Efficient hardware for the Tate pairing calculation in characteristic three

T. Kerins[1], W. P. Marnane[1], E. M. Popovici[2], and P.S.L.M. Barreto[3]

[1] Dept. of Electrical and Electronic Engineering,
University College Cork, Cork City, Ireland.
{timk,liam}@rennes.ucc.ie

[2] Dept. of Microelectronic Engineering,
University College Cork, Cork City, Ireland.
e.popovici@ucc.ie

[3] Dept. Computing and Digital Systems Engineering
Escola Politécnica, Universidade de São Paulo
São Paulo, Brazil.
pbarreto@larc.usp.br

Abstract. In this paper the benefits of implementation of the Tate pairing computation in dedicated hardware are discussed. The main observation lies in the fact that arithmetic architectures in the extension field $GF(3^{6m})$ are good candidates for parallelization, leading to a similar calculation time in hardware as for operations over the base field $GF(3^m)$. Using this approach an architecture for the hardware implementation of the Tate pairing calculation based on a modified Duursma-Lee algorithm is proposed.
keywords Tate pairing, hardware, characteristic three, tower fields

1 Introduction

In recent years an ever increasing number of pairing based cryptosystems have appeared in the literature, see [1]. In turn this has driven research into efficient algorithms for the implementation of bilinear pairings on elliptic curves. To date the Tate pairing (originally introduced to cryptography by Frey and Rück in [2]) has attracted attention as the most efficiently computable bilinear pairing on elliptic curves and over supersingular elliptic curves it achieves its maximum security in characteristic three

Until 2002 the best method of Tate pairing computation on elliptic curves was via the algorithm of Miller [3]. It is an extension of the well known double-and-add method of performing point scalar multiplication on elliptic curves. This involves performing point additions and point doublings, as well as evaluation of the intermediate line functions to elements of the underlying field. These are then accumulated to give the pairing value. In 2002 the work of Galbraith *et al.* and Barreto *et al.* furthered this development so that the Tate pairing became easier to compute in practice [4][5]. As described in the BKLS/GHS algorithms, prudent choice of points, by use of a distortion map of the type discussed in [6], as well as a triple-and-add algorithm in characteristic three greatly simplifies the pairing calculation. The utilization of so called tower fields of $GF(3^m)$ for arithmetic in $GF(3^{6m})$ was originally proposed by Galbraith *et al.* [4].

In 2003 further improvements in the implementation of the Tate pairing were described by Duursma and Lee in [7], leading the DL algorithm for Tate pairing computation. Here the pairing computation was extended to more general hyperelliptic curves. Also the distortion map was incorporated into the operation of into the algorithm itself, as well as as well as modifying the loop of the BLKS/GHS algorithms, to yield a more efficient implementation. Further enhancements to the DL algorithm for supersingular elliptic curves over fields of characteristic three were described in [9], [10] and [11]. As will be described in this paper this modified DL (MDL) algorithm described in [9] is an excellent candidate for implementation on dedicated hardware. Further work on even more efficient general pairing algorithms of which the MDL algorithm is a special case appeared recently in [12]

Despite the large body of work accumulating regarding the improving algorithmic efficiency of the Tate pairing computation to date the hardware implementation of such algorithms particularly over characteristic three has received scant attention in the literature. This is somewhat surprising given the well known speed and security advantages of dedicated cryptographic hardware [13]. The main contribution of this paper is the description of how the modified DL algorithm in characteristic three can be efficiently implemented in hardware and a number of conclusions are then derived about the expected calculation time of such an architecture.

This paper is organized as follows. Section 2 describes related work on the hardware implementation of Tate pairing and arithmetic circuitry in characteristic three. Section 3 describes the modified MDL algorithm for computation of the Tate pairing and issues related to its efficient the hardware implementation. Section 4 discusses feasibility and calculation time on dedicated hardware. The conclusions of this paper are presented in Section 5.

2 Related Work

Hardware architectures for polynomial basis arithmetic in characteristic three have appeared in [14–17] while architectures for normal basis arithmetic have appeared in [18]. In hardware and indeed software the basis representation is a significant design choice. For this paper the polynomial basis representation of $GF(3^m) \cong GF(3)[x]/f(x)$ was chosen, where $f(x)$ is a degree m irreducible polynomial over $GF(3)$. In polynomial basis multiplication in $GF(3^m)$ is possible in $d = \lceil m/D \rceil$ clock cycles for some digit size D following the architectures outlined by Bertoni *et al.* in [15]. The coefficient serial multiplier discussed in [16] and [17] is a special case of this. As will be described in Section 3 the primary required operations over $GF(3^m)$ for the MDL algorithm are addition, subtraction, multiplication and cubing. It has been outlined in [15–18] that addition and subtraction (and also negation as a special case) can be efficiently performed in hardware by small combinational gate circuits using various two bit binary encoding of $GF(3)$ elements and that the gate delay for these addition and subtraction architectures is low. This implies that additive operations in $GF(3^m)$ arithmetic hardware can be performed almost for free and will not significantly contribute to a processor's calculation time. In hardware elements of $GF(3^m)$ can be represented in $2m$ bits.

In [15] a digit serial multiplier over $GF(3^m)$ is described. This considers multiplication over $GF(3^m)$ as a series of matrix-vector multiplications with coefficients in $GF(3)$. This can also be implemented efficiently in hardware assuming a low weight irreducible polynomial $f(x) \in GF(3)[x]$ (trinomial or pentanomial) has been used to define arithmetic in $GF(3^m)$. Under this assumption cubing circuitry in $GF(3^m)$ can also be efficiently implemented in much less hardware than general multiplication and cubing can be performed in a single clock cycle. An efficient algorithm and hardware architecture for inversion in $GF(3^m)$ in $2m$ clock cycles based on the extended Euclidean algorithm appeared recently in [16] and [17].

