The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 14, No. 6, pp.25-31, Dec. 31, 2014. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2014.14.6.25
JIIBC 2014-6-5

비대칭키 RSA의 $\phi(n)$ 해독을 위한 역 아기걸음- 2^k-ary 성인걸음법

Reverse Baby-step 2^k-ary Adult-step Method for $\phi(n)$ Decryption of Asymmetric-key RSA

이상운*

Sang-Un Lee*

요 약 비대칭키 RSA의 공개키 e와 합성수 $n=pq$은 알고 있고 개인키 d를 모를 때, $\phi(n)=(p-1)(q-1)=n+1-(p+q)$ 을 구하여 $d=e^{-1}(\mathrm{mod}\,\phi(n))$으로 개인키 d를 해독한다. 암호해독은 일반적으로 $n/p=q$ 또는 $a^2\equiv b^2(\mathrm{mod}\,n), a=(p+q)/\ 2, b=(q-p)/2$ 를 구하는 소인수 분해법이 널리 적용되고 있다. 그러나 아직까지도 많은 RSA 수들이 해독되지 않고 있다. 본 논문은 $\phi(n)$을 직접 구하는 알고리즘을 제안하였다. 제안된 알고리즘은 이산대수의 아기걸음-거인걸음법과 모듈러 지수연산의 2^k-ary법을 적용하였다. 이 알고리즘은 역-아기걸음과 2^k-ary 성인걸음법을 적용하여 기본적인 성인걸음법 수행횟수를 $1/2^k$로 줄이고, $m=\lfloor\sqrt{n}\rfloor$ 의 저장 메모리 용량도 $l, a^l>n$로 감소시켜 $\phi(n)$을 l회 이내로 구하였다.

Abstract When the public key e and the composite number $n=pq$ are disclosed but not the private key d in an asymmetric-key RSA, message decryption is carried out by obtaining $\phi(n)=(p-1)(q-1)=(n+1)-(p+q)$ and subsequently computing $d=e^{-1}(\mathrm{mod}\,\phi(n))$. The most commonly used decryption algorithm is integer factorization of $n/p=q$ or $a^2\equiv b^2(\mathrm{mod}\,n), a=(p+q)/2, b=(q-p)/2$. But many of the RSA numbers remain unfactorable. This paper therefore applies baby-step giant-step discrete logarithm and 2^k-ary modular exponentiation to directly obtain $\phi(n)$. The proposed algorithm performs a reverse baby-step and 2^k-ary adult-step. As a results, it reduces the execution time of basic adult-step to $1/2^k$ times and the memory $m=\lceil\sqrt{n}\rceil$ to $l, a^l>n$, hence obtaining $\phi(n)$ by executing within l times.

Key Words : Euler's totient function, Integer factorization, Discrete logarithm, Baby-step giant-step, Modular exponentiation

Ⅰ. 서 론

대표적인 비대칭키 RSA의 n은 유사하거나 동일한 길이의 2개 소수 p와 q를 선택하여 곱한 값 $n=pq$로 쉽게 계산되는 합성수 (composite number)이다. 일단 n을 얻고 난 후에는 p,q를 없애 공개키를 만든 사람 조차도 p,q를 알 수 없다. 비대칭키를 사용하는 이유는 n을 p,q로 소인수분해 (integer factorization)하기 어렵다는데 기반

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 : 2014년 8월 13일, 수정완료 : 2014년 10월 25일
게재확정일자 : 2014년 12월 12일

Received: 13 August, 2014 / Revised: 25 October, 2014 / Accepted: 12 December, 2014
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

하고 있다.[1-3]

RSA의 암호를 해독하기 위해 $q=n/p$의 나눗셈 시행법 (trial division)이나 $a^2 \equiv b^2 (\mathrm{mod}\, n), a=(p+q)/2$, $b=(q-p)/2$의 제곱합동법 (congruence of squares)으로 p,q를 구하여 다시 $\phi(n)$을 구하는 간접법을 적용하고 있다.[4,5]

십진수 길이가 100자리 이상인 n만 알고 있는 상태에서 나눗셈 시행법으로 소인수분해하기 위해 최신 컴퓨터를 활용해도 80년 이상이 소요되어 현실적으로 거의 불가능한 것으로 알려져 있다.

