
			3	6	6981	40	3800
			9	6	15065	40	3000
Shu *et al.* [33]	$E(\mathbb{F}_{2^{283}})$	xc2vp100	6	32	37803	72	61
			1	4			
			1	1			
			1	2			
Ronan *et al.* [29]	$E(\mathbb{F}_{2^{313}})$	xc2vp100	14	4	34675	55	203
				8	41078	50	124
				12	44060	33	146
Ronan *et al.* [30]	$C(\mathbb{F}_{2^{103}})$	xc2vp100	20	4	21021	51	206
				8	24290	46	152
				16	30464	41	132

Table 6. FPGA-based accelerators for the modified Tate pairing over $\mathbb{F}_{3^{97}}$ in the literature. The parameter D refers to the number of coefficients processed at each clock cycle by a multiplier.

	FPGA	**#mult.**	**D**	**Area [slices]**	**Freq. [MHz]**	**Calculation time [μs]**
Grabher and Page [10]	xc2vp4	1	4	4481	150	432.3
Kerins *et al.* [20]	xc2vp125	18	4	55616	15	850

Table 7. FPGA-based accelerators for reduced η_T pairing over $\mathbb{F}_{3^{97}}$ in the literature. The parameter D refers to the number of coefficients processed at each clock cycle by a multiplier.

	FPGA	#mult.	D	Area [slices]	Freq. [MHz]	Calculation time [μs]
Ronan *et al.* [28]	xc2vp100	5	4	10540	84.8	187
Jiang [15]	xc4vlx200	Not specified	7	74105	77.7	20.9
Beuchat *et al.* [4]	xc2vp4	1	3	1833	145	192
Beuchat *et al.* [5]	xc2vp30	9	3	10897	147	33.0
	xc4vlx25	9	3	11318	200	24.2

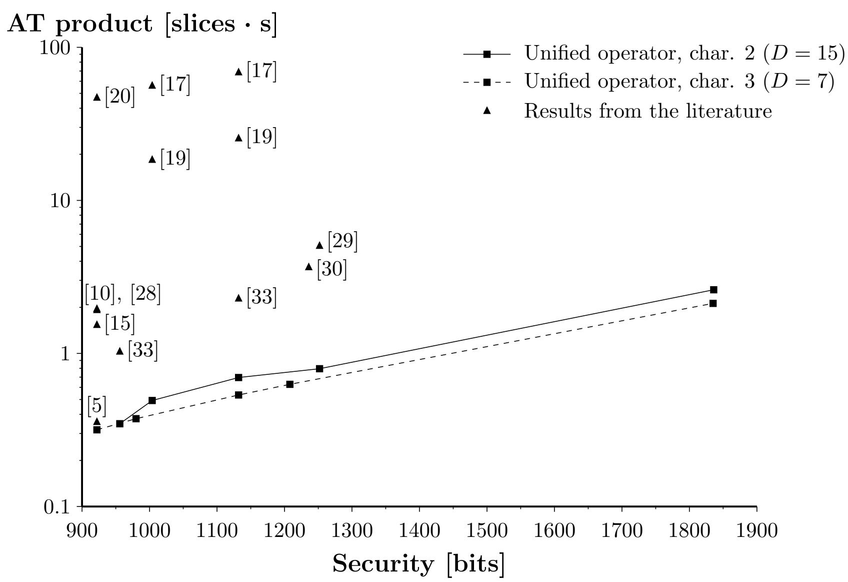

Fig. 2. Area-time product of the proposed coprocessor for the modified Tate pairing in characteristics two and three against the other solutions published in the literature.

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*, 42:239–271, 2007.
2. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, number 2442 in Lecture Notes in Computer Science, pages 354–368. Springer, 2002.
3. J.-L. Beuchat, N. Brisebarre, J. Detrey, and E. Okamoto. Arithmetic operators for pairing-based cryptography. In P. Paillier and I. Verbauwhede, editors, *Proceedings of CHES 2007*, number 4727 in Lecture Notes in Computer Science, pages 239–255. Springer, 2007.
4. J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, M. Shirase, and T. Takagi. Algorithms and arithmetic operators for computing the η_T pairing in characteristic three. Cryptology ePrint Archive, Report 2007/417, 2007.
5. J.-L. Beuchat, M. Shirase, T. Takagi, and E. Okamoto. An algorithm for the η_T pairing calculation in characteristic three and its hardware implementation. In P. Kornerup and J.-M. Muller, editors, *Proceedings of the 18th IEEE Symposium on Computer Arithmetic*, pages 97–104. IEEE Computer Society, 2007.
6. I. Duursma and H. S. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In C. S. Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, number 2894 in Lecture Notes in Computer Science, pages 111–123. Springer, 2003.
7. K. Fong, D. Hankerson, J. López, and A. Menezes. Field inversion and point halving revisited. *IEEE Transactions on Computers*, 53(8):1047–1059, August 2004.
8. G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62(206):865–874, April 1994.
9. S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In C. Fieker and D.R. Kohel, editors, *Algorithmic Number Theory – ANTS V*, number 2369 in Lecture Notes in Computer Science, pages 324–337. Springer, 2002.
10. P. Grabher and D. Page. Hardware acceleration of the Tate pairing in characteristic three. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, number 3659 in Lecture Notes in Computer Science, pages 398–411. Springer, 2005.
11. R. Granger, D. Page, and N. P. Smart. High security pairing-based cryptography revisited. In F. Hess, S. Pauli, and M. Pohst, editors, *Algorithmic Number Theory – ANTS VII*, number 4076 in Lecture Notes in Computer Science, pages 480–494. Springer, 2006.
12. 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.
13. F. Hess, N. Smart, and F. Vercauteren. The Eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, October 2006.
14. T. Itoh and S. Tsujii. A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. *Information and Computation*, 78:171–177, 1988.
15. J. Jiang. Bilinear pairing (Eta_T Pairing) IP core. Technical report, City University of Hong Kong – Department of Computer Science, May 2007.

