International Journal of Security and Its Applications
Vol.9, No.9 (2015), pp.165-178
http://dx.doi.org/10.14257/ijsia.2015.9.9.15

Fast Three-Input Multipliers over Small Composite Fields for Multivariate Public Key Cryptography

Haibo Yi[1] and Weijian Li[2,*]

[1]*School of Computer Engineering, Shenzhen Polytechnic*
518055 Shenzhen, China
[2]*School of Computer Science, Guangdong Polytechnic Normal University*
510665 Guangzhou, China
[*]*Corresponding Author (E-mail): weijianlee@126.com*

Abstract

Since quantum computer attacks will be threats to the current public key cryptographic systems, there has been a growing interest in Multivariate Public Key Cryptography (MPKC), which has the potential to resist such attacks. Finite field multiplication is playing a crucial role in the implementations of multivariate cryptography and most of them use two-input multipliers. However, there exist multiple multiplications of three elements in multivariate cryptography. This motivates our work of designing three-input multipliers, which extend the improvements on multiplication of three elements in three directions. First, since multivariate cryptography can be implemented over small composite fields, our multipliers are designed over such fields. Second, since it requires multiplications of two and three elements, our multipliers can execute both of them. Third, our multipliers adapt table look-up and polynomial basis, since they are faster over specific fields, respectively. We demonstrate the improvement of our design mathematically. We implement our design on a Field-Programmable Gate Array (FPGA), which shows that our design is faster than other two-input multipliers when computing multiplication of three elements, e.g. multiplier with field size 256 is 28.4% faster. Our multipliers can accelerate multivariate cryptography and mathematical applications, e.g. TTS is 14% faster.

Keywords: *Three-input multiplier, Composite field, Finite field, Table look-up, Polynomial basis, Gaussian elimination, Solving systems of linear equations, Multivariate Public Key Cryptography (MPKC), Field-Programmable Gate Array (FPGA)*

1. Introduction

Quantum computers will break most of the current public key cryptographic systems, including RSA, DSA and ECDSA [1]. One of the countermeasures is Multivariate Public Key Cryptography (MPKC), which has the potential to resist quantum computer attacks [2]. The main strength is that its underlying mathematical problem is to solve a set of Multivariate Quadratic (MQ) polynomial equations over a finite field, which is proven to be an NP-hard problem [3].

During the past thirty years, multivariate cryptography has been one of the subjects of much researches in many areas. Multivariate signatures are claimed to be one of the fastest signatures, in particular, Rainbow [4] and TTS [5]. Multiple methodologies for accelerating the implementation of multivariate cryptography have been proposed due to its applicability in the area of engineering [6-14]. Among these methodologies, optimization of finite field multiplication is playing a crucial role since it is the most fundamental arithmetic in multivariate cryptography. Most of these implementations use two-input multipliers over a finite field, aside from a high-speed implementation of

ISSN: 1738-9976 IJSIA

Rainbow signature over $GF(2^8)$ [9]. However, there exist multiple multiplications of three elements in multivariate cryptography.

Our contributions. This motivates our work of designing three-input multipliers for multivariate cryptography. We propose novel multipliers that extends the improvements on the multiplication of three elements in three directions compared with the known results. First, since most of multivariate cryptographic schemes can be implemented over small composite fields $GF((2^n)^2)$, our multipliers can execute multiplications over such fields. Second, since multivariate cryptography requires multiplication of two elements and three elements, our multipliers can compute both of them. Third, since table look-up and polynomial basis methods have advantages of speed in different fields, respectively, our multipliers can perform multiplications based on these methods.

We test and verify our design on a Field-Programmable Gate Array (FPGA). Experimental results show that our multipliers are faster than other two-input multipliers when computing multiplication of three elements. We demonstrate the improvement of our multipliers mathematically. Moreover, they can speed up Gaussian elimination and implementations of multivariate cryptography. Besides, our multipliers can also be used in other mathematical and cryptographic applications.

Organization. The rest of this paper is organized as follows: In Section 2, we introduce multiplication of three elements and how they work in multivariate cryptography. In Section 3, three-input multipliers over small composite fields are proposed. In Section 4, we evaluate the performances of our multipliers by mathematical analysis. In Section 5, our multipliers are implemented on a FPGA and experimental results are analyzed. In Section 6, we compare our design with others multipliers. In Section 7, we present applications of our multipliers. In Section 8, conclusions and future improvements are discussed.

2. Preliminaries

2.1. Finite Field Multiplier

Existing two-input multipliers over finite fields are mainly designed on: polynomial basis, normal basis and the other basis. Besides, there has been a growing interest to implement arithmetic using composite field representations, which has been employed in many applications.

There are few three-input multipliers, aside from [9], which uses a three-input polynomial basis multiplier over $GF(2^8)$ to speed up Rainbow signature generations. Compared with our work in this paper, it can be only used in a specific application, while our work can be used in most of multivariate cryptographic systems and other PKCs.

