논문 12-37A-12-03

한국통신학회논문지 '12-12 Vol.37A No.12
http://dx.doi.org/10.7840/kics.2012.37A.12.1031

유한체상의 제곱근과 세제곱근을 찾는 알고리즘과 그 응용

조 국 화◆, 하 은 혜*, 구 남 훈*, 권 순 학°

Square and Cube Root Algorithms in Finite Field and Their Applications

Gook Hwa Cho◆, Eunhye Ha*, Namhun Koo*, Soonhak Kwon°

요 약

Tonelli-Shanks 알고리즘을 변형한 새로운 알고리즘을 통해 효율적으로 제곱근 및 세제곱근을 찾을 수 있는 방법을 연구하였다. 이 논문에서 소개하는 제곱근을 찾는 알고리즘은 Number Field Sieve에 응용할 수 있다. 큰 합성수를 인수분해하는데 가장 효율적인 알고리즘으로 알려진 Number Field Sieve (NFS)는 법 N에 대하여 공통근을 갖는 두 다항식 선택한 후에, sieving, linear algebra, square root 단계를 차례대로 거친다. NFS의 마지막 단계에서는 수체(Number Field)상에서 제곱근을 구하는 과정이 필요한데 이를 유한체(Finite Field)상으로 내려서 계산한 후 CRT(Chinese Remainder Theorem)을 이용하여 수체 상에서의 제곱근으로 복원하는 과정에서 제안된 알고리즘이 사용될 수 있다.

Key Words : NFS, Tonelli-Shanks algorithm, CRT, Finite Field

ABSTRACT

We study an algorithm that can efficiently find square roots and cube roots by modifying Tonelli-Shanks algorithm, which has an application in Number Field Sieve (NFS). The Number Field Sieve, the fastest known factoring algorithm, is a powerful tool for factoring very large integer. NFS first chooses two polynomials having common root modulo N, and it consists of the following four major steps; 1. Polynomial Selection 2. Sieving 3. Matrix 4. Square Root. The last step of NFS needs the process of square root computation in Number Field, which can be computed via square root algorithm over finite field.

Ⅰ. 서 론

유한체 상에서 square root를 찾는 효율적인 방법 중 하나로 알려져 있는 Tonelli-Shanks 알고리즘은 F_p상에서 square root를 찾는 알고리즘으로써, 확장한 F_{p^k}상에서도 square root를 찾을 수 있다.[4,6,7]

마찬가지 아이디어에 의하면 유한체 상에서 cube root 역시 찾을 수 있다. 또한 이 논문에서 소개하는 제곱근을 찾는 알고리즘은 Number Field Sieve에 응용할 수 있다.

Number Field Sieve (NFS)는 120자리 이상의 큰 수를 인수분해 하는데 가장 효율적인 알고리즘으로 알려져 있다. RSA-768을 포함하여 대부분의 RSA challenge number의 인수분해는 NFS 알고리즘을 이용한 것이다.[5] NFS 알고리즘의 기본 4단계 중 마지막 단계인 square root step은 일반적으로 수체(Number Field)상에서 제곱근을 구하기 위하여 유한체(Finite Field)상으로 내려서 계산한 후

※ 이 논문은 2009년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (과제번호: 2009-0064393)
◆ 주저자 : 성균관대학교 수학과, achimheasal@nate.com, 준회원
° 교신저자 : 성균관대학교 수학과, shkwon@skku.edu, 정회원
* 성균관대학교 수학과, grace.eh.ha@gmail.com, 준회원, komaton@skku.edu, 정회원
논문번호 : KICS2012-10-003, 접수일자 : 2012년 10월 15일, 최종논문접수일자 : 2012년 12월 3일

CRT(Chinese Remainder Theorem)을 이용하여 수체 상에서의 제곱근으로 복원하는 과정에서 사용될 수 있다.[1,-3,8]

본 논문의 나머지 부분은 다음과 같다 : 2장에서는, Tonelli-Shanks가 제시한 제곱근과 세제곱근을 찾는 방법에 대하여 설명한다. 3장에서 Tonelli-Shank 알고리즘을 변형하여 유한체 상에서 제곱근과 세제곱근을 찾는 새로운 알고리즘에 대하여 설명한다. 4장에서는 2장에서 소개한 제곱근을 찾는 알고리즘을 Number Field Sieve에 어떻게 응용할 수 있는지 예제를 통해 설명한다. 5장에서는 위 방법을 기반으로 세제곱근을 찾는 예를 보여준다. 마지막으로 6장에서 복잡도를 분석해보고, 7장에서 결론짓는다.

