情報保護學會論文誌
第18卷 第6(A)號, 2008. 12

유한체 F_{p^k}에서 효율적으로 제곱근을 구하는 알고리즘들*

한 동 국[1†], 최 두 호[1], 김 호 원[2], 임 종 인[3]
[1]한국전자통신연구원, [2]부산대학교, [3]고려대학교

Efficient Computation of Square Roots in Finite Fields F_{p^k}*

Dong-Guk Han[1†], Doo Ho Choi[1], Howon Kim[2], Jongin Lim[3]
[1]ETRI, [2]Pusan National University, [3]Korea University

요 약

본 논문에서는 확장체 $F_{p^k}(k$ 홀수) 에서 효율적으로 제곱근을 구하는 방법을 제안한다. 제곱근을 구하는 알고리즘은 표수 p의 조건에 따라서 여러 가지 방법들이 제안되었다. 특히, 알고리즘을 구성하는 중심 되는 연산 중의 하나가 지수승 연산이다. 본 연구에서는 기존의 제곱근을 구하는 알고리즘에 사용되는 지수승의 지수들이 표수 p을 활용한 p-진법으로 표현할 경우, 특별한 형태의 주기성을 가지는 표현으로 나타내어짐을 증명하고, 이것을 활용해 기존의 알고리즘들을 효율적으로 계선하는 방법을 제안한다. 본 논문에서 제안한 방법은 표수 p의 조건에 의존하지 않고, 지수승 기반의 기존의 모든 제곱근 알고리즘에 적용 가능하다는 장점을 가지고 있다. 지금까지 알려진 바로는, 본 논문이 처음으로 Tonelli-Shanks 알고리즘을 효율적으로 개선하였으며, $p \equiv 1 \pmod{16}$인 경우 60% 이상의 효율성 증대가 있었다. 다른 제곱근 알고리즘에 적용한 결과들도 비교표들을 이용해 언급되어 있으며, 기존의 방법들에 비해 상당히 효율적임을 나타내고 있다.

ABSTRACT

In this paper we study exponentiation in finite fields $F_{p^k}(k$ is odd) with very special exponents such as they occur in algorithms for computing square roots. Our algorithmic approach improves the corresponding exponentiation independent of the characteristic of F_{p^k}. To the best of our knowledge, it is the first major improvement to the Tonelli-Shanks algorithm, for example, the number of multiplications can be reduced to at least 60% on average when $p \equiv 1 \pmod{16}$. Several numerical examples are given that show the speed-up of the proposed methods.

Keywords : square roots, finite fields, Tonelli-Shanks algorithm

Ⅰ. 서 론

확장체 $F_q(q=p^k$, 소수 p, 그리고 홀수 $k)$에서 제곱근을 구하는 문제는 계산 정수론과 같은 학문에서 고전적인 연구 주제 중에 하나이다. 제곱근을 구하는 알고리즘은 암호시스템을 구성하는 요소로도 활용되어지며, 예를 들어, 타원곡선 암호시스템의 경우 좌표 표현을 효율적으로 하는데 활용되어진다. 3차 다항식 $f(x)$에 대한 확장체 F_q 위에서의 타원곡선 방정식 $E: y^2 = f(x)$의 원소는 일반적으로 (a,b)와 같이 2개의 F_q 원소로 표현

접수일: 2008년 3월 4일; 채택일: 2008년 6월 30일
* "본 연구는 지식경제부 및 정보통신연구진흥원의 IT핵심기술개발사업[2005-S-088-04, 안전한 RFID/USN 을 위한 정보보호 기술] 사업의 일환으로 수행하였음
† 주저자, 교신저자 christa@korea.ac.kr

되어진다. 하지만, (a,b) 좌표 표현은 a와 한 비트 추가 정보만을 가지고도 표현 되어질 수 있다. 여기서 한 비트 추가 정보는 예를 들어, b가 x 좌표 위의 값이면 1 아래 값이면 0을 의미한다. y 좌표값 b는 $y^2=f(a)$ 의 근 중에서 한 비트 정도를 이용해 쉽게 결정이 가능하다. 즉, $2\log_2(q)$ 비트로 표현되어지던 타원곡선위의 좌표를 $\log_2(q)+1$ 만으로 표현이 가능하다.

표수 p 의 조건에 따라서 제곱근을 구하는 방법은 여러 가지로 구분된다. $p\equiv 3\,(\mathrm{mod}4)$인 경우 완전제곱수 $a\in F_q$ 의 제곱근 x 는 한 번의 지수승 $x=a^{(q+1)/4}$ 으로 계산되어 진다(k가 홀수인 경우, $q=p^k\equiv 3(\mathrm{mod}4)$ 를 만족한다).

유사하게, $q\equiv 5\ (\mathrm{mod}8)$의 경우에서도 한 번의 지수승 연산이면 제곱근을 구할 수 있다[1]. 앞의 두 경우와는 반대로, $q\equiv 1\ (\mathrm{mod}8)$의 경우에서는 제곱근을 구하는 알고리즘이 좀 더 복잡해 진다. 이 경우, 가장 효율적으로 제곱근을 구하는 방법은 Tonelli-Shanks[13] 또는 Cipolla-Lehmer[4] 알고리즘에 기반한 것들이다.

제곱근을 구하는 알고리즘을 효율적으로 개선하기 위해, 제곱근을 구하는 알고리즘의 핵심 연산인 지수승의 지수의 새로운 표현법에 대한 연구가 최근 이루어지고 있다. Crypto 2002에서 Barreto 등은 확장체 F_{p^k} (소수 p, 홀수 k)에서 $p\equiv 3\ (\mathrm{mod}4)$ 또는 $p\equiv 5\ (\mathrm{mod}8)$ 의 경우 제곱근 계산에 필요한 지수승의 지수의 표현법을 새롭게 함으로 F_p에서의 계산 복잡도를 $O(k^3\log_2 p)$에서 $O(k^2(\log_2 k+\log_2 p))$으로 감소시켰다 [2]. 최근들어, [7]에서 Kong 등은 $p\equiv 9\ (\mathrm{mod}16)$의 조건에 활용되어지는 제곱근 알고리즘의 내부 지수승의 지수에 대해 주기성을 가지는 새로운 표현법을 활용하여 기존 방법보다 월등히 효율적인 제곱근 구하는 알고리즘을 제안하였다. 기존 알고리즘의 F_p에서의 계산 복잡도가 $O(k^3\log_2 p)$인데 반해 새로운 알고리즘의 것은 $O(k^2(\log_2 k+\log_2 p))$이다. 이와 같은 특별한 지수들에 대한 새로운 표현법으로부터 도출된 효율성의 개선은 Itoh-Tsujii 알고리즘에 활용된 divide-and-conquer 기법 [5]과 정규기저의 특성에서 기인한 것이다. 유한체 F_{p^k}를 구성하는 조건을 활용해 효율적으로 제곱근을 구하는 방법들이 제안되기도 하였다[14,15].

본 논문에서는 확장체 F_{p^k}에서 제곱근을 구하는 알고리즘들의 핵심 요소 중의 하나인 지수승들을 효율적으로 계산하는 방법을 연구한다. 우리가 살펴보게 될 제곱근을 구하는 알고리즘들의 지수승들은, $\alpha,\beta\in Z$ 에 대해, $(p^k-\alpha)/\beta$ 같은 형태의 지수들을 가지고 있다.

본 논문의 첫번째 기여도는, 지수 $(p^k-\alpha)/\beta$을 표수 p에 대해 주기성을 가지는 $A+B\cdot\sum_{i=0}^{l-1}(p^w)^i$ 같은 표현법을 제안한 것이다(Proposition 1 참조). 기존에 제안된 방법들과의 차이점은, 기존 방법들은 고정된 α,β의 값에 의존한 표현법을 제안한 방면, 본 논문은 범용적으로 활용 가능한 형태로 제안함으로 표수 p의 조건에 의존하지 않고 일반적으로 모든 표수 p에 적용 가능하다는 것이다. 이렇게 표현된 지수를 이용해 밑 x에 대한 지수승을 할 경우, $x^A\cdot(x^B)^{\sum_{i=0}^{l-1}(p^w)^i}$과 같이 계산을 하게 된다.

본 논문의 두전째 기여도는, 동일한 밑 x에 대해 두번의 지수승 x^A과 x^B을 하게 되는데, 지수 B가 지수 A 와 표수 p을 이용해 표현되어짐을 보이고, 두번의 지수승을 중복되는 연산을 제거하여 효율적으로 계산하는 방법을 제안한 것이다.

