通信情報保護學會論文誌
第 10 券 第 1 號, 2000. 3

GF(p^m)상에서 정의되는 타원곡선을 위한 복합 좌표계 응용

정 재 욱*, 심 상 규**, 이 필 중***

Application of Mixed Coordinate Technique for Elliptic Curves Defined over GF(p^m)

Jae wook Chung*, Sang Gyoo Sim**, Pil Joong Lee***

요 약

타원곡선 이산대수 문제에 기초한 공개키 암호시스템에서 타원곡선 멱승은 반드시 필요한 연산이며 연산들 중에서 가장 복잡도가 크다. 따라서 효율적인 암호시스템 구현을 위해서는 타원곡선 멱승연산을 효율적으로 구현하는 것이 중요하다. 본 논문에서는 복합 좌표계(mixed coordinate system)[9]를 이용한 멱승 방법을 GF(p^m)상에서 정의되는 타원 곡선에 적용하여 최적의 효율성을 갖는 타원곡선 멱승 구현법을 제안한다. 또한 '곱셈을 이용한 역원 연산 알고리즘(IM; Inversion with Multiplication)'[2]을 이용하여 더욱 효율적인 구현이 가능함을 보인다.

ABSTRACT

In public key cryptosystems based on elliptic curve (EC) discrete logarithm problem, EC exponentiation is the most fundamental and also the most time-consuming function among EC functions. Thus it is important to implement efficient EC exponentiation to have an efficient EC cryptosystem. In this paper, we propose an efficient EC exponentiation algorithm which is optimized for *GF(p^m)*, by applying mixed coordinate technique.[9] In addition, we show that a more efficient implementation is possible by utilizing 'IM (inversion with multiplication),'[2] algorithm.

keywords : *elliptic curve, mixed coordinates, EC exponentiation, optimal extension field, public key cryptosystem*

Ⅰ. 서 론

타원곡선 암호시스템은 Koblitz[14,15]와 Miller[16]에 의해서 제안된 공개키 암호시스템이다. 이러한 타원곡선 암호시스템의 장점은 유한체에서의 이산대수 문제에 비해 타원곡선에서의 이산대수 문제가 매우 어렵다는 것과, 다른 암호시스템과 비교하여 같은 안전도를 유지하는데 필요한 키 길이가 매우 작아서 스마트카드와 같은 제약적인 환경에서 유용하게 사용될 수 있다는 것이다. 또한 타원곡선 이산대수는 다른 유한체에서의 이산대수와는 달리 부지수(sub-exponential) 알고리즘이 알려지지 않아서 매우 안전한 것으로 알려져 있다.

암호학적 용도를 위한 타원곡선은 유한체상에서 정의되는데, 이러한 유한체로 널리 쓰이는 것들은 *GF(p)*, *GF(2^m)* 그리고 *GF(p^m)*이다. 이제까지는 *GF(p)*와 *GF(2^m)*에서의 연구가 많이 행해져왔다. 하지만 *GF(p)*의 경우 p가 160비트 정도 되어야

*, **, *** 포항공과대학교 전자전기공학과 정보 보안 및 통신 연구실
({jwchung, sim}@oberon.postech.ac.kr, pil@postech.ac.kr)

암호학적 안전도를 유지할 수 있기 때문에 복잡한 다정도 정수 연산을 필요로 한다. $GF(2^m)$은 유한체가 0과 1로만 이루어진 비트열로 표현되는데 요즘의 프로세서들은 대부분 워드단위로 데이터를 처리하기 때문에 소프트웨어 구현시 비트단위 처리에 부담이 있다는 단점이 있다. 한편 $GF(p^m)$은 OEF (Optimal Extension Field)[4]의 조건을 만족시키도록 정의하면 프로세서가 워드단위로 데이터 처리하는 것을 최대한 활용할 수 있어서 매우 효율적인 구현이 가능하다. 이러한 이유로 최근에는 $GF(p^m)$에서 정의되는 타원곡선에 대한 연구가 많이 이루어지고 있다.

타원곡선에서의 이산대수 문제에 기초한 암호시스템들은 모두 타원곡선 멱승을 반드시 필요로 하고 있다. 그런데 타원곡선 암호시스템을 이루는데 필요한 연산 중 타원곡선 멱승 연산은 가장 복잡도가 크고 시간이 많이 걸리는 연산이므로 효율적인 타원곡선 암호시스템을 위해서는 효율적인 타원곡선 멱승 연산의 구현이 선행되어야 한다.

타원곡선 멱승을 효율적으로 연산하는 방법으로 복합 좌표계[9]를 이용한 멱승 연산 방법이 있으나 구현 결과들은 모두 $GF(p)$의 연산 속도에만 근거한 것이었다. 이 논문에서는 복합 좌표계를 이용한 멱승 방법을 $GF(p^m)$에서 정의된 타원곡선에도 적용하여 그 성능을 분석하고 실제 구현 결과를 보이도록 하겠다.

본 논문의 2절에서는 '곱셈을 이용한 유한체 역원 연산 알고리즘(IM)'과 여러 유한체 연산들의 구현 결과를 보인다. 3절에서는 사전 계산을 저장하지 않는 멱승 연산 알고리즘들 중에서 가장 효율적인 알고리즘이라고 알려진 signed window 방식의 타원곡선 멱승 알고리즘에 대한 설명을 한다. 4절에서는 기존의 연구들[2,9]에서 소개된 여러 좌표계들에 대해서 $GF(p^m)$상의 타원곡선 연산 알고리즘과 이에 필요한 유한체 연산수를 열거한다. 5절에서는 효율적인 복합 좌표계를 선택하는 구체적인 방법에 대한 기술을 하고 6절에서는 최적화된 타원곡선 멱승의 구현 결과를 보인다.

