Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.5, Oct. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.5.1121

GNFS를 위한 향상된 다항식 선택 기법*

김 수 리,[1†] 권 지 훈,[1] 조 성 민,[1] 장 남 수,[2] 윤 기 순,[3] 김 창 한,[4‡] 박 영 호,[2] 홍 석 희[1]
[1]고려대학교 정보보호연구원, [2]세종사이버대학교, [3]엔에스에이치씨, [4]세명대학교

Enhanced Polynomial Selection Method for GNFS*

Suhri Kim,[1†] Jihoon Kwon,[1] Sungmin Cho,[1] Nam Su Chang,[2] Kisoon Yoon,[3]
Chang Han Kim,[4‡] Young-Ho Park,[2] Seokhie Hong[1]
[1]Center for Information Security Technologies(CIST), Korea University,
[2]Sejong Cyber University, [3]NSHC, [4]Semyung University

요 약

RSA 암호 시스템은 가장 널리 사용되는 공개키 암호 알고리즘 중 하나이며, RSA 암호 시스템의 안전성은 큰 수의 인수분해의 어려움에 기반을 둔다. 따라서 RSA 암호 시스템의 합성수 n을 인수분해하려는 시도는 계속 진행 중에 있다. General Number Field Sieve는 현재까지 알려진 가장 빠른 인수분해 방법이고, RSA-704를 인수분해 하는데 사용된 소프트웨어인 CADO-NFS도 GNFS를 기반으로 설계되어 있다. 그러나 CADO-NFS는 다항식 선택 과정에서 입력된 변수로부터 항상 최적의 다항식을 선택하지 못하는 문제점이 있다. 본 논문에서는 CADO-NFS의 다항식 선택 단계를 분석하고 중국인의 나머지 정리와 유클리드 거리를 사용하여 다항식을 선택하는 방법을 제안한다. 제안된 방법을 이용하면 기존의 방법보다 좋은 다항식이 매번 선택되며, RSA-1024를 인수분해 하는데 적용할 수 있을 것으로 기대한다.

ABSTRACT

RSA cryptosystem is one of the most widely used public key cryptosystem. The security of RSA cryptosystem is based on hardness of factoring large number and hence there are ongoing attempt to factor RSA modulus. General Number Field Sieve (GNFS) is currently the fastest known method for factoring large numbers so that CADO-NFS – publicly well-known software that was used to factor RSA-704 – is also based on GNFS. However, one disadvantage is that CADO-NFS could not always select the optimal polynomial for given parameters. In this paper, we analyze CADO-NFS's polynomial selection stage. We propose modified polynomial selection using Chinese Remainder Theorem and Euclidean Distance. In this way, we can always select polynomial better than original version of CADO-NFS and expected to use for factoring RSA-1024.

Keywords: GNFS, CADO-NFS, polynomial selection

I. Introduction

Public key cryptosystem is useful for sending information via insecure channel and its security is based on hardness of solving number theoretic problems. For example, RSA cryptosystem is based on hardness of factoring large numbers and Elliptic Curve Cryptography is based on discrete logarithm problem. Among other public key cryptosystems, RSA cryptosystem is popular for its simplicity.

Received(09. 01. 2016), Accepted(09. 27. 2016)
* 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대 정보·컴퓨팅기술개발사업의 지원을 받아 수행된 연구입니다(No. NRF-2014M3C4A7030649)
† 주저자, suhrikim@gmail.com
‡ 교신저자, chkim@semyung.ac.kr(Corresponding author)

In RSA cryptosystem, public modulus n is chosen as product of two distinct primes p,q and primes p,q are kept secret[1]. It is easy to generate such number n but hard to find p,q such that $n=pq$ for n sufficiently large. Since RSA cryptosystem can be broken by factoring n, integer factorization is one of main topics for research. Currently, the best known factoring algorithm is the General Number Field Sieve (GNFS)[9]. Factoring based on GNFS is recently performed by T. Kleinjung et al. in 2009 for factoring RSA-768[6], and also by S. Bai et al. in 2012 for factoring RSA-704[5].