Few full hardware processor architectures for Tate pairing calculation in characteristic three have appeared in the literature. However, the authors are aware of an FPGA implementation of a pairing based cryptosystem coprocessor architecture based on the binary BLKS/GHS algorithm in [19].

3 Tate Pairing Calculation by Modified Duursma-Lee Algorithm

This section presents an outline of the modified Duursam-Lee algorithm along with some observations regarding its efficient calculation in hardware.

3.1 The Tate Pairing

Following from [5], [8] and [12] the modified Tate pairing is defined on the supersingular elliptic curve $E_{\pm}$ in affine coordinates defined over a Galois field $GF(3^m)$ where in practice m is generally prime

$$E_{\pm} : y^2 = x^3 - x \pm 1 \tag{1}$$

The set of points on $E_{\pm}$ along with the point at infinity $\mathcal{O}$ form a group of order $\#E$ under the well known chord-tangent law of composition [20]. The curve (1) is chosen so that it contains a large cyclic subgroup of prime order l, i.e. $l = \#E/n$, where n is small. Also l^2 does not divide $\#E$ but l divides $3^{6m} - 1$ and not any $3^{jm} - 1$, $j < 6$. In order to resist discrete logarithm solving attacks it is recommended that the binary representation of l is at least 150 bits long [21].

Now $E_{\pm}(GF(3^m))$ contains an l-torsion group $E_{\pm}[l](GF(3^m))$ and similarly $E_{\pm}(GF(3^{6m}))$ contains an l torsion group $E_{\pm}[l](GF(3^{6m}))$. Following [5] for our purposes the Tate pairing of order l is defined as a bilinear map between

$E_{\pm}[l](GF(3^m))$ and $E_{\pm}[l](GF(3^{6m}))$ to an element of the multiplicative subgroup of $GF(3^{6m})$, i.e. $GF(3^{6m})^*$

$$E_{\pm}[l](GF(3^m)) \times E_{\pm}[l](GF(3^{6m})) \rightarrow GF(3^{6m})^* \tag{2}$$

It is only defined up to l^{th} powers of unity; to obtain a unique value in $GF(3^{6m})$ suitable for cryptographic applications it is necessary to raise it to the power $\epsilon = (3^{6m}-1)/l$.

Now consider $P = (x_p, y_p), R = (x_r, y_r) \in E_{\pm}[l](GF(3^m))$, i.e. $x_p, y_p, x_r, y_r \in GF(3^m)$. The pairing is efficiently computed in practice by considering the point $\phi(R) \in E_{\pm}[l](GF(3^{6m}))$ where ϕ is a distortion map of the type introduced in [6]. The distortion map ϕ is defined as

$$\phi(R) = \phi((x_r, y_r)) = (\rho - x_r, \sigma y_r) \tag{3}$$

where $\rho, \sigma \in GF(3^{6m})$ such that $\rho^3 - \rho \mp 1 = 0$, ($\rho^3 - \rho - 1 = 0$ for E_+ (1) and $\rho^3 - \rho + 1 = 0$ for E_- (1)) and $\sigma^2 + 1 = 0$. Following [7–9] the the modified Tate pairing is now defined on points $P, R \in E[l](GF(3^m))$ as

$$\hat{e}(P,R) = e_{3^{3m}-1}(P, \phi(R))^{\epsilon_1} = e_l(P, \phi(R))^{\epsilon} = \tau \in GF(3^{6m}) \tag{4}$$

The calculation of (4) is performed in two stages :

– Evaluation of point $\phi(R) \in E_{\pm}[l](GF(3^{6m}))$ in the rational function f_P on $E_{\pm}(GF(3^m))$ such that $div(f_p) \sim l((P) - (\mathcal{O}))$, i.e. $e_{3^{3m}-1}(P, \phi(R)) = f_p(\phi(R)) = t \in GF(3^{6m})^*$ This is performed by modified Duursma-Lee algorithm illustrated as Algorithm 1
– Raising the resulting $t \in GF(3^{6m})$ element to the Tate power ϵ_1, i.e. $\tau = t^{\epsilon_1}$. Tate power $\epsilon_1 = \epsilon/3^{3m} = 3^{3m} - 1$ as the DL algorithm benefits from the equivalence property of the Tate pairing.

Algorithm 1: The Modified Duursma-Lee Algorithm (char 3)

```
input: $P = (x_p, y_p), R = (x_r, y_r) \in E_{\pm}[l](GF(3^m))$
output: $t = e_{3^{3m}-1}(P, \phi(R))) \in GF(3^{6m})^*$
01.  initialize : $t, \gamma, \in GF(3^{6m})$,
                 $\alpha = x_p, \beta = y_p, x = x_r^3, y = y_r^3, \mu = 0 \in GF(3^m)$
             $d = (\pm m) \bmod 3 \in GF(3)$     (* $+m \leftrightarrow E_+$, $-m \leftrightarrow E_-$ *)
02.   for $i$ in 0 to $m-1$ loop
03.     $\alpha = \alpha^9$,$\beta = \beta^9$     (* arithmetic in $GF(3^m)$ *)
04.     $\mu = \alpha + x + d$     (* arithmetic in $GF(3^m)$ *)
05.     $\gamma = -\mu^2 - \beta y \sigma - \mu\rho - \rho^2$     (* arithmetic in $GF(3^{6m})$ *)
06.     $t = t^3$     (* cubing in $GF(3^{6m})$ *)
07.     $t = t\gamma$     (* multiplication in $GF(3^{6m})$ *)
08.     $y = -y$     (* arithmetic in $GF(3^m)$ *)
09.     $d = (d \mp 1) \bmod 3$     (* $d = d - 1 \leftrightarrow E_+$, $d = d + 1 \leftrightarrow E_-$ *)
10.   end loop
return: $t$
```

After the calculation of $t = e_{3^{3m}-1}(P, \phi(R))) \in GF(3^{6m})$ by Algorithm 1 this Galois field element must then be raised to the power ϵ_1. This can be efficiently performed by representing $GF(3^{6m})$ as an extension field of $GF(3^m)$ as illustrated in Section 3.2.