제곱합동법을 적용하는 방법에는 2차 체 (quadratic Sieve), MPQS (multiple-polynomial quadratic sieve), NFS (Number field sieve), GNFS (General number field), Dixon, CFRAC (continued fraction factorization), SQUFOF (Shanks′ square forms factorization) 등이 있다.[6] 지금까지 소인수분해된 RSA 수들을 고찰해보면 대부분 GNFS를 적용하였으나 아직까지 해독되지 않은 RSA 수들이 많이 존재하고 있다.[7]

RSA 암호체계에서 송신자는 메시지 m에 대해 $e<n, \gcd(e,\phi(n))=1$을 선택하여 암호 코드 $c=m^e (\mathrm{mod}\, n)$로 변환시켜 전송하며, 수신자는 개인키 d를 이용하여 $m=c^d (\mathrm{mod}\, n)$로 해독한다.[1-3] 여기서 $ed(\mathrm{mod}\,\phi(n))=1$이며, (e,n)은 공개된다. $\phi(n)$은 n과 서로소인 개수로 오일러 totient 함수이다.[8] RSA를 적용하는 암호에서 $\phi(n)$만 알고 있으면 식 (1)에 의해 개인키 d를 해독할 수 있다.

$$ed(\mathrm{mod}\,\phi(n))=1,\ d=e^{-1}(\mathrm{mod}\,\phi(n)) \tag{1}$$

소수 p에 대해 $\phi(p)=p-1$, $\phi(p)/2=(p-1)/2$로 쉽게 구할 수 있다. 그러나 합성수 n의 $\phi(n)$은 식 (2)와 같다. 여기서 $(p+q-1)$는 p,q의 배수 개수로 n개에서 소인수 p,q의 배수 개수를 빼면 n과 서로소인 원소의 개수 $\phi(n)$이 구해진다.

$$\begin{aligned}\phi(n)&=(p-1)(q-1)=pq-(p+q-1)\\&=n-(p+q-1)\\(p+q)&=(n+1)-\phi(n)\end{aligned} \tag{2}$$

합성수 n에 대해서는 임의의 수 $a(a<n)$에 대해 식 (3)이 성립한다.

$$a^{\phi(n)}=a^{\phi(n)/2}\equiv 1 \tag{3}$$

$a^b \equiv c(\mathrm{mod}\, n)$에서 a,c,n이 주어졌을 때 b를 구하는 문제는 $b=\log_a c$로 풀 수 있어 이산대수 (discrete logarithm)라 한다.[2] 이산대수 알고리즘을 적용하면 $\phi(n)$을 구할 수 있다.

대표적인 이산대수 알고리즘으로는 아기걸음-거인걸음 (Baby-step Giant-step)과 Pollard의 rho 등이 있다.[9-11] 그러나 어떠한 알고리즘도 빠르게 해를 구하지 못하고 있다.[12]

본 논문은 아기걸음-거인걸음 알고리즘의 모듈러 연산 횟수와 저장 데이터의 거인걸음 보폭 크기를 성인걸음 보폭으로 획기적으로 줄이고, 2^k명이 동시에 걸어가는 형태로 $\phi(n)$을 찾는 알고리즘을 제안한다. 2장에서는 아기걸음-거인걸음법을 고찰한다. 3장에서는 역 아기걸음-4중 성인걸음 알고리즘을 예로 제안하고 4장에서는 제안된 알고리즘의 성능을 검증하여 본다.

II. 이산대수 알고리즘

아기걸음-거인걸음 알고리즘은 그림 1과 같이 n을 $m=\lceil \sqrt{n} \rceil$의 거인걸음 보폭으로 분할하고, 아기걸음 단계에서는 거인걸음 보폭 $[a^0, a^{m-1}]$에 대해 아기걸음으로 보폭 1인 m걸음의 모듈러 지수연산을 수행한다.

다음으로 거인걸음 시작점인 $a^m (\mathrm{mod}\, n)$의 역함수 $a^{-m} (\mathrm{mod}\, n)$을 유클리드 알고리즘으로 구한다. 마지막으로 거인걸음 단계에서는 보폭 m으로 j번째 걸음에서 아기걸음의 i번째 값과 일치하면, $b=jm+i$로 결정한다.[9,10] 이는 식 (4)로 증명된다.

$$\begin{aligned}&a^b\times(a^{-m})^j=a^i,\ a^{b-jm}=a^i,\\&a^b=a^{mj+i}, b=mj+i\end{aligned} \tag{4}$$

이 방법은 미지의 $b(a^b)$부터 시작하여 보폭 m의 $-j$걸음을 걸어가 0 $(a^0=1)$ 근방의 i를 찾아 $b=jm+i$를 계산하는 방식이며, $b/mj\neq 0$가 되지 못하는 경우가 대부분으로 $-j$번째 걸음이 $[0,m-1]$의 i번째와 일치하면 i만큼 더해주는 방식이다.

