Comm. Korean Math. Soc. 9 (1994), No. 1, pp. 241–251

수체 선별법(NFS) 알고리즘 실현에 관한 연구

조인호[1], 임종인[2], 이상진[2]

1. 서론

정수의 소인수 분해 문제는 가장 오래된 수학적 문제의 하나이지만 1987년 개발되었고 현재 가장 안전한 공개키 암호법의 하나인 RSA 공개키 암호법이 안전성을 소인수분해의 어려움에 근거하고 있기 때문에, 갈로아체에서의 이산로그(discrete logarithm) 문제와 더불어 암호학 분야에서 매우 중요한 문제의 하나로서 주목을 받아왔다.

컴퓨터 하드웨어의 발전과 더불어 80년대에 개발된 이차 선별법(QS), 복수다항식 이차선별법(MPQS)[12] 등은 소인수분해가 가능한 합성수의 자릿수를 30자리 정도에서 1990년 100자리 정도까지 끌어올렸다.

1990년 Lenstra[7]는 수체선별법(number field sieve)이라는 인수분해법을 개발하여 1991년 MPQS로서는 불가능했던 RSA 키사이즈인 155자리 합성수인 $F_9 = 2^{512} + 1$ 의 인수분해에 성공하였다[8]. 본 논문은 NFS 알고리즘을 프로그램화 하여 컴퓨터 실행을 한 결과을 수록하고 있다. 여건상 우리는 NFS를 39자리 합성수인 $2^{128} + 1$에 적용하여 $2^{128} + 1 = 59649589127497217 \times 5704689200685129054721$ 라는 결과를 얻었다. 이전의 방법과는 달리 NFS는 수체상의 fundamental units와 prime ideal 의 generator를 찾는 문제가 수반되기 때문에 수학적으로도 더 흥미가 있다.

1992년 Lenstra[16] 등은 일반화된 수체선별법(GNFS)를 발표 하였고, 1993년 여름 Buchman[17] 등은 이에 대한 구현을 발표하였지만 아직까지는 다항식의 선택 문제 등 몇가지 해결점이 남아 있는 것으로 보고되고 있다.

Received April 23, 1993. Revised September 24, 1993.
본 연구는 1992년도 고려대학교 교내 특별연구비 지원에 의한것임.

2. NFS 알고리즘

NFS 알고리즘은 N을 소인수분해 하려는 목표합성수라 할 때 이전의 MPQS에서와 마찬지로 $x^2 \equiv y^2 \mod N$을 만족하는 두정수 x, y를 구하여 $d = \gcd(x \pm y, N)$을 구한 후 $d \neq 1$일때 d를 N의 인수로서 택하는 방법을 사용하고 있다. $d \neq 1$일 확률은 1/2이상이라는 것은 쉽게 알 수 있기 때문에 위와 같은 이차 합동식을 만족하는 정수쌍(x, y)를 10개 정도 찾는다면 우리는 N의 인수를 찾는데 99%이상 성공 했다고 봐야 할 것이다. N의 자리수가 100자리 이상일 경우 MPQS로서는 위와 같은 이차합동식을 만족하는 정수쌍을 충분히 찾는 것이 현실적으로 불가능 하였으나 [10, 12], 이제부터 기술하는 NFS알고리즘을 쓰면 효과적으로 구할 수 있다.

목표합성수 N의 형태가 $N = r^e - s$(r은 작은 크기의 정수, s는 절대값이 작은 정수)라 하자. 이와 같은 수는 Cunningham project수로서 흔히 발견된다.

$d \sim (3 \log N / 2 \log \log N)^{1/3}$인, 정수 d와 $kd \geq e$를 만족하는 최소의 정수 k 를 계산하자.

$r^{kd} \equiv sr^{kd-e} \mod N$이므로 $m = r^k, c = sr^{kd-e}$라 하면 $m^d \equiv c \mod N$이 성립 한다. $f(x) = x^d - c \in \mathbb{Z}[x]$라 하자. $f(x)$가 가약 다항식이라면 $f(x)$을 인수분해하여[5] $f(x) \equiv 0 \mod N$이라는 조건으로부터 gcd를 구하여 N의 인수를 구하는데 성공할 가능성이 크다.

