Third International Conference on Systems

An Efficient Hardware Implementation of the Tate Pairing in Characteristic Three*

Giray Kömürcü
Bogaziçi University, Faculty of Engineering, Electrical and Electronics Engineering Dept., Istanbul, Turkey
giraykomurcu@su.sabanciuniv.edu,
TUBITAK, National Institute for Electronics and Cryptography, *Gebze, Kocaeli, Turkey*

Erkay Savas
Sabanci University, Faculty of Engineering and Natural Sciences, Computer Science Dept., Istanbul, Turkey
erkays@sabanciuniv.edu

Abstract

DL systems with bilinear structure recently became an important base for cryptographic protocols such as identity-based encryption (IBE). Since the main computational task is the evaluation of the bilinear pairings over elliptic curves, known to be prohibitively expensive, efficient implementations are required to render them applicable in real life scenarios. We present an efficient accelerator for computing the Tate Pairing in characteristic 3, using the Modified-Duursma-Lee algorithm. Our accelerator shows that it is possible to improve the area-time product by 12 times on FPGA, compared to estimated values from one of the best known hardware architecture [6] implemented on the same type of FPGA. Also the computation time is improved upto 16 times compared to software applications reported in [17]. In addition, we present the result of an ASIC implementation of the algorithm, which is the first hitherto.

Keywords: Bilinear pairings**,** Tate Pairing, Characteristic Three, FPGA Implementation, hardware accelerator.

1 Introduction

Identity-based encryption (IBE), is a public key cryptosystem that allows any arbitrary string to be used as a public key, such as recipients' email address. This vastly reduces the amount of work, to set up an online lookup for public keys and presents novel functionalities especially useful in access control systems while maintaining privacy and anonymity. Shamir introduced the concept of identity-based cryptography in 1984 [2]. However, the concept became practical only with Boneh and Franklin in 2003 [3]. Tate Pairing, originally developed by Frey and Rück [5], became popular, since it is efficiently computable and achieves its maximum security in characteristic three over supersingular elliptic curves [6]. Later, in [9,16] tower fields of $GF(3^m)$, $GF(3^{6m})$ was proposed. Duursma and Lee [11] further improved the implementation of Tate Pairing and proposed Duursma-Lee algorithm.

The algorithm used here first appeared in [12] with further improvements and eliminating the cube root operation at the expense of two extra cubing operations. However, implementing pairing operations in software falls short of matching speed requirements of many pairing-based cryptography applications, especially in embedded systems. Therefore, despite the fact and necessity that designing dedicated hardware architectures gained significant importance, there is not much work on this subject in the literature. The modified Duursma-Lee algorithm was previously implemented partially as a dedicated hardware only in [6] on FPGA. Our aim is to design an accelerator that reduces the computation time and area of Tate pairing computation in characteristic three, to realize it on FPGA's and build the first ASIC implementation of Tate pairing.

2 Related Work

One of the earliest work is by Page and Smart [13] which described $GF(3^m)$ arithmetic architectures for cryptographic applications. They, later, implemented Tate pairing with Duursma-Lee algorithm using an accelerator for arithmetic in $GF(3^{m})$[1].

The work by Kerins et. al. proposes a Tate pairing implementation [6] based on the modified Duursma-Lee algorithm. With their approach, it is possible to multiply two polynomials in $GF(3^{6m})$ using the same number of clock cycles as multiplying two $GF(3^m)$ polynomials, at the expense of area overhead and reduced clock frequency.

In our accelerator we use the parallel hardware architecture and optimize it in terms of area and speed especially working on sub-blocks. We optimize cubing

* This research is partially supported by the Scientific and Technological Research Council of Turkey under project number 105E089.

978-0-7695-3105-2/08 $25.00 © 2008 IEEE
DOI 10.1109/ICONS.2008.27

and multiplication units for specific irreducible polynomials used in the construction of ternary extension fields reducing the total area significantly. Additionally, we try to find an optimum algorithm and architecture to design a suitable Tate pairing accelerator for relatively constrained settings.

Our contribution is three-fold: i) we present a *full realization* of the accelerator on both FPGA and ASIC for Duursma-Lee algorithm for the first time, and ii) we demonstrate that sub-blocks in the accelerator can be improved in terms of both area and time complexity by applying good design techniques, and iii) we show that our actual implementation of Duursma-Lee algorithm is in fact faster and smaller than the estimated values given in the previous work [6].

