通信情報保護學會誌
第8卷 第2號, 1998. 6

소인수 분해 알고리즘

Factorization Algorithms

김진규*, 김영수**, 김성옥***

요 약

수많은 암호시스템과 관련 프로토콜의 안전이 소인수분해 문제의 어려움에 기반하고 있다. 본 논문에서는 암호해독과 설계에 영향을 줄 수 있는 소인수분해 알고리즘에 대하여 현재까지의 연구 동향과 성과를 기술하였으며, 연분수를 이용한 인수분해 알고리즘(CFRACT), QS(Quadratic Sieve), NFS(Number Field Sieve), 타원곡선 알고리즘 및 Pollard's p-1 알고리즘, Pollard's rho 알고리즘을 분석하였다.

Ⅰ. 서 론

합성수(composite number) n에 대한 인수분해 문제는 정수론의 가장 근본적인 문제의 하나로 Fermat나 Euler, Gauss 등 수많은 수학자들을 매혹시켜 왔다. 20세기 들어, 컴퓨터의 지속적인 발달로 컴퓨터의 방대한 계산능력을 활용할 수 있었고, 1977년 RSA 암호화시스템의 개발은 소인수분해 문제가 순수 수학적 영역을 벗어나는 새로운 전기가 되었다.

인수분해 알고리즘은 양의 정수 n을, $p_1^{e1} \cdot p_2^{e2} \cdots p_k^{ek}$로 표현($p_i$는 서로 다른 소수, $e_i \geq 1$)하는 알고리즘으로, 크게 special-purpose 알고리즘과 general-purpose 알고리즘으로 나눈다. Special-purpose 알고리즘은 인수분해 하려는 정수 n의 특별한 성질에 의존하며, general-purpose 알고리즘의 수행시간은 정수 n의 길이에만 의존한다. Trial Division이나, Pollard's p − 1 method, William's p+1 알고리즘, Pollard's rho 알고리즘, 타원곡선 알고리즘과 SNFS(Special Number Field Sieve)등 이 special − purpose 알고리즘에 속하며, QS(Quadratic Sieve), GNFS (General Number Field Sieve)등이 대표적인 general − purpose 알고리즘에 속한다[2][3][4][5].

1925년 A.J.C. Cunningham과 H.J. Wooldall에 의해 인수분해될 수들의 테이블($b^n \pm 1$ 형태의 수)이 발표되고, 이른바 "Cunningham project" 시작되어 현재까지도 진행되고 있다. 1983년 J. Brillhart, D. H. Lehmer, J. L. Selfridge, B.Tuckerman과 S. S. Wagstaff. Jr에 의해 두 번

* 한남대학교 컴퓨터공학과 석사과정
** 한국전자통신연구원 선임연구원
*** 한남대학교 컴퓨터공학과 교수

째 테이블이 발표되었다. 1988년 다음 버전의 테이블이 발표되었다. 이 프로젝트에 의해 인수분해 되지 않은 수들이 새로운 인수분해 알고리즘을 실제로 평가 기준으로 사용된다. Wagstaff는 정기적으로 "most-wanted" 수를 발표하는데, 현재의 "most-wanted" 수는 $2^{569}-1$, $6^{256}+1$과 $2^{4096}+1$이다[1].

표 1. 인수분해 기록

연도	자릿수	비트수	MY
1984	71	234	0.1
1988	106	352	140
1993	120	399	825
1994	129	429	5000
1995	119	395	250
1996	130	432	750

Ⅱ. Trial Division

페르마의 정리를 이용한 소인수 분해법으로 합성수(composite number) n이 두 정수의 크기가 비슷할 때, 즉 인수의 크기가 $\sqrt{n}$의 크기에 근접할 때 효율적이다. 홀수 n이 주어졌을 때 다음이 성립한다.

$n=a\cdot b\,(a\geq b>0) \Leftrightarrow n=t^2-s^2$의 형태로 표시할 수 있다. $(a=t+s, b=t-s)$

$$n=a\cdot b=\left(\frac{a+b}{2}\right)^2-\left(\frac{a+b}{2}\right)^2$$

이므로 위의 식이 성립함을 알 수 있다. $n=a\cdot b$의 형태일 때 $t>s$라 하면

$s=\dfrac{a+b}{2}$ 가 되고,

t를 $\sqrt{n}$보다 약간 더 큰 수라고 가정할 수 있다. Trial Division은 t를 찾기 위해 $[\sqrt{n}]+1$부터 $t^2-n=s^2$가 되는지를 계속해서 살펴나간다. 성공한다면 $n=(t+s)(t-s)$가 성립하므로 n의 인수분해가 가능하다.

200819를 인수분해 해보자. $(\sqrt{200819}\,)+1)+1=449$이고, $449^2-200819=782$가 된다. 782는 제곱수가 아니다. 계속해서 450에 대해 시도하면, $450^2-200819=1681=41^2$이므로

$$200819 = 450^2-41^2 = (450+41)(450-41)$$
$$= 491\ 409$$

로 인수분해 된다.

$n=a\cdot b$의 형태를 만족한다 하더라도 a와 b의 크기가 비슷하지 않다면 t를 구하기 위한 시도 횟수가 많아지게 되어 효율성이 떨어진다[2].

Ⅲ. Factor-Base 알고리즘

Factor-Base 알고리즘은 소수의 "factor-base"를 사용하여 인수분해에 이용할 수 있는 선형 방정식의 집합을 유도한다. 즉 factor-base를 정한 뒤 이를 이용하여 이차합동식 $x^2\equiv y^2 \bmod n\ (x\geq y)$을 유도한 후 $(x\pm y, n)$을 구하는 것이다. QS와 NFS가 이 계열에 속한다. 이러한 알고리즘은 병렬화(parallelized)하기 쉽다는 장점이 있어 네트워크로 연결하여 계산을 수행하기 용이하다[3][5].

