Performance Comparism Of Finite Fields Arithmetic In Elliptic Curve Based Cryptographic Schemes.

Aliyu Danladi Hina

Abstract
Finite fields are well studied discrete structures with a vast array of useful properties and are indispensable in the theory and application of cryptography. Arithmetic in finite field is an integral part of many public key algorithms. The performance of elliptic curve based schemes depends on the efficient arithmetic in the underlying field. ; Cryptography is one of the most prominent application areas of finite field arithmetic. Most of public-key cryptographic algorithms including the recent algorithms such as elliptic curve and pairing-based cryptography rely heavily on finite field arithmetic, which needs to be performed efficiently to meet the execution speed and design space constraints. These objectives constitute massive challenges that necessitate research efforts that will render the best algorithms, architectures, implementations, and design practices. This paper aims to provide a concise perspective for efficient finite field arithmetic in the most widely used finite field for usage in cryptography, The Optimal Extension Field.

Key words: cryptography, discrete structures, elliptic curve, Finite field, finite field arithmetic.

1.0 INTRODUCTION

To implement an ECC, one must select an underlying finite field in which to perform arithmetic calculations. A finite field is identified with the notation $GF(p^m)$ for p a prime and m a positive integer. It is well known that there exists a finite field for all primes p and positive integers m. Any such field is isomorphic to $GF(p)[x]/(P(x))$, where $P(x) = x^m + \sum_{i=0}^{m-1} p_i x_i,\ p_i \epsilon GF(p)$, is a monic-irreducible polynomial of degree m over $GF(p)$. In the following, each residue class will be identified with the unique polynomial of least degree in this class.
Various finite fields admit the use of different algorithms for arithmetic. Unsurprisingly, the choices of *p*, *m*, and *P(x)* can have a dramatic impact on the performance of the ECC. In particular, there are generic algorithms for arithmetic in an arbitrary finite field and there are specialized algorithms which provide better performance in finite fields of a particular form. In the following, we briefly describe field types proposed for ECC. The basic requirement for a fast and thus energy efficient implementation of ECC is a very fast multiplication in the prime field. The fastest known implementation was implemented by SUN Microsystems. [5]

2.0 FINITE FIELDS

Various finite fields admit the use of different algorithms for arithmetic. The choice of p, m and p(x) can have a dramatic impact on the performance of the elliptic curve cryptography (ECC). There are generic algorithms in an arbitrary field and there are specialized algorithms which provide better performance in a finite field of a particular form.

2.1 Binary Fields *GF(2^m)*: The finite field GF(2^m) called a binary finite field of 2^m elements implying that there exist a set of m elements $\{a_0, a_1, a_2, \dots a_{m-1}\}$ in $GF(2^m)$ such that each $\alpha \epsilon GF(2^m)$ can be written in the form $\alpha = \sum_{i=0}^{m-1} \alpha_i a_i$ where $\alpha_i \epsilon \{0,1\}$.
Implementing the binary field in designing elliptic curve based schemes, one often choose p = 2 and P(x) to be a trinomial or pentanomial. Such choices of irreducible polynomial lead to efficient methods for extension field modular reduction. We will refer to this type of field as a binary field, The elements of the subfield GF(2) can be represented by the logical signals 0 and 1. In this way, it is possible to construct fast and area efficient finite field arithmetic. Binary fields are also popular for software implementations of ECC. Many authors have suggested the use of p = 2 and m a composite number, In this case, the field GF(2^m) is isomorphic to $GF((2^s)^r)$, for m = sr and we call this a composite field.

2.2 Binary Composite Fields: An extension defined over a subfield of *GF(2^k)* is known as a composite field denoted by *GF((2^n)^m)*. Considering the fact that both binary and composite fields $GF((2^n)^m)$ refer to same field, efficient implementation can be obtained for composite fields, since this field provides efficient implementations for specific operations such as multiplication, inversion and exponentiation.
The composite field has the advantage that its operations are computed using arithmetic in the subfield GF(2^n) and the operations in the subfield can be efficiently performed by index table look-up if n is too large [3]. Thus instead of performing the computation in the binary field, it is more efficient to implement the composite field to perform the computations. This approach can provide superior performance when compared to the case of binary fields. However, a recent attack against ECCs over composite fields makes their use in practice questionable.

2.3 Prime Fields: Prime fields, GF(p^m) where m = 1 are perhaps the most obvious finite fields to use. For ECC, a typical prime is chosen to be larger than 2^{160}, and must be stored in multiple computer words. The problem with this representation is that during computation, the carries between words must be propagated, and the reduction modulo p must be performed over several words. There has been a large amount of research dealing with methods for doing long-number multi-precision arithmetic efficiently. Perhaps the most popular method in this context is based on Montgomery reduction.

2.4 Optimal Extension Fields: An Optimal Extension Field (OEF) is a finite field F(p^n) such that:

(1) p is a pseudo-Mersenne prime.

