타원곡선 공개키 생성을 위한 고속 스칼라곱 연산 시스템 구현

김갑열* · 이철수* · 박석천**

Development of High Speed Scalar Product Operation System for ECC Public Key

Kap-yol Kim* · Chul-soo Lee* · Seok-cheon Park**

이 논문은 2008년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2008-2-D01031)

요 약

온라인에서 유통되는 수많은 정보들은 데이터로 제작되어 전송되고 있으며 이들 정보들은 타인에 노출될 경우 큰 문제가 발생할 수 있는 정보들이 다수 차지하고 있다. 따라서 과거부터 최근까지 보호가 필요한 정보들의 안전한 유통채널 확보를 위해 다양한 방법들이 연구되고 있으며 그 대표적인 기술이 암호 시스템이다. 암호 시스템은 다양한 수학적 이론과 다양한 암호 알고리즘을 바탕으로 구현되므로 고속 연산의 중심을 두고 연구가 진행되어 지고 있다. 하지만 최근 들어 비공개키 암호 시스템에 비해 공개키 암호 시스템에 대한 연구는 지지부진한 형편이며 따라서 본 논문에서는 대표적인 공개키 암호 시스템인 ECC 암호 시스템에서 고속 공개키 생성을 위한 스칼라곱 연산 시스템을 연구하였다. 제안하는 시스템은 실제 구현을 통해 기존 이진 NAF 시스템과 이진 검색 시스템을 비교·분석 하였고 그 결과 본 논문에서 제안한 시스템이 기존 시스템보다 공개키 생성의 시간적 효율이 우수한 것을 확인하였다.

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.

키워드

암호화, 타원곡선, 공개키, 암호 시스템

Key word

RSA, ECC, Binary Search, Binary NAF

* 경원대학교 컴퓨터공학
** 경원대학교 컴퓨터공학 정교수 (교신저자)

접수일자 : 2009. 11. 25
심사완료일자 : 2009. 12. 14

Ⅰ. 서 론

암호화는 확산과 혼동을 사용하여 암호문 상에서 보여지는 평문의 통계적 특성을 없애고 평문과 암호문간의 대응관계를 최소화 시켜야하며 이러한 목적을 달성하기 위해 다양한 수학이론들이 사용되고 있다. 암호화를 사용하기 위해서는 암호 알고리즘과 키가 사용되는데 이때 암호화시키는 키에 의해서 크게 비밀키 암호 알고리즘과 공개키 암호 알고리즘으로 나눌 수 있다[1][2].

비밀키 암호 알고리즘은 암호문을 공개된 채널에서 전달하고 비밀키는 안전한 채널로 공유하는 방법을 사용하는 알고리즘으로 구현이 간단하고 암·복호화 속도가 빨라 최근까지도 많이 사용되있다. 하지만 비밀키 암호 알고리즘은 비밀키 공유의 안전한 채널 확보가 어렵고 보안상의 문제가 존재한다.

따라서 키 생성과 분배, 암·복호 등의 안전을 확보하고 있는 공개키 암호 알고리즘의 고속 구현 연구가 끝임없이 진행 중에 있다[3]. 공개키 암호알고리즘의 가장 대표적인 예로는 1978년 소인수분해의 어려움에 기반을 둔 RSA(Rivest Shamir Adleman) 암호 알고리즘과 유한체 위에 이산대수문제를 기반으로 한 ECC(Elliptic Curve Cryptosystem)알고리즘을 들 수 있으며 Diffie-Hellman 키 분배 알고리즘, 미국 연방 정보 처리 표준인 DSA(Digital Signature Algorithm) 등이 공개키 암호 알고리즘에 속한다[4~8].

본 논문은 공개키 암호 알고리즘 중 안전도와 연산 속도면에서 가장 효율적으로 알려진 ECC 암호 알고리즘의 고속 구현을 위해 고속 공개키 생성 시스템을 제안하였으며 제안한 시스템은 실제 구현을 통해 공개키 생성 시간을 측정하여 다른 스칼라곱 시스템과 비교 분석하였다.

Ⅱ. 관련 연구

2.1 ECC 알고리즘

ECC 알고리즘은 1985년 N. Koblitz와 V. Miller에 의해 제안된 알고리즘으로 타원곡선 알고리즘으로도 불리며 유한체 위에서 정의된 타원곡선 군(Group)에서의 이산대수 문제의 어려움에 기초한 암호 알고리즘이다[9]. Elliptic Curve는 약 150년 전부터 수학적으로 광범위한 연구가 있어 왔고 최근 Andrew Wiles의 Fermat's Last Theorem 증명에서 중요하게 사용되기도 하였다. ECC 알고리즘은 약 10년 전 부터 비트당 안전도가 타 공개키 암호 알고리즘보다 효율적이라는 것이 알려졌고 또한 ECM(Elliptic Curve Method)은 RSA 암호 알고리즘의 근간이 되는 인수분해 문제와 소수성 테스트를 위한 효율적 알고리즘을 제공하기도 하였다.

