RSA암호 키생성에 관한 연구

*최용진, *김재문, °염흥열, *이만영
*한양대학교 전자통신공학과, °순천향대학교 전자공학과

A Study on the Key Generation of RSA Cryptosystem

*Choe Young Jin, *Kim Jae Moon, °Youm Heung Youl, *Rhee Man Young
*Dept. of Electronic Communication Eng. Hanyang Univ.
°Dept. of Electronics Eng. Sooncheonhyang Univ.

요 약

본 논문에서는 RSA 암호시스팀의 안전한 암호화, 복호화키의 생성에 관하여 연구하였다. 공개키 암호시스팀인 RSA 암호시스팀은 일반적으로 암호화, 복호화에 요구되는 계산시간과, 안전한 키 생성문제가 중요한 해결과제이다. 안전한 키는 소인수분해 공격에 강한 조건을 만족하는 것이다. 따라서 본 논문에서는 (P±1) 소인수분해 공격에 강한 조건을 만족하면서, 키 생성시 필요한 10^{75}정도의 소수를 예비 소수판정과 소수판정 알고리듬을 사용하여 소수가 아닐 확률이 10^{-8}이하가 되도록 하였다. 그리고 생성된 키가 정확함을 입증하였다.

1. 서 론

공개키 암호시스팀인 RSA 알고리듬은 간단한 암호화,복호화기법 때문에 널리 사용되고 있다.[1] 그러나 일반적으로 큰 수를 다뤄야 하므로 암호화,복호화에 많은 계산량이 요구된다. 또한 안전한 RSA 공개키 암호시스팀을 구성하기 위해서는, (P±1) 소인수분해 해독에 강한 합성수를 생성할 필요가 있다. 그 해독에 강한 소수를 생성하는 부분에 다량의 계산량이 요구되어 왔으나, J.Gordon은 8비트 이하의 알고있는 소수에 의존하는 예비판정과 고차의 멱(high-order power) 계산을 포함한 계산식을 이용하여, (P±1) 소인수분해 공격에 강한 RSA 암호키의 생성방법을 제안하였다.[2] 그러나, J.Gordon의 방식에서는 (1) 최적의 예비판정용 소수설정을 하지 않았고, (2)고차의 멱계산을 수행하는 데는 더욱 간단한 계산 방법이 있다는 점을 고려하지 않았다. 따라서 본 논문에서는 (1), (2)의 문제를 개선하기 위해 알고 있는 소수 설정범위의 최적치를 구하고, 정수론을 이용하여 고차의 멱계산을 간단히 할 수 있음을 보이고, 그 타당성을 입증한다.

2. RSA 암호 알고리듬과 키 생성조건[3]

2.1 RSA 암호 알고리듬의 암호화 및 복호화

RSA 암호방식을 개략적으로 설명하면 다음과 같다. 먼저 암호화키,복호화키는 다음의 세 정수 e,d,N 을 이용한다.

$$\begin{aligned} &\text{암호화키} \quad (e,N) \\ &\text{복호화키} \quad (d,N) \end{aligned} \qquad (2.1)$$

암호화는 보내고자 하는 메시지, 즉 평문(plaintext) M을 0~(N-1)사이의 정수로 바꾼 후에 암호문(ciphertext) C로 변환한다.

$$\begin{aligned} &\text{암호화} \quad C \equiv M^e \quad (\bmod N) \\ &\text{복호화} \quad M \equiv C^d \quad (\bmod N) \end{aligned} \qquad (2.2)$$

RSA 암호방식에서 암호화키와 복호화키는 일대일 대응하므로, (e,N)으로 암호화된 경우는 (d,N) 으로만 복호화 할 수 있다. 그러나, RSA에서는 공개된 암호화키(e,N)에서 복호화키(d)를 구하는 것이 N의 소인수를 알지 못하면 계산량이 많아서 힘들게 된다. 또한 N의 소인수 분해이외의 방법을 통한 해독방법도 N의 소인수 분해와 동등하거나 그 이상의 시간이 필요함이 알려져 있다. 하지만 N을 단순한 큰 소수의 곱으로 설정하는 조건만으로는 RSA암호가 간단히 해독되어 버리는 경우가 발생한다. 따라서 소인수 분해가 어려운 N의 생성조건을 찾아야 한다.

2.2 일반적인 키 생성방법

일반적인 암호화키,복호화키의 생성순서는 다음과 같다.

가) 서로 다른 두개의 큰 소수 P,Q를 임의로 선정한다. 여기서 P,Q는 비밀로 한다.

나) P,Q의 곱을 N이라 한다. ($N = P \cdot Q$)

다) 암호화키 e는 (P-1)(Q-1)과 서로소인 정수를 임의로 선택한다.

$$\gcd(e,(P-1)(Q-1)) = 1 \qquad (2.3)$$

라) (P-1)(Q-1)을 모듈라로 하여 e의 역수가 되는 복호화키 d를 계산한다.

$$e \cdot d \equiv 1 \quad (\bmod (P-1)(Q-1)) \qquad (2.4)$$

위 순서에서 다), 라)는 다음에 소개하는 정수론의 몇가지 정리를 이용하면 계산량을 줄일 수 있다.