Ⅱ. Tonelli-Shanks 알고리즘

제곱근을 계산할 수 있는 Tonelli-Shanks 알고리즘에 대해 소개하겠다. 여기서 p는 소수이고, q는 p의 거듭제곱이라 하자. 그리고 F_q는 q개 원소의 유한체 표기법이다. Table 1이 제곱근을 찾는 Tonelli-Shanks 알고리즘이다.[7]

Table 1. The Tonelli-Shanks square root algorithm

Require: A quadratic residue a in F_q with odd characteristic
Ensure: x satisfying $x^2 = a$

Step 1. Compute the integers s,t such that $q-1 = 2^s t$ with t odd
Step 2. Select a quadratic non-residue b in F_q and compute $c = b^t$ and $r = a^t$
Step 3.
$h \leftarrow 1$, $c \leftarrow c^{-1}$
for $i = 1$ to $s-1$
Compute $d = r^{2^{s-i-1}}$
if $d \neq 1$, **then** $h \leftarrow h \cdot c$, $r \leftarrow r \cdot c^2$
$c \leftarrow c^2$
end for
Step 4. Return $a^{\frac{t+1}{2}} \cdot h$

Table 1의 알고리즘은 Step 2에서 이차 비잉여를 필요로 하므로 확률적 아이디어에 입각한 알고리즘이며 이 알고리즘의 복잡도는 $O(v_2(q-1)(\log q)^3)$이다. 여기서 $v_2(q-1)$은 $2^e \mid (q-1)$를 만족하는 최대의 음이 아닌 정수 e를 표시한 것이다. 최악의 복잡도는 $O((\log q)^4)$이긴 하지만, $v_2(q-1)$이 작은 경우에는 향상된 복잡도 $O((\log q)^3)$를 보여줄 때도 있다.

다음은 Adleman, Manders, Miller가 소개한 일반적인 $r-$th 근을 찾는 방법을 이용하여 제곱근을 찾는 Tonelli-Shanks 알고리즘을 확장한 세제곱을 찾는 알고리즘이다.[7]

Table 2. The cube root computation based on the Tonelli-Shanks algorithm

Require: A cubic residue a in F_q
Ensure: r satisfying $r^3 = a$

Step 1. Compute the integers s,t such that $p-1 = 3^s t$ with $t = 3k \pm 1$
Step 2. Select a cubic non-residue b in F_q and compute $c = b^t$ and $r = a^t$
Step 3.
$h \leftarrow 1$, $c' \leftarrow c^{3^{s-1}}$, $c \leftarrow c^{-1}$
for $i = 1$ to $s-1$
Compute $d = r^{3^{s-i-1}}$
if $d = c'$, **then** $h \leftarrow h \cdot c$, $r \leftarrow r \cdot c^3$
else if $d \neq 1$ (equivalently, $d = c'^2$),
then $h \leftarrow h \cdot c^2$, $r \leftarrow r \cdot (c^3)^2$
$c \leftarrow c^3$
end for
Step 4. $r \leftarrow a^k \cdot h$
if $t = 3k+1$, **then** $r \leftarrow r^{-1}$
Return r

위 알고리즘의 복잡도는 $\log q$의 크기뿐 아니라 $v_3(q-1)$의 값에 의존한다. 여기서 $v_3(q-1)$은 $3^{e'} \mid (q-1)$인 최대 음이 아닌 정수 e'를 표시한 것이다. 또한 step 4의 복잡도는 대략 한번의 지수승이고 이때 $(\log q)^3$ 정도의 비트 연산이 필요하고, Tonelli-Shanks 알고리즘의 복잡도는 $O(v_2(q-1)(\log q)^3)$이므로 $q-1$을 나누는 2^s에서 s가 작을수록 step 4의 복잡도가 전체복잡도에서 차지하는 부분이 커지게 된다.