3.1 Factor-base

Legendre는 이차합동식 $x^2\equiv y^2\ (\mathrm{mod}\ n),\ (x\neq\pm y),\ x^2\equiv y^2\ (\mathrm{mod}\ n),\ (x\neq\pm y)$을 만족하는 두 정수 x, y를 찾을 수 있다면, $d=$ g.c.d$(x\pm y, n)$을 계산하여 d가 non-trivial일 경우 인수를 찾을 수 있다는 사실을 발견하였다. d가 non-trivial일 확률은 50%이상 이므로 위의 2차 합동식을 만족하는 해를 10개정도 찾는다면 인수를 찾을 확률이 99.9% 이상이 된다.

factor-base $B=\{P_1,P_2,\cdots,P_k\}$가 $P_1=-1$

이고, 각 P_i가 서로 다른 소수들의 집합일 때 least absolute residue $b^2 \bmod n$이 B의 원소들의 곱으로 표현될 수 있으면 b를 B-number라 한다. 예를들어, $n=4633$이라 할 때, $67^2 \equiv 143 \bmod 4633$, $68^2 \equiv 9 \bmod 4633$, $69^2 \equiv 128 \bmod 4633$이므로 정수 67, 68, 69는 모두 B-number이다.

$F_2^h = F_2 \times \cdots \times F_2$를 0과 1의 두 원소로 구성된 필드상의 벡터 공간으로 정의한다. n과 h개의 원소를 포함하는 factor base B가 주어졌을 때, 벡터 $e \in F_2^h$의 B-number에 대응시킬 수 있다. 즉, $b^2 \bmod n$을 $\prod_{j=1}^{h} p_j^{\alpha}$의 형태로 표시하고, j번째 원소 e_j를 $\alpha_j \bmod 2$로 설정한다. 만약 α_j가 짝수이면 e_j는 0이 되고 α_j가 홀수이면 e_j는 1이 된다. 위의 예에서 각각

$$67 \longrightarrow \{1, 0, 0\}$$
$$68 \longrightarrow \{1, 0, 0\}$$
$$69 \longrightarrow \{0, 1, 0\}$$

와 같이 대응된다. Factor-Base 알고리즘은 다음과 같다.

step 1. 적당한 크기의 정수 y를 선택한다.

step 2. $B = \{p_i \mid p_i = 1, i > 1, p_i < y+1\}$, p_i : 소수

step 3. b_i를 여러개 선택하여 $b_i^2 \bmod n$이 B의 원소들의 곱으로 표시될 수 있는지 검사한다.

step 4. B-number b_i 들을 충분히 얻은 후 ($\pi(y)+2$ 정도) F_2^h에서의 대응되는 벡터들을 구하고 row reduction을 이용하여 대응되는 e_i들의 합이 0 벡터가 되도록 적당한 b_i의 값을 결정한다. $\pi(y)$는 y이하의 소수들의 개수이다.

step 5. 만약 $b \equiv \pm c \bmod n$이면 재 시도한다.

step 6. $b^2 \equiv c^2 \bmod n$, $b \not\equiv \pm c \bmod n$이면 $\gcd(b+c, n)$을 계산한다.

3.2 연분수 인수분해 알고리즘

연분수를 이용한 인수분해 방법(Continued Fraction Factoring Algorithm)은 RSA 암호시스템의 개발 이전에는 가장 효율적인 general-purpose 알고리즘으로 현재 쓰이고 있는 소인수 분해법 중 가장 오랜 역사를 가지고 있으며 40자리(133 비트) 이상의 수를 인수분해 하였다[5]. 1931년 D. H. Lehmer와 R. E. Powers에 의해 기본적인 개념이 제시되었고, 1975년 Michael Morrison과 John Brillhart에 의해 실용화되었다. 당시 Morrison과 Brillhart는 연분수를 이용한 인수분해법(CFRAC)으로 38자리의 페르마의 7번째 수(F_7)를 인수분해 하였다. √ 1981년, J. W. Smith와 Wagstaff는 EPOC(Extended Precision Operand Computer)라 불리는 CFRAC 알고리즘을 수행하기 위한 전용 컴퓨터를 설계하였으며, 이를 1개월 동안 수행하여 70자리수의 인수분해에 성공하였다[1].

CFRAC 알고리즘은 일반화된 페르마의 소인수 분해법($\sqrt{n}$ 대신 $\sqrt{kn}$을 이용)을 이용하며 연분수의 전개를 통해 $d_{n-1}^2 \equiv (-1)^n \cdot q_n \pmod n$ 형태의 합동식을 만들어 나간다. q_n 중 주어진 factor-base에 대한 B-number가 되는 수를 찾아 n의 인수분해를 시도한다[2].

실수 x에 대한 연분수를 다음과 같이 정의한다.

$$a_0 = [x],\ x_0 = x - a_0,\ a_1 = \left[\frac{1}{x_0}\right],\ x_1 = \frac{1}{x_0 - a_1}$$

이라 한다.

$i > 1$에 대해, $a_i = \left[\frac{1}{x_{i-1}}\right]$, $x_i = \frac{1}{x_{i-1} - a_i}$ 으로 정의하면,

$\frac{1}{x_{i-1}}$이 정수가 될 때, $x_i = 0$이 되어 이 과정은 멈추게 된다.

이 과정이 멈추게 될 필요충분조건은 x가 유리수임은 자명하다. 이 과정에 의해 x를 다음과 같이 표기할 수 있다.

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots + \cfrac{1}{a_i + x_i}}}$$

이 식을 $[a_0, a_1, a_2, \cdots, a_i, x_i]$와 같이 표기하기도 한다. $x_i \neq 0$이면 x는 무리수가 된다.

67과 24의 최대 공약수를 구하는 알고리즘은

$$67 = 2 \times 24 + 19$$
$$24 = 1 \times 19 + 5$$
$$19 = 3 \times 5 + 4$$
$$5 = 1 \times 4 + 1$$

이를 분수형태로 쓰면

$$67/24 = 2 + (19/24)$$
$$24/19 = 1 + (5/19)$$
$$19/5 = 3 + (4/5)$$
$$5/4 = 1 + (1/4)$$

가 된다. 이를 연분수 전개(continued fraction expansion) 하면,

$$\frac{67}{24} = 1 + \cfrac{1}{1 + \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{4}}}} = [2, 1, 3, 1, 4]$$

과 같이 표시할 수 있다[13].

