Optimal Extension Fields for XTR

Dong-Guk Han[1]*, Ki Soon Yoon[1], Young-Ho Park[2], Chang Han Kim[3], and Jongin Lim[1]

[1] Center for Information and Security Technologies(CIST),
Korea University, Anam Dong, Sungbuk Gu, Seoul, KOREA
{christa,ksyoon}@cist.korea.ac.kr,
jilim@tiger.korea.ac.kr
[2] Dept. of Information Security & System Engineering,
Sejong Cyber Univ., Seoul, KOREA
youngho@cybersejong.ac.kr
[3] Dept. of Information Security, Semyung Univ., Jechon, KOREA
chkim@venus.semyung.ac.kr

Abstract. Application of XTR in cryptographic protocols leads to substantial savings both in communication and computational overhead without compromising security [6]. XTR is a new method to represent elements of a subgroup of a multiplicative group of a finite field $GF(p^6)$ and it can be generalized to the field $GF(p^{6m})$ [6,9]. This paper proposes optimal extension fields for XTR among Galois fields $GF(p^{6m})$ which can be applied to XTR. In order to select such fields, we introduce a new notion of Generalized Optimal Extension Fields(GOEFs) and suggest a condition of prime p, a defining polynomial of $GF(p^{2m})$ and a fast method of multiplication in $GF(p^{2m})$ to achieve fast finite field arithmetic in $GF(p^{2m})$. From our implementation results, $GF(p^{36}) \rightarrow GF(p^{12})$ is the most efficient extension fields for XTR and computing $Tr(g^n)$ given $Tr(g)$ in $GF(p^{12})$ is on average more than twice faster than that of the XTR system[6,10] on Pentium III/700MHz which has 32-bit architecture.

Keywords XTR public key system, Pseudo-Mersenne prime, Karatsuba's method.

1 Introduction

Almost all public key systems have a large key size except Elliptic Curve Cryptosystem(ECC). This is impractical in many applications such as smart card and wireless telecommunication of which power and bandwidth are limited. So many cryptographers think that ECC is one of the most efficient public key systems applicable to many hardwares with limited environments.

* This work was supported by both Ministry of Information and Communication and Korea Information Security Agency, Korea, under project 2002-130

K. Nyberg and H. Heys (Eds.): SAC 2002, LNCS 2595, pp. 369–384, 2003.

The XTR public key system was introduced at Crypto 2000[6]. From a security point of view XTR is a traditional subgroup discrete logarithm system. But it uses a non-standard way to represent and compute subgroup elements to achieve substantial computational and communication advantages over traditional representations. It is the first method we are aware of that uses $GF(p^2)$ arithmetic to achieve $GF(p^6)$ security, without requiring explicit construction of $GF(p^6)$. As shown in [6], XTR of security equivalent to 1024-bit RSA achieves speed comparable to cryptosystems based on random elliptic curves over random prime fields (ECC) and of equivalent security. Here the XTR public keys are only twice as large as ECC keys, but parameter initialization from scratch for XTR takes a negligible amount of computing time, unlike RSA and ECC. So XTR is an excellent alternative to either RSA or ECC in applications such as smart card and wireless telecommunications. Therefore finding the optimized extension fields for XTR is very significant.

It was mentioned very briefly that XTR can be generalized in a straightforward way using the extension field of the form $GF(p^{6m})$ and systematic design of this generalization was proposed in [9]. But they did not propose optimized extension fields for XTR.

In this paper, we suggest optimized extension fields among several extension fields $GF(p^{6m})$ at 32-bit word system. It is sufficient to look around following five extension fields for XTR according to a size of prime p.

□ Generalized extension fields for XTR : $GF(p^{6m}) \rightarrow GF(p^{2m})$

- $GF(p^6) \rightarrow GF(p^2)$, the size of prime p is about 170 bits and $m = 1$.
- $GF(p^{12}) \rightarrow GF(p^4)$, the size of prime p is about 85 bits and $m = 2$.
- $GF(p^{18}) \rightarrow GF(p^6)$, the size of prime p is about 64 bits and $m = 3$.
- $GF(p^{36}) \rightarrow GF(p^{12})$, the size of prime p is about 32 bits and $m = 6$.
- $GF(p^{66}) \rightarrow GF(p^{22})$, the size of prime p is about 16 bits and $m = 11$.

We compare complexities of elementary operations of above five extension fields and select optimal extension fields for XTR. Moreover, we consider the cost of computing XTR single exponentiation that is to compute $Tr(g^n)$ given $Tr(g)$, $n \in Z$. The most frequently performed operations in XTR[6,9] are the following three types:

$$x^2, xy, xz - yz^{p^m} \text{ for } x, y, z \in GF(p^{2m})$$

These three operations play an important role to speed up XTR single exponentiation. To optimize elementary operations in $GF(p^{2m})$ we stipulate the following properties on the choice of p and defining polynomial:

1. Choose p to be a pseudo-Mersenne prime, that is, of the form $2^n \pm c$ for some $\log_2 c \leq \frac{1}{2}n$ to allow for efficient subfield modular reduction.

2. Choose a defining polynomial that is a binomial or $2m$-th all-one-polynomial (AOP) for efficient extension field modular reduction.
3. Choose p to be less than but close to the word size of the processor so that all subfield operations take advantage of the processor's fast integer arithmetic.