3.2 A Tower field representation for $GF(3^{6m})$

As discussed in Section 2, efficient hardware architectures exist for addition, subtraction, cubing and multiplication in the base field $GF(3^m)$. However as seen from Algorithm 1 the principal complexity in performing the modified Tate pairing (4) lies in the implementation of efficient arithmetic in $GF(3^{6m})$ as well as $GF(3^m)$. The suggestion of constructing the field $GF(3^{6m})$ as an extension field of $GF(3^m)$ originally appeared in [4] and [5] and is prudent for hardware implementation. In [11] much of the arithmetic developed in this section is explicitly described.

The choice of basis for construction for $GF((3^{6m}))$ from $GF(3^m)$ is motivated by a desire to simplify as much as possible the $GF(3^{6m})$ elements ρ and σ used in the distortion map ϕ (3) appearing in steps 05. of Algorithm 1. Elements of $a \in GF(3^{6m})$ are represented as $a = \sum_{i=0}^{5} a_i \zeta^i$ where $a_i \in GF(3^m)$. The basis $\{\zeta^0, \zeta^1, \zeta^2, \zeta^3, \zeta^4, \zeta^5\} = \{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ is equivalent to a tower field extension of $GF((3^m)^6) \cong GF(((3^m)^2)^3)$ where σ and ρ are zeros of $\sigma^2 + 1 = 0$ and $\rho^3 - \rho \mp 1$ as defined by the distortion map i.e.

$$GF(3^{2m}) \cong GF(3^m)[y]/g(y) \tag{5}$$

where $g(y) = y^2 + 1$ is an irreducible polynomial over $GF(3^m)$ (provided that m and 2 are coprime) and

$$GF(3^{6m}) \cong GF(3^{2m})[z]/h_{\pm}(z) \tag{6}$$

where $h_{\pm}(z) = z^3 - z \mp 1$ is an irreducible polynomial over $GF(3^{2m})$. Polynomial $h_{+}(z) = z^3 - z - 1$ is used for E_+ and $h_{-}(z) = z^3 - z + 1$ for E_- (1) provided that m and 3 are coprime.

In this basis the elements $GF(3^{6m})$ elements σ and ρ required by the distortion map so that $\sigma^2 + 1 = 0 \in GF(3^{6m})$ and $\rho^3 - \rho \mp 1 = 0 \in GF(3^{6m})$ are represented by

$$\sigma = 0\zeta^0 + 1\zeta^1 + 0\zeta^2 + 0\zeta^3 + 0\zeta^4 + 0\zeta^5 = (0,1,0,0,0,0)$$

and

$$\rho = 0\zeta^0 + 0\zeta^1 + 1\zeta^2 + 0\zeta^3 + 0\zeta^4 + 0\zeta^5 = (0,0,1,0,0,0)$$

Now implementation of multiplication by σ and ρ in steps 05. of Algorithm 1 now becomes much simpler in hardware. Consider calculation of $\gamma \in GF(3^{6m})$

$$\begin{aligned}\gamma &= -\mu^2 - \beta y\sigma - \mu\rho - \rho^2 \\ &= (-\mu^2)\zeta^0 + (-\beta y)\zeta^1 + (-\mu)\zeta^2 + (0)\zeta^3 + (-1)\zeta^4 + (0)\zeta^5\end{aligned} \tag{7}$$

Now calculation of γ involves only two multiplications of μ^2 and βy in the $GF(3^m)$ subfield which can be carried out in parallel. The $GF(3^m)$ negation operation does not need to be clocked can be carried out by a small amount of combinational gate circuitry. Calculation of μ from step 04. of Algorithm 1. requires only addition over $GF(3^m)$ which can also be carried out un-clocked using a small amount of combinational logic. Multiplication of the respective $GF(3^m)$ elements by ζ in (7) can be performed by a simple rewiring in hardware. As elements of $GF(3^m)$ are represented by $2m$ bits in hardware elements of $GF(3^{6m})$ are represented in $12m$ bits.

A further advantage of using this representation from a hardware perspective is that cubing and full multiplication in $GF(3^{6m})$ (steps 06., 07. Algorithm 1) can also be performed using only simpler cubing and multiplication operations respectively over the base field $GF(3^m)$ and similarly all these simpler operations can be carried out in parallel.

Multiplication Consider multiplication $c = ab$ of two elements $a = \sum_{i=0}^{5} a_i\zeta^i$ and $b = \sum_{j=0}^{5} b_j\zeta^j$ of $GF(3^{6m})$ where $a_i, b_j \in GF(3^m)$. In the equivalent tower field representation from (5) and (6) elements $a \in GF(3^{6m})$ are represented of triples of elements of $GF(3^{2m})$

$$a = \underbrace{(a_0 + a_1\sigma)}_{\tilde{a}_0} + \underbrace{(a_2 + a_3\sigma)}_{\tilde{a}_1}\rho + \underbrace{(a_4 + a_5\sigma)}_{\tilde{a}_2}\rho^2$$

