The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 15, No. 2, pp.23-29, Apr. 30, 2015. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2015.15.2.23
JIIBC 2015-2-4

대칭키 해독을 위한 아기걸음 2^k-ary 성인걸음 알고리즘

Baby-Step 2^k-ary Adult-Step Algorithm for Symmetric-Key Decryption

이상운*

Sang-Un Lee*

요 약 $a^b \equiv c(\mathrm{mod}\, p)$에서 a,c,p가 주어졌을 때 b를 구하는 이산대수 문제를 푸는 아기걸음-거인걸음 알고리즘은 p를 $m = \lceil \sqrt{p} \rceil$개의 원소를 가진 m개의 블록으로 분할하고 거인 1명이 보폭 m으로 단방향으로만 a^0로 걸어가면서 찾는 방법이다. 본 논문은 기본적으로 p를 $p/l, a^l > p$로 분할하고, 성인 1명이 보폭 l로 단방향으로 걸어가는 방법으로 변형시켰다. 또한, 성인 2^k명이 동시에 걸어가면서 b를 빠르게 찾는 방법으로 확장시켰다. 제안된 알고리즘을 $1 \le b \le p-1$의 범위에서 $2^k, (k=2)$를 적용한 결과 기본적인 성인걸음수의 1/4로 감소시키는 효과를 얻었다. 결론적으로, 제안된 알고리즘은 아기걸음-거인걸음 알고리즘의 보폭 수를 획기적으로 단축시킬 수 있었다.

Abstract The baby-step giant-step algorithm seeks b in a discrete logarithm problem when a,c,p of $a^b \equiv c(\mathrm{mod}\, p)$ are already given. It does so by dividing p by m block of $m = \lceil \sqrt{p} \rceil$ length and letting one giant walk straight toward a^0 with constant m strides in search for b. In this paper, I basically reduce $m = \lceil \sqrt{p} \rceil$ to $p/l, a^l > p$ and replace a giant with an adult who is designed to walk straight with constant l strides. I also extend the algorithm to allow 2^k adults to walk simultaneously. As a consequence, the proposed algorithm quarters the execution time of the basic adult-walk method when applied to $2^k, (k=2)$ in the range of $1 \le b \le p-1$.In conclusion, the proposed algorithm greatly shorten the step number of baby-step giant-step.

Key Words : Discrete logarithm, Discrete exponentiation, Multiplicative order, Euler's totient function, Baby-step giant-step

Ⅰ. 서 론

암호는 대칭키와 비대칭키로 분류된다. 대표적인 비대칭키 RSA의 n은 합성수 (composite number)로 2개 소수 p,q를 선택하여 $n = p \times q$로 쉽게 만들고, p,q는 없앤다. 송신자는 메시지 m에 대해 $e < n, \gcd(e, \phi(n)) = 1$을 선택하여 암호화된 코드 $c = m^e (\mathrm{mod}\, n)$로 변환시켜 전송하며, 수신자는 개인키 d를 이용하여 $m = c^d (\mathrm{mod}\, n)$로 해독한다.[1-5] 여기서 $ed (\mathrm{mod}\, \phi(n)) = 1$이며, (e,n)은 공개된다. $\phi(n)$은 n과 서로소인 개수로 오일러 totient 함수이다.[6] 합성수 n의 $\phi(n) = (p-1)(q-1) = (n+1) - (p+q)$이며, $d = e^{-1} (\mathrm{mod}\, \phi(n))$으로 $\phi(n)$만 알고 있으면 해독할 수 있다. 비대칭키는 n을 p,q로 소인수분해 하기 어렵다는데 기반하고 있다. 대표적인 대칭키 DSA

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 : 2014년 10월 28일, 수정완료 : 2015년 3월 30일
게재확정일자 : 2015년 4월 10일

Received: 28 October, 2014 / Revised: 30 March, 2015 / Accepted: 10 April, 2015
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

(digital signature algorithm)는 미 연방정부 표준으로 채택되어 활용되고 있다. DSA는 ElGamal 알고리즘에 기반하며, ElGamal 알고리즘은 이산대수 (discrete logarithm) 문제에 기반한 공개키 암호방식으로 암호화와 서명 알고리즘으로 구성되어 있다. 여기서 암호화 알고리즘은 Diffie-Hemann 키 교환 방식을 채택하고 있다.[7-9] 대칭키에서는 합성수 n 대신 소수 p를 사용한다. 따라서 $\phi(p)=p-1$에 의거 합성수와 소수는 식 (1)이 성립한다.

