DOI: 10.3745/KIPSTC.2008.15-C.6.507

RSA 알고리즘 부하 경감을 위한 고속 모듈러 멱승 연산 알고리즘 설계

김 갑 열[†] · 이 철 수[††] · 박 석 천[†††]

요 약

최근 정보통신의 급속한 발달로 온라인 서비스에 기반을 둔 기업들이 많이 구축되고 있다. 이들 기업들은 개인정보를 수집하여 고객관리를 하며 유료 서비스의 경우 결제정보를 요청하여 대금을 지불하는 방식을 취한다. 이와 같은 정보의 유통과 관리는 최근 큰 이슈가 되고 있으나 대부분의 기업들이 정보보호에 대한 인식이 부족한 현실이다. 실제로 국내 최대 오픈마켓에서 해킹으로 대량의 고객 개인정보가 노출되기도 했다. 따라서 본 논문에서는 고객관리를 위한 개인정보의 불법공격이나 해킹에 대비하기 위해 가장 보편화된 RSA 암호 알고리즘 부하를 줄이는 방법을 제안한다. 이때 부하를 줄이는 방법은 Binary NAF Method를 이용하여 RSA 핵심 연산인 모듈러 멱승 연산을 고속으로 처리할 수 있도록 설계하였고 기존 Binary Method와 Windows Method를 이용한 모듈러 멱승 알고리즘을 구현하여 비교 평가 하였다.

키워드 : 모듈러, 멱승, 유클리드 호제법, 암호화

Design of High Speed Modular Exponentiation Operation Method for RSA Algorithm

KapYol Kim[†] · ChulSoo Lee[††] · SeokCheon Park[†††]

ABSTRACT

At a recent, enterprises based on online-service are established because of rapid growth of information network. These enterprises collect personal information and do customer management. If customers use a paid service, company send billing information to customer and customer pay it. Such circulation and management of information is big issue but most companies don't care of information security. Actually, personal information that was managed by largest internal open-market was exposed. For safe customer information management, this paper proposes the method that decrease load of RSA cryptography algorithm that is commonly used for preventing from illegal attack or hacking. The method for decreasing load was designed by Binary NAF Method and it can operates modular Exponentiation rapidly. We implemented modular Exponentiation algorithm using existing Binary Method and Windows Method and compared and evaluated it.

Keywords : RSA, ECC, Montgomery, Binary NAF

1. 서 론

암호화란 공격자가 데이터를 도청하더라도 그 내용을 알 수 없도록 특정한 처리를 하는 것으로 수 천년 전부터 인류는 암호 기술을 사용해 왔다[1]. 암호 알고리즘은 원본데이터를 암호화 시키는 키의 종류에 크게 비밀키 암호 알고리즘과 공개키 암호 알고리즘으로 나누어지며 각 암호 알고리즘은 다양한 수학이론들이 사용되고 있다[2].

비밀키 암호 알고리즘은 암호키와 복호키가 서로 같은 형태로 암·복호 속도는 공개키 암호 알고리즘에 비해 빠르나 키 분배에 대한 문제가 생기며 키가 노출되게 되면 공격자가 암호문 해독이 용이해져 데이터 보호에 치명적일 수 있다. 공개키 암호 알고리즘은 암호키와 복호키가 서로 다른 형태로 키 관리에 대한 안전성을 보장할 수 있지만 구현의 복잡함과 연산 부하가 커서 비교적 고사양의 시스템에서만 사용 한다[3].

본 논문에서는 공개키 암호 알고리즘 중 가장 강력한 안정성을 확보하고 있는 RSA(Rivest Shamir Adleman) 알고리즘을 효율적으로 구현하기 위해 고속 모듈러 멱승 알고리즘을

† 준 회 원 : 경원대학교 전자계산학과 석사과정
†† 정 회 원 : 경원대학교 컴퓨터 소프트웨어학과 교수
††† 종신회원 : 경원대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2008년 11월 3일
수 정 일 : 1차 2008년 11월 24일
심사완료 : 2008년 11월 30일

제안한다. 제안 알고리즘은 ECC(Elliptic Curve Cryptosystem) 알고리즘에서 고속 스칼라곱 연산을 위해 사용되는 Binary NAF Method를 적용하여 고속 모듈러 멱승을 할 수 있도록 설계 하였다. 제안 알고리즘의 평가는 제안 알고리즘의 복호화 연산 기능 효율 측정과 기존 RSA 알고리즘에서 고속 모듈러 멱승 연산을 위해 사용되는 Binary Method와 Window Method를 이용한 모듈러 멱승 알고리즘을 구현하여 연산 시간을 비교하였다.