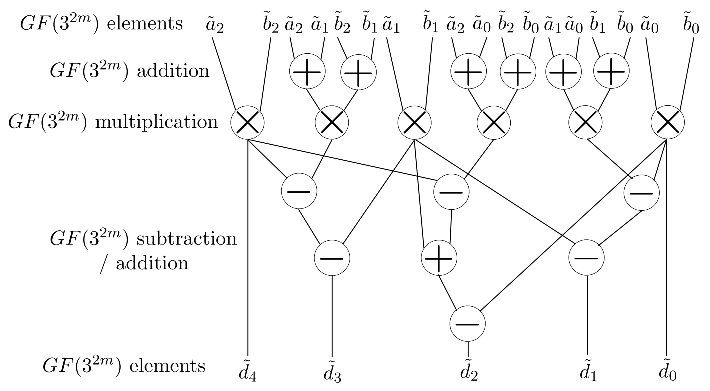
In this representation multiplication of $GF(3^{6m})$ elements $a = \sum_{i=0}^{2} \tilde{a}_i\rho^i$ and $b = \sum_{j=0}^{2} \tilde{b}_j\rho^j$, $\tilde{a}_i, \tilde{b}_j \in GF(3^{2m})$ is performed by Karatsuba multiplication [22] of a and b over $GF(3^{2m})$ to form a degree 4 polynomial $d = \sum_{k=0}^{4} \tilde{d}_k\rho^k$ over $GF(2^{2m})$

$$\begin{bmatrix} \tilde{d}_0 \\ \tilde{d}_1 \\ \tilde{d}_2 \\ \tilde{d}_3 \\ \tilde{d}_4 \end{bmatrix} = \begin{bmatrix} \tilde{a}_0\tilde{b}_0 \\ (\tilde{a}_1 + \tilde{a}_0)(\tilde{b}_1 + \tilde{b}_0) - \tilde{a}_1\tilde{b}_1 - \tilde{a}_0\tilde{b}_0 \\ (\tilde{a}_2 + \tilde{a}_0)(\tilde{b}_2 + \tilde{b}_0) + \tilde{a}_1\tilde{b}_1 - \tilde{a}_2\tilde{b}_2 - \tilde{a}_0\tilde{b}_0 \\ (\tilde{a}_2 + \tilde{a}_1)(\tilde{b}_2 + \tilde{b}_1) - \tilde{a}_2\tilde{b}_2 - \tilde{a}_1\tilde{b}_1 \\ \tilde{a}_2\tilde{b}_2 \end{bmatrix} \tag{8}$$

Polynomial d from (8) is then reduced modulo the irreducible polynomial $h_{\pm}(z)$ (6) over $GF(3^{2m})$ to form $c = \sum_{i=1}^{2} \tilde{c}_i\rho^i$ as illustrated in (10) for $h_+(z)$ and (10) in for $h_-(z)$

$$\begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} = \begin{bmatrix} \tilde{d}_0 + \tilde{d}_3 \\ \tilde{d}_1 + \tilde{d}_3 + \tilde{d}_4 \\ \tilde{d}_2 + \tilde{d}_4 \end{bmatrix} \quad (9) \qquad \begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} = \begin{bmatrix} \tilde{d}_0 - \tilde{d}_3 \\ \tilde{d}_1 + \tilde{d}_3 - \tilde{d}_4 \\ \tilde{d}_2 + \tilde{d}_4 \end{bmatrix} \quad (10)$$

As seen from (8) the composition stage of multiplication in $GF(3^{6m})$ is performed in six multiplications, seven additions and six subtractions in $GF(3^{2m})$ while the reduction stage is performed in either five additions for $h_+(z)$ (10) or three additions and two subtractions for $h_-(z)$ in $GF(3^{2m})$. Addition and subtraction in $GF(3^{2m})$ are performed coefficient-wise so are easy and cheap to perform in hardware using arrays of simple gate circuits previously discussed. The hardware complexity in $GF(3^{6m})$ multiplications lies in the required six multiplications in $GF(3^{2m})$. From the dataflow diagram for (8) illustrated as Figure 1 is is seen that the six required $GF(3^{2m})$ multiplications can be carried out in parallel.



$GF(3^{2m})$ elements
$\tilde{a}_2$ $\tilde{b}_2$ $\tilde{a}_2$ $\tilde{a}_1$ $\tilde{b}_2$ $\tilde{b}_1$ $\tilde{a}_1$ $\tilde{b}_1$ $\tilde{a}_2$ $\tilde{a}_0$ $\tilde{b}_2$ $\tilde{b}_0$ $\tilde{a}_1$ $\tilde{a}_0$ $\tilde{b}_1$ $\tilde{b}_0$ $\tilde{a}_0$ $\tilde{b}_0$
$GF(3^{2m})$ addition
$GF(3^{2m})$ multiplication
$GF(3^{2m})$ subtraction / addition
$GF(3^{2m})$ elements
$\tilde{d}_4$ $\tilde{d}_3$ $\tilde{d}_2$ $\tilde{d}_1$ $\tilde{d}_0$

Fig. 1. Dataflow for Karatsuba composition stage of multiplication in $GF(3^{6m}) \cong GF(3^{2m})[z]/h_{\pm}(z)$

Multiplication $\tilde{c} = \tilde{a}\tilde{b} \in GF(3^{2m})$ (5) of two elements $\tilde{a} = a_0 + \sigma a_1$ and $\tilde{b} = b_0 + \sigma b_1$, $a_1, a_0, b_1, b_0 \in GF(3^m)$ is performed by Karatsuba multiplication in three multiplications, two additions and three subtractions in $GF(3^m)$ as illustrated in (11)

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} a_0b_0 - a_1b_1 \\ (a_1 + a_0)(b_1 + b_0) - a_1b_1 - a_0b_0 \end{bmatrix} \tag{11}$$

Here both the polynomial composition and reduction steps are performed simultaneously by the observation that $\sigma^2 = -1 \in GF(2^{2m})$ from $g(y)$ (5). Again additive operations in $GF(3^m)$ are easily performed by simple gate circuits and multiplication in $GF(3^m)$ can be performed as discussed in Section 2. As illustrated from Figure 2 the three required $GF(3^m)$ multiplications can be carried out in parallel.