16. A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithmic Number Theory – ANTS IV*, number 1838 in Lecture Notes in Computer Science, pages 385–394. Springer, 2000.
17. M. Keller, T. Kerins, F. Crowe, and W. P. Marnane. FPGA implementation of a $\mathrm{GF}(2^m)$ Tate pairing architecture. In K. Bertels, J.M.P. Cardoso, and S. Vassiliadis, editors, *International Workshop on Applied Reconfigurable Computing (ARC 2006)*, number 3985 in Lecture Notes in Computer Science, pages 358–369. Springer, 2006.
18. M. Keller, T. Kerins, and W. P. Marnane. FPGA implementation of a $\mathrm{GF}(2^{4m})$ multiplier for use in pairing based cryptosystems. In *Field-Programmable Logic and Applications*, pages 594–597. IEEE, 2005.
19. M. Keller, R. Ronan, W. P. Marnane, and C. Murphy. Hardware architectures for the Tate pairing over $\mathrm{GF}(2^m)$. *Computers and Electrical Engineering*, 33(5–6):392–406, 2007.
20. T. Kerins, W. P. Marnane, E. M. Popovici, and P. S. L. M. Barreto. Efficient hardware for the Tate pairing calculation in characteristic three. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, number 3659 in Lecture Notes in Computer Science, pages 412–426. Springer, 2005.
21. N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In N. P. Smart, editor, *Cryptography and Coding*, number 3796 in Lecture Notes in Computer Science, pages 13–36. Springer, 2005.
22. S. Kwon. Efficient Tate pairing computation for elliptic curves over binary fields. In C. Boyd and J. M. González Nieto, editors, *Information Security and Privacy – ACISP 2005*, volume 3574 of *Lecture Notes in Computer Science*, pages 134–145. Springer, 2005.
23. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curves logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, September 1993.
24. V. S. Miller. Short programs for functions on curves. Available at http://crypto.stanford.edu/miller, 1986.
25. V. S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, 2004.
26. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–484, Feb 2002.
27. F. Rodríguez-Henríquez, G. Morales-Luna, and J. López. Low-complexity bit-parallel square root computation over $\mathrm{GF}(2^m)$ for all trinomials. *IEEE Transactions on Computers*, 57(4):472–480, April 2008.
28. R. Ronan, C. Murphy, T. Kerins, C. Ó hÉigeartaigh, and P. S. L. M. Barreto. A flexible processor for the characteristic 3 η_T pairing. *Int. J. High Performance Systems Architecture*, 1(2):79–88, 2007.
29. R. Ronan, C. Ó hÉigeartaigh, C. Murphy, M. Scott, and T. Kerins. FPGA acceleration of the Tate pairing in characteristic 2. In *Proceedings of the IEEE International Conference on Field Programmable Technology – FPT 2006*, pages 213–220. IEEE, 2006.
30. R. Ronan, C. Ó hÉigeartaigh, C. Murphy, M. Scott, and T. Kerins. Hardware acceleration of the Tate pairing on a genus 2 hyperelliptic curve. *Journal of Systems Architecture*, 53:85–98, 2007.
31. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *2000 Symposium on Cryptography and Information Security (SCIS2000), Okinawa, Japan*, pages 26–28, January 2000.

32. M. Scott. Optimal irreducible polynomials for GF(2^m) arithmetic. Cryptology ePrint Archive, Report 2007/192, 2007.
33. C. Shu, S. Kwon, and K. Gaj. FPGA accelerated Tate pairing based cryptosystem over binary fields. In *Proceedings of the IEEE International Conference on Field Programmable Technology – FPT 2006*, pages 173–180. IEEE, 2006.
34. L. Song and K. K. Parhi. Low energy digit-serial/parallel finite field multipliers. *Journal of VLSI Signal Processing*, 19(2):149–166, July 1998.

