타원곡선 암호시스템 응용을 위한 마이크로소프트 COM 소프트웨어 모듈 구현†

(Implementation of Microsoft COM Software Modules for Elliptic Curve Cryptographic Applications)

김 태 호*, 김 창 훈*, 남 인 길**, 홍 춘 표*
(Tae Ho Kim, Chang Hoon Kim, In Gil Nam, Chun Pyo Hong)

요 약 본 논문에서는 타원곡선 암호시스템 응용을 위한 마이크로소프트 COM 소프트웨어 모듈을 구현하고 그 성능을 평가한다. 개발된 COM 소프트웨어 모듈은 IEEE 1363의 모든 유한체 GF(p)와 GF(2m)상의 타원곡선 키 교환 프로토콜 및 전자서명 기능을 지원한다. 또한 이 모듈은 컴포넌트 기반 소프트웨어 개발 방법을 지향하기 때문에 생산성이 높으며 개방화, 표준화된 시스템 특성을 가진다. 따라서 C 라이브러리를 이용한 개발 방법에 비해 보다 쉽고 빠르게 소프트웨어를 개발할 수 있다. 게다가 마이크로소프트 COM 인터페이스를 따르기 때문에 타원곡선 암호 시스템에 대한 깊은 지식 없이도 타원곡선 암호 알고리즘에 기반한 보안 소프트웨어를 쉽게 개발할 수 있다.

핵심주제어 : 타원곡선 암호시스템, 유한체 연산, COM, 타원곡선 키 교환, 타원곡선 디지털 서명

Abstract In this paper, we implement Microsoft COM software modules for elliptic curve cryptographic applications and analyze its performance. The implemented COM software modules support all elliptic curve key exchange protocols and elliptic curve digital signature algorithm in IEEE 1363 finite fields GF(p) and GF(2m). Since the implemented software modules intend to focus on a component-based software development method, and thus it have a higher productivity and take systematic characteristics to be open outward and to be standardized. Accordingly, it enable a software to be developed easier and faster rather than a method using C library. In addition, it support the Microsoft COM interface, we can easily implement secure software applications based on elliptic curve cryptographic algorithms.

Key Words : Elliptic Curve Cryptosystems, Finite Field Arithmetic, COM, Elliptic Curve Key Exchange Protocol, Elliptic Curve Digital Signature Algorithm

1. 서 론

1980년대 중반 Victor Miller[1]와 Neal Koblitz[2]에 의해 제안된 타원곡선 암호 시스템(Elliptic Curve Cryptosystem: ECC)은 최근 학계나 산업계로부터 많은 관심을 모으고 있다. ECC의 가장 주된 장점은 RSA[3]나 ElGamal[4]과 같은 다른 암호 시스템에 비해 현저히 작은 키를 사용하면서(약 1/6 정도) 동일한 안전도를 가진다는 사실이다[5,6]. 작은 키를 사용한다는 것은 계산 시간,

† 본 논문은 2005년도 PoP-iT 인력양성 사업에 의하여 지원되었음

* 대구대학교 정보통신공학과
** 대구대학교 컴퓨터·IT공학부

전력 소모 그리고 저장 공간의 감소를 의미한다. 이러한 장점 때문에 최근 IEEE 1363[7] 및 NIST[8]는 공개키 암호 시스템을 위해 ECC에 기반한 타원곡선 전자서명 알고리즘(Elliptic Curve Digital Signature: ECDSA)[5]을 표준으로 채택하였다. ECDSA를 위해 유한체는 $GF(p)$와 $GF(2^m)$을, $GF(2^m)$상의 원소 표기법으로는 가우시안 정규기저(Gaussian Normal Basis: GNB)와 다항식 기저(Polynomial Basis: PB) 표기법을 사용한다. 여기서 p는 소수이고 GNB는 정규 기저(Normal Basis: NB)의 특별한 경우로서 8로 나누어지지 않는 모든 양의 정수 m에 대해 존재한다.

ECC기반 보안 소프트웨어를 개발하기 위해서는 유한체 연산 및 타원곡선에 대한 깊은 지식을 필요로 하기 때문에 RSA나 ElGamal과 같은 다른 암호 시스템에 비해 구현이 상당히 어렵다. 또한 타원곡선 암호 알고리즘 기반 보안 프로토콜을 구현하기 위해서는 AES(Advanced Enc- ryption Standard)[9]와 같은 비밀키 알고리즘, PRNG(Psuedo Random Number Generator)[10] 그리고 SHA-1(Secure Hash Algorithm)[11]과 같은 암호학적 해쉬 함수의 구현을 추가적으로 요구한다[5]. 따라서 ECC기반 보안 소프트웨어를 개발하기 위해서는 상당히 많은 시간과 노력이 필요하다. 뿐만 아니라 다양한 보안 프로토콜은 서로 다른 암호 알고리즘을 요구한다. 예를 들면, ECDH(Elliptic Curve Diffie-Hellman)[5]는 타원곡선 정수 곱셈기와 난수생성기를 요구하는 반면, ECDSA는 ECDH에 SHA-1의 기능을 추가적으로 요구한다. 또한 고속의 데이터 암·복호를 위해 AES와 같은 비밀키 알고리즘을 요구하기도 한다. 이러한 문제점들 때문에 소프트웨어 개발자들은 당연히 각 암호 알고리즘들의 분리된 구현을 요구할 것이다. 더욱이 최근 윈도우 기반 소프트웨어 개발 과정에 있어 대부분의 소프트웨어 모듈은 COM 인터페이스를 지원한다. 이는 소스코드 라이브러리 형태에 비해 훨씬 빠르고 쉬운 소프트웨어 개발 환경을 제공할 뿐만 아니라, 윈도우 기반 소프트웨어 개발 언어와 툴에 독립적이다. 따라서 각 암호 알고리즘의 COM 소프트웨어 모듈의 개발은 여러 가지 장점을 가진다[12].