This implies that by this method multiplication in $GF(3^{6m})$ requires eighteen multiplications in the base field $GF(3^m)$ plus a number of additive operations. The advantage of implementing this operation in dedicated hardware over serial general purpose processors lies in the fact that all eighteen $GF(3^m)$ multiplications can be carried out in parallel. By parallelizing this operation the calculation time for multiplication in $GF(3^{6m})$ can be made very close to that for that in $GF(3^m)$. Due to the large number of $GF(3^m)$ additions/subtractions required (124 in total) it is impractical to consider implementing these as pure combinational logic. In practice it is more prudent to implement a smaller number of additive gate circuits and schedule the required operations through these in

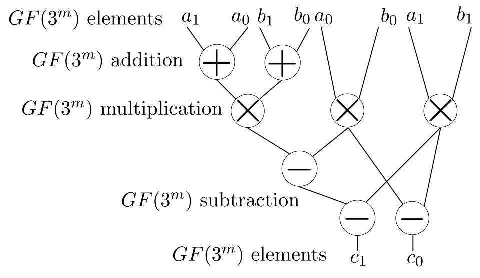


Fig. 2. Dataflow for Karatsuba multiplication in $GF(3^{2m}) \cong GF(3^m)[y]/(y^2+1)$

an extra few clock cycles. So using the digit serial multiplier of Bertoni *et al.* [15] in hardware implementation of multiplication in $GF(3^{6m})$ can be performed in $\lceil m/D \rceil + n_m$ clock cycles, where n_m is the relatively small number of extra clocks required for scheduling the additions/subtractions and register read/write operations.

Cubing Cubing $c = a^3 \in GF(3^{6m}) \cong GF(3^{2m})[z]/h_{\pm}(z)$ (6) of an element $a = \sum_{i=0}^{2} \tilde{a}_i \rho^i, \tilde{a}_i \in GF(3^{2m})$ is performed by (12) for $GF(3^{6m})$ generated by polynomial $h_+(z)$ and by (13) for $GF(3^{6m})$ generated by polynomial $h_-(z)$.

$$\begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} = \begin{bmatrix} \tilde{a}_0^3 + \tilde{a}_1^3 + \tilde{a}_2^3 \\ \tilde{a}_1^3 - \tilde{a}_2^3 \\ \tilde{a}_2^3 \end{bmatrix} \quad (12) \qquad \begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} = \begin{bmatrix} \tilde{a}_0^3 - \tilde{a}_1^3 + \tilde{a}_2^3 \\ \tilde{a}_1^3 + \tilde{a}_2^3 \\ \tilde{a}_2^3 \end{bmatrix} \quad (13)$$

Each involves three cubing operations, two additions and a subtraction in $GF(3^{2m})$. As illustrated in Figures 3 and 4 in both cases the three $GF(3^{2m})$ cubing operations can be carried out in parallel.

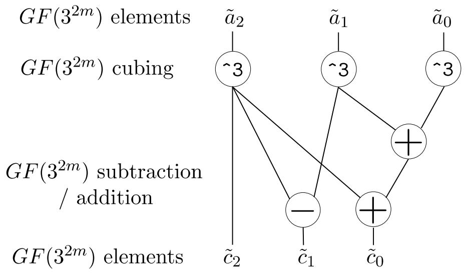


Fig. 3. Dataflow for cubing in $GF(3^{6m}) \cong GF(3^{2m})[z]/h_+(z)$

From (12) and (13) main complexity in cubing in $GF(3^{6m}) \cong GF(3^{2m})[z]/h_{\pm}(z)$ lies in performing the cubing operation in the field $GF(3^{2m}) \cong GF(3^m)[y]/g(y)$ (5). Consider an element $\tilde{a} = a_0 + \sigma a_1 \in GF(3^{2m})$ generated by $g(y) = y^2 + 1$,

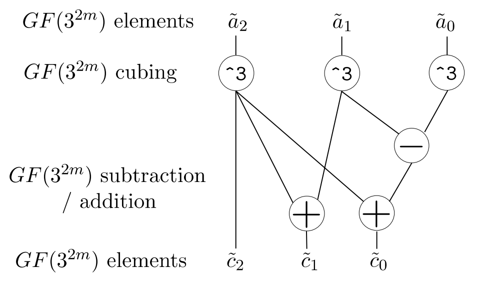


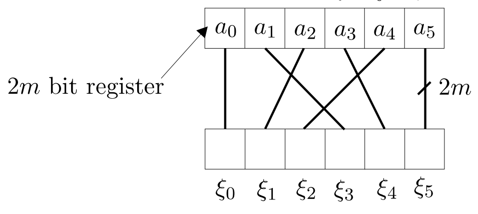
Fig. 4. Dataflow for cubing in $GF(3^{6m})[z] \cong GF(3^{2m})[z]/h_{-}(z)$

where $a_1, a_0 \in GF(3^m)$. Now $\tilde{c} = c_0 + \sigma c_1 = \tilde{a}^3 \in GF(3^{2m})$ is calculated by

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} a_0^3 \\ -a_1^3 \end{bmatrix} \tag{14}$$

which involves two cubing operations in $GF(3^m)$ which again can be performed in parallel. So the cubing operation in $GF(3^{6m})$ can be efficiently calculated in hardware by performing six $GF(3^m)$ cubing operations in parallel as well as three $GF(3^m)$ negation operations and six addition/subtraction operations. Following from [15] $GF(3^m)$ cubing can be performed efficiently in a single clock cycle and the additive operations can be performed by simple combinational gate circuits. Using this type of parallel cubing architecture with six $GF(3^m)$ cubing circuits $GF(3^{6m})$ cubing is performed in a single clock cycles and the six additive operations are performed by simple un-clocked gate circuits previously discussed.

