EFFICIENT FPGA-BASED MULTIPLIERS FOR $\mathbb{F}_{3^{97}}$ AND $\mathbb{F}_{3^{6\cdot 97}}$

Jamshid Shokrollahi [*]
B-IT
University of Bonn
Germany
jamshid@bit.uni-bonn.de

Elisa Gorla [†]
Department of Mathematics
University of Zürich
Switzerland
elisa.gorla@math.unizh.ch

Christoph Puttmann[*]
Heinz Nixdorf Institute
University of Paderborn
Germany
puttmann@hni.upb.de

ABSTRACT

In this work we present a new structure for multiplication in finite fields. This structure is based on a digit-level LFSR (Linear Feedback Shift Register) multiplier in which the area of digit-multipliers are reduced using the Karatsuba method. We compare our results with the other works of the literature for $\mathbb{F}_{3^{97}}$. We also propose new formulas for multiplication in $\mathbb{F}_{3^{6\cdot 97}}$. These new formulas reduce the number of $\mathbb{F}_{3^{97}}$-multiplications from 18 to 15. The finite fields $\mathbb{F}_{3^{97}}$ and $\mathbb{F}_{3^{6\cdot 97}}$ are important fields for pairing based cryptography.

Keywords: finite field multiplication, FPGA, pairing based cryptography.

1. INTRODUCTION

Efficient multiplication in finite fields is a central task in the implementation of most public key cryptosystems. A great amount of work has been devoted to this topic (see [1] or [2] for a comprehensive list). The two types of finite fields which are mostly used in cryptographic standards are binary finite fields of type $\mathbb{F}_{2^m}$ and prime fields of type $\mathbb{F}_p$, where p is a prime (cf. [3]). Efforts to efficiently fit finite field arithmetic into commercial processors resulted into applications of medium characteristic finite fields like those reported in [4] and [5]. Medium characteristic finite fields are fields of type $\mathbb{F}_{p^m}$, where p is a prime slightly smaller than the word size of the processor, and has a special form that simplifies the modular reduction. Mersenne prime numbers constitute an example of primes which are used in this context. The security parameter is given by the length of the binary representations of the field elements, and the extension degree m is selected appropriately. Due to security considerations, the extension degree for fields of characteristic 2 or medium characteristic is usually chosen to be prime.

With the introduction of the method of Duursma and Lee for the computation of the Tate pairing (cf. [6]), fields of type $\mathbb{F}_{3^m}$ for m prime have attracted special attention. Computing the Tate pairing on elliptic curves defined over $\mathbb{F}_{3^m}$ requires computations both in $\mathbb{F}_{3^m}$ and in $\mathbb{F}_{3^{6m}}$. In [7] calculations are implemented using the tower of extensions

$$\mathbb{F}_{3^m} \subset \mathbb{F}_{3^{2m}} \subset \mathbb{F}_{3^{6m}}$$

and the inherent parallelism of multiplication in extension fields is used to accelerate the operations. Hardware designs and especially FPGA-based ones are suitable platforms for parallel implementation of algorithms. In that work multiplications in the first and the second field extensions are computed via 3 and 6 multiplications in the ground fields, respectively, requiring 18 multiplications in $\mathbb{F}_{3^{97}}$.

In our current work, which is mostly based on [7], on the one hand, we use asymptotically fast methods to improve the performance of multiplication in $\mathbb{F}_{3^{97}}$, and on the other hand, we propose new multiplication formulas to speedup multiplication in $\mathbb{F}_{3^{6\cdot 97}}$. Using the new formulas, multiplication in $\mathbb{F}_{3^{6\cdot 97}}$ is done with only 15 multiplications instead of 18. We use the same extension tower, using 3 multiplications in $\mathbb{F}_{3^{97}}$ to multiply elements in $\mathbb{F}_{3^{2\cdot 97}}$, but only 5 multiplications in $\mathbb{F}_{3^{2\cdot 97}}$ for $\mathbb{F}_{3^{6\cdot 97}}$. Our proposed method has a slightly increased number of additions in comparison to the Karatsuba method. Notice however that a multiplication in $\mathbb{F}_{3^{97}}$ requires many more resources than an addition, therefore the overall resource consumption will be reduced. The details of our method to generate the new formulas have been omitted to limit the complexity and diversity of materials in this paper, and have been submitted as another paper for CHES 2007.

