UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Mathematics
Mathematics

Annabell Kuldmaa

# EFFICIENT MULTIPLICATION IN BINARY FIELDS

Bachelor Thesis (9 ECTS)

Supervisor: Lauri Tart

Author: ....................................... ” ......” June 2015

Supervisor: .................................. ” ......” June 2015

Head of Institute: ............................ ” ......” June 2015

TARTU 2015

# Efficient Multiplication in Binary Fields

**Abstract:** The thesis discusses the basics of efficient multiplication in finite fields, especially in binary fields. There are two broad approaches: polynomial representation and normal bases, used in software and hardware implementations, respectively. Due to the advantages of normal bases of low complexity, there is also a brief introduction to constructing optimal normal bases. Furthermore, as irreducible polynomials are of fundamental importance for finite fields, the thesis concludes with some irreducibility test.
**Keywords:** finite field multiplication, modular reduction, polynomial basis, normal basis, irreducible polynomials

# Tõhus korrutamine binaarsetes korpustes

**Lühikokkuvõte:** Käesolevas bakalaureusetöös käsitletakse tõhusa korrutamise põhitõdesid lõplikes korpustes, keskendudes peamiselt binaarsetele korpustele. Levinumad on kaks lähenemist: ühed toetuvad polünoombaasidele ning teised normaalbaasidele, mida kasutatakse vastavalt tarkvaras ja riistvaras. Madala keerukusega normaalbaaside eeliste tõttu vaatleme ka optimaalsete normaalbaaside konstrueerimist. Lisaks uurime töö lõpuosas taandumatute polünoomide leidumise tingimusi. Seda põhjusel, et taandumatud polünoomid on lõplike korpuste konstrueerimisel fundamentaalse tähtusega.
**Võtmesõnad:** korrutamine lõplikes korpustes, modulaarne taandamine, polünoombaasid, normaalbaasid, taandumatud polünoomid

# Contents

# Introduction

Finite fields are widely applied in coding theory and cryptography. Some prominent examples are Reed-Solomon codes, the Advanced Encryption Standard and elliptic curve cryptosystems. All of these applications require fast hardware and software implementations of arithmetic, especially multiplication, in large finite fields.

The aim of this thesis is to give an overview of selected methods for performing fast multiplication in finite fields. Over the last few decades this area has been intensely studied and numerous implementations have been introduced. The thesis reviews the basics of these schemes and provides references for more detailed resources.

The main focus of the thesis is on binary fields as these fields are widely used and particularly suitable for hardware designs, since the arithmetic involves basic bitwise operations. In general, it is common practice to use normal bases for hardware implementations of multiplication and polynomial bases for software ones.

While the hardware-oriented methods are compact and fast, they are also inflexible and expensive as the change of the field in hardware requires a complete redesign. On the other hand, despite software implementations being slower, they are more cost-effective and much more flexible. For example, the algorithms and field parameters can easily be modified without requiring redesign. Thus, the reader will be introduced to both software and hardware implementations.

The first chapter covers the fundamental definitions and properties of finite fields. The theory is discussed only to the extent needed for our purposes. The second chapter introduces several algorithms for multiplication via polynomial bases. This also includes a look into how Montgomery modular multiplication can be modified for binary fields. This is followed by two hardware implementations using normal bases, the Massey-Omura scheme and an algorithm due to Reyhani-Masoleh and Hasan. The existence of normal bases is discussed in Chapter 4 as normal bases of low complexity are desirable in hardware designs. Finally, since irreducible polynomials are of fundamental importance for finite fields, the thesis concludes with an irreducibility test due to Rabin. As the field reduction process can be accelerated using trinomials, the existence of irreducible trinomials is also briefly discussed.

The thesis draws on a number of sources. The basics of finite fields are taken from [7]. Chapter 2 relies heavily on [3], [4] and [5] and Chapter 3 on [6], [8] and [11]. Chapter 4 is based on [8] and Chapter 5 on [2] and [10].

# 1 Preliminaries

In this chapter we introduce a number of fundamental concepts that are frequently used in the following chapters. In general, we do not prove the results we cite, instead we refer the reader to [7]. Also, we assume that the reader is familiar with the concepts of modular arithmetic and primitive roots. If necessary, the reader can consult [1] or [9].

## 1.1 Rings and Fields

Recall that a ring $R$ is a *division ring* if the nonzero elements of $R$ form a group under multiplication and a *field* is a commutative division ring.

**Definition 1.1.** *A* finite field *is a field* $\mathbb{F}$ *which contains a finite number of elements. The number of distinct elements in* $\mathbb{F}$ *is called the order of* $\mathbb{F}$*.*

An important example of a finite field is the field of integers modulo a prime $p$, also called a *prime field.* We denote prime fields by

$$\mathbb{F}_p = \mathbb{Z}_p = \{0, 1, \dots, p-1\}.$$

**Theorem 1.2** (Wedderburn)**.** *Every finite division ring is a field.*

**Definition 1.3.** *For an arbitrary finite field* $\mathbb{F}$ *there exists a positive integer* $p$ *such that* $pa = 0$ *for every* $a \in \mathbb{F}$*. The least of such integers* $p$ *is called the characteristic of* $\mathbb{F}$ *and* $\mathbb{F}$ *is said to be of characteristic* $p$*.*

A basic property of the characteristic is the following.

**Theorem 1.4.** *Any finite field* $\mathbb{F}$ *is of prime characteristic.*

In the rest of the thesis, we mainly focus on finite fields of characteristic 2. Recall that a finite field of characteristic 2 is called a *binary field.*

**Theorem 1.5.** *Let* $p$ *be the characteristic of a field* $\mathbb{F}$*. Then for every* $a, b \in \mathbb{F}$ *and* $n \in \mathbb{N}$ *we have*

$$(a+b)^{p^n} = a^{p^n} + b^{p^n}.$$

## 1.2 Rings of Polynomials

Next, we give a brief overview of rings of polynomials and quotient rings of polynomial rings. Let $R$ be a commutative ring. A *polynomial* over $R$ is a formal expression of the following form:

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_n x^n, a_i \in R.$$

We define the sum of $f(x) = \sum\limits_{i=0}^{n} a_i x^i$ and $g(x) = \sum\limits_{j=0}^{m} b_j x^j$ over $F$ by

$$f(x) + g(x) = \sum_{i=0}^{\max(m,n)} (a_i + b_i) x^i,$$

and their product of two polynomials over $R$ by

$$f(x)g(x) = \sum_{k=0}^{n+m} c_k x^k, \text{ where } c_k = \sum_{i+j=k} a_i b_j.$$

**Definition 1.6.** *The set of all polynomials over $R$ with the operations defined above is called the ring of polynomials over $R$ and denoted by $R[x]$.*

In the following, we consider polynomials over fields. Let $F$ be a field. The concept of divisibility, applied to the ring $F[x]$, yields the following definition. A polynomial $f(x) \in F[x]$ *divides* another polynomial $g(x) \in F[x]$, if there exists a polynomial $h(x) \in F[x]$ such that $g(x) = f(x)h(x)$.

**Definition 1.7.** *A polynomial $p(x) \in F[x]$ is said to be irreducible over $F[x]$ if $p(x)$ has a positive degree and $p(x) = b(x)c(x)$ with $b(x), c(x) \in F[x]$ implies that either $b(x)$ or $c(x)$ is an invertible polynomial.*

Therefore, a non-constant polynomial is irreducible in $F[x]$ if it only allows trivial factorizations. Recall that a factorization is called trivial if at most one factor is non-constant. A non-constant polynomial in $F[x]$ that is not irreducible, is called *reducible in* $F[x]$.

Fix a polynomial $p(x) \in F[x]$. Recall that the *principal ideal* generated by $p(x)$ is the set

$$I := p(x)F[x] = \{p(x)f(x) \mid f(x) \in F[x]\},$$

and the *coset* of a polynomial $g(x)$ modulo $I$ is

$$[g(x)] = g(x) + I = \{g(x) + p(x)f(x) \mid f(x) \in F[x]\}.$$

If we define addition and multiplication of cosets by

$$[g(x)] + [h(x)] = [g(x) + h(x)],$$

$$[g(x)][h(x)] = [g(x)h(x)],$$

then $F[x]/I = \{[g(x)] \mid g(x) \in F[x]\}$ becomes a ring, called the *quotient ring* of $F[x]$ modulo $p(x)$. Observe that equality of cosets means

$$[g(x)] = [h(x)] \iff p(x) \mid g(x) - h(x) \iff g(x) \equiv h(x) \pmod{p(x)}.$$

**Theorem 1.8.** *The quotient ring $F[x]/p(x)F[x]$ is a field if and only if $p(x)$ is irreducible over $F$.*

## 1.3 Field Extensions

Now, let us consider field extensions. Let $F$ be a field. A subset $F'$ of $F$ that is itself a field under the operations of $F$ is called a *subfield* of $F$. Conversely, $F$ is called an *extension* field of $F'$. Note that if $F$ is an extension field of $F'$, then $F$ can be viewed as a vector space over $F'$. Elements $a \in F$ can clearly be multiplied by a scalar $r \in F'$ so that $ra \in F$. Furthermore, all the laws for multiplication by scalars are satisfied essentially by definition.

**Theorem 1.9.** *Every finite field $\mathbb{F}$ with prime characteristic $p$ contains a subfield which is isomorphic to $\mathbb{Z}_p$.*

The above can be used to deduce the possible number of elements in a finite field.

**Theorem 1.10.** *Let $\mathbb{F}$ be a finite field. Then $\mathbb{F}$ has $p^n$ elements, where prime $p$ is the characteristic of $\mathbb{F}$ and $n$ is called the degree of $\mathbb{F}$ over its prime subfield $\mathbb{Z}_p$.*

It can be shown that for every prime $p$ and $n \in \mathbb{N}$, there exists a finite field with $p^n$ elements and all these fields of the same cardinality are isomorphic. Therefore, we can speak of *the* finite field (also called the *the Galois field*) with $q = p^n$ elements, where $p \in \mathbb{P}$ and $n \in \mathbb{N}$. Such fields will be denoted by $\mathrm{GF}(q)$ in the rest of the thesis. For a finite field $\mathrm{GF}(q)$ we denote by $\mathbb{F}_q^*$ the multiplicative group of the nonzero elements of $\mathrm{GF}(q)$.