지수의 표현법을 새롭게 하고, 중복되는 연산을 제거하는 이와 같은 방법들은 앞에서 언급했듯이, 표수 p에 의존하지 않고 모든 제곱근을 구하는 방법에 적용가능하다(제곱근을 구하는 방법중의 하나인 Cipolla- Lehmer 알고리즘은 Lucas 수열을 기반한 방법으로, 최근 Muller는 Lucas 수열을 효율적으로 계산하는 방법을 통해 Cipolla-Lehmer 알고리즘의 계산 복잡도를 개선하였다[9].). 지금까지 우리가 알고 있는 바로는, Tonelli-Shanks 알고리즘을 효율적으로 개선한 결과는 본 논문이 처음이며, 표수가 $p\equiv 1\ (\mathrm{mod}16)$ 의 경우 최소 60% 이상의 효율성 개선이 있었다. 뿐만 아니라, 기존에 제안된 새로운 지수 표현법에 기반한 고속 제곱근 알고리즘들 [2,7] 과의 비교에서도 상당한 효율성 개선이 있었다(확장체 F_{p^k}의 사이즈에 따른 비교 결과 [표 5, 6, 7] 참조). 예를 들어, 홀수 k의 범위를 3에서 15까지로 하고, p^k의 사이즈를 100-비트 간격으로 100- 비트 부터 400-비트까지 고려할 경우, [2]에서 제안된 Barreto 방법과의 비교에선 $p\equiv 3\ (\mathrm{mod}4)$ 인 경우 최소 26% 이상의 효율성 향상이 있었고, 특별히, $k=3$인 경우에서는 평균 45%의 효율성 증대가 있었다. 그리고, [7]에서 제안된 Kong의 알고리즘과의 비교에선 $p\equiv 9\ (\mathrm{mod}16)$ 조건에서 최소 40% 이상의 효율성 향상이 있었고, 특별

히, $k=3$인 경우에서는 평균 60%의 효율성 증대가 있었다.

Ⅱ. 제곱근을 구하는 알고리즘들

본 절에서는 기존에 제안된 제곱근을 구하는 알고리즘에 대해서 살펴보고자 한다. 각 알고리즘들은 확장체 F_q에서 제곱근을 구하는 것을 목적으로 기술되었으며, 표수 p의 조건에 관계 없이 일반적으로 제곱근을 구하는 방법부터 p의 조건에 따라서 효율적으로 구하는 방법까지 총 4가지를 소개한다.

2.1 배경지식

본 논문에서 자주 사용되는 기호 및 용어들을 정리하자.

- F_q : 위수 q을 가지는 유한 확장체(finite extension field)로, 본 논문에서는 2가 아닌 소수 p과 홀수 정수 k에 대해 $q=p^k$로 정의한다. 여기서, 소수 p은 확장체 F_q의 표수 (characteristic)라고 한다.
- M_{F_q}은 확장체 F_q에서 곱셈에 필요한 연산량을 나타낸다. 즉, $a,b\in F_q$ 에서 $a*b$을 계산하는데 필요한 연산량을 M_{F_q} 로 정의한다.
- $\left(\frac{a}{q}\right)$: a in F_q에 대한 quadratic residue test의 결과를 나타낸다. a이 F_q에서 제곱근을 가지면 $\left(\frac{a}{q}\right)=1$ 이고, 그렇지 않으면 $\left(\frac{a}{q}\right)=-1$이다.

F_q은 F_p 위에서 k 차 확장체 (k extension field)라고 불리우고, F_q은 F_p 위에서 k 차원 벡터 공간으로 생각될 수 있다. 이해를 쉽게 하기 위해, 앞으로 F_q와 F_p을 간단히 각각 확장체와 유한체로 부른다. 만일 $\{\beta_0,\beta_1,\cdots,\beta_{k-1}\}$이 F_p 위에서 F_q의 기저이면, 원소 $a\in F_q$은 $a=\sum_{j=0}^{k-1}a_j\beta_j=(a_0,a_1,\cdots,a_{k-1}),\ a_j\in F_p,\ 0\le j\le k-1$ 같이 기저를 이용하여 유일하게 표현 될 수 있다.

유한체 F_p 위에서 정의된 k 차의 기약다항식 $f(x)$에 대해 β 이 기약다항식 $f(x)$ 의 한 근일 때, k 개의 원소를 갖는 $\{\beta,\beta^p,\cdots,\beta^{p^{k-1}}\}$ 집합을 우리는 F_p 위에서 F_q 의 "정규기저(Normal Basis)" 라고 부른다. 정규기저를 이용하여 확장체의 원소를 표현할 경우 다음과 같은 장점을 활용할 수 있다.

Theorem 1. ([10]) 정규기저 $(a_0,a_1,\cdots,a_{k-1})$ 로 표현된 F_q 의 원소 a에 대해서, a^{p^m}은 a 의 k 개 계수에 대한 m cyclic shift 로 표현 가능하다. 즉,

$$a^{p^m}=(a_{k-m},a_{k-m+1},\cdots,a_{k-1},a_0,a_1,\cdots,a_{k-m-2},a_{k-m-1}).$$

Theorem 1에서 언급한 정규기저의 특성을 활용하면, s가 표수 p의 지수승으로 표현될 경우(즉, 적당한 양의 정수 c로 $s=p^c$) $a\in F_q$와 $e=1+s+\cdots+s^{n-1}$에 대해 a^e 지수승은 다음과 같이 효율적으로 계산되어 질 수 있다.

$$a^e=\begin{cases}a^{1+s+\cdots+s^{t-1}}\cdot(a^{1+s+\cdots+s^{t-1}})^{s^t} & n=2t\text{인 경우,}\\ ((a^{1+s+\cdots+s^{t-1}})\cdot(a^{1+s+\cdots+s^{t-1}})^{s^t})^s\cdot a & n=2t+1\text{인 경우.}\end{cases}$$

이와 같은 과정을 반복적으로 수행 할 경우, a^e 지수승은 $\lfloor\log_2(n)\rfloor+HW(n)-1$번의 확장체 F_q 원소의 곱셈으로 계산되어 진다. 여기서, $HW(n)$은 n의 이진수 표현에서 1의 개수 (Hamming weight)을 의미한다. 이 방법은 Itoh-Tsujii에 의해서 제안된 역원계산 알고리즘에 사용된 기법을 변형한 것이다 [5].

Theorem 2 ([2,5]) $a\in F_q$. 적당한 정수 c 대해 $s=p^c$로 두자. 그러면, $a^{1+s+\cdots+s^{n-1}}$ 계산에 필요한 연산량은 최대 $[\lfloor\log_2(n)\rfloor+HW(n)-1]\cdot M_{F_q}$.

2.2 Tonelli-Shanks 알고리즘

Tonelli-Shanks 알고리즘은 유한체 F_p에서 제곱근을 구하는 것으로, 표수 p의 조건에 의존하지 않고, 만약 a이 F_p에서 제곱근을 가지면 Tonelli-Shanks 알고리즘은 $x^2\equiv a(\mathrm{mod}p)$의 근인 x을 찾는다. 다음에 기술되는 Tonelli- Shanks 알고리즘은 논문 [12]을 기반으로 확장체 F_q에서 제곱근을 구하는 것으로 변형된 것이다.

[11] 논문에 Algorithm 1의 연산량이 잘 정리되어 있다. 만약 고정된 확장체 F_q에서 제곱근을 구하는 환경인 경우, Algorithm 1에서 단계 1과 2의 결과값 $u,s,$ 그리고 t은 a과 함께 입력값으로 처리 가능하다. 단계 3과 While 루프에서 필요로 하는 연산량은 다음과 같다. 참고로, 단계 3은 2번의 지수승과 2번의 곱셈 연산으로 이루어져 있다.

Algorithm 1. Tonelli-Shanks 알고리즘

Input : $\left(\frac{a}{q}\right)=1$를 만족하는 $a\in F_q$
Output : $x^2=a$를 만족하는 x

1. $\left(\frac{u}{q}\right)=-1$를 만족하는 랜덤한 $u\in F_q$ 선택
2. 홀수 t 에 대해, $q-1=2^s\cdot t$를 만족하는 s 찾기
3. $v\leftarrow s$, $z\leftarrow u^t$, $w\leftarrow a^{\frac{t-1}{2}}$, $x\leftarrow a\cdot w$, and $b\leftarrow x\cdot w$.
4. **While** $b\neq 1$ **do**
5. $b^{2^m}=1$를 만족하는 가장 작은 양의 정수 m 찾기
6. $w\leftarrow z^{2^{v-m-1}}$, $z\leftarrow w^2$, $b\leftarrow b\cdot z$, $x\leftarrow x\cdot w$, $v\leftarrow m$
7. **end while**
8. **Return** x

- 단계 3: 평균 $[3\lfloor\log_2(q)\rfloor-3s+1]\cdot M_{F_q}$ (6절 참조, [11]).
- While 루프: 평균 $[(s^2+7s-12)/4+1/(2^{s-1})]\cdot M_{F_q}$ (Lemma 1 참조, [11]).