CADO-NFS is one of publicly well known factoring tool based on GNFS and it was used to factor RSA-704 and RSA-220[12]. It selects polynomial using Kleinjung's second algorithm. However CADO-NFS could not always select the optimal polynomial for given parameters.

This paper targets CADO-NFS's polynomial selection stage. We analyze the possible disadvantage that can occur when using CADO-NFS directly. By modifying CADO-NFS's polynomial selection procedure we can now select better polynomial compare to polynomial selected by CADO-NFS.

This paper is organized as follows. In Section 2, we present backgrounds for polynomial selection. Then in Section 3, we explain Kleinjung's second algorithm - algorithm for polynomial selection which is currently used in CADO-NFS. In Section 4, we analyze disadvantage in CADO-NFS and propose our new way of selecting polynomial. In Section 5, experimental results of our polynomial selection is presented.

II. Background

In this Section, underlying principle GNFS is first presented. Next, we describe earlier method for polynomial selection. Lastly, we introduce two measurements to score quality of selected polynomial.

2.1 General Number Field Sieve

Modern factoring algorithm is based on 'difference of squares'. It factors by selecting random integers x,y such that $x \neq y \bmod n$ and $x^2 \equiv y^2 \bmod n$. Then by computing $\gcd(x+y,n)$ and $\gcd(x-y,n)$ we can obtain non-trivial factors of n with high probability. This concept of using 'square of random number' is also employed in Quadratic Sieve(QS) and General Number Field Sieve.

GNFS is currently fastest known method for factoring numbers over 110 digits[9]. It can be seen as generalization of QS. QS uses quadratic polynomials whereas GNFS uses polynomials of higher degree so that a square is not produced directly in Z_n as before. It uses two polynomial $f(x),g(x)$ having common root $m \bmod n$. Let α and β be roots of $f(x),g(x)$ not in Z_n, respectively, and consider rings $Z[\alpha]$, $Z[\beta]$. Goal in GNFS is to find (a,b) pairs such that $a-b\alpha$ and $a-b\beta$ are smooth over chosen basis of primes. We say that an element is smooth if all of its factors are member of our chosen basis of primes.

We collect (a,b) pairs where $\prod(a-b\alpha)=X^2$ for $X \in Z[\alpha]$ and $\prod(a-b\beta)=Y^2$ for $Y \in Z[\beta]$. Consider homomorphisms from ring $Z[\alpha]$ and $Z[\beta]$ to Z_n that maps α and β to m. Then there exist $x,y \in Z_n$ such that X,Y are mapped to x,y respectively. Hence $x^2 \equiv y^2 \bmod n$ is again obtained and non-trivial

factors of n can be found with high probability.

Usually, GNFS is divided into four stages - polynomial selection, sieving, linear algebra, square roots - but we focus on first two stages of GNFS. Namely, polynomial selection where we select $f(x)$ and $g(x)$ and sieving where we collect pairs (a,b). It is known that sieving takes over 90% of total time for factoring using GNFS, and choice of polynomial dramatically affect time to complete sieving. In next Section, we briefly describe earlier method for polynomial selection.

2.2 Classical Polynomial Selection

Classic way to generate polynomial is using base-m method[10]. The base-m method expresses number n to be factored as $n = m^d + c_{d-1}m^{d-1} + ... + c_0$, such that $|c_i| \leq \frac{m}{2}$ for each i and generates monic polynomial $f(x)$ of degree d and monic polynomial $g(x) = x - m$ of degree 1 where $m \bmod n$ is common root of $f(x)$ and $g(x)$. To reduce size of coefficient of $f(x)$ this method is modified to select non-monic polynomial $f(x) = \sum_{i=0}^{d} c_i x^i$ such that $n = \sum_{i=0}^{d} c_i m^i$. We choose m to be close to $(n/c_d)^{1/d}$[4]. If c_{d-2} is not small enough, try another c_d. Otherwise we optimize the generated polynomial pair.