**Theorem 1.11.** *The multiplicative group $\mathbb{F}_q^*$ of every finite field $\mathrm{GF}(q)$ is cyclic.*

**Definition 1.12.** *Generators of the multiplicative group $\mathbb{F}_q^*$ are called the primitive elements of $\mathrm{GF}(q)$.*

The number of primitive elements of $\mathrm{GF}(q)$ is $\varphi(q-1)$, where $\varphi$ is Euler's totient function. In the next section, we will describe how elements of finite fields can be represented as powers of a primitive element of $\mathbb{F}_q$, but here we will conclude with two useful lemmas.

**Lemma 1.13.** *Let $\mathrm{GF}(q)$ be a finite field. Then for all $a \in \mathbb{F}_q^*$ we have that $a^{q-1} = 1$.*

Note that the previous lemma is an analog of Fermat's Little Theorem.

**Lemma 1.14.** *Every element $a \in \mathrm{GF}(q)$ satisfies $a^q = a$.*

As a matter of fact, the converse also holds.

**Lemma 1.15.** *For any finite extension $K$ of $\mathrm{GF}(q)$ and $a \in K$, if we have $a^q = a$, then $a \in \mathrm{GF}(q)$.*

Now, let us review splitting fields of polynomials. For that, we need to define the roots of a polynomial.

**Definition 1.16.** *An element $b$ is called a root of the polynomial $f(x) \in F[x]$ if $f(b) = 0$.*

Recall that the quotient ring $F[x]/p(x)F[x]$ is a field if and only if $p(x)$ is irreducible over $F$.

**Theorem 1.17.** *Let $F$ be a field, and suppose $p(x) \in F[x]$ is an irreducible polynomial of degree $n \geq 2$. Then $F[x]/p(x)F[x]$ is a field which contains a subfield $F'$ isomorphic to $F$ and, furthermore, $p(x)$ has a root in $F'$.*

Successive use of the previous theorem leads to an extension containing *all* the roots of $p(x)$.

**Theorem 1.18.** *Let $F$ be a field, and suppose $p(x) \in F[x]$ is an irreducible polynomial of degree $n \geq 1$. Then there exists an extension $F'$ of $F$ which contains $n$ roots of $p(x)$.*

**Definition 1.19.** *An extension field $F'$ of $F$ is said to be the splitting field of $f(x) \in F[x]$ over $F$ if $f(x)$ can be written as a product of linear factors, i.e.*

$$f(x) = b(x-a_1)\dots(x-a_n),$$

*where $a_1, \dots, a_n, b \in F'$ and $F'$ is the smallest field containing all the roots of $f(x)$.*

In fact, the preceding can be refined to a more concrete observation.

**Lemma 1.20.** *If $\mathbb{F}$ is an extension field of* $\mathrm{GF}(q)$ *of degree $n$, then the polynomial $x^{q^n} - x \in \mathbb{F}_q[x]$ factors in $\mathbb{F}[x]$ as*

$$x^{q^n} - x = \prod_{a \in \mathbb{F}} (x - a)$$

*and $\mathbb{F}$ is the splitting field of $x^{q^n} - x$ over* $\mathrm{GF}(q)$.

Now, let us state the main characterization theorem for finite fields.

**Theorem 1.21** (Existence and Uniqueness of Finite Fields)**.** *For every prime $p$ and every positive integer $n$ there exists a finite field with $p^n$ elements. Any finite field with $q = p^n$ elements is isomorphic to the splitting field of $x^q - x$ over $GF(p)$.*

**Lemma 1.22.** *Let $p(x)$ be an irreducible polynomial over $\mathbb{F}_q[x]$ of degree $m$. Then $p(x) \mid x^{q^n} - x$ if and only if $m \mid n$.*

**Theorem 1.23.** *If $p(x)$ is an irreducible polynomial of degree $n$ over $\mathbb{F}_q[x]$, then $p(x)$ has a root $\alpha$ in* $\mathrm{GF}(q^n))$. *All the roots of $p(x)$ are simple (i.e. $p(x)$ has no multiple roots) and are given by the $n$ distinct elements $\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}} \in$* $\mathrm{GF}(q^n)$.

**Corollary 1.24.** *Let $p(x)$ be an irreducible polynomial of degree $n$ over $\mathbb{F}_q[x]$, then the splitting field of $p(x)$ over* $\mathrm{GF}(q))$ *is given by* $\mathrm{GF}(q^n)$.

Let $F$ be an arbitrary field. Recall that $a \in F$ is an $n$-th root of unity in $F$ if $a^n = 1$. In the next chapters, we denote the set of all $n$-th roots of unity in $F$ by $H_n$. The next lemma will be used in the following sections.

**Lemma 1.25.** *Let $kl + 1$ be a prime,* $\mathrm{GF}(q)$ *be a finite field and suppose there exists a primitive $(kl + 1)$-th root of unity $a$ in* $\mathrm{GF}(q)$. *Then for any primitive $l$-th root of unity $\eta$ in $\mathbb{Z}_{kl+1}$, we have*

$$\sum_{i=0}^{l-1} a^{\eta^{i+j}} = \sum_{i=0}^{l-1} a^{\eta^i}, j \in \mathbb{N}.$$

## 1.4 Bases for Finite Fields

In this section we consider the finite extension $\mathrm{GF}(q^n)$ of the finite field $\mathrm{GF}(q)$ as a vector space over $\mathrm{GF}(q)$. Note that since $\{\alpha_1, \ldots, \alpha_n\}$ is a basis of $\mathrm{GF}(q^n)$ and since the dimension of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$ is $n$, each element $\alpha \in \mathrm{GF}(q^n)$ can be uniquely represented in the following form:

$$\alpha = c_1\alpha_1 + \cdots + c_n\alpha_n, \ c_i \in \mathrm{GF}(q).$$

**Definition 1.26.** *Fix $\alpha \in \mathrm{GF}(q^n)$. Then the elements $\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}$ are called the conjugates of $\alpha$ over $\mathrm{GF}(q)$.*

Now, let us introduce a useful linear mapping from $\mathrm{GF}(q)$ to $\mathrm{GF}(q^n)$.

**Definition 1.27.** *For $\alpha \in \mathrm{GF}(q^n)$, the trace of $\alpha$ is defined by*

$$\mathrm{Tr}_{\mathrm{GF}(q^n)}(\alpha) = \alpha + \alpha^q + \cdots + \alpha^{q^{n-1}}.$$

Since $(\mathrm{Tr}_{\mathrm{GF}(q^n)}(\alpha))^q = \mathrm{Tr}_{\mathrm{GF}(q^n)}(\alpha)$, the trace of an element $\alpha$ always lies in the base field $\mathrm{GF}(q)$.

As stated above, every finite field is a vector space over each of its subfields, and thus has a vector space basis over each of its subfields. In general, the number of distinct bases of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$ is rather large. Different kinds of bases facilitate certain computations. In the following, we introduce two different types of bases.

**Definition 1.28.** *A basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$ of the form $\{1, \alpha, \ldots, \alpha^n\}$, where $\alpha \in \mathrm{GF}(q^n)$, is called a polynomial basis of $GF(q^n)$ over $\mathrm{GF}(q)$. The element $\alpha$ is called the defining element of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$.*

A root of an irreducible polynomial $p(x)$ over $\mathrm{GF}(q)$ of degree $n$ can be viewed as a generator of a polynomial basis. In other words, we can take a root $\alpha$ of $p(x)$, and its powers $\alpha^i, 0 \leq i \leq n-1$, form a polynomial basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$.

**Definition 1.29.** *A basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$ of the form $\{\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}\}$, where $\alpha \in \mathrm{GF}(q^n)$, is called a normal basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$. Elements $\alpha \in \mathrm{GF}(q^n)$ that give rise to normal bases as above are called normal elements.*

Normal bases will be considered in much more detail in Chapters 3 and 4.

## 1.5 Representation of Field Elements

As discussed in previous section, there is more than one way to represent elements of a finite field. One way is to use quotient rings over an irreducible polynomial $p(x)$ and the other is to use the fact that $\mathbb{F}_q^*$ is cyclic and its elements can be represented as powers of a primitive element.

Addition in finite fields using quotient rings over an irreducible polynomial $p(x)$ is neither difficult nor costly. In the case of binary fields, it is simply the XOR operation. As multiplication is much more complicated, in the following we will focus on that operation.

First, let us study multiplication modulo an irreducible polynomial. Let elements $a, b \in \mathrm{GF}(q^n)$ be represented as polynomials

$$a(x) = \sum_{i=0}^{n-1} a_i x^i \text{ and } b(x) = \sum_{i=0}^{n-1} b_i x^i, \text{ where } a_i, b_i \in \mathrm{GF}(q).$$

In order to multiply two elements $a$ and $b$ in $\mathrm{GF}(q^n)$, we need an irreducible polynomial of degree $n$. Let $p(x) = x^n + r(x)$ be an irreducible polynomial of degree $n$ over the field $\mathrm{GF}(q)$. The product $c = ab \in \mathrm{GF}(q^n)$ is obtained by computing

$$c(x) = a(x)b(x) \pmod{p(x)},$$

where $c(x) = \sum\limits_{i=0}^{n-1} c_i x^i, c_i \in \mathrm{GF}(q)$. Therefore, multiplication in the field $\mathrm{GF}(q^n)$ is accomplished by multiplying the corresponding polynomials modulo the irreducible polynomial $p(x)$. It is easy to see that this kind of multiplication is complicated, especially the reduction modulo $p(x)$. Multiplication with powers of a primitive element is much easier, but leads to convoluted addition, necessitating conversion into polynomials.

Before proceeding to primitive elements and index tables, note that the absolutely fastest way to perform multiplication in finite fields is to precompute its Cayley table. As an example, let us examine the finite field $\mathrm{GF}(16)$ as an extension of $\mathrm{GF}(2)$. Therefore, we need an irreducible polynomial of degree $4$.

Take $p(x) = x^4 + x^3 + 1$ and verify that it is indeed irreducible over $\mathrm{GF}(2)$. If $p(x)$ is irreducible, it must contain a linear or a quadric factor. As $p(1) \neq 0$ and $p(0) \neq 0$, $p(x)$ does not contain linear factors. Thus, we need to show that there exist no polynomials $l(x)$ of degree 2, such that $l(x) \mid p(x)$. Note that, there are exactly four polynomials of degree 2 over $\mathrm{GF}(2)$: $x^2$, $x^2 + 1$, $x^2 + x$ and $x^2 + x + 1$. It is easy to check that none of these divides $p(x)$. Consequently, we have shown that $p(x)$ is irreducible over $\mathrm{GF}(2)$. Thus, we have that $[x^4] = [x^3 + 1]$. The Cayley table for multiplication consists of all possible binary polynomial multiplications, where polynomials have degree at most $3$ and the result is reduced modulo $x^4 + x^3 + 1$.

