University of Bristol

DEPARTMENT OF COMPUTER SCIENCE

A comparison of different finite fields for use in elliptic curve cryptosystems

N. P. Smart

June 2000 | CSTR-00-007

A COMPARISON OF DIFFERENT FINITE FIELDS FOR USE IN ELLIPTIC CURVE CRYPTOSYSTEMS

N.P. SMART

Abstract. We examine the relative efficiency of four methods for finite field representation in the context of elliptic curve cryptography (ECC). We conclude that a set of fields called the Optimized Extension Fields (OEFs) give greater performance, even when used with affine coordinates, when compared against the type of fields recommended in the emerging ECC standards. Although this performance advantage is only marginal and hence there is probably no need to change the current standards to allow OEF fields in standards compliant implementations.

1. Introduction

The efficient implementation of arithmetic in finite fields is crucial for the high performance of various cryptographic algorithms, such as those based on the difficulty of the discrete logarithm problem in finite fields, elliptic curves or hyperelliptic curves. In all of these schemes the efficiency of the underlying finite field operations is the dominant performance constraint, any effort spent optimizing the field operations is well spent. This has led a number of special choices of field to be used, each with its own performance characteristics. Due to different engineering constraints such as processor type, memory requirements etc there is no correct answer to the question: Which field should one use ?

In this paper we look in more detail at the choice of finite fields in the case of elliptic curve based systems. It is important when comparing one parameter choice against another that we use real world parameter choices and we look not only at the performance of the underlying field arithmetic but also at the performance of the overall cryptographic protocols. This is important since some cryptographic algorithms do not make use of general arithmetic but only require careful optimization of crucial parts. This is particularly true of modular exponentiation based systems where it makes more sense to spend a lot of time optimizing the squaring operation as opposed to the general multiplication operation. A similar situation holds for elliptic curves where one needs to optimize the point doubling operation more than the general point addition operation.

In this paper we will concentrate on the choice of finite field and not consider the use of special curves, such as the so-called Koblitz curves, which can provide performance advantages. Hence our conclusions will not be effected by security considerations as long as the overwhelming majority of curves over the given fields are considered to be secure.

It is also important to compare like for like, for example if one system uses machine code for the main arithmetics whilst the one being compared against uses no machine code then the comparison is not really fare. In addition one should only

Key words and phrases. finite fields, elliptic curves, cryptography.

compare algorithms using a single computer. Even comparisons on almost identical processors can lead to different conclusions. For example in fields of characteristic two arithmetic is often implemented via lookup tables, hence algorithms will behave differently on two processors which are identical bar the fact that one has a faster access time for its cache. Also comparing two systems which are programmed by different people one could be comparing programming ability rather than actual performance.

Often these considerations are ignored since complexity theory tells us that implementation details should not matter and that it is the complexity of the algorithm itself which should determine the relative merits. But complexity arguments only hold in the limit, which may not be applicable at the problem size under consideration. For example when multiplying two integers, complexity theory tells us that Fourier transform techniques or Karatsuba multiplication will work faster than school book multiplication, but when multiplying 200 bit numbers it is far more efficient in practice to use a school book multiplication algorithm.

Various other authors have compared implementation details for elliptic curve systems, but we would contend that such comparisons are to be taken with a pinch of salt. For example Bailary and Paar [5] compare their OEF based elliptic curve implementation against other peoples implementations of characteristic two fields of composite degrees. This is bad practice for a number of reasons. Firstly composite fields of characteristic two are not recommended for use in cryptographic standards (a view which has been reinforced by recent work in [10]). Secondly the authors of [5] have a reason to prefer OEF fields since they are putting them forward as a replacement for standard implementations, although in our comparison we give independent verification of their conclusion that OEF fields offer significant advantages. Thirdly the comparison was performed on 64 bit archetectures, which although these are now common in high end workstations, are not likely to be common in the small devices which are the target for elliptic curve based systems. Our comparison on a 32 bit RISC system is likely to be more indicative since such processors, like the StrongArm, are likely to be used in a number of such devices in the comming years.

In another another comparison in [9] the authors give a comparison between even characteristic fields and odd characteristic fields, the later being implemented using Barrett reduction. This is slightly flawed since standards compliant elliptic curve systems in odd characteristic are more likely to be implemented over fields defined by a Generalized Mersenne prime which provide different performance characteristics than general primes.

