Non-Conventional Basis of Finite Fields - implementing a fast communication between two elliptic curve cryptosystems in software and hardware.

Sang Ho Oh, Chang Han Kim, Joong Chul Yoon, Hee Jin Kim, and Jong In Lim

Abstract— **Finite field arithmetic is becoming increasingly important in cryptographic applications. In particular cryptographic primitives based on the discrete logarithm problem over elliptic curve groups are accomplished essentially by arithmetic in finite fields. It is well known that the efficiency of finite field arithmetic depends strongly on the particular way in which the field elements are represented. The finite field representation can be classified according to the choice of basis - a polynomial basis in software implementation and a normal basis in hardware implementation conventionally. The big problems of the communication between one Elliptic Curve Cryptosystem(ECC) in software implementation and another ECC in hardware implementation result from the difference in the choice of basis. In this paper we discuss the cost of the communication between such cryptosystems and propose the use of a non-conventional basis representation[3] providing the improved communicaton.**

Keywords— **Finite Fields, Basis Conversion, Elliptic Curve, and Public Key.**

I. Introduction

Elliptic Curve Cryptosystem(ECC) is well-suitable for use in constrained environments[1] such as mobile devices and smart cards, since it provides more efficient, high-strength security with smaller key sizes than any known Public-key cryptosystem. We focus on the interoperability of a hardware-based system(e.g. cryptographic VLSI chip-embedded mobile devices) and a software-based system (e.g. smart cards) for ECC. The implementation of ECC is accomplished essentially by arithmetic in finite fields, in particular $GF(2^n)$ of characteristic two or $GF(p)$ of odd prime p. In VLSI implementation or smart card with no additional coprocessor for modular exponentiation the common choice for the underlying finite field is a class of fields $GF(2^n)$. The efficient computation of field arithmetic depends greatly on the particular ways in which the field elements are represented. These most efficient ways are a polynomial basis representation and a normal basis representation. The difficulty of communication between these two systems for ECC results from the choice of basis; the most common choices of basis for software implementation and hardware implementation are a polynomial basis and a normal basis respectively. Interoperability between these systems using the two different types of field representation needs a conversion of basis by a appropriate change-of-basis matrix. Since the size of key for ECC is becoming increasingly large in proportion to the growth of computing power, this method seems to be a significant burden for the space-sensitive systems such as mentioned above. To enhance usability of the space-sensitive systems Kaliski Jr.*et al* presented algorithm for the storage-efficient finite field basis conversion[2]. The time complexity of this algorithm for a basis conversion amounts to $10\% \sim 20\%$ of one elliptic curve scalar multiplication according to a choice of basis. Such a fact gives us some motive for the communication with no basis conversion.

In this paper we discuss the details for the communication of systems using two different basis representations and propose the use of non-conventional basis representation providing the improved communication. Hereafter let's assume a matrix[2] for a basis conversion to be precomputed and consider finite fields $GF(2^n)$.

II. Description of Elliptic Curve Cryptosystems

In this section we describe briefly a public-key encryption and a digital signature for elliptic curve[3]and look into how a communication between two ECCs using different bases is performed.

A. Public-Key Encryption Scheme

Alice(using system A) wants to send a secure message M to Bob(using System B).

Case I: Between two systems using one basis

a. Setup
1. $E(a,b)$[4] is an elliptic curve defined over the field $GF(2^n)$.
2. P is a base point of prime order N in $E(GF(2^n))$.

b. Key Generation
1. Choose a statistically unique and unpredictable integer

S. H. Oh, J. C. Yoon,and J. I. Lim are with the Department of Mathematics, Korea University, Seoul, Korea. E-mail: gausmath@bora.dacom.co.kr .

H. J. Kim is with the Telemann Co., Seoul, Korea.

C. H. Kim is with the Department of Computational Mathematics, Semyung University, Jecheon, Korea. E-mail: chkim235@chollian.net . Research supported by The Basic Research Grants(97-0100-13-01-5),Korea Science and Engineering Foundation, 1997

[1] limits in memory usage and processing power.

[2] n-by-n matrix with entries in $GF(2)$.

[3] American National Standard X9.62 and X9.63.

[4] As a matter of convenience we denote a elliptic curve E as a pair of field elements E(a,b) where $E : y^2 + xy = x^3 + ax^2 + b$.