2. 관련연구

2.1 RSA 알고리즘

RSA 알고리즘은 1977년에 개발된 공개키 암호 알고리즘이다. 알고리즘의 보안성은 큰 수의 인수분해가 어렵다는 데에 기인하며 공개키와 비밀키는 큰 소수의 짝으로 이루어져 암호화된 메시지와 공개키만으로는 원래의 메시지를 찾기 어렵다[4]. 인수분해 문제와 관련된 연구에 의하면 RSA 암호 알고리즘의 키 길이는 적어도 1,024bit가 되어야 최소 몇 년 동안 적절한 안전성을 보장할 수 있다[5].

RSA 알고리즘에서 키 생성 과정은 먼저 서로 다른 임의의 두 개의 큰 소수 p, q를 선택하고 그 곱 n(p·q)를 구한다. 사용자는 Φ=(p-1)(q-1)에 관해 서로소인 e를 개인 공개키로 선택하고 e와 n을 공개한다. 사용자가 p, q를 알고 있다면 유클리드 알고리즘을 이용하여 e*d ≡1 (mod Φ)인 정수d를 개인 비밀키로 택할 수 있고 이들을 이용하여 데이터를 암호화 또는 복호화를 한다.

암호화 방법은 적당한 데이터를 블럭으로 나누어서 각 블럭을 mod n에 공개키(e)로 모듈러 멱승 연산을 하여 암호화한다. 복호화는 개인만이 알고 있는 비밀키(d)를 사용하여 mod n에 모듈러 멱승 연산을 하면 데이터가 복호된다[6].

2.2 모듈러 멱승 알고리즘

모듈러 멱승 알고리즘은 기본 연산인 모듈러 곱셈을 고속으로 처리하는 알고리즘과 모듈러 곱셈 수를 줄이는 기법 등에 의해서 고속으로 연산할 수 있다. 이때 모듈러 곱셈을 고속으로 처리하 알고리즘으로는 Yang 알고리즘, Barret 알고리즘, Montgomery 알고리즘 등이 있으며 현재까지는 Montgomery 알고리즘이 가장 효율적으로 알려져 있다[7]. 모듈러 곱셈 수를 줄이는 기법으로는 Binary Method와 Window Method가 있으며 구현의 용이성으로 Binary Method가 연산 속도가 떨어짐에도 불구하고 가장 많이 알려져 있다. 이 이외에도 모듈러 곱셈 수를 줄이는 기법으로 확장 이진 방식, 누승 테이블 방식이 있으나 미리 일정 값들을 계산하여야 하므로 매번 밑수가 바뀌는 RSA 알고리즘에서 적용하기에는 비효율적이다[8].

2.3 Binary NAF Method

Binary NAF(Non Adjacent Form) Method는 ECC 알고리즘에서 타원곡선상의 한점 P를 임의의 난수 k만큼 스칼라곱 할 때 고속으로 연산하기 위한 기법으로 사용되어 지고 있다. 정수 k를 NAF 함수로 분해하여 나오는 경우의 수에 따라 연산을 달리하여 kP에서 연산의 수를 줄일 수 있다. NAF의 기본 원리는 정수의 이진 값에서 1을 인접하지 않게 배치하여 + 및 - 연산을 통해 연산 횟수를 줄이고 정확한 수로 다시 조합하는데 있다[9]. <표 1>은 NAF를 이용한 정수 연산 예제를 비교한 것이다.

〈표 1〉 NAF를 이용한 정수 연산

이진 값	연산 결과
$(0\ 1\ 1\ 1)_2$	4 + 2 + 1 = 7
$(1\ 0\ -1\ 1)_2$	8 − 2 + 1 = 7
$(1\ -1\ 1\ 1)_2$	8 − 4 + 2 + 1 = 7
$(1\ 0\ 0\ -1)_2$	8 − 1 = 7

<표 1>과 같이 정수 7을 이진 값으로 나타내어 연산할 수 있는 경우의 수는 4가지가 나오지만 연산 횟수가 가장 적은 것은 $(100\text{-}1)_2$로 1의 값이 가장 멀리 떨어진 형태를 가진다. 이와 같은 원리는 모든 정수에 적용되며 적절하게 분해하여 연산하게 될 경우 획기적으로 연산 횟수를 줄일 수 있다. <표 1>의 $(100\text{-}1)_2$를 다시 조합하는 원리는 연산 결과에서 나오는 것과 같으나 각 자리수 마다 가중치를 두게 되면 그 가중치를 미리 연산하여야 하므로 전체 연산의 부하를 주게 된다. 따라서 다음과 같은 방법을 사용한다.

