ANALYSIS AND IMPLEMENTATION OF BINARY POLYNOMIAL MULTIPLICATION

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

BY

MARY ACHIENG OLUDO

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

SEPTEMBER 2021

Approval of the thesis:

ANALYSIS AND IMPLEMENTATION OF BINARY POLYNOMIAL MULTIPLICATION

submitted by **MARY ACHIENG OLUDO** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Selçuk-Kestel _____
Director, Graduate School of **Applied Mathematics**

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

Assoc. Prof. Dr. Murat Cenk _____
Supervisor, **Cryptography, METU**

Examining Committee Members:

Assoc. Prof. Dr. Oğuz Yayla
Cryptography, METU _____

Assoc. Prof. Dr. Murat Cenk
Cryptography, METU _____

Assoc. Prof. Dr. Fatih Sulak
Mathematics, Atılım University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MARY ACHIENG OLUDO

Signature :

ABSTRACT

ANALYSIS AND IMPLEMENTATION OF BINARY POLYNOMIAL MULTIPLICATION

OLUDO, Mary Achieng
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Murat Cenk

September 2021, 47 pages

N. Koblitz and V. Miller originally proposed the concept of elliptic curve cryptography in 1985. It is fast gaining popularity in public key cryptosystems as it boasts of an advantage over current public key cryptosystems, that is, the requirement of the key size be smaller and still maintain the same level of security. It takes advantage of elliptic curves' mathematical properties in finite fields. Elliptic curve cryptographic systems perform operations on points on elliptic curves. Elliptic curves can be represented over prime fields and can also be represented over binary fields. In the binary field representation, the elements of these finite fields are binary polynomials. Polynomial multiplication is a key operation in binary field elliptic curve cryptographic implementation and hence an area of interest in cryptography. In this thesis,we analyse some polynomial multiplication algorithms over binary fields. For 2-way algorithms we investigate schoolbook, Karatsuba and refined Karatsuba and for 3-way algorithms we investigate schoolbook, Karatsuba and CNH 3-way split algorithms. We define a 64-bit by 64-bit multiplication as the smallest unit of computation for a polynomial multiplication which can also be improved by the sliding window method. We have observed that an optimal size of eight for the window size is attained with a time memory trade off.

Keywords: Elliptic Curves, Karatsuba, CNH 3-Way Split

ÖZ

İKİLİK TABANDA POLİNOMLARIN ÇARPIMININ ANALİZİ VE UYGULANMASI

OLUDO, Mary Achieng
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Murat Cenk

Eylül 2021, 47 sayfa

N. Koblitz ve V. Miller ilk olarak 1985 yılında eliptik eğri kriptografisi kavramını önerdiler. Eliptik eğri kriptografi, aynı güvenlik seviyesi için daha küçük anahtar boyutu kullanarak mevcut açık anahtar şifreleme sistemlerine göre avantaj sağladığı için açık anahtarlı şifreleme sistemleri arasında hızla popülerlik kazanmaktadır. Bu sistem, sonlu cisimlerde eliptik eğrilerin matematiksel özelliklerinden yararlanır. Eliptik eğri şifreleme sistemleri, eliptik eğriler üzerindeki noktalar üzerinde işlem gerçekleştirir. Eliptik eğriler, asal cisimler üzerinde temsil edilebilir ve ayrıca ikili cisimler üzerinde de temsil edilebilir. İkili cisim gösteriminde, sonlu cisim elemanları ikili cisimler üzerindeki polinomlardır. Polinom çarpımı, ikili cisim eliptik eğri kriptografik uygulamasında önemli bir işlemdir ve bu nedenle kriptografide bir ilgi alanıdır. Bu tezde, ikili cisimler üzerinde bazı polinom çarpma algoritmalarını analiz ediyoruz. 2-yollu algoritmalar için okul kitabı, Karatsuba ve rafine Karatsuba ve 3-yollu algoritmalar için okul kitabı, Karatsuba ve CNH 3-yollu bölünmüş algoritmaları inceliyoruz. Bir polinom çarpımı için en küçük hesaplama birimi olarak 64 bit'e 64 bit çarpmayı tanımlıyoruz ve bunu da kayan pencere yöntemiyle inceliyoruz. Bir zaman belleği değiş tokuşu yaklaşımı kullanarak en uygun pencere boyutunun sekiz olduğunu gözlemledik.

Anahtar Kelimeler: Eliptik eğriler, Karatsuba, CNH 3-yollu bölünmüş

To Muffin and Steff

ACKNOWLEDGMENTS

I am very grateful to my thesis supervisor, Assoc. Prof. Dr. Murat Cenk for guiding me and giving me constant feedback throughout the process of writing this thesis. I sincerely appreciate his supervision and kind support throughout my study period and also his continuous words of encouragement.

I also want to thank my sister, Steffany Oludo for giving me unwavering support to push this through and to continue even when I was not motivated to do so. Without her love and support, this thesis would not have been made possible.

And finally to my cat, Muffin, for those late nights and early mornings that he kept me company.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF ABBREVIATIONS

ECC	Elliptic Curve Cryptography
AES	Advanced Encryption Standard
DES	Data Encryption Standard
RSA	Rivest Shamir Adleman Cryptosystem
ECDLP	Elliptic Curve Discrete Logarithm Problem
KA	Karatsuba 2-Way Algorithm
SB	Schoolbook 2-Way Algorithm
RK	Refined Karatsuba 2-Way Algorithm
SB3W	Schoolbook 3-Way Algorithm
K3W	Karatsuba 3-Way Algorithm
CNH3W	Cenk Negre Hasan 3-Way Algorithm
NIST	National Institute of Standards and Technology

CHAPTER 1

INTRODUCTION

Cryptography involves transforming data using established algorithms with the purpose of securing it. It is very important in this digital as age as information is primarily exchanged electronically. It is therefore essential to protect data from being accessed by malicious parties that the data was not intended to reach. According to [19], there are four key objectives that are to be achieved in a cryptographically secure system. Namely:

- Confidentiality: None other than the parties that are authorized to obtain the data have access to it.
- Data integrity: Makes sure that the exchanged data is not tampered with.
- Authentication: Only genuine data and parties are involved in the information exchange.
- Non-repudiation: Prevents the involved parties from retracting their previous actions.

Cryptography can be broadly divided into three main areas of interest:

1. Symmetric key cryptography whereby the participants share the same key, encryption and decryption methods.
2. Public key cryptosystems that use different keys for each participating party.
3. Cryptographic protocols which entail the standardized algorithms, how to use and apply both symmetric and public key cryptography.

1.1 Symmetric Key Cryptography

This is the branch of cryptography where both parties use an identical key for the encryption and the decryption of data. The key is initially exchanged using a secure method to prevent unauthorized parties from gaining access to it. This key exchange is usually done using public key algorithms which will be discussed more in the next section. Examples of symmetric key cryptographic schemes include AES [24] and DES [7] and its variants.

1.2 Asymmetric Key Cryptography

Public/Asymmetric key cryptography first came to light in the 1970s after a proposal by Diffie and Hellman [8]. It is different from symmetric key cryptography in that it uses different keys for the different parties involved in the communication. It is a very crucial branch of cryptography as it has vast applications in confidentiality where information encrypted by a party's public key can only be decrypted by that party's secret key. It can also be useful during authentication of data as done with digital signature algorithms.

Definition 1.2.1. *Public key cryptography is an encryption scheme that can be described by the following spaces and algorithms all depending on a security parameter* $k \in \mathbb{N}$ *[9]*
M_k *: The message space*
PK_k*: The public key space*
SK_k*: The secret key space*
C_k*: The ciphertext space*
For an $m \in M_k$*,* $pk \in PK_k$*,* $sk \in SK_k$ *and* $c \in C_k$ *it is required that:*

$$Decrypt(Encrypt(m, pk)sk) = m$$

Where there exists an algorithm that generates keys that produces the public key and private key with the security parameter, k, an encrypt algorithm accepts in input m and public key and outputs c, the ciphertext and a decrypt algorithm that takes c and sk and outputs m.

1.3 Security in Asymmetric Key Cryptography

The security of asymmetric key cryptographic systems depends on three main features. Namely:

1. An attacker cannot evaluate m from c.

2. Other than the length, an attacker cannot deduce any other property of m from c.

3. An attacker cannot distinguish between the encryption of any pair of different m_0 and m_1 both of equal length.

To ensure the security of the data as is required by the three features mentioned above special mathematical functions referred to as one-way/trapdoor functions have been used.

1.4 Trapdoor Functions

One-way functions also known as trapdoor functions are mathematically challenging functions which are simple to evaluate but the inverse is considered infeasible to evaluate without knowledge of specific information about the function [19]. That is, if f is a trapdoor/one-way function with some hidden information y, given $f(x)$, x can only be computed when y is known.

Examples of such problems in cryptography are the prime factorization problem used by RSA which is based on how difficult it is to factorise integers to their product of primes and another, the discrete logarithm problem either in a group defined over an elliptic curve or modulo a prime, specifically, for a group G, that has generator g and an element h in G, evaluate the discrete logarithm of h, to the base g. Because of this second example, Elliptic curves become a key area of interest in cryptography.

1.5 Aim of This Thesis

The main aim of this thesis is to discuss the performance of binary polynomial multiplication algorithms in software and present some optimisation techniques that improve their performance. This is important for cryptographic applications especially in binary field elliptic curve cryptosystems whereby polynomial multiplication is a key underlying arithmetic operation that is useful when performing operations on points on an elliptic curve.

In this thesis, we have analysed the schoolbook, Karatsuba-like, CNH, and windows methods for binary polynomial multiplication. After introducing the algorithms and their complexities, we have implemented them using combination of algorithms so that better results are achieved. In the base case, we have used the sliding window method for 64 bit sizes. To improve the results, we have optimal windows size has been searched and we found that the size 8 is a good choice with the reasonable amount of memory.