3 Tate Pairing Calculation and Modified Duursma-Lee Algorithm

The modified Tate pairing is basically a transformation that takes two points on an elliptic curve $E_{\pm} : y^2 = x^3 - x \pm 1$ defined over $GF(3^m)$ and outputs a nonzero element of in tower extension field $GF(3^{6m})$.

Arithmetic operations required to implement the modified Duusma-Lee Algorithm is addition, subtraction, cubing and multiplication in $GF(3^m)$ and cubing and multiplication in $GF(3^{6m})$.

Algorithm 1: The Modified Duursma-Lee Algorithm (char 3) [6]

```
input: $P = (x_p, y_p)$, $R = (x_r, y_r) \in E_{\pm}[l](GF(3^m))$
output: $t = e_3^{3m-1}(P, \phi(R)) \in GF(3^{6m})^*$
initialize : $t = 1 \in GF(3^{6m})$,
$\alpha = x_p$, $\beta = y_p$, $x = x_r^3$, $y = y_r^3$, $\mu = 0 \in GF(3^m)$
$d = (\pm m) \bmod 3 \in GF(3)$
for $i$ in 0 to $m$-1 loop
  $\alpha = \alpha^9$, $\beta = \beta^9$ (* arithmetic in $GF(3^m)$ *)
  $\mu = \alpha + x + \mathrm{d}$ (* arithmetic in $GF(3^m)$ *)
  $\gamma = (-\mu^2)\zeta^0 + (-\beta y)\zeta^1 + (-\mu)\zeta^2 + (0)\zeta^3 + (-1)\zeta^4 + (0)\zeta^5$
  (*$\zeta = 3^m$*)
  $t = t^3$ (* cubing in $GF(3^{6m})$ *)
  $t = t\gamma$ (* multiplication in $GF(3^{6m})$ *)
  $y = -y$ (* arithmetic in $GF(3^m)$ *)
  $d = (d \pm 1) \bmod 3$
end loop
return: $t$
```

Constructing the ternary extension field $GF(3^{6m})$ on the base field of $GF(3^m)$ is suggested in [16, 9] and described explicitly in [6]. Use of extension fields simplifies the arithmetic operations and allows parallelization for the cubing and multiplication operations.

4 Arithmetic in Characteristic Three and Our Sub-Blocks

In this section, we present hardware architectures for addition, subtraction, multiplication and cubing in $GF(3^m)$. Characteristic three arithmetic is slightly more complicated than characteristic two arithmetic since coefficients can take three values; {0, 1, 2}. Hence two bits are needed to represent each digit in GF(3) using the encoding {0, 1, 2} = {00, 01, 10}. The negation in this representation is performed by swapping the most and the least significant bits (which is almost free in hardware implementations) since $2 \equiv -1 \bmod 3$. Since negation operation is used very often especially in performing $GF(3^{6m})$ multiplication, this particular representation is very useful in our case. For arithmetic operations, m bit elements are expressed as $2m$ bit arrays as follows

$$A = (\{a^H_{m-1}, a^L_{m-1}\}, \ldots\ldots\ldots, \{a^H_1, a^L_1\}, \{a^H_0, a^L_0\})$$

4.1 Addition and Subtraction

Addition and subtraction is performed digit-wise by using the Boolean expression in [1], i.e.

$C_i = A_i + B_i$, for $i = 0, 1, \ldots, m-1$ and
$t = (A^L_i \vee B^H_i) \oplus (A^H_i \vee B^L_i)$
$C^H_i = (A^L_i \vee B^L_i) \oplus t$ and $C^L_i = (A^H_i \vee B^H_i) \oplus t$

where $\vee$ and $\oplus$ stands for logical OR and XOR operations, respectively. In the used representation, negation and multiplication of GF(3) element by two are equivalent operations and performed by swapping the most and least significant bits of the digit representing the element. Therefore, subtraction in $GF(3^m)$ is equally efficient as the addition in the same field and thus the same adder block is used for both operations. If subtraction is needed, bits in digits of subtrahend are individually swapped and connected to the adder block. Since this is achieved by only wiring no additional hardware resource is used.

When implemented on FPGAs, for each GF(3) addition, two 4-input "look-up tables" (LUTs) are used. Since one slice is composed of two LUTs, for m-bit long $GF(3^m)$ additions m slices are used. This result is almost the same in all papers implementing characteristic three addition such as [10]. The delay of the addition operation is 5,061 ns on Xilinx Virtex2p 100 device.