입력 : a, c, n, 출력 : b
$a^b \equiv c \pmod n$
$m = \lceil \sqrt{n} \rceil$ /* 거인 보폭 결정

/* Baby-step: 수행복잡도 $O(\sqrt{n})$ */
for $i = 0$ to $m-1$
$c_i = a^i \pmod n$ 계산, (i, c_i) 저장.
end
/* 거인걸음 시작점 결정 */
$c_m = a^m \pmod n$에 대해 $d = (c_m)^{-1} \pmod n$ 계산

/* Giant-step: 수행복잡도 $O(\sqrt{n})$*/
$j = 0$, $c_j = c$.
if $c_j = c_i$ then $b = i$, 알고리즘 종료.
for $j = 1$ to m
$c_j = c_{j-1} \times d \pmod n$ 계산 /* $c_j = c \times d^j \pmod n$
if $c_j = c_i$ then $b = mj + i$
else if $c_j \neq c_i$ then $j = j+1$, continue.
end

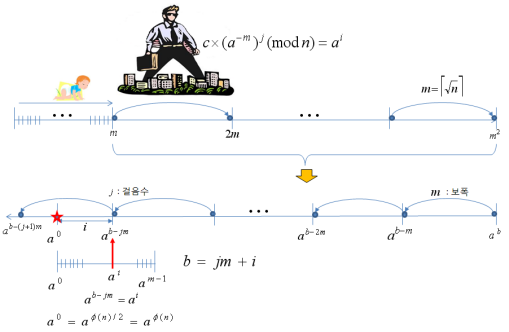


그림 1. 아기걸음-거인걸음 알고리즘
Fig. 1. Baby-step Giant-step Algorithm

아기걸음-거인걸음 알고리즘의 문제점은 아기걸음단계에서 구한 $m = \lceil \sqrt{n} \rceil$ 개의 모듈로 지수연산값을 계산하여 저장하고 있어야만 하며, 이로 인해 각 j걸음에서 탐색하는데 과다한 시간과 메모리를 필요로 한다는 점이다. 만약, RSA-100과 같이 n이 십진수 100자리이면 $m = \lceil \sqrt{n} \rceil$ 은 십진수 50자리로 이 데이터를 모듈로 지수연산 계산, 저장 및 탐색에는 현실적으로 불가능하다고 할 수 있다. 또 다른 문제점은 거인 1명이 단독으로 걸어가는 형태이다. 만약, 거인 2^k 명이 동시에 걸어가는 형태로 확장시킬 수 있다면 거인걸음 수행횟수를 $1/2^k$로 획기적으로 줄일 수 있을 것이다.

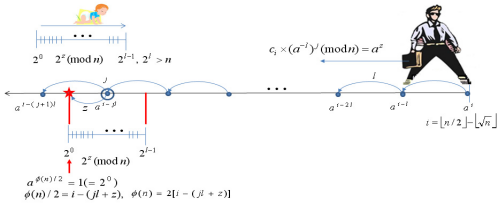
Ⅲ. 역 아기걸음 – 2^k-ary 성인걸음 알고리즘

본 장에서는 아기걸음-거인걸음 알고리즘을 변형시킨 방법으로 이산 대수를 계산하여 $\phi(n)$을 찾는 방법을 제안한다. 제안된 방법의 특징은 다음 전략을 적용한다.

[전략 1] $a^{\phi(n)} = a^{\phi(n)/2} \equiv 1 \pmod n$에서 $\phi(n)/2$을 찾는 것은 아기걸음-거인걸음 이산대수의 $a^0 \equiv 1 \pmod n$을 찾는 개념과 동일하다. 따라서 아기걸음-거인걸음 이산대수를 적용한다.

[전략 2] 아기걸음-거인걸음의 보폭 $m = \lceil \sqrt{n} \rceil$을 성인걸음 (adult-step) 보폭 $l, n < a^l$로 축소시켜 저장될 메모리 공간을 축소시킨다. 이는 성인 1명이 수행하는 기본적인 아기걸음-성인걸음법 (basic baby-step adult-step method)이 된다.