In the next chapter, we shall give some mathematical background and Galois field concepts. In chapter 3, we discuss elliptic curves, and their application in Cryptography. Chapter 4 shall discuss some polynomial multiplication algorithms. In chapter 5, we analyse the software implementation of the algorithms discussed in chapter 4. In chapter 6 will give a conclusion to this thesis.

CHAPTER 2

GROUPS, RINGS AND GALOIS FIELDS

Galois theory, that studies groups and finite fields was made popular by Evariste Galois [1811-1832].In the next sections, we briefly discuss concepts in Galois theory as they shall be used in the remainder of this thesis to perform mathematical operations on points on Elliptic curves. For a more comprehensive study, we suggest reading [26] [17] [3] [18] [5] [21].

2.1 Groups

A group can be defined as $G \neq \emptyset$, a non-empty set with an operation, $*$, on group G which satisfies:

- G is closed under $*$. That is, $a * b$ is in group G, for a, b which are elements in G.
- G is associative. Whereby $a * (b * c) = (a * b) * c = a * b * c$, for a, b, c which are elements in G.
- G has identity element, e with respect to $*$. That is, $\exists e \in G$ whereby $a * e = a = e * a$, for all a in G
- All elements in G have an inverse. For all a in G, $\exists a^{-1}$ also in G such that $a * a^{-1} = e$. The element a^{-1} is referred to as the inverse of a in G with respect to $*$

If the above four requirements are fulfilled, the pair $\{G, *\}$ is referred to as a group.

If also, the $*$ operation is commutative, such that is $a * b = b * a$, for a, b which are elements in G, then G is referred to as a commutative group.

2.1.1 Cyclic Groups

A group is cyclic when it is generated using one element in it. $G = <\alpha>$ for some element $\alpha \in G$ called the generator.

2.1.2 Subgroups

If there exists a G with $\{G, *\}$ and there is a subset K in G such that $\emptyset \neq K \subset G$. If $\{K, *\}$ is also a group under the same operation as G, that is $\forall a, b \in K$
1. $a * b \in K$
2. $\forall a \in K$, $a^{-1} \in K$
3. $a * e = a = e * a$, $\forall a \in K$
Then K is called a subgroup of group G, denoted by $K \leq G$. If $\forall k \in K$,$\forall g \in G$ and $g^{-1}kg \in K$ that is, the conjugate of k by g is invariant, then K is referred to as a normal subgroup of group G denoted by $K \triangleleft G$

2.1.3 Cyclic Subgroups

Suppose G, a group with an element $a \in G$. Therefore $<\alpha> = \{a^n | n \in \mathbb{Z}\}$ is a subgroup of group G. And <a> is a cyclic subgroup that is generated by element a.

2.1.4 Homomorphism and Isomorphism

If a group $\{G, *\}$ and $\{H, \circ\}$ are two groups, and if $f : G \to H$ is a function from group G to H. If function f preserves the operation $*$ and $\circ$, that is, for every $a, b \in G$, then $f(a * b) = f(a) \circ f(b)$ then f is called a group homomorphism.

$f : G \to H$ is injective for any two elements $a, b \in G$, then $f(a) = f(b)$ if and only if $a = b$. If f is injective then it is called a monomorphism.

$\forall y \in H$, there exists element $x \in G$ where $y = f(x)$ If the function f is surjective, then it is a group isomorphism.

If the function $f : G \to G$ is a group isomorphism, then f is called a group automorphism.

2.2 Rings

A ring can be defined as an non-empty set that has two operations, namely multiplication and addition and satisfies:

- $\{R, +\}$ is an additive commutative group
- It is closed under the multiplication operation. That is, if $a \in R$ and if $b \in R$ then $a * b \in R$
- It is associative. That is $a * (b * c) = (a * b) * c = a * b * c, \forall a, b, c \in R$

If $a * b = b * a$, $\forall a, b \in R$ then R is a commutative ring. If R has an identity element(unity) with respect to multiplication, then R is a ring with unity. If R is commutative and with unity, then R is called a commutative ring with unity.

2.2.1 Zero Divisors and Integral Domains

If $\{R, +, *\}$ is a ring and if a, b is in R and a is not equal to 0, b is not equal to 0 but $a * b = ab = 0$, then b is a right zero divisor and a is a left zero divisor in R.

An integral domain is defined as a commutative ring that has no zero divisors. In a ring with unity, a zero divisor does not have a multiplicative inverse.

2.2.2 Ideals and Quotient Rings

If I be a subgroup of R, that is $(I, +) \leq (R, +)$ If $rx \in I$, $\forall r \in R$, $\forall x \in I$ then $(I, +, *)$ is the left ideal of R.

If $xr \in I$, $\forall r \in R, \forall x \in I$ then $(I, +, *)$ is the right ideal of R.
If I is both a right ideal and also a left ideal of R then I is referred to as an ideal of R.
If R is a commutative ring, each ideal is two-sided. A quotient or factor ring of R by I is the set of equivalence classes of R with respect to I.

2.2.3 Units

An element a of R, a ring with unity is called a unit in R. That is, if a has a multiplicative inverse in R, there is an $a^{-1} \in R$ whereby $a * a^{-1} = 1_R = a^{-1} * a$. In a ring R, with unity, the units forms a group under multiplication.

2.2.4 Equivalence Class

Let I be a ideal with two sides(left and right ideal) of R. If a, b are in R then a is equivalent/congruent to b relative to I $(mod\ I)$ if $(b - a) \in I$.Then $\sim$ is the equivalence relation R when the following properties, namely, reflexivity, symmetry and transivity are fulfilled.

If A be a set and R is an equivalence relation on A. For any element $a \in A$, the equivalence class can be defined as $[a] = \{x \in A : x \sim A\}$. Then:

- For any element $a \in A$, $a \in [a]$
- For any two elements $a, b \in A$, $a \sim b \iff [a] = [b]$
- For any two elements $a, b \in A$, either $[a] \cap [b] = \emptyset$ or $[a] = [b]$

For the equivalence class S, any element $a \in S$ can be used to represent S guaranteeing $[a] = S$.

2.2.5 Characteristic of a Ring

Let $(R, +, \circ)$ be a ring. If there is a positive integer n, whereby $nx = x + x + x + \cdots + x = 0$, n-times the minimum such times, n, is the characteristic of the ring R.

If no such positive integer exists where $nx = 0 \; \forall x \in R$ then R has characteristic 0. An integral domain has characteristic 0 or a prime number.

2.3 Fields

A field, $(F, +, *)$, can be defined as an integral domain in which all non-zero elements have a multiplicative inverse. It has to have the following properties:

- $(F, +)$ is a commutative group.
- $(F^*, *)$ is a commutative group.
- The $*$ operation is distributative over $+$.

2.3.1 Galois Fields

These are fields that have a finite number of elements, p^n, whereby p is a prime number and n is in $\mathbb{Z}^+$.A finite field has characteristic of a prime number greater than 0. Examples of Galois fields include prime fields represented as $GF(p)$ and binary extension fields represented as $GF(2^n)$.

- Prime Fields: Let's take a number $n \in \mathbb{N}^+$. If $n = p$ where p is prime $\mathbb{Z}/p\mathbb{Z} = \tilde{0}, \tilde{1}, \tilde{2}, \ldots, p \tilde{-} 1$ is an integral domain that is finite so $\mathbb{Z}/p\mathbb{Z}$ is a field also called a prime field.
- Binary Extension Fields: These are Galois fields with 2^n elements whereby each element is a polynomial that has degree less than n and the coefficients of these elements are in $GF(2)$.

2.3.2 Finite Field Bases

Galois fields can be represented in many different ways. A basis for a field p^n relative to a prime p can be defined as a set with n elements of the field with p^n elements which are linearly independent with respect to p. The two most common bases for finite field representation are polynomial bases and normal bases.

2.3.2.1 Normal Bases

A normal basis for a field with p^n elements contains n conjugates of the primitive root α of the field. That is the set containing $\{\alpha, \alpha^p, \alpha^{p^2}, \ldots, \alpha^{p^{n-1}}\}$. This forms an n dimensional vector space. The primitive root α is normal over the field $\mathbb{F}_p$. If we denote $\alpha_{i's}$ as $\alpha_i = \alpha^{p^i}$ where α is in $\mathbb{F}_{p^n}$ and $i = 0, 1, \ldots, n-1$. Then the normal basis becomes the set $\{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{n-1}\}$ and $\alpha_i\alpha_j$ is a linear combination of the conjugates of the primitive root whose coefficients are in $\mathbb{F}_p$ and $0 \leq i, j \leq n$. If there is an n by n matrix, B, whose elements are in $\mathbb{F}_p$ as in the equation below, we can define B as the multiplication table of this basis. The number of non-zero elements in the multiplication table can be referred to as the density, δ. Therefore multiplication of two elements in $\mathbb{F}_{p^n}$ is done with at most $2n\delta$ multiplications in $\mathbb{F}_p$.

2.3.2.2 Polynomial Bases

A polynomial basis for a field with p^n elements contains the set $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$ whereby α is the primitive root. This also forms an n dimensional vector space. A Galois field element in the polynomial basis representation can be formed by a linear combination of the elements in the set, that is, $\sum_{k=0}^{n} a_k \cdot \alpha^k$ where a_k is in $\mathbb{F}_p$. Multiplication of any two elements in $\mathbb{F}_{p^n}$ in the polynomial basis representation is done by multiplying every term of the first element by every term of the second element as shown in equation 2.1.

$$\left(\sum_{k=0}^{n} a_k \cdot \alpha^k\right)\left(\sum_{k=0}^{n} b_k \cdot \alpha^k\right) = \sum_{k=0}^{n-1}\left(\alpha^k \sum_{l+m=n}^{n-1} a_l b_m\right) \tag{2.1}$$

This multiplication requires at most $2n^2$ operations in $\mathbb{F}_p$. Then, the result is reduced modulo $f(x)$ which is the defining polynomial of $\mathbb{F}_{p^n} = \frac{\mathbb{F}_p(x)}{<f(x)>}$.

