情報保護學會論文誌
第15券 第3號, 2005. 6

최적확장체 위에서 정의되는 타원곡선에서의 고속 상수배 알고리즘*

정 병 천,†‡ 이 수 진, 홍 성 민, 윤 현 수
한국과학기술원 전자전산학과

Fast Scalar Multiplication Algorithm on Elliptic Curve over Optimal Extension Fields*

Byungchun Chung,†‡ Soojin Lee, Seong-Min Hong, Hyunsoo Yoon
Dept. of Electrical Engineering & Computer Science, KAIST

요 약

EC-DSA나 EC-ElGamal과 같은 타원곡선 암호시스템의 성능 향상을 위해서는 타원곡선 상수배 연산을 빠르게 하는 것이 필수적이다. 타원곡선 특유의 Frobenius 사상을 이용한 base-ϕ 전개 방식은 Koblitz에 의해 처음 제안되었으며, Kobayashi 등은 최적확장체 위에서 정의되는 타원곡선에 적용할 수 있도록 base-ϕ 전개 방식을 개선하였다. 그러나 Kobayashi 등의 방법은 여전히 개선의 여지가 남아있다. 본 논문에서는 최적확장체에서 정의되는 타원곡선상에서 효율적인 상수배 연산 알고리즘을 제안한다. 제안한 상수배 알고리즘은 Frobenius 사상을 이용하여 상수 값을 Horner의 방법으로 base-ϕ 전개하고, 이 전개된 수식을 최적화된 일괄처리 기법을 적용하여 연산한다. 제안한 알고리즘을 적용할 경우, Kobayashi 등이 제안한 상수배 알고리즘보다 20%~40% 정도의 속도 개선이 있으며, 기존의 이진 방법에 비해 3배 이상 빠른 성능을 보인다.

ABSTRACT

Speeding up scalar multiplication of an elliptic curve point has been a prime approach to efficient implementation of elliptic curve schemes such as EC-DSA and EC-ElGamal. Koblitz introduced a base-ϕ expansion method using the Frobenius map. Kobayashi et al. extended the base-ϕ scalar multiplication method to suit Optimal Extension Fields(OEF) by introducing the table reference method. In this paper we propose an efficient scalar multiplication algorithm on elliptic curve over OEF. The proposed base-ϕ scalar multiplication method uses an optimized batch technique after rearranging the computation sequence of base-ϕ expansion usually called Horner's rule. The simulation results show that the new method accelerates the scalar multiplication about 20%~40% over the Kobayashi et al. method and is about three times as fast as some conventional scalar multiplication methods.

Keywords : *Elliptic curve, Scalar multiplication, Frobenius map, Batch technique, OEF*

접수일 : 2005년 2월 17일 ; 채택일 : 2005년 5월 11일
* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았고 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성 · 지원사업의 연구결과로 수행되었음.
† 주저자, ‡ 교신저자 : bcchung@camars.kaist.ac.kr

Ⅰ. 서 론

타원곡선 암호시스템(Elliptic Curve Cryptosystem)은 1985년에 Miller [1] 와 Koblitz [2] 에 의해 독자적으로 제안되었다. 그 이후 타원곡선 암호시스템은 안전하고 효율적 구현이 가능한 암호시스템으로 주목을 받아왔다. 타원곡선 암호시스템의 안전성은 타원곡선 이산대수 문제(ECDLP; Elliptic Curve Discrete Logarithm Problem)에 기반한다. 유한체 상에서 정의되는 타원곡선에서의 이산대수 문제는 일반적으로 타원곡선과 동일한 크기를 갖는 유한체에서의 이산대수 문제보다 더 어렵다고 알려져 있다 [3]. 따라서 타원곡선 암호시스템은 동등한 안전도를 갖는 유한체에서의 암호시스템보다 작은 크기의 키를 요구하기 때문에 고속의 소형 암호시스템 구현이 용이하다. 보통 160 비트 크기의 타원곡선과 1024 비트 크기의 유한체가 동등한 안전성을 가진 것으로 비교된다 [4]. 이러한 장점은 계산능력이나 대역폭, 메모리가 제한된 스마트카드나 무선 단말기 등의 응용에 적합함을 보여준다.

타원곡선 암호시스템에서는 타원곡선 위의 한 점 P에 대한 상수배(scalar multiplication) kP 연산이 주된 연산이다. 상수배 연산은 반복적인 덧셈과 두배 연산으로 이루어지기 때문에, 타원곡선 암호시스템의 성능은 덧셈과 두배 연산의 반복 횟수에 비례하게 된다. 반복 횟수를 줄이기 위한 방법들은 기존 모듈러 멱승연산에서 곱셈 횟수를 줄이려는 방법과 유사하다 [5]. 따라서 모듈러 멱승연산에서의 이러한 연구를 타원곡선 상수배 연산에 적용할 수 있다. 그러나 타원곡선에서는 타원곡선 덧셈과 뺄셈의 계산량 차가 거의 없기 때문에, 모듈러 멱승에서는 적용이 쉽지 않던 부호화된 이진 방법(signed binary method)이나 덧셈-뺄셈 사슬(addition-subtrac- tion chain), 부호화된 윈도우 방법(signed win- dow method) 등의 방법을 효과적으로 적용할 수 있다.

한편, 타원곡선에서는 타원곡선 특유의 방법인 Frobenius 사상(Frobenius map)을 이용한 base-ϕ 전개방법을 사용하여 덧셈/뺄셈 횟수를 획기적으로 줄일 수 있다. 이는 기존의 어떤 방법보다 효율적인 방법으로 알려져 있다 [5].

Frobenius 사상을 이용한 base-ϕ 전개 방식은 Koblitz에 의해 처음 제안되었다 [6]. Koblitz의 방법은 $GF(2)$, $GF(4)$, $GF(8)$, $GF(16)$ 유한체에서 정의되는 타원곡선에만 적용 가능한 방법이다. Muller [7] 와 Cheon 등 [8] 은 base-ϕ 전개 방식을 $GF(2^r)$, $r \leq 6$ 에서 정의되는 타원곡선으로 확장시켰다. 한편, Koblitz는 특성계수(characteris- tic) 2가 아닌 $GF(3)$과 $GF(7)$의 홀수 특성계수를 갖는 유한체에서 정의되는 타원곡선에서의 base-ϕ 전개 방식을 제안하였으며 [9], Smart는 128이하의 작은 홀수 특성계수를 갖는 유한체 $GF(q)$에서 정의되는 타원곡선에서의 일반화된 base-ϕ 전개 방식을 제시하였다 [10].