연분수 전개 알고리즘은 다음과 같다. 정수 n을 인수분해 될 수라고 가정한다.

step 1. $b_1 = 1$, $b_0 = a_0 = [\sqrt{n}]$, $x_0 = \sqrt{n} - a_0$

step 2. $b_0^2 \pmod n$을 계산한다. $i = 1, 2, 3, \cdots$ 에 대하여 다음을 반복한다.

step 3. $a_i = \left[\frac{1}{x_{i-1}}\right]$, $x_i = \frac{1}{x_{i-1} - a}$

step 4. $b_i = a_i \cdot b_{i-1} - b_{i-2} \pmod n$로 설정한다.

step 5. $b_i^2 \bmod n$를 계산한다.

step 6. (3)의 과정에서 나타나는 $b_0^2 \pmod n$를 인수분해 하여 나타나는 모든 소수들과 -1로 이루어진 factor base B를 만든다.

step 7. 대응하는 벡터들이 0 벡터가 되도록 하는 부분집합을 구한다.

$b = \prod b_i (\Sigma e_i = 0$ 벡터$)$

$c = \prod_{p}^{r(j)}$, $r(j) = (\frac{1}{2}\Sigma a(i,j))$

step 8. 만약 $b \neq \pm c \pmod n$이면 g.c.d$(b+c, n)$는 n의 non-trivial 인수가 된다. $b \equiv \pm c \pmod n$ 이면 $\Sigma e_i = 0$가 되는 i의 다른 부분집합을 찾는다.

step 9. $\Sigma e_i = 0$ 벡터인 i의 부분집합을 찾는 것이 불가능하다면, factor-base B를 증가시키면서 a_i, b_i 와 $b_i^2 \pmod n$를 계산한다. □

b_i^2를 만들고 인수분해 하는 과정은 시간은 많이 필요로 하지만 적은 기억용량을 차지하며, $b_i^2 \pmod n$의 크기가 작기 때문에 주어진 factor base에 대한 B-number가 커진다.

Ⅳ. Quadratic Sieve (QS)

QS 알고리즘은 1980년대 초, Maurice Kraitchek에 의해 소개되었고, 1984년 Carl Pomerance에 의해 완성되었다. 1983년 Davis와 Holdridge는 CRAY-1과 CRAY-XMP를 이용하여 70자리(233 bit)의 합성수를 인수분해 하였다. 이후 MPQS(Multi Polynomial Quadratic Sieve)와 같은 여러 변종으로 발전하였다.

정수 계수를 가지는 2차 다항식

$$f(x) = (x + [\sqrt{n}])^2 - n = f(x) \bmod n$$

에 대하여, x가 정수인 경우

$$(x + [\sqrt{n}])^2 \equiv f(x) \bmod n$$

이 성립한다. 따라서 적당한 $x_1, \cdots x_k$ 가 존재하여 $f(x_1) \cdots f(x_k) = y^2$를 만족한다고 가정하면,

$x = (x_1 + [\sqrt{n}]) \cdots (x_1 + [\sqrt{n}])$이 되고 $x^2 \equiv y^2 \bmod n$이 되어 g.c.d$(x \pm y, n)$를 구할 수 있으

므로 인수분해를 시도할 수 있다.

$f(x_1)\cdots f(x_k)$가 제곱수가 되는 $x_1,\cdots,x_k$를 얻기 위해 factor base B를 정하고 $f(x)$중 B-number가 되는 것들을 찾아 Gauss 소거법을 이용한 일차종속 관계식을 유도해야 한다.

모든 $k(\in Z)$에 대해 $p \mid f(x+kp)$이므로 $f(x) \equiv 0 \bmod p$를 이용 두 근 r_1, r_2를 구하면 $r_1 \pm kp$, $r_2 \pm kp$가 모두 근이 된다. 여기에서 Legendre Symbol $\left(\frac{n}{p}\right)=1$이 되어야 한다. QS 알고리즘은 다음과 같다.

step 1. 적당한 factor base $B=\{p_i : \left(\frac{n}{p}\right)=1, p_i$:prime, $i=1, 2, \cdots, h$ and $p_0=-1\}$을 택한다.

step 2. factor-base내의 모든 p_i $(i \geq 1)$에 이차합동식 $f(x) \equiv 0 \bmod p_i$를 푼다. 각 p_i에 대한 근을 각각 r_1, r_2라 한다.

step 3. 적당한 sieve interval $[-M, M]$을 정하고 sieve array를 0으로 한다.

step 4. B내의 각 p_i에 대해 r_1, $r_1 \pm p_i$, $r_1 \pm 2p_i$, $\cdots$, r_2, $r_2 \pm p_i$, $r_2 \pm 2p_i$, $\cdots$ 등의 위치에서 sieve array에 $[\log(p_i)]$ 값을 더한다.

step 5. $f(x)$의 값은 $[-M, M]$에 대해 $M\sqrt{7}$ 근방이므로 각 sieve location에서의 값을 $[\log\sqrt{2}+\log M]$과 비교한다. B-number가 되는 것은 Sieve location 값이 이 값과 근사한 값이다.

step 6. 각 B-number에 대해 f_i^b의 벡터를 대응시키고 factor-base 알고리즘을 적용한다.

QS 알고리즘은 기존의 연분수(CFRAC) 알고리즘을 완전히 대체할 정도로 효율적이어서 QS를 기본으로 효율을 향상시키려는 노력이 계속 진행되었다. Peter Montgomery는 1985년 복수개의 다항식을 사용하는 MPQS 알고리즘을 제안하여, QS 알고리즘에 비해 10배 이상의 속도 향상을 가져왔다. MPQS의 수행 시간은 $o(e^{(1+0(n))\sqrt{\ln n \ln\ln n}})$ 이다.

1987년 Robert D. Silverman은 이 알고리즘을 1265시간 동안 수행$2^{359}+1$의 인수분해 하는데 성공하였다[10][11].

QS에서 $f(x)=(x+[\sqrt{n}])^2-n$, 한 개의 다항식을 사용한 반면 MPQS에서는

$$f_{a,b}(x)=ax^2+2bx+c(b^2-ac=n,\ 0\leq b\leq a)$$

형태의 다항식을 사용한다. 이 식에서

$$a\cdot f_{a,b}(x)=(ax+b)^2-n$$

이 되고 QS의 경우와 유사한 다항식

$(ax+b)^2=a\cdot f_{a,b}(x) \bmod n$을 유도할 수 있다. 선별 구간(Sieving interval)을 $[-M,M]$으로 하면 $f_{a,b}(x)$의 최대값은 양 끝점에서 얻어지므로

$$f_{a,b}\left(-\frac{b}{a}\right)=\left(\frac{1}{a}\right)\cdot(a^2M^2-n)$$

이 되고 최소값은 $x=-\frac{b}{a}\left(-1<-\frac{b}{a}<0\right)$에서

$f_{a,b}\left(-\frac{b}{a}\right)=-\frac{n}{a}$이 된다.