II. 유한체 연산

타원곡선에서의 연산은 유한체 연산을 기본으로 하기 때문에 효율적인 타원곡선 암호시스템을 위해서는 효율적인 유한체 연산이 먼저 구현되어야 한다. 이 절에서는 유한체 연산들 중에서 가장 복잡도가 큰 역원 연산을 효율적으로 계산하기 위한 알고리즘으로 '곱셈을 이용한 유한체 역원 연산 알고리즘 (IM)'[2]을 소개하고, 유한체 연산의 구현 결과를 보인다.

1. $GF(p^m)$에서의 역원 연산

다음은 '곱셈을 이용한 유한체 역원 연산 알고리즘 (IM)'이다. 이 알고리즘을 통해 효율적인 유한체 역원 연산을 구현할 수 있었다. $\deg(A) < \deg(f) = m$인 $A(x)$와 $GF(p^m)$의 기약 다항식인 $f(x)$를 입력값이라고 할 때, 다음 알고리즘은 $A(x)B(x) \equiv 1 \bmod f(x)$인 $B(x)$를 리턴한다.

[알고리즘 IM]

Step 1. $B \leftarrow 0$, $C \leftarrow 1$, $F \leftarrow f(x)$, $G \leftarrow A(x)$로 초기화

Step 2. $\deg(F) = 0$이면 $B \leftarrow B \cdot (F_0^{-1} \bmod p)$, B를 리턴

Step 3. $\deg(F) < \deg(G)$이면 $\{F, B\}$와 $\{G, C\}$를 서로 교환

Step 4. $j = \deg(F) - \deg(G)$

(a) $j \neq 0$ 이면 다음을 수행

$\alpha \leftarrow G_{\deg(G)}^2 \bmod p$,

$\beta \leftarrow F_{\deg(F)} G_{\deg(G)} \bmod p$,

$\gamma \leftarrow G_{\deg(G)} F_{\deg(F)-1} - F_{\deg(F)} G_{\deg(G)-1} \bmod p$,

$F \leftarrow \alpha F - (\beta x^j + \gamma x^{j-1}) G$,

$B \leftarrow \alpha B - (\beta x^j + \gamma x^{j-1}) C$.

(b) $j = 0$ 이면 다음을 수행

$F \leftarrow G_{\deg(F)} F - F_{\deg(F)} G$,

$B \leftarrow G_{\deg(F)} B - F_{\deg(F)} C$.

Step 5. Step 2로 간다.

위 알고리즘에서 대문자로 표시된 문자는 x에 대한 다항식이다. $\deg(F)$는 F의 차수를, F_i는 다항식 F에서 x^i의 계수를 뜻한다.

IM은 기존의 유한체 역원 연산 알고리즘인 확장 유클리드 알고리즘 (EEA),[6] Almost Inverse 알고리즘[1] 또는 그 변형들 (IP,[2] MAIA[7])과는 달리 다항식의 차수 감소시 부분체 역원 연산을 필

요로 하지 않는다는 것이 장점이다. 부분체 역원 연산은 다른 부분체 연산들에 비해 상대적으로 매우 복잡도가 큰 연산이므로 부분체 역원 연산의 효율성이 유한체 역원 연산 알고리즘의 효율성을 크게 좌우한다. p의 비트수가 작은 경우(8또는 16비트) 모든 부분체 원소에 대한 역원을 사전 계산해 둘 것을 [3]에서 제안하고 있지만, p의 비트수가 그 이상 커질 경우 사전 계산에 필요한 메모리가 매우 커져서 유용하지 않다. 한편 [2]에서는 p의 비트수가 클 경우 부분적인 사전 계산을 하는 방법을 소개하고 있는데, IM에서의 부분체 역원 연산은 단 한번 밖에 사용되지 않으므로 사용되는 메모리 양에 비해 그 효율이 상당히 적다.

IM은 IP에서의 차수 감소시 필요한 부분체 역원 연산을 대략 m번의 부분체 곱셈으로 대체하고 있다.[2] 일반적으로 부분체 역원 연산은 유한체 곱셈 연산에 비해 복잡도가 비슷하거나 더 큰 연산이므로, IM은 기존의 유한체 역원 연산에 비해 효율적이라고 볼 수 있다. 다음의 표 1은 기약 다항식 $f(x)$가 이항식이고, $GF(p^5)$의 구현시에 필요한 부분체 곱셈과 부분체 역원수를 보인다. IM은 부분체 역원 회수가 다른 유한체 역원 알고리즘들과는 달리 단 1회만을 필요로 한다.

[표 1] 역원 연산 알고리즘들의 연산수 비교
[Table 1] Computation amount of finite field inversion algorithms

역원 알고리즘	부분체 곱셈	부분체 역원
EEA	41회	9회
IP	47회	6회
IM	67회	1회

2. $GF(p^m)$ 연산의 구현 결과

표 2는 $GF(p^m)$에서의 구현 결과를 보여주고 있다. 32비트 프로세서인 Pentium II 400MHz PC에서 구현하였기 때문에 p는 2^{32}보다 작으면서 가장 큰 소수인 $2^{32}-5$로 선택하였으며, 유한체의 크기가 160비트가 되도록 $m=5$를 선택하였다. 기약 다항식은 $f(x)=x^5-2$를 사용하고 Microsoft Visual C++ 6.0 컴파일러를 사용하여 C언어로만 구현하였다. 부분체 역원 연산을 위한 사전 계산은 사용하지 않았다.

