논문 2011-4-22

$k-$페르마 소인수분해 알고리즘

The $k-$Fermat's Integer Factorization Algorithm

최명복*, 이상운**

Myeong-Bok Choi, Sang-Un Lee

요 약 $n=pq$인 합성수 n을 p와 q로 소인수분해하는 것은 매우 어려운 문제이다. 대부분의 소인수분해 알고리즘은 $a^2 \equiv b^2 \pmod n$인 제곱 합동이 되는 (a,b)를 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 있다. 제곱합동의 기초가 되는 페르마 알고리즘은 $a^2-b^2=n$을 찾는다. 본 논문은 $a^2-b^2=kn, (k=1,2,\cdots)$를 찾는 방법을 제안하였다. 제안된 방법에서 b는 5의 배수로 $b_1=0$ 또는 5가 반드시 한 개는 존재한다고 가정한다. 첫 번째로, n_2n_1에 대해 $b_1=0$와 $b_1=5$을 만족하는 kn을 구하여 k를 결정한다. 두 번째로, $a^2-b^2=kn$이 되는 a_2a_1을 결정한다. 세 번째로, $kn<a^2<(k+1)n$ 범위에 속하는 $\sqrt{kn}<a<\sqrt{(k+1)n}$의 범위를 결정하여 a_2a_1 값들에 대해 $a^2-b^2=kn$으로 (a,b)를 구한다. 제안된 알고리즘을 몇 가지 사례에 적용한 결과 페르마 알고리즘에 비해 수행 속도를 현격히 단축시키는 효과를 얻었다.

Abstract It is very difficult problem to factorize composite number. 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). Fermat's algorithm that is base of congruence of squares finds $a^2-b^2=n$. This paper proposes the method to find $a^2-b^2=kn, (k=1,2,\cdots)$. It is supposed $b_1=0$ or 5 to be surely, and b is a double number. First, the proposed method decides k by getting kn that satisfies $b_1=0$ and $b_1=5$ about n_2n_1. Second, it decides a_2a_1 that satisfies $a^2-b^2=kn$. Third, it figures out (a,b) from $a^2-b^2=kn$ about a_2a_1 as deciding $\sqrt{kn}<a<\sqrt{(k+1)n}$ that is in $kn<a^2<(k+1)n$. The proposed algorithm is much more effective in comparison with the conventional Fermat algorithm.

Ⅰ. 서 론

소수 (prime number) p는 1과 자신 (p) 이외의 어떠한 수로도 나누어 질 수 없는 수로, 1은 소수가 아니다. 따라서 소수 $p=2,3,5,7,11,\cdots$이다.[1] RSA 암호체계 (cryptograph)는 큰 자리수의 숫자 n을 2개의 p와 q로 소인수분해 (factorization)하기가 어렵다는 이론에 기반을 두고 있다.[2] 공개 키 생성 (암호화 과정)에는 2개 소수의 곱 $n=pq$로 p,q를 찾는 소수 판별 (primality test)을 수행해야 하며, 암호 해독 과정에는 합성수 (composite number) n을 2개 소수 p,q로 소인수분해 해야 한다. 소수 판별법은 안전한 암호화 키를 생성하기 위해 필요한 연구이며, 소인수분해 방법은 암호체계를 깨트리기 위해 필요한 연구이다.[3,4] 여기서, p와 q는 크기 (자리수)가 비슷한 큰 소수들이 사용되고 있으며 각각을 소수 인자 (prime factor)라 한다. n이 공개되므로

*종신회원, 강릉원주대학교 멀티미디어공학과
**정회원, 강릉원주대학교, 멀티미디어공학과
접수일자 2011.5.19, 수정완료 2011.7.23
게재확정일자 2011.8.12

p만 결정하면 $q=n/p$로 계산될 수 있으나 p를 한 번에 결정하는 방법이 없다. 따라서 소인수분해의 모든 알고리즘은 p를 얼마나 빨리 찾을 수 있는가에 초점을 맞추고 있으며, 대부분은 체 (Sieve) 알고리즘을 적용하고 있다.[4] 100자리 십진수에 대해 1991년에 다중 다항식 2차 체 (MPQ Sieve) 알고리즘으로 2.2 GHz CPU를 사용하여 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 RSA Factoring Challenge가 활동을 중단한 이후 남아 있는 상금은 취소된 상태이다.[5] 즉, 현재까지 알려진 알고리즘으로는 더 이상 풀 수 없는 문제임을 알 수 있다.