2.2. Multivariate Public Key Cryptography

The core of multivariate cryptography is to evaluate multivariate polynomials over finite fields. Generally, multivariate polynomial consists of multiplication of two elements, multiplication of three elements and addition over a finite field. For example, multivariate polynomial in Rainbow signature scheme can be represented

$$\sum_{i\in O_l, j\in S_l} \alpha_{ij} x_i x_j + \sum_{i,j\in S_l} \beta_{ij} x_i x_j + \sum_{i\in S_{l+1}} \gamma_i x_i + \eta.$$

by the form

Some definitions of the coefficients are as follows. O_i is a set of Oil variables in the i-th layer; S_i is a set of Vinegar variables in the i-th layer. α , β and γ are coefficients; η is a constant.

Solving system of linear equations over finite fields is another part of multivariate cryptography, e.g. Gaussian elimination and Gauss-Jordan elimination. These methods consist of pivoting, normalization and elimination. Let a_{ij} be the element of i-th row and j-th column in a matrix and a_{ii} be the pivot element. Normalization includes a multiplication of two elements, which is $a_{it} = a_{ii}^{-1} a_{it}$. Elimination includes a multiplication of three elements, which is $a_{it} = a_{ii}^{-1} a_{it}$.

In sum, multiplications of two elements and three elements are both used in multivariate cryptography.

3. Design of Three-input Multipliers

3.1. Overview of our Multipliers

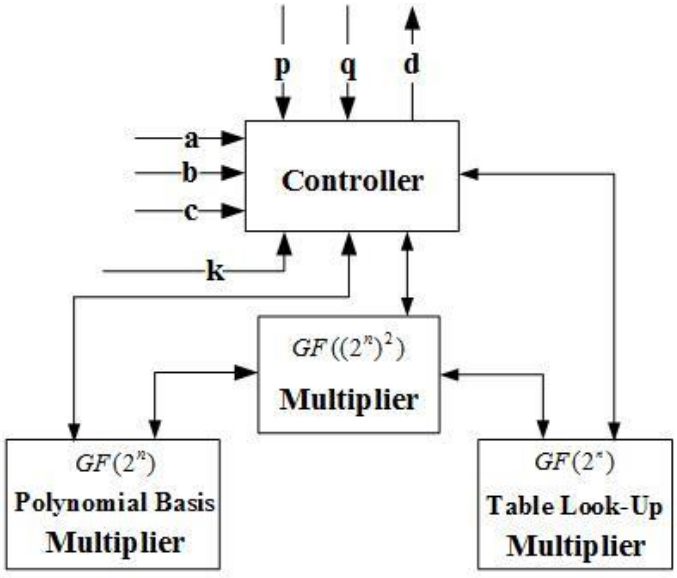

Figure 1. The Architecture of Our Multipliers

The architecture of our multipliers is depicted in Figure 1, which consists of four parts, i.e. controller, multiplier over $GF((2^n)^2)$, multiplier with polynomial basis over $GF(2^n)$ and multiplier with table look-up over $GF(2^n)$.

In our multipliers, $a \times b = d$, $a \times b \times c = d$ are performed, where a, b, c, d are elements in $GF((2^n)^2)$ and p, q are irreducible polynomials over $GF(2^n)$ and $GF((2^n)^2)$, respectively.

According to the values of k our multipliers can be reused for different applications, which is illustrated in Table 1. For example, when $k = (111)_2$, our multipliers execute multiplication of three elements by using table look-up over $GF((2^n)^2)$.

k is decoded by a 3-8 line decoder, where three bits of k are connected with S_1 , S_2 and S_3 of the decoder, respectively. D_1 , D_2 ,... D_8 are connected with other computational components.

3.2. Multiplier over $GF(2^n)$ on Polynomial Basis

Table 1. Our Multipliers for Different Parameters

Value of k	Finite Field	Number of Operands	Method of Multiplication
$(000)_2$	$GF(2^n)$	2	Polynomial basis
$(001)_2$	$GF(2^n)$	2	Table look-up
$(010)_2$	$GF(2^n)$	3	Polynomial basis
$(011)_2$	$GF(2^n)$	3	Table look-up
$(100)_2$	$GF((2^n)^2)$	2	Polynomial basis
$(101)_2$	$GF((2^n)^2)$	2	Table look-up
$(110)_2$	$GF((2^n)^2)$	3	Polynomial basis
$(111)_2$	$GF((2^n)^2)$	3	Table look-up

We adapt [15] to design multiplier over $GF(2^n)$ on polynomial basis. Field element is expressed in the polynomial form and field multiplication is performed in two steps. The first step is to perform the polynomial multiplication. The second step is to reduce modulo the irreducible polynomial p(x).

$$c(x) = a(x) \times b(x) (\mathrm{mod}\,(p(x))) = \sum_{i=0}^{n-1} c_i x^i$$

Let $a(x)$ and $b(x)$ be elements in $GF(2^n)$, and
be the expected multiplication result.

$$x^i \bmod p(x) = \sum_{j=0}^{n-1} v_{ij} x^j .$$

First, we compute v_{ij} for $i = 0,1,...,2(n-1)$ and $j = 0,1,...,n-1$ according to
Next, we compute S_i by AND logic gates for $i = 0,1,...,2(n-1)$ by

$$S_i = \sum_{j+k=i} a_j b_k .$$

After that, we compute c_i by XOR logic gates for $i = 0,1,...,n-1$ by

$$c_i = \sum_{j=0}^{2(n-1)} v_{ji} S_j .$$

Finally, the multiplication result is $c(x)$.