A Computation of the η_T Pairing in Characteristic Two: Direct Approach

Shu *et al.* [33] started from Equation (2) to design their square root-free η_T pairing algorithm. First, they compute:

$$\begin{aligned} g_{[2^i]P'}(\psi(Q)) = (x^2_{[2^i]P'} + 1)(x_Q + 1) + y^2_{[2^i]P'} + y_Q + b + \\ (x^2_{[2^i]P'} + x_Q + 1)s + t. \end{aligned}$$

For $i = \frac{m-1}{2}$, they have

$$\begin{aligned} g_{\left[2^{\frac{m-1}{2}}\right]P'}(\psi(Q)) = x_{P'}x_Q + \alpha x_{P'} + \alpha x_Q + y_{P'} + y_Q + \alpha + \beta + \\ (x_{P'} + x_Q + \alpha)s + t. \end{aligned}$$

Since

$$\begin{aligned} l_{P'}(\psi(Q)) = x^2_{P'} + x_{P'}x_Q + \alpha x_{P'} + \alpha x_Q + x_Q + y_{P'} + y_Q + \alpha + \delta + 1 + \\ (x_{P'} + x_Q + \alpha + 1)s + t, \end{aligned}$$

and $\alpha + \beta = \delta + 1$, they obtain:

$$l_{P'}(\psi(Q)) = g_{\left[2^{\frac{m-1}{2}}\right]P'}(\psi(Q)) + x^2_{P'} + x_Q + \alpha + s.$$

Algorithm 4 summarizes the computation of the η_T pairing according to the above equations.

Let us see what happens when computing

$$\begin{aligned} \eta_T(P,Q)^M &= \sqrt[2^{\frac{m-1}{2}}]{\left(\eta_T(P,Q)^{2^{\frac{m-1}{2}}}\right)^M} \\ &= \left(\eta_T\left(P, \left[2^{-\frac{m-1}{2}}\right]Q\right)^{2^{\frac{m-1}{2}}}\right)^M. \end{aligned}$$

We have to calculate

$$f_{T',P'}\left(\psi\left(Q'\right)\right)^{2^{\frac{m-1}{2}}} = \left(\prod_{i=0}^{\frac{m-1}{2}} \left(g_{[2^i]P'}\left(\psi\left(Q'\right)\right)\right)^{2^{m-i-1}}\right) l_{P'}\left(\psi\left(Q'\right)\right)^{2^{\frac{m-1}{2}}},$$

Algorithm 4 Computation of the η_T pairing: direct approach without square roots [33].

$y_P \leftarrow y_P + \bar{\delta}$;	($\bar{\delta}$ XOR)
$x_P \leftarrow x_P^2$; $y_P \leftarrow y_P^2$;	(2 S)
$x'_P \leftarrow x_P$;	
$x_P \leftarrow x_P + 1$; $y_Q \leftarrow y_Q + b$; $u \leftarrow x_Q + 1$;	($b + 2$ XOR)
$g_0 \leftarrow x_P \cdot u + y_P + y_Q$;	(1 M, 2 A)
$g_1 \leftarrow x_P + x_Q$;	(1 A)
$G \leftarrow g_0 + g_1 s + t$;	
$F \leftarrow G^2$;	(2 S, 1 A, 1 XOR)
$x_P \leftarrow x_P^4$; $y_P \leftarrow y_P^4$;	(4 S)
$x_P \leftarrow x_P + 1$; $y_P \leftarrow x_P + y_P$;	(1 A, 1 XOR)
$g_0 \leftarrow x_P \cdot u + y_P + y_Q$;	(1 M, 2 A)
$g_1 \leftarrow x_P + x_Q$;	(1 A)
$G \leftarrow g_0 + g_1 s + t$;	
$F \leftarrow F \cdot G$;	(3 M, 6 A, 2 XOR)
for $i \leftarrow 2$ to $\frac{m-1}{2}$ **do**	
$F \leftarrow F^2$;	(4 S, 4 A)
$x_P \leftarrow x_P^4$; $y_P \leftarrow y_P^4$;	(4 S)
$x_P \leftarrow x_P + 1$; $y_P \leftarrow x_P + y_P$;	(1 A, 1 XOR)
$g_0 \leftarrow x_P \cdot u + y_P + y_Q$;	(1 M, 2 A)
$g_1 \leftarrow x_P + x_Q$;	(1 A)
$G \leftarrow g_0 + g_1 s + t$;	
$F \leftarrow F \cdot G$;	(6 M, 14 A)
end for	
$g_2 \leftarrow x'_P + x_Q + \alpha$;	(1 A, α XOR)
$L \leftarrow (g_0 + g_2) + (g_1 + 1)s + t$;	(1 A, 1 XOR)
$F \leftarrow F \cdot L$;	(6 M, 14 A)
Return F^M;	

where

$$\begin{aligned} Q' &= \left[2^{-\frac{m-1}{2}}\right] Q \\ &= \left(x_Q^{2^{-m+1}} + \frac{m-1}{2}, y_Q^{2^{-m+1}} + \frac{m-1}{2} x_Q^{2^{-m+1}} + \tau\left(-\frac{m-1}{2}\right)\right). \end{aligned}$$

Noting that $(m-1)/2 = \alpha + 1$, and $\tau(-(m-1)/2) = \gamma$, we obtain

$$Q' = \left(x_Q^{2^{-m+1}} + \alpha + 1, y_Q^{2^{-m+1}} + (\alpha + 1) \cdot x_Q^{2^{-m+1}} + \gamma \right).$$

