논문 2011-2-16

$n+1$ 소인수분해 알고리즘

The $n+1$ Integer Factorization Algorithm

최명복*, 이상운**

Myeong-Bok Choi, Sang-Un Lee

요 약 $n=pq$인 합성수 n을 크기가 비슷한 p와 q로 소인수분해하는 것은 매우 어려운 문제이다. 대부분의 소인수분해 알고리즘은 $a^2 \equiv b^2 \pmod n$인 제곱 합동이 되는 (a,b)를 소수의 곱 (인자 기준, factor base, B)으로 찾아 $a^2-b^2=(a-b)(a+b)$ 공식에 의거 유클리드의 최대공약수 공식을 적용하여 $p=GCD(a-b,n)$, $q=GCD(a+b,n)$으로 구한다. 여기서 (a,b)를 얼마나 빨리 찾는가에 알고리즘들의 차이가 있으며, B를 결정하는 어려움이 있다. 본 논문은 좀 더 효율적인 알고리즘을 제안한다. 제안된 알고리즘에서는 $n+1$을 3자리 소수까지 소인수분해하여 B를 추출하고 B의 조합 f를 결정한다. 다음으로, $a=fxy$가 되는 값을 $\sqrt{n}<a<\sqrt{2n}$ 범위에서 구하여 $n-2$의 소인수분해로 x를 얻고, $y=\frac{a}{fx}$, $y_1=\{1,3,7,9\}$을 구한다. 제안된 알고리즘을 몇 가지 사례에 적용한 결과 $\sqrt{n}<a$를 순차적으로 찾는 기존의 페르마 알고리즘에 비해 수행 속도를 현격히 단축시키는 효과를 얻었다.

Abstract It is very difficult to factorize composite number, $n=pq$ to integer factorization, p and q that is almost similar length of digits. Integer factorization algorithms, for the most part, find (a,b) that is congruence of squares ($a^2 \equiv b^2 \pmod n$) with using factoring(factor base, B) and get the result, $p=GCD(a-b,n), q=GCD(a+b,n)$ with taking the greatest common divisor of Euclid based on the formula $a^2-b^2=(a-b)(a+b)$. The efficiency of these algorithms hangs on finding (a,b) and deciding factor base, B. This paper proposes a efficient algorithm. The proposed algorithm extracts B from integer factorization with 3 digits prime numbers of $n+1$ and decides f, the combination of B. And then it obtains x(this is, $a=fxy$, $\sqrt{n}<a<\sqrt{2n}$) from integer factorization of $n-2$ and gets $y=\frac{a}{fx}$, $y_1=\{1,3,7,9\}$. Our algorithm is much more effective in comparison with the conventional Fermat algorithm that sequentially finds $\sqrt{n}<a$.

Key Words : 소수 (prime number), 합성수 (composite number), 소인수 분해 (integer factorization), 제곱합동 (congruence of squares), 인자 기준 (factor base)

Ⅰ. 서 론

RSA 암호체계 (cryptograph)는 큰 자리수의 숫자 n을 p와 q로 소인수분해 (integer factorization)가 어렵다는 이론에 기반을 두고 있다.[1] RSA 암호를 해독하기 위해서는 $n=pq$인 합성수 (composite number) n을 2개 소수 p,q로 소인수분해 해야 한다.[2,3] 여기서, p와 q는 크기 (자리수)가 비슷한 큰 소수들이 사용되고 있으며 각각을 소수 인자 (prime factor)라 한다. n이 공개되므로 p만 결정하면 $q=n/p$로 계산될 수 있으나 p를 한 번에 결정하는 방법이 없다. 따라서 소인수분해의 모든 알고

*종신회원, 강릉원주대학교 멀티미디어공학과
**정회원, 강릉원주대학교 멀티미디어공학과
접수일자: 2011.3.13, 수정일자: 2011.4.13
게재확정일자: 2011.4.15

리즘은 p를 얼마나 빨리 찾을 수 있는가에 초점을 맞추고 있다.

