2000 년 한국정보처리학회 추계 학술발표논문집 제 7 권 제 2 호

소수성 시험방법의 개선방안 연구

김영진*, 홍순좌*, 박중길*
*국가보안기술연구소
e-mail : yjkim67@etri.re.kr

The Study of a Improving Method on a Primality Test

Young-Jin Kim*, Soon-Jwa Hong*, Joong-Gil Park*
*National Security Research Institute

요 약

이 논문에서는 먼저 소수성 시험 알고리즘의 기본 개념이 되는 Fermat 의 정리를 살펴본다. 그리고, 가장 널리 사용되고 구현이 용이한 소수성 시험 알고리즘인 Rabin-Miller 및 Rivest 알고리즘을 살펴보고, 이들 알고리즘의 수행 시간을 분석한다. 또한, Rivest 가 제시한 연구 결과를 바탕으로 소수성 시험 방법의 수행성능 향상 방안을 고려함으로써, Rivest 알고리즘을 효율적으로 구현하여 암호 시스템 등의 응용에 적용할 수 있도록 개선 방안을 제시한다.

1. 서론

소수를 구하는 문제는 수세기에 걸쳐 많은 수학자들의 관심을 끌고 있다. 13 세기 초에 Fibonacci 는 주어진 N 이 소수인지 여부를 $\sqrt{N}$ 개 만큼의 수로 나누어 보면 알 수 있다고 지적한 바 있다. 그 외에도 E. Lucas, M. Kraitchik, D. H. Lehmer 등이 Fermat 의 정리를 바탕으로 소수성 시험 방법을 개선 발전시켰다.

최근에 소수성 시험 방법이 전산과 암호 분야에서 관심을 끄는 이유는 RSA 와 같은 공개키 암호 시스템에서 소수가 사용되기 때문이다. 그렇기 때문에 소수 생성은 이러한 암호 시스템 구축에 필요한 주요 부분이다.

이 논문에서는 먼저 소수 판정 알고리즘의 기본 개념이 되는 Fermat 의 정리를 살펴본 뒤에, 가장 널리 사용되고 있고 구현이 용이한 소수성 시험 알고리즘인 Rabin-Miller 및 Rivest 알고리즘을 살펴보고, 이 알고리즘의 수행 시간을 분석한다. 또한, Rivest 가 제시한 연구 결과를 바탕으로 소수성 시험 단계의 수행성능 향상 방안을 고려함으로써 Rivest 알고리즘을 효율적으로 구현하여 암호 시스템 등의 응용에 적용할 수 있도록 구현 방안을 제시하고자 한다.

2. 소수성 시험

가. 개요

소수성 시험(Primality Test)은 주어진 수가 소수인지를 판단하는 시험을 말한다. 주어진 N 이 소수인지를 판단하는 가장 간단한 방법은 N 을 $\sqrt{N}$ 이하의 모든 수로 나눠보는 것이다[1].

정리 1. N 이 합성수이면, N 에 대하여 $p \le \sqrt{N}$ 인 소인수 p 가 존재한다.

길이가 n 비트인 N 이 주어졌을 때, $2^{n-1} \le N < 2^n$ 이고, 위의 방법을 사용하면 $\sqrt{N} \le \sqrt{2^n}$ 만큼 나누어야 하기 때문에 $O(2^{n/2})$의 수행 시간이 걸린다. 그렇기 때문에 n 이 커질수록 이 방법을 적용하는 것은 힘들다. 이보다 개선된 소수성 시험 방법이 p 가 소수이면 $1 \le a \le p-1$ 를 만족하는 모든 a 에 대하여 $a^{p-1} \equiv 1 \bmod p$ 라는 Fermat 의 정리를 이용하여 시험을 실시하는 것이다.

정리 2. (Euler 의 정리) 양의 정수 m 과 $(a, m) = 1$ 인 정수 a 에 대하여 $a^{\phi(m)} \equiv 1 \pmod m$ 이다.

정리 3. (Fermat 의 정리) p 가 소수일 때, $(a, p) = 1$ 인 정수 a 에 대하여 $a^{p-1} \equiv 1 \pmod p$ 이다.

Fermat 의 정리에 의하여 $1 \le a \le N-1$인 임의의 a 에 대해 $a^{N-1} \equiv 1 \bmod N$ 을 만족하면 N 이 "소수일 가능성이 높다"라고 말할 수 있게 된다. 이렇게 Fermat 의 정리를 소수성 시험에 적용하게 되면 효율적으로 소수를 판단할 수 있게 된다.