While 루프의 연산량의 경우, q가 $2^s\cdot t+1$의 형태로 표현된 경우에 대해서의 평균값이다. 따라서 단계 1과 2를 제외한 Algorithm 1에서 필요로 하는 전체 연산량은 다음과 같다.

$$\left[\frac{(s^2-5s-8)}{4}+3\lfloor\log_{2(q)}\rfloor+\frac{1}{2^{s-1}}\right]\cdot M_{F_q}. \qquad (1)$$

2.3 Finding Square Roots when $p\equiv 3\pmod 4$

본 절은 표수가 $p\equiv 3\pmod 4$을 만족할 경우, 효율적으로 제곱근을 구하는 알고리즘을 소개한다. $p\equiv 3\pmod 4$와 k이 홀수라는 조건으로부터 $p\equiv 3\pmod 4$이 쉽게 얻어지고, $q\equiv 3\pmod 4$으로 부터 Algorithm 1의 단계 2의 s 값과 단계 3의 b의 값이 모두 1이 된다. 따라서 단계 4 이하의 While 루프를 거치지 않고, 단계 3에서의 $x:=a^{\frac{t+1}{2}}$ 이 원하는 a의 제곱근이 된다. 즉, p이 위의 조건을 만족할 경우, 제곱근을 구하는 Algorithm 1은 단 한번의 지수승으로 단순화 되어질 수 있다.

square-and-multiply algorithm을 이용해 지수승 $a^{(q+1)/4}$을 계산할 경우, 전체 연산량은 $\lfloor\log_2(q+1)/4)\rfloor+HW((q+1)/4)-1)\cdot M_{F_q}$이다.

$\lfloor\log_2(q+1)/4)\rfloor\approx\lfloor\log_2(q)\rfloor-2$,
$HW(q+1)/4)\approx(\lfloor\log_2(q)\rfloor-1)/2$로부터 Algorithm 2에서 필요로 하는 전체 연산량은 평균 다음과 같다.

Algorithm 2. Tonelli-Shanks 알고리즘 ($p\equiv 3\pmod 4$를 만족할 경우)

Input : $\left(\frac{a}{q}\right)=1$ 를 만족하는 $a\in F_q$
Output : $x^2=a$ 를 만족하는 x

1. $x\leftarrow a^{\frac{q+1}{4}}$
2. **Return** x

$$[\frac{3}{2}\lfloor\log_2(q)\rfloor-\frac{7}{2}]\cdot M_{F_q}. \qquad (2)$$

2.4 Atkin Algorithm when $p\equiv 5\pmod 8$

앞서 소개된 알고리즘과 유사하게 $p\equiv 5\pmod 8$의 경우도 Algorithm 1보다 단순하게 제곱근을 구할 수 있다. 1992년 Atkin [1]에 의해서 제안되었으며, 2가 F_q에서 제곱근을 가지지 않는다는 것을 활용하였다. 필요한 연산량은 확장체 F_q에서 한번의 지수승과 4번의 곱셈량이 필요하다.

Algorithm 3. Atkin 알고리즘 ($p\equiv 5\pmod 8$)

Input : $\left(\frac{a}{q}\right)=1$를 만족하는 $a\in F_q$
Output : $x^2=a$를 만족하는 x

1. $b\leftarrow(2a)^{\frac{q-5}{8}}$
2. $i\leftarrow 2ab^2$
3. $x\leftarrow ab(i-1)$
4. **Return** x

Algorithm 3에서 필요로 하는 전체 연산량은 평균 다음과 같다.

$$[\frac{3}{2}\lfloor\log_2(q)\rfloor-1]\cdot M_{F_q}. \qquad (3)$$

2.5 Modified Atkin Algorithm when $p\equiv 9\pmod{16}$

지금까지 살펴 본 것처럼, $p\equiv 3\pmod 4$와 $p\equiv 5\pmod 8$의 경우는 Algorithm 1보다 간단하게 제곱근을 계산할 수 있었다. 하지만, $p\equiv 1\pmod 8$의 경우는 Algorithm 1을 활용하던지, 아니면 Cipolla-Lehmer

[4]를 활용해야 해서 앞의 경우들 보다 훨씬 연산량이 많이 필요로 한다. 최근 $p \equiv 1 \pmod 8$을 만족하는 p의 경우의 반에 해당하는 $p \equiv 9 \pmod{16}$의 경우에 제곱근을 기존 방법들 보다 효율적으로 계산하는 방법들이 제안되었다. 먼저, Muller 에 의해서 두번의 지수승으로 제곱근을 구하는 방법이 제안되었고 [9], Muller의 알고리즘을 향상시키는 방법이 2006년 Kong 등에 의해서 제안되었다[7]. 본 논문에서는 [7]에 제안된 알고리즘을 소개한다.

Algorithm 4. 향상된 Atkin 알고리즘($p \equiv 9 \pmod{16}$)

Input : $\left(\frac{a}{q}\right) = 1$를 만족하는 $a \in F_q$
Output : $x^2 = a$를 만족하는 x

1. $b \leftarrow (2a)^{\frac{q-9}{16}}$
2. $i \leftarrow 2ab^2$
3. $r \leftarrow i^2$
4. If $r = -1$ then $x \leftarrow ab(i-1)$
5. Else
   5.1 $\left(\frac{d}{q}\right) = -1$를 만족하는 랜덤수 d를 선택
   5.2 $u \leftarrow bd^{\frac{q-9}{8}}$
   5.3 $i \leftarrow 2u^2d^2a$
5.4 $x \leftarrow uda(i-1)$
8. **Return** x

Kong 등에 의해 제안된 Algorithm 4를 좀더 자세하게 살펴보면, 단계 3의 r은 $r^2 = 1$을 만족하는 값이므로, 단계 5이하로 넘어갈 확률은 1/2이 된다. 따라서, 평균적으로 1.5번 (단계 1에서 한번, 단계 5.2에서 1/2번)의 지수승 연산이 필요로 하게 된다.

Algorithm 4에서 필요로 하는 전체 연산량은 평균

$$\frac{9}{4} \lfloor \log_2(q) \rfloor \cdot M_{F_q} \quad (4)$$

(단계 5.1의 d을 선택하는 부분은 q이 고정된 환경에서는 사전에 계산하여 입력값으로 처리 가능함으로 이 부분의 연산량은 생략하였다.}

Ⅲ. 향상된 제곱근을 구하는 알고리즘들

3.1 중심 되는 Idea

확장체 F_q에서 제곱근을 구하는 알고리즘을 4가지로 살펴 보았다. 표수 p의 조건에 따라서

- $p \equiv 1 \pmod{16}$인 경우: Algorithm 1 활용,
- $p \equiv 3 \pmod 4$인 경우: Algorithm 2 활용,
- $p \equiv 5 \pmod 8$인 경우: Algorithm 3 활용,
- $p \equiv 9 \pmod{16}$인 경우: Algorithm 4 활용하여 제곱근을 구할 수 있다.

각 알고리즘들에서 가장 많은 연산량을 요구하는 부분은 공통적으로 확장체 F_q에서의 지수승 부분이다. 예를 들어, Algorithm 1에서는 단계 3에서 u^t 와 $a^{\frac{t-1}{2}}$ 이고, Algorithm 2에서는 단계 1에서 $a^{\frac{q+1}{4}}$ 계산 부분이다. 즉, 제곱근을 구하는 알고리즘을 효율적으로 계산하기 위해서는 지수승 부분의 연산을 효율적으로 계산하는 것이 우선적인 과제이다. 본 논문에서는 각 알고리즘들에 필요로 하는 지수승에 사용되어지는 지수들의 특성을 분석하여 공통적으로 만족되어지는 조건을 유도하고, 얻어진 조건들을 활용해 지수의 새로운 표현법을 제안하고자 한다. 앞서, 간단한 정리와 정의를 살펴본다.

Theorem 3. [Euler 정리] 만약 양의 정수 m과 a이 서로소이면 (즉, $\gcd(m,a) = 1$), $a^{\phi(m)} \equiv 1 (\mathrm{mod} m)$을 만족한다.