본 논문은 $a^2-b^2=kn, (k=1,2,\cdots)$의 제곱합동 (congruence of squares)을 빠르게 찾는 방법을 제안한다. 2장에서는 소인수 분해 알고리즘을 고찰해 본다. 3장에서는 제곱합동을 빠르게 찾는 개선된 페르마 알고리즘을 제안한다. 4장에서는 제안된 알고리즘의 적용성을 평가해 본다.

II. 소인수분해 알고리즘 고찰 및 연구배경

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 있다.[4] 또한 최근에는 Choi와 Lee[6]는 $n+1$ 소인수분해 알고리즘을 제안하였다.

가장 단순한 나눗셈 시행 방법과 Shor 알고리즘을 제외한 대부분의 소인수분해 알고리즘은 페르마 알고리즘[7]에 기반을 두고 있다. 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 처

표 1. 제곱 합동
Table 1. Congruence of Squares

n	$\sqrt{n}$	$\sqrt{n}<a<{}_{\max}l(\sqrt{n})$ $a^2(\mathrm{mod}\, n)=b^2$ (a,b)	범위 (제곱합동 수)	a^2-b^2 (kn)
18,206,927[8]	4266.96	(5676, 3743) (7609, 1810) (9542,0123)	[4267,9999] 5,733개중 3개	n $3n$ $5n$
6,012,707[9]	2452.08	(2454,0097) (4811,2260) (4908,0194) (7265,2163) (7362,0291) (9719,2066) (9816,0388)	[2453,9999] 7,547개중 7개	n $3n$ $4n$ $8n$ $9n$ $15n$ $16n$
84,923[10]	291.46	(342,179) (505,016) (668,147) (847,195)	[292,999] 708개중 4개	n $3n$ $5n$ $8n$
1,649[11]	40.61	(57,40) (74,23) (91,06)	[41,99] 59개중 3개	n $3n$ $5n$

리 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 $p=GCD(a-b,n)$, $q=GCD(a+b,n)$를 결정한다. 실제로 a 값을 랜덤하게 선택하여 제곱의 합동을 찾는 것은 비현실적으로 많은 시간이 소요된다. 따라서 대안으로 n보다 작은 몇 개만을 찾는다. 예로 표 1에서 a를 찾는 것이 얼마나 어려운지 고찰해 보자. 예로, $n=18,206,927$인 경우, $\sqrt{n}<a\le {}_{\max}l\ (\sqrt{n})$인 $4267\le a\le 9999$의 5,733개 중 제곱합동이 되는 a는 5676, 7609와 9542의 3개 뿐이다. 페르마 알고리즘은 항상 $a^2-b^2=n$이 되는 (a,b)를 찾는다.

Dixon과 2차 체, GNFS 등의 방법은 페르마 알고리즘이 a를 순차적으로 찾는 방법을 개선한 것임에도 불구하고 적용에 어려움이 있다. 따라서 본 논문에서는 a를 보다 쉽게 찾는 방법을 제안한다.

Ⅲ. $k-$페르마 소인수분해 알고리즘

표 1의 제곱의 합동이 되는 (a,b)를 10보다 작은 소수인 $\{2,3,5,7\}$의 배수인지를 고찰하면 표 2와 같은 공통점을 찾을 수 있다.

표 2 제곱 합동 공통점
Table 2. Common Point of Congruence of Squares

n	$a^2 \pmod n = b^2$ (a,b)	배수 분석							
		a				b			
		2	3	5	7	2	3	5	7
18,206,927	(5676, 3743) (7609, 1810) (9542,0123)	○ ○	○		○	○	○	○	
6,012,707	(2454,0097) (4811,2260) (4908,0194) (7265,2163) (7362,0291) (9719,2066) (9816,0388)	○ ○ ○ ○	○ ○ ○ ○	○		○ ○ ○ ○	○ ○	○	○
84,923	(342,179) (505,016) (668,147) (847,195)	○ ○	○	○	○	○	○	○	○
1,649	(57,40) (74,23) (91,06)	○	○		○	○ ○	○	○	

즉, a는 4개의 모든 수에서 2와 3의 배수를 갖고 있으며, b는 2,3,5의 배수를 갖고 있다. 따라서 본 장에서는 이러한 특징에 기반하여 a의 범위를 한정시켜 빠르게 찾는 알고리즘을 제안한다. 페르마 알고리즘의 $\sqrt{n}<a<n$에서 $a^2-b^2=n$을 찾는 방식인데 반해 제안된 알고리즘은 $\sqrt{kn}<a<\sqrt{(k+1)n}$에서 a를 찾는 차이점이 있다.

