ON THE REPRESENTATION OF FINITE FIELDS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SEDAT AKLEYLEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

DECEMBER 2010

Approval of the thesis:

ON THE REPRESENTATION OF FINITE FIELDS

submitted by **SEDAT AKLEYLEK** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Department of Cryptography, Middle East Technical University** by,

Prof. Dr. Ersan Akyıldız
Director, Graduate School of **Applied Mathematics**

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography**

Prof. Dr. Ferruh Özbudak
Supervisor, **Department of Mathematics, METU**

Examining Committee Members:

Prof. Dr. Ersan Akyıldız
Department of Mathematics, METU

Prof. Dr. Ferruh Özbudak
Department of Mathematics, METU

Assist. Prof. Dr. Erdal Kılıç
Department of Computer Engineering, Ondokuz Mayıs University

Dr. Murat Cenk
Department of Electrical and Computer Engineering, University of Waterloo, Canada

Dr. Tolga Yalçın
Institute of Applied Mathematics, METU

Date:

Name, Last Name:	SEDAT AKLEYLEK
Signature :	

ABSTRACT

ON THE REPRESENTATION OF FINITE FIELDS

Akleylek, Sedat
Ph.D., Department of Cryptography
Supervisor : Prof. Dr. Ferruh Özbudak

December 2010, 66 pages

The representation of field elements has a great impact on the performance of the finite field arithmetic. In this thesis, we give a modified version of redundant representation which works for any finite fields of arbitrary characteristics to design arithmetic circuits with small complexity. Using our modified redundant representation, we improve many of the complexity values. We then propose new representations as an alternative way to represent finite fields of characteristic two by using Charlier and Hermite polynomials. We show that multiplication in these representations can be achieved with subquadratic space complexity. Charlier and Hermite representations enable us to find binomial, trinomial or quadranomial irreducible polynomials which allows us faster modular reduction over binary fields when there is no desirable such low weight irreducible polynomial in other representations. These representations are very interesting for the NIST and SEC recommended binary fields $GF(2^{283})$ and $GF(2^{571})$ since there is no optimal normal basis (ONB) for the corresponding extensions. It is also shown that in some cases the proposed representations have better space complexity even if there exists an ONB for the corresponding extension.

Keywords: finite field representation, polynomial multiplication, subquadratic space complexity, elliptic curve cryptography

ÖZ

SONLU CİSİMLERİN GÖSTERİMİ ÜZERİNE

Akleylek, Sedat
Doktora, Kriptografi Bölümü
Tez Yöneticisi : Prof. Dr. Ferruh Özbudak

Aralık 2010, 66 sayfa

Cisim elemanlarının gösterimi sonlu cisim aritmetik uygulamalarının performansı üzerinde büyük öneme sahiptir. Bu tezde, herhangi bir karakteristiğe sahip sonlu cisimlerde düşük çarpımsal karmaşıklığa sahip devre ihtiyacı için tasarlanan, gerekenden fazla eleman kullanan gösterimin değiştirilmiş versiyonu veriliyor. Bu gösterimi kullanarak bir çok değerin karmaşıklığını azaltıyoruz. Sonra, karakteristiği 2 olan sonlu cisimlerin gösterimlerine alternatif bir yol olması için Charlier ve Hermite polinomların kullanılmasını öneriyoruz. Bu gösterimlerde çarpma işleminin logaritmik alan karmaşıklığı ile yapılabildiğini gösteriyoruz. Charlier ve Hermite gösterimleri, hızlı modüler aritmetik yapmamıza ve başka gösterimler kullanılarak istenilen düzeyde az terimli indirgenemez polinom elde edilemediği durumlarda, iki, üç ve dört terimli indirgenemez polinomları bulabilmemize olanak sağlamaktadır. Bu gösterimler, NIST ve SEC standartlarında karakteristiği 2 olan cisimlerde kullanılması önerilen $GF(2^{283})$ ve $GF(2^{571})$ cisim genişlemeleri için optimal normal gösterim bulunmadığından oldukça ilginç sonuçlar vermektedir. Bunlara ek olarak, bazı cisim genişlemeleri için optimal normal gösterim olsa bile önerilen bu yeni gösterimlerin daha iyi alan karmaşıklığına sahip olduğunu gösteriyoruz.

Anahtar Kelimeler: sonlu cisimlerin gösterimleri, polinom çarpımı, logaritmik alan karmaşıklığı, eliptik eğri kriptografisi

biricik yeğenime

ACKNOWLEDGMENTS

I would like to thank Prof. Dr. Ferruh Özbudak, my supervisor, for his many suggestions and constant support. This thesis would not have been completed without his guidance.

I also want to thank Dr. Murat Cenk sharing with me his knowledge of mathematics, and providing much useful advice and friendly encouragement.

I would like to express my sincere gratitude to Prof. Dr. Ersan Akyıldız for his encouragement and many advices to understand the life during my graduate studies.

I want to thank all friends who have provided me with the joy and warmth needed to keep me going through out my graduate studies.

I would like to thank to my family for the support they provided me through out my life.

I also greatfully acknowledge the financial support of TÜBİTAK BİDEB 2211 Program during this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Finite fields have many applications in coding theory, digital signal processing and cryptography ([12], [17], [26], [27], [28], [29], [31]). Efficient arithmetic of finite field is an important factor for cryptographic applications. Finite field multiplication is the main operation for most of the cryptographic applications. The measure of efficiency in hardware implementations is the number of AND gates and XOR gates. An improvement in complexity refers to a decrease in the number of AND gates and XOR gates simultaneously in hardware implementations. Its complexity depends on the representation of the field elements and choice of reduction polynomial [18]. From the implementation point of view, polynomial basis and normal basis representations are mostly recommended to represent finite field elements [22]. It is well-known that normal basis, especially optimal normal basis (ONB) has great advantegous on squaring. With an optimal choice of field, the space complexity of multiplication is about the same as for a polynomial representation. However, ONB does not exist for all extensions. Therefore, there is a big demand to represent finite fields in a different way for the extension degrees recommended in NIST and SEC standards ([32], [36]). In this thesis, we focus on the representation of elements of finite fields and multiplication of elements in a finite field of characteristic p, where $p \geq 2$.

The representation of field elements has a great impact on the performance of the finite field arithmetic [26]. There are mainly three types of representation of finite fields of characteristic p, namely canonical (polynomial) basis, normal basis and redundant representation. Recently, Dickson polynomial representation has been proposed to obtain efficient binary field multiplication using low weight irreducible polynomial in [20] and [21]. Hasan and Negre formulate the multiplication of two elements in the field as a product of Toeplitz or Hankel matrix. Dickson polynomials seem interesting when no optimal normal basis (ONB) in any type exists for the field. This is the case for NIST recommended binary fields $GF(2^{163})$ and $GF(2^{283})$. By using Dickson polynomial representation, one can obtain irreducible Dickson binomials or trinomials. Depending on the choice of basis, binary field multiplication can be performed in different ways.

The extension field multiplication can be performed in two steps: polynomial multiplication over $GF(p)$ and modular reduction over $GF(p^n)$. As the complexity of finite field multiplication depends on the number of non-zero terms in the reduction polynomials, it is desirable to use the reduction polynomials with as few non-zero terms as possible. Over binary fields, the use of trinomial or when trinomial does not exist for the corresponding extension, pentanomial is preferred since there is no irreducible binomial or quadranomial except for $x + 1$ in $GF(2)[x]$.

It is well-known that two parameters for hardware implementations are of vital importance: space complexity and time complexity. In this thesis, the complexity of the algorithms for arithmetic operations in finite fields are given by the number of operations in the ground field. For example, an addition and a multiplication in $GF(2)$ can be defined by a two-input XOR gate and a two-input AND gate, respectively. The space complexity of an algorithm for a given input is the number of AND gates and XOR gates that the algorithm needs to store during its execution. To find the space complexity of the algorithm is very helpful to obtain compact VLSI implementations. The time complexity i.e. the total gate delay of the circuit of an algorithm for a given input is the number of AND gates and XOR gates that the algorithm executes. This number is computed with respect to size of the input. A reduced space complexity is one of the crucial point for the applications based on smart cards and mobile phones. Similarly, if the performance is the most critical parameter, a greater space complexity is acceptable while there is an improvement in the total gate delay.

We classify our contributions in three parts. In Chapter 2, we give a modified redundant representation. Using our modified redundant representation, we improve many of the complexity values significantly. Our method works for any finite field. We give more emphasis for finite fields of characteristic 2. We also give some applications in cryptography.

In Chapter 3, we give a new way to represent certain finite fields $GF(2^n)$. This representation is based on Charlier polynomials. We show that multiplication in Charlier polynomial representation can be performed with subquadratic space complexity. One can obtain binomial or trinomial irreducible polynomials in Charlier polynomial representation which allows us faster modular reduction over binary fields when there is no desirable such low weight irreducible polynomial in other representations. This representation is very interesting for NIST recommended binary field $GF(2^{283})$ since there is no ONB for the corresponding extension. We also note that recommended NIST and SEC binary fields can be constructed with low weight Charlier polynomials such as $GF(2^{113})$, $GF(2^{131})$ and $GF(2^{233})$.

In Chapter 4, Hermite polynomial representation is proposed as an alternative way to represent finite fields of characteristic two. We show that multiplication in Hermite polynomial representation can be achieved with subquadratic space complexity. This representation enables us to find binomial, trinomial or quadranomial irreducible polynomials which allows us faster modular reduction over binary fields when there is no desirable such low weight irreducible polynomial in other representations. We then show that the product of two elements in Hermite polynomial representation can be performed as Toeplitz matrix-vector product. This representation is very interesting for NIST recommended binary field $GF(2^{571})$ since there is no ONB for the corresponding extension. We note that an advantage of this representation is that it can be used to obtain more efficient finite field arithmetic.

A summary of the thesis is presented in Chapter 5.

CHAPTER 2

MODIFIED REDUNDANT REPRESENTATION FOR DESIGNING ARITHMETIC CIRCUITS WITH SMALL COMPLEXITY

Efficient hardware implementations of the arithmetic operations in the finite field have been widely studied in coding theory, computer algebra and cryptographic applications. Itoh and Tsujii [23] first gave a method to construct a multiplier for a class of fields represented by irreducible all-one-polynomials and equally-spaced-polynomials. A redundant representation of field elements is proposed in [23] as well as later in [10], [37], [38] and [41]. Redundant representation allows multiplication and squaring to be achieved more simply than other representations. In cryptographic applications, finite fields, represented by low weight polynomials, are desired due to the efficiency of finite field operations.

The main idea in the redundant representation is to represent the finite field $GF(2^m)$ as a subring of the quotient polynomial ring $GF(2^n)/(x^n - 1)$, provided that $n > m$ and there is a subring of $GF(2^n)[x]/(x^n - 1)$ which is isomorphic to $GF(2^m)$ as a ring. This representation allows a kind of parallel multiplier design yielding small complexity for multiplication in $GF(2^m)$ for some m. For a better complexity, it is necessary to choose the smallest n satisfying the condition above.

Geiselmann, Quade and Steinwandt gave a characterization of the smallest value $n \in N$ with $GF(2)[x]/(x^n - 1)$ containing an isomorphic copy of $GF(2^m)$ in [14]. They showed that the values found in [10] are not optimal in many cases. Similarly, Wu, Hasan, Blake and Gao proved in [42] that some values found in [10] can be reduced significantly when there exists a type II optimal normal basis in $GF(2^m)$. Then, Geiselmann and Steinwandt generalized their idea to finite fields of arbitrary characteristic in [15].

In this chapter, we give a modified redundant representation which can be considered as a generalization of [10], [14] and [15]. Using our modified redundant representation, we improve many of the complexity values in [10], [14] and [15] significantly. Our method works for any finite field. We give more emphasis for finite fields of characteristic 2. We also give some applications in cryptography.

This chapter is organized as follows: Section 2.1 describes a modified redundant representation. In Section 2.2, we illustrate our method with an explicit example and we recall some facts which we need in order to compare our method with previous ones. In Section 2.3, we demonstrate our improvements in tables and give an analysis of the reasons of the improvements. We give certain applications in cryptography in Section 2.4.

The material presented in this chapter is partially included in [4].

2.1 A Modified Redundant Representation

In this section, we recall previous works and describe our method.

2.1.1 The Redundant Representation

Let p be an arbitrary prime number and $m \geq 2$ be a positive integer. Note that the finite field $GF(p^m)$ can be considered as a ring as well. Let $n > m$ be an integer and consider the finite quotient ring $GF(p)[x]/(x^n - 1)$. If there exists a subring of the quotient ring $GF(p)[x]/(x^n - 1)$ which is isomorphic to $GF(p^m)$ as a ring, then it is well-known that we can represent $GF(p^m)$ using the ring representation of the quotient ring $GF(p)[x]/(x^n - 1)$. Such a representation of $GF(p^m)$ is called a *redundant representation* ([14], [37], [41]) or a *polynomial ring representation* ([10]).

Let the canonical factorization of $(x^n - 1) \in GF(p)[x]$ be given as

$$x^n - 1 = f_1(x) f_2(x) \cdots f_t(x), \tag{2.1}$$

where $f_1(x), f_2(x), \cdots, f_t(x)$ are monic (not necessarily distinct) irreducible polynomials in $GF(p)[x]$.

In order to design arithmetic circuits with small complexity using redundant representation of $GF(p^m)$ in $GF(p)[x]/(x^n - 1)$, it is important to choose n as small as possible ([10], [14], [15], [41]).

Let $S_1(p, m)$ be the set of integers consisting of n such that $n > m$ and there exists at least one irreducible factor of degree m in (2.1).

Let $S_2(p, m)$ be the set of integers consisting of n such that $n > m$ and there exists at least one irreducible factor of degree $\ell \cdot m$, where ℓ is a positive integer, in (2.1). It is clear that $S_1(p, m) \subseteq S_2(p, m)$. Note that if $\ell = 1$, then $S_1(p, m) = S_2(p, m)$

In [10], using

$$n = \min\{S_1(p, m)\}, \tag{2.2}$$

arithmetic circuits with small complexity via redundant representation of $GF(p^m)$ in the quotient ring $GF(p)[x]/(x^n - 1)$ are obtained. Remember that the main idea in [10] is to find the smallest n such that $x^n - 1$ has an irreducible polynomial factor of degree m.

In [14], using

$$n = \min\{S_2(p, m)\}, \tag{2.3}$$

arithmetic circuits with small complexity via redundant representation of $GF(p^m)$ in the quotient ring $GF(p)[x]/(x^n - 1)$ are obtained. Remember that the main idea in [14] is to determine the smallest n such that $x^n - 1$ has an irreducible polynomial divisor of degree $\ell \cdot m$ where l is a positive integer.

It is shown that using $S_2(p, m)$ instead of $S_1(p, m)$ improves the complexity of various arithmetic operations for many values of p and m ([14]).

Now we explain the main idea of our contribution.

2.1.2 Our Contribution

Let k be a positive integer such that

$$k | m \quad \texttt{and} \quad k < m \tag{2.4}$$

Let m_k be the positive integer such that $m = k \cdot m_k$. Note that $1 < m_k \leq m$. Let the canonical factorization of $(x^n - 1) \in GF(p^k)[x]$ be given as

$$x^n - 1 = g_1(x)g_2(x)\cdots g_s(x), \tag{2.5}$$

where $g_1(x), g_2(x), \cdots, g_s(x)$ are monic (not necessarily distinct) irreducible polynomials in $GF(p^k)[x]$.

Let $T_k(p, m_k)$ be the set of integers consisting of n such that $n > m_k$ and there exists at least one irreducible factor of degree $\ell \cdot m_k$, where ℓ is a positive integer, in (2.5). Note that if $k = 1$, then $m_k = m$ and $T_k(p, m_k) = S_2(p, m)$. However if $k > 1$ and k satisfies (2.4), then we observe that

$$\min\{T_k(p, m_k)\} << \min\{S_2(p, m)\} \tag{2.6}$$

for many values of p and m (see Table 2.1 and Table 2.2 in Section 2.3).

In this paper, we use redundant representation in the following form: For an arithmetic operation, assume that we know an arithmetic circuit design for $GF(p^k)$. For $p = 2$, this means, in particular, we know the number of AND gates, XOR gates of the design for $GF(p^k)$. Then using

$$n_k = \min\{T_k(p, m_k)\} \tag{2.7}$$

via redundant representation of $GF(p^m)$ in $GF(p^k)[x]/(x^{n_k} - 1)$ and the design of the arithmetic operation for $GF(p^k)$, we get an arithmetic circuit for the corresponding arithmetic operation in $GF(p^m)$. Moreover, we optimize the complexity of the arithmetic operation considering all divisors k of m satisfying (2.4).

Our method depends on the following fact, which is a simple generalization of Proposition 1 in [14]. We call our method as modified redundant representation.

Proposition 2.1.1 *Let p be a prime number. Let $m \geq 2$ be a positive integer. Let k be a positive integer with $k|m$ and $k < m$. Let $m_k = m/k$. Then, the smallest positive integer n_k, such that there exists a subring of the quotient ring $GF(p^k)[x]/(x^{n_k} - 1)$ which is isomorphic to $GF(p^m)$ as a ring, is $\min\{n :$ n is a positive integer and the canonical factorization of $(x^n - 1) \in GF(p^k)[x]$ has an irreducible factor of degree $\ell \cdot m_k$ where ℓ is a positive integer $\}$. Moreover, n_k is not divisible by p.*

Proof. Assume that n_k is divisible by p. In this structure, one can find n'_k such that $(x^{n_k} - 1) = (x^{n'_k} - 1)^p$. Then, $(x^{n'_k} - 1)$ contains an irreducible divisor of degree $\ell \cdot m_k$ with $\ell \geq 1$ and this contradicts with the minimality of n_k. Therefore, n_k is not divisible by p.

The observation in the paragraph above implies that $(x^{n_k} - 1)$ has no multiple roots since

$$gcd((x^{n_k} - 1), \frac{\partial}{\partial x}(x^{n_k} - 1)) = 1.$$

Let $g_i(x)$ be the factor of $(x^{n_k} - 1)$ with degree m_{k_i}. Canonical factorization of $(x^{n_k} - 1) \in GF(p^k)[x]$ is given as

$$(x^{n_k} - 1) = g_1(x)g_2(x) \cdots g_s(x), \tag{2.8}$$

where $g_1(x), g_2(x), \cdots, g_s(x)$ are distinct and monic irreducible polynomials in $GF(p^k)[x]$. Then, by using Chinese Remainder Theorem, the ring is isomorphic to product of fields.

$$GF(p^k)[x]/(x^{n_k} - 1) \cong GF(p^k)[x]/g_1(x) \times ... \times GF(p^k)[x]/g_s(x) \tag{2.9}$$

It is well-known that $GF(p^{k \cdot m_k})$ is a subfield of $GF(p^k)[x]/g_i(x)$ if and only if $m_k | m_{k_i}$ for some $i \in \{1, 2, ..., s\}$. This completes the proof. ∎

Now we consider Proposition 2.1.1 with an algorithmic approach. To find the smallest n_k, it is important to obtain an algorithm. Theorem 2.1.2, a modification of Theorem 1 in [42] to our structure, characterizes the relationship between n_k and m_k.

