Commun. Korean Math. Soc. **18** (2003), No. 1, pp. 159–168

IMPLEMENTATION ISSUES FOR ARITHMETIC OVER EXTENSION FIELDS OF CHARACTERISTIC ODD

SANGHO OH, CHANG HAN KIM,
YONGTAE KIM, AND YOUNGHO PARK

ABSTRACT. In this paper we discuss the construction of a new extension field of characteristic odd and analyze the complexity of arithmetic operations over such a field. Also we show that it is suitable for Elliptic Curve Cryptosystems(ECC) and Digital Signature Algorithm(DSA, [7]) as an underlying field. In particular, our digital signature scheme is at least twice as efficient as DSA.

1. Introduction

A finite field $GF(2^n)$ has a great attractiveness in hardware implementations which stems from the fact that each element can be represented by n binary digits. But the binary representation of $GF(2^n)$ does not offer the same computational advantages in software implementations, since computer systems for a general purpose are designed to calculate results in units of data known as words. From such a viewpoint, an optimal extension field(OEF), $GF(p^n)$ where $p > 2$ is a pseudo-Mersenne prime, was recently introduced by Bailey and Parr [2]. If a pseudo-Mersenne prime p is of the size of word(say, 32 or 64-bit), it allows a fast subfield modular reduction. And since an OEF is constructed using an irreducible binomial $x^n - w$ for some w in $GF(p)$, a multiplication of two elements in $GF(p^n)$ is performed efficiently. In general, extension fields can be constructed as follows.

Let p be a prime and let f be an irreducible polynomial of degree n over $GF(p)$. Then

$$GF(p^n) \cong GF(p)[x]/(f),$$

Received April 22, 2002.
2000 Mathematics Subject Classification: 11Y40, 94A60.
Key words and phrases: extension field, Elliptic curve and digital signature algorithm.

that is, $GF(p^n)$ can be identified with a residue class of polynomial ring over $GF(p)$. Let α be a root of $f(x) = 0$. Then $B = \{1, \alpha, \alpha^2, \cdots, \alpha^{n-1}\}$ is a polynomial basis for $GF(p^n)$ and so we obtain

$$GF(p^n) = \{a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} | a_i \in GF(p)\}.$$

In this paper we introduce an extension field of characteristic odd with its irreducible polynomial $x^n + x^{n-1} + \cdots + 1$. It will be represented by the following non-conventional basis [5, 10]:

$$B' = \{\alpha, \alpha^2, \cdots, \alpha^n\}.$$

It can be easily checked that B' is another basis of $GF(p^n)$ over $GF(p)$. Then an extension field $GF(p^n)$ can be also identified with

$$\{a_0\alpha + a_1\alpha^2 + \cdots + a_{n-1}\alpha^n | a_i \in GF(p)\}.$$

In Section 2 we discuss the construction and arithmetic operations of the extension field mentioned above. In particular, we show that in exponentiation the new basis representation is more efficient than a polynomial basis representation. In Section 3 we compare the proposed extension field with the OEF for the complexities of a multiplication and a squaring and then present the results of our software implementation. In Section 4 the proposed extension field is applied to ECC and DSA.

2. Extension field

2.1. Construction

DEFINITION 2.1. Let p be a prime. We call p a pseudo-Mersenne prime if p is of the form $2^m - c$, where $log_2|c| < \frac{m}{2}$.

For a large prime p, arithmetic operations in $GF(p)$ are mostly influenced by a multiple-precision modular reduction. An efficient modular reduction method suitable for a pseudo-Mersenne prime is well-known from [7]. In particular, we always consider pseudo-Mersenne primes of the size of 32 or 64-bit word, since primes as large as possible while still within single-precision limits of CPU are efficient in arithmetic.

DEFINITION 2.2. We call f the n-th all-one-polynomial(AOP) if $f(x) = x^n + x^{n-1} + \cdots + 1$.

DEFINITION 2.3. Let p be a pseudo-Mersenne prime of the form $2^m - c$ and let f be the n-th irreducible AOP over $GF(p)$. An extension field $GF(p^n)$ over $GF(p)$ will be denoted $\mathbf{EF_n(m, c)}$ if it is constructed by f and the proposed non-conventional basis(B') representation.

The construction of $EF_n(m,c)$ can be achieved very simply and fast from the following theorem [8].