3.3. Multiplier over $GF(2^n)$ on Table Look-up

The number of elements in $GF(2^n)$ is 2^n. If α is chosen as the primitive element, all non-zero elements can be represented as a power of α. We store $k_i(x)$ in address i of the look-up table where

$$k_i(x) = \alpha^i \ mod \ p(x)$$

and p(x) is the irreducible polynomial over $GF(2^n)$.

Table 2 depicts a look-up table, where n=4 and the irreducible polynomial p(x) is $x^4 + x + 1$. The following are three examples for multiplications of two elements and

three elements using table look-up, respectively. Suppose that the element in $GF(2^4)$ is represented as $(xxxx)_2$, where $x \in GF(2)$, i.e. x=0 or x=1.

Table 2. Table Look-Up over $GF(2^4)$

Power Representation	Binary Representation
α^0	0001
α^1	0010
α^2	0100
α^3	1000
α^4	0011
α^5	0110
α^6	1100
α^7	1011
α^8	0101
α^9	1010
α^{10}	0111
α^{11}	1110
α^{12}	1111
α^{13}	1101
α^{14}	1001

Example 1. Compute $(1000)_2 \times (1100)_2$. Since $(1000)_2 = \alpha^3$ and $(1100)_2 = \alpha^6$, $(1000)_2 \times (1100)_2 = \alpha^3 \times \alpha^6 = \alpha^9$. By looking up Table 2, we have $\alpha^9 = (1010)_2$. Therefore, $(1000)_2 \times (1100)_2 = (1010)_2$.

Example 2. Compute $(1011)_2 \times (1101)_2$. Since $(1011)_2 = \alpha^7$ and $(1101)_2 = \alpha^{13}$, $(1011)_2 \times (1101)_2 = \alpha^7 \times \alpha^{13} = \alpha^{20} = \alpha^{20 \bmod 15} = \alpha^5$. By looking up Table 2, we have $\alpha^5 = (0110)_2$. Therefore, $(1011)_2 \times (1101)_2 = (0110)_2$.

Example 3. Compute $(1000)_2 \times (1011)_2 \times (1101)_2$. Since $(1000)_2 = \alpha^3$, $(1011)_2 = \alpha^7$ and $(1101)_2 = \alpha^{13}$, $(1000)_2 \times (1011)_2 \times (1101)_2 = \alpha^3 \times \alpha^7 \times \alpha^{13} = \alpha^{23} = \alpha^{23 \bmod 15} = \alpha^8$. By looking up Table 2, we have $\alpha^8 = (0101)_2$. Therefore, $(1000)_2 \times (1011)_2 \times (1101)_2 = (0101)_2$.

It can be observed from these examples that computing multiplication of three elements is very efficient when using table look-up.

3.4. Multiplier over $GF((2^n)^2)$

Multiplication of Two Elements. Let $a(x) = a_h x + a_l$ and $b(x) = b_h x + b_l$ be the elements in $GF((2^n)^2)$, where a_h , a_l , b_h , b_l are elements in $GF(2^n)$.

Then the multiplication of a(x) and b(x) can be expressed as

$$a(x) \times b(x) = (a_h x + a_l)(b_h x + b_l) = (a_h b_h x^2 + (a_h b_l + a_l b_h)x + a_l b_l) \bmod q(x).$$

We perform the polynomial multiplication and reduction module the irreducible polynomial $q(x) = x^2 + x + e$, where e is a constant in $GF(2^n)$. Then we have

$$c_h = (a_h + a_l)(b_h + b_l) + a_l b_l \ ,$$

$$c_l = a_l b_l + a_h b_h e \ .$$

It can be observed that the critical path of multiplication of two elements over $GF((2^n)^2)$ includes one multiplication, one constant multiplication and one addition over $GF(2^n)$.

Multiplication of Three Elements. Suppose that $a(x) = a_h x + a_l$, $b(x) = b_h x + b_l$ and $c(x) = c_h x + c_l$ are elements in $GF((2^n)^2)$. $d(x) = a(x) \times b(x) \times c(x) \bmod q(x)$ is computed through

$$d(x) = (d_h x + d_l) = (a_h b_h c_h x^3 + (a_h b_l c_h + a_l b_h c_h + a_h b_h c_l) x^2 + (a_l b_l c_h + a_h b_l c_l + a_l b_h c_l) x + a_l b_l c_l) \bmod q(x),$$

where

$$d_l = e(a_h b_h c_h + a_h b_l c_h + a_l b_h c_h + a_h b_h c_l) + a_l b_l c_l, d_h$$
$$= e a_h b_h c_h + a_h b_h c_h + a_h b_l c_h + a_l b_h c_h + a_h b_h c_l + a_l b_l c_h + a_h b_l c_l + a_l b_h c_l,$$
$$q(x) = x^2 + x + e, e \in GF(2^n).$$

Therefore, it can be observed that the critical path of multiplication of three elements over $GF((2^n)^2)$ includes one constant multiplication, one multiplication, and three additions over $GF(2^n)$.