$$a^{\phi(n)} = a^{\phi(n)/2} \equiv 1 (\mathrm{mod}\, n) \ : \text{합성수}$$
$$a^{\phi(p)} \equiv 1 (\mathrm{mod}\, p), a^{\phi(p)/2} \equiv \pm 1 (\mathrm{mod}\, p) : \text{소수} \qquad (1)$$

대칭키는 $a^b \equiv c (\mathrm{mod}\, p)$를 적용하며, a, c, p가 주어졌을 때 b를 구하는 문제는 $b = \log_a c$로 풀 수 있어 이산대수 (discrete logarithm)라 한다.[2] 즉, 대칭키를 사용하는 암호화된 메시지를 해독하는 능력은 이산대수 문제를 얼마나 빨리 풀 수 있는가에 달려 있다.

이산대수 알고리즘으로는 아기걸음-거인걸음 (baby-step giant-step)과 Pollard의 rho 등이 있다.[2,10-12] 그러나 어떠한 알고리즘도 빠르게 해를 구하지 못하고 있다.[13]

가장 간단한 알고리즘으로 알려져 있는 아기걸음-거인걸음 알고리즘은 p을 $m = \lceil \sqrt{p} \rceil$ 개의 원소를 갖고 있는 m 개의 블록 (block)으로 분할하고, 아기걸음 단계에서는 첫 번째 블록의 m 개에 대해 보폭 1인 m 걸음에 대한 모듈러 지수연산을 수행한다. 다음으로 거인걸음의 보폭 시작점인 $a^m (\mathrm{mod}\, p)$의 역함수 $a^{-m} (\mathrm{mod}\, p)$을 유클리드 알고리즘으로 구한다. 마지막으로 거인걸음 단계에서는 보폭 m으로 m 걸음을 걸어가면서 j번째 걸음에서 첫 번째 블록의 i번째 값과 일치하면, $b = jm + i$로 결정한다.[10,11]

아기걸음-거인걸음 알고리즘의 문제점은 아기걸음단계에서 구한 $m = \lceil \sqrt{p} \rceil$ 개의 모듈로 값을 계산 및 저장하고 탐색하는데 과다한 시간과 메모리를 필요로 한다는 점이다. p가 큰 수이면 이 데이터를 모듈로 계산과 저장 및 탐색에는 현실적으로 불가능하다고 할 수 있다.

본 논문은 아기걸음-거인걸음 알고리즘의 모듈러 연산 횟수와 저장 데이터의 크기를 획기적으로 줄일 수 있는 알고리즘을 제안한다. 2장에서는 암호 해독 방법에 관한 관련 연구를 고찰한다. 3장에서는 아기걸음-4중 성인걸음 알고리즘을 제안하고 4장에서는 제안된 알고리즘의 성능을 검증하여 본다.

II. 대칭키와 이산대수 알고리즘

대칭키의 Diffie-Hellman 키 교환방식은 그림 1에 제시되어 있다.[7,8] 여기서 생성함수 G는 실제적으로 2 또는 5를 적용한다. 키 s를 해독하는 방법은 $Y_A = G^{X_A} (\mathrm{mod}\, p)$의 이산대수에서 X_A를 얻고, $s = Y_B^{X_A} (\mathrm{mod}\, p)$에 대입하면 된다.

