Efficient Algorithms for Finite Fields, with Applications in Elliptic Curve Cryptography

by

Selçuk Baktır

A Thesis
Submitted to the Faculty
of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the
Degree of Master of Science
in

Electrical and Computer Engineering

by

April, 2003

Approved:

Dr. Berk Sunar Thesis Advisor ECE Department	Dr. William J. Martin Thesis Committee Mathematical Sciences Department
Dr. Kaveh Pahlavan Thesis Committee ECE Department	Dr. John A. Orr Department Head ECE Department

© Copyright by Selçuk Baktır
All rights reserved.

To my family;
to my parents Ayşe and Mehmet Baktır,
to my sisters Elif and Zeynep,
and to my brothers Selim and Oğuz

Abstract

This thesis introduces a new tower field representation, *optimal tower fields* (OTFs), that facilitates efficient finite field operations. The recursive direct inversion method presented for OTFs has significantly lower complexity than the known best method for inversion in optimal extension fields (OEFs), i.e., Itoh-Tsujii's inversion technique. The complexity of OTF inversion algorithm is shown to be $O(m^2)$, significantly better than that of the Itoh-Tsujii algorithm, i.e. $O(m^2(\log_2 m))$. This complexity is further improved to $O(m^{\log_2 3})$ by utilizing the Karatsuba-Ofman algorithm. In addition, it is shown that OTFs are in fact a special class of OEFs and OTF elements may be converted to OEF representation via a simple permutation of the coefficients. Hence, OTF operations may be utilized to achieve the OEF arithmetic operations whenever a corresponding OTF representation exists. While the original OTF multiplication and squaring operations require slightly more additions than their OEF counterparts, due to the free conversion, both OTF operations may be achieved with the complexity of OEF operations. Furthermore, efficient finite field algorithms are introduced which significantly improve OTF multiplication and squaring operations.

The OTF inversion algorithm was implemented on the ARM family of processors for a medium and a large sized field whose elements can be represented with 192 and 320 bits, respectively. In the implementation, the new OTF inversion algorithm ran at least six to eight times faster than the known best method for inversion in OEFs, i.e., Itoh-Tsujii inversion technique. According to the implementation results obtained, it is indicated that using the OTF inversion method an elliptic curve scalar point multiplication operation can be performed at least two to three times faster than the known best implementation for the selected fields.

Acknowledgements

In this thesis I describe the research I conducted in pursuit of my Master's Degree in Electrical and Computer Engineering at WPI.

Firstly, I would like to thank Berk Sunar for advising and supporting me throughout this research. I would also like to thank Kaveh Pahlavan for his suggestions and time in reviewing this work.

I would like to thank William J. Martin, a faculty member of Cryptography and Information Security Laboratory, for his suggestions and time in reviewing this work as well as for the interesting, enjoyable and beneficial lunch talks about cryptography, discrete math, number theory, complexity theory and everything. Some questions he asked kept my mind busy for hours until I found the answer, but the games he introduced, which usually could be played with mere coins and a handwritten chess board, were always very entertaining.

Further more, I would like to thank my colleagues at CRIS Laboratory, Gunnar Gaubatz, Jens Peter Kaps, Seth Hardy and Kaan Yüksel, for their friendship and the nice working environment they provided.

I remember when I was a child, being a ten year old and going to primary school, learning about Gauss finding his famous theorem for computing the sum of first n integers, and saying myself "Well, if Gauss as a kid found a short way of doing this why can't I do the same thing?" and coming up with an elegant geometrical proof and the exact formula of Gauss's theorem. With my child mind, I was happy that I did what Gauss did myself, but sad for not taking any credit for it since Gauss had already done it before me! However, my dear mom and dad always gave me credit reminding that sweet memory to me.

Finally, and most importantly, I would like to thank my parents Ayşe and Mehmet Baktır, my sisters Elif and Zeynep, and my brothers Selim and Oğuz. I learned a lot from them, and they have had the greatest influence on me in good ways. Without their unconditional love and support, this work wouldn't have been possible.

Worcester, Massachusetts — Selçuk Baktır

April 2003

Table of Contents

List of Tables

Chapter 1

Introduction

1.1 Background

The use of elliptic curves in public key cryptography was first proposed independently by Koblitz and Miller in 1980s. Since then, elliptic curve cryptography has been the focus of a lot of attention and gained great popularity due to the same level of security they provide with much smaller key sizes than conventional public key cryptosystems have.

The standard protocols in cryptography that utilize discrete logarithm problem have analogues in elliptic curve cryptography. The standard discrete logarithm problem has sub-exponential complexity, e.g. using a general number sieve method a discrete logarithm problem in $\mathbb{F}_p^*$ can be solved in sub-exponential time [16]. Whereas, a discrete logarithm on an elliptic curve $E(\mathbb{F}_q)$ has exponential complexity in the size $n = \lceil \log_2 q \rceil$ of the field elements, e.g. using the Pollard's Rho method, one of the best methods for solving discrete logarithm problem on elliptic curves, one can solve the discrete logarithm problem only in time $2^{\frac{n}{2}}$ [3].

Elliptic Curve Cryptosystems offer better security with smaller key sizes and computationally more efficient algorithms compared to traditional public key cryptosystems such as RSA [17] and discrete logarithm based systems like ElGamal [5] and Diffie-Hellman [4] algorithms. This makes them a good choice especially for constrained environments like smart cards and hand-held devices where resources as power, processing time and memory are limited.

1.2 Motivation

Elliptic curve cryptography relies on efficient algorithms for finite field arithmetic. For instance, the elliptic curve digital signature algorithm requires efficient addition, multiplication and inversion in finite fields of size larger than 2^{160}. This poses a significant problem in embedded systems where computational power is quite limited and public-key operations are unacceptably slow [22]. Efficient algorithms for finite field operations have been closely investigated [18, 13]. Besides the standard basis, alternative representations such as the normal bases [10, 11, 21], and dual bases [10, 7, 6, 9] have been studied. *Optimal Extension Fields* [2, 1] have been found to be especially successful in embedded software implementations of elliptic curve schemes. The arithmetic operations in OEFs are much more efficient than in characteristic two extensions or prime fields due to the use of a large characteristic ground field and the selection of a binomial as the field polynomial.

In the elliptic curve scalar point multiplication, a large number of field multiplications and inversions are computed. Inversion is inherently more complex and at least several times more costly than multiplication. One of the possible remedies is the use of projective coordinates for representing the points on the curve in order to avoid inversions. On the other hand, projective coordinates require significantly more temporary storage than affine coordinates. Using projective coordinates is not necessary if the complexity of inversion can be reduced significantly. The adaptation of Itoh-Tsujii method for standard basis [8], particularly for *Optimal Extension Fields* [2], has been effective in achieving fast inversion. However, despite recent improvements, inversion is still the slowest operation in elliptic curve implementations. In this thesis, this issue is addressed by proposing a new inversion method.

1.3 Contribution of the Thesis

In this thesis, a specialized finite field representation is introduced for a class of finite fields, named *Optimal Tower Fields*(OTFs). Efficient algorithms are introduced for performing finite field operations in OTFs.

The most significant contribution of this thesis is the recursive direct inversion method presented for OTFs which has significantly lower complexity than the known best method for inversion in optimal extension fields (OEFs), i.e., Itoh-Tsujii's inversion technique. The complexity of the new OTF inversion algorithm is shown to be $O(m^2)$, as little as the

complexity of multiplication and significantly better than the complexity of the Itoh-Tsujii algorithm, i.e. $O(m^2(\log_2 m))$. The complexity of OTF inversion algorithm is further improved to $O(m^{\log_2 3})$ by utilizing the Karatsuba-Ofman algorithm. Furthermore, efficient finite field algorithms are introduced which significantly improve OTF multiplication and squaring operations.

It is shown that OTFs are in fact a special class of OEFs and the conversion between the two field representations is a mere permutation. Hence, OTF operations may be utilized to achieve fast OEF arithmetic whenever a corresponding OTF representation exists.

In order to practically verify the theoretical results obtained, the OTF inversion algorithm was implemented on the ARM family of processors for a medium and a large sized field whose elements can be represented with 192 and 320 bits, respectively. Theory exactly matched practice. In the implementation, the new OTF inversion algorithm ran at least six to eight times faster than the known best method for inversion in OEFs, i.e., Itoh-Tsujii inversion technique. According to the implementation results obtained, it is indicated that using the OTF inversion method an elliptic curve scalar point multiplication operation can be performed at least two to three times faster than the known best implementation for the selected fields.

1.4 Outline of the Thesis

Chapter 2 starts by describing Optimal Extension Fields and their arithmetic. This chapter introduces a new theorem regarding the Frobenius Maps used for inversion in OEFs. The chapter ends with the explanation of the direct inversion technique for extension fields in standard basis.

Chapter 3 starts by introducing a new class of finite fields, *Optimal Tower Fields*(OTFs), and presenting their existence conditions. The chapter proves that OTFs are a class of OEFs, and the conversion between the two field representations is a mere permutation. The chapter shows that the special field representation for Optimal Tower Fields allows extremely efficient finite field arithmetic. This chapter introduces the recursive direct inversion method for OTFs, gives the complexity analysis of OTF arithmetic and compares the complexities of OTF and OEF arithmetic operations. This chapter also proposes using the Karatsuba algorithm for improving the complexities of OTF arithmetic operations. Finally, the chapter generalizes OTFs as Generalized OTFs to include a larger class of OEFs, yet conserving all

the properties and advantages of OTFs.

Finally, Chapter 4 presents implementations of both the Itoh-Tsujii technique and the OTF recursive direct inversion technique on the ARM family of processors for two selected fields. This chapter ends by comparing the performance of the two inversion techniques and making an estimate on how much faster an ECC scalar point multiplication can be performed by merely using OTF inversion technique than by using Itoh-Tsujii inversion technique.

The Appendix gives lists of some OTFs and generalized OTFs. It also gives lists of complexities of OEF, OTF and generalized OTF arithmetic for some practical values of extension degrees.

Acknowledgement:
The implementations in Chapter 4 were done in collaboration with my colleague Gunnar Gaubatz.

Chapter 2

Background

This chapter describes Optimal Extension Fields and their arithmetic. A new theorem is introduced regarding the Frobenius Maps used for inversion in OEFs. The chapter ends with the explanation of the direct inversion technique for extension fields in standard basis. The direct inversion technique introduced in this chapter is used in Chapter 3 for performing inversion in Optimal Tower Fields.

2.1 Optimal Extension Fields and their Arithmetic

Optimal extension fields were introduced by Bailey and Paar in [1]. The main idea is to use a generating polynomial of the form $P(x) = x^m - w$ to construct the extension field $GF(p^m)$, where p is selected as a *pseudo-Mersenne prime* given in the form $2^k \pm c$ with $\log_2 c < \lfloor \frac{k}{2} \rfloor$. The pseudo-Mersenne form allows efficient reduction in the ground field operations. Theorem 1 [14] provides a simple means to identify irreducible binomials that can be used in OEF construction.

In this thesis, we write

$$a \equiv b \pmod{n}$$

to mean $n|a - b$. For example,

$$7 \equiv -13 \pmod{10}$$

In contrast, we write

$$b = a \bmod n$$

to mean $b \equiv a \pmod{n}$ and $0 \leq b < n$. That is b is the canonical representative of a mod n. This b is easily found by the division algorithm. We will do likewise for polynomials.

Theorem 1 ([14]) *Let $m \geq 2$ be an integer and $a \in GF(q)^*$. Then the binomial $x^m - a$ is irreducible in $GF(q)[x]$ if and only if the following three conditions are satisfied:*

1. *each prime factor of m divides the order e of a in $GF(q)^*$;*
2. *the prime factors of m do not divide $\frac{q-1}{e}$;*
3. *$q \equiv 1 \pmod{4}$ if $m \equiv 0 \pmod{4}$.*

The representation of OEF elements utilizes the standard basis. An element $A \in GF(p^m)$ is represented as

$$A = \sum_{i=0}^{m-1} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_{m-1} x^{m-1}$$

where $a_i \in GF(p)$.

The OEF arithmetic operations are performed as follows.

Addition/Subtraction:

The addition/subtraction of two field elements $A, B \in GF(p^m)$ is performed in the usual way, by adding/subtracting the polynomial coefficients in $GF(p)$ as follows.

$$A \pm B = \sum_{i=0}^{m-1} a_i x^i \pm \sum_{i=0}^{m-1} b_i x^i = \sum_{i=0}^{m-1} (a_i \pm b_i) x^i$$

Multiplication:

Let $A, B \in GF(p^m)$. Their product $C = A \cdot B$ is computed in two steps:

1. Polynomial multiplication:
$$C' = A \cdot B = \sum_{i=0}^{2m-2} c'_i x^i$$
2. Modular reduction:
$$\begin{aligned} C &= C' \bmod P(x) \\ &= \sum_{i=0}^{2m-2} c'_i x^i \bmod x^m - w \\ &= \sum_{i=0}^{m-1} (c'_i + w c'_{i+m}) x^i \end{aligned}$$

Here we set $c'_{2m-1} = 0$. In the first step the ordinary product of two polynomials is computed. In the reduction step the binomial $P(x) = x^m - w$ facilitates efficient reduction. The reduction may be achieved using only $m - 1$ constant coefficient multiplications by w, and $m - 1$ additions.

Inversion:
An elegant method for inversion was introduced by Itoh and Tsujii [11]. Let $A \in GF(p^m)$. We want to find $B = A^{-1} \bmod P(x)$. The algorithm works in four steps:

1. Compute the exponentiation A^{r-1} in $GF(p^m)$, where $r = \frac{p^m-1}{p-1}$;
2. Compute the product $A^r = (A^{r-1}) \cdot A$;
3. Compute the inversion $(A^r)^{-1}$ in $GF(p)$;
4. Compute the product $A^{r-1} \cdot (A^r)^{-1} = A^{-1}$.

For the particular choice of

$$r = \frac{p^m - 1}{p - 1} ,$$

A^r belongs to the ground field $GF(p)$ [14]. This allows the inversion in Step 3 to be computed in the ground field $GF(p)$ instead of the larger field $GF(p^m)$. For the exponentiation A^{r-1} in Step 1, we expand the exponent $r - 1$ as follows

$$r - 1 = \frac{p^m - 1}{p - 1} - 1 = p^{m-1} + p^{m-2} + \ldots + p^2 + p .$$

The exponentiation requires the computation of powers A^{p^i} for $1 \leq i \leq m - 1$. The original Itoh-Tsujii algorithm proposes to use a normal basis representation over $GF(2)$ which turns these exponentiations into simple bitwise rotations. In [8], this technique was adapted to work efficiently in the standard basis as well. In the same reference it was shown that A^{r-1} can be computed by performing

$$\#MUL = \lfloor \log_2(m-1) \rfloor + HW(m-1) - 1$$

multiplications and

$$\#p^i - EXP = \lfloor \log_2(m-1) \rfloor + HW(m-1)$$

p^i-th power exponentiations in $GF(p^m)$, where $HW(m)$ denotes the hamming-weight of m. For the details of this algorithm the reader is referred to [8]. Instead, efficient computation

of A^{p^i} is briefly outlined for the special case of OEFs. The information given here is based on [2] and [8] which consider the computation of A^{p^i} in the standard basis. A^{p^i} is the i-th iterate of the *Frobenius map* defined as $\sigma(A) = A^p$. The following properties of the Frobenius map will be used:

- $\sigma(A + B) = \sigma(A) + \sigma(B)$ for any $A, B \in GF(p^m)$ (Linearity Property),
- $\sigma(a) = a^p = a$ for any $a \in GF(p)$ (Fermat's Little Theorem).