소인수분해 방법에는 나눗셈 시행 (Trial Division), Pollard의 rho 알고리즘, Pollad의 p-1 알고리즘, William의 p+1 알고리즘, Lenstra elliptic curve 인수분해, 페르마 (Fermat)의 인수분해 방법, 오일러 (Euler)의 인수분해 방법, Special Number Field Sieve, 2차 (Quadratic, Q), MPQ (Multiple-polynomial quadratic), NF (Number field), GNF (General number field), Dixon, CFRAC (continued fraction factorization), SQUFOF (Shanks' square forms factorization)과 양자 컴퓨터를 활용한 Shor 알고리즘이 있으며, 대부분은 $a^2 \equiv b^2 (\mathrm{mod}\, n)$의 제곱합동 (congruence of squares)에 기반하고 있다.[2]

100자리 십진수에 대해 1991년에 다중 다항식 2차 체 (MPQ Sieve) 알고리즘으로 4시간 만에 해독한 이래 지금까지 RSA-200까지 해독되었다. RSA-210부터 RSA-2048 (607)까지는 아직 해독되지 않고 있으며, RSA-704 (212 자리)는 $30,000, RSA-768 (232)는 $50,000, RSA-896 (270)은 $75,000, RSA-1024 (309)는 $100,000, RSA-1536 (463)은 $150,000, RSA-2048 (617)는 $200,000로, 총 $605,000의 상금이 걸려 있었으나 2007년 RSA Factoring Challenge가 활동을 중단한 이후 남아있는 상금은 취소된 상태이다.[3] 즉, 현재까지 알려진 알고리즘으로는 더 이상 풀 수 없는 문제임을 알 수 있다.

본 논문은 $a^2 \equiv b^2 (\mathrm{mod}\, n)$의 제곱합동을 보다 빠르게 찾는 방법을 제안한다. 2장에서는 제곱합동 소인수 분해 알고리즘을 고찰해 본다. 3장에서는 제곱합동을 단순히 $n+1$을 소인수분해하여 빠르게 찾는 알고리즘을 제안한다. 4장에서는 제안된 알고리즘의 적용성을 평가해 본다.

Ⅱ. 제곱합동 소인수분해 방법

가장 단순한 나눗셈 시행 방법과 Shor 알고리즘을 제외한 대부분의 소인수분해 알고리즘은 페르마 알고리즘[4]에 기반을 두고 있다. 제곱합동을 찾는 알고리즘들은 항상 $a^2 - b^2 = n$이 되는 (a,b)를 찾는다. Dixon[5]과 2차 체[6], GNFS[7] 등의 방법은 페르마 알고리즘이 a를 순차적으로 찾는 방법을 개선한 형태이다. 페르마 알고리즘은 데이터 수집 단계 (data collection phase)와 데이터 처리 단계 (data processing phase)를 수행한다. 데이터 수집 단계에서는 제곱 합동인 $a^2 \equiv b^2 (\mathrm{mod}\, n)$, $a^2 - b^2 \equiv 0 (\mathrm{mod}\, n)$, $a^2 - b^2 = n$을 만족하는 a와 b를 찾기 위해 $\sqrt{n} < a$에서 1씩 증가시키면서 a를 찾는다. 데이터 처리 단계에서는 데이터 수집 단계에서 획득한 a,b에 대해 $a^2 - b^2 = (a+b)(a-b)$공식에 기반하여 유클리드 (Euclid)의 최대 공약수 (Great Common Divisor, GCD) 법칙을 적용하여 $p = GCD(a-b, n)$, $q = GCD(a+b, n)$을 결정한다. 실제로 a 값을 랜덤하게 선택하여 제곱 합동을 찾는 것은 비현실적으로 많은 시간이 소요된다. 따라서 대안으로 n보다 작은 몇 개만을 찾는다.

$n = 18,206,927$에 대해, 페르마 알고리즘은 $\sqrt{n} < a$인 $[4267, 9999]$에서 순차적으로 찾은 결과 제곱 합동 $5676^2 (mod\, 18206927) = 14010049 = 3743^2$을 얻었다. $n = 6,012,707$, $\sqrt{n} = 2452.08$의 경우, 페르마 알고리즘은 $\sqrt{n} < a$인 $[2453, 9999]$에서 2번째인 $2454^2 (\mathrm{mod}\, 6012707) = 9409 = 97^2$을 얻었다.