[표 2] 유한체 연산의 구현결과
[Table 2] Implementation result of finite field operations

유한체 연산	시간(μs)
덧셈	0.191
뺄셈	0.213
곱셈 (M)	2.594
제곱 (S)	2.114
부분체 역원	2.564
EEA	33.539
IP	21.932
IM (I)	12.409
S/M	0.814
I/M	4.784

표 2를 통해 앞 절에서 소개한 IM 유한체 역원 연산 알고리즘이 기존의 EEA나, IP에 비해 매우 효율적임을 확인할 수 있다.

III. Signed window 방식의 타원곡선 멱승 알고리즘

Signed window 방식의 멱승 알고리즘은 사전 계산의 결과값을 저장하지 않는 멱승 알고리즘들 중에서 가장 효율적인 타원곡선 알고리즘으로 알려져 있다. l 비트의 길이를 갖는 정수 $k(|k|=1)$와 타원곡선상의 한 점 P에 대해 kP를 계산할 때, 윈도우의 크기를 w라 한다면 k는 다음과 같은 signed window로 표현될 수 있다.

$$k=2^{k_0}(2^{k_1}(\cdots 2^{k_{v-1}}(2^{k_v}W[v]+W[v-1]) + W[v-2]\cdots)+W[0]) \quad (1)$$

이때 $W[i]$는 홀수이고, $-2^w+1\le W[i]\le 2^w-1$의 범위를 가지며, $w\le ki$인 조건을 만족한다. Singed window 방식의 타원곡선 멱승 연산은 $Pi=iP(i=\pm1,\pm3,\ldots,\pm(2^w-1))$의 사전 계산과 다음과 같은 타원곡선 연산을 통해 이루어진다.

$$kP=2^{k_0}(2^{k_1}(\cdots 2^{k_{v-1}}(2^{k_v}P_{W[v]}+P_{W[v-1]}) + P_{W[v-2]}\cdots)+P_{W[0]}) \quad (2)$$

그런데 $-2^w+1\le W[i]\le -1$인 점들에 대한

사전 계산은 $P_{W[i]} = -P_{-W[i]}$로 쉽게 계산할 수 있으므로 사전 계산은 $1 \le W[i] \le 2^w - 1$에 대해서만 저장을 해두어도 된다. 이럴 경우, 타원곡선 뺄셈 연산을 따로 구현하여 사용하거나, 타원곡선 뺄셈 연산이 필요할 때마다 $-Pi$를 계산하여 더하는 방식을 취해야 한다.

식 (2)에서 맨 처음 연속되는 두배 연산은 더욱 효율적으로 연산할 수 있다. 만약 $W[v] = 1$인 경우 k_v번의 두배 연산은 다음 식과 같은 방법으로 1번의 덧셈 연산과 $k_v - w$번의 두배 연산으로 계산 가능하다.[9]

$$2^{k_v}P_1 = 2^{k_v - w}(P_{2^w-1} + P_1) \quad (3)$$

만약 $W[v] = 3$의 경우 다음 식과 같이 계산하면 k_v번의 두배 연산이 1번의 덧셈과 $k_v - w + 1$ 번의 두배 연산으로 감소된다.[9]

$$2^{k_v}P_3 = 2^{k_v - w + 1}(P_{2^{w-1}} + P_{2^{w-1}+1}) \quad (4)$$

모든 $W[v] < 2^{w-1}$에 대해 이와 비슷한 방법을 적용할 수 있다. 이 방법을 사용하는 것이 얼마나 효과적인지 연산수 측면에서 분석해 볼 필요가 있다. 기존의 연구[9]에서도 이러한 분석이 있으나 그 분석들이 정확하지 않고 오류가 있어서 이 논문에서 정확한 분석을 보인다.

k_v의 평균값은 $w+2$이므로 위 방법을 사용하지 않을 때는 평균 $w+2$번의 두배 연산이 필요하다. 위 방법을 사용할 때 평균적으로 줄어드는 두배 연산과 평균적으로 늘어나는 덧셈 연산의 수를 계산해 보자.

$W[v]$의 비트수가 j일 확률은 다음과 같다.

$$\Pr(|W[v]| = j) = \begin{cases} \frac{1}{2^{w-1}} & \text{if } j = 1 \\ \frac{1}{2^{w-j+1}} & \text{if } 2 \le j \le w \end{cases}$$

$W[v]$의 비트수가 j일 때 위 방법에 의해 줄어드는 두배 연산의 회수는 $(w+1-j)$번이고 1번의 덧셈 연산이 늘어난다. $j = w$일 때는 위의 방법을 사용하지 않는다는 사실에 유의하여 평균적으로 줄어드는 두배 연산의 수를 계산하면 $3/2 - 1/2^{w-1}$번이 되고, 평균적으로 1/2번의 덧셈 연산이 늘어난다는 사실을 쉽게 계산할 수 있다.

Ⅳ. 좌표계

타원곡선은 Affine 좌표계,[10,12] Projective 좌표계,[13] Jacobian 좌표계,[5] 그리고 Chudnovsky Jacobian 좌표계[5] 등의 여러 가지 좌표계에서 표현 될 수 있다. 이 절에서는 이러한 여러 좌표계에서의 타원곡선 연산 공식과 그에 필요한 유한체 연산수를 열거한다. 또한 변형된 Jacobian 좌표계도 함께 소개한다.[2, 9]

1. Affine 좌표계

Weierstrass 등식 $y^2 = x^3 + ax + b$를 만족시키는 Affine 좌표계로 표현된 점 $P = (x_1, y_1)$, $Q = (x_2, y_2)$에 대해 $P + Q = (x_3, y_3)$를 계산하는 연산식은 다음과 같다.