Using these rules the exponentiation $A^{p^i} = \sigma^i(A)$ is simplified as

$$A^{p^i} = \left(\sum_{j=0}^{m-1} a_j x^j\right)^{p^i} = \sum_{j=0}^{m-1} (a_j x^j)^{p^i} = \sum_{j=0}^{m-1} a_j x^{jp^i} \ . \tag{2.1}$$

Let's focus on the powers x^{jp^i} for $0 < i, j \leq m-1$ in the summation. Theorem 2 allows further simplification.

Theorem 2 ([8]) *Let $P(x)$ be an irreducible polynomial of the form $P(x) = x^m - w$ over $GF(p)$, e an integer, $P(\alpha) = 0$ in $GF(p^m)$, and it is understood that $p \geq 3$. Then*

$$\alpha^e = w^t \alpha^s$$

where $s = e \bmod m$ and $t = \frac{e-s}{m}$.

Hence, it is possible to precompute w^t and s for all values the exponent e takes in the summation in Equation (2.1), i.e., $e = jp^i$ for $0 < i, j \leq m-1$. Utilizing a lookup table with entries

$$c_j = w^{\frac{jp^i - (jp^i \bmod m)}{m}} \bmod p$$

one may realize the exponentiation A^{p^i} using only $m-1$ constant coefficient multiplications required to compute the terms $c_j a_j$ for $0 < j \leq m-1$ and some additions for properly collecting the reduced terms of the polynomial in Equation (2.1). The lookup table is relatively small in size, since for most OEFs m is small and $c_i \in GF(p)$. The following Claim was made in [2] to further simplify the reduction.

Claim 1 ([2])

$$(x^j)^{p^i} \bmod P(x) = w^t x^j$$

where $x^j \in GF(p)[x]$, i is an arbitrary positive rational integer, and other variables are as defined in Theorem 2.

"Proof" Since $P(x)$ is an irreducible binomial, by Theorem 1, $m|(p-1)$ which implies $p \bmod m = (p-1)+1 \bmod m = 1$. Thus $s \bmod m = jp^i \bmod m = j$. □

The Claim states that the positions of the terms in the summation in Equation (2.1) stay fixed when the p^i-th power is taken. This means that in the summation no two terms will have the same power and therefore no additions will be needed for the exponentiation. However, there is a flaw in the proof. The proof begins by assuming that m divides $p-1$ based on the first condition of Theorem 1. This will not always be correct. According to this condition, each prime factor of m has to divide the order of w. If m has repeated factors, that the order of w does not have with the same multiplicity, m will not divide the order of w, and hence will not divide $p-1$. The Claim may be fixed by explicitly requiring m to divide the order of w.

When corrected, Claim 1 eliminates additions in the computation of A^{p^i}, however, it adds yet another restriction to OEF construction. In this thesis, the following theorem is introduced which shows that the exponentiation A^{p^i} may be achieved by a simple scaled permutation of the coefficients of the polynomial representation of A.

Theorem 3 *For an irreducible binomial $P(x) = x^m - w$ defined over $GF(p)$ for constructing $GF(p^m)$, the following identity holds for an arbitrary positive integer i and $A = a_0 + a_1 x + a_2 x^2 + \cdots + a_{m-1}x^{m-1} \in GF(p^m)$,*

$$A^{p^i} = \left(\sum_{j=0}^{m-1} a_j x^j\right)^{p^i} = \sum_{j=0}^{m-1} (a_j c_{s_j}) x^{s_j}$$

where $s_j = jp^i \bmod m$ and $c_{s_j} = w^{\frac{jp^i - s_j}{m}}$. Furthermore, the s_j values are distinct for $0 \le j \le m-1$.

Proof of Theorem 3 Let $s_j = jp^i \bmod m$. Then $m|(jp^i - s_j)$ and the following expression can be written

$$x^{jp^i} = (x^m)^{\frac{jp^i - s_j}{m}} x^{s_j} = w^{\frac{jp^i - s_j}{m}} x^{s_j} \ .$$

Assigning $c_{s_j} = w^{\frac{jp^i - s_j}{m}}$ the summation

$$A^{p^i} = \left(\sum_{j=0}^{m-1} a_j x^j\right)^{p^i} = \sum_{j=0}^{m-1} a_j x^{jp^i} = \sum_{j=0}^{m-1} (a_j c_{s_j}) x^{s_j}$$

is obtained. Next, by contradiction it is proven that the s_j values are distinct for $0 \leq j \leq m-1$. Assume there is a collision $s_{j_1} = s_{j_2}$ with $0 \leq j_1 \neq j_2 \leq m-1$ then

$$\begin{aligned} j_1 p^i &= j_2 p^i \pmod{m} \\ (j_1 - j_2)p^i &= 0 \pmod{m} . \end{aligned}$$

According to Theorem 1, all prime factors of m divide $p-1$. Thus m and p are relatively prime and the above expression is satisfied only when $j_1 - j_2 = 0$, a contradiction. □

Using the method in Theorem 3, exponentiations of degree p^i may be implemented with the help of a lookup table of precomputed c_{s_j} values, using not more than $m-1$ constant coefficient multiplications and no additions.

2.2 Direct Inversion

A method for the direct computation of an inverse $B = A^{-1}$ in $GF(q^m)$ was introduced in [15]. The standard basis representation of $A(x) \in GF(q^m)$ is given as $A = \sum_{i=0}^{m-1} a_i x^i$, where $a_i \in GF(q)$, and similarly $B = \sum_{i=0}^{m-1} b_i x^i$. Consider the product $C(x) = A(x)B(x) \bmod P(x) = 1$, where $P(x)$ denotes the generating polynomial for constructing $GF(q^m)$ over $GF(q)$. This means the first coefficient of the product $A(x)B(x) \bmod P(x)$ is one and all other coefficients are zeroes. By expressing the coefficients of the product in terms of the coefficients of A and B, a system of m linear equations is formed. Solving these equations for the coefficients of B yields the inverse expressed in terms of the coefficients of A. The main advantage of using the Direct Inversion technique is that one may compute the inverse of an extension field element by doing operations only in the ground field $GF(q)$. This technique is illustrated with the following two examples:

Example 1 *Direct Inversion in $GF(q^2)$*
Let $A(x) \in GF(q^2)$ and $A(x) = a_0 + a_1 x$, where $a_0, a_1 \in GF(q)$ are known, with the irreducible field polynomial selected as $P(x) = x^2 - w$, where $w \in GF(q)$. Then

$$\begin{aligned} A(x)B(x) \bmod P(x) &= (a_0 + a_1 x)(b_0 + b_1 x) \bmod P(x) \\ &= (a_0 b_0 + w a_1 b_1) + (a_0 b_1 + a_1 b_0)x = 1 . \end{aligned}$$

The coefficients yield the following system of equations

$$\begin{pmatrix} a_0 & w a_1 \\ a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Solving the system of equations gives

$$b_0 = a_0\Delta^{-1} \quad and \quad b_1 = -a_1\Delta^{-1} \tag{2.2}$$

where $\Delta = a_0^2 - wa_1^2$.

Example 2 *Direct Inversion in* $GF(q^3)$
Let $A(x) \in GF(q^3)$ *and* $A(x) = a_0 + a_1x + a_2x^2$, *where* $a_0, a_1, a_2 \in GF(q)$ *are known, with the irreducible field polynomial selected as* $P(x) = x^3 - w$, *where* $w \in GF(q)$. *Then*

$$\begin{aligned} A(x)B(x) \bmod P(x) &= (a_0 + a_1x + a_2x^2)(b_0 + b_1x + b_2x^2) \bmod P(x) \\ &= a_0b_0 + a_1b_2w + a_2b_1w + (a_0b_1 + a_1b_0 + a_2b_2w)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 \\ &= 1 \ . \end{aligned}$$

The coefficients yield the following system of equations

$$\begin{pmatrix} a_0 & wa_2 & wa_1 \\ a_1 & a_0 & wa_2 \\ a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Solving the system of equations gives

$$b_0 = (a_0^2 - a_1a_2w)\Delta^{-1} \ , \quad b_1 = (a_2^2w - a_0a_1)\Delta^{-1} \ , \quad b_2 = (a_1^2 - a_0a_2)\Delta^{-1} \tag{2.3}$$

where

$$\Delta = a_0^3 - 3a_0a_1a_2w + a_1^3w + a_2^3w^2 \ .$$

Chapter 3

Optimal Tower Fields

This chapter introduces a new class of finite field representations, *Optimal Tower Fields*(OTFs), and presents their existence conditions. Next, it proves that OTFs are a class of OEFs, and an OTF element can be converted to OEF representation via a simple permutation of the coefficients. Hence, OTF operations may be utilized to perform OEF arithmetic whenever a corresponding OTF representation exists. This chapter shows that the field representation for Optimal Tower Fields allows efficient finite field arithmetic. The chapter introduces a new recursive direct inversion method for OTFs, gives the complexity analysis of OTF arithmetic and compares the complexities of OTF and OEF arithmetic operations. The new OTF inversion technique is shown to have $\mathcal{O}(m^2)$ complexity which is similar to the complexity of multiplication and a significant improvement over the $\mathcal{O}(m^2 \log_2 m)$ complexity of the known best inversion algorithm, i.e. Itoh-Tsujii algorithm, for OEFs. By using the Karatsuba algorithm, the complexity of inversion is further reduced to $\mathcal{O}(m^{\log_2 3})$. Finally, OTFs are generalized as Generalized OTFs to include a larger class of OEFs, yet conserving all the properties and advantages of OTFs.

The results of this chapter are used in Chapter 4 for implementing OEF inversion using OTFs.

3.1 Optimal Tower Fields

A field obtained by repeatedly extending a ground field with a series of same degree irreducible polynomials is commonly referred to as a tower field. To construct a tower field $GF(q^{t^k})$, one needs k irreducible polynomials $P_i(x)$ for $0 < i \leq k$, each of degree t and

irreducible over $GF(q^{t^{i-1}})$. The selection of these polynomials determines the representation and thus the efficiency of the field operations. In this thesis, we limit our attention to a subclass of tower fields which is introduced as follows:

Definition 1 *An Optimal Tower Field (OTF) is a finite field representation $GF(q^{t^k})$ such that*

1. *q is a pseudo-Mersenne prime,*
2. *$GF(q^{t^k})$ is constructed by an ensemble of binomials $P_i(x) = x^t - \alpha_{i-1}$ irreducible over $GF(q^{t^{i-1}})$ with $P_i(\alpha_i) = 0$, for $0 < i \leq k$.*

The definition requires a set of related irreducible binomials for the construction of Optimal Tower Fields. Before presenting an explicit construction, the following theorems are developed.

Lemma 1 *If $\alpha_i^r \in GF(q^{t^{i-1}})$ with $0 \leq r < t$, then $r = 0$.*

Proof of Lemma 1 Since $P_i(x)$ is a minimal polynomial of α_i over $GF(q^{t^{i-1}})$, the elements

$$1, \alpha_i, \alpha_i^2, \cdots, \alpha_i^{t-1}$$

are linearly independent over $GF(q^{t^{i-1}})$, so $\alpha_i^r - \beta = 0$ with $\beta \in GF(q^{t^{i-1}})$ forces $r = 0$. □

Theorem 4 *For an OTF $GF(q^{t^k})$ constructed using the binomials $P_i(x) = x^t - \alpha_{i-1}$ with $P_i(\alpha_i) = 0$, for $0 < i \leq k$, the binomial roots α_i are related as*

$$\text{ord}(\alpha_i) = t\ \text{ord}(\alpha_{i-1}) = t^i \text{ord}(\alpha_0)\ ,$$

where ord(a) *denotes the order of a field element a.*

Proof of Theorem 4 Consider the powers of α_i: $\alpha_i, \alpha_i^2, \alpha_i^3, \ldots, \alpha_i^t = \alpha_{i-1}$. Note that, the t-th power of α_i yields α_{i-1}. Likewise, t-th power of α_{i-1} yields α_{i-2} is t. This process may be repeated, so that we may conclude by induction that $\alpha_i^{t^{i-j}} = \alpha_j$. We want to prove that if $\alpha_i^k = \alpha_j$ then $k \geq t^{i-j}$. We will do this by contradiction. Now suppose $\alpha_i^k = \alpha_j$ with $0 < k < t^{i-j}$, $j < i$. Let's write

$$k = t^l(st + r)$$

with $0 \leq s$, and $0 < r < t$, i.e. $t^l || k$. Then

$$\begin{aligned} \alpha_i^k &= \alpha_i^{t^{l(st+r)}} \\ &= \alpha_{i-l-1}^s \cdot \alpha_{i-l}^r \\ &= \alpha_j \end{aligned}$$

giving

$$\alpha_{i-l}^r = \alpha_j (\alpha_{i-l-1})^{-s} \ .$$

Now $k < t^{i-j}$ gives

$$k = t^l(st + r) < t^{i-j} \ ,$$

but $st + r > 0$, so

$$t^l < t^{i-j}$$

giving

$$l < i - j$$

and

$$j < i - l \ .$$

Note that α_j and $(\alpha_{i-l-1})^{-s}$ both belong to $GF(q^{t^{i-l-1}})$. So $\alpha_{i-l}^r = \alpha_j \cdot (\alpha_{i-l-1})^{-s}$ also belongs to $GF(q^{t^{i-l-1}})$. According to Lemma 1, this is satisfied only when $r = 0$ which is a contradiction with our assumption that $r > 0$. Our assumption that $k < t^{i-j}$ was wrong. Therefore, t^{i-j} is the first power of α_i which gives α_j. Hence, $\text{ord}(\alpha_i) = t^{i-j}\text{ord}(\alpha_j)$ and $\text{ord}(\alpha_i) = t^i \text{ord}(\alpha_0)$.

□

Theorem 5 *If there exists an irreducible binomial $Q(x) = x^t - a$ over $GF(q)$, then $\frac{q^t-1}{q-1}$ is divisible by t.*

Proof of Theorem 5 If $Q(x) = x^t - a$ is irreducible over $GF(q)$, then all three conditions of Theorem 1 are satisfied. Another binomial is constructed by choosing an arbitrary primitive element $a' \in GF(q)$: $P(x) = x^t - a'$. This binomial is irreducible over $GF(q)$ since the three conditions of Theorem 1 are satisfied, i.e.,

1. the order of a primitive element in $GF(q)$ is $q - 1$. Since $\text{ord}(a)|q - 1$ and each prime factor of t divides $\text{ord}(a)$, each prime factor of t also divides $\text{ord}(a') = q - 1$;

2. prime factors of t do not divide $\frac{q-1}{e'} = 1$, where $e' = q-1$ is the order of a';

3. t and q are the same for $Q(x)$ and $P(x)$; since the condition $q \equiv 1 \pmod 4$ if $t \equiv 0 \pmod 4$ was satisfied for $Q(x)$, it will be satisfied for $P(x)$ as well.