만약 $f(x)$가 가약이고 gcd가 trivial하게 된다면 $f(x)$를 다르게 택하면 된다. 실제로는 대부분의 저계수 다항식이 기약다항식이므로 문제가 발생하는 경우는 거의 없다. 이제부터 $f(x)$를 기약이라 하자.

$f(\alpha) = 0$인 α와 수체$Q(\alpha)$, 그리고 준동형사상

$$\begin{array}{rccc} \varphi: & \mathbb{Z}[\alpha] & \longrightarrow & \mathbb{Z}/N\mathbb{Z} \\ & a & \longmapsto & m \end{array}$$

을 생각하자.

예)$N = 2^{128} + 1, d = 3, k = 43, c = sr^{kd-e} = -2, m = 2^{43}, f(x) = x^3 + 2, Q(\alpha) = Q(\sqrt[3]{-2})$

NFS 알고리즘은 다음의 세조건을 만족하는 정수쌍 (a,b)를 찾아서 N에 대한 관계식을 구하고자 하는 것이다. B_1, B_2를 적당한 bound라 할때 ① $\gcd(a,b)=1$, ② $a+mb : B_1-\text{smooth}$, ③ $\text{Norm}(a+\alpha b)=a^d-c(-b)^d$; B_2-smooth 여기서 B-smooth란 B 이하의 솟수만의 곱으로 인수분해 된다는 것이다.

일반적으로 성립되지 않지만 우리는 $\mathbb{Z}[\alpha]$가 UFD라고 가정 하겠다. 따라서 $Q(\alpha)$의 정수환은 $\mathbb{Z}[\alpha]$가되고 $\mathbb{Z}[\alpha]$는 PID가된다. 이 가정 역시 수학적으로는 틀린 것 이지만, 실제로 NFS알고리즘 실행시 극복 될 수 있는 문제이다. $2^{457}+1$의 인수분해시 $f(x)=x^5+8$이고, 이 경우 $\mathbb{Z}[\alpha]$는 UFD가 되지 않지만 $Q(\alpha)$의 정수환은 UFD가된다[3].

위와 같이 $Q(\alpha)$의 정수환이 UFD가 되지 않더라도 우리의 목표는 아래에서 기술하는 바와 같이 N을 법으로 한 합동 관계식을 구하려는 것이므로 위의 가정에 대한 반례가 되는 경우 발생시 그것을 제외 하면 된다.

p을 솟수, $C_p \in \{0,1,\cdots,p-1\}$을 $f(C_p)\equiv 0 \mod p$라 할때 $\mathbb{Z}[\alpha]$의 prime norm을 가지는 prime ideal들은 $\{p, C_p\}$ 쌍들과 일대일 대응 관계에 있다. 즉 각 $\{p,C_p\}$에 대해서 norm이 p인 prime ideal 은 $(p, \alpha - C_p)$가 된다. 따라서 $a+\alpha b$가 $(p,\alpha-C_p)$에 속하기 위한 필요충분조건은 $a+C_p b \equiv 0 \mod p$인 것이다[14].

이제 위의 세 조건을 만족하는 정수쌍 (a,b)에 대해서

$$|a+mb| = \prod_{\substack{p:\text{솟수} \\ p\le B_1}} p^{e_p} \tag{2.1}$$

$$|N(a+\alpha b)| = \prod_{\substack{p:\text{솟수} \\ p\le B_2}} p^{f_p} \tag{2.2}$$

가 성립한다고 하자($e_p, f_p \in \mathbb{Z} \ge 0$).
가정에서 $\mathbb{Z}[\alpha]$가 PID이므로 prime norm인 prime ideal들에 대해 geneator가 하나씩 존재한다. 이들의 집합을 G라 하고 U를 fundamental unit의 집

합이라 하면 (2.2)로 부터

$$a+\alpha b=\left(\prod_{u\in U}u^{f_u}\right)\left(\prod_{g\in G}g^{f_g}\right) \tag{2.3}$$

을 얻을 수 있다($f_u\in\mathbb{Z}, f_g\in\mathbb{Z}\geq 0$).
따라서 (2.1)과 (2.3)으로 부터 다음과 같은 관계식을 얻을 수 있다.