위에서 최대값과 최소값의 길이가 거의 같도록 a, b, c를 선택하는데, $f(x)$의 크기를 작게 하여 $f(x)$가 B-number가 될 확률을 높이려는 이유이다.

$a=\sqrt{\frac{2n}{M}}$ 으로 하면 이 조건을 만족시킬 수 있으며, 이와같이 a를 선택한 후 b를 몇가지로

$$b^2\equiv a \bmod n,\ 0\leq b\leq a$$

에 의해 선택하면 c는 $b^2-ac=n$에 의해 결정된다. 위의 식으로 b를 결정하기 위해서는 a의 소인수를 알아야 하기 때문에 초기에 $\left(\frac{n}{q}\right)=1$이고 $\left(\sqrt{\frac{2n}{M}}\right)^{1/k}$정도의 거의 같은 크기 를 갖는 k개의 소수들의 곱으로부터 a를 만들고 이로부터 b와 c를 만드는 것이 좋다.

이와 같이 $f_{a,b}(x)$를 만들고 난 후에는 QS와 동일한 과정을 거치며, 만족할만한 B-number가 생성되지 않는 경우 a, b를 바꾸어서 다시

시도한다. MPQS 알고리즘은 몇 가지 방향으로 변형되어 사용된다.

큰 소수를 사용하는 경우, Sieving을 수행한 결과,

$$v^2 = q_v \cdot \prod p^{e_{u_p}} \bmod n$$

과 같은 관계식을 발견하였다고 하자. q_v는 factor base에 속해있지 않은 큰 소수(large prime)로 factor base 내의 최대 소수의 제곱보다 작은 것이다. Sieving 후 남은 cofactor가 최대 소수의 제곱보다 작으면 cofactor는 소수이므로, 위에서와 같은 큰 소수가 된다. 따라서 large prime을 하나 추가하는 것은 실행시간의 오버헤드가 되지 않는다. 다른 합동식

$$u^2 = q_u \cdot \prod p^{e_{u_p}} (q_u = q_v)$$

이 발견되면

$$(uv)^2 = qu^2 \cdot \prod p^{e_{u_p}+e_{v_p}}$$

로부터 (qu n) = 1 인지를 확인한 후 q_u^2을 소거하여 B-number를 소거하는데 쓰인다. MPQS에서 large prime 사용을 통해 2배 이상의 속도 개선이 이루어졌다고 추정하고 있다.

또 다른 변형의 경우, 하나의 큰 소수를 사용하는 알고리즘에 2개의 large prime을 적용한다. 즉

$v^2 = q_{1v} \cdot q_{2v} \cdot \prod p^{e_{u_p}} (q_u = q_v) \bmod n$, q_{1v}, $q_{2v} \notin B$

로 q_{1v}, q_{2v}는 large prime이다. large prime을 그래프의 vertex에 대응시킨 후 두 large prime이 같은 합동식에서 발생하면 edge로 이어준다. 이 그래프에서의 cycle은 large prime들이 짝수이면 발생한 합동식들을 가져온다. 따라서 1.5.3에서와 같이 large prime을 소거함으로써 B-number를 얻을 수 있다.

1994년, Arjen Lenstra가 주도하는 연구원 그룹은 이 알고리즘을 이용하여, RSA-129 (429 bit)를 인수분해 하는데 성공하였다. RSA-n은 RSA 공개키 알고리즘에 이용되는 $n = p \cdot q$ 형태의 수로 1977년 영국의 수학자 Martin Gardner가 Scientific American지에 기고한 글에서 처음 제시되었다. RSA-129의 인수분해에는 8개월 동안 전 세계 1600대의 컴퓨터를 이용하였고, 5000 MY가 소요되었다. 이는 전 세계 인터넷 계산능력의 0.03%, 전세계 컴퓨터 계산능력의 3%에 해당하는 수치이다[4].

V. Number Field Sieve

1989년 말 A. K. Lenstra, H. W. Lenstra, M. S. Manasse등은 NFS라 불리는 $r^e \pm |s|$ 형태의 수에 적용되는 새로운 special-purpose 인수분해법을 개발하였다. NFS의 실행 시간은 $O(e^{(1.92+o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}})$으로, MPQS 보다 효율성이 뛰어나다. 현재 150 자릿수 이상을 인수분해할 수 있는 가장 빠른 알고리즘으로 알려져 있지만, 반면 가장 복잡한 알고리즘이기도 하다.

처음 제안되었을 당시에는 비실용적인 알고리즘으로 인식되었지만 현재는 110~120자릿수 이상의 수에 대해서는 현존하는 인수분해 알고리즘 가운데에서 가장 효율적이다. 1996년 Arjen Lenstra가 주도하는 연구원 그룹은 NFS를 이용 RSA 130을 인수분해 하는데 성공하였다. 이 작업에는 RSA-129의 인수분해에 필요로 했던 5000 MY의 15%만을 필요로 하였다[5].

