Parallelized Scalar Multiplication on Elliptic Curves Defined over Optimal Extension Field

Jaewon Lee, Heeyoul Kim, Younho Lee, Seong-Min Hong, and Hyunsoo Yoon
(Corresponding author: Jaewon Lee)

Department of EECS, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
305-701, Guseong-dong, Yuseong-gu, Daejeon, Korea.
(Email: {jaewon, hykim, yhlee, smhong, hyoon}@camars.kaist.ac.kr, hykim@camars.kaist.ac.kr)

(Received Sept. 21, 2005; revised and accepted Nov. 5, 2005)

Abstract

In this paper, we propose three algorithms to perform scalar multiplication on elliptic curves defined over higher characteristic finite fields such as the OEF (Optimal Extension Field). First, we propose an efficient scalar multiplication method in which the Frobenius expansion is used on an elliptic curve defined over OEF. Second, we propose a new finite field multiplication algorithm. Third, we propose a particular polynomial squaring algorithm. We show that the proposed algorithms, when used together, accelerate the scalar multiplication on elliptic curves by two-fold.

Keywords: Elliptic curves, frobenius expansion, scalar multiplication

1 Introduction

The elliptic curve cryptosystem was proposed by Koblitz and Miller independently [15, 20]. Because it requires a smaller key size than an RSA-type cryptosystem, it is possibe to implement fast and compact cryptosystems such as ECDSA [17]. However, since elliptic curve operations are very complicated and require finite field operations, there has been much research on the fast implementation of elliptic curve cryptosystems. They can be categorized into two groups : high-level algorithms which manage elliptic curve points and low-level algorithms that accelerate finite field operations.

First, we describe high-level algorithms. The core operation is a scalar multiplication($k \cdot P$) of a point P on an elliptic curve in the elliptic curve cryptosystems. Because the scalar multiplication is composed of repeated point additions and doublings, the throughput of the cryptosystem is linearly dependent on the number of repetitions. To reduce the number of repetitions, an addition/subtraction-chain or a Frobenius map can be used but the latter is known to be more efficient [8].

The base-ϕ expansion method which uses the Frobenius map was proposed by Koblitz [16]. Koblitz's method can be used only for the points on the elliptic curves defined over $\mathbb{F}_2$. Muller and Cheon *et al.* extended it to $\mathbb{F}_{2^r}$, where r is a small integer less than 6 [6, 21]. Koblitz also extended it to $\mathbb{F}_3$ and $\mathbb{F}_7$ [17]. Recently, Kobayashi *et al.* proposed a base-ϕ expansion method that can be applied to $\mathbb{F}_{p^m}$-rational points on an elliptic curve defined over $\mathbb{F}_p$ where p is a very large prime such as 64-bit [13].

Now we describe low-level algorithms. To accelerate finite field operations, there have been many proposals of finite fields $\mathbb{F}_{2^m}$, $\mathbb{F}_{(2^m)^n}$, $\mathbb{F}_{p^m}$. $\mathbb{F}_{2^m}$ has been widely used because it is efficient in hardware implementation [1, 2, 3]. However, it is not appropriate for software implementation, because it comprizes many bit operations. To remove these bit operations, composite fields $\mathbb{F}_{(2^m)^n}$ have been proposed [10, 22]. Two representative composite fields are $\mathbb{F}_{(2^8)^{13}}$ and $\mathbb{F}_{(2^{16})^{11}}$. However, it is infeasible to use composite fields such as $\mathbb{F}_{(2^{32})^7}$ or $\mathbb{F}_{(2^{64})^5}$, because it requires two tables for sub-field multiplication and the table size is too large. This means that the composite field can not fully utilize the current 32-bit or 64-bit computer architecture. To overcome this problem, Bailey and Paar proposed $\mathbb{F}_{p^m}$ and named it the OEF (Optimal Extension Field) when p satisfies some conditions. Because such p is a prime near the register size, we can use the CPU's built-in operations such as multiplication and addition as sub-field operations. Currently, most efficient software implementations use the OEF.

In this paper, we propose three algorithms. First, we propose an efficient scalar multiplication method when base-ϕ expansion is used on an elliptic curve defined over the OEF. Whereas the base-ϕ expansion method requires less point additions and doublings than the conventional addition/subtraction-chain method, there must be a special computation method when the used elliptic curve is defined over a higher characteristic finite field such as an OEF.