The main reason for our interest was to compare the new idea of OEF fields with the fields defined by GM-primes which occur in the standards documents. Hence all our implementations were coded from scratch and shared many core subprocedures. A similar length of time was spent optimizing each implementation and so the relative differences in performance should be indicative of completely optimized implementations. Although the resulting comparisons are not completely scientific we do hope that they are on a better foundation than previous ones. Hence hopefully they can be used to make further commerical considerations as to which fields are to be preferred.

We end this introduction with a caveat that our timings where performed on a 32 bit RISC processor (namely one of the Sparc family). This is probably indicative of

other RISC type processors but for smaller processors used in Smart cards, mobile phones and set top boxes different conclusions may possibly arise.

2. The Choices for Fields

Currently there are a number of field choices for elliptic curve systems that are mentioned in the literature. These can be divided into two classes:

2.1. **Prime Fields.**

Fields of large prime characteristic, $\mathbb{F}_p$, are very popular since they can be efficiently implemented using techniques borrowed from other finite field based cryptographic systems such as DSA and RSA. However using standard modular arithmetic is not very efficient since multi-precision remaindering operations are very expensive. Hence when used in elliptic curve systems there are various choices that are often made:

General Primes For general primes the most efficient implementation technique is almost always to use Montgomery Arithmetic, [15]. Although the authors of [9] use Barrett reduction the timing difference between Montgomery arithmetic and Barrett reduction is usually comparable. Montgomery arithmetic uses a special representation to perform efficient arithmetic, the division and remaindering essentially being performed by bit shifting. We do not cover this arithmetic here since it is covered in a number of text books e.g. [7] and [14].

Generalized Mersenne Primes Certain primes are highly suited for efficient reduction techniques, the most simple form of such primes being the Mersenne primes, which are primes of the form $p = 2^k - 1$. However the number of Mersenne primes of the correct size for cryptography is limited. This has led a number of authors to propose generalizations on the Mersenne primes.

Crandall [8] proposed the use of primes of the form $p = 2^k - c$ where c is a small integer, which is usually chosen to fit into a single word. Primes of the form $p = 2^k \pm c$ for a small value of c (in comparison to 2^k) are often called pseudo-Mersenne primes.

In another direction Solinas [16] introduced the concept of Generalized Mersenne Primes (GM-primes) which are primes of the form

$$p = f(2^k)$$

where f is a polynomial of small degree and weight and k is a multiple of the computer word size.

The use of GM-primes has become popular due to the adoption of these primes in the recommend curves in standards from such bodies as ANSI [1], NIST [2], SECG [3], and WAP [4]. As an example we take the following example, from Solinas' paper,

$$f(t) = t^3 - t - 1$$

then we obtain the 192-bit prime

$$p = 2^{192} - 2^{64} - 1 = f(2^{64}).$$

Reducing the result of a multi-precision multiplication is then a simple matter: After performing the multi-precision multiplication of two 192-bit integers we obtain a number of the form

$$N = (A_5||A_4||A_3||A_2||A_1||A_0)$$

where A_i is a 64-bit integer. The value of N modulo p can then be computed with 3 additions modulo p,

$$N \equiv T + S_1 + S_2 + S_3 \pmod{p}$$

where

$$T = (A_2||A_1||A_0), \quad S_1 = (0||A_3||A_3),$$
$$S_2 = (A_4||A_4||0), \quad S_3 = (A_5||A_5||A_5)$$

2.2. **Non-Prime Fields.** Again there are a number of choices here for the fields $\mathbb{F}_q$ with $q = p^n$. These are usually implemented via a polynomial basis where

$$\mathbb{F}_q = \mathbb{F}_p[x]/f(x)$$

where $f(x)$ is an irreducible polynomial of degree n over $\mathbb{F}_p$. Normal bases can be used but these are usually when considering hardware implementations of fields in characteristic two, hence we shall not consider normal bases further here.