1) Affine 좌표계에서의 덧셈 연산 $(P \neq \pm Q)$

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\lambda = (y_2 - y_1)/(x_2 - x_1)$$

2) Affine 좌표계에서의 두배 연산 $(P = Q)$

$$x_3 = \lambda^2 - 2_{x1}$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\lambda = (3x_1^2 + a)/(2y_1)$$

연산수 : $I(A + A) = I + 2M + S$

$$I(2A) = I + 2M + 2S$$

2. Projective 좌표계

Weierstrass 등식 $E_p: Y^2Z = X^3 + aXZ^2 + bZ^3 (x = X/Z, y = Y/Z)$를 만족시키는 점 $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$에 대해 $P + Q = (X_3, Y_3, Z_3)$를 계산하는 연산식은 다음과 같다.

1) Projective 좌표계에서의 덧셈 연산 $(P \neq \pm Q)$

$X_3 = vA$

$Y_3 = u(v^2X_1Z_2 - A) - v^3Y_1Z_2$

$Z_3 = v^3Z_1Z_2$

$u = Y_2Z_1 - Y_1Z_2$

$v = X_2Z_1 - X_1Z_2$

$A = u^2Z_1Z_2 - v^3 - 2v^2X_1Z_2$

2) Projective 좌표계에서의 두배 연산 ($P = Q$)

$X_3 = 2hs$

$Y_3 = w(4B - h) - 8Y_1^2s^2$

$Z_3 = 8s^3$

$w = aZ_1^2 + 3X_1^2$

$s = Y_1Z_1$

$B = X_1Y_1s$

$h = w^2 - 8B$

연산수 : $t(P + P) = 12M + 2S$

$t(2P) = 7M + 5S$

3. Jacobian 좌표계

Weierstrass 등식 E_j: $Y^2 = X^3 + aXZ^4 + bZ^6$ ($x = X/Z^2$, $y = Y/Z^3$)를 만족시키는 점 $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$에 대해 $P + Q = (X_3, Y_3, Z_3)$를 계산하는 연산식은 다음과 같다.

1) Jacobian 좌표계에서의 덧셈 연산 ($P \neq \pm Q$)

$X_3 = -H^3 - 2U_1H^2 + r^2$

$Y_3 = -S_1H^3 + r(U_1H^2 - X_3)$

$Z_3 = Z_1Z_2H$

$U_1 = X_1Z_2^2$

$U_2 = X_2Z_1^2$

$S_1 = Y_1Z_2^3$

$S_2 = Y_2Z_1^3$

$H = U_2 - U_1$

$r = S_2 - S_1$

2) Jacobian 좌표계에서의 두배 연산 ($P = Q$)

$X_3 = T$

$Y_3 = -8Y_1^4 + K(S - T)$

$Z_3 = 2Y_1Z_1$

$S = 4X_1Y_1^2$

$K = 3X_1^2 + aZ_1^4$

$T = -2S + K^2$

연산수 : $t(J + J) = 12M + 4S$

$t(2J) = 4M + 6S$

4. Chudnovsky Jacobian 좌표계

Jacobian 좌표계와 동일하지만, 덧셈 연산을 빠르게 계산하기 위하여 Z^2과 Z^3을 사전 계산 한다는 것이 다르다. $P = (X_1, Y_1, Z_1, Z_1^2, Z_1^3)$, $Q = (X_2, Y_2, Z_2, Z_2^2, Z_2^3)$에 대해 $P + Q = (X_3, Y_3, Z_3, Z_3^2, Z_3^3)$을 계산하는 연산식은 다음과 같다.

1) Chudnovsky Jacobian 좌표계에서의 덧셈 연산 ($P \neq \pm Q$)

$X_3 = -H^3 - 2U_1H_2 + r^2$

$Y_3 = -S_1H^3 + r(U_1H^2 - X_3)$

$Z_3 = Z_1Z_2H$

$Z_3^2 = Z_3^2$

$Z_3^3 = Z_3^3$

$U_1 = X_1(Z_2^2)$

$U_2 = X_2(Z_1^2)$

$S_1 = Y_1(Z_2^3)$

$S_2 = Y_2(Z_1^3)$

$H = U_2 - U_1$

$r = S_2 - S_1$

2) Chudnovsky Jacobian 좌표계에서의 두배 연산 ($P = Q$)

$X_3 = T$

$Y_3 = -8Y_1^4 + K(S - T)$

$Z_3 = 2Y_1Z_1$

$Z_3^2 = Z_3^2$

$Z_3^3 = Z_3^3$

$S = 4X_1Y_1^2$

$K = 3X_1^2 + aZ_1^4$

$T = -2S + K^2$

연산수 : $t(J^r + J^r) = 11M + 3S$

$t(2J^r) = 5M + 6S$

5. 변형된 Jacobian 좌표계

두배 연산을 효율적으로 행하기 위한 좌표계로 [9]에서 변형된 Jacobian 좌표계를 제안하였다. [2]에서도 이와 비슷한 좌표계를 제안하였지만 근본적으로는 [9]와 거의 동일한 기법을 사용한 것이고 단지 다음과 같은 변환식으로 $x = X/Z^2$, $y = Y/(2Z^3)$을 사용하였다는 점이 다르다.

이 변환식을 통해 일반적인 Jacobian 좌표계에 비해 타원곡선 덧셈 연산에서 유한체 원소간의 덧셈/뺄셈 회수를 1번 줄일 수 있는 장점이 있다. 이 논문에서는 위의 변환식을 사용하고 분석의 용이를 위해 [9]에서와 같이 4개의 유한체 원소로 타원곡선 상의 한 점을 표현하는 방법을 선택하였다. 실제 구현에서는 (X, Y, Z)와 같이 3개의 유한체 원소로 타원곡선 상의 한 점을 표현하고 나머지 한 원소 (aZ^4)는 전역 변수로 설정하여 사용하여도 된다.