Ⅲ. 새로운 변형 알고리즘 소개

이 절에서는 3장에서 소개한 Tonelli-Shanks 알고리즘을 NFS에서 제곱근을 찾기 위한 알고리즘으로 변형하였다.

여기서 홀수 $q=p^k$이고, 임의의 $\alpha^{\frac{q-1}{2}}=1$이면 α는 quadratic residue이고, $\alpha^{\frac{q-1}{2}}=-1$이면 α는 quadratic non-residue이다.

Table 3은 NFS에서 제곱근을 찾기 위한 변형된 알고리즘이다.

Table 3. An algorithm to find square roots

Require: δ such that $\delta^{\frac{q-1}{2}}=1$
Ensure: ω satisfying $\omega^2=\delta$

1. $q-1=2^s\cdot t$, where t is odd
2. Set $\lambda\leftarrow\delta^t$, $\omega\leftarrow\delta^{\frac{t+1}{2}}$
3. Find ζ such that $\zeta^{\frac{q-1}{2}}=-1$
4. Set $z\leftarrow\zeta^t$
5. **for** $0\le i\le s-1$ **do**
6. **if** $\lambda^{2^{s-i-1}}=-1$ **then**
7. $\lambda\leftarrow\lambda\cdot z^{2^i}$, $\omega\leftarrow\omega\cdot z^{2^{i-1}}$
8. **end if**
9. **end for**
10. **Return** ω

(증명) 먼저 $\delta^{\frac{q-1}{2}}=1$인 δ를 입력한다. line 2에서 $\omega=\delta^{\frac{t+1}{2}}$ 이므로 $\omega^2=\delta^{t+1}=\delta\cdot\delta^t=\delta\cdot\lambda$이다. line 7에서 $\omega_{i+1}=\omega_i\cdot z^{2^{i-1}}$이므로 $\omega_{i+1}^2=\omega_i^2\cdot z^{2^i}=\lambda_i\cdot\delta\cdot z^{2^i}=\lambda_{i+1}\cdot\delta$이다.
그러므로 $\lambda_{i+1}^{2^{s-i-1}}=(\lambda_i z^{2^i})^{2^{s-i-1}}=\lambda_i^{2^{s-i-1}}\cdot z^{2^{s-1}}$이고, line 6에서 $\lambda^{2^{s-i-1}}=-1$이므로 $\lambda_{i+1}^{2^{s-i-1}}=1$이다. $i=s-1$이면, $\lambda_s=1$이므로 $\omega_s^2=\delta$이다.

이와 같은 방법으로 세제곱근도 찾을 수 있다. 마찬가지로 $q=p^k$이고, 임의의 $\alpha^{\frac{q-1}{3}}=1$이면 α는 cubic residue이고, $\alpha^{\frac{q-1}{3}}\neq 1$이면 α는 cubic non-residue 이다. 다음 Table 4는 세제곱근을 찾는 알고리즘이다.

Table 4. An algorithm to find cube roots

Require: δ such that $\delta^{\frac{q-1}{3}}=1$
Ensure: ω satisfying $\omega^3=\delta$

1. $q-1=3^s\cdot t$
2. Set $\lambda\leftarrow\delta^t$, $\omega\leftarrow\delta^{\frac{t+1}{3}}$
3. Find ζ such that $\zeta^{\frac{q-1}{3}}\neq 1$
4. Set $z\leftarrow\zeta^t$
5. **for** $0\le i\le s-1$ **do**
6. **if** $\lambda^{3^{s-i-1}}=z^{3^{s-1}}$ **then**
7. $\lambda\leftarrow\lambda\cdot(z^{3^i})^2$, $\omega\leftarrow\omega\cdot(z^{3^{i-1}})^2$
8. **else if** $\lambda^{3^{s-i-1}}=(z^{3^{s-1}})^2$ **then**
9. $\lambda\leftarrow\lambda\cdot z^{3^i}$, $\omega\leftarrow\omega\cdot z^{3^{i-1}}$
10. **end if**
11. **end for**
12. **Return** ω