$n = p \times q, (p \neq q)$에서 $p < q$인 경우, $p < \sqrt{n}$, $\sqrt{n} < q$로 $p + q > 2 \lfloor \sqrt{n} \rfloor$이다. 왜냐하면 $p = q$인 경우 $p + q = 2\sqrt{n}$이기 때문이다. 따라서 $\phi(n) \leq n - 2 \lfloor \sqrt{n} \rfloor$, $\phi(n)/2 \leq \lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor$으로 결정할 수 있다. 즉, $\phi(n)/2$의 상한 (upper limit)은 $\lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor$이다.

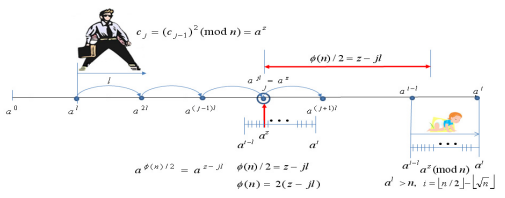


[전략 3]아기걸음-거인걸음은 $[0, m-1]$의 $a^i \pmod n$을 구한 다음 $a^m \equiv c_m \pmod n$에 대해 a^{-m}의 역함수를 구하여 b에서 0으로 역방향으로 $-jm$ 걸음을 걷는 방식이다. 반면에, 제안된 방법은 $[a^{i-m}, a^i], i = \lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor$에 대해 $a^z \pmod n$을 구하고, $a^l \pmod n$ 부터 시작하여 $a^{jl} \pmod n$로 l에서 i로 순방향으로 걸어가면서 $\phi(n)/2 = z - jl$을 구한다. 이를 역 아기걸음-2^0 성인걸음 (reverse baby- step 2^0 adult-step)이라 하자.

[전략 3]은 역함수를 구하지 않는 장점이 있다.

$a^m \equiv c_m \pmod n$의 역함수 a^{-m}은 확장된 유클리드 알고리즘을 적용하지 않고 식 (5)로도 구할 수 있다.

$$(kn+1) \pmod{c_m} = 0, (c_m)^{-1} = (kn+1)/c_m \qquad (5)$$

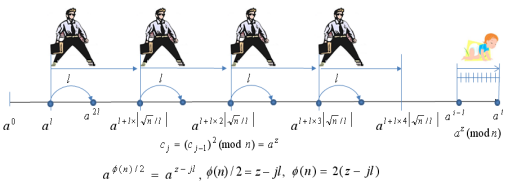


$\phi(n)2$의 하한 (lower limit)은 $l(p)=l(q)=2$인 [11,97]의 경우 $\lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor - 0.66 \lfloor \sqrt{n} \rfloor$, $l(p)=2$, $l(q)=3$인 경우 $\lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor - 3.84 \lfloor \sqrt{n} \rfloor$, $l(p)=l(q)=3$인 [101,997]의 경우 $\lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor - 0.73 \lfloor \sqrt{n} \rfloor$ 이다.

$10 \lfloor \sqrt{n} \rfloor < (p+q) \le \lfloor \sqrt{n} \rfloor$ 이므로 $5 \lfloor \sqrt{n} \rfloor < (p+q)/2 \le \lfloor \sqrt{n} \rfloor$ 이다. 이로부터 $\lfloor n/2 \rfloor - 4 \lfloor \sqrt{n} \rfloor < \phi(n)/2 \le \lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor$ 을 적용해 2등분, 4등분, 8등분 등으로 계속해서 k회 양분하면 2^k명이 동시에 수행하는 2^k-ary로 확장된다. 이는 $l \le j \le l+4 \lfloor \sqrt{n}/l \rfloor$ 에 대해 수행하면 된다. 여기서는 4등분한 4중 탐색을 예를 들어 설명한다.

[전략 4] $l \le j \le l+4 \lfloor \sqrt{n}/l \rfloor$ 에 대해 $j_1 = l, j_2 = l+2 \lfloor \sqrt{n}/l \rfloor$ 이면 2^1 성인걸음법으로 확장할 수 있으며, $j_1 = l, j_2 = l+ \lfloor \sqrt{n}/l \rfloor, j_3 = l+ 2 \lfloor \sqrt{n}/l \rfloor, j_4 = l+3 \lfloor \sqrt{n}/l \rfloor$ 이면 2^2-ary 성인걸음법으로 확장된다. 이와 같이 k회 양분하면 2^k명이 동시에 수행하는 2^k-ary 형태가 된다.