The problem with Cayley tables is that they are not feasible to compute or store for large $n$, and may otherwise lead to memory architecture-related vulnerabilities. Furthermore, we will end up with huge tables which cannot be stored efficiently. Therefore, we need an alternative.

Now let us combine the two representations, leading to much more memory-efficient multiplication. For that we use the so-called *index tables*.

Let $a$ be primitive in $\mathrm{GF}(q)$ and let $p(x) = x^n + r(x)$ be an irreducible polynomial of degree $n$ over the field $\mathrm{GF}(q)$. As $a$ is a generator of $\mathbb{F}_q^*$, we have that

$$\mathrm{GF}(q^n) = \{0, 1, a, a^2, \ldots, a^{q^n-1}\}.$$

Also, note that since multiplication includes reduction modulo $p(x)$, we shall also use the fact that $[a^n + r(a)] = [0]$, meaning $[a^n] = [-r(a)]$.

Let us proceed with our example and check whether $[x]$ is primitive in $\mathrm{GF}(16)$. As prime factors of $|\mathbb{F}_{16}^*| = 15$ are 3 and 5, and we have that neither $[x]^3 \neq [1]$ nor $[x]^5 = [x]^4[x] = [x^3 + x + 1] \neq [1]$, we can take $[x]$ as a primitive element. Thus, we have that

$$\begin{aligned}
[x]^4 &= [x^3 + 1], \\
[x]^5 &= [x]^4[x] = [x^3 + x + 1], \\
[x]^6 &= [x]^5[x] = [x^3 + x + 1][x] = [x^4 + x^2 + x] = [x^3 + x^2 + x + 1], \\
&\dots \\
[x]^{14} &= [x]^{13}[x^2 + x][x] = [x^3 + x^2], \\
[x]^{15} &= [x]^{14}[x^3 + x^2][x] = [x^4 + x^3] = [x^3 + x^3 + 1] = [1].
\end{aligned}$$

The result can be represented in the index table, where instead of writing $a^k$ we simply write $k$ and instead of writing a polynomial $k_{n-1}a^{n-1} + \cdots + k_1 a + k_0$, we write $k_{n-1} \ldots k_1 k_0$. Thus, we arrive at the following result:

| $k$ | $k_3k_2k_1k_0$ |
|---|---|
| 0 | 0001 |
| 1 | 0010 |
| 2 | 0100 |
| 3 | 1000 |
| 4 | 1001 |
| 5 | 1011 |
| 6 | 1111 |
| 7 | 0111 |
| 8 | 1110 |
| 9 | 0101 |
| 10 | 1010 |
| 11 | 1101 |
| 12 | 0011 |
| 13 | 0110 |
| 14 | 1100 |
| 15 | 1111 |

This index table can be used for fast multiplication in $\mathrm{GF}(16)$. Therefore, it is useful to know a generator of a multiplicative group of a finite field and an irreducible polynomial whose root this generator is. Note that finding irreducible polynomials is quite difficult in general and we will return to this topic in subsequent chapters. Also, index tables, while more memory-efficient by a factor of $q$, inherit the same problem that make Cayley tables infeasible for very large $q$.

# 2 Multiplication Using Polynomial Bases

As discussed in the previous section, there is more than one way to implement multiplication in finite fields viewed as vector spaces over subfields. In order to specify a multiplication rule, it is necessary to choose a basis. When it comes to software implementations, using polynomial bases is more efficient than normal bases. In the following we describe several algorithms for binary field arithmetic using polynomial bases.

## 2.1 Standard Field Multiplication

Let $a, b \in \mathrm{GF}(2^n)$ be represented as polynomials

$$a(x) = \sum_{i=0}^{n-1} a_i x^i \text{ and } b(x) = \sum_{i=0}^{n-1} b_i x^i, \text{ where } a_i, b_i \in \mathrm{GF}(2)$$

or, equivalently, as vectors $a = (a_{n-1}, \ldots, a_1, a_0)$ and $b = (b_{n-1}, \ldots, b_1, b_0)$. Let $p(x) = x^n + r(x)$ be an irreducible polynomial of degree $n$ over $\mathrm{GF}(2)$.

The simplest algorithm for field multiplication using polynomial representation is the shift-and-add method. This method is based on the observation that

$$a(x)b(x) = a_{n-1}x^{n-1}b(x) + \ldots + a_1 x b(x) + a_0 b(x).$$

Thus, we successively compute $x^i b(x)$ modulo $p(x)$ for all $1 \leq i \leq n-1$ and add all the results for which $a_i = 1$. If $b(x) = b_{n-1}x^{n-1} + \ldots + b_2x^2 + b_1x + b_0$, then

$$\begin{aligned} b(x)x &= b_{n-1}x^n + \ldots + b_2x^3 + b_1x^2 + b_0x \\ &\equiv b_{n-1}r(x) + (b_{n-2}x^{n-1} + \ldots + b_2x^3 + b_1x^2 + b_0x) \pmod{p(x)}. \end{aligned}$$

Therefore, $b(x)x \pmod{p(x)}$ can be computed by a left-shift of the vector representation of $b(x)$, followed by the addition of $r(x)$ to $b(x)$, if the most significant bit $b_{n-1}$ is 1. This algorithm is presented as Algorithm 1.

The shift-and-add method is not particularly suitable for software implementations as the bitwise shifts are costly to implement for processor architecture based on fixed-length words.

**Algorithm 1** Right-To-Left Shift-and-Add Field Multiplication

```
Input: $a(x) = \sum_{i=0}^{n-1} a_i x^i, b(x) = \sum_{i=0}^{n-1} b_i x^i, a_i, b_i \in \mathrm{GF}(2)$
Output: $c(x) = a(x)b(x) \pmod{p(x)} = \sum_{i=0}^{n-1} c_i x^i, c_i \in \mathrm{GF}(2)$
if $a_0 = 1$ then
    $c(x) \leftarrow b(x)$
else
    $c(x) \leftarrow 0$
for $i = 0 \ldots n-1$ do
    $b(x) \leftarrow b(x)x \pmod{p(x)}$
    if $a_i = 1$ then
        $c(x) \leftarrow b(x) + c(x)$
return $c(x)$
```

## 2.2 Polynomial Multiplication

Next, we consider faster methods for field multiplication. First, we have two improved algorithms for simply multiplying field elements that are represented as polynomials. Then we present an algorithm for faster reduction modulo an irreducible polynomial $p(x)$.

To begin, it is necessary to describe how polynomials are stored in software. Let $w$ be the word-length of the processor (usually $w$ is a multiple of $8$) and $t = \lceil \frac{n}{w} \rceil$. Thus, the vector $a = (a_{n-1}, \ldots, a_1, a_0)$ may be stored in an array of $t$ $w$-bit words:

$$A = (A[t-1], \ldots, A[1], A[0]),$$

where the rightmost bit is $a_0$ and the leftmost $wt - n$ bits are set to $0$. The $i$-th bit of the $j$-th word is denoted by $A[j][i]$. We also use the following notion: if $C = (C[n], \ldots, C[1], C[0])$, then $C\{j\} = (C[n], \ldots, C[j+1], C[j])$.

Now, let us introduce a more efficient method for polynomial multiplication: the comb method. Here, multiplication is implemented in two separate steps, first performing polynomial multiplication to obtain a $2n$-bit-length polynomial and then reducing it via reduction polynomials.

The right-to-left comb method for polynomial multiplication is based on the observation that if $b(x)x^i$ has been computed for some $i \in \{0, \ldots, w-1\}$, then $b(x)x^{wj+i}$ can be easily computed by appending $j$ zero words to the right of the vector representation of $b(x)x^i$. Algorithm 2 processes the bits of the words of $A$ from right to left.

**Algorithm 2** Right-to-Left Comb Method for Polynomial Multiplication

```
Input: $a(x) = \sum_{i=0}^{n-1} a_i x^i, b(x) = \sum_{i=0}^{n-1} b_i x^i, a_i, b_i \in \mathrm{GF}(2)$
Output: $c(x) = a(x)b(x) = \sum_{i=0}^{2n-2} c_i x^i, c_i \in \mathrm{GF}(2)$
$C \leftarrow 0$
for $i = 0 \dots w-1$ do
    for $j = 0 \dots t-1$ do
        if $A[j][i] = 1$ then
            $C\{j\} = C\{j\} + B$
    if $i \neq w-1$ then
        $B \leftarrow Bx$
return $C$
```

Note that Algorithm 2 can be also improved. One way to do this is the left-to right comb method which processes the bits of $a$ from left to right, as follows:

$$a(x)b(x) = (\dots((a_{n-1}b(x)x+(a_{n-2}b(x))x+(a_{n-3}b(x))x+\dots+a_1b(x))x+a_0b(x).$$

Note that this modification does not accelerate multiplication in comparison with Algorithm 2, but this left-to right comb method can be considerably accelerated at the expense of some storage overhead by first computing $u(x)b(x)$ for all polynomials $u(x)$ of degree less than some fixed $w'$ ($w' \mid w$), and then processing the bits of $A[j]$ one at a time. This modified method is called the left-to-right comb method with windows of width $w'$. The corresponding algorithm is presented below as Algorithm 3.

As stated above, the previous two algorithms only perform polynomial multiplication. The maximal degree of the output polynomial $c(x)$ is $2n-2$. In some cases, the modular reduction required for field multiplication is done separately. In others, the irreducible polynomial $p(x)$ is included as an input to the algorithm and reduction is done mid-step. For example, Algorithm 3 can be modified to calculate $u(x)b(x) \pmod{p(x)}$ which may allow optimization in the next steps.

**Algorithm 3** Left-to-Right Comb Method for Polynomial Multiplication with Windows of Width $w'$

```
Input: $a(x) = \sum_{i=0}^{n-1} a_i x^i, b(x) = \sum_{i=0}^{n-1} b_i x^i, a_i, b_i \in \mathrm{GF}(2)$
Output: $c(x) = a(x)b(x) = \sum_{i=0}^{2n-2} c_i x^i, c_i \in \mathrm{GF}(2)$
Precompute $B_u = u(x)b(x)$ for all polynomials $u(x)$ of degree at most $w' - 1$
for all $i = \frac{w}{w'} \dots 0$ do
    for all $j = 0 \dots t-1$ do
        $u \leftarrow (u_{w'-1}, \dots, u_1, u_0)$ where $u_k = A[j][w'i + k]$
        $C\{j\} \leftarrow C\{j\} + B_u$
    if $i \neq 0$ then
        $C \leftarrow Cx^{w'}$
return $C$
```

## 2.3 Field Reduction

In this section we explore how to efficiently reduce polynomials of degree $2n-2$, as the output polynomial of the comb algorithms discussed above have degree at most $2n-2$.