Theorem 2.1.2 *Let $m = k \cdot m_k$ and $q = p^k$, where p is a prime number and $k > 0$ is a positive integer. There exists a subring of the quotient ring $GF(q)[x]/(x^{n_k} - 1)$ isomorphic to $GF(q^{m_k})$ as a ring with $n_k > m_k$ is and only if m_k divides the multiplicative order of $q \pmod{n_k}$.*

Algorithm 1 gives the smallest n_k such that there exists a subring of $GF(q)[x]/(x^{n_k} - 1)$ isomorphic to $GF(q^{m_k})$ as a ring with $n_k > m_k$. Algorithm 1 is a consequence of Theorem 2.1.2. Note that Algorithm 1 is a simple generalization of the idea given in [41].

Algorithm 1 Computing the smallest n_k such that there exists a subring of the quotient ring $GF(q)[x]/(x^{n_k-1})$ isomorphic to $GF(q^{m_k})$ as a ring with $n_k > m_k$

Input: $q = p^k, m = k \cdot m_k$

Output: n_k

```
1: Find all the factors, $d_i \geq (m_k + 1)$ of $2^{m_k} - 1$ and list them in an increasing order: $d_1 \cdot d_2 \cdot \cdots \cdot d_c = 2^{m_k} - 1$
2: while $i \leq c$ do
3:     if $m_k | \varphi(d_i)$ and the multiplicative order of $q$ in $\mathbb{Z}_{d_i^*}$ is $m_k$ then
4:         $t \leftarrow d_i$ and BREAK
5:     else
6:         $i \leftarrow i + 1$
7:     end if
8: end while
9: Let $h$ be the largest positive integer such that $t > hm$
10: if $h \geq 2$ then
11:     for $i = 2$ to $h$ do
12:         Find all the factors, $d_i > i \cdot m_k$ of $2^{i \cdot m_k} - 1$ and list them in an increasing order: $d_1 \cdot d_2 \cdot \cdots \cdot d_{c_i} = 2^{i \cdot m_k} - 1$
13:         while $i \leq c_i$ do
14:             if $i \cdot m_k | \varphi(d_i)$ and the multiplicative order of $q$ in $\mathbb{Z}_{d_i^*}$ is $i \cdot m_k$ then
15:                 $n \leftarrow min\{t, d_i\}$ and BREAK
16:             else
17:                 $i \leftarrow i + 1$
18:             end if
19:         end while
20:     end for
21: end if
```

2.2 Multiplication Using Modified Redundant Representation and Complexity

In this section, we keep the notation of Section 2. Recall that in the modified redundant representation method, we assume that we know an arithmetic circuit design for the intermediate fields $GF(p^k)$. In our examples, we use intermediate fields with $k = 2, 3$ and 5. In this section, we explain various finite field multiplication circuit designs for the intermediate field $GF(p^k)$. We now summarize the schoolbook method, Karatsuba method and Tom-Cook method in view of required multiplications and additions. These methods can be used for the corresponding arithmetic circuits design in the intermediate fields.

2.2.1 Overview of Multiplication Methods

Let $f(x)$ be an irreducible polynomial of degree k in $GF(p)[x]$. Multiplication in

$$GF(p^k) = GF(p)[x]/ < f(x) >$$

is computed as a multiplication of polynomials with modulo $f(x)$ reduction. A simple and generic design for finite field multiplication in $GF(p^k)$ is the schoolbook method. Consider two $k-$term polynomials

$$a(x) = \sum_{i=0}^{k-1} a_i x^i, b(x) = \sum_{j=0}^{k-1} b_j x^j \tag{2.10}$$

By using the schoolbook multiplication method, one can compute $c(x) = a(x) \cdot b(x)$ as

$$c(x) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_i b_j x^{i+j} \tag{2.11}$$

Assume that reduction polynomial, $f(x)$, is binomial. Then,

$$x^{i+j} = \begin{cases} x^{i+j} \ \texttt{if} \ i+j < k \\ x^{i+j-k} \ \texttt{if} \ i+j \geq k \end{cases} \tag{2.12}$$

Then, for binomials, schoolbook multiplication method can be performed at most k^2 multiplications and $k(k-1)$ additions in $GF(p)$. By using the same idea, if the reduction polynomial, $f(x)$, is trinomial or pentanomial, then the number of additions is $(k+1)(k-1)$ or $(k+3)(k-1)$, respectively [25].

Karatsuba method splits elements into 2 parts [24]. We show this technique with an example. Let $GF(2^2) = GF(2)[x]/ < (x^2 + x + 1) >$, $a(x) = a_0 + a_1 x$ and $b(x) = b_0 + b_1 x$. Karatsuba method computes $a(x) \cdot b(x)$ as

$$c(x) = c_0 + c_1 x, \tag{2.13}$$

where

$$\begin{aligned} c_0 &= a_0 b_0 + a_1 b_1 \\ c_1 &= (a_0 + a_1)(b_0 + b_1) + a_0 b_0 + a_1 b_1. \end{aligned}$$

Multiplication in $GF(p^2)$ can be computed with three multiplications and four additions in $GF(p)$ by using Karatsuba method.

Tom-Cook method is based on interpolation and uses $2k - 2$ distinct elements of finite field with point at ∞ [8]. This method works in the following order: Let x_i be the interpolation points. Choose a family $\{x_i\}$ for $0 \leq i < 2k - 1$ of distinct points in $GF(p)$. Evaluate the product $a(x_i)b(x_i) \in GF(p)$ for each i. Then, interpolate the evaluation points to obtain $a(x) \cdot b(x) \in GF(p)[x]$.

One multiplication in $GF(p^3)$ theoretically costs 5 multiplication and 33 additions in $GF(p)$ by using Tom-Cook multiplication method. In this structure, Tom-Cook method cannot be used for all $GF(p)$. For example, let $k = 3$. We need $2 \cdot 3 - 2 = 4$ elements in $GF(p)$. If $p \leq 3$, then, there is not enough points to apply Tom-Cook method for cubic extension $(4 > p)$. For this reason, Karatsuba method is one of the best choice for $GF(2^3)$ and $GF(3^3)$ [30]. One multiplication in $GF(p^3)$ theoretically costs 6 multiplication and 13 additions in $GF(p)$ by using Karatsuba method.

Montgomery gave explicit formula for the extension degree 5 in [30]. One multiplication in $GF(p^5)$ theoretically costs 13 multiplication and 22 additions in $GF(p)$.

2.2.2 Multiplication Using Modified Method

In this part, we explain how we obtain an arithmetic design for finite field multiplication in $GF(p^m)$ assuming an arithmetic design for finite field multiplication in the intermediate field $GF(p^k)$ in the case that $p = 2$. The same method works for the general characteristic p.

The multiplication in $GF(p^k)[x]/(x^{n_k} - 1)$ can be performed with a linear feedback shift register with feedback polynomial $(x^{n_k} - 1)$. Let $a(x) = \sum_{i=0}^{n_k-1} a_i \cdot x^i$ and $b(x) = \sum_{i=0}^{n_k-1} b_i \cdot x^i$ where

$a_i, b_i \in GF(p^k)$. Then,

$$c(x) = a(x) \cdot b(x) = \sum_{j=0}^{n_k-1} \left(\sum_{i=0}^{n_k-1} a_i \cdot b_{(j-i) \pmod{n_k}} \right) \cdot x^j \tag{2.14}$$

Multiplier structure of redundant basis is given in Figure 2.1.

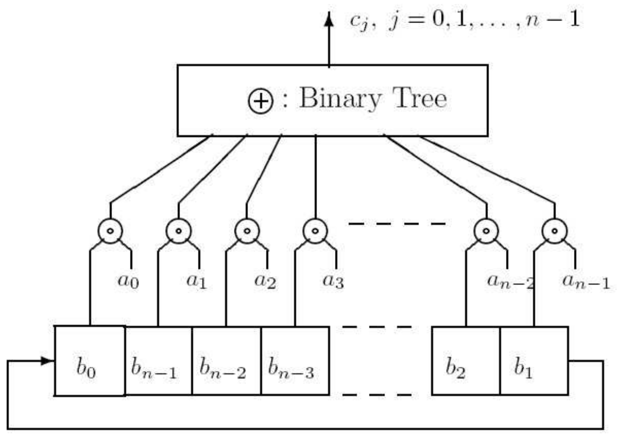

Figure 2.1: Bit Serial Multiplier for Redundant Basis

Figure 2.2 shows a parallel version of the multiplier using redundant basis. Note that in Figure 2.2, B refers to Figure 2.1.

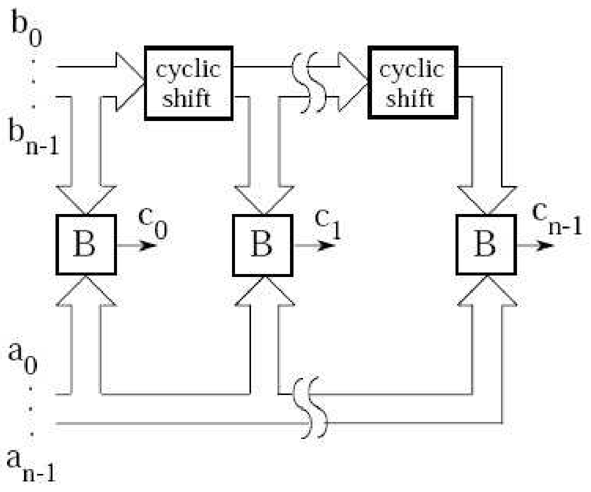

Figure 2.2: Parallelization of the Bit Serial Multiplier for Redundant Basis

In the following example, we give an illustration of our method and we show how to calculate the cost of multiplication.

Example 2.2.1 *Let $p = 2$ and $m = 12$. Let $k = 3$. Then, $m_k = 4$. This means that we need to find the smallest $n_k \in N$ with $GF(2^k)[x]/(x^{n_k} - 1)$ containing an isomorphic copy of $GF(8^{m_k})$. We build $GF(2^{12})$ as*

$$GF(8)[x]/(x^5 - 1) \cong GF(8)[x]/g_1(x) \times GF(8)[x]/g_2(x), \tag{2.15}$$

with $GF(8) = GF(2)[y]/(y^3 + y + 1)$ since

$$(x^5 - 1) = g_1(x)g_2(x) = (x - 1)(x^4 + x^3 + x^2 + x + 1) \tag{2.16}$$

over $GF(8)$. An element $a(x) \in GF(2^{12})$ is represented as $a_0 + a_1x + a_2x^2 + a_3x^3$, where $a_i \in GF(8)$.

Let $a(x), b(x) \in GF(8)[x]/(x^5 - 1)$ and

$$c(x) = a(x) \cdot b(x) = \sum_{i=0}^{4} c_i \cdot x^i, \tag{2.17}$$

where $c_i \in GF(8)$. Then, one can compute c_i's by using the formula defined below. This is a parallel-in-parallel-out multiplier [10]. The multiplication of two elements in the ring $GF(8)[x]/(x^5 - 1)$ can be performed by a matrix multiplication.

$$\begin{bmatrix} b_1 & b_2 & b_3 & b_4 & b_0 \\ b_2 & b_3 & b_4 & b_0 & b_1 \\ b_3 & b_4 & b_0 & b_1 & b_2 \\ b_4 & b_0 & b_1 & b_2 & b_3 \\ b_0 & b_1 & b_2 & b_3 & b_4 \end{bmatrix} \cdot \begin{bmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \tag{2.18}$$

If one uses the schoolbook method for intermediate field multiplication, this matrix multiplication needs $3^2 \cdot 5^2$ AND gates since it requires 5^2 multiplications in $GF(2^3)$ and one multiplication in $GF(2^3)$ requires 3^2 multiplications in $GF(2)$. The number of XOR gates is $8 \cdot 5^2 + 3 \cdot 4 \cdot 5$ since one needs $3^2 - 1$ additions in $GF(2)$ for one multiplication in $GF(2^3)$ and 3 additions in $GF(2)$ for one addition in $GF(2^3)$. Note that one requires $5 \cdot 4$ additions in $GF(2^3)$ to perform one multiplication in $GF(2^{12})$. If one uses Karatsuba-like method defined above for intermediate field multiplication, then the cost of multiplication in $GF(8)[x]/(x^5 - 1)$ is $6 \cdot 5^2$ AND gates and $13 \cdot 5^2 + 3 \cdot 4 \cdot 5$ XOR gates.

One can determine the number of AND gates and XOR gates for the extension degree multiple of k by using the following formulas for binary fields. These are the upper bounds for the multiplication. AND_k or XOR_k refers to the number of multiplications or additions in $GF(2)$ to perform a multiplication in $GF(2^k)$, respectively.

$$\begin{aligned}\#ANDs &= AND_k \cdot n_k^2 \\ \#XORs &= XOR_k \cdot n_k^2 + k \cdot n_k \cdot (n_k - 1)\end{aligned}$$

Now, we explain how to compute the number of AND gates and XOR gates for any k and n_k. Assume that one needs to work in a field $GF(2^m)$ with $m = k \cdot m_k$. Let $(x^{n_k} - 1)$ have an irreducible factor with degree $\ell \cdot m_k$ where ℓ is a positive integer. To perform a multiplication in $GF(2^m)$ n_k^2 multiplications in $GF(2^k)$ and $n_k(n_k - 1)$ additions in $GF(2^k)$ are required. One multiplication in $GF(2^k)$ requires AND_k multiplications in $GF(2)$ and XOR_k additions in $GF(2)$, with $AND_k = k^2$ by using the schoolbook multiplication for any extension, $AND_k = 3$, $XOR_k = 5$ for quadratic extension, $AND_k = 6$, $XOR_k = 13$ for cubic extension and $AND_k = 13$, $XOR_k = 22$ for quintic extension by using Karatsuba-like method. If one uses trinomial or pentanomial as a reduction polynomial while performing the schoolbook multiplication, then $XOR_k = (k + 1)(k - 1)$ or $(k + 3)(k - 1)$, respectively. One addition in $GF(2^k)$ requires k additions in $GF(2)$.

These formulae are used to construct Table 2.1 in Section 4. Note that we prefer to use the schoolbook multiplication method for intermediate field arithmetic.

We observe that multiplication by x^i and squaring are also as efficient as given in [15]. This observation is stated as a remark.

Remark 2.2.2 *Let $a = (a_{n_k-1}, a_{n_k-2}, \cdots, a_1, a_0)$, where $a_i \in GF(p^k)$. Multiplication by x^i for $0 < i < n_k$ is i times cyclic shift. Then,*

$$x^i \cdot a = (a_{n_k-1-i}, a_{n_k-2-i}, \cdots, a_1, a_0, \cdots, a_{n_k-i+1}, a_{n_k-i}).$$

Remark 2.2.3 *Similarly, squaring is a permutation of the element's coordinates. Let $a = (a_{n_k-1}, a_{n_k-2}, \cdots, a_1, a_0)$, where $a_i \in GF(p^k)$.*

$$a^2 = (a^2_{\frac{n_k-1}{2}}, a^2_{n_k-1}, \cdots, a^2_1, a^2_{\frac{n_k+1}{2}}, a^2_0).$$

2.3 Improved Results

In this section, we obtain n_k's by using modified redundant representaion. Then, we compute the number of AND gates and XOR gates according to the derived formulas in Section 2.2.2. Remember that, in this paper, we use the complexity to compute the required number of AND gates and XOR gates for multiplication of two elements.

All values listed in the tables are found by using Magma Computational Algebra System [6]. we obtain n_k's by using Algorithm 1. Then, we check these results by factorizing all binomials with an odd extension degree n_k into irreducible polynomials over corresponding finite field.

We demonstrate some of our improvements in complexities, i.e. the number of AND gates and XOR gates, according to [14] in Table 2.1. These are computed by using schoolbook multiplication method. The complexity of all values listed in Table 2.1 of our method is much better than the one given in [14] in view of the number of AND gates and XOR gates. In Table 2.1, we give the respective change in percentage of AND gates and XOR gates. Table 2.1 also shows the smallest n and n_k with $GF(2)[x]/(x^n - 1)$ and $GF(2^k)[x]/(x^{n_k} - 1)$ an isomorphic copy of $GF(2^{k \cdot m_k})$. According to Table 2.1, all values of n_k is much more smaller than the corresponding n given in [14]. Note that if one uses Karatsuba-like method for intermediate field arithmetic, the complexity of all values listed in Table 2.1 of our method is also much better than the one given in [14] in view of the number of AND gates and XOR gates.

Now we give an example to show how the smallest n and n_k are computed.

Example 2.3.1 *Let us consider $GF(2^{262})$. Then, $m_k = 131$ and $k = 2$. By using the method given in [14], one computes the smallest value $n = 789$ with $GF(2)[x]/(x^{789}-1)$ containing an isomorphic copy of $GF(2^{262})$. To illustrate our modified method, we need to build $GF(2^{262})$ as*

$$GF(4)[x]/(x^{n_k} - 1) \cong GF(4)[x]/g_1(x) \times \cdots \times GF(4)[x]/g_s(x)$$

with $GF(4) = GF(2)[y]/(y^2 + y + 1)$. Then, by using our modified method the smallest n_k is 263 such that $(x^{n_k} - 1)$ over $GF(4)$ has an irreducible factor of degree $\ell \cdot m_k$, where ℓ is a positive integer. Note that $deg(g_1(x)) = 1$ and $deg(g_2(x)) = deg(g_3(x)) = 131$.

Some observations for the Table 2.1 can be listed as remarks. Remark 2.3.2 shows how to find better values than given in [14].