합성수 $n = r^e - s$라 하자. 먼저 extension degree $d \in Z_{70}$을 선택한다. $k \in (Z_{70})$를 $kd > e$가 설립하는 최소의 수라 가정한다. 따라서, $r^{kd} \equiv sr^{kd-e} \pmod n$가 성립하고 $m = r^k$, $c = sr^{kd-e}$라 놓으면 $m^d \equiv c \pmod n$가 성립한다. $f(x) = x^d - c \in Z[x]$라 하자. $f(x)$가 가약(reducible)인 다항식

표 2. RSA-n의 인수분해

RSA-n	년도	MY	알고리즘
RSA-100	1991	7	QS
RSA-110	1992	75	QS
RSA-120	1993	830	QS
RSA-129	1994	5000	QS
RSA-130	1996	500	GNFS

표 3. NFS를 이용한 인수분해

비트수	GNFA(MY)	SNFS(MY)
512	$3 \cdot 10^4$	—
768	$2 \cdot 10^8$	$1 \cdot 10^7$
1024	$3 \cdot 10^{11}$	$3 \cdot 10^7$
1280	$1 \cdot 10^{14}$	$3 \cdot 10^9$
1536	$3 \cdot 10^{16}$	$2 \cdot 10^{11}$
2048	$3 \cdot 10^{20}$	$4 \cdot 10^{14}$

이기 위한 필요충분조건은 $p \mid d$인 소수 d가 있어서 p번째 승(p-th power)이 되는지 $4 \mid d$이고 $-4c$가 4번째 승($4th$ power)이 라는 것이다. 따라서 d를 잘 선택하면 $f(x)$가 기약 다항식 (irreducible polynomial)이라 가정해도 좋다.

$f(x)$가 기약이므로 $f(a)=0$을 만족하는 a에 대해 필드 $K=Q(\alpha)$를 만들 수 있다. 무리없이 $Z[\alpha]$를 UFD(Unique Factorization Domain)이라 가정할 수 있다. 이 때, K의 정수 링(ring of integer)은 $Z[\alpha]$가 된다.

$\phi : Z[\alpha] \rightarrow Z/nZ$를 $\phi(\alpha) \equiv m \pmod n$인 ring homomorphism 이라 하자.

예를들어 $n = 3239^{-1}$은 $d=5$, $m=3^{48}$, $f(x) = x^5 - 3$, $K=Q(3^{1/5})$이었고, $Z[3^{1/5}]$는 UFD가 된다.

NFS의 개념은 $a+\alpha b$, $a+mb$가 smooth가 되게하는 작은 서로소인 정수쌍 a, b들을 sieving을 통해 찾아내는 것이다. 대수적 정수 $a+\alpha b$가 smooth라는 의미는 $a+\alpha b$가 작은 노름(small norm), 특히 소수 노름인 소수 아이디얼(ideal)에 의해서만 나누어진다는 의미이다. $a+\alpha b$의 prime ideal factorization은 $N(a+\alpha b) = a^d\, c(b)^\alpha$의 소인수 분해에 대응되고 있다. $B \in R > 0$ 를 smoothness bound라 할 때 $N(a+\alpha b)$와 $a+mb$ 가 B-smooth라 하자. 즉,

$$|N(a+\alpha b)| = \prod_{p\text{prime},\ p \le B} p^{v_p}$$

$$|a+mb| = \prod_{p\text{prime},\ p \le B} p^{w_p}$$

$N(a+\alpha b)$의 소인수 분해로부터 $a+\alpha b$의 units이 소인수로의 인수분해

$$a+\alpha b = \left(\prod_{u \in U} u^{t_u}\right)\left(\prod_{g \in G} g^{v_g}\right)$$

를 유도하였다고 하자. 이 식에서

$$\left(\prod_{u \in U} \varnothing(u)^{t_u} \prod_{g \in G} \varnothing(g)^{v_g}\right) = \left(\prod_{p\text{prime},\ p \le B} p^{w_p} \bmod n\right)$$

을 얻을 수 있다. 만약 sieving을 통해 충분히 많은 이러한 a, b의 쌍을 ($|U|+|G|+|\pi|(B)$ 보다 많은) 얻을 수 있다면

$$\prod_{a,b} (a+\alpha b)^{x(a,b)} = \left(\left(\prod u^{\overline{t_u}}\right)\left(\prod u^{\overline{v_g}}\right)\right)$$와

$$\prod_{a,b} (a+mb)^{x(a,b)} = \left(\prod_{p\text{prime},\ p \le B} p^{w_p}\right)^2$$

이 성립하게 되는 $x(a,b) \in \{0,1\}$들을 앞의 factor-base 알고리즘과 같이 지수 벡터들에 대한 modulo 2의 Gauss 소거법에 의해 구할 수 있다. 위의 두 식으로부터

$$\left(\left(\prod_{u \in U} \varnothing \overline{u^{t_u}}\right)\left(\prod_{g \in G} \varnothing g^{\overline{v_g}}\right)\right)^2 = \left(\prod_{p\text{prime},\ p \le B} p^{w_p}\right)^2 \bmod n$$

이 성립한다. 따라서 이차합동식 $x^2 \equiv y^2 \bmod n$이 성립하는 정수 x, y를 얻을 수 있다. 다시 $g.c.d(x \pm y, n)$를 계산하여 non-trivial이면 n의 인수를 찾을 수 있다.

NFS는 이산 로그 문제의 해결에 있어서도 현재 가장 효율적인 알고리즘으로 알려져 있다. 이산 로그 문제는 인수분해의 문제와 같은 정도의 난이도를 가진 것으로 평가되고 있는데, k-bit (modulo p)의 로그를 찾는 문제는 k-bit 합성수의 인수분해문제와 거의 같다고 할 수 있다. 1990년 Brian LaMacchia와 Andrew Oldzko는 Gaussian integer method라

이산 로그를 구했다. NFS를 이용한 기록으로, 독일 Saarlandes 대학의 학생이었던 Weber, Denny와 Zayer는 248 bit 소수 modulo에서의 이산 로그를 계산하였다[5].

NFS(와 QS)는 쉽게 병렬화(parallelized)가 가능하기 때문에 분산 네트워크(distributed network)를 통한 작업이 용이하다[8].

Ⅵ. 타원곡선을 이용한 소인수 분해

타원곡선을 이용한 소인수 분해 알고리즘(ECM)은 현재 가장 효율적인 인수 분해 알고리즘의 하나로 1986년 H. W. Lenstra에 의해 개발되었다. 타원곡선법은 Pollard 알고리즘이나 Williams p+1 알고리즘을 일반화시킨 special-purpose 알고리즘으로 상대적으로 크기가 작은 소인수를 찾는데 적합하다[8]. 1995년 독일 Saarlands 대학의 Andreas Mueller는 이 ECM 방법을 이용하여 99 자리수 (329 bit)의 44자리수(147 bit) 소인수를 발견했다고 발표하였다. 이 작업은 네트워크로 연결된 워크스테이션에 의해 60 MY가 소요되었다[4][5]. 1995년에는 Peter Montgomery가 135자리수(449 bit) 합성수의 47자리(157 bit)의 소인수를 찾아낸 기록이 있다.