1998년, Bailey와 Paar는 32 또는 64 비트 고성능 마이크로프로세서에서 소프트웨어 구현에 적합한 $GF(p^m)$ 형태의 유한체인 최적확장체(OEF; Optimal Extension Field)를 제안하였다 [11]. OEF에서는 프로세서를 최대한 활용할 수 있도록 특성계수 p를 프로세서 워드 크기에 적합한 2^{32} 또는 2^{64}에 가까운 큰 소수를 사용한다. 따라서 Muller나 Smart의 base-ϕ 전개 방식을 OEF 위에서의 타원곡선에 그대로 적용하기 위해서는 2^{32} 또는 2^{64} 크기의 방대한 테이블이 필요하게 된다. Kobayashi 등은 Muller나 Smart의 방법을 작은 크기의 테이블을 사용하여 OEF에 적용할 수 있도록 개선하였으나 [12], 여전히 개선의 여지가 남아있다.

본 논문에서는 OEF 위에서의 타원곡선에서 base-ϕ 전개 방식을 적용한 효율적인 상수배 연산 알고리즘을 제안한다. 본 논문에서 제안하는 base-ϕ 전개 방식은 Horner의 법칙(Horner's rule) [13] 을 적용한 방법으로 Kobayashi 등의 방법과는 다른 base-ϕ 전개식을 사용한다. 이와 같은 전개 방식을 적용하게 되면 큰 상수배 연산 문제가 여러 개의 작은 상수배 연산 문제로 바뀌게 된다. 이 때, 작은 상수는 유한체의 특성계수 정도의 크기가 된다. 이렇게 해서 생긴 여러 개의 작은 상수배 연산은 일괄처리 기법 [14] 을 이용하여 계산의 효율을 더욱 높일 수 있다. 일괄처리 기법은 여러 개의 상수배 연산을 동시에 계산하는 과정에서 중복되는 연산 부분을 제거함으로써, 각각 독립적으로 처리하는 것보다 계산의 효율을 높일 수 있는 방법이다. 기존의 이러한 일괄처리 기법은 인증서버와 같이 동시에 많은 서명을 일괄적으로 생성하거나 검증하는 과정에서 발생하는 상수배 연산에 대하여 적용할 수 있는 방법이었다. 그러나 본 논문에서는 한 서명 내에서 일괄처리 기법을 적용하며, 또한 일괄처리 기법을 최적화

함으로써 계산의 효율을 극대화한다. 본 논문에서 제안하는 방법은 Kobayashi 등의 방법보다 20%~40% 정도의 속도 향상이 있고, 또한 기존의 이진 방법에 비해서는 3배 이상 빠른 성능을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 OEF 위에서의 타원곡선과 이 타원곡선에서 정의되는 Frobenius 사상, 그리고 Kobayashi 등의 base-ϕ 전개 방식을 이용한 상수배 연산 알고리즘을 살펴본다. 3장에서는 일괄처리 기법에 대해 살펴보고, 4장에서는 Horner 방법을 이용한 base-ϕ 전개에 최적화된 일괄처리 기법을 적용한 효율적인 상수배 연산 알고리즘을 제안한다. 5장에서는 4장에서 제안한 상수배 연산 알고리즘을 분석하고, 다른 상수배 연산 알고리즘과 성능 비교를 한다. 마지막으로 6장에서 결론을 맺는다.

II. 최적확장체 위에서의 타원곡선

1. 타원곡선의 정의

$GF(p)$, $p>3$ 상에서 정의되는 정규 타원곡선(non-supersingular elliptic curve) E는 다음과 같은 형태의 방정식으로 표현된다.

$$E: y^2 = x^3 + ax + b, \tag{1}$$

여기서 $a, b \in GF(p)$ 이고, $4a^3 + 27b^2 \neq 0$을 만족한다. $GF(p)$상에서 타원곡선 E를 정의하는 이유는 상수배 연산에서 Frobenius 사상을 이용한 base-ϕ 전개 방식을 적용하기 위함이다.

$E(GF(p^m))$는 $GF(p)$의 확장체 $GF(p^m)$ 상의 타원곡선 E 위의 점들의 집합으로 정의된다. $E(GF(p^m))$는 O(point at infinity)와 식 (1)을 만족하는 점 $(x, y) \in GF(p^m) \times GF(p^m)$ 들로 구성되며, O을 항등원으로 하는 덧셈에 대한 가환군(abelian group)을 이룬다.

2. Frobenius 사상

$P=(x, y)$를 타원곡선 E 위의 $GF(p^m)$-점이라 하면, Frobenius 사상 ϕ는 다음과 같이 정의된다.

$$\phi : (x, y) \rightarrow (x^p, y^p)$$

이 Frobenius 사상 ϕ는 $E(GF(p^m))$의 자기준동형(endomorphism) 사상이 되며, 다음의 방정식을 만족한다.

$$\phi^2 - t\phi + p = 0, \quad -2\sqrt{p} \le t \le 2\sqrt{p}$$

Frobenius 사상을 구현 관점에서 살펴보면, 다항식 기저 표현의 원소일 경우 Frobenius 사상 ϕ를 계산하는 데 $2(m-1)$번의 $GF(p)$ 곱셈이 필요하다[12]. 따라서 Frobenius 사상 연산은 m^2번의 $GF(p)$ 곱셈을 필요로 하는 확장체 곱셈보다 빠르게 구현될 수 있는 연산이다.

3. Frobenius 사상을 이용한 base-ϕ 전개

타원곡선에서의 상수배 연산은 상수 k와 타원곡선 위의 한 점 P가 주어졌을 때, kP를 구하는 연산을 말한다. 여기서 상수 k를 Frobenius 사상 ϕ의 다항식으로 전개할 수 있다[12].

$$k = \sum_{i=0}^{l} u_i \phi^i, \text{ where } -\frac{p}{2} \le u_i \le \frac{p}{2}. \tag{2}$$

이 식에서 l은 대략 $2m+3$ 정도 된다.

길이가 $2m+4$인 k에 대한 ϕ의 다항식 표현은 다음 정리 1에 의하여 길이가 m인 ϕ의 다항식 표현으로 축소될 수 있다.

정리 1 $E(GF(p^m))$에서 Frobenius 사상 ϕ를 m번 반복하면 항등사상(identity map)이 된다. 즉,

$$\phi^m = 1 \text{ on } E(GF(p^m)).$$

이는 $x^{p^m} = x$, $\forall x \in GF(p^m)$이기 때문이다. 따라서 ϕ의 다항식 표현은 다음과 같이 축소된다.

$$\sum_{i=0}^{2m+3} u_i \phi^i = \sum_{i=0}^{m-1} (u_i + u_{i+m} + u_{i+2m}) \phi^i$$

$$= \sum_{i=0}^{m-1} d_i \phi^i, \text{ where } -\frac{3p}{2} \le d_i \le \frac{3p}{2}. \tag{3}$$

Kobayashi 등은 이를 Type I 전개라 하였다.