Table 2.1: The Smallest n and n_k with $GF(2)[x]/(x^n-1)$ and $GF(2^k)[x]/(x^{n_k}-1)$ an Isomorphic Copy of $GF(2^{k\cdot m_k})$ and Complexities

	Our Results				Redundant Representation [14]			Improvements in Gates	
m	k	n_k	$\#ANDs$	$\#XORs$	n	$\#ANDs$	$\#XORs$	$ANDs$	$XORs$
15	3	11	1089	1298	61	3721	3660	%70.7	%64.5
22	2	23	2116	2599	67	4489	4422	%52.8	%41.2
30	3	11	1089	1298	61	3721	3660	%70.7	%64.5
46	2	47	8836	10951	139	19321	19182	%54.2	%42.9
84^{*}	3	29	7569	9164	203	41209	41006	%81.6	%77.6
102	2	103	42436	52839	307	94249	93942	%54.9	%43.7
140^{*}	5	29	21025	24244	319	101761	101442	%79.3	%76.1
174	3	59	31329	38114	349	121801	121452	%74.2	%68.6
190	2	191	145924	182023	573	328329	327756	%55.5	%44.4
246^{*}	3	83	62001	75530	581	337561	336980	%81.6	%77.5
249	3	167	251001	306278	1169	1366561	1365392	%81.6	%77.5
260^{*}	5	53	70225	81196	521	271441	270920	%74.1	%70.0
262	2	263	276676	345319	789	622521	621732	%55.5	%44.4
270	2	271	293764	366663	811	657721	656910	%55.3	%44.1
290^{*}	5	59	87025	100654	649	421201	420552	%79.3	%76.0
300^{*}	3	101	91809	111908	707	499849	499142	%81.6	%77.5
310	2	311	386884	482983	933	870489	869556	%55.5	%44.4
318^{*}	3	107	103041	125618	729	531441	530712	%80.6	%76.3
330	5	67	112225	129846	661	436921	436260	%74.3	%70.2
358	2	359	515524	643687	1077	1159929	1158852	%55.5	%44.4
390^{*}	3	131	154449	188378	869	755161	754292	%79.5	%75.0
410	5	83	172225	199366	821	674041	673220	%74.4	%70.3
444^{*}	3	149	199809	243764	1043	1087849	10868067	%81.6	%77.5
478	2	479	917764	1146247	1437	2064969	2063532	%55.5	%44.4
502	2	503	1012036	1264039	1509	2277081	2275572	%55.5	%44.4
516^{*}	3	173	269361	328700	1211	1466521	1465310	%81.6	%77.5
530	5	107	286225	331486	1061	11257721	1124660	%74.5	%70.5
534^{*}	3	179	288369	351914	1253	1570009	1568756	%81.6	%77.5
588^{*}	3	197	349281	426308	1379	1901641	1900262	%81.6	%77.5
598	2	599	1435204	1792807	1797	3229209	3227412	%55.5	%44.4
646	2	647	1674436	2091751	1941	3767481	3765540	%55.5	%44.4
678^{*}	3	227	463761	566138	1589	2524921	2523332	%81.6	%77.5
718	2	719	2067844	2583367	2157	4652649	4650492	%55.5	%44.4
804^{*}	3	269	651249	795164	1883	3545689	3543806	%81.6	%77.5
1380	3	461	19412689	2336348	5699	32478601	32472902	%94.1	%92.8
1380	5	611	9333025	10823254	5699	32478601	32472902	%71.2	%66.6

Remark 2.3.2 *Let $m = k \cdot m_k$ where $k > 2$ is a prime number. If $m_k + 1$ is prime and 2 is a primitive element of $GF(2^k)$, then the smallest $n_k = m_k + 1$. This ensures that $(x^{m_k+1} - 1)$ has an irreducible factor with degree m_k over $GF(2^k)$. In this setting, $GF(q)[x]/(x^{n_k} - 1)$ is isomorphic to $GF(q^{m_k}) \times GF(q)$, where $q = 2^k$.*

Note that Remark 2.3.2 is a simple generalization of the idea given in [42].

Remark 2.3.3 *Let $m = k \cdot m_k$ where $k > 2$ is a prime number. Then, for $n_k \geq 2 \cdot m_k + 1$, the cost of multiplication in generalized redundant representation increases.*

Remark 2.3.2 and Remark 2.3.3 explain in which cases arithmetic circuits with small complexity can be obtained by using generalized redundant representation. Now we compare the modified redundant representation with the polynomial basis and optimal normal basis (ONB).

Although the modified redundant representation offers almost free squaring as normal basis does, the number of required AND gates and XOR gates in the modified redundant representation is not as small as that of polynomial basis representation to multiply two elements in $GF(2^m)$ since one needs $k \cdot n_k$ bits to represent each element in the modified redundant representation, where $m = k \cdot m_k$ with $n_k > m_k$ and this causes a modest increase in the space complexity (see Example 2.3.4). Thus, the polynomial basis multiplier is much superior compared to the modified redundant representation in view of the number of AND gates and XOR gates. However, the modified redundant representation yields a very modular architecture. As a consequence, the modified redundant representation leads to much more efficient design than polynomial basis in some applications having many repeated squarings and very few multiplications.

Although the number of required AND gates in the modified redundant representation is much more than ONB to multiply two elements in $GF(2^m)$, the modified redundant representation gives better addition complexity i.e. the number of required XOR gates compared to type II ONB for values of $m \in \{30, 174, 330, 410, 530, 690, 810, 1398, 1758\}$. We use the complexity formulae defined for ONB multipliers in composite fields in [34] to give comparison. We give an illustrating example to compare the multipliers in view of the number of required AND gates and XOR gates.

Example 2.3.4 *Let us consider $m = 174$. Then, $m_k = 58$ and $k = 3$. By using the modified redundant representation, the smallest n_k is 59 such that $(x^{n_k} - 1)$ over $GF(8)$ has an irreducible factor of degree m_k. Then, one needs 31329 AND gates and 38114 XOR gates to perform the multiplication in $GF(2^k)[x]/(x^{n_k} - 1)$. If one uses polynomial basis or type II ONB, then the cost of multiplication in $GF(2^m)$ is 30276 AND gates and 36830 XOR gates or 30276 AND gates and 50457 XOR gates, respectively. Note that we build $GF(2^m)$ as $GF((2^k)^{m_k})$.*

Similarly, one reduces the addition complexity for $m = 1380$ where type I ONB exists. Furthermore, for values of m marked with an asteriks in Table 2.1 in which there is no ONB, the modified redundant representation can be used to reduce the number of required XOR gates. As a result, in some applications needing small space complexity the modified redundant representation is advantageous over ONB.

Table 2.2 compares the selected m for the corresponding smallest n_k for the method in [15] and given in this study for the finite fields of characteristic 3. Since the complexity computation is similar to binary fields given in Section 3, we focus on the smallest n_k for this case. For values of m marked with asterisk, the listed value of n_k is smaller than given in [15]. According to Table 2.2, in many cases, the method given in this thesis gives better results.

Table 2.2: The Smallest n and n_k with $GF(3)[x]/(x^n-1)$ and $GF(3^k)[x]/(x^{n_k}-1)$ an Isomorphic Copy of $GF(3^{k \cdot m_k})$

	Our Results		[15]
m	k	n_k	n
12	2	35	35
12^*	3	5	35
$3 \cdot 97^*$	3	389	1747
$5 \cdot 97^*$	5	389	971
$6 \cdot 97^*$	3	389	1747
$6 \cdot 97^*$	6	389	1747
$3 \cdot 193^*$	3	773	6949
$6 \cdot 193^*$	3	773	6949
$6 \cdot 193^*$	6	773	6949
$2 \cdot 239^*$	2	479	958
$3 \cdot 239^*$	3	479	6227
$5 \cdot 239^*$	5	479	5269

2.4 Inversion in the Redundant Basis

In this section we briefly describe inversion operation in $GF(2^k)[x]/(x^{n_k} - 1)$. Inversion in $GF(2^k)[x]/(x^{n_k} - 1)$ can be achieved with different methods. Since not all elements of $GF(2^{k \cdot n_k})$ is invertible, we only consider the elements of $GF(2^m)$. Let $a(x) = \sum_{i=0}^{n_k-1} a_i \cdot x^i$ and $b(x) = \sum_{i=0}^{n_k-1} b_i \cdot x^i$ where $a_i, b_i \in GF(2^k)$. Then, from $a(x) \cdot b(x) = 1$ we have a set of linear equations

$$a(x) \cdot b(x) = \sum_{j=0}^{n_k-1} \left(\sum_{i=0}^{n_k-1} a_i \cdot b_{(j-i) \pmod{n_k}} \right) \cdot x^j = 1 \tag{2.19}$$

We can write the set of equations in matrix form. Note that this matrix is always singular and is a special case of Toeplitz matrix. Any algorithm for solving Toeplitz system can also be used to solve this matrix.

$$\begin{bmatrix} b_1 & b_2 & \cdots & b_{n-1} & b_0 \\ b_2 & b_3 & \cdots & b_0 & b_1 \\ b_3 & b_4 & \cdots & b_1 & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ b_0 & b_1 & \cdots & b_{n-2} & b_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{n-1} \\ a_{n-2} \\ a_{n-3} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} \tag{2.20}$$

2.5 Some Applications of The Modified Method

In this section, we show that our modified representation can be used securely and efficiently in elliptic curve cryptographic applications.

If one needs to work in a field $GF(2^{k \cdot m_k})$ with a very fast multiplication and squaring operations for elliptic curve cryptographic applications, then the values specified in Table 2.3 can be used. The values of m marked with an asterisk are stated to be secure for elliptic curve cryptography in [7]. For example, $GF(2^{226})$ offers the same security level against Gaudry, Hess and Smart attack [13] as the recommended field $GF(2^{233})$ in elliptic curve digital signature algorithm standards. The values of $m \in \{502, 718, 862\}$ give better complexity than the ones in [14].

Special attention to the finite fields of characteristic three has been given in cryptographic

Table 2.3: Some Values of the Smallest n_k with $GF(2^k)[x]/(x^{n_k} - 1)$ for Elliptic Curve Cryptographic Applications

m	k	m_k	n_k
174	3	58	59
226^*	2	113	227
410	5	82	83
502^*	2	251	503
718^*	2	359	719
862^*	2	431	863

applications since there are useful properties in pairing-based cryptography [19]. In characteristic three, most commonly used extensions are of the form $3 \cdot m$, $5 \cdot m$ and $6 \cdot m$, where $m \in \{97, 193, 239\}$. The modified method in this thesis leads to much more efficient designs than the ones in [15] (See Table 2.2).

CHAPTER 3

POLYNOMIAL MULTIPLICATION OVER BINARY FIELDS USING CHARLIER POLYNOMIAL REPRESENTATION WITH LOW SPACE COMPLEXITY

In this chapter, we give a new way to represent certain finite fields $GF(2^n)$. This representation is based on Charlier polynomials. We show that multiplication in Charlier polynomial representation can be performed with subquadratic space complexity. One can obtain binomial or trinomial irreducible polynomials in Charlier polynomial representation which allows us faster modular reduction over binary fields when there is no desirable such low weight irreducible polynomial in other representations. This representation is very interesting for NIST recommended binary field $GF(2^{283})$ since there is no ONB for the corresponding extension. We also note that recommended NIST and SEC binary fields can be constructed with low weight Charlier polynomials such as $GF(2^{113})$, $GF(2^{131})$ and $GF(2^{233})$.

This chapter is organized as follows: Section 3.1 describes Charlier polynomial and gives some general results on Charlier polynomials in $GF(2)[x]$. In Section 3.2, we present the general method to multiply two polynomials in Charlier polynomial representation and give the total arithmetic complexity. We compare complexity of multipliers in view of $\#AND$ and $\#XOR$ gates in Section 3.3. We give an idea about other arithmetic operations in Section 3.4.

This chapter was presented in [2].

3.1 Charlier Polynomials

In this section, we give preliminaries and describe a new representation of binary fields. Charlier polynomials are the monic orthogonal polynomials associated with the inner product [16].

Definition 3.1.1 *The Charlier polynomials are $C_0(x) = 1$, $C_1(x) = x$ with the recursion*

$$C_n(x) = (x - n + 1) \cdot C_{n-1}(x)$$

for $n \geq 2$.

Since we work in binary fields, we give the Charlier polynomials in $GF(2)[x]$ for $n \leq 10$ in Table 3.1. All values in Table 3.1 are computed by using Software for Algebra and Geometry Experimentation (Sage) [35].

Table 3.1: Charlier Polynomials in $GF(2)[x]$

$C_0(x)$	1
$C_1(x)$	x
$C_2(x)$	$x^2 + x$
$C_3(x)$	$x^3 + x^2$
$C_4(x)$	$x^4 + x^2$
$C_5(x)$	$x^5 + x^3$
$C_6(x)$	$x^6 + x^5 + x^4 + x^3$
$C_7(x)$	$x^7 + x^6 + x^5 + x^4$
$C_8(x)$	$x^8 + x^4$
$C_9(x)$	$x^9 + x^5$
$C_{10}(x)$	$x^{10} + x^9 + x^6 + x^5$

3.1.1 Conversion of Coefficients From Polynomial Representation to Charlier Polynomial Representation

The polynomial basis $\{1, x, x^2, \cdots, x^{n-1}\}$, where x is a root of an irreducible polynomial of degree n over $GF(2)$ is usually preferred to represent the elements of $GF(2)[x]$. Let $a(x) = a'_{n-1}x^{n-1} + \cdots + a'_1 x + a'_0$, where $a'_i \in GF(2)$ be a polynomial with the standard (canonical) representation. Let $C_n(x) = \beta_n$ be the n-th Charlier polynomial in $GF(2)[x]$, where $n \geq 0$.

Algorithm 2 Conversion of Coefficients From Polynomial Representation to Charlier Polynomial Representation

Input: $a(x) = \sum_{i=0}^{n-1} a'_i x^i$

Output: $(a_0, a_1, \cdots, a_{n-1})$, where $a = \sum_{i=0}^{n-1} a_i \beta_i$

```
 1: $T \leftarrow a$
 2: for $i = n$ downto 1 do
 3:     if $deg(T) = i$ then
 4:         $a_i \leftarrow 1$
 5:         $T \leftarrow T + \beta_i$
 6:     else
 7:         $a_i \leftarrow 0$
 8:     end if
 9: end for
10: $a_0 \leftarrow T$
```

$a(x)$ can be represented by using Charlier polynomials as $a = a_{n-1}\beta_{n-1} + \cdots + a_1\beta_1 + a_0\beta_0$, where $a_i \in GF(2)$ by using Algorithm 2.

Note that since we are working in characteristic two, Algorithm 2 is self-inverse. That is in Algorithm 2, if the input is a polynomial representation, then the output is the Hermite representation, and conversely if the input is an Hermite representation, then the output is the polynomial representation.

3.1.2 Charlier Basis

A basis for the finite field $GF(2^n)$ is a set of n elements $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\} \in GF(2^n)$ such that every element of the binary field can be represented uniquely as a linear combination of basis elements. For a given $a \in GF(2^n)$, we can write

$$a = \sum_{i=0}^{n-1} a_i \cdot \beta_i$$

where $a_i \in GF(2)$ for $0 \leq i \leq n-1$.

Theorem 3.1.2 *Let $f = \sum_{i=0}^{n} f_i \cdot \beta_i$ be an irreducible polynomial of degree n in $GF(2)[x]$. The set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ forms a basis of $GF(2^n) \cong GF(2)[x]/(f)$.*

Proof. Consequences of Algorithm 2 show that the set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ is linearly independent and each element in $GF(2^n)$ is uniquely expressed by using the set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$. Then, the set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ forms a basis of $GF(2^n) \cong GF(2)[x]/(f)$. ■

Theorem 3.1.3 *Let $C_n(x) = \beta_n$ be the n-th Charlier polynomial in $GF(2)[x]$, where $n \geq 0$. Then, for all $i, j \geq 0$ Charlier basis satisfies the following equation*

$$\beta_i \cdot \beta_j = \beta_{i+j} + \ell \cdot \beta_{i+j-1}$$

where $\ell \in GF(2)$. If i and j are both odd number, then $\ell = 1$. If i or j is an even number, then $\ell = 0$.

Proof. We will prove the theorem by induction on i and j. By using Table 3.1, the theorem is true for few terms. Assume that theorem is true for $i = n - 1$. Then we need to show that it is true for $i = n$. We have four cases:

i. n is even and j is odd

ii. n is even and j is even

iii. n is odd and j is odd

iv. n is odd and j is even

Let $j < n - 1$.

i. Let n be even and j be odd. Note that $\beta_1 = x$

$$\begin{aligned}
\beta_n\beta_j &= (\beta_1\beta_{n-1} + \beta_{n-1})\beta_j \\
&= \beta_{n-1}(\beta_j(\beta_1 + 1)) \\
&= \beta_{n-1}(\beta_{j+1} + \beta_j + \beta_j) \\
&= \beta_{n+j}
\end{aligned}$$

ii. Let n and j be even.

$$\begin{aligned}
\beta_n\beta_j &= (\beta_1\beta_{n-1} + \beta_{n-1})\beta_j \\
&= \beta_{n-1}(\beta_j(\beta_1 + 1)) \\
&= \beta_{n-1}(\beta_{j+1} + \beta_j) \\
&= \beta_{n+j} + \beta_{n+j-1} + \beta_{n+j-1} \\
&= \beta_{n+j}
\end{aligned}$$

iii. Let n and j be odd. Remember that addition of two odd integers is even.

$$\begin{aligned}
\beta_n\beta_j &= (\beta_1\beta_{n-1})\beta_j \\
&= \beta_{n-1}(\beta_{j+1} + \beta_j) \\
&= \beta_{n+j} + \beta_{n+j-1}
\end{aligned}$$

iv. Let n be odd and j be even.

$$\begin{aligned}
\beta_n\beta_j &= (\beta_1\beta_{n-1})\beta_j \\
&= \beta_{n-1}\beta_{j+1} \\
&= \beta_{n+j}
\end{aligned}$$

Note that if $j = n - 1$ or $j = n$, then this case can be proved by considering the factors of β_j or β_{j+1} as shown above. ■

3.2 Polynomial Multiplication Using Charlier Polynomials Over Binary Fields

In this section, we describe polynomial multiplication in Charlier polynomial representation for binary fields and give the total arithmetic complexity. Remember that multiplication in finite fields can be performed in two steps: multiplication over $GF(2)$ and modular reduction over $GF(2^n)$. Therefore, we divide this section into multiplication and reduction parts. Throughout this section, $M(n)$ and $A(n)$ denote the minimum number of multiplications and the minimum number of additions for corresponding algorithm for two n-term polynomials multiplication, respectively. The upper bounds of the required number of multiplications and additions to multiply polynomials in Charlier basis is given in the following theorem.

Theorem 3.2.1 *Let $a = a_{n-1}\beta_{n-1} + \cdots + a_0\beta_0$ and $b = b_{n-1}\beta_{n-1} + \cdots + b_0\beta_0$ be n-term polynomials over $GF(2)$ and $a \cdot b = c = c_{2n-2}\beta_{2n-2} + \cdots + c_0\beta_0$. Then, the coefficients of the polynomial, c are computed with*

$$M(n) + M(\left\lfloor \frac{n}{2} \right\rfloor) \textit{ multiplications and}$$
$$A(n) + A(\left\lfloor \frac{n}{2} \right\rfloor) + 2\left\lfloor \frac{n}{2} \right\rfloor - 1 \textit{ additions}$$

by using any multiplication method.