여기서, Euler 함수 $\phi(m)$은 m보다 작은 양의 정수들 중에서 m과 서로소가 되는 정수들의 개수를 의미한다.

Definition 1. a와 m은 서로소가 되는 양의 정수라고 하자. 그러면, Theorem 3에 의해서 $a^x \equiv 1 \pmod m$을 만족하는 x이 존재한다. 이런 x들 중에서 가장 작은 정수를 법 (modulo) m에 대한 a의 위수 (order)라고 부른다. 간단하게 이것을 $\mathrm{ord}_m(a)$라고 표현한다.

위의 정리와 제곱근 구하는 알고리즘들에 사용되어지는 지수들의 조건들을 이용하여 지금부터 지수들의 새로운 표현법을 제안하고자 한다.

Assumptions: 주어진 소수 p과 양의 정수 α, β, m에 대해서 다음의 조건 4개를 만족한다.

1. $p \equiv \alpha \pmod \beta$,
2. β가 $p^m - \alpha$를 나눈다,

3. β가 $p^{m-1}-1$를 나눈다,
4. $\gcd(p,\beta)=1$.

Proposition 1. 위의 4개의 조건을 모두 만족 시키는 p,α,β,m에 대해, $\frac{p^m-\alpha}{\beta}$은 다음과 같이 표현되어진다.

$$\frac{p^m-\alpha}{\beta}=A+B\sum_{i=0}^{l-1}(p^w)^i, \tag{5}$$

여기서, $A=\frac{p-\alpha}{\beta}$, $B=p\cdot\frac{p^w-1}{\beta}$, $w=\mathrm{or}\,d_\beta(p)$, $l=\frac{m-1}{w}$이고, A,B,l은 정수이다.

증명.

$w=\mathrm{or}\,d_\beta(p)$ 와 세번째 조건 $\beta\mid(p^{m-1}-1)$로 부터 $w\mid(m-1)$을 만족한다. $w\mid(m-1)$ 조건으로 l을 $l=(m-1)/w$로 둔다. 그러면,

$$\begin{aligned}\frac{p^m-\alpha}{\beta}&=p\cdot\left[\frac{p^{m-1}-1}{\beta}\right]+\frac{p-\alpha}{\beta}\\&=p\cdot\left[\left(\frac{p^{w-1}}{\beta}\right)\cdot\sum_{i=0}^{l-1}(p^w)^i\right]+\frac{p-\alpha}{\beta}\end{aligned}$$

여기서, 두 번째 등호 부분은 $p^{m-1}-1=(p^w)^{l-1}=(p^w-1)(1+p^w+\cdots+(p^w)^{(l-1)})$ 로부터 유도된 것이다.

조건과 증명으로 부터 A,B,l이 정수라는 것은 명확하다.

Corollary 1. 만약 위의 가정에서, $\alpha=1$일 경우,

$$\frac{p^m-1}{\beta}=\frac{p-1}{\beta}\cdot\sum_{i=0}^{m-1}p^i. \tag{6}$$

다음 절들에서는 앞에서 언급한 제곱근을 구하는 알고리즘들에 사용되어지는 지수승의 지수들이 Proposition 1 의 조건 4개를 모두 만족함을 보이고, 식 (5)을 만족하는 명확한 값들을 제시한다. 그리고, 새로운 지수의 표현을 활용한 지수승 연산의 효율성을 언급한다.

3.2 향상된 Tonelli-Shanks 알고리즘

본 절에서는 Algorithm 1의 단계 3에서 계산되는 지수승 u^t와 $a^{(t-1)/2}$의 지수 t와 $(t-1)/2$이 Proposition 1에 적용됨을 보이고, Proposition 1의 결과를 활용해 효율적으로 지수승을 하는 방법을 제안한다.

Lemma 1. 홀수 k에 대해 $q=p^k$을 만족하는 두 수 q, p을 $q=2^s\cdot t+1$ 과 $p=2^{s_1}\cdot t_1+1$ 같이 표현했을 경우, $s=s_1$을 만족한다. 여기서, t, t_1은 홀수이다

증명.

$$\begin{aligned}&q-1=p^k-1\\&=(2^{s_1}t_1+1)^k-1\\&=(2^{s_1}t_1)\cdot p^{k-1}+(2^{s_1}t_1)\cdot p^{k-2}+\cdots+(2^{s_1}t_1)\cdot p+(2^{s_1}t_1)\end{aligned}$$

여기서, t_1, k, 그리고 p가 모두 홀수이므로, $(t_1\cdot p^{k-1}+t_1\cdot p^{k-2}+\cdots+t_1\cdot p+t_1)$도 홀수이다. 따라서, $s_1=s$.

Algorithm 1의 단계 2에서 $q=2^s\cdot t+1$로 표현된다고 가정하면, Lemma 1에 의해 소수 p도 적당한 홀수 x에 대해 $p=2^s\cdot x+1$로 표현된다.

3.2.1 지수승 u^t 에서 지수 t의 새로운 표현

$t=\frac{p^{k-1}}{2^s}$이므로 Proposition 1의 기호로 대응시키면, $\alpha\leftarrow1$, $\beta\leftarrow2^s$, $m\leftarrow k$ 이다. 앞서 살펴본 것처럼, 적당한 홀수 x에 대해 표수 p은 $p=2^s\cdot x+1$으 표현되므로 Proposition 1의 조건 1은 만족된다. 조건 2는 당연하고, 조건 3은 p이 $2^s\cdot x+1$로 표현된다는 것으로 부터 쉽게 유도된다. 그리고, 조건 4는 p이 홀수 소수라는 조건으로부터 당연하다. 따라서, 지수 t의 경우 $\alpha=1$이면서 조건 4개를 모두 만족하므로, Corollary 1의 식 (6)에 적용하면,

$$t=\frac{p^{k-1}}{2^s}=x\cdot\sum_{i=0}^{k-1}p^i. \tag{7}$$

따라서, u^t 지수승 계산은 다음과 같이 두 단계로 계산되어 질 수 있다.

1. $z_1\leftarrow u^x$
2. $z_2\leftarrow(z_1)^{\sum_{i=0}^{k-1}p^i}$

u^t을 계산하는데 필요한 연산량 z_1와 z_2은 각각 $\lfloor\log_2(x)\rfloor-1+HW(x)$와 $\lfloor\log_2(k)\rfloor+HW(k)-1$ 곱

셈이 요구된다. 여기서, 랜덤한 홀수 정수 v에 대해서 평균적으로 $HW(v)=3/2+\frac{\lfloor \log_2(v) \rfloor}{2}$ 이 된다. 왜냐하면, 최상위 비트와 최하위 비트가 모두 1이기 때문이다. 따라서, u^t을 계산하는데 필요한 확장체 F_q에서 평균 곱셈량은 다음과 같다. 참고로, x과 k은 모두 홀수이다.

$$\left[\frac{3}{2}\left(\lfloor \log_2(x) \rfloor + \lfloor \log_2(k) \rfloor\right)+1\right] \cdot M_{F_q}. \quad (8)$$

3.2.2 지수승 $a^{(t-1)/2}$ 에서 지수 $(t-1)/2$ 의 새로운 표현

$(t-1)/2=\frac{p^k-(2^s+1)}{2^{s+1}}$ 이므로 Proposition 1의 기호로 대응시키면, $\alpha \leftarrow -1+2^s$, $\beta \leftarrow 2^{s+1}$, $m \leftarrow k$이다.

Lemma 2. 지수 $(t-1)/2$은 Proposition 1의 조건 4개를 모두 만족한다.

증명.

조건 2와 4는 명백하게 만족을 하고, 조건 1은 $p=2^s \cdot x+1$ 와 $\alpha=1+2^s$ 로부터, $p-\alpha=2^s(x-1)$이다. 여기서, $x-1$은 짝수가 되므로, $2^{s+1} | (p-\alpha)$을 만족한다. 조건 3의 경우, $p^{k-1}-1=2^s(xp^{k-2}+\cdots+xp+x)$로 표현된다. x, p, k이 모두 홀수이므로 $(xp^{k-2}+\cdots+xp+x)$이 짝수가 되고, 따라서 $2^{s+1} | (p^{k-1}-1)$을 만족한다.

Proposition 1의 성립 조건 4개를 모두 만족시키므로, 수식 (5)의 명확한 표현 (explicit formulae)을 유도한다.

Lemma 3. $\mathrm{or}_{d_{2^{s+1}}}(p)$은 2이다.

증명.