다음의 사실을 이용하여 Type I 전개를 더 최적화 할 수 있다. 이는 d_i의 비트 표현에서 '1'의 수를 줄이기 위한 방법이다.

$$\sum_{i=0}^{m-1}\phi^i = 0 \tag{4}$$

이는 정리 1과 $\phi \neq 1$라는 사실로부터 유도된 것이다. 식 (4)를 이용하면 동일한 값을 d_i에 더하거나 빼도 결과는 변화가 없다. 따라서 식 (3)의 표현은 다음과 같이 바뀐다.

$$\sum_{i=0}^{m-1} d_i\phi^i = \sum_{i=0}^{m-1}(d_i - z)\phi^i, \text{ where } -\frac{p}{2} < z < \frac{p}{2}$$
$$= \sum_{i=0}^{m-1} c_i\phi^i, \text{ where } -2p < c_i < 2p. \tag{5}$$

여기서 z는 $\sum w_H(c_i)$[2]를 최소화 하는 정수이다. Kobayashi 등은 이를 Type II 전개라 하였다.

4. Base-ϕ 상수배 연산 알고리즘

식 (5)를 이용하면, 상수배 연산 kP를 다음과 같이 표현할 수 있다.

$$kP = \sum_{i=0}^{m-1} c_i\phi^i P, \text{ where } -2p < c_i < 2p$$
$$= c_0P + c_1\phi P + c_2\phi^2 P + \cdots + c_{m-1}\phi^{m-1}P \tag{6}$$
$$= c_0P + \phi(c_1P + \phi(c_2P + \phi(\cdots + \phi(c_{m-1}P)\cdots))) \tag{7}$$

전통적인 base-ϕ 상수배 연산 방식들은 $GF(2^r)$, $r \leq 6$ 또는 $GF(q)$, $q < 128$과 같은 작은 유한체에서 정의된 타원곡선에서만 적용되었다[6-8] [10]. 이들 방법은 식 (7)에서 iP, $1 \leq i < 2p$를 미리 계산하여 테이블에 저장해 놓고, 식을 연산하는 과정에서 c_iP는 테이블 참조를 통하여 처리하는 방식을 취하였기 때문이다. 따라서 2^{32} 또는 2^{64}에 가까운 큰 소수를 특성계수로 갖는 OEF에서는 이러한 테이블 이용 방식을 적용할 수 없다. Kobayashi 등은 이러한 base-ϕ 상수배 연산 방식을 OEF에 적용할 수 있도록 이전의 방법을 개선하였다[12].

Kobayashi 등의 방법을 간단한 예를 들어 설명한다. 어떤 상수배 kP가 식 (6)의 형태로 전개되었다고 하자.

$$kP = 15P + 13\phi^2P + 2\phi^3P$$
$$= (1111)_2P + (1101)_2\phi^2P + (0010)_2\phi^3P$$

우선, ϕP, $\phi^2 P$, $\phi^3 P$를 계산하여 테이블에 저장한다. X는 O로 초기화한다. 그리고 base-ϕ 전개식에서 계수의 이진표현 중에서 첫 번째 비트(MSB)가 '1'인 항들을 찾아 테이블을 참조하여 덧셈하고 두배 한다. 따라서 $X \leftarrow X + P + \phi^2 P$를 계산하고, $X \leftarrow 2X$를 수행한다. 마찬가지로 두 번째 비트가 '1'인 항들을 찾아 덧셈하고 두배 한다. 그리고 마지막 비트(LSB)에서는 덧셈만 수행한다.

이 알고리즘은 Type I 전개의 경우 평균적으로 $m(\lceil \log_2 p \rceil + 1)/2$번의 덧셈과 $\lceil \log_2 p \rceil$번의 두배 연산을 필요로 한다. Type II 전개의 경우는 $m \cdot \overline{w_H}$번의 덧셈과 $\lceil \log_2 p \rceil$번의 두배 연산을 필요로 한다. 여기서 $\overline{w_H}$는 c_i들의 평균 해밍중을 의미한다.

III. 일괄처리 기법(Batch Technique)

일괄처리 기법은 여러 개의 상수배 연산을 동시에 처리하는 과정에서 중복되는 연산 부분을 제거함으로써, 각각 독립적으로 처리하는 것보다 계산의 효율을 높일 수 있는 방법이다. 이는 인증 서버와 같이 동시에 많은 서명을 일괄적으로 생성하거나 검증하는 과정에서 발생하는 상수배 연산에 대해서 적용할 수 있다. 본 장에서는 일괄처리 기법의 기본 개념을 기술한다[14].

우선, 일괄처리 개념 설명에 필요한 간단한 병렬 이진 상수배 연산 알고리즘을 소개한다. n개의 t비트 정수 k_j $(0 \leq j \leq n-1)$와 타원곡선 위의 한 점 P가 주어졌을 때, 상수배 k_jP $(0 \leq j \leq n-1)$을 구하는 알고리즘은 알고리즘 1과 같다.

이 알고리즘은 $t-1$번의 타원곡선 두배 연산과 $\sum(w_H(k_j)-1)$번의 타원곡선 덧셈 연산을 필요로 한다.

2) $w_H(c_i)$는 c_i의 해밍중(Hamming weight)을 의미한다.

알고리즘 1. Parallel Binary Scalar Multiplication (PBSM)

Input : $k_j=\sum_{i=0}^{t-1}k_{ji}2^i,\ 0\le j\le n-1$ and a variable point P

Output : $Q_j=k_jP,\ 0\le j\le n-1$

1 Initialize $Q_j\leftarrow O,\ 0\le j\le n-1$

2 **for** $i=0$ **to** $t-1$ **do**

3 $\quad P_i\leftarrow 2^iP$

4 **for** $j=0$ **to** $n-1$ **do**

5 $\quad$ **if** $k_{ji}=1$ **then** $Q_j\leftarrow Q_j+P_i$

따라서 이 알고리즘의 연산 속도를 향상시키기 위해서는 $\sum w_H(k_j)$를 줄이는 것이 중요하다.

다음으로, 내용 전개에 있어 필요한 집합 연산자를 살펴본다. 집합 연산자 $\cap, \cup, \Delta$은 각각 교집합, 합집합, 차집합을 의미하며, 다음과 같이 구현될 수 있다.

$$a\cap b := a\&b,\ a\cup b := a|b,\ a\Delta b := a\hat{}(a\&b),$$

여기서 &, |, ^는 비트단위(bitwise) 연산자 AND, OR, XOR를 나타낸다.