일반적으로 160비트 키 사이즈를 가지는 ECC 알고리즘은 1,024비트 키 사이즈를 가지는 RSA 알고리즘과 대등한 안전도를 가진다고 알려져 있으며 특히 160비트 키 사이즈와 193비트 키 사이즈를 가지는 ECC 알고리즘은 각각 10년에서 20년 이상 보안강도를 가진 것으로 평가되어 여타 암호 알고리즘에 비해 효율성을 인정받고 있다[10]. 다음 표 1은 RSA와 ECC의 키 사이즈에 비례한 보안 강도를 보여준다[11].

표 1. RSA 키 와 ECC 키의 보안 강도
Table. 1 Security Strength of RSA and Ecc key

RSA	ECC	RSA/ECC
152 bit	106 bit	5 : 1
768 bit	132 bit	6 : 1
1,024 bit	160 bit	7 : 1
2,048 bit	210 bit	10 : 1
21,000 bit	600 bit	35 : 1

위와 같이 ECC 알고리즘은 작은 키를 이용하여 높은 보안강도를 기대할 수 있는 특징으로 제한적 메모리 공간과 저 전력을 가지는 모바일 환경의 단말기(예, 휴대폰, PDA)등에 구현하기 용이하다. 또한 현재까지 가장 사용량이 많은 RSA 암호 알고리즘이 유한체위의 곱셈 연산을 주로 사용하는 반면 ECC 알고리즘은 타원곡선상의 덧셈 연산을 주로 사용하기 때문에 계산에 대한 속도 역시 타 시스템 보다 빠르며 하드웨어와 소프트웨어 구현이 용이하게 된다[12].

2.2 이진 NAF 메소드

이진 NAF(Non Adjacent Form) 메소드는 ECC 알고리즘에서 타원곡선상의 한점 G를 임의의 난수 k만큼 스칼라곱할 때 고속으로 연산하기 위한 기법으로 사용되어지고 있다. 정수 k를 NAF 함수로 분해하여 나오는 경우의 수에 따라 연산을 달리하여 kG에서 연산의 수를 줄일 수 있다. 정수를 분해하는 NAF 함수는 다음 그림 1과 같

