논문 13-38A-09-04

한국통신학회논문지 '13-09 Vol.38A No.09
http://dx.doi.org/10.7840/kics.2013.38A.9.759

특정한 유한체 F_q상에서의 제곱근 알고리즘

구 남 훈◆, 조 국 화*, 권 순 학°

Square Root Algorithm in Fq for Special Class of Finite Fields

Namhun Koo◆, Gooc Hwa Jo*, Soonhak Kwon°

요 약

$q \equiv 5 \pmod 8$의 경우에 유한체 F_q상에서 Atkin의 제곱근 알고리즘과 $q \equiv 9 \pmod{16}$의 경우에 Kong의 알고리즘으로부터 일반적인 제곱근 알고리즘을 제안한다. 우리의 알고리즘은 s가 $2^s | q-1$을 만족하는 가장 큰 양의 정수라 할 때, 2^s차 원시근 ξ를 미리 계산하였고 s의 값이 작을 때 적용가능하다. 제시한 알고리즘은 제곱근을 계산하기 위해 한 번의 지수계산이 필요하고, Akin, Müller, Kong의 알고리즘과 비교해보아도 유리하다.

Key Words : square root algorithm; finite field; Tonelli-Shanks algorithm; Cipolla-Lehmer algorithm

ABSTRACT

We present a square root algorithm in F_q which generalizes Atkin's square root algorithm [9] for finite field F_q of q elements where $q = 5 \pmod 8$ and Kong et al.'s algorithm [11] for the case $q \equiv 9 \pmod{16}$. Our algorithm precomputes ξ a primitive 2^s-th root of unity where s is the largest positive integer satisfying $2^s | q-1$, and is applicable for the cases when s is small. The proposed algorithm requires one exponentiation for square root computation and is favorably compared with the algorithms of Atkin, Müller and Kong et al.

Ⅰ. 서 론

유한체 F_q상에서 제곱근을 찾는 문제, 즉 $x^2 = c$의 근을 찾는 문제는 많은 곳에서 응용이 되고 있다. 예를 들어 미 NIST(National Institute of Standards and Technology)제안한 규정집 (Federal Information Processing Standard 186-3)의 전자서명(Digital Signature) 프로토콜에서는 타원곡선을 이용하여 전자서명을 구현하는 방법을 설명하였는데 여기에서 유한체상의 제곱근을 찾는 알고리즘이 필요하다[1].

한체상에서 square root를 찾는 표준적인 두 개의 알고리즘으로는 Tonelli-Shanks 알고리즘[2,3], 그리고 Cipolla-Lehmer 알고리즘[4,5]이 있다. Tonelli-Shanks 알고리즘은 $2^s | q-1$이고 $2^{s+1} | q-1$인 2의 지수 s에 의존하고 최악의 경우 복잡도는 $O(\log^4 q)$[6]이지만 Cipolla-Lehmer의 복잡도는 $O(\log^3 q)$[4,5]이다. 하지만 Cipolla-Lehmer는 큰 확대체에서의 연산이 필요하기 때문에, s가 비교적 작은 경우에는 Tonelli-Shanks 알고리즘이 Cipolla-Lehmer보다 더 효율적이다[6,7]. Tonelli-Shanks의 방법에서는 $q = p^k$일때 $q-1$의 p진 전개식을 이용하여 Tonelli-Shanks 알고리즘에서 요구되는 곱하기의 개수를 줄이는 방법도 제안되었다[8].

한편 s가 상당히 작을 때는, 한번 또는 두 번의

※ 이 논문은 2010년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (과제번호: 2010-0022419)
◆ 주저자 : 성균관대학교 수학과, komaton@skku.edu, 정회원
° 교신저자 : 성균관대학교 수학과, shkwon@skku.edu, 정회원
* 성균관대학교 수학과, achimheasal@nate.com, 준회원
논문번호 : KICS2012-12-569, 접수일자 : 2012년 12월 26일, 최종논문접수일자 : 2013년 8월 13일