홀수 x에 대해 $p=2^s \cdot x+1$이므로, 2^{s+1}은 $p-1$를 나누지 못한다. $p^2-1=2^{s+1}(2^{s-1}x^2+x)$ 로부터, $2^{s+1} | (p^2-1)$을 만족한다.

Collary 2. $\frac{t-1}{2}$은 다음과 같이 표현되어진다.

$$\frac{t-1}{2}=A+B\sum_{i=0}^{(k-3)/2}(p^2)^i,$$

여기서 $A=\frac{x-1}{2}$, $B=px(2^{s-1}x+1)$.

따라서, $a^{(t-1)/2}$ 지수승 계산은 다음과 같이 네 단계로 계산되어 질 수 있다.

1. $z_1 \leftarrow a^A$,
2. $z_2 \leftarrow a^B$,
3. $z_3 \leftarrow (z_2)^{\sum_{i=0}^{(k-3)/2}(p^2)^i}$,
4. $z_4 \leftarrow z_1 . z_3$.

z_1 과 z_2 지수승을 살펴보면, 동일한 밑 a에 대한 지수승 연산임을 알 수 있다. 즉, 지수 A 와 B에 공통 요소가 있으면 반복되는 연산을 줄일 수 있게 된다. 다음의 식은 지수 A과 표수 p을 이용해 지수 B을 전개한 것이다.

$$B=px(2^{s-1}x+1)=p \cdot (A \cdot p+\frac{x+1}{2}+2^{s-1}x). \quad (9)$$

[표 1]은 식 (9)을 이용해 중복되는 연산을 제거하고, 효율적으로 z_1 과 z_2을 계산하는 방법이다.

[표 1] 효율적인 z_1, z_2 계산 과정

Input : a, p, x, s
Output : z_1, z_2
1) $z_1 \leftarrow a^{(x-1)/2}$
2) $y_1 \leftarrow z_1 \cdot a$ $(=a^{(x+1)/2})$
3) $y_2 \leftarrow z_1 \cdot y_1$ $(=a^x)$
4) $y_2 \leftarrow (y_2)^{2^{s-1}}$ $(=a^{x2^{s-1}})$
5) $y_2 \leftarrow y_1 \cdot y_2$ $(=a^{(x+1)/2+x2^{s-1}})$
6) $y_1 \leftarrow z_1^p$ $(=a^{p \cdot (x-1)/2})$
7) $y_1 \leftarrow y_1 \cdot y_2$ $(=a^{A \cdot p+(x+1)/2+2^{s-1}x})$
8) $z_2 \leftarrow y_1^p$ $(=a^B)$

$a^{(t-1)/2}$을 계산하는데 필요한 연산량은 z_1, z_2에 $\frac{3}{2}\lfloor \log_2(x) \rfloor + s+1$, z_3에 $\frac{3}{2}\lfloor \log_2(k) \rfloor - 2$, 그리고, z_4에 한번의 곱셈이 요구된다. 따라서 전체 계산량은 평균 다음과 같다.

$$[\frac{3}{2}(\lfloor \log_2(x) \rfloor + \lfloor \log_2(k) \rfloor)+s] \cdot M_{F_q}. \quad (10)$$

3.2.3 향상된 Tonelli-Shanks 알고리즘의 효율성

향상된 Tonelli-Shanks 알고리즘은 Algorithm 1의 단계 3을 효율적으로 개선한 것이다. 단계 3을 계산하는데 필요한 연산량은 식 (8)과 (10)로 부터

$3(\lfloor \log_2(x) \rfloor + \lfloor \log_2(k) \rfloor) + s + 3$번의 곱셈이다 (단계 3의 x, b계산에 필요한 두번의 곱셈을 포함). 단계 4이하의 While 루프 부분은 바뀐 부분이 없기 때문에, 단계 1과 2를 제외한 향상된 Tonelli-Shanks 알고리즘에서 필요로 하는 전체 곱셈량은 평균 다음과 같다.

$$\left[\frac{1}{4}(s^2+11s)+3(\lfloor \log_2(x) \rfloor + \lfloor \log_2(k) \rfloor)+\frac{1}{2^{s-1}}\right] \cdot M_{F_q}. \quad (11)$$

3.3 $p \equiv 3 \pmod 4$ 조건에서 향상된 Tonelli-Shanks 알고리즘

$p \equiv 3 \pmod 4$이므로 적당한 정수 x에 대해 $p = 4x+3$으로 표현한다. Algorithm 2의 단계 1의 지수승 $a^{(q+1)/4}$ 계산을 Proposition 1을 이용하여 효율적으로 하기 위해 $a^{(q-3)/4}$ 계산을 먼저하고, 그 결과에 a을 곱해서 원하는 결과 $a^{(q+1)/4}$를 얻는 방식을 선택한다. 지금부터 살펴보게 될 지수승 $a^{(q-3)/4}$의 지수 $(q-3)/4$을 Proposition 1의 기호로 대응시키면, $\alpha \leftarrow 3$, $\beta \leftarrow 4$, $m \leftarrow k$이다.

Lemma 4. 지수 $(q-3)/4$은 Proposition 1의 조건 4개를 모두 만족한다.

Lemma 4의 결과로 부터, $\mathrm{ord}_4(p) = 2$이 됨을 알 수 있다. Proposition 1의 결과를 활용한 지수 $(q-3)/4$의 새로운 표현은 다음과 같다.

Corollary 3. $\frac{q-3}{4}$은 다음과 같이 표현되어진다.

$$\frac{q-3}{4} = A + B\sum_{i=0}^{(k-3)/2}(p^2)^i,$$

여기서 $A = x$, $B = p(4x^2+6x+2)$.

따라서, $a^{(q+1)/4}$ 지수승 계산은 다음과 같이 다섯 단계로 계산되어 질 수 있다. $z_1 \leftarrow a^A$, $z_2 \leftarrow a^B$, $z_3 \leftarrow (z_2)^{\sum_{i=0}^{(k-3)/2}(p^2)^i}$, $z_4 \leftarrow z_1 \cdot z_3$, $z_5 \leftarrow z_4 \cdot a$. z_1 과 z_2 지수승을 살펴보면, 동일한 밑 a에 대한 지수승 연산임을 알 수 있다. 즉, 지수 A 와 B에 공통 요소가 있으면 반복되는 연산을 줄일 수 있게 된다. 뿐만 아니라, B의 전개식에서 표수 p이 사용되어 지도록 재전개되어 진다면, p승은 정규기저에서 cyclic shift로 계산되어 지므로 훨씬 효율적으로 계산할 수 있게 된다. 다음의 식은 지수 B의 새로운 표현법이다.

$$B = p(A \cdot p + (3x+2)). \quad (12)$$

[표 1]과 유사하게 식 (12)을 이용해 중복되는 연산을 제거하고, 효율적으로 z_1과 z_2을 전개가 가능하다.

[표 2] 효율적인 z_1, z_2 계산 과정

Input : a, p, x, s
Output : z_1, z_2
1) $z_1 \leftarrow a^x$
2) $y_1 \leftarrow z_1 \cdot a$ $\quad (= a^{x+1})$
3) $y_1 \leftarrow y_1^2$ $\quad (= a^{2x+2})$
4) $y_1 \leftarrow y_1 \cdot z_1$ $\quad (= a^{3x+2})$
5) $y_2 \leftarrow z_1^p$ $\quad (= a^{xp})$
6) $y_1 \leftarrow y_1 \cdot y_2$ $\quad (= a^{xp+(3x+2)})$
7) $z_2 \leftarrow y_1^p$ $\quad (= a^B)$

$a^{(q+1)/4}$을 계산하는데 필요한 연산량은 z_1, z_2에 $\frac{3}{2}\lfloor \log_2(x) \rfloor + \frac{7}{2}$, z_3에 $\frac{3}{2}\lfloor \log_2(k) \rfloor - 2$, 그리고 z_4, z_5에 두번의 곱셈이 요구된다. 따라서 전체 곱셈량은

$$\left[\frac{3}{2}(\lfloor \log_2(x) \rfloor + \lfloor \log_2(k) \rfloor)+\frac{7}{2}\right] \cdot M_{F_q}. \quad (13)$$

3.4 $p \equiv 5 \pmod 8$ 조건에서 향상된 Atkin 알고리즘

표수 p의 조건이 $p \equiv 5 \pmod 8$이므로 적당한 정수 x에 대해 $p = 8x+5$로 표현한다. Algorithm 3의 단계 1의 지수승 $(2a)^{(q-5)/8}$ 계산에 Proposition 1의 적용 여부를 확인하기 위해 먼저 지수 $(q-5)/8$을 Proposition 기호로 대응시키면, $\alpha \leftarrow 5$, $\beta \leftarrow 8$, $m \leftarrow k$이다.