나. 소수의 분포

소수 $p(p>1)$는 1 과 자기 자신에 의해서만 나눠지는 정수를 말한다. Euclid 는 무한개의 소수가 존재한다는

것을 그의 저서(IX of Elements, Proposition 20)에서 다음과 같이 밝힌 바 있다.

> $p_1, p_2, \cdots, p_n$을 소수의 집합이라 하자. 그리고 정수 $N = p_1 p_2 \cdots p_n + 1$을 고려해보자. N은 소수의 곱으로 표현된다. 이 때, p를 $p|N$을 만족하는 소수라 하자. 그러면, $1 \le i \le n$에 대하여 $p_i \nmid N$이기 때문에 모든 i에 대하여 $p \ne p_i$가 성립한다. 그러므로, p는 소수 집합에 속하지 않는다. 이와 마찬가지로, 소수의 유한 목록이 완전하지 않기 때문에 소수는 무한하다.

소수는 정수의 특성을 파악하는데 사용되는 기본적인 요소이다. 또한, 소수와 관련해서 많은 문제들이 존재한다. 소수 목록은 어떻게 생성되는가? 알려진 것 중에 가장 큰 소수는 무엇인가? 임의의 수가 소수인지 합성수인지 어떻게 판단할 수 있는가? 언제 소수성을 판단할 수 없고, 언제 임의의 수가 소수인지 어떻게 확신할 수 있는가?

암호와 같은 많은 응용들이 큰 소수를 필요로 한다. 다행히도 크기가 큰 소수는 많아서, 소수를 찾을 때까지 적절한 크기의 난수를 발생시켜 시험하는 방법은 많은 시간을 필요로 하지는 않는다. 여기에서 소수의 분포를 예측하는데 소수 분포 함수 및 소수 정리를 사용한다. 소수 분포 함수 $\pi(N)$은 N 이하의 소수의 개수를 말한다. 예를 들어, 10 이하의 소수는 2, 3, 5, 그리고 7 이므로, $\pi(10)=4$ 이다. 다음의 소수 정리는 $\pi(N)$에 대한 근사값을 찾는데 사용한다[2].

정리 4. (소수 정리) $\lim_{N\to\infty} \frac{\pi(N)}{N/\ln N} = 1$

N 이 작더라도 $\pi(N)$은 $N/\ln N$ 에 아주 근접한다. 예를 들어, $N = 10^9$일 때, $\pi(N) = 50{,}847{,}478$ 이고 $N/\ln N = 48{,}254{,}942$ 으로 6%의 차이밖에 나지 않는다. 여기에서 우리는 이러한 소수 정리를 바탕으로 해서 임의로 선택한 정수 N 이 소수로 판명될 확률이 대략 $1/\ln N$ 이라고 말할 수 있다. 그러므로, N 과 같은 크기의 소수를 찾기 위해서는 N 근처에서 선택되는 정수 $\ln N$ 개 정도를 조사해야 한다. 만일, 100 자리 이하의 소수를 찾기 위해서는 $\ln 10^{100} \approx 230$ 개의 임의의 100 자리 크기의 수를 선택하여 소수성 시험을 실시해야 한다.

다. Rabin-Miller 소수성 시험

$N-1 = 2^s t$ (t 는 홀수)이라 하자. 주어진 a 에 대하여 다음과 같이 u_i를 계산할 수 있다.

$$u_0 = a^t \bmod N$$
$$u_{i+1} = u_i^2 \bmod N \qquad (0 \le i \le s-2)$$
$$u_s = u_{s-1}^2 = a^{2^s t} = a^{N-1} \bmod N$$

Fermat 의 정리를 사용하면, $u_s \equiv 1 \bmod N$일 때 N은 확률적으로 소수라고 말할 수 있다. 또한, $u_0 \equiv 1 \bmod N$ 이거나 $r < s$ 에 대하여 $u_r \equiv -1 \bmod N$ 이더라도 N은 확률적으로 소수라고 말할 수 있다. 이 개념을 사용하여 만들어진 RABIN-MILLER 알고리즘은 다음과 같다.

