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 on 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 accelerator, characteristic three, tower fields

1 Introduction

In recent years an ever increasing number of pairing based cryptosystems have appeared in the literature, see [8]. In turn this has driven research into efficient algorithms for the implementation of bilinear pairings on elliptic curves. The Tate pairing (originally introduced to cryptography by Frey and Rück in [10]) has attracted attention as an efficiently computable bilinear pairing 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 [21]. 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 [11, 1]. As described in the BKLS/GHS algorithms, prudent choice of points, by use of a distortion map of the type discussed in [27], 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 proposed in [11, 23].

In 2003 further improvements in the implementation of the Tate pairing were described by Duursma and Lee in [9], leading to the Duursma-Lee 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 the algorithm itself, as well as modifying the loop of the BLKS/GHS algorithms, to yield a more efficient implementation. Further enhancements to the Duursma-Lee algorithm for supersingular elliptic curves over fields of characteristic three were described in [20, 25, 12]. As will be described in this paper, this modified Duursma-Lee algorithm described in [20] is an excellent candidate for implementation on dedicated hardware. Further work on even more efficient general pairing algorithms appeared recently in [3].

Despite the large body of work regarding improving the algorithmic efficiency of the Tate pairing computation, to date the hardware implementation of such algorithms, particularly over characteristic three, have received scant attention in the literature. This is somewhat surprising given the well known speed and security advantages of dedicated cryptographic hardware [24]. The main contribution of this paper is the description of how the modified Duursma-Lee 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. The architecture described in this paper has application as a hardware accelerator for pairing based cryptographic protocols in an internet server, where high speed pairing calculation is the primary design consideration.

2 Related Work

Hardware architectures for polynomial basis arithmetic in characteristic three have appeared in [23, 5, 16, 18] while architectures for normal basis arithmetic have appeared in [13]. 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 $GF(3)$. Polynomial basis multiplication in $GF(3^m)$ is possible in $d = \lceil m/D \rceil$ clock cycles for a digit size D following the architectures outlined by Bertoni *et al.* in [5]. The coefficient serial multiplier discussed in [16, 18] is a special case of this. As will be described in Section 3 the primary required operations over $GF(3^m)$ for the modified Duursma-Lee algorithm are addition, subtraction, multiplication and cubing.

It has been outlined in [5, 16, 18, 13] 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. Arithmetic in $GF(3)$ can be performed in two 4:1 FPGA lookup tables using 2-bit encoding [16] and will not significantly contribute to a processor's calculation time. In hardware, elements of $GF(3^m)$ can be represented in $2m$ bits, hence additive operations over $GF(3^m)$ cost $4m$

4:1 FPGA lookup tables. This is a relatively small hardware cost compared to that of multiplicative architectures (see Section 4.2).

In [5] 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, 18]. Few full hardware processor architectures for Tate pairing calculation in characteristic three have appeared in the literature. However, an FPGA implementation of a pairing based cryptosystem coprocessor architecture based on the binary BLKS/GHS algorithm appears 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 [1–3] 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 $\#E_\pm$ under the well known chord-tangent law of composition [26]. The curve (1) is chosen so that it contains a large cyclic subgroup of prime order l. Also l^2 does not divide $\#E_\pm$ 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 l is at least 150 bits long [6].

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 [1] 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 [27]. 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 [9, 2, 20] 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. First the modified pairing $e_{3^{3m}-1}(P, \phi(R)) = t \in GF(3^{6m})^*$ is evaluated. This is performed by the modified Duursma-Lee algorithm illustrated as Algorithm 1. Then the resulting $t \in GF(3^{6m})$ is raised to the Tate power ϵ_1, i.e. $\tau = t^{\epsilon_1}$. Tate power $\epsilon_1 = \epsilon/3^{3m} = 3^{3m} - 1$ as the Duursma-Lee 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

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 $GF(3^m)$ appeared in [11, 1] and is prudent for hardware implementation. The suggestion of the application of Karatsuba multiplication to this arithmetic

hardware appeared in [23]. In [12] much of the arithmetic developed in this section is explicitly described. Tower fields have previously been used in the implementation of Galois field arithmetic for elliptic curve cryptography in [7, 14, 22].