[전략 4]는 모듈러 지수연산법 (modular exponentiation)에서 기본적인 방법이 이진법 (binary method)을 확장한 방법이 2^k-ary 법 개념을 적용하였다.[13,14]



[전략 5]성인걸음 보폭 l을 최대로 할 수 있는 a의 값은 2이다. 따라서 $a=2$를 적용한다. 또한, 2를 적용하면 이진수를 사용하는 컴퓨터에서 계산도 용이하기 때문이다.

즉, 제안된 알고리즘은 역 아기걸음-2^k-ary성인걸음 알고리즘이라 할 수 있으며, 그림 2에서는 2^2-ary 성인걸음법을 제시하였다.

역 아기걸음-2^k-ary 성인걸음 알고리즘은 성인 걸음 단계의 수행횟수를 $1/2^k$로 감소시켜 $\phi(n)/2$이 $\lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor$에서 가장 먼 $\lfloor n/2 \rfloor - 5 \lfloor \sqrt{n} \rfloor$ 근방에 위치할 경우에도 매우 빠르게 찾을 수 있다. 또한, 아기걸음 단계의 저장 데이터를 $m = \lceil \sqrt{n} \rceil$에서 $l, a^l > n$로 줄이는 장점이 있다.

입력 : $a^b \equiv c \pmod n$, a, c, n, $a^l > n$
출력 : $\phi(n)$

/* 역-아기걸음: 수행복잡도 $O(l)$ */
$\phi(n) = 0$
$i = \lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor$
$z = i - l, 2^z \pmod n = c_z$
for $z = [i-l+1, l]$ do
  $c_z = c_{z-1} \times 2 \pmod n$
  if $c_z = 2^i, (0 \le i \le l-1)$ then
    $\phi(n) = 2(z-i)$, 알고리즘 종료
end

/* 2^k-ary 성인걸음 기준값 설정 */
$j_1 = l, j_2 = l + \lfloor \sqrt{n}/l \rfloor, j_3 = l+2 \lfloor \sqrt{n}/l \rfloor,$
$j_4 = l+3 \lfloor \sqrt{n}/l \rfloor$
$c_{j_k} = 2^{j_k} \pmod n$ 계산
if $c_{j_k} = 2^z$ then $\phi(n) = 2(z - j_k)$, 알고리즘 종료

/* 2^k-ary 성인걸음: 수행복잡도 $O(\sqrt{n}/l)$ */
$j_k = (j_k - 1) + l$
Do {
  $c_{j_k} = (c_{j_k-1})^2 \pmod n$ 계산
  if $c_{j_k} = c_z$ then $\phi(n) = 2(z - lj_k)$, 알고리즘 종료
  else $j_k = (j_k - 1) + l$.
} while $\phi(n) > 0$

그림 2. 역 아기걸음-2^k 성인걸음법
Fig. 2. Reverse Baby-step 2^k Adult-step Algorithm

만약, 제안된 알고리즘에 적용된 개념을 아기걸음-거인걸음 알고리즘에 적용하면 거인걸음 수행 횟수도 2^k로 줄일 수 있다. 그러나 아기걸음 단계에서 저장된 데이터 개수 $m=\lceil \sqrt{n} \rceil$은 여전히 줄이지 못한다.

Ⅳ. 실험 및 결과 분석

$\phi(n)/2$이 i로부터 멀리 떨어져 있는 $n=10967(11\times997), p+q-1=1007, \phi(n)=9960, \lfloor \sqrt{n} \rfloor=104, l=14$에 대해 기본적인 아기걸음-성인걸음법과 역 아기걸음-2^2성인걸음법을 적용한 결과는 표 1과 같다.

기본적인 아기걸음-성인걸음은 아기걸음 14회, 성인걸음 출발점 1회, 성인걸음 28회를 수행하여 총 43회 연산을 수행한다. 여기서 역함수 계산 횟수는 생략하였다. 참고로 $2^{-14} \pmod{10967}$은 $(10967\times1955+1)/5417=3958$로 1955회 수행해야 얻을 수 있다.