본 논문에서는 타원곡선 암호 시스템에 대한 깊은 지식 없이 타원곡선 암호 알고리즘 기반 보안 소프트웨어를 쉽게 개발할 수 있는 COM 모듈을 설계 및 구현한다. 이를 위해, 우리는 유한체 $GF(p)$, $GF(2^m)$상의 연산에 필요한 자료구조를 정의하고 타원곡선 정수 곱셈 알고리즘의 COM 인터페이스를 설계 및 구현한다. 또한 ECDH와 ECDSA와 같은 ECC기반 응용 프로토콜을 위한 부가적인 암호 알고리즘의 COM 인터페이스를 설계 및 구현한다. 암호 모듈의 기능검증을 위해 구현된 결과 중 $GF(p)$, $GF(2^m)$상의 타원곡선 정수 곱셈 모듈을 사용하여 ActiveX 컨트롤을 구현하였으며, Visual Basic 6.0을 이용한 기능검증을 통해 정확히 동작함을 확인하였다.

본 논문의 구성은 다음과 같다. 2절에서는 ECC 구현을 위해 필요한 수학적 이론인 유한체 $GF(p)$, $GF(2^m)$을 살펴보고, 본 논문에서 구현한 타원곡선 정수 곱셈 알고리즘에 대해 기술한다. 3절에서 ECDH 및 ECDSA 프로토콜 구현을 위한 부가적인 암호 알고리즘을 살펴보고, 4절에서 COM 소프트웨어 모듈 구현을 위한 자료구조 및 인터페이스 설계한다. COM 모듈의 구현 결과는 5절에서 기술하며, 6절에서 결론을 맺는다.

2. 유한체 및 타원곡선 정수 곱셈 알고리즘

본 절에서는 타원곡선 암호시스템 구현에 필요한 수학적 이론 중 가장 대표적인 유한체 $GF(p)$, $GF(2^m)$에 대해 기술하며, 본 논문에서 구현한 타원곡선 정수 곱셈 알고리즘을 살펴본다. 본 논문에서 설명된 모든 알고리즘은 32비트 마이크로프로세서 플랫폼에서 구현한다고 가정한다.

2.1 유한체

2.1.1 GF(p)

p가 소수이면 원소 p개를 가지는 유일한 $GF(p)$가 존재하며, $GF(p)$상의 원소는 다음과 같은 정수 집합으로 정의된다.

$$Z_p = \{0, 1, 2, \cdots, p-1\} \quad (1)$$

$GF(p)$상의 원소는 32비트 배열에 바이너리로 표현될 수 있다. 즉, $m=\lceil \log_2 p \rceil$ 이고 $t=\lceil m/16 \rceil$ 이면, 원소 a는 배열 $(a_{t-1},\cdots,a_2,a_1,a_0)$의 하위 16비트에 저장될 수 있다. 각 배열의 상위 16비트는 연산에서 캐리가 발생할 경우를 대비하여 비워둔다.

2.1.2 GF(2m)

유한체 $GF(2^m)$은 $GF(2)$상의 m차원 벡터 공간으로 $A \in GF(2^m)$에 대해서

$$A=\sum_{i=0}^{m-1} a_i\alpha_i = a_{m-1}\alpha_{m-1}+\cdots+a_1\alpha_1+a_0\alpha_0, \quad a_i \in GF(2) \tag{2}$$

로 유일하게 표현하는 $GF(2^m)$상의 m개의 원소 집합 $\{\alpha_0,\alpha_1,\cdots,\alpha_{m-1}\}$ 가 존재한다. 이러한 집합 $\{\alpha_0,\alpha_1,\cdots,\alpha_{m-1}\}$ 를 $GF(2)$에 대한 $GF(2^m)$의 기저(basis)라 하며, A를 벡터 $(a_0, a_1, \cdots, a_{m-1})$로 표현한다. 따라서 $GF(2^m)$상의 원소는 길이 m의 비트 스트링으로 나타낼 수 있다. 예를 들어서 $t=\lceil m/32 \rceil$ 일 때, 바이너리 벡터 A는 배열 $(A[t-1],\cdots,A[2],A[1],A[0])$에 저장한다. $GF(2^m)$상의 연산은 캐리가 발생하지 않으므로 덧셈 연산의 경우 XOR로 간단히 구현이 가능하다. 우리는 $GF(2)$에 대한 $GF(2^m)$의 기저로 IEEE 및 NIST 표준인 다항식 기저와 가우시안 정규기저를 구현한다. 우선 $GF(2)$상의 차수 m인 기약 다항식

$$P(x)=x^m+\sum_{i=0}^{m-1} p_i x^i = x^m+p_{m-1}x^{m-1}+\cdots+p_1x+p_0, \quad p_i \in GF(2) \tag{3}$$

의 근을 α라 하면, $P(\alpha)=0$이므로,

$$\alpha^m=\sum_{i=0}^{m-1} p_i\alpha^i = p_{m-1}\alpha^{m-1}+\cdots+p_1\alpha+p_0, \quad p_i \in GF(2) \tag{4}$$

이다. 이러한 성질을 이용하면, 아래의 식 (5)와 같이 $GF(2^m)$의 모든 원소는 차수가 m보다 작은 다항식 기저 $\{\alpha_0,\alpha,\alpha^2,\cdots,\alpha^{m-1}\}$ 의 선형 결합으로 나타낼 수 있다.