Let $p(x) = x^n + r(x)$ be an irreducible polynomial of degree $n$ over $\mathrm{GF}(2)$ and $c(x) = \sum_{i=0}^{2n-2} c_i x^i, c_i \in \mathrm{GF}(2)$. The following algorithm is based on the observation that

$$\begin{aligned} x^n &\equiv r(x) \pmod{p(x)}, \\ x^{n+k} &\equiv r(x)x^k \pmod{p(x)}, \end{aligned}$$

and thus

$$\begin{aligned} & c(x) \\ & = c_{2n-2}x^{2n-2} + \cdots + c_1 x + c_0 \\ & \equiv (c_{2n-2}x^{n-2} + \cdots + c_n)r(x) + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0 \pmod{p(x)}. \end{aligned}$$

In the following algorithm, reduction modulo $p(x)$ is done one bit at a time, starting from the leftmost bit. In order to accelerate reduction, polynomials $x^k r(x)$ $0 \leq k \leq w-1$ are precomputed. Moreover, Algorithm 4 works on the bit, not word level. Recall that $t = \lceil \frac{n}{w} \rceil$, where $w$ is the word length of the processor.

Multiplication in a finite field is simply the product of two field polynomials, reduced modulo $p(x)$, but there are polynomials modulo which reduction is more efficient than for others.

**Algorithm 4** Bit-Level Modular Reduction

```
Input: $c(x) = \sum_{i=0}^{2n-2} c_i x^i, c_i \in \mathrm{GF}(2), p(x) = x^n + r(x)$
Output: $c(x) \pmod{p(x)}$
Precompute $u_k(x) = x^k r(x)$ for all $x^n r(x), 0 \leq k \leq w - 1$
for all $i = (2n-2) \ldots n$ do
    if $c_i = 1$ then
        $j \leftarrow \frac{i-n}{w}$
        $k \leftarrow (i-n) - wj$
        $C\{j\} \leftarrow C\{j\} + u_k(x)$
return $(C[t-1], \ldots, C[0])$
```

Specifically, the reduction of polynomials modulo $p(x)$ is particularly efficient if $p(x)$ has a small number of terms. The irreducibles with the least number of terms are the trinomials $x^n + x^a + 1$. Thus, it is common practice to choose a trinomial for the field polynomial, provided that one exists. For example, Algorithm 4 can be accelerated even more when using trinomials, because the space requirements will be smaller and the additions involving $x^k r(x)$ become faster.

If an irreducible trinomial of degree $n$ does not exist, then the next best polynomials are the pentanomials $x^n + x^a + x^b + x^c + 1$. In binary fields for every $n$ up to 1000, there exists either an irreducible trinomial or pentanomial of degree $n$. Such polynomials are widely recommended in all major standards. For software implementation, reduction polynomials with middle terms close to each other are more suitable. The existence of irreducible polynomials will be discussed further in Chapter 5.

## 2.4 Montgomery Multiplication in $\mathrm{GF}(2^n)$

There is another variation of multiplication using polynomial representation. Instead of computing $a(x)b(x)$ in $\mathrm{GF}(2^n)$, we calculate $a(x)b(x)r(x)^{-1}$ in $\mathrm{GF}(2^n)$, where $r(x)$ is a special fixed element of $\mathrm{GF}(2^n)$. This method is based on an idea by Montgomery for modular multiplication of integers to replace division in classical reduction algorithms with less-expensive operations. The method is not efficient for a single modular multiplication, but can be used effectively in computations such as modular exponentiation, where many multiplications are performed for a given input.

It can be shown that Montgomery's technique is applicable to the field $\mathrm{GF}(2^n)$ as well. The selection of $r(x) = x^n$ turns out to be very useful in obtaining fast software multiplication. Thus, $r(x)$ is the element of the field, represented by the polynomial $r(x) \pmod{p(x)}$, i.e., if $p = (p_n, p_{n-1}, \ldots, p_1, p_0)$, then $r = (p_{p-1}, \ldots, p_1, p_0)$, where $p(x)$ is the field polynomial. Montgomery multiplication requires that both $r(x)$ and $p(x)$ are relatively prime, i.e. $\gcd(p(x), r(x)) = 1$, since $r(x) = x^n$, we must have $x \nmid p(x)$. As $p(x)$ is an irreducible polynomial over the field GF$(2)$, this will always be the case. Since $r(x)$ and $p(x)$ are relatively prime, there exist two polynomials $r(x)^{-1}$ and $p'(x)$ with the property that

$$r(x)r(x)^{-1} + p(x)p'(x) = 1,$$

where $r(x)^{-1}$ is the inverse of $r(x)$ modulo $p(x)$. The polynomials $r(x)^{-1}$ and $p'(x)$ can be computed using the extended Euclidean algorithm.

The Montgomery product of $a(x)$ and $b(x)$ is defined as the product

$$c(x) = a(x)b(x)r(x)^{-1} \pmod{p(x)}.$$

For an implementation see Algorithm 5.

Note that from $u(x) = t(x)p'(x) \pmod{r(x)}$ implies that there exists an polynomial $k(x)$ over GF$(2)$ such that

$$u(x) = t(x)p'(x) + k(x)r(x).$$

Therefore, as $r(x) = x^n$ we can write

$$\begin{aligned} c(x) &= \frac{1}{x^n}(t(x) + u(x)p(x)) = \frac{1}{x^n}(t(x) + (t(x)p'(x) + k(x)r(x))p(x)) \\ &= \frac{1}{x^n}(t(x) + t(x)p'(x)p(x) + k(x)r(x)p(x)). \end{aligned}$$

Now using that $r(x)r(x)^{-1} + p(x)p'(x) = 1$, $c(x)$ can be computed as

$$\begin{aligned} c(x) &= \frac{1}{x^n}(t(x) + t(x)(1 + r(x)r(x)^{-1}) + k(x)r(x)p(x)) \\ &= \frac{1}{x^n}(t(x)r(x)r(x)^{-1} + k(x)r(x)p(x)) \\ &= t(x)r(x)^{-1} + k(x)p(x) \equiv a(x)b(x)r(x)^{-1} \pmod{p(x)}, \end{aligned}$$

as required. As mentioned before, the algorithm is similar to the algorithm given by Montgomery for the multiplication of integers. The difference is that the final subtraction step required in the integer case is not necessary for polynomials, as the degree of polynomial $c(x)$ computed by this algorithm is less than equal to $n - 1$. Indeed, the degree of $c(x)$ can be found to be

$$\begin{aligned} \deg c(x) &\leq \max(\deg t(x), \deg u(x) + \deg p(x)) - \deg r(x) \\ &\leq \max(2n - 2, n - 1 + n) - n = 2n - 1 - n = n - 1. \end{aligned}$$

**Algorithm 5** Montgomery Multiplication in $\mathrm{GF}(2^n)$

```
Input: $a(x) = \sum_{i=0}^{n-1} a_i x^i,\ b(x) = \sum_{i=0}^{n-1} b_i x^i,\ p'(x) = \sum_{i=0}^{n-1} p'_i x^i,$
$r(x) = x^n,\ a_i, b_i, p'_i \in \mathrm{GF}(2)$
Output: $c(x) = \sum_{i=0}^{n-1} c_i x^i = a(x)b(x)r(x)^{-1} \pmod{p(x)}, c_i \in \mathrm{GF}(2)$
$t(x) \leftarrow a(x)b(x)$
$u(x) \leftarrow t(x)p'(x) \pmod{r(x)}$
$c(x) \leftarrow \frac{t(x)+u(x)p(x)}{r(x)}$
return $c(x)$
```

Thus, the polynomial $c(x)$ is already reduced.

In the above algorithm, the computation of $c(x)$ involves a regular polynomial multiplication, a modulo $r(x)$ multiplication, and finally a regular multiplication and a division by $r(x)$ as the last step. Note that the modular multiplication and division operations are fast operations since we have $r(x) = x^n$.

Also, it turns out that the computation of $p'(x)$ can be completely avoided if the coefficients of $a(x)$ are scanned one bit at a time. Recall that we need to compute $a(x)b(x)r(x)^{-1}$. Again, we follow the analogy with the integer algorithm.

Firstly, if $c_0 = 1$, we add $p(x)$ to $c(x)$. Therefore, since $p(x)$ is irreducible, $p_0 = 1$ and $c(x)$ becomes divisible by $x$. If $c_0 = 0$, we do not add $p(x)$ to $c(x)$. In other words, we compute $c(x) = c(x)+c_0p(x)$ after the addition step, making $c(x)$ always divisible by $x$. This bit-level algorithm for Montgomery multiplication is given as Algorithm 6.

**Algorithm 6** Bit-Level Algorithm for Montgomery Multiplication

```
Input: $a(x) = \sum_{i=0}^{n-1} a_i x^i, b(x) = \sum_{i=0}^{n-1} b_i x^i, p(x) = \sum_{i=0}^{n-1} p_i x^i, a_i, b_i, p_i \in \mathrm{GF}(2)$
Output: $c(x) = \sum_{i=0}^{n-1} c_i x^i = a(x)b(x)r(x)^{-1} \pmod{p(x)}, c_i \in \mathrm{GF}(2)$
$c(x) \leftarrow 0$
for $i \leftarrow 0 \ldots n-1$ do
    $c(x) \leftarrow c(x) + a_i b(x)$
    $c(x) \leftarrow c(x) + c_0 p(x)$
    $c(x) \leftarrow c(x)/x$
return $c(x)$
```

Note that the bit-level algorithm for Montgomery multiplication can also be generalized to a world-level algorithm as we did in the previous section for the comb methods by proceeding word by word.

Observe that in order to compute $a(x)b(x) \pmod{p(x)}$ there is a need for conversion of the output. To achieve that, the output of the algorithm must be multiplied with $r(x)$. Therefore, actual speedup is achieved only when performing a whole series of modular multiplications on intermediate results. For example, this technique can be used to obtain fast software implementations of discrete exponentiation.