Characteristic Two In this case due to work described in [10] we need to choose n to be prime. One chooses $f(x)$ to be a trinomial or pentanomial for efficiency. In other words we choose either

$$f(x) = x^n + x^k + 1$$

or

$$f(x) = x^n + x^k + x^l + x^m + 1.$$

These have been a popular choice in standards bodies and for implementors due to the advantages that they offer in hardware and on some RISC processors. The literature on these is quite extensive so we just refer the reader to [7] or [13] for more details.

Optimized Extension Fields These fields have been proposed by Bailey and Paar in [5] and [6]. They appear to offer a number of advantages which we shall outline below. An Optimized Extension Field (OEF) is one of the form

- $p = 2^k \pm c$ is a pseudo-Mersenne prime with $\log_2 c \leq k/2$ and such that p fits into a computer word.
- $f(x) = x^n - \omega$ is irreducible.

For efficiency reasons it is often sensible to insist that $\omega = 2$.

In OEF fields addition of elements in $\mathbb{F}_q$ is relatively simple and can be accomplished without carries propagating, since elements of $\mathbb{F}_q$ are implemented as polynomials modulo $f(x)$. Multiplication is also very simple since reduction of a polynomial modulo $f(x)$ is particular simple. Multiplication can also be simplified via using Karatsuba multiplication, which even provides a performance advantage for polynomial multiplication for very small degree polynomials. Finally inversion is particularly easy since one can use a technique due to Itoh and Tsujii [12] combined with an efficient method to compute the action of the Frobenius mapping, see [6] for more details on this.

It should be noted that the technique of Weil descent which is described in [10] could be applied to curves defined over OEFs, since n is typcially small. However the resulting curve does not seem to have the nice properties that one observes in the even characteristic case. This is because the function field extensions are not Artin-Schreier in nature. Hence to the best current knowledge there are no security concerns with using OEFs, however this could change given the rapid progress made in studying the EC-DLP in recent years.

There has been no comprehensive comparison of the above choices of finite fields in the literature. Since the use of Montgomery arithmetic and even characteristic finite fields are quite standard [9] these are not the most interesting cases.

However the comparison of OEFs against GM-prime fields has not to our knowledge been carried out. But this is the most important comparison to make, since GM-prime fields are those which are being used by various standards bodies, e.g. ANSI, NIST and SECG.

In this paper we describe an independent evaluation of the performance of the four types of finite field mentioned above. The implementations we describe had a similar level of optimization applied to them, this is because, as mentioned previously, one cannot compare performance information across different processors or with different levels of optimization performed.

The implementation described was written in portable C++, with only a few lines of machine code. The target archetecture was assumed to have a 32-bit word length, the actual timings being implemented on a Sparc Ultra 5 Workstation with 64MB of RAM.

3. Field Operations

The fields we used for comparison where the following:

$K_1 = \mathbb{F}_p$ where $p = 2^{192} - 2^{64} - 1$. This was used for the GM-prime implementation and the Montgomery implementation.

$K_2 = \mathbb{F}_{p^n} = \mathbb{F}_p[z]/(z^6 - 2)$, where $p = 2^{31} - 19$. This was used for the OEF implementation.

$K_3 = \mathbb{F}_{2^n} = \mathbb{F}_2[z]/(z^{191} + z^9 + 1)$.

Notice that roughly the same bitlength was used for all fields, namely 192 and 186. The following table describes the timings we obtained.

Field Type	Addition	Multiplication	Square	Inversion
Montgomery	$0.88\mu s$	$7.56\mu s$	$7.36ms$	$0.18ms$
GM-prime	$0.92\mu s$	$5.48\mu s$	$5.44ms$	$0.20ms$
OEF	$1.00\mu s$	$5.04\mu s$	$4.84ms$	$0.02ms$
$\mathbb{F}_{2^{191}}$	$0.16\mu s$	$12.96\mu s$	$1.56ms$	$0.18ms$

Notice that the even characteristic case appears to be the worst since multiplication is almost twice as slow as the next slowest field type, namely the Montgomery representation. This however does not translate into a 100% increase in the required CPU time for the final cryptographic operation since for fields of even characteristic the squaring operation comes almost for free.