$$A=\sum_{i=0}^{m-1} a_i\alpha^i = a_{m-1}\alpha^{m-1}+\cdots+a_1\alpha+a_0, \quad a_i \in GF(2) \tag{5}$$

$GF(2^m)$에서 모든 양의 정수 m에 대한 정규기저는 항상 존재한다. 그러나 가우시안 정규기저는 m이 8로 나누어지지 않을 경우 존재한다. m, k를 소수 $p \neq 2$에 대해서 $p=mk+1$인 양의 정수라 하고, $K=<\tau>$는 $GF(p)^\times$에서 위수(order) k인 유일한 부분군이라 하자. 이 경우 β가 $GF(2^{mk})$상의 p번째 원시근이라면 다음 원소

$$\alpha=\sum_{j=0}^{k-1}\beta^{r^j} \tag{6}$$

를 $GF(2)$상의 (m, k)타입의 가우스 주기(Gauss period)라 한다. 이때 $\mathrm{ord}_p 2$를 mod p에 대한 2의 위수라 하고, $\gcd(mk/\mathrm{ord}_p 2, m)=1$이라고 가정하면, α는 $GF(2^m)$ 상에서 정규 기저의 원소이다. 즉, $0 \le i \le m-1$에 대해, $\alpha_i=\alpha^{2^i}$ 라 놓으면, $\{\alpha_0,\alpha_1,\cdots,\alpha_{m-1}\}$ 은 $GF(2)$상의 $GF(2^m)$에 대한 기저이다. 이러한 기저를 우리는 $GF(2^m)$상의 (m, k) 타입의 가우스 정규 기저 또는 가우스 정규 기저 타입 k라 부른다.

2.2 타원곡선 정수 곱셈 알고리즘

타원곡선은 a와 b가 고정된 실수일 경우에 방정식 $y^2 = x^3 + ax + b$ 을 만족하는 (x, y)점들의 집합을 의미하며, 타원곡선의 이산대수 문제를 암호에 이용한다. 실수 타원곡선군에서 원소들을 계산하는 것은 느리고 오차가 있으므로, 암호에는 일정한 조건을 만족하는 $GF(p)$ 또는 $GF(2^m)$ 상의 타원곡선군을 사용한다. $GF(p)$에서의 타원곡선 방정식은

$$y^2 \bmod p = x^3+ax+b \bmod p \tag{7}$$

와 같은 형태로 표현된다. $GF(p)$는 0 부터 p-1 까지의 수를 원소들로 가지고 있으며 모듈러 연산을 적용한다. $GF(p)$에서 $P+Q$는 점 $P=(x_P,y_P)$, $Q=(x_Q,y_Q)$가 $P\neq -Q$인 서로 다른 점이라면,

$$\lambda=(\frac{y_Q-y_P}{x_Q-x_P}) \bmod p \tag{8}$$

$$\begin{aligned} x_R&=(\lambda^2-x_P-x_Q) \bmod p,\\ y_R&=(\lambda(x_P-x_R)-y_P) \bmod p \end{aligned} \tag{9}$$

이며, λ는 P와 Q를 잇는 직선의 기울기이다. $P=Q$가 같은 점이라면 $2P$는

$$\lambda=(\frac{3x_P^2+a}{2y_P}) \bmod p \tag{10}$$

$$\begin{aligned} x_R&=(\lambda^2-x_P-x_Q) \bmod p,\\ y_R&=(\lambda(x_P-x_R)-y_P) \bmod p \end{aligned} \tag{11}$$

으로 구할 수 있다. 알고리즘 1은 본 논문에서 구현한 $GF(p)$상의 타원곡선 정수 곱셈 알고리즘으로 k의 최상위 비트부터 차례로 읽으면서 0이면 포인트 배 연산, 1이면 포인트 덧셈과 포인트 배 연산을 각각 수행한다. 알고리즘 1에서 만약 k의 각 비트들의 0, 1 비율이 동일하다면, $m/2$번의 포인트 덧셈과 m번의 포인트 배 연산이 필요하다 [13].

[알고리즘 1] : Left-to-right 바이너리 타원곡선 정수 곱셈 알고리즘[13]

Input : $P = (x, y) \in E(GF(p))$, an integer $k \geq 0$
Output : $kP = (x_0, y_0)$
1. $Q \leftarrow O$
2. For i = 0 to m - 1
   2.1 $Q \leftarrow 2Q$
   2.2 If k_i = 1 then $Q \leftarrow Q + P$
3. Return $kP = (x_0, y_0)$

$GF(2^m)$ 상의 타원곡선군은 이진수 표현이 가능하므로 컴퓨터 연산에서 매우 빠르게 동작할 수 있는 장점이 있다. $GF(2^m)$에서의 타원곡선 방정식은

$$y^2+xy=x^3+ax^2+b \tag{12}$$

와 같은 형태로 표현된다. $P+Q$는 점 $P=(x_P,y_P)$, $Q=(x_Q,y_Q)$가 $P\neq -Q$이고 $P\neq Q$인 서로 다른 두 점일 경우

$$\lambda=(\frac{y_Q+y_P}{x_Q+x_P}) \tag{13}$$

$$\begin{aligned} x_R&=\lambda^2+\lambda+x_P+x_Q+a,\\ y_R&=\lambda(x_P+x_R)+x_R+y_P \end{aligned}, \tag{14}$$