4.2 Cubing

The modified Duursma-Lee algorithm requires cubing operation in $GF(3^{6m})$ and it is possible to build a parallel architecture by using $GF(3^m)$ cubing blocks as explained in the next section. Cubing is a linear operation in characteristic three and we adopt the technique presented in [15]. For characteristic three, Frobenius map is written as follows:

$$A^3 \equiv \Big(\sum_{i=0}^{m-1} a_i x^i\Big)^3 \bmod p(x) = \sum_{i=0}^{m-1} a_i x^{3i} \bmod p(x)$$

This formula can be represented as follows:

$$A^3 \equiv \sum_{\substack{i=0\\ i\equiv 0 \bmod 3}}^{3(m-1)} a_{i/3} x^i \bmod p(x) \equiv T+U+V \bmod p(x) \equiv$$

$$\Big(\sum_{\substack{i=0\\ i\equiv 0 \bmod 3}}^{m-1} a_{i/3} x^i\Big)+\Big(\sum_{\substack{i=m\\ i\equiv 0 \bmod 3}}^{2m-1} a_{i/3} x^i\Big)+\Big(\sum_{\substack{i=2m\\ i\equiv 0 \bmod 3}}^{3(m-1)} a_{i/3} x^i\Big) \bmod p(x)$$

Here the degrees of the second U and the third terms V are bigger than m and need to be reduced. For $p(x) = x^m + p_t x^t + p_0$ and $t < m/3$, the terms can be represented as follows as also showed in [15]:

$$U = \sum_{\substack{i=m\\ i\equiv 0 \bmod 3}}^{2m-1} a_{i/3} x^{i-m}(-p_t x^t - p_0) \bmod p(x)$$

$$V = \sum_{\substack{i=2m\\ i\equiv 0 \bmod 3}}^{3(m-1)} a_{i/3} x^{i-2m}(a^{2t} – p_t p_0 a^t + 1) \bmod p(x)$$

Reduction is basically done by additions. For irreducible polynomial $p(x) = x^m + p_t x^t + p_0$, every occurances of x^m and x^{2m} are replaced with $(-p_t x^t - p_0)$ and $(a^{2t} – p_t p_0 a^t + 1)$[1], respectively.

Table 1: Comparison of cubing circuits for GF(3^{97})

Cubing circuit	Proposed circuit	Circuit in [6]	Circuit in [15]
No. of Slices	116	514	-
No. of LUTs	222	-	388*
Max. Frequency	144MHz	-	-

*Estimation by authors of [15]; not the result of an actual implementation.

We optimize the reduction for the well known polynomial $p(x) = x^{97} + x^{16} + 2$ and calculate the terms to be added to achieve reduction in the same clock cycle. The optimization for a specific polynomial results in a very efficient implementation. We use 111 GF(3) adders to complete the cubing operation. And critical path of the system consists of three serially connected GF(3) adders. As seen from Table 1, our implementation is 2.5 times more efficient than the implementations in [6] and [15]. Although the implementation details of the cubing circuits are not clear in [6] and [15], the improvement in the slice and LUT numbers should be due to register free design and doing the reduction for a given polynomial.

4.3 Multiplication

Multiplication is the most important operation for pairing implementations due to its complexity. Since the modified, as explained in the next section. Duursma Lee algorithm requires GF(3^{6m}) multiplications, we need 18 parallel GF(3^m) multipliers in parallel. Therefore, designing efficient multiplier architecture is the key for an efficient hardware accelerator.

Hardware architectures proposed in the literature for GF(3^m) multiplication can be treated in three major classes: parallel, serial and digit multipliers. Firstly, parallel multipliers are not appropriate on constrained devices.

Secondly, serial multipliers process a single coefficient of the multiplier at each clock cycle. These types of multipliers require m clock cycles for each GF(3^m) multiplication, while their area consumption and critical path delay are relatively small compared to other types of multipliers.

Finally, digit multipliers are very similar to serial multipliers but they process w coefficients of the multiplier at each clock cycle rather than a single coefficient. Consequently, the operation is completed in $\lceil m/w \rceil$ cycles. The area consumption is more than the serial multipliers and increases with w. Since the area and critical path delay also increase with w, choosing w is an important decision influenced by area and time constraints. We prefer to use serial multipliers in our implementation, which incur increased number of clock cycles, while providing a better solution in terms of area and frequency. Serial multipliers can also be treated in two classes: i) least-significant-element-first (LSE) and ii) most-significant-element-first (MSE). Although there is not much difference between the two types we implement the LSE Multiplier. As illustrated in Algorithm 2, the reduction is performed in interleaved fashion.