Proof.

$$\begin{aligned}
c_0 &= a_0b_0 \\
c_1 &= a_0b_1 + a_1b_0 + a_1b_1 \\
c_2 &= a_0b_2 + a_2b_0 + a_1b_1 \\
c_3 &= a_0b_3 + a_3b_0 + a_1b_2 + a_2b_1 + a_1b_3 + a_3b_1 \\
&\vdots \\
c_{2n-3} &= a_{n-2}b_{n-1} + a_{n-1}b_{n-2} + \ell \cdot a_{n-1}b_{n-1} \\
c_{2n-2} &= a_{n-1}b_{n-1}
\end{aligned}$$

where $\ell = 1$ if $n - 1$ is odd, otherwise, $\ell = 0$. There are extra terms when we compare this multiplication with ordinary multiplication. The extra terms can be expressed with $a_{2i+}b_{2j+1}$, where $0 \le i, j \le \left\lfloor \frac{n}{2} - 1 \right\rfloor - 1$. These elements correspond to multiplication of two $\left\lfloor \frac{n}{2} \right\rfloor$-term polynomials. Therefore, the total multiplication complexity is $M(n) + M(\left\lfloor \frac{n}{2} \right\rfloor)$. We need $2\left\lfloor \frac{n}{2} \right\rfloor - 1$ extra additions to combine these. Similarly, the total addition complexity is $A(n) + A(\left\lfloor \frac{n}{2} \right\rfloor) + 2\left\lfloor \frac{n}{2} \right\rfloor - 1$. ■

Example 3.2.2 *Let $n = p^j$, where p is a prime number and j is a positive integer. Let $a = a_{n-1}\beta_{n-1} + \cdots + a_0\beta_0$ and $b = b_{n-1}\beta_{n-1} + \cdots + b_0\beta_0$ be n-term polynomials over $GF(2)$ and $a \cdot b = c = c_{2n-2}\beta_{2n-2} + \cdots + c_0\beta_0$. Then, by using Karatsuba multiplication method [39],*

1. *If $p = 2$, the required number of multiplications is $n^{log_2 3} + \left\lfloor \frac{n}{2} \right\rfloor^{log_2 3}$ and the required number of additions is $8n^{log_2 3} - 11n + 3$.*

2. *If* $p = 3$, *the required number of multiplications is* $n^{log_3 6} + \left\lfloor \frac{n}{2} \right\rfloor^{log_3 6}$ *and the required number of additions is* $\frac{116}{15} n^{log_3 6} - \frac{29}{5} n + \frac{7}{5}$.

We give an example to show that Theorem 3.2.1 is working.

Example 3.2.3 *Let* $a = a_3\beta_3 + a_2\beta_2 + a_1\beta_1 + a_0\beta_0$ *and* $b = b_3\beta_3 + b_2\beta_2 + b_1\beta_1 + b_0\beta_0$ *be* 4*-term polynomials over* $GF(2)$ *in Charlier polynomial representation. Let* $a \cdot b = c = c_6\beta_6 + \cdots + c_0\beta_0$. *Then,*

$$\begin{aligned}
c_0 &= a_0 b_0 \\
c_1 &= a_0 b_1 + a_1 b_0 + \underline{a_1 b_1} \\
c_2 &= a_0 b_2 + a_2 b_0 + a_1 b_1 \\
c_3 &= a_0 b_3 + a_3 b_0 + a_1 b_2 + a_2 b_1 + \underline{a_1 b_3 + a_3 b_1} \\
c_4 &= a_1 b_3 + a_3 b_1 + a_2 b_2 \\
c_5 &= a_2 b_3 + a_3 b_2 + \underline{a_3 b_3} \\
c_6 &= a_3 b_3
\end{aligned}$$

a_1b_1, $(a_1b_3 + a_3b_1)$ *and* a_3b_3 *are the extra terms when we compare this multiplication with ordinary multiplication. The computation of these extra terms can be achieved by the following method:*
Let $x_0 = a_1, y_0 = b_1, x_1 = a_3$ *and* $y_1 = b_3$. *Then, the extra terms can be computed as follows:*

$$\begin{aligned}
m'_1 &= x_0 y_0, \\
m'_2 &= (x_0 + x_1)(y_0 + y_1) - m'_1 - m'_3, \\
m'_3 &= x_1 y_1
\end{aligned}$$

The computation of extra terms requires 3 multiplications and 4 additions. One needs at most $9 + 3 = 12$ *multiplications and* $24 + 4 + 3 = 31$ *additions by using Karatsuba method to compute* $a \cdot b = c = c_6\beta_6 + \cdots + c_0\beta_0$.

Remark 3.2.4 *Note that some or all elements of extra terms may be obtained without any cost, i.e. these are computed in n-term polynomials product. This, of course, depends your choice on multiplication method. Therefore, this reduces multiplication and addition complexity. However, in this chapter, we give upper bounds.*

Example 3.2.5 *Let us consider Example 3.2.3. Consider two* 4*-term polynomials in standard representation* $a(x) = \sum_{i=0}^{3} a_i x^i$ *and* $b(x) = \sum_{i=0}^{3} b_i x^i$*. Karatsuba algorithm computes the product* $c(x) = a(x)b(x) = \sum_{i=0}^{6} c_i x^i$ *with the following 9 multiplications:*

$$\begin{aligned}
m_0 &= a_0 b_0 \\
m_1 &= a_1 b_1 \\
m_2 &= a_2 b_2 \\
m_3 &= a_3 b_3 \\
m_4 &= (a_0 + a_1)(b_0 + b_1) \\
m_5 &= (a_0 + a_2)(b_0 + b_2) \\
m_6 &= (a_1 + a_3)(b_1 + b_3) \\
m_7 &= (a_2 + a_3)(b_2 + b_3) \\
m_8 &= (a_0 + a_1 + a_2 + a_3)(b_0 + b_1 + b_2 + b_3)
\end{aligned}$$

The extra terms in Charlier polynomial representation, i.e. $a_1 b_1$, $(a_1 b_3 + a_3 b_1)$ *and* $a_3 b_3$*, are obtained without any cost:*

$$m'_1 = m_0, m'_2 = m_6 + m_0 + m_3, m'_3 = m_3$$

Thus, one needs 9 multiplications and 24+3=27 additions by using Karatsuba method to compute $c = a \cdot b = c_6\beta_6 + \cdots + c_0\beta_0$

Now, we show how modular reduction process can be performed for irreducible Charlier polynomials.

3.2.1 Irreducible Charlier Binomials

Selected irreducible Charlier binomials are given in Table 3.2.

3.2.1.1 Reduction

By using irreducible Charlier binomial, one can perform reduction operation as follows:
Let $f = \beta_n + \beta_0$ be an irreducible polynomial of degree n over $GF(2)$. Let $n \leq i \leq 2n-2$. Then,

Table 3.2: Irreducible Charlier Binomials

$\beta_2 + \beta_0$	$\beta_3 + \beta_0$
$\beta_5 + \beta_0$	$\beta_7 + \beta_0$
$\beta_9 + \beta_0$	$\beta_{41} + \beta_0$
$\beta_{63} + \beta_0$	$\beta_{71} + \beta_0$
$\beta_{105} + \beta_0$	$\beta_{127} + \beta_0$
$\beta_{169} + \beta_0$	$\beta_{177} + \beta_0$

$$\begin{aligned}
\beta_n\beta_{i-n} &= \beta_i + \beta_{i-1} \cdot \ell \\
\beta_0\beta_{i-n} &= \beta_i + \beta_{i-1} \cdot \ell \\
\beta_i &= \beta_{i-n} + \beta_{i-1} \cdot \ell \\
\beta_i &= \beta_{i-n} + (\beta_{i-n-1}\beta_n + \beta_{i-2} \cdot \ell_1) \cdot \ell \\
\beta_i &= \beta_{i-n} + \beta_{i-n-1} \cdot \ell
\end{aligned}$$

If $i - n$ and n or $i - n - 1$ and n are both odd, then $\ell = 1$ or $\ell_1 = 1$, respectively. Otherwise, $\ell = 0$ or $\ell_1 = 0$. Note that $\beta_n = 1$. $\ell \cdot \ell_1 = 0$ since if $i - n$ is odd, then, $i - n - 1$ is even. Same trick is applicable for trinomial case.

3.2.2 Irreducible Charlier Trinomials

Table 3.3 tabulates selected irreducible Charlier trinomials. According to Table 3.3, it should be noted that recommended NIST or SEC binary fields such as $GF(2^{113})$, $GF(2^{131})$, $GF(2^{233})$ and $GF(2^{283})$ can be constructed with irreducible Charlier trinomials [32] and [36].

3.2.2.1 Reduction

By using irreducible Charlier trinomial, one can perform reduction operation as follows:
Let $f = \beta_n + \beta_k + \beta_0$ be an irreducible polynomial of degree n over $GF(2)$ and k be even. Let

Table 3.3: Irreducible Charlier Trinomials

$\beta_{113} + \beta_{10} + \beta_0$	$\beta_{131} + \beta_{29} + \beta_0$
$\beta_{167} + \beta_2 + \beta_0$	$\beta_{169} + \beta_3 + \beta_0$
$\beta_{171} + \beta_4 + \beta_0$	$\beta_{187} + \beta_3 + \beta_0$
$\beta_{211} + \beta_{10} + \beta_0$	$\beta_{221} + \beta_2 + \beta_0$
$\beta_{227} + \beta_3 + \beta_0$	$\beta_{231} + \beta_{13} + \beta_0$
$\beta_{233} + \beta_{11} + \beta_0$	$\beta_{233} + \beta_{17} + \beta_0$
$\beta_{283} + \beta_3 + \beta_0$	$\beta_{283} + \beta_{14} + \beta_0$
$\beta_{291} + \beta_{21} + \beta_0$	$\beta_{311} + \beta_3 + \beta_0$
$\beta_{323} + \beta_{22} + \beta_0$	$\beta_{331} + \beta_{21} + \beta_0$
$\beta_{347} + \beta_{28} + \beta_0$	$\beta_{359} + \beta_{24} + \beta_0$
$\beta_{401} + \beta_{13} + \beta_0$	$\beta_{403} + \beta_{26} + \beta_0$
$\beta_{419} + \beta_{14} + \beta_0$	$\beta_{443} + \beta_9 + \beta_0$
$\beta_{463} + \beta_{11} + \beta_0$	$\beta_{469} + \beta_{23} + \beta_0$
$\beta_{511} + \beta_{11} + \beta_0$	$\beta_{541} + \beta_3 + \beta_0$
$\beta_{551} + \beta_{18} + \beta_0$	$\beta_{557} + \beta_{11} + \beta_0$

$n \le i \le 2n - 2$. Then,

$$\begin{aligned}
\beta_n\beta_{i-n} &= \beta_i + \beta_{i-1} \cdot \ell \\
(\beta_k + \beta_0)\beta_{i-n} &= \beta_i + \beta_{i-1} \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n} + \beta_{i-1} \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n} + (\beta_{i-n+k-1} + \beta_{i-n-1}) \cdot \ell
\end{aligned}$$

If $i - n$ and n are odd, then $\ell = 1$. Otherwise, $\ell = 0$.

Let $f = \beta_n + \beta_k + \beta_0$ be an irreducible polynomial of degree n over $GF(2)$ and k be odd. Let $n \le i \le 2n - 2$. Then,

$$\begin{aligned}
\beta_n\beta_{i-n} &= \beta_i + \beta_{i-1} \cdot \ell \\
(\beta_k + \beta_0)\beta_{i-n} &= \beta_i + \beta_{i-1} \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n} + (\beta_{i+k-n-1} + \beta_{i-1}) \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n} + \beta_{i-n-1} \cdot \ell
\end{aligned}$$

If $i - n$ and n are odd, then $\ell = 1$. Otherwise, $\ell = 0$.

3.2.3 Reduction Complexity

Table 3.4 shows reduction complexity for irreducible Charlier binomials and trinomials.

Table 3.4: Reduction Complexity

	Form	$\#XOR$
Charlier Binomial	$\beta_n + \beta_0$	$\frac{3n}{2}$
Charlier Trinomial	$\beta_n + \beta_k + \beta_0$	$3n$

Remember that the cost of reduction process in polynomial basis representation strictly depends on the choice of reduction polynomial. Then, if one uses trinomial or pentanomial, reduction process requires $2n$ or $4n$ XOR gates, respectively.

3.3 Multiplication Complexity

In this section, we give modular multiplication complexity of multipliers in view of $\#AND$ and $\#XOR$ gates. Let $n = p^j$. Table 3.5 compares the space complexity and time complexity of selected multipliers. Note that this table is prepared by using Karatsuba multiplication method for Charlier basis [39]. According to Table 3.5, Charlier polynomial representation has better complexity than Dickson polynomial representation and ONB II. Therefore, binary fields can be constructed with low weight Charlier polynomials efficiently when there does not exist ONB for the corresponding extension. Remember that we give upper bounds for Charlier binomials and trinomials. The complexity of the field multiplication for Charlier polynomials can be further reduced by cleverly combining computed values (see Example 3.2.5). Therefore, for some cases, multiplication complexity for Charlier polynomial representation is also comparable with ONB I.

Remark 3.3.1 *NIST recommended binary field $GF(2^{283})$ can be constructed efficiently by using Charlier polynomials since there is no ONB for the corresponding extension.*

Table 3.5: Space Complexity Comparison of Selected Multipliers

	p	$\#AND$	$\#XOR$	Critical Delay
Charlier Binomial	2	$n^{log_2 3} + \lfloor \frac{n}{2} \rfloor^{log_2 3}$	$8n^{log_2 3} - 10n + 3$	$3log_2(n)T_X + T_A$
Charlier Binomial	3	$n^{log_3 6} + \lfloor \frac{n}{2} \rfloor^{log_3 6}$	$\frac{116}{15}n^{log_3 6} - \frac{29}{5}n + \frac{7}{5}$	$4log_3(n)T_X + T_A$
Charlier Trinomial	2	$n^{log_2 3} + \lfloor \frac{n}{2} \rfloor^{log_2 3}$	$8n^{log_2 3} - 8n + 3$	$3log_2(n)T_X + T_A$
Charlier Trinomial	3	$n^{log_3 6} + \lfloor \frac{n}{2} \rfloor^{log_3 6}$	$\frac{116}{15}n^{log_3 6} - \frac{14}{5}n + \frac{7}{5}$	$4log_3(n)T_X + T_A$
Polynomial Basis [39]	2	$n^{log_2 3}$	$6n^{log_2 3} - 8n + 2$	$(3log_2(n) - 1)T_X + T_A$
Polynomial Basis [39]	3	$n^{log_3 6}$	$\frac{29}{5}n^{log_3 6} - 8n + \frac{11}{5}$	$(4log_3(n) - 1)T_X + T_A$
Dickson Binomial [20]	2	$2n^{log_2 3}$	$11n^{log_2 3} - 11n$	$(2log_2(n) + 1)T_X + T_A$
Dickson Binomial [20]	3	$2n^{log_3 6}$	$\frac{48}{5}n^{log_3 6} - 11n + \frac{3}{5}$	$(3log_3(n) + 1)T_X + T_A$
Dickson Trinomial [20]	2	$2n^{log_2 3}$	$11n^{log_2 3} - 4n + 1$	$(2log_2(n) + 6)T_X + T_A$
Dickson Trinomial [20]	3	$2n^{log_3 6}$	$\frac{48}{5}n^{log_3 6} - 2n + \frac{1}{5}$	$(3log_3(n) + 6)T_X + T_A$
ONB I [11]	2	$n^{log_2 3} + n$	$\frac{11}{2}n^{log_2 3} - 4n - \frac{1}{2}$	$(2log_2(n) + 1)T_X + T_A$
ONB I [11]	3	$n^{log_3 6} + n$	$\frac{24}{5}n^{log_3 6} - 3n - \frac{4}{5}$	$(3log_3(n) + 1)T_X + T_A$
ONB II [11]	2	$2n^{log_2 3}$	$11n^{log_2 3} - 12n + 1$	$(2log_2(n) + 1)T_X + T_A$
ONB II [11]	3	$2n^{log_3 6}$	$\frac{48}{5}n^{log_3 6} - 10n - \frac{2}{5}$	$(3log_3(n) + 1)T_X + T_A$

3.4 Other Arithmetic Operations

3.4.1 Squaring

Let $a = a_{n-1}\beta_{n-1} + \cdots + a_0\beta_0$ be n-term polynomial over $GF(2)$ and $a^2 = c = c_{2n-2}\beta_{2n-2} + \cdots + c_0\beta_0$. Then,

$$c = a_{n-1}\beta_{2n-2} + a_{n-1} \cdot \ell\beta_{2n-3} + \cdots + 0\beta_3 + a_1\beta_2 + a_1\beta_1 + a_0\beta_0,$$

where $\ell = 1$ if $n - 1$ is odd. Otherwise, $\ell = 0$. Proof of squaring is very similar to Theorem 3.2.1. Note that the cost of squaring in Charlier polynomial representation is just reduction.

3.4.2 Inversion

Let a and b be the polynomials in Charlier representation over $GF(2)$ of degree $(m - 1)$ and $(k - 1)$, respectively and $m > k > 0$. Then, there exist polynomials q (quotient) and r (remainder) in $GF(2)[x]$ such that

$$a = q \cdot b + r \text{ with } 0 \leq r < b.$$

Note that $\deg(r) < \deg(b)$. The idea of the division algorithm is the consequence of the multiplication of β_i and β_j which is more complicated compared to the usual multiplication in $GF(2)[x]$ (see Theorem 3.1.3). By using division algorithm recursively, one easily obtains the Euclidean algorithm for F to find the greatest common divisors of a and b. Therefore, Euclidean algorithm can be used efficiently to find an inversion of an element. Moreover, one can use the following two well-known methods.

Problem 3.4.1 *Let $a = a_{n-1}\beta_{n-1} + \cdots + a_0\beta_0$. Find a^{-1} such that $a \cdot a^{-1} = 1$.*

- Compute a^{2^n-2} equal to a^{-1}. This method is based on Fermat theorem.
- Use extended Euclidean algorithm in polynomial representation. By using Algorithm 2, conversion between polynomial representation and Charlier representation can be achieved very efficiently. Inversion of an element a can be computed as follows:
    1. Convert a given in Charlier representation to polynomial representation.
    2. Compute the inverse a^{-1} of a modulo $f(x)$ by using extended Euclidean algorithm.
    3. Convert a^{-1} expressed in polynomial representation to Charlier representation.

CHAPTER 4

A NEW REPRESENTATION OF ELEMENTS OF BINARY FIELDS WITH SUBQUADRATIC SPACE COMPLEXITY

In this chapter, Hermite polynomial representation is proposed as an alternative way to represent finite fields of characteristic two. We show that multiplication in Hermite polynomial representation can be achieved with subquadratic space complexity. This representation enables us to find binomial, trinomial or quadranomial irreducible polynomials which allows us faster modular reduction over binary fields when there is no desirable such low weight irreducible polynomial in other representations. We then show that the product of two elements in Hermite polynomial representation can be performed as Toeplitz matrix-vector product. This representation is very interesting for NIST recommended binary field $GF(2^{571})$ since there is no ONB for the corresponding extension. We note that an advantage of this representation is that it can be used to obtain more efficient finite field arithmetic.