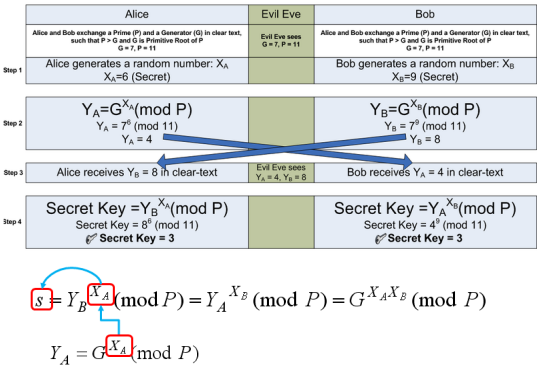


그림 1. Diffie-Hellman 키 교환 방식
Fig. 1. Diffie-Hellman key exchange

또 다른 방식은 그림 2의 ElGamal 알고리즘이 있으며, 이 방법에 근거하여 개발된 DSA가 미 연방정부 표준으로 채택되어 활용되고 있다.[9]

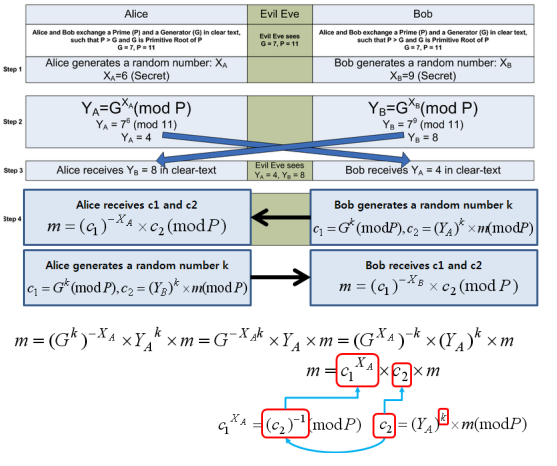


그림 2. ElGamal 알고리즘
Fig. 2. ElGamal algorithm

ElGamal 알고리즘을 해독하려면 역함수와 이산대수를 적용하여야 한다. 여기서 n은 소수로 $a^b \equiv c(\mathrm{mod}\, n)$의 역함수는 식 (2)로 쉽게 풀 수 있다.

$$(a^b)\times(a^{-b}) = 1 = a^{\phi(n)},\ n=\text{소수이면},\ \phi(n)=n-1$$
$$a^{b+(-b)} = a^{n-1},\ a^{-b} = a^{(n-1)-b}(\mathrm{mod}\, n) \quad (2)$$

$a^b \equiv c(\mathrm{mod}\, p)$에서 b를 찾기 위한 이산대수 문제를 푸는 아기걸음-거인걸음 알고리즘은 그림 3에 제시되어 있다.[10,11]

입력 : a, c, p, 출력 : b
$a^b \equiv c(\mathrm{mod}\, p)$
$m = \lceil \sqrt{p} \rceil$ /* 블록 개수 결정

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

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

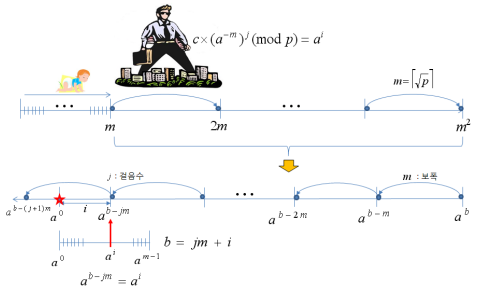


그림 3. 아기걸음-거인걸음 알고리즘
Fig. 3. Baby-step giant-step algorithm

이 알고리즘은 주어진 수 n을 $m = \lceil \sqrt{p} \rceil$ 개의 원소를 갖는 m개의 블록으로 분할한다. 아기걸음 단계에서는 첫 번째 블록에 대해 보폭 1, 걸음수 m인 $a^i = c_i (\mathrm{mod}\, p), 0 \le i \le m-1$을 계산하여 저장한다. 다음으로 거인걸음 시작점인 $a^m = c_m (\mathrm{mod}\, p)$에 대한 $(a^{-m}) = d (\mathrm{mod}\, p)$의 역함수를 구하여 거인걸음 단계에서는 식 (3)과 같이 보폭이 m인 걸음수 j까지 이동하였을 때 아기걸음의 i번째 값과 동일하면 $b = jm+i$로 결정한다. 왜냐하면, $a^b \times (a^{-m})^j = a^i$에서 $a^b = a^{mj+i}, b = mj+i$가 되기 때문에 (a^{-m})을 적용하여 거인걸음 단계에서 b를 구한다.

$$c \times d^j (\mathrm{mod}\, p) = c_i,\ 0 \le j \le m,\ b = m \times j + i \quad (3)$$