지수 계산을 이용하여 제곱근을 찾는 방법들이 있으며 이러한 방법은 Tonelli-Shanks 알고리즘보다 더 빠르게 구현할 수 있다. 예를 들어, 유한체 상에서 c가 제곱 잉여이고, $q \equiv 3 \pmod 4$인 경우 c의 제곱근은 $c^{\frac{q+1}{4}}$으로 찾을 수 있다. 왜냐하면, $c = a^2$을 이용하면 $(c^{\frac{q+1}{4}})^2 = c$임을 확인할 수 있다. s가 2 또는 3일 때, Atkin[9], Müller[10], Kong[11]이 제안한 비슷한 방법이 있다. 그리고 이들의 방법은 Tonelli-Shanks와 Cipolla-Lehmer의 방법보다 좀 더 성능이 좋다.

하지만 이와 같은 '누승법' (exponentiation method) 접근 방법이 F_q 상에서 제곱근을 찾는 문제에 잘 활용되지 않는 것처럼 보인다. 우리의 목표는 제곱근 풀이에 '누승법' 접근 방법을 일반화 시키고 구체적인 예를 보이는 것이다. 우리는 미리 계산된 원시원소를 이용하여 새로운 제곱근을 찾는 방법을 소개하고, F_q 상에서 오직 한 번의 지수계산만을 요구하는 새로운 알고리즘을 보여주겠다.

Ⅱ. 작은 지수 s를 이용하는 제곱근 알고리즘들

2.1. Atkin의 알고리즘

$q \equiv 1 \pmod 4$일 때, 한 번의 지수계산으로 제곱근을 찾는 알고리즘은 알려져 있지 않다. 하지만, $q \equiv 5 \pmod 8$일 때 한 번의 지수계산을 이용하는 Atkin이 제안한 효율적인 제곱근 알고리즘이 있다[9].

Table 1. Atkin's algorithm when $q \equiv 5 \pmod 8$ [9]

Input: A square a in F_q **Output:** x satisfying $x^2 = a$ in F_q
1. $b \leftarrow (2a)^{\frac{q-5}{8}}$
2. $i \leftarrow 2ab^2$
3. $x \leftarrow ab(i-1)$
4. Return x

Table 1의 Atkin의 알고리즘은 $(2a)^{\frac{q-1}{2}} = -1$이고 $(2a)^{\frac{q-1}{4}} = \sqrt{-1} = i$라는 사실을 이용한다. 위 알고리즘은 1번의 지수계산과 4번의 곱셈계산을 필요로 하며 Tonelli-Shanks 또는 Cipolla-Lehmer 알고리즘보다 더욱 효율적이다.

2.2. Müller의 알고리즘

Müller[10]는 $q \equiv 9 \pmod{16}$일 때 Atkin 방법을 확장하였다. $q \equiv 9 \pmod{16}$인 경우에는 $(2a)^{\frac{q-1}{4}} = \pm 1$이고, 만약에 $(2a)^{\frac{q-1}{4}} = 1$이면 이차 비잉여인 d, $(2a)^{\frac{q-1}{4}} = -1$이면 이차 잉여인 d를 필요로 한다. Table 2에서 알 수 있듯이 Müller 알고리즘은 2번의 지수계산(step1, step3)을 사용한다.

Table 2. Müller's algorithm when $q \equiv 9 \pmod{16}$ [10]

Input: A square a in F_q **Output:** x satisfying $x^2 = a$ in F_q
1. $b \leftarrow (2a)^{\frac{q-1}{4}}$
2. Find randomly d satisfying $-b = \eta(d)$
3. $u \leftarrow (2ad^2)^{\frac{q-9}{16}}$
4. $i \leftarrow 2u^2d^2a$
5. $x \leftarrow uda(i-1)$
6. Return x

2.3. Kong 등의 알고리즘

Table 3. Kong et al.'s algorithm when $q \equiv 9 \pmod{16}$ [11]

Input: A square a in F_q **Output:** x satisfying $x^2 = a$ in F_q
1. $b \leftarrow (2a)^{\frac{q-9}{16}}$
2. $i \leftarrow 2ab^2$, $r \leftarrow i^2$
3. if $r = -1$ **then**
4. $x \leftarrow ab(i-1)$
5. else
6. Find a quadratic nonresidue d
7. $u \leftarrow bd^{\frac{q-9}{8}}$
8. $i \leftarrow 2u^2d^2a$
9. $x \leftarrow uda(i-1)$
10. end if
11. Return x