To meet the third condition, we choose m large enough. And then we can select p to be less than but close to the word size of the processor. On the other hand, a number of subfield multiplications and additions in $GF(p)$ required for polynomial multiplicaton in $GF(p^{2m})$ would increase rapidly. As customary we do not count the cost of additions in $GF(p)$, however, when p is small and m is large the cost of additions in $GF(p)$ is not negligible. In this paper, we use Karatsuba-like method at polynomial multiplicaton in $GF(p^{2m})$ to reduce the number of subfield multiplications and modify the Karatsuba-like method to reduce the cost of additions in $GF(p)$. Due to the above considerations for prime p and fast polynomial multiplication method, our proposed system is on average more than twice faster than the XTR[6,10] to compute a single exponentiation.

As the construction in this paper, there are fewer appropriate prime p than XTR case. This means that users may share the same extension field. But Elliptic Curve Cryptosystems has this property.

The rest of the paper is organized as follows. In Section 2 we shall briefly review on the XTR Public Key Cryptosystems[6,9,10]. In Section 3 and 4 deal with Generalized Optimal Extension Field(GOEF) and an efficient arithmetic in GOEF, respectively. In Section 5 we present our implementation results and propose the optimized extension field for XTR. The final Section contains our conclusions to the present work.

2 Review on the XTR Public Key Cryptosystems

In this section we review some of the results from [6,9,10].

Definition 1. *The trace $Tr(h)$ over $GF(p^{2m})$ of $h \in GF(p^{6m})$ is the sum of the conjugates over $GF(p^{2m})$ of h,i.e.,*

$$Tr(h) = h + h^{p^{2m}} + h^{p^{4m}}.$$

For constructing XTR, primes p, q and positive integer m must satisfy following conditions :

- p and $2m+1$ are prime numbers, p $(mod\ 2m+1)$ is a primitive element in Z_{2m+1}.
- $\Phi_{6m}(p)$ has a prime factor q whose the size is more than 160 bits.

Note that $\Phi_n(X)$ is $n-th$ cyclotomic polynomial for a positive integer n. The first above condition guarantees $GF(p^{2m})$ has an optimal normal basis of type I[11, Theorem 5.2]. The subgroup with order q cannot be embedded in the

multiplicative group of any true subfield of $GF(p^{6m})$ by the second condition[7, Lemma 2.4].

Definition 2. *For $c \in GF(p^{2m})$ let $F(c, X)$ be the polynomial $X^3 - cX^2 + c^{p^m}X - 1 \in GF(p^{2m})[X]$ with (not necessarily distinct) roots h_0, h_1, h_2 in $GF(p^{6m})$, and let $c_n = h_0^n + h_1^n + h_2^n$ for $n \in Z$.*

Note. If $F(c, X)$ is irreducible over $GF(p^{2m})$ then c_n is equal to $Tr(h_0^n)$.

Lemma 1 (9, Lemma 2.1).

i. $c = c_1$.
ii. $h_0h_1 + h_1h_2 + h_0h_2 = c^{p^m}$.
iii. $h_0h_1h_2 = 1$.
iv. $c_n = c_{np^m} = c_n^{p^m}$ *for* $n \in Z$.
v. Either all h_i *have order dividing* $p^{2m} - p^m + 1$ *and* > 3 *or all* $h_i \in GF(p^{2m})$.
vi. $(c_n)_t = c_{nt} = (c_t)_n$.
vii. $c_n \in GF(p^{2m})$ *for* $n \in Z$. *[6, Lemma 2.3.2]*

Corollary 1 (9, Lemma 2.3). *Let c, c_{n-1}, c_n and c_{n+1} be given.*

i. $c_{2n} = c_n^2 - 2c_n^{p^m}$.
ii. $c_{n+2} = c * c_{n+1} - c^{p^m} * c_n + c_{n-1}$.
iii. $c_{2n-1} = c_{n-1} * c_n - c^{p^m} * c_n^{p^m} + c_{n+1}^{p^m}$.
iv. $c_{2n+1} = c_n * c_{n+1} - c * c_n^{p^m} + c_{n-1}^{p^m}$.

In XTR, an algorithm to compute $Tr(g^n)$ given $Tr(g)$ and $n \in Z$ is needed like that to compute g^n in public key system based on discrete logarithm problem. The size of n depends on the size of the order g.

Definition 3. *Let $S_n(c) = (c_{n-1}, c_n, c_{n+1}) \in GF(p^{2m})^3$*

To compute $S_n(c)$ from any given $c \in GF(p^{2m})$, $m \in Z^+$, the algorithm 2.3.7[6] for $m = 1$ and the algorithm 4.2 [9] were proposed. Another XTR single exponentiation method [10] is on average more than 35% faster than the algorithm 2.3.7[6]. The most frequently performed operation in the algorithm to compute $S_n(c)$[6,9,10] was organized with Corollary 1 *i,iii,iv.* Thus the complexity of the algorithm computing $S_n(c)$ depends on the complexity of computing $x^2, xy, xz - yz^{p^m}$ for $x, y, z \in GF(p^{2m})$.

Lemma 2 (10, Lemma 2.2). *$x, y, z \in GF(p^2)$ with $p \equiv 2 \bmod 3$.*

i *Computing x^p is free.*
ii *Computing x^2 takes two multiplications in $GF(p)$.*
iii *Computing xy costs the same as two and a half multiplications in $GF(p)$.*
iv *Computing $xz - yz^p$ costs the same as three multiplications in $GF(p)$.*

Lemma 3 (9, Lemma 4.3). *Let p and $2m+1$ be prime numbers, where p (mod $2m+1$) is a primitive element in Z_{2m+1}. Then for $x, y, z \in GF(p^{2m})$,*

i *Computing x^{p^m} is free.*
ii *Computing x^2 takes 80% of the complexity taken for multiplications in $GF(p^{2m})$.*
iii *Computing xy takes $4m^2$ multiplications in $GF(p)$.*
iv *Computing $xz - yz^{p^m}$ takes $4m^2$ multiplications in $GF(p)$.*

Remark 1. In $GF(p^2)$, $S_n(c)$ can be computed in $7\log_2 n$ multiplications in $GF(p)$ [10].