THEOREM 2.4. *Let q be a prime.*
$g(x) = x^n + x^{n-1} + \cdots + 1$ is irreducible over $GF(q)$ if and only if $n+1$ is a prime and q is primitive in Z_{n+1}.

In Theorem 2.4, note that a prime q can be reduced as a positive integer $< n+1$. Also when primes of the size of 32 bits or 64 bits are considered, n is very small in practical applications, say $4, 6, 10, 12, 16, 18, 22, 28, 30, 40, 42, 46$. Hence it is very easy to determine whether q is primitive or not in Z_{n+1}.

THEOREM 2.5. *Let p be a prime and let n be a positive integer satisfying that $n+1$ is a prime and p is a primitive element in Z_{n+1}. For any fixed non-negative integer k, ϕ_k is a one-to-one correspondence of a set $Z_{n+1}^{\times}$ onto itself with $\phi_k(i) = ip^k \ (mod\ n+1)$.*

PROOF. ϕ_k is clearly well-defined. Let $l \in Z_{n+1}^{\times}$. First we can find a unique non-negative integer $u < n$ satisfying that $l = p^u (\mathrm{mod}\ n+1)$. Let v and i be remainders of $u-k$ and p^v divided by n and $n+1$ respectively. Then

$$ip^k \ (mod\ n+1) = p^{v+k} \ (mod\ n+1) = p^u \ (mod\ n+1) = l.$$

Therefore ϕ_k is surjective, so the proof is done. □

2.2. Arithmetic

Here we describe a method to perform a multiplication and a squaring efficiently in $EF_n(m,c)$, where $p = 2^m - c$ is a pseudo Mersenne prime. In general, a multiplication of two elements in $GF(p^n)$ consists of a product of two elements - they can be considered as polynomials - and a reduction of the product by an irreducible polynomial. In the case of $EF_n(m,c)$, we replace the second step with some few precomputations.

Let $\mathbf{a}, \mathbf{b} \in EF_n(m,c)$, where $\mathbf{a} = \sum_{i=0}^{n-1} a_i \alpha^{i+1}$, $\mathbf{b} = \sum_{i=0}^{n-1} b_i \alpha^{i+1}$ and $a_i, b_i \in GF(p)$. Then using the identity $\alpha^{n+1} = 1$, we can obtain

$$\mathbf{a} \times \mathbf{b} = \sum_{i=0}^{n-1} c_i \alpha^{i+1} - \sum_{i=0}^{n-1} c_n \alpha^{i+1},$$

where

$$c_k = \sum_{\substack{i,j=0,\cdots,n-1 \\ i+j \equiv k-1 (mod\ n+1)}} a_i b_j \ , 0 \le k \le n.$$

REMARK 2.6. Each c_k has $n-1$ terms for a_ib_j except for c_n with n terms.

After setting up a precomputed table for c_k, we can perform a multiplication of $\mathbf{a}$ and $\mathbf{b}$ through a subfield arithmetic process. A multiplication of two elements in $EF_n(m,c)$ with the given table requires n^2 multiplications and n^2-1 additions in $GF(p)$. For example, let $n=6$, then the precomputed table is given as follows.

c_k	a pair of subscripts of terms for a_ib_j					
c_0	(1,5)	(2,4)	(3,3)	(4,2)	(5,1)	
c_1	(0,0)	(2,5)	(3,4)	(4,3)	(5,2)	
c_2	(0,1)	(1,0)	(3,5)	(4,4)	(5,3)	
c_3	(0,2)	(1,1)	(2,0)	(4,5)	(5,4)	
c_4	(0,3)	(1,2)	(2,1)	(3,0)	(5,5)	
c_5	(0,4)	(1,3)	(2,2)	(3,1)	(4,0)	
c_6	(0,5)	(1,4)	(2,3)	(3,2)	(4,1)	(5,0)

TABLE 1. Precomputed Table for a Multiplication

For a squaring, a precomputed table can be obtained trivially from that of a multiplication. In order to compute a square of an element in $EF_n(m,c)$, a total of $\frac{n(n+1)}{2}$ multiplications, $\frac{(n-1)(n+2)}{2}$ additions and $n+1$ doublings in $GF(p)$ is required.