$1 < d < N$.
2. Compute $Q = dP$.
3. Bob has the private key d and announces the public key Q.

c. Encryption
1. Choose a statistically unique and unpredictable integer $1 < k < N$.
2. Compute $P' = kP$, $Q' = kQ$ and $C = Q' \bigoplus M$ [5].
3. Alice sends (P', C) to Bob.

d. Decryption
1. Compute $dP' \bigoplus C$.
2. Bob obtains Alice's message M.

Case II: Between two systems using different bases

Suppose a basis B_0 of system A differs from another basis B_1 of system B and let Γ be a change-of-basis matrix from B_0 to B_1.

First of all for system B's user to decrypt a message M the parameter $E(a, b)$ and the point P' defined in system A must be converted to adequate forms(denoted by $\overline{E(a,b)}$ and $\overline{P'}$ respectively) by a matrix Γ, since they were represented by a basis B_0.[6]

d′. Decryption
1. Compute $d\overline{P'}$.
2. Convert $d\overline{P'}$ to dP' by a matrix Γ^{-1}.
3. Compute $dP' \bigoplus C$.
4. Bob obtains Alice's message M.

Note 1: $d\overline{P'} = \overline{dP'}$ where $\overline{dP'}$ is the converted form of dP' by a matrix Γ

In order to decrypt a message M we must convert 6 field elements. A basis conversion and a field multiplication in a finite field $GF(2^n)$ require $2n^2 - n$ and n^2 bit operations respectively. As a result, a communication between two systems using different bases takes additionally times for 12 field multiplications and storages for a matrix Γ and an inverse matrix of Γ.

B. Digital Signature Scheme

Alice(using system A) wants to send a message M with her signature (r,s) to Bob(using System B).

Case I: Between two systems using one basis

The Setup and Key-Generation of Digital Signature Scheme is identical with those of Public-Key Encryption Scheme.

c. Signature Generation
1. Choose a statistically unique and unpredictable integer $1 < k < N$.
2. Compute $kP = (x_1, y_1)$.
3. Compute $r = x_1 \bmod N$.
4. Compute $e = H(M)$[7].
5. Compute $s = k^{-1}(e + dr) \bmod N$.
6. Alice sends $(r, s), M$ to Bob.

d. Signature Verification
1. Compute $e = H(M)$.
2. Compute $s^{-1} \bmod N$.
3. Compute $u_1 = es^{-1} \bmod N$.
4. Compute $u_2 = rs^{-1} \bmod N$.
5. Compute $u_1P + u_2Q = (x_1, y_1)$.
6. Compute $v = x_1 \bmod N$.
7. Bob accepts the signature if $v = r$.

Case II: Between two systems using two bases

Like a case of Public-Key Encryption, consider a basis B_0 of system A, a basis B_1 of system B and a change-of-basis matrix Γ.

The verification of a signature (r, s) can be described using the converted forms (denoted by $\overline{E(a,b)}$, $\overline{P}$ and $\overline{Q}$ respectively) of the parameter $E(a, b)$, the point P and the point Q by a matrix Γ.

d′. Signature Verification
1. Compute $e = H(M)$.
2. Compute $s^{-1} \bmod N$.
3. Compute $u_1 = es^{-1} \bmod N$.
4. Compute $u_2 = rs^{-1} \bmod N$.
5. Compute $u_1\overline{P} + u_2\overline{Q} = (\overline{x_1}, \overline{y_1})$, where $\overline{x_1}$ and $\overline{y_1}$ are the converted forms of x_1 and y_1 by a matrix Γ.
6. Convert $\overline{x_1}$ to x_1 by a matirix Γ^{-1}.
7. Compute $v = x_1 \bmod N$.
8. Bob accepts the signature if $v = r$.

Note 2: $u_1\overline{P} + u_2\overline{Q} = \overline{kP}$ where $\overline{kP}$ is the converted form of kP by a matrix Γ, from the following fact:

$$\begin{aligned} u_1\overline{P} + u_2\overline{Q} &= (es^{-1} \ mod \ N)\overline{P} + (rs^{-1} \ mod \ N)(d\overline{P}) \\ &= (es^{-1} \ mod \ N)\overline{P} + (drs^{-1} \ mod \ N)\overline{P} \\ &= ((e + dr)s^{-1} \ mod \ N)\overline{P} \\ &= k\overline{P} = \overline{kP}. \end{aligned}$$

Also in case of Digital Signature a communication between two systems using different bases takes additionally times for 14 field multiplications and spaces for a matrix Γ and an inverse matrix of Γ.