Theorem 1. *Let $c \in GF(p^{2m})$ and a positive integer n be given. Then it takes $(2a+b)\log_2 n$ multiplications in $GF(p)$ to compute $S_n(c)$, where a is the number of multiplications in $GF(p)$ to compute x^2 for $x \in GF(p^{2m})$ and b is the number of multiplications in $GF(p)$ to compute $xz - yz^{p^m}$ for $x, y, z \in GF(p^{2m})$.*

3 Generalized Optimal Extension Field

The performance of field arithmetic in $GF(p^m)$ mainly depends on the choice of parameters for extension field, such as a prime p and a defining polynomial. The reduction step in multiplication has the biggest time complexity. So there are many methods proposed as follows to reduce complexity in reduction steps.

Definition 4. *Let c be a positive rational integer. A pseudo-Mersenne prime is a prime number of the form $2^n \pm c$, $\log_2 c \leq \lfloor \frac{1}{2} n \rfloor$.*

Definition 5. *[3] An Optimal Extension Field(OEF) is a finite field $GF(p^m)$ such that:*

1. *p is a pseudo-Mersenne prime,*
2. *An irreducible binomial $P(x) = x^m - \omega$ exists over $GF(p)$.*

Theorem 2. *[11] Let $m \geq 2$ be an integer and $\omega \in GF(p)^*$. Then the binomial $x^m - \omega$ is irreducible in $GF(p)[x]$ if and only if the following two conditions are satisfied:*

i. *each prime factor of m divides the order e of ω over $GF(p)$, but not $(p-1)/e$,*
ii. *$p \equiv 1$ mod 4 if $m \equiv 0$ mod 4.*

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

There is another method to reduce the complexity of extension field modular reduction.

Definition 6. *We call f_m the m-th all-one-polynomial(AOP) if*

$$f(x) = \Phi_{m+1}(x) = \frac{x^{m+1}-1}{x-1} = x^m + x^{m-1} + \cdots + x + 1.$$

The following theorem shows when AOP is irreducible [11].

Theorem 3. *Let p be a prime. $f_m(x)$ is irreducible over $GF(p)$ if and only if $m+1$ is a prime and p is primitive in Z_{m+1}.*

Theorem 4. *If $m+1$ is a prime and p is primitive in Z_{m+1}, where p is a prime or prime power. Let α be a root of m-th AOP then $\{\alpha, \alpha^p, \cdots, \alpha^{p^{m-1}}\}$ is a basis of $GF(p^m)$ over $GF(p)$. Furthermore $\{\alpha, \alpha^p, \cdots, \alpha^{p^{m-1}}\} = \{\alpha, \alpha^2, \cdots, \alpha^m\}$.*

Proof. The first assertion is from [11]. To prove the second, it is sufficient to show that $\{1, p, p^2, \cdots, p^{m-1}\} = \{1, 2, \cdots, m\}$ in Z_{m+1}. Suppose $0 \le j \le i \le m-1$ and $p^i \equiv p^j \ mod\ (m+1)$ then $p^j(p^{i-j}-1) \equiv 0 \ mod\ (m+1)$. Thus $p^{i-j} \equiv 1 \ mod\ (m+1)$ and order of p divides $i-j$. But the order of p is m and $i-j < m$. Therefore $i = j$. Hence $\{1, p, p^2, \cdots, p^{m-1}\}$ are all distinct and $\{1, p, p^2, \cdots, p^{m-1}\} = \{1, 2, \cdots, m\}$.

Definition 7. *If a set $A = \{\alpha, \alpha^2, \cdots, \alpha^m\}$ in $GF(p^m)$ is a basis over $GF(p)$ then we call it a non-conventional basis of $GF(p^m)$ over $GF(p)$.*

In this paper, we use a non-conventional basis representation for $GF(p^m)$ whose defining polynomial is AOP. Because the property that is $\alpha^{m+1} = 1$ and $1 = -\alpha - \alpha^2 - \cdots - \alpha^m$ can be used to speed up extension field modular reduction in a non-conventional basis. The detail explanation can be covered in the section 4.3.

We introduce a new type of Galois field.

Definition 8. *Generalized Optimal Extension Field(GOEF) is a finite field $GF(p^m)$ such that:*

1. *p is a pseudo-Mersenne prime,*
2. *Either an binomial $x^m - \omega$ or $m-th$ AOP is irreducible.*

The construction of GOEF can be achieved from a pair of a pseudo-Mersenne prime and an irreducible polynomial. Either a binomial or an AOP must be chosen as the irreducible polynomial to construct GOEF. But in GOEF, determination of irreducible polynomial is related to the choice of a pseudo-Mersenne prime. The choice of defining polynomial determines the complexity of the operations required to perform the extension field modular reduction. And the selection of a pseudo-Mersenne prime affects the complexity in subfield modular

reduction. So it is very important to make a pair of a pseudo-Mersenne prime and a defining polynomial. We consider a polynomial basis or normal basis representation of a field element $A \in GF(p^m)$ depending on f.

$$A(\alpha) = a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha + a_0, \ \textit{when } f(x) \textit{ is binomial.}$$

or

$$A(\alpha) = a_{m-1}\alpha^{m} + \cdots + a_1\alpha^2 + a_0\alpha, \ \textit{when } f(x) \textit{ is AOP.}$$

where $a_i \in GF(p)$ and α is a root of $f(x)$. Note that since we choose p to be less than the processor's word size, we can represent $A(x)$ with m registers.