Lemma 5. 지수 $(q-5)/8$은 Proposition 1의 조건 4개를 모두 만족한다.

Lemma 5의 결과로 부터, $\mathrm{ord}_8(p) = 2$이 됨을 알 수 있다. Proposition 1의 결과를 활용한 지수 $(q-5)/8$의 새로운 표현은 다음과 같다.

Corollary 4. $\frac{q-5}{8}$은 다음과 같이 표현되어진다.

$$\frac{q-5}{8} = A + B\sum_{i=0}^{(k-3)/2}(p^2)^i,$$

여기서 $A = x$, $B = p(8x^2+10x+3)$.

따라서, $(2a)^{(q-5)/8}$ 지수승 계산은 다음과 같이 네 단

계로 계산되어 질 수 있다. $z_1 \leftarrow (2a)^A$, $z_2 \leftarrow (2a)^B$, $z_3 \leftarrow (z_2)^{\sum_{i=0}^{(k-3)/2}(p^2)^i}$, $z_4 \leftarrow z_1 . z_3$. z_1 과 z_2 지수승을 살펴보면, 동일한 밑 $2a$에 대한 지수승 연산임을 알 수 있다. 즉, 지수 A 와 B에 공통 요소가 있으면 반복되는 연산을 줄일 수 있게 된다. 뿐만 아니라, B의 전개식에서 표수 p이 사용되어 지도록 재전개되어 진다면, p승은 정규 기저에서 cyclic shift로 계산되어 지므로 훨씬 효율적으로 계산할 수 있게 된다. 다음의 식은 지수 B의 새로운 표현법이다.

$$B = p(A \cdot p + (5x+3)). \tag{14}$$

앞의 절들과 유사하게 식 (14)을 이용해 중복되는 연산을 제거하고, 효율적으로 z_1 과 z_2의 전개가 가능하다.

[표 3] 효율적인 z_1, z_2 계산 과정

Input : $2a$, p, x, s
Output : z_1, z_2
1) $z_1 \leftarrow (2a)^x$
2) $y_1 \leftarrow z_1 \cdot (2a)$ $(=(2a)^{x+1})$
3) $y_2 \leftarrow z_1 \cdot y_1$ $(=(2a)^{2x+1})$
4) $y_2 \leftarrow (y_2)^2$ $(=(2a)^{4x+2})$
5) $y_2 \leftarrow y_1 \cdot y_2$ $(=(2a)^{5x+3})$
6) $y_1 \leftarrow z_1^p$ $(=(2a)^{xp})$
7) $y_1 \leftarrow y_1 \cdot y_2$ $(=(2a)^{xp+5x+3})$
8) $z_2 \leftarrow y_1^p$ $(=(2a)^B)$

$(2a)^{(q-5)/8}$을 계산하는데 필요한 연산량은 z_1, z_2에 $\frac{3}{2}\lfloor \log_2(x) \rfloor + \frac{9}{2}$, z_3에 $\frac{3}{2}\lfloor \log_2(k) \rfloor - 2$, 그리고 z_4에 한번의 곱셈이 요구된다. 따라서 전체 곱셈량의 평균은

$$[\frac{3}{2}(\lfloor \log_2(x) \rfloor + \lfloor \log_2(k) \rfloor) + \frac{15}{2}] \cdot M_{F_q}. \tag{15}$$

3.5 $p \equiv 9 \pmod{16}$ 조건에서 향상된 Modified Atkin 알고리즘

$p \equiv 9 \pmod{16}$이므로 적당한 정수 x에 대해 $p = 16x+9$으로 표현할 수 있다. Algorithm 4에는 단계 1과 5.2에서 각각 지수승 연산 $(2a)^{(q-9)/16}$과 $d^{(q-9)/8}$이 필요하다. 본 논문에서는 지수 $(q-9)/16$만 Proposition 1의 새로운 지수 표현법을 적용하여 효율적인 지수승 계산 방법을 제안한다. $d^{(q-9)/8}$의 경우, $(d^{(q-9)/16})^2$과 같은 방법으로 계산이 가능하므로, 지수 $(q-9)/16$의 효율적인 표현법이 두 번째 지수승에도 적용가능하게 된다. 지수 $(q-9)/16$을 Proposition 1의 기호로 대응시키면, $\alpha \leftarrow 9$, $\beta \leftarrow 16$, $m \leftarrow k$이다.

Lemma 6. 지수 $(q-9)/16$은 Proposition 1의 조건 4개를 모두 만족한다.

Lemma 6의 결과로 부터, $\mathrm{or}_{d_{16}}(p) = 2$이 됨을 알 수 있다. Proposition 1의 결과를 활용한 지수 $(q-9)/16$의 새로운 표현은 다음과 같다.

Corollary 5. $\frac{q-9}{16}$은 다음과 같이 표현되어진다.

$$\frac{q-9}{16} = A + B\sum_{i=0}^{(k-3)/2}(p^2)^i,$$

여기서 $A = x$, $B = p(16x^2+18x+5)$.

따라서, $(2a)^{(q-9)/16}$ 지수승 계산은 다음과 같이 네 단계로 계산되어 질 수 있다. $z_1 \leftarrow (2a)^A$, $z_2 \leftarrow (2a)^B$, $z_3 \leftarrow (z_2)^{\sum_{i=0}^{(k-3)/2}(p^2)^i}$, $z_4 \leftarrow z_1 . z_3$. z_1 과 z_2 지수승을 살펴보면, 동일한 밑 $2a$에 대한 지수승 연산임을 알 수 있다. 즉, 지수 A 와 B에 공통 요소가 있으면 반복되는 연산을 줄일 수 있게 된다. 뿐만 아니라, B의 전개식에서 표수 p이 사용되어 지도록 재전개되어 진다면, p승은 정규 기저에서 cyclic shift로 계산되어지므로 훨씬 효율적으로 계산할 수 있게 된다. 다음의 식은 지수 B의 새로운 표현법이다.

$$B = p(A \cdot p + (9x+5)). \tag{16}$$

앞의 절들과 유사하게 식 (16)을 이용해 중복되는 연산을 제거하고, 효율적으로 z_1 과 z_2의 전개가 가능하다. $(2a)^{(q-9)/16}$을 계산하는데 필요한 연산량은 z_1, z_2에 $\frac{3}{2}\lfloor \log_2(x) \rfloor + \frac{11}{2}$, z_3에 $\frac{3}{2}\lfloor \log_2(k) \rfloor - 2$, 그리고 z_4에 한번의 곱셈이 요구된다. 따라서 총 $(\frac{3}{2}(\lfloor \log_2(x) \rfloor + \lfloor \log_2(k) \rfloor) + \frac{9}{2}) \cdot M_{F_q}$. 2.5절에서 언급했듯이, Algorithm 4의 단계 5 이하는 확률 1/2로 발생한다. 따라서,

[표 4] 효율적인 z_1, z_2 계산 과정

Input : $2a$, p, x, s
Output : z_1, z_2
1) $z_1 \leftarrow (2a)^x$
2) $y_1 \leftarrow z_1 \cdot (2a)$ $(=(2a)^{x+1})$
3) $y_2 \leftarrow z_1 \cdot y_1$ $(=(2a)^{2x+1})$
4) $y_2 \leftarrow (y_2)^{2^2}$ $(=(2a)^{8x+4})$
5) $y_2 \leftarrow y_1 \cdot y_2$ $(=(2a)^{9x+5})$
6) $y_1 \leftarrow z_1^p$ $(=(2a)^{xp})$
7) $y_1 \leftarrow y_1 \cdot y_2$ $(=(2a)^{xp+9x+5})$
8) $z_2 \leftarrow y_1^p$ $(=(2a)^B)$

Algorithm 4에서 필요로 하는 전체 연산량은 평균

$$[\frac{9}{4}(\lfloor \log_2(x) \rfloor + \lfloor \log_2(k) \rfloor) + \frac{65}{4}] \cdot M_{F_q}. \quad (17)$$

Ⅳ. 기존의 알고리즘들과의 비교

본 절에서는 2 절에서 소개한 제곱근을 구하는 각 알고리즘들을 효율적으로 개선하기 위해 기존에 제안된 방법들을 소개하고, 제안한 알고리즘들과 효율성을 비교한다.