III. Space and Time Complexities

Compared to the time complexity of a scalar multiplication [8] over an elliptic curve group, that of a basis conversion is hardly critical. But the space complexity of two change-of-basis matrices Γ and Γ^{-1}, that is the additional storage of $2n^2$ bits is considered to be a serious load in practical implemetations, in particular in hardware implemetation. Recently the time complexity of algorithms of the stoage-efficient finite field basis conversion introduced by Kaliski Jr.*et al*[2] amounts to 10% $\sim$ 20% of one elliptic curve scalar multiplication according to a choice of basis.

[5] M can be consider as a bits-string and $\bigoplus$ is an exclusive OR.
[6] A conversion of basis may be done before or after sending a message .
[7] H is a hash function
[8] it needs $10n$ field multiplications and $6.5n$ field squarings.

The cryptosystems based on the arithmetic operations of finite fields depend greatly on the finite field basis representation. It is known typically that a polynomial basis representation and a normal basis representation are most suitable for a software implementation and a hardware implementation respectively. The communication of two cryptosystems in software implementation and hardware implementation needs a basis conversion. Now we are ready for introducing a non-conventional basis representation .

IV. Non-Conventional Basis Representation

In this section we show that the proposed basis representation gives the very efficient results in both of software implementation and hardware implementation in comparison with the previous results.

Definition 1: If a subset $B = \{\alpha, \alpha^2, \alpha^3, \cdots, \alpha^n\}$ of $GF(2^n)$ is linearly independent over $GF(2)$, then we'll call a set B *an anomalous basis* of $GF(2^n)$ over $GF(2)$.

Let f be a monic irreducible polynomial of degree n over $GF(2)$ where α is a root of f. Then the set $\{\alpha, \alpha^2, \alpha^3, \cdots, \alpha^n\}$ is linearly independent over $GF(2)$. Note that an anomalous basis $\{\alpha, \alpha^2, \alpha^3, \cdots, \alpha^n\}$ is not equal to a polynomial basis $\{1, \alpha, \alpha^2, \cdots, \alpha^{n-1}\}$ over $GF(2)$. We'll pay attention to a special case of a generating polynomial f, that is all-one-polynomials(AOP), since the finite field arithmetic operations derived from such polynomials are performed very efficiently in hardware implementation.

A. Software Implementation

Let $B = \{\alpha, \alpha^2, \alpha^3, \cdots, \alpha^n\}$ be an anomalous basis of $GF(2^n)$ over $GF(2)$ where α is a root of AOP of the degree n. Let $\mathbf{a}$ and $\mathbf{b}$ be the elements of a finite field $GF(2^n)$. Then they can be represented by a basis B as follows:

$$\mathbf{a} = \Sigma_{i=1}^{n} a_{i-1}\alpha^i \text{ and } \mathbf{b} = \Sigma_{i=1}^{n} b_{i-1}\alpha^i$$

where $a_i, b_i \in GF(2)$ for each $i \in \{0, 1, 2, \cdots, n-1\}$.

The field multiplication consists of two steps, that is the product of two polynomials and the reduction using the identity $\alpha^{n+1} = 1$. The squaring operation is of the simple vector-form [9]. For example, if $n = 10$ then $\mathbf{a}^2$ can be described as $(a_5, a_0, a_6, a_1, a_7, a_2, a_8, a_3, a_9, a_4)$.

Note 3:

$$\begin{aligned}
\mathbf{a}^2 &= \left(\sum_{i=1}^{n} a_{i-1}\alpha^i\right)^2 \\
&= \left(\sum_{i=1}^{2m} a_{i-1}\alpha^i\right)^2 \\
&= \sum_{i=1}^{2m} a_{i-1}\alpha^{2i} \\
&= \sum_{i=1}^{m} a_{i-1}\alpha^{2i} + \sum_{i=m+1}^{2m} a_{i-1}\alpha^{2i} \\
&= \sum_{i=1}^{m} a_{i-1}\alpha^{2i} + \sum_{i=1}^{m} a_{m+i-1}\alpha^{2i-1}.
\end{aligned}$$

where $n = 2m$ for some positive integer m.