Now consider the root of $P(x) = x^t - a'$. According to Theorem 4, the root of $P(x)$ will have order $t(q-1)$. The order of the multiplicative group of $GF(q^t)$ is $q^t - 1$. Thus, the order of x, i.e. $t(q-1)$, divides the multiplicative group order $q^t - 1$. Consequently, $\frac{q^t-1}{q-1}$ is divisible by t. □

Theorem 6 *Let $P(x) = x^t - a$ be an irreducible binomial over $GF(q)$ and t' denote the product of the prime factors of t (i.e., t' is the square free part of t). Then $q^t \equiv 1 \pmod{t't}$.*

Proof of Theorem 6 Since $P(x)$ is given as irreducible, the first condition of Theorem 1, i.e., $t'|e$ is met. The order e always divides the group order $q-1$, and therefore $t'|q-1$. According to Theorem 5, $\frac{q^t-1}{q-1}$ is divisible by t or equivalently $t(q-1)|q^t-1$. Hence $t't|q^t-1$ and $q^t \equiv 1 \pmod{t't}$. □

The following theorem establishes necessary and sufficient conditions for the existence of OTFs:

Theorem 7 *Given an irreducible binomial $P_1(x) = x^t - \alpha_0$ over $GF(q)$ with $P_1(\alpha_1) = 0$ in $GF(q^t)$, all binomials of the form $P_i(x) = x^t - \alpha_{i-1}$ over $GF(q^{t^{i-1}})$, where $P_i(\alpha_i) = 0$ in $GF(q^{t^i})$ for $i > 0$, are also irreducible provided that none of the prime factors of t divides $\frac{q^t-1}{t(q-1)}$.*

Proof of Theorem 7 It needs to be proven that the conditions in Theorem 1 are satisfied for all binomials $P_i(x) = x^t - \alpha_{i-1}$ for $i > 1$. The first condition is always satisfied since all $\mathrm{ord}(\alpha_i)$ are multiples of t for $i > 0$ (Theorem 4). The third condition is also always satisfied, since it was satisfied for the first irreducible polynomial, i.e., if $t \equiv 0 \pmod 4$ and $q \equiv 1 \pmod 4$, then always $q^n \equiv 1 \pmod 4$. It will now be proven that once the second condition is satisfied for the first irreducible binomial $P_1(x)$, it will be satisfied for all the other binomials, $P_i(x)$ for $i > 1$, provided that none of the prime factors of t divides $\frac{q^t-1}{t(q-1)}$. Induction is used for the proof. It will be proven that if no prime factor of t divides $\frac{q^{t^n}-1}{t^n \cdot \mathrm{ord}(\alpha_0)}$ then no prime factor of t divides $\frac{q^{t^{n+1}}-1}{t^{n+1} \cdot \mathrm{ord}(\alpha_0)}$. The latter is factorized as follows.

$$\frac{q^{t^{n+1}}-1}{t^{n+1} \cdot \mathrm{ord}(\alpha_0)} = \frac{q^{t^n}-1}{t^n \cdot \mathrm{ord}(\alpha_0)} \cdot \frac{\sum_{i=0}^{t-1} q^{t^n i}}{t} \tag{3.1}$$

The first factor is not divisible by any prime factor of t. The second factor must be shown to be not divisible by any prime factor of t either. For this the result of Theorem 6 is used, i.e.,

$$q^t \equiv 1 \pmod{t't}$$

to simplify the summation for $n > 0$ as follows

$$\sum_{i=0}^{t-1} (q^t)^{t^{n-1}i} \equiv \sum_{i=0}^{t-1} (1)^{t^{n-1}i} \pmod{t't} \equiv t \pmod{t't}.$$

Hence,

$$\sum_{i=0}^{t-1} q^{t^n i} = k't't + t$$

for some integer k' and

$$\frac{\sum_{i=0}^{t-1} q^{t^n i}}{t} = k't' + 1$$

and therefore

$$\frac{\sum_{i=0}^{t-1} q^{t^n i}}{t} \bmod u = 1$$

for any prime factor u of t. Hence $\frac{\sum_{i=0}^{t-1} q^{t^n i}}{t}$ isn't divisible by any prime factor of t. The condition still remains for $n = 0$, i.e., $\frac{q^t-1}{t(q-1)}$ should not be divisible by any prime factor of t. □

Theorem 7 provides a simple means for checking the existence of OTFs for chosen values of q, t, and α_0. It should be noted that the existence condition is not dependent on k, which greatly simplifies the construction. Table 5.1 and Table 5.2 (Appendix) provide lists of practical OTF constructions for $GF(q^{2^k})$ and $GF(q^{3^k})$.

3.2 Conversion between OTFs and OEFs

An OTF $GF(q^{t^k})$ is isomorphic to an OEF $GF(q^m)$ if $m = t^k$. Before explaining how an associated OEF is obtained from a given OTF, the following theorem is introduced.

Theorem 8 *For a given OTF* $GF(q^{t^k})$, *if* $t \equiv 2 \pmod 4$ *then* $q \equiv 1 \pmod 4$.

Proof of Theorem 8 From Theorem 5, it is known that $\frac{q^t-1}{q-1}$ is divisible by t. The extension degree t is given as even ($t \equiv 2 \pmod 4$). Likewise, $q - 1$ is even since q is a

prime. Note that $q \neq 2$ since for $q = 2$ there are no irreducible binomials (and OTFs) over $GF(q)$. Therefore, $(q-1)t$ and $q^t - 1$ are divisible by 4. Since $q^t \equiv 1 \pmod 4$, then either $q \equiv 1 \pmod 4$ or $q \equiv 3 \pmod 4$. Note that for $q = 0$ or $q = 2$, the equality $q^t \equiv 1 \pmod 4$ is never satisfied. Next, it will be shown that $q \equiv 3 \pmod 4$ is never satisfied either, hence $q \equiv 1 \pmod 4$.

If $t \equiv 2 \pmod 4$, it can be written as $t = 4r + 2 = 2(2r + 1)$ for some integer $r \geq 0$. According to Theorem 7, none of the prime factors of t divides $\frac{q^t-1}{t(q-1)}$, hence 2 does not divide

$$\frac{q^t - 1}{t(q-1)} = \frac{q^{2(2r+1)} - 1}{2(2r+1)(q-1)} = \frac{(q^2-1)(\sum_{i=0}^{2r} q^{2i})}{2(2r+1)(q-1)} = \frac{(q+1)(\sum_{i=0}^{2r} q^{2i})}{2(2r+1)} \ .$$

Therefore 4 does not divide

$$\frac{(q+1)(\sum_{i=0}^{2r} q^{2i})}{2r+1} \ ,$$

and $q + 1$ is not divisible by 4. Since $q \not\equiv 3 \pmod 4$ it follows that $q \equiv 1 \pmod 4$. □

Constructing an equivalent OEF representation from a given OTF representation is established by the following theorem.

Theorem 9 *For a given OTF representation of $GF(q^{t^k})$ with $P_i(x) = x^t - \alpha_{i-1}$ for $0 < i \leq k$, there exists an associated OEF representation with irreducible polynomial $Q(x) = x^m - w$ such that $Q(\alpha_k) = 0$, $m = t^k$ and $w = \alpha_0$.*

Proof of Theorem 9 Consider the set of relations $P_i(\alpha_i) = \alpha_i^t - \alpha_{i-1} = 0$ for $0 < i \leq k$. In the first relation $P_1(\alpha_1) = \alpha_1^t - \alpha_0 = 0$, by repeatedly substituting α_i^t in place of α_{i-1}, the following relation is obtained:

$$Q(\alpha_k) = \alpha_k^{t^k} - \alpha_0 = 0 \ .$$

Hence, the binomial $Q(x) = x^{t^k} - \alpha_0$ with $Q(\alpha_k) = 0$ is obtained. This binomial has the form of an OEF binomial $x^m - w$, with $w = \alpha_0$ and $m = t^k$. It will be shown that $Q(x)$ satisfies the three conditions of Theorem 1, hence is irreducible over $GF(q)$ and generates the field $GF(q^m)$. It is known that $P_1(x) = x^t - \alpha_0$ is irreducible and satisfies the three conditions of Theorem 1. Since the prime factors of t and t^k are the same, $w = \alpha_0$, and q is identical for both of the binomials $P_1(x)$ and $Q(x)$, the first two of the three conditions of Theorem 1 are satisfied for $Q(x) = x^{t^k} - \alpha_0$ as well. For the third condition, if $t \equiv 0 \pmod 4$ then $t^k \equiv 0 \pmod 4$, and therefore $q \equiv 1 \pmod 4$ for both cases. On the other hand, if $t \not\equiv 0 \pmod 4$ then there are three cases that need to be considered:

- If $t \equiv 1 \pmod 4$, then $t^k \equiv 1 \pmod 4$, hence condition 3 disappears for $Q(x)$.
- If $t \equiv 2 \pmod 4$, then $t^k \equiv 0 \pmod 4$, thus condition 3 applies in this case. Theorem 8 confirms that $q \equiv 1 \pmod 4$ whenever $t \equiv 2 \pmod 4$. Therefore, condition 3 is satisfied.
- If $t \equiv -1 \pmod 4$, then either $t^k \equiv 1 \pmod 4$ or $t^k \equiv -1 \pmod 4$, hence condition 3 disappears.

It has been proven that $Q(x)$ satisfies the three conditions of Theorem 1, hence it is irreducible over $GF(q)$ and constructs an OEF. □

In the above Theorem 9, it has been shown that the OTF construction leads to an associated OEF representation. Likewise it is possible to construct an OTF representation from a given OEF representation as introduced in the following theorem.

Theorem 10 *For an OEF representation of a finite field $GF(q^m)$ with irreducible polynomial $Q(x) = x^m - w$, if $m = t^k$ and none of the prime factors of t divide $\frac{q^t-1}{t(q-1)}$ then there exists an OTF representation such that $P_1(x) = x^t - \alpha_0$ and $\alpha_0 = w$.*

Proof of Theorem 10 The irreducibility of $P_1(x) = x^t - w$ over $GF(q)$ needs to be shown, so that all the conditions of Theorem 7, which establishes the necessary and sufficient conditions for the existence of OTFs, are satisfied. As stated, $Q(x) = x^m - w = x^{t^k} - w$ is irreducible. Therefore, it satisfies the three conditions of Theorem 1. Since the prime factors of t are identical to the prime factors of t^k and w is same for $Q(x)$ and $P_1(x)$, conditions 1 and 2 of Theorem 1 are both satisfied for $P_1(x)$ too. For condition three of Theorem 1 there are two cases, either $t \equiv 0 \pmod 4$ or $t \not\equiv 0 \pmod 4$. In the first case, $t^k \equiv 0 \pmod 4$ and therefore $q \equiv 1 \pmod 4$ for both cases. Alternatively, if $t \not\equiv 0 \pmod 4$ then the condition disappears. It has been shown that $P_1(x) = x^t - w$ is irreducible, hence all conditions of Theorem 7 are satisfied. □

In order to construct explicit conversion rules between the OTF and OEF representations, some notation is briefly introduced. The OTF representation of an element $A \in GF(q^{t^k})$ is given as follows.

$$A = a_0 + a_1\alpha_k + a_2\alpha_k^2 + \ldots + a_{t-1}\alpha_k^{t-1}$$

where $a_i \in GF(q^{t^{k-1}})$ for $0 \leq i < t$. Similarly, a_i are represented as polynomials over $GF(q^{t^{k-2}})$ in the subfield:

$$a_i = \sum_{j=0}^{t-1} a_{ij} \alpha_{k-1}^j \quad , \quad \text{for } 0 \leq i < t \ . \tag{3.2}$$

This process continues for k levels until the ground field is reached. Note that, in this notation in each level a new value from the range $[0, t-1]$ is appended to coefficient indices. Hence, a coefficient in the ground field $GF(q)$ will have a k-digit t-ary number as index.

To obtain the standard basis representation of A each α_{i-1} is repeatedly replaced by α_i^t for $1 < i \leq k$ until a univariate polynomial in α_k with coefficients in the ground field $GF(q)$ is obtained. This polynomial is now in a standard basis representation over $GF(q)$ with $Q(\alpha_k) = \alpha_k^{t^k} - \alpha_0$ as the modulus polynomial. The relation between the OTF representation and the OEF representation defines the conversion. The following theorem constructs an explicit rule for conversion.

Theorem 11 *For a given OTF/OEF association the conversion from one representation to the other is a simple permutation of the coefficients. The permutation maps a coefficient a_ℓ of an element $A \in GF(q^{t^k})/GF(q^m)$ to the corresponding coefficient in $GF(q^m)/GF(q^{t^k})$ whose index is determined by the t-ary value of the mirror image of ℓ.*

Proof of Theorem 11 The following observation is made in Equation (3.2). The index ℓ of an element a_ℓ determines its position and the power of α_i it multiplies with in the $(k+1-i)$-th level of the OTF representation for $1 \leq i \leq k$. For instance, in the $(k+1-i)$-th level, the i-th digit ℓ_{i-1} of ℓ is appended to the index ℓ. ℓ_{i-1} gives the power of α_i that the coefficient a_ℓ multiplies with in this level. Collecting the α_i in k levels, the following multiplier is obtained for the coefficient a_ℓ in the ground field

$$\prod_{i=0}^{k-1} \alpha_{i+1}^{l_i} = \prod_{i=0}^{k-1} \alpha_k^{t^{k-1-i} l_i} \ .$$

The RHS follows from $\alpha_i = \alpha_k^{t^{k-i}}$. Considering the exponent $t^{k-1-i} l_i$, notice that the list l is effectively reversed. Therefore, a coefficient a_l is mapped to the location specified by the mirror image of l. Since the indices of the coefficients are not repeated, neither will the indices of the converted coefficients repeat and the conversion will always be a permutation. □

The conversion technique is illustrated by the following example.

Example 3 *Let $GF(q^{2^3})$ denote an OTF. The binomials used in the construction are given as $P_1(x) = x^2 - \alpha_0$, $P_2(x) = x^2 - \alpha_1$ and $P_3(x) = x^2 - \alpha_2$, defined over $GF(q)$, $GF(q^2)$ and $GF(q^{2^2})$ respectively, and $P_1(\alpha_1) = 0$, $P_2(\alpha_2) = 0$ and $P_3(\alpha_3) = 0$. An element $A \in GF(q^{2^3})$ has the following OTF representation*

$$A = ((a_{000} + a_{001}\alpha_1) + (a_{010} + a_{011}\alpha_1)\alpha_2) + ((a_{100} + a_{101}\alpha_1) + (a_{110} + a_{111}\alpha_1)\alpha_2)\alpha_3$$

According to Theorem 11, the indices of the permuted coefficients are obtained by taking the mirror image of their indices in the OTF representation. For instance, a_{011} will be mapped to the coefficient whose index is the binary value of the mirror image of its index, $(110)_2 = 6$, and thus a_{011} will become the coefficient of α_3^6. The OEF standard basis representation is obtained as

$$A = a_{000} + a_{100}\alpha_3 + a_{010}\alpha_3^2 + a_{110}\alpha_3^3 + a_{001}\alpha_3^4 + a_{101}\alpha_3^5 + a_{011}\alpha_3^6 + a_{111}\alpha_3^7$$

with field polynomial $Q(x) = x^8 - \alpha_0$, and $Q(\alpha_3) = 0$. The result is easily verified by converting the OTF representation directly by replacing α_2 with α_3^2 and α_1 with α_3^4.

3.3 Complexity Analysis

In this section the complexities for arithmetic operations in OTF representations are derived for second and third degree extensions. The following notation is introduced for the complexities:

- $\mathcal{A}_k$: Complexity of addition operation in $GF(q^{t^k})$,
- $\mathcal{S}_k$: Complexity of squaring operation in $GF(q^{t^k})$,
- $\mathcal{M}_k$: Complexity of multiplication operation in $GF(q^{t^k})$,
- $\mathcal{C}_k$: Complexity of multiplication of a $GF(q)$ element with an element of $GF(q^{t^k})$,
- $\mathcal{I}_k$: Complexity of inversion operation in $GF(q^{t^k})$.

Before deriving the computational complexities for OTF arithmetic operations, the following observation is made. Let $A \in GF(q^{t^i})$, where $A = a_0 + a_1\alpha_i + a_2\alpha_i^2 + a_3\alpha_i^3 + \cdots + a_{t-1}\alpha_i^{t-1}$

and $a_0, a_1, \ldots, a_{t-1} \in GF(q^{t^{i-1}})$. Consider the product

$$\begin{aligned} A\alpha_i &= \left(\sum_{j=0}^{t-1} a_j \alpha_i^j\right) \alpha_i = \sum_{j=0}^{t-1} a_j \alpha_i^{j+1} \\ &= a_{t-1}\alpha_{i-1} + \sum_{j=1}^{t-1} a_{j-1}\alpha_i^j \end{aligned} \tag{3.3}$$

The effect of multiplying an element $A \in GF(q^{2^i})$ with α_i is to rotate the coefficients $a_0, a_1, \ldots, a_{t-1}$ of A to the right and scale the first coefficient after rotation with α_{i-1}. However, note that multiplication of the first coefficient after rotation with α_{i-1}, i.e. $a_{t-1}\alpha_{i-1}$, will be similarly transformed in the subfield. Hence, the transformation progresses until $GF(q)$ is reached where the modulus polynomial is $\alpha_1^2 - \alpha_0$ and scaling a coefficient in $GF(q)$ with $\alpha_0 \in GF(q)$ means a constant multiplication in $GF(q)$. The complexity of this operation is denoted by $\mathcal{C}_0$.