- 분해된 정수의 모든 자리수의 가중치는 곱하고자 하는 값을 가짐
- 7X에서 X가 2라면 7의 분해값 $(100\text{-}1)_2$의 각 자리의 가중치는 2를 가짐
- 분해값이 1이 나오면 2X+X를 연산함 단, 최초 분해값 1은 연산하지 않음
- 분해값이 -1이 나오면 2X-X를 연산함
- 분해값이 0이 나오면 2X만 연산함
- $(100\text{-}1)_2 \cdot 2 = 2*2 \rightarrow 2*4 \rightarrow 2*8-2 = 14$

위와 같은 원리로 7X 계산하게 되면 6번 더하는 연산 횟수를 4번의 연산으로 줄일 수 있다. 이때 곱 연산은 단순히 변수의 값을 복사하여 다시 더하는 작업이므로 실제로는 곱하기 연산을 하지 않고 더하기 연산을 하게 된다. 정수를 분해하는 NAF 함수는 다음 (그림 1)과 같다[10].

```
Input: k = (e_{m-1} e_{m-2} ··· e_1 e_0)_10
Output: k = (z_m z_{m-1} ··· z_1 z_0)_NAF
i ← 0
while k > 0 do
    if k is odd then
        z_i ← 2 − (k mod 4)
    else
        z_i ← 0
    k ← (k − z_i)/2
    i ← i + 1
return z
```

(그림 1) NAF 동작

(그림 1)에서 입력 값 k가 7이라고 하면 7은 홀수 이므로 조건에 의해 2-(7 mod 4) = -1의 값을 z_0에 저장한다. 이후 k는 ((7-(-1))/2) = 4의 값을 가지며 4는 짝수이므로 z_1에는 0이 저장된다. k는 (4-0)/2 = 2의 값을 가지게 되며 역시 k가 짝수 이므로 z_2에는 0이 저장되고 k는 (2-0)/2 = 1의 값으로 변경된다. 다음 k는 홀수인 1의 값을 가지므로 조건에 의해 2-(1 mod 4) = 1을 연산하여 z_3에 저장된다. 이때 k = 1은 0보다 큰 제일 작은 수이므로 저장된 z(z_3 z_2 z_1 z_0) = (100-1) 값을 리턴 한다.

2.4 유클리드 호제법

일반적으로 암호 알고리즘에서 나누기 연산은 시스템 전체의 부하를 주기 때문에 꼭 필요한 경우를 빼고 사용되지 않는다. 하지만 Binary NAF Method를 이용한 모듈러 멱승 알고리즘에서는 나누기 연산의 비중이 크게 차지하게 되는데 이때 확장 유클리드 호제법(Euclidean algorithm)을 이용하여 모듈러 n에 대해 a의 역원을 구하면 곱하기 연산으로 바꿀 수 있다.

확장 유클리드 호제법은 유클리드 호제법과 일차결합(Linear Combination)을 이용한 합동식 ax ≡ 1 mod n을 해결하는 문제로 유클리드 호제법을 이용하여 a와 n의 최대공약수를 구하고 일차결합을 이용하여 역으로 정리하면 a mod n에서 a의 역원 x를 찾을 수 있다[11][12]. 즉 a로 나누기 연산을 하여야 할 때 x를 곱하여도 같은 결과를 얻을 수 있게 된다.

3. 고속 모듈러 멱승 연산 알고리즘 설계 및 평가

본 논문에서 제안하는 고속 모듈러 멱승 연산 알고리즘은 X^k mod n에서 지수 값 k를 분해하고 일정한 원리를 통해 멱승 연산을 하는 방법이다. 제안 알고리즘에서 지수 값 k를 분해하는 기법은 ECC 알고리즘에서 고속 스칼라곱을 위해 사용되는 Binary NAF Method를 이용하나 Binary NAF Method의 원래 목적인 스칼라곱을 위해 사용하지 않으므로 Method명을 RENAF(RSA Exponentiation Non Adjacent Form) Method라 새롭게 정의 하였다.

3.1 제안 기법의 적용

기존 Binary NAF Method는 ECC 알고리즘에서 타원곡선상의 한 점의 스칼라곱을 빠르게 하기 위해 더하기 연산을 줄이는 방법으로 사용되며 모듈러 멱승 연산에서도 곱의 횟수를 줄이는 방법으로 적용할 수 있다. <표 2>는 Binary NAF Method와 RENAF Method를 이용한 스칼라곱 연산과 멱승 연산의 예를 나타낸다.