이 좌표계에서의 점 $P = (X_1, Y_1, Z_1, aZ_1^4)$, $Q = (X_2, Y_2, Z_2, aZ_2^4)$에 대해 $P + Q = (X_3, Y_3, Z_3, aZ_3^4)$를 계산하는 연산식은 다음과 같다.

1) 변형된 Jacobian 좌표계에서의 덧셈 연산 $(P \neq \pm Q)$

$X_3 = D^2 - 4AB^2$

$Y_3 = D(4AB^2 - 2X_3) - 4B^3C$

$Z_3 = 2BZ_1Z_2$

$aZ_3^4 = aZ_3^4$

$A = X_1Z_2^2 + X_2Z_1^2$

$B = X_1Z_2^2 - X_2Z_1^2$

$C = Y_1Z_2^3 + Y_2Z_1^3$

$D = Y_1Z_2^3 - Y_2Z_1^3$

2) 변형된 Jacobian 좌표계에서의 두배 연산 $(P = Q)$

$X_3 = A^2 - B$

$Y_3 = A(B - 2X_3) - C$

$Z_3 = Y_1Z_1$

$aZ_3^4 = C(aZ_1^4)$

$A = 3X_1^2 + (aZ_1^4)$

$B = 2X_1Y_1^2$

$C = Y_1^4$

연산수 : $t(J^m + J^m) = 13M + 6S$

$t(2J^m) = 4M + 4S$

6. 타원곡선 연산의 계산량

표 3은 가능한 타원곡선 연산들 중에서 비교적 효율적인 것들을 유한체 연산수로 비교하고 있다. Affine 좌표계는 A, Jacobian 좌표계는 J, 변형된 Jacobian 좌표계는 J^m, Chudnovsky Jacobian 좌표계는 J^r 그리고 Projective 좌표계는 P로 표시하였다. 그리고 $t(C^1 + C^2 = C^3)$는 C^1좌표계의 점과 C^2좌표계의 점을 더하여 그 결과를 C^3좌표계의 점으로 표현한다는 것을 뜻하고, $t(2C^1 = C^2)$는 C^1좌표계의 점을 두배 연산하여 그 결과를 C^2좌표계의 점으로 표현한다는 것을 뜻한다.

V. 효율적인 복합 좌표계의 선택

앞 절의 표 3에서 볼 수 있듯 이 논문에서 고려된 5가지의 좌표계 중 어느 것도 덧셈 연산과 두배 연산이 동시에 효율적인 것은 없다. 예를 들어, 변형된 Jacobian 좌표계는 두배 연산이 가장 빠르지만 덧셈 연산이 매우 느리다. 반면 Projective 좌표계는 두배 연산이 가장 느리지만 덧셈 연산이 상대적으로 매우 빠르다. 이와 같은 사실을 통해 타원곡선 멱승 연산에서 두배 연산을 할 때는 두배 연산이 빠른 좌표계에서 연산을 행하고 덧셈 연산을 할 때는 덧셈 연산이 빠른 좌표계에서 연산을 행하는 것이 효율적일 것이라고 생각 할 수 있다. 이 절에서는 앞 절에서 보인 유한체 $GF(p^m)$연산의 구현 결과를 토대로 효율적인 멱승 연산을 할 수 있는 최적의 복합 좌표계를 선택하는 과정을 보이도록 하겠다.

여러 좌표를 복합적으로 이용하는 signed window 멱승 알고리즘은 사전 계산과 다음 식의 반복적 연산으로 이루어진다.[9]

$$2^k P' + P_{W[i-1]} = 2(2^{k_i-1}P') + P_{W[i-1]} \quad (5)$$

이때 P'는 중간 계산값이다. 복합 좌표계를 이용하여 식 (5)을 계산하는 과정은 다음과 같이 세 단계로 행해진다.

Step 1. C^1좌표계로 표현된 P'를 k_i-1번 두배 연산한다.

Step 2. k_i번째 두배 연산에서는 결과값이 C^2좌표계로 표현되도록 한다.

Step 3. Step 2의 결과값에 최종적으로 C^3좌표계로 표현된 사전 계산값 $P_{W[i-1]}$를 더하여 그 결과값을 C^1좌표계로 표현한다.

위의 단계를 반복적으로 적용하여 타원곡선 멱승을 계산할 수 있다. 식 (5)의 연산에 필요한 시간은 다음과 같다.

〔표 3〕 여러 좌표계에서의 타원곡선 덧셈 연산과 두배 연산의 연산수 비교[9]
〔Table 3〕 Computation amount of elliptic curve point addition and doubling

두배 연산		덧셈 연산	
연산	연산수	연산	연산수
$t(2P)$	$7M+5S$	$t(J^m+J^m)$	$13M+6S$
$t(2J)$	$5M+6S$	$t(J^m+J=J^m)$	$12M+5S$
$t(2J)$	$4M+6S$	$t(J+J=J^m)$	$12M+5S$
$t(2J^m=J)$	$4M+5S$	$t(J+J)$	$12M+4S$
$t(2J^m)$	$4M+4S$	$t(P+P)$	$12M+2S$
$t(2A=J)$	$3M+5S$	$t(J+J=J^m)$	$11M+4S$
$t(2J^m=J)$	$3M+4S$	$t(J+J)$	$11M+3S$
$t(2A=J^m)$	$3M+4S$	$t(J+J=J)$	$11M+3S$
$t(2A=J)$	$2M+4S$	$t(J+J=J)$	$10M+2S$
-	-	$t(J+A=J^m)$	$9M+5S$
-	-	$t(J^m+A=J^m)$	$9M+5S$
-	-	$t(J+A=J^m)$	$8M+4S$
-	-	$t(J+A=J)$	$8M+3S$
-	-	$t(J+A=J)$	$8M+3S$
-	-	$t(J^m+A=J)$	$8M+3S$
-	-	$t(A+A=J^m)$	$5M+4S$
-	-	$t(A+A=J)$	$5M+3S$
$t(2A)$	$2M+2S+I$	$t(A+A)$	$2M+S+I$