2.3.3 Polynomials Over a Finite Field

If $\mathbb{F}$ is a finite field, then a polynomial f over field $\mathbb{F}$ is defined as a sum

$$f(\,x) = \sum_{k=0}^{n} a_k \cdot x^k = a_0 + a_1 x^1 + \cdots + a_{n-1}x^{n-1} + a_n x^n \tag{2.2}$$

Where the coefficients $a_0, a_1, \ldots, a_n \in F$ and x is indeterminate. The degree of f is n. $a_n x^n$ is the leading term a_n is the leading coefficient. The polynomial can be described as monic if the leading coefficient is 1. If the polynomial has addition and multiplication operations where addition is commutative, associative and has an identity and have inverses and the multiplication operation is commutative, associative and has an identity and is distributive over the addition operation, then the set of all the polynomials $\mathbb{F}[x]$ over field $\mathbb{F}$ will form a commutative ring. If

$$f(x) = \sum_{k=0}^{n} a_k \cdot x^k = a_0 + a_1 x^1 + \cdots + a_{n-1} x^{n-1} + a_n x^n \tag{2.3}$$

$$g(x) = \sum_{k=0}^{n} b_k \cdot x^k = b_0 + b_1 x^1 + \cdots + b_{n-1} x^{n-1} + b_n x^n \tag{2.4}$$

2.3.3.1 Polynomial Addition

Given polynomial $f(x)$ and $g(x)$, polynomial addition is the summation of the same degree terms of two these two polynomials.

$$f(x) + g(x) = \sum_{k=0}^{Max(m,n)} (a_k + b_k) x^k \tag{2.5}$$

2.3.3.2 Polynomial Multiplication

Given polynomial $f(x)$ and $g(x)$, polynomial multiplication is the multiplication of each term of the first polynomial by each term of the second polynomial.

$$f(x)g(x) = \sum_{k=0}^{n-1} \left(x^k \sum_{l+m=n}^{n-1} a_l b_m \right) \tag{2.6}$$

2.3.3.3 Polynomial Division

If $\mathbb{F}$ is a finite field, if $f(x)$ and $g(x)$ are polynomials, and degree of $g(x)$ is greater or equal to the degree of $f(x)$ we can say that $f(x)$ divides $g(x)$ if $g(x) = f(x)q(x)$ whereby $q(x)$ is also a polynomial in the field.

2.3.3.4 Irreducible Polynomials

If $\mathbb{F}$ is a finite field and polynomial $f(x)$ is monic over field $\mathbb{F}$ with degree $(f) \geq 1$. Then $f(x)$ is called an irreducible polynomial if the only divisors of $f(x)$ that are monic are 1 and itself. If $\mathbb{F}$ is a field and there is an $m(x)$ which is an irreducible polynomial over $\mathbb{F}[x]$. Then $\mathbb{F}[x]/m(x)$ is also a field.

2.3.4 Binary Finite Field Extensions

Binary extension fields, Galois fields with 2^m elements also denoted as $\mathbb{F}_{2^m}$ are finite fields with characteristic of 2. They are represented by binary polynomials. They can be constructed with $f(x)$, which is irreducible over $GF(2)$ of degree m as $\mathbb{F}_{2^m} = \frac{\mathbb{F}_2[x]}{<f(x)>}$ where

$$f(x) = x^m + a_{m-1}x^{m-1} + \cdots + (a_1 x) + 1 \tag{2.7}$$

and $a_i \in GF(2)$. The elements of $\mathbb{F}_{2^m}$ are polynomials which have degree at most $m - 1$.

CHAPTER 3

ELLIPTIC CURVES

Elliptic curve cryptographic systems were first proposed by both Koblitz [16] and Miller independently in 1985 [20]. Since then, they have gained popularity in public key cryptographic systems. In this chapter, we briefly discuss elliptic curves, their applications in cryptography and give examples of elliptic curves used in modern cryptosystems. We begin by giving a background of elliptic curves in Mathematics, that is the basic definition, the group definition and the mathematical operations applied to points that are on an elliptic curve. Namely, point addition, doubling and multiplication operations. We shall then give some examples of elliptic curves recommended for cryptographic purposes.

3.1 Basic Definition

This section explains the basic definition of elliptic curves. For more indepth material, one can refer to [15] [25] [6] [12] [11].

Definition 3.1.1. *An elliptic curve over a Galois field, K, is described as the set containing all points defined by the Weierstrass Equation:*

$$E : y^2 + c_1xy + c_3y = x^3 + c_2x^2 + c_4x + c_6 \tag{3.1}$$

Where c_1, c_2, c_3, c_4, c_6 are all elements in K and whereby the discriminant of E is

defined as follows:

$$\begin{aligned}
\Delta &= -{m_2}^2 m_8 - 8{m_4}^3 - 27{m_6}^2 + 9m_2 m_4 m_6 \\
m_2 &= {c_1}^2 + 4c_2 \\
m_4 &= 2c_4 + c_1 c_3 \\
m_6 &= {c_3}^2 + 4c_6 \\
m_8 &= {c_1}^2 c_6 + 4c_2 c_6 - c_1 c_3 c_4 + c_2 c_3 c_4 + c_2 {c_3}^2 - {c_4}^2
\end{aligned} \tag{3.2}$$

Restricting the discriminant to non-zero values makes sure that the elliptic curve is not singular.

Definition 3.1.2. *For a Galois field K, that has characteristic 2, if* $c_1 = 0$*, and change the variables from* $(x, y) \rightarrow (x + c_2, y)$ *will transform the curve equation to* $y^2 + cy = x^3 + ax^2 + b$ *whereby* $a, b, c \in K$ *and* $c \neq 0$ *such curves are called supersingular and have discriminant* $\Delta = c^4$

Definition 3.1.3. *For a Galois field K with characteristic 2,* $c_1 = 0$*, and a change of variables* $(x, y) \rightarrow ({c_1}^2 x + \frac{c_3}{c_1}, {c_3}^2 y + \frac{{c_1}^2 c_4 + {c_3}^2}{{c_1}^3})$ *the elliptic curve equation transforms to* $y^2 + xy = x^3 + ax^2 + b$ *whereby* $a, b, c \in K$ *and* $b \neq 0$*. These kinds of curves are called non-supersingular and have discriminant* $\Delta = c^4$*.*

Definition 3.1.4. *If K is a Galois field that has characteristic of 2 that is,* $K = GF(2^m)$ *then the elliptic curve over K is*

$$E : y^2 + cy = x^3 + ax + b; for, a = 0, \Delta = c^4 \neq 0 (supersingular curve) \tag{3.3}$$

$$E : y^2 + xy = x^3 + ax^2 + b; for, a \neq 0, \Delta = b \neq 0 (non - supersingular curve) \tag{3.4}$$

3.2 Group Definition

Taking a point at infinity, O, the identity element together with all the points on an elliptic curve are an Abelian group. This group is defined by a binary operator, $\circ$, and upholds the following conditions:

1. Identity: $O + P = P + O = P$, for all P in E/K

2. Negative: If point P is in E/K, then $P + (-P) = O$ and $-P$ is referred to as the negative of P and $-P$ is also in E/K. Also, $-O = O$

3. Addition of points: If P, Q is in E/K where $P \neq \pm Q$. Then there is a point $P + Q = R$ also in E/K

4. Doubling of points: If P is in E/K where $P \neq -P$. Then there is a point $2P = S$ also in E/K

3.2.1 Point Addition

Point addition which is defined as adding two distinct points of an elliptic curve is:

Definition 3.2.1. *Let point P and Q in E/K be taken as two distinct points on the curve. If we draw a straight line that passes through point P and point Q, the third point on the curve E/K at which the line intersects the curve that is the reflection about the x-axis is the point $P + Q$. The formula can be defined algebraically as follows in affine coordinates:*
For a non-supersingular curve, $E/\mathbb{F}_{2^m} : y^2 + xy = x^3 + ax^2 + b$:

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1 \\ where : \lambda &= \frac{(y_1 + y_2)}{(x_1 + x_2)} \end{aligned} \tag{3.5}$$

For the supersingular curve $E/\mathbb{F}_{2^m} : y^2 + cy = x^3 + ax + b$:

$$\begin{aligned} x_3 &= \lambda^2 + x_1 + x_2 \\ y_3 &= \lambda(x_1 + x_3) + y \\ where : \lambda &= \frac{(y_1 + y_2)}{(x_1 + x_2)} \end{aligned} \tag{3.6}$$

3.2.2 Point doubling

The doubling of a point on the curve is defined as:

Definition 3.2.2. *If $P \in E/K$ be a point on the curve. If the curve's tangent at P is drawn, the point on the curve E/K at which the line intersects the curve is the reflection about the x-axis of the point 2P. The formula can be defined algebraically as follows in affine coordinates: For a non-supersingular curve $E/\mathbb{F}_{2^m} : y^2+xy = x^3+ax^2+b$:*

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + a = {x_1}^2 + \frac{b}{{x_1}^2} \\ y_3 &= {x_1}^2 + \lambda x_3 + x_3 \\ where : \lambda &= x_1 + \frac{y_1}{x_1} \end{aligned} \tag{3.7}$$

For the supersingular curve $E/\mathbb{F}_{2^m} : y^2 + cy = x^3 + ax + b$:

$$\begin{aligned} x_3 &= \lambda^2 \\ y_3 &= \lambda(x_1 + x_3) + y_1 + c \\ where : \lambda &= \frac{({x_1}^2 + a)}{c} \end{aligned} \tag{3.8}$$

3.2.3 Scalar Multiplication

Scalar or point multiplication can be described as follows:

Definition 3.2.3. *Let $P \in E/K$ be a point on the curve and let k be an integer, scalar or point multiplication can be defined as the addition of P to itself k times. This results to a point, kP which is also on the curve.*

3.2.4 Elliptic Curves over $GF(p)$

One form of representing elliptic curves is over the Galois field $GF(p)$. This field contains p, number of elements whereby p is prime and whose elements are the integers mod p and arithmetic is done in terms of the arithmetic of integers $mod\ p$. They are the elliptic curves:

$$y^2 = x^3 + ax + b \mod p \tag{3.9}$$

where the discriminant is $4a^3 + 27b^2 \neq 0 \mod p$

$E_p\,(a, b)$ will be a represention of all the points (x, y) that satisfy the conditions above along with point O at infinity where $(x, y \in Zp)$, $(a, b \in Zp)$. The set of points is

the collection of all discrete points on the (x, y) plane. That is the Cartesian product $\mathbb{Z}_p$ x $\mathbb{Z}_p$.