4.1 기존의 알고리즘들

Crypto 2002에서 Barreto 등은 확장체 F_q에서 표수 p이 $p \equiv 3 \pmod 4$ 또는 $p \equiv 5 \pmod 8$인 경우 Algorithm 2와 3을 효율적으로 계산하는 방법을 제안했다 [2]. 예를 들어, $p \equiv 3 \pmod 4$인 경우, Algorithm 2에서의 지수승 $a^{(q+1)/4}$의 계산을 위해 지수 $(q+1)/4$을 다음과 같이 전개하였다.

$$\frac{q+1}{4} = \frac{p+1}{4}[p(p-1)\sum_{i=0}^{(k-3)/2}(p^2)^i + 1]. \quad (18)$$

식 (18)을 이용해 $a^{(q+1)/4}$을 계산하려면, Theorem 2을 이용한 지수승 계산과, $(p-1)$ 과 $(p+1)/4$에 대한 지수승 계산, 그리고 한번의 곱셈이 필요하다. p 승 연산은 정규기저를 사용할 경우 cyclic shift로 처리 가능하다. 따라서, 전체 연산량은

$$[\frac{3}{2}(2 \cdot \lfloor \log_2(p) \rfloor + \lfloor \log_2(k) \rfloor) - 3] \cdot M_{F_q}. \quad (19)$$

최근, Kong 등에 의해 $p \equiv 9 \pmod{16}$인 경우, 즉 Algorithm 4에서 지수승 $(2a)^{(q-9)/16}$ 계산을 효율적으로 계산하는 방법이 제안되었다 [7]. Kong 등도 Theorem 2의 특성을 활용하기 위해 지수 $(q-9)/16$의 새로운 표현법을 제안하였다.

$$\frac{q-9}{16} = p^{k-1} \cdot (\frac{p-9}{16}) + 9 \cdot (\frac{p^2-1}{16}) \cdot \sum_{i=0}^{(k-3)/2}(p^2)^i \quad (20)$$

식 (20)을 이용해 $(2a)^{(q-9)/16}$을 계산하려면, Theorem 2을 이용한 지수승 계산과, $(p-9)/16$ 과 $9(p^2-1)/16$에 대한 지수승 계산, 그리고 한번의 곱셈이 필요하다. p 승 연산은 정규기저를 사용할 경우 cyclic shift로 처리 가능하다. 따라서, $(2a)^{(q-9)/16}$을 계산하는데 필요한 연산량은

$$[\frac{3}{2}(\lfloor \log_2(x) \rfloor + \lfloor \log_2(\frac{9(p^2-1)}{16}) \rfloor + \lfloor \log_2(k) \rfloor) - 2] \cdot M_{F_q} \quad (21)$$

여기서, $x = (p-9)/16$이다.

Algorithm 4에서 식 (4)을 유도하던 방법과 유사하게 식 (21)을 이용해 Algorithm 4의 전체 연산량을 계산하면

$$[\frac{9}{4}(\lfloor \log_2(x) \rfloor + \lfloor \log_2(9(p^2-1)/16) \rfloor + \lfloor \log_2(k) \rfloor) + \frac{13}{2}] \cdot M_{F_q} \quad (22)$$

4.2 $p \equiv 1 \pmod{16}$인 경우

확장체 F_q의 표수가 $p \equiv 1 \pmod{16}$을 만족할 경우 제곱근을 가지는 $a \in F_q$에 대해 $x^2 = a$을 만족하는 제곱근 x을 구하는 것은 Algorithm 1을 이용해 계산할 수 있다. 지금까지 알려진 바로는, 표수가 $p \equiv 1 \pmod{16}$을 만족할 때 Algorithm 1을 효율적으로 개선한 결과는 본 논문에서 제안한 방법이 처음이다. 비록, 3.2 절에서 Algorithm 1을 효율적으로 개선한 방법들은 표수의 조건에 의존하지는 않지만, 표수가 $p \equiv 1 \pmod{16}$이 아닌 경우는 더 효율적인 방법들이 존재하기 때문에 본 절에서는 표수의 조건을 $p \equiv 1 \pmod{16}$으로 제한한 환경에서 기존 Algorithm 1의 효율성과 본 논문에서

제안한 개선된 알고리즘의 효율성을 비교하고자 한다. [표 5]에서 우리는 확장체 F_q의 크기와 확장차수 (extension degree) k의 크기에 따라 식 (1)와 (11)의 값의 평균값을 비교한다. 홀수 확장차수 k의 범위를 3에서 15까지로 하고, $q=p^k$의 사이즈를 100-비트 간격으로 100-비트 부터 400-비트까지 고려한다. 여기서, 향상율 (Improvemen t) 항목은 개선된 알고리즘의 효율성을 나타내는 것으로 기존의 알고리즘보다 어느 정도 효율성이 좋아졌는지를 나타내는 수치이다. 예를 들어, 식 (1)와 (11)의 평균 곱셈량이 각각 A과 B이면 향상율은 (A-B)/A*100으로 계산된 값이다. [표 2]에서 알 수 있듯이, 확장체의 크기 및 k의 값에 관계없이 본 논문에서 제안한 방법이 월등이 효율적임을 알 수 있다. 동일한 확장차수에 대해서는 확장체의 크기가 클수록 효율성이 좋아지는 것으로 확인되었고, 확장체의 크기가 같을 경우에는 확장차수의 크기가 클수록 더 효율적임을 알 수 있다. 확장차수 k은 홀수로 가정하였기 때문에, $k=3$인 경우가 가장 작은 효율성 증대를 나타내지만, 이 경우에도 60% 이상의 효율성이 향상되었다.

4.3 $p\equiv 3 \pmod 4$ 또는 $p\equiv 5 \pmod 8$인 경우

표수의 조건이 $p\equiv 3 \pmod 4$ 또는 $p\equiv 5 \pmod 8$의 경우 Algorithm 2와 3을 이용해 계산할 수 있었고, 이 알고리즘들을 좀더 효율적으로 개선한 기존의 방법을 4.1 절에서 소개하였다. Algorithm 2와 3의 연산량이 비슷하고, 개선된 방법들의 증가된 효율성의 비율도 비슷해서 본 논문에서는 표수의 조건이 $p\equiv 3 \pmod 4$인 경우만 살펴본다. 식 (19)과 (13)로 부터 표수 p이 작고 확장차수 k이 크면 두 식의 값은 거의 비슷한 연산량을 나타낸다. 하지만, [표 6]로부터 $p\equiv 3 \pmod 4$인 경우 최소 26% 이상의 효율성 향상이 나타났고, 특별히, $k=3$인 경우에서는 평균 45%의 효율성 증대가 있었다. [표 6]의 향상율 항목 (Improvement)은 Barreto 등이 제안한 방법과의 비교 결과를 나타내는 것으로, Barreto 알고리즘이 가지는 연산량(식 (19))과 비교해 제한한 방법의 연산량 식(13)이 어느 정도의 효율성 향상을 가져왔는지 보여준다.

[표 5] $p\equiv 1 \pmod{16}$인 경우의 비교

q의 크기 (bit)		100	200	300	400
수식 (1) 의 결과		292	593	893	1192
$k=3$	수식 (11) 결과	104	208	305	404
	향상율 (%)	64.4	65.0	65.8	66.1
$k=5$	수식 (11) 결과	67	129	188	248
	향상율 (%)	77.0	78.3	78.9	79.2
$k=7$	수식 (11) 결과	55	95	137	180
	향상율 (%)	81.5	83.9	84.6	84.9
$k=9$	수식 (11) 결과	44	81	114	148
	향상율 (%)	84.9	86.4	87.3	87.7
$k=11$	수식 (11) 결과	41	68	95	122
	향상율 (%)	86.3	88.5	89.3	89.7
$k=13$	수식 (11) 결과	37	61	80	104
	향상율 (%)	87.3	89.8	91.0	91.3
$k=15$	수식 (11) 결과	32	56	71	92
	향상율 (%)	89.0	90.7	92.1	92.2