다[13].

```
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
Fig. 1 movement of NAF

NAF의 기본 원리는 정수의 이진 값에서 1을 인접하지 않게 배치하여 + 및 - 연산을 통해 연산 횟수를 줄이고 정확한 수로 다시 조합하는데 있다[14]. 다음 표 2는 NAF를 이용한 정수 연산 예제를 비교한 것이다.

표 2. NAF를 이용한 정수 연산
Table. 2 number operation used NAF

이진 값	연산 결과
$(0111)_2$	4+2+1=7
$(10-11)_2$	8−2+1=7
$(1-111)_2$	8−4+2+1=7
$(100-1)_2$	8−1=7

표 2와 같이 정수 7을 이진 값으로 나타내어 연산할 수 있는 경우의 수는 4가지가 나오지만 연산 횟수가 가장 적은 것은 (100-1)2로 1의 값이 가장 멀리 떨어진 형태를 가진다. 이와 같은 원리는 모든 정수에 적용되며 적절하게 분해하여 연산하게 될 경우 획기적으로 연산 횟수를 줄일 수 있다. 즉 NAF 함수를 이용하여 7X 계산하게 되면 6번 더하는 연산 횟수를 4번의 연산으로 줄일 수 있다. 이때 곱 연산은 단순히 변수의 값을 복사하여 다시 더하는 작업이므로 실제 연산에서는 곱하기 연산을 하지 않고 더하기 연산을 하게 된다.

2.3 ECDH

ECC 알고리즘을 이용한 암호 시스템은 우선 난수와 결합한(예, G+G+G... = kG) 공개키를 송·수신 디바이스에 공유하여 공격자가 유추할 수 없는 안전한 비밀키를 동기화하고 동기화된 비밀키를 이용 메시지 및 인증 데이터와 결합하여 암호화하는 순서로 진행된다. 이와 같은 암호 시스템을 구현하기 위해서는 키 분배 알고리즘과 메시지 암호 알고리즘이 구성되어야 하며 ECC 기반의 키 분배 알고리즘 ECDH(Elliptic Curve Diffie-Hellman)가 대표적이다. ECDH 알고리즘은 유한체위의 Diffie-Hellman 알고리즘을 그대로 타원곡선 위에서 변환한 것으로 본질적으로 Diffie-Hellman 알고리즘과 동작 방법이 같다[9]. 다음 그림 2는 ECDH 알고리즘의 동작 과정이다.

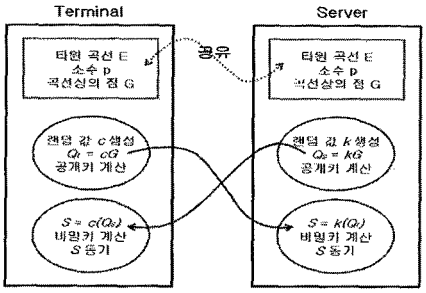

그림 2. Exponent Module 동작
Fig. 2 movement of Exponent Module

그림 2에서 최초 터미널과 서버는 타원곡선 E(E를 구성하는 파라미터 포함)와 곡선의 범위를 나타내는 p, 곡선상의 임의 점 G를 공유한다. 각 디바이스들은 랜덤 값 c, k를 생성하고 임의 점 G를 랜덤 값과 스칼라곱 (G+G+G...)하여 공개키를 생성, 교환한다. 교환한 공개키는 다시 자신의 랜덤 값과 스칼라곱하며 계산된 결과 값은 서로 같은 값을 가지는 비밀키 S가 된다. 이와 같은 방법으로 동기된 S는 서명을 위한 도구나 메시지를 암호화하는 암호키로 이용할 수 있고 메시지를 암호화하는 방법으로는 EC-ElGamal이 대표적이다. 본 논문에서는 ECDH 키 분배 알고리즘 동작 과정에서 공개키를 고속으로 생성하기 위한 독립된 시스템을 구현한다.

III. 고속 스칼라곱 연산 시스템 설계

ECC 알고리즘에서 고속 스칼라곱 연산은 이진 NAF 메소드만을 이용하나 이는 한점 G에 대해 매번 랜덤 값만큼의 스칼라곱 연산을 하는 부담을 가진다. 따라서 본 논문에서 제안하는 고속 스칼라곱 연산 시스템은 타원곡선상의 한 점 G에 대해 1비트에서 160비트까지 연산된 좌표 테이블과 이진 NAF 메소드를 이용한 스칼라곱

과 타원곡선상의 덧셈을 이용하여 조합된 공개키를 생성하는 방법을 사용한다.

3.1 제안 시스템 개요

기존 이진 NAF 메소드는 ECC 알고리즘에서 타원곡선상의 한 점의 스칼라곱을 빠르게 하기 위해 타원곡선상의 덧셈 연산을 줄이는 방법으로 사용되며 현존하는 스칼라곱 연산중 가장 빠른 것으로 알려져 있다. 하지만 이진 NAF 메소드에 의한 스칼라곱 연산은 점 G를 1로 정의하여 생성되는 랜덤값 만큼 더하는 연산을 하므로 비슷한 연산을 매번 반복할 수 밖에 없다. 일반적으로 미리 공유된 점 G는 특별한 경우가 아니면 교체되지 않으므로 암호 시스템의 업데이트를 하기 전까지는 계속 유지되어야 한다. 따라서 점 G에 대한 스칼라곱 연산의 중간 과정들은 언제나 노출 위험이 있고 실제 노출되더라도 타원곡선의 각 파라미터 중 한 가지 값이라도 모르거나 ECC 알고리즘의 점에 대한 타원곡선상의 덧셈 또는 뺄셈 등의 기법을 알지 못한다면 노출된 중간 과정의 값들은 키를 해석하데 의미가 없게 된다.