This chapter is organized as follows: Section 4.1 describes Hermite polynomial and gives some general results on Hermite polynomials in $GF(2)[x]$. In Section 4.2, we present the general method to multiply two polynomials in Hermite polynomial representation and give the total arithmetic complexity. In Section 4.3, we give polynomial multiplication in Hermite basis with Toeplitz matrix-vector multiplication design. We compare multipliers in view of $\#AND$ and $\#XOR$ gates and the complexity of the multiplication of two basis elements in Section 4.4.

The material presented in this chapter is partially included in [3].

4.1 Hermite Polynomials

In this section, we give preliminaries and describe a new representation of binary fields. Hermite polynomials are in the class of orthogonal polynomials. Hermite polynomials have several applications in quantum mechanics, numerical computation, algorithms [1], [5]. There are two kinds of Hermite polynomials: probabilists and physicists. In this chapter, we are interested in probabilists Hermite polynomials [1].

Definition 4.1.1 *[9] The Hermite polynomials are $H_0(x) = 1$, $H_1(x) = x$ with the recursion*

$$H_n(x) = x \cdot H_{n-1}(x) - (n-1) \cdot H_{n-2}(x)$$

for $n \geq 2$.

It should be noted that $deg(H_n) = n$. We give the Hermite polynomials in $GF(2)[x]$ for $n \leq 10$ in Table 4.1.

Table 4.1: Hermite Polynomials in $GF(2)[x]$

$H_0(x)$	1
$H_1(x)$	x
$H_2(x)$	$x^2 + 1$
$H_3(x)$	$x^3 + x$
$H_4(x)$	$x^4 + 1$
$H_5(x)$	$x^5 + x$
$H_6(x)$	$x^6 + x^4 + x^2 + 1$
$H_7(x)$	$x^7 + x^5 + x^3 + x$
$H_8(x)$	$x^8 + 1$
$H_9(x)$	$x^9 + x$
$H_{10}(x)$	$x^{10} + x^8 + x^2 + 1$

4.1.1 Conversion of Coefficients From Polynomial Representation to Hermite Polynomial Representation

The polynomial basis $\{1, x, x^2, \cdots, x^{n-1}\}$ is usually preferred to represent the elements of $GF(2)[x]$, where x is a root of an irreducible polynomial of degree n over $GF(2)$. Let

$a(x) = a'_{n-1}x^{n-1} + \cdots + a'_1 x + a'_0$, where $a'_i \in GF(2)$ be a polynomial with the standard representation. Let $H_n(x) = \beta_n$ be the n-th Hermite polynomial in $GF(2)[x]$, where $n \geq 0$. The polynomial $a(x)$ can be represented by using Hermite polynomials as $a = a_{n-1}\beta_{n-1} + \cdots + a_1\beta_1 + a_0\beta_0$, where $a_i \in GF(2)$. Algorithm 3 shows how conversion of coefficients from polynomial representation to Hermite polynomial representation can be performed.

Algorithm 3 Conversion of Coefficients From Polynomial Representation to Hermite Polynomial Representation

```
Input: $a(x) = \sum_{i=0}^{n-1} a'_i x^i$
Output: $(a_0, a_1, \cdots, a_{n-1})$, where $a = \sum_{i=0}^{n-1} a_i\beta_i$
 1: $T \leftarrow a$
 2: for $i = n$ downto 1 do
 3:     if $deg(T) = i$ then
 4:         $a_i \leftarrow 1$
 5:         $T \leftarrow T + \beta_i$
 6:     else
 7:         $a_i \leftarrow 0$
 8:     end if
 9: end for
10: $a_0 \leftarrow T$
```

Note that Algorithm 3 is self-inverse. That is in Algorithm 3, if the input is a polynomial representation, then the output is the Hermite representation, and conversely if the input is an Hermite representation, then the output is the polynomial representation.

4.1.2 Hermite Basis

A basis for the finite field $GF(2^n)$ is a set of n elements $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\} \in GF(2^n)$ such that every element of the binary field can be represented uniquely as a linear combination of basis elements. For a given $a \in GF(2^n)$, we can write

$$a = \sum_{i=0}^{n-1} a_i \cdot \beta_i$$

where $a_i \in GF(2)$ for $0 \leq i \leq n-1$.

Note that $GF(2)[x]$ is a free vector space over $GF(2)$ with the free basis $\{1, x, x^2, \cdots, x^{n-1}, \cdots\}$.

Moreover, $GF(2)[x]$ is a commutative algebra over $GF(2)$. In particular for arbitrary polynomials $f_1(x), f_2(x) \in GF(2)[x]$, the product $f_1(x) \cdot f_2(x)$ is defined in $GF(2)[x]$. Next using another free basis we obtain another commutative algebra on $GF(2)[x]$ over $GF(2)$. Assume that $\{\beta_0 = I, \beta_1, \cdots, \beta_{n-1}, \cdots\}$ is the Hermite basis. Let $(F, +)$ be the free vector space over $GF(2)$ generated by B. We explain this in detail in Theorem 4.1.8. We put a commutative algebra on F over $GF(2)$ by introducing the multiplication $\cdot$ on F. Proof of the following theorem is given after defining the algebra (see 4.1.3).

Theorem 4.1.2 *Let $H_n(x) = \beta_n$ be the n-th Hermite polynomial in $GF(2)[x]$, where $n \geq 0$. Then, for all $i, j \geq 0$ the Hermite basis $\{\beta_0, \beta_1, \cdots, \beta_{n-1}, \cdots\}$ satisfies the following equation*

$$\beta_i \cdot \beta_j = \beta_{i+j} + \ell \cdot \beta_{i+j-2}$$

where $\ell \in GF(2)$ is defined as

$$\ell = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are both odd number} \\ 0 & \text{otherwise} \end{cases}$$

4.1.3 The Algebra

Let $B = \{\beta_0 = I, \beta_1, \cdots, \beta_{n-1}, \cdots\}$ be the Hermite basis. Let $(F, +)$ be the free vector space over $GF(2)$ generated by B. The multiplication $\cdot$ on F has the following properties:

i. For the zero element 0 of $GF(2)$ we have

$$\beta_i \cdot 0 = 0 \cdot \beta_i = 0$$

for all $i = 0, 1, \cdots$.

ii. The identity element of F is the element $\beta_0 = I$ that is

$$\beta_i \cdot I = I \cdot \beta_i = \beta_i$$

for all $i = 0, 1, \cdots$.

iii. For all $c \in GF(2)$ and $\beta_i, \beta_j \in B$ we have

$$c(\beta_i \cdot \beta_j) = (c\beta_i) \cdot \beta_i = \beta_1 \cdot (c\beta_2).$$

Note that it is enough to define and check the properties of the multiplication only on the Hermite basis, as the multiplication is extended to F via linearity.

Remark 4.1.3 *The multiplication $\cdot$ is associative, that is, for any $\beta_i, \beta_j, \beta_k \in B$ we have*

$$\beta_i \cdot (\beta_j \cdot \beta_k) = (\beta_i \cdot \beta_j) \cdot \beta_k.$$

Proof. Using Theorem 4.1.2 we obtain that

$$\beta_i \cdot (\beta_j \cdot \beta_k) = \beta_{i+j+k} + \beta_{i+j+k-2}(\ell_1 + \ell_2) + \beta_{i+j+k-4}(\ell_1 \cdot \ell_2), \tag{4.1}$$

and

$$(\beta_i \cdot \beta_j) \cdot \beta_k) = \beta_{i+j+k} + \beta_{i+j+k-2}(\ell_1' + \ell_2') + \beta_{i+j+k-4}(\ell_1' \cdot \ell_2') \tag{4.2}$$

where $\ell_1, \ell_2, \ell_1', \ell_2' \in GF(2)$. If j and k are both odd numbers, then $\ell_1 = 1$. If i and $j + k$ are both odd numbers, then $\ell_2 = 0$. If i and j are both odd numbers, then $\ell_1' = 1$. If $i + j$ and k are both odd numbers, then $\ell_2 = 0$. For the other cases $\ell_1 = 0, \ell_2 = 0, \ell_1' = 0, \ell_2' = 0$. Note that ℓ_1 and ℓ_2 cannot be equal to 1 at the same time since the addition of two odd integers is an even integer. This case is also applicable for ℓ_1' and ℓ_2'. Therefore, using 4.1 and 4.2 we conclude that

$$\beta_i \cdot (\beta_j \cdot \beta_k) = (\beta_i \cdot \beta_j) \cdot \beta_k).$$

■

Remark 4.1.4 *The distributive law holds; that is, for all $\beta_i, \beta_j, \beta_k \in B$*

$$\beta_i \cdot (\beta_j + \beta_k) = \beta_i \cdot \beta_j + \beta_i \cdot \beta_k,$$

$$(\beta_j + \beta_k) \cdot \beta_i = \beta_j \cdot \beta_i + \beta_k \cdot \beta_i.$$

Let $a = a_0 \cdot \beta_0 + \sum_{i=1}^{n-1} a_i \cdot \beta_i$ be an element of F where $a_i \in GF(2)$ for $i = 0, 1, \cdots, (n-1)$ and $m - 1$ is the largest positive integer such that $a_{n-1} = 1$. We define the degree $\deg(a)$ of the element $a \in F$ as follows:

$$\deg(a) = \begin{cases} -\infty & \text{if } a = 0, \\ 0 & \text{if } a = I, \\ n-1 & \text{otherwise.} \end{cases}$$

Note that $a = 0$ means all coefficients of a i.e. a_i for $i = 0, 1, \cdots, (n-1)$ are equal to 0. Similarly, $a = I$ if and only if $a_0 = 1$ and $a_1 = a_2 = \cdots = a_{n-1} = 0$.

We define the division algorithm on F as follows: Let a and b be elements of F over $GF(2)$ of degree $(m-1)$ and $(k-1)$, respectively and $m > k > 0$. Then, there exist polynomials q (quotient) and r (remainder) in F such that

$$a = q \cdot b + r \text{ with } 0 \leq r < b.$$

Note that $\deg(r) < \deg(b)$. The idea of the division algorithm is the consequence of the multiplication of β_i and β_j which is more complicated compared to the usual multiplication in $GF(2)[x]$ (see Theorem 4.1.2). By using division algorithm recursively, one easily obtains the Euclidean algorithm for F to find the greatest common divisors of a and b.

Remark 4.1.5 *Division algorithm for F allows a suitable generalization of the Euclidean algorithm. Thus, F is a Euclidean domain and also a principal ideal domain.*

Let $f = \sum_{i=0}^{n-1} f_i \cdot \beta_i$, $a = \sum_{i=0}^{m-1} a_i \cdot \beta_i$ and $b = \sum_{i=0}^{k-1} b_i \cdot \beta_i$ be arbitrary elements of F with $f_i, a_i, b_i \in GF(2)$. We define an equivalence relation on F using f as follows: we call $a \equiv b \pmod{f}$ if and only if $f|(a-b)$. This relation is reflexive, symmetric and transitive. Recall that $f \in F$, non-constant, is irreducible if it cannot be written as the product of two non-constant elements in F.

Remark 4.1.6 *Let f be an irreducible polynomial over $GF(2)$ of degree n. Then, $K = F/(f)$ is a finite field of dimension n over $GF(2)$.*

In Proposition 4.1.7 we show that any β_i cannot be obtained by linear combination of β_j's with $i \neq j$.

Proposition 4.1.7 *The set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\} \in GF(2^n)$ is linearly independent.*

Proof. We prove the proposition by using induction. Let $a_0\beta_0 = 0$. Then, it is clear that $a_0 = 0$ since $\beta_0 = 1$. This corresponds to $k = 1$ case. Assume that the set $\{\beta_0, \beta_1, \cdots, \beta_{n-2}\}$ is linearly independent, i.e. proposition is true for $k = n-1$.

$$a_0\beta_0 + a_1\beta_1 + \cdots + a_{n-2}\beta_{n-2} = 0 \Leftrightarrow a_0 = a_1 = \cdots = a_{n-2}$$

Now, we need to show that the theorem is true for $k = n$.

$$a_0\beta_0 + a_1\beta_1 + \cdots + a_{n-1}\beta_{n-1} = 0$$

Then, we obtain

$$a_{n-1}\beta_{n-1} = 0$$

Remember that $deg(\beta_{n-1}) > 0$ for $n > 1$ and β_{n-1} cannot be expressed as a linear combination of $\{\beta_0, \beta_1, \cdots, \beta_{n-2}\}$. Therefore, $a_{n-1} = 0$. Thus, the set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ is linearly independent. ■

The following theorem shows that $B = \{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ is a basis of $GF(2^n) \cong GF(2)[x]/(f)$ for an irreducible polynomial f.

Theorem 4.1.8 *Let $f = \sum_{i=0}^{n} f_i \cdot \beta_i$ be an irreducible polynomial of degree n in $GF(2)[x]$. The set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ forms a basis of $GF(2^n) \cong GF(2)[x]/(f)$.*

Proof. Since $GF(2^n)$ is a vector space, the set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ is linearly independent by Proposition 4.1.7 and each element in $GF(2^n)$ is uniquely expressed by using the set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ by using Algorithm 3, the set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ spans $GF(2^n)$. Then, the set $\{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ forms a basis of $GF(2^n) \cong GF(2)[x]/(f)$. ■

Note that the set $\{\beta_0, \beta_1, \cdots, \beta_n\}$ is linearly dependent since $\deg(f) = n$ and $\beta_n = \sum_{i=0}^{n-1} f_i\beta_i$.

Theorem 4.1.9 *F is isomorphic to $GF(2)[x]$.*

Now we give proof of Theorem 4.1.2

Proof. We will prove the theorem by induction on i and j. By using Table 4.1, the theorem is true for few terms. Assume that theorem is true for $i = n - 1$. Then we need to show that it is true for $i = n$. Let $j < n - 1$. We have four cases:

i. n is even and j is odd

ii. n is even and j is even

iii. n is odd and j is odd

iv. n is odd and j is even

i. Let n be even and j be odd. Note that $\beta_1 = x$

$$\begin{aligned}
\beta_n\beta_j &= (\beta_1\beta_{n-1} + \beta_{n-2})\beta_j \\
&= \beta_{n-1}(\beta_j\beta_1) + \beta_{n-2}\beta_j \\
&= \beta_{n-1}(\beta_{j+1} + \beta_{j-1}) + \beta_{n+j-2} \\
&= \beta_{n-j} + \beta_{n+j-2} + \beta_{n+j-2} \\
&= \beta_{n+j}
\end{aligned}$$

ii. Let n and j be even.

$$\begin{aligned}
\beta_n\beta_j &= (\beta_1\beta_{n-1} + \beta_{n-2})\beta_j \\
&= \beta_{n-1}(\beta_j\beta_1) + \beta_{n-2}\beta_j \\
&= \beta_{n-1}\beta_{j+1} + \beta_{n+j-2} \\
&= \beta_{n-j} + \beta_{n+j-2} + \beta_{n+j-2} \\
&= \beta_{n+j}
\end{aligned}$$

iii. Let n and j be odd. Remember that addition of two odd integers is even.

$$\begin{aligned}
\beta_n\beta_j &= (\beta_1\beta_{n-1})\beta_j \\
&= \beta_{n-1}(\beta_{j+1} + \beta_{j-1}) \\
&= \beta_{n+j} + \beta_{n+j-2}
\end{aligned}$$

iv. Let n be odd and j be even.

$$\begin{aligned}
\beta_n\beta_j &= (\beta_1\beta_{n-1})\beta_j \\
&= \beta_{n-1}\beta_{j+1} \\
&= \beta_{n+j}
\end{aligned}$$

Note that if $j = n - 1$ or $j = n$, then this case can be proved by considering the factors of β_j or β_{j+1} as shown above. ■

Note that addition of two elements in Hermite basis representation is just coefficient-wise.

Remark 4.1.10 *The set $B = \{\beta_0, \beta_1, \beta_2, \cdots\}$ is multiplicatively closed since it satisfies the following properties: i) $I = \beta_0 \in B$ ii) if $\beta_i, \beta_j \in B$, then $\beta_i \cdot \beta_j \in B$.*

4.2 Polynomial Multiplication Using Hermite Polynomials Over Binary Fields

In this section, we describe polynomial multiplication in Hermite polynomial representation for binary fields and give the total arithmetic complexity. Remember that multiplication in finite fields can be performed in two steps: multiplication of polynomials over $GF(2)$ and modular reduction over $GF(2^n)$. Therefore, we divide this section into multiplication and reduction parts. Throughout this section, $M(n)$ and $A(n)$ denote the minimum number of multiplications and the minimum number of additions for corresponding algorithm for two n-term polynomials multiplication, respectively. The required number of multiplications and additions to multiply polynomials in Hermite basis is given in the following theorem.

Theorem 4.2.1 *Let $a = a_{n-1}\beta_{n-1} + \cdots + a_0\beta_0$ and $b = b_{n-1}\beta_{n-1} + \cdots + b_0\beta_0$ be n-term polynomials over $GF(2)$ and $a \cdot b = c = c_{2n-2}\beta_{2n-2} + \cdots + c_0\beta_0$. Then, the coefficients of the polynomial c are computed with*

$$M(n) + M(\left\lfloor \frac{n}{2} \right\rfloor) \textit{ multiplications and}$$
$$A(n) + A(\left\lfloor \frac{n}{2} \right\rfloor) + 2\left\lfloor \frac{n}{2} \right\rfloor - 1 \textit{ additions}$$

by using any multiplication method.

Proof. By using Theorem 4.1.2, we write explicitly,

$$\begin{aligned}
c_0 &= a_0b_0 + a_1b_1 \\
c_1 &= a_0b_1 + a_1b_0 \\
c_2 &= a_0b_2 + a_2b_0 + a_1b_1 + a_1b_3 + a_3b_1 \\
&\vdots \\
c_{2n-3} &= a_{n-2}b_{n-1} + a_{n-1}b_{n-2} \\
c_{2n-2} &= a_{n-1}b_{n-1}
\end{aligned}$$

There are extra terms when we compare this multiplication with ordinary multiplication. The extra terms can be expressed with $a_{2i+1}b_{2j+1}$, where $0 \le i, j \le \left\lfloor \frac{n}{2} \right\rfloor - 1$. These elements correspond to multiplication of two $\left\lfloor \frac{n}{2} \right\rfloor$-term polynomials. Therefore, the total multiplication complexity is $M(n) + M(\left\lfloor \frac{n}{2} \right\rfloor)$. We need $2\left\lfloor \frac{n}{2} \right\rfloor - 1$ extra additions to combine these. Similarly, the total addition complexity is $A(n) + A(\left\lfloor \frac{n}{2} \right\rfloor) + 2\left\lfloor \frac{n}{2} \right\rfloor - 1$. ■

Example 4.2.2 *Let $n = p^j$, where p is a prime number and j is a positive integer. Let $a = a_{n-1}\beta_{n-1} + \cdots + a_0\beta_0$ and $b = b_{n-1}\beta_{n-1} + \cdots + b_0\beta_0$ be n-term polynomials over $GF(2)$ and $a \cdot b = c = c_{2n-2}\beta_{2n-2} + \cdots + c_0\beta_0$. Then, by using Karatsuba multiplication method [39],*

1. *If $p = 2$, the required number of multiplications is $n^{log_2 3} + \left\lfloor \frac{n}{2} \right\rfloor^{log_2 3}$ and the required number of additions is $8n^{log_2 3} - 11n + 3$.*
2. *If $p = 3$, the required number of multiplications is $n^{log_3 6} + \left\lfloor \frac{n}{2} \right\rfloor^{log_3 6}$ and the required number of additions is $\frac{116}{15}n^{log_3 6} - \frac{29}{5}n + \frac{7}{5}$.*

Remark 4.2.3 *Let $a = a_{n-1}\beta_{n-1} + \cdots + a_0\beta_0$ be n-term polynomial over $GF(2)$ and $a^2 = c = c_{2n-2}\beta_{2n-2} + \cdots + c_0\beta_0$. Then,*

$$c = a_{n-1}\beta_{2n-2} + (a_{n-1} \cdot \ell + a_{n-2})\beta_{2n-3} + \cdots + (a_3 + a_2)\beta_4 + a_1\beta_2 + a_1\beta_1 + (a_1 + a_0)\beta_0$$

Proof of this remark is very similar to Theorem 4.2.1. Note that the cost of squaring in Hermite polynomial representation is just reduction and $\left\lfloor \frac{n}{2} \right\rfloor$ additions.