The choice of basis for construction of $GF((3^{6m}))$ 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 Step 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 $\sigma^2 + 1$ 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

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

The implementation of multiplication by σ and ρ in Step 05 of Algorithm 1 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 and 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 as 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 [15] 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 $GF(3^{2m})$

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

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 (9) for $h_+(z)$ and in (10) for $h_-(z)$

$$\begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} = \begin{bmatrix} \tilde{d}_0 + \tilde{d}_3 \\ \tilde{d}_1 + \tilde{d}_3 + \tilde{d}_4 \\ \tilde{d}_2 + \tilde{d}_4 \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} = \begin{bmatrix} \tilde{d}_0 - \tilde{d}_3 \\ \tilde{d}_1 + \tilde{d}_3 - \tilde{d}_4 \\ \tilde{d}_2 + \tilde{d}_4 \end{bmatrix} \tag{10}$$

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 as 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 it is seen that the six required $GF(3^{2m})$ multiplications can be carried out in parallel.

Multiplication $\tilde{c} = \tilde{a}\tilde{b} \in GF(3^{2m})$ (5) of 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 $GF(3^m)$, see (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 $\sigma^2 = -1 \in GF(3^{2m})$ from $g(y)$ in (5). Again, additive operations in $GF(3^m)$ are easily performed by simple gate circuits and

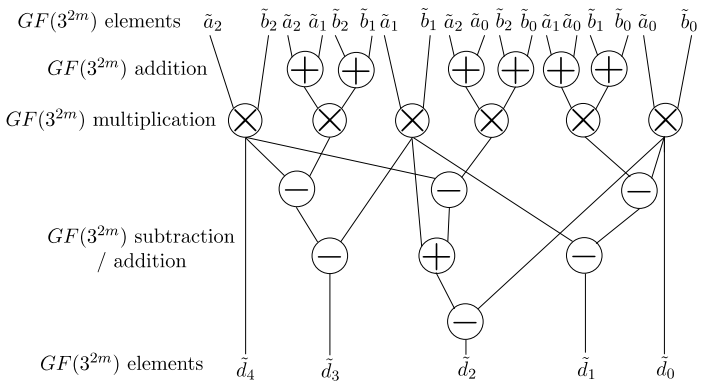

Fig. 1. Dataflow for Karatsuba composition stage of multiplication in $GF(3^{6m}) \cong GF(3^{2m})[z]/h_{\pm}(z)$

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.

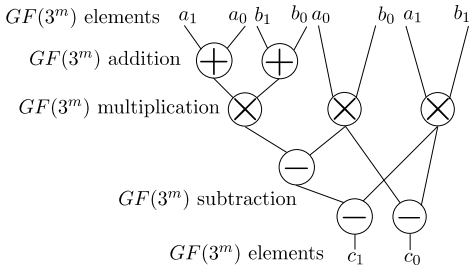

Fig. 2. Dataflow for Karatsuba multiplication in $GF(3^{2m}) \cong GF(3^m)[y]/(y^2+1)$

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 in $GF(3^m)$. Due to the large number of $GF(3^m)$ additions/subtractions required (124 in total), it may be impractical to implement these as pure combinational logic. Depending on hardware resource usage considerations, it may be more prudent to implement a smaller number of additive gate circuits and schedule the required operations through these in an extra few clock cycles. Therefore, using the digit serial multiplier of Bertoni *et al.* [5] the 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. For ease of implementation these would be controlled via finite state machines.

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 $h_-(z)$.

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

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.

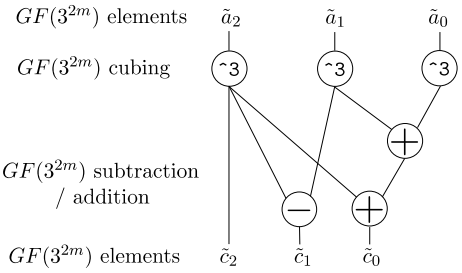

Fig. 3. Dataflow for cubing in $GF(3^{6m}) \cong GF(3^{2m})[z]/h_+(z)$

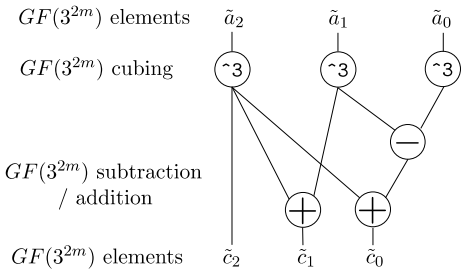

Fig. 4. Dataflow for cubing in $GF(3^{6m})[z] \cong GF(3^{2m})[z]/h_-(z)$

From (12) and (13) the main complexity of 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 $g(y) = y^2 + 1$, 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

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

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 [5] $GF(3^m)$ 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 cycle and the six additive operations are performed by simple un-clocked gate circuits previously discussed.