Algorithm 2: LSE-Multiplier [15]

```
input: $A = \sum_{i=0}^{m-1} a_i\alpha^i$, $B = \sum_{i=0}^{m-1} b_i\alpha^i$, $a_i, b_i \in$ GF(3)
output: C $\equiv A \,.\, B = \sum_{i=0}^{m-1} c_i\alpha^i$, where $c_i \in$ GF(3)
  $C \leftarrow 0$
  for $i = 0$ to $m - 1$ do
      $C \leftarrow b_i A + C$
      $A \leftarrow A\,\alpha \bmod p(\alpha)$
  end for
  return: $C$
```

[1] Note that $x^{2m} = (x^m)^2 = (-p_t x^t - p_0)^2 = a^{2t} – p_t p_0 a^t + 1$ in GF(3).

For interleaved reduction, we subtract $a_m(p_{m-1}x^{m-1} + \ldots + p_1x + p_0x)$ from the partial result C whenever $a_m \neq 0$ since $x^m = -p_{m-1}x^{m-1} - \ldots - p_1x - p_0$.

Two LSE multipliers are designed to examine the effects of fixed versus generic polynomials on time and space complexities. The advantage of the generic design is that it can be used with any polynomial in characteristic three which is flexible. In case of fixed polynomials, the coefficients of the polynomial can be hard-coded into the multiplier unit resulting in reduction of design complexity. For the fixed irreducible polynomial of $x^{97} + x^{16} + 2$, used in many IBE implementations in literature, only two GF(3) additions are needed in each iteration of interleaved reduction. As illustrated in Table 2, the multiplier with hard-coded irreducible polynomial is 30% better than the generic multiplier in terms of area.

Table 2: Comparison of GF(3^{97}) multiplication circuits

GF(3^{97}) Multiplier	Fixed LSE	Generic LSE	LSE in [6][2]	LSE in [15]
# of Slices	389	599	1006	
# of LUTs	727	1166	-	600* (LUT+FF)
Frequency	161MHz	161MHz	-	-
Total time (µs	0,61	0,61	-	-

*Estimation by paper's author.

The proposed GF(3^m) LSE multiplier architecture is shown in Figure 1. The proposed multiplier is implemented for m = 97 on virtex2p-100 for comparison purposes since it is the same Xilinx device used in [6]. As observed in Table 2, the fixed multiplier is nearly 2.5 times smaller than the architecture in [6] and the generic multiplier consumes around 60% of the area of the same architecture In Table 2. As a result our architecture is better than the architectures in the literature to the best of our knowledge.

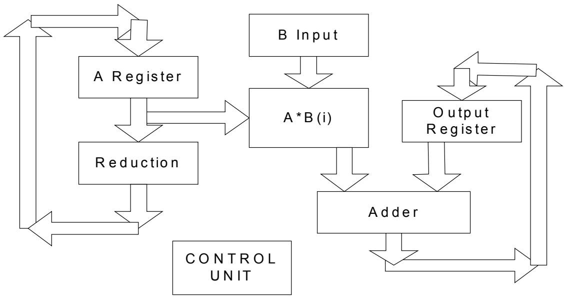

Figure 1: LSE multiplier architecture over GF(3^m)

5 Hardware Implementation of Tate Pairing Based on Modified Duursma Lee

GF(3^{6m}) can be considered as an extension field over GF(3^{2m}) with irreducible polynomial $z^3 - z \pm 1$ [6]. Also as suggested in [6], the multiplication in GF(3^{6m}) can be done in two steps: i) Karatsuba multiplication for polynomials with coefficients from GF(3^{2m}), and ii) reduction with irreducible polynomial $z^3 - z \pm 1$. Reader can profitably refer to [6] for further details.