# 3 Multiplication Using Normal Bases

In this chapter we discuss how multiplication in $\mathrm{GF}(q^n)$ can be done in general using multiplication tables for normal bases (which are *not* Cayley tables).

We view $\mathrm{GF}(q^n)$ as a vector space over $\mathrm{GF}(q)$. Let $\alpha_0, \alpha_1, ..., \alpha_{n-1} \in \mathrm{GF}(q^n)$ be linearly independent over $\mathrm{GF}(q)$. Then every element $A \in \mathrm{GF}(q^n)$ can be represented as $A = \sum\limits_{i=0}^{n-1} a_i\alpha_i,\ a_i \in \mathrm{GF}(q)$, or, equivalently, as a vector $A = (a_0, a_1, ..., a_{n-1})$. Let $B = (b_0, b_1, ..., b_{n-1})$ and

$$A \times B =: C = (c_0, c_1, ..., c_{n-1}).$$

We have $\alpha_i\alpha_j = \sum\limits_{k=0}^{n-1} t_{i,j}^{(k)}\alpha_k = (t_{i,j}^{(0)}, t_{i,j}^{(1)}, ..., t_{i,j}^{(n-1)})$ for some $t_{i,j}^{(k)} \in \mathrm{GF}(q)$. Thus, we can write

$$c_k = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j t_{i,j}^{(k)} = AT_kB^T, 0 \leq k \leq n-1,$$

where $T_k = (t_{i,j}^{(k)})$ are $n \times n$ matrices over $\mathrm{GF}(q)$ and $B^T$ is the transpose of $B$. The collection of matrices $\{T_k\}$ is called the *multiplication table* for $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$. Note that the matrices $\{T_k\}$ are independent of $A$ and $B$.

For some bases the corresponding multiplication tables $\{T_k\}$ are simpler than others as they may have fewer non-zero entries and more regularities so that one may choose some multiplication algorithm to make a hardware or software design of finite field multiplication feasible for large finite fields.

## 3.1 Multiplication Algorithm of Massey-Omura

In the following we present a multiplication algorithm for normal bases, which is based on the Massey-Omura scheme.

Let $\mathrm{GF}(q^n)$ be a finite field and $N = \{\alpha_0, \alpha_1, ..., \alpha_{n-1}\}$ be a normal basis, where $\alpha_i = \alpha^{q^i}$ for a fixed $\alpha \in \mathrm{GF}(q^n)$. Thus, we have $\alpha_i^{q^k} = \alpha_{i+k}$ for any integer $k$, where the subscripts of $\alpha$ are reduced modulo $n$.

As before, take

$$\alpha_i\alpha_j = \sum_{k=0}^{n-1} t_{i,j}^{(k)}\alpha_k,$$

with coordinates $t_{i,j}^{(k)} \in \mathrm{GF}(q)$. Then by the above, Theorem 1.5 and Lemma 1.14, we have for all $l \in \mathbb{N} \cup \{0\}$

$$\alpha_i\alpha_j = (\alpha_{i-l}\alpha_{j-l})^{q^l} = \left(\sum_{k=0}^{n-1} t_{i-l,j-l}^{(k)}\alpha_k\right)^{q^l} = \sum_{k=0}^{n-1}(t_{i-l,j-l}^{(k)})^{q^l}\alpha_{k+l} = \sum_{k=0}^{n-1} t_{i-l,j-l}^{(k-l)}\alpha_k,$$

where the subscripts and superscripts of $t_{i,j}^{(k)}$ are reduced modulo $n$. Therefore, we get that

$$t_{i,j}^{(k)} = t_{i-l,j-l}^{(k-l)} \text{ for all } 0 \leq i, j, l \leq n-1.$$

In particular, if we take $k = l$,

$$t_{i,j}^{(k)} = t_{i-k,j-k}^{(0)}.$$

For simplicity we denote $t_{i,j}^{(0)} := t_{i,j}$ and take $T = (t_{i,j})$ to be the matrix with entries $t_{i,j}$. Hence, we can write that

$$\alpha_0\alpha_i = \alpha\alpha_i = \sum_{j=0}^{n-1} t_{i,j}\alpha_j.$$

Therefore, instead of $n$ matrices, we need only one. The number of non-zero entries in T is called the *complexity* of the normal basis and denoted by $c_N$ .

**Theorem 3.1.** *For any normal basis N of* $\mathrm{GF}(q^n)$ *over* $\mathrm{GF}(q)$, $c_N$ *is at least* $2n-1$.

*Proof.* Let $\{\alpha, \alpha^q, ..., \alpha^{q^{n-1}}\} = \{\alpha_0, \alpha_1, ..., \alpha_{n-1}\}$ be a normal basis of $\mathrm{GF}(q^n)$. Then $b = \sum\limits_{i=0}^{n-1} \alpha_i = \mathrm{Tr}_{\mathrm{GF}(q^n)}(\alpha) \in \mathrm{GF}(q)$. Therefore, we have

$$\begin{aligned}\alpha_0 b + \alpha_1 0 + \ldots + \alpha_{n-1}0 = \alpha_0 b = \alpha b = \alpha\sum_{i=0}^{n-1}\alpha_i = \sum_{i=0}^{n-1}\alpha\alpha_i \\ = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} t_{i,j}\alpha_j = \sum_{j=0}^{n-1}\alpha_j(\sum_{i=0}^{n-1} t_{i,j}).\end{aligned}$$

Hence, $\sum\limits_{i=0}^{n-1} t_{i,j} = b$ when $j = 0$, otherwise $\sum\limits_{i=0}^{n-1} t_{i,j} = 0$.

Since $\alpha$ is non-zero and generates the normal basis, $\{\alpha\alpha_i : 0 \leq i \leq n-1\}$ is also a basis of GF($q^n$), so the matrix $T = (t_{i,j})$ is invertible. Thus, for each $j$ there is at least one non-zero $t_{i,j}$. Note that for each $j \neq 0$, there must be at least two non-zero elements in each column, since the column must sum to zero. So there are at least $2n-1$ non-zero terms in $T$. This completes the proof. □

A normal basis N is called *optimal* if $c_N = 2n-1$.

Now, take $A = \sum_{i=0}^{n-1} a_i\alpha_i \in \mathrm{GF}(q^n)$ and $B = \sum_{i=0}^{n-1} b_i\alpha_i \in \mathrm{GF}(q^n)$. Then $C = A \times B = \sum_{k=0}^{n-1} c_k\alpha_k \in \mathrm{GF}(q^n)$. On the other hand, we can write

$$\begin{aligned} C = A \times B &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_j\alpha_i\alpha_j = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_j \left(\sum_{k=0}^{n-1} t_{i,j}^{(k)}\alpha_k\right) \\ &= \sum_{k=0}^{n-1}\left(\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_jt_{i,j}^{(k)}\right)\alpha_k. \end{aligned}$$

Using our result that $t_{i,j}^{(k)} = t_{i-k,j-k}$, we have the coefficients $c_k$ of $C = A \times B$ as

$$c_k = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_jt_{i,j}^{(k)} = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_jt_{i-k,j-k} = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_{i+k}b_{j+k}t_{i,j}, \qquad (1)$$

where the subscripts and the superscripts of $a, b$ and $t_{i,j}$ are reduced modulo $n$.

Therefore, the coordinates of $C$ can be computed by cyclically shifting the coordinates of $A$ and $B$ and multiplying them with $T$.

## 3.2 Multiplication Algorithm of Reyhani-Masoleh and Hasan

The multiplication algorithm for $\mathrm{GF}(2^n)$ by Reyhani-Masoleh and Hasan is based on the use of $\alpha\alpha_i$ instead of $\alpha_i\alpha_j$ and the symmetry between $\alpha\alpha_i$ and $\alpha\alpha_{n-1}$.

Let $N = \{\alpha_0, \alpha_1, ..., \alpha_{n-1}\}$ be a normal basis of $\mathrm{GF}(2^n)$ over $\mathrm{GF}(2)$, then $\alpha_i^{2^k} = \alpha_{i+k}$. Take $A = \sum_{i=0}^{n-1} a_i\alpha_i \in \mathrm{GF}(2^n)$ and $B = \sum_{i=0}^{n-1} b_i\alpha_i \in \mathrm{GF}(2^n)$, then

$$\begin{aligned} C = A \times B &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_j\alpha_i\alpha_j = \sum_{i=0}^{n-1} a_ib_i\alpha_i^2 + \sum_{i=0}^{n-1}\sum_{j\neq i} a_ib_j(\alpha\alpha_{j-i})^{2^i} \\ &= \sum_{i=0}^{n-1} a_ib_i\alpha_{i+1} + \sum_{i=0}^{n-1}\sum_{j\neq i} a_ib_j(\alpha\alpha_{j-i})^{2^i} = \sum_{i=0}^{n-1} a_ib_i\alpha_{i+1} + \sum_{i=0}^{n-1}\sum_{j\neq 0} a_ib_{j+i}(\alpha\alpha_j)^{2^i}. \end{aligned}$$