We give an example to show that Theorem 4.2.1 is working.

Example 4.2.4 *Let $a = a_3\beta_3 + a_2\beta_2 + a_1\beta_1 + a_0\beta_0$ and $b = b_3\beta_3 + b_2\beta_2 + b_1\beta_1 + b_0\beta_0$ be 4-term polynomials over $GF(2)$. Let $a \cdot b = c = c_6\beta_6 + \cdots + c_0\beta_0$. Then,*

$$\begin{aligned}
c_0 &= a_0b_0 + a_1b_1 \\
c_1 &= a_0b_1 + a_1b_0 \\
c_2 &= a_0b_2 + a_2b_0 + a_1b_1 + a_1b_3 + a_3b_1 \\
c_3 &= a_0b_3 + a_3b_0 + a_1b_2 + a_2b_1 \\
c_4 &= a_1b_3 + a_3b_1 + a_2b_2 + a_3b_3 \\
c_5 &= a_2b_3 + a_3b_2 \\
c_6 &= a_3b_3
\end{aligned}$$

a_1b_1, $(a_1b_3 + a_3b_1)$ and a_3b_3 are the extra terms when we compare this multiplication with polynomial basis representation. The computation of these extra terms can be achieved by the following method:
Let $x_0 = a_1$, $y_0 = b_1$, $x_1 = a_3$ and $y_1 = b_3$. Then, the extra terms can be computed as follows:

$$\begin{aligned} m_1 &= x_0y_0, \\ m_2 &= (x_0 + x_1)(y_0 + y_1) - m_1 - m_3, \\ m_3 &= x_1y_1 \end{aligned}$$

The computation of extra terms requires 3 multiplications and 4 additions. One needs $9+3 = 12$ multiplications and $24 + 4 + 3 = 31$ additions by using Karatsuba method to compute $a \cdot b = c = c_6\beta_6 + \cdots + c_0\beta_0$.

Remark 4.2.5 *Note that some or all elements of extra terms may be obtained without any cost, i.e. these are computed in n-term polynomials product. This, of course, depends your choice on multiplication method. Therefore, this reduces multiplication and addition complexity.*

To obtain better multiplication complexity, we recall Karatsuba multiplication method over $GF(2)$ and then we present our observation.
Let a and b be polynomials of degree $n - 1$ over $GF(2)$ where n is a power of 2. By splitting a and b into two blocks of size $\frac{n}{2}$, one obtains

$$\begin{aligned} a &= a_Lx^{\frac{n}{2}} + a_R \\ b &= b_Lx^{\frac{n}{2}} + b_R \end{aligned}$$

Then, multiplication of a and b, i.e., $c = a \cdot b$ by using Karatsuba multiplication method is computed as follows:

$$\begin{aligned} c &= a \cdot b \\ &= (a_Lx^{\frac{n}{2}} + a_R)(b_Lx^{\frac{n}{2}} + b_R) \\ &= a_Lb_Lx^n + (a_Lb_R + a_Rb_L)x^{\frac{n}{2}} + a_Rb_R \\ &= a_Lb_Lx^n + ((a_L + a_R)(b_L + b_R) - a_Lb_L - a_Rb_R)x^{\frac{n}{2}} + a_Rb_R \end{aligned}$$

Karatsuba multiplication method uses divide-conquer idea recursively to obtain better multiplication complexity. Therefore, the complexity of this idea can be viewed as

$$M(n) = 3M(\frac{n}{2})$$

Remark 4.2.6 *Let $a = a_{n-1}x^{n-1} + \cdots + a_1x + a_0$ and $b = b_{n-1}x^{n-1} + \cdots + b_1x + b_0$ be n-term polynomials over $GF(2)$ where n is a power of 2. Let d and e be $(\frac{n}{2})$-term polynomials over $GF(2)$ with the coefficients a_{1+2i} and b_{1+2i}, where $0 \leq i \leq \frac{n}{2}$, respectively. Let $c = a \cdot b$ and $f = d \cdot e$. Assume that we have all required multiplications to compute the coefficients of c by using Karatsuba multiplication method. Then, the coefficients of f are obtained with at most $\frac{(\frac{n}{2})^{log_2 3}}{3} = \frac{n^{log_2 3}}{9}$ multiplications. Similarly, for $n = 3^k$, where k is a positive integer, the required number of multiplications is $\frac{(\frac{n}{2})^{log_3 6}}{3}$.*

Remark 4.2.6 is the consequence of recursive structure of Karatsuba multiplication method. Note that this observation can be used in [2] since they have similar structure.

Example 4.2.7 *Let us remember Example 4.2.4. Consider two 4-term polynomials in standard representation $a(x) = \sum_{i=0}^{3} a_i x^i$ and $b(x) = \sum_{i=0}^{3} b_i x^i$. Karatsuba algorithm computes the product $c(x) = a(x)b(x) = \sum_{i=0}^{6} c_i x^i$ with the following 9 multiplications:*

$$\begin{aligned}
m_0 &= a_0b_0 \\
m_1 &= a_1b_1 \\
m_2 &= a_2b_2 \\
m_3 &= a_3b_3 \\
m_4 &= (a_0 + a_1)(b_0 + b_1) \\
m_5 &= (a_0 + a_2)(b_0 + b_2) \\
m_6 &= (a_1 + a_3)(b_1 + b_3) \\
m_7 &= (a_2 + a_3)(b_2 + b_3) \\
m_8 &= (a_0 + a_1 + a_2 + a_3)(b_0 + b_1 + b_2 + b_3)
\end{aligned}$$

The extra terms in Hermite polynomial representation, i.e. a_1b_1, $(a_1b_3 + a_3b_1)$ and a_3b_3, are obtained without any cost:

$$m'_1 = m_0, m'_2 = m_6 + m_0 + m_3, m'_3 = m_3$$

Thus, one needs 9 multiplications and 24+3=27 additions by using Karatsuba method to compute $c = a \cdot b = c_6\beta_6 + \cdots + c_0\beta_0$

Now, we show how reduction process is performed for irreducible Hermite polynomials.

4.2.1 Irreducible Hermite Binomials

Table 4.2 gives selected irreducible Hermite binomials. We have mainly two kinds of irreducible Hermite binomials of the form: $\beta_n + \beta_0$ and $\beta_n + \beta_1$.

Table 4.2: Irreducible Hermite Binomials

$\beta_3 + \beta_0$	$\beta_2 + \beta_1$
$\beta_7 + \beta_0$	$\beta_4 + \beta_1$
$\beta_9 + \beta_0$	$\beta_6 + \beta_1$
$\beta_{15} + \beta_0$	$\beta_{22} + \beta_1$
$\beta_{63} + \beta_0$	$\beta_{28} + \beta_1$
$\beta_{127} + \beta_0$	$\beta_{46} + \beta_1$
$\beta_{471} + \beta_0$	$\beta_{52} + \beta_1$

4.2.1.1 Reduction

By using irreducible Hermite binomial, one can perform reduction operation as follows:
Let $f = \beta_n + \beta_0$ be an irreducible polynomial of degree n over $GF(2)$. Let $n \leq i \leq 2n-2$. Then,

$$\begin{aligned}
\beta_n \beta_{i-n} &= \beta_i + \beta_{i-2} \cdot \ell \\
\beta_0 \beta_{i-n} &= \beta_i + \beta_{i-2} \cdot \ell \\
\beta_i &= \beta_{i-n} + \beta_{i-2} \cdot \ell \\
\beta_i &= \beta_{i-n} + (\beta_{i-n-2}\beta_n + \beta_{i-n-4} \cdot \ell) \cdot \ell \\
\beta_i &= \beta_{i-n} + (\beta_{i-n-2} + \beta_{i-n-4}) \cdot \ell
\end{aligned}$$

If $i-n$ is odd, then $\ell = 1$. Otherwise, $\ell = 0$. Note that $\beta_n = \beta_0$ and if $i-n$ is odd, then $i-n-2$ is also odd.

Let $f = \beta_n + \beta_1$ be an irreducible polynomial of degree n over $GF(2)$. Let $n \leq i \leq 2n-2$. Then,

$$\begin{aligned}
\beta_n\beta_{i-n} &= \beta_i + \beta_{i-2}\cdot\ell \\
\beta_1\beta_{i-n} &= \beta_i + \beta_{i-2}\cdot\ell \\
\beta_i &= \beta_{i-n+1} + (\beta_{i-2} + \beta_{i-n-1})\cdot\ell \\
\beta_i &= \beta_{i-n+1} + (\beta_{i-n-2}\beta_n + \beta_{i-n-4}\cdot\ell + \beta_{i-n-1})\cdot\ell \\
\beta_i &= \beta_{i-n+1} + (\beta_{i-n-1} + \beta_{i-n-3}\cdot\ell + \beta_{i-n-4}\cdot\ell + \beta_{i-n-1})\cdot\ell \\
\beta_i &= \beta_{i-n+1} + (\beta_{i-n-3} + \beta_{i-n-4})\cdot\ell
\end{aligned}$$

If $i - n$ is odd, then $\ell = 1$. Otherwise, $\ell = 0$.

4.2.2 Irreducible Hermite Trinomials

Table 4.3 gives selected irreducible Hermite trinomials. We have mainly two kinds of irreducible Hermite trinomials of the form: $\beta_n + \beta_k + \beta_0$ and $\beta_n + \beta_k + \beta_1$. According to Table 4.3, it should be noted that recommended NIST or SEC binary fields such as $GF(2^{113})$, $GF(2^{233})$, $GF(2^{283})$ and $GF(2^{571})$ can be constructed with irreducible Hermite trinomials [32], [36].

Table 4.3: Irreducible Hermite Trinomials

$\beta_{137} + \beta_{17} + \beta_0$	$\beta_{113} + \beta_{12} + \beta_1$
$\beta_{169} + \beta_5 + \beta_0$	$\beta_{199} + \beta_{28} + \beta_1$
$\beta_{223} + \beta_{17} + \beta_0$	$\beta_{271} + \beta_{24} + \beta_1$
$\beta_{233} + \beta_5 + \beta_0$	$\beta_{209} + \beta_{26} + \beta_1$
$\beta_{271} + \beta_7 + \beta_0$	$\beta_{281} + \beta_6 + \beta_1$
$\beta_{311} + \beta_{25} + \beta_0$	$\beta_{283} + \beta_{66} + \beta_1$
$\beta_{383} + \beta_{21} + \beta_0$	$\beta_{361} + \beta_{16} + \beta_1$
$\beta_{431} + \beta_{65} + \beta_0$	$\beta_{457} + \beta_{24} + \beta_1$
$\beta_{497} + \beta_3 + \beta_0$	$\beta_{491} + \beta_{26} + \beta_1$
$\beta_{577} + \beta_7 + \beta_0$	$\beta_{571} + \beta_{22} + \beta_1$
$\beta_{641} + \beta_{11} + \beta_0$	$\beta_{653} + \beta_2 + \beta_1$

4.2.2.1 Reduction

By using irreducible Hermite trinomial, one can perform reduction operation as follows: Let $f = \beta_n + \beta_k + \beta_0$ be an irreducible polynomial of degree n over $GF(2)$. Let $n \le i \le 2n-2$. Note that $\beta_n = \beta_k + \beta_0$. Then,

$$\begin{aligned}
\beta_i &= \beta_{i-n+k} + \beta_{i-n} + \beta_{i-2} \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n} + (beta_{i-n-2}\beta_n + \beta_{i-n-4} \cdot \ell) \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n} + (\beta_{i-n+k-2} + \beta_{i-n+k-4} \cdot \ell + \beta_{i-n-2} + \beta_{i-n-4} \cdot \ell) \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n} + (\beta_{i-n+k-2} + \beta_{i-n+k-4} + \beta_{i-n-2} + \beta_{i-n-4}) \cdot \ell
\end{aligned}$$

If $i-n$ is odd, then $\ell = 1$. Otherwise, $\ell = 0$. Note that if $i-n$ is odd, $i-n-2$ is also odd.

Let $f = \beta_n + \beta_k + \beta_1$ be an irreducible polynomial of degree n over $GF(2)$. Let $n \le i \le 2n-2$. Note that $\beta_n = \beta_k + \beta_1$. Then,

$$\begin{aligned}
\beta_i &= \beta_{i-n+k} + \beta_{i-n+1} + (\beta_{i-n-1} + \beta_{i-2}) \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n+1} + (\beta_{i-n-1} + \beta_{i-n-2}\beta_n + \beta_{i-n-4} \cdot \ell) \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n+1} + (\beta_{i-n-1} + \beta_{i-n+k-2} + \beta_{i-n-1} + \beta_{i-n-2} \cdot \ell + \beta_{i-n-4} \cdot \ell) \cdot \ell \\
\beta_i &= \beta_{i-n+k} + \beta_{i-n+1} + (\beta_{i-n+k-2} + \beta_{i-n-2} + \beta_{i-n-4}) \cdot \ell
\end{aligned}$$

If $i-n$ is odd, then $\ell = 1$. Otherwise, $\ell = 0$.

4.2.3 Reduction Complexity

Table 4.4 shows reduction complexity for irreducible Hermite binomials and trinomials.

4.3 Toeplitz Matrix Vector Product for Hermite Basis with Subquadratic Space Complexity

In this section we recall Toeplitz matrix-vector multiplication scheme. Then, we give polynomial multiplication in Hermite basis with Toeplitz matrix-vector multiplication design. Remember that a Toeplitz matrix is defined as

Table 4.4: Reduction Complexity

	Form	$\#XOR$
Hermite Binomial	$\beta_n + \beta_0$	$2n$
Hermite Binomial	$\beta_n + \beta_1$	$2n$
Hermite Trinomial	$\beta_n + \beta_k + \beta_0$	$4n$
Hermite Trinomial	$\beta_n + \beta_k + \beta_1$	$\frac{7n}{2}$

Definition 4.3.1 *An $n \times n$ Toeplitz matrix is a matrix $(T_{i,j})$ with $T_{i,j} = T_{i-1,j-1}$ for $2 \leq i, j \leq n$.*

Let $n = 2^k$ where k is a positive integer. Let A be an $n \times n$ Toeplitz matrix. Let B be an $n \times 1$ column vector. Let $C = A \cdot B$ over $GF(2)$. Then,

$$\begin{bmatrix} A_1 & A_0 \\ A_2 & A_1 \end{bmatrix} \cdot \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = \begin{bmatrix} C_0 \\ C_1 \end{bmatrix}$$

where each A_0, A_1 and A_2 is Toeplitz matrix in size of $\frac{n}{2} \times \frac{n}{2}$. B_0, B_1, C_0 and C_1 are $\frac{n}{2} \times 1$ column vectors. Then, by [40]

$$\begin{aligned} C_0 &= P_0 + P_2 \\ C_1 &= P_1 + P_2 \end{aligned}$$

where

$$\begin{aligned} P_0 &= (A_0 + A_1)B_1 \\ P_1 &= (A_1 + A_2)B_0 \\ P_2 &= A_1(B_0 + B_1) \end{aligned}$$

For $n = 2$, the required number of multiplications and additions to compute C is 3 and 5 respectively. In [11], they give complexity results for computing C for binary fields as follows:

$$\begin{aligned} \#AND &= n^{log_2 3} \\ \#XOR &= 5.5n^{log_2 3} - 6n + 0.5 \end{aligned}$$

Similarly, for $n = 3^k$ where k is a positive integer, one can use the following formula to compute Toeplitz matrix-vector multiplication [40]

$$\begin{bmatrix} A_2 & A_1 & A_0 \\ A_3 & A_2 & A_1 \\ A_4 & A_3 & A_2 \end{bmatrix} \cdot \begin{bmatrix} B_0 \\ B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix}$$

where A_0, A_1, A_3 and A_4 are individually Toeplitz matrix of size $\frac{n}{3} \times \frac{n}{3}$, and $B_0, B_1, B_2 C_0, C_1$ and C_2 are $\frac{n}{3} \times 1$ column vectors. Then, by [40]

$$\begin{aligned} C_0 &= P_0 + P_3 + P_4 \\ C_1 &= P_1 + P_3 + P_5 \\ C_2 &= P_2 + P_4 + P_5 \end{aligned}$$

where

$$\begin{aligned} P_0 &= (A_0 + A_1 + A_2)B_2 \\ P_1 &= (A_0 + A_1 + A_3)B_1 \\ P_2 &= (A_2 + A_3 + A_4)B_0 \\ P_3 &= A_1(B_1 + B_2) \\ P_4 &= A_2(B_0 + B_2) \\ P_5 &= A_3(B_0 + B_1) \end{aligned}$$

For $n = 3$, the required number of multiplications and additions to compute C is 6 and 14 respectively. In [11], they give complexity results for computing C for binary fields as follows:

$$\begin{aligned} \#AND &= n^{log_3 6} \\ \#XOR &= \frac{24}{5} n^{log_3 6} - 5n + \frac{1}{5} \end{aligned}$$

Note that it is also possible to obtain similar complexity results for $n = 2^i 3^j$ by combining the above approaches in the recursive manner. Now, we show that the multiplication of two elements, i.e., A and B over $GF(2^n)$ in Hermite polynomial representation can be computed as a Toeplitz matrix-vector product.