Table 2에서 $\eta(d) := d^{\frac{q-1}{2}} = \pm 1$이며 이 계산은 이차잉여의 법칙을 이용하면 $O(\log^2 q)$만에 계산가능하다. Table 2의 Müller 알고리즘에서 만약 $(2a)^{\frac{q-1}{4}} = -1$이면, Table 3의 Kong 알고리즘[11]의 1-4단계와 같이 한 번의 지수승으로 처리할 수가 있다. 만약 $(2a)^{\frac{q-1}{4}} = 1$이면, Kong 알고리즘의 6-9단계에서는 Müller 알고리즘과 비슷한 방법으로 제곱근을 찾을 수 있다. 한번의 지수승만으로 Kong 알고리즘이 구현될 확률이 $\frac{1}{2}$이고, 또한 두 번의 지수승으로 Kong 알고리즘이 구현될 확률이 $\frac{1}{2}$이므로 Kong 알고리즘은 평균적으로 $1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2} = \frac{3}{2} = 1.5$번의 지수승 연산이 필요하다. 따라서 두 번의 지수승 연산이 필요한 Müller 알고리즘과 비교할 때 $\frac{1.5}{2} = 0.75$이므로 Kong 알고리즘은 Müller 알고리즘에 비하여 25%의 속도 향상을 기대할 수 있다.

Ⅲ. $q \equiv 2^s + 1 \pmod{2^{s+1}}$ 일 때 F_q상에서 새로운 제곱근 공식

q는 홀수인 소수의 거듭제곱이고 s는 $2^s | q-1$인 (즉 2^s가 $q-1$의 약수인) 가장 큰 양의 정수이다. $\frac{q-1}{2^s} \equiv 1 \pmod 2$이므로 $q \equiv 2^s + 1 \pmod{2^{s+1}}$이다. 즉, 임의의 q에 대해서 $q \equiv 2^s + 1 \pmod{2^{s+1}}$을 만족하는 유일한 s가 존재한다. 주어진 F_q 상에서 제곱수 c로부터,

$$b = c^{\frac{q-(2^s+1)}{2^{s+1}}}$$

로 b를 정의한다. 그리고 ζ를 다음과 같이

$$\zeta = c^{\frac{q-1}{2^s}} = c \cdot c^{\frac{q-(2^s+1)}{2^s}} = c\left\{c^{\frac{q-(2^s+1)}{2^{s+1}}}\right\}^2 = cb^2$$

정의한다. c가 F_q 상에서 제곱이므로, ζ는 $t \le s-1$에서 유일하게 결정되는 적당한 t에 대하여 2^t차 원시근이다. 즉,

$$\zeta^{2^t} = 1,\ \zeta^{2^{t-1}} = -1$$

을 만족하는 $t < s$가 존재한다. 또한 ξ는 이 논문에서 미리 계산되어서 쓰이는 F_q 상에서 2^s차 원시근이라 하자.

$\xi^{2^{s-t}}$는 단위원 2^t차 원시근이기 때문에,

$$\xi^{2^{s-t}} = \zeta^i,\ (\xi^{2^{s-t}})^j = \zeta$$

를 만족하는 $i, j \pmod{2^t}$가 유일하게 존재한다.

$\xi^{2^{s-t}} = \zeta^i = (\xi^{2^{s-t}})^{ij}$ 로부터,

$$ij \equiv 1 \pmod{2^t}$$

이다. 적당한 조건아래 한 번의 지수계산만 이용하여 제곱근을 찾는 새로운 Theorem 1을 소개하겠다.

Theorem 1. $u \equiv j(2^t - 1)2^{s-t-1} \pmod{2^{s-1}}$ 일 때, c의 제곱근은 $cb\xi^u$이다.

