ECSSL(Elliptic Curve SSL) 기반 DIT(Digital Investment Trust) 에이전트

정 은 희[†] · 이 병 관[††]

요 약

본 논문에서 제안하는 DIT(Digital Investment Trust) 에이전트는 전자상거래에 투자 신탁의 개념을 도입한 인터넷 기반의 은행업무 프로젝트로서, 소액지불뿐만 아니라, 계좌 생성 및 계좌 이체를 할 수 있으며, 또한 여기서 사용되는 인터넷 보안 프로토콜로는 기존의 SSL(Secure Socket Layer)보다 보안 기능을 강화한 ECSSL(Elliptic Curve SSL)이라는 프로토콜을 구축하여, 이를 기반으로 개발한 DIT 에이전트을 이용하여 고객의 정보와 자산을 제 3자로부터 보호하고자 한다.

DIT (Digital Investment Trust) Agent Based on ECSSL (Elliptic Curve SSL)

Eun-Hee Jeong[†] · Byung-Kwan Lee[††]

ABSTRACT

This paper proposes DIT (Digital Investment Trust) agent based on ECSSL (Elliptic Curve SSL). This DIT agent is a banking project using IT (Investment Trust) conception based on EC (Electronic Commerce) and can manage micro payment, account opening and account transferring. In addition, ECSSL (Elliptic Curve SSL) protocol is implemented which consists of much better encryption functions than existing SSL (Secure Socket Layer) protocol. Therefore, This DIT agent based on ECSSL protocol protects a customer's information and asset from third party.

키워드 : DIT, SSL, ECSSL, ECC, RSA, DES, SHA, 전자상거래, 디지털 투자 시스템

1. 서 론

인터넷을 활용한 사회 활동이 급증하면서, 전자상거래 분야에 대한 관심이 고조되고 있다. 컴퓨터와 네트워크를 기반으로 하는 소비자, 가상상점, 은행 등이 암호화된 전자 상거래 체계와 프로토콜을 통해 온라인으로 연결되며, 상거래에 필요한 여러 부분이 전자적으로 실현된다. 이러한 전자 상거래 환경에서는 정보의 보안성, 인증 체계, 전자 지불 수단 등의 기본 조건이 만족되어야 한다[1].

전자 지불 방식으로 디지털 캐쉬 시스템이 주로 사용되고 있는 이유는 현금 또는 신용카드에 의한 결제가 갖는 문제점인 소액지불, 개인정보 누출 등을 해결할 수 있기 때문이다. 디지털 캐쉬 시스템이 갖는 특징들은 첫째, 결제 비용의 절감이고 둘째, 안정성 향상, 지속성을 확보할 수 있으며, 셋째, 기존의 금융제도와의 호환성을 들 수 있다[2, 3].

본 논문에서 제안하는 DIT 에이전트는 이러한 디지털 캐쉬 시스템을 이용해, 소액지불 뿐만아니라, 계좌 생성 및 계좌 이체를 할 수 있으며, 투자 개념을 도입한 인터넷 기반 은행업무 프로젝트이다.

DIT 에이전트에선 현재 인터넷 보안으로 널리 이용되는 기존의 SSL(Secure Socket Layer)의 보안 기법에 공개키 보안을 강화한 ECC와 기존의 비밀키에 암호 기능을 강화시킨 ADES(Advanced DES)를 이용한 ECSSL(Elliptic Curve SSL)을 기반으로 한다.

본 논문에서 제안한 DIT 에이전트는 크게 세 가지 부분으로 나누어 볼 수 있는데, 첫째 디지털 투자 신탁 시스템 내에서 계좌 생성과 거래를 익명화하였고, 둘째 고객이 인터넷을 통해 안전하게 접근하여 화폐를 전송하는 것을 허용하였으며, 셋째 고객이 예금한 자금으로 가장 적합한 곳을 검색하여 투자를 하여 이윤을 남기는 투자 신탁이다.