Dixon 알고리즘[5]은 데이터 수집 단계에서 소수의 집합을 어떤 B보다 작은 것으로 선정한다. 이 소수의 집합을 인자 기준 (factor base, FB)이라 한다. 다음으로 다항식 $p(a) = a^2 (\mathrm{mod}\, n)$이 되는 a의 많은 값들을 시험하여 $(a, p(a))$를 저장한다. 이를 관계 (relation)라 한다. 일단 관계의 수가 인자 기준의 크기를 초과하면 데이터 처리 단계를 수행한다.

$n = 84,923$, $\sqrt{n} = 291.46$의 경우, Dixon 알고리즘은 $B = 10$인 $FB = \{2,3,5,7\}$로 설정하고, 랜덤하게 찾은 결과 $513^2 (\mathrm{mod}\, 84923) = 8400 = 2^4 \times 3 \times 5^2 \times 7^1$, $537^2 (\mathrm{mod}\, 84923) = 33600 = 2^6 \times 3^1 \times 5^2 \times 7^1$을 얻었다. $(513 \times 537)^2 (\mathrm{mod}\, 84923) = 2^{10} \times 3^2 \times 5^4 \times 7^2$로 $(513 \times 537) = 20,712 (\mathrm{mod}\, 84923) = 20712^2 (\mathrm{mod}\, 84923) = (2^5 \times 3 \times 5^2 \times 7)^2 (\mathrm{mod}\, 84923) = 16800^2 (\mathrm{mod}\, 84923)$이 성립한다. $a^2 - b^2 = (a+b)(a-b)$에 의거 $20712 - 16800 = 3912$와 $20712 + 16800 = 37512$을 얻는다.

2차 (Quadratic, Q) 체 알고리즘[6]은 Dixon 알고리즘의 데이터 수집 단계를 개선한 것으로 다수의 a에 대해 $a^2 (\mathrm{mod}\, n)$을 계산하고 이들 값의 곱이 제곱이 되는 것들을 제곱 합동으로 결정한다.

$n=1{,}649, \sqrt{n}=40.61$인 경우, 2차 체 알고리즘은 $\sqrt{n}<a$인 $a=41,42,43$을 선택한다. $41^2 \pmod{1649} = 32$, $42^2 \pmod{1649} = 115$, $43^2 \pmod{1649} = 200$ 모두 제곱이 되지 않는다. 그러나 이들의 곱인 $(32)(200)=6400=80^2$으로 제곱의 합동이 된다. 즉, $(41\times43) \pmod{1649} = 114, 114^2 \equiv 80^2 \pmod{1649}$로 $a^2 \equiv b^2 \pmod{n}$을 얻는다.

체 방법에는 Q, MPQ (Multiple-polynomial quadratic), NF (Number field), GNF (General number field) 등 다양한 방법들이 있다. GNFS[7]는 100자리보다 큰 정수를 소인수분해하는 가장 효율적인 알고리즘으로 알려져 있으며, Rational Sieve[8]의 확장 형태이다. $n=35$에 대해 Rational Sieve는 다음과 같이 적용된다. 먼저, 임의로 $B=19$로 설정하여 인자 기준 $FB=\{2,3,11,13,17,19\}$를 얻는다. 여기서 5,7은 35의 약수로 포함되지 않는다. 다음으로 z와 $z+n$이 B-smooth (그들의 모든 소인수가 $\le B$인 경우)인 양의 정수 z를 찾는다. 즉, 처음 몇 개의 z를 찾으면 1, 3, 4, 9, 13, 19, 22이다. 각각에 대해 곱셈 관계 (mod 35)를 계산하면 다음과 같다.