【정리 1】 Euler의 정리[4]

$$\gcd(X,N)=1 \quad \Rightarrow \quad X^{ph(N)} \equiv 1 \quad (\bmod N)$$

여기서 ph(N)은 Euler's totient 함수이고, $N=p_1 \cdot p_2 \cdots p_n$ 으로 소인수분해될 때 $ph(N) = (p_1-1)(p_2-1)\cdots(p_n-1)$로 된다.

【정리 2】 Fermat의 정리

P가 소수이고 gcd(X,P)=1이면

$$X^{P-1} \equiv 1 \quad (\bmod P)$$

【정리 3】 중국인의 잉여정리

$N = p_1 \cdot p_2 \cdots p_n$으로 소인수분해될 때, 다음 합동식(congruence)들은 (0~N)안의 해를 갖는다.

$$X \equiv X_i \pmod{p_i} \quad i=1,\ldots,n$$

다),라)의 식은 다음 관계식에 의해서 유도되었다.[5] X를 보내고자 하는 메시지라고 하는 경우, 【정리1】에 의해 식(2-5)와 같이 된다.

$$X^{mph(N)} \equiv 1 \pmod{N} \qquad (2\text{-}5)$$

여기서 m은 정수이고, N=P·Q이다. 양변에 X를 곱하면

$$X^{mph(N)+1} \equiv X \pmod{N} \qquad (2\text{-}6)$$

따라서 X에 mph(N)+1승을 하면 원래의 메시지가 복원된다. 그러므로 암호화키와 복호화키는 다음 조건을 만족하면 된다.

$$e \cdot d = m \cdot ph(N) + 1 \Rightarrow e \cdot d \equiv 1 \pmod{ph(N)} \qquad (2\text{-}7)$$

앞에서 소개한 정리들을 이용하면 식(2-3), (2-4)는 다음처럼 바꿀 수 있다. 먼저 식(2-6)을 다음 식(2-8)로 놓아보자.[6]

$$X^{r(N)} \equiv 1 \pmod{N} \qquad (2\text{-}8)$$

【정리3】을 이용하면 식(2-8)은 다음과 같은 두 식으로 쓸 수 있다.

$$X^{r(N)} \equiv 1 \pmod{P} \qquad (2\text{-}9)$$

$$X^{r(N)} \equiv 1 \pmod{Q} \qquad (2\text{-}10)$$

【정리2】에 의해 식(2-9)를 만족하는 r(N)은 P-1의 배수이면 되고, 식(2-10)의 r(N)은 Q-1의 배수이면 되므로, 두 식을 모두 만족하는 r(N)은 P-1과 Q-1의 최소공배수이면 된다. 즉,

$$r(N) = lcm(P-1, Q-1) \qquad (2\text{-}11)$$

따라서 식(2-7)을 다음처럼 바꿔쓸 수 있다.

$$e \cdot d = m \cdot r(N) + 1 \Rightarrow e \cdot d \equiv 1 \pmod{r(N)} \qquad (2\text{-}12)$$

그러므로 다), 라)는 다음처럼 변형된다.

다) 암호화키 e는 r(N) = lcm(P-1,Q-1)과 서로소이고, r(N)보다 작은 정수를 임의로 선택한다.

$$\gcd(e, r(N)) = 1; \quad 1 < e < r(N)$$

라) r(N)을 모듈라로 하여 e의 역수가 되는 복호화키 d를 계산한다.

$$e \cdot d \equiv 1 \pmod{r(N)}$$

그러나 이와같은 조건만으로 생성된 키는 깨지기 쉽다. 다음 절에서는 일반적인 키 생성조건을 사용시 (P±1)법에 의해 N이 소인수분해되는 과정을 알아보겠다.

2.3 (P±1) 소인수분해 방법

N=P·Q로 소인수분해되고, 임의로 선택한 a가 gcd(a,N)=1을 만족하는 경우, gcd(a,P)=1이 성립하고 【정리2】에 의해 식(2.13)가 성립한다.

$$a^{P-1} \equiv 1 \pmod{P} \qquad (2.13)$$

P-1이 작은 소수만의 곱으로 표현되는 경우, 임의로 선택한 x값이 P-1의 배수가 되면, 즉 x=k·(P-1)이면 【정리2】를 이용하여 식 (2.14)처럼 쓸 수 있다.

$$a^x = a^{(P-1)^k} \equiv 1 \pmod{P} \qquad (2.14)$$

따라서 $a^x-1=m\cdot P$가 되므로 식(2.15)에 의해 N의 소인수 P를 구할 수 있다.

$$\gcd(a^x-1, N) = \gcd(m\cdot P, P\cdot Q) = P \qquad (2.15)$$

또한 (P+1)법도 같은 방법을 적용한다. 따라서 P-1, Q-1은 큰 소인수를 가져야 한다.

2.4 안전한 키 생성조건

앞에서 설명한 소인수 분해 해독에 강한 키를 생성하기 위해서는 다음 조건을 만족해야 한다.