Second, we propose a new algorithm to improve the finite field multiplication which is the most time-consuming

operation. When the polynomial basis representation is used, the multiplication can be implemented by using a polynomial multiplication algorithm. Among the existing polynomial multiplication algorithms, the KOA (Karatsuba-Ofman Algorithm) [12] shows the best performance when the number of terms is relatively small. However, the number of terms must be a power of 2, e.g., 4, 8, 16, $\cdots$. In other cases, the KOA shows very poor performance acceleration. This means that the KOA has not much advantage when the OEF is used. We propose a modified KOA that can be used with the OEF.

Third, we propose a new efficient polynomial squaring algorithm. Generally, polynomial squaring is implemented by usual multiplication algorithms. We propose a new divide-and-conquer algorithm which is faster than our new multiplication algorithm.

This paper is composed as follows. In Section 2, we explain the elliptic curve and the scalar multiplication. In Section 3, we propose a high-level algorithm that accelerates the scalar multiplication procedure using base-ϕ expansion. In Section 4, we propose a low-level algorithm that accelerates the finite field multiplications and squarings. In Section 5, we describe the implementation and the experimental result. Finally, we conclude this paper in Section 6.

2 Background

Elliptic curves are smooth curves of genus 1 having a specified basepoint O. Such curves can be written as the locus in the projective plane of a cubic equation called by the Weierstrass equation of the form, $E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$ with only one point (the base point) $O = (0 : 1 : 0)$ on the line at infinity. For ease of notation, we usually write the Weierstrass equation for our elliptic curve using non-homogeneous coordinates $x = X/Z$ and $y = Y/Z$, $E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, always remembering that there is the extra point O out at infinity. As usual, if $a_1, \cdots, a_6 \in K$ for some field K, then E is said to be defined over K. Moreover, if $(X : Y : Z) \in \mathbb{P}^2(K)$ satisfies the Weierstrass equation, then it is call the K-rational point and $E(K) = \{(X : Y : Z) \in \mathbb{P}^2(K) | (X : Y : Z)$is a K-rational point$\}$ forms an abelian group with an identity element O. Moreover, we may consider elliptic curves defined over $\mathbb{Z}/n\mathbb{Z}$ for an integer n. For more details, see [19].

The discrete log problem on the elliptic curve is to compute the constant k for a given elliptic curve point P and a scalar multiple $k \cdot P$, and this is known to be very difficult. Because this is similar to the discrete log problem on the finite field, previous cryptosystems based on the discrete log problem can be transformed to the elliptic curve cryptosystem, where the core operation is a scalar multiplication $k \cdot P$ for a given elliptic curve point P and a constant k.

3 Scalar Multiplication Using Frobenius Expansion

For a given elliptic curve E defined over $\mathbb{F}_q$, Frobenius map ϕ on $E(\mathbb{F}_q)$ is defined as follows: $\phi : (x, y) \rightarrow (x^q, y^q)$ Because $\phi^2 - t\phi + q = 0$ where t is a trace of ϕ (i.e., $q = t\phi - \phi^2$), the scalar multiplication by q can be replaced by the Frobenius map and a scalar multiplication by t. Repeating this procedure, we can replace any integer by a polynomial of the Frobenius map.

Theorem 1. *For a given elliptic curve E defined over $\mathbb{F}_q$, scalar multiplication by k on $E(\mathbb{F}_{q^n})$ can be represented as the following Frobenius equation.*

$$k = \sum_{j=0}^{n+1} c_j \phi^j, \quad where \quad c_j \in \{i \in Z | -q/2 < i \leq q/2\}$$

As we can see in Theorem 1, if an elliptic curve E is defined over a small finite field $\mathbb{F}_q$, the scalar multiplication $k \cdot P$ can be replaced by scalar multiplications by the smaller integers less than or equal to $q/2$ and the Frobenius map computation. The latter operation can be done by a simple algorithm and does not significantly affect the total computation time.

However, when the definition field is large such as OEF, because the coefficients (c_j's) are also large, an efficient computation method is required. Kobayashi *et al.* proposed two computation algorithms which use the table lookup in [13]. One uses Brickell *et al.*'s BGMW method [5], but this cannot be applied when the definition field is too large such as 2^{32} or 2^{64}. The other uses the bit patterns of the coefficients, and its complexity is the same as the left-to-right binary exponentiation method. We propose the improvement of the latter method in this section.