4 Efficient Arithmetic in GOEF

This section describes a basic construction for arithmetic in fields $GF(p^m)$, of which GOEF is a special case.

4.1 Addition and Subtraction

Addition and subtraction of two field elements are implemented in a straightforward manner by adding or subtracting the coefficients of their polynomial or normal basis representation and if necessary, performing a modular reduction by subtracting or adding p once from the intermediate result.

4.2 Multiplication

Field multiplication can be performed in two steps. First, we perform a multiplication of two field elements $A(\alpha)$ and $B(\alpha)$. If we use $f(x)$ as a binomial and polynomial basis, then resulting in an intermediate product $C'(\alpha)$ of degree less than or equal to $2m-2$.

$$C'(\alpha) = A(\alpha)B(\alpha) = c'_{2m-2}\alpha^{2m-2} + \cdots + c'_1\alpha + c'_0 \ \textit{where } c'_i \in GF(p).$$

When AOP is used, the intermediate product $C'(\alpha)$'s degree is less than or equal to $2m$.

$$C'(\alpha) = A(\alpha)B(\alpha) = c'_{2m-2}\alpha^{2m} + \cdots + c'_1\alpha^3 + c'_0\alpha^2 \ \textit{where } c'_i \in GF(p).$$

The schoolbook method to calculate the coefficients $c'_i, i = 0, 1, \cdots, 2m-2$, requires m^2 multiplications and $(m-1)^2$ additions in the subfield $GF(p)$.

Since field multiplication is the time critical task in many public-key algorithms, in this paper, we use Karatsuba-like Method[5] to calculate the coefficients which requires $O(m^{1.59})$ cost for multiplication at the cost of more subfield additions[1]. Using this method gives considerable advantages to the cost of multiplication in the subfield. In general, the costs of addition and subtraction in $GF(p)$ do not be counted. But if m is chosen such that the size of p is as small as the word size of common processors, the complexity of addition and

subtraction must be considered. For example, it is required 62 additions, subtractions and 18 multiplications in a base field to compute a multiplication in $GF(p^6)$, but the number of additions,subtractions and multiplications in a base field increases to 722 and 147 in $GF(p^{22})$, respectively. From the results in Table 6, $T_{mul}/T_{add+sub} = 5.76$ in $GF(p^6)$ and $T_{mul}/T_{add+sub} = 1.7$ in $GF(p^{22})$ where T_{mul} is the time required for a subfield multiplication and $T_{add+sub}$ for a subfield addition and subtraction. The detail description of Karatsuba-like method won't be covered here for the lack of space. Instead we give an example to compute multiplication in $GF(p^{12})$ by our modified Karatsuba-like method in Appendix.

Also, there is Schonhage-Strassen FFT based Method which requires $O(m (\log_2 m)(\log_2 \log_2 m))$ complexity for multiplication. But this method is better than the classical method approximately when $m \geq 300$[2].

In Section 4.3 we present an efficient method to calculate the residue $C(\alpha) \equiv C'(\alpha) \ mod \ f(\alpha), C(\alpha) \in GF(p^m)$.

4.3 Extension Field Modular Reduction

After performing a multiplication of field elements in a polynomial representation, we obtain the intermediate result $C'(\alpha)$. In general, the degree of $C'(\alpha)$ will be greater than or equal to m when $f(x)$ is a binomial, and $m+1$ when AOP is used. In this case, we need to perform a modular reduction. The canonical method to carry out this calculation is long division with remainder by the defining polynomial. However, defining polynomials of special form allow for computational efficiencies in the modular reduction.

When the Defining Polynomial Is Binomial : $x^m - \omega$

Theorem 5. *[3] Given a polynomial $C'(\alpha)$ over $GF(p)$ of degree less than or equal to $2m-2$, $C'(\alpha)$ can be reduced module $f(x) = x^m - \omega$ requiring at most $m-1$ multiplications by ω and $m-1$ additions, where both of these operations are performed in $GF(p)$.*

A general expression for the reduced polynomial is given by :

$$C(\alpha) \equiv c'_{m-1}\alpha^{m-1} + [\omega c'_{2m-2} + c'_{m-2}]\alpha^{m-2} + \cdots + [\omega c'_m + c'_0] \ mod \ f(\alpha).$$

As an optimization, when possible we choose those fields with an irreducible binomial $x^m - 2$, allowing us to implement the multiplications as shifts.

When the Defining Polynomial Is AOP : $x^m + x^{m-1} + \cdots + x + 1$

Theorem 6. *Given a polynomial $C'(\alpha)$ over $GF(p)$ of degree less than or equal to $2m$, $C'(\alpha)$ can be reduced module $f(x) = x^m + x^{m-1} + \cdots + x + 1$ requiring at most $m-2$ additions and m subtractions, where both of these operations are performed in $GF(p)$.*

Using the property that is $\alpha^{m+1} = 1$ *and* $1 = -\alpha - \alpha^2 - \cdots - \alpha^m$ then a general expression for the reduced representation is given by :

$$C(\alpha) \equiv [c'_{m-2} - c'_{m-1}]\alpha^m + [c'_{m-3} - c'_{m-1} + c'_{2m-2}]\alpha^{m-1} + \cdots + [c'_0 - c'_{m-1} + c'_{m+1}]\alpha^2 + [c'_m - c'_{m-1}]\alpha \ mod \ f(\alpha).$$

Comparison : In general, multiplication is more expensive than subtraction. In Theorem 5, modular reduction requires at most $m - 1$ multiplications by ω. If ω is 2 then the complexity of reduction of the above two methods is almost equal. When ω is greater than 2, however, extension field modular reduction using AOP is more efficient.

Combining or Postponing the Reduction Steps : For a regular multiplication of $a, b \in GF(p)$, an integer multiplication step and an integer reduction step are needed.