RENAF Method에서 멱승 연산은 <표 2>와 같이 Binary NAF Method의 2X(2*2)연산을 X^2(2*2)연산으로 2X-X(2*8-2)연산을 X^2/X(16*16/2)연산으로 대체하면 된다. 일반적으로 나누기 연산은 곱 연산에 비해 시스템 부하와 연산 시간 저하에 원인이 된다. 따라서 본 논문에서 제안하는 알고리즘에서는 모듈러 n에 대한 /X의 역원을 구해 곱해준다. 이때 /X의 역원은 최초 연산시 한번만 구하게 되므로 전체 모듈러 멱승 연산에 주는 부하는 아주 미비하다.

〈표 2〉 Binary NAF Method 스칼라곱 연산과 RENAF Method 멱승 연산

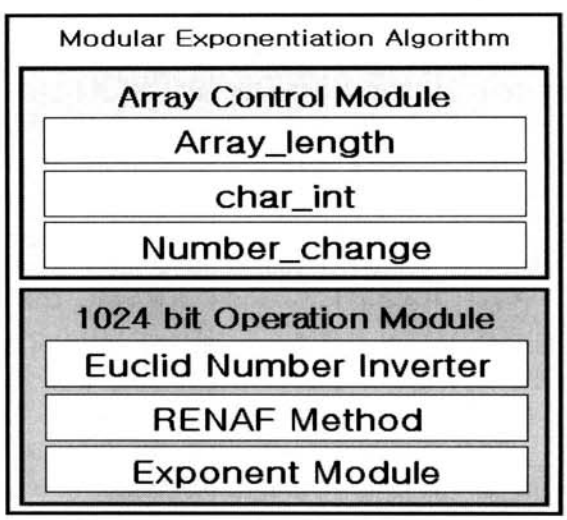
종류	연산 결과(X=2)
Binary NAF(7X)	$(100-1)_{NAF}$·2 = 2*2 → 2*4 → 2*8-2 = 14
RENAF(X^7)	$2^{(100-1)_{RENAF}}$ = 2*2 → 4*4 → 16*16/2 = 128

3.2 제안 알고리즘의 설계

(그림 2)는 본 논문에서 제안하는 RENAF Method를 이용한 모듈러 멱승 알고리즘의 구조이다. RENAF Method를 이용한 모듈러 멱승 알고리즘을 구현하기 위해서는 기본적으로 1,024비트 이상의 사칙연산과 mod 연산을 할 수 있는 모듈, 정수의 역원을 구할 수 있는 유클리드 알고리즘이 설계되어야 한다. (그림 2)에 나타난 각 모듈의 기능은 다음과 같다.

- RENAF Method : 모듈러 멱승 연산을 줄이기 위해 지수 값을 분해하는 모듈
- Array-length : 배열값의 길이를 리턴하여 효율적인 배열 컨트롤을 지원하는 모듈
- char-int : char형의 배열 값을 int형의 배열로 변경하는 모듈
- Euclide Number Inverter : mod 연산에서 나눗셈 연산을 곱셈으로 대처할 수 있도록 정수의 역원을 구하는 모듈
- Number Change : 값이 존재하는 배열을 빈 배열에 복사하는 모듈
- Exponent Module : 모듈러 멱승 연산을 하는 핵심 모듈

제안하는 알고리즘에서 핵심 연산을 하는 Exponent Module의 동작은 (그림 3)과 같이 설계하였다.



(그림 2) RENAF Method를 이용한 제안 알고리즘 구조

```
Input: X, k, n
Output: (X^k mod n) = Q
Q ← X, X'← Inv(X)
R[] ← RENAF(k), i ← strlen(R)-2
while i >= 0 do
    Q ← Q*Q
    Q ← Q mod n
    if R[i] == 1 then
        Q ← Q*X
        Q ← Q mod n
    else if R[i] == -1 then
        Q ← Q*X'
        Q ← Q mod n
    i ← i - 1
return Q
```

(그림 3) Exponent Module 동작

제안 알고리즘에서 Exponent Module 동작 절차는 설명은 (알고리즘 1) 같다.