3.2.5 Elliptic Curves Over $GF(2^n)$

Another representation of elliptic curves is over binary extension fields. A binary field $GF(2^n)$ contains 2^n elements where the degree of the field is n and whose elements are strings of bits with length m and their arithmetic can be computed in terms of the operations on this bits. Addition done in $GF(2^n)$ was described as the bitwise XOR operation since a finite field $GF(2^n)$ has characteristic 2. The elliptic curve that is used when $GF(2^n)$ is the underlying field is

$$y^2 + xy = x^3 + ax^2 + b, b \neq 0 \tag{3.10}$$

The discriminant of these curves can be singular even when b is not 0. This changes the behavior of the operator of this group. Given a point $P = (x, y)$ then its additive inverse is $P = (x, -(x + y))$. Given $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ addition of these two points $R = (x_{P+Q}, y_{P+Q})$ is given by

$$x_{P+Q} = \lambda^2 + \lambda - x_P - x_Q - a \tag{3.11}$$

$$y_{P+Q} = -\lambda(x_{P+Q} - x_P) - x_{P+Q} - y_P \tag{3.12}$$

With

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P} \tag{3.13}$$

For point doubling, that is getting $2P$ from P

$$x_{2P} = \lambda^2 + \lambda - a - 2x_P \tag{3.14}$$

$$y_{2P} = -\lambda^2 + \lambda + a + (2 + \lambda)x_p - \lambda x_{2P} - y_P \tag{3.15}$$

With

$$\lambda = \frac{3{x_P}^2 + 2ax_P - y_P}{2y_P + x_P} \tag{3.16}$$

$\lambda' s$ value is obtained as follows:

When $y^2 + xy = x^3 + ax^2 + b$ is differentiated with respect to x on both sides and an expression written for $\frac{dy}{dx}$.

$E_{2^n}(a, b)$ is all the points (x, y) in $GF(2^n)$ that satisfy $y^2+xy = x^3+ax^2+b$ whereby $(a, b) \in GF(2^n)$ and distinguished point O which is the additive identity element, b is not allowed to be the identity with respect to addition, of the finite field $GF(2^n)$. If there is g, a generator for $GF(2^n)$ then the elements of $GF(2^n)$ can be expressed as

$$0, 1, g, g^2, \ldots, g^{2n-2} \tag{3.17}$$

So, most points on an elliptic curve $E_{2^n}(a, b)$ can be expressed as g^i, g^j where $i, j = 0, 1, \ldots, 2^r - 2$ and also the point O. The order of the elliptic curve is the number of points in the group $E_{2^n}(a, b)$

3.3 Elliptic Curve Cryptography

Elliptic curve cryptography makes use of elliptic curves defined over Galois fields. Suppose G is a base point chosen by the user on a curve $E_q(a, b)$ such that $q = p$ for a prime number p when the Galois field has $q = 2^n$ elements then the underlying Galois field is a binary field. With respect to the group's operator then $kG = G + G + \ldots + G(k - times)$.

The core notion of elliptic curve cryptography is that when G is chosen properly, it is very easy to calculate $C = M$ x G but difficult to obtain M from the C obtained even when the eavesdropper knows the G and the curve $E_q(a, b)$ used. This problem is the elliptic curve discrete logarithm problem.

3.3.1 Elliptic Curve Discrete Logarithm Problem

With E, the elliptic curve over a Galois field $\mathbb{F}$ and P a point in $E(\mathbb{F}_q)$ that has order n, and a point $Q \in <P>$ find an integer $k \in [0, n - 1]$ where $Q = kP$. The integer k is referred to as the discrete logarithm of Q, to the base P, denoted by $k = log_p Q$

3.3.2 Elliptic Curve Cryptographic Security

Elliptic curve cryptography relies on the difficulty of the discrete logarithm calculation for large numbers. This is the elliptic curve discrete logarithm problem. Before

using an elliptic curve cryptographic scheme, careful consideration should be taken as the security of elliptic curve cryptographic systems can be compromised by various factors such as:

1. Elliptic curves whereby the sum of points equal the number of elements in the Galois field. These curves are not cryptographically secure.

2. When using binary fields, the Weil descent attack can be performed therefore when using these fields, n has to be prime.

3. In 1993, MOV showed that for elliptic curves that are supersingular, the problem of solving the elliptic curve discrete logarithm problem where $E_q(a, b)$ can be reduced to finding logarithms in a finite field.

4. Another attack that can be performed on elliptic curve cryptosystems is the Frey-Ruck attack which is a generalisation of the MOV attack that applies to curves $E/\mathbb{F}_{p^n}$ with group order $|E(\mathbb{F}_{p^n})|$ coprime to p.

5. The SSSA attack determines the discrete logarithm of some point P with respect to the base point G by reducing the ECDLP of a prime field anomalous curve $E(\mathbb{F}_p)$ to the DLP of the additive group $(\mathbb{Z}_p, +)$ of integers modulo p.

3.4 Binary Curves

Elliptic curves defined over binary extension fields $E/\mathbb{F}_{2^m}$ are represented as non-supersingular curves as follows: $y^2 + xy = x^3 + ax^2 + b$.
There are two main types of curves over $\mathbb{F}_{2^m}$:

1. Pseudo-random curves over $\mathbb{F}_{2^m}$ which have a pseudo-random structure where the parameters are chosen by a specified algorithm.

2. Koblitz curves over $\mathbb{F}_{2^m}$ that are defined over $E/\mathbb{F}_{2^m}$ by equations of form $y^2 + xy = x^3 + ax^2 + 1$ and where $a \in \{0, 1\}$.

Specific parameters have to be defined to generate a curve over $\mathbb{F}_{2^m}$.
The domain parameters for both pseudo-random curves and Koblitz curves are defined as the following:

1. M, defined as the extension of the field $\mathbb{F}_{2^m}$
2. A reduction polynomial $f(z)$ with degree m.
3. Coefficients $a, b \in \mathbb{F}_{2^m}$ of the curve.
4. The order of point P, the base point, that is represented by n.
5. The cofactor that is defined as $h = \#(E/\mathbb{F}_{2^m})/n$ where $\#(E/\mathbb{F}_{2^m})$ is the order of the curve E over $\mathbb{F}_{2^m}$
6. The $(x_p, y_p) \in E/\mathbb{F}_{2^m}$ the coordinates of point P.

A seed is also applied for random generation of the curve's coefficients.

3.5 NIST Curves

NIST recommends some binary curves over fields $\mathbb{F}_{2^m}$ where m is prime to be used for cryptographic purposes. Some NIST recommended elliptic curves are as follows:

3.5.1 Koblitz Curves over $\mathbb{F}_{2^m}$

1. Curve K-163
   This curve is used for legacy purposes only.
2. Curve K-233
   A Weierstrass curve with $m = 233$, $a = 0$, $b = 1$. The order of which is $h \cdot n$ whereby $h = 4$ and n is a prime and a reduction polynomial $f(z) = z^{233} + z^{74} + 1$.
3. Curve K-283
   A Weierstrass curve with $m = 283$, $a = 0$, $b = 1$. The order of which is

$h \cdot n$ whereby $h = 4$ and n is a prime and a reduction polynomial $f(z) = z^{283} + z^{12} + z^7 + z^{12} + 1$.

4. Curve K-409

   A Weierstrass curve with $m = 409$, $a = 0$, $b = 1$. The order of which is $h \cdot n$ whereby $h = 4$ and n is a prime and a reduction polynomial $f(z) = z^{409} + z^8 + 1$.

5. Curve K-571

   A Weierstrass curve with $m = 571$, $a = 0$, $b = 1$. The order of which is $h \cdot n$ whereby $h = 4$ and n is a prime and a reduction polynomial $f(z) = z^{571} + z^{10} + z^5 + z^2 + 1$.

3.5.2 Pseudo-random Curves over $\mathbb{F}_{2^m}$

1. Curve B-163

   This curve used for legacy purposes only.

2. Curve B-233

   A Weierstrass curve with $m = 233$, $a = 1$. The order of which is $h \cdot n$ whereby $h = 2$ and n is a prime and a reduction polynomial $f(z) = z^{233} + z^{74} + 1$.

3. Curve B-283

   A Weierstrass curve with $m = 283$, $a = 1$. The order of which is $h \cdot n$ whereby $h = 2$ and n is a prime and a reduction polynomial $f(z) = z^{283} + z^{12} + z^7 + z^{12} + 1$.

4. Curve B-409

   A Weierstrass curve with $m = 409$, $a = 1$. The order of which is $h \cdot n$ whereby $h = 2$ and n is a prime and a reduction polynomial $f(z) = z^{409} + z^8 + 1$.

5. Curve B-571

   A Weierstrass curve with $m = 571$, $a = 1$. The order of which is $h \cdot n$ whereby $h = 2$ and n is a prime and a reduction polynomial $f(z) = z^{571} + z^{10} + z^5 + z^2 + 1$.