RABIN-MILLER(N, s)
1. $\{1, 2, \ldots, N-1\}$에서 난수 a 를 선택한다.
2. $N-1 = 2^s \cdot d$ 형태로 변환한다. 단, d 는 홀수이다.
3. $k = s$ 또는 $a_k \equiv 1 \pmod N$일 때까지 연속적으로 $a_0 = a^d \pmod N$, $a_1 = a_0^2 \pmod N$, ... , $a_k = a_{k-1}^2 \pmod N$을 계산한다.
4. **if** $k=s$ **and** $a_k \not\equiv 1 \pmod N$
5. **then return** '합성수'
6. **else if** $k=0$
7. **then return** '소수'
8. **else if** $a_{k-1} \not\equiv -1 \pmod N$
9. **then return** '합성수'
10. **else return** '소수'

RABIN-MILLER 에서 N 은 소수성 시험 대상이 되는 수이고, k 는 이 알고리즘을 반복해서 시험을 수행해야 할 회수를 나타낸다. N 이 홀수 소수일 경우에 RABIN-MILLER(N)은 항상 '소수'를 반환한다. N이 홀수 합성수일 때에는 $1 \le a \le N-1$인 모든 a 에 대해 최소한 3/4 의 확률로 '합성수'를 반환한다. 이 알고리즘은 $O((\ln N)^3)$ 비트 연산을 수행한다[3].

라. Rivest 소수성 시험

어떤 수 N 이 합성수이고 다음 식을 만족하면, N을 (2 기반의) 가상 소수(Pseudoprime)이라 한다.

$$2^{N-1} \equiv 1 \pmod N \qquad (1)$$

모든 소수들은 식(1)를 만족한다. 그러나, 몇 안 되는 합성수들이 식(1)을 만족하여 가상 소수가 된다. 만일 가상 소수가 아주 드물다면 일반적인 응용에서 사용할 수 있는 큰 소수를 (암호 사용을 위해서) 찾는 것은 단지 난수 N 을 발생시켜서 식(1)를 만족하는 어떤 수 N을 선택하는 것이 될 것이다.

이를 확인하기 위해서 Rivest 는 한 네트워크 상의 33 대 SUN Sparcstation 들을 이용하여 대략 718 백만 개의 256 비트 난수 값에 대한 "Small Divisor 시험"을 실시했다. 여기에 사용한 Small Divisor 시험이란 10,000 이하의 소수를 이용하여 시험하고자 하는 수를 차례로 나누어 보는 시험을 말한다.

Rivest 는 난수 값들이 Small Divisor 시험을 통과하면 그 다음에 식(1)에 대한 시험을, 식(1)를 만족시키면 그 다음으로 Rabin-Miller 의 확률적 소수 시험을 8 회 실시했다. 시험을 실시한 수들 중에서 43,741,404 개가 Small Divisor 시험을 통과했다. 또한, 그 들 중의 4,058,000 개가 식(1)를 만족시켰다. 그리고, 이 모든 수가 8 회의 Rabin-Miller 의 확률적 소수 시험을 통과했다. 즉, 가상 소수가 발견되지 않았다. 다시 말해서, Small Divisor 시험과 식(1)를 만족시키는 모든 수는 (확률적으로) 소수로 판단되었다. 그러므로, 적어도 Small Divisor 를 갖지 않는 수들 중에는 가상 소수가 매우 드물다고 경험적으로 말할 수 있다. Pomerance 도 많은 경험과 분석을 바탕으로 n 이하의 가상 소수의 개수가 많아야

$$n / L(n)^{1+o(1)}, \quad L(n) = \exp\frac{\log n \log\log\log n}{\log\log n}! \qquad (2)$$

이라고 추측했다.

이 추측이 맞고 추가로 식(2)에서 $o(1)$을 무시할 수 있다고 가정하면, 2^{256} 보다 작은 가상 소수의 개수는

많아야 4×10^{52} 이 된다고 추측할 수 있다. 반면에 256 비트 이하의 소수 개수는 대략 6.5×10^{74} 이 된다. 그러므로, Pomerance 의 추측이 맞다면(그리고 $o(1)$ 항을 무시하는 것이 가능해지면) 임의로 선택된 256 비트 수가 합성수이면서 식(1)를 만족시킬 가능성은 $1/10^{22}$ 보다 작다. 따라서, 실제 환경에서 가상 소수성(Pseudoprimality)은 소수성(Primality)를 충분히 보장할 수 있을 것이다. 이러한 결과는 앞서 제시한 이론적 추측과도 일치한다[4].