Denoting $v = \lfloor \frac{n-1}{2} \rfloor$, for odd $n$ we have on the right side

$$\sum_{i=0}^{n-1}\sum_{j=1}^{v} a_ib_{j+i}(\alpha\alpha_j)^{2^i} + \sum_{i=0}^{n-1}\sum_{j=n-v}^{n-1} a_ib_{j+i}(\alpha\alpha_j)^{2^i},$$

and for even $n$ we have

$$\sum_{i=0}^{n-1}\sum_{j=1}^{v} a_i b_{j+i}(\alpha\alpha_j)^{2^i} + \sum_{i=0}^{n-1}\sum_{j=n-v}^{n-1} a_i b_{j+i}(\alpha\alpha_j)^{2^i} + \sum_{i=0}^{n-1} a_i b_{v+1+i}(\alpha\alpha_{v+1})^{2^i},$$

where summation has been rearranged with respect to $j$. Furthermore,

$$\begin{aligned}\sum_{i=0}^{n-1}\sum_{j=n-v}^{n-1} a_i b_{j+i}(\alpha\alpha_j)^{2^i} &= \sum_{i=0}^{n-1}\sum_{j=1}^{v} a_i b_{n-j+i}(\alpha\alpha_{n-j})^{2^i}\\ = \sum_{i=0}^{n-1}\sum_{j=1}^{v} a_{i+j} b_i(\alpha\alpha_{n-j})^{2^{i+j}} &= \sum_{i=0}^{n-1}\sum_{j=1}^{v} a_{i+j} b_i(\alpha\alpha_j)^{2^i},\end{aligned}$$

where summation has been rearranged with respect to $i$ and then all the subscripts are reduced modulo $n$. Therefore, we get that

$$\begin{aligned}C = A \times B &= \sum_{i=0}^{n-1} a_i b_i \alpha_{i+1} + \sum_{i=0}^{n-1}\sum_{j=1}^{v}(a_i b_{j+i} + a_{j+i} b_i)(\alpha\alpha_j)^{2^i}\\ &= \sum_{i=0}^{n-1}(a_i b_i \alpha_1 + \sum_{j=1}^{v}(a_i b_{j+i} + a_{j+i} b_i)\alpha\alpha_j)^{2^i}\end{aligned}$$

or

$$\begin{aligned}C &= A \times B\\ &= \sum_{i=0}^{n-1} a_i b_i \alpha_{i+1} + \sum_{i=0}^{n-1}\sum_{j=1}^{v}(a_i b_{j+i} + a_{j+i} b_i)(\alpha\alpha_j)^{2^i} + \sum_{i=0}^{n-1} a_i b_{v+1+i}(\alpha\alpha_{v+1})^{2^i}\\ &= \sum_{i=0}^{n-1}(a_i b_i \alpha_1 + \sum_{j=1}^{v}(a_i b_{j+i} + a_{j+i} b_i)\alpha\alpha_j + a_i b_{v+1}(\alpha\alpha_{v+1}))^{2^i}.\end{aligned}$$

Now, combining the above two equations, the following theorem can be obtained.

**Theorem 3.2** ([11], Theorem 1). *Let* $\alpha_0, \alpha_1, ..., \alpha_{n-1} \in \mathrm{GF}(2^n)$ *be a normal basis of* $\mathrm{GF}(2^n)$ *over* $\mathrm{GF}(2)$, $A = \sum_{i=0}^{n-1} a_i\alpha_i$ *and* $B = \sum_{i=0}^{n-1} b_i\alpha_i,\ a_i, b_i \in \mathrm{GF}(2)$. *Then*

$$C = A \times B = (((F_{n-1}^2 + F_{n-2})^2 + F_{n-3})^2 + \ldots + F_1)^2 + F_0, \qquad (2)$$

*where* $F_i = F_i(A, B) = a_{i-g} b_{i-g}\alpha + \sum_{j=1}^{v} z_{i,j}\alpha\alpha_j$, $g \in \{0, 1\}$ *and*

$$z_{i,j} = \begin{cases} (a_i + a_{i+j})(b_i + b_{i+j}) & \textit{if } g = 0 \textit{ and } n \textit{ is odd}, \\ a_i b_{j+i} + a_{j+i} b_i & \textit{if } q = 1 \textit{ and } n \textit{ is odd}, \\ b_i(a_i + a_{i+v}) & \textit{if } g = 0 \textit{ and } n \textit{ is even}, \\ a_i b_{i+v} & \textit{if } q = 1 \textit{ and } n \textit{ is even}. \end{cases}$$

Note that (2) is the key equation for the multiplier architectures AESMPO and XESMPO proposed by Reyhani-Masoleh and Hasan in [11].

# 4 Construction of Normal Bases

## 4.1 Normal Bases over $\mathrm{GF}(q^n)$

We have seen that normal bases of low complexity are desirable in implementing finite field arithmetic. In this chapter we describe a general construction of such bases. First, we need the following lemma.

**Lemma 4.1.** *Let $n$, $k$ be positive integers such that $nk+1$ is a prime, and suppose that $\gcd(\frac{nk}{e}, n) = 1$, where $e$ is the order of $q$ modulo $nk+1$. Let $\eta$ be a primitive $k$-th root of unity in $\mathbb{Z}_{nk+1}$. Then, every non-zero element $r$ in $\mathbb{Z}_{nk+1}$ can be written uniquely in the following form:*

$$r = \eta^i q^j, \ 0 \leq i \leq k-1, \ 0 \leq j \leq n-1.$$

*Proof.* Take $l = \frac{nk}{e}$. Using the properties of indices, there exists a primitive element $g$ in $\mathbb{Z}^*_{nk+1}$ such that $q = g^l$, as $l$ is the index of $q$ relative to $g$. Since $g$ is primitive, the order of $g$ is $nk$, and the order of $\eta$ is $k$, we have that $\eta = g^{na}$ for some integer $a$. As $\eta$ is a primitive $k$-th root of unity, $\gcd(a, k) = 1$. Next, let us suppose that there are $0 \leq i, s \leq k-1$ and $0 \leq j, t \leq n-1$ such that

$$\eta^i q^j \equiv \eta^s q^t \pmod{nk+1}.$$

Therefore, we have

$$\eta^{i-s} \equiv q^{t-j} \pmod{nk+1},$$
$$g^{na(i-s)} \equiv g^{l(t-j)} \pmod{nk+1}.$$

In particular, as $nk+1$ is a prime and $g$ is a primitive root modulo $(nk+1)$,

$$na(i-s) \equiv l(t-j) \pmod{nk}.$$

As $\gcd(l, n) = 1$, we have, by Euclid's Lemma, $n|(t-j)$. Hence, $t = j$, because $0 \leq j, t \leq n-1$. Thus,

$$a(i-s) \equiv 0 \pmod{k}.$$

But as $\gcd(a, k) = 1$, we get by Euclid's Lemma that $k \mid (i-s)$, and thus $i = s$ as $0 \leq i, s \leq k-1$. We have proven thus that

$$\eta^i q^j, \ 0 \leq i \leq k-1, \ 0 \leq j \leq n-1$$

are all distinct. As $\eta$ and $q$ are invertible, $\eta^i q^j \not\equiv 0 \pmod{nk+1}$, and thus every non-zero element $r$ in $\mathbb{Z}_{nk+1}$ can be expressed uniquely in the required form. The proof is complete. □

In the following define $r(i, j) := \eta^i q^j : \mathbb{Z}_k \times \mathbb{Z}_n \to \mathbb{Z}^*_{nk+1}$.

**Theorem 4.2.** *Let $q = p^l$ be a prime or a prime power, and let $n$, $k$ be positive integers such that $nk + 1$ is a prime and $nk + 1 \nmid q$. Suppose that $\gcd(\frac{nk}{e}, n) = 1$, where $e$ is the order of $q$ modulo $nk + 1$. Then, for any primitive $(nk + 1)$-th root of unity $\beta$ in $\mathrm{GF}(q^{nk})$ and any primitive $k$-th root of unity $\eta$ in $\mathbb{Z}_{nk+1}$, the element*

$$\alpha = \sum_{i=0}^{k-1} \beta^{\eta^i},$$

*generates a normal basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$ with complexity at most $kn - 1$ if $k \equiv 0 \pmod p$, otherwise with complexity at most $(k + 1)n - k$.*

*Proof.* Note that $k \mid nk$ and by Fermat's Little Theorem $nk + 1 \mid q^{nk+1-1}$, thus $\eta$ and $\beta$ always exist.

Let us first show that $\alpha \in \mathrm{GF}(q^n)$. Using Fermat's Little Theorem, we have that $q^{nk} \equiv 1 \pmod{nk + 1}$. Therefore, $(q^n)^k \equiv 1 \pmod{nk + 1}$, and $q^n$ is a $k$-th root of unity in $\mathbb{Z}_{nk+1}$. Thus, as $\eta$ generates $H_k$, there exists $l \in \mathbb{N}$ such that $q^n = \eta^l$. Then using Theorem 1.5 and Lemma 1.25, we have

$$\alpha^{q^n} = \left( \sum_{i=0}^{k-1} \beta^{\eta^i} \right)^{q^n} = \sum_{i=0}^{k-1} \beta^{\eta^i q^n} = \sum_{i=0}^{k-1} \beta^{\eta^i \eta^l} = \sum_{i=0}^{k-1} \beta^{\eta^{i+l}} = \sum_{i=0}^{k-1} \beta^{\eta^i} = \alpha.$$

As $\alpha = \alpha^{q^n}$, we have $\alpha \in \mathrm{GF}(q^n)$ by Lemma 1.15.

Next, we need to prove that $\alpha, \alpha^q, \ldots, \alpha^{q-1}$ are linearly independent over $\mathrm{GF}(q)$. For that purpose suppose that

$$\sum_{i=0}^{n-1} \lambda_i \alpha^{q^i} = \sum_{i=0}^{n-1} \lambda_i \sum_{j=0}^{k-1} \beta^{\eta^j q^i} = 0, \ \lambda_i \in \mathrm{GF}(q).$$

For $\alpha, \alpha^q, \ldots, \alpha^{q-1}$ to be linearly independent, we must have $\lambda_i = 0$ for all $0 \leq i \leq n - 1$.

Now, let $\gamma$ be any $(nk + 1)$-th root of unity in $\mathrm{GF}(q^{nk})$. By Lemma 4.1, $\eta^j q^i$ runs through $\mathbb{Z}^*_{nk+1}$ for $0 \leq i \leq n - 1, \ 0 \leq j \leq k - 1$, thus there exist unique $u_i \in \mathrm{GF}(q)$ for $0 < i \leq kn$, such that the following holds for all $\gamma$:

$$\sum_{i=0}^{n-1} \sum_{j=0}^{k-1} \lambda_i \gamma^{\eta^j q^i} = \sum_{i=0}^{n-1} \sum_{j=0}^{k-1} \lambda_i \gamma^{r(i,j)} = \sum_{m=1}^{nk} u_m \gamma^m = \gamma \sum_{m=0}^{nk-1} u_{m+1} \gamma^m.$$

Note that $u_i$ and $\lambda_j$ $(0 < i, j \leq kn)$ are equal in some order. Now, let us define $f(x) := \sum_{j=0}^{nk-1} u_{j+1} x^j$. Note that again by Lemma 4.1, for all $0 < r \leq nk$ there exist integers $u$ and $v$ such that $r = \eta^i q^v$. As $\beta^r$ is also a $(nk+1)$-th root of unity, by the above, Theorem 1.5, Lemma 1.14 and Lemma 1.25, we can write

$$\begin{aligned}
\beta^r f(\beta^r) &= \beta^r \sum_{j=0}^{nk-1} u_{j+1}(\beta^r)^j = \sum_{i=0}^{n-1}\sum_{j=0}^{k-1} \lambda_i (\beta^r)^{\eta^j q^i} \\
&= \sum_{i=0}^{n-1} \lambda_i \sum_{j=0}^{k-1} (\beta^{\eta^u q^v})^{\eta^j q^i} = \sum_{i=0}^{n-1} \lambda_i \left( \sum_{j=0}^{k-1} \beta^{\eta^{u+j} q^i} \right)^{q^v} \\
&= \left( \sum_{i=0}^{n-1} \lambda_i \sum_{j=0}^{k-1} \beta^{\eta^j q^i} \right)^{q^v} = \left( \sum_{i=0}^{n-1} \lambda_i \alpha^{q^i} \right)^{q^v} = 0.
\end{aligned}$$

Therefore, as $f(\beta^r) = 0$ for all $0 < r \leq nk$, $\beta^r$ is a root of $f(x)$. Thus, as $\beta$ is primitive, we have shown that there are $nk$ distinct roots of $f(x)$. Since $f(x)$ is of degree at most $nk-1$, this is possible only if $f(x) = 0$. Therefore, we have that $u_j = 0$ for all $0 < j \leq nk-1$, and thus $\lambda_i = 0$ for all $0 \leq i \leq n-1$. Hence, $\alpha, \alpha^q, \ldots, \alpha^{q-1}$ are linearly independent over $\mathrm{GF}(q)$ and form a normal basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$.