3.3.1 Cost of $GF(q^{2^k})$ Operations in OTF Representation

It is assumed that the tower field $GF(q^{2^k})$ is constructed using a series of irreducible binomials of the form $P_i(x) = x^2 - \alpha_{i-1}$ over $GF(q^{2^{i-1}})$ for $0 < i \leq k$ where $\alpha_i \in GF(q^{2^i})$ is a root of $P_i(x)$.

Addition: For $A, B \in GF(q^{2^i})$ the addition operation

$$A + B = (a_0 + a_1\alpha_i) + (b_0 + b_1\alpha_i) = (a_0 + b_0) + (a_1 + b_1)\alpha_i$$

requires two $GF(q^{2^{i-1}})$ additions, hence $\mathcal{A}_i = 2\mathcal{A}_{i-1}$ and complexity of $GF(q^{2^k})$ addition in non-recursive form is

$$\mathcal{A}_k = 2^k \mathcal{A}_0 \ .$$

Multiplication: The multiplication operation $AB = (a_0 + a_1\alpha_i)(b_0 + b_1\alpha_i)$ becomes

$$\begin{aligned} AB &= a_0b_0 + (a_0b_1 + a_1b_0)\alpha_i + a_1b_1\alpha_i^2 \\ &= (a_0b_0 + a_1b_1\alpha_{i-1}) + (a_0b_1 + a_1b_0)\alpha_i \ . \end{aligned}$$

The computation requires four subfield multiplications, two additions and the multiplication of $a_1b_1 \in GF(q^{2^{i-1}})$ by α_{i-1}. Thus,

$$\mathcal{M}_i = 4\mathcal{M}_{i-1} + 2\mathcal{A}_{i-1} + \mathcal{C}_0 \ .$$

The nonrecursive form for the complexity of $GF(q^{2^k})$ multiplication is obtained as

$$\begin{aligned}\mathcal{M}_k &= 4^k\mathcal{M}_0 + \sum_{j=1}^{k} 4^{k-j}(2^j\mathcal{A}_0 + \mathcal{C}_0) \\ &= 4^k\mathcal{M}_0 + (4^k - 2^k)\mathcal{A}_0 + \frac{1}{3}(4^k - 1)\mathcal{C}_0 \ . \end{aligned} \tag{3.4}$$

The complexity may be improved by using the Karatsuba-Ofman algorithm [12]. In order to achieve the four products only three multiplications will be needed:

$$AB = (a_0b_0 + a_1b_1\alpha_{i-1}) + ((a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1b_1)\alpha_i \ . \tag{3.5}$$

With the application of the Karatsuba method the complexity changes as follows.

$$\mathcal{M}_i^{KOA} = 3\mathcal{M}_{i-1}^{KOA} + 5\mathcal{A}_{i-1} + \mathcal{C}_0$$

The complexity of $GF(q^{2^k})$ multiplication with the Karatsuba technique is found as

$$\begin{aligned}\mathcal{M}_k^{KOA} &= 3^k\mathcal{M}_0 + \sum_{j=1}^{k} 3^{k-j}(2^{j-1}5\mathcal{A}_0 + \mathcal{C}_0) \\ &= 3^k\mathcal{M}_0 + 5(3^k - 2^k)\mathcal{A}_0 + \frac{1}{2}(3^k - 1)\mathcal{C}_0 \ . \end{aligned}$$

Squaring: The squaring operation $A^2 = (a_0 + a_1\alpha_i)^2$ becomes

$$\begin{aligned}A^2 &= a_0^2 + 2a_0a_1\alpha_i + a_1^2\alpha_i^2 \\ &= (a_0^2 + a_1^2\alpha_{i-1}) + 2a_0a_1\alpha_i\end{aligned}$$

which may be achieved by two squarings, one multiplication, two additions in the subfield, and one constant multiplication in the ground field:

$$\mathcal{S}_i = 2\mathcal{S}_{i-1} + \mathcal{M}_{i-1} + 2\mathcal{A}_{i-1} + \mathcal{C}_0 \ .$$

Since the multiplication complexity $\mathcal{M}_{i-1}$ depends on the use of the Karatsuba algorithm, two non-recursive complexity equations are obtained for the cost of squaring in $GF(q^{2^k})$:

$$\begin{aligned}\mathcal{S}_k &= 2^k\mathcal{S}_0 + \sum_{j=1}^{k} 2^{k-j}(\mathcal{M}_{j-1} + \mathcal{C}_0 + 2^j\mathcal{A}_0) \\ &= 2^k\mathcal{S}_0 + \frac{1}{2}(4^k - 2^k)\mathcal{M}_0 + \frac{1}{2}(k2^k + 4^k - 2^k)\mathcal{A}_0 + \frac{1}{6}(4^k + 3\cdot 2^k - 4)\mathcal{C}_0 \end{aligned} \tag{3.6}$$

$$\begin{aligned}\mathcal{S}_k^{KOA} &= 2^k\mathcal{S}_0 + \sum_{j=1}^{k} 2^{k-j}(\mathcal{M}_{j-1} + 2^j\mathcal{A}_0 + \mathcal{C}_0) \\ &= 2^k\mathcal{S}_0 + (3^k - 2^k)\mathcal{M}_0 + \frac{1}{2}(10\cdot 3^k - 3k2^k - 10\cdot 2^k)\mathcal{A}_0 + \frac{1}{2}(3^k - 1)\mathcal{C}_0\end{aligned}$$

Inversion: The inverse of an element in $GF(q^{2^i})$ may be computed by the application of the Direct Inversion technique as shown in (4.1):

$$b_0 = a_0(a_0^2 - \alpha_{i-1}a_1^2)^{-1} \quad \text{and} \quad b_1 = -a_1(a_0^2 - \alpha_{i-1}a_1^2)^{-1} \ .$$

The computation requires two squarings, one addition, one inversion, and two multiplications in $GF(q^{2^{i-1}})$, and one multiplication in the ground field. This can be expressed as follows.

$$\mathcal{I}_i = \mathcal{I}_{i-1} + 2\mathcal{S}_{i-1} + 2\mathcal{M}_{i-1} + \mathcal{A}_{i-1} + \mathcal{C}_0$$

The aggregate cost of inversion in $GF(q^{2^k})$is found as the summation

$$\mathcal{I}_k = \mathcal{I}_0 + \sum_{j=0}^{k-1}(2\mathcal{S}_j + 2\mathcal{M}_j + \mathcal{A}_j + \mathcal{C}_0)$$

which is simplified as follows.

$$\mathcal{I}_k = \mathcal{I}_0 + (4^k - 2^k)\mathcal{M}_0 + 2(2^k - 1)\mathcal{S}_0 + (k2^k + 4^k - 4\cdot 2^k + 3)\mathcal{A}_0 + \frac{1}{3}(4^k + 3\cdot 2^k - 3k - 4)\mathcal{C}_0 \quad (3.7)$$

It is possible to slightly improve the number of constant multiplications by applying OEF multiplication and squaring to OTF inversion. The conversion is a mere permutation and therefore comes for free. The improved complexity is found as

$$\mathcal{I}_k = \mathcal{I}_0 + (4^k + 2^k - 2)\mathcal{M}_0 + (4^k + 2^{k+1} - 6k - 3)\mathcal{A}_0 + (2^{k+2} - 3k - 4)\mathcal{C}_0 \ . \quad (3.8)$$

Instead, if the Karatsuba Algorithm introduced in (3.5) is used for all multiplications, the complexity is further reduced to

$$\mathcal{I}_k^{KOA} = \mathcal{I}_0 + 2(3^k - 2^k)\mathcal{M}_0 + 2(2^k - 1)\mathcal{S}_0 + (10\cdot 3^k - 3k2^k - 13\cdot 2^k + 3)\mathcal{A}_0 + (3^k - k - 1)\mathcal{C}_0 \ . \quad (3.9)$$

3.3.2 Cost of $GF(q^{3^k})$ Operations in OTF Representation

In the OTF representation the tower field $GF(q^{3^k})$ is constructed using a series of irreducible binomials of the form $P_i(x) = x^3 - \alpha_{i-1}$ over $GF(q^{3^{i-1}})$ for $0 < i \leq k$ where $\alpha_i \in GF(q^{3^i})$ is a root of $P_i(x)$.

Addition: Similar to the $GF(q^{2^k})$ case the addition operation in $GF(q^{3^i})$ requires three $GF(q^{3^{i-1}})$ additions, hence $\mathcal{A}_i = 3\mathcal{A}_{i-1}$ and in non-recursive form the complexity of addition in $GF(q^{3^k})$ is

$$\mathcal{A}_k = 3^k \mathcal{A}_0 \ .$$

Multiplication: The multiplication $AB = (a_0 + a_1\alpha_i + a_2\alpha_i^2)(b_0 + b_1\alpha_i + b_2\alpha_i^2)$ may be computed as

$$\begin{aligned} AB &= a_0b_0 + (a_0b_1 + a_1b_0)\alpha_i + (a_2b_0 + a_1b_1 + a_0b_2)\alpha_i^2 + (a_2b_1 + a_1b_2)\alpha_i^3 + a_2b_2\alpha_i^4 \\ &= a_0b_0 + (a_2b_1 + a_1b_2)\alpha_{i-1} + (a_0b_1 + a_1b_0 + a_2b_2\alpha_{i-1})\alpha_i + (a_2b_0 + a_1b_1 + a_0b_2)\alpha_i^2 \ . \end{aligned}$$

Hence, the complexity of $GF(q^{3^i})$ multiplication is

$$\mathcal{M}_i = 9\mathcal{M}_{i-1} + 6\mathcal{A}_{i-1} + 2\mathcal{C}_0 \ .$$

The nonrecursive equation for the complexity of multiplication in $GF(q^{3^k})$ is obtained as

$$\begin{aligned} \mathcal{M}_k &= 9^k\mathcal{M}_0 + \sum_{j=1}^{k} 9^{k-j}(6 \cdot 3^{j-1}\mathcal{A}_0 + 2\mathcal{C}_0) \\ &= 9^k\mathcal{M}_0 + (9^k - 3^k)\mathcal{A}_0 + \frac{1}{4}(9^k - 1)\mathcal{C}_0 \ . \end{aligned} \tag{3.10}$$

Using the ternary version of the Karatsuba method the complexity might be improved:

$$AB = D_0 + (D_5 - D_1 - D_2)\alpha_{i-1} + (D_3 - D_1 - D_0 + D_2\alpha_{i-1})\alpha_i + (D_4 - D_2 - D_0 + D_1)\alpha_i^2$$

where

$$\begin{aligned} D_0 &= a_0b_0 \\ D_1 &= a_1b_1 \\ D_2 &= a_2b_2 \\ D_3 &= (a_0 + a_1)(b_0 + b_1) \\ D_4 &= (a_0 + a_2)(b_0 + b_2) \\ D_5 &= (a_1 + a_2)(b_1 + b_2) \ . \end{aligned} \tag{3.11}$$

This reduces the number of multiplications in exchange of extra additions. The complexity of $GF(q^{3^i})$ multiplication becomes

$$\mathcal{M}_i^{KOA} = 6\mathcal{M}_{i-1} + 15\mathcal{A}_{i-1} + 2\mathcal{C}_0 \ .$$

In nonrecursive form the complexity in $GF(q^{3^k})$ is obtained as

$$\begin{aligned} \mathcal{M}_k^{KOA} &= 6^k\mathcal{M}_0 + \sum_{j=1}^{k} 6^{k-j}(3^{j-1}15\mathcal{A}_0 + 2\mathcal{C}_0) \\ &= 6^k\mathcal{M}_0 + \frac{2}{5}(6^k-1)\mathcal{C}_0 + 5(6^k-3^k)\mathcal{A}_0 \ . \end{aligned}$$

Squaring: The squaring operation of $A \in GF(q^{3^i})$ is achieved as follows.

$$\begin{aligned} A^2 &= a_0^2 + 2a_0a_1\alpha_i + (2a_2a_0 + a_1^2)\alpha_i^2 + 2a_2a_1\alpha_i^3 + a_2^2\alpha_i^4 \\ &= a_0^2 + 2a_2a_1\alpha_{i-1} + (2a_0a_1 + a_2^2\alpha_{i-1})\alpha_i + (2a_2a_0 + a_1^2)\alpha_i^2 \ . \end{aligned}$$

This may be realized with complexity

$$\mathcal{S}_i = 3\mathcal{S}_{i-1} + 3\mathcal{M}_{i-1} + 6\mathcal{A}_{i-1} + 2\mathcal{C}_0 \ .$$

The nonrecursive complexities for $GF(q^{3^k})$ are obtained as

$$\begin{aligned} \mathcal{S}_k &= 3^k\mathcal{S}_0 + \sum_{j=1}^{k} 3^{k-j}(3\mathcal{M}_{j-1} + 6\cdot 3^{j-1}\mathcal{A}_0 + 2\mathcal{C}_0) \\ &= 3^k\mathcal{S}_0 + \frac{1}{2}(9^k-3^k)\mathcal{M}_0 + \frac{1}{8}(9^k+4\cdot 3^k-5)\mathcal{C}_0 + \frac{1}{2}(9^k-3^k+2k3^k)\mathcal{A}_0 \end{aligned} \quad (3.12)$$

$$\begin{aligned} \mathcal{S}_k^{KOA} &= 3^k\mathcal{S}_0 + \sum_{j=1}^{k} 3^{k-j}(3\mathcal{M}_{j-1} + 6\cdot 3^{j-1}\mathcal{A}_0 + 2\mathcal{C}_0) \\ &= 3^k\mathcal{S}_0 + (6^k-3^k)\mathcal{M}_0 + \frac{2}{5}(6^k-1)\mathcal{C}_0 + (5\cdot 6^k - 3k3^k - 5\cdot 3^k)\mathcal{A}_0 \end{aligned}$$

Inversion: The inverse of an element $A \in GF(q^{3^i})$ may be computed by the application of the Direct Inversion technique as shown in (2.3):

$$b_0 = \Delta^{-1}(a_0^2 - a_1a_2\alpha_{i-1}) \ , \quad b_1 = \Delta^{-1}(a_2^2\alpha_{i-1} - a_0a_1) \ , \quad b_2 = \Delta^{-1}(a_1^2 - a_0a_2)$$

where

$$\Delta = a_0^3 + ((a_1^2 - 3a_0a_2)a_1 + a_2^3\alpha_{i-1})\alpha_{i-1} \ .$$

The computation requires three squarings, six multiplications, seven addition, and one inversion in $GF(q^{3^{i-1}})$, and one multiplication in the ground field with the assumption that multiplication by 3 is realized with two additions. The resulting complexity is expressed as follows.

$$\mathcal{I}_i = \mathcal{I}_{i-1} + 3\mathcal{S}_{i-1} + 9\mathcal{M}_{i-1} + 8\mathcal{A}_{i-1} + 4\mathcal{C}_0$$

The aggregate cost of inversion in $GF(q^{3^k})$ is expressed as the summation

$$\mathcal{I}_k = \mathcal{I}_0 + \sum_{i=0}^{k-1} 3\mathcal{S}_{i-1} + 9\mathcal{M}_{i-1} + 8\mathcal{A}_{i-1} + 4\mathcal{C}_0$$

which is simplified as follows.