$$2^0 3^0 11^0 13^0 19^0 = 1 \equiv 36 = 2^2 3^2 11^0 13^0 19^0 \quad ①$$
$$2^0 3^1 11^0 13^0 19^0 = 3 \equiv 38 = 2^1 3^0 11^0 13^0 19^1 \quad ②$$
$$2^2 3^0 11^0 13^0 19^0 = 4 \equiv 39 = 2^0 3^1 11^0 13^1 19^0 \quad ③$$
$$2^0 3^2 11^0 13^0 19^0 = 9 \equiv 44 = 2^2 3^0 11^1 13^0 19^0 \quad ④$$
$$2^0 3^0 11^0 13^1 19^0 = 13 \equiv 48 = 2^4 3^1 11^0 13^0 19^0 \quad ⑤$$
$$2^0 3^0 11^0 13^0 19^1 = 19 \equiv 54 = 2^1 3^3 11^0 13^0 19^0 \quad ⑥$$
$$2^1 3^0 11^1 13^0 19^0 = 22 \equiv 57 = 2^0 3^1 11^0 13^0 19^1 \quad ⑦$$

이들 7개에 대해 다른 방법으로 결합시켜 지수항이 짝수가 되는 ②×④×⑦인 $2^1 3^3 11^1 13^0 19^0 \equiv 2^3 3^1 11^1 13^0 19^2$로 결정하고 공통 인자를 삭제하면 $3^{3-1} \equiv 2^{3-1} 19^{2-0} = 3^2 \equiv 2^2 19^2$으로 $3^2 \equiv 38^2$을 얻는다. 따라서 $p=GCD(38-3,35)=35, q=GCD(38+3,35)=1$이 된다. a를 찾는 것이 쉽지 않음을 표 1에서 고찰해 보자. 표 1의 일부 데이터는 김승주[9]와 Jensen[10]의 pGNFS에서 인용되었다.

표 1. 제곱 합동
Table 1. Congruence of Squares

n	$\sqrt{n}<a<\sqrt{2n}$ $a^2 \pmod{n} = b^2$	p,q
18,206,927	$a=5676, b=3743$	$p=1933, q=9419$
6,012,707	$a=2454, b=97$	$p=2357, q=2551$
112,729	$a=475, b=336$	$p=139, q=811$
84,923	$a=342, b=179$	$p=163, q=521$
2,352,854,039	$a=49008$, $b=6995$	$p=42013$, $q=56003$
8,229,944,909,131,434,961	$a=2,925,355,081$ $b=572,501,040$	$p=2,352,854,041$ $q=3,497,856,121$
982,301,348,481,613,682,763,349,336,546,115,836,409	$a=33,894,332,773,738,340,605$ $b=1,904,435,117,249,313,796$	$p=20,989,897,656,489,026,809$ $q=46,798,767,890,987,654,401$

예로, $n=18{,}206{,}927$은 $\sqrt{n}<a\le {}_{\max}l(\sqrt{n})$인 $4267 \le a \le 9999$의 5,733개 중 제곱합동이 되는 a는 5676, 7609와 9542의 3개 뿐이다. 이들 값 중에서 항상 $\sqrt{n}<a<\sqrt{2n}$의 1개만을 찾는 것은 어려움이 있다. 또한, a에 대한 소인수 분해의 공통점이 없다. 예로, $5676=2^2\times3^1\times11^1\times43$, $2454=2^1\times3^1\times409$, $342=2^1\times3^2\times19$, $49008=2^4\times3^1\times19^1\times59$, $2925355081=11^1\times439^1\times605789$, $338943327737 38340605=5^1\times13^1\times101^1\times5162883895466617$로 소인수분해 된다. 이들 제곱합동 알고리즘을 적용하기 위해서는 FB를 설정하는 문제와 제곱합동이 되는 a,b를 결정하는 문제가 제기된다. 따라서 3장에서는 a를 보다 쉽게 찾는 방법을 제안한다.

Ⅲ. $n+1$ 소인수분해 알고리즘

기존의 제곱합동 알고리즘 단점은 FB를 선정하는 어려움과 a,b를 찾는 어려움을 동시에 갖고 있다. 본 장에서는 제곱합동이 되는 a,b를 특이하게 구하는 방법을 제안한다. 주어진 합성수 n은 p와 q를 제외한 어떠한 소수로도 인수분해가 되지 않는다. 반면에, $n+1$과 a는 합성수임에도 불구하고 소인수분해가 된다. 또한, n, p와 q의 1자리는 1,3,7 또는 9이다. 이러한 특징에 기반하여 알고리즘을 다음과 같이 제안한다.