2.3 Quantifying Quality of Polynomials

In Section 2.1, we emphasized the importance of selecting good polynomial. In this Section, we introduce measurements to evaluate whether generated polynomial can be considered 'good'. Note that main goal of sieving stage is to collect many pairs (a,b) such that $a - b\alpha$ is smooth over chosen basis of primes where α is root of $f(x)$. Usually basis of primes consist of 'small' primes and small value norm of $f(x)$ is more likely to be factored by small primes and hence more likely to be smooth over basis than larger norm. Thus one of measurement to quantify quality of polynomial is called 'lognorm'. Lognorm is logarithmic average of polynomial values across sieving region and lognorm of function is computed as below[11], where s refers to skewness of sieving region, calculated by ratio of a,b[4].

$$\frac{1}{2}\log\left(s^{-d}\int_0^{2\pi}\int_0^1 F^2(s\cos\theta,\sin\theta)\, r^{2d+1} dr d\theta\right).$$

Hence small lognorm means size of polynomial is small so that it is more likely to be smooth over our chosen basis of primes. Thus we are searching for polynomial with smaller lognorm as possible. We may assume that size of $g(x)$ does not vary much across sieving regions than $f(x)$ due to the fact that $g(x)$ is linear. So in practice, we only consider lognorm of $f(x)$.

Combining $f(x)$ and $g(x)$ we can approximate number of sieving reports as equation below[11].

$$\frac{6}{\pi^2}\iint_{\Omega}\rho\left(\frac{\log|F(x,y)|+\alpha(F)}{\log B_1}\right)\rho\left(\frac{\log|G(x,y)|+\alpha(G)}{\log B_2}\right)dxdy$$

Above measurement is called 'murphy E' of polynomials. Since collecting as many relation as possible is goal in sieving stage, larger murphy E implies it is likely to have large number of sieving reports in sieving stage. Hence we focus on selecting $f(x), g(x)$ pairs with larger murphy E value.

III. Kleinjung's Second Algorithm and Its Implementation

In this Section, we describe Kleinjung's second algorithm for selecting polynomial and how it is implemented in CADO-NFS. The algorithm gives an efficient way to control size of c_{d-2} while producing polynomial with large skewness.

3.1 Kleinjung's Second Algorithm

Kleinjung's second algorithm extends current GNFS's polynomial selection procedure[2]. Instead of using base-m method as in GNFS, Kleinjung's second algorithm uses base-(m_1,m_2) method and generates non-monic polynomial $f(x)$ of degree greater than 1 and non-monic polynomial $g(x)$ of degree 1. Base-(m_1,m_2) represents the number n to be factored as $n=\sum_{i=0}^{d} c_{d-i}m_1^{d-i}m_2^i$ and selects $f(x), g(x)$ of the form $f(x)=\sum_{i=0}^{d} c_i x^i$ and $g(x)=m_2x-m_1$ having $m_1m_2^{-1} \bmod n$ as common root.

Let Q be set of small primes and P be set of primes in $[B,2B]$ for some bound B so that $P\cap Q=\varnothing$ and primes in P are larger than primes in Q. Leading coefficient m_2 of $g(x)$ is chosen to be of form $m_2=p_1p_2q$ where $q=\prod q_i$ for $q_i\in Q$ and $p_1,p_2\in P$.

Kleinjung's second algorithm generates polynomial $f(x)$ with smaller c_{d-2} than as in first algorithm[2,4]. Method of selecting smaller c_{d-2} is described below.

In equation $n=\sum_{i=0}^{d} c_{d-i}m_1^{d-i}m_2^i$, we expand to

$$\begin{aligned} n &= \sum_{i=0}^{d} c_{d-i}m_1^{d-i}m_2^i \\ &= c_d\left(m_1^d+\frac{c_{d-1}}{c_d}m_1^{d-1}m_2\right)+\sum_{i=0}^{d-2} c_i m_1^i m_2^{d-i} \end{aligned} \quad (1)$$

and express (1) as in terms of d-th degree by using

$$c_d\left(m_1+\frac{c_{d-1}}{dc_d}m_2\right)^d = c_d m_1^d + c_{d-1}m_1^{d-1}m_2 + m_2^2R_0,$$

where $R_0=c_d\sum_{i=2}^{d}{}_dC_i m_1^{d-i}\left(\frac{c_{d-1}}{dc_d}m_2\right)^i$.