Generally, extension field multiplication using Schoolbook method takes m^2 multiplications in a base field. That is m^2 integer reduction step must be performed. If m is large, the cost of reduction step is very high. So a method which reduces the number of reduction step, contributes to the overall performance. We can reduce the number of reduction step by combining individual product terms as many as possible, and then reducing the accumulated sum *mod* p only once. Therefore, only m reduction step is needed. Because the intermediate results are greater than p^2 in absolute value when p is selected as a prime near the word size, the cost of the resulting final reductions is higher than that of the original reductions. According to the our implementation results, the case when p is word size and m is large is not attractive. Because modern workstation CPUs are optimized to perform integer arithmetic on operands of size up to the width of their registers and a double or triple-word integer arithmetic generated in combining stage is considerably less efficient than single-word integer arithmetic. Combining or postponing the reduction steps is not at all new. See for instance [4] for much earlier applications.

4.4 Fast Subfield Multiplication with Modular Reduction

In general, fast subfield multiplication is essential for fast multiplication in $GF(p^m)$. Subfield arithmetic in $GF(p)$ is implemented with standard modular integer techniques. For efficient implementation of GOEF arithmetic, optimization of subfield arithmetic is critical to performance. Modern workstation CPUs are optimized to perform integer arithmetic on operands of size up to the width of their registers. GOEF takes advantage of this fact by constructing subfields whose elements may be represented by integers in a single register.

We perform multiplication of two single-word integers and in general obtain a double-word integer result. In order to finish the calculation, we must perform a modular reduction. It is well known that fast modular reduction is possible

with moduli of the form $2^n \pm c$, where c is a *small* integer[12]. Integers of this form allow modular reduction without division. The operators << and >> mean **left shift** and **right shift**, respectively.

Algorithm : Reduction modulo $p = 2^m - c$, where $\log_2 c \leq \frac{1}{2}n$.
INPUT : a base 2 positive integer $x < p^2$ and a modulus p.
OUTPUT : $r \equiv x \ mod \ p$.
1. $q_0 \leftarrow (x >> n)$, $r_0 \leftarrow x - (q_0 << n)$, $r \leftarrow r_0$, $i \leftarrow 0$.
2. While $q_i > 0$ do the following:
   2.1 $q_{i+1} \leftarrow q_i c >> n$, $r_{i+1} \leftarrow q_i c - (q_{i+1} << n)$.
   2.2 $i \leftarrow i + 1$, $r \leftarrow r + r_i$.
3. While $r \geq p$ do: $r \leftarrow r - p$.
4. Return(r).

Remark 2. Algorithm 1 can be modified if $p = 2^n + c$ for some positive integer c such that $\log_2 c \leq \frac{1}{2}n$: in step 2.2, replace $r \leftarrow r + r_i$ with $r \leftarrow r + (-1)^i r_i$.

The Algorithm 1 terminates after a maximum of two iterations of the while loop, so we require at the most two multiplications by c. In practice, this leads to a dramatic performance enhancement over performing explicit division with remainder. If $p = 2^n + c$ is used then the number of iterations of the second while loop is smaller than that of $p = 2^n - c$ used in step 3. Since Algorithm 1 is organized with shift, addition, subtraction and multiplication by c, and if c is well chosen then multiplication by c can be substituted by shift the reduction of the number of subtraction is meaningful.

5 Implementation Results

We have implemented various field and XTR arithmetic using the techniques presented in previous Sections on typical microprocessors: Pentium III/700MHz (32-bit μP; Windows 2000, MSVC).

5.1 Application to XTR

In this Section 5.1,first,we propose optimized parameters for XTR such as prime p and a defining polynomial in Table 1. For the convenience to find primes, we give the prime p with the size of prime q is greater than 160bit. But we can also find prime p such that the size of prime q is as large as the original XTR.

In this paper, we used Karatsuba-like Method to achieve the efficiency of multiplication in $GF(p^{2m})$. Table 2 shows the number of multiplication, addition and subtraction in $GF(p)$ required to compute multiplication in $GF(p^{2m})$. Here, the defining polynomial is used as AOP. Note that values in the parenthesis are the numbers when Schoolbook method is used.

extension field	characteristic p	$f(x)$	the size q
$GF(p^6) \rightarrow GF(p^2)$	$2^{174}+7$	AOP	323 bit
$GF(p^{12}) \rightarrow GF(p^4)$	$2^{88}+7$	AOP	234 bit
$GF(p^{18}) \rightarrow GF(p^6)$	$2^{61}+15$	AOP	354 bit
$GF(p^{36}) \rightarrow GF(p^{12})$	$2^{30}+3$	AOP	302 bit
$GF(p^{66}) \rightarrow GF(p^{22})$	$2^{16}-17$	AOP	320 bit

Table 1. Construction of extension fields for XTR

extension field	Mul in $GF(p)$	Add in $GF(p)$	Sub in $GF(p)$
$GF(p^6) \rightarrow GF(p^2)$	3	5	2
$GF(p^{12}) \rightarrow GF(p^4)$	9(16)	12(11)	12(4)
$GF(p^{18}) \rightarrow GF(p^6)$	18(36)	39(29)	23(6)
$GF(p^{36}) \rightarrow GF(p^{12})$	54(144)	130(131)	118(12)
$GF(p^{66}) \rightarrow GF(p^{22})$	147(484)	345(461)	377(22)

Table 2. Multiplication in $GF(p^{2m})$ by using Karatsuba-like Method

In Table 3 we give the results of computing the linear recurrence $S_n(c)$ over $GF(p^{2m})$ from the values in Table 1,2,6.

extension field	prime p	$S_n(c)$ ($m$sec)
$GF(p^6) \rightarrow GF(p^2)$	$2^{174}+7$	10.174
$GF(p^{12}) \rightarrow GF(p^4)$	$2^{88}+7$	28.393
$GF(p^{18}) \rightarrow GF(p^6)$	$2^{61}+15$	50.276
$GF(p^{36}) \rightarrow GF(p^{12})$	$2^{30}+3$	4.932
$GF(p^{66}) \rightarrow GF(p^{22})$	$2^{16}-17$	6.789

Table 3. Comparison of $S_n(c)$ performance

It can be seen that $GF((2^{30}+3)^{12})$ yields XTR single exponentiation speeds which are more than twice as fast as the original XTR single exponentiation[10].