(1) $n+1$을 3자리 소수까지로 소인수분해하여 $B \leq 19$로 한정시킨 정준분해 (canonical decomposition) $B=2^i3^j5^17^1\cdots19^1$을 선택한다. $B<C$인 2자리 또는 3자리 소수를 C라 한다. 만약, $C=\{\phi\}$이면 $C=97$로 결정한다.

(2) B를 다음 기준으로 수정한다.

if $i=even$ then 2^i의 $i=i/2$로 대체

if $i/2=even$ then 3^j의 $j=1$로 대체

else if $i=odd\geq 3$ then 2^i의 $i=i+1$로 대체

else if $i=1$ then 2 삭제.

$B_{\max}$ = B의 최대 소수 값.

(3) B 소수의 조합을 f라 하면, $a=fz$가 된다. 또한, z가 소인수분해되어 $z=xy$, $a=fxy$, $y=\frac{a}{fx}$이다. 따라서 y 값에 대해 a가 $a^2\equiv b^2 \pmod n$을 만족하는 a와 b를 구한다.

if $f\neq 5$ then $n-2/x, (B_{\max}<x<C)$ 계산

if $x, (x\neq 5)$ 소수 존재 then 첫 번째 x 선택

else if x 미 존재 then

if $C<97$ then $100<x<3C$ 선택

else if $194<C<291$ then $C<x<2C$ 선택.

if x 미 존재 then $x=1$.

$\lceil \sqrt{n} \rceil < a < \lfloor \sqrt{2n} \rfloor$에서 $y=\frac{a}{fx}$, $y_1=\{1,3,7,9\}$인 y 탐색.

else if $f=5$ then $\lfloor \sqrt{n} \rfloor \leq a \leq \lceil \sqrt{2n} \rceil$에서 $x=5$로 결정, $y=\frac{a}{fx}$, $y_1=\{1,3,7,9\}$인 y 탐색.

(4) $a^2-b^2=(a-b)(a+b)$와 유클리드의 최대공약수 알고리즘을 적용하여 $p=GCD(a-b,n)$, $q=GCD(a+b,n)$을 구한다.

제안된 알고리즘은 인자 기준 B를 $n+1$을 소인수분해하여 얻기 때문에 "$n+1$ 소인수분해 알고리즘"이라 칭한다.

Ⅳ. 알고리즘 적용 결과 분석

표 1의 데이터에 대해 제안된 $n+1$ 소인수분해 알고리즘을 적용하여 보자.

$n=18{,}206{,}927$인 경우, $n+1=2^4\times3^2\times59^1\times2143$으로 $B=2^4\times3^2$, $B_{\max}=3$, $C=59$로 결정된다. $i=even$으로 $2^2\times3^2$으로 교체되고, 다시 $i/2=even$으로 $f=2^2\times3^1=12$, $a=12z$이다. $n-2$를 $3<x<59$, $(x\neq5)$로 나눌 수 있는 첫 번째 숫자는 11이다. 따라서 $x=11$로 결정된다. 즉, $a=12\times11\times y$이다. $\sqrt{n}=4266<a<\sqrt{2n}=6034$에 대해 $32.63<y<45.54$이며, $y_1=\{1,3,7,9\}$인 값들은 $33\leq y\leq43$인 33, 37, 41, 43의 4개가 존재한다. 이들 중 $a=132y$가 $a^2\equiv b^2 \pmod n$을 만족하는 것은 $y=43$인 $a=5676, b=3743$으로 알고리즘은 4회 수행되고 $p=1933$, $q=9419$를 얻는다.