Then equation (1) can be rewritten as

$$n=c_d\left(m_1+\frac{c_{d-1}}{dc_d}m_2\right)^d - m_2^2R_0 + \sum_{i=0}^{d-2} c_i m_1^i m_2^{d-i}.$$

To eliminate the denominator, we multiply $d^d c_d^{d-1}$ on both side and obtain

$$\begin{aligned} & d^d c_d^{d-1} n \\ &= (dc_d m_1 + c_{d-1}m_2)^d - d^d c_d^{d-1} m_2^2 R_0 \\ &\quad + d^d c_d^{d-1}\sum_{i=0}^{d-2} c_i m_1^i m_2^{d-i} \end{aligned} \quad (2)$$

Let $\tilde{n}=d^d c_d^{d-1}n$ and $\widetilde{m}=dc_d m_1+c_{d-1}m_2$. By simplifying terms that has degree lower than d, equation (2) can be represented as

$$\tilde{n}=\widetilde{m}^d+m_2^2\tilde{R}.$$

Taking modulo by m_2^2, we have

$$\tilde{n}=\widetilde{m}^d \bmod m_2^2. \quad (3)$$

By obtaining solution for $\widetilde{m}$ in (3), $\widetilde{m}$ equals to value who satisfies $\widetilde{m}=\tilde{n}^{\frac{1}{d}} \bmod m_2^2$. Since $dc_d m_1+c_{d-1}m_2=\widetilde{m}$, and two values d, c_d are known, by taking modulo dc_d we have

$$c_{d-1}m_2 \equiv \widetilde{m} \bmod dc_d \quad (4)$$

By multiplying inverse of m_2 modulo dc_d on both side of equation (4), c_{d-1} is determined. Due to the fact that we obtained c_{d-1} by taking dc_d as modulus, $|c_{d-1}| \le dc_d$ meaning that we have c_{d-1} with appropriate size.

3.2 Implementation in CADO-NFS

The most direct way to solve equation (3) is to first solve $\tilde{n} = x^d \bmod p$ for each prime p in m_2, and solve $\tilde{n} \equiv x^d \bmod p^2$ by using Hensel's lemma. After obtaining d solutions for each prime p, by using Chinese Remainder Theorem(CRT) we can obtain solution for $\tilde{n} = x^d \bmod m_2^2$. To have larger cardinality of polynomial set to select the best polynomial, CADO-NFS uses only the primes p having d roots for $\tilde{n} = x^d \bmod p$. Hence if m_2 consists of l primes, this means that by using CRT we have d^l number of roots in total. However, since we only need root $\widetilde{m}$ close to $\tilde{n}^{\frac{1}{d}}$[4], calculating all d^l solution is tedious. Hence to avoid using CRT and solve equation (3), CADO-NFS uses collision between roots to search for $\widetilde{m}$ close to $\tilde{n}^{\frac{1}{d}}$ efficiently.

Given input n, d, c_d, let $\widetilde{m}_0$ be integral part of $\tilde{n}^{\frac{1}{d}}$. We first compute roots for $\tilde{n} \equiv (\widetilde{m}_0 + r)^d \bmod p$ where $p \in P$. Then roots r are lifted to r_p modulo p^2 and recorded as (p, r_p). Second, we search for collision on r. This means that we are searching for r and r' where r is root of $\tilde{n} \equiv (\widetilde{m}_0 + r)^d \bmod p_1^2$ and r' is root of $\tilde{n} \equiv (\widetilde{m}_0 + r')^d \bmod p_2^2$ and $r = r'$. Then we can obtain $\tilde{n} \equiv (\widetilde{m}_0 + r)^d \bmod p_1^2 p_2^2$ without using CRT. If collision is not detected, then we use primes q_i in set Q and let $m_2 = p_1 p_2 q$ where $q = \prod q_i$.