(알고리즘 1) Exponent Module 동작 절차

1. X^k mod n 연산시 X 값을 Q에 복사함
2. Euclide Number Inverter모듈을 이용하여 X의 역원 X'를 구함
3. RENAF Method를 이용하여 k값을 분해한 후 배열 R[]에 역으로 저장함
4. 처음 분해된 값 1은 연산하지 않으므로 R[]의 길이를 구한 후 -2하여 i에 저장
5. Q = Q*Q 연산한 Q 값을 mod n 연산함
6. 배열 R[i] 확인하여 분해값이 0이고 i가 0이 아니라면 다시 Q = Q*Q, Q = Q mod n 실행으로 옴
7. 배열 R[i] 값이 -1이면 Q = Q*X' 연산후 Q = Q mod n을 실행하고 i가 0이 아니라면 Q = Q*Q, Q = Q mod n 실행으로 돌아옴
8. 배열 R[i] 값이 1이면 Q = Q*X 연산후 Q = Q mod n을 실행하고 i가 0이 아니라면 Q = Q*Q, Q = Q mod n실행으로 돌아옴
9. 만약 배열 R[i]의 i가 0이라면 종료
10. Q에는 X^k mod n 연산 결과가 저장됨
11. Q를 리턴 함

(알고리즘 1)에서 X 값이 3, k 값이 7, n 값이 17라고 가정하면 X의 역원 X'=6이 된다. RENAF Method에 의해 분해된 k 값은 100-1이 되며 배열 R[]에는 역으로 저장되므로 R[]={-1,0,0,1}이 저장된다. i는 2가되며 현재 Q의 값은 X 값이 복사되므로 3을 가지게 된다. 최초 연산 Q=Q*Q는 9=3*3이므로 Q에는 9가 저장되며 이후 연산 Q=Q mod n의 결과는 9이므로 Q의 값은 변화가 없다. R[2]를 확인하면 0이므로 다시 Q=Q*Q 연산으로 돌아오고 연산 결과 Q에는 81이 저장되며 Q=Q mod n의 연산 결과는 13이 된다. R[1]을 확인하면 역시 0이므로 Q=Q*Q 연산으로 돌아오고 연산 결과는 Q=169가 되며 169 mod 17의 결과는 Q=16이 된다. R[0]를 확인하면 -1이므로 Q=Q*X'(96=16*6) 연산을 하며 Q에는 96이 저장되고 96 mod 17의 결과인 11이 최종 Q의 값이 되어 리턴한다. 3^7의 값은 2,187이며 이 값을 mod 17 하면 11이므로 제안 알고리즘의 연산 결과와 같은 결과를 가지게 된다.

3.3 제안 알고리즘의 성능 평가

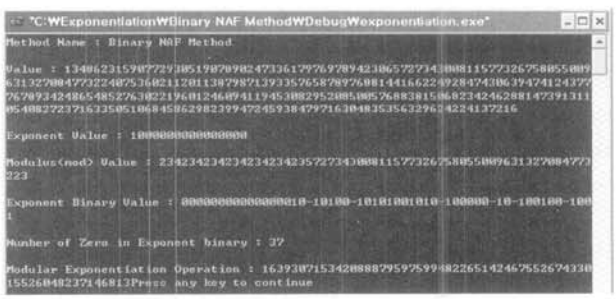
제안 알고리즘의 성능 평가는 제안 알고리즘의 복호화 연산 기능 효율과 기존 Method를 이용한 모듈러 멱승 알고리즘의 연산 시간을 비교하는 방식을 채택하였다. 비교 대상 알고리즘에서 사용되는 지수 분해 Method는 Binary Method와 Window Method를 선택하여 구현하였다.

제안 알고리즘은 두 가지 버전을 구현하였는데 이유는 초기 지수 값이 10비트 단위일 때 연산 시간이 1ms 단위가 측정되기 때문이다. 1ms 단위의 연산 시간은 프로그램 작동시 오차범위에 포함되어 기존 알고리즘과 정확한 비교를 할 수 없다. 따라서 Base가 되는 사칙 연산 알고리즘 기능을 낮춰 초기 연산부터 10ms 이상 연산 시간을 가지도록 하였다. 다음 <표 3>은 제안 알고리즘의 구현환경이며 구현 언어는 C를 사용하였다.

〈표 3〉 제안 알고리즘의 구현환경

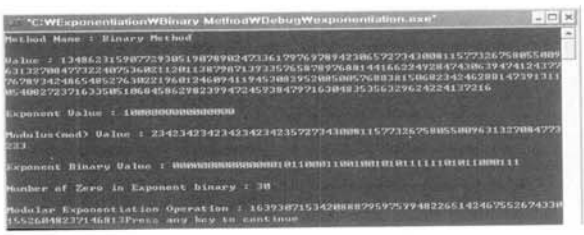
구분	구성요소		사양
H/W	SA	CPU	Intel Duo Core 2.66Ghz
		RAM	2GB
		이더넷 카드	Realtek RTL-8168
		디스플레이	GeForce 7300
S/W	운영 체제		Windows XP pro
	개발 플랫폼		Visual Studio 6.0

제안 알고리즘의 구현 결과는 콘솔화면에 출력할 수 있도록 하였고 실제 RSA 알고리즘에서는 내부 연산이므로 출력을 하지 않아도 된다. 다음 (그림 4)는 제안 알고리즘의 구현 결과이다.