Raising to Tate Power The basis $\{\zeta^0, \zeta^1, \zeta^2, \zeta^3, \zeta^4, \zeta^5\} = \{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ of $GF(3^{6m})$ over $GF(3^m)$ described by the distortion map as previously discussed is converted to the other basis $\{\xi, \xi^i, \xi^2, \xi^3, \xi^4, \xi^5\} = (1, \rho, \rho^2, \sigma, \sigma\rho, \sigma\rho^2)$ described by the distortion map by a simple rewiring in hardware as illustrated in Figure 5 . This is analogous to the tower field representation



ζ_0 ζ_1 ζ_2 ζ_3 ζ_4 ζ_5
a_0 a_1 a_2 a_3 a_4 a_5
$2m$ bit register
$2m$
ξ_0 ξ_1 ξ_2 ξ_3 ξ_4 ξ_5

Fig. 5. Hardware rewiring for $GF(3^{6m})$ basis change from $\{\zeta_i\}$ to $\{\xi_j\}$

$$GF(3^{3m}) \cong GF(3^m)[y]/h_{\pm}(y) \tag{15}$$

where $h_{\pm}(y) = y^3 - y \mp 1$ is an irreducible polynomial over $GF(3^m)$ $(E_+ \leftrightarrow h_+, E_- \leftrightarrow h_-)$

$$GF(3^{6m}) \cong GF(3^{3m})[z]/g(z) \tag{16}$$

where $g(z) = z^2 + 1$ is an irreducible polynomial over $GF(3^{3m})$.

In this basis $a \in GF(3^{6m})$ is represented a pair of elements $\check{a}_0, \check{a}_1 \in GF(3^{3m})$

$$a = \underbrace{(a_0 + a_1\rho + a_2\rho^2)}_{\check{a}_0} + \underbrace{(a_3 + a_4\rho + a_5\rho^2)}_{\check{a}_1}\sigma$$

As described in [11] raising $a = \sum_{i=0}^{5} a_i\xi_i \in GF(3^{6m})$ to the Tate power $\epsilon_1 = 3^{3m} - 1$ in this basis can be performed in a much more efficient manner that typical multiply-and-accumulate methods of exponentiation by the observation that for m odd

$$a^{3^{3m}} = (\check{a}_0 + \sigma\check{a}_1)^{3^{3m}} = \check{a}_0 - \sigma\check{a}_1 \tag{17}$$

as $\sigma^2 = -1 \in GF(3^{3m})$. Thus (17) implies that $c = a^{\epsilon_1} \in GF(3^{6m})$ is calculated by

$$c = \check{c}_0 + \sigma\check{c}_1 = \frac{\check{a}_0 - \sigma\check{a}_1}{\check{a}_0 + \sigma\check{a}_1} = \left[1 + \check{a}_1^2\nu^{-1}\right] + \sigma\left[1 - (\check{a}_0 + \check{a}_1)^2\nu^{-1}\right] \tag{18}$$

where $\nu = (\check{a}_0^2 + \check{a}_1^2) \in GF(3^{3m})$. Thus raising to the Tate power ϵ_1 involves five multiplications, three additions and a subtraction and an inversion in $GF(3^{3m})$. Multiplication in the field $GF(3^{3m})$ (15) is carried out in a similar manner to that outlined in (8),(9) and (10) except in this case the base field is $GF(3^m)$. The six required $GF(3^m)$ multiplications can be carried out in parallel and the additive operations are carried out by the gate circuits previously discussed. The calculation time for multiplication in $GF(3^{3m})$ is given as $\lceil m/D \rceil + n_m$. Inversion in $GF(3^{3m})$ is carried out by arithmetic in $GF(3^m)$ as illustrated in Appendix A. As this operation is performed only once it does not require to be heavily parallelized.

4 A Hardware Architecture for Tate Pairing Calculation based on Duursma-Lee Algorithm

This section considers a prospective hardware implementation for Tate pairing calculation $\hat{e}(P, R) = \tau$ (4) over elliptic curves (1) based on Algorithms 1 considering the observations from Section 3.2 on the efficient calculation time achievable by parallelizing $GF(3^{6m})$ arithmetic.

4.1 Observations on the Modified Tate Pairing Calculation

It is interesting to consider the number of clock cycles required for the main iteration loop (steps 03.-09.) of Algorithm 1 on a dedicated hardware architecture. Here eighteen $GF(3^m)$ digit size multipliers (digit size D, $d = \lceil m/D \rceil$)

and six $GF(3^m)$ cubing circuits are available in parallel, along with a suitable amount of simpler $GF(3^m)$ arithmetic circuits for performing addition, subtraction and negation. Also required on such an architecture are $2m$ bit registers for storage of elements of $GF(3^m)$ and $12m$ bit bus lines for elements of $GF(3^{6m})$ elements. The calculation time for an iteration of Algorithm 1 using this type of architecture is illustrated in Table 1. An extra two clock cycles are added to the calculation time of each operation for register read/write operations.

Table 1. Number of clock cycles required for an iteration of the Modified Duursma-Lee algorithm implemented on a parallel $GF(3^m)$ hardware architecture

step	operations	logic	clock cycles
03.	$\alpha = \alpha^3$, $\beta = \beta^3$	$\times 2$ $GF(3^m)$ cube	1+2
	$\alpha = \alpha^3$, $\beta = \beta^3$	$\times 2$ $GF(3^m)$ cube	1+2
04.	$\mu = \alpha + x + d$	combinational	0+2
05.	γ see (7)	$\times 2$ $GF(3^m)$ mul	$d + 2$
06.	$t = t^3$	$\times 6$ $GF(3^m)$ cube	$1 + 2$
07.	$t = t\gamma$	$\times 18$ $GF(3^m)$ mul	d+n_m+2
08. 09.	$y = -y$, $d = d \mp 1$	combinational	0+2

From Table 1 the modified Duursma-Lee Algorithm, Algorithm 1., can be performed on the type of dedicated hardware discussed in Section 3 in $\theta_{DL} = m(2\lceil m/D \rceil + 17 + n_m)$ clock cycles.