3. 난수 소수 생성

이 장에서는 난수 발생에 의한 소수 생성 방법을 적용할 경우에, Rivest 가 제시한 연구 결과를 바탕으로 난수 소수 생성 알고리즘의 수행 성능 향상 방안을 고려하고자 한다.

가. 난수 소수 생성 알고리즘 및 알고리즘 성능 향상 방안 고찰

N 과 k 가 주어졌을 때, N 이하의 난수를 발생하고 그 난수에 대한 소수성 시험을 거쳐 k 개의 소수를 생성하는 난수 소수 생성 알고리즘 RANDOM-PRIME 을 사용한다. RANDOM-PRIME 의 PRIME-TEST 는 2 장에서 보았듯이 Small Divisor 시험 단계, Rivest 의 Fermat 시험 단계 및 Rabin-Miller 소수성 시험 단계를 적용하여 수행해야 하지만, Rivest 의 연구 결과에 의해 Small Divisor 시험 단계와 Rivest 의 Fermat 시험 단계만을 실시하더라도 소수성 검사 결과에는 큰 영향이 없으므로 두 단계 만을 수행하도록 한다.

RANDOM-PRIME 알고리즘은 다음과 같다.

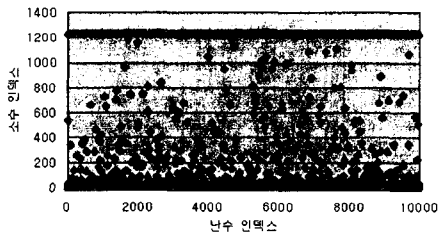
```
RANDOM-PRIME(N, k)
1. P[1...k] ← 0, i ← 0
2. repeat
3. do
4. j ← RANDOM(1, (N-1)/2)
5. p ← 2j + 1
6. if PRIME-TEST(p) = '소수'
7. then P[i] ← p, i ← i + 1
8. until i < k
9. return P
```

RANDOM-PRIME 은 N, k 를 입력으로 받아 N 이하의 k 개 소수 P 를 생성하여 반환한다. 이를 위해 라인 2 에서 8 까지의 루틴을 소수가 생성될 때까지 반복 수행한다. 먼저, 라인 4 와 5 에서 홀수에 해당하는 n 비트 길이의 난수 p 를 생성한다. 그 다음, 라인 6 에서 PRIME-TEST 라는 서브루틴을 이용하여 선택한 난수에 대한 소수성 검사를 수행한다. 여기에서 PRIME-TEST 를 통해 소수성 시험을 통과하면 라인 7 에서 생성된 소수 p 를 소수 목록 P 에 저장한다. 만일, 소수성 시험을 통과하지 못하면 다시 난수 p 를 생성하고 PRIME-TEST 수행을 실시한다. 이 과정을 k 개의 소수가 만들어질 때까지 반복한다.

여기에 기술한 RANDOM-PRIME 에서 한번 소수성 검사를 거쳐 라인 7 을 수행할 기대 횟수는 $O(\ln N)$이고, k 개의 소수를 생성하기 위한 RANDOM-PRIME 의 수행 시간은 $O(k\ln N)$이 된다. N 이하의 소수의 개수는 $N/\ln N$ 이기 때문에 소수성 시험을 위해 사용하는 알고리즘의 특성에 상관없이 난수 소수 생성 알고리즘 RANDOM-PRIME 의 라인 7 을 수행할 기대 횟수는 $O(\ln N)$로 크게 달라질 것이 없다. 하지만, 3, 5, 7 가운데 적어도 하나로 나눠지는 홀수는 전체 홀수의 54% 에 해당하고, 100 이하의 소수 가운데 적어도 하나로 나눠지는 홀수는 전체 홀수의 76%가 된다. 더 나아가 256 이하의 소수 가운데 적어도 하나로 나눠지는 홀수는 전체 홀수의 80%까지 이른다. 그러므로, 시험을 하기 전에 특정수의 배수는 제거하고 소수로 판정될 가능성이 높은 수들을 집중적으로 발생시켜 소수성 검사를 수행하면, 이 알고리즘 구현에 따른 수행 성능은 개선될 수 있다[2][5][6][7][8].

이러한 사실은 다음의 실험 결과에서도 확인할 수 있다. (그림 1), (그림 2)는 10,000 이하의 소수를 이용하여 10,000 개의 512 비트 난수에 대해 Small Divisor 시험을 실시하고 난 뒤의 결과를 나타내고 있다.