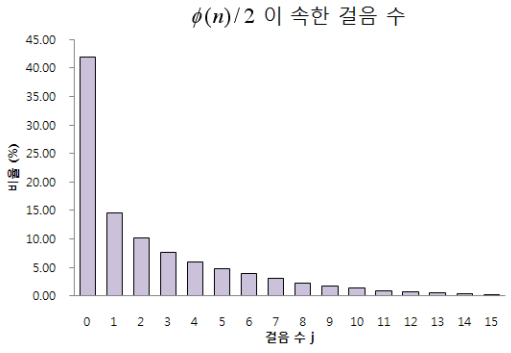
반면에, 역 아기걸음-2^2 성인걸음법은 아기걸음 14회, 성인걸음 출발점 4회와 성인걸음 4회로 총 22회를 수행한다. 즉, 역 아기걸음-2^2 성인걸음법은 역함수를 계산하는 횟수를 제외시키더라도 기본적인 아기걸음-성인걸음법의 수행횟수를 약 1/2로 줄일 수 있다.

그림 3은 $l(p)=l(q)=2$인 [11,97] 범위의 소수들에 대한 $n=p\times q, (p\neq q)$ 210개, $l(p)=2, l(q)=3$인 3,003개와 $l(p)=l(q)=3$인 10,153개로 총 13,366개의 데이터를 $l(n)=3$ 56개, $l(n)=4$ 887개, $l(n)=5$ 4493개, $l(n)=6$ 7930개로 분류하여 역 아기걸음-2^2성인걸음법으로 $\phi(n)$을 구한 횟수를 제시하였다.

그림 3에서 성인걸음 출발위치에서 41.88%를 찾을 수 있으며, 2걸음까지 걸으면 57.58%로 약 60%는 찾을 수 있다. 또한 성인걸음의 최대 걸음수 (수행횟수)는 보폭 l보다 작아 l회 이내에서 찾을 수 있다. 만약, n이 *RSA*와 같이 100자리 이상으로 큰 수이면 역 아기걸음-2^2성인걸음법은 수행 복잡도는 $O(\sqrt{n}/l)$로 아기걸음-거인걸음의 $O(\sqrt{n})$을 $1/l$회로 감소시킨다.

표 1. $n=10967$의 $\phi(n)$ 탐색법 비교
Table 1. Comparison of $\phi(n)$ Search Methods for $n=10967$

구분	기본적인 아기걸음-성인걸음법		역 아기걸음-2^2성인걸음법	
	k	$c=2^k \pmod n$	i	c
아기걸음	0	1	**5365** $(i-l)$	**4652**
	1	2	5366	9304
	2	4	5367	7641
	3	8	5368	4315
	4	16	5369	8630
	5	32	5370	6293
	6	64	5371	1619
	7	**128**	5372	**3238**
	8	256	5373	6476
	9	512	5374	1985
	10	1024	5375	3970
	11	2048	5376	7940
	12	4096	5377	4913
	13	8192	5378	9826
	14	5417	5379 (i)	8585
$2^{-14}=2^{4966}=3958 \pmod n$ $i=5379,\ 2^{5379}=8685 \pmod n$			$j_1=14, 2^{14}=5417 \pmod n$ $j_2=112, 2^{112}=1247 \pmod n$ $j_3=224, 2^{224}=8662 \pmod n$ $j_4=336, 2^{336}=9986 \pmod n$	
성인걸음	j	$8685\times3958^j \pmod n$	jl	$2^l\times2^{jl} \pmod n$
	1	4652	28	7164
			126	9621
			238	5228
			350	4918
	2	9990	42	6142
			140	6370
			252	3282
			364	1963
	3	4385	56	8303
			154	4108
			266	1087
			378	6548
	4	6036	70	1684
			168	993
			280	9967
			392	**3238**
	...	...		
	28	**128**(2^7)		
$\phi(n)/2=5379-(28\times14+7)$ $\phi(n)=2\times4980=9660$			$\phi(n)/2=5372-392$ $\phi(n)=2\times4980=9660$	



j	$l(n)$				계	비율
	3 $l=[8,10]$	4 $l=[10,14]$	5 $l=[14,17]$	6 $l=[17,20]$		
0	46	325	1221	4006	5598	41.88 %
1	10	166	571	1195	1942	14.53 %
2	0	133	444	782	1359	10.17 %
3	0	108	372	543	1023	7.65 %
4	0	82	319	390	791	5.92 %
5	0	51	295	300	646	4.83 %
6	0	20	277	223	520	3.89 %
7	0	1	236	169	406	3.04 %
8	0	1	174	132	307	2.30 %
9	0	0	145	83	228	1.71 %
10	0	0	141	47	188	1.41 %
11	0	0	100	25	125	0.94 %
12	0	0	77	25	102	0.76 %
13	0	0	65	10	75	0.56 %
14	0	0	41	0	41	0.31 %
15	0	0	15	0	15	0.11 %
계	56	887	4493	7930	13366	