이 알고리즘의 단점은 아기걸음 단계에서 구한 m개의 c_i값을 거인이 걸을 때마다 동일한 값이 존재하는지 계속적으로 확인해야 하기 때문에, $\lceil \sqrt{n} \rceil$개의 메모리와 탐색에 과다한 시간이 소요된다. 또한 거인걸음의 수행 복잡도는 $O(\sqrt{p})$이다. 이의 단점을 보완하기 위해 아기걸음 단계를 수행하지 않는 방법이 Pollard의 rho 알고리즘이다.[12]

Pollard rho 알고리즘은 $\alpha^\gamma = \beta (\mathrm{mod}\, p)$에서 γ를 찾기 위해 $\alpha^a \beta^b \equiv \alpha^A \beta^B$의 합동을 찾아 $(B-b)\gamma = (a-A) (\mathrm{mod}\, p)$으로 구한다. 이 알고리즘은 m개의 데이터를 저장하지 않는 장점은 있지만 아기걸음-거인걸음과 동일한 $O(\sqrt{p})$의 수행시간 복잡도이다.

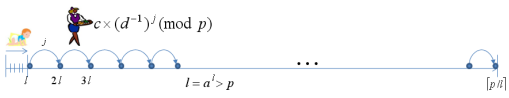
Ⅲ. 아기걸음-2^k 성인걸음 알고리즘

본 장에서는 아기걸음-거인걸음 알고리즘을 변형시킨 방법으로 이산 대수를 계산하는 방법을 제안한다. 제안된 방법의 특징은 다음과 같은 전략을 적용한다.

[전략 1] $a^b \equiv c(\mathrm{mod}\, p)$에서 a, c, p가 주어졌을 때 b를 구하기 위해 아기걸음의 걸음수를 $l, a^{l-1} < p < a^l$로 축소시켜 저장될 메모리 공간을 축소시킨다. 또한, 아기걸음-거인걸음 알고리즘의 거인걸음 보폭 $m = \lceil \sqrt{p} \rceil$을 성인걸음 보폭 l로 축소시킨다.

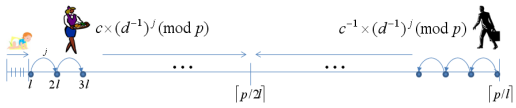
이 전략은 식 (4)에 제시되어 있으며, 2^0-성인걸음법으로 단방향으로만 진행한다. 이로 인해 성인걸음 단계의 수행 횟수 (걸음 수)가 거인걸음 단계보다 많아질 수 있다.

$$a^{l-1} < p < a^l,\ a^l (\mathrm{mod}\, p) = d,\ (a^{-l}) (\mathrm{mod}\, p) = d^{-1}$$
$$c \times (d^{-1})^j (\mathrm{mod}\, p) = c_i (= a^i),\ b = jl + i \quad (4)$$



[전략 1]의 단점을 보완하기 위해 식 (4)와 식 (5)를 적용하면 아기걸음 2^1-성인걸음법으로 양방향으로만 진행할 수 있다.

$$c^{-1}(\mathrm{mod}\,p) = a^{(p-1)-b}(\mathrm{mod}\,p)$$
$$(kp+1)(\mathrm{mod}\,c) = 0,\ c^{-1} = (kp+1)/c$$
$$c^{-1}\times c_l^j(\mathrm{mod}\,p) = c_i\,(=a^i)$$
$$b = jl+i = (p-1)-(jl+i) \qquad (5)$$



식 (4)+(5)는 개의 성인걸음수 범위에 대해 성인 2명이 양방향에서 걸어와 $p/2l$에서 만나는 형태이다. 따라서 $p/2l$을 다시 시작지점으로 설정하고 식 (6)과 식 (7)을 적용하면 아기걸음 2^1-성인걸음법으로 2지점에서 4명이 동시에 걸어오는 형태로 확장될 수 있다. 이를 계속 분할하면 $2^k, (k=0,1,2,\cdots)$ 지점 성인걸음법으로 확장할 수 있다.

$$m = p/2l,\ a^m \equiv c_m(\mathrm{mod}\,p)$$
$$c^{-1}\times c_m \times c_l^j(\mathrm{mod}\,p) = c_i\,(=a^i)$$
$$b = m+(jl-i)(\mathrm{mod}\,p-1) \qquad (6)$$

$$c^{-1}\times c_m \times (c_l^{-1})^j(\mathrm{mod}\,p) = c_i\,(=a^i)$$
$$b = m-(jl+i)(\mathrm{mod}\,p-1) \qquad (7)$$

[전략 2] 성인 1명이 걸어가는 방식을 2^k 지점에서 성인 $2^k\times 2$명이 동시에 걸어가는 형태로 확장하기 위해 p/l을 2^k개로 계속 분할한다. 즉, $1,2,4,8,16,\cdots$등으로 분할된다.