Note that we already have recorded roots (p, r_p) so that we only need to calculate roots for primes in Q. For each $q \in Q$ solve $\tilde{n} \equiv (\widetilde{m}_0 + r_q)^d \bmod q^2$. Then for each q and for all $p \in P$ we calculate $i_p \in [0, q^2)$ by solving $r_q + i_p q^2 \equiv r_p \bmod p^2$ such that equation $\tilde{n} \equiv (\widetilde{m}_0 + r_q + i_p q^2)^d \bmod p^2$ is satisfied. Then pair (p, i_p) is recorded and we search for collision on i between (p_1, i) and (p_2, i). If collision occurs

$$\tilde{n} \equiv (\widetilde{m}_0 + r_q + iq^2)^d \bmod p_1^2 p_2^2 q^2 \quad (5)$$

is satisfied. Again, CRT is not used to calculate common roots between p_1^2, p_2^2, q^2, and obtained $\widetilde{m} = \widetilde{m}_0 + r_q + iq^2$ is reasonably close to $\tilde{n}^{\frac{1}{d}}$ since $\widetilde{m}_0$ was chosen to be integral part of $\tilde{n}^{\frac{1}{d}}$.

As a summary, total procedure of Kleinjung's second algorithm is described in Algorithm 1.

Algorithm 1

Input : n, d, c_d, prime sets P, Q
Output : (m_1, m_2) for the base expansion of n

1. Compute $\tilde{n}, \widetilde{m}_0$ from n, d, c_d
2. For $p \in P$ do
   2.1 Solve r_p in $\tilde{n} \equiv (\widetilde{m}_0 + r_p)^d \bmod p^2$
   2.2 Record (p, r_p) and detect collisions on r_p
3. For $q \in Q$ do
   3.1 Solve r_q in $\tilde{n} \equiv (\widetilde{m}_0 + r_q)^d \bmod q^2$
   3.2 For each r_q do

3.2.1 For $p \in P$ do
3.2.1.1 For each r_p do
3.2.1.2. Solve i_p in
$\tilde{n} \equiv (\tilde{m}_0 + r_q + i_p q^2) \bmod p^2 q^2$
3.2.1.3 Record (p, i_p) and detect collisions on i_p

IV. Proposed Method

In this Section, we analyze the disadvantage in CADO-NFS and propose new method to find $\tilde{m}$ that can generate polynomial with larger murphy E value than other roots. We used CRT to find root of $\tilde{n} \equiv x^d \bmod m_2^2$ and polynomial is selected from $\tilde{m}$ with smallest $|\tilde{n}^{\frac{1}{d}} - \tilde{m}|$.

4.1 Analysis of CADO-NFS

The major disadvantage of using collision is that there is a probability of existence of root closer to $\tilde{n}^{\frac{1}{d}}$ than root founded by using collision. More specifically, CADO-NFS uses one solution of $\tilde{n} \equiv x^d \bmod m_2^2$ to generate polynomial not by searching all d^l roots but by fixing $\tilde{m}_0$ and detecting collision between i in equation (5). Hence solution that generates better polynomial can be missed. Since good selection of polynomial can reduce time for searching relation in sieving stage [4,8], not checking candidates that has higher probability of generating polynomial of larger murphy E might be a problem. In fact for RSA-768 with degree 6 and ad=265482057982680, CADO-NFS found
20567264542987682470035388620696444486268244739208 12
as $\tilde{m}$ and generated polynomial with 8.53×10^{-18} as murphy E value. But polynomial that was actually used for factoring RSA-768 has $\tilde{m}$ with
20567226635308133413947387352976918391973791082516 32
and 6.99×10^{-17} as murphy E. Therefore we focused on checking all roots of $\tilde{n} \equiv x^d \bmod m_2^2$.

4.2 Proposed Method

In order to check all d^l roots, CRT is necessary. Naive way of checking all d^l roots and observe which root generates better polynomial is to generate polynomial for each root, calculate lognorm, and compare if lognorm is smaller than lognorm calculated previously. However, calculation of lognorm for one polynomial takes 4.52×10^{-7}s so that calculation of lognorm for all d^l polynomial takes $4.52 \times 10^{-7} \times d^l$s. If 11 primes and degree 6 are used as in RSA-768 this will take 12 hours for just searching 15 pairs of p_1, p_2 while it takes 38.5 minutes when using collision.