4. Theoretical Evaluation of Performance

Multiplications of two elements and three elements are computed by invoking two-input multiplier (CM2) and three-input multiplier (CM3) over $GF((2^n)^2)$, respectively.

In the following, we prove invoking CM3 is faster than invoking CM2 when computing multiplication of three elements. We give some definitions at first. M_2 , M_3 , M_C and A stand for multiplication of two elements and three elements, constant multiplication and addition over $GF(2^n)$, respectively.

We give Lemma 1 from the view of critical path.

Lemma 1. The critical path by invoking CM3 is shorter than the one by invoking CM2 when computing multiplication of three elements.

First, the critical path of computing multiplication of two elements by invoking CM2 can be evaluated as follows,

$$T_2 = M_2 + M_C + A.$$

Second, the critical path of computing multiplication of three elements by invoking CM2 is double, i.e.

$$2T_2 = 2(M_2 + M_C + A).$$

Third, the critical path of computing multiplication of three elements by invoking CM3 can be evaluated as follows,

$$T_3 = M_3 + M_C + 3A.$$

The difference between $2T_2$ and T_3 is,

$$D = 2T_2 - T_3 = 2M_2 - M_3 + M_C - A.$$

Finally, we need to prove $2M_2 > M_3$ and $M_C > A$.

There are no overlaps in $a_0 \times a_1 = b_0$ and $a_2 \times a_3 = b_1$, where $a_0, a_1, a_2, a_3, b_0, b_1$ are elements over a finite field. On the other hand, there are overlaps in $a_4 \times a_5 \times a_6 = b_2$, where a_4, a_5, a_6, b_2 are elements over a finite field. Therefore, multiplication of three elements is faster than two multiplication of two elements and we have $2M_2 > M_3$. Since addition is much faster than constant multiplication over finite fields, we have $M_C >> A$.

Therefore, $D >> 0$ is proved. In other words, when computing multiplication of three elements, three-input multipliers are faster than two-input multipliers.

5. Implementation

5.1. Overview of the Implementation

We implement our multipliers on $n = 2,3,4,5,6,7,9,11,13,15$, respectively. Our multipliers can perform each multiplication over a finite field within one clock cycle. Our design is programmed in VHDL by using Quartus II, and implemented on a EP2S130F1020I4 FPGA device, which is a member of ALTERA Stratix II family.

Furthermore, this design can be generalized to cover other devices of FPGA. All the experimental results mentioned in the following are extracted after place and route.

5.2. Multiplication over $GF(2^n)$

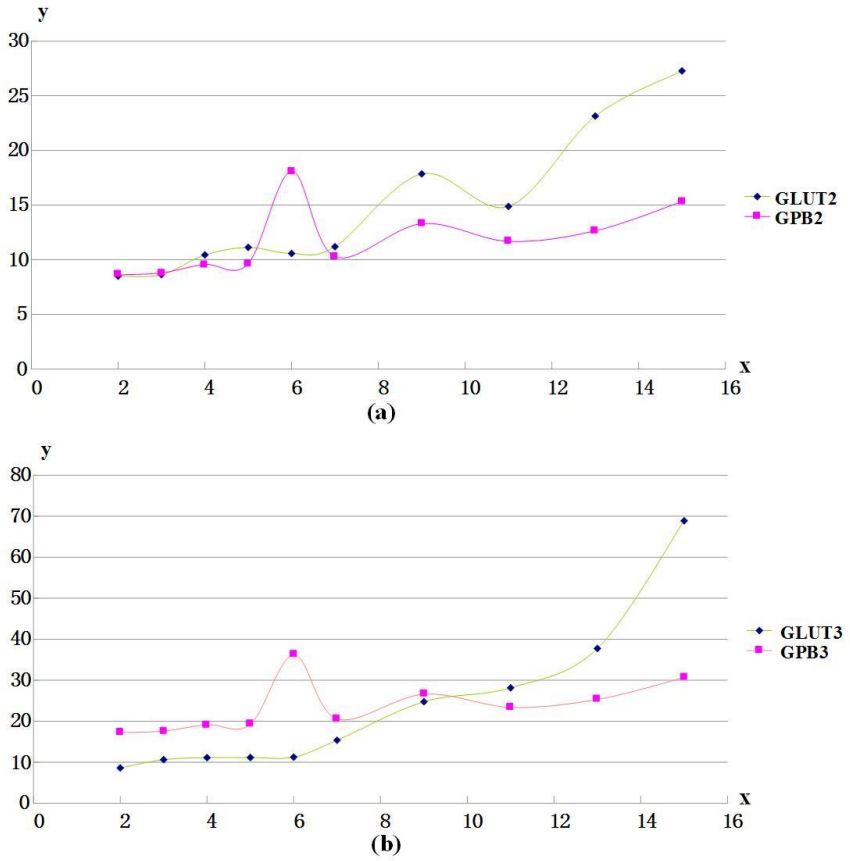

Figure 2. Comparison on Multiplications over $GF(2^n)$

Table 3 depicts the performance of our multipliers over $GF(2^n)$ for multiplication of two elements with table look-up (GLUT2) and polynomial basis (GPB2), multiplication of three elements with table look-up (GLUT3) and polynomial basis (GPB3), respectively.