A consistent amount of work has been done on hardware-based multiplication in finite fields, especially those of characteristic 3. The authors of [8] propose a least significant digit-element (LSDE) multiplier for $\mathbb{F}_{3^m}$. This multiplier divides the input polynomials into digits of length D. Whereas the digits of one input polynomial are processed in parallel, the digits of the other input polynomial are handled serially. Then the result is reduced modulo the irreducible polynomial. The same structure has also been used in [7] for multiplication in $\mathbb{F}_{3^{97}}$. Our multiplier, on the other hand, is based

[*]Partially funded by the German Research Foundation (DFG) under project RU 477/8

[†]Partially funded by the Swiss National Science Foundation under grant no. 107887

on the digit-serial implementation of LFSR (Linear Feedback Shift Register) multiplier which is widely used in the literature (see [9] or [10]), and performs the modular reduction during the multiplication. The first contribution of our current work is the application of the Karatsuba multiplier inside the digit-multipliers, which results in smaller area for these multipliers. Our results demonstrate the efficiency of this design compared to other works. The second contribution is the application of a method using only 5 multiplications in $\mathbb{F}_{3^{2\cdot 97}}$ for multiplication in $\mathbb{F}_{3^{6\cdot 97}}$. This results in an area-saving of almost 17% compared to the Karatsuba method which is used in [7].

Our work is organized as follows. Section 2 is devoted to the general structure of our multiplier for $\mathbb{F}_{3^{97}}$. In Section 3 we describe some improvements on the traditional LFSR multiplier and compare our results with other works from the literature. In Section 4 the new formulas for $\mathbb{F}_{3^{6\cdot 97}}$ together with suggestions for a new multiplier are presented, and Section 5 concludes the paper.

2. MULTIPLICATION IN $\mathbb{F}_{3^{97}}$

The finite field $\mathbb{F}_{3^{97}}$ can be represented as a vector space over $\mathbb{F}_3$. In this representation, elements of $\mathbb{F}_{3^{97}}$ are vectors of length 97 over $\mathbb{F}_3$. Addition of elements is computed by adding corresponding vectors. Multiplication is more complicated, and depends on the selected basis for $\mathbb{F}_{3^{97}}$. There are two popular bases which are used often in the literature, namely polynomial and normal bases. A polynomial basis is generally more suitable for multiplication, hence we choose this basis in our work.

In the polynomial basis, elements of $\mathbb{F}_{3^{97}}$ are represented as polynomials of degree at most 96 over $\mathbb{F}_3$. Two elements are added by adding of the corresponding polynomials. Multiplication is based on polynomial multiplication followed by reduction modulo the irreducible polynomial, which generates the polynomial basis. In our case the irreducible polynomial, which we denote by $f(x)$, is

$$x^{97} + x^{16} + 2. \tag{1}$$

In the next sections we show the details of polynomial arithmetic in our designs.

2.1. Arithmetic in $\mathbb{F}_3$

The element $a \in \mathbb{F}_3$ is represented using the vector (a_1, a_0) of two bits such that the elements 0, 1, and 2 are $(0,0)$, $(0,1)$, $(1,0)$, respectively. In this representation the operations addition, multiplication, and negation (multiplication by 2) are done, as shown in [11], using Equations 2, 3, and 4, respectively.

$$(a_1, a_0) + (b_1, b_0) = ((a_0 \vee b_0) \oplus t, (a_1 \vee b_1) \oplus t), \tag{2}$$
where $t = (a_0 \vee b_1) \oplus (a_1 \vee b_0)$

$$(a_1, a_0) \cdot (b_1, b_0) = ((a_1 \wedge b_0) \vee (a_0 \wedge b_1), \tag{3}$$
$$(a_0 \wedge b_0) \vee (a_1 \wedge b_1),$$

$$-(a_1, a_0) = (a_0, a_1). \tag{4}$$

The implementation of Equations 2 and 3 is done using 2 LUTs in the FPGA, whereas (4) is only a permutation of bits.