One checks that:

$$\begin{aligned} g_{[2^i]P'}(\psi(Q'))^{2^{m-i-1}} = {} & (x_{P'}^{2^i} + \alpha)(x_Q^{2^{-i}} + \alpha) + y_{P'}^{2^i} + y_Q^{2^{-i}} + \beta + \\ & (x_{P'}^{2^i} + x_Q^{2^{-i}} + \alpha)s + t. \end{aligned}$$

For $i = \frac{m-1}{2}$, we obtain:

$$g_{\left[2^{\frac{m-1}{2}}\right]P'}(\psi(Q'))^{2^{\frac{m-1}{2}}} = x_{P'}^{2^{\frac{m-1}{2}}} x_Q^{2^{-\frac{m-1}{2}}} + \alpha x_{P'}^{2^{\frac{m-1}{2}}} + \alpha x_Q^{2^{-\frac{m-1}{2}}} + y_{P'}^{2^{\frac{m-1}{2}}} + y_Q^{2^{-\frac{m-1}{2}}} + \alpha + \beta + \left(x_{P'}^{2^{\frac{m-1}{2}}} + x_Q^{2^{-\frac{m-1}{2}}} + \alpha\right) s + t.$$

Since $\alpha + \beta = \delta + 1$ and

$$l_{P'}(\psi(Q'))^{2^{\frac{m-1}{2}}} = \left(x_{P'}^{2^{\frac{m-1}{2}}}\right)^2 + x_{P'}^{2^{\frac{m-1}{2}}} x_Q^{2^{-\frac{m-1}{2}}} + \alpha x_{P'}^{2^{\frac{m-1}{2}}} + \alpha x_Q^{2^{-\frac{m-1}{2}}} + x_Q^{2^{-\frac{m-1}{2}}} + y_{P'}^{2^{\frac{m-1}{2}}} + y_Q^{2^{-\frac{m-1}{2}}} + \alpha + \delta + 1 + \left(x_{P'}^{2^{\frac{m-1}{2}}} + x_Q^{2^{-\frac{m-1}{2}}} + \alpha + 1\right) s + t,$$

we deduce that:

$$l_{P'}(\psi(Q'))^{2^{\frac{m-1}{2}}} = g_{\left[2^{\frac{m-1}{2}}\right]P}(\psi(Q'))^{2^{\frac{m-1}{2}}} + \left(x_{P'}^{2^{\frac{m-1}{2}}}\right)^2 + x_Q^{2^{-\frac{m-1}{2}}} + \alpha + s.$$

We obtain an algorithm with square roots to compute the η_T pairing in characteristic two (Algorithm 5).

Table 8 summarizes the cost of these algorithms. The reversed-loop approach allows one to save a single multiplication over $\mathbb{F}_{2^m}$ and to reduce the size of the code.

Table 8. Cost of the direct algorithms for computing the reduced η_T pairing in characteristic two in terms of operations over the underlying field $\mathbb{F}_{2^m}$.

	η_T pairing without square root (Algorithm 4)	**η_T pairing with square roots (Algorithm 5)**
Additions	$11m - 3$	$11 + 17 \cdot \frac{m-1}{2}$
XORs	$7 + \bar{\delta} + \alpha + b + \frac{m-1}{2}$	$3 + \bar{\delta} + \alpha + (2\alpha + \beta) \cdot \frac{m+1}{2}$
Multiplications	$4 + 7 \cdot \frac{m-1}{2}$	$4 + 7 \cdot \frac{m-1}{2}$
Squarings	$4m - 4$	$m - 1$
Square roots	–	$m - 1$

B Computation of the Modified Tate Pairing in Characteristic Three

Beuchat *et al.* [4] proposed a cube root-free reversed-loop algorithm for the reduced η_T pairing (Algorithm 6). Recall that the modified Tate pairing is given

Algorithm 5 Computation of the η_T pairing: direct approach with square roots.

$y_P \leftarrow y_P + \bar{\delta}$;	($\bar{\delta}$ XOR)
$u \leftarrow x_P + \alpha$; $v \leftarrow x_Q + \alpha$;	(2α XOR)
$g_0 \leftarrow u \cdot v + y_P + y_Q + \beta$;	(1 M, 2 A, β XOR)
$g_1 \leftarrow u + x_Q$;	(1 A)
$F \leftarrow g_0 + g_1 s + t$;	
$x_P \leftarrow x_P^2$; $y_P \leftarrow y_P^2$;	(2 S)
$x_Q \leftarrow \sqrt{x_Q}$; $y_Q \leftarrow \sqrt{y_Q}$;	(2 R)
$u \leftarrow x_P + \alpha$; $v \leftarrow x_Q + \alpha$;	(2α XOR)
$g_0 \leftarrow u \cdot v + y_P + y_Q + \beta$;	(1 M, 2 A, β XOR)
$g_1 \leftarrow u + x_Q$;	(1 A)
$G \leftarrow g_0 + g_1 s + t$;	
$F \leftarrow F \cdot G$;	(3 M, 6 A, 2 XOR)
for $i \leftarrow 2$ to $\frac{m-1}{2}$ **do**	
$\quad x_P \leftarrow x_P^2$; $y_P \leftarrow y_P^2$;	(2 S)
$\quad x_Q \leftarrow \sqrt{x_Q}$; $y_Q \leftarrow \sqrt{y_Q}$;	(2 R)
$\quad u \leftarrow x_P + \alpha$; $v \leftarrow x_Q + \alpha$;	(2α XOR)
$\quad g_0 \leftarrow u \cdot v + y_P + y_Q + \beta$;	(1 M, 2 A, β XOR)
$\quad g_1 \leftarrow u + x_Q$;	(1 A)
$\quad G \leftarrow g_0 + g_1 s + t$;	
$\quad F \leftarrow F \cdot G$;	(6 M, 14 A)
end for	
$g_2 \leftarrow x_P^2 + x_Q + \alpha$;	(1 S, 1 A, α XOR)
$l_0 \leftarrow g_0 + g_2$;	(1 A)
$l_1 \leftarrow g_1 + 1$;	(1 XOR)
$L \leftarrow l_0 + l_1 s + t$;	
$F \leftarrow F \cdot L$;	(6 M, 14 A)
Return F^M;	