After $e_{3^{3m}-1}(P, \phi(R)) = t \in GF(3^{6m})$ has been performed by Algorithm 1. it is then necessary to raise this $GF(3^{6m})$ element to the Tate power ϵ_1 by (18) to generate the required unique result $\tau = t^{\epsilon_1} \in GF(3^{6m})$. This operation can be efficiently performed on much of the same underlying hardware as required for Algorithm 1. The only operations required are multiplication, and additive operations and a single inversion in the base field $GF(3^m)$. Performing the $GF(3^m)$ multiplications as required in parallel implies that (18) can be performed in $\theta_{TP} = 9(\lceil m/D \rceil + n_m) + 2m$ clock cycles.

Assuming a worst case situation where the register read/write operations and scheduling through the simple gate circuits take the same number of clock cycles as a multiplication operation (i.e. $n_m \approx \lceil m/D \rceil$) this implies that using this type of hardware architecture the number of clock cycles for calculation of (4) is given by

$$\begin{aligned} \theta_{TATE} &\approx \quad \theta_{DL} \quad + \quad \theta_{TP} \\ &\approx m(2\lceil m/D \rceil + 17 + n_m) + 9(\lceil m/D \rceil + n_m) + 2m \\ &\approx \quad 3m(\lceil m/D \rceil + 17) \quad + \quad 18\lceil m/D \rceil + 2m \end{aligned} \tag{19}$$

4.2 Implementation Aspects

The question remains : How practical is the parallel architecture as discussed in Section 4.1? In order to gauge the feasibility of the architecture the required the main arithmetic units the $GF(3^m)$ multiplier and cubing cores were captured in the VHDL hardware design language and prototyped on the Xilinx Virtex2Pro125 device [23] for the field $GF(3^{97}) \cong GF(3)[x]/x^{97} + x^{16} + 2$. The FPGA resource usage (in FPGA slices : total on device 55616) and the post place-and-route (PPR) clock frequency of the $GF(3^{97})$ digit serial multiplier are illustrated for digit sizes $D = 1, 4, 8, 12$ in Table 2.

Table 2. Prototype $GF(3^{97})$ digit serial multiplier implementation results on the Xilinx Virtex2Pro technology

D	FPGA slices	% device	PPR / MHz
1	1006	1	146
4	1821	3	84
8	2655	4	60
12	4335	7	50

The fast $GF(3^{97})$ cubing circuitry was also implemented on this target technology and occupied 514 slices (0.5%) and had a post place-and-route clock frequency of 118 MHz. The $GF(3^m)$ inverter architecture achieved a clock frequency of 62 MHz and occupied 2210 (4 % device) FPGA slices.

These preliminary results indicate that eighteen $GF(3^{97})$ multipliers with a digit size of $D = 4$ can be implemented on approximately 60% of the target device and the six $GF(3^{97})$ cubing circuits and inversion circuit on approximately 7% of the device. This leaves the remaining 33% of the target device for storage registers, control data-path and arrays of gate circuits for the simple $GF(3^m)$ addition and subtraction logic.

Using the pessimistic (19) for the required number of clock cycles this implies that calculation of (4) on E_+ from (1) over $GF(3^{97})$ with a digit size of $D = 4$ could be performed in 12,866 clock cycles. A conservative 10 MHz clock frequency on the target technology translates this into a calculation time of approximately 1.3 *ms*. This represents at least a three fold improvement over the calculation times of 4.05 *ms* and 4.33 *ms* reported recently for optimized software implementations of the same calculation on serial general purpose processors [11] [24].

5 Conclusions

In this paper the suitability of the modified Duursma-Lee (MDL) algorithm for implementation in dedicated hardware has been illustrated. Prudent choice of basis construction for the fields $GF(3^{6m})$ allows the efficient implementation of multiplication and cubing operations and only arithmetic in the $GF(3^m)$ subfield is required. Multiplication in $GF(3^{6m})$ can be performed by eighteen $GF(3^m)$ multipliers in parallel along with some combinational logic and cubing in $GF(3^{6m})$ can be performed by six $GF(3^m)$ cubing circuits in parallel along with some combinational logic. This leads to a low number of clock cycles for arithmetic in $GF(3^{6m})$ compared to those required on serial processors. Modern FPGA devices such as the Virtex2Pro currently have enough resources to contain an implementation of this type of parallel hardware for calculation of the MDL algorithm. Assuming pessimistic operating parameters this dedicated hardware is projected to at least third the calculation time currently possible using optimized software implementations.