(증명) 먼저 $\delta^{\frac{q-1}{3}}=1$인 δ를 입력한다. line 2에서 $\omega=\delta^{\frac{t+1}{3}}$ 이므로 $\omega^3=\delta^{t+1}=\delta\cdot\delta^t=\delta\cdot\lambda$이다. line 7에서 $\omega_{i+1}=\omega_i\cdot(z^{3^{i-1}})^2$이므로 $\omega_{i+1}^3=\omega_i^3\cdot(z^{3^i})^2=\lambda_i\cdot\delta\cdot(z^{3^i})^2=\lambda_{i+1}\cdot\delta$이 다 . 그러므로 $\lambda_{i+1}^{3^{s-i-1}}=(\lambda_i(z^{3^i})^2)^{3^{s-i-1}}=\lambda_i^{3^{s-i-1}}\cdot(z^{3^{s-1}})^2$이 고 , line 6에서 $\lambda^{3^{s-i-1}}=z^{3^{s-1}}$이므로 $\lambda_{i+1}^{3^{s-i-1}}=1$이다. $i=s-1$이면, $\lambda_s=1$이므로 $\omega_s^3=\delta$이다.
만약 line 8에서 $\lambda^{3^{s-i-1}}=(z^{3^{s-1}})^2$인 경우에는 line 9와 같이 $\omega_{i+1}=\omega_i\cdot z^{3^{i-1}}$이고, $\omega_{i+1}^3=\omega_i^3\cdot z^{3^i}=\lambda_i\cdot\delta\cdot z^{3^i}=\lambda_{i+1}\cdot\delta$이므로 $\lambda_{i+1}^{3^{s-i-1}}=(\lambda_i z^{3^i})^{3^{s-i-1}}=\lambda_i^{3^{s-i-1}}\cdot z^{3^{s-1}}$이다.
즉, $\lambda_{i+1}^{3^{s-i-1}}=(z^{3^{s-1}})^2\cdot z^{3^{s-1}}$이다. 마찬가지로 $i=s-1$이면, $\lambda_s=1$이므로 $\omega_s^3=\delta$이다.

마지막 지수승을 생략한 위 알고리즘은 일반적으로 NFS에서 제곱근을 찾거나 유한체에서 세제곱근을 찾을 때 효율적으로 이용될 수 있다.

Ⅳ. 제곱근 알고리즘을 NFS에 응용

4.1. Number Field Sieve algorithm

두 다항식 $f_1(x), f_2(x) \in Z[x]$가 기약 다항식이며 정수 m은 법 N으로부터 두 다항식의 공통근이라 하자. 즉,

$$f_1(m) \equiv f_2(m) \equiv 0 \bmod N$$

이다. α_1과 α_2를 각각 f_1과 f_2의 복소근이라 할 때 환 동형사상을 다음과 같이 정의할 수 있다.

$$\phi_1 : Z[\alpha_1] \to Z_N,\ \phi_2 : Z[\alpha_2] \to Z_N$$
$$\alpha_1 \mapsto m \bmod N,\ \alpha_2 \mapsto m \bmod N$$

다음을 만족하는 서로소 쌍 (a,b)의 집합 S가 존재한다고 가정하자.

$$\prod_{(a,b)\in S}(a+b\alpha_1) = \beta_1^2,\ \beta_1 \in Z[\alpha_1]$$
$$\prod_{(a,b)\in S}(a+b\alpha_2) = \beta_2^2,\ \beta_2 \in Z[\alpha_2]$$