by $\eta_T\left([-\mu b]P, \left[3^{\frac{3m-1}{2}}\right]Q\right)^M$, where $\left[3^{\frac{3m-1}{2}}\right]Q = \left(\sqrt[3]{x_Q} - b, \lambda\sqrt[3]{y}\right)$. Thus, we can modify Algorithm 6 as follows:

- Since $(-\mu b)^2 = 1$, we remove line 2.
- It is no longer necessary to compute the cube of x_P and y_p (line 3).
- t is now given by $x_P - b + x_Q - b = x_P + x_Q$ and we save an addition.

Algorithm 7 summarizes these modifications.

C Inversion over $\mathbb{F}_{2^{2m}}$

Let $V = v_0 + v_1 s \in \mathbb{F}_{2^{2m}}$ be the multiplicative inverse of $U = u_0 + u_1 s \in \mathbb{F}_{2^{2m}}$, $U \neq 0$, where u_0, u_1, v_0, and $v_1 \in \mathbb{F}_{2^m}$. Since $UV = 1$, we obtain

$$\begin{cases} u_0 v_0 + u_1 v_1 = 1, \\ u_0 v_1 + u_1 v_0 + u_1 v_1 = 0. \end{cases}$$

Algorithm 6 Cube-root-free reversed-loop algorithm for computing the reduced η_T pairing in characteristic three [4].

Input: $P, Q \in E(\mathbb{F}_{3^m})[\ell]$.
Output: $\eta_T(P, Q) \in \mathbb{F}^*_{3^{6m}}$.

$x_P \leftarrow x_P + b$;	(1 A)
$y_P \leftarrow -\mu b y_P$;	
$x_Q \leftarrow x_Q^3$; $\quad y_Q \leftarrow y_Q^3$;	(2 C)
$t \leftarrow x_P + x_Q$;	(1 A)
$R \leftarrow (\lambda y_P t - \lambda y_Q \sigma - \lambda y_P \rho) \cdot (-t^2 + y_P y_Q \sigma - t\rho - \rho^2)$;	(6 M, 1 C, 6 A)
for $j \leftarrow 1$ **to** $\frac{m-1}{2}$ **do**	
$\quad R \leftarrow R^3$;	(6 C, 6 A)
$\quad x_Q \leftarrow x_Q^9 - b$; $\quad y_Q \leftarrow -y_Q^9$;	(4 C, 1 A)
$\quad t \leftarrow x_P + x_Q$; $\quad u \leftarrow y_P y_Q$;	(1 M, 1 A)
$\quad S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$;	(1 M)
$\quad R \leftarrow R \cdot S$;	(12 M, 59 A)
end for	
return R^M;	

The solution of this system of equations is then given by

$$v_0 = w^{-1} \cdot (u_0 + u_1), \text{ and} \qquad v_1 = w^{-1} \cdot u_1,$$

where $w = u_0^2 + (u_0 + u_1)u_1 \in \mathbb{F}_{2^m}$. Thus, inversion over $\mathbb{F}_{2^{2m}}$ involves three multiplications, two additions, one squaring, and an inversion over $\mathbb{F}_{2^m}$ (Algorithm 8).

D Arithmetic over $\mathbb{F}_{2^{4m}}$

D.1 Squaring over $\mathbb{F}_{2^{4m}}$

Let $U = u_0 + u_1 s + u_2 t + u_3 st \in \mathbb{F}_{2^{4m}}$. $V = U^2$ is given by $U^2 = u_0^2 + u_1^2 s^2 + u_2^2 t^2 + u_3^2 s^2 t^2$. Since $s^2 = s + 1$, $t^2 = t + s$, and $s^2 t^2 = 1 + t + st$, we obtain the following coefficients for $V = U^2$:

$$\begin{aligned} v_0 &= u_0^2 + u_1^2 + u_3^2, & v_1 &= u_1^2 + u_2^2, \\ v_2 &= u_2^2 + u_3^2, & v_2 &= u_3^2. \end{aligned}$$

Thus, four squarings and four additions over $\mathbb{F}_{2^m}$ allow one to compute $V = U^2$.