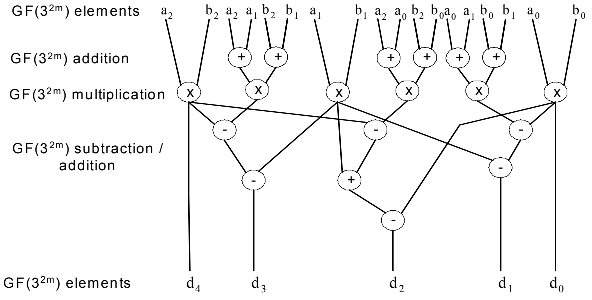

Figure 2: GF(3^{6m}) multiplier unit from [6]

In Figure 2, GF(3^{6m}) Karatsuba multiplier unit, as proposed in [6], is illustrated, where nodes represent the GF(3^{2m}) adders, subtracters, and multipliers. Similarly, GF(3^{2m}) can also be seen as an extension field over GF(3^m) with irreducible polynomial $y^2 + 1$. Since the adder/subtracter units operate on the corresponding coefficients of the operand polynomials, their structure is the same as GF(3^m) adders. GF(3^{2m}) multiplier, on the other hand, consists of GF(3^m) adders, subtracters, and multipliers as seen in Figure 3.

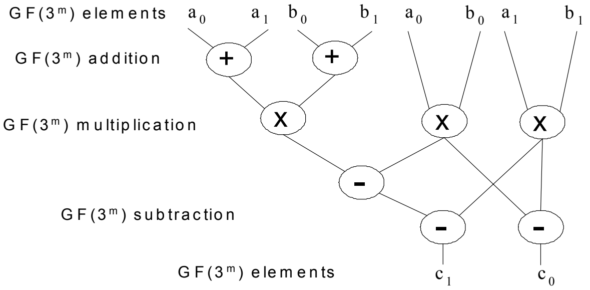

Figure 3: GF(3^{2m}) multiplier unit from [6]

As seen in Figure 2, GF(3^{6m}) Karatsuba multiplier has five GF(3^{2m}) elements as output. The result of the Karatsuba multiplier has the form $\tilde{d}_4z^4 + \tilde{d}_3z^3 + \tilde{d}_2z^2 + \tilde{d}_1z + \tilde{d}_0$. Since $z^3 = z + 1$ from the irreducible polynomial, we have $(\tilde{d}_2 + \tilde{d}_4)z^2 + (\tilde{d}_1 + \tilde{d}_4 + \tilde{d}_3)z + (\tilde{d}_0 + \tilde{d}_4)$.

[2] 1006 slices for digit size D = 1 is included for a fair comparison to our serial multiplier. Note that the clock count given in 4th column in Table 4 is for D = 4, whic consumes 1821 slices.

To summarize, 18 GF(3^m) multipliers and 52 GF(3^m) adders are used in one GF(3^{6m}) multiplier. The advantage of the proposed architecture is that multiplication is completed within m clock cycles as a GF(3^m) multiplication[3]. In order to explore reduction strategies, we develop two implementations: i) all the blocks are parallel and ii) we limit the number of adders after the multipliers to four and the operations are scheduled. This approach increases the number of clock cycles by five (2.5% of all operations), but significantly reduces the amount space consumed by adders. Similarly, we try to use scheduling approach to decrease the number of multipliers. However, not only that scheduling does not give successful results on FPGA implementation but also increases the number of slices around by 5%. We leave the scheduling approach for ASIC implementations as the future work since it may save chip space in ASIC. Lastly, for additions and subtractions we used the same adder block by just rewiring the inputs to swap the bits of the subtrahend since it negates the GF(3) elements in the employed representation.

The second GF(3^{6m}) block is for performing cubing operation and as in the case of the multiplier it is constructed using arithmetic units of the base field GF(3^{2m}) as proposed in [6]. As shown in Figure 4, GF(3^{6m}) cubing circuitry includes three adder/subtracter and three cubing blocks in GF(3^{2m}), while GF(3^{2m}) cubing circuit includes two GF(3^m) cubing circuit without any additional overhead but negation. Recall that $(a_2z^2 + a_1z + a_0)^3 = (a_2^3z^6 + a_1^3z^3 + a_0^3)$ and $z^3 = z + 1$ and $z^6 = z^2 - z - 1$. Thanks to the efficient GF(3^m) cubing blocks, implementing GF(3^{6m}) cubing block with parallel blocks does not consume much area and allows to finish the operation in one clock cycle.