We introduced the concept of elliptic curves in this chapter and how they are applied in cryptography. We then discussed some security parameters to consider when making a choice for an elliptic curve for cryptographic purposes. We finally concluded by giving examples of acceptable binary elliptic curves recommended by NIST. In the next chapter, we shall dive deeper into binary polynomial multiplication algorithms which play a major role in performing operations on binary elliptic curves discussed in section 3.2.

CHAPTER 4

POLYNOMIAL MULTIPLICATION ALGORITHMS

Until the 1960s, the complexity of multiplying two polynomials was $O(n^2)$ using the schoolbook method. Karatsuba and Ofman discovered a faster multiplication algorithm which had a complexity of $O(n^{1.58})$ called the Karatsuba method [14]. Since then, further optimizations have been done to decrease the time complexity hence increasing the speed of computing polynomial multiplications. Some of these multiplication algorithms shall be discussed in this chapter.

4.1 2-Way Algorithms

2-way algorithms divide each polynomial into two parts before evaluating the product. In such algorithms, if the degree of the polynomial is not divisible by 2, this can be achieved by padding the higher degree coefficients with zeros before the computation. In the next sections, some 2-way algorithms shall be discussed.

4.1.1 Schoolbook 2-Way Algorithm

This is the basic way of multiplying two polynomials. Say there is a polynomial $A(x)$ and $B(x)$ each of degree equal to $n-1$. Their product can be computed as

$$A(x) = \sum_{i=0}^{n-1} a_i \cdot x^i = A_h(x)\, X + A_l(x) \tag{4.1}$$

$$B(x) = \sum_{i=0}^{n-1} b_i \cdot x^i = B_h(x)\, X + B_l(x) \tag{4.2}$$

where $X = x^{n/2}$

The product is

$$C(x) = A_h B_h X^2 + (A_h B_l + A_l B_h)X + A_l B_l \tag{4.3}$$

The time complexity for the schoolbook algorithm can be computed by the number of multiplication operations and addition operations needed to carry out this polynomial multiplication. It can be observed from equation 4.1, 4.2 and 4.3 that there are 4 multiplication each of size $\frac{n}{2}$ and $(n-1)$ additions for the middle terms and $(n-2)$ additions are needed for the final computation [13]. In total, $M(n) = 4M(\frac{n}{2}) + 2n - 3$. Considering $n = 2^k$ and taking $M(1) = 1$, solving for this results in $M(n) = 2n^2 - 2n + 1$ and therefore this algorithm's complexity is $O(n^2)$.

4.1.2 Karatsuba 2-Way Algorithm

This is a faster multiplication algorithm than the schoolbook 2-way that was discovered in 1963 by Karatsuba and Ofman [14]. Say we have two polynomials $A(x)$ and $B(x)$ as shown in equations 4.1 and 4.2. Their product is computed as

$$C(x) = A_h B_h X^2 + ((A_l + A_h)(B_l + B_h) + A_h B_h + A_l B_l)X + A_l B_l \tag{4.4}$$

The time complexity for the Karatsuba algorithm can be computed by observing that there are 3 multiplication each of size $\frac{n}{2}$ and $3n-4$ additions in total [13]. This results in, $M(n) = 3M(\frac{n}{2}) + 3n - 4$. Considering $n = 2^k$ and $M(1) = 1$, solving for this results in $M(n) = 7n^{log_2 3} - 8n + 2$ and therefore the complexity of this algorithm is $O(n^{1.58})$.

4.1.3 Refined Karatsuba 2-Way Algorithm

Bernstein in [1], [2] proposed an improvement of the Karatsuba 2-way algorithm as follows. Consider a polynomial $A(x)$ and $B(x)$ as in equations 4.1 and 4.2. Their product is

$$C(x) = (1 + X)(A_l B_l + X(A_h B_h)) + X(A_h + A_l)(B_h + B_l) \tag{4.5}$$

The time complexity for the refined Karatsuba algorithm can be computed by observing that there are 3 multiplication each of size $\frac{n}{2}$ and 2 additions each of size $n-1$ for the $(A_h + A_l)$ and $(B_h + B_l)$ terms and $\frac{n}{2} - 1$ additions for the first term therefore $\frac{5n}{2} - 3$ additions in total. This results in, $M(n) = 3M(\frac{n}{2}) + \frac{5n}{2} - 3$. Considering $n = 2^k$ and $M(1) = 1$, solving for this results in $M(n) = 6.5n^{log_2 3} - 7n + \frac{3}{2}$ and therefore the complexity of this algorithm is $O(n^{1.58})$.

4.2 3-Way Algorithms

3-way algorithms divide each polynomial into 3 parts before evaluating the product. In such algorithms, if the degree of the polynomial is not divisible by 3, this can be achieved by padding the higher degree coefficients with zeros before the computation. In the next sections, some 3 way algorithms shall be discussed.

4.2.1 Schoolbook 3-Way Algorithm

In the schoolbook 3-way algorithm, polynomials $A(x)$ and polynomial $B(x)$ are first divided into three equal parts:

$$A(x) = \sum_{i=0}^{n-1} a_i \cdot x^i = A_h(x)\, X^2 + A_m(x)\, X + A_l(x) \tag{4.6}$$

$$B(x) = \sum_{i=0}^{n-1} b_i \cdot x^i = B_h(x)\, X^2 + B_m(x)\, X + B_l(x) \tag{4.7}$$

where $X = x^{n/3}$

The product is computed as

$$\begin{aligned} C(x) = {} & A_h B_h X^4 + (A_m B_h + A_h B_m) X^3 + (A_l B_h + A_m B_m + A_h B_l) X^2 + \\ & (A_m B_l + A_l B_m) X + A_l B_l \end{aligned} \tag{4.8}$$

The time complexity for the schoolbook 3-way algorithm can be computed by observing that there are 9 multiplications each of size $\frac{n}{3}$ and 4 additions each of size $n-1$ therefore $4n - 8$ additions in total [13]. This results in, $M(n) = 9M(\frac{n}{3}) + 4n - 8$. Solving for this results in $M(n) = 2n^2 - 2n + 1$ and therefore the complexity of this algorithm is $O(n^2)$.

4.2.2 Karatsuba 3-way Algorithm

The Karatsuba 3-way algorithm requires polynomial $A(x)$ and polynomial $B(x)$ be divided into three parts as equations 4.6 and 4.7 . Their product can be computed as

$$\begin{aligned} C(x) = A_h B_h X^4 + ((A_m + A_h)(B_m + B_h)) + A_m B_m + A_h B_h)X^3 + \\ ((A_l A_h) + (B_l B_h) + A_l B_l + A_m B_m + A_h B_h)X^2 + \\ ((A_l + A_m)(B_l + B_m)) + A_l B_l + A_m B_m)X + A_l B_l \end{aligned} \tag{4.9}$$

The time complexity for the Karatsuba 3-way algorithm can be computed by observing that there are 6 multiplications each of size $\frac{n}{3}$ and 7 additions each of size $\frac{2n}{3} - 1$ and 4 additions each of size $\frac{n}{3} - 1$ for the middle terms therefore $6n - 11$ additions and 6 additions of $\frac{n}{3}$ for reconstruction. In total there are $8n - 11$ additions [13]. This results in, $M(n) = 6M(\frac{n}{3}) + 8n - 11$. This results in $M(n) = \frac{34}{5} n^{log_3 6} - 8n + \frac{11}{5}$ and therefore the complexity of this algorithm is $O(n^{1.63})$.

4.2.3 CNH 3-Way Split Algorithm

Cenk, Negre and Hasan in [4], [22], [23] proposed a 3-way split algorithm as follows. Say there are two polynomials $A(x)$ and $B(x)$ that are divided into three parts as equations 4.6 and 4.7 . The product is computed as

$$\begin{aligned} P_1 &= P_{1l} + P_{1h}X = A_0 B_0 \\ P_2 &= P_{2l} + P_{2h}X = (A_0 + A_1)(B_0 + B_1) \\ P_3 &= P_{3l} + P_{3h}X = A_1 B_1 \\ P_4 &= P_{4l} + P_{4h}X = (A_0 + A_2)(B_0 + B_2) \\ P_5 &= P_{5l} + P_{5h}X = A_2 B_2 \\ P_6 &= P_{6l} + P_{6h}X = (A_1 + A_2)(B_1 + B_2) \end{aligned} \tag{4.10}$$

where $X = x^{n/3}$

$$\begin{aligned} C(x) = P_{5h}X^5 + (P_{3h} + P_{5l} + P_{5h} + P_{6h})X^4 + \\ (P_{1h} + P_{3l} + P_{3h} + P_{4h} + P_{5l} + P_{5h} + P_{6l})X^3 + \\ (P_{1l} + P_{1h} + P_{2h} + P_{3l} + P_{3h} + P_{4l} + P_{5l})X^2 + \\ (P_{1l} + P_{1h} + P_{2l} + P_{3l})X + P_{1l} \end{aligned} \tag{4.11}$$

The time complexity for the CNH 3-way split algorithm can be computed by observing that there are 6 multiplications each of size $\frac{n}{3}$ and 9 additions each of size $\frac{n}{3} - 1$ and 7 additions each of size $\frac{n}{3}$ for the middle terms therefore $\frac{16n}{3} - 9$ additions and 6 additions of $\frac{n}{3}$ for reconstruction. In total there are $\frac{22n}{3} - 9$ additions [13]. This results in, $M(n) = 6M(\frac{n}{3}) + \frac{22n}{3} - 9$. This results in $M(n) = \frac{98}{15}n^{log_3 6} - \frac{22n}{3} + \frac{9}{5}$ and therefore the complexity of this algorithm is $O(n^{1.63})$.

4.3 Shift and Add Method For Binary Polynomial Multiplication

Consider a degree m binary field in a polynomial basis. In this representation, each element $a(x)$ in the field is a polynomial represented as:

$$a(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + a_{m-3}x^{m-3} + \cdots + a_2x^2 + a_1x^1 + a_0$$

The coefficients of these elements can be represented in n 64-bit words where $\lceil \frac{m}{64} \rceil$. This polynomial can therefore be stored in an array of length n in software. The multiplication of a and b can be computed using different algorithms as discussed in [12]. We shall focus only on three of these algorithms, the shift and add, left to right comb method and finally we shall consider the left to right comb method with precomputation.