본 논문에서 구현한 고속 스칼라곱 연산 시스템은 이러한 이론을 바탕으로 실제 구현할 kG의 값을 점 G에서부터 스칼라곱 하는 것이 아니라 점 G에 대한 1비트(1G)에서 160비트(1461501637330902918203684832716283019655932542976G)까지의 연산 결과를 테이블로 구성하고 랜덤 값 k가 속한 비트의 10진수를 빼기(자연수 연산) 연산한 후 선택된 비트 테이블의 좌표 값과 k-(k가 속한 비트의 10진수)G를 타원곡선상의 덧셈 연산을 한다. 따라서 본 논문에서 구현한 고속 스칼라곱 연산 시스템은 기존 타원곡선 암호 시스템에서 스칼라곱 연산 범위를 줄일 수 있어 시간과 연산 부하 효율을 얻을 수 있다.

3.2 제안 시스템의 설계

본 논문에서 제안하는 고속 스칼라곱 연산 시스템의 타원곡선 파라미터 값은 다음 표 3과 같다.

표 3. 제안 시스템의 파라미터
Table. 3 parameter of proposed system

파라미터	설 명
E(GF(p))	유한체 GF(p) 위에서 a, b ∈ GF(p)에 의해 정의된 타원곡선
p	타원곡선이 정의되는 유한체의 크기
a, b	타원곡선을 결정하는 방정식의 상수이며 GF(p)에 속한 원소
G, Q, P	타원곡선상의 임의 점
Gx, Gy 등	타원곡선상의 한 점의 X · Y좌표 값
kG	k만큼 스칼라곱 연산한 타원곡선상의 한 점 (공개키)
k	생성된 랜덤 난수(160bit 이하 수)
(k)2	랜덤 난수의 2진 표현
(bit)10	bit 값의 10진 표현
(+), (-)	타원곡선상의 덧셈, 타원곡선상의 뺄셈

다음 그림 3은 본 논문에서 제안하는 고속 스칼라곱 연산 시스템의 구조이다.

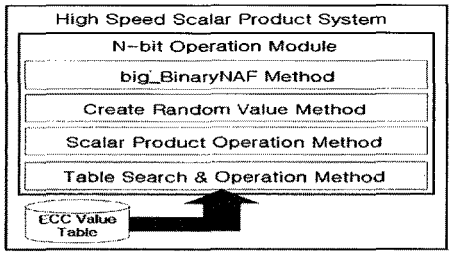

그림 3. 제안 시스템 구조
Fig. 3 architecture of proposed system

본 논문에서 제안하는 고속 스칼라곱 연산 시스템을 구현하기 위해서는 기본적으로 160비트 이상의 사칙연산을 할 수 있는 모듈, 거대 정수의 역원을 구할 수 있는 유클리드 알고리즘, 고속의 mod 연산을 할 수 있는 몽고메리 알고리즘, 랜덤 값을 NAF 함수에 의해 분해 할 수 있는 모듈, 타원곡선상의 점을 연산할 수 있는 모듈, 또한 저장된 테이블에서 값을 추출하기 위해 거대 정수를 비교할 수 있는 모듈과 조합된 타원곡선 점을 연산할 수 있는 모듈 등이 설계되어야 한다. 그림 3.에 나타난 각 모듈의 기능은 다음과 같다.

- N-bit Operation Module : 160비트 이상 정수의 값을 사칙연산 할 수 있는 모듈과 정수의 역원을 구하는 유클리드 알고리즘, 몽고메리 알고리즘, 거대 정수 값을 비교연산 기능 등이 포함된 기본 연산 모듈
- big_BinaryNAF Method : 생성된 160비트 이하의 거대 랜덤 정수를 NAF 함수를 이용 분해 하는 모듈
- Create Random Value Method : 160비트 이하의 거대 랜덤 정수를 추출하는 모듈
- Scalar Product Operation Method : 입력된 정수 만큼 타

원곡선상의 한점(G)을 스칼라곱 하는 기능, 타원곡선 상의 두 점을 (+),(-)연산하는 기능의 모듈

- Table Search & Operation Method : 생성된 랜덤 값과 ECC 비트 테이블의 값을 비교하는 기능, ECC 비트 테이블의 값을 추출하는 기능, 비트 테이블의 타원곡선 점과 스칼라곱된 임의의 점과 (+)연산을 하는 기능이 포함된 모듈

제안하는 시스템에서 핵심 연산을 하는 Table Search & Operation Method를 설계하였으며 동작은 다음 그림 4와 같다.