When considering RSA-1024, it is too inefficient to check all roots for each pair of (p_1, p_2, q), Since exact value of leading coefficient of $f(x)$ is unknown, this leads to at least $d^l \times$(admax-admin)/incr amount of checking where admax and admin denotes maximal and minimal value of c_d we are searching for, respectively.

Thus instead of generating all d^l polynomials to calculate lognorm, we only generated polynomial for roots with smallest $|\tilde{n}^{\frac{1}{d}} - \tilde{m}|$. This means that instead of generating polynomial and calculate lognorm for each root, only euclidean distance is calculated for each root and polynomial is generated only once for $\tilde{m}$ with smallest $|\tilde{n}^{\frac{1}{d}} - \tilde{m}|$. The total procedure for selecting polynomial is described in Algorithm 2.

Algorithm 2

Input : n, d, c_d, prime sets P, Q

incr: number added to c_d after each round

S : Set of 15 polynomials recorded as $(\widetilde{m}, f(x), g(x)$,lognorm)

Output : $f(x), g(x)$ with lowest lognorm among founded polynomials

1. For each c_d
  1.1 Compute $\tilde{n}$, $\widetilde{m}_0$ from n, d, c_d
  1.2. For $p \in P$ do
    1.2.1 Solve r_p in $\tilde{n} \equiv (\widetilde{m}_0 + r_p)^d \bmod p^2$
    1.2.2. Record (p, r_p) and detect collision on r_p
  1.3. For $q \in Q$ do
    1.3.1 Solve x in $\tilde{n} \equiv x^d \bmod q^2$
  1.4. For (p_1, p_2) $(p_1, p_2 \in P)$ such that p_1, p_2 has collision
    1.4.1 Solve x in $\tilde{n} \equiv x^d \bmod p_i^2$ for $i = 1,2$
    1.4.2. Solve $\tilde{n} \equiv x^d \bmod m_2^2$ by CRT where $m_2 = p_1 p_2 \prod q$
    1.4.3. For each solution $\widetilde{m}$ in $\tilde{n} \equiv x^d \bmod m_2^2$ calculate $D = |\tilde{n}^{\frac{1}{d}} - \widetilde{m}|$
    1.4.4. Output smallest D
  1.5. Calculate lognorm for (p_1, p_2) and $\widetilde{m}$ with smallest D
  1.6. If lognorm $\leq$ max lognorm in S
    1.6.1. Insert-sort $(\widetilde{m}, f(x), g(x)$,lognorm)
2. If c_d+incr $\leq$ admax goto step 1
3. Output $f(x), g(x)$ with lowest lognorm

In Step 1.2 in Algorithm 2, we modified CADO-NFS's function 'collision_on_sq'. Instead of moving to function 'match' in 'hash-add' if collision is found, we recorded prime pair (p_1, p_2) for detected collision. With this recorded prime pair and primes in q we solve $\tilde{n} \equiv x^d \bmod k^2$ for prime $k | m_2$. At Step 1.6 we calculate euclidean distance D between solution x of $\tilde{n} \equiv x^d \bmod m_2^2$ and $\tilde{n}$.

For each prime pair, polynomial is generated for $\widetilde{m}$ having smallest D and lognorm is calculated. Then $(\widetilde{m}, f(x), g(x)$,lognorm) is insert-sorted if calculated lognorm is smaller than maximal lognorm of set calculated previously. In practice, we kept only 15 values of lognorm. Hence regardless of number of prime pairs and number of rounds occurs, our Algorithm 2 outputs 15 polynomials.

V. Implementation Results

Experiments were performed using gcc version 4.9.2 with processor Intel(R) Core(TM) i5-4690K CPU at 3.5 GHz with 8GB RAM. Size optimization and root optimization are same for both group. We used cado-2.1.1 version for optimization. Although latest version of CADO-NFS is 2.2.0, released in Dec. 2015, only optimization part of polynomial selection has been changed. Since we are only considering the generation of polynomial which is same in both versions, this experimental result will be same for cado-2.2.0 also.

Single round implies that test was done for one ad value and multiple rounds implies that test was done for range of ad values.