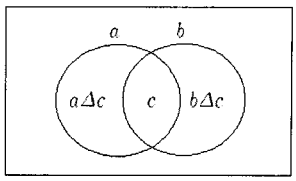
다음 보조정리 2는 일괄처리 기법의 기본 아이디어를 제공한다.

보조정리 2 임의로 선택된 $a=\sum_{i=0}^{t-1}a_i2^i,\ b=\sum_{i=0}^{t-1}b_i2^i$가 주어졌을 때, a와 b의 교집합을 $c=a\cap b$라 하면 다음의 부등식을 만족한다.

$$w_H(a\Delta c)+w_H(b\Delta c)+w_H(c)\le w_H(a)+w_H(b)$$

증명. t비트 a와 b가 임의로 선택되었으므로, 교집합 $c=a\cap b=\sum_{i=0}^{t-1}a_ib_i2^i$의 평균해밍중 $w_H(c)$는 $\frac{t}{4}$ 이다.

따라서 $w_H(a)+w_H(b)-(w_H(a\Delta c)+w_H(b\Delta c)+w_H(c))=w_H(c)\ge 0$, 단 등호는 $c=\phi$일 때만 성립한다.



실제로 알고리즘 1을 이용하여 aP와 bP를 구하는 데 필요한 덧셈의 횟수를 살펴보면, aP와 bP를 독립적으로 계산할 경우는 $t-2$번의 덧셈이 필요하고, 교집합 c를 이용하여 $aP=(a\Delta c)P+cP$, $bP=(b\Delta c)P+cP$를 계산할 경우는 $3t/4-1$번의 덧셈이 필요하다. 따라서 교집합을 이용하면 약 25% 정도의 덧셈을 줄일 수 있다. 보조정리 2는 n개의 경우로도 확장이 가능하다. 단, n개로 확장된 경우는 상수배 연산에 필요한 덧셈의 횟수가 교집합을 어떻게 생성하고 결합하느냐에 따라 차이가 발생하게 된다. 따라서 효율적인 교집합 생성 및 결합 알고리즘이 필요하다.

Ⅳ. 제안하는 상수배 연산 알고리즘

본 장에서는 OEF 위에서의 타원곡선에서 base-ϕ 전개 방식에 일괄처리 기법을 적용한 효율적인 상수배 연산 알고리즘을 제안한다. 제안하는 방식은 식 (7)처럼 Horner의 법칙(Horner's rule)[13]을 적용한 방법으로 Kobayashi 등의 방법과는 다른 base-ϕ 전개식을 사용한다. 이와 같이 식을 전개하면 큰 상수배 문제가 여러 개의 작은 상수배 문제로 바뀌게 된다. 즉 kP 문제가 여러 개의 작은 c_iP 문제로 바뀌는 것이다. 이렇게 해서 생긴 여러 개의 작은 상수배 문제에 일괄처리 기법을 적용하여 계산의 효율성을 높일 수 있다. 본 논문에서는 이 일괄처리 기법을 최적화하여 상수배 연산 알고리즘에 적용함으로써 계산의 효율성을 극대화한다.

1. 최적화된 일괄처리 알고리즘

본 절에서는 $n(>1)$개의 상수배 연산이 동시에 주어졌을 때, 이를 효율적으로 처리할 수 있는 최적화된 일괄처리 알고리즘을 제안한다. 이 알고리즘은 크게 분할 단계(partitioning stage)와 결합 단계(combining stage)로 구성된다. 분할 단계에서는 주어진 상수 c_i들을 서로소(disjoint)인 부분집합으

로 나누는 기능을 하고, 결합 단계에서는 분할 단계에서 만들어진 부분집합들을 결합하여 원래의 각 집합들을 복원하는 기능을 한다.

1.1 분할 단계 (Partitioning Stage)

n개의 상수 c_j가 주어지면 공집합이 아닌 서로소인 부분집합은 최대 2^n-1개 까지 가능하다. 여기서 공집합은 0을 의미한다.

인덱스 i를 $(e_{n-1}\cdots e_1e_0)_2$와 같이 이진수로 표현하면, 서로소인 부분집합 s_i, $1\le i\le 2^n-1$는 다음의 공식에 의해 쉽게 구할 수 있다.

$$s_i=\bigcap_{e_j=1} c_j \,\Delta\, \bigcup_{e_j=0} c_j$$

예를 들어 $n=3$인 경우, 서로소인 부분집합을 벤 다이어그램으로 표현하면 그림 1과 같다. s_{101}은 $(c_2\cap c_0)\,\Delta\, c_1$과 같이 구할 수 있다.

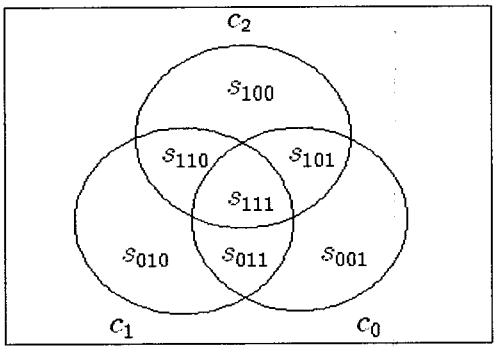


그림 1. $n=3$일 때의 분할 벤 다이어그램

1.2 결합 단계 (Combining Stage)

분할 단계에서 서로소인 부분집합으로 나누어진 조각들은 다시 결합과정을 통하여 원래의 집합으로 복원되어야 한다.

$n=3$인 경우를 예로 들어 설명한다. [14]에서는 다음과 같은 단순한 방법으로 원래의 집합을 복원하였다.

$$c_0=s_{001}\cup s_{011}\cup s_{101}\cup s_{111}$$
$$c_1=s_{010}\cup s_{011}\cup s_{110}\cup s_{111}$$
$$c_2=s_{100}\cup s_{101}\cup s_{110}\cup s_{111}$$

그러나 $s_{110}\cup s_{111}$을 미리 계산해 둔다면 c_1과 c_2에서 1번의 $\cup$을 줄일 수 있다. 여기서 $\cup$은 실제로 상수배 연산 알고리즘에서 타원곡선 덧셈 연산에 해당한다. 따라서 $\cup$연산을 최소화하는 방법이 필요하다. $n=3$인 경우는 눈으로 쉽게 찾을 수 있지만, n이 커지면 이러한 방법은 쉽지 않으므로 정형화된 방법이 필요하다.