Almost Inverse algorithm[6] has been known as the most improved algorithm of multiplicative inversion. it is applicable to computing a multiplicative inverse of non-zero field elements represented by an anomalous basis. To describe multiplicative inversion and Almost Inverse algorithm we identify $GF(2^n)$ with a residue class of polynomial ring over $GF(2)$, that is

$$GF(2^n) \cong GF(2)[x]/(f)$$

Given a non-zero polynomial $A(x)$ of degree less than or equal to $n-1$, the inverse of $A(x)$ is a polynomial $B(x)$ of degree less than or equal to $n-1$ such that

$$A(x)B(x) \equiv 1 \ mod \ f.$$

Almost Inverse algorithm computes $B(x)$ and k such that

$$A(x)B(x) \equiv x^k \ mod \ f.$$

where degree of $B < n$ and a nonnegative integer k is uniquely determined. Suppose the degree of $A(x)$ is equal to n and $A(x)$ is not same with f. We'll show Almost Inverse algorithm to find the almost inverse $B(x)$ of $A(x)$.

Algorithm 1: (Almost Inverse Algorithm)
Intialize integer $k = 0$, and polynomials $B = 1, C = 0$,
$F = A$, $G = f$.
loop:

1. While the constant term of F is zero, do $F = F/x$, $C = C * x$, $k = k + 1$.
2. If $F = 1$, then return B, k.
3. If $\deg(F) < \deg(G)$, then exchange F, G and exchage B, C.
4. $F = F + G$, $B = B + C$.
   Goto loop.

Let $F(x) = x^{k_0} * F'(x)$ for a polynomial $F'(x)$ with the non-zero constant term and a positive[10] integer k_0. Almost Inverse algorithm applied to $F'(x)$ computes the almost inverse $B'(x)$ of $F'(x)$ such that

$$F'(x)B'(x) \equiv x^{k'} \ mod \ f$$

for a nonnegative integer k'. To compute the multiplicative inverse of $F(x)$ we need to divide $B'(x)$ by $x^{k_0+k'}$. The remainder is left to the reader.

[9] The coordinate vector $(a_0, a_1, \cdots, a_{n-1})$ with the ordered anomalous basis B can be interpreted as $\Sigma_{i=1}^{n} a_{i-1}\alpha^i$.

[10] The anomalous basis representation does not include a constant term.

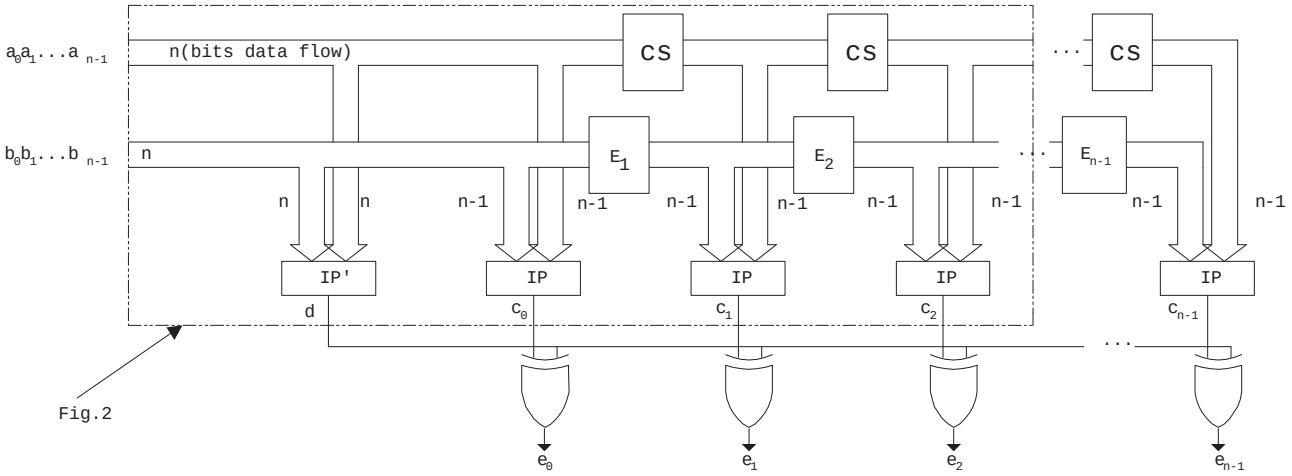

Fig. 1. Multiplier Architecture

Arithmetic in the fields $GF(2^{180})$ derived from an anomalous basis, was implemented in the C-language on a Pentium 120. In Table 1 we give the running times for the operations of squaring, multiplication and inversion in μ seconds and compare our results with those of [7] [11].