(1) 소수 P,Q의 값을 크게 함

N은 두 소수의 곱이므로 $\sqrt{N}$ 보다 작은 소수가 존재한다. 따라서 간단한 소인수분해법을 사용해도 $\sqrt{N}$ 단계만에 구할 수 있다. 그러므로 P,Q의 차, 즉 |P-Q|가 되도록 작은 값을 선택한다. 또한 N이 커지면 소인수분해에 필요한 계산량이 비약적으로 증가한다. 일반적으로 N은 512비트(10^{154}) P, Q는 256비트(10^{77}) 정도가 타당하다고 고려되고 있다.

(2) 특정 소수 P,Q의 선택

(P±1)법의 소인수분해 공격방법을 피하기 위해 소수 P는 다음 조건을 만족해야 한다.

(a) P-1은 매우 큰 소수 R을 인수로 갖는다. 즉, P=2jR + 1
(b) P+1은 매우 큰 소수 S를 인수로 갖는다. 즉, P=2kS - 1
(c) R-1은 매우 큰 소수 T를 인수로 갖는다. 즉, R=2iT + 1

소수 Q에도 같은 조건이 요구된다. 3장에서는 이 조건들을 만족하는 효율적인 소수 생성방법에 대해 설명하겠다.

3. 효율적인 소수 생성방법

3.1 확률적인 소수 판정 알고리듬

소수 판정 알고리듬에는 확률적인 소수판정인 Solovay-Strassen(SS)판정방법[7]과 Knuth의 알고리듬 P가[8] 있다. Solovay-Strassen의 알고리듬은 다음과 같다.

【SS방법】[9] prime = true
for i=1 to k do

begin 1 < a < M 인 임의의 난수를 선택한다.
만일 gcd(a,M)≠1 또는 $a^{(m-1)/2} \not\equiv J(a,M) \bmod M$이면
prime = false
end.

여기서 J(a,M)은 Jacobi 기호이다.

이 알고리듬은 k회 시행시 2^{-k}의 확률로 소수가 아니라고 판정한다. 따라서 0.999999의 신뢰도로 소수를 찾기까지 SS방법은 적어도 20번을 시행해야 한다. 그러나, Knuth의 알고리듬 P는 k회 시행시 4^{-k}의 확률로 소수가 아님을 판정하므로 같은 신뢰도를 얻기 위해서는 10번 정도만 시행하면 된다. 또한 Knuth의 알고리듬 P를 만족하는 소수후보는 항상 SS판정을 통과하므로 더 우수한 것으로 증명되었다.[8] 따라서 본 논문에서는 Knuth의 알고리듬 P를 사용하였다. 알고리듬 P는 다음과 같다.

【알고리듬 P】

(1) 임의의 기수난수 $N = (1+2^k \cdot q)$을 생성하여 k값과 홀수 q값을 찾는다.

(2) 1부터 N 사이의 임의의 난수 X를 발생한다.

(3) 먼저 j = 0으로 설정하고, $(X^q \bmod N)$값을 Y에 넣는다.

$j = 0, \quad Y \leftarrow X^q \bmod N$

(4) $Y = X^{2^j q} \bmod N$

① 만일 j = 0, Y = 1 또는 Y = N - 1 이면
알고리듬을 종료하고 " N이 소수일 것이다 "라고 판정한다.

② 만일 j > 0, Y = 1 이면
(6)으로 간다.

(5) j를 1 증가시킨다.
만일 j < k 이면, $Y \leftarrow Y^2 \bmod N$ 하고 (4)로 간다.

(6) 알고리듬을 끝마치고 " N이 확실히 소수가 아니다. "라고 판정한다.

3.2 예비 판정용 소수의 설정

X이하의 모든 소수를 $\pi(X)$로 나타낼 때, 다음과 같은 정리가 알려져 있다.

【정리 4】 소수 갯수의 정리(Prime Number Theorem)

$$\lim_{x\to\infty} \frac{\pi(X)}{X/\ln(X)} = 1$$

【정리 4】에 의해 $\pi(X)$는 근사적으로 다음 수식으로 표현할 수 있다.

$$\pi(X) \fallingdotseq \frac{X}{\ln(X)}$$

따라서 2부터 X까지의 구간에서 임의로 선택한 값이 소수가 될 확률은 $\frac{\pi(X)}{(X-1)}$ 이 된다. 그러므로 평균적으로 소수를 찾기까지 $\frac{(X-1)}{\pi(X)} \fallingdotseq \ln(X)$ 개의 값을 검사해야

한다. 예를들어 P, Q가 2^{256} 정도이면 $\ln(2^{256})$=177개 (또는 홀수값만 조사하는 경우 89개)의 값을 검사해야 소수를 찾을 수 있다. 즉, 약 177개의 간격으로 소수가 나타날 것으로 기대된다. 따라서 2^{256}정도의 임의의 큰 수가 소수인지를 알기위해 직접 소수판정 알고리듬을 적용하면 상당히 많은 시간이 걸린다. 그러므로 에라토스테네스 체(Eratosthenes' sieve)의 원리를 이용하여 미리 정해놓은 소수집합의 배수가 되는 수를 제거함으로 소수판정 알고리듬이 판정할 대상을 감소시켜 놓는다. 예비판정을 포함한 전체적인 소수생성 순서는 다음과 같다.