The use of multiplication is also slightly faster for the OEF field compared to the GM-prime field. But the most striking improvement is in the time required to perform an inversion in the field. As we shall comment later, this leads to important decisions on how one actually implements an elliptic curve cryptographic system. It is important to look at the ratio, $r = I/M$, of the time to perform an inversion, I, to the time to compute a multiplication, M, which in our examples comes out to be,

$$r_{\text{Mont}} = 23.81,\ r_{\text{GM-prime}} = 36.49,\ r_{\text{OEF}} = 3.97,\ r_{\mathbb{F}_{2^{191}}} = 13.88$$

4. Curve Operations

The basic elliptic curve operation required in cryptography is point multiplication. That is given $P \in E(\mathbb{F}_q)$ and $k \in_R [1, \ldots, \#E(\mathbb{F}_q) - 1]$ compute $[k]P$. There are various techniques to perform this which are described in [7] and [14].

A first observation is that if P is a fixed point which is required to be multiplied by a large number of values, k, then one can use a great deal of precomputation. Such a point, P, is often the generator of the group $\#E(\mathbb{F}_q)$ and is hence called a base point. However sometimes we do not know the value of P in advance and so different optimizations need to be performed, in such a situation we call P a general point.

In standard implementations for the EC-DH protocol each party needs to perform one multiplication of a general point and one multiplication of the fixed base point. In the EC-DSA protocol the signer needs to perform one multiplication of the base point and the verifier needs to perform a multiplication of the base point and a multiplication of a general point.

A second observation is that in both multiplication of a general point and of the base point can be done in either affine or 'mixed' coordinates. 'Mixed' coordinates refers to the fact that we use a projective representation of the points, we, however, take into account that some of the intermediate points may be in affine representation. Mixed coordinates are to be preferred when the ratio, r, of inversion to multiplication is large, since one is trading off inversions for a larger number of multiplications. On the other hand affine coordinates require 33% less storage.

For our timings we used the following curves which are suitably strong for cryptographic use:

$\underline{E(K_1)}$. We used the curve labelled $P - 192$ by NIST, [2]. This is the curve **secp192r1** in SECG and **prime192v1** in ANSI X9.62. This curve is given by

$$E_1 : Y^2 = X^3 - 3X + b$$

where

$$b = 2455155546008943817740293915197451784769108058161191238065.$$

This curve has group order

$$6277101735386680763835789423176059013767194773182842284081,$$

which is a prime. The use of a curve with a coefficient of -3 for the X term in the equation provides a certain performance advantage, whilst a curve of prime group order is clearly a security advantage.

$\underline{E(K_2)}$. In this case we needed to generate our own curve so we took the curve given by

$$E_2 : Y^2 = X^3 - 3X + 131072z^5.$$

This curve has group order

$$9807970940881741910790475986522413956726171926140144424 4,$$

which is four times a prime. Again notice the coefficient of X in the curve equation is minus three.

$E(K_3)$. In this case we use the curve in Example 16 on page 187 of [7]. This is given by

$$E_3 : Y^2 + XY = X^3 + X^2 + b$$

where

$$b = 7BC86E2102902EC4D5890E8B6B4981FF27E0482750FEFC03.$$

This curve has group order

$$2 \times 1569275433846670190958947355834614995815261150867795429199.$$

which is two times a prime. Here we have chosen a curve with coefficient of X^2 equal to one, this gives greater performance in characteristic two. In addition for characteristic two fields the best type of group order we can use is one which is twice a prime. Hence this curve is typical of ones used in real life systems, although NIST does not have a curve in characteristic two at this level of security.

In the following table we see that for OEF fields we not only obtain a improvement when using affine coordinates but we also obtain a small improvement over GM-prime fields when using mixed coordinates. We also reduce the amount of memory required for the tables in the window multiplication methods since we no longer need to store the z-coordinates.

Operation	Montgomery	GM-prime	OEF	$\mathbb{F}_{2^{191}}$
Addition : Affine	$239\mu s$	$230\mu s$	$47\mu s$	$218\mu s$
Addition : Mixed	$140\mu s$	$100\mu s$	$92\mu s$	$226\mu s$
Doubling : Affine	$262\mu s$	$237\mu s$	$59\mu s$	$212\mu s$
Doubling : Mixed	$85\mu s$	$60\mu s$	$54\mu s$	$88\mu s$
Base Point Mult. : Affine	$16ms$	$15ms$	$3ms$	$15ms$
Base Point Mult. : Mixed	$8ms$	$6ms$	$5ms$	$13ms$
General Point Mult. : Affine	$60ms$	$54ms$	$13ms$	$50ms$
General Point Mult. : Mixed	$22ms$	$15ms$	$13ms$	$26ms$

We also notice that for general point multiplications the performance of curves over fields of even characteristic is not as bad as one would be led to believe from just looking at the timings for the field arithmetic. In some small systems, to avoid attacks like DPA [11] one often alters the base point on every run of the protocol. Hence one never actually uses the special optimizations for multiplying a base point and all point multiplications become general ones.