Now we are ready for introducing exponentiation in $EF_n(m,c)$. Let $(t_k,\cdots,t_1,t_0)_p$ be the base p representation of a positive integer t. Then

$$\begin{aligned}
&\mathbf{a}^t \\
=\ &\prod_{j=0}^{k}\mathbf{a}^{t_jp^j}=\prod_{j=0}^{k}\left(\sum_{i=0}^{n-1}a_i\alpha^{i+1}\right)^{t_jp^j} \\
=\ &\prod_{j=0}^{k}\left(\sum_{i=0}^{n-1}a_i^{p^j}\alpha^{(i+1)p^j}\right)^{t_j}=\prod_{j=0}^{k}\left(\sum_{i=0}^{n-1}a_i^{p^j(mod\ p-1)}\alpha^{(i+1)p^j(mod\ n+1)}\right)^{t_j} \\
=\ &\prod_{j=0}^{k}\left(\sum_{i=0}^{n-1}a_i\alpha^{\phi_j(i+1)}\right)^{t_j}\triangleq\prod_{j=0}^{k}\mathbf{y_j}^{t_j},
\end{aligned}$$

where ϕ_j is a one-to-one correspondence of a set $Z_{n+1}^{\times}$ onto itself with $\phi_j(i)=ip^j\ (mod\ n+1)$.

REMARK 2.7. Note that for each fixed j, $\phi_j(i)$ is just a permutation of i's. Since some few precomputations will cut off the computation of $\phi_j(i)$, when computing $\mathbf{a}^t$ in $EF_n(m,c)$, we have only to consider $\prod_{j=0}^{k} \mathbf{y_j}^{t_j}$ for the known $\mathbf{y_j}$'s.

A simultaneous multiple exponentiation method [6] computes $\prod_{j=0}^{k} \mathbf{y_j}^{t_j}$ by performing $m-1$ squarings and at most $(2^{k+1}-2)+m-1$ multiplications. In Section 4 this efficient exponentiation is applied to public key cryptosystems based on a discrete logarithm in a multiplicative group of $EF_n(m,c)$.

As a special case of exponentiation, computing the p-th power of $\mathbf{a}$ is just a permutation of its coefficients, that is does not need any cost for time. Itoh and Tsujii inversion method(for example, see [3] or [11]) of using the identity $\mathbf{a}^{-1} = (\mathbf{a}^r)^{-1}\mathbf{a}^{r-1}$ for $r = p^{n-1} + p^{n-1} + \cdots + p + 1$ is available to the efficient computation of the multiplicative inverse of a non-zero element in $EF_n(m,c)$. We can perform an inversion in $EF_n(m,c)$ by $\lfloor log_2(n-1) \rfloor + HW(n-1)$ multiplications in $EF_n(m,c)$ and an inversion in $GF(p)$ where $HW(n)$ is the Hamming weight of n. In the case of OEF, an inversion needs $\lfloor log_2(n-1) \rfloor + HW(n-1)$ multiplications in $GF(p^n)$ and an inversion and $(n-1)^2$ multiplications in $GF(p)$. Hence the time complexity for an inversion is reduced by 18%, under the practical consideration of ECC, that is for $3 \leq n \leq 10$.

3. Comparison of complexities

In Table 2 and 3 we assume that $x^n - w$ and n-th AOP are irreducible over $GF(p)$ where p is a pseudo-Mersenne prime of the form $2^m - c$ and $w \in GF(p)$. Also a squaring in $GF(p)$ will be considered like a multiplication in $GF(p)$. Note that our assumption is very theoretical, since it is not necessary to consider the same subfield $GF(p)$. In fact, we only have to assume nearly similar subfields in size.

	Mul. in $GF(p)$	Add. in $GF(p)$	Mul. by w in $GF(p)$
OEF with $x^n - w$	n^2	$n^2 - n$	$n-1$
$EF_n(m,c)$	n^2	$n^2 - 1$	No

TABLE 2. Multiplication in $GF(p^n)$

	Mul. in $GF(p)$	Add. in $GF(p)$	Mul. by w in $GF(p)$	Mul. by 2 in $GF(p)$
OEF with $x^n - w$	$\frac{n(n+1)}{2}$	$(*)$	$n-1$	$2n-3$
$EF_n(m,c)$	$\frac{n(n+1)}{2}$	$\frac{(n-1)(n+2)}{2}$	No	$n+1$

TABLE 3. Squaring in $GF(p^n)$

$(*)$ If n is even then $\frac{n(3n-4)}{4}$, otherwise $\frac{(3n+1)(n-1)}{4}$.