【소수생성】

1) 소수집합 {3,5,7,11,...,31,...i(h)}을 설정함. 여기서 i(h)는 소수 3에서부터 h번째의 소수가 된다. 또한 홀수난수를 생성하므로 소수 2는 포함시키지 않는다.

2) 2^{256} 의 소수후보인 홀수난수 A를 한개 생성한다.

3) 난수 A에 대해 k=1에서 k=h까지 다음의 V(i(k))를 계산한다.

$$V(i(k)) \Leftarrow A \pmod{i(k)}$$

계산결과 { k=1에서 k=h까지 모두 V(i(k))≠0이면, 6)으로 가고
k를 증가시키면서 V(i(k))=0이 나오면, 4)로 간다.

4) $A \Leftarrow A + 2$

5) k=1부터 k=h까지 다음의 V(i(k))를 계산한다.

$$V(i(k)) \Leftarrow (V(i(k)) + 2) \pmod{i(k)}$$

계산결과 V(i(k))가 0이 되면, A를 합성수로 판정하고 4)로 간다.

6) A에 대해 소수판정 알고리듬 P를 적용하여, A가 합성수로 판정되면 4)로 돌아간다.

7) 소수 A를 출력하고 끝마침

이상의 순서로 끝마치면 소수판정 알고리듬에서 계산되는 multi-precision 연산횟수를 줄일 수 있다. 즉, 미리 배수 제거용 소수갯수 h를 적절히 정하면, 그러한 배수들은 소수판정 알고리듬을 통과하지 못하므로 소수를 찾을 때까지의 계산시간을 단축할 수 있다. 다음 절에서는 최적의 h값을 찾는 방법에 대해 알아 보겠다.

3.3 예비 판정용 소수 설정범위의 최적화

3부터 h개까지의 소수집합 {3,5,7,...,31,..i(h)}을 효율적으로 설정하기 위해서는 다음 순서의 계산을 수행해야 한다.

【최적치 h 설정】

1) 임의의 홀수가 에라토스테네스 체를 통과한 소수후보가 될 확률 Z(h)는 M=3·5 ···i(h)라 하면,

$$Z(h) = \frac{2}{3}\cdot\frac{4}{5}\cdots\frac{i(h)-1}{i(h)} = \frac{ph(M)}{M}$$

여기서 ph(M)= Euler's totient 함수이다.

2) 【소수생성】의 1)~3) 단계까지 걸리는 시간을 $t_0(h)$라 한다.

3) 【소수생성】의 4)~5)단계의 1회 처리시간을 $t_a'(h)$, 4)~5)단계가 종료하기까지의 시간, 즉 배수 제거용 소수 h개의 어느 것의 배수도 아닌 A를 얻을 때까지 걸리는 평균시간을 $t_a(h)$라 하면,

$$t_a(h) = \frac{t_a'(h)}{Z(h)}$$

4) 【소수생성】의 6)단계의 실행시간, 즉 소수판정 알고리듬 P의 소수판정에 필요한 처리시간을 t_b라 한다.

5) 【소수생성】의 5)단계가 종료하는 시점의 A가 소수가 될 확률 P(h)는, 2^{256} 정도의 수는 평균 약 177개의 정수간격으로 소수가 나타나므로, W=177이라 하면,

$$P(h) = \frac{1}{Z(h)\cdot\frac{1}{2}W}$$

6) 따라서 소수를 얻을 때까지의 기대시간 T(h)는,

$$\begin{aligned} T(h) = {} & t_0(h) + P(h)\cdot(t_a(h) + t_b) \\ & + (1-P(h))\cdot P(h)\cdot 2\cdot(t_a(h) + t_b) \\ & + (1-P(h))^2\cdot P(h)\cdot 3\cdot(t_a(h) + t_b) \\ & + \cdots \\ & + (1-P(h))^j\cdot P(h)\cdot(j+1)\cdot(t_a(h) + t_b) \end{aligned}$$

$$= t_0(h) + (t_a(h) + t_b)\cdot P(h)[1 + \{\sum_{j=0}^{\infty}(1-P(h))^j\cdot(j+1)\}]$$

$$= t_0(h) + (t_a(h) + t_b)\cdot P(h)\{1 + \frac{1-P(h)}{(P(h))^2}\}$$

$$= t_0(h) + (t_a(h) + t_b)\cdot\{P(h) - 1 + \frac{1}{P(h)}\}$$

7) 기대시간 T(h)식의 우변이 최소치가 되는 점을 h의 최적치로 한다. h값은 사용하는 컴퓨터의 기종과 프로그램 설계등에 의존하여 달라질 수 있다. h의 최적치는 5.2절에서 계산하였다.

4. 안전한 RSA키 생성

본 장에서는 3장에서 서술한 효율적인 소수생성 방법을 이용하여 안전한 RSA키 생성순서를 설명한다. 2.4절에서 설명한 조건을 만족하는 키 생성순서는 다음과 같다.

(1) P=2kS + 1, R=2iT + 1 식에 있는 소수 S,T를 생성한다.
(2) T를 이용하여 R=2iT + 1식을 만족하는 소수 R을 계산한다.
(3) R,S를 이용하여 P=2jR + 1, P=2kS - 1 두 식을 모두 만족하는 P를 계산한다.