```
Input: k, G
Output: kG
T ← true
Bit ← (k)2
flag ← (bit)10 > k
while(T)
  if flag is true then
    Q ← Search((bit-1)10G)
    r ← k - (bit-1)10
    P ← scalar(Gx,Gy,Gx,Gy,r,1)
    kG ← scalar(Qx,Qy,Px,Py,0,2)
    T ← false
  else
    if (bit)10 != k then
      Q ← Search((bit)10G)
      r ← k - (bit)10
      P ← scalar(Gx,Gy,Gx,Gy,r,1)
      kG ← scalar(Qx,Qy,Px,Py,0,2)
      T ← false
    else
      k ← Create Random()
return kG
```

그림 4. Table Search & Operation Method 동작
Fig. 4 movement of Table Search & Operation Method

Table Search & Operation Method의 동작 절차의 설명은 다음과 같다.

- 반복문에 대한 제어 변수 T 선언
- 랜덤 값 k를 2진수로 표현하여 Bit 변수에 저장
- Bit 변수의 10진수 정수 값과 랜덤 값 k 비교하여 flag 변수에 저장
- flag = true면 즉, $(bit)_{10}$이 크면 $(bit-1)_{10}$연산하여 해당하는 타원곡선 점을 검색하여 Q에 저장
- 랜덤 값 k에 $(bit-1)_{10}$ 값을 정수 뺄셈 연산하여 r에 저장
- r값 만큼 타원곡선 점 G를 스칼라곱하여 P에 저장
- 점 Q와 점 P를 (+)하여 kG를 구하고 제어 변수 T를 false로 변경함
- flag = false이고 $(bit)_{10}$!= k이면 즉, $(bit)_{10}$이 작으면 $(bit)_{10}$에 해당하는 타원곡선 점을 검색하여 Q에 저장
- 랜덤 값 k에 $(bit)_{10}$ 값을 정수 뺄셈 연산하여 r에 저장
- r값 만큼 타원곡선 점 G를 스칼라곱하여 P에 저장
- 점 Q와 점 P를 (+)하여 kG를 구하고 제어 변수 T를 false로 변경함
- flag = false이고 $(bit)_{10}$ == k이면 랜덤 값 k를 다시 생성
- kG를 반환함

제안하는 시스템에서 두 번째 핵심 연산을 하는 Scalar Product Operation Method를 설계 하였으며 동작은 다음 그림 5와 같다.

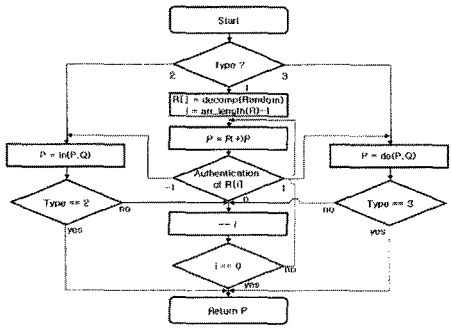

그림 5. Scalar Product Operation Method 동작
Fig. 5 movement of Scalar Product Operation Method

Scalar Product Operation Method는 마지막 Type 인자 값 형태의 따라 (+),(-)연산 또는 스칼라곱 연산을 할 수 있다. Type 인자 값이 1이면 인자 값 중 한 점을 랜덤 값 만큼 스칼라곱 연산을 하며 Type 인자 값이 2이면 인자 값 중 두 타원곡선 점을 (+)연산 한다. 마지막으로 Type 인자 값이 3이면 인자 값 중 두 타원곡선 점을 (-)연산 한다. 구체적인 동작 절차는 다음과 같다.

- 가정 : scalar(G_x, G_y, G_x, G_y, r, 1)
- G_x와 G_x 값이 타원곡선 점을 저장할 수 있는 P, Q 변수에 복사됨, 즉 P와 Q는 G의 값을 가짐
- Type 값 확인 후 만약 1이면 r(Random) 값을 분해함
- Random 값의 분해는 big_binaryNAF Method 즉 decomp()에 의함, Random 값 10의 분해는 010이 됨
- 첫 번째 분해 값이 0이면 P=P(+)P 연산, 즉 P=2P가 됨
- 두 번째 분해 값은 1이므로 P=P+P 연산 후 P=P+Q를 연산. 즉 P=2P+2P+P=5P가 됨(in(P, Q)의 결과는 P+Q와 같음, de(P, Q)의 결과는 P-Q와 같음)
- 다시 세 번째 분해 값이 0이므로 P=P+P 연산 즉

P=5P+5 P=10P의 결과 값을 가짐
- 세 번째 분해 값이 마지막이므로 10P를 반환함

3.3 제안 시스템 구현

본 논문에서 제안 하는 시스템은 ECC 표준에서 정하는 160비트 이상의 파라미터 값을 가지도록 설정하였다. 따라서 프로그램 설치 시 미리 동기 되어야 하는 파라미터 p, a, b, G의 값은 다음 표 4와 같이 정의하였다.

표 4. 고정 파라미터
Table. 4 fix parameter

구분	값
p	1461501637330902918203684832716283019653785059327
a	1461501637330902918203684832716283019653785059324