본 논문에서는 식 (1)을 만족하는 5의 배수인 b는 반드시 1개가 존재한다고 가정한다.

$$a^2=b^2+kn,(k=1,2,3\cdots,10),\ b^2=a^2-kn \quad (1)$$

$n=pq$의 특징을 살펴보자. 자리수를 아래첨자로 표기하면, 소수 p,q의 $p_1=\{1,3,7,9\}$, $q_1=\{1,3,7,9\}$이며, 합성수 $n=pq$의 $n_1=\{1,3,7,9\}$이 된다. 따라서 $a_1^2-b_1^2=n_1$을 구하여 $n_1=\{1,3,7,9\}$이 되는 경우를 고찰해 보면 표 3과 같다. 표 3에서 $n_1=\{1,9\}$는 5의 배수인 0 또는 5가 a_1과 b_1에 존재하지만 $n_1=\{3,7\}$은 존재하지 않음을 알 수 있다. $n_1=3$인 경우, $3n=9$로 $n_1=9$를 적용할 수 있으며, $n_1=7$인 경우, $3n=21$로 $n_1=1$을 적용할 수 있다.

표 3 $a_1^2-b_1^2=n_1$
Table 3. $a_1^2-b_1^2=n_1$

n_1	b_1									
	1	2	3	4	5	6	7	8	9	0
a_1 1	0	**7**	2	5	6	5	2	**7**	0	**1**
2	**3**	0	5	8	**9**	8	5	0	**3**	4
3	8	5	0	**3**	4	**3**	0	5	8	**9**
4	5	2	**7**	0	**1**	0	**7**	2	5	6
5	4	**1**	6	**9**	0	**9**	6	**1**	4	5
6	5	2	**7**	0	**1**	0	**7**	2	5	6
7	8	5	0	**3**	4	**3**	0	5	8	**9**
8	**3**	0	5	8	**9**	8	5	0	**3**	4
9	0	**7**	2	5	6	5	2	**7**	0	**1**
0	**9**	6	**1**	4	5	4	**1**	6	**9**	0

n_1	a_1	b_1
1	1, 4, **5**, 6, 9, **0**	2, 3, **5**, 7, 8, **0**
3	2, 3, 7, 8	1, 4, 6, 9
7	1, 4, 6, 9	2, 3, 7, 8
9	2, 3, **5**, 7, 8, **0**	1, 4, **5**, 6, 9, **0**

5의 배수인 b의 1 자리수 (b_1)은 0또는 5이다. 즉, $b_1=0$이면 $(a_2a_1)^2=n_2n_1$, $b_1=5$이면 $(a_2a_1)^2$

$-(b_1)^2 = n_2n_1$이다. $a_2a_1 = [01,99]$ 범위에서 n_2n_1을 구하면 표 4와 같다. 여기서는 편의상 홀수만을 표현하였으며, $b_2 = [0,9]$이다. 표 4의 결과를 kn의 n_2n_1으로 요약 정리한 결과는 표 5와 같다.

표 4. $n_2n_1 = (a_2a_1)^2 - (b_1)^2$

Table 4. $n_2n_1 = (a_2a_1)^2 - (b_1)^2$

구분	a^2-b^2		n_2n_1	
a_2a_1	$b_1=0$	$b_1=5$	$b_1=0$	$b_1=5$
01	**01**	**76**	01	76
03	**09**	**84**	09	84
07	**49**	**24**	49	24
09	**81**	**56**	81	56
11	**121**	96	21	96
13	**169**	**144**	69	44
17	**289**	**264**	89	64
19	**361**	**336**	61	36
21	**441**	**416**	41	16
23	**529**	**504**	29	04
27	**729**	**704**	29	04
29	**841**	**816**	41	16
31	**961**	**936**	61	36
33	1,0**89**	1,0**64**	89	64
37	1,3**69**	1,3**44**	69	44
39	1,5**21**	1,4**96**	21	96
41	1,6**81**	1,6**56**	81	56
43	1,8**49**	1,8**24**	49	24
47	2,2**09**	2,1**84**	09	84
49	2,4**01**	2,3**76**	01	76