즉, 제안된 알고리즘은 아기걸음-2^k 성인걸음 알고리즘이라 할 수 있으며, 그림 4에서는 2^2-성인걸음법까지만 제시하였다.

입력 : $a^b \equiv c(\mathrm{mod}\,p),\ a,c,p,\ a^{l-1} < p < a^l$
출력 : b

if $c=1$ then $b=0$, 알고리즘 종료
else if $c=a^k$ then $b=k$, 알고리즘 종료
else /* 아기걸음단계: 수행 복잡도 $O(l)$
$c_0=1, c_1=a,\ i=2$
Do {
$c_i = (c_{i-1}\times a)(\mathrm{mod}\,p)$, (i, c_i) 저장
if $i=l$ then $a^{-l} \equiv c_l^{-1}(\mathrm{mod}\,p)$ 계산
} while $i=l$
$m = \lceil (p+1)/4 \rceil$, $n = \lceil (p+1)/2 \rceil$,
$o = \lceil 3(p+1)/4 \rceil$
$a^m \equiv c_m(\mathrm{mod}\,p), a^{-m} \equiv c_m^{-1}(\mathrm{mod}\,p)$ 계산
$a^n \equiv c_n(\mathrm{mod}\,p), a^{-n} \equiv c_n^{-1}(\mathrm{mod}\,p)$ 계산
$a^o \equiv c_o(\mathrm{mod}\,p), a^{-o} \equiv c_o^{-1}(\mathrm{mod}\,p)$ 계산

/* 성인걸음단계: 수행 복잡도 $O(\sqrt{p}/2l)$
$k=0$.
$c_{1k} = c(\mathrm{mod}\,p),\ c_{1k}^{-1} = c^{-1}(\mathrm{mod}\,p)$
$c_{2k} = c_{2k}^{-1} = (c_m\times c^{-1})(\mathrm{mod}\,p)$
$c_{3k} = c_{3k}^{-1} = (c_n\times c^{-1})(\mathrm{mod}\,p)$
$c_{4k} = c_{4k}^{-1} = (c_o\times c^{-1})(\mathrm{mod}\,p)$
if $c_{1k} = c_i$ then $b=i$
else if $c_{1k}^{-1} = c_i$ then $b=i$
else if $c_{2k} = c_i$ then $b=m+i$
else if $c_{2k}^{-1} = c_i$ then $b=m-i$
else if $c_{3k} = c_i$ then $b=n+i$
else if $c_{3k}^{-1} = c_i$ then $b=n-i$
else if $c_{4k} = c_i$ then $b=o+i$
else if $c_{4k}^{-1} = c_i$ then $b=o-i$.
if $b>0$ then 알고리즘 종료.

$k=1$.
do {
$c_{1k} = c_{1(k-1)}\times c_l^{-1}(\mathrm{mod}\,p)$
$c_{1k}^{-1} = c_{1(k-1)}^{-1}\times c_l(\mathrm{mod}\,p)$
for $j=2,3,4$
$c_{jk} = (c_{j(k-1)}\times c_l)(\mathrm{mod}\,p)$
$c_{jk}^{-1} = (c_{j(k-1)}^{-1}\times c_l^{-1})(\mathrm{mod}\,p)$
end
if $c_{1k} = c_i$ then $b=lk+i$
else if $c_{1k}^{-1} = c_i$ then $b=lk+i$
else if $c_{2k} = c_i$ then
$b = m+(lk-i)(\mathrm{mod}\,p-1)$
else if $c_{2k}^{-1} = c_i$ then $b=m-(lk+i)$
else if $c_{3k} = c_i$ then
$b = n+(lk-i)(\mathrm{mod}\,p-1)$
else if $c_{3k}^{-1} = c_i$ then $b=n-(lk+i)$
else if $c_{4k} = c_i$ then
$b = o+(lk-i)(\mathrm{mod}\,p-1)$
else if $c_{4k}^{-1} = c_i$ then $b=o-(lk+i)$
else $k=k+1$.
} while $b>0$