D.2 Multiplication over $\mathbb{F}_{2^{4m}}$

General Algorithm. Multiplication over $\mathbb{F}_{2^{4m}}$ is performed according to Karatsuba-Ofman's technique (Algorithm 9). It requires nine multiplications and twenty additions over $\mathbb{F}_{2^m}$. Note that we managed to save two additions over $\mathbb{F}_{2^m}$ compared to the solution proposed by Keller *et al.* [18].

Algorithm 7 Computation of the modified Tate pairing in characteristic three.

Input: $P, Q \in E(\mathbb{F}_{3^m})[\ell]$.
Output: $\eta_T(P,Q) \in \mathbb{F}^*_{3^{6m}}$.

$t \leftarrow x_P + x_Q$;	(1 A)
$R \leftarrow (\lambda y_P t - y_Q \sigma - \lambda y_P \rho) \cdot (-t^2 + \lambda y_P y_Q \sigma - t\rho - \rho^2)$;	(6 M, 1 C, 6 A)
for $j \leftarrow 1$ **to** $\frac{m-1}{2}$ **do**	
$R \leftarrow R^3$;	(6 C, 6 A)
$x_Q \leftarrow x_Q^9 - b$; $\quad y_Q \leftarrow -y_Q^9$;	(4 C, 1 A)
$t \leftarrow x_P + x_Q$; $\quad u \leftarrow \lambda y_P y_Q$;	(1 M, 1 A)
$S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$;	(1 M)
$R \leftarrow R \cdot S$;	(12 M, 59 A)
end for	
return R^M;	

Algorithm 8 Computation of $(u_0 + u_1 s)^{-1}$.

Input: $U = u_0 + u_1 s \in \mathbb{F}_{2^{2m}}$, $U \neq 0$.
Output: $V = u^{-1} = v_0 + v_1 \in \mathbb{F}_{2^{2m}}$.

$a_0 \leftarrow u_0 + u_1$;	(1 A)
$m_0 \leftarrow u_0^2$; $m_1 \leftarrow a_0 \cdot u_1$;	(1 S, 1 M)
$a_1 \leftarrow m_0 + m_1$;	(1 A)
$i_0 \leftarrow a_1^{-1}$;	(1 I)
$v_0 \leftarrow a_0 \cdot i_0$;	(1 M)
$v_1 \leftarrow u_1 \cdot i_0$;	(1 M)
Return $v_0 + v_1 s$;	

Computation of $(u_0 + u_1 s + t) \cdot (v_0 + v_1 s + t)$. The first multiplication over $\mathbb{F}_{2^{4m}}$ of Algorithm 4 (line 14) and Algorithm 5 (line 12) involves three multiplications and six additions over $\mathbb{F}_{2^m}$, as well as two XORs (Algorithm 10):

$$\begin{aligned}(u_0 + u_1 s + t) \cdot (v_0 + v_1 s + t) = {} & u_0 v_0 + u_1 v_1 + \\ & ((u_0 + u_1)(v_0 + v_1) + u_0 v_0 + 1)s + \\ & (u_0 + v_0 + 1)t + (u_1 + v_1)st.\end{aligned}$$

Computation of $(g_0 + g_1 s + t) \cdot ((g_0 + g_2) + (g_1 + 1)s + t)$. The first multiplication over $\mathbb{F}_{2^{4m}}$ of Algorithm 1 (line 7) and Algorithm 2 (line 10) can be significantly simplified. Since

$$\begin{aligned}(g_0 + g_1 s + t) \cdot ((g_0 + g_2) + (g_1 + 1)s + t) = {} & g_0 \cdot (g_0 + g_2) + g_1^2 + g_1 + \\ & (g_0 + g_1 \cdot g_2 + g_1^2 + g_1 + 1)s + \\ & (g_2 + 1)t + st,\end{aligned}$$

This operation involves one squaring, two XORs, two multiplications, and five additions over $\mathbb{F}_{2^m}$ (Algorithm 11).

Algorithm 9 Multiplication over $\mathbb{F}_{2^{4m}}$.

Input: $U = u_0 + u_1 s + u_2 t + u_3 st \in \mathbb{F}_{2^{4m}}$ and $V = v_0 + v_1 s + v_2 t + v_3 st \in \mathbb{F}_{2^{4m}}$.
Output: $W = U \cdot V$.

1. $a_0 \leftarrow u_0 + u_1$; $a_1 \leftarrow v_0 + v_1$; (2 A)
2. $a_2 \leftarrow u_0 + u_2$; $a_3 \leftarrow v_0 + v_2$; (2 A)
3. $a_4 \leftarrow u_1 + u_3$; $a_5 \leftarrow v_1 + v_3$; (2 A)
4. $a_6 \leftarrow u_2 + u_3$; $a_7 \leftarrow v_2 + v_3$; (2 A)
5. $a_8 \leftarrow a_0 + a_6$; $a_9 \leftarrow a_1 + a_7$; (2 A)
6. $m_0 \leftarrow u_0 \cdot v_0$; $m_1 \leftarrow u_1 \cdot v_1$; $m_2 \leftarrow u_2 \cdot v_2$; $m_3 \leftarrow u_3 \cdot v_3$; (4 M)
7. $m_4 \leftarrow a_0 \cdot a_1$; $m_5 \leftarrow a_2 \cdot a_3$; $m_6 \leftarrow a_4 \cdot a_5$; $m_7 \leftarrow a_6 \cdot a_7$; $m_9 \leftarrow a_8 \cdot a_9$; (5 M)
8. $a_{10} \leftarrow m_0 + m_1$; $a_{11} \leftarrow m_0 + m_4$; (2 A)
9. $w_0 \leftarrow a_{10} + m_2 + m_7$; (2 A)
10. $w_1 \leftarrow a_{11} + m_3 + m_7$; (2 A)
11. $w_2 \leftarrow a_{10} + m_5 + m_6$; (2 A)
12. $w_3 \leftarrow a_{11} + m_5 + m_8$; (2 A)
13. **Return** $w_0 + w_1 s + w_2 t + w_3 st$;