4.1 소수 S,T,R의 생성

S와 T는 3.2절에서 설명한 【소수생성】 순서에 따라 구한다. 구한 T를 기초로 (R-1)이 큰 소인수 T를 가지도록 하는 방법은 다음과 같다. 생성된 소수 T를 사용하여 R=2iT + 1이 되는 R을 구하고, R의 소수판정을 수행한다. 단 판정순서는 【소수생성】 순서에서 기수난수 A를 소수후보 R로 치환하고, 2)~5)단계는 다음 것으로 바꾼다.

2) $R \Leftarrow 2iT + 1$ 이 되는 R을 설정한다.

3) 소수후보 R에 대하여 k=1부터 k=h까지, 다음의 V(i(k))및 V'(i(k))를 계산한다.

$$V(i(k)) \Leftarrow R \pmod{i(k)}$$
$$V'(i(k)) \Leftarrow 2 \cdot T \pmod{i(k)}$$

계산결과 { k=1에서 k=h까지 모두 V(i(k))≠0이면, 6)으로 가고
k를 증가시키면서 V(i(k))=0이 나오면, 4)로 간다.

4) $R \Leftarrow R + 2T$

5) k=1부터 k=h까지 다음의 V(i(k))를 계산한다.

$$V(i(k)) \Leftarrow (V(i(k)) + V'(i(k))) \pmod{i(k)}$$

계산결과 V(i(k))가 0이 되는 경우가 있으면 R을 합성수로 판정하고 4)로 돌아간다.

4.2 소수 P의 계산

그러나 P-1이 큰 소인수 R을 갖고 동시에 P+1이 큰 소인수 S를 갖는 조건을 만족하는 소수 P를 찾는 것은 어렵다. 즉 다음의 두 식을 동시에 만족하는 P를 찾아야 한다.

$$P \equiv 1 \pmod{R} \qquad (4\text{-}1)$$
$$P \equiv -1 \pmod{S} \qquad (4\text{-}2)$$

식(4-1),(4-2)를 동시에 만족하는 P를 구하기 위해 【정리3】의 보조정리를 이용한다.

【보조정리】 r,s를 서로소인 정수라 하고, g를 모듈라 r의 역원이라 하면, 다음 식처럼 쓸 수 있다.

$$s \cdot g \equiv 1 \pmod{r} \qquad (4\text{-}3)$$

이 경우 임의의 정수 x,y에 대해 【정리3】을 이용하여 다음 두 식을 만족하는 P를 구해보면,

$$P \equiv y \pmod{r}$$
$$P \equiv x \pmod{s}$$

$$P \equiv \frac{r \cdot s}{r} \cdot a_1 \cdot y + \frac{r \cdot s}{s} \cdot a_2 \cdot x \pmod{rs}$$

여기서 { a_1 : $s \cdot a_1 \equiv 1 \pmod{r}$ 즉, $a_1 = g$
a_2 : $r \cdot a_2 \equiv 1 \pmod{s}$ 즉, $r \cdot a_2 = 1 + k \cdot s$

따라서 $r \cdot a_2 = 1 + k \cdot s$에서 k는 임의의 정수이므로 k=-g로 놓을 수 있다. 그러므로 이 조건들을 포함하여 정리하면 식(4-4)와 같이 된다.

$$P \equiv s \cdot g \cdot y + r \cdot a_2 \cdot x$$
$$\equiv sg \cdot y + (1 - sg) \cdot x$$
$$\equiv x + (y - x) \cdot sg \pmod{rs} \qquad (4\text{-}4)$$

그러므로 식(4-1),(4-2)를 만족하는 P는 식(4-4)에 x=-1,y=1을 대입하면 된다.

$$P \equiv -1 + 2s \cdot g \pmod{rs} \qquad (4\text{-}5)$$

따라서 이 【보조정리】를 이용하면 식(4-1), (4-2)를 동시에 만족하는 P는 식(4-3)의 g를 먼저 계산한 후 식(4-5)를 이용하여 구한다. P를 구하는 식(4-5)는 J.Gordon이 제안한 식 $P \equiv s^{r-1} - r^{s-1} \pmod{rs}$에 비해 계산량이 적으므로, RSA 키생성에 소요되는 시간을 줄일 수 있다. 식(4-5)에서 구한 P를 P_0로 놓고, P_0가 짝수일 경우에는 $P_0 = P_0 + R \cdot S$로 놓는다. 이렇게 얻어진 소수후보 P_0를, $P \Leftarrow P_0 + 2jR \cdot S$로 설정하고 3.2절의 【소수생성】순서에 따라 소수 판정을 행한다. 단 기수난수 A는 소수후보 P_0로 치환하고, 2)~5)단계는 다음 것으로 바꾼다.

2) 식(4-3)과 (4-5)를 이용하여 소수후보 P_0를 계산한다.

3) 소수후보 P_0에 대해 k=1부터 k=h까지, 다음의 V(i(k))를 계산한다.

$$V(i(k)) \Leftarrow P_0 \pmod{i(k)}$$
$$V'(i(k)) \Leftarrow 2 \cdot R \cdot S \pmod{i(k)}$$

계산결과 { k=1에서 k=h까지 모두 $V(i(k)) \neq 0$이면, 6)으로 가고
k를 증가시키면서 $V(i(k))=0$이 나오면, 4)로 간다.