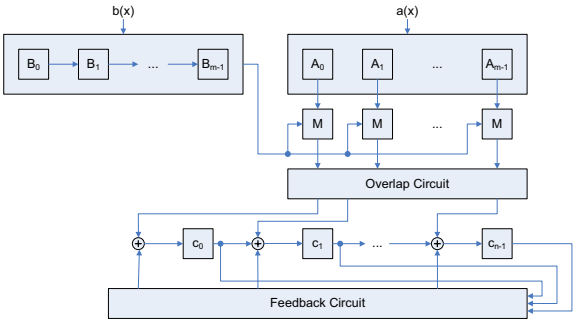

Fig. 1. Structure of a digit-level LFSR multiplier

2.2. Structure of the multiplier for $\mathbb{F}_{3^{97}}$

The structure of a digit-level LFSR multiplier is shown in Figure 1. In this figure the two input polynomials $a(x)$, and $b(x)$ are loaded into registers A and B, respectively, and divided into digits of length D. In each clock cycle the most significant digit of B is multiplied by the words of A, through digit-multipliers specified by M, and added to the content of the register in the feedback circuit. Inputs to the digit multipliers are two polynomials of degree $D-1$ in x. The product is a polynomial of degree $2(D-1)$. Powers x^D to $x^{2(D-1)}$ of each multiplier must be added to the powers x^0 to x^{D-2} of the next multiplier. This is done by the overlap circuit. In each clock cycle the register B and LFSR are shifted by D bits to the right. Shifting LFSR to right is equivalent to multiplication by x^D which generates the powers x^{97} to x^{96+D}. These powers are reduced modulo $f(x)$ of (1) using the feedback circuit. The name Linear Feedback Shift Register descends from these feedback structures. For more information about the digit-level LFSR multiplier and its costs for classical methods see [10]. In the next section we discuss our improvements to the traditional LFSR multiplier.

3. THE KARATSUBA METHOD

In this section we use asymptotically fast methods to reduce the size of digit-multipliers. We use a similar approach to [12] and combine the classical and the Karatsuba methods to build small digit-multipliers. Two linear polynomials $a_1x + a_0$ and $b_1x + b_0$ are multiplied classically using the formula

$$a_1b_1x^2 + (a_1b_0 + a_0b_1)x + a_0b_0 \quad (5)$$

with 4 multiplications and 1 addition. The same product can also be computed via

$$a_1b_1x^2 + ((a_1 + a_0)(b_1 + b_0) - a_1b_1 - a_0b_0)x + a_0b_0. \quad (6)$$

The new formula is called the Karatsuba method (see [13]). It requires 7 operations instead of 5, but only 3 multiplications, and uses fewer resources when the coefficients a_0, a_1, b_0, b_1 are replaced by polynomials. The classical method for multiplication of two polynomials of degree $n-1$ requires $O(n^2)$ operations. Recursive application of the Karatsuba method reduces the cost of a multiplication to $O(n^{1.59})$ operations. We represent the classical multiplication of two polynomials of degree $n-1$ by $\mathcal{C}_n$ and the method of (6) by $\mathcal{K}$. The methods $\mathcal{C}_n$ for $n \in \mathbb{N}$, and $\mathcal{K}$ constitute a set of polynomial multiplication methods. We call this set T. Using the elements of T we define the set of recursive multiplication methods T^* which contains the elements of T and all recursive combinations of elements of T^*. The recursive combination of the two methods $\mathcal{M}$ and $\mathcal{N}$, for polynomials of lengths m and n, respectively, is the multiplication method $\mathcal{MN}$ for polynomials of length mn. Let

$$a(x) = a_{mn-1}x^{mn-1} + \cdots + a_0, \text{ and}$$
$$b(x) = b_{mn-1}x^{mn-1} + \cdots + b_0$$

be given polynomials. In order to apply $\mathcal{MN}$, we write these polynomials as

$$a(x) = A_{m-1}X^{m-1} + \cdots + A_0, \text{ and}$$
$$b(x) = B_{m-1}X^{m-1} + \cdots + B_0,$$

where $X = x^n$ and $A_0, \cdots A_{m-1}, B_0, \cdots B_{m-1}$ are polynomials of degree $n-1$. If the polynomials A_i and B_i were coefficients, the two polynomials $a(x)$ and $b(x)$ would be multiplied using $\mathcal{M}$. The product using the method $\mathcal{MN}$ consists of several multiplications of the polynomials A_i and B_i, which are performed using $\mathcal{N}$. We implement the digit-multipliers using the elements of T^* to reduce their size. Our approach is similar to [12].