The shift and add method is an algorithm for multiplying binary polynomials that is computed on noticing that

$$a \cdot b = a_{m-1}x^{m-1}b + a_{m-2}x^{m-2}b \cdots + a_2x^2b + a_1x^1b + a_0b \tag{4.12}$$

The i^{th} iteration of the algorithm computes x^ib. If $a_i = 1$, the result obtained is added to the solution c. From the above, it can be observed that this computation requires

a large amount of shift operations, that is equivalent to the bit length of a making it slow in software implementation. An improvement of this algorithm is the left to right comb method [10].

4.4 Left to right comb method For Binary Polynomial Multiplication

The left to right comb method is based by observing that once $b(x) \cdot x^k$ is precomputed for $k \in [0, 63]$ then it is easy to obtain $b(x) \cdot x^{64j+k}$ by adding to the right side of $b(x) \cdot x^k$, j zero words. This is done by left shifting the solution vector at each iteration. This algorithm can be improved by precomputing some values with a memory trade off.

4.5 Left to Right Comb method with Precomputation For Binary Polynomial Multiplication

As discussed in the previous section, the left to right comb method can be further sped up by some precomputation of $u(x)b(x)$ where polynomial $u(x)$ with degree less than w, and w is the window size which divides the word length. These precomputed values are stored for all w from $[0, 2^w - 1]$. In the multiplication loop, this is done by taking w bits of the $A[j]'s$ at a time. Considering $b(x)$ is of length m, this results in additional memory usage of $2^w(w + m)$ bits.

The implementation of the algorithms described above shall be discussed further in the next chapter. Considering polynomial multiplication of large field sizes, we shall first recursively subdivide the problem into smaller size using the recursive implementation of the algorithms discussed in 4.1 and 4.2, the binary polynomial multiplication shall then be done using a binary multiplication algorithm and finally the solution shall be computed by reconstructing the results of these calls made recursively to get the result.

CHAPTER 5

SOFTWARE IMPLEMENTATION OF POLYNOMIAL ALGORITHMS

In this chapter, the software implementation of the algorithms discussed in chapter 4 shall be discussed. We take the recursive approach for higher degree polynomials [27]. We then analyze the timing results. Once this is done, we shall optimize the algorithms by combining them. The implementation is written in C++ and timings obtained on a 2GHz Quad-Core Intel Core i5.

Each algorithm was run a 100 times and an average of the total time taken computed to get the final result. The bit sizes ranged from 64 bit by 64 bit multiplication to 2048 bit by 2048 bit multiplication for the 2-way algorithms, doubling the bit sizes at each step. For the 3-way algorithms, the bit sizes ranged from 64 bit by 64 bit multiplication to 1728 bit by 1728 bit multiplication, with each next step being triple the size of the previous.

For the base case, which will be considered as the smallest unit of polynomial multiplication computation, that is, the multiplication of a 64 bit by a 64 bit polynomial, two approaches were taken. The first was using the shift and add algorithm, the most primitive method to multiply two binary polynomials and the second using a sliding window of precomputed values. An experiment of varying window sizes is also done in order to find the best window size to achieve optimal performance.

5.1 Implementation of Binary Polynomial Multiplication Algorithms

Consider two binary polynomial $a(x) = x^4+x^2+x+1$ and polynomial $b(x) = x^3+1$. The binary representation of polynomial $a(x)$ and of $b(x)$ is $a = 10111$ and $b = 01001$. The multiplication of $a(x)$ and $b(x)$ results in $c(x) = x^7+x^5+x^3+x^2+x+1$ and in binary representation 10101111.

Binary polynomial multiplication of the two operands a and b can simply be described in two steps; a left shift of b at each iteration and an XOR operation of the left shift result to the solution vector if the value of bit i in operand a is 1. This is the primitive shift and add multiplication described in the next section.

5.1.1 Shift and Add Multiplication

This algorithm is used to carry out binary polynomial multiplication for two 64 bit polynomials. The result shall be stored in an array of size two whereby each element of the array is 64 bits long. As shown by Algorithm 1.

Algorithm 1 Shift and Add Method

Input: Polynomials $a(x)$ and $b(x)$ of degree less or equal to $n-1$
Output: $c(x) = a(x) \cdot b(x)$

if $a_0 = 1$ **then**
  $c \leftarrow b$
else
$c \leftarrow 0$
end if
for $i \leftarrow 1$ to $m-1$ **do**
  $b \leftarrow b \cdot x$ ▷ left shift
  if $a_i = 1$ **then**
    $c \leftarrow c + b$
  end if
end for
return c

5.1.2 Left to Right comb method

This algorithm is similar to the shift and add algorithm but instead of left shifting operand b at each iteration, the solution vector is instead shifted whenever the value of bit i in operand a is 1. This algorithm is also used to carry out binary polynomial multiplication for two 64 bit polynomials. It is computed as as shown by Algorithm 2.

Algorithm 2 Left to Right Comb Method

Input: Polynomials $a(x)$ and $b(x)$ of degree less or equal to $n-1$

Output: $c(x) = a(x) \cdot b(x)$, $C\{j\} = (C[\,n\,], \ldots, C[\,j+1\,], C[\,j\,])$

$c \leftarrow 0$
for $k \leftarrow n-1$ to 0 **do**
  for $j \leftarrow 0$ to $t-1$ **do**
    if k^{th} bit of $A[j]is1$ **then**
      add B to $C\{j\}$
    end if
  end for
  if $k \neq 0$ **then**
    $C \leftarrow Cx$
  end if
end for
return C

5.1.3 Left to right Comb method with Precomputation

If extra memory is available, the left to right comb method can be optimized by first choosing an arbitrary window size, w which divides the word size 64, so the window sizes can range from values 2 to 64. We define a vector $u(x)$ which contains the values from 0 to $2^w - 1$.

We can then precompute $u(x)b(x)$ for each u_i value in the $u(x)$ vector and store the results in a look up table which shall be used to fetch these values in the computation of $a(x)b(x)$. The left to right comb method with precomputation is done as shown by

Algorithm 3.

Algorithm 3 Left to Right Comb Method with precomputation

Input: Binary field polynomials $a(x), b(x)$ of degree less or equal to $n-1$

Output: $c(x) = a(x) \cdot b(x)$, $C\{j\} = (C[\,n\,], \ldots, C[\,j+1\,], C[\,j\,])$

Compute $B_u = u(x)b(x)$ for u from [0,255]

$c \leftarrow 0$

for $k \leftarrow w-1$ to 0 **do**

  for $j \leftarrow 0$ to 63 **do**

    Let $u = (u_7, u_6, u_5, u_4, u_3, u_2, u_1, u_0)$ where u_i is bit $(8k+i)$ of A[j]

    add B_u to Cj

    if $k \neq 0$ **then** $C \leftarrow Cx^8$

    end if

  end for

  if $k \neq 0$ **then**

    $C \leftarrow Cx$

  end if

end for

return C

5.2 2-Way Algorithms

Implementation of 2-way algorithms is as follows; at each step, the polynomials $a(x)$ and $b(x)$ are each recursively divided into 2 lower degree polynomials of half the size of the previous step. Once the base case is reached, the binary polynomial algorithm of choice is used to compute the 64 bit by 64 bit polynomial multiplication after which a reconstruction step follows to compute the final solution.

5.2.1 Schoolbook 2-Way Algorithm

Consider two high degree binary polynomials each of degree which is less than or equal to $n-1$. We can compute the schoolbook 2-way algorithm by first dividing

$A(x)$ and $B(x)$ to 4 polynomials of size $X^{\frac{N}{2}}$ then recursively calling the schoolbook algorithm to the lower degree polynomials and finally reconstructing before returning the solution as shown by Algorithm 4.

Algorithm 4 Schoolbook Algorithm(SB)

Input: Binary field polynomials $a(x), b(x)$ of degree less or equal to $n-1$

Output: $c(x) = a(x) \cdot b(x)$

$N \leftarrow Max(degree(A), degree(B)) + 1$

if $N = 64$ **then return** $A \cdot B$

end if

$A(x) = A_h(x)X^{\frac{N}{2}} + A_l(x)$ and $B(x) = B_h(x)X^{\frac{N}{2}} + B_l(x)$

$P_0 \leftarrow SB(A_l, B_l)$

$P_1 \leftarrow SB(A_h, B_l)$

$P_2 \leftarrow SB(A_l, B_h)$

$P_3 \leftarrow SB(A_h, B_h)$

return $P_3 \cdot X^N + (P_1 + P_2) \cdot X^{\frac{N}{2}} + P_0$

5.2.2 Karatsuba 2-Way Algorithm

Consider two high degree binary polynomials each of degree which is less than or equal to $n-1$. We can compute the Karatsuba 2-way algorithm similary by first dividing $A(x)$ and $B(x)$ to 4 polynomials each having size $X^{\frac{N}{2}}$, then recursively calling the karatsuba algorithm to the lower degree polynomials and finally reconstructing before returning the solution as shown by Algorithm 4. The Karatsuba 2-way algorithm can be computed as shown by Algorithm 5.

5.2.3 Refined Karatsuba 2-Way Algorithm

The Bernstein's improvement of the karatsuba 2-way algorithm can be computed by subdiving $A(x)$ and $B(x)$ to 4 polynomials of size $X^{\frac{N}{2}}$, then recursively calling the refined Karatsuba algorithm to the lower degree polynomials before reconstructing the solution as shown by Algorithm 6.