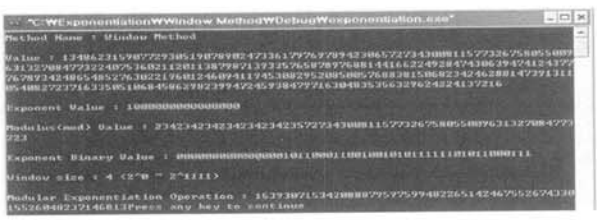


(그림 4) RENAF Method를 이용한 제안 알고리즘의 구현 결과

(그림 4)에서 지수 1,000,000,000,000,000의 분해 값은 Modulus(mod) 아래 화면에 출력되며 결과는 0000...-1001로 이때 0이 많이 출력 될수록 기존 Binary Method를 이용한 모듈러 멱승 알고리즘보다 연산 횟수가 줄게 된다. 제안 알고리즘을 구현하여 실제 연산 시간을 측정한 결과 지수 값이 10비트 이하일 경우 연산 시간이 1ms~3ms로 측정되었으며 모든 변수를 1,024Bit 정수로 설정하였을 때는 0.7sec~0.8sec로 측정되었다. 다음 (그림 5)와 (그림 6)은 연산 시간 비교를 위한 Binary Method와 Window Method를 이용한 모듈러 멱승 알고리즘을 구현한 결과이다.



(그림 5) Binary Method를 이용한 모듈러 멱승 알고리즘 구현 결과



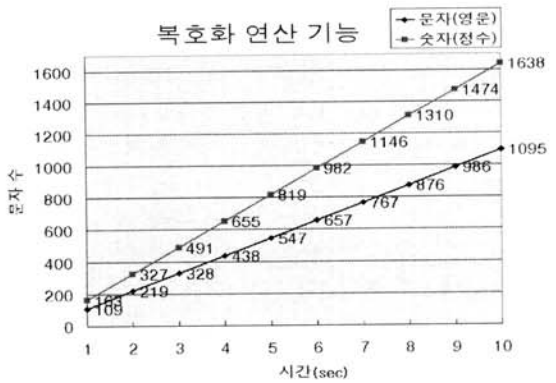
(그림 6) Window Method를 이용한 모듈러 멱승 알고리즘 구현 결과

(그림 5)의 Binary Method를 이용한 모듈러 멱승 알고리즘은 X^k mod n에서 k 값을 단순히 이진 분해하여 연산한다. 이때 분해값이 0이면 자승 연산을 하고 분해값이 1이면 자승 연산후 승산 연산을 실행한다[13]. (그림 6)의 Window Method를 이용한 모듈러 멱승 알고리즘은 X^k mod n에서 k 값에 해당하는 일정 값을 미리 연산하여 배열에 저장한 후 대입하는 방법이다. 즉 Window가 4라면 $X^0 \sim^{(1111)_2}$ 값을 미리 계산하고 k의 분해 값에 따라 해당하는 값을 대입한다[14].

구현된 결과에서 알 수 있듯이 Binary Method의 k 분해값의 0의 수는 RENAF Method보다 작다. 이는 제안 알고리즘의 연산 횟수가 Binary Method를 이용한 모듈러 멱승 알고리즘보다 적다는 걸 증명한다. 따라서 전체 연산 시간 역시 제안 알고리즘이 더 효율적이게 된다. Window Method를 이용한 모듈러 멱승 알고리즘의 경우 Window를 4로 설정하였으며 실제 시간을 체크하여 비교한 결과 Binary Method를 이용한 모듈러 멱승 알고리즘보다 효율적이나 제안 알고리즘보다는 느린걸 알 수 있었다.