(그림 1)에서 한 점의 좌표 값 (i, j)는 $1\le i\le 10{,}000$, $1\le j\le 1{,}230$ 에 대하여 i 번째 발생된 난수의 최소 소인수가 10,000 이하의 소수 중에서 j 번째 소수임을 의미한다. 예를 들어, (그림 1)에서 좌표 값 (6536, 4)은 6,536 번째 발생된 난수가 2, 3, 5 의 배수가 아니며 7 의 배수임을 나타낸다. 또한, 좌표 값이 $(i, 1)$은 i 번째 난수가 짝수임을 뜻하며, 좌표 값 $(i, 1230)$은 i 번째 난수가 10,000 이하의 어떤 소수로도 나누어지지 않음을 나타낸다. (그림 1)에 나타난 바와 같이 난수를 발생하였을 때, 10,000 이하의 소수 중 200 번째 소수이하의 소수 배수가 많이 분포하고 있음을 확인할 수 있다.

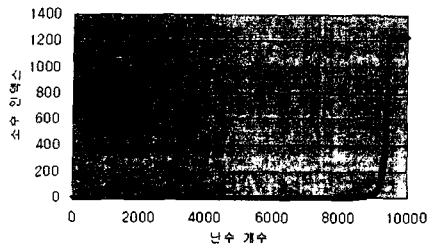


(그림 1) 난수 발생에 의한 최소 소인수 분포 상태

(그림 2)는 (그림 1)의 모든 데이터를 Y 축값 기준으로 오름차순 정렬하여 다시 그린 것이다. (그림 2)에서 X 축은 난수의 개수를 나타내고, Y 축은 (그림 1)과 마찬가지로 10,000 이하 소수 중의 최소 소인수 인덱스를 나타내고 있다. 예를 들어, (그림 2)에서 좌표값 (5079, 0)가 뜻하는 것은 전체의 50%가 짝수임을 의미하며, 좌표 값 (7811, 4)가 뜻하는 것은 2, 3, 5, 7 중 적어도 하나로 나누어지는 숫자는 경험적으로 전체의 78%임을 의미한다는 것으로 해석하면 된다. (그림 2)에 의하면, 200 번째 이하의 소수 배수가 되는 난수의 개수가 9,021 개이다.

그러므로, RANDOM-PRIME 알고리즘을 구현하고자 할 때에는 임의의 난수보다는 특정 소수의 배수가 되지 않는 난수 형태를 발생시켜 소수성 시험의 입력으로 하면 알고리즘 구현에 따른 수행 성능이 개선될

수 있다는 것을 짐작할 수 있다.



(그림 2) 특정 소수의 배수가 되는 난수 개수

나. 알고리즘 구현 결과 및 결과 분석

이 논문에서는 소수성 시험 방법으로 Small Divisor 시험 및 Rivest 의 Fermat 시험 단계만을 적용하여 512 비트의 난수 소수를 생성하도록 다음과 같은 입력 값에 대해 RANDOM-PRIME 알고리즘을 구현하였다.

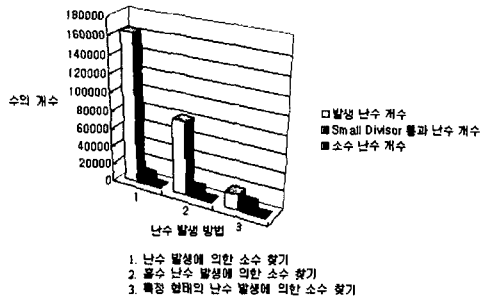
- 난수 발생에 의한 소수 찾기 : N 이하의 난수를 발생시켜서 소수성 시험
- 홀수 난수 발생에 의한 소수 찾기 : N 이하의 수 중에서 홀수 난수만을 택해서 소수성 시험
- 특정 형태의 난수 발생에 의한 소수 찾기 : K=2 x 3 x 5 x 7 x ... x 181 x 191 라 하면, $1 \le R \le (N-1)/K$ 인 난수 R 을 선택하여 $K \times R + 1$ 형태의 수만을 택해서 소수성 시험

특정 형태의 난수 발생 방법에 의한 소수 찾기의 경우에는 $1 \le R \le (N-1)/K$ (K = 2 x 3 x 5 x 7 x ... x 181 x 191)인 난수 R 을 선택하여 $K \times R + 1$ 형태의 수만을 택해서 소수성 시험을 실시하였다. 여기에서 사용한 K 의 값은 250 비트 크기이며, 난수 R 을 선택하여 $K \times R + 1$ 의 값이 512 비트가 되도록 하였다. 이 때 소수 곱 K 에 사용한 소수 목록은 다음과 같다.