(2) An irreducible binomial $p(x) = x^n - \omega$ exists over Fp.

There are two types of OEF which yield additional arithmetic advantages a Type I OEF which has $p = 2^n \mp 1$ allows for subfield modular reduction with very low complexity and a Type II OEF which has an irreducible binomial $x^m - 2$ permits for a reduction in the complexity of extension field modular reduction.

Optimal extension fields $GF((2^n \mp c)^m)$ offer considerable computational advantages by selecting p and m specifically to match the underlying hardware use to perform the required arithmetic.

An alternative construction is to use optimal extension fields (OEFs), defined as follows. Choose p of the form $2^n \mp c$, for n; c arbitrary positive integers, where $\log_2(c) \leq \frac{1}{2}n$. In this case, one chooses p of appropriate size to use the multiply instructions available on the target platform. In addition, m is chosen so that an irreducible binomial $P(x) = x^m - \omega$, *exists*, $\omega \epsilon GF(p)$. Finite field arithmetic in extension fields is greatly influenced by the choice of basis in addition to the choice of p, m and $p(x)$. Of the proposed bases for application namely: standard (polynomial) basis, normal basis and other basis (dual basis, triangular basis etc), the polynomial basis is highly recommended for implementation in elliptic curve cryptosystems. The representation of optimal extension field's elements utilizes the polynomial basis. An element $A \epsilon\, GF(p^m)$ is represented as $A = \sum_{i=0}^{m-1} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_{m-1} x^{m-1}$ *where* $a_i \epsilon\, GF(p)$.

3.0 RELEVANT ALGORITHMS

The key to a successful and efficient implementation of a cryptosystem is the choice of algorithms to optimize the arithmetic. While a multitude of algorithms exist, it is important to carefully choose the best combination. In this chapter, we will discuss the primary algorithms used in this implementation.

3.1 Karatsuba Multiplication: Extension field multiplication is the most costly basic arithmetic function in OEFs. For a given extension field of order n, n^2 subfield multiplications are required to multiply two values using traditional polynomial multiplication. It is shown in [29] that this can be reduced drastically in certain cases. Using a method developed by Karatsuba and Ofman, the number of multiplications can be reduced in exchange for an increased number of additions. As long as the time ratio for executing a multiplication vs. an addition is high, this tradeoff is more efficient.

A basic example of Karatsuba is given here to demonstrate its usefulness.

Given two degree-1 polynomials, A(x) and B(x), we can demonstrate the traditional and the Karatsuba methods.

$$A(x) = a_1x + a_0$$
$$B(x) = b_1x + b_0$$

For the traditional method, we must calculate the product of each possible pair of coefficients.

$$D_0 = a_0b_0$$
$$D_1 = a_0b_1$$
$$D_2 = a_1b_0$$
$$D_3 = a_1b_1$$

Now we can calculate the product

$C(x) = A(x) * B(x)$ as: $C(x) = D_3x_2 + (D_2 + D_1)x + D_0$

The Karatsuba method begins by taking the same two polynomials, and calculating the following three products:

$$E_0 = a_0b_0$$
$$E_1 = a_1b_1$$
$$E_2 = (a_0 + a_1)(b_0 + b_1)$$

These are then used to assemble the result C(x) = A(x) * B(x):

$C(x) = E_1x_2 + (E_2 - E_1 - E_0)x + E_0$

It is easy to verify the results are equal. We can now look at how many operations are required for each method. The traditional method requires four multiplications and one addition, while the Karatsuba method requires three multiplications and four additions. Thus we have traded a single multiplication for three additions. If the cost to multiply on the target platform is as least three times the cost to add, then the method is effective. While this basic form of Karatsuba was presented in the original paper, there are a number of ways this method may be expanded to handle larger degree polynomials. This therefore indicates that the Karatsuba algorithm is found to be

3.2 Itoh-Tsujii Inversion: Extension field inversion is normally a costly operation, but the nature of OEFs allows the reduction of the extension field inversion to a subfield inversion. The Itoh-Tsujii algorithm which was originally developed for use with composite fields $GF(2^{n^m})$ in a normal basis representation can be applied to extension fields $GF(q^m)$ in polynomial representation. It is assumed that the subfield inverse can be calculated by efficient means, such as table-lookup or the Euclidean algorithm, given a small order of the subfield. To perform the OEF inversion, we use the following expression:

$$A^{-1} = (A^r)^{-1}A^{r-1}, \qquad where\ r = \frac{q^m - 1}{q - 1}$$

The Algorithm below shows the general case for inversion. It is key to observe that $A^r \epsilon\, GF(q)$. Since r is known ahead of time, an efficient addition chain for the exponentiation in Step 1 can be pre-computed and hardcoded into the algorithm.