It can be observed from Table 3 that if we choose GLUT2 over $GF(2^2)$, multiplication of three elements requires 2×8.505 ns. If we choose GLUT3 over $GF(2^2)$, it requires 8.591 ns. Since $2 \times 8.505 > 8.591$, GLUT3 is faster than GLUT2 for multiplication of three elements. If we choose GPB2 over $GF(2^2)$, multiplication of three elements requires 2×8.620 ns. If we choose GPB3

over $GF(2^2)$, it requires 17.238 ns. Since $2\times8.620>17.238$, GPB3 is faster than GPB2 for multiplication of three elements.

Table 3. Executing Time (ns) of Multiplications over $GF(2^n)$

Finite Field	p(x)	GLUT2	GPB2	GLUT3	GPB3
$GF(2^2)$	x^2+x+1	8.505	8.620	8.591	17.238
$GF(2^3)$	x^3+x+1	8.631	8.791	10.616	17.580
$GF(2^4)$	x^4+x+1	10.440	9.569	11.106	19.133
$GF(2^5)$	x^5+x^2+1	11.128	9.634	11.139	19.263
$GF(2^6)$	x^6+x+1	10.575	18.102	11.229	36.198
$GF(2^7)$	x^7+x+1	11.187	10.276	15.359	20.547
$GF(2^9)$	x^9+x^4+1	17.854	13.298	24.749	26.595
$GF(2^{11})$	$x^{11}+x^2+1$	14.879	11.681	28.117	23.360
$GF(2^{13})$	$x^{13}+x^4+x^3+x+1$	23.145	12.643	37.737	25.282
$GF(2^{15})$	$x^{15}+x^4+x^2+x+1$	27.275	15.322	68.881	30.640

The comparison on multiplications over $GF(2^n)$ is given in Figure 2, where (a) is multiplications of two elements by invoking GLUT2 and GPB2, respectively and (b) is multiplications of three elements by invoking GLUT3 and GPB3, respectively.

In Figure 2, x and y axes are n and executing time (ns) of multiplication, respectively. The curves of the figure show that multiplication on table look-up is faster than multiplication on polynomial basis when the field size is small and slower when the field size is large.

5.3. Multiplication over $GF((2^n)^2)$

Table 4 depicts the performance of our multipliers over $GF((2^n)^2)$ for multiplication of two elements with table look-up (LUT2) and polynomial basis (PB2), multiplication of three elements with table look-up (LUT3) and polynomial basis (PB3), respectively.

It can be observed from Table 4 that if we choose LUT2 over $GF((2^2)^2)$, multiplication of three elements requires 2×9.456 ns. If we choose LUT3 over $GF((2^2)^2)$, it requires 10.229 ns. Since $2\times9.456>10.229$, LUT3 is faster than LUT2 for multiplication of three elements. If we choose PB2 over $GF((2^2)^2)$, multiplication of three elements requires 2×12.608 ns. If we choose LUT3 over $GF((2^2)^2)$, it requires 13.530 ns. Since $2\times12.608>13.530$, PB3 is faster than PB2 for multiplication of three elements.

The comparison on the multiplications over $GF((2^n)^2)$ is given in Figure 3, where (a) is multiplications of two elements by invoking LUT2 and PB2, respectively and (b) is multiplications of three elements by invoking LUT3 and PB3, respectively.

Table 4. Executing Time (ns) of Multiplications over $GF((2^n)^2)$

Field	p(x)	q(x)	LUT2	PB2	LUT3	PB3
$(2^2)^2$	x^2+x+1	x^2+x+3	9.456	12.608	10.229	13.530
$(2^3)^2$	x^3+x+1	x^2+x+5	10.304	12.226	13.215	14.873
$(2^4)^2$	x^4+x+1	x^2+x+9	11.422	12.842	22.062	16.354
$(2^5)^2$	x^5+x^2+1	x^2+x+8	15.072	13.792	16.653	17.706
$(2^6)^2$	x^6+x+1	x^2+x+33	14.535	14.698	21.234	20.425
$(2^7)^2$	x^7+x+1	$x^2+x+113$	15.563	14.689	21.698	20.989
$(2^9)^2$	x^9+x^4+1	x^2+x+32	19.621	16.143	27.215	23.268
$(2^{11})^2$	$x^{11}+x^2+1$	$x^2+x+1367$	27.541	17.558	34.348	24.441
$(2^{13})^2$	$x^{13}+x^4+x^3+x+1$	$x^2+x+3085$	30.133	19.125	36.794	27.642
$(2^{15})^2$	$x^{15}+x^4+x^2+x+1$	$x^2+x+16395$	48.575	19.947	57.344	26.917

In Figure 3, x and y axes are n and executing time (ns) of multiplication, respectively. The curves of the figure show that multiplication on table look-up is faster than multiplication on polynomial basis when the field size is small and slower when the field size is large.