$n=6{,}012{,}707$의 경우, $n+1=6012708=2^2\times3^1\times13^1\times38543$으로 $B=2^2\times3^1\times13^1$, $B_{\max}=13$, $C=97$로 결정된다. $i=even$이므로 $2^1\times3^1\times13^1$으로 대체되며, 이들 소수의 조합은 $f_1=2^1\times3^1=6$, $f_2=2^1\times13^1=26$과 $f_3=3^1\times13^1=39$가 된다. $n-2$를 $13<x<97, (x\neq5)$의 소수로 나눌 수 없어 $x=1$로 결정된다. $\sqrt{n}=2452<a<\sqrt{2n}=3467$에 대해 $409\leq y_6\leq573$, $97\leq y_{26}\leq133$, $63\leq y_{39}\leq87$을 얻는다. $y_1=\{1,3,7,9\}$인 값들에 대해 $a^2\equiv b^2 \pmod n$이 되는 a,b를 구하면, $6y$에서 $y=409$인 $a=2454$, $b=97$을 얻는다. 결국, $p=2357$ $q=2551$이다. 즉, 알고리즘은 $6y$에서 1회 수행된다.

$n=112{,}729$의 경우, $n+1=112{,}730=2^1\times5^1\times11{,}273$으로 $B=2^1\times5^1$을 추출한다. 2^i에서 $i=1$로 삭제되어 $f=5^1=5$로 $x=5$이다. 결국, $a=25y$가 된다. $\sqrt{n}=335.75<a<\sqrt{2n}=474.82$에서 $335\leq a\leq475$, $13.40<y<19.00$에 대해 $y_1=\{1,3,7,9\}$는 $17\leq y\leq19$로 17과 19의 2개가 된다. $a^2\equiv b^2 \pmod n$이 되는 a,b는 $y=19$에서 $a=475$, $b=336$을 얻는다. 즉, 알고리즘은 2회 수행된다. 결국, $p=139$, $q=811$을 얻는다.

$n=84{,}923$의 경우, $n+1=84924=2^2\times3^2\times7^1\times337$로 $B=2^2\times3^2\times7^1$을 추출한다. 2^i에서 $i=even$으로 2^1이 되어 B는 $2^1\times3^2\times7^1$, $B_{\max}=7$, $C=337$로 결정된다. 이들 소수의 조합은 $f_1=2^1\times3^2=18$,

$f_2 = 2^1 \times 7^1 = 14$, $f_2 = 3^2 \times 7^1 = 63$이 된다. $n-2$를 $7 < x < 337$로 나눌 수 없고, $C > 291$로 $x = 1$로 결정된다. $\sqrt{n} = 291.416 < a < \sqrt{2n} = 412.124$에 대해 $y_1 = \{1,3,7,9\}$인 값은 $17 \le y_{18} \le 21$을, $21 \le y_{14} \le 29$, $4 < y_{63} < 6$이다. 그러나 $4 < y_{63} < 6$에는 $y_1 = \{1,3,7,9\}$ 값이 존재하지 않아 $63y$는 생략된다. $18y$와 $14y$에 대해 $y_1 = \{1,3,7,9\}$ 값을 탐색하여 $a^2 \equiv b^2 \pmod n$이 되는 a,b를 구한 결과 $18y$의 $y = 19$에서 $a = 342$, $b = 179$를 얻는다. 즉, 알고리즘은 2회 수행된다. 결국, $p = 163$, $q = 521$을 얻는다.

$n = 2,352,854,039$에 대해, $n+1 = 2^3 \times 3^1 \times 5^1 \times 19607117$로 $B = 2^3 \times 3^1 \times 5^1$, $B_{\max} = 5$, $C = 97$로 결정된다. $i = odd \ge 3$으로 $B = 2^4 \times 3^1 \times 5^1$로 대체된다. 따라서 $f_1 = 2^4 \times 3^1 = 48$, $f_2 = 2^4 \times 5^1 = 80$, $f_3 = 3^1 \times 5^1 = 15$이다. $n-2$를 $5 < x < 97$로 나눌 수 있는 첫 번째 소수는 19로 $x = 19$가 선택되어 $a_1 = 912y$, $a_2 = 1520y$, $a_3 = 285y$가 된다. $\sqrt{n} = 48506 < a < \sqrt{2n} = 68598$에 대해 $y_1 = \{1,3,7,9\}$는 $57 \le y_{912} \le 73$, $33 \le y_{1520} \le 43$, $173 \le y_{285} \le 239$이다. $a^2 \equiv b^2 \pmod n$는 $912y$의 $y = 59$인 $a = 49008$, $b = 6995$이다. 즉, 알고리즘은 2회 수행된다. 결국, $p = 42013$, $q = 56003$이다.