3.1 Proposed Computation Algorithm

We describe the proposed computation method. The basic idea is to use a batch operation technique by changing the computation sequence as follows.

$$kP = \sum_{j=0}^{m-1} c_j \phi^j P, \quad (1)$$

$$= (\cdots (c_{m-1}P\phi + c_{m-2}P)\phi + \cdots + c_2P)\phi + c_0P, \quad (2)$$

where $c_j \in \{i \in Z | -q/2 \times (m-1) < i \leq q/2 \times (m-1)\}$. Equation (1) is the result of the type II expansion that is described in [13], and the original computation sequence is to compute the repeated Frobenius map to the P (i.e., $\phi^j P$) first, and after this, to compute scalar multiplications by c_j's. But, if we follow Equation (2), we do the scalar multiplication (i.e., $c_j \cdot P$) first, and the Frobenius map afterwards. This can be abstracted as follows.

Table 1: Required time for scalar multiplication on the elliptic curve defined over $\mathbb{F}_{p^m}$, where $p = 2^{31} - 1$, m=7. The proposed algorithm uses type II expansion in [13], so the average Hamming weight is 11/32.

Algorithm	Point Addition	Point Doubling	Total
Kobayashi (type I)	$7b_p/2$(=108)	b_p	$9b_p/2$(=139)
Kobayashi (type II)	$77b_p/32$(=75)	b_p	$109b_p/32$(=106)
Proposed	$44b_p/32$(=43)	b_p	$76b_p/32$(=76)

Theorem 2. *We denote* $\lceil \log p \rceil + 1$ *by* b_p. *Let* E *be an elliptic curve defined over* $\mathbb{F}_p$. *Consider a scalar multiplication* $k \cdot P$, *where* P *is a* $\mathbb{F}_{p^m}$*-rational point of* E *and* k *is a* mb_p*-bit integer. This can be replaced by* $m-1$ *Frobenius map and* m *scalar multiplications on the same curve where the constants are* b_p*-bit integers.*

We describe the proposed computation algorithm using an example. For simplicity, we assume m=3 without loss of generality. That is, we show how to compute $a \cdot P$, $b \cdot P$, and $c \cdot P$ for the following three 12-bit integers.

$$a = 010100110111_2$$
$$b = 110001101100_2$$
$$c = 101100100101_2$$

First, we pre-compute $2 \cdot P$, $2^2 \cdot P$, $\cdots$, $2^{11} \cdot P$. We denote by $Comm(a, b, c)$ common bit stream of a, b, and c, and by $Comm(b, c)$ that of b and c. We compute the following values.

$$\alpha = Comm(a, b, c) = 000000100100_2$$
$$\beta = Comm(b, c) - Comm(a, b, c) = 100000000000_2$$
$$\gamma = a - Comm(a, b, c) = 010100010011_2$$
$$\delta = b - Comm(b, c) - Comm(a, b, c) = 010001001000_2$$
$$\eta = c - Comm(b, c) - Comm(a, b, c) = 001100000001_2$$

Then the result can be computed as follows.

$$a \cdot P = \gamma \cdot P + \alpha \cdot P$$
$$b \cdot P = \delta \cdot P + \beta \cdot P + \alpha \cdot P$$
$$c \cdot P = \eta \cdot P + \beta \cdot P + \alpha \cdot P$$

The number of required point doublings is 11 for precomputation. The number of required point additions is 13 (1 for $\alpha \cdot P$, 0 for $beta \cdot P$, 4 for $\gamma \cdot P$, 2 for $\delta \cdot P$, 2 for $\eta \cdot P$, and 5 for the final merging.). The following theorem is obtained by generalization.