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101,
103, 107, 109, 113, 127, 131, 137, 139, 149,
151, 157, 163, 167, 173, 179, 181, 191

위의 3 가지 서로 다른 방법에 따라 10,000 개의 Small Divisor 시험 통과 수를 구하고, 이 수들에 대하여 Rivest 가 제시한 Fermat 시험 단계를 실시한 결과간의 비교는 (그림 3)과 같다. 수행 결과 난수 발생 방법의 차이에 따라 10,000 개의 Small Divisor 시험 통과 수를 얻기 위해서 각각 163,012, 81,516, 17,329 개의 난수 발생을 필요로 했다. 이렇게 소수를 생성하기 위해 필요한 난수의 개수가 난수 발생 방법에 따라 급격한 차이를 보이고 있음을 확인했다. 그리고, 10,000 개의 Small Divisor 시험 통과 수 중에서 Rivest 의 Fermat 시험 단계를 통과한 수는 각각 487, 428, 475 개였다.

구현 결과에서 나타나듯이 Small Divisor 시험 단계를 통과한 수에 대해서 다시 Fermat 시험 단계를 실시하여 소수 난수를 구했을 때에 이를 만족시킨 비율은 3 가지 방법간에 큰 차이를 보이지 않고 있다. 왜냐하면, Small Divisor 시험을 통과한 수가 다시 Fermat 시험을 통과할 비율은 어떤 방법을 사용하더라도 동일하기 때문이다. 그러므로, 난수로부터 소수를 찾는 방법에서 수행 성능에 큰 영향을 미치는 부분이 입력에 사용하는 난수의 형태를 특정 소수의 배수가 되지 않도록 하는 부분임을 확인할 수 있다.



(그림 3) 소수 찾기 시험 결과간의 비교

4. 결론

난수 소수 생성을 위한 RANDOM-PRIME 알고리즘을 구현하고자 할 때에는 임의의 난수를 발생시켜서 소수성 시험을 실시하지 않는 것이 중요하다. 이 연구에서는 특정 소수의 배수가 되지 않는 난수 형태를 발생시켜 소수성 시험의 입력으로 사용하는 알고리즘을 구현하여 다음과 같이 알고리즘 수행 성능이 개선될 수 있다는 것을 확인하였다.

- 소수성 시험을 함에 있어 작은 소수의 곱이 되지 않는 홀수들을 사용하는 것이 좋다.
- 실제 구현에 있어서는 알고리즘의 Upper bound 수행 시간 $O(f(n))$뿐 만이 아니라 수행 시간 $O(c\,f(n))$에서의 상수 값 c 를 줄이는 것도 필요하다.

그리고, 특정 소수의 배수가 되지 않는 난수 발생 시에는 $(2 \times 3 \times 5 \times 7 \times ... \times 181 \times 191 \times R) + 1$ 에 사용된 1 뿐만이 아니라 2 x 3 x 5 x 7 x ... x 181 x 191 과 서로소가 되는 $0 < a < (2 \times 3 \times 5 \times 7 \times ... \times 181 \times 191)$인 임의의 a 를 사용할 수 있다. 이 연구의 향후 과제로는 Small Divisor 시험 단계의 수행 성능 향상 방안에 대한 연구가 될 것이다.

참고 문헌

[1] Peter D. Schumer, *Introduction to Number Theory*, PWS, 1996.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to Allgorithms*, MIT Press, 1996.

[3] "*Prime Number Generation and Primality Testing*", http://www.middlebury.edu/~pemerson/thesis/thesis.html

[4] Ronald L. Rivest, "*Finding Four Million Large Random Primes*", Advances in Cryptology - CRYPTO '90 Proceedings. Springer-Verlag. 1991, pp.625-6.

[5] "*Prime number generation and Primality testing*", http://www.middlebury.edu/~pemerson/thesis/thesis.html

[6] "*The Prime Pages*", http://www.utm.edu/research/primes

[7] Johan Hastad, *Advanced Algorithms Lecture Notes*, March 6, 1995

[8] Eric Bach, Jeffrey Shallit, *Algorithmic Number Theory VOL.1 Efficient Algorithms - Chapter.9 Prime Numbers: Basic Algorithms*, 1998