$n = 8,229,944,909,131,434,961$에 대해, $n+1 = 2^1 \times 11^1 \times 271^1 \times 1380400018304501$로 2^i의 $i = 1$로 2를 삭제하면 $f = 11$을 얻는다. $B_{\max} = 11$, $C = 271$이다. $n-2$가 $11 < x < 271$의 소수로 나누어지지 않아 $C < x < 2C$인 $x = 277$, 281, 283, …, 541을 선택한다. $\sqrt{n} = 2868788055 < a < \sqrt{2n} = 4057078976$에 대해 $260798917 \le f_{11} \le 368825361$로 $y_1 = \{1,3,7,9\}$ 탐색 결과 $x = 439$, $y = 605,789$를 얻었다. 즉, 알고리즘은 $x = 439$에 대한 $594,077 \le y_{4829} \le 840,147$에서 4,686회 수행되어, $a = 2,925,355,081$, $b = 572,501,040$를 얻는다. 결국, $p = 2,352,854,041$, $q = 3,497,856,121$이다.

$n = 982,301,348,481,613,682,763,349,336,546,115,836,409$에 대해, $n+1 = 2^1 \times 3^1 \times 5^1 \times 13^1 \times 41^1 \times 61432229422364833193435241597630759$로 2^i의 $i = 1$로 2를 삭제하면 $B = 3^1 \times 5^1 \times 13^1$, $B_{\max} = 13$, $C = 41$이다. 소수의 조합은 $f_1 = 3^1 \times 5^1 = 15$, $f_2 = 3^1 \times 13^1 = 39$, $f_3 = 5^1 \times 13^1 = 65$이다. $n-2$가 $13 < x < 41$의 소수로 나누어지지 않아 $100 < x < 3C$인 $x = 101$, 103, 107, 109, 113이 된다. $\sqrt{n} = 3.13416870714 \times 10^{19} < a < \sqrt{2n} = 4.4323838924 \times 10^{19}$로부터 $8.03633001 \times 10^{17} \le f_{39} \le 1.13650869 \times 10^{18}$, $2.08944580 \times 10^{18} \le f_{15} \le 2.95492255 \times 10^{18}$, $4.82179801 \times 10^{17} \le f_{65} \le 6.81905214 \times 10^{17}$에 대해 $y_1 = \{1,3,7,9\}$을 탐색 결과 $f = 65$, $x = 101$에서 $y = 5,162,883,895,466,617$를 얻어, $a = 33,894,332,773,738,340,605$, $b = 1,904,435,117,249,313,796$를 구하였다. 즉, 알고리즘은 $4.774057437 \times 10^{15} < 6565y < 6.751536774 \times 10^{15}$에 서 1.55531×10^{14}회 수행되었다. 결국, $p = 20,989,897,656,489,026,809$와 $q = 46,798,767,890,987,654,401$이다.

제안된 $n+1$ 알고리즘은 표 2와 같이 $\sqrt{n} < a$에서 순차적으로 $a^2 \equiv b^2 \pmod n$이 되는 제곱 합동 a,b를 찾는 페르마 알고리즘에 비해 수행 속도를 현격히 줄이는 효과를 얻었다.

표 2. 알고리즘 수행횟수 비교
Table 2. Compare to Algorithm's Runtime

n	페르마 알고리즘	제안 알고리즘
18,206,927	1,410회	4회
6,012,707	2회	1회
112,729	140회	2회
84,923	51회	2회
2,352,854,039	482회	2회
8,229,944,909,131,434,961	56,567,026회	4,686회
982,301,348,481,613,682,763,349,336,546,115,836,409	2.55265×10^{18} 회	1.55531×10^{14} 회

V. 결 론

본 논문은 $a^2 \equiv b^2 \pmod n$의 제곱합동을 빠르게 찾는 알고리즘을 제안하였다. 제안된 알고리즘은 합성수 n은 소인수분해가 되지 않지만 합성수 a는 소인수 분해가 된다는 특징을 이용하였다. 즉, a는 소수들의 곱인 정준분해로 표기되며, 이들 소수들 중에서 $B \le 19$을 인자기준 B로 결정하였다.