$$(k_i-1)\cdot t(2C^1) + t(2C^1=C^2) + t(C^2+C^3=C^1)$$

이제 위의 식을 가장 효율적으로 계산할 수 있는 좌표 C^1, C^2 그리고 C^3를 선택해야 한다. 우선 C^1에서의 두배 연산이 가장 빈번하므로 C^1으로는 $t(2C^1)$이 가장 효율적인 변형된 Jacobian 좌표 (J^m)를 선택한다 ($C^1=J^m$).

Signed window 방식의 멱승 연산은 매번 멱승을 계산할 때마다 사전계산을 행하게 된다. 이러한 사전계산도 알고리즘의 수행 시간의 일부분이므로 가장 최단 시간내에 계산할 수 있도록 최적화 하여야 한다. 사전 계산을 효율적으로 행할 수 있는 좌표계를 선택하는 방법을 알아보자.

1. 효율적인 사전 계산이 가능한 좌표계의 선택

[9]에서는 Affine 좌표계 또는 Chudnovsky·Jacobian 좌표계를 사전 계산점의 좌표계로 선택할 것을 제안하였고, 단순히 $t(J^r+J^s)$와 $t(A+A)$의 속도를 비교하여 두 좌표중 하나를 선택하는 방법을 취하고 있다. 여기서는 두 경우에 대해 모두 연산수를 비교하여 최적의 좌표계와 최적의 사전 계산 알고리즘을 선택하기로 한다. [9]에서는 유한체 $GF(p)$에서의 역원 연산이 매우 느리다는 사실에 의해 그 두 연산에 의한 대략적 짐작만으로도 충분하였지만, $GF(p^m)$의 경우에선 앞에서 보인 것과 같이 유한체 역원 연산이 유한체 곱셈 연산에 비해 그리 느리지 않기 때문에 그러한 대략적 짐작이 충분치 않을 수도 있다.

사전 계산점의 좌표계가 Affine 좌표계인 경우 단순히 점 P에 $2P$를 계속적으로 더해감으로써 사전 계산을 할 수 있고, 이때 사전 계산의 연산량은 다음과 같다.

$$t(2A) + (2^{w-1}-1)\cdot t(A+A) = 2^{w-1}I + 2^wM + (2^{w-1}+1)S \quad (6)$$

그런데 유한체 역원 연산이 매우 느린 경우라면 'Montgomery trick of simultaneous inversion' 알고리즘[8]을 이용하여 유한체 역원 연산수를 줄이고, 대신 유한체 곱셈과 제곱 연산을 더 많이 하여 사전 계산을 할 수도 있다. 이 경우 사전 계산의 연산수는 다음과 같다.

〔표 4〕 사전 계산의 연산량 비교
〔Table 4〕 Comparison of computation amounts of various pre-computation methods

사전 계산량	연 산 수
식(6)	$8I+16M+9S\approx 61.6M$
식(7)	$4I+38M+13S\approx 67.7M$
식(8)	$77M+26S\approx 98.2M$
식(9)	$I+55M+23S\approx 78.5M$

$$wI+(5\cdot 2^{w-1}+2w-10)M + (2^{w-1}+2w-3)S \quad (7)$$

사전 계산점의 좌표계로 Chudnovsky Jacobian 좌표계를 선택할 경우 두 가지 방법에 의해 사전 계산 할 수 있고, 그 때의 연산량은 다음과 같다.

$$t(2A=J^c)+(2^{w-1}-2)\cdot t(J^c+J^c)+t(A+J^c = J^c)=(11\cdot 2^{w-1}-11)M+(3\cdot 2^{w-1}+2)S \quad (8)$$

$$t(2A+(2^{w-1}-2)\cdot t(A+J^c=J^c) + t(A+A=J^c)=I+(2^{w-2}-9)M \quad (9)$$

표 4는 $w=4$일 때 (6), (7), (8), (9)를 표 2의 결과를 이용하여 간단히 곱셈의 비로 나타낸 것이다. 사전 계산은 Affine 좌표계에서 단순히 점 $2P$를 반복적으로 더하는 방법이 최적임을 알 수 있다. 따라서 $C^3=A$로 선택하고 'Montgomery trick of simultaneous inversion'은 사용하지 않는다.

2. 사전 계산 이후 연산을 최적화 하는 복합 좌표계 선택

사전 계산을 끝내면 식(5)를 반복적으로 연산하여 멱승 연산을 행한다. 여기서는 식(5)의 연산 속도를 최적화 할 수 있는 좌표계를 선택하는 방법에 대하여 알아본다.

앞에서 $C^3=A$로 결정되었으므로 $t(2J^m=C^2)+t(C^2+A=J^m)$을 최소화하는 C^2를 구하기만 하면 된다. 가능한 C^2로는 앞에서 보인 다섯 가지 좌표계가 모두 가능하겠지만 그 중에서 효율적일 것이라 생각되는 것만을 고려해본다.