Last, we need to examine the complexity of the normal basis. Therefore, let us compute the multiplication table of the basis. As to its rows, for $0 \leq i \leq n-1$ we have by Lemma 1.25 that

$$\begin{aligned}
\alpha\alpha^{q^i} &= \left( \sum_{u=0}^{k-1} \beta^{\eta^u} \right) \left( \sum_{v=0}^{k-1} \beta^{\eta^v q^i} \right) = \sum_{u=0}^{k-1}\sum_{v=0}^{k-1} \beta^{\eta^u + \eta^v q^i} \\
&= \sum_{u=0}^{k-1}\sum_{v=0}^{k-1} \beta^{\eta^u(1+\eta^{v-u}q^i)} = \sum_{v=0}^{k-1} \left( \sum_{u=0}^{k-1} \beta^{\eta^u(1+\eta^v q^i)} \right). \qquad (3)
\end{aligned}$$

Now, let us examine the pairs $(v, i)$. There exists a unique pair $(v', i')$ with $0 \leq v' \leq k-1,\ 0 \leq i \leq n-1$ such that $1 + \eta^{v'} q^{i'} \equiv 0 \pmod{nk+1}$. If $(v, i) \neq (v', i')$, then by Lemma 4.1, we have $1 + \eta^v q^i \equiv \eta^w q^j \pmod{nk+1}$, where $0 \leq w \leq k-1,\ 0 \leq j \leq n-1$. Again by Theorem 1.5, Lemma 1.25 and the fact that $\beta$ has $n$ distinct powers, we get

$$\begin{aligned}
\sum_{u=0}^{k-1} \beta^{\eta^u(1+\eta^v q^i)} &= \sum_{u=0}^{k-1} \beta^{\eta^u(\eta^w q^j)} = \sum_{u=0}^{k-1} \beta^{\eta^{u+w} q^j} \\
&= \sum_{u=0}^{k-1} \beta^{\eta^u q^j} = (\sum_{u=0}^{k-1} \beta^{\eta^u})^{q^j} = \alpha^{q^j}.
\end{aligned}$$

On the other hand, if we have $(v, i) = (v', i')$, then

$$\sum_{u=0}^{k-1} \beta^{\eta^u(1+\eta^v q^i)} = k1,$$

which is 0 if $k \equiv 0 \pmod p$. Thus, for all $i \neq i'$, the sum (3) for $\alpha\alpha^{q^i}$ is a sum of at most $k$ basis elements. Therefore, the complexity of the basis is at most $(n-1)k + n = (k+1)n - k$.

If we have $k \equiv 0 \pmod p$, then the sum (3) for $\alpha\alpha^{q^{i'}}$ consists of at most $k-1$ basis elements. Therefore, if $k \equiv 0 \pmod p$, there will be at most $(n-1)k + k - 1 = nk - 1$ non-zero elements.

In conclusion, we have shown that $\alpha, \alpha^q, \ldots, \alpha^{q-1}$ form a normal basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$ with complexity at most $(kn-1)$ if $k \equiv 0 \pmod p$, where $p$ is the characteristic of $GF(q)$, otherwise with complexity at most $(k+1)n - k$. This completes the proof. □

## 4.2 Type I Optimal Normal Bases

In this section we examine a special case of Theorem 4.2 for $k = 1$. Thus, we get a construction of an optimal normal basis for $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$.

**Theorem 4.3.** *Suppose $n+1$ is a prime and $q$ is primitive in $\mathrm{GF}(n+1)$, where $q$ is a prime or a prime power. Then the $n$ non-unit $(n+1)$-th roots of unity in $\mathrm{GF}(q^n)$ are linearly independent and form an optimal normal basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$.*

We call any optimal normal basis of this construction a *type I optimal normal basis*. In the following we examine the multiplication tables for type I optimal normal bases showing that these have complexity at most $2n-1$.

Let $\alpha$ be a primitive $(n+1)$-th root of unity in $\mathrm{GF}(q^n)$, then $\alpha^{n+1} = 1$. Since $q$ is primitive in $\mathrm{GF}(n+1)$, we have

$$N := \{\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}\} = \{\alpha, \alpha^2, \ldots, \alpha^n\}.$$

By Theorem 4.2, $\{\alpha, \alpha^2, \ldots, \alpha^n\}$ is a basis, so $N$ is a normal basis of $\mathrm{GF}(q^n)$ over $\mathrm{GF}(q)$. Because $\alpha \neq 1$ and

$$(\sum_{i=0}^{n} \alpha^i)(\alpha - 1) = \alpha^{n+1} - 1 = 0,$$

$\alpha$ is a root of the polynomial $x^n + \ldots + x + 1$. Note that $\alpha\alpha^i = \alpha^{i+1} \in N$ for $1 \leq i < n$ and $\alpha\alpha^n = 1 = -\mathrm{Tr}_{\mathrm{GF}(q)}(\alpha) = -\sum_{i=1}^{n} \alpha^i$. Therefore, there are $n - 1 + n = 2n - 1$ non-zero terms, and thus $N$ is optimal.

The matrix $T$ corresponding to this optimal normal basis has the following properties: there is exactly one $1$ in each row, except for one row, where all the $n$ entries are equal to $-1$; all other entries are zeros.

As we are interested in finding normal bases for binary fields, using Theorem 4.3 we get the following sufficient result for the existence of an optimal normal basis of $\mathrm{GF}(2^n)$ over $\mathrm{GF}(2)$ :

Optimal normal basis of type I exists for a given binary field $\mathrm{GF}(2^n)$ if:

1. $n + 1$ is a prime,
2. $2$ is a primitive root modulo $n + 1$.

## 4.3 Type II Optimal Normal Bases

Again we examine a special case of Theorem 4.2, where $k = 2$ and $q = 2$. Thus, we get a construction for an optimal normal basis of $\mathrm{GF}(2^n)$ over $\mathrm{GF}(2)$.

**Theorem 4.4.** *Let* $2n + 1$ *be a prime and assume either*

1. $2$ *is primitive in* $\mathrm{GF}(2n + 1)$ *i.e.* $2$ *is a primitive root modulo* $2n + 1$*, or*
2. $2n+1 \equiv 3 \pmod 4$ *and* $2$ *generates the quadratic residues modulo* $2n+1$.

*Then* $\alpha = \gamma + \gamma^{-1}$ *generates an optimal normal basis of* $GF(2^n)$ *over* $GF(2)$*, where* $\gamma$ *is a primitive* $(2n + 1)$*-th root of unity in* $\mathrm{GF}(2^{2n})$.

Note that condition 2. is equivalent to $n$ being odd and $2$ generating the quadratic residues modulo $2n + 1$.

We call any optimal normal basis obtained via this construction a *type II optimal normal basis*. In Theorem 4.2, it was proved that $\alpha \in GF(2^n)$ and that $\alpha, \alpha^2, ..., \alpha^{2^{n-1}}$ are linearly independent over $GF(2)$. Thus $N = \{\alpha, \alpha^2, ..., \alpha^{2^{n-1}}\}$ is a normal basis of $GF(2^n)$ over $GF(2)$. If condition 1. holds, then

$$\{2, 2^2, 2^3, \ldots, 2^{2n-1}, 2^{2n}\} \equiv \{1, 2, 3, \ldots, 2n\} \pmod{2n + 1}.$$

Therefore, we can write $\gamma^{2^i} + \gamma^{-2^i} = \gamma^j + \gamma^{-j}$ for all $1 \leq j \leq 2n$. Furthermore, if $n+1 \leq j \leq 2n$, then $\gamma^j + \gamma^{-j} = \gamma^{(2n+1)-j} + \gamma^{-(2n+1)+j}$ and $1 \leq 2n+1-j \leq n$.

By condition 2., $\{2, 2^2, 2^3, \ldots, 2^{n-1}, 2^n\}$ is the set of quadratic residues modulo $2n + 1$. Since $2n + 1 \equiv 3 \pmod 4$, we have that $\left(\frac{-1}{2n+1}\right) = -1$ and $\{-2, -2^2, -2^3, \ldots, -2^{n-1}, -2^n\}$ is the set of quadratic non-residues. Therefore, $1, 2, \ldots, 2n$ are congruent to $2, 2^2, \ldots, 2^n, -2, -2^2, \ldots, -2^n$ modulo $2n + 1$ in some order.

Observing that if $2^i \pmod{2n + 1}$ is in the range of $n + 1$ to $2n$, then $-2^i \pmod{2n+1}$ is in the range of 1 to $n$, we can always write $\gamma^{2^i} + \gamma^{-2^i} = \gamma^j + \gamma^{-j}$ for all $1 \leq j \leq n$.

So in both cases $\alpha^{2^i} = \gamma^j + \gamma^{-j}$ for some $1 \leq j \leq n$. Thus,

$$N = \{\gamma + \gamma^{-1}, \gamma^2 + \gamma^{-2}, ..., \gamma^n + \gamma^{-n}\}.$$

The cross-product terms are

$$\alpha(\gamma^i + \gamma^{-i}) = (\gamma + \gamma^{-1})(\gamma^i + \gamma^{-i}) = (\gamma^{(1+i)} + \gamma^{-(1+i)}) + (\gamma^{(1-i)} + \gamma^{-(1-i)}),$$

which is a sum of two distinct elements in $N$ except when $i = 1$. In that case, sum is just $\alpha^2$ which is in $N$. Thus, $c_N = 2(n - 1) + 1 = 2n - 1$ and $N$ is an optimal normal basis of $\mathrm{GF}(2^n)$ over $\mathrm{GF}(2)$.