4) $P \Leftarrow P_0 + 2j \cdot R \cdot S$

5) k=1부터 k=h까지 다음의 V(i(k))를 계산한다.

$$V(i(k)) \Leftarrow (V(i(k)) + V'(i(k))) \pmod{i(k)}$$

계산결과 V(i(k))가 0이 되는 경우가 있으면 P를 합성수로 판정하고 4)로 돌아간다.

다음장에서는 이와같은 방법에 의해 RSA키를 프로그램에 의해 계산하고 그 결과를 검토한다.

5. 프로그램 실행결과 및 검토

본 장에서 수행한 결과는 Intel 80386프로세서 25Mhz에서, C언어와 어셈블러를 이용하여 실행하였다.[10]

5.1 RSA 키생성 순서도

RSA 암호화, 복호화키 생성의 전체 순서도는 그림 5.1과 같다.

5.2 프로그램 실행 결과

(1) 예비판정 소수 후보의 최적치 h값 결정

최적의 h값을 결정하기 위해 h값을 100에서부터 증가시키면서 10^{75}정도의 기

수난수가 소수가 아닐 확률이 10^{-8} 이하가 될 때까지의 시간을 측정하였다. 여기서 각 h값에 대해 10^{75}의 기수난수를 100개씩 생성하여 평균시간을 취하였다. 그 결과 그림5.2과 같이 h값은 600정도에서 포화상태에 도달한다. 즉, 소수 3부터 소수 4583 사이의 소수를 이용하여 예비판정을 하면 가장 적절함을 알게 되었다.

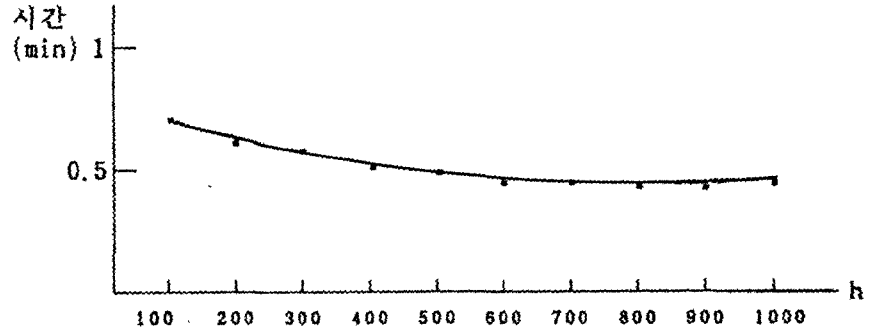

그림 5.2 h값에 따른 10^{-8}이하의 확률을 가진 소수 생성도

(2) RSA 키 생성시간

N=10^{150}, P=Q=10^{75} 정도의 수를 최적치 h=600개의 예비 소수판정 방법을 이용하고 키를 생성하기까지의 총 시간을 측정하여 그림5.3에 나타내었다. 여기서 100개의 키 생성시간을 각각 측정하였다. 그 결과 41개의 키가 150초~ 200초사이에서 생성되었다.

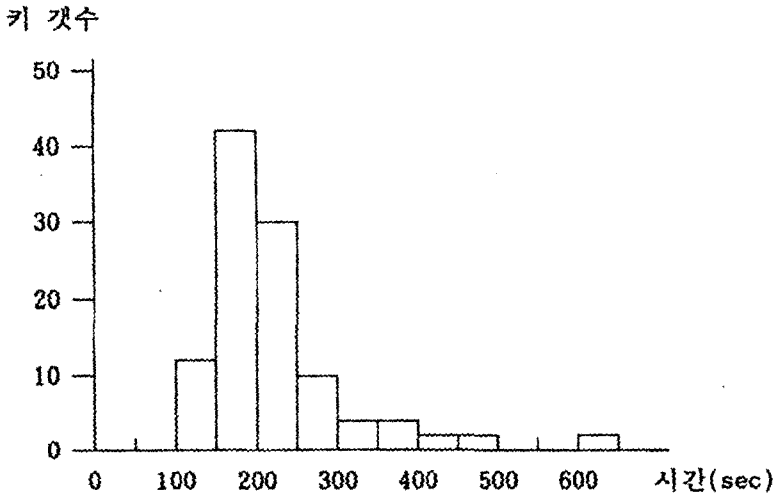

그림 5.3 RSA 키 생성 계산시간

(3) 암호화,복호화키 생성 예와 검토

(가) 소수 P생성

S = 32074064573258056200711037105464143

T = 86921264703429351279010620540936667

R = 60844885292400545895307434378655666691

g = 24308150347238805335623820121776016O7

R*S = 19515427798209344951577964928202369179993143197403228197063430727566O813

P_0 = 15593223677876053670221929345646387064761874386429678933352555255

55355601

P = 157450280290346080846420616614767559461392352914934316057664777216737970879

(나) 소수 Q 생성

S = 661829903648817071818933790894361 43

T = 475206152450630656055821250688500 87

R = 807850459166072115294896126170451479 1

g = 456304554565344414436797138093650557 0

R*S = 534659591552534138340337970999496438387809419496633902993734197593491113

Q0 = 603991998764996571838053818577703369441884607577303411249931632157633019

Q = 244408765746720563655032168594348079274282979898042363176392725734789580547

(다) 암호화키(e), 복호화키(d) 생성

N=P*Q = 384822286722386890883138450762098738223082152726650533676141806553745374525988219805290058471737697480898250256994076025118848501686740551004788310890813