제안된 알고리즘은 인자기준 B를 $n+1$을 소인수분

해하여 얻는다. 다음으로 B 값에 특정 기준을 적용하여 f 값을 얻고, $\sqrt{n} < a < \sqrt{2n}$ 의 범위에서 $a = fxy$가 되는 x는 $n-2$를 소인수분해하여 얻었다. 최종적으로 $y = \frac{a}{fx}, y_1 = \{1,3,7,9\}$인 값들 중에서 $a^2 \equiv b^2 (\mathrm{mod}\, n)$이 되는 y를 구하였다.

결국 제안된 $n+1$ 소인수분해 알고리즘을 몇 가지 사례에 적용한 결과 $\sqrt{n} < a$를 순차적으로 찾는 페르마 알고리즘에 비해 수행 횟수를 획기적으로 축소시킬 수 있었다.

참 고 문 헌

[1] Wikipedia, *"RSA,"* http://en.wikipedia.org/wiki/Rsa, 2010.

[2] Wikipedia,*"Integer Factorization,"* http://en.wikipedia.org/wiki/Integer_factorization, 2010.

[3] Wikipedia, *"RSA Factoring Challenge,"* http://en.wikipedia.org/wiki/RSA_Factoring_Challenge, 2010.

[4] Wikipedia, *"Fermat's Factorization Method,"* *http://en.wikipedia.org/wiki/Fermat's_factorization_method*, 2010.

[5] Wikipedia, *"Dixon's Factorization Method,"* http://en.wikipedia.org/wiki/Dixon%27s_factorization_method, 2010.

[6] Wikipedia, *"Quadratic Sieve,"* http://en.wikipedia.org/wiki/Quadratic_sieve, 2010.

[7] Wikipedia, *"General Number Field Sieve,"* http://en.wikipedia.org/wiki/General_number_field_sieve, 2010.

[8] Wikipedia, *"Rational Sieve,"* http://en.wikipedia.org/wiki/Rational_sieve, 2010.

[9] 김승주, *"공개키 암호 방식,"* School of Information and Communication Engineering, Sungkyunkwan University, http://dosan.skku.or.kr/~sjkim/Lecture Notes/SKKU/2006/ECE3063/Lec05(crypto).pdf, 2007.

[10] P. L. Jensen, *"pGNFS,"* http://pgnfs.org/index.php?page= Results, 2009.

저자 소개

최 명 복(종신회원)

- 1992년 : 호서대학교 전자계산학과(학사)
- 1994년 : 아주대학교 컴퓨터공학과(석사)
- 2001년 : 아주대학교 컴퓨터공학과(박사)
- 1997~현재 : 강릉원주대학교 멀티미디어공학과 교수
- 2004. 1~현재 : 한국인터넷방송통신학회 이사
- <주관심분야 : 지능형 정보검색, 퍼지응용, 지식표현, 신경망, 임베디드 및 유비쿼터스 응용, 지능형 교통제어, 소프트웨어 공학, 알고리즘>
- e-mail : cmb5859@gmail.com

이 상 운(정회원)

- 1983년~1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1995년 ~ 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 1998년 ~ 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007.2 : 국립 원주대학 여성교양과 조교수
- 2007.3 ~ 현재 : 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수
- <주관심분야 : 소프트웨어 프로젝트 관리, 소프트웨어 개발방법론, 소프트웨어 척도 (소프트웨어 규모, 개발노력, 개발기간, 팀 규모), 분석과 설계 방법론, 소프트웨어 시험 및 품질보증, 소프트웨어 신뢰성, 신경망, 뉴로-퍼지, 그래프 알고리즘>
- e-mail : sulee@gwnu.ac.kr