Theorem 3. *For* m *random* b_p*-bit integers,* $a_0, a_1, \cdots, a_{m-1}$, *scalar multiplications* $(a_i \cdot P)$ *require* b_p*-1 point doublings and* $\frac{m+1}{2}\overline{Hw} - 1$ *point additions, where* $\overline{Hw}$ *means the average Hamming weight of* a_i*s.*

Theorems 2 and 3 allow the following statement : *For a given* $E\mathbb{F}_p$ *and a rational* $\mathbb{F}_{p^m}$*-point* P, *the scalar multiplication for a* $m \cdot b_p$*-bit integer can be done with* b_p*-1 point doublings and* $\frac{m+1}{2}\overline{Hw} - 1$ *point additions.*

We consider the OEF $\mathbb{F}_{p^m}$, where $p = 2^{31} - 1$ and $m = 7$, that is used in [14]. Whereas Kobayashi *et al.*'s scheme requires 106 point operations (addition or doubling), the proposed scheme requires only 76 point operations. Because the number of required Frobenius map operations is the same, there is no overhead. Therefore, the proposed method accelerates the scalar multiplication about 30%. Table 1 shows the required time for scalar multiplication. Among the number of required point operations, all the reduced operations are point additions which are more time consuming operations than point doublings. Therefore, we can expect a performance of upgrade more than 30%.

4 Faster Multiplication and Squaring on Finite Fields

4.1 Multiplication

Usually, polynomial multiplication algorithms can be used for multiplication of finite field elements in polynomial basis. For the finite field multiplication, the school book method [4] which is simple but poor, the KOA [18], or their hybrid method [9] is usually used. Among them, the KOA is the fastest for finite field multiplication in elliptic curve arithmetics. Since the KOA utilizes the divide-and-conquer technique, it suffers from skewed splitting or redundant computation, if the degree of polynomial is not $2^t - 1 (t = 1, 2, 3, \cdots)$ To overcome this problem, the even-and-odd technique was used in [18], which is intuitive and the best solution so far. But, this still causes redundancy.

However, for the cryptographic safety of the elliptic curve E over the finite field F_p, the order of E over F_{p^m}, $N_m (= \#E(F_{p^m}))$ should have a large prime factor [19]. In general, if n divide m, N_n can divide N_m. Hence, if m is even or the power of two, N_m can not have a large prime factor. Thus a finite field F_{p^m} with prime m, should be used for a safe elliptic curve cryptosystem, in which case the KOA can not work in its maximal efficiency.

From these reasons, we present a fast multiplication algorithm for a finite field, F_{p^m} where $m = 2^t + l$ $(1 \leq l < 2^t)$, in polynomial basis. Consider two polynomials of degree $m - 1$:

$$A(x) = a_{m-1}x^{m-1} + \cdots + a_1 x + a_0,$$
$$B(x) = b_{m-1}x^{m-1} + \cdots + b_1 x + b_0$$

They can be multiplied using the result and the intermediate terms from the multiplication procedure of two

Table 2: Number of multiplications in various algorithms

degree of poly.	school book method	KOA	proposed multiplication	proposed squaring
2	9	7	6	5
4	25	17	14	13
5	36	21	18	15
6	49	25	24	23
8	81	43	36	27
9	100	51	41	39
10	121	59	50	43
11	144	63	54	45
12	169	71	66	61
13	196	75	71	69
14	225	79	78	67

Table 3: Times needed for multiplication and squaring

	School book	KOA	propsed mul.	proposed sqr.
Pentium-Pro	10.89	6.55	5.66	4.94
UltraSparc-2	12.69	10.38	7.56	6.13

polynomials

$$A^{(1)}(x) = a_{m-2}x^{m-2} + \cdots + a_1x + a_0,$$
$$B^{(1)}(x) = b_{m-2}x^{m-2} + \cdots + b_1x + b_0$$

with additional $m-1$ coefficient multiplications. By repeating this procedure recursively, we get two polynomials $A^l(x), B^l(x)$ with degree $2^t - 1$, then apply the KOA to them. It can be formalized with Theorem 4.

Theorem 4. *Consider two arbitary polynomials in one variable of degree $m-1$, where $m = 2^t + l$, with coefficient in a finite field F_p. Those polynomials can be multiplied with :*

$$\#MUL \leq 2^{t\log_2 3} + ml - \frac{l(l-1)}{2}$$
$$\#ADD \leq 6m^{\log_2 3} - 8m + 2 + 5\left(ml - \frac{l(l-2)}{2}\right)$$

multiplications and additions, respectively, in F_p.

Refer appendix for the proof and an example of this theorem. This theorem results in fewer coefficient multiplications as shown in Figure 1(a).