$$\varphi(a+\alpha b)=\left(\prod_{u\in U}\varphi\left(u^{f_u}\right)\right)\left(\prod_{g\in G}\varphi\left(g^{f_g}\right)\right)=\left(\prod_{\substack{p:\text{숫수}\\ p\leq B_1}}p^{e_p} \mod N\right) \tag{2.4}$$

B_1 이하의 숫수의 갯수를 $\pi(B_1)$이라 할 때, 위와 같은 순서쌍(a,b)를 $\#U+\#G+\pi(B_1)$개 이상 찾아내면 지수에 Gauss소거법을 적용하여 (2.5)를 얻을 수 있다.

G와 U를 구하는 과정은 5절에서 언급하겠다.

$$\left(\left(\prod\varphi\left(u^{e_u}\right)\right)\left(\prod\varphi\left(g^{e_g}\right)\right)\right)^2=\left(\prod p^{e_p}\right)^2 \quad (\mod N) \tag{2.5}$$

식(2.5)를 구하면 앞에서와 같이 gcd를 구하므로써 N을 인수분해 할 수 있다.

NFS의 추정실행시간은 $\exp\left((c+O(1))\right)(\log N)^{1/3}(\log\log N)^{2/3}\big)$, $(c\sim 1.526)$ 으로서 MPQS의 $\exp\left((1+O(1))\right)(\log N)^{1/2}(\log\log N)^{1/2}\big)$ 보다 훨씬 작다.

3. Sparse matrix에서의 Gauss소거법

Sieve method 를 이용하여 sieve bound 안에 있는 prime ideal로 인수분해한 것을 가지고 다음과 같이 행렬을 만든다. 이 행렬은 "0"이 매우 많

은 Sparse matrix가 된다. 각행은 관계식에 대응하는 것이고, 각 열은 sieve bound안에 있는 솟수로 보자. 각 성분은 관계식에서 솟수의 지수를 2로 나눈 나머지를 의미한다. 행렬의 열은 fundamental unit 갯수+솟수의 갯수+prime ideal의 갯수를 더한 것이 될 것이고, 행은 이보다 많을 것이다. sieve bound크기에 따라 행렬의 크기가 결정되며, 본 연구에서 $2^{128}+1$에 대한 NFS적용시의 것은 1863×1568의 크기를 가지고 있다. 행렬에 대해서 Gauss소거법을 시행하면 메모리가 급격히 증가 하므로 소거법이 빠르게 수행되도록 다음과 같은 Wiedemann의 방법[13]을 사용하였다.

Step 1. 각 열중 "0"이 아닌 것의 갯수가 1인 것에 대응하는 행과 열을 버림
(그 이유는, 이 행은 다른 행과 일차독립임)

Step 2. 각 열중 "0"이 아닌 것의 갯수가 최소인 열을 선택

Step 3. 선택한 열에 대응 하는 행중 "0"의 갯수가 최소인 행을 선택

Step 4. 선택한 행을 가지고 Gauss 소거법 시행

Step 5. 일차 종속인 행의 관계식을 구할때까지 Step 1.-Step 4.을 반복수행

이 방법은 일차종속인 행의 관계식을 구하기 위한 것이므로 모든 행과 일차독립인 것으로 판명된 행은 버림으로써 메모리를 줄일 수 있다. Step 2.는 수행시간을 줄이기 위하여 선택했으며, Step 3.은 메모리 증가를 최대한 억제하기 위하여 선택하였음.

4. 기 타

NFS 알고리즘을 수행하기 위하여 필요한 곳이 fundamental unit 와 prime ideal의 generator를 구하기 위한 알고리즘이다. 일반적인 수체에서 이들을 구하는 것은 discriminant가 급격히 커지기 때문에 불가능 하지만 우리의 경우에는 다음과 같이 구할 수 있다.