Algorithm 1: General Inversion Algorithm in $GF(q^m)$

Require: $A \,\epsilon\, GF(q^m)$

Ensure: $C \equiv A^{-1} mod p(x)$

```
1: $B \leftarrow A^{r-1}$ (using an addition chain)
2: $b \leftarrow BA = A^{r-1}A = A^{r} \epsilon GF(q)$
3: $b \leftarrow b^{-1} = (A^{r})^{-1}$
4: $C \leftarrow bB = (A^{r})^{-1}A^{r-1} = A^{r-1}$
```

can be formed utilizing $[log_2(m-1)] + W_H(m-1) - 1$ extension field multiplications,
where $W_H(m-1)$ denotes the Hamming weight.
To further reduce the complexity, we utilize the Frobenius map to compute the exponentiations of A occurring in the addition chain. As shown in [1], for an OEF with a binomial field polynomial, the pth iteration of the Frobenius map requires at most m-1 multiplications in *GF(q)*, this therefore shows that the stress of inversions in optimal extension fields has be reduced to mere subfield inversion in which the norm function maps the elements of the extension field to the subfield by raising them to a certain power [16].

3.3 de Rooij Point Multiplication: The primary operation in an elliptic curve cryptosystem is point multiplication, *C = kP*. For large *k*, computing *kP* is a costly endeavor. However, well-studied techniques used for ordinary integer exponentiation can be advantageously adapted to this Setting. The most basic of these algorithms is the binary-double-and-add algorithm [29]. It has a complexity of $log_2(k) + W_H(k)$ group operations, where W_H is the Hamming weight of the multiplier k. On average, then, we can expect this algorithm to require 1:5 $log_2(k)$ group operations. Using more advanced methods, such as signed digit, k-ary or sliding window, the complexity may be reduced to approximately 1:2 $log_2(k)$ group operations on average. The situation is much better in applications where the point is known ahead of time. The most common public-key operation for a smart card or PDA is to provide a digital signature. The ECDSA algorithm involves the multiplication of a fixed curve point by the user-generated private key as the core operation. Because the curve point is known ahead of time, pre-computations may be performed to expedite the signing process. Using a method devised by de Rooij in [dR98], we are able to reduce the number of group operations necessary by a factor of four over the binary-double-and-add algorithm. The de Rooij algorithm is a variant of that devised by Brickell, Gordon, McCurley, and Wilson [BGMW93], but requires far fewer pre-computations. A modified form of de Rooij is shown in Algorithm 2. Note that the step shown in line 10 requires general point multiplication of AM by q, where 0 · q < b. This is accomplished using the binary-double-and-add algorithm. In [21], the author remarks that during execution, q is rarely greater than 1. The choice of t and b are very important to the operation of this algorithm. They are defined such that $b^{t+1} \geq \#E(GF(p^m))$. The algorithm must be able to handle a multiplier, s, not exceeding the order of the elliptic curve. The number of point pre-computations and temporary storage locations is determined by t + 1, while b represents the maximum size of the exponent words. Thus we need to find a compromise between the two parameters.

Algorithm 2: Elliptic Curve Fixed Point Multiplication

```
Require: $\{b^0A; b^1A; : : : ; b^tA\}, A \epsilon E(GF(p^m))$, and $s = \sum_{i=0}^{t} s_i b^i$
Ensure: $C = sA, C \epsilon E(GF(p^m))$
1: Define $M \epsilon [0; t]$ such that $z_M \geq z_i$ for all $0 \leq i \leq t$
2: Define $N \epsilon [0; t]; N \neq M$ such that $z_M \geq z_i$ for all $0 \leq i \leq t; i \neq M$
3: for $i \leftarrow 0$ to $t$ do
4:     $Ai \leftarrow biA$
5:     $zi \leftarrow si$
6: end for
7: Determine M and N for $\{z0; z1; : : : ; zt\}$
8: while $z_N \geq 0$ do
9:     $q \leftarrow [z_M/z_N]$
10:    $A_N \leftarrow qA_M + A_N$    general point multiplication
11:    $z_M \leftarrow z_M \bmod z_N$
12:    Determine M and N for $\{z0; z1; : : : ; zt\}$
13: end while
14: $C \leftarrow z_M A_M$
```

CONCLUSION

The underlying finite field is an important instrument in the efficiency of elliptic curve cryptosystems. The merits of optimal extension fields over other finite fields were demonstrated. Multiplication and squaring operations in OEFs were found to be slightly more efficient. Algorithms based on these theories and several techniques used to speed up the optimal extension field arithmetic were explained. This therefore shows that optimal extension fields are specially attractive for use in an elliptic curve based cryptographic scheme.

Referrences

[1] D. V. Bailey and C. Paar. Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. *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 *Fqn*. *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] C¸ . K Ko¸c 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, Link¨oping University, Department of Electrical Engineering, Link¨oping, 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. *BIBLIOGRAPHY*
52

[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.

Author Details:
Aliyu Danladi Hina
DEPT OF PRE-ND,
SCHOOL OF GENERAL STUDIES,
THE FEDERAL POLYTECHNIC BAUCHI,
NIGERIA.
E-mail: leeyo17@yahoo.com