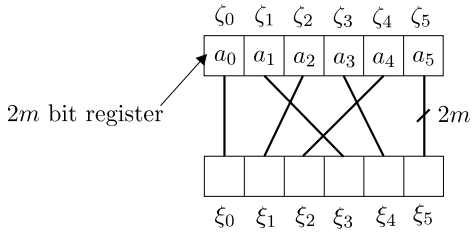

Fig. 5. Hardware rewiring for $GF(3^{6m})$ basis change from $\{\zeta_i\}$ to $\{\xi_j\}$

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 $\{\xi, \xi^1, \xi^2, \xi^3, \xi^4, \xi^5\} = (1, \rho, \rho^2, \sigma, \sigma\rho, \sigma\rho^2)$ described by the distortion map by a simple rewiring in hardware as illustrated in Figure 5 . This is analogous to the tower field representation

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

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 by 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 [12] 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 than 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, optimizing the calculation is not as important as for the $GF(3^{6m})$ multiplication and cubing previously discussed.

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 $GF(3^{6m})$. 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, it is then necessary to raise this $GF(3^{6m})$ element to the Tate power ϵ_1 using (18). This generates 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, 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. $n_m \approx \lceil m/D \rceil$) using this type of hardware architecture the number of clock cycles for calculation of (4) is given by

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

4.2 Implementation Aspects

The question remains : How practical is the parallel architecture as discussed in Section 4.1? The primary hardware complexity in this type of architecture is the implementation of the $GF(3^{6m})$ multiplier circuit using eighteen $GF(3^m)$ digit serial multipliers in parallel.

In order to gauge the feasibility of the architecture, the $GF(3^m)$ multiplier and cubing cores were captured in the VHDL hardware design language and prototyped on the Xilinx Virtex2Pro125 device [28] for the field $GF(3^{97}) \cong GF(3)[x]/x^{97} + x^{16} + 2$. The FPGA resource usage of the $GF(3^{97})$ digit serial multiplier for digit sizes $D = 1, 4, 8, 12$ is 1,006 (1% device), 1,821 (3% device), 2,655 (8% device) and 4,335 (12% device) FPGA slices, respectively. The fast $GF(3^{97})$ cubing circuitry was also implemented on this target technology and occupied 514 slices (0.5%). The $GF(3^m)$ inverter architecture of occupied 2210 (4% device) FPGA slices.

The $GF(3^{6m})$ parallel multiplier is the most complex part of the proposed architecture. It was implemented on the target technology, using eighteen $GF(3^{97})$ multipliers with a digit size of $D = 4$, and all of the additive operations were performed in parallel (multiplication in $\lceil 97/4 \rceil = 25$ clock cycles). In this case, all of the arithmetic was hardwired into the design. In total, it occupied 32,403 FPGA slices (inc. routing) which represents 58% of the target device. A post place and route clock frequency of 29.3 MHz was achieved for the $GF(3^{6m})$ multiplier and this translates into a calculation time of 0.9 μs. This preliminary result indicates that a parallel $GF(3^{97})$ multiplier using eighteen $GF(3^{97})$ multipliers with a digit size of $D = 4$ can be accommodated on the target device. The six required $GF(3^{97})$ cubing circuits and inversion circuit in total occupy approximately 7% of the device. This leaves the remaining 35% of the device for storage registers, control and arrays of gate circuits for the simple $GF(3^m)$ addition and subtraction logic.

The data path for the parallel multiplier, the principal complexity of this processor architecture, has already been fully implemented. The simpler parallel cubing logic can be implemented in a similar manner. In our experience the addition of a register bank for storage, and control via finite state machines (as in [17, 19]) is a straightforward matter. Using this method of control some redesign of hardware is involved to accommodate future algorithmic improvements. However, the ability to reconfigure the target FPGA makes such possible changes easy to accommodate. 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 estimated 15 MHz clock frequency for the entire processor (data path, control and memory) implies a calculation time for (4) over $E_+(GF(3^{97}))$ of 0.85 ms. This represents a considerable improvement over the calculation times of 4.05 ms and 4.33 ms reported for optimized software implementations on serial general purpose processors [12, 4].