환 동형사상의 성질에 의하면,

$$\phi_1(\beta_1^2) = \prod_{(a,b)\in S}(\phi_1(a+b\alpha_1)) \equiv \prod_{(a,b)\in S}(a+bm) \bmod N$$
$$\phi_2(\beta_2^2) = \prod_{(a,b)\in S}(\phi_2(a+b\alpha_2)) \equiv \prod_{(a,b)\in S}(a+bm) \bmod N$$

이고, 다음을 얻을 수 있다.

$$\phi_1(\beta_1)^2 \equiv \phi_2(\beta_2)^2 \bmod N$$

따라서 N이 합성수라는 가정 하에 최대공약수 $(\phi_1(\beta_1) \pm \phi_2(\beta_2), N)$는 N의 자명하지 않는 인수가 될 확률이 $\frac{1}{2}$이상이다.

집합 S를 형성하는 방법은 매끄러운(smooth) 다항식 값과 관계가 있으며 f_1과 f_2는 다음 이항 동차다항식과 관련이 있다. 즉,

$$F_1(x,y) = y^d f_1(x/y),\ F_2(x,y) = y^d f_2(x/y).$$

집합 S는 다항식 F_1, F_2의 매끄러운 값을 모으는 것으로 형성된다. 특히, $F_1(a,b)$와 $F_2(a,b)$가 어떤 매끄러운 유계 B로부터 $B-smooth$가 되는 서로소인 쌍 (a,b)를 모을 수 있다. 이때, (a,b) 쌍을 *relation*이라고 부른다. 사실 $F_1(a,b)$ 또는 $F_2(a,b)$가 '*almost*' 매끄럽더라도 충분하다. 충분히 많은 (a,b) 쌍의 정보를 얻으면 $\prod_{(a,b)\in S} F_1(a,b)$, $\prod_{(a,b)\in S} F_2(a,b)$가 각각 Z에서 제곱이 되도록 만드는 집합 S를 선형대수의 아이디어를 이용하여 찾을 수 있다.

실제로, sieving stage는 *relation*을 확인하여 $F_1(a,b)$, $F_2(a,b)$ 모두 $B-smooth$ 가 되는 (a,b)쌍을 모으는 과정이다. 이때 많은 매끄러운 값들이 요구되지만 실제로 매끄러운 값이 많지 않기 때문에, 이 단계에서 많은 시간이 소요된다. 또한 전체 알고리즘에서도 제일 많은 시간이 소요된다. 좋은 sieving 다항식을 선택하는 것이 N을 인수분해하는 시간을 줄여주는 중요한 이유이다. 또한 좋은 다항식은 F로부터 생성되는 매끄러운 값의 증가시켜준다.

sieving 단계가 끝나고, 큰 희박한 행렬을 법 2로 축소하여 집합 S를 찾을 수 있다. 이 단계는 sieving 단계만큼 많은 시간이 걸리지 않지만 차수가 매우 큰 행렬에서는 Gauss소거법 등을 사용할 수가 없으므로 Block Lanczos method등의 방법이 필요하다.

4.2. NFS에서 제곱근 찾는 방법

일반적으로 square root 단계에서 제곱근을 찾는 방법은 다음과 같다. f가 법 p로부터 기약다항식이라 하고, $g = x^p - x \pmod f$라 하자. 그러면 f가 기약 다항식이므로 $\gcd(g,f) = 1$이다.

다음 quadratic non-residue를 찾아야 한다.

$\zeta^{\frac{p^d-1}{2}} = -1 \in \mathrm{F}_{p^d}$ 인 ζ를 찾는다. 그러면 위 3단계를 통해 찾은 (a,b)쌍을 이용하여 다음을 계산한다.

$$\delta_p = f'(x)^2 \cdot \prod(a+bx) \pmod p$$

s가 홀수인 $p^d - 1 = 2^s \cdot t$라면, $\gamma^2 = \delta_p^t$ 인 γ를 찾을 수 있다. 이 때, $\gamma \in \{1, \zeta^t, \zeta^{2t}, \cdots, \zeta^{2^{s-1}t}\}$이다. 그러면 다음과 같이 β_p를 정의하면

$$\beta_p = \delta_p^{\frac{t+1}{2}} \gamma^{-1} \pmod p$$

$\beta_p^2 = \delta_p$임을 알 수 있고 위의 관계식을 만족하는 적당한 γ를 찾는 것이 Tonelli-Shanks 알고리즘의 핵심이다.[3]