1.3 최적화된 결합 알고리즘

분할 단계에서 생성된 부분집합 s_i에서 인덱스 $i=(e_{n-1}\cdots e_1e_0)_2$들 중에서 $e_0=1$인 s_i들을 찾아 합집합을 구한다. 이것이 c_0가 된다. 그리고 e_0을 제외한 $(e_{n-1}\cdots e_1)_2$의 비트 패턴이 동일한 s_i들의 합집합을 구하여 $s_{(e_{n-1}\cdots e_10)_2}$에 저장하고, 나머지는 삭제한다. 이렇게 해서 한 단계가 끝난다. 다음에 $e_1=1$인 s_i를 찾아 합집합을 구한다. 이는 c_1이 된다. 그리고 $(e_{n-1}\cdots e_2)_2$의 비트 패턴이 동일한 s_i들의 합집합을 구하여 $s_{(e_{n-1}\cdots e_200)_2}$에 저장하고, 나머지는 삭제한다. 계속 이 과정을 반복하면 c_i, $0\le i\le n-1$을 구할 수 있게 된다. 구현 측면에서 볼 때, 연결 리스트(linked list) 자료구조를 사용하면 효율적인 구현이 가능하다. 자세한 알고리즘은 알고리즘 2를 참고하기 바란다.

알고리즘 2. Optimized Batch Scalar Multiplication (OBSM)

Input : $\{c_0, c_1, \cdots, c_{n-1}\}$ and a variable point P
Output : $Q_i=c_iP,\ 0\le i\le n-1$

Partitioning Stage

1 **for** $i=(e_{n-1}\cdots e_1e_0)_2=1$ **to** 2^n-1 **do**
2 $s_i \leftarrow \bigcap_{c_j=1} c_j \,\Delta\, \bigcup_{c_j=0} c_j$

Combining Stage

3 $\{P_i\} \leftarrow PBSM(\{s_i\}, P),\ 1\le i\le 2^n-1$
4 Initialize $Q_i \leftarrow O,\ 0\le i\le n-1$
5 **for** $i=0$ **to** $n-1$ **do**
6 **for** $j=(e_{n-1}\cdots e_1e_0)_2=1$ **to** 2^n-1 **do**
7 **if** $e_i=1$ **then**
8 $Q_i \leftarrow Q_i+P_j$
9 **if** $j\neq(e_{n-1}\cdots e_{i+1}0\cdots0)_2$ **then**
10 $P_{(e_{n-1}\cdots e_{i+1}0\cdots0)_2} \leftarrow P_{(e_{n-1}\cdots e_{i+1}0\cdots0)_2}+P_j$
11 Delete P_j

이해를 돕기 위해 $n=3$인 간단한 예를 들어본다.

$c_0 = s_{001} \cup s_{011} \cup s_{101} \cup s_{111}$
$s_{010} = s_{010} \cup s_{011}$, delete s_{011}
$s_{100} = s_{100} \cup s_{101}$, delete s_{101}
$s_{110} = s_{110} \cup s_{111}$, delete s_{111}
$c_1 = s_{010} \cup s_{110}$
$s_{100} = s_{100} \cup s_{110}$, delete s_{110}
$c_2 = s_{100}$

일반적으로 [14]의 경우는 $n(2^{n-1}-1)$번의 $\cup$이 사용되고, 제안 알고리즘의 경우 $2^{n+1}-2n-2$번의 $\cup$이 사용된다. 증명은 정리 4를 참고하기 바란다. $n=3$인 경우 [14]는 9번의 $\cup$이 사용되고, 제안 알고리즘은 8번의 $\cup$이 사용된다. n이 커짐에 따라 차이는 더 커지게 되는데 이를 표 1에 정리하였다.

2. 상수배 연산 알고리즘

본 절에서는 최적화된 일괄처리 기법을 base-ϕ 상수배 연산 알고리즘에 적용한 새로운 상수배 연산 알고리즘을 제안한다. 일괄처리 기법에서 설명한 방법을 그대로 상수배 연산 알고리즘에 적용할 수 있다. 먼저 식 (7)에 있는 상수배 연산 kP의 base-ϕ 전개를 다시 기술하면 다음과 같다.

$$kP = \sum_{i=0}^{m-1} c_i \phi^i P, \text{ where } -2p < c_i < 2p$$
$$= c_0 P + \phi(c_1 P + \phi(c_2 P + \phi(\cdots + \phi(c_{m-1} P) \cdots))) \quad (8)$$

제안 알고리즘은 이 식에 있는 $c_0P, c_1P, \cdots, c_{m-1}P$를 일괄처리 기법으로 계산하고, 그 결과에 몇 번의

표 1. n에 따른 결합에 필요한 합집합 연산의 수

n	[14]	Proposed
2	2	2
3	9	8
4	28	22
5	75	52
6	186	114
7	441	240

알고리즘 3. Base-ϕ Scalar Multiplication with Batch Technique

Input : $k = \sum_{i=0}^{m-1} c_i \phi^i$ and
a variable point $P \in E(GF(p^n))$
Output : $Q = kP$

1 Choose an optimal value n
2 Initialize $Q_i \leftarrow O$, $0 \le i \le m-1$
3 **for** $i=0$ **to** $\lfloor m/n \rfloor$ **do**
4 $\{Q_{ni}, Q_{ni+1}, \cdots, Q_{ni+n-1}\} \leftarrow$
$OBSM(\{c_{ni}, c_{ni+1}, \cdots, c_{ni+n-1}\}, P)$
5 $Q \leftarrow \phi(Q_{m-1}) + Q_{m-2}$
6 **for** $i=m-3$ **to** 0 by -1 **do**
7 $Q \leftarrow \phi(Q) + Q_i$

Frobenius 사상과 덧셈을 수행하면 상수배 연산을 완결 짓게 된다.

식 (8)을 연산하기 위한 모든 과정을 알고리즘 3에 기술하였다.

V. 알고리즘 분석 및 성능 비교

본 장에서는 제안한 상수배 연산 알고리즘의 성능을 분석한다. 타원곡선에서 상수배 연산은 반복적인 덧셈과 두배 연산으로 이루어지기 때문에, 타원곡선 암호 시스템의 성능은 덧셈과 두배 연산의 반복 횟수에 비례하게 된다. 따라서 본 장에서는 알고리즘의 성능을 덧셈과 두배 연산의 반복 횟수 관점에서 분석하고 다른 알고리즘과 성능을 비교한다.

제안 알고리즘은 상수배 연산 kP를 수행하는 데, 식 (8)에서 작은 상수배 연산 c_iP와 Frobenius 사상, 그리고 나머지 덧셈 연산을 필요로 한다. 제안 알고리즘에 필요한 모든 연산의 수는 정리 3과 같다. 식 (8)을 보면 직관적으로 알 수 있으므로 증명은 생략한다.