In Table 1 we show the results of implementing $\mathbb{F}_{3^{97}}$ multipliers on a XC2VP20-6FF896 FPGA. In this table the first column is the digit-size D. In a digit-level multiplier with digit-size D, inputs are preceded by enough zeros so that their length becomes a multiple of D. Hence it is natural to choose a value of D such that the difference $\lceil m/D \rceil - m/D$ is as small as possible. Our values for D are selected using this criteria and hence differ from other standard values like multiples of 4 in other works (see [8] and [7]). The string in the second column shows the recursive combination of the Karatsuba and classical methods which is applied. It is important to notice that the method $\mathcal{KC}_2$, which we used for polynomials of degree 6, applies to polynomials of length 7. Therefore, we add a zero in front of the polynomial and then remove all the gates containing an operation with the coefficients which are known to be zero. Hence this multiplier requires fewer resources than a complete $\mathcal{KC}_2$. This point distinguishes our approach from that in [12]. In the third, fourth, and fifth columns are the number of slices, maximum working frequency of the multiplier, and the required clock cycles for our designs.

Table 1. Timing and area costs of digit-level LFSR multipliers in $\mathbb{F}_{3^{97}}$ for different values of digit-size D

D	**Multiplication**	**# of slices**	**Maximum frequency (MHz)**	**# of clock cycles** = $\lceil 97/D \rceil$
1	–	327	300	97
2	$\mathcal{C}_2$	800	174	49
4	$\mathcal{C}_4$	1716	125	25
7	$\mathcal{KC}_4$	2954	111	14
14	$\mathcal{KKC}_4$	4006	72	7

The results of comparing our results with those in [7] are shown in Figure 2. Different digit-levels result in different circuits, which we compare with respect to both time and area. Area is the number of slices, whereas time is the product of clock cycles and minimum period. Both designs are on the same technology, but the speed grade of the FPGA in [7] is not available. As it is shown, our designs have better area-time performance. These improvements result, on the one hand, by using asymptotically faster methods, and on the other hand, by integrating the modular reduction stage into the LFSR. When a small digit-serial multiplier is used even the small size of a modular reduction must be taken into account.

4. MULTIPLICATION IN $\mathbb{F}_{3^{6\cdot 97}}$

Multiplication in $\mathbb{F}_{3^{6\cdot 97}}$ is done in the same way as in [7], as a tower of extensions of degrees 2 and 3, i.e.

$$\begin{aligned} \mathbb{F}_{3^{97}} &\cong \mathbb{F}_3/(x^{97} + x^{16} + 2) \\ \mathbb{F}_{3^{2\cdot 97}} &\cong \mathbb{F}_{3^{97}}/(y^2 + 1) \\ \mathbb{F}_{3^{6\cdot 97}} &\cong \mathbb{F}_{3^{2\cdot 97}}/(z^3 - z - 1). \end{aligned}$$

The elements of $\mathbb{F}_{3^{2\cdot 97}}$ are polynomials of degree 1 in s over $\mathbb{F}_{3^{97}}$, for s a root of $y^2 + 1$ in $\mathbb{F}_{3^{2\cdot 97}}$. The polynomials are multiplied by applying (6) and then reduced modulo $s^2 + 1$. The elements of $\mathbb{F}_{3^{6\cdot 97}}$ are polynomials of degree 3 in r, a root of $z^3 - z - 1$ in $\mathbb{F}_{3^{6\cdot 97}}$. They are multiplied using the

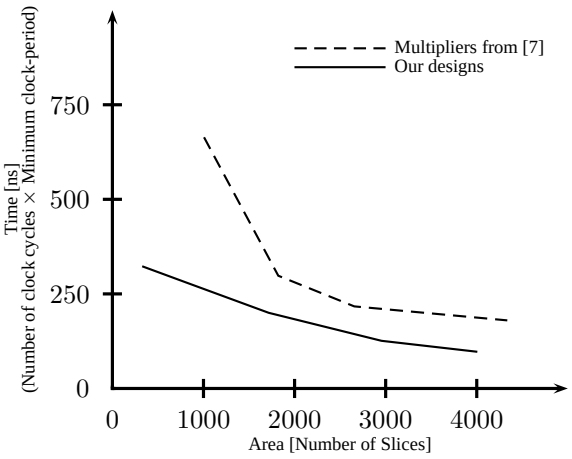

Fig. 2. Time vs. area comparisons of our multipliers with those in [7]

formulas (7) and then reduced modulo $r^3 - r - 1$.