그림 4. 아기걸음-2^k 성인걸음법
Fig. 4. Baby-step 2^k adult-step algorithm

아기걸음-2^k 성인걸음 알고리즘은 성인 걸음 단계의 수행횟수를 2^k로 감소시키는 특징과 더불어 b가 p에 근

접한 경우에도 매우 빠르게 찾을 수 있으며, $1 \le b \le p-1$의 전체 범위에서도 평균적으로 수행횟수를 2^k로 감소시켜 수행 복잡도는 $O(2^k n/l)$이다. 또한, 아기걸음 단계의 저장 데이터를 $m = \lceil \sqrt{p} \rceil$에서 $l, a^l > p$로 줄이는 장점이 있다.

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

Ⅳ. 실험 및 결과 분석

$a^b \equiv c(\mathrm{mod}\, p)$에서 $1 \le b \le p-1$에서 임의의 수를 선택한다. $p = 1019$, $\sqrt{p} = 31.92, m = \lceil \sqrt{p} \rceil = 12$, $2^{10} = 1024 > p, l = 10$인 경우에 다음 5개의 이산대수 값을 구하여 보자.

(1) $2^{165} \equiv 71(\mathrm{mod}\, 1019)$
(2) $2^{709} \equiv 301(\mathrm{mod}\, 1019)$
(3) $2^{501} \equiv 203(\mathrm{mod}1019)$
(4) $2^{551} \equiv 557(\mathrm{mod}1019)$
(5) $2^{1012} \equiv 207(\mathrm{mod}1019)$

위 5개 데이터 각각에 대한 역함수는 다음과 같이 계산된다.

(1) $2^{-165} = 2^{1018-165} = 2^{853} \equiv 244(\mathrm{mod}\, 1019)$
$(1019 \times 17 + 1)(\mathrm{mod}\, 71) = 0, (1019 \times 17 + 1)/71 = 244$
(2) $2^{-709} = 2^{1018-709} = 2^{309} \equiv 325(\mathrm{mod}\, 1019)$
(3) $2^{-501} = 2^{1018-501} = 2^{517} \equiv 763(\mathrm{mod}1019)$
(4) $2^{-551} = 2^{1018-551} = 2^{467} \equiv 236(\mathrm{mod}1019)$
(5) $2^{-1012} = 2^{1018-1012} = 2^6 \equiv 64(\mathrm{mod}1019)$

$1019/10 = 102, 102/2 = 51$로 아기걸음-성인걸음의 중간 기준값은 510이다. 따라서 $2^{510} \equiv 1017(\mathrm{mod}\, 1019)$, $2^{-510} = 2^{508} \equiv 509(\mathrm{mod}\, 1019)$ 이다.

아기걸음-거인걸음은 아기걸음 단계에서 다음 $m = 32$개의 모듈로 지수 연산 값들을 계산하여 저장한다. 즉 32회를 수행한다.

$2^0 \equiv 1(\mathrm{mod}\, 1019), 2^1 \equiv 2(\mathrm{mod}\, 1019)$
$2^2 = (2^1)(\mathrm{mod}\, 1019) \times 2 \equiv 4,\ \ 2^3 = (2^2)(\mathrm{mod}\, 1019) \times 2 \equiv 8$
$2^4 = 8 \times 2(\mathrm{mod}\, 1019) = 16,\ \ 2^5 = 16 \times 2(\mathrm{mod}\, 1019) = 32$
$2^6 = 32 \times 2(\mathrm{mod}\, 1019) = 64,\ \ 2^7 = 64 \times 2(\mathrm{mod}\, 1019) = 128$
$2^8 = 128 \times 2(\mathrm{mod}\, 1019) = 256,$
$2^9 = 256 \times 2(\mathrm{mod}\, 1019) = 512$
$2^{10} = 512 \times 2(\mathrm{mod}\, 1019) = 5,\ \ 2^{11} = 5 \times 2(\mathrm{mod}\, 1019) = 10$
…