정리 3 k는 $m\lceil \log_2 p \rceil$ 비트 정수이고, P는 타원곡선 $E(GF(p^n))$위의 한 점이라 할 때, 상수배 연산 kP는 m개의 $\lceil \log_2 p \rceil + 1$ 비트 상수배 연산과 $m-1$번의 Frobenius 사상, 그리고 $m-1$번의 타원곡선 덧셈 연산을 필요로 한다.

그럼, 이제 정리 3에서 m개의 $\lceil \log_2 p \rceil + 1$ 비트 상수배 연산에 필요한 타원곡선 덧셈과 두배 연산 회수를 알아본다. 제안 알고리즘에서는 이를 위해 최적화된 일괄처리 방식을 사용한다. 이 때 일괄처리 방식에 필요한 덧셈과 두배 연산은 다음 정리를 기반으로 구할 수 있다.

정리 4 $\overline{w_H} > 2^{n-1}$을 만족하는 임의의 n개의 $\lceil \log_2 p \rceil + 1$ 비트 정수 $c_0, c_1, \cdots, c_{n-1}$이 주어졌을 때, 상수배 연산 $c_i P$ $(0 \le i \le n-1)$는 $\lceil \log_2 p \rceil$ 번의 두배 연산과 평균 $\frac{2^n-1}{2^{n-1}}\overline{w_H} + 2^n - 2n - 1$ 번의 덧셈 연산을 필요로 한다. 여기서 $\overline{w_H}$는 c_i들의 평균 해밍중을 나타낸다.

증명. 알고리즘 2에서 타원곡선 연산이 사용되는 부분은 3, 8, 10번째 줄이다.

3번째 줄을 살펴보면, 각 부분집합 s_i의 평균 해밍중은 $\frac{\overline{w_H}}{2^{n-1}}$이 된다. $\overline{w_H} > 2^{n-1}$의 조건을 만족한다면, s_i의 평균 해밍중은 1보다 크게 된다. 그러므로 이에 필요한 덧셈 연산은 $\frac{\overline{w_H}}{2^{n-1}} - 1$ 회가 된다. 따라서 3번째 줄의 $2^n - 1$개의 $s_i P$에는 $(2^n-1)\cdot(\frac{\overline{w_H}}{2^{n-1}} - 1)$ 번의 덧셈과 $\lceil \log_2 p \rceil$ 번의 두배 연산이 필요하다.

8, 10번째 줄에서는 $i=0$일 때 6번째 줄 **for** 루프에서 각각 $2^{n-2}-1$번의 덧셈이 필요하다. 이 둘을 더하면 $2^{n-1}-2$가 된다. 이런 식으로 계속하면 $i=n-1$일 때는 각각 2^0-1번의 덧셈이 필요하다. 따라서 이들을 모두 더하면 다음과 같다.

$$\sum_{i=0}^{n-1} 2^n - 2 = 2^{n+1} - 2n - 2$$

따라서 전 과정에 필요한 연산을 정리하면 $\lceil \log_2 p \rceil$ 번의 두배 연산과 평균 $\frac{2^n-1}{2^{n-1}}\overline{w_H} + 2^n - 2n - 1$ 번의 덧셈 연산이 필요하다.

정리 4에서 $\overline{w_H} > 2^{n-1}$의 조건이 필요한 이유를 살펴본다. c_i가 $\lceil \log_2 p \rceil + 1$ 비트 이므로 알고리즘 2의 분할 단계에서 0이 아닌 서로소인 부분집합의 수는 최대 $\lceil \log_2 p \rceil + 1$개 까지 가능하다. 따라서 필요 이상의 c_i를 사용하면 많은 수의 부분집합이 공집합이 되므로 일괄처리 방식의 장점을 최대로 활용하지 못하게 되는 결과를 초래한다. 따라서 알고리즘 3에서 일괄 처리 방식을 사용할 때 평균 해밍중을 고려하여 최적의 n을 선택해야 한다. 이렇게 n개씩 나누어 처리하는 것이 최적의 방법이 된다. 물론 이 때 한번 계산된 $2^i P$ 테이블은 다시 계산할 필요 없다.

예를 들어 EC-KCDSA 표준에서 권고하고 있는 32비트 프로세서에 적합한 파라미터 ECP31M07K $(p = 2^{31}-1, m=7)$과 64비트 프로세서에 적합한 파라미터 ECP61M05K$(p = 2^{61}-1, m=5)$의 경우를 고려해보자 [15].

우선 $p = 2^{31}-1$, $m=7$의 경우를 살펴보면, 상수 k를 Type I과 Type II로 base-ϕ 전개했을 경우 n을 1부터 7까지 선택했을 때, $c_i P$당 필요한 덧셈 횟수의 분석 값과 시뮬레이션 값은 각각 그림 2와 그림 3과 같다. 시뮬레이션 결과 Type I 전개의 경우는 $n=4$일 때 가장 좋았고, Type II 전개의 경우는 $n=5$일 때 가장 좋은 결과를 보였다. $m=7$이므로 이 경우는 3과 4 두 개로 나누어 처리하는 것이 가장 바람직하다.

그림 2와 그림 3을 분석해 보자. Type I 전개의 경우 n이 4보다 커질 때 분석 값과 시뮬레이션 값이 차이가 나는 것을 볼 수 있는 데, 이는 분석 값에는 부분집합이 공집합인 경우도 덧셈 연산 횟수에 포함되었기 때문이다. Type II 전개의 경우 시뮬레이션 값이 분석 값보다 성능이 떨어지는 것을 볼 수

표 2. $E(GF(2^{31}-1)^7)$ 에서 상수배 연산 알고리즘에 필요한 연산수의 비교

알고리즘	전개Type	두배	덧셈	총합
Binary		$mA-1$ (216)	$mA/2$ (108)	324
Signed Binary		$mA-1$ (216)	$3mA/8$ (81)	297
Kobayashi 등[12]	Type I	A (31)	$m(A+1)/2$ (112)	143
	Type II	A (31)	$m\overline{w_H}$ (77)	108
Proposed	Type I	A (31)	72‡	103
	Type II	A (31)	60‡	91

$m=7$, $A = \lceil \log_2 p \rceil = 31$, $\overline{w_H} = 11$, ‡ $m=7$을 $n=3, 4$ 로 처리

있는 데, 이는 Type II 전개 과정에서 해밍중을 최소화하기 위해 공통되는 비트 패턴을 많이 제거했기 때문이다. 이는 일괄처리 방식에서 공통되는 비트 패턴을 최대한 이용하려는 의도와 개념상 상반된다. 이 모든 방법을 적용하여 상수배 연산을 수행 할 경우, 정리 3에 의해 Type I 전개의 경우는 31번의 두배 연산과 72번의 덧셈 연산이 소요되고, Type II의 경우는 31번의 두배 연산과 60번의 덧셈 연산이 소요된다. 이는 Kobayashi 등의 방법이 Type I의 경우 143회와 Type II의 경우 108회인 것에 비해 각각 40%, 20% 정도 향상된 수치이다. 또한 이진 방법보다는 3배 이상 빠름을 알 수 있다. 이를 표 2에 정리하였다.