b	618161358937170673988121756987436237099350920727
Gx	1168983055804381306122964739888353823030159703303
Gy	591673640518579811944247307252990789873540735386

표 4와 같이 값이 설정될 경우 생성되는 키 값은 1,024 비트 키 사이즈를 가지는 RSA 암호 알고리즘과 동등한 수준의 안전성을 가지게 된다. 제안 시스템의 구현은 기본적으로 C++언어로 작성 하였고 C언어로 구성되어야 할 부분은 함수로 구현하여 extern 키워드로 참조 하였다. 다음 표 5.는 제안 시스템의 구현환경이다.

표 5. 제안 알고리즘의 구현환경
Table. 5 development environment of proposed system

구분	구성요소		사양
H/W	SA	CPU	Dual Core 2.66Ghz
		RAM	2GB
		디스플레이	GeForce 7300
S/W	운영 체제		Windows XP pro
	개발 플랫폼		Visual Studio 6.0

제안 시스템 내부에서 큰 정수의 값들의 연산형태는 10진수 배열을 word 단위의 값으로 분할하고 16진수로 변환하여 연산을 하며 대부분의 연산들이 비트 단위로 연산된다. 따라서 10진수의 일반 배열을 이용한 연산보다 월등하게 빠른 연산을 할 수 있다.

제안 시스템의 구현 결과는 다음 그림 6과 같으며 실제 암호 시스템에서는 내부 연산이므로 출력이 되지 않는다. 따라서 모든 값들을 10진수로 변환 후 표준 출력 객체를 이용하여 출력하였다.

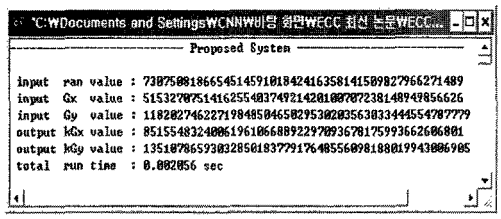

그림 6. 제안 시스템 구현 결과
Fig. 6 development effect of proposed system

Ⅳ. 제안 시스템의 평가

본 논문에서 제안한 시스템의 평가를 위해 타원곡선 암호 시스템에서 현재까지 가장 빠른 스칼라곱을 연산하는 것으로 알려진 이진 NAF 메소드를 이용한 공개키 생성 시스템과 테이블에 저장된 값을 추출하여 랜덤 값 k로 조합하는 형태 즉, k가 7이면 7G를 계산하기 위해 테이블에 1G와2G, 3G를 추출하여 단순히 더하는 공개키 생성 시스템을 구현하였다. 이들 시스템의 구현은 연산 속도 측정의 정당성을 위해 같은 큰 정수 연산 모듈 위에서 동작한다. 다음 그림 7, 8은 이진 NAF 시스템과 이진 검색 시스템의 구현 결과이다.

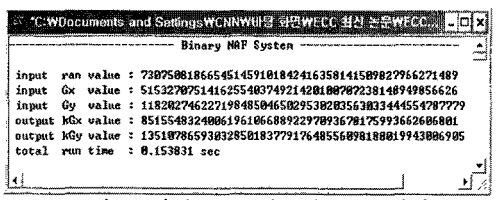

그림 7. 이진 NAF 시스템 구현 결과
Fig. 7 development effect of Binary NAF System

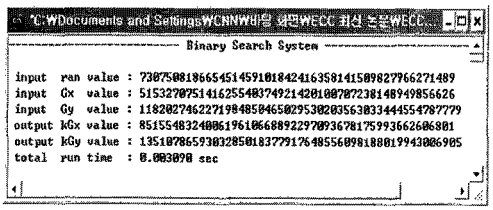

그림 8. 이진 검색 시스템 구현 결과
Fig. 8 development effect of Binary Search System

제안 시스템의 평가는 비교 대상 시스템과 공개키 연산 시간을 측정하여 분석하였으며 공개키를 생성하기 위해 생성되는 랜덤 값 k는 각 비트에서 가장 큰 정수를 가질 수 있는 (n bit-1)값과 각 비트에서 가장 작은 정수를 가질 수 있는 (n bit+1)값을 임의로 생성하여 공개키 연산을 하였다. 이때 n의 값은 50비트에서 160비트까지의 값을 가지며 (50bit-1)은 49비트가 가질 수 있는 가장 큰 정수의 값, (50bit+1)은 50비트에서 가질 수 있는 가장 작은 정수의 값이다. (n bit-1)G의 연산 시간 결과는 다음 표 6과 같다.

표 6. (n bit-1)G의 연산 시간(단위 : ms)
Table. 6 operation time of (n bit-1)G

종류 \ n	50	70	90	110	130	150	160
Binary Search	49	69	90	111	130	151	162
Binary NAF	47	66	85	104	123	142	152
Proposed	44	62	80	97	116	134	143

(n bit-1)G의 연산 시간은 표 6과 같이 연산 시스템의 종류와 상관없이 생성된 랜덤 값이 커질수록 연산 시간이 늘어나는 것을 확인 할 수 있다.

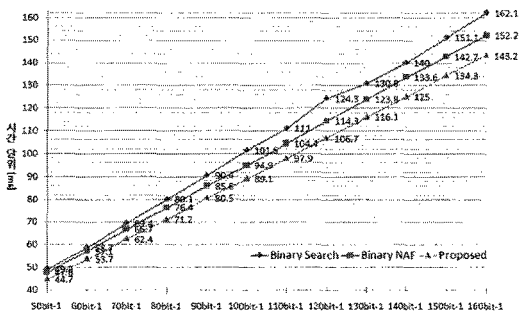

그림 9. (n bit-1)G의 연산 시간
Fig. 9 operation time of (n bit-1)G

표 6에 대한 결과를 그래프로 표현하면 위 그림 9와 같다. 그림 9에서 (n bit-1)G의 연산 시간은 각 비트(n-1 bit)에서 최대의 정수 k 값을 가지게 된다. 제안한 시스템은 미리 연산된 비트 테이블에서 49비트의 G값을 검색하고 나머지 (k-4bit)G 값을 이진 NAF 메소드를 통해 연산한 후 최종적으로 더하는 결과를 가지므로 이진 NAF 시스템에서 처음부터 kG값을 연산하는 것보다 작은 연산을 하게 된다.