(증명) $x = cb\xi^u$이면,

$$x^2 = c \cdot cb^2\xi^{2u} = c \cdot \zeta \cdot \xi^{2u}$$

이다. 적당한 정수 k에 대해서 $u = j(2^t-1)2^{s-t-1} + 2^{s-1}k$로 쓸 수 있고 $\xi^{2^s} = 1$이므로,

$$\begin{aligned}\zeta\xi^{2u} &= (\xi^{2^{s-t}})^j \cdot \xi^{2u} = \xi^{j2^{s-t}+2u} \\ &= \xi^{j2^{s-t}+j(2^t-1)2^{s-t}+2^sk} \\ &= \xi^{j2^{s-t}+j(2^t-1)2^{s-t}} = \xi^{j2^{s-t}(1+2^t-1)} \\ &= \xi^{j2^s} = 1\end{aligned}$$

이다. 그러므로 $x^2 = c \cdot \zeta\xi^{2u} = c$이다.

예 1. $q \equiv 3 \pmod 4$일 때 제곱근:

$s = 1$인 경우이다. 그러므로 $\zeta = c^{\frac{q-1}{2}} = 1$이고 $t = 0$이다. 또한 $\xi = -1$이고 $u = 0$이다. 그러므로 c의 제곱근은 $cb = c \cdot c^{\frac{q-3}{4}} = c^{\frac{q+1}{4}}$이다.

예 2. $q \equiv 5 \pmod 8$일 때 제곱근:

$s = 2$인 경우이다. 그러므로 $\zeta = c^{\frac{q-1}{4}} = \pm 1$이고 $t = 0$ 또는 1이다. 또한 ξ는 $\xi^{2^{2-t}} = \zeta^i$를 만족하는 2^2차 원시근이다. c의 제곱근은 $cb\xi^u$이고 여기서 $u \equiv j(2^t-1)2^{1-t} \pmod 2$이다. $t = 0$이면 $u = 0$이고 $x = cb$가 제곱근이다. $t = 1$이면 $i = j = 1$이고 $u = 1$이므로 $x = cb\xi$가 제곱근이다.

예 3. $q \equiv 9 \pmod{16}$일 때 제곱근:

$s = 3$인 경우이다. 그러므로 $\zeta = c^{\frac{q-1}{2^3}}$은

$t=0,1,2$인 적당한 t에 대하여 위수가 2^t이다. 또한 ξ는 $\xi^{2^{3-t}}=\zeta^i$를 만족하는 2^3차 원시근이다. 이 때 c의 제곱근은 다음과 같이 구한다.

1) $t=0$이면, $u=0$이고 $x=cb$가 제곱근이다.

2) $t=1$이면, $i=j=1$이고 $u\equiv 2j\equiv 2\pmod{2^2}$이므로 $x=cb\xi^2$이 제곱근이다.

3) $t=2$이면, $\xi^2=\zeta^i$로부터 $(i,j)=(1,1),(3,3)$이고 $u\equiv 3j\pmod{2^2}$이므로 $x=cb\xi^{3j}$가 제곱근이다. 즉,

(1) $\xi^2=\zeta$이면, $x=cb\xi^3$가 제곱근이다.

(2) $\xi^2=\zeta^3$이면, $x=cb\xi$가 제곱근이다.

예 4. $q\equiv 17\pmod{32}$일 때 제곱근:

$s=4$인 경우이다. 그러므로 $\zeta=c^{\frac{q-1}{2^4}}$는 $t=0,1,2,3$인 적당한 t에 대하여 위수가 2^t이다. 또한 ξ는 $\xi^{2^{4-t}}=\zeta^i$를 만족하는 2^4차 원시근이다. 그러므로 c의 제곱근은 다음과 같이 구한다.

1) $t=0$이면, $u=0$이고 $x=cb$가 제곱근이다.

2) $t=1$이면, $\xi^8=\zeta$로부터 $i=j=1$이고 $u\equiv 4j\equiv 4\pmod{2^3}$이므로 $x=cb\xi^4$가 제곱근이다.