$$\begin{aligned}\mathcal{I}_k &= \mathcal{I}_0 + \frac{3}{2}(3^k-1)\mathcal{S}_0 + \frac{1}{16}(21\cdot 9^k - 12\cdot 3^k - 9)\mathcal{M}_0 + \frac{1}{64}(21\cdot 9^k + 48\cdot 3^k - 8k - 69)\mathcal{C}_0 \\ &\quad + \frac{1}{16}(21\cdot 9^k + 24\cdot 3^k k - 56\cdot 3^k + 35)\mathcal{A}_0 \end{aligned} \tag{3.13}$$

The complexity $\mathcal{I}_k$ in $GF(q^{3^k})$ when OEF multiplication and squaring are utilized for all multiplications and squarings is found as

$$\begin{aligned}\mathcal{I}_k &= \mathcal{I}_0 + \frac{3}{16}(7\cdot 9^k + 4\cdot 3^k - 11)\mathcal{M}_0 + 2(3^{k+1} - 4k - 3)\mathcal{C}_0 + \frac{1}{16}(21\cdot 9^k + 4\cdot 3^k - 253)\mathcal{A}_0 \end{aligned} \tag{3.14}$$

On the other hand if the Karatsuba Algorithm introduced in (3.11) is used for all multiplications, the complexity is further reduced to

$$\begin{aligned}\mathcal{I}_k^{KOA} &= \mathcal{I}_0 + \frac{3}{2}(3^k-1)\mathcal{S}_0 + \frac{1}{10}(24\cdot 6^k - 15\cdot 3^k - 9)\mathcal{M}_0 + \frac{1}{25}(24\cdot 6^k - 20k - 24)\mathcal{C}_0 \\ &\quad + \frac{1}{4}(48\cdot 6^k - 18\cdot 3^k k - 77\cdot 3^k + 29)\mathcal{A}_0 \ . \end{aligned} \tag{3.15}$$

3.4 Comparison of OTF and OEF Complexities

The treatment of OEFs is based on the detailed complexity analysis given in [1]. In the derivation of the complexities small w values are assumed for OEFs and small α_0 values are assumed for OTFs, and therefore $\mathcal{C}_0 \cong \mathcal{A}_0$. It is also assumed that $\mathcal{S}_0 = \mathcal{M}_0$ for simplification. Below, Table 3.1 summarizes the complexities compiled from [1] and from Equations (3.4), (3.6), (3.7), (3.10), (3.12), (3.13). The complexities are derived in terms of m, the extension degree for OEFs ($GF(q^m)$), and $m = t^k$ for OTFs ($GF(q^{t^k})$). To differentiate OTFs of form $GF(q^{2^k})$ and $GF(q^{3^k})$, OTF2 and OTF3 are used, respectively.

Operation	$\mathcal{M}_0$	$\mathcal{A}_0$
$\mathcal{M}(m)$ (OEF)	m^2	m^2-1
$\mathcal{M}(m)$ (OTF2)	m^2	$\frac{1}{3}(4m^2-3m-1)$
$\mathcal{M}(m)$ (OTF3)	m^2	$\frac{1}{4}(5m^2-4m-1)$
$\mathcal{S}(m)$ (OEF)	$\frac{1}{2}(m^2+m)$	$\frac{1}{2}(m^2+5m-8)$, m even $\frac{1}{2}(m^2+3m-2)$, m odd
$\mathcal{S}(m)$ (OTF2)	$\frac{1}{2}(m^2+m)$	$\frac{1}{6}(4m^2+3m\log_2 m-4)$
$\mathcal{S}(m)$ (OTF3)	$\frac{1}{2}(m^2+m)$	$\frac{1}{8}(5m^2+8m\log_3 m-5)$
$\mathcal{I}(m)$ (OEF)	$(\delta-1)m^2+\delta(m-1)+2m$	$(\delta-1)(m^2-1)+m-1$
$\mathcal{I}(m)$ (OTF2)	m^2+m-2	$\frac{1}{3}(4m^2+3m\log_2 m-9m+5)$
$\mathcal{I}(m)$ (OTF3)	$\frac{1}{16}(21m^2+12m-33)$	$\frac{1}{64}(105m^2-176m-8\log_3 m+96m\log_3 m+71)$

Table 3.1: Comparison of OEF and OTF complexities ($\delta=\lfloor\log_2(m-1)\rfloor+HW(m-1)$)

Table 3.1 shows that the number of ground field multiplications used in performing OTF multiplications and squarings are identical to those of OEF operations. Note that the single inversion $\mathcal{I}_0$ in the ground field required for $\mathcal{I}(m)$ is not shown in the table. OEF multiplication and squaring appear to be slightly more efficient than OTF multiplication and squaring, due to a reduced number of additions. However, it was shown earlier that OTFs can be converted to an OEF representation and back via simple permutation. Therefore, it shall be assumed that the OEF multiplication and squaring algorithms are used to achieve OTF multiplications and squarings with no overhead in the conversion. In Table 5.3 and Table 5.4 (Appendix) the number of operations required for multiplication and squaring are tabulated for practical values of m. These tables show that performing the two operations in OTFs and OEFs can be considered to be of equal complexity for practical considerations.

For inversion the new algorithm presents a significant improvement over the OEF inversion algorithm. In 3.1, the number of multiplications required for OEF inversion is given as $(\delta-1)m^2+\delta(m-1)+2m$. The value of δ depends on the bit length and the Hamming weight of $m-1$. By carefully selecting m, the Hamming weight may be minimized to 1, leading to a $\lfloor\log_2(m-1)\rfloor m^2+(\lfloor\log_2(m-1)\rfloor+1)(m-1)+2m$ complexity. The number of additions grows similarly with δ. In practical terms this means that the Itoh-Tsujii inversion technique will cost at least $\log_2(m-1)$ field multiplications when applied to OEFs. In elliptic curve implementations based on OEFs, typically an inversion/multiplication cost ratio of larger than 4 is observed. On the other hand, as seen in 3.1, the inversion complexity for OTFs grows linearly with m^2, both in the number of multiplications and additions. For

both $GF(q^{2^k})$ and $GF(q^{3^k})$ OTFs, the inversion/multiplication cost ratio is slightly larger than one. The asymptotic complexity of the OTF inversion algorithm makes it even more desirable for larger values of m. For instance, for $m = 32$ the complexity of inversion in OEFs is found as $\mathcal{I}_0 + 8535\mathcal{M}_0 + 8215\mathcal{A}_0$, whereas for OTFs it is only $\mathcal{I}_0 + 1054\mathcal{M}_0 + 1426\mathcal{A}_0$.

The complexity of OTF inversion may be slightly improved by utilizing OEF multiplication and squaring in implementing the OTF inversion. Then the complexity of $GF(q^{2^k})$ inversion is derived in Equation (3.8) as

$$\mathcal{I}(m)^* = \mathcal{I}_0 + (m^2 + m - 2)\mathcal{M}_0 + (m^2 + 2m - 6\log_2 m - 3)\mathcal{A}_0 + (4m - 3\log_2 m - 4)\mathcal{C}_0$$

and for $GF(q^{3^k})$ in Equation (3.14) as

$$\mathcal{I}(m)^* = \mathcal{I}_0 + \frac{3}{16}(7m^2 + 4m - 11)\mathcal{M}_0 + 2(3m - 4\log_3 m - 3)\mathcal{C}_0 + \frac{1}{16}(21m^2 + 4m - 25)\mathcal{A}_0 \ .$$

In Table 5.5 (Appendix) the number of ground field operations are summarized for practical values of m.

These complexities are dramatically reduced by using the Karatsuba-Ofman algorithm for multiplications, yielding the following inversion complexities in terms of the word length m. For $GF(q^{2^k})$, the complexity is derived in Equation (3.9) as

$$\mathcal{I}(m)^{KOA} = (2m^{\log_2 3} - 2)\mathcal{M}_0 + (11m^{\log_2 3} - 3m\log_2 m - 13m - \log_2 m + 2)\mathcal{A}_0$$

and for $GF(q^{3^k})$ in Equation (3.15) as

$$\mathcal{I}(m)^{KOA} = \frac{12}{5}(m^{\log_3 6} - 1)\mathcal{M}_0 + (\frac{324}{25}m^{\log_3 6} - \frac{9}{2}m\log_3 m - \frac{77}{4}m - \frac{4}{5}\log_3 m + \frac{629}{100})\mathcal{A}_0 \ .$$

Table 5.6 (Appendix) provides the number of ground field operations for several values of m. For $m = 32$, the complexity of inversion is found as $I_0 + 484\mathcal{M}_0 + 1774\mathcal{A}_0$, with a significantly lower number of multiplications compared to the standard version of the OTF inversion technique.

3.5 Generalization of OTFs

The OTF definition given in Section 3 restricts the ground field to $GF(q)$, a prime field, and the extension degree to a power of an integer. For instance, for an extension degree of $12 = 3 \cdot 2^2$ or $72 = 3^2 \cdot 2^3$, Definition 1 does not allow an OTF construction. This may

be too restrictive for certain applications. However, by allowing $GF(q)$ to be an extension field this restriction may be overcome. Note that $GF(q)$ may itself be in a specialized field representation, e.g. an OTF with $GF(q) = GF(p^{t_1^{k_1}})$. In this case, the roots of the irreducible binomials constructing the two OTFs may be linked by choosing α_0 of the OTF $GF((p^{t_1^{k_1}})^{t_2^{k_2}})$ to be the root of the generating binomial of the OTF $GF(p^{t_1^{k_1}})$.

For such a generalization, the theorems introduced for OTF construction, i.e. Theorems 4, 5, 6 and 7, will still apply. Furthermore, the complexity analysis presented in Section 5 and the conversion rule described in Section 4, will continue to hold over $GF(p^{t_1^{k_1}})$. Rules for conversion between the OTF $GF((p^{t_1^{k_1}})^{t_2^{k_2}})$ and the OEF $GF(p^{t_1^{k_1} t_2^{k_2}})$ representations similar to those described in Section 3.2 will apply. Conversion between the two field representations becomes a simple permutation of the ground field $GF(p)$ coefficients as before. Also, the advantage shown in (3.3) still applies when multiplying α_0 of the OTF $GF(q^{t_2^{k_2}})$, with an element in $GF(q)$.

It is possible to attain extension degrees of the form $t_1^{k_1} \cdot t_2^{k_2} \cdots t_n^{k_n}$ by repeatedly extending OTFs, and by linking their representations through the constant term α_0 as described earlier. For instance, one may construct

- $GF(p^{t_1^{k_1}})$ by extending the prime field $GF(p)$, with $P_i(x) = x^{t_1} - \alpha_{i-1}^{(1)}$ and $P_i(\alpha_i^{(1)}) = 0$ for $1 \leq i \leq k_1$,
- $GF((p^{t_1^{k_1}})^{t_2^{k_2}})$ by extending $GF(p^{t_1^{k_1}})$, with $P_i(x) = x^{t_2} - \alpha_{i-1}^{(2)}$ and $P_i(\alpha_i^{(2)}) = 0$ for $1 \leq i \leq k_2$,
- $GF((p^{t_1^{k_1}})^{t_2^{k_2}})^{t_3^{k_3}})$ by extending $GF((p^{t_1^{k_1}})^{t_2^{k_2}})$, with $P_i(x) = x^{t_3} - \alpha_{i-1}^{(3)}$ and $P_i(\alpha_i^{(3)}) = 0$ for $1 \leq i \leq k_3$ etc.

The second tower field $GF((p^{t_1^{k_1}})^{t_2^{k_2}})$ is linked to the first tower field $GF(p^{t_1^{k_1}})$ by choosing $\alpha_0^{(2)} = \alpha_{k_1}^{(1)}$. Likewise, the third tower field is linked to the second by setting $\alpha_0^{(3)} = \alpha_{k_2}^{(2)}$. In general, the j-th tower field is linked to the $(j-1)$-st by selecting $\alpha_0^{(j)} = \alpha_{k_{j-1}}^{(j-1)}$. This procedure is continued until the desired extension degree is reached. For the construction, Theorems 1 and 7 may still be used with minimal or no change.

Table 5.7 and Table 5.8 (Appendix) give lists of $p = 2^n + c$ and a values constructing generalized OTFs of the form $GF((p^{3^2})^2)$ where an OTF is constructed on top of the OTF $GF(p^{3^2})$ with first irreducible binomial $P_1(x) = x^3 - a$, and $GF((p^3)^{2^k})$ where an OTF is constructed on top of the OEF $GF(p^3)$ with irreducible binomial $x^3 - a$, for $-5 \leq c \leq 5$,

$7 \leq n \leq 16$ and $-5 \leq a \leq 5$. Table 5.9 (Appendix) gives a list of $q = 2^n + c$ and a values constructing generalized OTFs of the form $GF((q^{3^2})^{2^k})$ where an OTF is constructed on top of the OTF $GF(q^{3^2})$ with first irreducible binomial $P_1(x) = x^3 - a$, for $-5 \leq c \leq 5$, $7 \leq n \leq 16$ and $-5 \leq a \leq 5$. The computational complexity of arithmetic operations for some generalized OTFs of the form $GF((p^3)^{2^k})$, $GF((q^{3^2})^2)$ and $GF((q^{3^2})^{2^k})$ are listed in Tables 8, 11 and 13 (Appendix).

Chapter 4

Fast OEF Inversion Using Optimal Tower Fields

In this chapter OTF recursive direct inversion technique introduced in Chapter 3 is utilized to perform OEF inversion. Both OTF inversion and Itoh-Tsujii inversion for OEFs are implemented on the ARM family of processors and the timings are obtained. The performance of the two inversion techniques are compared and finally an estimate is made on how long an ECC scalar point multiplication would take depending on which inversion technique is used.

4.1 Introduction

Elliptic curve point multiplication requires a large number of multiplications and inversions. Hence, efficient arithmetic is very important for elliptic curve cryptography. *Optimal Extension Field* arithmetic is found to be efficient for elliptic curve implementations in embedded systems [2]. Especially the adaptation of Itoh-Tsujii's method in [8] has been effective in achieving fast inversion. However, inversion is still the most costly operation and is several times slower than multiplication. In this chapter, this problem is solved by implementing OEF inversion by utilizing *Optimal Tower Fields* introduced in Chapter 3.

4.1.1 Recursive Direct Inversion for Optimal Tower Fields

The information regarding OTF recursive direct inversion technique given in this section are drawn from Chapters 2 and 3.

Let $A(x) \in GF(q^2)$ and $A(x) = a_0 + a_1 x$, where $a_0, a_1 \in GF(q)$, with the irreducible field polynomial selected as $P(x) = x^2 - w$, where $w \in GF(q)$. Then, writing the inverse $A(x)$ in $GF(q^2)$ as $B(x) = b_0 + b_1 x$, where $b_0, b_1 \in GF(q)$,

$$\begin{aligned} A(x)B(x) \bmod P(x) &= (a_0 + a_1 x)(b_0 + b_1 x) \bmod P(x) \\ &= (a_0 b_0 + w a_1 b_1) + (a_0 b_1 + a_1 b_0)x. \end{aligned}$$

So $A(x)B(x) \bmod P(x) = 1$ yields the system

$$\begin{pmatrix} a_0 & w a_1 \\ a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} .$$

Solving the system of equations gives

$$b_0 = a_0 \Delta^{-1} \quad \text{and} \quad b_1 = -a_1 \Delta^{-1} \tag{4.1}$$

where $\Delta = a_0^2 - w a_1^2$. Hence, the inverse of $A(x) = a_0 + a_1 x$ in $GF(q^2)$ is obtained by carrying out an inversion only in $GF(q)$ for computing Δ^{-1}.

In this chapter, OTF recursive direct inversion technique is implemented for OTFs of the form $GF(q^{2^4})$ and $GF(q^{2^5})$. This is achieved by recursively applying the direct inversion technique 3 and 4 times for $GF(q^{2^4})$ and $GF(q^{2^5})$, respectively. Hence, inversion is performed only in the ground field $GF(q)$ which may be carried out by table lookup.