4.2 Squaring

Finite field squaring is used in elliptic curve arithmetics at the same frequency with multiplication. But, in the previous works, squaring was considered as very simple operation because, in the finite fields of characteristic two, its time complexity is $O(n)$ or $O(1)$. However, in the higher characteristic finite field, squaring is time consuming operation as well and computed by same algorithm of multiplications. We noticed that squaring could be calculated in more efficient way than multiplying where the polynomial is degree of $3^t - 1(t = 1, 2, 3, \cdots)$.

For simplicity, suppose squaring a polynomial of degree two:

$$C(x) = \sum_{i=0}^{4} c_i x^i = (a_2x^2 + a_1x + a_0)^2 \tag{3}$$

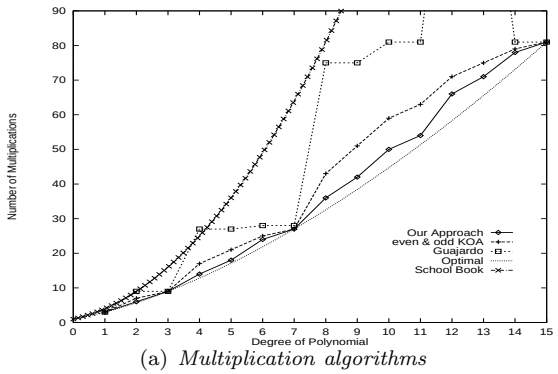
First of all, get the intermediate term ,d_i's, as follows:

$$\begin{aligned} d_0 &= a_0^2 \\ d_1 &= 2a_1a_0 \\ d_2 &= (a_2 + a_1 + a_0)^2 \\ d_3 &= 2a_2a_1 \\ d_4 &= a_2^2 \end{aligned} \tag{4}$$

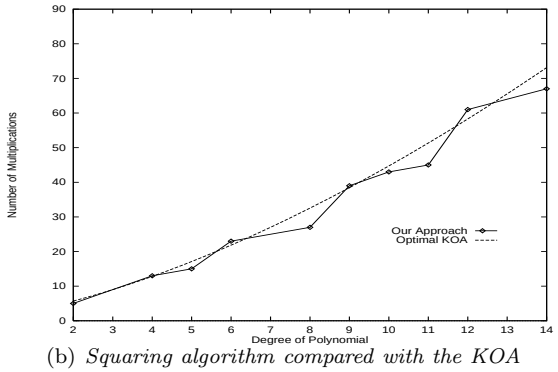
Using terms from Equation (4), the final result can be obtained as follows:

$$\begin{aligned} c_0 &= d_0 \\ c_1 &= d_1 \\ c_2 &= d_2 - (d_0 + d_1 + d_3 + d_4) \\ c_3 &= d_3 \\ c_4 &= d_4 \end{aligned} \tag{5}$$

Because in Equation (4), multiplication by two can be implemented with the shift or addition operation, so they are not counted as coefficient multiplications. Thus, the result of Equation (3) can be obtained with five coefficient multiplications while six is necessary in usual polynomial



(a) *Multiplication algorithms*



(b) *Squaring algorithm compared with the KOA*

Figure 1: Number of coefficient multiplications for each algorithm

multiplication algorithms. This procedure can be applied recursively and it needs fewer number of coefficient multiplications than usual polynomial multiplication algorithms. If the degree of polynomial is not $3t-1$ ($t \geq 2$), the squaring procedure can go on with our multiplicatin algorithm. The number of coefficient multiplicatins for polynomials of degree 2 through 14 is plotted in Figure 1(b).

5 Implementation

5.1 Environment

Our implemenation environment is as follows. We used a PC with Pentium Pro 200 MHz microprocessor and UltraSparc-2 168 MHz workstation. Usually a CISC (Complex Instruction Set Computer) based PC and a RISC (Reduced Instruction Set Computer) based workstation possess different characteristics in various sides. Since RISC architecture is slower than CISC in single-precision (word size) multiplication and our approach needs fewer number of multiplications, we believe that more improvement will be achieved with a RISC workstation.

We implemented our algorithms and the others over the finite field F_{p^m}, where $p = 2^{31} - 1$ and irreducible polynomial of $x^7 - 3$, in C language only, then examined the time taken for computation. We assumed that all scalar multiplication algorithms use type II expansion in [13], so the average Hamming weight is 11/32.