ph=(P-1)·(Q-1) = 384822286722386890883138450762098738223082152726650533676141806553745259878179462440214050931960281130411413553403497860355250075064935018367833393 88

gcd(P-1,Q-1) = 2

lcm(P-1,Q-1) = 192411143361193445441569225381049369111541076363325266838070903276872629939089731220107025465980140565205706776701748930177625037532467509183916696 94

암호화키 e = 887

복호화키 d = 396970002650489295555886902420879758144442130490287754581363870345746237642090426305294088616396456859443566405145660137796002050377018649161857446 9

(라) 암호화,복호화

앞에서 생성된 암호화키, 복호화키가 맞는지를 알아보기 위하여 다음과 같은 평문을 암호화 하여 보았다. 여기서 메시지 M은 0≤M≤N 조건을 만족해야 한다. 앞에서 구한 N은 10^{149}, 약 495비트이고 각 문자는 8비트로 표현되므로 1블럭당 최대 60문자(≅495÷8)를 취할 수 있다. 따라서 다음 평문은 최소 3블럭으로 나눌 수 있다. 다음처럼 암호화,복호화를 수행하면 보낸 평문이 정확히 복원되므로 생성된 키의 정확성을 입증하였다.

① 암호화 할 평문

The RSA system was invented by Rivest, Shamir, and Adleman. It is defined applying two numerical problems, namely the discrete logarithm problem and the factorization problem.

② 제 1 블럭의 암호화,복호화

평문
The RSA system was invented by Rivest, Shamir, and Adleman. I

평문의 16진수 표현
6854206553522041797374736D6577207361692076 6E6E6565742064796252
20766973652C7453206168696D2C726120646E41206C646D656E61492E0000

암호문의 16진수 표현
05F320D13DDDA4E278125191DE8DDC13BF1AE0AFDABBBA1038E2ED419D3DCE
F5EA398135B72C9F9192343A5E309F561BA31D19441CBAC183D6336C4412EA

복호문의 16진수 표현
6854206553522041797374736D6577207361692076 6E6E6565742064796252
20766973652C7453206168696D2C726120646E41206C646D656E61492E0000

복호된 평문
The RSA system was invented by Rivest, Shamir, and Adleman. I

③ 제 2블럭의 암호화,복호화

평문
t is defined applying two numerical problems, namely the dis

평문의 16진수 표현
2074736964206665 6E6964656120707079 6C6E6920677774206F 756E656D69
726163206C7270626F 656C736D202C616E 656D796C7420656864 2073690000

암호문의 16진수 표현
A068EC4FA322E7107FEB549BD5D5913D0D51E156A5C62C6B0B818307046556
1E085482B21693BAC07B018C6E2199909018754 0E7AC4937AFD9C9B3030E12

복호문의 16진수 표현
2074736964206665 6E6964656120707079 6C6E6920677774206F 756E656D69
726163206C7270626F 656C736D202C616E 656D796C7420656864 2073690000

복호된 평문
t is defined applying two numerical problems, namely the dis

④ 제 3블럭의 암호화,복호화

평문
crete logarithm problem and the factorization problem.

평문의 16진수 표현

```
  7263746520656F6C616769726874206D7270626F656C206D6E612064687420
  6561667463726F7A6974616F69206E7270626F656C2E6D0000000000000000

암호문의 16진수 표현
  1EFB0DA30DEF3B0C934B076FA2711F225F5E3F3BE9B97E4BBF6903126AC875
  53C8BB1D03DE938CE9D31BF110B0DC25E52298543EDFEFB300B638E16A2CCC

복호문의 16진수 표현
  7263746520656F6C616769726874206D7270626F656C206D6E612064687420
  6561667463726F7A6974616F69206E7270626F656C2E6D0000000000000000

복호된 평문
  crete logarithm problem and the factorization problem.
```

6. 결 론

본 논문에서는 (P±1)소인수분해에 강하고, RSA 암호키의 조건을 만족하는 키를 프로그램을 통하여 생성하여 보았다. 즉 키를, 최적의 소수판정 후보설정과 고차의 멱계산을 정수론의 정리를 이용하여 좀더 간단하고 빠르게 생성하였다. 그러나, 본 프로그램에서는 기본적인 연산 알고리듬을 적용하였기 때문에 소프트웨어만으로는 만족할만한 속도로 키를 생성할 수 없었다. 즉, 실용화할 수 있는 키의 크기인 2^{256} 정도의 큰 수의 연산을 위해서는 보다 효율적인 알고리듬과 하드웨어의 결합이 요구된다.