3) $t=2$이면, $\xi^4=\zeta^i$로부터 $(i,j)=(1,1),(3,3)\pmod{2^2}$이고 $u\equiv 6j\pmod{2^3}$이므로 $x=cb\xi^{6j}$가 제곱근이다. 즉,

(1) $\xi^4=\zeta$이면, $x=cb\xi^6$가 제곱근이다.

(2) $\xi^4=\zeta^3$이면, $x=cb\xi^2$가 제곱근이다.

4) $t=3$이면, $\xi^2=\zeta^i$로부터 $(i,j)=(1,1),(3,3),(5,5),(7,7)\pmod{2^3}$이고 $u\equiv 7j\pmod{2^3}$이므로 $x=cb\xi^{7j}$가 제곱근이다. 즉,

(1) $\xi^2=\zeta$이면, $x=cb\xi^7$가 제곱근이다.

(2) $\xi^2=\zeta^3$이면, $x=cb\xi^5$가 제곱근이다.

(3) $\xi^2=\zeta^5$이면, $x=cb\xi^3$가 제곱근이다.

(4) $\xi^2=\zeta^7$이면, $x=cb\xi$가 제곱근이다.

위 예들은 Table 4,5,6에서 제안하는 제곱근 알고리즘으로 표현될 수 있다. 모든 알고리즘은 주어진 q에 대하여 미리 계산된 2^s차 원시근 ξ가 필요하며 제안된 알고리즘은 오직 한 번의 지수계산이 필요하다. (즉, 1단계에서 b를 계산할 때) 한편, Müller 알고리즘은 2번의 지수계산이 필요하다. (1,3단계에서) 그리고 Müller 알고리즘은 미리 계산하는 것(precomputation)이 가능하지 않다. 또한 Kong 알고리즘은 평균적으로 1.5개의 지수계산이 필요하며 역시 precomputation이 가능하지 않다.

Table 4. $q\equiv 5\pmod 8$일 때 제안된 제곱근 알고리즘

Input: A square c in F_q
Output: x satisfying $x^2=c$ in F_q

1. $b\leftarrow c^{\frac{q-5}{8}}$
2. $\zeta\leftarrow cb^2$
3. **if** $\zeta=1$, **then** $x\leftarrow cb$
4. **else then** $x\leftarrow cb\xi$
5. **Return** x

Table 5. $q\equiv 9\pmod{16}$일 때 제안된 제곱근 알고리즘

Input: A square c in F_q
Output: x satisfying $x^2=c$ in F_q

1. $b\leftarrow c^{\frac{q-9}{16}}$
2. $\zeta\leftarrow cb^2$
3. **if** $\zeta=1$, **then** $x\leftarrow cb$
4. **else if** $\zeta=-1$ **then** $x\leftarrow cb\xi^2$
5. **else if** $\xi^2=\zeta$ **then** $x\leftarrow cb\xi^3$
6. **else then** $x\leftarrow cb\xi$
7. **Return** x

Table 6. $q\equiv 17\pmod{32}$일 때 제안된 제곱근 알고리즘

Input: A square c in F_q
Output: x satisfying $x^2=c$ in F_q

1. $b\leftarrow c^{\frac{q-17}{32}}$
2. $\zeta\leftarrow cb^2$
3. **if** $\zeta=1$, **then** $x\leftarrow cb$
4. **else if** $\zeta=-1$ **then** $x\leftarrow cb\xi^4$
5. **else if** $\xi^4=\zeta$ **then** $x\leftarrow cb\xi^6$
6. **else if** $\xi^4=\zeta^3$ **then** $x\leftarrow cb\xi^2$
7. **else if** $\xi^2=\zeta$ **then** $x\leftarrow cb\xi^7$
8. **else if** $\xi^2=\zeta^3$ **then** $x\leftarrow cb\xi^5$
9. **else if** $\xi^2=\zeta^5$ **then** $x\leftarrow cb\xi^3$
10. **else then** $x\leftarrow cb\xi$
11. **Return** x