먼저 수체의 원소에 대한 norm을 계산하는 식을 구한다. norm계산하는 프로그램을 사용하여 적당한 한계치 내에서 전수조사를 통하여 unit을 구한다. unit들 중 크기가 작은 것을 선택하여 fundamental unit인지를 조사 한다.

prime ideal의 generator는 크기를 증가시키면서 norm을 조사하여 구할 수 있다. 앞에서와 같은 쌍 p, C_p 에 대응 되는 generator를 $g(p, C_p)$라 하면, 이것은 $\sum_{i=0}^{d-1} a_i\alpha^i \in \mathbb{Z}[\alpha](a_i \in \mathbb{Z})$로서 norm이 $\pm p$이고 $\sum_{i=0}^{d-1} a_i C_p^i \equiv 0 \mod p$이다.

5. NFS 알고리즘 구현 예

아주 큰 수에 대한 인수분해는 우리나라 실정상 시도하기가 불가능하다. 100자리 이상의 합성수에 대한 인수분해는 수백대의 workstation을 연결시켜 병렬처리 하여도 많은 시간이 소요된다. 예를 들어 138자리수인 $2^{457}+1$의 NFS를 사용한 인수분해는 9주가 소요되었다. 우리의 첫번째 목표는 NFS 알고리즘의 프로그램화 및 실현 성공에 있었으므로 먼저 PC로 수행 가능할 것으로 예측되는 수를 가지고 구현하였다.

- 인수분해 대상 선정 : $N = 2^{128}+1 = r^e - s$ 알고리즘 구성을 위한 기본 상수 계산

  $d = 3, k = 43, c = sr^{kd-e} = -2, m = r^k = 2^{43}$, 다항식 $\Rightarrow f(x) = x^d - c = x^3 + 2$, 사용하는 수체 $\Rightarrow Q(-2^{1/3}), f(\alpha) = 0$, 사용하는 동형사상 $\Rightarrow \varphi : \mathbb{Z}[\alpha] \to \mathbb{Z}/N\mathbb{Z}, \varphi(\alpha) = m$

- 알고리즘 수행을 위한 bound 결정

  $a + mb$의 bound는 $B_1 = 3571$로 하였고 $N(a + \alpha b)$의 bound는 $B_2 = 3557$로 하였다. $\pi(B_1) = 500$이었고 norm이 B_2 이하인 prime norm의 prime ideal의 갯수는 496개였다.

  bound B_1, B_2의 선택은 전체 추정 실행 시간을 최소화시키는 것으로서 Lenstra[7]를 따랐다.

- 알고리즘의 원활한 수행을 위해 자주 쓰이는 데이타를 미리 계산하였다. sieving시 사용하는 데이타인 500개의 솟수 p_i에 대응되는 $m \mod p_i$를 미리 계산 하여 $f_i k$로 보관하였다.

앞에서 언급했듯이 norm이 $\pm p$인 prime ideal 은 $f(C_p) \equiv 0 \mod p$일 때 $(p, \alpha - C_p)$ 형태가 된다. 이때 $f(C_p) \equiv 0 \mod p$를 만족하는 C_p 는 GF(p)에서 $\gcd(x^p - x, f(x))$를 구한후 근을 구하면 쉽게 구할 수 있다.

또한, $(p, \alpha - C_p)$의 generator $g(p, C_p) = g = \sum_{i=0}^{d-1} a_i \alpha^i$ 은 $\sum_{c=0}^{d-1} a_i C_p^i \equiv 0 \mod p$를 만족해야 된다. 또한 norm이 $\pm p$인 것에서 계수가 간단한 것을 선택하였다.

아래의 예에서 보듯 generator g는 계수를 크게 해나가면서 몇 가지 기법을 써서 조사해 나가면 우리의 경우에는 쉽게 구할수 있었다. 그러나, 일반적인 수체의 경우 discriminant가 너무나 커질것으로 예상되기 때문에 지금까지 개발된 방법으로는 generator를 구할 수 없다. Lenstra 등은[16] GNFS에서 Pollard lattice sieve[18]를 사용하여 discriminant를 사용하지 않고 일반형태의 정수에 적용할 수 있도록 확장 하였다.

예) 위에서 설명한 것과 같이 실행하여 우리는 $g_0 = 0 + 1 * \alpha + 0 * \alpha^2$ $(norm(g_0) = -2)$ 부터 $g_{495} = 3 + (-16) * \alpha + (-16) * \alpha^2$ $(norm(g_{495}) = -3557)$까지 496개의 Prim ideal generator를 구할 수 있었다. 우리의 경우에 fundamental unit는 1개 이므로 시행착오법으로

$$u = 1 + 1 * \alpha + 0 * \alpha^2 \ (norm(u) = -1)$$
$$u^{-1} = -1 + 1 * \alpha + (-1) * \alpha^2 \ (norm(u^{-1}) = -1)$$

를 쉽게 구할 수 있었다.