Algorithm 10 Computation of $(u_0 + u_1 s + t) \cdot (v_0 + v_1 s + t)$.

Input: $U = u_0 + u_1 s + t \in \mathbb{F}_{2^{4m}}$ and $V = v_0 + v_1 s + t \in \mathbb{F}_{2^{4m}}$.
Output: $W = U \cdot V$.

1. $a_0 \leftarrow u_0 + u_1$; $a_1 \leftarrow v_0 + v_1$; (2 A)
2. $m_0 \leftarrow u_0 \cdot v_0$; $m_1 \leftarrow u_1 \cdot v_1$; $m_2 \leftarrow a_0 \cdot a_1$; (3 M)
3. $w_0 \leftarrow m_0 + m_1$; (1 A)
4. $w_1 \leftarrow m_0 + m_2 + 1$; (1 A, 1 XOR)
5. $w_2 \leftarrow u_0 + v_0 + 1$; (1 A, 1 XOR)
6. $w_3 \leftarrow u_1 + v_1$; (1 A)
7. **Return** $w_0 + w_1 s + w_2 t + w_3 st$;

Computation of $(g_0 + g_1 s + t) \cdot (f_0 + f_1 s + f_2 t + f_3 st)$. The main loop of the η_T pairing algorithms studied in Section 2 requires the multiplication of $F \in \mathbb{F}_{2^{4m}}$ by $G = g_0 + g_1 s + t$. Taking advantage of the sparsity of G, we obtain an algorithm involving only six multiplications and fourteen additions over $\mathbb{F}_{2^{4m}}$ (Algorithm 12).

D.3 Computation of U^{2^m+1} over $\mathbb{F}_{2^{4m}}$

Let $U = u_0 + u_1 s + u_2 t + u_3 st \in \mathbb{F}_{2^{4m}}$. Seeing that $s^{2^m} = s + 1$ and $t^{2^m} = t + s + \alpha + 1$, we obtain:

$$U^{2^m} = \begin{cases} (u_0 + u_1 + u_3) + (u_1 + u_2)s + (u_2 + u_3)t + u_3 st & \text{if } \alpha = 1, \\ (u_0 + u_1 + u_2) + (u_1 + u_2 + u_3)s + (u_2 + u_3)t + u_3 st & \text{if } \alpha = 0. \end{cases}$$

A first solution to compute U^{2^m+1} would be to multiply U^{2^m} by U according to Algorithm 9. There is however a faster way to raise U to the power of $2^m + 1$. Defining $m_0 = (u_0 + u_1)(u_2 + u_3)$, $m_1 = u_0 u_1$, $m_2 = u_0 u_3$, $m_3 = u_1 u_2$, and

Algorithm 11 Computation of $(g_0 + g_1 s + t) \cdot ((g_0 + g_2) + (g_1 + 1)s + t)$.

Input: $U = g_0 + g_1 s + t \in \mathbb{F}_{2^{4m}}$ and $V = (g_0 + g_2) + (g_1 + 1)s + t \in \mathbb{F}_{2^{4m}}$.
Output: $W = U \cdot V$.
1. $s_0 \leftarrow g_1^2$; (1 S)
2. $a_0 \leftarrow g_0 + g_2$; $a_1 \leftarrow g_1 + s_0$; (2 A)
3. $m_0 \leftarrow g_0 \cdot a_0$; $m_1 \leftarrow g_1 \cdot g_2$; (2 M)
4. $w_0 \leftarrow m_0 + a_1$; (1 A)
5. $w_1 \leftarrow m_1 + g_0 + a_1 + 1$; (2 A, 1 XOR)
6. $w_2 \leftarrow g_2 + 1$; (1 XOR)
7. $w_3 \leftarrow 1$;
8. **Return** $w_0 + w_1 s + w_2 t + w_3 st$;

