타원곡선 암호 응용을 위한 개선된 최적확장체 연산

이문규
인하대학교 컴퓨터공학부
mklee@inha.ac.kr

Improved Arithmetic in Optimal Extension Fields with Application in Elliptic Curve Cryptography

Mun-Kyu Lee
School of Computer Science and Engineering, Inha University

요 약

최적확장체(Optimal Extension Field: OEF)는 유한체의 일종으로서, 타원곡선 암호시스템의 소프트웨어 구현에 있어 매우 유용하다. Bailey 및 Paar는 p^i 거듭제곱 연산을 비롯하여 다수의 효율적인 OEF 연산 알고리즘을 제안하였으며, 또한 암호 응용에 적합한 OEF를 생성하기 위한 효과적인 알고리즘을 제안하였다. 본 논문에서는 Bailey-Paar의 p^i 거듭제곱 알고리즘이 적용되지 않는 반례를 제시하며, 또한 그들의 OEF 생성 알고리즘은 실제로 OEF가 아닌 유한체를 OEF로 출력하는 오류가 있음을 보인다. 본 논문에서는 이러한 문제들을 해결한 개선된 알고리즘들을 제시하고, OEF의 개수에 관한 수정된 통계치를 제시한다.

1. 서 론

Koblitz [1]와 Miller [2]가 타원곡선 암호(Elliptic Curve Cryptography)를 처음 제안한 이래, 이에 관한 다양한 연구가 이루어진 바 있다. 타원곡선의 효율적인 실현을 위해서는 그 하위의 유한체 연산을 효율적으로 구현하는 것이 중요하다.

유한체는 소수 p와 양의 정수 m에 대해 $GF(p^m)$ 형태로 표현되는데, $p=2$를 이용하는 이진체(binary field), 큰 소수 p와 $m=1$을 이용하는 소수체(prime field) 등 다양한 유한체들이 제안된 바 있다. 그러나, 타원곡선의 소프트웨어 구현에 있어서는 최적확장체(Optimal Extension Field: 이하 OEF) [3]가 가장 효율적임이 알려져 있다 [4].

OEF는 다음을 만족하는 유한체 $GF(p^m)$이다[3].

- p는 pseudo-Mersenne 소수이다.
  즉, $\log_2 c \le \lfloor n/2 \rfloor$에 대해 $p = 2^n \pm c$ 형태의 소수이다.
- $GF(p)$상의 기약 이항식 $P(x) = x^m - w$가 존재한다.

[3]의 정리 1은 유한체가 OEF가 되기 위한 조건을 기술하고 있다.

정리 1 [3]. *$m \ge 2$을 정수라 하고, $w \in GF(p)^*$라 하자. 이때 이항식 $x^m - w$가 $GF(p)[x]$ 상에서 기약다항식일 필요충분조건은 다음과 같다.*
(i) *m의 각 소인수는 $GF(p)$ 상에서 w의 위수 e를 나누며, $(p-1)/e$는 나누지 않는다.*
(ii) *$m \equiv 0 \bmod 4$ 이면 $p \equiv 1 \bmod 4$ 이다.*

OEF 상의 원소를 표현하기 위해서는 다항식 기저 표현이 사용된다. 즉, OEF $GF(p^m)$ 상의 원소 $A(x)$는 $a_i \in GF(p)$에 대해

$$A(x) = a_{m-1}x^{m-1} + \Lambda + a_1 x + a_0$$

와 같이 표현된다.

OEF 상의 산술 연산은 기약다항식 $P(x)$에 대한 modulo 연산으로 정의된다. 따라서, 두 원소의 덧셈 및 뺄셈은 그들의 다항식 표현에서 같은 차수의 계수들에 대한 modulo 덧셈 및 modulo 뺄셈으로 정의된다. 두 원소의 곱셈은 다항식 곱셈 및 $P(x)$에 대한 modulo 연산의 두 단계로 구성된다.

한편, 역원(inversion) 연산은 타 연산에 비해 상대적으로 복잡한데, Bailey와 Paar [3]는 $A(x) \in GF(p^m)$, $r = (p^m - 1)/(p-1)$에 대해 $A^r(x)$이 하위체 $GF(p)$ 상에 있다는 사실을 이용한 효율적인 역원 계산 알고리즘을 제안한 바 있다 (알고리즘 1).