Algorithm 5 Karatsuba Algorithm(KA)

Input: Binary field polynomials $a(x), b(x)$ of degree less or equal to $n-1$

Output: $c(x) = a(x) \cdot b(x)$

$N \leftarrow Max(degree(A), degree(B)) + 1$

if $N = 64$ **then return** $A \cdot B$

end if

$A(x) = A_h(x)X^{\frac{N}{2}} + A_l(x)$ and $B(x) = B_h(x)X^{\frac{N}{2}} + B_l(x)$

$P_0 \leftarrow KA(A_l, B_l)$

$P_1 \leftarrow KA((A_h + A_L), (B_h + B_L))$

$P_2 \leftarrow KA(A_h, B_h)$

return $P_2 \cdot X^N + (P_1 - P_0 - P_2) \cdot X^{\frac{N}{2}} + P_0$

Algorithm 6 Refined Karatsuba Algorithm(RK)

Input: Binary field polynomials $a(x), b(x)$ of degree less or equal to $n-1$

Output: $c(x) = a(x) \cdot b(x)$

$N \leftarrow Max(degree(A), degree(B)) + 1$

if $N = 64$ **then return** $A \cdot B$

end if

$A(x) = A_h(x)X^{\frac{N}{2}} + A_l(x)$ and $B(x) = B_h(x)X^{\frac{N}{2}} + B_l(x)$

$P_0 \leftarrow RK(A_l, B_l)$

$P_1 \leftarrow RK((A_h + A_L), (B_h + B_L))$

$P_2 \leftarrow RK(A_h, B_h)$

return $(1 + X^{\frac{N}{2}})(P_0 + X^{\frac{N}{2}}P_2) + X^{\frac{N}{2}}P_0$

5.3 3-Way Algorithms

Implementation of 3-way algorithms is as follows; at each step, the polynomials $a(x)$ and $b(x)$ are each recursively divided into 3 lower degree polynomials of a third the size of the previous step. Once the base case is reached, the binary polynomial algorithm of choice is used to compute the 64 bit by 64 bit polynomial multiplication after which a reconstruction step follows to compute the final solution.

5.3.1 Schoolbook 3-Way Algorithm

Consider two high degree binary polynomials each of degree which is less than or equal to $n-1$. We can compute the schoolbook 3-way algorithm by first dividing $A(x)$ and $B(x)$ to 6 polynomials of size $X^{\frac{N}{3}}$ then recursively calling the schoolbook 3 algorithm to the lower degree polynomials and finally reconstructing before returning the solution as shown by Algorithm 7.

5.3.2 Karatsuba 3-Way Algorithm

Consider two high degree binary polynomials each of degree which is less than or equal to $n-1$. We can compute the schoolbook 3-way algorithm by first dividing $A(x)$ and $B(x)$ to 6 polynomials of size $X^{\frac{N}{3}}$ then recursively calling the Karatsuba 3-way algorithm to the lower degree polynomials and finally reconstructing before returning the solution as shown by Algorithm 8.

5.3.3 CNH 3-way Split Algorithm

Consider two high degree binary polynomials each of degree which is less than or equal to $n-1$. We can compute the schoolbook 3-way algorithm by first dividing $A(x)$ and $B(x)$ to 6 polynomials of size $X^{\frac{N}{3}}$ then recursively calling the CNH 3-way split algorithm to the lower degree polynomials and finally reconstructing before returning the solution as shown by Algorithm 9.

Algorithm 7 Schoolbook 3-Way Algorithm(S3W)

Input: Binary field polynomials $a(x), b(x)$ of degree which is less than or equal to $n-1$

Output: $c(x) = a(x) \cdot b(x)$

$N \leftarrow Max(degree(A), degree(B)) + 1$

if $N = 64$ **then return** $A \cdot B$

end if

$A(x) = A_h(x)X^{\frac{2N}{3}} + A_m(x)X^{\frac{N}{3}} + A_l(x)$

and $B(x) = B_h(x)X^{\frac{2N}{3}} + B_m(x)X^{\frac{N}{3}} + B_l(x)$

$P_0 \leftarrow S3W(A_l, B_l)$

$P_1 \leftarrow S3W(A_m, B_l)$

$P_2 \leftarrow S3W(A_l, B_m)$

$P_3 \leftarrow S3W(A_l, B_h)$

$P_4 \leftarrow S3W(A_m, B_m)$

$P_5 \leftarrow S3W(A_h, B_l)$

$P_6 \leftarrow S3W(A_m, B_h)$

$P_7 \leftarrow S3W(A_h, B_m)$

$P_8 \leftarrow S3W(A_h, B_h)$

return $P_8X^{\frac{4N}{3}} + (P_7 + P_6)X^N + (P_5 + P_4 + P_3)X^{\frac{2N}{3}} + (P_2 + P_1)X^{\frac{N}{3}} + P_0$

Algorithm 8 Karatsuba 3-Way Algorithm(K3W)

Input: Binary field polynomials $a(x), b(x)$ of degree less or equal to $n-1$

Output: $c(x) = a(x) \cdot b(x)$

$N \leftarrow Max(degree(A), degree(B)) + 1$

if $N = 64$ **then return** $A \cdot B$

end if

$A(x) = A_h(x)X^{\frac{2N}{3}} + A_m(x)X^{\frac{N}{3}} + A_l(x)$

and $B(x) = B_h(x)X^{\frac{2N}{3}} + B_m(x)X^{\frac{N}{3}} + B_l(x)$

$P_0 \leftarrow K3W(A_l, B_l)$

$P_1 \leftarrow K3W((A_l + A_m), (B_l + B_m))$

$P_2 \leftarrow K3W(A_m, B_m)$

$P_3 \leftarrow K3W((A_l + A_h), (B_l + B_h))$

$P_4 \leftarrow K3W((A_m + A_h), (B_m + B_h))$

$P_5 \leftarrow K3W(A_h, B_m)$

$P_6 \leftarrow (P_2 + P_4 + P_5)$

$P_7 \leftarrow (P_0 + P_2 + P_3 + P_5)$

$P_8 \leftarrow (P_0 + P_1 + P_2)$

return $P_5X^{\frac{4N}{3}} + P_6X^N + P_7X^{\frac{2N}{3}} + P_8X^{\frac{N}{3}} + P_0$

Algorithm 9 CNH 3-Way Split Algorithm(CNH3W)

Input: Binary field polynomials $a(x), b(x)$ of degree less or equal to $n-1$

Output: $c(x) = a(x) \cdot b(x)$

$N \leftarrow Max(degree(A), degree(B)) + 1$

if $N = 64$ **then return** $A \cdot B$

end if

$A(x) = A_h(x)X^{\frac{2N}{3}} + A_m(x)X^{\frac{N}{3}} + A_l(x)$

and $B(x) = B_h(x)X^{\frac{2N}{3}} + B_m(x)X^{\frac{N}{3}} + B_l(x)$

$P_0 \leftarrow CNH3W(A_l, B_l)$

$P_1 \leftarrow CNH3W((A_l + A_m), (B_l + B_m))$

$P_2 \leftarrow CNH3W(A_m, B_m)$

$P_3 \leftarrow CNH3W((A_l + A_h), (B_l + B_h))$

$P_4 \leftarrow CNH3W(A_h, B_m)$

$P_5 \leftarrow CNH3W((A_m + A_h), (B_m + B_h)$

$P_6 \leftarrow (P_{3h} + P_{5l} + P_{5h} + P_{6h})$

$P_7 \leftarrow (P_{1h} + P_{3l} + P_{3h} + P_{4h} + P_{5l} + P_{5h} + P_{6l})$

$P_8 \leftarrow (P_{1h} + P_{1h} + P_{2h} + P_{3l} + P_{3h} + P_{4l} + P_{5l})$

$P_9 \leftarrow (P_{1l} + P_{1h} + P_{2l} + P_{3l})$

return $P_{5h}X^{\frac{5N}{3}} + P_6X^{\frac{4N}{3}} + P_7X^N + P_8X^{\frac{2N}{3}} + P_9X^{\frac{N}{3}} + P_{1l}$

5.4 Timing Results

Table 5.1 shows the 2-way algorithms computed by the schoolbook, Karatsuba and refined Karatsuba algorithms. The base case for degree of $a(x), b(x)$ equal to 64 is handled by the shift and add binary polynomial multiplication algorithm explained in algorithm 1. We can observe that for the smaller bit sizes, the schoolbook algorithm outperforms the other two algorithms but as the bit sizes get bigger, the refined Karatsuba algorithm is faster, almost 6 times faster than the schoolbook algorithm for bit size 2048 by 2048 bits.

Table 5.2 shows the 3-way algorithms computed by the schoolbook, Karatsuba and CNH 3-way split algorithms. Similarly, the base case for degree of $a(x), b(x)$ equal to 64 is handled by the shift and add binary polynomial multiplication algorithm explained in algorithm 1. We can observe that for the smaller bit sizes, the Karatsuba algorithm outperforms the other two algorithms but as the bit sizes get bigger, the CNH 3-way algorithm is faster, almost 3 times faster than the schoolbook algorithm for bit size 1728 by 1728 bits.

In the next section, some optimization techniques are applied to the algorithms discussed in order to make them more time efficient.

Table 5.1: 2-Way Multiplication With Shift And Add Method in (ms)

m bits	Schoolbook	Karatsuba	Refined Karatsuba
64	0.010	0.010	0.012
128	0.026	0.029	0.031
256	0.117	0.189	0.143
512	0.783	0.262	0.238
1024	2.407	0.819	0.742
2048	12.702	2.058	2.004

Table 5.2: 3-Way Multiplication With Shift And Add Method in (ms)

m bits	Schoolbook	Karatsuba	CNH-3 Way Split
64	0.014	0.010	0.012
192	0.137	0.063	0.065
576	0.956	0.372	0.381
1728	6.964	2.209	2.162

Table 5.3: Left to Right comb with precomputation for base case m = 64 bits in (μs)

w=2	w=4	w=8	w=16
11	6	5	3

Table 5.4: 2-Way Multiplication With Precomputation in (ms)

m bits	Schoolbook	Karatsuba	Refined karatsuba
64	0.004	0.004	0.007
128	0.017	0.014	0.017
256	0.060	0.071	0.040
512	0.254	0.192	0.159
1024	0.075	0.435	0.579
2048	3.038	1.252	1.211

5.5 Discussion

In the following sections, we discuss the software implementation of multiplication algorithms discussed previously. In section 5.4, the results of polynomial multiplication using different 2-way and 3-way algorithms of various bit length was discussed. These algorithms can be optimised for more efficiency during computation. In the next section two such optimisation techniques are discussed in detail.