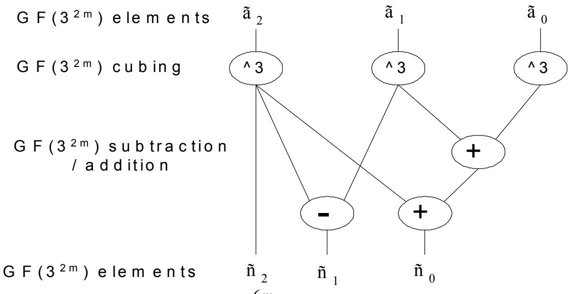

Figure 4: GF(3^{6m}) cubing unit from [6]

5.1 Proposed Coprocessor Architecture

After building the efficient blocks that are needed for our accelerator, we design a control unit and a datapath for the Tate Pairing operation. The operation may be divided into two big phases as initialization and loop. In Table 3 operations are described in detail.

Table 3: Number of clock cycles required for modified Duursma-Lee algorithm

	Step	Operation	Clock Cycles	total cycle for m = 97
Init.	1-4	$\alpha = x_{p,}$ ß = y_p $x = x_r^3$ $y = y_r^3$	4	4
Loop	5	$\alpha = \alpha^3$, ß = ß^3	1	97
Loop	6	$\alpha = \alpha^3$, ß = ß^3	1	97
Loop	7	$u = \alpha + x + d$	1	97
Loop	8	$\Upsilon = (-\mu^2)\zeta^0 + (-\beta y)\zeta^1 + (-\mu)$	97	97*97
Loop	9	$t = t^3$	1	97

Loop	10	t = t*ϒ, y=-y,
		d=d-1mod3
			97	97*97

				19210

When the initialization is completed, accelerator starts operating in a loop. For the entire operation, we use only one GF(3^{6m}) multiplier for step 10, one GF(3^{6m}) cubing circuit for step 9, two GF(3^m) cubing circuits for steps 5 and 6, two GF(3^m) multipliers for step 8 and a number of adders. The main advantage of our accelerator is that most of the operations are completed in single clock cycle. If the adder and cubing circuits were implemented with registers, clock count would increase around by 400 and registers would increase the area of the accelerator.

5.2 Implementation Aspects

Table 4: Comparison of our work with previous calculations of Modified Duursma-Lee Algorithm.

	Tate Pairing with fixed multiplier	Tate Pairing with generic multiplier	Tate pairing in [6]*	Tate Pairing in Software
	14267	16955	53406[4]	-
Frequency	77,37 MHz	69,73 MHz	15 MHz	-
Clock Count	19210	19210	12222[5]	-
Total Time	250,72 µs	278,19 µs	0.815 ms	4,05 ms
Area*Time	1	1,33	12.2	-

*Estimation by the authors.

FPGA Implementation: In this work we mapped our blocks and the whole accelerator to Xilinx Virtex2Pro100 device, since the previous works on the subject used the same device. Two different versions of our hardware

[3] The adders before and after 18 GF(3^m) multipliers are, in fact, in the critical path; therefore they do not add to the cycle count.

[4] 55616-2210 = 53406 since the inverter circuit required for final exponentiation operation is not implemented. Total slice count of 55616 is taken from [18] since the estimation figures in [6] are not clear.

[5] 12866 – 644 = 12222. We subtract the number of clock cycles spent on the final exponentiation (644) for a fair comparison.

pairing accelerator is synthesized as seen from the Table 4 below. First accelerator uses the GF(3^m) multiplier with fixed reduction polynomial of $p(x) = x^{97} + x^{16} + 2$. It occupies 14267 slices (32% of device) with an operating frequency of 77 MHz. In this case total calculation time is about 251 μs. Our second version of the accelerator is implemented with generic GF(3^m) multiplier and results are presented in table 4.

As seen from Table 4, our implementation of Modified Duursma-Lee algorithm is almost three times (2.93) better than the previous implementation in the literature in terms of execution time and consumes nearly one-fourth of the estimated area in other implementations(namely [6]). In terms of area-time product, our Tate pairing accelerator with fixed multiplier is 12 times better than the one in [6] and the one with generic multiplier is 9 times better than the same implementation. In addition to these, our hardware implementation shortens the calculation time nearly sixteen times compared to software implementation reported in [17].

ASIC Implementation: We also synthesized VHDL codes of the algorithm for 0.25 μm CMOS technology. The total cell area is 4.3mm^2 excluding the buffers that are needed to satisfy the clock tree and static timing analysis specifications. The implementation consumes around 10 mm^2 chip area after routing with 5 metal technology. Our ASIC implementation has reached the frequency of 78 MHz and completes the pairing in 250 μs. We should note here that Virtex-2 devices are based on 90 nm CMOS technology with 9 metal routing layers.