알고리즘 1. OEF상의 역원 계산 [3]
입력: 0이 아닌 $A(x) \in GF(p^m)$.
출력: $B(x) \in GF(p^m)$ s.t. $A(x)B(x) \equiv 1 \bmod P(x)$
1. $B(x) \leftarrow A^{r-1}(x)$.
2. $c_0 \leftarrow A(x)B(x)$. ▷ $c_0 = A^r(x) \in GF(p^m)$.
3. $c \leftarrow c_0^{-1} \bmod p$. ▷ $c = A^{-r}(x) \in GF(p^m)$.
4. $B(x) \leftarrow B(x)c$.

이 알고리즘에서 가장 시간이 많이 소요되는 부분은 1번 과정이다. r은 $r = p^{m-1} + p^{m-2} + \Lambda + p + 1$을 만족하므로, 우리는 p진 표현 $r - 1 = (11\mathrm{K}\ 10)_p$를 얻을 수

있으며, 따라서 $A^{r-1}(x)$는 p진 덧셈 사슬을 이용하여 계산 가능하다. 예를 들어, $m=6$에 대해 $A^{r-1}(x)$의 계산은 다음과 같이 할 수 있다.

$$B \leftarrow A^p = A^{(10)}; C \leftarrow BA = A^{(11)}; B \leftarrow C^{p^2} = A^{(1100)}; B \leftarrow BC = A^{(1111)};$$
$$B \leftarrow B^p = A^{(11110)}; B \leftarrow BA = A^{(11111)}; B \leftarrow B^p = A^{(111110)}$$

이 과정을 위해서는 위에서 언급한 유한체 곱셈 알고리즘 이외에 유한체에서의 효율적인 p^i거듭제곱 알고리즘이 필요하다. [3]에서 Bailey와 Paar는 사전계산 테이블을 이용한 효율적인 p^i거듭제곱 알고리즘을 제안하였으며, 다양한 플랫폼을 위한 OEF 생성 알고리즘도 제안하였다. 또한 이 OEF 생성 알고리즘을 이용하여 다양한 n에 대해 존재하는 OEF 개수에 관한 통계치를 제시하고, 이들 OEF 중 일부를 포함하는 테이블을 제시한 바 있다.

본 논문에서는 먼저 [3]에서 제안된 p^i거듭제곱 알고리즘의 오류를 지적하는 반례를 제시하는데, 이것은 Baktir와 Sunar가 관찰한 오류[5]와 일치하는 결과이다. 다음에 [3]에서 제안된 OEF 생성 알고리즘 또한 OEF가 아닌 유한체를 OEF로 표시하여 출력하는 오류가 있음을 보인다. 이것은 [3]의 OEF 생성 알고리즘에서 유한체에 대한 여과 과정(filtering procedure)이 완전하지 않기 때문인데, 본 논문은 이러한 단점을 보완한 개선된 OEF 생성 알고리즘 및 개선된 p^i거듭제곱 알고리즘을 제안하고 OEF 개수에 관한 새로운 통계치를 제시한다.

2. OEF 상에서 p^i 거듭제곱 계산

이 절에서는 p^i 거듭제곱 연산을 살펴본다. 먼저, $a_j{}^p \equiv a_j \bmod p$ 이므로 $GF(p^m)$ 상의 원소

$$A(x) = a_{m-1}x^{m-1} + \Lambda + a_1 x + a_0$$

의 p^i 거듭제곱은 다음과 같이 표현할 수 있다.

$$A^{p^i}(x) = a_{m-1}x^{(m-1)p^i} + \Lambda + a_1 x^{p^i} + a_0 \bmod P(x) \quad (1)$$

[3]의 따름정리 2는 (1)의 각 $(x^j)^{p^i}$ 부분이

$$(x^j)^{p^i} \equiv w^{q_{ij}} x^j \bmod P(x) \quad (2)$$

를 만족한다고 주장하고 있다 (단, $q_{ij} = \lfloor jp^i / m \rfloor$). 이렇게 되면 (1)은

$$A^{p^i}(x) = a_{m-1}w^{q_{i,m-1}}x^{m-1} + \Lambda + a_1 w^{q_{i,1}} x + a_0 \bmod P(x) \quad (3)$$

로 표현 가능하며, (3)의 모든 $1 \le i, j \le m-1$에 대해 $w^{q_{ij}}$ 들이 사전 계산 가능하므로 (3)은 $m-1$ 개의 $GF(p)$ 곱셈만을 필요로 한다.