	anomalous basis using AOP	standard basis using trinomial
multiplication	51.4	71.8
squaring	1.5	2.7
inversion	161.4	225

Table 1. Time for field operations

B. Hardware Implementation

The hardware implementation of field arithmetic using an anomalous basis representation was introduced already in [3][12] . We'll brief the reader on a hardware architecture for a multiplier. The definitions and the notations in the section IV-A is also valid here. The multiplication of two field elements $\mathbf{a}, \mathbf{b}$ can be rewritten using matrix forms as follows:

$$\mathbf{a} \times \mathbf{b} = (C + D)[\mathbf{b}]_B = E$$

, where

$$C = \begin{pmatrix} 0 & a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_1 \\ a_0 & 0 & a_{n-1} & a_{n-2} & \cdots & a_2 \\ a_1 & a_0 & 0 & a_{n-1} & \cdots & a_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & a_{n-4} & a_{n-5} & \cdots & 0 \end{pmatrix}$$

$$, D = \begin{pmatrix} a_{n-1} & a_{n-2} & a_{n-3} & a_{n-4} & \cdots & a_0 \\ a_{n-1} & a_{n-2} & a_{n-3} & a_{n-4} & \cdots & a_0 \\ a_{n-1} & a_{n-2} & a_{n-3} & a_{n-4} & \cdots & a_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & a_{n-4} & \cdots & a_0 \end{pmatrix}$$

$$and \quad [\mathbf{b}]_B = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{pmatrix}$$

Let $C[\mathbf{b}]_B = (c_0, c_1, c_2, \cdots, c_{n-1})$, $D[\mathbf{b}]_B = (d, d, d, \cdots, d)$ and $E = (e_0, e_1, e_2, \cdots, e_{n-1})$. Then Fig 1. outlines the architecture for the parallel multiplier using an anomalous basis representation. Also Fig 2. gives a full detail of the multiplier architecture. The multiplier consists of three units, that is Inner Product(IP), Cyclic Shift(CS) and Exchange(E_i) as illustrated in Fig 3.

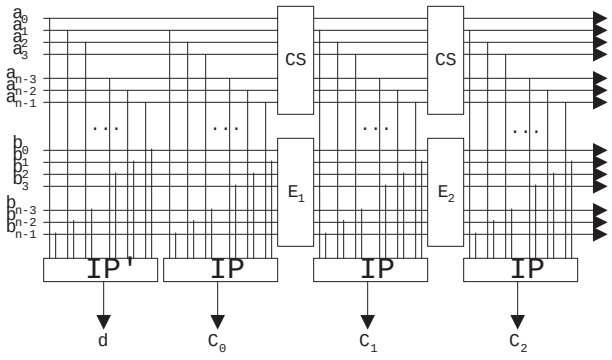

Fig. 2. Multiplier Architecture

A squaring operation is a rewiring with no delays and no gates as mentioned in Note 3. In Table 2 we list space(for gates) and time(for delays due to gates:D_A, D_X[13]) com-

[11]The work implemented arithmetic in $GF(2^{177})$ using Pentium 133.

[12]The multiplier mentioned here was originated from [3].

[13]D_A and D_X are delays for AND gates and XOR gates respectively.

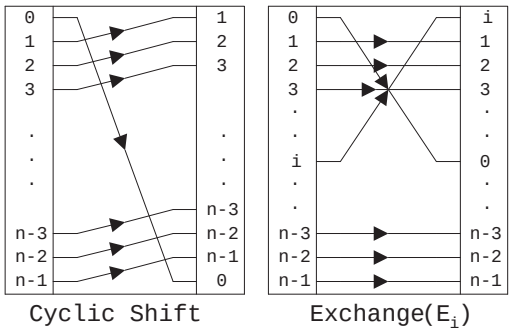

Fig. 3. Multiplier Architecture

plexities of the previous known parallel multipliers including the above multiplier. The following inverse algorithm[5] for computing a multiplicative inverse of a non-zero field element takes 9 to 12 multiplications and 149 to 199 squarings for $150 \leq n \leq 200$. For reference, an elliptic curve point addition using affine coordinates requires two field multiplications, a squaring and a multiplicative inversion. Therefore the total of 11 to 14 multiplications in progress of the inversion algorithm is required, while an elliptic curve point addition using projective coordinates needs 10 to 15 multiplications.