5.4. Comparison

Suppose that GLUT2, GPB2, GLUT3, GPB3 are performed over $GF(2^m)$ and LUT2, PB2, LUT3, PB3 are performed over $GF((2^{(m/2)})^2)$. The comparison on multiplications over $GF(2^m)$ and $GF((2^{(m/2)})^2)$ is given in Figure 4, where (a) is multiplications of two elements by invoking GLUT2, GPB2, LUT2, PB2, respectively and (b) is multiplications of three elements by invoking GLUT3, GPB3, LUT3, PB3, respectively.

In Figure 4, x and y axes are m and executing time (ns) of multiplication, respectively. The curves of the figure show that LUT2, LUT3 are faster than GLUT2, GLUT3 and GPB2, GPB3 are faster than PB2, PB3. In other words, table look-up and polynomial basis are more efficient over $GF((2^n)^2)$ and $GF(2^n)$, respectively.

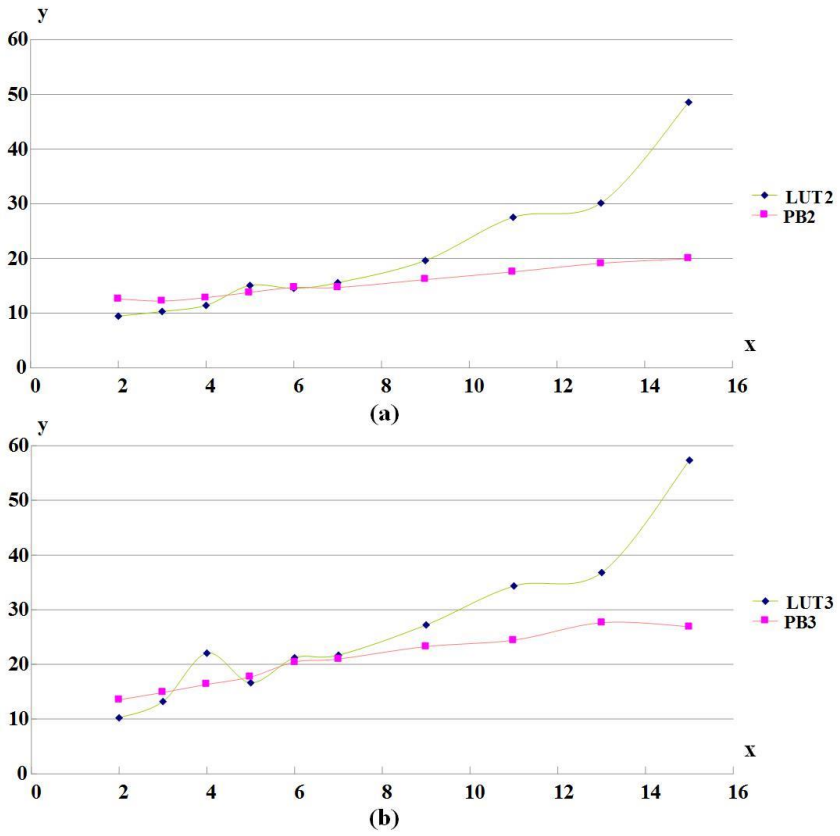

Figure 3. Comparison on Multiplications over $GF((2^n)^2)$

5.5. Example

Suppose that we compute multiple multiplications of two elements and three

$$\sum_{i=0}^{m-1}(\sum_{j=0}^{m-1}\alpha_{ij}x_i x_j)+\sum_{i=0}^{m-1}\beta_i x_i.$$

elements, e.g.

If we use two-input multipliers, $2m^2+m$ multiplications of two elements are required. If we use three-input multipliers, m^2+m multiplications are required. If the computation is performed over $GF((2^2)^2)$, the execution time for using LUT2 and LUT3 is $9(2m^2+m)$ and $10(m^2+m)$, respectively. Since $m \geq 1, 18m^2+9m > 10m^2+10m$.

It can be observed that when performing multiple multiplications, three-input multipliers are faster