그러나 Baktir와 Sunar[5]가 지적한 대로, [3]의 따름정리 2의 증명에는 오류가 있으며, 위의 간소화는 모든 OEF에 적용 가능한 것이 아니다. (2)의 증명은 $jp^i \bmod m = j$ 라는 주장에 근거하고 있는데, 우리는 이것이 성립하지 않는 반례를 찾을 수 있다. 예를 들어, [3]의 표 9에 주어진 OEF들 중 $p = 2^8 - 15 = 241$, $m = 27$ 및 $w = 2$인 OEF를 생각해 보자. $i = 1$의 경우, $j = 9, 18$ 이외의 모든 $1 \le j \le 26$ 에 대해 우리는 $j \cdot 241^1 \bmod 27 \ne j$ 임을 확인할 수 있다. 이것은 (2)를

$$(x^j)^{p^i} \equiv w^{q_{ij}} x^{jp^i \bmod m} \bmod P(x) \quad (4)$$

로 고쳐 씀으로써 쉽게 해결될 수 있다. 즉, (4)를 바탕으로 [6]의 알고리즘을 간단히 확장하면 개선된 p^i 거듭제곱 연산 알고리즘을 얻을 수 있다. 우선 실용적인 OEF의 경우 $m << p$ 이므로 $\gcd(m, p) = 1$가 성립한다는 사실을 이용하면, $j_1 p^i \bmod m = j_2 p^i \bmod m$ 는 $j_1 = j_2$ 의 필요충분조건이 된다. 그러므로 사상

$$\pi : j \;\alpha\; jp^i \bmod m$$

는 일대일대응이 되며, 우리는 [3]에서와 똑같은 사전계산 및 온라인 계산을 하되 결과로 얻은 항들을 위 사상에 따라 재배열하면 된다. 다음 알고리즘은 사전 계산 테이블에 $w^{q_{ij}}$ $(1 \le j \le m-1)$들이 계산되어 있다고 가정하고 p^i 거듭제곱을 계산하는 알고리즘이다.

알고리즘 2. p^i 거듭제곱 계산
입력: $A(x) = a_{m-1}x^{m-1} + \Lambda + a_1 x + a_0$.
출력: $A'(x) = a'_{m-1}x^{m-1} + \Lambda + a'_1 x + a'_0 = A^{p^i}(x)$.
1. $b_j \leftarrow a_j w^{q_{ij}}$ for $1 \le j \le m-1$. ▷ 계수 계산 [3]
2. $a'_{\pi(j)} \leftarrow b_j$ for $0 \le j \le m-1$. ▷ 항 재배열

이 알고리즘은 $m-1$개의 $GF(p)$ 곱셈 및 π 사상 계산을 위한 추가적인 연산을 필요로 한다. 그러나, π 계산을 위한 추가 비용은 무시할 수 있는 정도로 작다.

3. OEF 생성을 위한 개선된 알고리즘

모든 OEF는 타원곡선의 효율적인 소프트웨어 구현을 가능하게 하나, 다음과 같이 추가적인 효율성을 제공하는 특별한 형태의 OEF들이 존재한다.

- Type I OEF: $p = 2^n \pm 1$, 즉, $c = 1$.
- Type II OEF: 기약다항식 $P(x) = x^m - 2$, 즉, $w = 2$.

Type I OEF에서는 하위체 $GF(p)$에서 더욱 효율적인 modulo 연산이 가능하며, Type II OEF에서는 modulo $P(x)$를 이용한 더욱 효율적인 확장체 연산이 가능하다.

[3]의 8절에서는 Type II OEF들을 생성하기 위한 알고리즘을 소개하고, 이 알고리즘을 이용해 생성한 OEF들을 나열한 표를 제시하고 있다. 그러나 이 표([3]의 표 9)는 실제로는 OEF가 아닌 다음과 같은 유한체들을 포함하는 오류를 범하고 있다.

- $p = 251$, $m = 25$ 에서 $GF(251)$ 상의 2의 위수는 $e = 50$이다. 한편, m의 소인수는 5뿐이며, 5는 e 뿐 아니라 $(p-1)/e = 5$ 도 나누므로, $GF(251^{25})$ 는 OEF가 아니다. 마찬가지 이유로 $GF(257^{32})$도 OEF가 아니다.
- $p = 241$, $m = 25$ 에서 $GF(241)$ 상의 2의 위수는 $e = 24$이다. 한편, m의 소인수 5는 $e = 24$를 나누지 않으므로, $GF(241^{25})$는 OEF가 아니다.

위 두 가지 반례 중 첫번째는 m의 소인수가 $(p-1)/e$를 나눌 경우 OEF가 되지 않는다는 점을 확인하지 않은 데서 기인한 것으로 보인다. 실제로 [3]의 OEF 생성 알고리즘([3]의 알고리즘 3)에는 이러한 여과 과정이 빠져 있다.