구분	a^2-b^2		n_2n_1	
a_2a_1	$b_1=0$	$b_1=5$	$b_1=0$	$b_1=5$
51	2,6**01**	2,5**76**	01	76
53	2,8**09**	2,7**84**	09	84
57	3,2**49**	3,2**24**	49	24
59	3,4**81**	3,4**56**	81	56
61	3,7**21**	3,6**96**	21	96
63	3,9**69**	3,9**44**	69	44
67	4,4**89**	4,4**64**	89	64
69	4,7**61**	4,7**36**	61	36
71	5,0**41**	5,0**16**	41	16
73	5,3**29**	5,3**04**	29	04
77	5,9**29**	5,9**04**	29	04
79	6,2**41**	6,2**16**	41	16
81	6,5**61**	6,5**36**	61	36
83	6,8**89**	6,8**64**	89	64
87	7,5**69**	7,5**44**	69	44
89	7,9**21**	7,8**96**	21	96
91	8,2**81**	8,2**56**	81	56
93	8,6**49**	8,6**24**	49	24
97	9,4**09**	9,3**84**	09	84
99	9,8**01**	9,7**76**	01	76

제안된 알고리즘은 $a^2-b^2 = kn$이 되는 (a,b)를 표 5를 이용하여 찾기 때문에 "k－페르마 알고리즘"이라 칭하며 다음과 같이 5단계로 수행된다.

표 5. n_2n_1 **값에 따른** a_2a_1 **또는** b_2b_1 **값 할당**

Table 5. Allocation of a_2a_1 or b_2b_1 by n_2n_1

n_2n_1	$(a_2a_1)^2-(b_1)^2=n_2n_1$ $\sqrt{kn}<a<\sqrt{(k+1)n}$		$(a_2a_1)^2-(b_2b_1)^2=n_2n_1$ $\min l(\sqrt{n})<b<\sqrt{n}$	
	a_2a_1		b_2b_1	
	$b_1=0$	$b_1=5$	$a_1=0$	$a_1=5$
01	01, 49, 51, 99	-	-	18, 32, 68, 82
09	03, 47, 53, 97	-	-	04, 46, 54, 96
11	-	06, 44, 56, 94	17, 33, 67, 83	-
19	-	12, 38, 62, 88	09, 41, 59, 91	-
21	11, 39, 61, 89	-	-	02, 48, 52, 98
29	23, 27, 73, 77	-	-	14, 36, 64, 86
31	-	16, 34, 66, 84	13, 37, 63, 87	-
39	-	08, 42, 58, 92	19, 31, 69, 81	-
41	21, 29, 71, 79	-	-	22, 28, 72, 78
49	07, 43, 57, 93	-	-	24, 26, 74, 76
51	-	24, 26, 74, 76	07, 43, 57, 93	-
59	-	22, 28, 72, 78	21, 29, 71, 79	-
61	19, 31, 69, 81	-	-	08, 42, 58, 92
69	13, 37, 63, 87	-	-	16, 34, 66, 84
71	-	14, 36, 64, 86	23, 27, 73, 77	-
79	-	02, 48, 52, 98	11, 39, 61, 89	-
81	09, 41, 59, 91	-	-	12, 38, 62, 88
89	17, 33, 67, 83	-	-	06, 44, 56, 94
91	-	04, 46, 54, 96	03, 47, 53, 97	-
99	-	18, 32, 68, 82	01, 49, 51, 99	-

04	02, 48, 52, 98	23, 27, 73, 77	14, 36, 64, 86	11, 39, 61, 89
16	04, 46, 54, 96	21, 29, 71, 79	22, 28, 72, 78	03, 47, 53, 97
24	18, 32, 68, 82	07, 43, 57, 93	24, 26, 74, 76	01, 49, 51 ,99
25	05, 15, …, 95	-	-	10, 20, …, 90
36	06, 44, 56, 94	19, 31, 69, 81	08, 42, 58, 92	17, 33, 67, 83
44	12, 38, 62, 88	13, 37, 63, 87	16, 34, 66, 84	09, 41, 59, 91
56	16, 34, 66, 84	09, 41, 59, 91	12, 29, 71, 79	13, 37, 63, 87
64	08, 42, 58, 92	17, 33, 67, 83	06, 44, 56, 94	19, 31, 69, 81
75	-	10, 20, …, 90	05, 15, …, 95	-
76	24, 26, 74, 76	01, 49, 51, 99	18, 32, 68, 82	07, 43, 57, 93
84	22, 28, 72, 78	03, 47, 53, 97	04, 46, 54, 96	21, 29, 71, 79
96	14, 36, 64, 86	11, 39, 61, 89	02, 48, 52, 98	23, 27, 73, 77

(1) k**와** a_2a_1 **결정**

if $n_1 = \{1|9\}$ then $k=1$로 설정, 표 5에서 n_2n_1 값에 대응하는 $b_1=0$ 또는 $b_1=5$의 a_2a_1 선택

else if $n_1 = \{3|7\}$ then /* 표 5에 n_2n_1이 미존재

if $n_2n_1 = i \times j, (i=3,7,9,\ j=1,2,\cdots,9)$ then

$k=3$인 $3n$에 대응하는 $b_1=0$의 a_2a_1 선택

else if $n_2n_1 \neq i \times j$ then $b_1=5$에 해당하는 a_2a_1 값을 가진 kn으로 k 결정.

if $b_1=0$ then a_1은 i,j 값으로 한정.