이진 검색 시스템은 랜덤 값 k를 기존에 연산된 각 비트 테이블 좌표 값과 조합하는 형태여서 계산의 부하는 적을 것으로 기대 하였으나 실제 Binary Method를 이용한 스칼라곱 시스템과 다를 것이 없다. 즉 미리 연산된 비트 테이블을 이용하는 것과 처음부터 Binary 연산을 하는 것은 차이가 없는 것이다. 이미 순수하게 Binary 연산을 하는 스칼라곱 시스템은 Binary NAF를 이용한 스칼라곱 시스템보다 느리다는 것은 증명된바 있다. 또한 이진 검색 시스템은 랜덤 값 k만큼 비트 테이블을 이용하여 조합하기 위해 k값에 더해야 하는 비트의 좌표 값이 아니더라도 비교와 검색을 통해 더할 수 있는 자리인지 검증한다. 따라서 이진 검색 시스템은 (n bit-1)G 연산에서 가장 불리한 조건의 연산을 하게 되며 실제 다른 시스템과 공개키 생성 시간이 오래 걸리는 것으로 나타났다. 또한 각각의 시스템의 (50 bit-1)G에서 (160 bit-1)G의 연산시간의 평균을 측정한 결과 이진 NAF 시스템은 1199.7(ms), 이진 검색 시스템은 1268.6(ms), 제안 시스템은 1124.8(ms)의 연산 시간을 나타내었으며 제안 시스템을 기준으로 시간 효율은 이진 NAF 시스템이 6.65%의 시간 딜레이를 가졌고 이진 검색 시스템은 12.78%의 시간 딜레이를 가졌다. 다음 표 7은 (n bit+1)G의 연산 시간 결과를 나타 낸 것이다.

표 7. (n bit+1)G의 연산 시간(단위 : ms)
Table. 7 operation time of (n bit+1)G

종류 \ n	50	70	90	110	130	150	160
Binary Search	1.2	1.4	1.7	2.2	2.6	3.0	3.3
Binary NAF	47	66	86	105	124	142	152
Proposed	1.2	1.4	1.5	1.7	1.9	2.1	2.3

(n bit+1)G의 연산 시간은 표와 같이 이진 NAF 시스템의 공개키 생성 시간이 다른 시스템과 비교할 수 없을 정도로 느린 것이 확인 가능하다. 이는 순수하게 랜덤 값 k를 이용하여 kG까지 순차적으로 연산해 나가는 이진 NAF 시스템과 달리 제안 시스템인 이진 검색 시스템은 이미 계산된 좌표 값을 이용하기 때문에 k값이 n bit에서 가까울수록 연산이 크게 줄기 때문이다. 따라서 (n bit+1)G의 연산 시간 그래프는 제안 시스템과 이진 검색 시스템의 연산 시간 결과만을 이용하여 다음 그림 10과 같이 표현하였다.

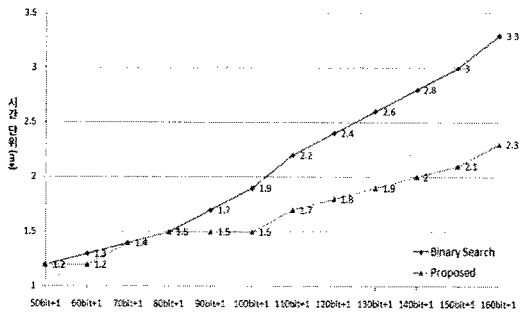

그림 10. (n bit+1)G의 연산 시간
Fig. 10 operation time of (n bit+1)G

그림 10에서 (n bit+1)G의 시간은 제안한 시스템이 임의로 생성되는 랜덤 값 k가 클수록 이진 검색 시스템보다 효율적인 것을 알 수 있다. (n bit+1)G의 연산은 미리 계산된 비트 테이블에 n bit 값의 좌표를 검색하고 이 값에 G값만을 더하는 연산을 한다. 이때 제안 시스템은 n bit 값의 좌표만을 검색하고 나머지 더해야 할 값에 대해서는 개의치 않고 무조건 이진 NAF 메소드에서 값을 구해 더할 수 있도록 구현되어 있다. 즉, 더 이상 랜덤 값 k에 의해 몇 비트의 G값을 더해야하는지 비교・검색할 필요가 없게 된다. 이진 NAF 메소드는 1G의 값을 연산하게 되지만 실제 1G는 현재 시스템에 고정된 G값이므로 연산없이 바로 n bit의 좌표 값과 더하게 된다.

이진 검색 시스템의 경우 n bit의 좌표를 검색하고 이후 k 값이 조합될 때까지 차례로 k 값에 비트 테이블의 좌표 값이 더할 수 있는 값인지 비교하는 과정을 거치게 되고 최종적으로 G값을 더하므로 비교 검색하는 과정의 부하가 연산 시간에 영향을 미치게 된다. 또한 n 값이 커질수록 비교・검색 알고리즘은 부하가 큰 정수 연산을 하게 되므로 n 값에 의해 시간 차이가 커지는 것을 알 수 있다.

최종적으로 이진 검색 시스템과 제한 시스템의 (50 bit+1)G에서 (160 bit+1)G의 연산시간 평균에 대한 효율성을 측정한 결과 이진 검색 시스템은 약 25.3(ms)의 평균 연산 시간을 가졌으며 제안 시스템은 20.1(ms)의 평균 연산 시간을 나타냈다. 따라서 제안 시스템을 기준으로 이진 검색 시스템은 25.8%의 시간적 딜레이를 가진 것을 확인 할 수 있다.

V. 결 론

본 논문에서는 보안성과 안정성을 확보하고 있는 공개키 암호 시스템 중 타원곡선 암호 시스템을 위한 고속 스칼라곱 공개키 생성 시스템을 제안하였다.