표 5에서는 세가지 경우에 대해 식 (5)의 계산에 필요한 연산수와, 이를 단일 좌표계에서 수행한 것

〔표 5〕 식(5)의 계산에 필요한 연산수
〔Table 5〕 Computation amount required to compute (5)

복합 좌표계	식(5)의 연산량	최악	평균
(J^m, J, A)	$(7.3k_i+12.1)M$	$48.6M$	$55.2M$
(J^m, J^c, A)	$(7.3k_i+12.1)M$	$48.6M$	$55.2M$
(J^m, J^m, A)	$(7.3k_i+13.1)M$	$49.6M$	$56.2M$
(A, A, A)	$(8.4k_i+7.6)M$	$49.6M$	$58.0M$
(P, P, P)	$(11.1k_i+13.6)M$	$69.1M$	$80.2M$
(J, J, J)	$(8.9k_i+15.3)M$	$59.8M$	$68.7M$
(J^c, J^c, J^c)	$(9.9k_i+13.4)M$	$62.9M$	$72.8M$
(J^m, J^m, J^m)	$(7.3k_i+17.9)M$	$54.4M$	$61.7M$

과의 연산수 비교를 보여주고 있다(유한체 연산 구현 결과의 $I/M\approx 0.8$과 $S/M\approx 4.8$을 이용하였다). 그리고 최악의 경우는 window간 거리가 1일 경우로 계산한 것이고, 평균값은 window간 거리가 2일 경우로 계산하였다. 표 5에서 최악의 경우가 평균값보다 더 작은데, 최악의 경우는 식 (5)의 계산시간이 짧더라도 window의 수가 더 많아지므로 결과적으로 전체 멱승 시간이 더 늘어나게 된다는 것에 주의해야 한다. 표 5를 통해 signed window 타원곡선 멱승을 행할 때, 단일 좌표계를 사용하는 것보다 복합 좌표계를 사용하는 것이 유리하다는 것을 알 수 있고, $(C^1, C^2, C^3)=(J^m, J, A)$ 또는 (J^m, J^c, A)이 모두 가장 효율적인 복합 좌표계임을 보여주고 있다. 그런데 $C^2=J^c$의 경우는 $C^2=J$의 경우보다 속도면의 이득이 없으면서도 2개의 유한체 원소가 더 필요한 불이익이 있다. 따라서 (J^m, J, A)를 최적의 복합 좌표계로 선택하는 것이 바람직하다.

최적의 복합 좌표계 (J^m, J, A)를 사용할 때 사전 계산을 포함한 전체 멱승의 평균 연산량은 다음과 같다. $w=4, |k|=160$을 가정하였고 유한체 연산간의 속도비는 표 2의 결과를 이용하여 계산하였다.

$$\begin{aligned} &t(2A)+(2^{w-1}-1)\cdot t(A+A)\\ &+1/2\cdot t(A+A=J^m)\\ &+(w+1/2^{w-1}-1/2)\cdot t(2J^m)\\ &+t(2J^m=J)+t(J+A=J^m)\\ &+[\{|k|-w+1-(1/2)^{w-1}\}/(w+2)-1]\\ &\cdot\{(w+1)\cdot t(2J^m)+t(2J^m=J)+t(J+A=J^m)\}\\ &\approx 8I+849.7M+763.7S\approx 1499M \end{aligned}$$

그리고 최악의 경우는 첫 번째 window 값이 1이고, window간 평균 간격이 1일 때이다. 최악의 경우의 연산량은 다음과 같다.

$$
\begin{aligned}
&t(2A) + (2^{w-1} - 1) \cdot t(A + A) + t(A + A = J^m) \\
&+ t(2J^m = J) + t(J + A = J^m) \\
&+ (|k| - 1)/(w + 1) - 1 \\
&\cdot \{w \cdot t(2J^m) + t(2J^m = J) + t(J + A = J^m)\} \\
&\approx 8I + 895.4M + 792S \approx 1567M
\end{aligned}
$$

또한 단일 좌표계(C)에서 멱승을 행할 때의 평균 연산량은 다음과 같다.

$$
\begin{aligned}
&t(2C) + (2^{w-1} - 1) \cdot t(C + C) + 1/2 \cdot t(C + C) \\
&+ (w + 1/2^{w-1} + 1/2) \cdot t(2C) + t(C + C) \\
&+ [\{|k| - w + 1 - (1/2)^{w-1}\}/(w+2) - 1] \\
&\cdot (w+1) \cdot t(2C) + t(C + C)
\end{aligned}
$$

Ⅵ. 구현 결과

타원곡선을 정의하는 유한체는 앞 절에서 사용한 $GF(p^5)$를 사용하였다 ($p = 2^{32} - 5$). 그리고 타원곡선 파라메터로는 $a = 1$, $b = x + 214783695$를 선택하였다. 이 때 a와 b는 $GF(p^5)$에서의 원소이고 x에 대한 다항식으로 표현한 것이다.

표 6은 160비트 길이의 $k(|k| = 160)$그리고 $w = 4$일 때의 구현 결과이다. 사용된 시스템은 Pentium Ⅱ 400MHz이고 Visual C++ 6.0 컴파일러를 사용하여 C 언어만으로 구현하였다. 여러 좌표계를 복합적으로 이용하여 멱승을 계산하는 알고리즘이 하나의 좌표계만을 사용하는 멱승 알고리즘보다 약 9.1% 정도 빠른 속도를 낼 수 있었다.