The matrix $T$ corresponding to this optimal normal basis has the following properties: there are exactly two 1's in each row, except for the first row in which there is exactly one 1; all the other entries are zeros.

# 5 On Irreducible Polynomials

In this chapter we present some criteria for testing the irreducibility of polynomials.

## 5.1 Rabin's Irreducibility Test

We begin with a result due to Rabin, which forms the basis of an irreducibility-testing algorithm named after him.

**Theorem 5.1.** *Let $p(x)$ be a polynomial of degree $n$, and let $r_1, r_2, \ldots, r_t$ be the distinct prime divisors of $n$. Then $p(x)$ is irreducible over $\mathrm{GF}(q)$ if and only if*

1. $p(x) \mid x^{q^n} - x$,

2. $\gcd(x^{q^{\frac{n}{r_i}}} - x, p(x)) = 1$ *for all* $i \in \{1, \ldots, t\}$.

*Proof.* Let us first prove necessity. Assume that $p(x)$ is irreducible over $\mathrm{GF}(q)$. By Corollary 1.24 the splitting field of $p(x)$ is given by $\mathrm{GF}(q^n)$. Thus by Lemma 1.20 now implies $p(x) \mid x^{q^n} - x$, that is, condition 1. holds.

For condition 2., assume that there exists such an $i$ that

$$r(x) := \gcd(x^{q^{\frac{n}{r_i}}} - x, p(x)) \neq 1.$$

Thus, $r(x) \mid x^{q^{\frac{n}{r_i}}} - x$ and $r(x) \mid p(x)$ for some $i$. As $p(x)$ is irreducible, we have that $r(x) = \pm p(x)$, which implies $p(x) \mid x^{q^{\frac{n}{r_i}}} - x$. By Lemma 1.22, $n \mid \frac{n}{r_i}$ for some $i$ and we arrive at a contradiction since $\frac{n}{r_i} < n$. Hence, condition 2. holds.

For sufficiency, assume that both conditions hold. Note that $\mathbb{F}_q[x]$ is factorial, so $p(x)$ can be written as a product of irreducible polynomials. From condition 1., it follows by Lemma 1.20 that all the roots of $p(x)$ are in $\mathrm{GF}(q^n)$. Assume that there exists an irreducible factor $p_1(x)$ of $p(x)$ of degree $m < n$. As $p_1(x)$ is irreducible, by Lemma 1.22, we have $m|n$.

Therefore, since $m < n$, we have that $m|\frac{n}{r_i}$ for one of the divisors $r_i$, and all the roots of $p_1(x)$ lie in $\mathrm{GF}(q^{\frac{n}{r_i}})$). But then $p_1(x) \mid \gcd(x^{q^{\frac{n}{r_i}}} - x, p(x))$ and this contradicts condition 2. Thus, $p(x)$ must indeed be irreducible and sufficiency is proven. This completes the proof. □

Rabin's irreducibility test is presented as Algorithm 7.

**Algorithm 7** Rabin's Irreducibility Test

```
Input: $f(x) = \sum_{i=0}^{n} a_i x^i$, where $a_n = 1$ and $r_1, r_2, \ldots, r_t$ all distinct prime divisors of $n$
Output: true if $f(x)$ is irreducible, otherwise false
for all $i = 1 \ldots t$ do
    $n_i \leftarrow \frac{n}{r_i}$
for all $j = 1 \ldots t$ do
    $g(x) \leftarrow \gcd(f(x), x^{q^{n_i}} - x \pmod{f(x)})$
    if $g(x) \neq 1$ then return false
$g(x) \leftarrow (x^{q^n} - x \pmod{f(x)}$
if $g(x) = 0$ then return true
return false
```

## 5.2 Specific Trinomials

As described in Chapter 2, when using representation via polynomial bases for implementing finite field multiplication, the reduction step is done modulo an irreducible polynomial. In order to accelerate the reduction process, trinomials and pentanomials are used. Recall that a trinomial is a polynomial with three and a pentanomial with five nonzero terms, one of them being the constant term.

**Theorem 5.2.** *Let $a \in \mathrm{GF}(q)$, where $q$ is a prime power $p^n$. The trinomial*

$$x^p - x - a$$

*is irreducible over $\mathrm{GF}(q)$ if and only if $\mathrm{Tr}_{\mathrm{GF}(q)}(a) \neq 0$.*

*Proof.* Let us start with proving necessity. Take $\alpha$ to be a root of $x^p - x - a$. It follows that

$$\begin{aligned}
\alpha^p &= \alpha + a, \\
\alpha^{p^2} &= (\alpha + a)^p = \alpha^p + a^p = \alpha + a + a^p, \\
\alpha^{p^3} &= (\alpha + a + a^p)^p = \alpha^p + a^p + a^{p^2} = \alpha + a + a^p + a^{p^2}, \\
&\cdots \\
\alpha^{p^n} &= (\alpha + a + a^p + \cdots + a^{p^{n-2}})^p \\
&= \alpha^p + a^p + a^{p^2} + \cdots + a^{p^{n-1}} = \alpha + \mathrm{Tr}_{\mathrm{GF}(q)}(a).
\end{aligned}$$

Thus, $\mathrm{Tr}_{\mathrm{GF}(q)}(a) = 0$ if and only if $\alpha^q = \alpha$. In other words, $\mathrm{Tr}_{\mathrm{GF}(q)}(a) = 0$ if and only if $\alpha \in \mathrm{GF}(q)$. Note that $\alpha$ is an arbitrary root of $x^p - x - a$, thus every root of $x^p - x - a$ is in $\mathrm{GF}(q)$ in this case. Therefore, the splitting field of $x^p - x - a$ is $\mathrm{GF}(q)$ if and only if $\mathrm{Tr}_{\mathrm{GF}(q)}(a) = 0$. Since $p(x)$ is irreducible over $\mathrm{GF}(q)$, $\mathrm{Tr}_{\mathrm{GF}(q)}(a) \neq 0$, which gives us necessity.

For sufficiency, let $\eta := \mathrm{Tr}_{\mathrm{GF}(q)}(a) \neq 0$. Since the trace of an element always lies in the base field, $\eta \in \mathrm{GF}(q)$. Thus by the above and Lemma 1.14, we can write

$$\alpha^{q^i} = (\alpha^q)^{q^i} = (\alpha + \eta)^{q^{i-1}} = \alpha^{q^{i-1}} + \eta = \ldots = \alpha + i\eta,\ i \in \mathbb{N}.$$

Since $\mathrm{GF}(q)$ is of characteristic $p$, we have that $\alpha + \eta = \alpha + (p+1)\eta$. Therefore, $\alpha$ has exactly $p$ distinct conjugates over $\mathrm{GF}(q)$. Recall that the minimal polynomial of $\alpha$ over $\mathrm{GF}(q)$ is the unique monic polynomial of least degree among all polynomials over $\mathrm{GF}(q)$ having $\alpha$ as a root. Suppose that the minimal polynomial

$$m(x) = m_d x^d + \ldots + m_1 d + m_0$$

of $\alpha$ is of degree $d$. Then by Theorem 1.5 and Lemma 1.14,

$$\begin{aligned} m(\alpha^q) &= m_d \alpha^{qd} + \ldots + m_1 \alpha^q + m_0 = m_d^q \alpha^{qd} + \ldots + m_1^q \alpha^q + m_0^q \\ &= (m_d \alpha^d + \ldots + m_1 \alpha + m_0)^q = m(\alpha)^q = 0. \end{aligned}$$

Therefore, $\alpha, \alpha^q, \ldots, \alpha^{q^{p-1}}$ are $p$ distinct roots of $m(x)$ and we have that $d \geq p$. As $x^p - x - a$ is a monic polynomial of degree $p \leq d$ having $\alpha$ as a root, we have that $m(x)$ must be equal to $x^p - x - a$. Since minimal polynomials are irreducible over the base field $\mathrm{GF}(q)$, so is $x^p - x - a$. This completes the proof. □

# References

[1] D.M. Burton, Elementary Number Theory. Allyn and Bacon, Boston, 1980.

[2] S. Gao, D. Panario. *Tests and constructions of irreducible polynomials over finite fields*. In: Foundations of Computational Mathematics, 346-361, Springer, Berlin, 1997.

[3] J. Guajardo, S.S. Kumar, C. Paar, J. Pelzl. *Efficient software-implementation of finite fields with applications to cryptography*. Acta Appl. Math. vol 93(1), 3-32, 2006.

[4] D. Hankerson, A. Menezes, S. Vanstone. Guide to Elliptic Curve Cryptography. Springer, New York, 2004.

[5] Ç.K. Koç, T. Acar. *Montgomery multiplication in $GF(2^k)$*. Des. Codes Cryptogr. vol 14(1), 57-69, 1998.

[6] S. Kwon, K. Gaj, C.H. Kim, C.P. Hong. *Efficient linear array multiplication in $GF(2^m)$ using normal basis for elliptic curve cryptography*. In: CHES 2004. LNCS vol 3156, 76-91, Springer, Heidelberg, 2004.

[7] R. Lidl, H. Niederreiter. Finite Fields. Cambridge University Press, Cambridge, 1997.

[8] A.J. Menezes (ed.), I.F. Blake, X. Gao, R.C. Mullin, S.A. Vanstone, T. Yaghoobian. Applications of Finite Fields. Kluwer, Boston, 1993.

[9] R.A. Mollin. Fundamental Number Theory with Applications, Second Eddition. Chapman & Hall/CRC, Boca Raton, 2008.

[10] M.O. Rabin. *Probabilistic algorithms in finite fields*. SIAM J. Comput. vol 9(2), 273-280, 1979.

[11] A. Reyhani-Masoleh, M.A. Hasan. *Low complexity sequential normal basis multipliers over* $\mathrm{GF}(2^n)$. In: Proceeding of the 16th IEEE Symposium on Computer Arithmetic, 188-195, 2003.

## Non-exclusive licence to reproduce thesis and make thesis public

I, Annabell Kuldmaa (date of birth: 17.07.1993),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   ”Efficient Multiplication in Binary Fields”, supervised by Lauri Tart,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 05.06.2015