Note that 'ad' refers to leading coefficient of $f(x)$ and CADO-NFS uses this value as input parameter. 'lq' means number of primes used in m_2, and 'pairs' means number of (p_1, p_2) that has collision. Hence total number of roots generated by l lq and n pairs is $n \times d^l$.

Below is single-round test result for RSA-768 using degree 6, $P=[100000,200000]$.

Below is single-round test result for RSA-768 using parameters that includes parameters that were actually used for factoring RSA-768[6]. Namely, degree 6 and $P=[900,1800]$ were used.

Multiple rounds were tested with RSA-704 number with degree 6, increment 60 and $P=[900,1800]$.

In summary, the results of experiments show that by using our modified version of polynomial selecting method, we can search for polynomial with murphy E larger or equal to murphy E founded by CADO-NFS in practical time. This is guaranteed by the fact that we are actuatlly checking all the roots of equation (5).

Table 1. Implementation results for RSA-768 single round

	CADO-NFS	CRT
lq=8, ad=265482058083480, 5 pairs		
MurphyE	3.27×10^{-17}	3.27×10^{-17}
Time	24 min	7 min
lq=6, ad=265482058083480, 6 pairs		
MurphyE	1.32×10^{-17}	1.32×10^{-17}
Time	25.7 min	6.7 min

Table 2. Implementation results for RSA-768 single round

	CADO-NFS	CRT
lq=11, ad=265482057982680, 15 pairs		
MurphyE	8.53×10^{-18}	6.98×10^{-17}
Time	28.8 min	70.5 min

Table 3. Implementation results for RSA-704 multiple round

	CADO-NFS	CRT
lq=7, ad=1614120 ~ 1614180 , 2 pairs		
MurphyE	1.99×10^{-16}	1.99×10^{-16}
Time	19 min	6.7 min
lq=7, ad=1614180 ~ 1614300 12 pairs		
MurphyE	2.11×10^{-16}	2.11×10^{-16}
Time	63.6 min	6.6 min

In other words, we never generate polynomial with lower murphy E value than CADO-NFS. Since generating polynomial having larger murphy E as possible is most important in polynomial selection stage, it can be said that our modified version selects better polynomial in reasonable time.

VI. Conclusion

In this paper we propose modified version of polynomial selection in CADO-NFS. CADO-NFS uses collision to avoid CRT and find common roots between primes. In this way CADO-NFS generates polynomial with moderate - not best - murphy E. However largest murphy E value is necessary for shorter sieving time. Hence we used CRT to generate all roots but estimate performance of root by euclidean distance instead of calculating lognorm. In this way polynomial with murphy E value larger than CADO-NFS is guaranteed in practical time and can be expected to implemented in polynomial selection for RSA-1024.

References

[1] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signature and Public-Key Cryptosystems," ACM, vol.21(2), pp.120-126, 1978

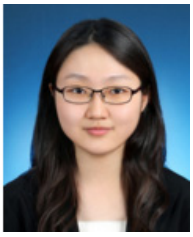
[2] T. Kleinjung. "Polynomial selection". In CADO workshop on integer factorization, INRIA Nancy, 2008. http://cado.gforge.inria.fr/workshop/slides/kleinjung.pdf.

[3] T. Kleinjung. "On polynomial selection for the general number field sieve". Mathematics of Computation, pp. 2037 - 2047, 2006.