〔표 6〕 타원곡선 멱승의 구현 결과
〔Table 6〕 Implementation result of elliptic curve exponentiation

타원곡선 멱승(μsec)			
역원 알고리즘	좌표계	평균 연산량	구현 결과
EEA (Inv/Mult = 12.9)	(J^m, J, A)	$1571M$	4747.8
	Affine	$3110M$	8935.8
IP (Inv/Mult = 8.5)	(J^m, J, A)	$1536M$	4650.6
	Affine	$2274M$	6864.8
IM (Inv/Mult = 4.8)	(J^m, J, A)	$1506M$	4624.6
	Affine	$1570M$	5092.3
Jacobian 좌표계		$1889M$	5823.3
Projective 좌표계		$1928M$	6422.2
변형된 Jacobian 좌표계		$1727M$	5086.3

Ⅶ. 결 론

본 논문에서는 복합 좌표계를 이용한 타원곡선 멱승 방법[9]을 유한체 $GF(p^m)$에서 정의되는 타원곡선에 대해 적용하였고 그 구현 결과를 제시하였다. 또한 IM 유한체 역원 연산 알고리즘을 이용하여 더욱 효율적인 구현을 할 수 있었다. 그리고 복합좌표계를 $GF(p^m)$ ($|p| = 32$, $m = 5$)상에서 정의되는 타원곡선에 적용함으로써 가장 효율적인 타원곡선 멱승을 구현하였다. 하지만 이 논문의 구현결과에 이용된 복합좌표계 (J^m, J, A)가 반드시 모든 타원곡선 암호시스템에서 최적의 선택이지는 않다는 것에 주의해야 한다. 최적의 타원곡선 암호시스템의 구현을 위해서는 유한체 구현결과와 이 논문에서 보인 방법을 통해 최적의 복합좌표계를 선택하는 과정이 필요하다.

참 고 문 헌

〔1〕 A. Schroeppel, H. Orman, S. O'Malley and O. Spatschek, "Fast key exchange with elliptic curve systems," Springer-Verlag, *Advances in Cryptology-CRYPTO'95*, LNCS 963, pp. 43-56, 1995.

〔2〕 C. H. Lim and H. S. Hwang, "Fast Implementation of Elliptic Curve Arithmetic in $GF(p^n)$," Springer-Verlag, Public Key Cryptography, LNCS 1751, pp. 405-421, 2000.

〔3〕 김덕수, "$F(p^m)$에서의 효율적인 연산과 이를 이용한 타원곡선 암호시스템의 구현," 석사졸업논문, 포항공과대학교 전자전기공학과, Dec. 1998.

〔4〕 D. V. Bailey and C. Paar, "Optimal extension field for fast arithmetic in public key algorithms," Springer-Verlag,

Advances in Cryptology-Crypto'98, LNCS 1462, pp. 472-485, 1998.

[5] 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 Math.*, 7, pp. 385-434, 1986.

[6] E. D. Win, A. Bosselaers, S. Vandenderghe, P. D. Gersem, and J. Vandewelle, "A fast software implementation for arithmetic operations in GF(2n)," Springer-Verlag, *Advances in Cryptology-Asiacrypt'96*, pp. 65-76, 1996.

[7] E. J. Lee, D. S. Kim, & P. J. Lee, "Speed-Up of Fp^m arithmetic for Elliptic Curve Cryptosystem," *Proceeding of ICISC'98*, pp. 81-91, Seoul, Korea, 1998.

[8] Henri Cohen, *A course in computational algebraic number theory*, Springer-Verlag, Graduate Texts in Math., 138, 1993.

[9] Henri Cohen, Atsuko Miyaji and Takatoshi Ono, "Efficient ellitic curve exponentiation using mixed coordinates," Springer-Verlag, *Advances in Cryptology-Asiacrypt'98*, LNCS 1514, pp. 50-65, 1998.

[10] IEEE P1363: Standard Specifications for Public Key Cryptography, Working Draft 13, Oct. 1999.

[11] J. A. Solina, "An improved algorithm for arithmetic on a family of elliptic curves," Springer-Verlag, *Advances in Cryptology-Crypto'97*, LNCS 1294, pp. 357-371, 1997.

[12] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, GTM 106, New York, 1986.

[13] K. Koyama and Y. Tsuruoka, "Speeding up elliptic cryptosystems by using a signed binary window method," Springer-Verlag, *Advances in Cryptology-Proceedings of Crypto'92*, LNCS 740, pp. 345-357, 1993.

[14] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comp.*, 48, pp.203-209, 1987.

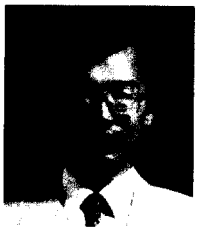
[15] N. Koblitz, *A course in number theory and cryptography*, Springer-Verlag, 2nd Edition, Graduate Texts in Math., 114, 1994.

[16] V. S. Miller, "Use of elliptic curves in cryptography", Springer-Verlag, *Advances in Cryptology-Proceedings of Crypto'85*, LNCS 218, pp. 417-426, 1986.

〈著者紹介〉

정 재 욱 (Jae Wook Chung) 학생회원
1999년 2월 : 포항공과대학교 전자전기공학과 졸업
1999년 2월~현재 : 포항공과대학교 전자전기공학과 석사과정 (정보보안 및 통신 연구실)

심 상 규 (Sang Gyoo Sim) 학생회원
1996년 2월 : 포항공과대학교 전자전기공학과 졸업
1998년 2월 : 포항공과대학교 전자전기공학과 석사졸
1998년 3월~현재 : 동대학원 박사과정(정보보안 및 통신 연구실)



이 필 중 (Pil Joong Lee) 종신회원
1974년 2월 : 서울대학교 전자공학과 졸업
1977년 2월 : 서울대학교 전자공학과 석사 졸업
1982년 6월 : U.C.L.A. System Science, Engineer
1985년 6월 : U.C.L.A. Electrical Engineering, Ph.D.
1980년 6월~1985년 8월 : Jet Propulsion Laboratory, Senior Engineer
1985년 8월~1990년 2월 : Bell Communications Research, M.T.S
1990년 2월~현재 : 포항공과대학교 전자전기공학과, 교수