4.2 Construction of OTFs $GF(q^{2^4})$ and $GF(q^{2^5})$

In this section, the construction of $GF(q^{2^4})$ and $GF(q^{2^5})$ is explained. For the construction of OTFs, Theorems 1 and 7 are used.

4.2.1 Construction of $GF(q^{2^4})$

$GF(q^{2^4})$ is constructed with $q = 2^{12} - 3 = 4093$ and $\alpha_0 = 2$. For the selected q and α_0, one can easily check using Theorem 1 and Theorem 7 that the binomials $P_i(x) = x^2 - \alpha_{i-1}$ with

$P_i(\alpha_i) = 0$ are irreducible over $GF(q^{2^{i-1}})$, for $0 < i \leq 4$, hence they construct the OTF $GF(q^{2^4})$. The irreducible binomials $P_i(x)$ satisfy $P_i(\alpha_i) = 0$ for $0 < i \leq 4$. Therefore the following relations hold:

$$\begin{aligned} P_1(\alpha_1) &= \alpha_1^2 - 2 = 0 \\ P_2(\alpha_2) &= \alpha_2^2 - \alpha_1 = 0 \\ P_3(\alpha_3) &= \alpha_3^2 - \alpha_2 = 0 \\ P_4(\alpha_4) &= \alpha_4^2 - \alpha_3 = 0 \ . \end{aligned}$$

By repeatedly substituting α_i^2 in place of α_{i-1} for $2 \leq i \leq 4$ in the first relation $P_1(\alpha_1) = \alpha_1^2 - 2 = 0$, the following relation is obtained

$$Q(\alpha_4) = \alpha_4^{2^4} - 2 = 0 \ .$$

Hence, the binomial $Q(x) = x^{2^4} - 2$ is obtained with $Q(\alpha_4) = 0$. This binomial has the form of an OEF binomial $x^m - w$, with $w = 2$ and $m = 2^4$. Using Theorem 1 or Theorem 9 one may verify in a straightforward manner that $Q(x)$ is irreducible over $GF(q)$ and generates the OEF $GF(q^m)$.

It is shown above that the OTF $GF(q^{2^4})$ with $\alpha_0 = 2$ is associated with the OEF $GF(q^m)$, for $w = 2$ and $m = 2^4$. In Chapter 3, Theorem 11 tells that in an OTF/OEF association the conversion between the two field representations is a simple permutation of the coefficients. The rest of this subsection explains how an element in OTF representation maps to an element in OEF representation and vice versa using a simple permutation. An element $A \in GF(q^{2^4})$ is represented as follows:

$$\begin{aligned} A &= a_0 + a_1\alpha_1 + (a_2 + a_3\alpha_1)\alpha_2 + (a_4 + a_5\alpha_1 + (a_6 + a_7\alpha_1)\alpha_2)\alpha_3 \\ &\quad + (a_8 + a_9\alpha_1 + (a_{10} + a_{11}\alpha_1)\alpha_2 + (a_{12} + a_{13}\alpha_1 + (a_{14} + a_{15}\alpha_1)\alpha_2)\alpha_3)\alpha_4 \ . \end{aligned}$$

In order to obtain the standard basis representation of A Theorem 11 may be used. Alternatively, all α_i may be substituted by $\alpha_4^{2^{4-i}}$ for $1 \leq i \leq 3$, and by simple algebraic manipulation a univariate polynomial in α_4 with coefficients in the ground field $GF(q)$ is obtained. A is now in standard basis representation over $GF(q)$ with $Q(\alpha_4) = \alpha_4^{16} - 2$ as the modulus polynomial:

$$\begin{aligned} A &= a_0 + a_8\alpha_4 + a_4\alpha_4^2 + a_{12}\alpha_4^3 + a_2\alpha_4^4 + a_{10}\alpha_4^5 + a_6\alpha_4^6 + a_{14}\alpha_4^7 + a_1\alpha_4^8 + a_9\alpha_4^9 + a_5\alpha_4^{10} + a_{13}\alpha_4^{11} \\ &\quad + a_3\alpha_4^{12} + a_{11}\alpha_4^{13} + a_7\alpha_4^{14} + a_{15}\alpha_4^{15} \ . \end{aligned}$$

4.2.2 Construction of $GF(q^{2^5})$

The field $GF(q^{2^5})$ is constructed using the same procedure as explained in the previous subsection with $q = 2^{10} - 3 = 1021$ and $\alpha_0 = 2$. It can easily be verified using Theorem 1 and Theorem 7 that the binomials $P_i(x) = x^2 - \alpha_{i-1}$ with $P_i(\alpha_i) = 0$ are irreducible over $GF(q^{2^{i-1}})$, for $0 < i \leq 5$, hence they construct the OTF $GF(q^{2^5})$. The irreducible binomials $P_i(x)$ satisfy $P_i(\alpha_i) = 0$ for $0 < i \leq 5$, therefore the following relations hold:

$$\begin{aligned} P_1(\alpha_1) &= \alpha_1^2 - 2 = 0 \\ P_2(\alpha_2) &= \alpha_2^2 - \alpha_1 = 0 \\ P_3(\alpha_3) &= \alpha_3^2 - \alpha_2 = 0 \\ P_4(\alpha_4) &= \alpha_4^2 - \alpha_3 = 0 \\ P_5(\alpha_5) &= \alpha_5^2 - \alpha_4 = 0 \ . \end{aligned}$$

In the first relation $P_1(\alpha_1) = \alpha_1^2 - 2 = 0$, by repeatedly substituting α_i^2 in place of α_{i-1} for $2 \leq i \leq 5$, the following relation is obtained

$$Q(\alpha_5) = \alpha_5^{2^5} - 2 = 0 \ .$$

Hence, the binomial $Q(x) = x^{2^5} - 2$ is obtained with $Q(\alpha_5) = 0$. This binomial has the form of an OEF binomial $x^m - w$, with $w = 2$ and $m = 2^5$. As before, using Theorem 1 or Theorem 9, it can be checked that $Q(x)$ is irreducible over $GF(q)$ and generates the OEF $GF(q^m)$.

Having shown that the OTF $GF(q^{2^5})$ with $\alpha_0 = 2$ is associated with the OEF $GF(q^m)$, for $w = 2$ and $m = 2^5$, it is easy to demonstrate how an element can be transformed from one representation to the other by permutation. An element $A \in GF(q^{2^5})$ is represented as

$$A = A_0 + A_1\alpha_5$$

where

$$\begin{aligned} A_0 &= a_0 + a_1\alpha_1 + (a_2 + a_3\alpha_1)\alpha_2 + (a_4 + a_5\alpha_1 + (a_6 + a_7\alpha_1)\alpha_2)\alpha_3 \\ &\quad +(a_8 + a_9\alpha_1 + (a_{10} + a_{11}\alpha_1)\alpha_2 + (a_{12} + a_{13}\alpha_1 + (a_{14} + a_{15}\alpha_1)\alpha_2)\alpha_3)\alpha_4 \end{aligned}$$

and

$$\begin{aligned} A_1 &= (a_{16} + a_{17}\alpha_1 + (a_{18} + a_{19}\alpha_1)\alpha_2 + (a_{20} + a_{21}\alpha_1 + (a_{22} + a_{23}\alpha_1)\alpha_2)\alpha_3 \\ &\quad +(a_{24} + a_{25}\alpha_1 + (a_{26} + a_{27}\alpha_1)\alpha_2 + (a_{28} + a_{29}\alpha_1 + (a_{30} + a_{31}\alpha_1)\alpha_2)\alpha_3)\alpha_4 \ . \end{aligned}$$

In order to obtain the standard basis representation of A, Theorem 11 may be used, or alternatively all α_i are replaced by $\alpha_5^{2^{5-i}}$ for $1 \leq i \leq 4$ and a univariate polynomial in α_5 with coefficients in the ground field $GF(q)$ is obtained.

$$\begin{aligned} A &= a_0 + a_{16}\alpha_5 + a_8\alpha_5^2 + a_{24}\alpha_5^3 + a_4\alpha_5^4 + a_{20}\alpha_5^5 + a_{12}\alpha_5^6 + a_{28}\alpha_5^7 \\ &\quad + a_2\alpha_5^8 + a_{18}\alpha_5^9 + a_{10}\alpha_5^{10} + a_{26}\alpha_5^{11} + a_6\alpha_5^{12} + a_{22}\alpha_5^{13} + a_{14}\alpha_5^{14} + a_{30}\alpha_5^{15} \\ &\quad + a_1\alpha_5^{16} + a_{17}\alpha_5^{17} + a_9\alpha_5^{18} + a_{25}\alpha_5^{19} + a_5\alpha_5^{20} + a_{21}\alpha_5^{21} + a_{13}\alpha_5^{22} + a_{29}\alpha_5^{23} \\ &\quad + a_3\alpha_5^{24} + a_{19}\alpha_5^{25} + a_{11}\alpha_5^{26} + a_{27}\alpha_5^{27} + a_7\alpha_5^{28} + a_{23}\alpha_5^{29} + a_{15}\alpha_5^{30} + a_{31}\alpha_5^{31} \end{aligned}$$

This polynomial is now in standard basis representation over $GF(q)$ with $Q(\alpha_5) = \alpha_5^{32} - 2$ as the modulus polynomial.

4.3 Complexity Comparison of OTF vs. Itoh-Tsujii Inversion

Based on Table 3.1, Table 4.1 summarizes the complexities of OEF and OTF inversion in terms of m, the extension degree for OEFs ($GF(q^m)$), and $m = 2^k$ for OTFs ($GF(q^{2^k})$). Remember that inversion in ground field, $\mathcal{I}_0$, is ignored since it can be performed by table lookup for moderate sizes of q. Also, it is assumed that $\mathcal{S}_0 = \mathcal{M}_0$ for simplification, and $\mathcal{C}_0 = \mathcal{A}_0$ since multiplication with a small constant, e.g. 2, can be achieved with only an addition or a shift operation. According to [8] an inversion in an OEF $GF(p^m)$, may be achieved by computing $\delta - 1$ multiplications in $GF(p^m)$, at most δ Frobenius maps in $GF(p^m)$, $2m$ multiplications in $GF(p)$, and $m-1$ additions in $GF(p)$, where $\delta = \lfloor \log_2(m-1) + HW(m-1) \rfloor$ and $HW(m)$ denotes the hamming-weight of m.

Operation	$\mathcal{M}_0$	$\mathcal{A}_0$
$\mathcal{I}(m)$ (OEF)	$(\delta-1)m^2 + \delta(m-1) + 2m$	$(\delta-1)(m^2-1) + m - 1$
$\mathcal{I}(m)$ (OTF)	$m^2 + m - 2$	$\frac{1}{3}(4m^2 + 3m\log_2 m - 9m + 5)$

Table 4.1: Comparison of OEF and OTF complexities

Below, Table 4.2 compares the complexities of OTF vs. Itoh-Tsujii inversion for the specific cases of the OTFs $GF(q^{2^4}), GF(q^{2^5})$ and corresponding OEFs $GF(q^{16}), GF(q^{32})$.

Field	Itoh-Tsujii Inversion	OTF Inversion
$GF(q^{24})/GF(q^{16})$	$1673\mathcal{M}_0 + 1545\mathcal{A}_0$	$270\mathcal{M}_0 + 355\mathcal{A}_0$
$GF(q^{25})/GF(q^{32})$	$8535\mathcal{M}_0 + 8215\mathcal{A}_0$	$1054\mathcal{M}_0 + 1426\mathcal{A}_0$

Table 4.2: Complexities of Itoh-Tsujii vs. OTF Inversion

4.4 Implementation

In order to verify the theoretical performance benefits of OTF inversion over OEF, arithmetic routines for both methods have been implemented for the ARM7TDMI platform [19]. ARM family of processors are popular platforms for embedded systems due to their low power consumption and comparatively high performance. It is expected that OTF arithmetic will primarily be used in constrained environments, with applications ranging from mobile computing as in PDAs to smartcard security systems.

The arithmetic routines were developed in plain C using the ARM Developer Suite version 1.0.1, which includes the Metrowerks Codewarrior compiler. The performance of the arithmetic routines were measured using the "ARMulator" emulation engine. The emulation engine reports a statistic of elapsed clock cycles which is made available through the AXD debugger's user interface.

4.4.1 Implementation of Field Arithmetic

The inversion operation was implemented for two extension fields for which an OTF representation exists. The sizes of these finite fields were selected as representative for elliptic curve cryptosystems of medium and large size. In the first case the field $GF(q^{24}) = GF(q^{16})$ was selected, where the ground field is defined by the Pseudo-Mersenne prime $q = 2^{12} - 3 = 4093$, and the field elements are polynomials with 16 coefficients. Each coefficient is $n = \lceil \log_2 q \rceil = 12$ bits long, and hence every field element can be represented in 192 bits. The second field selected is $GF(q^{25}) = GF(q^{32})$, with $q = 2^{10} - 3 = 1021$. Each field element is now a polynomial with $m = 32$ coefficients of $n = 10$ bits each.

Performing an arithmetic operation in $GF(q)$ typically requires a regular integer operation followed by a modular reduction step. Using Pseudo-Mersenne primes as the field characteristic facilitate efficient modular reduction in the ground field. Often, when multiplying or squaring extension field elements, a number of coefficients of length $n = \lceil \log_2 q \rceil$ have to be multiplied and accumulated before the reduction step. Each multiplication results

in no more than a $2n$-bit product. During the computation of each coefficient of a polynomial multiplication, at most m of these products are accumulated. The lengths of intermediate results therefore exceed $2n$ by at most $e = \lceil \log_2 m \rceil$ bits, i.e. 4 bits for $GF(4093^{16})$ and 5 bits in the case of $GF(1021^{32})$. Therefore, it can safely be assumed that an intermediate result never exceeds $3n > 2n + e$ bits, and always fits into the 32 bits wide registers of the ARM processor.

An intermediate result a can be represented as a number in base 2^n, i.e. $a = (a_2 a_1 a_0)_{2^n} = a_2 2^{2n} + a_1 2^n + a_0$. Considering that $2^n \equiv c \pmod q$ and $2^{2n} \equiv c^2 \pmod q$ for a Pseudo-Mersenne prime $q = 2^n - c$ [16], the reduction can be performed simply as

$$\begin{aligned} a' &\equiv a \pmod q \\ &\equiv a_2 c^2 + a_1 c + a_0 \pmod q \\ &\equiv a'_1 2^n + a'_0 \pmod q \end{aligned}$$

where a' does not exceed $n + 4$ bits for $c = 3$, and in a second step

$$\begin{aligned} a'' &\equiv a' \pmod q \\ &\equiv a'_1 2^n + a'_0 \pmod q \\ &\equiv a'_1 c + a'_0 \pmod q. \end{aligned}$$

At most one further subtraction of the modulus is required to fully reduce the result to be in $GF(q)$.

The implementation of OTF inversion was optimized for speed by unrolling the recursion into a single section of linear code, and avoiding loops or additional function calls in order to reduce overhead. The same effort was put into optimizing the code for Itoh-Tsujii inversion, e.g. by loop unrolling and deferring modular reduction as much as possible.

The Multiply-Accumulate instruction, a feature commonly only found in DSP architectures, enables a particularly efficient implementation, since multiplications and additions constitute the majority of operations. Another benefit of the ARM architecture is that the latency of multiplication is data dependent. The ARM possesses a 32×8 bit multiplier, which requires between 2 and 7 clock cycles for a 32×32 bit operation, dependent on which bytes of the multiplier actually contain nonzero data. Since all coefficients fit into at most two bytes, less than the maximum number of clock cycles are necessary.