The multiplication in $EF_6(32, 99)$ derived from the non-conventional basis representation, is implemented in the C-language on a Pentium MMX 233Mhz. In Table 4 we give running times for a squaring and a multiplication in μ seconds and compare our results with the previous works.

	Mul. in extension fields	Platform
OEF with $x^6 - 7$ and $p = 2^{31} - 1$ [3]	5.82	233Mhz, Pentium
$EF_6(32, 99)$	6.51	//

TABLE 4. Time for field arithmetic operations.

4. Applications

4.1. Elliptic curve cryptosystem

Efficient implementations of ECC are closely related to arithmetic operations in underlying finite fields; The elliptic curve addition, which is one of the most significant primitives in elliptic curve cryptography, is performed by finite field arithmetic operations. Recently Bailey and Parr [2, 3] showed that an OEF yields a considerable speed advantage over previous software implementations of finite field arithmetic for elliptic curve cryptography. Therefore the proposed extension field is available to practical applications based on ECC as mentioned in the previous sections. For reference, there exist a number of extension fields of the proposed type. For example, we can construct 967 $EF_6(32, c)$s, 1158 $EF_{10}(32, c)$s, and 747 $EF_4(64, c)$s for $0 < c < 2^{16}$.

4.2. Digital signature algorithm

First of all we start from comparing complexities for arithmetic operations in a ring Z_q and an extension field $EF_n(m, c)$ for a positive integer $q \approx (2^m)^n$. In Table 5 we consider exponents of the size of 160 bits on 32-bit word system and of the size of 255 bits on 64-bit word system respectively. And let q_1 and q_2 be moduli of the size of 960 bits and of the size of 1024 bits respectively. In particular, the complexity for a modular exponentiation in Z_{q_1} and Z_{q_2} is obtained using a modified k-ary exponentiation method [7]. Here k is considered to be 4.

	on 32-bit word system		on 64-bit word system	
	$EF_n(32, c)$	Z_{q_1}	$EF_n(64, c)$	Z_{q_2}
Mul.	n^2 mul. in $GF(2^{32} - c)$	(1) $2n^2 + n$ mul. [8] (2) n^2 mul. [5] of 32-bit integers	n^2mul. in $GF(2^{64} - c)$	(1) $2n^2 + n$ mul. (2) n^2 mul. of 64-bit integers
Average for an Exp.	31 sqr. 46 mul.	157 sqr. 37.5 mul.	63 sqr. 70 mul.	252 sqr. 59 mul.
Worst for an Exp.	31 sqr. 61 mul.	157 sqr. 39 mul.	63 sqr. 77 mul.	252 sqr. 62 mul.

TABLE 5. Complexities for a multiplication and an exponentiation.

REMARK 4.1. Note that in a ring Z_q, the complexity of a squaring is about 80% of that of a multiplication except that q is of the form l^k for a small integer l.

DSA is a digital signature scheme based on a discrete logarithm in a multiplicative group of a prime field and an exponentiation plays the most important role in efficient implementations of a discrete logarithm based cryptography. Table 5 shows that an exponentiation in the proposed extension field is at least twice as efficient as that in a ring Z_q. Hence DSA is worth being extended to on $EF_n(m, c)$. Our scheme is the same with DSA except for a key generation procedure. Table 6 lists each step of key generation procedures of DSA and our scheme.

REMARK 4.2. In our scheme,

1. $x^{16} - 1 = (x^8 + 1)(x^4 + 1)(x^2 + 1)(x + 1)(x - 1)$.
2. The number of $EF_{16}(64, c)$ satisfying our scheme condition and $0 < c < 2^{32}$ is more than two millions.
3. Since $q \simeq 255$ bits, one exponentiation takes a total of 63 squarings and at most 77 multiplications(70 on average).

DSA	Our Scheme
1. Select a prime $q \approx 2^{160}$.	1. Set up an $EF_{16}(64, c)$ where $p = 2^{64} - c$.
2. Select a prime $p \approx 2^{1024}$ satisfying $q \mid p - 1$.	2. Select a prime $q \approx p^4$ satisfying $q \mid (p^8 + 1)$.
3. Select an element $g \in GF(p)^\times$ and compute $\alpha = g^{\frac{p-1}{q}} (\neq 1)$.	3. Select an element $g \in EF_{16}(64, c)$ and compute $\alpha = g^{\frac{p^{16}-1}{q}} (\neq 1)$.
4. Select a random integer a for $1 < a < q$.	4. Select a random integer a for $1 < a < q$.
5. Compute α^a.	5. Compute α^a.
6. Public key(p, q, α, α^a) and Private key(a)	6. Public key$(EF_{16}(64, c), q, \alpha, \alpha^a)$ and Privatekey(a)

TABLE 6. Key Generation Procedures.