References

1. R. Dutta, R. Barua and P. Sarkar. Pariring-based cryptography : A survey. Cryptology ePrint Archive, Report 2004/064 2004. `http://eprint.iacr.org/2004/64`.
2. G. Frey and H. Rück. A remark considering m-divisibility in the divisor class group of curves. *Mathematics of Computation,* 62:865-874, 1994.
3. V. S. Miller. Short Programs for functions on curves. Unpublished manuscript. 1986. `http://crypto.stanford.edu/miller/miller.pdf`.
4. S. Galbraith, K. Harrison and D. Soldera. Implementing the Tate pairing. In *Algorithm Number Theory Symposium - ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 324-337. Springer-Verlag 2002.
5. P.S.L.M. Barreto, H.Y. Kim, B. Lynn and M. Scott. Efficient implementation of pairing based cryptosystems. In *Advances in Cryptology - CRYPTO'2002* volume 2442 of *Lecture Notes in Computer Science*, pages 354-368. Springer-Verlag 2002.
6. E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In *Advances in Cryptology - Eurocrypt 2001* volume 2045 of *Lecture Notes in Computer Science*, pages 195-210. Springer-Verlag 2001.
7. I. Duursma and H.-S. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In *Advances in Cryptology - Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 111-123. Springer-Verlag, 2003.
8. P. S. L. M. Barreto. The well-tempered pairing. Bochum, Germany 2004. 8th Workshop on Elliptic Curve Cryptography - ECC'2004. Invited talk.
9. S. Kwon. Efficient Tate pairing computation for supersingular elliptic curves over binary fields. Cryptology ePrint Archive, Report 2004/303, 2004. `http://eprint.iacr.org/2004/303`.
10. 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. Springer-Verlag, 2004. Updated version: Cryptology ePrint Archive, Report 2004/032. `http://eprint.iacr.org/2004/303`
11. R. Granger, D. Page and M. Stam. On Small Characteristic Algebraic Tori in Pairing-Based Cryptography. Cryptology ePrint Archive, Report 2004/132, 2004. `http://eprint.iacr.org/2004/132`
12. P. S. L. M. Barreto, S. Galbraith, C. O hEigeartaigh and M. Scott. Efficient Pairing Computation on Supersingular Abelian Varieties. Cryptology ePrint Archive, Report 375/2004. 2004. `http://eprint.iacr.org/2004/375`.
13. B. Schneier. *Appplied Cryptography* second edition. John Wiley & Sons, 1996.
14. D. Page and N. P. Smart. Hardware implementation of Finite Fields of Characteristic Three. In *Cryptographic Hardware and Embedded Systems - CHES 2002* volume 2523 of *Lecture Notes in Computer Science* pages 529-539. Springer-Verlag 2002.

15. G. Bertoni, J. Guajardo, S. Kumar, G. Orlando C. Paar and T. Wollinger. Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications. In *Topics in Cryptology - CT RSA 2003* volume 2612 of *Lecture Notes in Computer Science* pages 158-175. Springer-Verlag 2003.
16. T. Kerins, E. Popovici and W. P. Marnane. Algorithms and Architectures for use in FPGA implementations of Identity Based Encryption Schemes. In *Field Programmable Logic and Applications- FPL 2004* volume 3203 of *Lecture Notes in Computer Science*, pages 74-83, Springer-Verlag 2004.
17. T. Kerins, W. P. Marnane and E. M. Popovici. Hardware Architectures for Arithemtic in $GF(p^m)$ for use in Public Key Cryptography. Preprint. 2004.
18. R. Granger, D. Page and M. Stam. Hardware and Software Normal Basis Arithemtic for Paring Based Cryptography in Characteristic Three. Cryptology ePrint Archive, Report 157/2004. 2004. `http://eprint.iacr.org/2004/157`.
19. T. Kerins, W. P. Marnane, E. M. Popovici and P. S. L. M. Barreto. A Hardware Accelerator for Pairing Based Cryptosystems. Preprint. 2004.
20. J. H. Silverman. *The Arithemtic of Elliptic Curves.* Number 106 in Graduate Studies in Mathematics. Springer-Verlag, Berlin, Germany, 1986.
21. I. Blake, G. Seroussi and N. Smart. *Elliptic Curves in Cryptography* volume 265 of *London Mathemtatical Lecture Note Series*, Cambridge University Press, 1999.
22. A. Karatsuba and Y. Ofman. Multiplication of Multidigit numbers on Automata. *Sov. Phys. Dokl (english translation)*, 7(7):595-596, 1963
23. Xilinx Inc. Virtex-2 Platform FPGAs : Complete Data Sheet. Ds031. 2004 `http://www.xilinx.com/bvdocs/publications/ds031.pdf`
24. P. S. L. M. Barreto. A note on efficient computation of cube roots in characteristic 3. Cryptology ePrint Archive, Report 035/2004 2004. `http://eprint.iacr.org/2004/305`.

Appendix A

Element $\check{c} = c_0 + c_1\rho + c_2\rho^2 = \check{a}^{-1}$ the inverse of $\check{a} = a_0 + a_1\rho + a_2\rho^2 \in GF(3^{3m})$ is calculated efficiently by the observation that $\check{c}\check{a} = 1 \in GF(3^{3m})$. The $GF(3^m)$ coefficients of $\check{c} \in GF(3^{3m})$ from defined by $h_\pm(y)$ from (15) are calculated as illustrated in (20) for $h_+(y)$:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \delta_+^{-1} \begin{bmatrix} a_0^2 + a_2^2 - a_0a_2 - a_1(a_1 + a_2) \\ -a_0a_1 + a_2^2 \\ a_1^2 - a_0a_2 - a_2^2 \end{bmatrix} \tag{20}$$

where $\delta_+ = (a_0 - a_2)a_0^2 + (-a_0 + a_1)a_1^2 + (a_0 - a_1 + a_2)a_2^2$ and by (21) for $h_-(y)$:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \delta_-^{-1} \begin{bmatrix} a_0^2 - a_1^2 + (a_1 - a_0)a_2 + a_2^2 \\ -a_0a_1 - a_2^2 \\ a_1^2 - a_0a_2 - a_2^2 \end{bmatrix} \tag{21}$$

where $\delta_- = (a_0 - a_2)a_0^2 + (-a_0 - a_1)a_1^2 + (a_0 + a_1 + a_2)a_2^2 \in GF(3^m)$

Calculation of δ_+ and δ_- from (20) and (21) involves six multiplication operations in $GF(3^m)$ then these are inverted in $2m$ clock cycles using the $GF(3^m)$ inversion architecture discussed in [16] and [17]. The calculation of $\check{c}$ in (20) and

(21) then involves a further six $GF(3^m)$ multiplication operations. In hardware this operation can be partly parallelized by performing three multiplication operations in parallel. This implies that inversion in $GF(3^m)$ can be performed in $4(\lceil m/D \rceil + n_m) + 2m$ clock cycles.