(2) a 범위와 초기값 결정

$\sqrt{kn} < a < \sqrt{(k+1)n}$에 대해 $l(a)=\left\lceil \frac{l(n)}{2} \right\rceil, l(b)=l(a)-1$가 되도록 $a_{\min}$의 $l(b)$ 자리수 $a_{l(b)}$ 값을 +1, a_3까지 0으로, a_2a_1는 (1)에서 구한 값으로 설정한다. 각 a 값 $\times\times a_2a_1$에 대한 +100 증분, 10개 숫자에 대해 a_2a_1이 홀수인 경우, $i_2i_1=07,11,13,17$의 곱을, 짝수인 경우 $i_2i_1=08,12,14,18$의 곱을 찾아 초기 값 (a_{init})으로 결정한다.

(3) a와 b 결정

초기값 a_{init}과 $a=a_{init}+i_2i_100\times 2^1\times 5^j\times k, (j\geq 2, k=101:2:997)$에 대해 $a^2-b^2=kn$을 만족하는 a와 b를 탐색한다. 만약, a,b가 존재하지 않으면 표 5에서 $a_2a_1=\times 0$ 또는 $\times 5$인 경우에 대해 $a_2=[0,9]$를 초기 값으로 하여 +100 증분으로 탐색한다. 왜냐하면 11, 12, 13, 14, 17과 18의 곱은 $2^2\times 5^2\times i_2i_1$ 형태이다. 예로, $1100=2^2\times 5^2\times 11$이 된다. 만약, a,b가 존재하지 않으면 (4)를 수행한다.

(4) $(a_2a_1)^2-(b_2b_1)^2=n_2n_1$, $(a_2a_1)^2=(b_2b_1)^2+n_2n_1$의 b_2b_1를 구한다. 이 경우 $_{\min}l(\sqrt{n})<b<\sqrt{n}$의 범위를 갖으며, b_2b_1에 대해 $(b_3b_2b_1)^2+n_3n_2n_1=(a_2a_1)^2$, $b_3=[0,9]$ 이 5의 곱이 정수가 되는 값을 초기 값으로 결정한다. 이들 초기 값에 대해 $_{\min}l(\sqrt{n})<b<\sqrt{n}$ 범위에서 $\min l(\sqrt{n})$의 $l(b)$자리수를 1로 하고, +1,000 증분으로 찾는다.

(5) p와 q 결정

$a^2-b^2=(a-b)(a+b)$ 공식에 의거 유클리드 알고리즘으로 $p=GCD(a-b,n), q=GCD(a+b,n)$을 구한다.

Ⅳ. 알고리즘 적용 결과 분석

표 1의 4개 데이터와 pGNFS[12] 데이터에 제안된 알고리즘을 적용하여 보자. pGNFS[11]는 $n=$2,352,854,039을 $p=$42,013과 $q=$56,003로, $n=$8,229,944, 909,131, 434,961를 $p=$2,352, 854,041과 $q=$3,497,856,121로, $n=$982,301,348,481,613,682,763,34 9,336,546,115,836,409을 $p=$20,989,897,656,489,026,809와 $q=$46,79 8,767,890,987, 654,401로 소인수분해 하였다.

$n=18,206,927$인 경우, $n_1=7, n_2n_1=27=3\times 9$로 $i,j=\{3,9\}$, $3n=81$로 $k=3$으로 결정된다. $b_1=0$인 $a_2a_1==\{09,41,59,91\}$ 중에서 $i,j=\{3,9\}$를 포함하는 $\{09,59\}$축소된다. $kn<a^2<(k+1)n$, $\sqrt{3n}<a<\sqrt{4n}$ 은7309.5873$<a<$8553.9151이며, $l(n)=8, l(a)=4, l(b)=3$으로 $a_{\min}$의 $l(b)$자리 값이 3에서 4로 증가되어 7400이 된다. 또한, a_2a_1은 09, 59이므로 7409, 7459로 결정된다. 각 값에 대해 +100의 증분으로 10개의 값들에 대해 7,11,13,17의 곱이 되는 첫 번째 값은 xx09는 7609(7), 7909(11), 7709(13), 8109(17)로, xx59는 7959(7), 8459 (11), 8359(13)로 a_{init}이 결정된다. 각 a_{init} 값과 $a=a_{init}+i_2i_100\times 2^1\times 5^j\times k, (j\geq 2, k=101:2:997)$을 탐색 결과 xx09의 7의 곱 a_{init}인 $a=7609$, $b=1810$으로 알고리즘은 1회 수행된다. 결국, $p=GCD(7609-1810,18206927)=1933$, $q=GCD(7609+1810,18206927)=9419$이다.