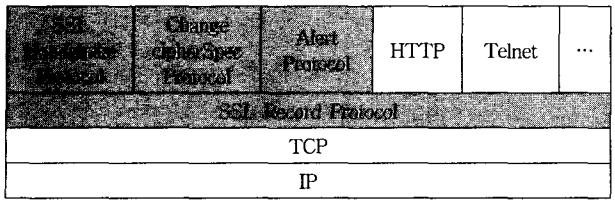
본 논문의 구성은 제 2장에서 관련연구에 대해 설명하고 제 3장에선 ECSSL의 설계에 대해서 설명하였다. 그리고 제 4장에서 DIT 에이전트의 설계에 대해서 설명하였고, 제 5장에서 시뮬레이션 결과에 대해 설명하였으며, 마지막으로 제 6장에서 결론 및 향후 연구 방향을 제시하였다.

† 준 회 원 : 관동대학교 대학원 전자계산공학과
†† 종신회원 : 관동대학교 컴퓨터공학과 교수
논문접수 : 2002년 7월 25일, 심사완료 : 2002년 10월 28일

2. 관련 연구

2.1 SSL

SSL(Secure Socket Layer)은 넷스케이프사에서 처음 제안되었으며, 현재 웹 보안의 대명사로 잘 알려져 있는 보안 프로토콜이며, TCP/IP 계층과 어플리케이션 계층(HTTP, TELNET, FTP등) 사이에 위치하여 데이터를 송수신하는 두 컴퓨터 사이의 보안 서비스를 제공한다[4, 5].



(그림 2-1) SSL 프로토콜의 구조

2.1.1 SSL Handshake Protocol

SSL Handshake Protocol은 한 세션동안 이용되는 암호 매개변수를 생성하고, 한 세션에서 사용되는 비밀 정보를 공유하기 위해 이용된다.

클라이언트와 서버가 처음으로 통신을 시작할때 서로 프로토콜 버전을 확인하고, 암호 알고리즘을 선택하고, 서로 인증하고, 공유할 비밀 정보를 생성하기 위해 공개키 암호 기법을 사용한다.

2.1.2 SSL Record Protocol

SSL Record Protocol은 상위 응용 계층으로부터 임의의 크기의 데이터를 전달받은 후, 전송할 메시지를 SSL에서 처리할 수 있는 데이터 블록 크기만큼 단편화하고, 압축을 한 후 암호화를 하여 TCP 계층으로 전달한다.

데이터를 수신한 경우에는 복호화하고, MAC(Message Authentication Code)에 대해 조사한 후 다시 압축 해제를 한 후, 단편화된 데이터 조각을 모아 어플리케이션 계층으로 전달한다.

2.2 RSA

1976년 Diffie와 Hellman이 공개키 암호 방식의 개념을 발표한 후, 1978년 MIT의 Rivest, Shamir, Adleman이 처음 RSA라고 하는 유망한 공개키 암호 방식을 제안하였다. 이들은 합성수의 소인수 분해의 어려움을 이용하여 RSA 암호 방식을 실현하였다. 가입자는 백 자리 크기 이상의 두 개의 소수 p, q를 선택하여 $n = p \cdot q$를 계산한다. 소수 p, q의 크기가 클수록 암호화시키는 비중은 커지게 되며 보안의 강도는 비례적으로 강하게 된다. 이때 p와 q를 알고있는 사람은 n을 계산하기 쉽지만 n만 알고 있는 사람은 n으로부터 p와 q를 찾는 소인수 분해는 어렵다.

p와 q를 선택하여 n을 계산한 다음 Euler 함수 값 $\Phi(n) = (p-1)(q-1)$와 서로소인 암호화 키인 $K(e)$를 선택한다. 다시 유클리드 알고리즘을 이용하여 다음 식을 만족하는 복호화 키인 $K(d)$를 계산한다.

$$K(e) \cdot K(d) \equiv 1 \bmod \Phi(n)$$

$K(e)$와 n은 공개 목록에 등록하여 공개하고 $K(d)$와 n는 비밀리에 보관한다. RSA 암호 방식의 구성 절차와 암·복호화 과정은 (그림 2-2)와 같다.

수신자 B에게 평문 M을 비밀리에 전달하려는 송신자 A는 공개목록에서 수신자 B의 공개 암호화 키 $K(e)_B$를 찾아, 암호문 $C \equiv M^{k(e)_B} \bmod n_B$를 계산하여 수신자 B에게 전송한다. 수신자 B는 송신자 A로부터 수신된 암호문 C를 자신이 비밀리에 보관하고 있는 복호화 키 $K(d)_B$로 평문 $M \equiv C^{K(d)_B} \bmod n_B$를 복원한다.