$$
\begin{aligned}
&(a_0 + a_1 r + a_2 r^2)(b_0 + b_1 r + b_2 r^2) = \\
&c_0 + c_1 r + c_2 r^2 + c_3 r^3 + c_4 r^4, \text{where} \\
&P_0 = (a_0 + a_1 + a_2)(b_0 + b_1 + b_2) \\
&P_1 = (a_0 + s a_1 - a_2)(b_0 + s b_1 - b_2) \\
&P_2 = (a_0 - a_1 + a_2)(b_0 - b_1 + b_2) \\
&P_3 = (a_0 - s a_1 - a_2)(b_0 - s b_1 - b_2) \\
&P_4 = a_2 b_2, \text{ and} \\
&c_0 = P_0 + P_1 + P_2 + P_3 - P_4 \\
&c_1 = P_0 - s P_1 - P_2 + s P_3 \\
&c_2 = P_0 - P_1 + P_2 - P_3 \\
&c_3 = P_0 + s P_1 - P_2 - s P_3 \\
&c_4 = P_4,
\end{aligned}
\tag{7}
$$

Combining (6), (7) we have the following theorem.

Theorem 1 *Let $\alpha, \beta \in \mathbb{F}_{3^{6 \cdot 97}}$ be given as:*

$$
\begin{aligned}
\alpha =& a_0 + a_1 s + a_2 r + a_3 rs + a_4 r^2 + a_5 r^2 s \\
\beta =& b_0 + b_1 s + b_2 r + b_3 rs + b_4 r^2 + b_5 r^2 s.
\end{aligned}
$$

Let further their product $\gamma = \alpha\beta \in \mathbb{F}_{3^{6 \cdot 97}}$ be

$$
\gamma = c_0 + c_1 s + c_2 r + c_3 rs + c_4 r^2 + c_5 r^2 s.
$$

Then the coefficients $c_0 \cdots c_5$ of the product can be computed using only 15 multiplications in $\mathbb{F}_{3^{97}}$.

Closed-form formulas for this multiplication are shown in Appendix A. Scalar multiplications are particularly simple using these formulas. Scalar multiplications are multiplications by -1, s, and $-s$. Negation of coefficients and consequently of polynomials is only a permutation of bits, as seen in Section 2. Indeed multiplication of an element in $\mathbb{F}_{3^{2 \cdot 97}}$ by s is a permutation, too. Let $\alpha = a_1 s + a_0 \in \mathbb{F}_{3^{2 \cdot 97}}$, then

$$
s\alpha = a_1 s^2 + a_0 s \mod s^2 + 1 = a_0 s - a_1.
$$

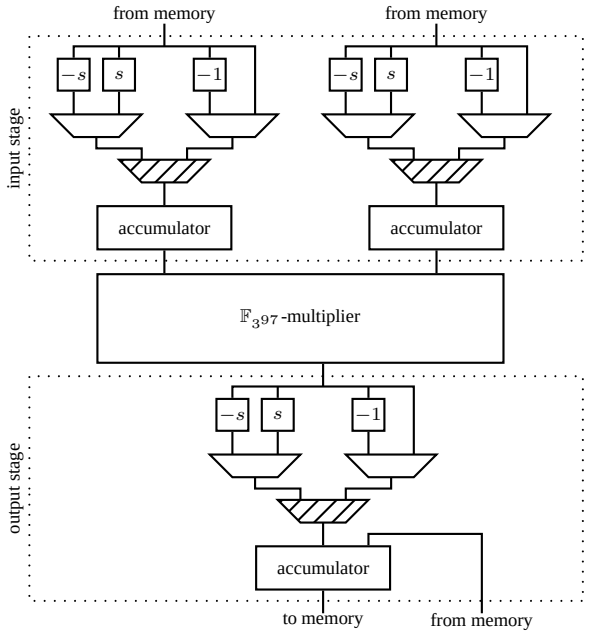

Fig. 3. The proposed structure block for implementing the formulas of Appendix A

All of the $\mathbb{F}_{3^{97}}$-multiplications can be done in parallel. This property allows designers to implement as many of these multipliers as possible, according to their time-area constraints. On the other hand, these multipliers are used for other computations such as point addition and doubling on elliptic curves for pairing-based cryptography. Reading and writing intermediate values into register files in such applications is time-consuming. To solve this problem we propose a new multiplier which is shown in Figure 3. The new multiplier consists of three pipeline stages, namely, input, multiplication, and output. During the time of each multiplication in $\mathbb{F}_{3^{97}}$, the input stage loads the coefficients a_i and b_i from memory for the next multiplication, and computes the linear combinations in (8) to compute P_is. In this time the output stage adds the last computed product P_i to memory variables according to (9). In this structure the hatched multiplexers can select either one of their inputs or the sum of the inputs. In this way all possible multiples of input polynomials can be selected and added to the accumulators.