각 p에 대해 찾은 β_p를 Chinese Remainder Theorem을 이용하여 x를 찾는다. 우선, $\zeta \equiv x_i \pmod{p_i}$이고, p_i들이 서로소이면 $\zeta \equiv \sum(x_i P_i a_i) \pmod P$이다. 여기서, $P = \prod p_i$이고, $P_i = P/p_i$, $a_i \equiv P_i^{-1} \pmod{p_i}$이다.

마지막으로 $x \equiv \sum_{i=1}^{d} x_i P_i a_i \pmod n - rP \pmod n$을 계산하여 x를 찾을 수 있다. 여기서 $r = \frac{\zeta}{P} = \sum \frac{a_i x_i}{p_i}$이다.[2]

4.3. NFS의 4단계

Number Field Sieve는 다음 4단계[2]를 포함하고 있다.

4.3.1. Polynomial selection step

많은 매끄러운 값을 생성할 수 있고, 법 N으로부터 공통근을 갖으며 계수가 작은 다항식 f_1과 f_2를 선택한다. 이 때 두 다항식 f_1, f_2의 resultant $Res(f_1,f_2)$가 작은 값을 가지도록 계수들을 선택하여야 한다.

4.3.2. Sieving step

relation을 모은다. 즉, $F_1(a,b)$와 $F_2(a,b)$가 어떤 유계 B로부터 둘 다 $B-$smooth가 되는 서로소인 (a,b)를 찾는다.

4.3.3. Matrix step

큰 희박한 행렬을 법 2로부터 축소하여 집합 S를 찾는다.

4.3.4. Square root step

주어진 $\prod_{(a,b)\in S}(a+b\alpha_i)=\beta_i^2$이 되는 β_i를 찾는다. 그리고 $\phi_i(\beta_i)$도 찾아준다. 이때, $i=1,2$이고 α_i는 f_i의 복소근이다.

4단계를 모두 마치고나면, $(\phi_1(\beta_1)\pm\phi_2(\beta_2), N)$의 최대공약수를 계산할 수 있다. 계산된 최대공약수는 N의 인수가 된다.

4.4. NFS에서 제곱근 찾는 예제

우리가 제시한 알고리즘을 이용하여 NFS에서 제곱근을 찾는 예이다.

$$N=45113=229\cdot 197$$

$f=x^3+15x^2+29x+8$이고, $f(m)\equiv 0 \pmod N$인 해 $m=31$이다.

Dependent Pairs (a,b)

$(-2,1)$	$(13,1)$	$(61,1)$	$(33,2)$	$(2,3)$	$(5,3)$
$(19,4)$	$(14,5)$	$(11,7)$	$(119,11$	$(5,17)$	$(35,19)$
$(375,29$	$(9,32)$	$(1,33)$	$(78,37)$	$(5,41)$	$(9,41)$

1. 위 표에서 나타난 (a,b)를 이용하여 다음 g를 계산한다.

$$g=\prod(a+bx)\pmod f = 9728848250924236265780730970 1x^2 + 216002640202516839427136973936x + 609015460249676734336165513 76$$

2. $p_1=1301149,\ p_2=179424673$

$p_1\cdot p_2$ 의 크기와 다항식 g의 계수의 크기가 비슷하도록 임의의 p_1,p_2를 잡는다.

3. $q_1=p_1^3,\ q_2=p_2^3$으로

$q_1-1=2^{s_1}\cdot t_1,\ q_2-1=2^{s_2}\cdot t_2$

$s_1=2,\ t_1=550707645075202737$

$s_2=5,\ t_2=1805080239309637202346 63$

4. $\delta_1^{\frac{q_1-1}{2}}=1,\ \delta_2^{\frac{q_2-1}{2}}=1$ 을 만족하는,

즉 quadratic residue인

$\delta_1=g\pmod{p_1}=1273599x^2+154892x+602517$

$\delta_2=g\pmod{p_2}=90945823x^2+37640377x+104521782$

을 1에서 찾은 g를 이용하여 찾는다.