Good pair (a, b)는 다음의 알고리즘 수행 1)에서와 같은 방법으로 찾았다. 예를들어 $a = 4346$, $b = 7$일 경우

$$a + mb = 2 * 31 * 229 * 1091 * 1291 * 3079$$
$$a + \alpha b = u^{-2} * g_0 * g_1 * g_2^2 * g_{62} * g_{115} * g_{370}$$

가 된다. 여기에서

$$g_1 = 1 + 0 * \alpha + (-1) * \alpha^2 \ (norm(g_1) = -3)$$
$$g_2 = 1 + 0 * \alpha + 1 * \alpha^2 \ (norm(g_2) = 5)$$
$$g_{62} = 5 + 3 * \alpha + (-3) * \alpha^2 \ (norm(g_{62}) = -307)$$
$$g_{115} = 1 + (-11) * \alpha + (-10) * \alpha^2 \ (norm(g_{115}) = -677)$$
$$g_{370} = 13 + (-4) * \alpha + (-1) * \alpha^2 \ (norm(g_{370}) = 2633)$$

이다.

— 알고리즘 수행

1) relation pair (a,b)를 찾음 $\gcd(a,b) = 1, a + mb$에 대해서 sieve method를 행함

Step 1. a를 배열로 선언하고 0으로 초기화

Step 2. $mb \equiv -k(\mod prime)$인 k에 해당하는 배열 a에 $\log_2(prime)$를 더함

$mb \equiv -k\ (\mod prime^{power})$인 k에 해당하는 배열 a에 $\log_2(prime^{power})$를 더함

Step 3. 배열 a의 값이 $\log_2(a + mb)$와 같은 값을 갖는 (a,b)를 선택

Step 4. $a + \alpha b$에 대응하는 별도의 sieve method를 사용하지 않고 선택된 (a,b)가 인수분해 되는지 시행착오 방법으로 찾음

2) 찾은 관계식을 가지고 $x^2 \equiv y^2(\mod N)$이 되는 (x,y)를 구함

행렬을 unit, prime ideal generator, 숫수에 대응하는 지수들을 2로 나눈 나머지로 구성하면 이러한 행렬은 영의 갯수가 매우 많은 sparse행렬이 된다. 이 sparse행렬의 각 행에 대한 일차 종속이 되는 관계식을 구하기 위해서는 Gauss소거법을 사용한다[13].

3) N을 인수분해 함

– 수행결과

1) sieve method를 사용한 결과 1863개의 관계식을 구하였다. 수행시간 : 2일, full-full relation의 개수 : 701 , full-partial relation의 개수 : 1162 (10,000 이하의 large prime을 허용)

2) sparse행렬을 구축 : 크기 : 1863 by 1568

가우스 소거법 수행 : 3시간 정도 소요되며 계속적인 memory의 증가 (1M 이상) 로 인하여 Hard Disk를 사용하였음. 300개의 일차 종속인 행의 관계식을 구함

3) 인수분해를 시도한 결과 : 147개의 관계식이 성공함

- 사용한 컴퓨터 : 386DX, Ram : 4M, OS : window 3.0
- 프로그래밍 언어 : Turbo C^{++} 3.0

6. 참고사항

(1) D. Coppersmith[3]는 1991년 NFS를 일반적인 형태의 합성수에 적용할 수 있도록 확장하였다. 추정 실행시간은 $\exp((1.902+O(1))(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{1}{3}})$인 알고리즘이지만 일반적인 수체에서의 연산, prime ideal의 generator 및 fundamental unit을 찾는데 있어서의 어려움 때문에 당분간 컴퓨터 실현은 어려울 것으로 생각된다.