5. CONCLUSION

In this paper we proposed a new structure for multiplication in $\mathbb{F}_{3^{97}}$. This structure is based on digit-level LFSR multipliers, where the area of digit-multipliers are reduced using the Karatsuba method. Another advantage of this approach is performing the modular reduction during the multiplication. Our synthesis results showed the performance improvement compared to other designs in the literature. We have also presented new formulas for multiplication in $\mathbb{F}_{3^{6\cdot 97}}$ using only 15 multiplications in $\mathbb{F}_{3^{97}}$. When the Karatsuba method is applied 18 multiplications are required. Furthermore, we have introduced a feasible hardware structure for realizing our proposed formulas. Our formulas are for the case that $\mathbb{F}_{3^{6\cdot 97}}$ is constructed from $\mathbb{F}_{3^{2\cdot 97}}$ using the irreducible polynomial $z^3 - z - 1$. In case that the finite field is constructed using $z^3 - z + 1$, the formulas require slight modifications.

6. REFERENCES

[1] D. E. Knuth, *The Art of Computer Programming, vol. 2, Seminumerical Algorithms*, 3rd ed. Reading MA: Addison-Wesley, 1998, first edition 1969.

[2] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, 2nd ed. Cambridge, UK: Cambridge University Press, 2003, first edition 1999.

[3] *Digital Signature Standard (DSS)*, U.S. Department of Commerce / National Institute of Standards and Technology, January 2000, federal Information Processings Standards Publication 186-2. [Online]. Available: http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf

[4] D. V. Bailey and C. Paar, "Optimal extension fields for fast arithmetic in public-key algorithms," in *Advances in Cryptology: Proceedings of CRYPTO '98*, Santa Barbara CA, ser. Lecture Notes in Computer Science, H. Krawczyk, Ed., no. 1462. Springer-Verlag, 1998, pp. 472–485.

[5] R. M. Avanzi and P. Mihăilescu, "Generic efficient arithmetic algorithms for PAFFs (processor adequate finite fields) and related algebraic structures (extended abstract)," in *Selected Areas in Cryptography (SAC 2003)*. Springer-Verlag, 2003, pp. 320–334.

[6] I. Duursma and H. Lee, "Tate-pairing implementations for tripartite key agreement." [Online]. Available: citeseer.ist.psu.edu/duursma03tatepairing.html

[7] T. Kerins, W. P. Marnane, E. M. Popovici, and P. S. L. M. Barreto, "Efficient hardware for the tate pairing calculation in characteristic three," in *Cryptographic Hardware and Embedded Systems, CHES2005*, ser. Lecture Notes in Computer Science, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer-Verlag, 2005, pp. 412–426.

[8] 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: the Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13–17, 2003: Proceedings*, ser. Lecture Notes in Computer Science, M. Joye, Ed., vol. 2612. Springer-Verlag, 2003, pp. 158–175.

[9] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*. Kluwer Academic Publishers, 1987.

[10] J. Shokrollahi, "Efficient implementation of elliptic curve cryptography on fpgas," Ph.D. dissertation, Bonn University, Bonn, December 2006. [Online]. Available: http://hss.ulb.uni-bonn.de/diss_online/math_nat_fak/2007/shokrollahi_jamshid/

[11] R. Granger, D. Page, and M. Stam, "Hardware and software normal basis arithmetic for pairing-based cryptogaphy in characteristic three," *IEEE Transactions on Computers*, vol. 54, no. 7, pp. 852–860, 2005. [Online]. Available: http://

[12] J. von zur Gathen and J. Shokrollahi, "Fast arithmetic for polynomials over $\mathbb{F}_2$ in hardware," in *IEEE Information Theory Workshop (2006)*. Punta del Este, Uruguay: IEEE, March 2006, pp. 107–111.

[13] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Soviet Physics–Doklady*, vol. 7, no. 7, pp. 595–596, January 1963, translated from Doklady Akademii Nauk SSSR, Vol. 145, No. 2, pp. 293–294, July, 1962.