제안 시스템은 타원곡선상의 공개키 kG 연산을 위해 미리 분배된 점 G에서부터 스칼라곱 하는 것이 아니라 (1bit)G에서 (160bit)G까지의 연산 결과를 비트 테이블로 구성하고 랜덤 값 k가 속한 비트의 10진수를 빼기(자연수 연산) 연산한 후 선택된 비트 테이블의 좌표 값과 (k-((k)2)10)G를 타원곡선상의 (+)연산을 하도록 구성하였으며 성능 평가를 위해 이진 NAF 메소드를 이용한 공개키 생성 시스템과 비트 테이블의 연산 결과를 이용한 이진 검색 시스템을 각각 구현하여 kG 연산 시간을 측정하였다. 측정 결과 제안 시스템은 (n bit-1)G과 (n bit+1)G의 연산에서 다른 시스템보다 효율적인 것으로 평가 되었으며 각각의 시간 효율을 측정한 결과 (n bit-1)G 연산에서는 제안 시스템이 이진 NAF 시스템보다 약 6.65%의 시간 효율을 보였고 이진 검색 시스템에 비해서는 약 12.78%의 시간 효율을 보였다. 또한 (n bit+1)G 연산에서는 제안 시스템이 이진 검색 시스템보다 약 25.8%의 시간 효율을 보인 것으로 평가되었다.

본 논문에서 제안하는 시스템은 안전한 데이터 유통 채널을 확보하기 위해 정기적으로 160비트 테이블에 대한 업데이트가 요구되므로 업데이트에 대한 비용 부담이 증가될 수 있다. 따라서 비용적인 측면에 부담을 감수할 수 있는 대규모 운영 시스템에서의 구축에 제한된다는 단점을 가질 수 있다. 하지만 본 논문에서 제안한 시스템을 이용하여 타원곡선 암호 시스템을 구축한다면 서버와 클라이언트 사이의 전체 암호 연산 시간을 효율적으로 단축하게 되고 160비트 테이블 값을 정기적으로 서버측에서 새로운 값으로 업데이트 한다면 보안상의 안전도 역시 문제없이 안전한 데이터 유통채널을 확보할 수 있을 것으로 기대된다.

감사의 글

본 연구는 이 논문은 2008년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었습니다. (KRF-2008-2-D01031) 관계부처에 감사 드립니다.

참고문헌

[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] 유영준, "디지털 암호화 기술 현황", 전자정보센터, 2006. 11.

[5] A.J Menezes,P.C Van Oorschot, S.A Vanstorne, "Handbook of Applied Cryiptography," CRC Press, Boca Raton, Florida, 1997.

[6] W. Diffe and M. Hellman, "New directions In cryptography," IEEE Transactions on Information Theory, pp. 644~654, Nov. 1976.

[7] http://en.wikipedia.org/wiki/Digital_Signature_Algorithm

[8] Raymond G. Kammer, Director, "DIGITAL SIGNATURE STANDARD(DSS)," U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, 27. January. 2000.

[9] SEC1, "Elliptic Curve Cryptography," v.1.0, pp.62, Sept. 2000.

[10] 포항공대, "부가형 전자서명 방식 표준(안) - 제 3부: 타원곡석을 이용한 인증서 기반 전자서명 알고리즘", TTA.KO, 2000. 12.

[11] Randall K. Nichols, "ICSA Guide to Cryptography," McGraw-Hill, Dec. 1999.

[12] C. G. Pollman, "XML Pool Encryption," XMLSEC02, USA, pp.1-9, 22, Nov. 2002.

[13] 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.

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

저자소개

김갑열(Kap-yol Kim)

e-mail : kkl81@naver.com
2006년 한국교육개발원(학사)
2009년 경원대학교 전자계산학과(석사)

※ 관심분야 : RFID/USN, 암호 알고리즘, 네트워크

이철수(Chul-soo Lee)

e-mail : csl100@kyungwon.ac.kr
1964년 육군사관학교 졸업(학사)
1972년 서울대학교 수학과 졸업 (학사)
1977년 KAIST 전산학과 졸업(석사)
1980년 KAIST 전산학과 졸업(박사)
1993년 ~ 1998년 한국전산원 원장
1998년 ~ 2000년 한국정보보호진흥원장
2003년 ~ 현재 경원대학교 소프트웨어 대학 정교수
※관심분야 : 정보보안, 정보시스템 감리, 정보화 정책

박석천(Seok-cheon Park)

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 ~ 현재 경원대학교 컴퓨터공학과 정교수
※관심분야 : 차세대 인터넷, 멀티미디어 통신, 네트워크 시큐리티, 액티브 네트워크