이며, $2P$는

$$\lambda=x_P+\frac{y_P}{x_P} \tag{15}$$

$$x_R=\lambda^2+\lambda+a,\ y_R=x_P^2+(\lambda+1)x_R \tag{16}$$

으로 구할 수 있다[14]. 지금까지 설명된 타원곡선 연산은 Affine 좌표계를 사용한 경우이다. 그러나 지금까지 설명된 바와 같이 Affine 좌표계를 사용할 경우 매 연산마다 역원 연산을 필요로 한다. 이러한 역원 연산을 피하기 위해 Projec- tive 좌표계를 사용하며, 현재까지 Projective 좌표계를 이용한 다양한 연구결과가 발표 되었다. 이 중 최근 López-Dahab[15]은 비밀키에 상관없이 동일한 연산을 수행 할 수 있는 알고리즘을 제안 하였으며, 그 결과는 알고리즘 2와 같다. 이 알고리즘에서 비밀키에 상관없이 동일한 연산을 수행하게 되면, 시간 및 전력 공격에 대해서 면역을 가질 수 있다. 따라서 본 논문에서는 EC- DH를 위해 López-Dahab 알고리즘을 사용한다.

3. 타원곡선 보안 프로토콜

ECDH와 ECDSA는 가장 많이 사용되는 ECC 기반 보안 프로토콜이다. 본 절에서는 이 두 프로토콜의 기능에 대해 간단히 살펴본다. 또한 이 두

프로토콜의 구현을 위해 필요한 부가적인 암호 알고리즘에 대해서도 알아본다.

[알고리즘 2] : López-Dahab 타원곡선 정수 곱셈 알고리즘[15]

Input : $P=(x,y) \in E(GF(2^m))$, an integer $k \geq 0$
Output : $kP=(x_0,y_0)$

1. If $k=0$ or $x=0$, then stop and output $kP=\boldsymbol{O}$ or P
2. $k \leftarrow (k_{s-1},k_{s-2},\cdots,k_1,k_0)_2$
3. $(X_1,Z_1) \leftarrow (x,1)$, $(X_2,Z_2) \leftarrow (x^4+b,x^2)$
4. for $i=s-2$ down to 0 do
5. $\quad Z_3 \leftarrow (X_1Z_2 + X_2Z_1)^2$
6. $\quad$ if $k_i=1$ then
$\quad\quad X_1 \leftarrow xZ_3 + (X_1Z_2)(X_2Z_1)$, $Z_1 \leftarrow Z_3$,
$\quad\quad X_2 \leftarrow X_2^4 + bZ_2^4$, $Z_2 \leftarrow X_2^2Z_2^2$
$\quad$ else
$\quad\quad X_2 \leftarrow xZ_2 + (X_1Z_2)(X_2Z_1)$, $Z_2 \leftarrow Z_3$,
$\quad\quad X_1 \leftarrow X_1^4 + bZ_1^4$, $Z_1 \leftarrow X_1^2Z_1^2$
$\quad$ end if
$\quad$ end for
7. $x_0 \leftarrow \frac{x1}{z1}$
8. $y_0 \leftarrow \frac{1}{x} \cdot (x+\frac{X_1}{Z_1})\left\{(x+\frac{X_1}{Z_1})(x+\frac{X_2}{Z_2})+x^2+y\right\}+y$
9. return $kP=(x_0,y_0)$

3.1 ECDH

ECDH는 임의의 두 사용자가 안전한 랜덤 키를 사용할 수 있도록 하며, 이 프로토콜은 타원곡선상의 이산대수 문제에 기반한다. 타원곡선 알고리즘을 이용한 키 교환은 다음과 같이 이루어지며, 선택된 타원곡선 E상의 임의의 점 B를 선택한다. 또한 사용자 A, B는 각각의 개인키 k_A, k_B, 및 점 B를 이용하여 각각의 공개키를 생성한다.

$$P_A = k_AB, \quad P_B = k_BB \tag{17}$$

생성된 공개키를 A, B가 서로 교환하고, 전달 받은 상대방의 공개키와 자신의 개인키를 계산하면,

$$P_S = k_A(k_BB) = k_B(k_AB) \tag{18}$$

형태로 나타나며 서로 동일한 키를 생성하게 된다.

3.2 ECDSA

ECDSA는 서명자의 비밀키와 메시지의 해쉬값을 사용하여 서명을 생성하며, 서명자의 공개키로 서명을 검증한다. 만약, A가 B로 전자서명 된 메시지를 보낸다면, 우선 A, B는 타원곡선상의 점 P를 선택한다. 그리고 A는 개인키 d를 이용하여 알고리즘 3에 나타난 순서로 서명을 생성한다. 여기서 $H()$는 암호학적 해쉬 함수로 n보다 작은 비트 길이의 출력을 가진다. B는 A에게서 받은 서명 (r, s)를 A의 공개키 Q를 이용하여 알고리즘 4에 나타난 순서로 서명을 검증한다.