5.2 Result and Analysis

Our implementation result is summarized in Table 3.

Our finite field multiplication algorithm minimizes the number of coefficient multiplications and improve performance by about 27%. Also, we proposed specialized efficient finite field squaring algorithm different from usual multiplication. Our squaring algorithm needs fewer number of coefficient multiplications in particular cases and results in about 40.1% of improvement than general multiplications.

5.3 Scalar Multiplication Speed

In an elliptic curve cryptosystem, the number of finite field operations required for point addition and doubling varies according to the coordinate systems [11]. Among various coordinate systems, we assume that the *Modified Jacobian* coordinate is used [7]. The numbers of finite field operations for one elliptic curve addition and doubling are specified in Table 4.

Based on the above specification, we can estimate the time taken for an elliptic curve addition, a doubling, and a scalar multiplication as following Table 5. From this result, we can see the overall performance of elliptic curve cryptosystem can be improved by about two-fold.

6 Conclusion

Through the use of the elliptic curve cryptosystems, the area of information security has tremendous potential to assist commercial success, provide peace of mind, and improve the quality of life. This paper scratches the surface of this exciting field by outlining how security services are provided by elliptic curve systems.

We provide the efficient ways of using the elliptic curve cryptosystems. These can be used in mobile communucations, tiny computing systems, or ubiquitous computing system due to the efficiency of key management and computational overhead.

Acknowledgement

This work was supported by the Ministry of Science and Technology(MOST)/Korea Sceince and Engineering Foundation(KOSEF) through the Advanced Information Technology Research Center(AITrc).

References

[1] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone, "An implementation for a fast public-key cryptosystem," *Journal of Cryptology*, vol. 3, pp. 63-79, 1991.

Table 4: Number of finite field multiplication and squaring per EC addition and doubling

	EC addition	EC doubling	Scalar Mul.	
			Kobayashi [14]	**Proposed**
FF mul.	13	4	1095	679
FF sqr.	6	4	570	378

Table 5: Estimated time for the elliptic curve operations

		KOA+Kob.	P.M+Kob.	**P.M+P.F**
EC addition (μsec)	Pentium-Pro	124.45	103.22	←
	UltraSparc-2	197.22	135.06	←
EC double (μsec)	Pentium-Pro	52.40	42.40	←
	UltraSparc-2	83.04	54.76	←
EC scalar mul. (msec)	Pentium-Pro	10.91	9.01	**5.71**
	UltraSparc-2	17.28	11.77	**7.45**

Kob. : Kobayashi algorithm [14]
P.M : Proposed Multiplication and squaring algorithm
P.F : Proposed Frobenius map

[2] G. B. Agnew, R. C. Mullin, and S. A. Vanstone, "An implementation of elliptic curve cryptosystems over $\mathbb{F}_{2^{155}}$," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 804-813, 1993.

[3] G. B. Agnew, T. Beth, R. C. Mullin, and S. A. Vanstone", "Arithmetic Operations in GF(2^m)," *Journal of Cryptology*, vol. 6, pp. 3-13, 1993.

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

[5] E. F. Brickell, D. M. Gordon, K. S. McCurley, and D. B. Wilson, "Fast exponentiation with precomputation," *Advances in Cryptology - Eurocrypt'92*, LNCS 658, pp. 200-207, Springer-Verlag, 1993.

[6] J. H. Cheon, S.Park, S. Park, and D. Kim, "Two efficient algorithms for arithmetic of elliptic curves using Frobenius map," *Public Key Cryptography: Proceedings of the First international workshop, PKC'98*, LNCS 1431, pp. 195-202, Springer-Verlag, 1998.

[7] D. V. Chudnovsky and G. V. Chudnovsky, "Sequences of numbers generated by addition in formal groups and new primality and factorization tests," *Advances in Applied Mathmetics*, vol. 7, pp. 385-434, 1986.

[8] D. M. Gordon, "A survey of fast exponentiation methods," *Journal of Algorithms*, vol. 27, pp. 129-146, 1998.

[9] J. Guajardo and C. Paar, "Efficient algorithms for elliptic curve cryptosystems," *Advances in Cryptology - Crypto'97*, LNCS 1233, pp. 342-356, Springer-Verlag, 1997.