REMARK 4.3. Since $p^8 - 1$ is not divisible by q, the security of our scheme relies on two distinct but related discrete logarithm problems. One is the logarithm problem in $GF(p^n)^\times$; the other is the logarithm problem in the cyclic subgroup of order q, where the best current methods run in "square-root" time. For reference, the fastest algorithm for computing a discrete logarithm problem in $GF(p^n)^\times$ is a number field sieve method of subexponential time [1].

5. Conclusion and future research

The discrete logarithm problem is one of most important mathematical problems of Public Key Cryptography. The international standardization of Public Key Cryptography have dealt with this problem in conjunction with a large prime field and an elliptic curve defined over a prime field or $GF(2^n)$. In this paper we have shown that arithmetic operations in an extension field of characteristic odd are very efficient as compared to the previous works, so the proposed extension field is suitable for ECC and DSA as an underlying field.

Computing a p-th power of an element in $EF_n(m, c)$ for $p = 2^m - c$ is just a permutation of coefficients. An inversion method(for example, see [3]) of using the identity $\mathbf{a}^{-1} = (\mathbf{a}^r)^{-1}\mathbf{a}^{r-1}$ where $\mathbf{a} \in GF(p^n)$ and $r = p^{n-1} + p^{n-2} + \cdots + p + 1$, is considerably efficient in the case of $EF_n(m, c)$. Also it is well-known that all inversion methods of using a p-th power for a field characteristic p are suitable for a hardware implementation. Lastly we suggest that the extension field mentioned above be applied to hardware implementations of ECC.

ACKNOWLEDGEMENTS. Authors wish to thank all of the members of CIST for their valuable remarks and improvements to this paper.

References

[1] L. M. Adleman and Jonathan DeMarrais, *A subexponential algorithm for discrete logarithms over all finite fields*, Mathematics of Computation **61** (1993), 1–15.
[2] D. V. Bailey and Christof Paar, *Optimal extension fields for fast arithmetic in public-key algorithms*, Crypto '98, LNCS 1462 (1998), 472–485.
[3] ———, *Inversion in Optimal Extension Fields*, Conference on The Mathematics of Public Key Cryptography, 1999.
[4] E. De Win, A. Bosselaers, S. Vandenberghe, P De Gersem and J. Vandewalle, *A fast software implementation for arithmetic operations in* $GF(2^t)$, Asiacrypt '96, LNCS 1163 (1996), 65–76.
[5] C. H. Kim, S. Oh, and J. Lim, *A new Hardware architecture for operations in* $GF(2^m)$, IEEE Trans. Computers **51** (2002), no. 1, 90–92.
[6] S. M. Hong, S. Y. Oh and H. S. Yoon, *New modular multiplication algorithms for fast modular exponentiation*, Eurocrypt '96, Springer-Verlag (1996), 166–177.
[7] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1996.
[8] A. J. Menezes, *Applications of finite fields*, Kluwer Academic Publishers, 1993.
[9] P. L. Montgomery, *Modular multiplication without trial division*, Mathematics of Computation **44** (1985), 519–521.
[10] S. Oh, C. H. Kim, *Proposing the use of non-conventional basis of finite fields*, Contribution to IEEE P1363, 1999.
[11] T. Itoh and S. Tsujii, *A fast algorithm for computing multiplicative inverses in* $GF(2^m)$ *using normal bases*, Information and Computation **78** (1988), 171–177.

Sangho Oh
Center for Information Security Technologies
Korea University
Seoul 136-701, Korea
E-mail: gausmath@dreamwiz.com

Chang Han Kim
Department of Information Security
Semyung University
Jecheon 390-711, Korea
E-mail: chkim@semyung.ac.kr

Yongtae Kim
Department of Mathematics Education
Kwangju National University of Education
Kwangju 500-703, Korea
E-mail: ytkim@gnue.ac.kr

Youngho Park
Department of Information Security and Systems
Sejong Cyber University
Seoul 143-747, Korea
E-mail: youngho@cybersejong.ac.kr