From the results of Table 4, we can see the need for a pseudo-Mersenne prime and Karatsuba-like Method to speed up XTR single exponentiation.

Finally, we recommend parameters of GOEF for XTR in Table 5.

6 Conclusion

We presented various speed-up techniques for field arithmetic in $GF(p^m)$ and introduced a class of finite fields, known as Generalized Optimal Extension Fields, which take advantage of well-known optimizations for finite field arithmetic on

extension field	prime p	Method of multiplication	$f(x)$	$S_n(c)$ ($m$sec)
$GF(p^6)$	general	Karatsuba method[10]	AOP	11.910
$\Rightarrow GF(p^2)$	$2^{174}+7$	Karatsuba method[10]	AOP	10.174
$GF(p^{36})$ $\Rightarrow GF(p^{12})$	general	Schoolbook method	AOP	129.338
	general	Karatsuba method(Appendix)	AOP	67.822
	$2^{30}+3$	Schoolbook method	AOP	8.218
	$2^{30}+3$	Karatsuba method(Appendix)	AOP	4.932

Table 4. Comparison XTR in $GF(p^2)$ [6,10] with XTR in $GF(p^{12})$

extension field	prime p	$f(x)$	the size q
$GF(p^{36})$ $\Rightarrow GF(p^{12})$	$2^{30}+3$	$x^{12}-2$	302 bit
	$2^{30}+7, (7=2^3-1)$	AOP	355 bit
	$2^{30}+129, (129=2^7+1))$	AOP	351 bit
	$2^{30}-257, (257=2^8+1))$	AOP	269 bit
	$2^{30}-513, (513=2^9+1)$	AOP	341 bit
	$2^{30}-513$	$x^{12}-3$	341 bit

Table 5. Recommended primes and $f(x)$ for $GF(p^{12})$

microprocessors. The main improvements presented in this paper consist of optimization in field multiplication and careful choices of field parameters to speed up field arithmetic. With above results, we proposed the optimized extension field for XTR that is $GF(p^{36}) \rightarrow GF(p^{12})$. The defining polynomial of $GF(p^{12})$ is the 12-th AOP and a candidate of nice prime is $2^{30}+3$. From our implementation results, our proposed field is about twice faster than the XTR[6] to compute $Tr(g^n)$. The key size of it is equal as that of the original XTR system. So our proposed optimal extension field for XTR is the more excellent alternative to either RSA or ECC than XTR[6] in applications such as SSL/TLS(Secure Sockets Layer, Transport Layer Security), public key smartcards, WAP/WTLS(Wireless Application Protocol, Wireless Transport Layer Security).

References

1. Aho,A.,Hopcroft,J.,Ullman,J., *The Design and Analysis of Computer Algorithms.*, Addison-Wesley,Reading Mass,1974.
2. Bach,E, Shallit,J., *Algorithmic Number Theory.*, Vol 1, The MIT Press, Mass, 1996.
3. Bailey. D.V. and Paar C, *Optimal extension fields for fast arithmetic in public-key algorithms.*, Crypto '98, Springer-Verlag pp.472-485, 1998.
4. H.Cohen, A.K. Lenstra, *Implementation of a new primality test.*, Math.Comp.48 (1987) 103-121.
5. D.E. Knuth, *The art of computer programming.*, Volume 2, Seminumerical Algorithms, second edition, Addison-Wesley, 1981.
6. A.K. Lenstra, E.R. Verheul, *The XTR public key system.*, Proceedings of Crypto 2000, LNCS 1880,Springer-Verlag, 2000,1-19; available from www.ecstr.com.

7. A.K. Lenstra, *Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields.*, Proceedings of ACISP 1997, LNCS 1270,Springer-Verlag, 1997,127-138.
8. A.K. Lenstra, *Lip 1.1*, available at www.ecstr.com.
9. Seongan Lim, Seungjoo Kim, Ikkwon Yie, Jaemoon Kim, Hongsub Lee, *XTR Extended to GF(p^{6m}).* Procee dings of SAC 2001,317-328, LNCS 2259, Springer-Verlag, 2001,125-143.
10. Martijn Stam, A.K. Lenstra, *Speeding Up XTR.* Proceedings of Asiacrypt 2001, LNCS 2248, Springer-Verlag, 2001,125-143; available from www.ecstr.com.
11. A.J Menezes, *Applications of Finite Fields.*, Waterloo, 1993.
12. S.B.Mohan and B.S.Adiga, *Fast Algorithms for Implementating RSA Public Key Cryptosystem.*, Electronics Letters, 21917):761,1985.
13. S.Oh, S.Hong,D.Cheon,C.Kim,J.Lim and M.Sung, *An Extension Field of Characteristic Greater than Two and its Applicatins.* Technical Report 99-2, CIST,1999. Available from http://cist.korea.ac.kr/.