A. MULTIPLICATION FORMULAS FOR $\mathbb{F}_{3^{6\cdot 97}}$

Let $\alpha, \beta \in \mathbb{F}_{3^{6\cdot 97}}$ be given as:

$$\begin{aligned}\alpha &= a_0 + a_1 s + a_2 r + a_3 rs + a_4 r^2 + a_5 r^2 s,\\ \beta &= b_0 + b_1 s + b_2 r + b_3 rs + b_4 r^2 + b_5 r^2 s,\end{aligned}$$

where $a_0, \cdots, b_5 \in \mathbb{F}_{3^{97}}$ and $s \in \mathbb{F}_{3^{2\cdot 97}}$, $r \in \mathbb{F}_{3^{6\cdot 97}}$ are roots of $y^2 + 1$ and $z^3 - z - 1$, respectively. Let their product $\gamma = \alpha\beta \in \mathbb{F}_{3^{6\cdot 97}}$ be

$$\gamma = c_0 + c_1 s + c_2 r + c_3 rs + c_4 r^2 + c_5 r^2 s.$$

Then the coefficients $c_0 \cdots c_5 \in \mathbb{F}_{3^{97}}$ of the product can be computed using the following formulas.

$$
\begin{aligned}
P_0 &= (a_0 + a_2 + a_4)(b0 + b2 + b4) \\
P_1 &= (a_0 + a_1 + a_2 + a_3 + a_4 + a_5) \\
&\quad (b_0 + b_1 + b_2 + b_3 + b_4 + b_5) \\
P_2 &= (a_1 + a_3 + a_5)(b_1 + b_3 + b_5) \\
P_3 &= (a_0 + sa_2 - a_4)(b_0 + sb_2 - b_4) \\
P_4 &= (a_0 + a_1 + sa_2 + sa_3 - a_4 - a_5) \\
&\quad (b_0 + b_1 + sb_2 + sb_3 - b_4 - b_5) \\
P_5 &= (a_1 + sa_3 - a_5)(b_1 + sb_3 - b_5) \\
P_6 &= (a_0 - a_2 + a_4)(b_0 - b_2 + b_4) \\
P_7 &= (a_0 + a_1 - a_2 - a_3 + a_4 + a_5) \\
&\quad (b_0 + b_1 - b_2 - b_3 + b_4 + b_5) \\
P_8 &= (a_1 - a_3 + a_5)(b_1 - b_3 + b_5) \\
P_9 &= (a_0 - sa_2 - a_4)(b_0 - sb_2 - b_4) \\
P_{10} &= (a_0 + a_1 - sa_2 - sa_3 - a_4 - a_5) \\
&\quad (b_0 + b_1 - sb_2 - sb_3 - b_4 - b_5) \\
P_{11} &= (a_1 - sa_3 - a_5)(b_1 - sb_3 - b_5) \\
P_{12} &= a_4 b_4 \\
P_{13} &= (a_4 + a_5)(b_4 + b_5) \\
P_{14} &= a_5 b_5
\end{aligned}
\tag{8}
$$

$$
\begin{aligned}
c_0 &= -P_0 + P_2 + (s+1)P_3 - (s+1)P_5 - \\
&\quad (s-1)P_9 + (s-1)P_{11} - P_{12} + P_{14} \\
c_1 &= P_0 - P_1 + P_2 - (s+1)P_3 + (s+1)P_4 - \\
&\quad (s+1)P_5 + (s-1)P_9 - (s-1)P_{10} + \\
&\quad (s-1)P_{11} - P_{12} - P_{13} + P_{14} \\
c_2 &= -P_0 + P_2 + P_6 - P_8 + P_{12} - P_{14} \\
c_3 &= P_0 - P_1 + P_2 - P_6 + P_7 - P_8 - P_{12} \\
&\quad + P_{13} - P_{14} \\
c_4 &= P_0 - P_2 - P_3 + P_5 + P_6 - P_8 - P_9 + P_{11} + \\
&\quad P_{12} - P_{14} \\
c_5 &= P_0 + P_1 - P_2 + P_3 - P_4 + P_5 - P_6 + P_7 - \\
&\quad P_8 + P_9 - P_{10} + P_{11} - P_{12} + P_{13} - P_{14}
\end{aligned}
\tag{9}
$$