$n=6,012,707$의 경우, $n_1=7, n_2n_1=07=7\times 1$로 $i,j=\{1,7\}$, $3n=11$로 $k=3$으로 결정된다. $b_1=0$의 $a_2a_1=\{11,39,61,89\}$중에서 $i,j=\{1,7\}$을 포함하는 $a_2a_1=\{11,61\}$이 선택된다. $kn<a^2<(k+1)n$인 $\sqrt{3n}<a<\sqrt{4n}$ 은 4247.1309$<a<$4904.1643이며, $a_{\min}$은 $l(a)=4, l(b)=3$으로 4247이 4300으로, $a_2a_1=11$로 4311로 결정된다. 결국, $4311\leq a\leq 4811$에서 4311과 4361 에 대해 증가분 +100을 하면서 10개씩을 선택하여 7,11,13,17 17의 곱을 결정하면 4711(7), 4411 (11), 4511(13), 4811(17)과 4361(7)이 a_{init}으로 결정된다. 각 a_{init} 값들에 대해 $a=a_{init}+i_2i_100\times 2^1\times 5^j\times k$는 $4311\leq a\leq 4811$에 존재하지 않으므로 탐색 대상은 5개만이 존재한다. 여기서 $a^2-b^2=3n$을 만족하는 값은 xx11의 17의 곱 a_{init}인 $a=4811$, $b=$

2260을 얻고 알고리즘이 종료되었다. $p = GCD(4811-2260, 6012707) = 2551$, $q = GCD(4811+2260, 6012707) = 2357$을 얻는다.

$n = 84,923$의 경우, $n_1 = 3, n_2 n_1 = 23 \neq i \times j$로 $b_1 = 5$에 해당하는 kn을 구하면 $8n = 81$로 $k = 8$로 결정된다. 따라서 $a_2 a_1 = \{03,47,53,97\}$이다. $kn < a^2 < (k+1)n$인 $\sqrt{8n} < a < \sqrt{9n}$은824.2475 874.265이며, $l(a) = 3, l(b) = 2$로 $a_{\min}$의 2번째 자리를 +1로 하면 834가 되며, $a_2 a_1 = 47$로 $847 \leq a \leq 847$로 $a = 847$의 1개만이 존재한다. 즉, $a = 847$, $b = 195$를 1회 만에 얻고 알고리즘이 종료되었다. $p = GCD(847-195, 84923) = 163$, $q = GCD(847+195, 84923) = 521$이다.

$n = 1,649$의 경우, $n_1 = 9, n_2 n_1 = 49 = 7 \times 7$로 $k = 1$이며, $a_2 a_1 = \{07,43,57,93\}$중에서 $i,j = \{7\}$을 포함하는 $a_2 a_1 = \{07,57\}$로 결정된다. $40.6079 < a < 57.4282$에서 $l(a) = 2, l(b) = 1$로 $a_{\min}$의 1번째 자리를 +1로 하면 41이 되며, $a_2 a_1 = 07$이므로 $47 \leq a \leq 57$이 된다. 이들 중 $a_2 a_1 = \{07,43,57,93\}$을 포함하는 수는 57인 1개만이 존재한다. 즉, $a = 57$, $b = 40$을 1회에 얻고 알고리즘이 종료되었다. $p = GCD(57-40, 1649) = 17$, $q = GCD(57+40, 1649) = 97$이다.