6.1 Pollard's p-1 Method

Pollard's p-1 method는 1974년 J. M. Pollard에 의해 발표된 special-purpose 알고리즘으로 페르마의 소정리(小定理)를 이용하며, 타원곡선을 이용한 소인수 분해와 비슷한 개념을 가진다. Pollard의 알고리즘의 기본 개념은 다음과 같다.

n을 인수분해 한다고 가정하면, 어떤 n의 소인수 p,q에 대해 $a^e \equiv 1 \pmod{p}$와 $a^e \not\equiv 1 \pmod{p}$를 만족하는 정수 e와 a를 찾을 수 있고, $p \mid (a^e-1)$이고 $p \nmid (a^e-1)$이 되어, $g.c.d(a^e-1, n)$는 p에 의해서는 나누어지지만 q로는 나누어지지 않는 n의 nontrivial한 인수가 된다는 것이다.

입력 n에 대해 Pollard의 알고리즘은 다음과 같이 수행된다[7].

step 1. 어떤 bound B보다 작은 모든 정수의 최소 공배수가 되는 정수 e를 선택한다. 예를들어 M보다 작은 모든 정수의 최소 공배수가 될 것이다. 이를 간단히 하기 위해 $e = \prod_{i=1}^{\pi(B)} p_i^{\alpha_i}$로 설정하며 여기에서 $p_1, p_2, \ldots, p_{\pi(B)}$는 B보다 작은 소수 그리고 α_i 는 $p_i^{\alpha_i} \geq \sqrt{n} > \mathrm{MIN}_{p,n}\{p-1\}$가 되도록 하는 α_i중 최소의 것을 선택한다.

step 2. 2와 $n-2$ 사이에서 임의의 정수 a를 선택한다. 예를들어 a는 2나 3 또는 임의로 선택된 정수이다.

step 3. square 연산을 반복하여 $a^e \bmod n$을 계산한다.

step 4. 유클리드 알고리즘을 사용하여 $a = g.c.d(a^e-1, n)$을 계산한다. 만약 $1<d<n$이면 nontrivial factor d를 출력한다. 이외의 경우에는 a를 다시 선택하여 step 2부터 반복한다.

6.2 Pollard's rho 알고리즘

Pollard's rho 알고리즘은 작은 인수를 찾는데 적합한 special-purpose 알고리즘이다. $f: S \rightarrow S$를 random 함수라 하고, S를 기수 n(cardinality n)의 유한 집합이라 한다. x_0을 S의 임의의 원소라 하고, $x_{i+1} = f(x_i)$ $(i \geq 0)$에 의해 정의되는 순열 $x_0, x_1, x_2, \cdots$ 를 고려해보자. S는 유한이므로 이 순열은 결국 cyclic이 되며, 길이 $\sqrt{\pi n/8}$ 의 tail과 $\sqrt{\pi n/8}$ 의 무한히 반복되

는 cyclic로 구성될 것이다. 여기에서 collision을 찾게 되는데, 즉 $x_i = x_j$가 되는 서로 다른 i와 j를 찾는다. collision을 발견하는 가장 간단한 방법은 $i = 0,1,2,\cdots$ 에 대하여 x_i를 계산, 저장한 다음 중복(duplicate)을 찾아내는 것이다. 중복이 검출되기 이전에 필요한 입력의 기대치는 $\sqrt{\pi n/2}$ 이다. 이 방법은 $O\sqrt{n}$ 의 메모리와 $O\sqrt{n}$ 시간이 소요된다.

p를 n의 소인수라 하면, Pollard's rho 알고리즘은 n을 인수분해 하기위해, $x_0 = 2$, $x_i + 1 = f(x_i) = x_i^2 + 1 \ (mod\ p)$에 의해 정의된 정수 순열 $x_0, x_1, x_2, \cdots$ 에서 중복(duplicate)을 찾는다. $x_m \equiv x_{2m} \ (mod\ p)$이 되는 x_m과 x_{2m}을 찾기 위해 Floyd's cyclic finding 알고리즘을 이용한다[3]. $p|n$이지만 p를 알지 못하기 때문에, $x_i \ mod\ n$을 계산하고 $g.c.d(x_m - x_{2m}, n) > 1$인지를 검사해 본다. $g.c.d(x_m - x_{2m}, n) < n$이라면 n의 non-trivial 인수를 얻게 된다. $g.c.d(x_m - x_{2m}, n) = n$이 될 확률은 극히 미미하다. 입력 n에 대하여 Pollard's rho 알고리즘은 다음과 같다.

step 1 $a = 2$, $b = 2$로 설정한다. $i = 1,2,\cdots$에 대하며 다음을 수행한다.

step 2 $a = a^2 + 1 \ mod\ n$, $b = b^2 + 1 \ mod\ n$을 계산한다.

step 3 $d = g.c.d(a - b, n)$을 계산한다.

step 4 $1 < d < n$ 이면 d를 반환한다. (성공)

step 5 $d = n$ 이면 알고리즘을 종료한다. (실패)

Pollard's rho 알고리즘은 ECDLP(Elliptic Curve Discrete Logarithm Problem)에 가장 최적의 알고리즘으로 알려져 있으며 $\sqrt{\pi n/2}$ step (타원곡선의 덧셈)을 필요로 한다. 1993년 Paul von Oorschot와 Michael Wiener는 r개의 프로세서 상에서 Pollard's rho 알고리즘을 병렬화하는 방법을 소개하였다. 각 프로세서가 수행하는 연산의 기대치는 $(\sqrt{\pi n/2})/r$이 된다.

6.4 Lenstra의 타원곡선 인수분해법 (ECM)

ECM은 Special-purpose 형태의 알고리즘 중 속도면에서 가장 빠르다[future]. 1987년 말 일본의 Kenji Koyama는 이를 이용하여 $2^{713}-1$의 23자리 소수를 발견하였으며[12], A. Lenstra에 의해 145 비트수의 인수분해에 성공한 기록이 있다[6].

Lenstra의 알고리즘은 Pollard의 방법과 유사하며 가산(addtion) 연산을 사용한다. n을 홀수인 합성수이고 d를 n의 소인수라 하자. 즉, $d|n$이고 $1<d<n$이다.