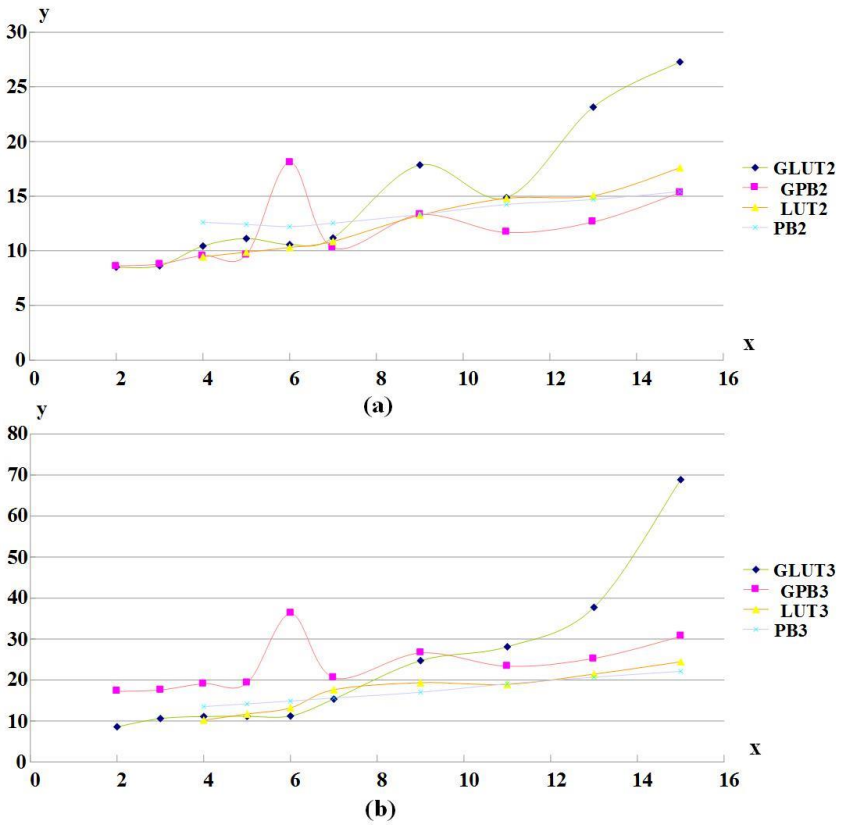

Figure 4. Comparison Multiplications over Different Fields

6. Comparison

To the best of our knowledge, the existing fastest multipliers over small finite fields on FPGAs is [16]. Then we compare our multipliers with [16], which is presented in Table 5.

It can be observed from Table 5 that if we choose our multipliers over $GF(2^5)$, multiplication of two elements and three elements require 9.634 ns and 11.106 ns, respectively. Since 9.634 ns <13.2 ns (First class from [16]), our multiplier is faster when computing multiplication of two elements. On the other hand, since 11.106 ns < 2×13.2 ns (First class from [16]), our multiplier is faster when computing multiplication of three elements.

Therefore, the comparison clearly demonstrates that our multipliers are faster than other multipliers when computing multiplications of three elements.

Table 5. Comparison of Our Design and other Multipliers

	Proposed two-input(ns)	Proposed three-input(ns)	First class(ns) [16]	Second class(ns) [16]
$GF(2^5)$	9.634	11.106	13.2	13.3
$GF(2^7)$	10.276	15.359	13.9	14.3
$GF(2^9)$	13.298	24.749	15.9	15.1
$GF(2^{15})$	15.322	30.640	19.5	19.2

7. Applications

Multiplication of three elements over finite fields is playing a key role in mathematical and cryptographic applications. In the following, we present two applications of our multipliers, i.e. Gaussian elimination and multivariate signature schemes.

7.1. Gaussian Elimination over Finite Fields

Gaussian elimination can be performed by transforming Ax=b into the equivalent system A'x = b', where A' is an upper triangular matrix.

Table 6. Comparison of Gaussian Eliminations Using Different Multipliers

	Time(ns)	Frequency(MHz)	Cycles	Ratio
Two-Input(LUT)	4235	90.9	385	1.0*
Three-Input(LUT)	3630	90.9	330	1.14
Two-Input(PB)	5390	71.4	385	1.0*
Three-Input(PB)	4620	71.4	330	1.14

*The speed of Gaussian Elimination using two-input multipliers is normalized to 1.0.

The original Gaussian elimination uses two-input multipliers, which requires 385 multiplications with matrix size 10×11.Gaussian elimination using three-input multipliers requires 330 multiplications with matrix size 10×11.

The comparisons of two-input and three input multipliers for Gaussian elimination over $GF((2^2)^2)$ are depicted in Table 6, where we adapt LUT2 and PB2 for two-input multipliers and LUT3 and PB3 for three-input multipliers, respectively. It can be observed from Table 6 that three-input multipliers are faster and require less clock cycles than two-input multipliers when performing Gaussian elimination.

7.2. Multivariate Signature Schemes

There are multiple multiplications of three elements in multivariate signature schemes, e.g. Rainbow and TTS. In the following, we use our multipliers to speed up Rainbow and TTS.

Table 7 is the implementations of Rainbow and TTS using two-input and three-input multipliers, respectively. It can be observed from Table 7 that three-input multipliers are able to speed up Rainbow and TTS.

Table 7. Rainbow and TTS Signatures Using Two-Input and Thee-Input Multipliers

	2-input Multi.	3-input Multi.	Clock Cycle	Frequency (MHz)	Time (us)	Ratio of Speed
2-input(Rainbow)	5810	-	5876	50	118	1.14
3-input (Rainbow)	2340	2109	4515	50	90.3	1.0*
2-input (TTS)	2088	-	2136	50	42.7	1.14
3-input (TTS)	1184	647	1879	50	37.6	1.0*

* The speed of using three-input multipliers is normalized to 1.0.