역으로 송신자 B가 수신자 A에게 평문을 비밀리에 전송하려면 수신자 A의 공개 암호화 키 $K(e)_A$를 공개 목록에서 찾아 암호문 $C \equiv M^{K(e)_A} \bmod n_A$를 계산하여 수신자 A에게 전송한다. 수신자 A는 송신자 B로부터 수신한 암호문 C를 자신이 비밀리에 보관하고 있던 비밀 복호화 키 $K(d)_A$로 평문 $M \equiv C^{K(d)_A} \bmod n_A$을 복원한다.

송신자 A	p_A, q_A $n_A = p_A \cdot q_A$ $\Phi(n_A) = (p_A-1)(q_A-1)$ $\gcd(K(e)_A, \Phi(n_A)) = 1$ $K(e)_A \cdot K(d)_A \equiv 1 \bmod \Phi(n_A)$ $C \equiv M^{K(e)_B} \bmod n_B$
공개정보 $K(e)_A, n_A, K(e)_B, n_B$	C
수신자 B	p_B, q_B $n_B = p_B \cdot q_B$ $\Phi(n_B) = (p_B-1)(q_B-1)$ $\gcd(K(e)_B, \Phi(n_B)) = 1$ $K(e)_B \cdot K(d)_B \equiv 1 \bmod \Phi(n_B)$ $M \equiv C^{K(d)_B} \bmod n_B$

(그림 2-2) RSA 공개키 암호 방식

2.3 DES

ANSI에 의해 Data Encryption Algorithm(DEA)와 ISO에 의해 DEA-1로 알려진 DES는 세계적인 표준으로 사용

된 64비트 블록 암호 알고리즘이다.

DES의 구조는 크게 데이터 암호화부와 키 생성부로 구성되어있다. 먼저 키 생성부에서 생성된 48비트의 16개 라운트 키(round key)는 데이터 암호화부의 각 라운드로 들어가 평문 블록과 함께 치환, 대치 그리고 키 스케쥴 등을 이용하여 암호문을 만들고, 복호화는 암호화의 역순이다[3].

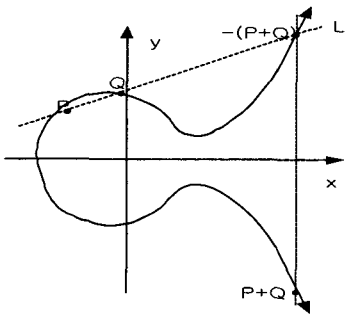
2.4 ECC

ECC(Elliptic Curve Cryptography)는 1985년 N.Kobitz와 V.Miller에의해 제안되었다. RSA와 이산 대수처럼 일반적으로 사용되는 공개키 보안 시스템과 비교해 볼때, ECC는 좀 더 작은 키 크기, 그에따라 더 나은 보안 그리고 더 작은 하드웨어 구현에 이롭다.

ECC는 이미 알려진 공개키 시스템보다 비트당 더 높은 보안을 제공하기 때문에 많은 관심의 초점이 되는 분야이다. 다른 암호화 기법에 비해 좀더 작은 키 크기로 응용분야에 적용했을때 키 크기에서의 부담을 줄일 수 있고, 시간을 절약할 수도 있다는 장점을 갖는다.

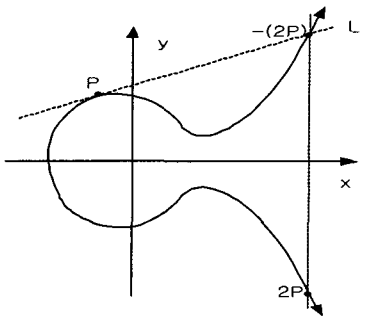
이러한 타원곡선은 유한체 상에 정의된 타원곡선에 대하여 타원곡선군은 3차 방정식을 만족하는 순서쌍들과 무한점을 포함한 집합을 말한다. 두 점의 덧셈군을 계산하는 방식으로 두 가지이다. 동일한 점 즉, $P = Q$의 경우와 점 $P \neq Q$인 경우이다.