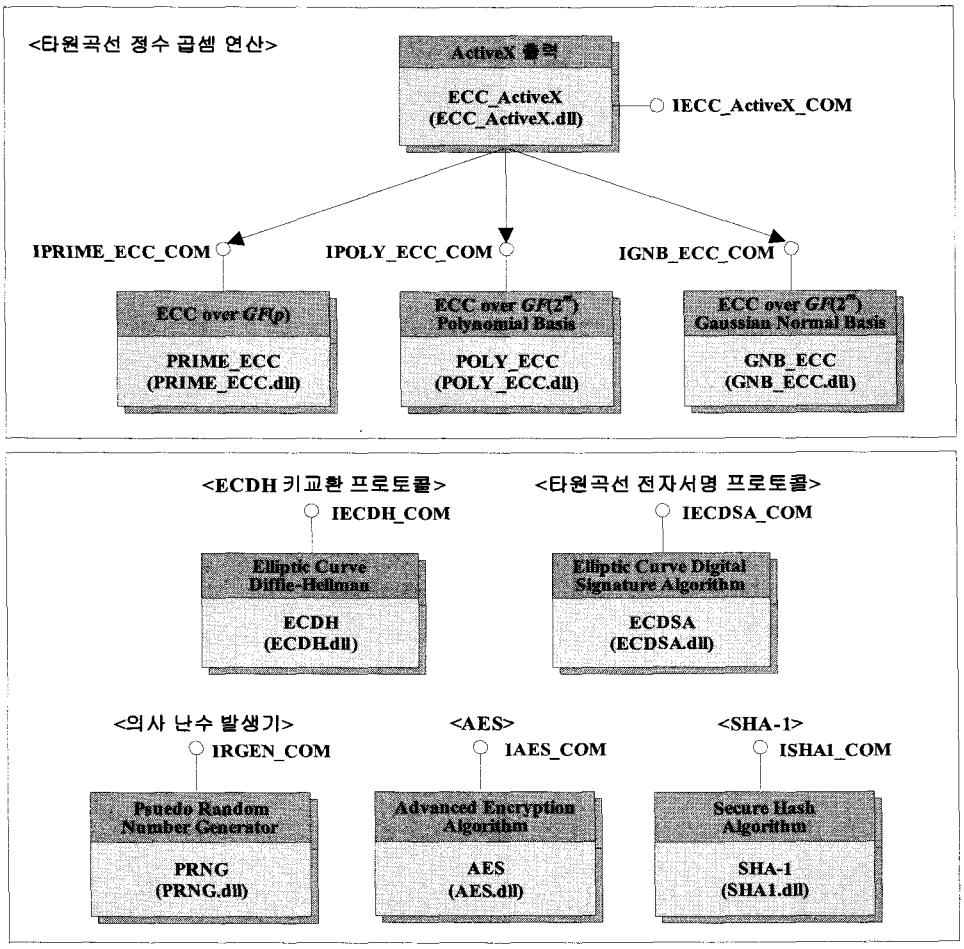
알고리즘 3에서 임의의 k를 선택하기 위해 PRNG를 사용하며, PRNG에서는 AES를 사용한다. 또한 서명에 사용될 메시지 다이제스트를 생성하기 위해 SHA-1을 사용한다.

[알고리즘 3] : 서명 생성 ECDSA[5]

1. Choose a random number k : $1 \leq k \leq n-1$.
2. Compute $kP = (x_1, y_1)$ and $r = x_1 \bmod n$.
   If $r = 0$ then go to step 1.
3. Compute $k^{-1} \bmod n$.
4. Compute $e = H(M)$.
5. Compute $s = k^{-1}(e+dr) \bmod n$.
   If $s = 0$ then go to step 1.
6. Return (r, s).

[알고리즘 4] : 서명 검증 ECDSA[5]

1. Verify that r, s are integers in the interval $[1, n-1]$.
2. Compute $e = H(M)$.
3. Compute $w = s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $X = u_1B + u_2Q$. If $X = \boldsymbol{O}$ then reject the signature.
   Otherwise compute $v = x_1 \bmod n$ where $X = (x_1, y_1)$
6. If $v = r$ then return ("Accept the signature") else return ("Reject the signature").



<그림 1> 타원곡선 암호시스템을 위한 암호 모듈 구성도

4. 암호 모듈 구현

본 절에서는 본 논문에서 구현한 암호 모듈의 전체적인 구성, 인터페이스 정의, 그리고 자료구조에 대해 살펴본다.

4.1 암호 모듈 구성 및 기능

서론 부분에서 언급했듯이 보안 프로토콜은 그 기능에 따라 서로 다른 암호알고리즘을 요구하기 때문에 암호 모듈을 하나로 만드는 것은 바람직하지 않다. 이런 문제를 해결하기 위하여 본 논문에서는 그림 1과 같이 알고리즘들을 각각의 모듈단위로 구성하였다. 암호 모듈은 총 9개로 구성되며,

각각의 모듈은 IUnknown 인터페이스와 자기 자신의 고유한 인터페이스를 가진다.

ECC_ActiveX는 타원곡선 정수 곱셈 모듈 3개(PRIME_ECC, POLY_ECC, GNB_ECC)를 통합 구현하였다. 따라서 IECC_ActiveX_COM 인터페이스를 통해 3개 모듈의 인터페이스에 모두 접근할 수 있다. ECC_ActiveX를 제외한 나머지 모듈들은 각각의 독립모듈로 사용할 수 있으며, 각 모듈에 대한 기능을 표 1에 요약 하였다.

4.2 인터페이스 구성

모든 인터페이스는 IDL(Interface Definition Language)를 사용하여 작성하였으며, 컴파일은 마이크로소프트 Visual C++ 6.0의 IDL 컴파일러를 사용하였다. 각 모듈의 인터페이스 구성은 다음과 같다.

<표 1> 각 암호모듈의 기능

모듈 이름	기 능
ECC_ActiveX	- $GF(p)$, $GF(2^m)$상의 ECC 연산 모듈 통합 - ActiveX GUI 지원 - 이벤트 처리 - 각 모듈 제어
PRIME_ECC	$GF(p)$상의 타원곡선 정수 곱셈 수행
POLY_ECC	$GF(2^m)$상의 다항식 기저를 이용한 타원곡선 정수 곱셈 수행
GNB_ECC	$GF(2^m)$상의 가우시안 정규기저를 이용한 타원곡선 정수 곱셈 수행
PRNG	난수 생성
AES	128비트 블록 암호화 및 복호화 수행
SHA-1	메시지를 해싱하여 160비트 메시지 다이제스트 생성
ECDH	타원곡선 Diffie-Hellman 키 교환 프로토콜 수행
ECDSA	타원곡선 전자 서명 알고리즘 수행

■ $GF(p)$상의 타원곡선 정수 곱셈 모듈

```
interface IPRIME_ECC_COM : IDispatch
{
  [id(1), helpstring("method ECC_Multiplication")]
  HRESULT ECC_Multiplication([in] int In_M,
              [in] VARIANT vEC_Param,
              [out, retval] VARIANT* vkP);
}
```

- ECC_Multiplication 메서드 : In_M(연산에 사용되는 비트 수)와 vEC_Param(타원곡선 매개변수, 점 P, 정수 k, 소수)를 사용하여 연산을 수행하며 vkP(kP)를 리턴한다.