Appendix

A. Comparison of Arithmetic Performance in $GF(p)$

The first three results were made with freelip version 1.1 [8] and the last two were made by our own hands.

extension field	prime p	Mul (μsec)	Add (μsec)	Sub (μsec)
$GF(p^6) \rightarrow GF(p^2)$	$2^{174}+7$	7.884	0.761	0.431
$GF(p^{12}) \rightarrow GF(p^4)$	$2^{88}+7$	4.095	0.53	0.67
$GF(p^{18}) \rightarrow GF(p^6)$	$2^{61}+15$	3.365	0.5	0.31
$GF(p^{36}) \rightarrow GF(p^{12})$	$2^{30}+3$	0.131	0.006	0.006
$GF(p^{66}) \rightarrow GF(p^{22})$	$2^{16}-17$	0.0507	0.006	0.006

Table 6. GOEF arithmetic timings on a 700 MHz

B. Karatsuba-like Multiplication of Two Elements with Degree 12

For a prime p and $A, B \in GF(p^{12})$, let $A = a_1x + a_2x^2 + \cdots + a_{12}x^{12}$ and $B = b_1x + b_2x^2 + \cdots + b_{12}x^{12}$
Now, we shall compute $AB = C = c_1x + c_2x^2 + \cdots + c_{12}x^{12} \bmod 1 + x + \cdots + x^{12}$ using Karatsuba-like multiplication where $a_i, b_i \in GF(p)$.
We previously compute the followings.

Step 1

$$
\begin{array}{lll}
G_1 = a_1 b_1 & G_2 = a_2 b_2 & G_3 = G_{17} G_{18} \\
G_4 = G_3 - G_1 - G_2 & G_5 = a_3 b_3 & G_6 = a_4 b_4 \\
G_7 = G_{19} G_{20} & G_8 = G_7 - G_5 - G_6 & G_9 = (a_1 + a_3)(b_1 + b_3) \\
G_{10} = (a_2 + a_4)(b_2 + b_4) & G_{11} = (G_{17} + G_{19})(G_{18} + G_{20}) & G_{12} = G_9 - G_1 - G_5 \\
G_{13} = G_{11} - G_4 - G_8 & G_{14} = G_{10} - G_2 - G_6 & G_{15} = G_{12} + G_2 \\
G_{16} = G_{14} + G_5 & G_{17} = a_1 + a_2 & G_{18} = b_1 + b_2 \\
G_{19} = a_3 + a_4 & G_{20} = b_3 + b_4 &
\end{array}
$$

Step 2

$$
\begin{array}{lll}
H_1 = a_5 b_5 & H_2 = a_6 b_6 & H_3 = H_{17} H_{18} \\
H_4 = H_3 - H_1 - H_2 & H_5 = a_7 b_7 & H_6 = a_8 b_8 \\
H_7 = H_{19} H_{20} & H_8 = H_7 - H_5 - H_6 & H_9 = (a_5 + a_7)(b_5 + b_7) \\
H_{10} = (a_6 + a_8)(b_6 + b_8) & H_{11} = (H_{17} + H_{19})(H_{18} + H_{20}) & H_{12} = H_9 - H_1 - H_5 \\
H_{13} = H_{11} - H_4 - H_8 & H_{14} = H_{10} - H_2 - H_6 & H_{15} = H_{12} + H_2 \\
H_{16} = H_{14} + H_5 & H_{17} = a_5 + a_6 & H_{18} = b_5 + b_6 \\
H_{19} = a_7 + a_8 & H_{20} = b_7 + b_8 &
\end{array}
$$

Step 3

$$
\begin{array}{lll}
I_1 = I_{17} I_{18} & I_2 = I_{21} I_{22} & I_3 = (G_{17} + I_{17})(G_{18} + I_{18}) \\
I_4 = I_3 - I_1 - I_2 & I_5 = I_{19} I_{20} & I_6 = I_{23} I_{24} \\
I_7 = I_{25} I_{26} & I_8 = I_7 - I_5 - I_6 & I_9 = I_{27} I_{28} \\
I_{10} = I_{29} I_{30} & I_{11} = (I_{27} + I_{29})(I_{28} + I_{30}) & I_{12} = I_9 - I_1 - I_5 \\
I_{13} = I_{11} - I_4 - I_8 & I_{14} = I_{10} - I_2 - I_6 & I_{15} = I_{12} + I_2 \\
I_{16} = I_{14} + I_5 & I_{17} = a_1 + a_5 & I_{18} = b_1 + b_5 \\
I_{19} = a_3 + a_7 & I_{20} = b_3 + b_7 & I_{21} = a_2 + a_6 \\
I_{22} = b_2 + b_6 & I_{23} = a_4 + a_8 & I_{24} = b_4 + b_8 \\
I_{25} = G_{19} + I_{19} & I_{26} = G_{20} + I_{20} & I_{27} = I_{17} + I_{19} \\
I_{28} = I_{18} + I_{20} & I_{29} = I_{21} + I_{23} & I_{30} = I_{22} + I_{24}
\end{array}
$$

Step 4

$$
\begin{array}{lll}
J_1 = a_9 b_9 & J_2 = a_{10} b_{10} & J_3 = J_{17} J_{18} \\
J_4 = J_3 - J_1 - J_2 & J_5 = a_{11} b_{11} & J_6 = a_{12} b_{12} \\
J_7 = J_{19} J_{20} & J_8 = J_7 - J_5 - J_6 & J_9 = J_{21} J_{22} \\
J_{10} = J_{23} J_{24} & J_{11} = (J_{17} + J_{19})(J_{18} + J_{20}) & J_{12} = J_9 - J_1 - J_5 \\
J_{13} = J_{11} - J_4 - J_8 & J_{14} = J_{10} - J_2 - J_6 & J_{15} = J_{12} + J_2 \\
J_{16} = J_{14} + J_5 & J_{17} = a_9 + a_{10} & J_{18} = b_9 + b_{10} \\
J_{19} = a_{11} + a_{12} & J_{20} = b_{11} + b_{12} & J_{21} = a_9 + a_{11} \\
J_{22} = b_9 + b_{11} & J_{23} = a_{10} + a_{12} & J_{24} = b_{10} + b_{12}
\end{array}
$$