그림 3. 역 아기걸음-2^2 성인걸음법의 $\phi(n)/2$ 탐색 걸음수
Fig. 3. The number of step to $\phi(n)/2$ for reverse baby-step 2^2 adult-step method

V. 결론

본 논문은 합성수 n을 사용하는 비대칭키 RSA의 $\phi(n)$을 $a^{\phi(n)}=a^{\phi(n)/2}\equiv 1(\mathrm{mod}\, n)$의 이산대수를 빠르게 찾는 알고리즘을 제안하였다.

제안된 알고리즘은 아기걸음-거인걸음의 이산대수법에 대해 첫 번째로, 거인 보폭 $m=\lceil \sqrt{n} \rceil$을 $l, a^l > n$으로 감소시켜 저장 메모리 용량과 탐색 시간을 줄였다. 두 번째로, $a=2$로 설정하여 성인걸음 보폭 l을 최대로 크게 하였다. 세 번째로, 아기걸음 $[0,l-1]$ 범위를 역으로 $[i-l,i], i=\lfloor n/2 \rfloor - \lfloor \sqrt{n} \rfloor$을 구하였다. 네 번째로, $2^0=1$명이 수행하는 방식을 모듈러 지수연산법의 2^k-ary 방법을 적용하여 2^k명이 동시에 수행하는 방법으로 확장시켰다. 이와 같은 방법을 적용한 결과 $\phi(n)$을 빠르게 구할 수 있음을 보였다.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," 2nd Ed., MIT Press and McGraw-Hill. pp. 887 - 896, 2001.

[2] D. R. Stinson, "Cryptography: Theory and Practice," 3rd ed., London, CRC Press, 2006.

[3] B. Raiter, "How the RSA Cipher Works", http://www.tutorialized.com/tutorial/How-the-RSA-Cipher-Works/42395, 2009.

[4] M. Seysen, "A probabilistic factorization algorithm with quadratic forms of negative discriminant", Mathematics of Computation, Vol. 48, No. 178, pp. 757 - 780, 1987.

[5] C. P. Schnorr, "Refined analysis and improvements on some factoring algorithms", Journal of Algorithms, Vol. 3, No. 2, pp. 101 - 127, 1982.

[6] Wikipedia, "Integer Factorization," http://en.wikipedia.org/wiki/Integer_factorization, 2014.

[7] Wikipedia, "RSA Factoring Challenge," http://en.wikipedia.org/wiki/RSA_Factoring_challenge, 2014.

[8] K. Ford, "The Number of Solutions of $\phi(x)=m$", Annals of Mathematics, Vol. 150, No. 1, pp. 283-311, 1999.

[9] A. Stein and E. Teske, "Optimized Baby step-Giant step Methods," Journal of the Ramanujan Mathematical Society, Vol. 20, No. 1, pp. 1-32, 2005.

[10] D. C. Terr, "A modification of Shanks' Baby-step Giant-step algorithm," Mathematics of Computation, Vol. 69, pp. 767-773, 2000.

[11] J. Pollard, "Monte Carlo methods for index computation mod p", Mathematics of Computation, Vol. 32, 1978.

[12] A. A. Razborov and S. Rudich, "Natural proofs", Journal of Computer and System Sciences, Vol. 55, pp. 24-35, 1997.

[13] S. Bruce, "Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed.", Wiley, 1996.

[14] S. T. Klein, "Should One Always Use Repeated Squaring for Modular Exponentiation?", Information Processing Letters, Vol. 106, Issue. 6, pp. 232-237, 2008.

저자 소개

이 상 운(정회원)

- 1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007년 2월 : 국립 원주대학 여성교양과 조교수
- 2007년 3월 ~ 현재 : 강릉원주대학교 멀티미디어공학과 부교수
- 관심분야 : 소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 그래프 알고리즘
- E-Mail : sulee@gwnu.ac.kr