Theorem 4.3.2 *Let $f = \beta_n + \beta_1$ be an irreducible polynomial over $GF(2^n)$. Let $A = a_{n-1}\beta_{n-1} + \cdots + a_0\beta_0$ and $B = b_{n-1}\beta_{n-1} + \cdots + b_0\beta_0$ be n-term polynomials over $GF(2)$ and $A \cdot B \pmod{f} = C = c_{n-1}\beta_{n-1} + \cdots + c_0\beta_0$. Then, the coefficients of the polynomial C are computed with*

$$\begin{bmatrix} a_0 & a_1 & a_{n-1} & a_{n-2} & \cdots & a_3 & a_2 \\ a_1 & a_0 + a_{n-1} & a_{n-2} & a_{n-1} + a_{n-3} & \cdots & a_2 & a_1 + a_3 \\ \vdots & & & & & & \vdots \\ a_{n-2} & a_{n-1} + a_{n-3} & a_{n-4} & a_{n-3} & \cdots & a_0 + a_{n-1} & a_1 + a_{n-2} \\ a_{n-1} & a_{n-2} & a_{n-3} & a_{n-4} + a_{n-1} & \cdots & a_1 & a_0 + a_{n-1} \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix}$$

Proof. We give the sketch of the proof. Let $A \cdot B = C' = c'_{2n-2}\beta_{2n-2} + \cdots + c'_0\beta_0$ and $A \cdot B \pmod f) = C = c_{n-1}\beta_{n-1} + \cdots + c_0\beta_0$. Let $\beta_n + \beta_1$ be an irreducible polynomial over $GF(2^n)$. Then, $\beta_n = \beta_1$. By Theorem 4.1.2,

$$\begin{aligned} \beta_{n+1} &= \beta_2 + \beta_0 \\ \beta_{n+2} &= \beta_3 + \beta_1 + \beta_n = \beta_3 \\ \beta_{n+3} &= \beta_4 + \beta_2 \\ \beta_{n+4} &= \beta_5 + \beta_3 + \beta_{n+2} = \beta_5 \\ &\vdots \\ \beta_{2n-3} &= \beta_{n-2} + \beta_{n-4} \\ \beta_{2n-2} &= \beta_{n-1} \end{aligned}$$

Then,

$$C' = (c'_{2n-2}+c'_{n-1})\beta_{n-1}+(c'_{2n-3}+c'_{n-2})\beta_{n-2}\cdots+(c'_{n+3}+c'_{n+1}+c'_2)\beta_2+(c'_n+c'_1)\beta_1+(c'_{n+1}+c'_0)\beta_0 = C$$

$$\begin{aligned} c_0 &= c'_{n+1} + c'_0 \\ c_1 &= c'_n + c'_1 \\ c_2 &= c'_{n+3} + c'_{n+1} + c'_0 \\ &\vdots \\ c_{n-2} &= c'_{2n-3} + c'_{n-2} \\ c_{n-2} &= c'_{2n-2} + c'_{n-1} \end{aligned}$$

After writing these variables in the matrix-vector product form, one obtains the desired form. ■

Now, we give an example.

Example 4.3.3 *Let $f = \beta_4 + \beta_1$ be an irreducible polynomial over $GF(2)$. Let $A = a_3\beta_3 + a_2\beta_2 + a_1\beta_1 + a_0\beta_0$ and $B = b_3\beta_3 + b_2\beta_2 + b_1\beta_1 + b_0\beta_0$ be 4-term polynomials over $GF(2)$. Let $A \cdot B = C' = c'_6\beta_6 + \cdots + c'_0\beta_0$ and $A \cdot B \pmod{f} = C = c_3\beta_3 + \cdots + c_0\beta_0$.*

$$\begin{aligned}\beta_4 &= \beta_1\\ \beta_5 &= \beta_2 + \beta_0\\ \beta_6 &= \beta_3 + \beta_1 + \beta_4 = \beta_3\end{aligned}$$

$$\begin{aligned}C' &= c'_6\beta_3 + c'_5(\beta_2 + \beta_0) + c'_4\beta_1 + c'_3\beta_3 + c'_2\beta_2 + c'_1\beta_1 + c'_0\beta_0\\ C' \pmod{f} &= (c'_6 + c'_3)\beta_3 + (c'_5 + c'_2)\beta_2 + (c'_4 + c'_1)\beta_1 + (c'_5 + c'_0)\beta_0 = C\end{aligned}$$

where

$$\begin{aligned}c'_0 &= a_0b_0 + a_1b_1\\ c'_1 &= a_0b_1 + a_1b_0\\ c'_2 &= a_0b_2 + a_2b_0 + a_1b_1 + a_1b_3 + a_3b_1\\ c'_3 &= a_0b_3 + a_3b_0 + a_1b_2 + a_2b_1\\ c'_4 &= a_1b_3 + a_3b_1 + a_2b_2 + a_3b_3\\ c'_5 &= a_2b_3 + a_3b_2\\ c'_6 &= a_3b_3\end{aligned}$$

and

$$\begin{aligned}c_0 &= c'_5 + c'_0\\ c_0 &= a_0b_0 + a_1b_1 + a_3b_2 + a_2b_3\\ c_1 &= c'_4 + c'_1\\ c_1 &= a_1b_0 + a_0b_1 + a_3b_1 + a_2b_2 + a_1b_3 + a_3b_3\\ c_2 &= c'_5 + c'_2\\ c_2 &= a_2b_0 + a_1b_1 + a_3b_1 + a_0b_2 + a_3b_2 + a_1b_3 + a_2b_3\\ c_3 &= c'_6 + c'_3\\ c_3 &= a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3 + a_3b_3\end{aligned}$$

Then, by using above formula one can obtain the following Toeplitz matrix.

$$\begin{bmatrix} A_0 & A_1\\ A_1 & A_2\end{bmatrix} \cdot \begin{bmatrix} B_0\\ B_1\end{bmatrix} = \begin{bmatrix} C_0\\ C_1\end{bmatrix}$$

where

$$A_0 = \begin{bmatrix} a_0 & a_1 \\ a_1 & a_0 + a_3 \end{bmatrix}, \ A_1 = \begin{bmatrix} a_3 & a_2 \\ a_2 & a_1 + a_3 \end{bmatrix}, \ A_2 = \begin{bmatrix} a_1 & a_0 + a_3 \\ a_0 + a_3 & a_1 + a_2 \end{bmatrix}$$

$$B_0 = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}, \ B_1 = \begin{bmatrix} b_2 \\ b_3 \end{bmatrix}, \ C_0 = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}, \ C_1 = \begin{bmatrix} c_3 \\ c_2 \end{bmatrix}$$

Then, by changing of rows we obtain,

$$\begin{bmatrix} A_1 & A_2 \\ A_0 & A_1 \end{bmatrix} \cdot \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = \begin{bmatrix} C_1 \\ C_0 \end{bmatrix}$$

Then, we compute

$$\begin{aligned} C_0 &= (A_1 + A_2)B_1 + A_1(B_0 + B_1) \\ C_1 &= (A_0 + A_1)B_0 + A_1(B_0 + B_1) \end{aligned}$$

4.4 Multiplication Complexity

In this section, we give modular multiplication complexity of multipliers in view of $\#AND$ and $\#XOR$ gates. Table 4.5 compares the complexity of selected multipliers. T_X and T_A represent respectively the delay of an XOR gate and an AND gate. Note that this table is prepared by using Karatsuba multiplication method for Hermite basis [39]. According to Table 4.5, Hermite polynomial representation has better complexity than Charlier polynomial representation, Dickson polynomial representation, ONB II and in some cases ONB I. Therefore, binary fields can be constructed with low weight Hermite polynomials efficiently when there does not exist ONB for the corresponding extension.

Remark 4.4.1 *NIST recommended binary field $GF(2^{571})$ can be constructed efficiently by using Hermite polynomials since there is no ONB for the corresponding extension.*

Table 4.5: Complexity Comparison of Selected Multipliers

	p	$\#AND$	$\#XOR$	Critical Delay
Hermite Binomial	2	$n^{log_2 3}$	$\frac{11}{2}n^{log_2 3} - 6n + \frac{1}{2}$	$2log_2(n)T_X + T_A$
Hermite Binomial	3	$n^{log_3 6}$	$\frac{24}{5}n^{log_3 6} - 5n + \frac{1}{5}$	$3log_3(n)T_X + T_A$
Hermite Trinomial	2	$n^{log_2 3} + \frac{n^{log_2 3}}{9}$	$8n^{log_2 3} - 7n + 3$	$(2log_2(n) + 1)T_X + T_A$
Hermite Trinomial	3	$n^{log_3 6} + \frac{(\frac{n}{2})^{log_3 6}}{3}$	$\frac{116}{15}n^{log_3 6} - \frac{9}{5}n + \frac{7}{5}$	$(3log_3(n) + 1)T_X + T_A$
Polynomial Basis [39]	2	$n^{log_2 3}$	$6n^{log_2 3} - 8n + 2$	$(3log_2(n) - 1)T_X + T_A$
Polynomial Basis [39]	3	$n^{log_3 6}$	$\frac{29}{5}n^{log_3 6} - 8n + \frac{11}{5}$	$(4log_3(n) - 1)T_X + T_A$
Charlier Binomial [2]	2	$n^{log_2 3} + \lfloor\frac{n}{2}\rfloor^{log_2 3}$	$8n^{log_2 3} - 10n + 3$	$3log_2(n)T_X + T_A$
Charlier Binomial [2]	3	$n^{log_3 6} + \lfloor\frac{n}{2}\rfloor^{log_3 6}$	$\frac{116}{15}n^{log_3 6} - \frac{29}{5}n + \frac{7}{5}$	$4log_3(n)T_X + T_A$
Charlier Trinomial [2]	2	$n^{log_2 3} + \lfloor\frac{n}{2}\rfloor^{log_2 3}$	$8n^{log_2 3} - 8n + 3$	$3log_2(n)T_X + T_A$
Charlier Trinomial [2]	3	$n^{log_3 6} + \lfloor\frac{n}{2}\rfloor^{log_3 6}$	$\frac{116}{15}n^{log_3 6} - \frac{14}{5}n + \frac{7}{5}$	$4log_3(n)T_X + T_A$
Dickson Binomial [20]	2	$2n^{log_2 3}$	$11n^{log_2 3} - 11n$	$(2log_2(n) + 1)T_X + T_A$
Dickson Binomial [20]	3	$2n^{log_3 6}$	$\frac{48}{5}n^{log_3 6} - 11n + \frac{3}{5}$	$(3log_3(n) + 1)T_X + T_A$
Dickson Trinomial [20]	2	$2n^{log_2 3}$	$11n^{log_2 3} - 4n + 1$	$(2log_2(n) + 6)T_X + T_A$
Dickson Trinomial [20]	3	$2n^{log_3 6}$	$\frac{48}{5}n^{log_3 6} - 2n + \frac{1}{5}$	$(3log_3(n) + 6)T_X + T_A$
ONB I [11]	2	$n^{log_2 3} + n$	$\frac{11}{2}n^{log_2 3} - 4n - \frac{1}{2}$	$(2log_2(n) + 1)T_X + T_A$
ONB I [11]	3	$n^{log_3 6} + n$	$\frac{24}{5}n^{log_3 6} - 3n - \frac{4}{5}$	$(3log_3(n) + 1)T_X + T_A$
ONB II [11]	2	$2n^{log_2 3}$	$11n^{log_2 3} - 12n + 1$	$(2log_2(n) + 1)T_X + T_A$
ONB II [11]	3	$2n^{log_3 6}$	$\frac{48}{5}n^{log_3 6} - 10n - \frac{2}{5}$	$(3log_3(n) + 1)T_X + T_A$

The normalized number of required AND gates and XOR gates to multiply two elements in $GF(2^n)$ for $p = 2$ case for the selected multipliers given in Table 4.5 are depicted in Figure 4.1. Dashed lines correspond to the normalized number of required XOR gates. The normalized number of AND gates and XOR gated are obtained by simply dividing the number of required AND gates and XOR gates by the number of required AND gates and XOR gates in polynomial basis, respectively. It should be noticed that the number of required AND gates in Hermite polynomial representation with a binomial is as good as polynomial basis. Furthermore, the number of required XOR gates in Hermite polynomial representation with a binomial is the least one among all of those multipliers.

Figure 4.2 demonstrates the normalized number of required AND gates and XOR gates to multiply two elements in $GF(2^n)$ for $p = 3$ for the selected multipliers given in Table 4.5. Dashed lines correspond to the normalized number of required XOR gates. Note that the normalized number of AND gates and XOR gated are obtained by simply dividing the number of required AND gates and XOR gates by the number of required AND gates and XOR gates

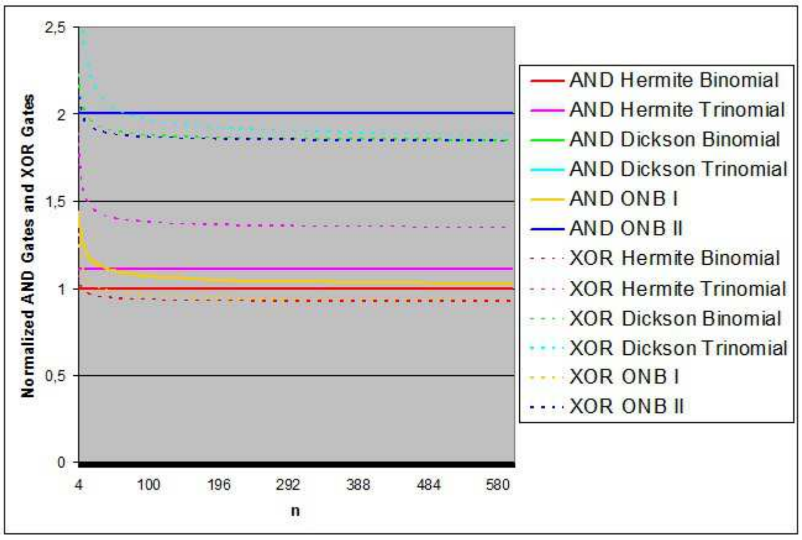

Figure 4.1: The Normalized Number of Required AND Gates and XOR Gates for $p = 2$

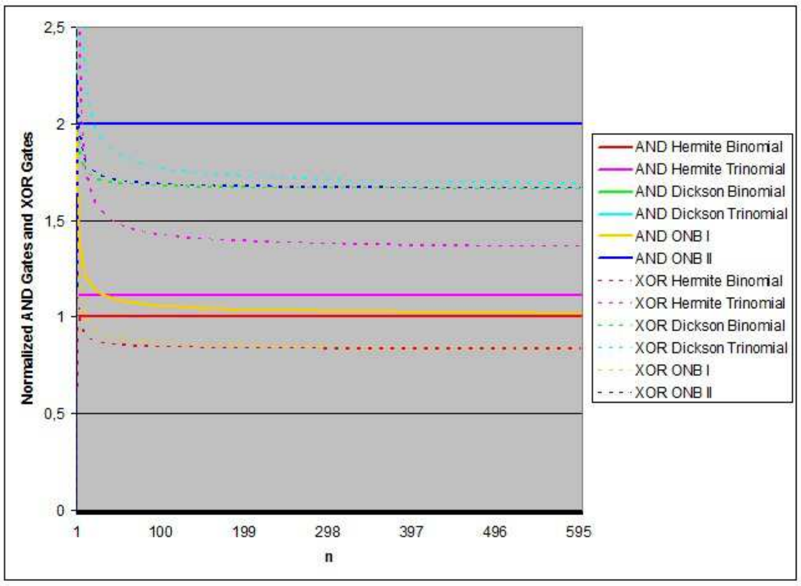

Figure 4.2: The Normalized Number of Required AND Gates and XOR Gates for $p = 3$

in polynomial basis, respectively. It should be noticed that the number of required *AND* gates in Hermite polynomial representation with a binomial is as good as polynomial basis. Furthermore, the number of required *XOR* gates in Hermite polynomial representation with a binomial is the least one among all of those multipliers.

Now, we give the complexity of the multiplication of two basis elements. This idea is restated in [38]. Let $B = \{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ be a basis for $GF(2^n)$. Then, multiplication of two basis elements, i.e. multiplication law, can be viewed as

$$\beta_i\beta_j = \sum_{k=0}^{n-1} \lambda_{ij}^{(k)}\beta_k$$

where $\lambda_{ij}^{(k)} \in GF(2)$. The complexity of the multiplication law relative to the basis is computed by the number of $\lambda_{ij}^{(k)} = 1$.

$$C(B) = \frac{1}{n}\sum_{k=0}^{n-1}\sum_{i,j=0}^{n-1} \lambda_{ij}^{(k)}$$

Note that $C(B) \geq n$ since we are working in $GF(2^n)$. This complexity notion can be considered as basis density [33]. We give some examples to compare different representations of binary fields.

Example 4.4.2 *Let $R = GF(2)[x]/(f(x))$, where $f(x)$ is an irreducible polynomial of degree n. Assume that there is an element $\beta \in R$ is a primitive element of degree n over $GF(2)$ such that the set $B = \{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{n-1}}\}$ is a basis for R. A basis of this form is called a normal basis for R. The complexity of normal basis satisfies te following inequality*

$$C(B) \geq 2n - 1$$

A basis satisfying $C(B) = 2n - 1$ is called optimal normal basis.

Example 4.4.3 *Let $R = GF(2)[x]/(x^{n'} - 1)$ be the finite quotient ring which is isomorphic to $GF(2^n)$ as a ring with the basis $B\{1, x, x^2, \cdots, x^{n'-1}\}$. Note that $n' = n + r$, where r is the redundancy. Then*

$$x^i \cdot x^j = \begin{cases} x^{i+j} & \text{if } i + j < n' \\ x^{i+j-n'} & \text{if } i + j \geq n' \end{cases}$$

$$\lambda_{ij}^{(k)} = \begin{cases} 1 & \text{if } k = i + j \text{ or } k = i + j - n' \\ 0 & \text{otherwise} \end{cases}$$

The complexity of B is

$$C(B) = \frac{1}{n'}\left(\frac{n'(n'+1)}{2} + \frac{(n'-1)n'}{2}\right) = n'$$

since there is exactly one k with $\lambda_{ij}^{(k)} \neq 0$ *for each* (i, j)*.*

Example 4.4.4 *Let* $R = GF(2)[x]/(x^{n'} - x - 1)$ *be the finite quotient ring which is isomorphic to* $GF(2^n)$ *as a ring with the basis* $B\{1, x, x^2, \cdots, x^{n'-1}\}$*. Note that* $n' = n + r$*, where r is the redundancy. Then*

$$x^i \cdot x^j = \begin{cases} x^{i+j} & \text{if } i + j < n' \\ x^{i+j-n'+1} + x^{i+j-n'} & \text{if } i + j \geq n' \end{cases}$$

The complexity of B is

$$C(B) = \frac{1}{n'}\left(\frac{n'(n'+1)}{2} + \frac{(n'-1)n'}{2} \cdot 2\right) = \frac{3n'-1}{2}$$

.