4. 제안 알고리즘의 성능 분석

제안 알고리즘의 성능 분석에서 복호화 연산 기능 측정은 밑수 X 값과 지수 k 값, mod n 값을 1,024bit 정수로 설정하고 10sec 까지 복호 할 수 있는 문자수를 도출하였다. (그림 7)은 제안 알고리즘의 복호화 연산 기능을 나타낸다. (그림 7)과 같이 본 논문에서 제안하는 알고리즘을 이용해 복호화를 수행할 경우 1초에 영문자는 약 109자, 숫자는 정수를 기준으로 163자를 복호할 수 있다. 일반적으로 암호 알고리즘이 적용되는 결제 시스템 등은 주민등록번호, 카드번호 등 중요 정보를 취급함에 있어 약 50자가 넘지 않으므로 1초 안에 중요 개인정보를 모두 암·복호 할 수 있는 것이다. 또한 한번 공유된 키들은 일정 시간 동안 업데이트 하지 않고도 사용할 수 있는 것을 고려하면 10초에 1,000자 이상의 문자를 연속적으로 복호할 수 있다.



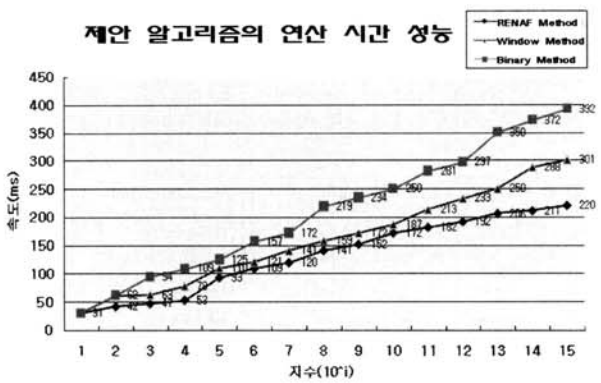
(그림 7) 제안 알고리즘의 복호화 연산 기능

제안 알고리즘의 성능 분석에서 연산 시간 비교는 밑수 X 값과 mod n의 값을 1,024bit 정수로 설정하고 지수 k 값을 10~10^{15} 까지 증가시켜 연산 시간을 측정하였다. 이때 Base로 사용되는 사칙연산 알고리즘은 10진수를 1자리 단위로 연산하는 알고리즘으로 초기 지수 값이 작아도 10ms 이상 연산 시간이 걸리게 된다. 따라서 초기 지수 값이 10비트 이하가 되더라도 연산 시간이 오차범위에 들지 않아 연산 시간 비교를 정확히 할 수 있다. 다음 <표 4>는 RENAF Method를 이용한 제안 모듈러 멱승 알고리즘과 Binary Method, Window Method를 이용한 모듈러 멱승 알고리즘의 연산 시간을 나타낸다.

〈표 4〉 제안 알고리즘과 Binary Method, Window Method를 이용한 모듈러 멱승 알고리즘의 연산 시간(단위 : ms)

종류/지수	10^2	10^4	10^6	10^8	10^{10}	10^{12}	10^{14}
RENAF Method	42	53	109	141	172	192	211
Binary Method	62	109	157	219	250	297	372
Window Method	62	78	121	159	187	233	288

(그림 8)은 각 모듈러 멱승 알고리즘의 연산 시간을 그래프로 표현한 것이다. (그림 8)에서 각 모듈러 멱승 알고리즘은 지수 값이 증가할 때마다 연산 시간이 증가함을 알 수 있다. 이때 제안 알고리즘은 Binary Method를 이용한 모듈러



(그림 8) 제안 알고리즘과 Binary Method, Window Method를 이용한 모듈러 멱승 알고리즘 연산 시간 비교

멱승 알고리즘을 기준으로 평균 38.3% 연산 시간 단축 효과를 나타냈고 Window Method를 이용한 모듈러 멱승 알고리즘 기준으로는 평균 17%의 연산 시간 단축 효과를 나타냈다. 또한 지수 값이 클수록 연산 시간 차이가 커지게 되는데 이는 제안 알고리즘이 기존 Method들 보다 자승 연산 횟수가 늘고 자승 연산 후 승산 연산을 하는 횟수가 줄기 때문이다. 따라서 본 논문에서 제안하는 모듈러 멱승 알고리즘은 시스템 부하가 큰 RSA 알고리즘에 적용할 경우 상당한 효과를 볼 수 있을 것으로 판단된다.

5. 결 론

본 논문에서는 보안성과 안정성을 확보하고 있는 공개키 암호 알고리즘 중 가장 보편화되어 사용되어지는 RSA 알고리즘의 부하를 줄이기 위해 핵심 연산인 모듈러 멱승연산을 고속으로 연산할 수 있는 알고리즘을 설계했다. 제안 알고리즘은 ECC 알고리즘에서 고속 스칼라곱 연산을 위해 사용되는 Binary NAF Method를 수정하여 적용하였으며 X^k mod n 에서 k 값을 NAF 함수를 이용하여 분해한 후 다시 X 값을 이용해 가중치를 대입하여 연산하도록 하였다. 제안하는 알고리즘은 C언어를 기반으로 Windows XP에서 Visual Studio 6.0으로 구현하였으며 구현된 알고리즘의 성능 평가를 위해 Binary Method와 Window Method를 이용한 모듈러 멱승 알고리즘을 같은 조건에서 구현하였다.