6 Conclusion

We first developed the sub-blocks for arithmetic in GF(3^m) needed for the operations in characteristic three. After achieving good results in sub-blocks, we implemented GF(3^{6m}) arithmetic using them in parallel manner. Finally we developed our accelerator that computes the Tate Pairing in 250 μs by using commonly available FPGA. The implementation results showed that final area-time product of the design is 12 times better than the estimated values given in a previous implementation of the same algorithm in the literature and total computation time is 16 times better than the software applications. Another advantage of our accelerator is its suitability for ASIC implementation since no flip-flops are used in cubing and addition blocks and there is limited usage in the multiplier blocks. Based on these assumptions we build the first ASIC implementation of the modified Duursma Lee algorithm using 0.25 μm CMOS technology and presented the results of the area and operating frequency of the circuit. For future work, we plan to build a crypto coprocessor based on our Tate pairing implementation.

References

[1] P. Grabher and D. Page. *Hardware Acceleration of the Tate Pairing in Characteristic Three*. CHES 2005, LNCS 3659, pp. 398–411, 2005. International Association for Cryptologic Research 2005

[2] A. Shamir. *Identity-Based Cryptosystems and Signature Schemes*. In Advances in Cryptology (CRYPTO), Springer-Verlag LNCS 196, 47–53, 1985.

[3] D. Boneh and M. Franklin. *Identity-Based Encryption from the Weil Pairing*. In SIAM Journal on Computing, 32(3), 586–615, 2003.

[4] R. Sakai, K. Ohgishi and M. Kasahara. *Cryptosystems Based on Pairings*. In Symposium on Cryptography and Information Security (SCIS), 2000.

[5] G. Frey and H. Rück. *A remark considering m-divisibility in the divisor class group of curves*. Mathematics of Computation, 62:865-874, 1994.

[6] T. Kerins, W. P. Marnane, E. M. Popovici and P.S.L.M. Barreto. *Efficient Hardware for the Tate Pairing Calculation in Characteristic Three*. CHES 2005, LNCS 3659, pp. 412–426, 2005.International Association for Cryptologic Research 2005

[7] V. S. Miller. *Short Programs for functions on curves*. Unpublished manuscript.1986. http://crypto.stanford.edu/miller/miller.pdf.

[8] 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, volume LNCS 2442, pages 354–368. Springer-Verlag, 2002.

[9] S. Galbraith, K. Harrison and D. Soldera. *Implementing the Tate pairing*. In Algorithm Number Theory Symposium - ANTS V, vol 2369 of Lecture Notes in Computer Science, pages 324-337. Springer-Verlag 2002.

[10] R. Ronan, C. Eigeartaigh, C. Murphy, T. Kerins and M. Barreto. *Hardware implementation of the ηT pairing in characteristic 3*.

[11] 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.

[12] 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.

[13] D. Page and N.P. Smart. *Hardware implementation of finite fields of characteristic three*. In B. S. Kaliski, Jr., C¸ . K. Koc, and C. Paar, editors, Cryptographic Hardware and Embedded Systems — CHES 2002, volume LNCS. Springer-Verlag, 2002. 160, 161, 170, 172

[14] T. Kerins, E. Popovici, and W. Marnane. *Algorithms and Architectures for Use in FPGA Implementations of Identity Based Encryption Schemes*. FPL 2004, LNCS 3203, pp. 74–83, 2004.Springer-Verlag Berlin Heidelberg 2004

[15] G. Bertoni, J. Guajardo, S. Kumar, G. Orlando, C. Paar, and T. Wollinger. *Efficient* GF(p^m) *Arithmetic Architectures for Cryptographic Applications*. CT-RSA 2003, LNCS 2612, pp. 158–175, 2003. Springer-Verlag Berlin Heidelberg 2003

[16] 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.

[17] P. S. L. M. Barreto. *A note on efficient computation of cube roots in characteristic* 3. Cryptology ePrint Archieve, Report 035/2004 2004. http://eprint.iacr.org/2004/375

[18] J.-L. Beuchat, M. Shirase, T. Takagi and E. Okamoto, *An Algorithm for the ηT Pairing Calculation in Characteristic Three and its Hardware Implementation,* available at http://eprint.iacr.org/2006/.