$(0,1), (1,2), (2,4), (3,8), (4,16), (5,32), (6,64), (7,128),$
$(8,256), (9,512), (10,5), (11,10), (12,20), (13,40), (14,80),$
$(15,160), (16,320), (17,640), (18,261), (19,522), (20,25),$
$(21,50), (22,100), (23,200), (24,400), (25,800), (26,581),$
$(27,143), (28,286), (29,572), (30,125), (31,250)$

다음으로 $m = 32$에 대한 2회 연산 수행으로 모듈로 지수연산과 역함수를 구한다.

$2^{32} = (2^{31}) \times 2(\mathrm{mod}\, 1019) = 250 \times 2 = 500(\mathrm{mod}\, 1019)$
$2^{-32} = 2^{986} = 858(\mathrm{mod}\, 1019)$

$2^{165} \equiv 71(\mathrm{mod}\, 1019)$의 165를 구하기 위해, 거인걸음 단계에서는 $71 \times 858^5(\mathrm{mod}\, 1019) = 32(= 2^5)$으로 6회를 수행하여 $b = 32 \times 5 + 5 = 165$를 구한다. 결국, $2^{165} \equiv 71(\mathrm{mod}\, 1019)$을 구하기 위해 32+2+6=40회를 수행한다. 여기서는 거인걸음 단계 6회 수행 각각에 대해 아기걸음 단계의 32개 데이터 중에서 동일한 값이 있는지 검증하는 $6 \times (32-2) = 180$회의 과정은 생략하였다. 여기서 $32-2$를 한 이유는 $2^0 = 1, 2^1 = 2$는 저장하지 않아도 되기 때문이다.

아기걸음-성인걸음 알고리즘은 $2^{165} \equiv 71(\mathrm{mod}\, 1019)$을 구하기 위해, 아기걸음 단계에서는 $(2,4), (3,8), (4,16), (5,32), (6,64), (7,128), (8,256), (9,512)$ 의 8회 모듈러 지수 연산을 수행하여 값을 저장한다.

다음으로, $k = 2$의 성인걸음 기준값인 $2^{10} \equiv 5(\mathrm{mod}\, 1019)$, $2^{-10} = 204(\mathrm{mod}\, 1019), 2^{-165} = 2^{1018-165} = 2^{853} \equiv 244(\mathrm{mod}\, 1019)$를 구하고, $2^{255} \equiv 241(\mathrm{mod}\, 1019)$, $2^{510} \equiv 1017(\mathrm{mod}\, 1019), 2^{765} \equiv 537(\mathrm{mod}\, 1019)$으로 3회의 모듈러 지수연산과 2회의 역함수를 구한다.

성인걸음 단계에서는 다음 $2^k (k = 2)$가지 경우 수 중 가장 먼저 해를 구하면 알고리즘이 종료되며, 10회 수행으로 해를 구하였다.

$71\times 204^{16}(\mathrm{mod}\,1019)=32(=2^5),\ \ b=10\times 16+5=165$

$244\times 5^{17}(\mathrm{mod}\,1019)=32(=2^5),\ \ b=10\times 17-5=165$

$244\times 241\times 5^{93}(\mathrm{mod}\,1019)=4(=2^2)$

$b=255+(93\times 10-2)=1183(\mathrm{mod}\,1018)=165$

$244\times 241\times 204^{9}(\mathrm{mod}\,1019)=1(=2^0)$

$b=255-(9\times 10+0)=165$

$244\times 1017\times 5^{68}(\mathrm{mod}\,1019)=128(=2^7)$

$b=510+(68\times 10-7)=1183(\mathrm{mod}\,1018)=163$

$244\times 1017\times 204^{34}(\mathrm{mod}\,1019)=32(=2^5)$

$b=510-(34\times 10+5)=165$

$244\times 537\times 5^{42}(\mathrm{mod}\,1019)=4(=2^2)$

$b=765+(34\times 10-5)=1183(\mathrm{mod}\,1018)=163$

$244\times 537\times 204^{60}(\mathrm{mod}\,1019)=1(=2^0)$

$b=765-(60\times 10+0)=165$

즉, 2^2 성인걸음법은 8+5+10=23회로 아기걸음-거인걸음 알고리즘의 40회를 13회 감소시킬 수 있었으며, 아기걸음 단계에서 저장된 데이터 탐색에도 $8\times 10=$ 80회로 100회 감소시켰다.