[10] G. Harper, A. Menezes, and S. Vanstone, "Public-key cryptosystems with very small key kengths", Advances in Cryptology Eurocrypt'92, pp. 163-173, Springer-Verlag, 1992.

[11] T. Izu, J. Kogure, M. Noro, and K. Yokoyama, "Efficient implementation of Schoof's algorithm," *Asiacrypt'98*, LNCS 1514, pp. 66-79, Springer-Verlag, 1998.

[12] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Soviet Physics-Doklady (English translation)*, vol. 7, no. 7, pp. 595-596, 1963.

[13] T. Kobayashi, H. Morita, K. Kobayashi, and F. Hoshino, "Fast elliptic curve algorithm combining frobenius map and table reference to adapt to higher characteristic," *Advances in Cryptology - Eurocrypt'99*, LNCS 1592, pp. 176-189, Springer-Verlag, 1999.

[14] T. Kobayashi, H. Morita, K. Kobayashi, and F. Hoshino, "Fast elliptic curve algorithm combining frobenius map and table reference to adapt to higher characteristic," *Advances in Cryptology - Proceeding of Eurocrypt'99*, LNCS 1592, pp. 176-189, Springer-Verlag, 1999.

[15] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203-209, 1987.

[16] N. Koblitz, "CM-curves with good cryptographic properties," *Advances in Cryptology - Crypto'91*, LNCS 576, pp. 279-287, Springer-Verlag, 1991.

[17] N. Koblitz, "An elliptic curve implementation of the finite field digital signature algorithm," *Advances in Cryptology - Crypto'98*, LNCS 3796, pp. 327-337, Springer-Verlag, 1998.

[18] E. J. Lee, D. S. Kim, and P. J. Lee, "Speed-up of F_{p^m} arithmetic for elliptic curve cryptosystems," *ICISC'98*, pp. 81-91, 1998.

[19] A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

[20] V. S. Miller, "Use of elliptic curve in cryptography," *Advances in Cryptology - Crypto'85*, LNCS 218, pp. 417-426, Springer-Verlag, 1985.

[21] V. Muller, "Fast multiplication on elliptic curves over small fields of characteristic two," *Journal of Cryptology*, vol. 11, pp. 219-234, 1998.

[22] E. D. Win, A. Bosselaers, S. Vandenberghe, P. D. Gersem, and J. Vandewalle, "A fast software implementation for arithmetic operations in GF(2^n)," *Asiacrypt'96*, LNCS 1163, pp. 65-76, Springer-Verlag, 1996.

Appendix

Proof of Theorem 4

Let the result of multiplication of two polynomials ,A and B, of degree $m-2$ be $C'(x) = \sum_{i=0}^{2m-4} c'_i x^i$. If a term of degree $m-1$ is added, the multiplication $C(x) = \sum_{i=0}^{2m-2} c_i x^i$ can be obtained as follows.

$$\begin{cases} c_i = c'_i, & i = 0, 1, \cdots, m-2 \\ c_i = c'_i + (a_{m-1} + a_{i-m+1})(b_{m-1} + b_{i-m+1}) \\ \quad -a_{m-1}b_{m-1} - a_{i-m+1}b_{i-m+1} & i = m-1, \cdots, 2m-3 \\ c_i = a_{m-1}b_{m-1}, & i = 2m-2 \end{cases} \tag{6}$$

In the second case, $a_{i-m+1}b_{i-m+1}$ $(i = m-1, \cdots, 2m-3)$ can be obtained during computing $C'(x)$ and additional multiplication caused by the term of degree $m-1$ is only $(a_{m-1} + a_{i-m+1})(b_{m-1} + b_{i-m+1})$.

Therefore, the multiplication of two polynomial of degree $m-1$ can be calculated by KOA from the constant term through the term of degree $2^t - 1, (t = 1, 2, \cdots)$ which is less than $m-1$ but maximal, and by repeated application of above equation to the term of degree $m-1$.