$p=2^{61}-1$, $m=5$의 경우는 상수 k를 Type I과 Type II로 base-ϕ 전개했을 경우 n을 1부터 5까지 선택했을 때, c_iP당 필요한 덧셈 횟수의 분석 값과 시뮬레이션 값은 각각 그림 4와 그림 5와 같다. 이 경우는 $n=5$일 경우 최적의 값을 갖는 것을 확인할 수 있다. 따라서 $m=5$이므로 한 번에 일괄처리 기법을 이용하여 처리하면 된다. 상수배 연산에 소요되는 총 연산수를 표 3에 정리하였다.

마지막으로 상수배 연산에 소요되는 메모리양을 분석한다. $p=2^{31}-1$, $m=7$ 경우와 같이 한 번에 일괄처리 기법으로 처리가 안 되는 경우는 알고리즘 1에서 2^iP를 테이블에 저장할 필요가 있다. 따라서

표 3. $E(GF(2^{61}-1)^5)$에서 상수배 연산 알고리즘에 필요한 연산수의 비교

알고리즘	전개 Type	두배	덧셈	총합
Binary		$mA-1$ (304)	$mA/2$ (152)	456
Signed Binary		$mA-1$ (304)	$3mA/8$ (101)	405
Kobayashi 등[12]	Type I	A (61)	$m(A+1)/2$ (155)	216
	Type II	A (61)	$m\overline{w_H}$ (100)	161
Proposed	Type I	A (61)	81	142
	Type II	A (61)	70	131

$m=5$, $A=\lceil \log_2 p \rceil=61$, $\overline{w_H}=20$

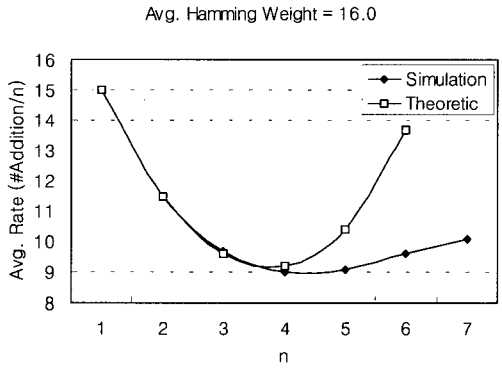


그림 2. $\overline{w_H}=16$일 때의 c_iP당 필요한 덧셈 수(Type I)

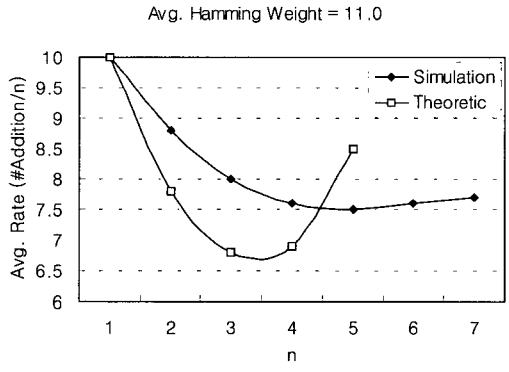


그림 3. $\overline{w_H}=11$ 일 때의 c_iP당 필요한 덧셈 수(Type II)

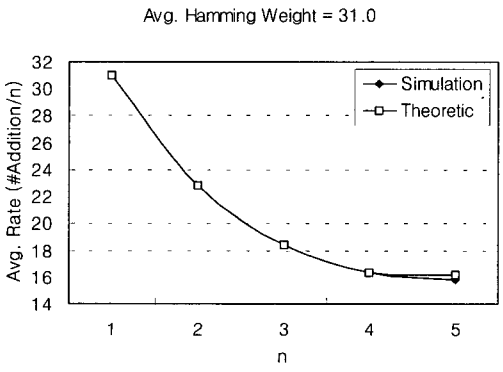


그림 4. $\overline{w_H}=31$ 일 때의 c_iP당 필요한 덧셈 수(Type I)

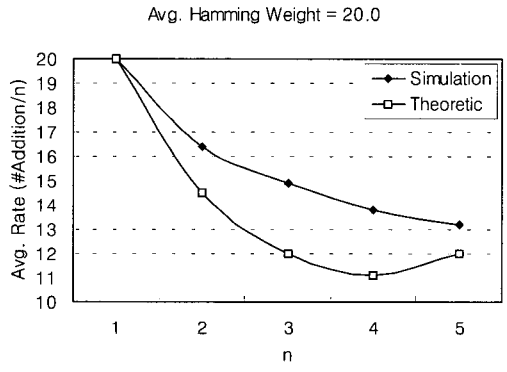


그림 5. $\overline{w_H}=20$ 일 때의 c_iP당 필요한 덧셈 수(Type II)

표 4. 상수배 연산 알고리즘에 필요한 메모리양 분석 (단위 비트)

	사례	알고리즘 1	알고리즘 2	총 합
$m>n$	$p=2^{31}-1$,	$\lceil \log_2 p \rceil \lceil \log_2 p^m \rceil$	$2^n \lceil \log_2 p^m \rceil$	$(\lceil \log_2 p \rceil + 2^n) \lceil \log_2 p^m \rceil$
	$m=7, n=4$	(6,944)	(3,584)	(10,528)
$m \leq n$	$p=2^{61}-1$,		$2^m \lceil \log_2 p^m \rceil$	$2^m \cdot \log_2 p^m$
	$m=5$		(10,240)	(10,240)

$\lceil \log_2 p^m \rceil$ 비트 원소를 $\lceil \log_2 p \rceil$ 개의 엔트리에 저장하려면 $\lceil \log_2 p \rceil \lceil \log_2 p^m \rceil$ 비트 메모리를 필요로 한다. 또한 알고리즘 3에서 n개 단위로 일괄처리를 하므로 알고리즘 2에 필요한 메모리양은 결합 과정에서 2^n개의 $\lceil \log_2 p^m \rceil$ 비트 원소를 테이블에 저장해야 하므로 $2^n \lceil \log_2 p^m \rceil$ 비트 메모리를 필요로 한다. 반면 $p=2^{61}-1$, $m=5$와 같은 경우는 한번에 일괄처리 기법을 적용할 수 있으므로 알고리즘 1을 한번만 실행하면 된다. 따라서 $2^i P$를 테이블에 저장할 필요가 없다. 따라서 알고리즘 2에 필요한 메모리양만을 고려하면 된다. 알고리즘 2에서는 결합 단계를 위해 2^m개의 $\lceil \log_2 p^m \rceil$ 비트 원소를 테이블에 저장해야 하므로 $2^m \lceil \log_2 p^m \rceil$ 비트 메모리만을 필요로 한다. 이를 종합하여 표 4에 정리하였다.