$m_4 = u_2 u_3$, we have

$$\begin{aligned}
U^{2^m+1} &= (u_0u_1 + u_0u_3 + u_1u_2 + u_0^2 + u_1^2) + \\
&\quad (u_0u_2 + u_1u_2 + u_1u_3 + u_2u_3 + u_2^2 + u_3^2)s + \\
&\quad (u_0u_3 + u_1u_2 + u_2u_3 + u_2^2 + u_3^2)t + (u_2u_3 + u_2^2 + u_3^2)st \\
&= (u_0u_1 + u_0u_3 + u_1u_2 + (u_0 + u_1)^2) + \\
&\quad ((u_0 + u_1)(u_2 + u_3) + u_0u_3 + u_2u_3 + (u_2 + u_3)^2)s + \\
&\quad (u_0u_3 + u_1u_2 + u_2u_3 + (u_2 + u_3)^2)t + (u_2u_3 + (u_2 + u_3)^2)st \\
&= (m_1 + m_2 + m_3 + (u_0 + u_1)^2) + (m_0 + m_2 + m_4 + (u_2 + u_3)^2)s + \\
&\quad (m_2 + m_3 + m_4 + (u_2 + u_3)^2)t + (m_4 + (u_2 + u_3)^2)st,
\end{aligned}$$

when $\alpha = 1$, and

$$\begin{aligned}
U^{2^m+1} &= (u_0u_1 + u_0u_2 + u_1u_2 + u_1u_3 + u_0^2 + u_1^2) + \\
&\quad (u_0u_2 + u_0u_3 + u_1u_3 + u_2u_3 + u_2^2 + u_3^2)s + \\
&\quad (u_0u_3 + u_1u_2)t + (u_2u_3 + u_2^2 + u_3^2)st \\
&= ((u_0 + u_1)(u_2 + u_3) + u_0u_1 + u_0u_3 + (u_0 + u_1)^2) + \\
&\quad ((u_0 + u_1)(u_2 + u_3) + u_1u_2 + u_2u_3 + (u_2 + u_3)^2)s + \\
&\quad (u_0u_3 + u_1u_2)t + (u_2u_3 + (u_2 + u_3)^2)st \\
&= (m_0 + m_1 + m_2 + (u_0 + u_1)^2) + (m_0 + m_3 + m_4 + (u_2 + u_3)^2)s + \\
&\quad (m_2 + m_3)t + (m_4 + (u_2 + u_3)^2)st,
\end{aligned}$$

when $\alpha = 0$. Thus, computing U^{2^m+1} involves only five multiplications, two squarings, and nine additions over $\mathbb{F}_{2^m}$ (Algorithm 13).

Algorithm 12 Computation of $(g_0 + g_1 s + t) \cdot (f_0 + f_1 s + f_2 t + f_3 st)$.

Input: $G = g_0 + g_1 s + t \in \mathbb{F}_{2^{4m}}$ and $F = f_0 + f_1 s + f_2 t + f_3 st \in \mathbb{F}_{2^{4m}}$.
Output: $W = G \cdot F$.

1. $a_0 \leftarrow g_0 + g_1$; $a_1 \leftarrow f_0 + f_1$; $a_2 \leftarrow f_2 + f_3$; (3 A)
2. $m_0 \leftarrow g_0 \cdot f_0$; $m_1 \leftarrow g_1 \cdot f_1$; $m_2 \leftarrow g_0 \cdot f_2$; $m_3 \leftarrow g_1 \cdot f_3$; (4 M)
3. $m_4 \leftarrow a_0 \cdot a_1$; $m_5 \leftarrow a_0 \cdot a_2$; (2 M)
4. $w_0 \leftarrow m_0 + m_1 + f_3$; (2 A)
5. $w_1 \leftarrow m_0 + m_4 + f_2 + f_3$; (3 A)
6. $w_2 \leftarrow m_2 + m_3 + f_0 + f_2$; (3 A)
7. $w_3 \leftarrow m_2 + m_5 + f_1 + f_3$; (3 A)
8. **Return** $w_0 + w_1 s + w_2 t + w_3 st$;

Algorithm 13 Computation of U^{2^m+1} over $\mathbb{F}_{2^{4m}}$.

Input: $U = u_0 + u_1 s + u_2 t + u_3 st \in \mathbb{F}_{2^{4m}}$.
Output: $V = U^{2^m+1}$.

1. $a_0 \leftarrow u_0 + u_1$; $a_1 \leftarrow u_2 + u_3$; (2 A)
2. $m_0 \leftarrow a_0 \cdot a_1$; $m_1 \leftarrow u_0 \cdot u_1$; $m_2 \leftarrow u_0 \cdot u_3$; (3 M)
3. $m_3 \leftarrow u_1 \cdot u_2$; $m_4 \leftarrow u_2 \cdot u_3$; (2 M)
4. $s_0 \leftarrow a_0^2$; $s_1 \leftarrow a_1^2$; (2 S)
5. $v_3 \leftarrow m_4 + s_1$; (1 A)
6. $v_2 \leftarrow m_2 + m_3$; (1 A)
7. **if** $\alpha = 1$ **then**
8. $\quad v_1 \leftarrow v_3 + m_0 + m_2$; (2 A)
9. $\quad v_0 \leftarrow v_2 + m_1 + s_0$; (2 A)
10. $\quad v_2 \leftarrow v_2 + v_3$; (1 A)
11. **else**
12. $\quad v_1 \leftarrow v_3 + m_0 + m_3$; (2 A)
13. $\quad v_0 \leftarrow m_0 + m_1 + m_2 + s_0$; (3 A)
14. **end if**
15. **Return** $v_0 + v_1 s + v_2 t + v_3 st$;