□

Example of Theorem 4

Following is the example of Theorem 4 for the polynomial of degree four.

$$\begin{aligned} C(x) &= A(x) \times B(x) \\ &= (a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0)(b_4x^4 + \\ &\quad b_3x^3 + b_2x^2 + b_1x + b_0) \\ &= \sum_{i=0}^{8} c_i x^i \end{aligned} \tag{7}$$

The intermediate terms as Equation (8) can be obtained by KOA through the term of degree three.

$$\begin{aligned} d_0 &= a_0b_0 \\ d_1 &= (a_1 + a_0)(b_1 + b_0) \\ d_2 &= a_1b_1 \\ d_3 &= (a_2 + a_0)(b_2 + b_0) \\ d_4 &= (a_3 + a_2 + a_1 + a_0)(b_3 + b_2 + b_1 + b_0) \\ d_5 &= (a_3 + a_1)(b_3 + b_1) \\ d_6 &= a_2b_2 \\ d_7 &= (a_3 + a_2)(b_3 + b_2) \\ d_8 &= a_3b_3 \end{aligned} \tag{8}$$

Using Equation (8), the multiplication of polynomials of degree 3 can be computed by Equation (9).

$$\begin{aligned} C'(x) &= \left(\sum_{i=0}^{3} a_i x^i\right)\left(\sum_{i=0}^{3} b_i x^i\right) = \sum_{i=0}^{6} c'_i x^i \\ c'_0 &= d_0 \\ c'_1 &= -d_0 + d_1 - d_2 \\ c'_2 &= -d_0 + d_2 + d_3 - d_6 \\ c'_3 &= d_0 - d_1 + d_2 - d_3 + d_4 - d_5 + d_6 - \\ &\quad d_7 + d_8 \\ c'_4 &= -d_0 - d_2 + d_5 + d_6 - d_8 \\ c'_5 &= d_0 - d_1 - d_6 + d_7 - d_8 \\ c'_6 &= d_0 - d_3 + d_8 \end{aligned} \tag{9}$$

The intermediate terms for the term of degree four are in Equation (10).

$$\begin{aligned} d_9 &= (a_4 + a_0)(b_4 + b_0) \\ d_{10} &= (a_4 + a_1)(b_4 + b_1) \\ d_{11} &= (a_4 + a_2)(b_4 + b_2) \\ d_{12} &= (a_4 + a_3)(b_4 + b_3) \\ d_{13} &= a_4b_4 \end{aligned} \tag{10}$$

Finally, the coefficients of $C(x)$ can be calculated as Equation (11), using Equations (8), (9), and (10).

$$\begin{aligned} c_0 &= c'_0 \\ c_1 &= c'_1 \\ c_2 &= c'_2 \\ c_3 &= c'_3 \\ c_4 &= c'_4 + d_9 - d_0 - d_{13} \\ c_5 &= c'_5 + d_{10} - d_2 - d_{13} \\ c_6 &= c'_6 + d_{11} - d_6 - d_{13} \\ c_7 &= c'_7 + d_{12} - d_8 - d_{13} \\ c_8 &= d_{13} \end{aligned} \tag{11}$$

By previous KOA, 17 coefficient multiplications are needed to multiply the polynomials of degree four, but we need only 14.

Jaewon Lee Received the B.S., M.S., and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 1997, 1999, and 2005, respectively. He is currently a senior engineer of Samsung Electronics Co., LTD. His research interests include Digital Rights Management and Tamper-Resistant Software.

Heeyoul Kim Received the B.E. degree in computer science from Korea Advance Institute of Science and Technology (KAIST), South Korea, in 2000, the M.S. degree in computer science from KAIST, in 2002. He is currently working toward the Ph.D. degree at the Division of Computer Science, KAIST.

Younho Lee Received the B.E. degree in computer science from Korea Advance Institute of Science and Technology (KAIST), South Korea, in 2000, the M.S. degree in computer science from KAIST, in 2002. He is currently working toward the Ph.D. degree at the Division of Computer Science, KAIST.

Seong-Min Hong Received the B.E. degree in computer science from KAIST, South Korea, in 1994. He also received the M.S. degree and Ph.D degree in computer engineering from KAIST in 1996 and 2000, respectively. He is currently an research professor in the division of Computer Science at KAIST.

Hyunsoo Yoon Received the B.E. degree in electronics engineering from SNU, South Korea, in 1979, the M.S. degree in computer science from KAIST, in 1981, and the Ph.D. degree in computer and information science from the Ohio State University, Columbus, Ohio, in 1988. From 1988 to 1989, with the AT & T Bell Labs. as a Member of Technical Staff. Since 1989 he has been a faculty member of Division of Computer Science at KAIST.