참고 문헌

[1] 이 만영, "공개키 암호시스팀에 관한 연구(1)," 통신정보보호학회지, 제1권, 제1호, pp.94-99, 1991. 4

[2] J.Gordon, "Strong Primes are Easy to Find," Lecture Note in Computer Science, 209, pp.216-223, 1984

[3] R.L.Rivest, A.Shamir and L.Adleman, "A Method of Obtaining Digital Signatures and Public-key Cryptosystems," Communications of the ACM 21 (2). pp.120-126, 1978

[4] U.Dudley, *Elementary Number Theory*, W.H.Freemanand Company, 1969

[5] Meyer, C.H. and S.M.Matyas, *Cryptography: A New Dimension in Computer Data Secutity*, John Wiley, 1982

[6] J. Seberry and J. Pieprzyk, *Cryptography : An Introduction to Computer Security*, Prentice-Hall, 1989

[7] R.Solovay and V.Strassen,"A Fast Monte-Carlo Test for Primality," SIAM Journal of Computing 6(1), 84-85(1977): 7,118(1978)

[8] Knuth, *The Art of Computer Programming, Seminumerical Algorithms*, Addison Wesley, Reading, Mass., 2nd, 1981

[9] Van Tilborg, *An Introduction to Cryptography*, Kluwer Academic Pub., 1988

[10] D.G.N.Hunter, "RSA Key Calculation in ADA," The Computer Journal, Vol.28, No.3, pp343-348, 1985

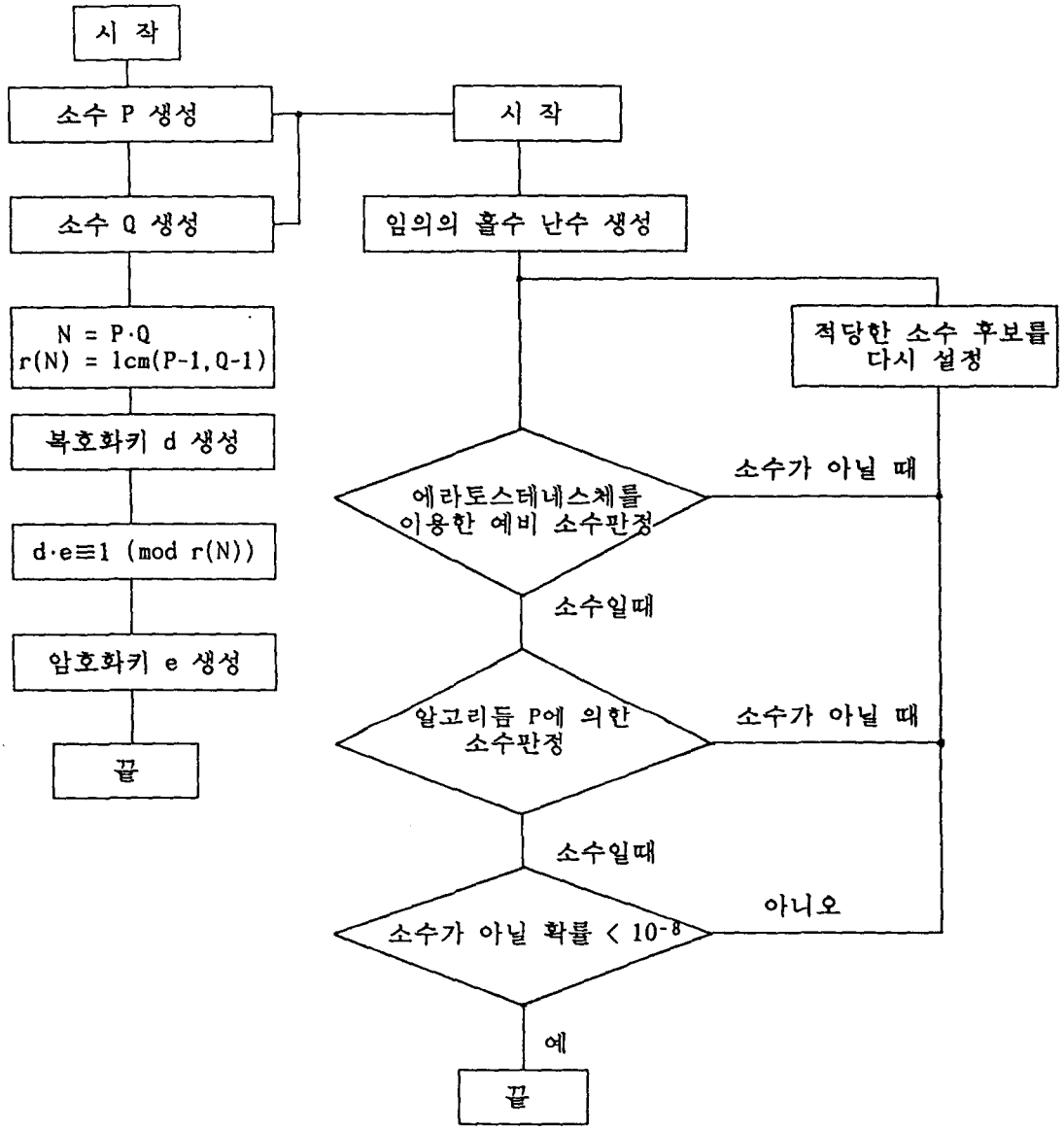

그림 5.1 RSA 키생성 순서도