5. $\lambda_1=\delta_1^{t_1}=1301148,\ \lambda_2=\delta_2^{t_2}=89053323$

6. $\omega_1=\delta_1^{\frac{t_1+1}{2}}=957036x^2+484516x+692167$

$\omega_2=\delta_2^{\frac{t_2+1}{2}}=100719928x^2+173251485x+119988325$

7. $\zeta_1^{\frac{q_1-1}{2}}=-1,\ \zeta_2^{\frac{q_2-1}{2}}=-1$을 만족하는,

즉 quadratic non-residue인 임의의

$\zeta_1=x+1,\ \zeta_2=x-132$을 선택한다.

8. $z_1=\zeta_1^{t_1}=396072,\ z_2=\zeta_2^{t_2}=77268072$

9. **for** $0\le i\le s_j-1$ **do**

if $\lambda^{2^{s_j-i-1}}=-1$ **then**

$\lambda\leftarrow\lambda\cdot z^{2^i},\ \omega\leftarrow\omega\cdot z^{2^{i-1}}$

end if

end for

10. return $w_1=532465x^2+658589x+1078320$

$\omega_2=136870979x^2+120340270x+142235968$

11. Chinese Remainder Theorem을 이용하여

$\omega=-26790931919974x^2-59513310702967x-16784961545772$

12. $x^2 = (\omega(m))^2 \pmod N = 29661^2$

$y^2 = \prod(a+bm) = \beta^2 \pmod N = 36531^2$

13. $\gcd(x-y, N) = 229$

$(\omega(m))^2 \pmod N$과 $\prod(a+bm)$의 square root를 찾으면 $x^2 = y^2 \pmod N$의 형태이므로 $N = 45113$의 인수는 $\gcd(x \pm y, N)$으로 찾을 수 있다.

Ⅴ. 세제곱근 찾는 예제

$f = x^3 + 15x^2 + 29x + 8$이다.

1. $g = 22028305803008109 34x^2$
$+ 22028305803008109 19x$
$+ 2202830580300810941$

2. $p_1 = 179424673$

3. $q_1 = p_1^3$ 으로

$q_1 - 1 = 3^{s_1} \cdot t_1$

$s_1 = 3,\ t_1 = 81586317788918924$

4. $\delta_1^{\frac{q_1-1}{3}} = 1$ 을 만족하는,

즉 quadratic residue인

$\delta_1 = g \pmod{p_1} = 1301134x^2 + 1301119x + 1301141$

5. $\lambda_1 = \delta_1^{t_1} = 643659$

6. $\omega_1 = \delta_1^{\frac{t_1+1}{3}} = 793093x^2 + 114203x + 561024$

7. $\zeta_1^{\frac{q_1-1}{3}} \neq 1$을 만족하는,

즉 quadratic non-residue인 임의의 $\zeta_1 = x - 5$을 선택한다.

8. $z_1 = \zeta_1^{\ t_1} = 927075x^2 + 760675x + 338713$

9. **for** $0 \le i \le s_1 - 1$ **do**

$\lambda \leftarrow \lambda \cdot (z^{3^i})^2,\ \omega \leftarrow \omega \cdot (z^{3^{i-1}})^2$

else if $\lambda^{3^{s-i-1}} = (z^{3^{s-1}})^2$ **then**

$\lambda \leftarrow \lambda \cdot z^{3^i},\ \omega \leftarrow \omega \cdot z^{3^{i-1}}$