The proposed architecture could also be adapted, (with a small amount of extra control) to also perform scalar multiplication on $[k]P \in E_{\pm}(GF(3^m))$ and exponentiation $t^k \in GF(3^{6m})$ as required in most pairing based protocols. Using the formulae in [1], point cubing is performed by four cubings in $GF(3^m)$ and point addition is performed by an inversion, two multiplications and a cubing in $GF(3^m)$ (neglecting the cost of the simpler additive operations). Assuming $k \approx m = 97$ (Hasse's Theorem [6]), the base three representation of k is approximately 97 trits [11]. In general, k is chosen so that these are mostly zero (say 25% nonzero). Under these assumptions $[k]P \in E_+(GF(3^{97}))$ is performed in approximately 0.5 ms using a serial triple-and-add algorithm at the same clock frequency. Using a cube-and-multiply method for $t^k \in GF((3^{97})^6)$ this is performed using the parallel multiplier and cubing circuitry previously discussed in approximately 0.1 ms.

5 Conclusions

In this paper the suitability of the modified Duursma-Lee 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 and cubing in $GF(3^{6m})$ can be performed by six $GF(3^m)$ cubing circuits in parallel, along with some combinational logic for additive operations. 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 modified Duursma-Lee algorithm. Assuming pessimistic operating parameters this type of dedicated parallel hardware is projected to drastically reduce the calculation time currently possible using optimized software implementations.

Acknowledgement

The authors would like to acknowledge the help of the anonymous referees, whose comments were valuable in the preparation of this paper. This research was funded as part of an Enterprise Ireland Research innovation fund project.

References

1. 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.
2. P. S. L. M. Barreto. The well-tempered pairing. Bochum, Germany 2004. 8th Workshop on Elliptic Curve Cryptography - ECC 2004. Invited talk.
3. 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`.
4. 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`.
5. 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.
6. I. Blake, G. Seroussi and N. Smart. *Elliptic Curves in Cryptography* volume 265 of *London Mathemtatical Lecture Note Series*, Cambridge University Press, 1999.
7. E. DeWin, A. Bosselaers, S. Vandenberghe, P. De Gersem, and J. Vandewalle. A fast software implementation for arithemtic operations in $GF(2^n)$. In *Advances in Cryptology - Asiacrypt 1996*, volume 1163 of *Lecture Notes in Computer Science*, pages 65-76. Springer-Verlag, 1996.
8. R. Dutta, R. Barua and P. Sarkar. Pairing-based cryptography : A survey. Cryptology ePrint Archive, Report 2004/064 2004. `http://eprint.iacr.org/2004/64`.
9. 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.
10. G. Frey and H. Rück. A remark considering m-divisibility in the divisor class group of curves. *Mathematics of Computation,* 62:865-874, 1994.
11. 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.
12. 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`
13. 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`.
14. J. Guajardo and C. Paar. Efficient Algorithms for Elliptic Curve Cryptosystems. In *Advances in Cryptology - Crypto 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 342-355. Springer-Verlag, 1997.

15. A. Karatsuba and Y. Ofman. Multiplication of Multidigit numbers on Automata. *Sov. Phys. Dokl (english translation)*, 7(7):595-596, 1963
16. T. Kerins, E. M. 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, E. M. Popovici, W. P. Marnane. An FPGA Implementation of a Flexible Secure Elliptic Curve Crpytography Processor. In *Applied Reconfigurable Computing - ARC 2005*, pages 22-30, IADIS press 2005.
18. 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.
19. T. Kerins, W. P. Marnane, E. M. Popovici and P. S. L. M. Barreto. A Hardware Accelerator for Pairing Based Cryptosystems. Preprint. 2005.
20. 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`.
21. V. S. Miller. Short Programs for functions on curves. Unpublished manuscript. 1986. `http://crypto.stanford.edu/miller/miller.pdf`.
22. C. Paar and P. Soria-Rodriguez. Fast Arithemtic Architecturs for Public Key Algorithms over Galois Fields $GF((2^n)^m)$. In *Advances in Cryptology - Eurocrypt 1997* volume 1233 of *Lecture Notes in Computer Science* pages 363-378. Springer-Verlag 1997.
23. 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.
24. B. Schneier. *Appplied Cryptography* second edition. John Wiley & Sons, 1996.
25. 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`
26. J. H. Silverman. *The Arithemtic of Elliptic Curves*. Number 106 in Graduate Studies in Mathematics. Springer-Verlag, Berlin, Germany, 1986.
27. 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.
28. Xilinx Inc. Virtex-2 Platform FPGAs : Complete Data Sheet. Ds031. 2004 `http://www.xilinx.com/bvdocs/publications/ds031.pdf`

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 $\check{c} \in GF(3^{3m})$ 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 polynomial $h_-(y)$:

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

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, 18]. 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.