Step 5

$$K_1 = K_{17}K_{18} \quad K_2 = K_{21}K_{22} \quad K_3 = K_{29}K_{30}$$
$$K_4 = K_3 - K_1 - K_2 \quad K_5 = K_{19}K_{20} \quad K_6 = K_{23}K_{24}$$
$$K_7 = K_{31}K_{32} \quad K_8 = K_7 - K_5 - K_6 \quad K_9 = K_{25}K_{26}$$
$$K_{10} = K_{27}K_{28} \quad K_{11} = (K_{25} + K_{27})(K_{26} + K_{28}) \quad K_{12} = K_9 - K_1 - K_5$$
$$K_{13} = K_{11} - K_4 - K_8 \quad K_{14} = K_{10} - K_2 - K_6 \quad K_{15} = K_{12} + K_2$$
$$K_{16} = K_{14} + K_5 \quad K_{17} = a_1 + a_9 \quad K_{18} = b_1 + b_9$$
$$K_{19} = a_3 + a_{11} \quad K_{20} = b_3 + b_{11} \quad K_{21} = a_2 + a_{10}$$
$$K_{22} = b_2 + b_{10} \quad K_{23} = a_4 + a_{12} \quad K_{24} = b_4 + b_{12}$$
$$K_{25} = K_{17} + K_{19} \quad K_{26} = K_{18} + K_{20} \quad K_{27} = K_{21} + K_{23}$$
$$K_{28} = K_{22} + K_{24} \quad K_{29} = K_{17} + K_{21} \quad K_{30} = K_{18} + K_{22}$$
$$K_{31} = K_{19} + K_{23} \quad K_{32} = K_{20} + K_{24}$$

Step 6

$$L_1 = (K_{17}+a_5)(K_{18}+b_5) \quad L_2 = (K_{21}+a_6)(K_{22}+b_6) \quad L_3 = (K_{29}+H_{17})(K_{30}+H_{18})$$
$$L_4 = L_3 - L_1 - L_2 \quad L_5 = (K_{19}+a_7)(K_{20}+b_7) \quad L_6 = (K_{23}+a_8)(K_{24}+b_8)$$
$$L_7 = (K_{31}+H_{19})(K_{32}+H_{20}) \quad L_8 = L_7 - L_5 - L_6 \quad L_9 = L_{17}L_{18}$$
$$L_{10} = L_{19}L_{20} \quad L_{11} = (L_{17}+L_{19})(L_{18}+L_{20}) \quad L_{12} = L_9 - L_1 - L_5$$
$$L_{13} = L_{11} - L_4 - L_8 \quad L_{14} = L_{10} - L_2 - L_6 \quad L_{15} = L_{12} + L_2$$
$$L_{16} = L_{14} + L_5 \quad L_{17} = I_{27} + J_{21} \quad L_{18} = I_{28} + J_{22}$$
$$L_{19} = I_{29} + J_{23} \quad L_{20} = I_{30} + J_{24}$$

Step 7

$$M_1 = I_1 - G_1 - H_1 + G_{16} \quad M_2 = I_4 - G_4 - H_4 + G_8$$
$$M_3 = I_{15} - G_{15} - H_{15} + G_6 \quad M_4 = I_{16} - G_{16} - H_{16} + H_1$$
$$M_5 = I_8 - G_8 - H_8 + H_4 \quad M_6 = I_6 - G_6 - H_6 + H_{15}$$
$$Q = H_{13} + K_{13} - G_{13} - J_{13}$$

Step 8

The following are the table of the coefficients of multiplication of two polynomials after reduction $mod\ 1 + x + \cdots + x^{12}$.

Degree	Coefficients	Addition	Subtraction
1	$H_{16} + K_{16} + L_1 - K_1 - H_1 - M_1 - J_{16} - Q$	2	5
2	$G_1 + H_8 + K_8 + L_4 - K_4 - H_4 - M_2 - J_8 - Q$	3	5
3	$G_4 + H_6 + K_6 + L_{15} - K_{15} - H_{15} - M_3 - J_6 - Q$	3	5
4	$G_{15} + L_{13} - K_{13} - H_{13} - I_{13} + G_{13} + H_{13} - Q$	3	4
5	$G_{13} + H_1 + L_{16} - K_{16} - H_{16} - M_4 + J_1 - Q$	3	4
6	$M_1 + H_4 + L_8 - K_8 - H_8 - M_5 + J_4 - Q$	3	4
7	$M_2 + H_{15} + L_6 - K_6 - H_6 - M_6 + J_{15} - Q$	3	4
8	$M_3 + J_{13} - Q$	1	1
9	$I_{13} - G_{13} - H_{13} + J_{16} - Q$	1	3
10	$K_1 - G_1 - J_1 + M_4 + J_8 - Q$	2	3
11	$K_4 - G_4 - J_4 + M_5 + J_6 - Q$	2	3
12	$K_{15} - G_{15} - J_{15} + M_6 - Q$	1	3
Sum		**27**	**44**

Table 7. Amount of operation in a reduction step

Step	1	2	3	4	5	6	7	8	**Total**
Multiplication	9	9	9	9	9	9	0	0	54
Addition	12	12	20	12	20	20	7	27	130
Subtraction	10	10	10	10	10	10	14	44	118

Table 8. Total amount of operation