Ⅵ. 결 론

본 논문에서는 OEF에서 정의되는 타원곡선에서 효율적인 상수배 연산 알고리즘을 제안하였다. 이 알고리즘은 base-ϕ 전개 방식에 최적화된 일괄처리 기법을 적용한 방법으로 Kobayashi 등의 방법에 비해 Type I 전개에 대해서는 40%정도, Type II 전개에 대해서는 20%정도의 속도 개선이 있었다. 또한 이는 기존의 이진 방법에 비해서는 3배 이상 빠른 성능을 보였다.

일괄 처리 기법은 주로 인증 서버와 같이 동시에 많은 서명을 생성하고 검증하는 경우에 적용 가능한 방법이었다. 그러나 본 논문에서는 한 서명 내에서도 일괄처리 기법을 적용할 수 있도록 함으로써, 클라이언트 측에서도 효율적으로 서명을 생성하고 검증할 수 있는 방법을 제공하였다. 그러나 이 방법은 일괄처리 기법을 사용함에 따라 Kobayashi 등의 방법보다는 약간의 메모리를 더 사용하는 단점이 있다. 이는 작은 상수배 계산 과정과 결합 단계에서 필요한 부분인데, 그 양은 수 KB 정도이며 그리 부담되는 양은 아니다.

일괄처리 기법을 이용한 base-ϕ 상수배 연산을 타원곡선 암호시스템에 적용함으로써, 고속의 소형 암호시스템을 필요로 하는 임베디드 시스템과 같은 분야에서 타원곡선 암호시스템이 효과적으로 활용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] V. Miller, "Use of Elliptic Curve in Cryptography," in *Advances in Cryptology - CRYPTO'85, LNCS*, pp. 417-426, 1985.

[2] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203-209, 1987.

[3] M. Wiener and R. Zuccherato, "Fast Attacks on Elliptic Curve Cryptosystems," in *Selected Areas in Crypto- graphy - SAC'98*, 1998.

[4] A. Menezes, "Elliptic Curve Cryptosystems," *CryptoBytes*, vol. 1, no. 2, pp. 1-4, 1995.

[5] D. Gordon, "A Survey of Fast Exponentiation Methods," *Journal of Algorithms*, vol. 27, pp. 129-146, 1998.

[6] N. Koblitz, "CM-curves with Good Cryptogrpahic Properties," in *Advances in Cryptology - CRYPTO'91, LNCS 576*, pp. 279-287, 1991.

[7] V. Muller, "Fast Multiplication on Elliptic Curves over Small Fields of Characteristic Two," *Journal of Cryptology*, vol. 11, pp. 219-234, 1998.

[8] J. Cheon, S. Park, S. Park and D.

Kim, "Two Efficient Algorithms for Arithmetic of Elliptic Curves Using Frobenius Map," in *Proceedings of the First International Workshop - PKC'98, LNCS*, pp.195-202, 1998.

[9] N. Koblitz, "An Elliptic Curve Implementation of the Finite Field Digital Signature Algorithm, in *Advances in Cryptology - CRYPTO'98, LNCS 1462*, pp. 327-337, 1998.

[10] N. Smart, "Elliptic Curve Cryptosystem over Small Fields of Odd Characteristic," *Journal of Cryptology*, vol. 12, pp. 141-151, 1999.

[11] D. Bailey and C. Paar, "Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms," in *Advances in Cryptology - CRYPTO'98, LNCS 1462*, pp. 472-485, 1998.

[12] T. Kobayashi, H. Morita, K. Kobayashi and F. Hoshino, "Fast Elliptic Curve Algorithm Combining Frobenius Map and Table Reference to Adapt to Higher Characteristic, "in *Advances in Cryptology - EUROCRYPT'99, LNCS 1592*, pp. 176-189, 1999.

[13] D. Knuth, *The Art of Computer Programming. Volume 2: Seminumerical Algorithms*, Addition-Wesley, 1981.

[14] D. M'Raihi and D. Naccache, "Batch Exponentiation : A First DLP-based Signature Generation Strategy," in *3rd ACM Conference on Computer and Communication Security*, pp. 58-61, 1996.

[15] TTA 표준 : 부가형 전자 서명 방식 표준 - 제3부 : 타원곡선을 이용한 인증서 기반 전자 서명 알고리즘, 2001년 11월.

〈著者紹介〉

정 병 천 (Byungchun Chung)
1998년 2월: 성균관대학교 정보공학과 졸업
2001년 2월: 한국과학기술원 전산학과 석사
2001년 3월~현재: 한국과학기술원 전자전산학과 박사과정
〈관심분야〉 암호학, 네트워크 보안

이 수 진 (Soojin Lee)
1992년 3월: 육군사관학교 전산과 졸업
1996년 2월: 연세대학교 컴퓨터과학과 석사
1996년~1999년: 육군교육사령부 전산개발처
1999년~2001년: 육군사관학교 전산소
2002년 3월~현재: 한국과학기술원 전자전산학과 박사과정
〈관심분야〉 정보보호, 침입탐지, Ad-hoc 및 센서 네트워크 보안

홍 성 민 (Seong-Min Hong)
1994년 2월: 한국과학기술원 전산학과 졸업
1996년 2월: 한국과학기술원 전산학과 석사
2000년 8월: 한국과학기술원 전산학과 박사
2000년~2004년: (주)트러스컴 개발실장
2004년 9월~현재: 한국과학기술원 전자전산학과 BK 초빙교수
〈관심분야〉 암호학, 암호 프로토콜

윤 현 수 (Hyunsoo Yoon)
1979년 2월: 서울대학교 전자공학과 졸업
1981년 2월: 한국과학기술원 전산학과 석사
1988년 8월: 오하이오 주립대학 전산학 박사
1988년~1989년: AT&T Bell Labs. 연구원
1989년~현재: 한국과학기술원 전자전산학과 교수
〈관심분야〉 정보보호; Ad-hoc 및 센서 네트워크 및 보안, 차세대 이동통신