Lenstra의 타원곡선 인수분해 알고리즘은 ring Z_n상에 정의된 타원곡선 $E_{a,b}(Z_n)$상에서 행하여진다. 타원곡선 조건 $4a^3+27b^2 \neq 0$은 $g.c.d(4a^3+27b^2, n) = 1$이라는 조건으로 대치된다.

Lenstra 알고리즘의 아이디어는 $E_{a,b}$에서 P+Q가 정의되지 않는 P,Q ($\in E_{a,b}(Z_n)$)를 찾는 것이다. 알고리즘이 시작되기 전에 나누어 떨어질 수 있기 때문에, 알고리즘이 수행되는 동안 n이 2나 3을 인수로 가지지 않는다고 가정한다. 입력 n에 대하여 알고리즘은 다음과 같이 실행된다.

step 1. x,y와 a를 랜덤하게 선택하고 b를 $y^2 - x^3 - ax \ (mod\ n)$으로 설정함으로서 타원곡선 $E_{a,b}(Z_n)$과 $E_{a,b}(Z_n)$상의 한 점 $P = (x, y)$를 생성한다.

step 2. $g.c.d(4a^3+27b^2, n)$를 계산한다. 만약 $1 < g.c.d(4a^3+27b^2, n) < n$ 이면 n의 약수를 찾게되고 알고리즘을 멈춘다. $g.c.d(4a^3+27b^2, n) = 1$이면 n의 모든 소인수 p에 대하여 $(4a^3+27b^2, n) \not\equiv 0 \ (mod\ P)$이 되고 $E_{a,b}$는 n의 각 소인수 p에 대해 Zp상의 타원곡선이 되어 알고리즘은 계속 진행될

것이다. 그러나 g.c.d$(4a^3+27b^2, n)$이 n이라면 또다른 타원곡선 $E_{a,b}$를 생성해야 한다.

step 3. e를 $\prod_{i=1}^{\pi(B)} p_i^{\alpha i}$ 로 설정한다. $p_1, p_2, \cdots, p_{\pi(B)}$는 B보다 작거나 같은 소수이며 αi는$p_i^{\alpha i} \leq C$가 되는 최대 수로 선택한다. B와 C는 실행 시간을 최적화하도록 결정되는 범위이며 알고리즘이 반드시 성공하도록 보장한다.

step 4. doubling 연산을 반복하여, $E_{a,b}(Z_n)$에서 e P를 계산한다. 두 중간점 $P_1 = (x_1, y_1)$와 $P_2 = (x_2, y_2)$를 더하기 이전에 g.c.d$(x_1 - x_2, n)$ 또는 g.c.d(y_1+y_2, n)가 n의 nontrivial 약수가 되는지를 검사한다. 만약 약수가 된다면 이 약수를 출력하고 알고리즘을 종료한다. 그렇지 않으면 step 1부터 다시 시작한다.

타원곡선 $E_{a,b}(Z_n)$상에서 p,q가 n의 소인수이며, $E_{a^p,\ b^p}(Z_n)$상에서 $e \cdot P_p = O$ 이지만 P_q가 $E_{a^p,\ b^p}(Z_n)$상에서 e를 나누는 order를 가지지 않는 경우, nontrivial 한 인수분해를 행할 수 있다.

Lenstra의 타원곡선 알고리즘과 Pollard's의 $p-1$ 알고리즘 사이의 유사점은, Pollard의 알고리즘에서는 e가 a ($\in Z_p^*$)의 order의 배수가 되도록 n의 소인수 p, q를 찾는다는 것이다. 유사하게, Lenstra의 알고리즘에서는 e가 P_p ($\in$ ' $E_{a_p,\ b_p}(Z_p)$)의 order의 배수이지만 P_q ($\in$ ' $E_{a_q,\ b_q}(Z_q)$)의 배수는 되지 않도록 n의 소인수 p, q를 찾는다. 두 알고리즘의 주된 차이점은 융통성에 있다. Pollard 알고리즘에서, group Z_p^*의 p는 범위가 n의 소인수로 고정되어 있어, 이 그룹이 e를 나누는 order를 가지지 않는 경우 이 방법은 실패하게 된다. Lenstra의 타원곡선 알고리즘에서, 그룹 $E_{a_p,\ b_p}(Z_p)$는 a,b 값을 조절함으로서 다양화될 수 있다. 따라서, n의 모든 소인수 p에 대해, $|E_{a_p,\ b_p}(Z_p)| \nmid e$이면, 또다른 타원곡선을 선택하여 작업을 계속할 수 있는데, 단순히 새로운 a, b값을 선택하기만 하면 된다[7].

Ⅶ. 결 론

어떤 정수가 합성수(composite)인지 소수인지를 판별하는 일은 일반적으로 정수의 인수분해 문제보다 쉬운 것으로 알려져 있다. 따라서 어떤 정수를 소인수 분해하기 전에 이의 합성수 여부를 먼저 판정하는 방식을 취한다[3]. 또한 $n \geq 2$ 이면, n이 완전 제곱수(perfect power)인지 아닌지의 여부를, 즉 $n = x^k$ ($n \geq 2$, $k \geq 2$) 인지를, 효율적으로 검사할 수 있다. 소수 $p(\leq lg\ n)$에 대하여 $n^{1/p}$의 정수 근사값 x가 계산한다. 구간 $[2, 2^{\lceil lg(n/p)+1 \rceil}]$에서 $n = x^p$를 만족하는 x에 대한 이진 탐색(binary search)을 수행하면, $O((lg^3\ n) lg\ lg\ lg\ n))$의 비트 연산이 소요된다. Bach와 Sorenson이 소개한 완전 제곱수 검출하는 알고리즘의 경우, 최악의 경우 O(lg3 n)의 비트 연상을, 평균 O(lg2 n)의 비트 연산을 수행한다. 합성수 여부의 판별과 완전 제곱수의 판정이 수행되었다고 가정한다면 일반적인 인수분해 전략은 다음과 같다[2][3].

임의의 bound b_1 이하의 작은 소수로 Trial Division을 적용한다. Trial Division이나 Pollard's $p-1$ 알고리즘, Pollard's rho 알고리즘은 아주 작은 소수를 발견하는데 유리하므로 가장 먼저 적용하게 된다.