4.4 $p\equiv 9 \pmod{16}$인 경우

본 절에서는 표수의 조건이 $p\equiv 9 \pmod{16}$인 경우 Algorithm 4과, Kong 등에 의해 이 알고리즘을 좀더

[표 6] $p\equiv 3 \pmod 4$인 경우의 비교

q의 크기 (bit)		100	200	300	400
수식 (2) 의 결과		144	294	444	594
$k=3$	수식 (19) 결과	94	196	295	394
	수식 (13) 결과	50	101	151	200
	향상율 (%)	46.8	48.5	48.8	49.2
$k=5$	수식 (19) 결과	57	117	177	237
	수식 (13) 결과	32	62	92	122
	향상율 (%)	43.9	47.0	48.0	48.5
$k=7$	수식 (19) 결과	42	84	126	168
	수식 (13) 결과	25	46	67	88
	향상율 (%)	40.5	45.2	46.8	47.6
$k=9$	수식 (19) 결과	31	67	100	133
	수식 (13) 결과	20	38	55	71
	향상율 (%)	35.5	43.3	45.0	46.6
$k=11$	수식 (19) 결과	25	55	82	109
	수식 (13) 결과	17	32	46	59
	향상율 (%)	32.0	41.8	43.9	45.9
$k=13$	수식 (19) 결과	22	46	67	91
	수식 (13) 결과	16	28	38	50
	향상율 (%)	27.3	39.1	43.3	45.1
$k=15$	수식 (19) 결과	19	40	58	79
	수식 (13) 결과	14	25	34	44
	향상율 (%)	26.3	37.5	41.4	44.3

효율적으로 개선한 방법(4.1절 참조)과 본 논문에서 제안한 방법과의 효율성 비교를 살펴본다. [표 7]로부터, Kong등에 의해 제안된 알고리즘의 경우 확장차수 k이 3인 경우는 Algorithm 4의 연산량과 큰 차이가 없음을 알 수 있다. 하지만, 본 논문에서 제안한 방법은 k이 3인 경우에서 대략 60% 이상의 효율성 개선이 있었고, q의 크기가 100 비트 이상의 경우에서는 k의 값에 관계없이 최소 40% 이상의 효율성 증대를 가져왔다.

[표 7] $p \equiv 9 \pmod{16}$인 경우의 비교

q의 크기 (bit)		100	200	300	400
수식 (4) 의 결과		220	445	670	895
$k=3$	수식 (22) 결과	215	442	667	890
	수식 (17) 결과	81	157	232	306
	향상율 (%)	62.3	64.5	65.2	65.6
$k=5$	수식 (22) 결과	129	264	399	534
	수식 (17) 결과	54	99	144	189
	향상율 (%)	58.1	62.5	63.9	64.6
$k=7$	수식 (22) 결과	93	188	285	381
	수식 (17) 결과	43	74	106	137
	향상율 (%)	53.8	60.6	62.8	64.0
$k=9$	수식 (22) 결과	73	150	224	298
	수식 (17) 결과	36	63	88	112
	향상율 (%)	50.7	58.0	60.7	62.4
$k=11$	수식 (22) 결과	64	123	183	244
	수식 (17) 결과	33	54	74	94
	향상율 (%)	48.4	56.1	59.6	61.5
$k=13$	수식 (22) 결과	53	102	154	206
	수식 (17) 결과	28	47	63	81
	향상율 (%)	47.2	53.9	59.1	60.7
$k=15$	수식 (22) 결과	46	89	132	179
	수식 (17) 결과	26	43	56	72
	향상율 (%)	43.5	51.7	57.6	59.8

V. 결 론

본 논문에서는 $q=p^k$, $k>1$, k는 홀수, p는 소수를 만족하는 확장체 F_q에서 효율적으로 제곱근을 구하는 알고리즘을 제안하였다. 특히, $p \equiv 1 \pmod{16}$인 경우에 활용되는 Tonelli-Shan ks 알고리즘을 효율적으로 개선한 것을 본 논문이 처음으로, 60% 이상의 효율성 증대가 있었다. 하지만 본 논문의 제안 기법은 지수 k가 홀수 일 때 만 적용이 가능한 것으로, 앞으로 k가 짝수인 경우에 대한 연구를 진행할 예정이다.

참고문헌

[1] A.O.L. Atkin, “Probabilistic primality testing,” Summary by F. Morain, INRIA Res. Rep. 1779, pp. 159-163, 1992.

[2] P.S.L.M. Barreto, H. Kim, B. Lynn, and M. Scott, “Efficient Algorithms for Pairing-Based Cryptosystems,” CRYPTO 2002, LNCS 2442, pp. 354-368, 2002.

[3] P.S.L.M. Barreto and J.F. Voloch, “Efficient Computation of Roots in Fintie Fields,” Journal of Design, Codes and Cryptography, vol. 39, pp. 275-280, 2006.

[4] R. Crandall and C. Pomerance, “Prime Numbers,” A Computational Perspective, Springer-Verlag, New York, 2001.

[5] T. Itoh and S. Tsujii, “A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases,” Information and Computation, vol. 78, pp. 171-177, 1988.

[6] N. Koblitz, “Elliptic curve cryptosystems,” In Mathematics of Computation, volume 48, pp. 203-209, 1987.

[7] F. Kong, Z. Cai, J. Yu, and D. Li, “Improved generalized Atkin algorithm for computing square roots in finite fields,” Information processing Letters, vol. 98 (1), pp. 1-5, 2006.

[8] V.S. Miller, “Use of elliptic curves in cryptography,” In Advances in Cryptology-CRYPTO' 85, LNCS 218, pp. 417-426, 1986.

[9] S. Muller, “On the computation of square roots in finite fields,” Journal of Design, Codes and Cryptography, vol. 31, pp. 301-312, 2004.

[10] R. Lidl and H. Niederreiter, “Finite Field,” Encyclopedia of Mathematics and Its Applications, vol. 20, Cambridge University Press, Cambridge, 1997.

[11] S. Lindhurst, “An analysis of Shanks's algorithm for computing square roots in finite fields,” CRM Proceedings and Lecture Notes,

vol. 19, pp. 231-242, 1999.

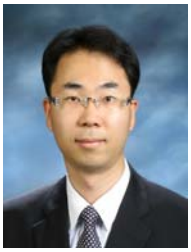
[12] D. Shanks, “Five number-theoretic algorithms,” Proceedings of the second Manitoba conference on numerical mathematics, pp.51-70, 1972.

[13] A. Tonelli, “Bemerkung $\ddot{u}$ber die Aufl$\ddot{o}$sung quadratischer Congruenzen,” G$\ddot{o}$ttinger Nachrichten, pp. 344-346, 1891.

[14] W. Feng, Y. Nogami, Y. Morikawa, “A fast square root computation using the Frobenius mapping,” ICICS 2003, LNCS 2836, pp. 1-10.

[15] M.K. Lee, H. Kim, D. Hong, K. Chung, “Efficient exponentiation in $GF(p^m)$ using the Frobenius map,” ICCSA 2006 (Part 4), LNCS 3983, pp. 584-593.

<著 者 紹 介>



한 동 국 (Dong-Guk Han) 정회원
1999년 : 고려대학교 수학과 졸업(학사)
2002년 : 고려대학교 수학과 석사 (이학석사)
2005년 : 고려대학교 정보보호대학원 박사 (공학박사)
2004년 4월~2005년 4월 : 일본 Kyushu Univ., 방문연구원
2005년 4월~2006년 4월 : 일본 Future Univ.-Hakodate, Post.Doc.
2006년 6월~현재 : 한국전자통신연구원 정보보호연구본부 선임연구원
<관심분야> 공개키 암호시스템 안전성 분석 및 고속 구현, 부채널 분석, RFID/USN 정보 보호 기술

최 두 호 (Dooho Choi) 정회원
1994년 : 성균관대학교 수학과 졸업(학사)
1996년 : 한국과학기술원(KAIST) 수학과 석사(이학석사)
2002년 : 한국과학기술원(KAIST) 수학과 박사(이학박사)
2002년 1월~현재 : 한국전자통신연구원(ETRI) 정보보호연구본부 선임연구원./팀장
<관심분야> RFID/USN 정보보호 기술, 페어링 기반 암호 이론, 암호시스템 안전성 증명, 비가환군 암호 이론

김 호 원 (Ho Won Kim) 종신회원
1993년 2월 : 경북대학교 전자공학과 졸업(학사)
1995년 2월 : 포항공과대학교 전자전기공학과 석사(공학석사)
1999년 2월 : 포항공과대학교 전자전기공학과 박사(공학박사)
1998년 12월~2008년 2월 : 한국전자통신연구원(ETRI) 정보보호연구단 선임연구원/팀장
2008년 3월~현재 : 부산대학교 정보컴퓨터공학부 조교수
<관심분야> RFID/USN 정보보호 기술, 타원곡선 및 초타원곡선 암호 이론, VLSI 설계, embedded system

임 종 인 (Jongin Lim) 종신회원
1980년 2월 : 고려대학교 수학과 학사
1982년 2월 : 고려대학교 수학과 석사
1986년 2월 : 고려대학교 수학과 박사
1999년 2월~현재 : 고려대학교 정보경영공학전문대학원 원장, CIST 센터장
<관심분야> 암호이론, 정보보호정책