아래의 알고리즘 3은 이 여과 과정(18-21행)을 추가한 개선된 OEF 생성 알고리즘이다.

알고리즘 3. Type II OEF의 생성
입력: n, $mMax$,
유한체 위수 범위 low, $high$.
출력: Type II OEFs with $low \le mn \le high$.

```
1.  cMax ← 2^⌊n/2⌋.
2.  if cMax is even then
3.    cMax ← cMax − 1.
4.  end if
5.  c ← −cMax.
6.  while c ≤ cMax do
7.    p ← 2^n + c.
8.   if p is prime then
9.     e ← (the order of 2 in GF(p)).
10.    for m ← 2 to mMax do
11.     if low ≤ mn ≤ high then
12.       BadMValue ← 0.
13.       for each prime factor d of m do
14.        if d does not divide e then
15.          BadMValue ← 1.
16.          break.
17.        end if
18.        if d divides (p−1)/e then
19.          BadMValue ← 1.
20.          break.
21.        end if
22.       end for ▷ line 13
23.       if BadMValue = 0 then
24.        if m ≡ 0 mod 4 then
25.         if p ≡ 1 mod 4 then
26.           output p, m.
27.         end if
28.        else
29.          output p, m.
30.        end if ▷ line 24
31.       end if ▷ line 23
32.     end if ▷ line 11
33.   end for ▷ line 10
34.  end if ▷ line 8
35.   c ← c + 2.
36. end while ▷ line 6
```

위 알고리즘은 $p = 2^n \pm c$ 형태의 소수를 고정시키고 m을 바꾸면서 해당 m에 대해 기약이항식이 존재하는지 확인하는 방식으로 동작한다. 소수 p 내의 비트 수 n은 목표로 하는 마이크로프로세서의 특성에 근거하여 정해지는데, Pentium과 같은 32비트 프로세서의 경우 $n \approx 32$를 사용하면 된다. low, $high$ 및 $mMax$ 등의 인자는 OEF가 타원곡선암호를 위한 적절한 수준의 안전성 및 효율성을 제공할 수 있도록 OEF의 범위를 한정하기 위한 것으로, OEF $GF((2^n \pm c)^m)$의 위수가 대략 2^{mn}이라는 사실을 이용하여 $low \le mn \le high$이 되도록 m을 제한하는 데 사용된다. (일반적으로, $low = 130$, $high = 256$ 정도면 적당하다.) 또한, 위 알고리즘을 약간만 변형하면 Type I OEF 및 일반적인 OEF 생성도 가능하다.

4. 각 (m, n)별 OEF의 개수

[3]에서는 다양한 m 및 n에 대해 존재하는 OEF 개수에 관한 통계를 제시하고 있다. 그러나 이 통계는 오류가 있는 [3]의 알고리즘 3을 이용하여 만들어진 것이므로 역시 오류를 포함하고 있다. 이 절에서는 개선된 OEF 생성 알고리즘으로부터 생성된 OEF의 개수에 관한 새로운 통계를 제시한다. 아래 표 1은 다양한 m 및 n에 대해 존재하는 Type II OEF의 수를 나타낸다. 단, 공간을 절약하기 위해 원래 표([3]의 표 5, 6, 7)에서 오류가 없는 행은 생략하였다.

표 1. Type II OEF의 수

n	m													
	6	7	8	9	10	11	12	13	14	15	16	25	27	32
8												1	2	1
16				19	5	1	4	6	4	4	13			
19		9	21	25	4	7	6	4						
25	84	75	129	168	60									

참고문헌

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

[2] V. Miller, " Use of elliptic curves in cryptography, " Advances in Cryptology-CRYPTO ' 85, LNCS, vol. 218, 1986, pp. 417-428, Springer.

[3] D.V. Bailey and C. Paar, " Efficient arithmetic in finite field extensions with application in elliptic curve cryptography," Journal of Cryptology, vol. 14, 2001, pp.153-176.

[4] N.P. Smart, " A comparison of different finite fields for elliptic curve cryptosystems, " Computers and Mathematics with Applications, vol. 42, 2001, pp. 91-100.

[5] S. Baktir and B. Sunar, " Optimal Tower Fields," IEEE Transactions on Computers, vol. 53, no. 10, 2004, pp.1231-1243.

[6] T. Kobayashi, " Base-Φ method for elliptic curves over OEF," IEICE Trans. Fundamentals, vol. E83-A, no. 4, 2000, pp.679-686.