4.4.2 OEF Inversion using Itoh-Tsujii's Method

Inversion in Optimal Extension Fields was implemented using the method by Itoh and Tsujii [8]. Let $A \in GF(q^m)$. The algorithm to compute $B = A^{-1} \bmod P(x)$ works in four steps:

1. Compute the exponentiation A^{r-1} in $GF(q^m)$, where $r = \frac{q^m-1}{q-1}$
2. Compute the product $A^r = A \cdot A^{r-1}$
3. Compute the inversion $(A^r)^{-1}$
4. Compute the product $A^{r-1} \cdot (A^r)^{-1} = A^{-1}$

In Step 2, A^r is an element in the ground field $GF(q)$ [14], thus the inversion in Step 3 also takes place in $GF(q)$. For the exponentiation A^{r-1} in Step 1 the exponent is expanded as

$$r - 1 = \frac{q^m - 1}{q - 1} - 1 = q^{m-1} + q^{m-2} + \ldots + q^2 + q.$$

Exponentiation of A to the $(r-1)$-st power requires the computation of powers A^{q^i} for $1 \leq i \leq m-1$. This leads to an alternative representation of the exponent in basis q. For the field $GF(4093^{16})$ where $m = 16$ one can write

$$r - 1 = (1111\ 1111\ 1111\ 1110)_q$$

and for $GF(1021^{32})$ with 32 coefficients

$$r - 1 = (1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110)_q.$$

These exponentiations can be implemented efficiently using *Frobenius maps*. A^{q^i} is the i-th iterate of the Frobenius map defined as $\sigma(A) = A^q$. Applying an i-th iterate of the Frobenius map can be viewed as shifting the exponent to the left by i digits, e.g.

$$\sigma^4(A) = A^{q^4} = A^{(10000)_q}.$$

Consequently, for $m = 16$, one can perform exponentiation by $r - 1$ in the following steps:

1. $B_0 = A^q = A^{(10)_q}$ $\sigma(A)$
2. $B_1 = B_0 \cdot A = A^{(11)_q}$ Multiplication
3. $B_2 = B_1^{q^2} = A^{(1100)_q}$ $\sigma^2(A)$

4. $B_3 = B_2 \cdot B_1 = A^{(1111)_q}$ Multiplication
5. $B_4 = B_3^{q^4} = A^{(1111\ 0000)_q}$ $\sigma^4(A)$
6. $B_5 = B_4 \cdot B_3 = A^{(1111\ 1111)_q}$ Multiplication
7. $B_6 = B_5^{q^8} = A^{(1111\ 1111\ 0000\ 0000)_q}$ $\sigma^8(A)$
8. $B_7 = B_6 \cdot B_4 = A^{(1111\ 1111\ 1111\ 0000)_q}$ Multiplication
9. $B_8 = B_7 \cdot B_2 = A^{(1111\ 1111\ 1111\ 1100)_q}$ Multiplication
10. $B_9 = B_8 \cdot B_0 = A^{(1111\ 1111\ 1111\ 1110)_q}$ Multiplication

In the case of $m = 32$ the last three steps above are different and three additional steps are needed:

8. $B_7 = B_6 \cdot B_5 = A^{(1111\ 1111\ 1111\ 1111)_q}$ Multiplication
9. $B_8 = B_7^{q^{16}} = A^{(1111\ 1111\ 1111\ 1111\ 0000\ 0000\ 0000\ 0000)_q}$ $\sigma^{16}(A)$
10. $B_9 = B_8 \cdot B_6 = A^{(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0000\ 0000)_q}$ Multiplication
11. $B_{10} = B_9 \cdot B_4 = A^{(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0000)_q}$ Multiplication
12. $B_{11} = B_{10} \cdot B_2 = A^{(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100)_q}$ Multiplication
13. $B_{12} = B_{11} \cdot B_0 = A^{(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110)_q}$ Multiplication

Frobenius maps for OEFs can be implemented efficiently through the use of lookup tables for permutation and scaling of the coefficients [8],[2]. For each coefficient of a field element one table lookup determines the scaling factor by which that coefficient is multiplied. Another table lookup determines the new position of the coefficient after permutation.

For $GF(4093^{16})$ there are only four different Frobenius maps to be computed and field elements contain 16 coefficients, therefore the size of the lookup tables are sufficiently small for practical implementations. In the case of $GF(1021^{32})$ the memory requirements are slightly higher, albeit not significantly, since there are 32 coefficients and one additional lookup table for the 16^{th} order Frobenius map $\sigma^{16}(A)$.

The routines for multiplication of field elements have been implemented in a rather straightforward fashion. The choice of a binomial $P(x) = x^m - w$ as the irreducible field

polynomial proved helpful for fast modular reduction. Since $w = 2$, scaling by w can be performed by a simple shift operation. On the ARM platform shift operations can be performed simultaneously with other operations, due to the barrel-shifter that is integrated in the data path of one of the source operands.

4.4.3 OTF Inversion

OTF Inversion is based on recursively performing direct inversion through solving a system of linear equations, as described in Section 2.2. Before the recursion starts, the polynomial A has to be converted from OEF into OTF notation through a simple permutation of the coefficients.

Each level i of the recursion can be described in the following manner for $k \geq i \geq 1$. Note that for each level A_i is the input polynomial, and B_i the resulting inverse:

1. Split A_i into an upper and a lower part $A_{i,h}$ and $A_{i,l}$, which are elements of the underlying subfield
2. Compute the squares $A_{i,h}^2$ and $A_{i,l}^2$ in the subfield
3. Multiply $A_{i,h}^2$ with the root α_{i-1} of the subfield. This step can be easily integrated with Step 2, since it is simply a permutation of the coefficients and one single ground field multiplication with α_0
4. Compute the determinant of the linear system $\Delta_i = A_{i,l}^2 - \alpha_{i-1} A_{i,h}^2$
5. Compute the inverse Δ_i^{-1}, either by executing the next level of recursion, i.e. letting $A_{i-1} = \Delta_i$ and $\Delta_i^{-1} = B_{i-1}$, or by using a lookup table once the ground field is reached
6. Compute the upper and the lower part of the inverse $B_{i,l} = A_{i,l}\Delta_i^{-1}$ and $B_{i,h} = -A_{i,h}\Delta_i^{-1}$ by multiplication in the subfield and rejoin both halves of the polynomial to form the inverse B_i

It would be easier to see how this method works if the recursion was unrolled into a linear sequence of steps. The depth of recursion, k, is 4 and 5, for the fields $GF(4093^{16})$ and $GF(1021^{32})$, respectively. Steps 1 through 5 are executed k times, with Δ_i as the input to the next round, until the ground field is reached ($i = 1$). After the inverse Δ_1^{-1} has been

Field	$GF(4093^{16})$	$GF(1021^{32})$
OEF (Itoh-Tsujii)	23933	109259
OTF	3826	12780
Speedup	6.3	8.5

Table 4.3: Comparison between OTF and OEF inversion (clock cycles)

found through table lookup, Step 6 is repeated k times, working the way back up out of the recursion. Each Δ_i^{-1} is the previous round's result B_{i-1}.

Since the result of this procedure is still in OTF notation, one last permutation is necessary to bring it back into OEF representation. This, however is easily taken care of by simply storing the coefficients of the results in permuted order.

4.4.4 Performance Analysis and Impact on ECC Operations

Table 4.3 gives the execution time of the two inversion methods as implemented in number of clock cycles. It is clearly visible that OTF inversion is at least 6–8 times faster than OEF inversion. The basic operation of Elliptic Curve Cryptosystems (ECC), i.e. the scalar point multiplication, is typically implemented using the NAF method [20] in conjunction with EC point doubling and point addition. Recall the equations for point addition and point doublings using affine coordinates:

$$\lambda = \begin{cases} \frac{y_2-y_1}{x_2-x_1} \bmod P(x) \text{ ; Point Addition} \\ \\ \frac{3x_1^2+a}{2y_1} \bmod P(x) \text{ ; Point Doubling} \end{cases} \tag{4.2}$$

$$x_3 = \lambda^2 - x_1 - x_2 \bmod P(x) \tag{4.3}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \bmod P(x) \tag{4.4}$$

It can be seen that point addition requires one inversion, two multiplications, one squaring and six subtractions. Point doubling requires one inversion, two multiplications, two squarings and six additions/subtractions. Based on these requirements one can determine an estimate of how much faster an ECC implementation based on OTFs could be.

Table 4.4 shows the timings in clock cycles of the relevant OEF arithmetic operations, which, in conjunction with inversion, are used to implement ECC point addition and doubling. According to (4.2), (4.3) and (4.4), the time needed for an ECC point addition is

$$t_{ECADD} = t_{Inv} + 2t_{Mul} + t_{Sq} + 6t_{Add}$$

Operation	$GF(4093^{16})$	$GF(1021^{32})$
Multiplication	3099	11598
Squaring	1763	6270
Addition/Subtraction	313	556

Table 4.4: Timings for OEF arithmetic operations (clock cycles)

Field		$GF(4093^{16})$		$GF(1021^{32})$	
Inversion Method		OEF	OTF	OEF	OTF
ECADD	1 Inversion	23,933	3,826	109,259	12,780
	2 Multiplications	6,198		23,196	
	1 Squaring	1,763		6,270	
	6 Add / Sub	1,878		3,336	
ECADD cycles		33,772	13,665	142,061	45,582
ECDBL	1 Inversion	23,933	3,826	109,259	12,780
	2 Multiplications	6,198		23,196	
	2 Squarings	3,526		12,540	
	6 Add / Sub	1,878		3,336	
ECDBL cycles		35,535	15,428	148,331	51,852
# Point Doublings		192		320	
# Point Add/Sub		64		107	
Total clock cycles		8,984,128	3,836,736	62,666,447	21,469,914
Speedup factor		2.34		2.92	

Table 4.5: Clock cycles for ECC scalar point multiplication

and that for an ECC point doubling is

$$t_{ECDBL} = t_{Inv} + 2t_{Mul} + 2t_{Sq} + 6t_{Add} \ .$$

It is clear that inversion and multiplication/squaring are the dominant operations.

For an improved implementation of the ECC scalar point multiplication $\mathcal{Q} = k\mathcal{P}$, the integer k is typically transformed into non-adjacent form (NAF) [20]. The NAF representation uses signed digits and has the beneficial property that on average 2/3 of the digits are zeroes. Therefore, for an integer k of length $l = \lceil \log_2 k \rceil$, a scalar point multiplication requires on average l point doublings and $\frac{l}{3}$ point additions (or subtractions).

For an elliptic curve over $GF(4093^{16})$, where inversion is 6.3 times faster, a full scalar point multiplication with a 192 bit integer would require 192 point doublings and on average 64 point additions. Using OEF inversion, scalar point multiplication takes 8,984,128 clock cycles, while it only takes 3,836,736 cycles using OTF inversion. This corresponds to a speedup factor of 2.34.

For a curve over $GF(1021^{32})$ and scalar point multiplication with a 320 bit integer, where 320 point doublings and 107 point additions are necessary on average, similar results are obtained. With OEF inversion the scalar point multiplication takes 62,666,447 cycles. On the other hand, OTF inversion is 8.5 times faster and thus the computation of a scalar point multiplication only requires 21,469,914 cycles. This results in a speedup factor of 2.92, nearly three times faster than with OEF inversion. Table 4.5 summarizes the necessary operations and their timings in terms of clock cycles.

Acknowledgement:
I would like to acknowledge my colleague and friend Gunnar Gaubatz for his collaboration in performing the implementations described in this chapter. The timings shown in Tables 4.3, 4.4 and 4.5 were obtained as a result of our joint work.

Chapter 5

Conclusion

5.1 Summary and Conclusions

In this thesis a new tower field representation, Optimal Tower Fields, is introduced and a construction technique which establishes the conditions for their existence is outlined. It is also shown that OTF elements can be converted to OEF representation and back with a simple permutation. Thus, OTF operations (such as OTF inversion technique) are accessible from the OEF representation whenever a suitable OTF exists. This also means that cryptographic applications built over OTFs inherit the security characteristics of the ones built over OEFs.

Multiplication and squaring operations in OEFs are found to be slightly more efficient (in the number of additions) than the same operations in OTFs. However, since OEF operations are directly accessible to OTFs via a simple permutation conversion that comes for free, these two operations may be realized in OTFs with the same complexity as in OEFs.

The main advantage in using OTFs is in the recursive direct inversion method introduced. It is determined that OTF inversion is at least a few times more efficient than the OEF Itoh-Tsujii inversion technique, even when fast Frobenius maps apply. OTF inversion requires $m^2 + m - 2$ ground field multiplications, whereas an OTF multiplication may be realized by using m^2 ground field multiplications. Hence, for practical purposes OTF inversion may be considered to have the same complexity as OTF/OEF multiplication, assuming the ground field inversion may be realized efficiently. Furthermore, the asymptotic complexity of OTF inversion, i.e. $O(m^2)$, is surprisingly lower than the $O(m \log_2 m)$ complexity of Itoh-Tsujii inversion technique. By using the Karatsuba-Ofman algorithm an improved version of the OTF direct inversion algorithm is presented which achieves an even better $O(m^{\log_2 3})$

asymptotic complexity.

OTF inversion algorithm was implemented for the OEFs $GF(4093^{16})$ and $GF(1021^{32})$. The implementation results show that OTF inversion is at least 6 to 8 times faster than Itoh-Tsujii inversion algorithm for the selected fields. Furthermore, it is demonstrated that this speedup in inversion results in a 2 to 3 times improvement in the speed of an ECC scalar point multiplication. Hence, for the selected fields, an elliptic curve cryptosystem using OTF inversion technique runs at least 2 to 3 times faster than one that uses any other inversion algorithm.

5.2 Direction of Future Research

No weaknesses of using OEFs for elliptic curve cryptography have been found so far. The security of OTFs and OEFs are related to each other in the sense that OTFs are a special class of OEFs. Investigating the cryptanalytic properties of OTFs and showing how secure or insecure they are for usage in elliptic curve cryptography may be an interesting research topic.

In the construction of OTFs, binomials are used as irreducible polynomials. The availability of irreducible trinomials or irreducible polynomials of some other special type may be investigated for similar constructions.