$n = 2,352,854,039$에 대해, $n_1 = 9, n_2 n_1 = 39 \neq i \times j$로 $k = 1$이며, $a_2 a_1$은 $b_1 = 5$에 존재하는 $a_2 a_1 = \{08,42,58,92\}$이다. $kn < a^2 < (k+1)n$인 $\sqrt{n} < a < \sqrt{2n}$은 $48506.2268 < a < 68598.1638$이며, $l(n) = 10, l(a) = 5, l(b) = 4$로 $a_{\min}$의 $l(b)$ 자리수를 +1로 하면 49000이며, $a_2 a_1 = \{08,42,58,92\}$로 49, 008, 49042, 49058, 49092가 초기 값이 된다. 각 초기 값에 대해 +100의 증가분으로 10개씩 값들을 선정하여 $a_2 a_1$이 짝수이므로, 8,12,14,18의 곱을 찾으면 xx08은 49,008(8), 49,008(12), 49,308(14), 49,608(18), xx42는 49, 242(12), 49,042(14), 49,842(18), xx58은 49,058(8), 49,158(12), 49,658(14), 49,158(18), xx92는 49,092(8), 49,092(12), 49,392(14), 49,392 (18)을 a_{init}로 얻는다. a,b는 49,008의 8의 곱 a_{init}인 $a = 49008, b = 6995$를 얻어 알고리즘은 1회를 수행하였다. 즉, $p = GCD(49008-6995, 2352854039) = 42013$, $q = GCD(49008+6995, 2352854039) = 56003$이다.

$n = 8,229,944,909,131,434,961$에 대해, $n_1 = 1$, $n_2 n_1 = 61 \neq i \times j$로 $k = 1$이며, $a_2 a_1 = \{19,31,69,81\}$ 모두 선택된다. $\sqrt{n} < a < \sqrt{2n}$은 $2868788055.8054 < a < 4057078976.0939$이며, $l(n) = 19, l(a) = 10$, $l(b) = 9$로 $a_{\min}$의 $l(b)$ 자리수를 +1로 하면 2,900,000,019, 2,900,000,031, 2,900,000,069, 2,900,000,081의 4개가 초기 값이 된다. 각각에서 7,11,13,17의 곱을 결정하면 xx19에 대해서는 2,900,000,019(7), 2,900,000,719(11), 2,900,000,51 9(13), xx31에 대해서는 2,900,000,131(7), 2,900,000,631(11), 2,900, 000,831(13), xx69에 대해서는 2,900,000,369(7), 2,900,000,169(11), xx81에 대해서는 2,900,000,481(7), 2,900,000,081(11), 2,900,000,18 1(13), 2,900,000,981(17)이 초기 값으로 결정되었다. 12개의 각 초기 값에 대해 $a = a_{init} + i_2 i_1 00 \times 2^1 \times 5^j \times k (j \geq 2, k = 101:2:997)$에 대해 탐색 결과 xx81의 11의 곱 3 6 1 번 째 인 $2,900,000,081 + 1100 \times 2 \times 5^2 \times 461 = 2,900,000,081 + 1100 \times 23,050 = 2,925,355,081$에서 $a^2 - b^2 = n$을 만족하여 알고리즘은 361회를 수행하였다. 이 경우, $a = 2,925,355,081$, $b = 572,501,040$이다.

$n = 982,301,348,481,613,682,763,349,336,546,115,836,409$에 대해, $n_1 = 9, n_2 n_1 = 09 = 9 \times 1, 3 \times 3$으로 $i,j = \{1,3,9\}$, $k = 1$이다. $a_2 a_1 = \{03,47,53,97\}$ 중에서 $i,j = \{1,3,9\}$를 포함하는 $a_2 a_1 = \{03,53\}$가 선택되었다. $kn < a^2 < (k+1)n$인 $\sqrt{n} < a < \sqrt{2n}$은 $3.13416870714008 \times 10^{19} < a < 4.43238389240286 \times 10^{19}$이다. $l(n) = 39, l(a) = 20, l(b) = 19$로 $a_{\min}$의 $l(b)$ 자리수를 +1로 하여 32,000,000,000,000,000,003과 32,000,000,000,000,000,053에 대해 +100의 증가분으로 10개씩에 대해 7,11,13,17의 곱을 결정하면 xx03에 대해서는 xx003(7), xx703 (11), xx53에 대해서는 x00,353(7), x,00,353(11), x00,553(13), x00,353(17)의 초기 값이 결정되었다. 6개의 각 초기 값에 대해 증분을 탐색 결과 실패하였다. 다음으로 $(b_2 b_1)^2 - (a_1)^2 = n_2 n_1$의 $a_1 = 5$인 경우의 $b_2 b_1 = \{04, 46, 54, 96\}$에 대해 $(b_3 b_2 b_1)^2 + n_3 n_2 n_1 = (a_2 a_1)^2, b_3 = [0,9]$이 5의 곱인 정수가 되는 값을 구한 결과 x04, x46, x54는 없으며, x96에 대해서는 096 296, 396, 596, 796, 896을 얻었다. 예로,

$(096)^2+409=625=25^2$으로 $a_2a_1=25$, $a_1=5$이다. 따라서 이들 6개 초기 값에 대해 $110000000000000000<b<\sqrt{n}$ 범위에서 +1,000 증분으로 찾은 결과 x796의 1,904,435,117,249,313 번째에서 $a=33,894,332,773,738,340,605$, $b=1,904,435,117,249,313,796$을 얻었다.