■ $GF(2^m)$상의 다항식 기저를 이용한 타원곡선 정수 곱셈 모듈

```
interface IPOLY_ECC_COM : IDispatch
{
  [id(1), helpstring("method ECC_Multiplication")]
  HRESULT ECC_Multiplication([in] int In_M,
              [in] VARIANT vEC_Param,
              [out, retval] VARIANT *vkP);
}
```

- ECC_Multiplication 메서드 : In_M(연산에 사용되는 비트 수)와 vEC_Param(타원곡선 매개변수, 점 P, 정수 k, 기약다항식)을 사용하여 연산을 수행하며 vkP(kP)를 리턴한다.

■ $GF(2^m)$상의 GNB를 이용한 타원곡선 정수 곱셈 모듈

```
interface IGNB_ECC_COM : IDispatch
{
  [id(1), helpstring("method ECC_Multiplication")]
  HRESULT ECC_Multiplication([in] int In_M,
              [in] int In_TYPE,
              [in] VARIANT vEC_Param,
              [out, retval] VARIANT* vkP);
}
```

- ECC_Multiplication 메서드 : In_M(연산에 사용되는 비트 수), In_TYPE(GNB 타입), vEC_Param(타원곡선 매개변수, 점 P, 정수 k)을 사용하여 연산을 수행하며 vkP(kP)를 리턴한다.

■ PRNG 모듈

```
interface IPRNG_COM : IDispatch
{
  [id(1), helpstring("method PRNG")]
  HRESULT PRNG(
              [in] VARIANT vPRNG_Param,
              [out, retval] VARIANT* vRN);
}
```

- PRNG 메서드 : vPRNG_Param(현재 시간 DT_i, 초기값 V_i, 키값 K_1, K_2)를 사용하여 연산을 수행하며 vRN(난수 R_i)를 리턴한다.

■ AES 모듈

```
interface IAES_COM : IDispatch
{
  [id(1), helpstring("method AES")]
  HRESULT AES([in] int OP,
        [in] VARIANT vAES_Param,
        [out, retval] VARIANT* vCipherText);
}
```

- AES 메서드 : OP(암호화 또는 복호화 선택), AES_Param(메시지 M, 키 K)를 사용하여 암호화 또는 복호화를 수행하며 vCipherText(암호문)을 리턴한다.

■ SHA-1 모듈

```
interface ISHA1_COM : IDispatch
{
  [id(1), helpstring("method SHA1")]
  HRESULT SHA1(
            [in] VARIANT vSHA1_Param,
            [out, retval] VARIANT* vMD);
}
```

- SHA1 메서드 : vSHA1_Param(메시지 M)을 사용하여 연산을 수행하며 vMD(메시지 다이제스트)를 리턴한다.

■ ECDH 모듈

```
interface IECDH_COM : IDispatch
{
  [id(1), helpstring("method ECDH Protocol")]
  HRESULT ECDH([in] int In_Field,
            [in] VARIANT vECDH_Param,
            [out, retval] VARIANT* vkP);
}
```

- ECDH 메서드 : In_Field(유한체를 선택), vECDH_Param(타원곡선 매개변수, 공개키 P, 개인키 k)를 사용하여 연산을 수행하며 vkP(kP)를 리턴한다.

■ ECDSA 모듈

ECDSA 인터페이스는 두 개의 메서드를 가지며, ECDSA_Gen 메서드는 서명을 생성, ECDSA_Verify 메서드는 서명을 검증하는 기능을 수행한다.

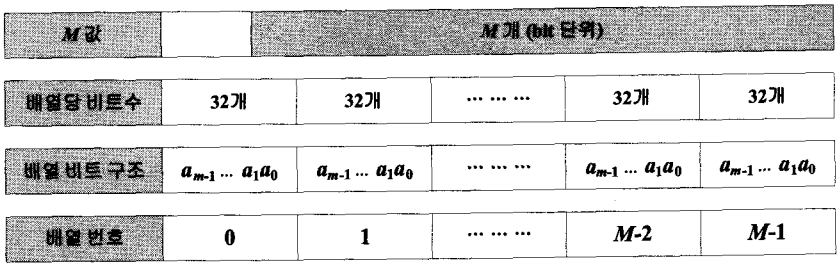
```
interface IECDSA_COM : IDispatch
{
  [id(1), helpstring("method ECDSA_Gen")]
  HRESULT ECDSA_Gen(
            [in] VARIANT vECDSA_Param,
            [out, retval] VARIANT* vSN);
  [id(2), helpstring("method ECDSA_Verify")]
  HRESULT ECDSA_Verify(
            [in] VARIANT vECDSA_Param,zzzzzzz
            [out] int Match);
}
```

- ECDSA_Gen 메서드 : vECDSA_Param(ECDSA 매개변수, 개인키 d, 메시지 m)을 사용하여 연산을 수행하며 vSN(서명 s, r)을 리턴한다.
- ECDSA_Verify 메서드 : vECDSA_Param (ECD-SA 매개변수, 공개키 Q, 메시지 m, 서명 s, r)을 사용하여 연산을 수행하며 Match(서명이 일치 1, 불일치 0)을 리턴한다.