Remark 4.4.5 *By changing* n' *to n, one obtains the complexity result for polynomial representation with an irreducible trinomial polynomial.*

Example 4.4.6 *Let* $R = GF(2)[x]/(x^{n'} - x^2 - x - 1)$ *be the finite quotient ring which is isomorphic to* $GF(2^n)$ *as a ring with the basis* $B\{1, x, x^2, \cdots, x^{n'-1}\}$*. Note that* $n' = n + r$*, where r is the redundancy. Then*

$$x^i \cdot x^j = \begin{cases} x^{i+j} & \text{if } i + j < n' \\ x^{i+j-n'+2} + x^{i+j-n'+1} + x^{i+j-n'} & \text{if } n' \leq i + j \geq 2n' - 3 \\ x^{i+j-n'+1} + x^{i+j-n'} + x^2 + x + 1 & \text{if } i + j = 2n' - 2 \end{cases}$$

The complexity of B is

$$C(B) = \frac{1}{n'}\left(\frac{n'(n'+1)}{2} + \frac{(n'-1)n' + 2n' - 2}{2}\cdot 3 + 5\right) = \frac{2n'^2 + 2n' + 2}{n'}$$

.

Example 4.4.7 *Let* $B = \{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ *be the Charlier basis. Let* $f = \beta_n + \beta_0$ *be an irreducible polynomial of degree n over* $GF(2)$ *and* $R = GF(2)[x]/(f)$. *Then*

$$\beta_i \cdot \beta_j = \begin{cases} \beta_{i+j} + \beta_{i+j-1}\cdot \ell & \text{if } i+j < n \\ \beta_{i+j-n} + \beta_{i+j-n-1}\cdot \ell & \text{if } i+j \geq n \end{cases}$$

If i and j are both odd, then $\ell = 1$. *Otherwise* $\ell = 0$.

The complexity of B is

$$C(B) = \frac{1}{n}\left(\frac{n(n+1)}{2} + \frac{n(n+1)}{2\cdot 2} + \frac{(n-1)n}{2} + \frac{(n-1)n}{2\cdot 2}\right) = \frac{3n}{2}$$

Example 4.4.8 *Let* $B = \{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ *be the Charlier basis. Let* $f = \beta_n + \beta_k + \beta_0$ *be an irreducible polynomial of degree n over* $GF(2)$ *and* $R = GF(2)[x]/(f)$. *Then*

$$\beta_i \cdot \beta_j = \begin{cases} \beta_{i+j} + \beta_{i+j-1}\cdot \ell & \text{if } i+j < n \\ \beta_{i+j-n+k} + \beta_{i+j-n} + (\beta_{i+j-n+k-1} + \beta_{i+j-n-1})\cdot \ell & \text{if } i+j \geq n \end{cases}$$

If i and j are both odd, then $\ell = 1$. *Otherwise* $\ell = 0$.

The complexity of B is

$$C(B) = \frac{1}{n}\left(\frac{n(n+1)}{2} + \frac{n(n+1)}{2\cdot 2} + \frac{(n-1)n}{2}\cdot 2 + \frac{(n-1)n}{2\cdot 2}\cdot 2\right) = \frac{9n-3}{4}$$

Example 4.4.9 *Let* $B = \{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ *be the Hermite basis. Let* $f = \beta_n + \beta_0$ *be an irreducible polynomial of degree n over* $GF(2)$ *and* $R = GF(2)[x]/(f)$. *Then*

$$\beta_i \cdot \beta_j = \begin{cases} \beta_{i+j} + \beta_{i+j-2}\cdot \ell & \text{if } i+j < n \\ \beta_{i+j-n} + (\beta_{i+j-n-2} + \beta_{i+j-n-4})\cdot \ell & \text{if } i+j \geq n \end{cases}$$

If i and j are both odd, then $\ell = 1$. *Otherwise* $\ell = 0$.

The complexity of B is

$$C(B) = \frac{1}{n}\left(\frac{n(n+1)}{2} + \frac{n(n+1)}{2\cdot 2} + \frac{(n-1)n}{2} + \frac{(n-1)n}{2\cdot 2}\cdot 2\right) = \frac{7n-1}{4}$$

Example 4.4.10 *Let $B = \{\beta_0, \beta_1, \cdots, \beta_{n-1}\}$ be the Hermite basis. Let $f = \beta_n + \beta_k + \beta_0$ be an irreducible polynomial of degree n over $GF(2)$ and $R = GF(2)[x]/(f)$. Then*

$$\beta_i \cdot \beta_j = \begin{cases} \beta_{i+j} + \beta_{i+j-2} \cdot \ell & \text{if } i + j < n \\ \beta_{i+j-n+k} + \beta_{i+j-n} + (\beta_{i+j-n+k-2} + \beta_{i+j-n+k-4} + \beta_{i+j-n-2} + \beta_{i+j-n-4}) \cdot \ell & \text{if } i + j \geq n \end{cases}$$

If i and j are both odd, then $\ell = 1$. Otherwise $\ell = 0$.
The complexity of B is

$$C(B) = \frac{1}{n}\left(\frac{n(n+1)}{2} + \frac{n(n+1)}{2 \cdot 2} + \frac{(n-1)n}{2} \cdot 2 + \frac{(n-1)n}{2 \cdot 2} \cdot 4\right) = \frac{11n-5}{4}$$

Example 4.4.11 *Let $B = \{\beta_0 = 1, \beta_1, \cdots, \beta_{n-1}\}$ be the Dickson basis given in [20]. Let $f = \beta_n + \beta_k + \beta_0$ be an irreducible polynomial of degree n over $GF(2)$ and $R = GF(2)[x]/(f)$. Then*

$$\beta_i \cdot \beta_j = \begin{cases} \beta_{i+j} + \beta_{|i-j|} & \text{if } i + j < n \\ \beta_{i+j-n+k} + \beta_{|i+j-n-k|} + \beta_{i+j-n} + \beta_{2n-i-j} & \text{if } i + j \geq n \end{cases}$$

The complexity of B is

$$C(B) = \frac{1}{n}\left(\frac{n(n+1)}{2} + \frac{(n-1)n}{2} \cdot 4\right) = 3n - 1$$

According to examples, in some cases especially when there is no ONB exists, Charlier representation and Hermite representation has better multiplication complexity. Moreover, if we have a Charlier binomial or an Hermite binomial for the corresponding extension, the multiplication complexity is the lowest one even if there exists an ONB for this extension. Note that in Example 4.4.3, 4.4.4 and 4.4.6 if the redundancy, r, is very small for example $r = 1$ as explained in Chapter 2, then the (modified) redundant representation is one of the best choice for corresponding extension for efficient implementations in hardware.

CHAPTER 5

CONCLUSION

In Chapter 2, we give a modified redundant representation which can be considered as a generalization of [10], [14] and [15]. Using our modified redundant representation, we improve many of the complexity values in [10], [14] and [15] significantly. Our method works for any finite field. We give more emphasis for finite fields of characteristic 2. We also give some applications in cryptography.

In Chapter 3, we give a new way to represent certain finite fields $GF(2^n)$. This representation is based on Charlier polynomials. We show that multiplication in Charlier polynomial representation can be performed with subquadratic space complexity. One can obtain binomial or trinomial irreducible polynomials in Charlier polynomial representation which allows us faster modular reduction over binary fields when there is no desirable such low weight irreducible polynomial in other representations. This representation is very interesting for NIST recommended binary field $GF(2^{283})$ since there is no ONB for the corresponding extension. We also note that recommended NIST and SEC binary fields can be constructed with low weight Charlier polynomials such as $GF(2^{113})$, $GF(2^{131})$ and $GF(2^{233})$.

In Chapter 4, we propose a new representation of finite fields of characteristic two by using Hermite polynomials. After recalling well-known finite field multiplication technique, we show that multiplication in Hermite polynomial representation can be achieved with subquadratic space complexity. Then, we discuss how to implement them efficiently. This representation enables us to find binomial, trinomial or quadranomial irreducible polynomials which allows us faster modular reduction over binary fields when there is no desirable such low weight irreducible polynomial in other representations. We then show that the product of two elements in Hermite polynomial representation can be performed as Toeplitz matrix-vector product. It is shown that in some cases the proposed representation has better space complexity even if there exists an ONB for the corresponding extension. Therefore, this representation is very interesting for NIST and SEC recommended binary fields since these can be constructed with low weight Hermite polynomials. This representation is also quite generic in the sense that it is independent on the choice of extension degree. We also note that this work naturally extends to other characteristics, especially characteristic 3.

REFERENCES

[1] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Table*, Dover Publications, 1965.

[2] S. Akleylek, M. Cenk and F. Özbudak, “Polynomial Multiplication Over Binary Fields Using Charlier Polynomial Representation with Low Space Complexity”, 11th International Conference on Cryptology (INDOCRYPT 2010), G. Gong and K.C. Gupta (Eds.), LNCS 6498, pp.227-237, 2010.

[3] S. Akleylek, M. Cenk and F. Özbudak, “A New Representation of Elements of Binary Fields with Subquadratic Space Complexity Multiplication of Polynomials”, submitted, 2010.

[4] S. Akleylek and F. Özbudak “Modified Redundant Representation for Designing Arithmetic Circuits with Small Complexity”, IEEE Transactions on Computers, 2011 (In press).

[5] A. Bohm, *Quantum Mechanics: Foundations and Applications*, third edition, Springer, 2001.

[6] W. Bosma, J. Cannon and C. Playoust, “The Magma Algebra System I: The User Language”, Journal Symbolic Computation, vol. 24, no. 3-4, pp. 235-265, Sept. 1997.

[7] M. Ciet, J. Quisquater and F. Sica, “A Secure Family of Composite Finite Fields for Fast Implementation of Elliptic Curve Cryptography”, Progress in Cryptology - INDOCRYPT 2001, LNCS vol. 2247, pp. 108-116, Springer-Verlag, Dec 2001.

[8] S.A. Cook, “On the Minimum Computation Time of Functions”, pp. 51-77, Thesis, Harvard University, Cambridge, MA, 1966.

[9] D. Drake, “The Combinatorics of Associated Hermite Polynomials”, European Journal of Combinatorics, vol.30 no.4, pp.1005-1021, 2009.

[10] G. Drolet, ”A New Representation of Elements of Finite Fields $GF(2^m)$ Yielding Small Complexity Arithmetic Circuits”, IEEE Trans. Computers, vol. 47, no. 9, pp. 938-946, Sept. 1998.

[11] H. Fan and M.A. Hasan, “A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields”, IEEE Trans. Computers, Vol.56-2, pp.224-233, 2007.

[12] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, second ed. Cambridge University Press, Cambridge, UK, 2003.

[13] P. Gaudry, F. Hess and N. Smart “Constructive and Destructive Facets of Weil Descent on Elliptic Curves”, Journal of Cryptology, vol. 15, no. 1, Springer-Verlag, March 2002

[14] W. Geiselmann, J. Muller-Quade and R. Steinwandt, “On ’A New Representation of Elements of Finite Fields $GF(2^m)$ Yielding Small Complexity Arithmetic Circuits’”, IEEE Trans. Computers, vol. 51, no. 12, pp. 1460-1461, Dec. 2002.

[15] W. Geiselmann and R. Steinwandt, “A Redundant Representation of $GF(q^n)$ for Designing Arithmetic Circuits”, IEEE Trans. Computers, vol. 52, no. 7, pp. 848-853, July 2003.

[16] C.D. Godsil, *Algebraic Combinatorics*, Chapman Hall/CRC Mathematics Series, 1993.

[17] D. Hankerson, D.G. Hoffman, D.A. Leonard, C.C. Lindner, K.T. Phelps, C.A. Rodger and J.R. Wall, *Coding Theory and Cryptography: The Essentials*, second ed. Marcel Dekker, 2000.

[18] D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2003.

[19] K. Harrison, D. Page and N.P. Smart, “Software Implementation of Finite Fields of Characteristic Three, For Use in Pairing-Based Cryptosystems”, LMS Journal Computation and Mathematics, vol. 5, pp. 181-193, 2002.

[20] M.A. Hasan and C. Negre, “Subquadratic Space Complexity Multiplication over Binary Fields with Dickson Polynomial Representation”, WAIFI 2008, LNCS 5130, pp.88-102, 2008.

[21] M.A. Hasan and C. Negre, “Low Space Complexity Multiplication over Binary Fields with Dickson Polynomial Representation”, to appear IEEE Trans. on Computers, doi: 10.1109/TC.2010.132, 2010.

[22] IEEE 1363 Standard Specifications for Public-Key Cryptography, 2000. http://grouper.ieee.org/groups/1363/ index.html

[23] B. Itoh and S. Tsujii, “Structure of a Parallel Multiplier for a Class of Fields $GF(2^m)$”, Information and Computers, vol. 83, pp. 21-40, 1989.

[24] A. Karatsuba and Y. Ofman, “Multiplication of Multidigit Numbers by Automata”, Soviet Physics-Doklady, vol. 7, pp. 595-596, 1963.

[25] R. Katti and J. Brennan, “Low Complexity Multiplication in a Finite Field Using Ring Representation”, IEEE Trans. Computers vol. 52, no. 4, pp. 418-427, April 2003.

[26] R.Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications*, Cambridge University, 1997.

[27] R.J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Boston: Kluwer Academic, 1987.

[28] A.J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Boston, MA: Kluwer Academic Publishers, 1993.

[29] A.J. Menezes, I.F. Blake, X. Gao, R.C. Mullen, S.A. Vanstone and T. Yaghobian, *Applications of Finite Fields*, Boston: Kluwer Academic, 1993.

[30] P.L. Montgomery, “Five, Six and Seven-term Karatsuba-like Formulae”, IEEE Trans. Computers, vol. 54, no. 3, pp. 362-369, March 2005.

[31] G. Mullen and C. Mummert, *Finite Fields and Applications*, American Mathematical Society, 2007.

[32] National Institute of Standards and Technology, Recommended Elliptic curves for Federal Government Use, 1999.

[33] C. Nègre, “Finite field arithmetic using quasi-normal bases”, Finite Fields and Their Applications vol.13, no.3 pp.635-647, 2007.

[34] S. Oh, C.H. Kim, J. Lim and D.H. Cheon, “Efficient Normal Basis Multipliers in Composite Fields”, IEEE Transactions on Computers, vol.49, no.10, pp.1133-1138, Oct. 2000.

[35] Sage: Open Source Mathematics Software, The Sage Group, http://www.sagemath.org

[36] Standards for Efficient Cryptography Group (SECG), SEC 2: Recommended Elliptic Curve Domain Parameters, 2010.

[37] J.H. Silverman, “Fast Multiplication in Finite Fields $GF(2^N)$”, Proc. Cryptographic Hardware and Embedded Systems, First Int’l Workshop, C.K. Koc and C. Paar (Eds.), pp. 122-134, 1999.

[38] J.H. Silverman, ”Rings of Low Multiplicative Complexity”, Finite Fields and Their Applications, vol.6, pp.175-191, 2000.

[39] A. Weimerskirch and C. Paar, *Generalizations of the Karatsuba Algorithm for Efficient Implementations*, http://eprint.iacr.org/2006/224, 2006.

[40] S. Winograd, *Arithmetic Complexity of Computations*, SIAM, 1980.

[41] H. Wu, M.A. Hasan and I.F. Blake, “Highly Regular Architectures for Finite Field Computation Using Redundant Basis”, Proc. Cryptographic Hardware and Embedded Systems, First Int’l Workshop, Ç.K. Koç and C. Paar (Eds.), pp. 269-279, 1999.

[42] H. Wu, M.A. Hasan, I.F. Blake and S. Gao, “Finite Field Multiplier Using Redundant Representation”, IEEE Trans. Computers vol. 51, no. 11, pp. 1306-1316, Nov. 2002.

VITA

Sedat Akleylek received the B.Sc. degree in Mathematics with the part of Computer Science from Ege University in 2004 in İzmir, Turkey and M.Sc. degree in Cryptography from Middle East Technical University in 2008 in Ankara, Turkey. He is currently employed as a research assistant at the Graduate School of Applied Mathematics of Middle East Technical University since 2005. His current research interests are in the areas of cryptography, algorithms and architectures for computations in Galois fields, computer algebra and e-learning.

Publications

- S. Akleylek and F. Özbudak, "Modified Redundant Representation for Designing Arithmetic Circuits with Small Complexity", IEEE Transactions on Computers, 2011 (In press).
- S. Akleylek, M. Cenk and F. Özbudak, "On the Polynomial Multiplication in Chebyshev Form", submitted, 2010.
- S. Akleylek, M. Cenk and F. Özbudak, "A New Representation of Elements of Binary Fields with Subquadratic Space Complexity Multiplication of Polynomials", submitted, 2010.
- S. Akleylek, M. Cenk and F. Özbudak, "Faster Interleaved Montgomery Modular Multiplication without Pre-computational Phase", submitted, 2010.
- S. Akleylek, M. Cenk and F. Özbudak, "Polynomial Multiplication over Binary Fields Using Charlier Polynomial Representation with Low Space Complexity", 11th International Conference on Cryptology (INDOCRYPT 2010), G. Gong and K.C. Gupta (Eds.), LNCS 6498, pp.227-237, Springer, 2010.
- S. Akleylek, B. B. Kırlar, Ö. Sever, Z. Yüce, "A New Short Signature Scheme with Random Oracle from Bilinear Pairings", Journal of Telecommunications and Information Technology, 2010 (In press)

- S. Akleylek, M. Cenk and F. Özbudak, “Faster Montgomery Modular Multiplication without Pre-computational Phase For Some Classes of Finite Fields”, Computer and Information Sciences, E. Gelenbe et.al (Eds.), LNEE vol.62, pp.405-408, Springer, 2010.

- S. Akleylek, M. Cenk and F. Özbudak, “A New Representation of Elements of Binary Fields with Subquadratic Space Complexity Multiplication”, 10th Central European Conference on Cryptology, Poznan, Poland, 2010.

- S. Akleylek and M. Cenk, “Speeding Up Montgomery Modular Multiplication For Prime Fields”, XII Antalya Algebra Days, 2010, Turkey.

- S. Akleylek, M. Cenk and F. Özbudak, “Modified Discrete Fourier Transform for Efficient Polynomial Multiplication”, Proceedings of 4th International Information Security and Cryptology Confence (ISCTURKEY 2010), pp. 150-153, 2010, Ankara, Turkey.

- S. Akleylek, B. B. Kırlar, Ö. Sever and Z. Yüce, “Short Signature Scheme from Bilinear Pairings”, NATO Information Assurance and Cyber Defense (IST-091), Antalya, Turkey, 2010.

- S. Akleylek, B. B. Kırlar, Ö. Sever and Z. Yüce, “Short Signature Scheme from Bilinear Pairings”, Proceedings of Western European Workshop on Research in Cryptology (WEWoRC 2009), pp. Graz, Austria.

- S. Akleylek, B. B. Kırlar, Ö. Sever and Z. Yüce, “Arithmetic on Pairing-Friendly Fields”, Proceedings of 3th International Information Security and Cryptography Conference (ISCTURKEY 2008), pp. 115-120, Ankara, Turkey

- S. Akleylek, B. B. Kırlar, Ö. Sever and Z. Yüce, “Pairing-Based Cryptography: A Survey”, Proceedings of 3th International Information Security and Cryptography Conference, (ISCTURKEY 2008), pp. 121-125, Ankara, Turkey.