(2) Lenstra[16]등이 1992년 발표한 GNFS는 Coppersmith의 방법과 달리 Pollard lattice sieve 법을 사용하여 Prime ideal의 generator를 찾는 문제, fundamental unit를 찾는 문제 등을 피하였다. 그러나 nice polynominal를 찾는 일반적인 방법이 제시되고 있지않다. 1993년 8월 Crypto '93에서 [17] 등은 GNFS를 사용하여 49자리 합성수를 인수분해 하였다고 발표하였다.

(3) NFS의 확장방안은 위의 Coppersmith방식으로 나아갈 경우 상당기간 실현이 어려울것으로 예상되므로 우리 팀은 현재 이차 다항식을 여러개 선택하여 이차체를 여러개 선택함으로써 관계식을 만들어 내는 방법을 시도하고 있다. 이것은 MPQS및 Coppersmith방법에서 idea를 얻은것으로서 완성시 효과가 클것으로 예상된다.

(4) 본 알고리즘의 실현시 효율을 높이기 위하여 Large prime variation[8]을 사용하였다. 이 방법은 큰 솟수 p가 인수로 나타날때 이것을 버리지 않고 이용하는 것으로서 $x^2 \mod n$이 p^2을 인수로 가지면 $\gcd(p,n)=1$을 확인한 후 $(p^{-1} \mod n)^2$을 곱하여 $x^2 \mod n$으로부터 제곱성을 파괴하지 않고 큰 솟수 p^2을 제거함으로서 B-smooth가 되게 하는 것이다. 우리의 경우는 10000이하의 큰 솟수를 사용하였다.

7. 참고문헌

1. J. Brillhart, D. H. Lehmer, J. I. Selfridge, B. Tuckerman, S. S. Wagstaff, Jr., *Factorizations of* $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ *up to high powers*, Second edition, Contemp. Math. AMS, **22** (1988).
2. T. R. Caron, R. D. Silverman, *Parallel implementation of the quadratic sieve*, J. Supercomputing **1** (1988), 273-290.
3. Don. Coppersmith, *Modification to the NFS*, IBM Reaserch Report #RC16264, 1991.
4. J. A. Davis, D. B. Holdridge, *Most wanted factorizations using the quadratic sieve*, Sandia. Tech. Report SAND84-1658UC-32(August), 1984.
5. D. E. Knuth, *The art of computer programming*, Seminumerical algorithms, Second edition, Addison-Wesley, 1981.
6. A. K. Lenstra, H. W. Lenstra, Jr., *Algorithms in number theory, Handbook of theoretical computer science*, North- Holland, Amsterdam, New York, Oxford, 1992.
7. A. K. Lenstra, H. W. Lenstra, Jr. M. S. Manasse, J. M. Pollard, *The number field sieve*, Comm. ACM. (1990), 564-572.
8. A. K. Lenstra, M. S. Manasse, *Factoring with two large primes*, Preprint, 1992.
9. C. Pomerance, *The quadratic sieve factoring algorithm*, crypt'84, Lecture Notes in Comp. Sci. Springer-Verlag **209** (1985), 169- 182.
10. H. J. J. te Riesel, W. M. Lioen, D. T. Winter, *Factoring with the quadratic sieve on large vector computers*, J. Comp. Appl. Math. **27** (1989), 267-278.
11. R. Rivest, A. Shamir and L. M. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Comm. ACM **21** (1978), 120-128.

12. R. D. Silverman, *The multiple polynomial quadratic sieve*, Math. Comp. **48** (1987), 329-339.
13. D. H. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Trans. Inform. Theory **IT32** (1986), 54-62.
14. 조인호, 임종인, *NFS*알고리즘 실현에 관한 연구, ETRI 연구보고서, 1992.
15. 조인호, 임종인, *MPQS*알고리즘 실현에 관한 연구, ETRI 연구보고서, 1991.
16. J. P. Bubler, H. W. Lenstra , C. Pomerance, *Factoring integers with the number field sieve*, Preprint, 1992.
17. J. Buchman, J. Loho, J. Zayer, *An implementation of the general number field sieve*, Crypto '93, 1993.
18. J. M. Pollard, *The lattice sieve*, Draft, september, 1991.

서울시 성북구 안암동 5가 1번지(136-701), 고려대학교 이과대학 수학과[1]
충남 연기군 조치원읍 (339-800), 고려대학교 자연과학대학 수학과[2]