8. Conclusions and Future Improvements

We propose three-input multipliers that extends the improvements on the multiplication of three elements in three directions compared with the known results. First, since most of multivariate cryptographic schemes can be implemented over small composite fields $GF((2^n)^2)$, our multipliers can execute multiplications over such fields. Second, since multivariate cryptography requires multiplication of two elements and three elements, our multipliers can compute both of them. Third, since table look-up and polynomial basis methods have advantages of speed in different fields, respectively, our multipliers can perform multiplications based on these methods.

We test and verify our design on a FPGA. Experimental results show that our multipliers are faster than other two-input multipliers when computing multiplication of three elements, e.g. multiplication of three elements over $GF((2^4)^2)$ is 28.4% faster (executing time 16.354 ns). We demonstrate the improvement of our multipliers mathematically. Moreover, they can speed up Gaussian elimination and implementations of multivariate cryptography, e.g. TTS is 14% faster (executing time 37.6 ns). Besides, our multipliers can also be used in other mathematical and cryptographic applications.

There is still space for improvements of our work and the upper limit of the speed of our multipliers is not reached yet. First, our multipliers can adapt other methods for multiplication, e.g. normal basis, dual basis. Second, our design can be extended to $GF((2^n)^m)$. Third, our design are also suitable for multiplication of multiple elements, namely the number of elements larger than three.

Acknowledgements

This work is Supported by Foundation for Distinguished Young Talents in Higher Education of Guangdong, China (No.2014KQNCX177), and Quality Engineering Project of Department of Education of Guangdong Province (2014).

References

[1] P. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”, SIAM Review, vol. 41, no. 2, **(1999),** pp. 303–332.
[2] D. J. Bernstein, “Introduction to post-quantum cryptography”, In Post-Quantum Cryptography, **(2009)**, pp. 1–14.
[3] D. S Johnson, “The NP-completeness column: An ongoing guide”, Journal of Algorithms, vol. 3, no. 2, **(1982),** pp. 182 – 195.
[4] J. Ding and D. Schmidt, “Rainbow, a new multivariable polynomial signature scheme”, In Applied Cryptography and Network Security, **(2005)**, pp. 164–175.
[5] J.-M. Chen and B.-Y. Yang, “A more secure and efficacious TTS signature scheme”, In Information Security and Cryptology - ICISC 2003, **(2004)**, pp. 320–338.
[6] M.-L. Akkar, N. T. Courtois, R. Duteuil, and L. Goubin, “A fast and secure implementation of Sflash”, In Public Key Cryptography - PKC 2003, **(2002)**, pp. 267–278.
[7] B.-Y. Yang, J.-M. Chen, and Y.-H. Chen, “TTS: High-speed signatures on a low-cost smart card’. In Cryptographic Hardware and Embedded Systems - CHES 2004, **(2004)**, pp. 371–385.
[8] S. Balasubramanian, H. W. Carter, A. Bogdanov, A. Rupp, and Jintai Ding, “Fast multivariate signature generation in hardware: The case of Rainbow”, In International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2008**, (2008)**, pp. 25–30.
[9] S. Tang, H. Yi, J. Ding, H. Chen, and G. Chen, “High-speed hardware implementation of Rainbow signature on FPGAs’. In Post-Quantum Cryptography, **(2011),** pp. 228–243.
[10] A. Bogdanov, T. Eisenbarth, A. Rupp, and C. Wolf, “Time-area optimized public-key engines: MQ-cryptosystems as replacement for elliptic curves?”, In Cryptographic Hardware and Embedded Systems - CHES 2008, **(2008),** pp. 45–61.
[11] B.-Y. Yang, C.-M. Cheng, B.-R. Chen, and J.-M. Chen, “Implementing minimized multivariate PKC on low-resource embedded systems”, In Security in Pervasive Computing, **(2006),** pp. 73–88.
[12] C. Berbain, Olivier Billet, and Henri Gilbert, “Efficient implementations of multivariate quadratic systems. In Selected Areas in Cryptography”, **(2007),** pp.174–187.

[13] P. Czypek, S. Heyse, and E. Thomae, "Efficient implementations of MQPKS on constrained devices", In Cryptographic Hardware and Embedded Systems – CHES 2012, **(2012),** pp. 374–389.
[14] A. Petzoldt, E. Thomae, S. Bulygin, and C. Wolf, "Small public keys and fast verification for multivariate quadratic public key systems", In Cryptographic Hardware and Embedded Systems - CHES 2011, **(2011),** pp. 475–490.
[15] E. D. Mastrovito, "VLSI designs for multiplication over finite fields $GF(2^m)$ ", In Proceedings of the 6th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, **(1989),** pp. 297–309.
[16] J. L. Imana, "Reconfigurable implementation of bit-parallel multipliers over $GF(2^n)$ for two classes of finite fields", In 2004 IEEE International Conference on Field-Programmable Technology, **(2004),** pp. 287 – 290.