위 5개 데이터에 대해 거인걸음과 성인걸음의 수행횟수만을 단순 비교한 결과는 표 1과 같다. 여기서는 아기걸음 단계의 데이터 $m-2=30$, $l-2=8$를 탐색하는 횟수는 생략하였다.

표 1. 연산횟수 비교
Table 1. Comparison of the number of computation

$2^b\equiv c(\mathrm{mod}\,1019)$		아기걸음-거인걸음 알고리즘	아기걸음-2^2 성인걸음 알고리즘
b	c	거인걸음 수행횟수	2^2 성인걸음 수행횟수
165	71	$b=32\times 5+5$ (6회)	$b=255-(9\times 10+0)=165$ (10회)
501	203	$b=32\times 15+21$ (16회)	$b=510-(10\times 0+9)$ (1회)
551	557	$b=32\times 17+7$ (18회)	$b=510+(5\times 10-9)$ (6회)
709	301	$b=32\times 22+5$ (23회)	$b=765-(5\times 10+6)$ (6회)
1012	207	$b=32\times 31+20$ (32회)	$b=1018-(0\times 10+6)$ (1회)

표 1에서 $b=p/4l$ 근방인 165에 대해서는 성인걸음이 거인걸음에 비해 많은 수행횟수를 보여주고 있다. $b=p/2l$, $b=3p/4l$과 $b=p/l$ 근방에서는 성인걸음이 매우 빠르게 해를 구할 수 있음을 알 수 있다. 즉, 공개키 암호를 아기걸음-거인걸음 알고리즘으로 해독하기 불가능하도록 가능한 $p/2l\le b\le p/l$에서 선택할 경우에도 제안된 알고리즘은 빠르게 암호를 해독할 수 있음을 알 수 있다.

만약, p가 큰 수인 경우 암호를 빠르게 해독하려면 제안된 알고리즘의 2^k 성인걸음법을 $k=3,4,5,\cdots$로 확장시키면 보다 빠르게 해독이 가능하다.

V. 결론

본 논문은 소수 p를 사용하는 대칭키의 $a^b\equiv c(\mathrm{mod}\,p)$ 이산대수를 빠르게 찾는 알고리즘을 제안하였다. 제안된 알고리즘은 아기 걸음수 $m=\lceil\sqrt{p}\,\rceil$를 $l, a^l>p$로 감소시켜 저장 메모리 용량과 탐색 시간을 줄였다. 또한, 본 논문에서는 $2^2=4$명의 성인이 동시에 걸어가는 방식으로 확장시켜 해를 빠르게 구할 수 있음을 보였다.

일반적으로 대칭키에 적용되는 큰 수 p에 대해 $a^b\equiv c(\mathrm{mod}\,p)$의 개인키 b는 2^k의 $k=3,4,5,\cdots$로 성인의 인원수를 확장시키면 보다 빠르게 찾을 수 있을 것이다.

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] M. Alfred, P. C. Oorschot, and S. A. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[4] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol. 21, No. 2, pp. 120 - 126, 1978.

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

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

[7] D. Boneh, "The Decision Diffie - Hellman Problem",

Lecture Notes in Computer Science Vol.1423, pp. 48 - 63, 1998.

[8] W. Diffie and M. E. Hellman, "New Directions in Cryptography", IEEE Trans. on Information Theory, Vol. IT-22, pp. 644-654, 1976.

[9] T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transactions on Information Theory, Vol.31 No. 4, pp. 469 - 472. 1985.

[10] 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.

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

[12] J. Pollard, "Monte Carlo Methods for Index Computation mod p", Mathematics of Computation, Vol.32, 1978.

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

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 ~ 2015년 3월 : 강릉원주대학교 멀티미디어공학과 부교수
- 2015년 4월 ~ 현재 : 강릉원주대학교 멀티미디어공학과 정교수

<관심분야> : 소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 그래프 알고리즘