ECC_ActiveX 인터페이스인 IECC_ActiveX_COM의 경우 ActiveX의 핵심적인 부분이 이미 ATL에 의해 구현이 되어 있으므로 자기 자신의 고유한 인터페이스는 메서드를 가지고 있지 않다. 또한 ECDSA_Verify 메서드를 제외한 나머지 메서드는 VARIANT형의 매개변수를 리턴한다. VARIANT는 BSTR, SAFEARRAY와 함께 자동화 데이터형의 하나이다. 만약 C언어에서만 사용하는 데이터형을 사용한다면 다른 언어에서 인식할 수 없는 문제가 발생한다. 따라서 본 논문에서는 자동화 데이터형인 VARIANT를 사용하여 다른 언어와 호환성을 유지 할 수 있도록 하였다. 뿐만 아니라 스크립트 언어에서는 변수의 포인터를 넘겨 줄 수 없기 때문에 마지막 매개변수는 [out, retval] 키워드를 주어 리턴 값으로 사용할 수 있도록 하였다.

4.3 큰 정수의 구현

그림 2는 본 논문에서 사용한 자료의 저장 형태를 나타낸다. 각 배열은 32비트로 구성되며, 각 배열의 비트는 big-endian형태를 가진다. 배열번호는 최상위 배열이 0이 되며, 하위 배열로 가면서 오름차순으로 번호가 부여된다. $GF(2^m)$상의 연산은 캐리가 발생하지 않기 때문에 배열의 모든 비트를 데이터 저장에 사용 가능하지만 $GF(p)$상의 연산



<그림 2> 자료의 저장형태

은 캐리가 발생할 수 있기 때문에 배열의 상위 16 비트는 캐리저장을 위해 비워둔다. 이 경우 배열의 16비트가 낭비될 수 있지만 캐리를 처리하는 방법이 간단하기 때문에 전체적인 코드의 크기를 감소시킬 수 있다.

5. 구현결과

본 논문의 4절에서 구현한 ECC_ActiveX 컨트롤을 이용하여 $GF(p)$, $GF(2^m)$상의 타원곡선 정수 곱셈 모듈의 기능을 검증하였으며, 본 논문에서 구현한 타원곡선 암호 모듈의 구현환경을 표 2에 요약하였다.

<표 2> 타원곡선 암호 모듈 구현환경

구현환경	세 부 사 항
운영체제	Microsoft Windows XP Professional
CPU	Intel Pentium 4, 2.8 GHz
Memory	512 MB
프로그래밍 환경	Microsoft Visual Studio 6.0
프로그래밍 언어	C++, IDL

본 논문에서 개발한 암호 모듈의 기능검증을 위해 $GF(p)$와 $GF(2^m)$상의 타원곡선 정수 곱셈 모듈을 사용하여 ActiveX 컨트롤을 구현하고, 기능검증 도구로 마이크로소프트 Visual Basic 6.0을 사용하였다. ECC_ActiveX 컨트롤은 $GF(p)$, $GF(2^m)$ 다항식 기저 그리고 $GF(2^m)$ 가우시안 정규기저상의 타원곡선 정수 곱셈 모듈을 포함하기 때문에 3개의 모듈에 모두 접근이 가능하다. 따라서 ECC_ActiveX 컨트롤에서 3가지의 연산을 수행할 수 있다.

본 논문에서 개발된 각 모듈들은 고유한 인터페이스를 가지고 이를 통해 필요한 암호 연산을 수행한다. 따라서 암호시스템 개발자는 필요한 암호 모듈만을 선택적으로 구성하여 타원곡선 기반의 암호시스템을 쉽게 구현할 수 있다. 본 논문에서 제안한 COM 소프트웨어 모듈과 C 라이브러리의 특성을 표 3에 요약하였다.

<표 3> COM 소프트웨어 모듈과 C 라이브러리와의 특성 비교

구분	COM 소프트웨어 모듈	C 라이브러리
구조	모듈	단일
컴포넌트	Interface & Black Box	Implementation & White Box
개발 방법	조립	스크래치
재사용 단위	컴포넌트	함수, 클래스
재사용 방법	컴포넌트 저장소	라이브러리
재사용 방식	실행코드 기반	소스코드 기반
생산성	높음	낮음
시스템 특성	개방화, 표준화	폐쇄적

표 3에 기술된 바와 같이 본 논문에서 제안된 COM 소프트웨어 모듈은 Black Box형태인 모듈화된 컴포넌트를 조립하는 방식의 개발 방법을 지향하기 때문에 생산성이 높다. 또한 개방화, 표준화된 시스템 특성을 가지므로 C 라이브러리를 이용한 방식에 비해 보다 쉽고 빠르게 소프트웨어를

개발할 수 있다[16]. C 라이브러리를 이용한 개발 방법의 재사용 수준은 소스코드 기반의 재사용으로 원시코드의 내부를 모두 이해해야하는 단점이 있을 뿐만 아니라 다른 언어에서 사용할 경우 해당언어에 맞게 코드를 변환해야하기 때문에 많은 시간과 노력을 필요로 한다. 반면에 제안된 COM 소프트웨어 모듈의 재사용 방식은 실행코드 기반의 재사용으로 컴포넌트 내부에 대한 구현은 모두 은폐시키고 외부로는 인터페이스를 제공하여 쉽게 컴포넌트를 재사용 할 수 있도록 해준다. 또한 언어 독립적이기 때문에 컴포넌트 표준을 따르는 언어에서 사용할 경우 코드 변환 없이 사용할 수 있다는 장점이 있다[12].