(그림 2-3)은 타원곡선 위의 두 점 P와 Q가 서로 다른 경우로 이 두 점의 합 $P + Q$를 구하는 그래프이다. 먼저 서로 다른 점 P와 Q를 지나는 직선 L을 긋는다. 그러면 L과 원래의 타원곡선과 만나는 점이 반드시 존재하게 되고 그 점을 바로 $-(P + Q)$로 정의한다. 그리고 x축과 대칭 이동되는 점이 암호화된 점인 $P + Q$가 된다.



(그림 2-3) P ≠ Q 경우의 ECC 그래프

(그림 2-4)는 같은 점 P를 더한 경우로 $P + P = 2P$인 점을 구하는 방식이다. $P \neq Q$와 같이 점 P에서 $2P$를 덧셈 연산하여 구하고, 접선 L을 긋고 만나는 점을 $-(2P)$로 정하고, x축을 기준으로 대칭 이동하여 암호화된 점 $2P$를 얻는다. 이렇게 정의된 연산에 의해서 유한군이 되고 그 군의 원소의 개수 즉, 좌표 점의 개수는 타원곡선이 정의된 유한체의 원소의 개수가 된다.



(그림 2-4) P=Q 경우의 ECC 그래프

타원곡선 E는 $y^2 (\mathrm{mod}\ p) = x^3 + bx + c (\mathrm{mod}\ p)$ $(b, c \in Z_p)$로 표현하며 무한원점(0)으로 구성되어 있다. p개의 원소를 갖는 유한체의 점들로 구성된 타원곡선 $E(Z_p)$는 두 개의 점으로 구성되어 있다. 즉 $P(x_1, y_1)$, $Q(x_2, y_2)$로 되어 있으며 $\{(x, y) \in (Z_P, Z_Q) \mid y^2 = x^3 + bx + c\} \cup \{0\}$로 표현한다.

여기서 $E(Z_p)$의 원소인 P, Q가 주어졌을때 타원곡선의 암·복호화는 $Q = aP$일때 a를 계산해 내는것이 중요한 문제이다. 타원곡선 암호화 시스템은 타원곡선 상에서 점 P를 a번 더하는 계산이 주를 이룬다. 즉, $Q = aP$를 구하는 더하기 연산은 modular 덧셈을 통해 이루어진다.

또 타원곡선 암호화가 RSA 암호화 시스템보다 능률적인 이유는 소수 p의 값이 작아도 안전성이 뛰어나다는 것이다. 즉, 타원곡선 암호시스템의 안전도는 타원곡선 이산대수문제에 의존하고 있으며, 효율성은 aP를 얼마나 빠르게 계산해 낼 수 있는가에 달려있다.

타원곡선 알고리즘은 최초의 점에서 소수의 횟수만큼 덧셈연산에 의한 이동으로 마지막 최종의 점을 암호화 키로 갖기 때문에 암호화 키를 통한 원래의 점을 역추적하기가 상당히 어렵다. 즉 $Q = aP$에서 암호화된 결과의 점 Q의 값이 (0,502)라고 하면 정수형태의 비밀키 a와 초기의 점 P를 예측하기는 불가능하다는 것이다.

이러한 ECC 암호화 방법은 유한체 k위에서 정의된 타원곡선 E위의 점들이 덧셈군의 형태를 이룬다. 이 덧셈군의 더하기 연산은 기초 체(field) k에서의 산술 연산 몇 개를 포함하며, 하드웨어와 소프트웨어로 구현하기가 쉽다. 또한 이 덧셈군에서 이산대수를 사용하는 암호시스템은 유한체의 곱셈군에 기초한 시스템에 비해 두 가지 장점을 가지고 있다.

첫째, 이 군에서의 이산대수 문제는 매우 어렵다. 특히 같은 크기인 k 유한체에서의 이산대수 문제보다 더 어렵다. 다시 말하면, ECC는 작은 키 길이를 가지고 현존하는 공개키 암호화 방법의 안전도를 보장받을 수 있다. 짧은 키 길이는 몇몇의 응용에서 중요 요소인 작은 대역폭과 적은 메모리로 ECC를 사용할 수 있음을 의미한다. ECC는 메모리와 처리

능력이 제한된 IC카드 또는 휴대용 전화기의 시스템 설계에 유용하게 이용할 수 있다.