Pollard's rho 알고리즘이나 Pollard's rho 알고리즘을 적용하여 임의의 bound b_2보다 작은 소인수를 찾는다. ($b_1 < b_2$). Pollard's $p-1$ 알고리즘은 p가 n의 소인수일 때 비교적 작은 수 B에 대하여 $p-1$이 B이하인 소수들의 곱으로 이루어지는 아주 특수한 성질에 의존하며, 아주 작은 인수나 특수한 인수를 발견하는데 유리하다. Lenstra의 알고리즘은 이에 비해 여러

타원곡선을 선택할 수 있게 함으로서 융통성이 뛰어나다.

타원곡선 인수분해 알고리즘을 적용하여 bound b_3 보다 작은 소인수를 찾는다. ($b_3 > b_2$). 타원곡선을 이용하여 20~30자리 인수를 발견하고, 나머지 인수가 합성수이면 QS나 NFS를 적용하는 것이 효율적이다. 타원곡선 알고리즘은 n이 비슷한 크기의 두 인수의 곱으로 구성되어 있을 때, QS와 같은 수행시간을 가진다. 그러나 일반적으로 QS가 single-precision 연산을 수행하기 때문에 multi-precision 연산을 수행하는 타원곡선 알고리즘에 비해 효율이 좋다. 그러나 타원곡선법은 200자리 합성수의 25자리 인수의 발견시간과 100자리 합성수의 25자리 인수의 발견 시간이 거의 같다는 특징을 가지고 있으며, 30자리 정도의 상대적으로 작은 소인수를 발견하는데 가장 효율적인 방법이다. 또한 타 알고리즘에 비해 가장 적은 기억 장소만을 차지한다.

최종적으로 좀더 강력한 general-purpose 인수분해 알고리즘 MPQS나 NFS 알고리즘을 적용한다. SNFS는 $r^e \pm s$ (r과 s 가 상대적으로 작은) 형태의 합성수에만 적용되지만, GNFS는 현존 인수분해 알고리즘 중 최고의 효율성을 보이고 있다.

Ⅷ. 참고문헌

[1] Eric Bach, Jeffrey Shallit, Algorithmic Number Theory, Volume I, Efficient Algorithms, MIT press, 1997.

[2] 한국전자통신연구소, 소인수 분해 알고리즘 분석에 관한 연구, 1991.

[3] A. J. Menezes, P. C. van O.orschot, and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, Florida, 1997.

[4] A. Oldyzko, "The future of integer factorization", CryptoBytes, Volume 1, number 2 pp.5~12, 1995.

[5] Certicom Corp., "Remarks on the Security of the elliptic curve cryptosystem", 1997

[6] Adi Shamir, "RSA for Paranoids", CryptoBytes, Volume 1, number 3 pp. 1~4, 1995.

[7] Shafi Goldwasser, Mihir Bellare, Lecture note on Cryptography, http://www-cse.ucsd.edu/user/ mihir, July, 10, 1996.

[8] Jonathan S. Greenfield, Distributed Programming Paradigms with Cryptography Application, Springer-Verlag, pp.61~69, 1994.

[9] H. E. Rose, A Course in Number Theory, 2nd Edition, Clarendon Press, 1994.

[10] B, Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, New York, 2nd edition, 1996.

[11] D. R. Stinson, Cryptography Theory and Practice, CRC Press, 1995.

[12] 임종인, 김창한, "소인수 분해와 암호학", 한국통신정보보호학회지, 제 1권, 제 1호, 1991, 4.

[13] 조인호, "연분수와 암호학", 한국통신정보보호학회지, 제 1권, 제 1호, 1991, 4.

[14] Alfres Menezes, Elliptic Curve Public Key Cryptosystem, Kluwer Academic Publishers, 1993.

[15] Evangelos Kranakis, Primality and Cryptography, B.G.TEUBNER, 1985.

[16] John B. Fraleigh, A First Course in Abstract Algebra, 3rd Edition, Addison-Wesley, 1982.

[17] David M. Bressoud, Factorization and Primality Testing, Springer-Verlag, 1989.

[18] Burt Kaliski, Matt Robshaw, "The Secure Use of RSA", CryptoBytes, Volume 1, number 3, pp.7~13, 1995.

[19] Bruce Dodson, Arjen K. Lenstra, "NFS with Four Large Primes: An Explosive Experiment" Advances in Cryptology, Crypto'95, pp.372~385, 1995.

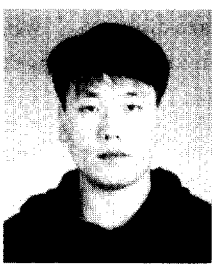
[20] Neal Koblitz, A Course in Number Theory and Cryptography, Springer-Verlag, 1994.

[21] 양대현, 송주석, "타원곡선을 이용한 암호 시스템," 통신정보보호학회지, 제 7권, 제 4호, 1997.12.

[22] J. Buchmann, J. Loho, J. Zayer, "An implementation of the general number field sieve", Advances in Cryptology, Crypto'93, pp.159~165, 1993.

□ 著者紹介

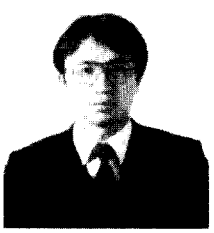
김 진 규



1993년 한남대학교 수학과(학사)
1998년 한남대학교 컴퓨터공학과 석사과정 재학중

※ 주관심분야 : 암호이론, 컴퓨터/네트워크 보안

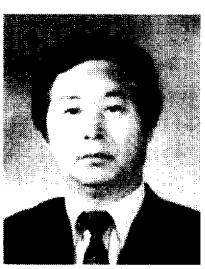
김 영 수



1986년 한남대학교 전자계산공학과(학사)
1990년 한남대학교 수학과(석사)
1986년 - 현재 한국전자통신 연구원 선임 연구원
1998년 7월 - 현재 한국정보보호센터 연구원

※ 주관심분야 :

김 성 옥



연세대학교 수학과(학사)
University of Minnnesota 전자계산학과 (석사)
연세대학교 수학과 (박사)
한남대학교 컴퓨터 공학과 교수

※ 주관심분야 :