end if

end for

10. return $\omega_1 = 1069433x^2 + 409248x + 270441$

11. $\omega_1^3 \equiv \delta_1$

Ⅵ. 복잡도 분석

Tonelli-Shanks 알고리즘(Table 1,2)과 우리가 제안한 변형 알고리즘(Table 3,4)의 차이점은 Step 4를 거치지 않고 제곱근을 바로 찾을 수 있다는 점이다. Tonelli-Shanks 알고리즘의 복잡도는 최악의 경우 $O((\log q)^4)$이고, 평균적으로 $O((\log q)^3)$이다. s가 커질수록 우리가 제안한 알고리즘의 복잡도도 $O((\log q)^4)$이고, 평균적으로 $O((\log q)^3)$이다. 그러나 Tonelli-Shanks 알고리즘의 Step 4를 보면 마지막 지수승이 필요하고 또한 곱셈량이 k에 의존하게 되어 k가 클수록 곱셈량이 커진다. 그러나 우리가 제안하는 알고리즘에서는 마지막 지수승이 필요없고 k에 의존하지 않기 때문에 우리가 제안한 알고리즘이 효율적이라 할 수 있다.

Ⅶ. 결 론

본 논문에서는 Tonelli-Shanks 알고리즘을 확장, 변형하여 유한체에서의 제곱근 또는 세제곱근을 찾는 방법에 대해 생각해 보았다. 제시된 알고리즘은 F_p뿐만 아니라 일반적인 유한체 F_{p^k} $(k \ge 1)$상에서도 적용가능하며 square root step, cube root step 등 n'th root step과 같은 일반적인 경우로도 확장 가능하다. 소개한 알고리즘을 이용하여 다른 부분에 응용할 수도 있다.

참 고 문 헌

[1] E. Bach, "A note on square roots in finite fields," *IEEE Trans. Inform. Theory* vol. 36, no. 6, pp. 1494-1498, Oct. 1990.

[2] J. P. Buhler, H. W. Lenstra, and C. Pomerance, "Factoring integers with the number field sieve," *Reprinted in The Development of the Number Field Sieve, Lecture Notes in Mathematics 1554*. A.K. Lenstra, HW. Lenstra, Jr., Eds., Jun. 1993

[3] J. Dreibelbis, *Implementing the General Number Field Sieve*, Rochester Institute of Technology, Jun. 2003.

[4] D. G. Han, D. Choi, and H. Kim,

"Improved computation of square roots in specific finite fields," *IEEE Trans.* Comput., vol. 58, no. 02, pp. 188-196, Feb. 2009.

[5] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thome, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann, "Factorization of a 768-bit RSA modulus," in *Proc. IACR Crypto*, pp. 333-350, Aug. 2010.

[6] F. Kong, Z. Cai, J. Yu, and D. Li, "Improved generalized Atkin algorithm for computing square roots in finite fields," *Inform. Process. Lett.*, vol. 98, no. 1, pp. 1-5, April. 2006

[7] N. Nishihara, R. Harasawa, Y. Sueyoshi, and A. Kudo, "A remark on the computation of cube roots in finite fields," *IACR Cryptology ePrint Archive*, Sep. 2009

[8] G. H. Jo, N. Koo, S. Kwon, "Two cubic polynomial selection for the number field sieve," *J. KICS*, vol. 36, no. 10. pp. 614-620, Oct. 2011

조 국 화 (Gook Hwa Cho)

2007년 8월 전북대학교 수학과 학사
2011년 2월 성균관대학교 수학과 석사
2011년 3월~현재 성균관대학교 수학과 박사과정
〈관심분야〉 정수론, 공개키 암호 시스템, NFS

하 은 혜 (Eunhye Ha)

2011년 2월 성균관대학교 수학과 학사
2011년 3월~현재 성균관대학교 수학과 석사과정
〈관심분야〉 공개키 암호 시스템, NFS

구 남 훈 (Namhun Koo)

2007년 8월 성균관대학교 수학과 학사
2009년 2월 성균관대학교 수학과 석사
2009년 3월~현재 성균관대학교 수학과 박사과정
〈관심분야〉 공개키 암호 시스템, NFS

권 순 학 (Soonhak Kwon)

1990년 2월 KAIST 수학과 학사
1997년 5월 Johns Hopkins University 박사
1998년 3월~현재 성균관대학교 수학과, 정교수
〈관심분야〉 정수론, 공개키 암호