[4] S. Bai "Polynomial Selection for the Number Field Sieve" , Ph.D. Thesis ,The

Australian National University,2011
〔5〕 S. Bai, E. Thom´e, P. Zimmermann. Factorisation of RSA-704 with CADO-NFS. Report, 2012.http://eprint.iacr.org/2012/369.pdf.
〔6〕 T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thom´e, J. W. Bos, P. Gaudry, A. Kruppa,P. L. Montgomery, D. A. Osvik, H. J. J. te Riele, A. Timofeev, and P. Zimmermann. "Factorization of a 768-bit RSA modulus". CRYPTO '10, vol.6223 LNCS, pp 333 - 350, 2010
〔7〕 A. K. Lenstra and H. W. Lenstra, Jr., editors. "The Development of the Number Field Sieve", vol. 1554 of Lecture Notes in Mathematics. Springer, 1993.
〔8〕 Matthew E. Briggs "An Introduction to the General Number Field Sieve", Master Thesis. Virginia Polytechnic Institute and State University. April, 1998.
〔9〕 A.K. Lenstra, H.W. Lenstra, Jr., editors, "The Development of the Number Field Siece", Lecture Notes in Mathmatics, vol.1554, 1993.
〔10〕 B. A. Murphy, R. P. Brent, "On Quadratic Polynomials for the Number Field Sieve", CATS'98, pp 199-231, 1998.
〔11〕 B. A. Murphy, "Polynomial Selection for the Number Field Sieve Integer Factorization Algorithm", Ph.D. Thesis, The Australian National University, 1999.
〔12〕 S. Bai, P. Gaudry, A. Kruppa, E. Thome, P. Zimmermann "Factorization ofr RSA-220 with CADO-NFS", 2016

〈저자소개〉



김 수 리 (Suhri Kim) 학생회원
2014년 2월: 고려대학교 수학과 학사
2014년 8월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 부채널 공격, 공개키 암호시스템

권 지 훈 (Jihoon Kwon) 학생회원
2010년 2월: 고려대학교 수학과 학사
2010년 3월~현재: 고려대학교 정보보호대학원 석박사 통합과정
〈관심분야〉 정보보호, 공개키 암호시스템

조 성 민 (Sung Min Cho) 학생회원
2008년 2월: 광운대학교 수학과 학사 졸업
2011년 8월: 고려대학교 정보경영공학전문대학원 석사 졸업
2011년 8월~현재: 고려대학교 정보보호대학원 정보보호학과 박사과정
〈관심분야〉 부채널 공격, 공개키 암호 알고리즘, 암호구현

장 남 수 (Nam Su Chang) 정회원
2002년 2월: 서울 시립대학교 수학과 이학사
2004년 8월: 고려대학교 정보보호 대학원 공학석사
2010년 2월: 고려대학교 정보경영공학전문대학원 공학박사
2010년 7월~현재: 세종사이버대학교 정보보호학과 조교수
〈관심분야〉 암호칩 설계 기술, 부채널 공격, 공개키 암호 알고리즘, 공개키 암호 암호분석

윤 기 순 (Kisoon Yoon) 정회원
1998년 8월: 경희대학교 수학과 이학사
2007년 8월: 고려대학교 정보보호학과 공학석사
2013년 11월: Université de Caen 수학과 이학박사
2013년 11월~현재: 엔에스에이치씨 암호기술팀 팀장
〈관심분야〉 정수론, 암호학, 정보보호

김 창 한 (Chang Han Kim) 종신회원
1985년 2월: 고려대학교 수학과 이학사
1987년 2월: 고려대학교 수학과 이학석사
1992년 2월: 고려대학교 수학과 이학박사
1992년 8월~현재: 세명대학교 정보통신학부 교수
〈관심분야〉 정수론, 공개키암호, 암호프로토콜

박 영 호 (Young-Ho Park) 종신회원
1990년 2월: 고려대학교 수학과 이학사
1993년 2월: 고려대학교 수학과 이학석사
1997년 2월: 고려대학교 수학과 이학박사
2002년 3월~현재: 세종사이버대학교 정보보호학과 교수
〈관심분야〉 공개키 암호, 암호 프로토콜, 부채널 공격, 암호안전성평가

홍 석 희 (Seokhie Hong) 종신회원
1995년 2월: 고려대학교 수학과 학사
1997년 2월: 고려대학교 수학과 석사
2001년 8월: 고려대학교 수학과 박사
1999년 8월~2004년 2월: (주) 시큐리티 테크놀로지스 선임연구원
2003년 8월~2004년 2월: 고려대학교 정보보호기술연구센터 선임연구원
2004년 4월~2005년 2월: K.U.Leuven, ESAT/SCD-COSIC 박사후연구원
2005년 3월~2013년 8월: 고려대학교 정보보호대학원 부교수
2013년 9월~현재: 고려대학교 정보보호대학원 정교수
〈관심분야〉 대칭키·공개키 암호 분석 및 설계, 컴퓨터 포렌식