5. Cryptographic Operations

Finally we timed three basic cryptographic operations which are popular using ECC namely unsigned Diffie-Hellman (EC-DH), the EC variant of the digital signature algorithm (EC-DSA), and the EC variant of the MQV primitive (MQV). The timings we give below for our four specimen fields. The times for EC-DH and EC-MQV are the times required by one of the parties to perform their calculations. In the following we assumed that any base point multiplication was done using the optimizations alluded to above.

Operation	Montgomery	GM-prime	OEF	$\mathbb{F}_{2^{191}}$
EC-DSA Sign	$9ms$	$7ms$	$4ms$	$20ms$
EC-DSA Verify	$30ms$	$22ms$	$17ms$	$60ms$
EC-DH	$30ms$	$21ms$	$16ms$	$63ms$
MQV	$42ms$	$30ms$	$24ms$	$84ms$

6. Conclusion

We have shown that OEF fields appear to offer performance advantages over other field representations used in ECC. This is not only in terms of overall performance but also in terms of storage memory requirements. The present author has no vested interests in any of the four fields types under consideration and hopefully the results can be taken as completely independent of commercial bias or the use of agressive optimization techniques applied to one of the cases only.

On the other hand it should be noted that the performance difference between OEF fields and fields based on Generalized Mersenne numbers is probably not large enough to warrant additions to the various standards since addition of OEF fields would degrade attempts to obtain interoperability between various implementations.

References

[1] ANSI X9.62 : Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). *American National Standards Institute*, 1999.

[2] NIST FIPS PUB 186-2 : DIGITAL SIGNATURE STANDARD (DSS) *National Institute for Standards and Technology*, 2000.

[3] SECG SEC 2: Recommended Elliptic Curve Domain Parameters *Standards for Efficient Cryptography Group*, 1999.

[4] WAP: WTLS : Wireless Transport Layer Security Specification *Wireless Application Forum Ltd*, 1999.

[5] D.V. Bailey and C. Paar. Optimal extension fields for fast arithmetic in public-key algorithms. *Advances in Cryptology - CRYPTO 98*, Springer-Verlag LNCS 1462, 472–485, 1998.

[6] D.V. Bailey and C. Paar. Efficient arithmetic in finite field extensions with applications in elliptic curve cryptography. To appear *J. Cryptology.*

[7] I.F. Blake, G. Seroussi and N.P. Smart. *Elliptic Curves in Cryptography.* Cambridge University Press, 1999.

[8] R. Crandall. Method and apparatus for public key exchange in a cryptographic system. U.S. Patent Number 5159632, 1992.

[9] E. De Win, S. Mister, B. Preneel and M. Wiener. On the performance of signature schemes based on elliptic curves. *Algorithmic Number Theory - ANTS-III*, Springer-Verlag LNCS 423, 252–266, 1998.

[10] P. Gaudry, F. Hess and N.P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Preprint*, 2000.

[11] P. Kocher, J. Jaffe and B. Jun. Differential Power Analysis. In *Advances in Cryptology, CRYPTO '99*, Springer LNCS 1666, pp 388–397, 1999.

[12] 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.

[13] R. Lidl and H. Niederreiter. *Finite Fields,* in *Encyclopedia of Mathematics and its Applications,* G.-C. Rota, editor, Addison-Wesley, 1983.

[14] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[15] P.L. Montgomery. Modular multiplication without trial division. *Math. Comp.*, **44**, 519–521, 1985.

[16] J.A. Solinas. Generalized Mersenne Numbers. Preprint, 1999.

Computer Science Department,, Woodland Road,, University of Bristol, BS8 1UB, UK

E-mail address: nigel@cs.bris.ac.uk