Algorithm 2 (Optimal Inverse Algorithm)
Input: a positive integer n and $\alpha \in GF(2^n)$.
Output: $\alpha^{-1} = \alpha^{2+2^2+\cdots+2^{n-1}}$.
1. Set $t \leftarrow n-1, x \leftarrow 1, u \leftarrow \alpha^2$.
2. While $t > 0$ do
  2.1 While $t_0 = 0$ and $t > 1$ do (where $(t_i)_2$ is the binary representation of t)
    2.1.1 $t \leftarrow t \gg 1$ (one right shift).
    2.1.2 Set $u \leftarrow u * u^{2^t}$.
  2.2 Set $x \leftarrow x * u$.
  2.3 If $t = 1$, then stop.
  2.4 else set $u \leftarrow u^2$.
  2.5 Set $t_0 \leftarrow 0$.

	multiplication (squaring)
anomalous basis (AOP)[3]	n^2 AND, n^2-1 XOR gates, $D_A + (1 + \lceil \log_2(n-1) \rceil) D_X$ Delays (Rewiring)
standard basis (AOP)[4]	n^2 AND, n^2-1 XOR gates, $D_A + (2 + \lceil \log_2(n-1) \rceil) D_X$ Delays ($n-1$ XOR gates, $1D_X$ Delay)
normal basis (type I)[1]	n^2 AND, n^2-1 XOR gates, $D_A + (1 + \lceil \log_2(n-1) \rceil) D_X$ Delays (Rewiring)

Table 2. Space and time complexities for field operations

V. Conclusion

Elliptic Curve Cryptography which offers the highest security for the same key-size of any known Public-key Cryptosystem plays a very important role in the practical applications of cryptography, in particular, in hardware implementation including cellular phones, smart cards, small-size computers(HPC) and the like. Let us assume that in years to come it is possible for you to pay securely for your car to a motor company using a cellular phone. Then maybe in both of your phone and the sever computer of the bank with which you do a deal, you may find Elliptic Curve Cryptosystems. If it happens, we can consider the communication between your phone and the server of the bank. Under the assumption that the former has an Elliptic Curve Cryptosystem in hardware implementation, while the latter has that in software implementation the communication needs a basis conversion. We have shown that a basis conversion is a hindrance to a fast communication. Such a fact gives us the motive for the communication with no basis conversion, that is the implementation by one basis. In conclusion, anomalous basis representation gives the very remarkable performance of arithmetic operations in both of hardware implementation and software implementation as compared to the previous results.

Acknowledgments

The authors would like to acknowledge the suggestions of many people.

References

[1] M. A. Hasan, M. Z. Wang, and V. K. Bhargava, *"A Modified Massey-Omura parallel multiplier for a class of finite fields"*, IEEE Transactions on Computers, v42, no.10, pp.1278-1280, October 1993.
[2] B. S. Kaliski Jr and Y. L. Yin, *"Storage-Efficient Finite Field Basis Conversion"*, Contribution to IEEE Standard P1363, 1998.
[3] C. H. Kim, S. H. Oh, and J. I. Lim, *"A New Hardware Architecture of Operations in $GF(2^n)$"*, Submitted to IEEE Transactions on Computers, 1998.
[4] C. K. Koc and B. Sunar, *"Low-complexity Bit-parallel Canonical and Normal Basis Multiplier for a Class of Finite Fields"*, IEEE Transactions on Computers, v47, no.3, pp.353-356, March 1998.
[5] S. H. Oh and C. H. Kim, *"Algorithm of Inverse Operation in $GF(2^n)$"*, Submitted to IEEE Transactions on Information Theory, 1998.
[6] R. Schroeppel, H. Orman, S. O'Malley, and O Spatscheck, *"Fast Key Exchange with Elliptic Curve Systems"*, Proc. Crypto'95, Springer-Verlag, 1995, pp.43-56.
[7] E. De Win, A. Bosselaers, S. Vandenberghe, P De Gersem, and J. Vandewalle, *"A Fast Software Implementation for Arithmetic Operations in $GF(2^t)$"*, Asia Crypto'96, Springer-Verlag, 1996, pp.65-76.
[8] IEEE P1363, *Standard Specifications For Public Key Cryptography, Annex A*, 1998.