Appendix

n	c	α_0	n	c	α_0	n	c	α_0	n	c	α_0
8	1	−5	9	−3	2	11	5	2	14	−3	−2
8	1	−3	9	−3	3	11	5	5	14	−3	2
8	1	3	10	−3	−2	12	−3	−5	16	1	−5
8	1	5	10	−3	2	12	−3	−2	16	1	−3
9	−3	−3	11	5	−5	12	−3	2	16	1	3
9	−3	−2	11	5	−2	12	−3	5	16	1	5

Table 5.1: $GF(q^{2^k})$ OTFs with $q = 2^n + c$, $7 \leq n \leq 16$; $-5 \leq c \leq 5$; $-5 \leq \alpha_0 \leq 5$.

n	c	α_0	n	c	α_0	n	c	α_0	n	c	α_0
7	−1	3	11	5	−4	12	3	2	14	−3	−4
7	−1	−3	11	5	−5	12	3	−2	14	−3	−5
10	−3	5	12	−3	5	12	3	−4	16	3	4
10	−3	−5	12	−3	4	12	3	−5	16	3	3
11	5	5	12	−3	2	14	−3	5	16	3	2
11	5	4	12	−3	−2	14	−3	4	16	3	−2
11	5	3	12	−3	−4	14	−3	3	16	3	−3
11	5	2	12	−3	−5	14	−3	2	16	3	−4
11	5	−2	12	3	5	14	−3	−2			
11	5	−3	12	3	4	14	−3	−3			

Table 5.2: $GF(q^{3^k})$ OTFs with $q = 2^n + c$, $7 \leq n \leq 16$; $-5 \leq c \leq 5$; $-5 \leq \alpha_0 \leq 5$.

m	$\mathcal{M}(m)$	$\mathcal{S}(m)$	$\mathcal{I}(m)$
8	$64\mathcal{M}_0 + 77\mathcal{A}_0$	$36\mathcal{M}_0 + 54\mathcal{A}_0$	$\mathcal{I}_0 + 70\mathcal{M}_0 + 84\mathcal{A}_0$
9	$81\mathcal{M}_0 + 92\mathcal{A}_0$	$45\mathcal{M}_0 + 68\mathcal{A}_0$	$\mathcal{I}_0 + 111\mathcal{M}_0 + 136\mathcal{A}_0$
16	$256\mathcal{M}_0 + 325\mathcal{A}_0$	$136\mathcal{M}_0 + 202\mathcal{A}_0$	$\mathcal{I}_0 + 270\mathcal{M}_0 + 355\mathcal{A}_0$
27	$729\mathcal{M}_0 + 884\mathcal{A}_0$	$378\mathcal{M}_0 + 536\mathcal{A}_0$	$\mathcal{I}_0 + 975\mathcal{M}_0 + 1244\mathcal{A}_0$
32	$1024\mathcal{M}_0 + 1333\mathcal{A}_0$	$528\mathcal{M}_0 + 762\mathcal{A}_0$	$\mathcal{I}_0 + 1054\mathcal{M}_0 + 1426\mathcal{A}_0$

Table 5.3: Number of $GF(q)$ operations for OTF arithmetic

m	$\mathcal{M}(m)$	$\mathcal{S}(m)$	$\mathcal{I}(m)$
6	$36\mathcal{M}_0 + 35\mathcal{A}_0$	$21\mathcal{M}_0 + 29\mathcal{A}_0$	$\mathcal{I}_0 + 140\mathcal{M}_0 + 110\mathcal{A}_0$
8	$64\mathcal{M}_0 + 63\mathcal{A}_0$	$36\mathcal{M}_0 + 48\mathcal{A}_0$	$\mathcal{I}_0 + 307\mathcal{M}_0 + 259\mathcal{A}_0$
9	$81\mathcal{M}_0 + 80\mathcal{A}_0$	$45\mathcal{M}_0 + 53\mathcal{A}_0$	$\mathcal{I}_0 + 293\mathcal{M}_0 + 248\mathcal{A}_0$
12	$144\mathcal{M}_0 + 143\mathcal{A}_0$	$78\mathcal{M}_0 + 98\mathcal{A}_0$	$\mathcal{I}_0 + 810\mathcal{M}_0 + 726\mathcal{A}_0$
16	$256\mathcal{M}_0 + 255\mathcal{A}_0$	$136\mathcal{M}_0 + 164\mathcal{A}_0$	$\mathcal{I}_0 + 1673\mathcal{M}_0 + 1545\mathcal{A}_0$
18	$324\mathcal{M}_0 + 323\mathcal{A}_0$	$171\mathcal{M}_0 + 203\mathcal{A}_0$	$\mathcal{I}_0 + 1758\mathcal{M}_0 + 1632\mathcal{A}_0$
24	$576\mathcal{M}_0 + 575\mathcal{A}_0$	$300\mathcal{M}_0 + 344\mathcal{A}_0$	$\mathcal{I}_0 + 4264\mathcal{M}_0 + 4048\mathcal{A}_0$
27	$729\mathcal{M}_0 + 728\mathcal{A}_0$	$378\mathcal{M}_0 + 404\mathcal{A}_0$	$\mathcal{I}_0 + 4610\mathcal{M}_0 + 4394\mathcal{A}_0$
32	$1024\mathcal{M}_0 + 1023\mathcal{A}_0$	$528\mathcal{M}_0 + 588\mathcal{A}_0$	$\mathcal{I}_0 + 8535\mathcal{M}_0 + 8215\mathcal{A}_0$
36	$1296\mathcal{M}_0 + 1295\mathcal{A}_0$	$666\mathcal{M}_0 + 734\mathcal{A}_0$	$\mathcal{I}_0 + 9424\mathcal{M}_0 + 9100\mathcal{A}_0$

Table 5.4: Number of $GF(p)$ operations for OEF arithmetic

m	$\mathcal{M}(m)$	$\mathcal{S}(m)$	$\mathcal{I}(m)$
8	$64\mathcal{M}_0 + 63\mathcal{A}_0$	$36\mathcal{M}_0 + 48\mathcal{A}_0$	$\mathcal{I}_0 + 70\mathcal{M}_0 + 80\mathcal{A}_0$
9	$81\mathcal{M}_0 + 80\mathcal{A}_0$	$45\mathcal{M}_0 + 53\mathcal{A}_0$	$\mathcal{I}_0 + 111\mathcal{M}_0 + 136\mathcal{A}_0$
16	$256\mathcal{M}_0 + 255\mathcal{A}_0$	$136\mathcal{M}_0 + 164\mathcal{A}_0$	$\mathcal{I}_0 + 270\mathcal{M}_0 + 311\mathcal{A}_0$
27	$729\mathcal{M}_0 + 728\mathcal{A}_0$	$378\mathcal{M}_0 + 404\mathcal{A}_0$	$\mathcal{I}_0 + 975\mathcal{M}_0 + 1091\mathcal{A}_0$
32	$1024\mathcal{M}_0 + 1023\mathcal{A}_0$	$528\mathcal{M}_0 + 588\mathcal{A}_0$	$\mathcal{I}_0 + 1054\mathcal{M}_0 + 1166\mathcal{A}_0$

Table 5.5: Number of $GF(q)$ operations for OTF arithmetic with OEF multiplication and squaring

m	$\mathcal{M}(m)$	$\mathcal{S}(m)$	$\mathcal{I}(m)$
8	$27\mathcal{M}_0 + 108\mathcal{A}_0$	$27\mathcal{M}_0 + 72\mathcal{A}_0$	$\mathcal{I}_0 + 52\mathcal{M}_0 + 120\mathcal{A}_0$
9	$36\mathcal{M}_0 + 149\mathcal{A}_0$	$36\mathcal{M}_0 + 95\mathcal{A}_0$	$\mathcal{I}_0 + 84\mathcal{M}_0 + 217\mathcal{A}_0$
16	$81\mathcal{M}_0 + 365\mathcal{A}_0$	$81\mathcal{M}_0 + 269\mathcal{A}_0$	$\mathcal{I}_0 + 160\mathcal{M}_0 + 489\mathcal{A}_0$
27	$216\mathcal{M}_0 + 1031\mathcal{A}_0$	$216\mathcal{M}_0 + 788\mathcal{A}_0$	$\mathcal{I}_0 + 516\mathcal{M}_0 + 1919\mathcal{A}_0$
32	$243\mathcal{M}_0 + 1176\mathcal{A}_0$	$243\mathcal{M}_0 + 936\mathcal{A}_0$	$\mathcal{I}_0 + 484\mathcal{M}_0 + 1774\mathcal{A}_0$

Table 5.6: Number of $GF(q)$ operations for OTF arithmetic with Karatsuba

n	c	a	n	c	a	n	c	a	n	c	a
7	−1	3	12	−3	5	12	3	−4	16	3	2
11	5	5	12	−3	2	12	3	−5	16	3	−4
11	5	2	12	−3	−2	14	−3	2			
11	5	−2	12	−3	−5	14	−3	−2			
11	5	−5	12	3	2	16	3	3			

Table 5.7: $GF((p^{3^2})^2)$ generalized OTFs with irreducible binomial $P(x) = x^3 - a$ for constructing $GF(p^3)$, $p = 2^n + c$, $7 \leq n \leq 16$; $-5 \leq c \leq 5$; $-5 \leq a \leq 5$.

n	c	a	n	c	a	n	c	a	n	c	a
11	5	5	11	5	−5	12	−3	−2	14	−3	−2
11	5	2	12	−3	5	12	−3	−5			
11	5	−2	12	−3	2	14	−3	2			

Table 5.8: $GF((p^3)^{2^k})$ generalized OTFs with irreducible binomial $P(x) = x^3 - a$ for constructing $GF(p^3)$, $p = 2^n + c$, $7 \leq n \leq 16$; $-5 \leq c \leq 5$; $-5 \leq a \leq 5$.

n	c	a	n	c	a	n	c	a	n	c	a
11	5	−5	11	5	5	12	−3	2	14	−3	2
11	5	−2	12	−3	−5	12	−3	5			
11	5	2	12	−3	−2	14	−3	−2			

Table 5.9: $GF((q^{3^2})^{2^k})$ generalized OTFs with irreducible binomial $P(x) = x^3 - a$ for constructing $GF(q^3)$, $q = 2^n + c$, $7 \leq n \leq 16$; $-5 \leq c \leq 5$; $-5 \leq a \leq 5$.

m	$\mathcal{M}(m)$	$\mathcal{S}(m)$	$\mathcal{I}(m)$
6	$36\mathcal{M}_0 + 39\mathcal{A}_0$	$21\mathcal{M}_0 + 31\mathcal{A}_0$	$\mathcal{I}_0 + 42\mathcal{M}_0 + 48\mathcal{A}_0$
12	$144\mathcal{M}_0 + 169\mathcal{A}_0$	$78\mathcal{M}_0 + 114\mathcal{A}_0$	$\mathcal{I}_0 + 156\mathcal{M}_0 + 195\mathcal{A}_0$
18	$324\mathcal{M}_0 + 387\mathcal{A}_0$	$171\mathcal{M}_0 + 247\mathcal{A}_0$	$\mathcal{I}_0 + 363\mathcal{M}_0 + 466\mathcal{A}_0$
24	$576\mathcal{M}_0 + 701\mathcal{A}_0$	$300\mathcal{M}_0 + 422\mathcal{A}_0$	$\mathcal{I}_0 + 600\mathcal{M}_0 + 774\mathcal{A}_0$
36	$1296\mathcal{M}_0 + 1585\mathcal{A}_0$	$666\mathcal{M}_0 + 918\mathcal{A}_0$	$\mathcal{I}_0 + 1353\mathcal{M}_0 + 1753\mathcal{A}_0$

Table 5.10: Number of ground field operations for generalized OTF arithmetic

m	$\mathcal{M}(m)$	$\mathcal{S}(m)$	$\mathcal{I}(m)$
6	$36\mathcal{M}_0 + 35\mathcal{A}_0$	$21\mathcal{M}_0 + 29\mathcal{A}_0$	$\mathcal{I}_0 + 42\mathcal{M}_0 + 48\mathcal{A}_0$
12	$144\mathcal{M}_0 + 143\mathcal{A}_0$	$78\mathcal{M}_0 + 98\mathcal{A}_0$	$\mathcal{I}_0 + 156\mathcal{M}_0 + 183\mathcal{A}_0$
18	$324\mathcal{M}_0 + 323\mathcal{A}_0$	$171\mathcal{M}_0 + 203\mathcal{A}_0$	$\mathcal{I}_0 + 363\mathcal{M}_0 + 412\mathcal{A}_0$
24	$576\mathcal{M}_0 + 575\mathcal{A}_0$	$300\mathcal{M}_0 + 344\mathcal{A}_0$	$\mathcal{I}_0 + 600\mathcal{M}_0 + 678\mathcal{A}_0$
36	$1296\mathcal{M}_0 + 1295\mathcal{A}_0$	$666\mathcal{M}_0 + 734\mathcal{A}_0$	$\mathcal{I}_0 + 1353\mathcal{M}_0 + 1483\mathcal{A}_0$

Table 5.11: Number of ground field operations for generalized OTF arithmetic with OEF multiplication and squaring

m	$\mathcal{M}(m)$	$\mathcal{S}(m)$	$\mathcal{I}(m)$
6	$18\mathcal{M}_0 + 67\mathcal{A}_0$	$18\mathcal{M}_0 + 40\mathcal{A}_0$	$\mathcal{I}_0 + 36\mathcal{M}_0 + 66\mathcal{A}_0$
12	$54\mathcal{M}_0 + 232\mathcal{A}_0$	$54\mathcal{M}_0 + 160\mathcal{A}_0$	$\mathcal{I}_0 + 108\mathcal{M}_0 + 287\mathcal{A}_0$
18	$108\mathcal{M}_0 + 493\mathcal{A}_0$	$108\mathcal{M}_0 + 358\mathcal{A}_0$	$\mathcal{I}_0 + 228\mathcal{M}_0 + 715\mathcal{A}_0$
24	$162\mathcal{M}_0 + 757\mathcal{A}_0$	$162\mathcal{M}_0 + 577\mathcal{A}_0$	$\mathcal{I}_0 + 324\mathcal{M}_0 + 1084\mathcal{A}_0$
36	$324\mathcal{M}_0 + 1570\mathcal{A}_0$	$324\mathcal{M}_0 + 1246\mathcal{A}_0$	$\mathcal{I}_0 + 660\mathcal{M}_0 + 2436\mathcal{A}_0$

Table 5.12: Number of ground field operations for generalized OTF arithmetic when Karatsuba-Ofman algorithm is used

Bibliography

[1] D. V. Bailey and C. Paar. Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume LNCS 1462, pages 472–485, Berlin, Germany, 1998. Springer-Verlag.

[2] D. V. Bailey and C. Paar. Efficient Arithmetic in Finite Field Extensions with Application in Elliptic Curve Cryptography. *Journal of Cryptology*, 14(3):153–176, 2001.

[3] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, London Mathematical Society Lecture Notes Series 265, 1999.

[4] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, 1976.

[5] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[6] S. T. J. Fenn, M. Benaissa, and D. Taylor. Finite Field Inversion Over the Dual Base. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4(1):134–136, March 1996.

[7] W. Geiselmann and D. Gollmann. Self-Dual Bases in F_{q^n}. *Designs, Codes and Cryptography*, 3:333–345, 1993.

[8] J. Guajardo and C. Paar. Itoh-Tsujii Inversion in Standard Basis and Its Application in Cryptography. *Design, Codes, and Cryptography*, (25):207–216, 2002.

[9] M. A. Hasan. Double-Basis Multiplicative Inversion Over $GF(2^m)$. *IEEE Transactions on Computers*, 47(9):960–970, September 1998.

[10] I. S. Hsu, T. K. Truong, L. J. Deutsch, and I. S. Reed. A Comparison of VLSI Architecture of Finite Field Multipliers Using Dual-, Normal-, or Standard Bases. *IEEE Transactions on Computers*, 37(6):735–739, June 1988.

[11] T. Itoh and S. Tsujii. A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases. *Information and Computation*, 78:171–177, 1988.

[12] A. Karatsuba and Y. Ofman. Multiplication of Multidigit Numbers on Automata. *Sov. Phys. Dokl. (English translation)*, 7(7):595–596, 1963.

[13] Ç. K Koç and T. Acar. Montgomery Multplication in $GF(2^k)$. *Design, Codes, and Cryptography*, 14(1):57–69, 1998.

[14] R. Lidl and H. Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, Reading, Massachusetts, USA, 1983.

[15] E. D. Mastrovito. *VLSI Architectures for Computation in Galois Fields*. PhD thesis, Linköping University, Department of Electrical Engineering, Linköping, Sweden, 1991.

[16] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, Florida, USA, 1997.

[17] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[18] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. Fast Key Exchange with Elliptic Curve Systems. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO '95*, volume LNCS 963, pages 43–56, Berlin, Germany, 1995. Springer-Verlag.

[19] David Seal. *ARM Architecture Reference Manual*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, second edition, 2000.

[20] D. R. Stinson. *Cryptography, Theory and Practice*. Chapman & Hall/CRC, Boca Raton, Florida, USA, second edition, 2002.

[21] B. Sunar. *Fast Galois Field Arithmetic for Elliptic Curve Cryptography and Error Control Codes*. PhD thesis, Department of Electrical & Computer Engineering, Oregon State University, Corvallis, Oregon, USA, November 1998.

[22] A. Woodbury, D. V. Bailey, and C. Paar. Elliptic Curve Cryptography on Smart Cards Without Coprocessors. In *IFIP CARDIS 2000, Fourth Smart Card Research and Advanced Application Conference*, Bristol, UK, September 20–22 2000. Kluwer.