성능 평가와 분석을 통해 본 논문에서 제안하는 모듈러 멱승 알고리즘의 복호화 연산 효율성을 측정했으며 연산 시간 비교에서는 제안 알고리즘이 Binary Method와 Window Metho를 이용한 모듈러 멱승 알고리즘에 비해 각각 평균 38.3%, 17%의 연산 시간 단축 효과를 나타냄을 확인 했다. 특히 지수 k가 클수록 상대적으로 Binary Method와 Window Method를 이용한 모듈러 멱승 알고리즘 보다 연산 횟수가 크게 줄게 되며 이러한 특징으로 제안 알고리즘은 지수가 큰 복잡한 연산일수록 효율성이 극대화됨을 알 수 있다.

참 고 문 헌

[1] 이주철, "정보보호와 전자문서 유통확대를 위한 전자서명의 효율적 이용방안", 전북대학교 학위논문, pp.18-19, 2006. 2.

[2] 민병관, "암호화 기술의 최근 동향", 전자부품연구원, pp.3-4, 2004. 02. 19.

[3] 곽미숙, "정보보호에서의 기술적 보안(암호화 기법)", 서울대학교의과대학, pp.3-9, 2005. 10. 25.

[4] A.J Menezes, P.C Van Oorschot and S.A Vanstorne, "Handbook of Applied Cryptography," CRC Press, Boca Raton, Florida, pp.175, pp.285-290, 1997.

[5] R. Rivest, A. Rhamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," Communications of the ACM, pp.8-10, February, 1978.

[6] http://ko.wikipedia.org/wiki/RSA

[7] P. Findlay and B. Johnson, "Modular exponentiation using recursive sums of residues," in Crypto 89, pp.371-386, 1990.

[8] 원동호 외, "전자결재용 서명 고속화를 위한 32비트 고속연산 S/W 개발", 한국정보보호센터, pp.5-7, 1997. 9.

[9] F. Al-Somani and M. .K. Ibrahim, "High Performance Elliptic Curve GF(2m) Cryptoprocessor Secure Against Timing Attacks," IJCSNS, Vol.6, No.1B, pp.179-181, 2006.

[10] http://en.wikipedia.org/wiki/Non-adjacent_form

[11] http://en.wikipedia.org/wiki/Euclidean_algorithm

[12] 이광호 외, "개선된 확장 유클리드 알고리듬을 이용한 유한체 나눗셈 연산기의 하드웨어 설계", 한국정보과학회 학술대회, pp.64-65, 2005. 11.

[13] JH Yang and CC Chang, "Efficient residue number system iterative modular multiplication algorithm for fast modular exponentiation," Computers & Digital Techniques, pp.4-5, November, 20, 2008.

[14] Naofumi Homma et al, "Collision-based Power Analysis of Modular Exponentiation Using Chosen-message Pairs," CHES2008, LNCS5154, pp.15-29, August, 2008.

김 갑 열

e-mail : kkl81@naver.com

2006년 한국교육개발원(학사)

2007년~현 재 경원대학교 전자계산학과 석사과정

관심분야 : RFID/USN, 암호 알고리즘, 네트워크 시큐리티

이 철 수

e-mail : csl100@kyungwon.ac.kr

1964년 육군사관학교(학사)

1972년 서울대학교 수학과(학사)

1977년 KAIST 전산학과(석사)

1980년 KAIST 전산학과(박사)

1993년~1998년 한국전산원 원장

1998년~2000년 한국정보보호진흥원장

2003년~현 재 경원대학교 컴퓨터소프트웨어학과 정교수

관심분야 : 정보보안, 정보시스템 감리, 정보화 정책

박 석 천

e-mail : scpark@kyungwon.ac.kr

1977년 고려대학교 전자공학과(학사)

1982년 고려대학교 컴퓨터공학(석사)

1989년 고려대학교 컴퓨터공학(박사)

1979년~1985년 금성통신연구소

1991년~1992년 UC, Irvine Post Doc.

1988년~현 재 경원대학교 컴퓨터공학과 정교수

관심분야 : 차세대 인터넷, 멀티미디어 통신, 네트워크 시큐리티, 액티브 네트워크