6. 결 론

본 논문에서는 타원곡선 암호시스템 응용을 위한 마이크로소프트 COM 소프트웨어 모듈을 구현하고 기능검증을 위해 타원곡선 정수 곱셈 모듈을 사용하여 ActiveX 컨트롤을 구현하였다. 개발된 COM 소프트웨어 모듈은 IEEE 1363의 모든 유한체 $GF(p)$와 $GF(2^m)$상의 타원곡선 키 교환 프로토콜 및 전자서명 기능을 지원한다. 본 논문에서 개발된 소프트웨어 모듈은 마이크로소프트 COM 인터페이스를 따르기 때문에 타원곡선 암호 시스템에 대한 깊은 지식 없이 타원곡선 암호 알고리즘 기반 보안 소프트웨어를 쉽게 개발할 수 있다. 다시 말해서, 본 논문에서 개발된 COM 소프트웨어 모듈을 이용하면, 마이크로소프트 Visual Basic과 같은 RAD(Rapid Application Development)툴을 이용하여 짧은 시간 내에 타원곡선 암호 시스템을 쉽게 구현 할 수 있다.

참 고 문 헌

[1] V. Miller, "Uses of elliptic curves in cryptography," *Advances in Cryptology Crypto'85, LNCS* 218, pp. 417-426, 1986.

[2] N. Koblitz, "Elliptic curve cryptosystems," *Math- ematics of Computation*, Vol. 48, pp. 203-209, 1987.

[3] J.-J. Quisquater and C. Couvreur, "Fast Deciph- erment Algorithm for RSA Public-key Crypto- system," *IEE Electronics Letters*, Vol. 18, No. 21, pp. 905-907, 1982.

[4] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logari- thms," *in Advances in Cryptology-Proc. of CR- YPTO'84*, pp. 10-18, 1985.

[5] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[6] I. F. Blake, G. Seroussi, and N. P. Smart, "Elliptic Curves in Cryptography," *Cambridge University Press*, 1999.

[7] IEEE P1363, Standard Specifications For Public Key Cryptography, 2000

[8] NIST, Recommended elliptic curves for federal government use, May. 1999.
http://csrc.nist.gov/encryption.

[9] J. Daemen, V.Rijmen, "Rijndael: The Advanced Encryption Standard," *Dr. Dobb's Journal*, Mar. 2001.

[10] NIST-Recommended Random Number Generator Based on ANSI X9.31 appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Jan. 2005.

[11] Federal Information Processing Standards Pub- lication 180-2, Aug. 2002.

[12] 전병선, Component Development with Visual C++ & ATL, 영진닷컴, 2005.

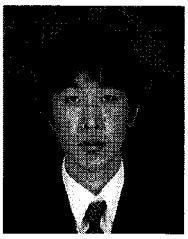
[13] M. Brown, D. Hankerson, J. López, and A. Menezes, "Software Implementation of the NIST Elliptic Curves Over Prime Fields," *Proc. of CT-RSA 2001, LNCS 2020*, pp. 250-265, 2001.

[14] D. Hankerson, J. Hernandez, and A.

Menezes, "Software Implementation of Elliptic Curve Cry- ptography Over Binary Fields," *Proc. of CHES 2000, LNCS 1965*, pp. 1-24, 2000.

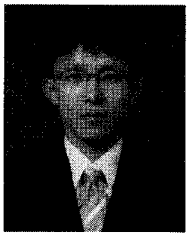
[15] J. Lopez, R. Dahab, "Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precompu- tation," *CHES'99, LNCS 1717*, pp. 316-327, 1999.

[16] Aoyama, M., "New Age of Software Develop- ment: How Component-Based Software Enginee- ring Changes the Way of Software Develop- ment?," *Proceedings of the 1998 International Workshop on CBSE*, 1998.



김 태 호 (Tae Ho Kim)

- 2006년 2월 대구대학교 컴퓨터·IT공학부, 학사
- 2006년 3월~현재 대구대학교 정보통신공학과, 석사과정
- 관심분야 : 암호 시스템, Embedded System, RFID/USN 보안

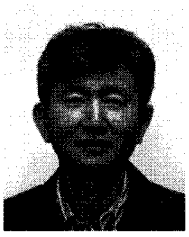


김 창 훈 (Chang Hoon Kim)

- 2001년 2월 대구대학교 컴퓨터정보공학부, 학사
- 2003년 2월 대구대학교 컴퓨터정보공학과, 석사
- 2006년 8월 대구대학교 컴퓨터정보공학과, 박사
- 2006년 9월~현재 대구대학교 정보통신공학과, BK21 연구교수
- 관심분야 : 암호 시스템, Embedded System, RFID/USN 보안

남 인 길 (In Gil Nam)

- 1978년 2월 경북대학교 전자공학과, 학사
- 1981년 8월 영남대학교 전자공학과, 석사
- 1992년 2월 경북대학교 전자공학과, 박사
- 1990년 3월~현재 대구대학교 컴퓨터·IT공학부, 교수
- 관심분야 : 데이터베이스, Embedded System, 보안 소프트웨어



홍 춘 표 (Chun Pyo Hong)

- 1978년 2월 경북대학교 전자공학과, 학사
- 1986년 12월 Georgia Institute of Technology ECE, 석사
- 1991년 12월 Georgia Institute of Technology ECE, 박사
- 1994년 9월~현재 대구대학교 정보통신공학부, 교수
- 관심분야 : DSP 하드웨어 및 소프트웨어, 컴퓨터 구조, VLSI 신호처리, Embedded System