7개의 데이터에 대해 제안된 알고리즘을 적용한 결과를 페르마 알고리즘과 비교하면 표 6과 같다.

알고리즘 성능비교에서는 Dixon과 2차 체 알고리즘은 생략하였다. 왜냐하면 Dixon 알고리즘[10]은 $n=84,923$에 대해서만, 2차 체 알고리즘[11]은 $n=1,649$에 대해서만 기술되어 있으며, 실제로 구하기가 어렵기 때문이다.

7개의 데이터에 대해 제안된 $k-$페르마 알고리즘을 적용한 결과 $n=6,012,707$을 제외한 6개 데이터에서 a는 $a_{\min}$ 근방에 위치하여 빠르게 탐색할 수 있음을 알 수 있다.

그러나 $n=982,301,348,481,613,682,763,349,336,546,115,836,409$에 대해서는 b가 $b_{\min}$ 근방에 위치함에도 불구하고 자리수가 크고 빠른 탐색 방법을 찾지 못하여 수행 횟수가 많음을 알 수 있다.

V. 결 론

본 논문은 $a^2-b^2=kn$의 a가 존재하는 범위를 빠르게 찾는 방법을 제안하였다. 제안 알고리즘은 "5의 배수인 b_1이 0 또는 5인 a는 $\sqrt{kn}<a<\sqrt{(k+1)n}$에 반드시 1개는 존재한다."는 가정에 기반하여 $k=1,2,\cdots,10$인 kn 중에서 n_2n_1을 찾고, $n_2n_1=i\times j$, $(i=3,7,9,\ j=1,3,7,9)$이면 $b_1=0$을 그렇지 않으면 $b_1=5$로 설정하고 k와 a_2a_1를 결정한다. 다음으로, $\sqrt{kn}<a<\sqrt{(k+1)n}$ 범위에서 $a^2-b^2=kn$이 되는 (a,b)를 찾는다.

페르마 알고리즘과 비교한 결과 제안된 $k-$페르마 알고리즘이 제곱합동 (a,b)를 월등히 빠르게 찾을 수 있음을 알 수 있었다.

참 고 문 헌

[1] Wikipedia, "Prime Number," http://en.wikipedia.org/wiki/ Prime_number, 2010.

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

[3] Wikipedia, "Primality Test," http://en.wikipedia.org/wiki/Primality_test, 2010.

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

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

표 6. 알고리즘 성능 비교
Table 6. Compare to Algorithm's Runtime

n	성능 비교 (탐색 횟수)	
	페르마 알고리즘 (a,b)	$k-$페르마 알고리즘 (a,b)
18,206,927	1,410회 (5676, 3743)	1회 (7609, 1810)
6,012,707	2회 (2454,0097)	1회 (4811,2260)
84,923	51회 (342,179)	1회 (847,195)
1,649	17회 (57,40)	1회 (57,40)
2,352,854,039	502회 (49008, 6995)	1회 (49008, 6995)
8,229,944,909, 131,434,961	25,355,000회 (2925355081, 572501040)	361회 (2925355081, 572501040)
982,301,348,481,613,682,763,349,336,546,115,836,409	25,526,457,023,37,540,605회	1,904,435,117,249,313회

[6] 최명복, 이상운, "n+1 소인수분해 알고리즘," 한국인터넷방송통신학회, 제11권, 제2호, 2011년 4월 30일.
[7] Wikipedia, "Fermat's Factorization Method," http://en.wikipedia.org/wiki/Fermat's_factorization _method, 2010.
[8] 김승주, "공개키 암호 방식," School of Information and Communication Engineering, Sungkyunkwan University, http://dosan.skku.or.kr/~sjkim/LectureNotes/SKKU/2006/ECE3063/Lec05(crypto).pdf, 2007.
[9] 진 훈, "Code and RSA," 전자계산학과 대학원, 그룹웨어 연구실, 경기대학교, 2010.
[10] Wikipedia, "Dixon's Factorization Method," http://en.wikipedia.org/wiki/Dixon%27s_factorization_method, 2010.
[11] Wikipedia, "Quadratic Sieve," http://en.wikipedia.org/wiki/ Quadratic_sieve, 2010.
[12] 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