Ⅳ. 결론 및 고찰

$q \equiv 2^s+1 \pmod{2^{s+1}}$일 때 F_q상에서 새로운 제곱근 알고리즘을 제안했다. $s=2,3,4$인 경우에 구체적인 알고리즘을 제안하였고 모든 알고리즘이 오직 한 번의 지수계산을 필요로 했다. 반면에 다른 알고리즘들은 (즉, Table 2,3) 한 번 이상의 지수계산을 필요로 했다. 또한, $s=4$인 경우는 (즉, Table 6) 다른 알고리즘들[9,10,11]에서는 제시되지 않은 새로운 결과이다. Table 7의 지수승 개수 비교에서 알 수 있듯이 제안된 알고리즘은 항상 한 번의 지수승만을 필요로 하므로 기존 알고리즘들에 비해 유리하다. 제안된 알고리즘은 s의 값이 작을 때 유용하다. 왜냐하면 주어진 s에 대하여 2^{s-1}개의 경우의 수가 고려되기 때문이다. 즉 s가 커질수록 경우의 수는 기하급수적으로 증가한다. 제안된 알고리즘이 비록 s가 작은 경우에 대하여 유용하지만 소수들의 대부분이 실제로 작은 s의 형태를 가지고 있다. 예를 들어, 전체 소수 q의 50%가 $s=1$(i.e., $q \equiv 3 \pmod 4$)이며, 전체 소수의 25%가 $s=2$(i.e., $q \equiv 5 \pmod 8$)이며, 일반적으로 전체 소수의 $\frac{100}{2^s}$%가 $q \equiv 2^s+1 \pmod{2^{s+1}}$의 형태로 표현된다. 따라서 이미 잘 알려진 $s=1$인 경우와 본 논문에서 제안된 $s=2,3,4$인 경우는 전체 소수의 $100(\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^4}) \approx 94\%$를 포함하므로 본 논문의 알고리즘들은 대부분의 소수에 대하여 유용하게 사용될 수 있다.

Table 7. number of necessary exponentiations in square root algorithms

$s=2$		Atkin[9]	Table 4
no. of exp.		1	1
$s=3$	Müller [10]	Kong[11]	Table 5
no. of exp.	2	1.5	1
$s=4$			Table 6
no. of exp.			1

References

[1] NIST, *Digital Signature Standard*, Federal Information Processing Standard 186-3, 2000, http://csrc.nist.gov/publications/fips/.

[2] D. Shanks, “Five number-theoretic algorithms,” in *Proc. Second Manitoba Conf. Numerical Math.*, pp. 51-70, Winnipeg, Canada, Oct. 1972.

[3] A. Tonelli, “Bemerkung über die Auflösung Quadratisher Congruenzen,” *Göttinger Nachrichten*, pp. 344-346, 1891.

[4] M. Cipolla, “Un metodo per la risoluzione della congruenza di secondo grado,” *Rendiconto dell’Accademia Scienze Fisiche e Matematiche*, vol. 9, no. 3, pp. 154-163, 1903.

[5] D. H. Lehmer, “Computer technology applied to the theory of numbers,” *Studies in Number Theory, Math. Assoc. Amer. (distributed by Prentice-Hall, Englewood Cliffs, N.J.)*, pp. 117-151, 1969.

[6] S. Lindhurst, “An analysis of Shanks’s algorithm for computing square roots in finite fields,” *CRM Proc. Lecture Notes*, vol. 19, pp. 231-242, 1999.

[7] D. G. Han, D. Choi, and H. Kim, “Improved computation of square roots in specific finite fields,” *IEEE Trans. Comput.*, vol. 58, no. 2, pp. 188-196, Feb. 2009.

[8] D.-G. Han, D. H. Choi, H. Kim, and J. Lim, “Efficient computation of square roots in finite fields F_{p^k},” *J. Korea Inst. Inform. Security Cryptology (KIISC)*, vol. 18, no. 6A, pp. 3-15, Dec. 2008.

[9] A. O. L. Atkin, “Probabilistic primality testing,” *summary by F. Morain, Inria Research Report 1779*, pp. 159-163, 1992.

[10] S. Müller, “On the computation of square roots in finite fields,” *Designs, Codes and Cryptography*, vol. 31, no. 3, pp. 301-312, Mar. 2004.

[11] 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, Apr. 2006.

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

조 국 화 (Gook Hwa Jo)

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