5.5.1 Precomputation of Sliding Window

Precomputation involves calculating some preknown values ahead of time and storing them in memory for look up whenever required for a faster runtime computation if memory overhead can be traded off to achieve this.

In section 4.3.3, for the left to right comb method with precomputation, a sliding window of size, w, that divides the word length could be chosen ranging from 2 to 32 and a vector, $u(x)$, of size 2^w containing all possible values of the window size generated. Assuming the base case is reached and $b(x)$ is a 64 bit polynomial, all possible values of the product $u_i b(x)$ can be can be precomputed then stored in a table for look up.

At runtime, when the base case is reached, this algorithm is then used instead of the

Table 5.5: 3-Way Multiplication With Precomputation in (ms)

m bits	Schoolbook	Karatsuba	CNH 3-Way Split
64	0.007	0.008	0.007
192	0.103	0.035	0.030
576	0.278	0.319	0.233
1728	3.475	2.189	1.052

primitive shift and add method.
An experiment of this algorithm for window sizes 2, 4, 8 and 16 was carried out and the results shown in Table 5.3.

The memory overhead is given as $2^w(w+m)$ where w is the size of the window and m is the bit length, in this case 64 bits, can be computed as follows:

- For window size, $w = 2$, is $2^w(w+m) = 2^2(2+64) \approx 0.03KB$
- For window size, $w = 4$, is $2^w(w+m) = 2^4(4+64) \approx 0.1KB$
- For window size, $w = 8$, is $2^w(w+m) = 2^8(8+64) \approx 2KB$
- For window size, $w = 16$, is $2^w(w+m) = 2^{16}(16+64) \approx 500KB$
- For window size, $w = 32$, is $2^w(w+m) = 2^{32}(32+64) \approx 34GB$

Optimally, the larger the window size the faster the algorithm but considering the space complexity of window sizes 16 and 32, the window size chosen as 8 is an acceptable trade off for this implementation.

Considering for bit size 2048 by 2048 multiplication, there are 32 different $b(x)$ where $b(x)$ is 64 bits hence a total of 64KB memory overhead. We can observe that if we repeat the 2-way and 3-way algorithms in Tables 5.1 and 5.2, as seen in Tables 5.4 and 5.5, the computation is almost twice as fast when using these precomputed values.

5.5.2 Algorithm Combination

As seen from Table 5.1 and Table 5.2, the speed of the polynomial algorithms also varies depending on the bit size of the polynomials, some algorithms dominate for

larger bit sizes namely refined Karatsuba for 2-way and CNH 3-way split for the 3-way algorithms but we can also observe that the schoolbook 2-way algorithm and the Karatsuba 3-way algorithms are faster for smaller bit sizes in comparison to the rest for the 2-way and 3-way algorithms respectfully. This is brought about by the decomposition and reconstruction steps of the algorithms slowing down the lower degree polynomials. Since these algorithms are computed recursively, we can then combine the faster algorithms depending on the step at which the algorithm is in.

For 2-way algorithms, refined Karatsuba algorithm can be used until bit size is 256 then switch to the schoolbook algorithm and for the 3-way algorithms CNH 3-way algorithm can be chosen until the bit size is equal to 192 then switched to the Karatsuba 3-way which is faster for smaller bit sizes. As we can observe from Table 5.6 and Table 5.7, this yields the optimal solution for such polynomial multiplications.

Table 5.6: Refined Karatsuba 2-Way Multiplication With Schoolbook Combination (ms)

m bits	Refined Karatsuba
64	0.011
128	0.021
256	0.121
512	0.330
1024	0.657
2048	0.946

Table 5.7: CNH 3-Way Multiplication With Karatsuba Combination (ms)

m bits	CNH 3-Way Split
64	0.009
192	0.039
576	0.181
1728	0.813

CHAPTER 6

CONCLUSION

Binary polynomial multiplication algorithms are useful in cryptography, especially for elliptic curve cryptosystems over binary fields. Different binary polynomial multiplication algorithms have been analysed in this thesis, namely, the schoolbook 2-way, the Karatsuba 2-way, the refined Karatsuba 2-way, the schoolbook 3-way, the Karatsuba 3-way, and the CNH 3-way split algorithms and their optimisations with the sliding window approach.

We observed that time optimisation of the computations of these algorithms can be done by trading off some memory using a precomputed look-up table and the sliding window method with an optimal size of 8, instead of using the shift and add method to carry out binary multiplication. When we analysed the results, we found that for the 2-way algorithms, the refined Karatsuba algorithm poses a significant improvement over the schoolbook algorithm and Karatsuba algorithm in higher degree polynomials. For the 3-way algorithms, the CNH 3-way split algorithm poses a significant improvement over the schoolbook and the Karatsuba algorithm for the higher degree polynomials.

For smaller degree polynomials, however, the schoolbook algorithm is much faster in the 2-way algorithms and the Karatsuba 3-way algorithm is faster than both the CNH 3-way split and schoolbook 3-way hence these algorithms can be combined and at every iteration the optimal algorithm is chosen. This results in the most time-efficient algorithm, especially for high degree polynomial multiplication.

REFERENCES

[1] D. J. Bernstein, Batch binary edwards., in S. Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pp. 317–336, Springer, 2009, ISBN 978-3-642-03355-1.

[2] D. J. Bernstein, C. Chuengsatiansup, and T. Lange, Curve41417: Karatsuba revisited., IACR Cryptology ePrint Archive, 2014, p. 526, 2014.

[3] J. Buchmann, *Introduction to Cryptography*, Springer, 2002, ISBN 0-387-95034-6.

[4] M. Cenk, C. Negre, and M. Hasan, *Improved three-way split formulas for binary polynomial and toeplitz matrix vector products*, 2012.

[5] B.-Z. Chor, Arithmetic of finite fields., Inf. Process. Lett., 14(1), pp. 4–6, 1982.

[6] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, editors, *Handbook of Elliptic and Hyperelliptic Curve Cryptography.*, Chapman and Hall/CRC, 2005, ISBN 978-1-4200-3498-1.

[7] D. Coppersmith, The data encryption standard (des) and its strength against attacks., IBM J. Res. Dev., 38(3), pp. 243–250, 1994.

[8] W. Diffie and M. E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, 22(6), pp. 644–654, November 1976.

[9] S. D. Galbraith, *Mathematics of Public Key Cryptography*, Cambridge University Press, 2012, ISBN 9781107013926.

[10] D. Hankerson, J. López Hernandez, and A. Menezes, Software implementation of elliptic curve cryptography over binary fields, in Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, Springer Berlin Heidelberg, 2000.

[11] D. Hankerson and A. Menezes, Elliptic curve discrete logarithm problem., in H. C. A. van Tilborg and S. Jajodia, editors, *Encyclopedia of Cryptography and Security (2nd Ed.)*, pp. 397–400, Springer, 2011, ISBN 978-1-4419-5905-8.

[12] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, 2004, ISBN 0-387-95273-X.

[13] M. B. Ilter and M. Cenk, Efficient big integer multiplication, International Journal of Information Security Science, 6, 2017.

[14] A. Karatsuba and Y. Ofman, Multiplication of multidigit numbers on automata, Soviet Physics-Doklady, 7(7), pp. 595–596, 1963, cited By 510.

[15] Koblitz, *A course in number theory and cryptography*, Springer, 2 edition, 1994.

[16] N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation, 48(177), pp. 203–209, January 1987, ISSN 0025-5718.

[17] N. Koblitz, *Algebraic aspects of cryptography*, volume 3 of *Algorithms and computation in mathematics*, Springer, 1998, ISBN 978-3-540-63446-1.

[18] R. Lidl and H. Niederreiter, *Finite Fields*, Addison-Wesley Pub. Co., Advanced Book Program/World Science Division, 1983, ISBN 9780521302401.

[19] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 2001.

[20] V. S. Miller, Use of elliptic curves in cryptography., in H. C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pp. 417–426, Springer, 1985, ISBN 3-540-16463-4.

[21] G. L. Mullen and D. Panario, editors, *Handbook of Finite Fields.*, Discrete mathematics and its applications, CRC Press, 2013, ISBN 978-1-439-87378-6.

[22] C. Negre, Improved three-way split approach for binary polynomial multiplication based on optimized reconstruction, Technical Report hal-00788646, Team DALI/LIRMM, on Hyper Articles en Ligne (HAL), 2013, cited By 3.

[23] C. Negre, Efficient binary polynomial multiplication based on optimized karatsuba reconstruction, Journal of Cryptographic Engineering, 4(2), pp. 91–106, 2014, cited By 6.

[24] N. I. of Standards and Technology, Advanced encryption standard, NIST FIPS PUB 197, 2001.

[25] R. E. Overill, Review: Advances in elliptic curve cryptography., J. Log. Comput., 15(5), p. 815, 2005.

[26] K. H. Rosen, *Elementary number theory and its applications (3. ed.).*, Addison-Wesley, 1993, ISBN 978-0-201-57889-8.

[27] A. Weimerskirch and C. Paar, Generalizations of the karatsuba algorithm for efficient implementations., IACR Cryptology ePrint Archive, 2006, p. 224, 2006.

APPENDIX A

LINK TO SOURCE CODE

https://github.com/oludoachieng/multiplications/blob/main/windowoptimized.cpp