둘째, 타원곡선을 사용함에 따른 또 다른 이점은 비록 모든 사용자들이 같은 기초에 k를 사용하더라도 각 사용자가 다른 곡선 E를 선택할 수 있다. 이는 $P+Q$의 점이 그래프의 선상에 무수히 많은 벡터의 점들을 가지고 있으며 수신측 또는 송신측에서 암호화키를 선택하여 점을 덧셈연산 하는 경우가 넓다는 것이다.

결과적으로 모든 사용자들은 같은 하드웨어로 체(field) 연산을 실행하고, 요구되는 안전도를 위해 주기적으로 곡선 E를 변환시킬 수 있다.

2.5 디지털 서명

디지털 서명의 개념은 Diffie와 Hellman에 의해 처음으로 시작되었으며, 디지털 서명은 비트 스트링으로 표현되는 디지털 신호이며, 수신자 또는 제 3자에 의한 서명의 위조와 송신자에 의한 메시지 전송 행위의 부인과 같은 두 가지 부정행위를 방지하기 위해 사용한다.

2.5.1 RSA 디지털 서명

RSA 디지털 서명은 서명 시스템의 안전성을 위해 큰 수의 곱셈이나 거듭제곱 등의 계산량이 많은 연산들이 요구되며 특히 메시지 양이 클 경우 서명 생성 및 검증시 계산량이 급격히 증가하여 많은 오버헤드가 발생한다. 이런 문제점을 극복하기 위한 방법으로 해쉬 함수를 이용하여 원래의 메시지를 압축한 후, 압축한 메시지에 대한 서명만을 계산하여 전송하는 방식을 채택한다.

RSA 디지털 서명 과정을 살펴보면 다음과 같다.

우선, 송신자는 메시지 m을 안전한 해쉬 함수 h를 이용하여 해쉬값 h(m)을 구한 후에, 자신의 비밀키 d를 이용하여 서명을 생성한다.

$$s = (h(m))^d \bmod n$$

송신자는 메시지 m과 디지털 서명 s를 수신자에게 전송한다. 수신자는 메시지 m의 해쉬값을 h(m)을 계산한 후, 공개키 리스트에서 송신자의 공개키 e를 선택한 후 아래의 식이 성립하는지를 확인한다. 만약 아래의 식이 만족한다면, s를 메시지 m의 서명으로 받아들인다.

$$h(m) = s^e \bmod n$$

2.5.2 DSA(Digital Signature Algorithm)

1991년에 미국 국립 표준·기술 연구소(NIST)가 공포한 디지털 서명 방법으로, 미국 연방 표준(FIPS 186)으로 발전되었고, 현재는 디지털 서명 표준(DSS : digital signature standards)이라고도 한다. DSA는 이산 대수를 계산하는 어려움에 기초를 두고 있으며, Schnorr와 ElGamal에 의해 제안된 디지털 서명 방식에 기반을 두고 있다.

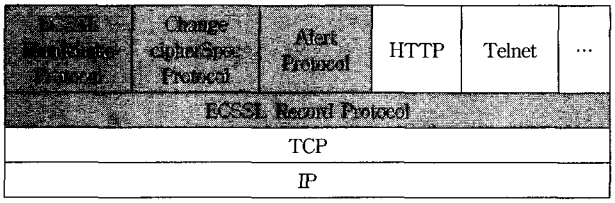
DSA는 인증만을 위한 것으로, DSA에선 서명 생성이 서명 검증보다 더 빠르며, 반면에 RSA에선 서명 검증이 서명 생성보다 더 빠르다.

3. ECSSL 설계

3.1 ECSSL

기존의 SSL(Secure Socket Layer)은 넷스케이프사에서 처음 제안되었으며, 현재 웹 보안의 대명사로 잘 알려져 있는 보안 프로토콜이다[3, 4].

본 논문에서 제안한 ECSSL은 아래 (그림 3-1)처럼 TCP/IP 계층과 어플리케이션 계층(HTTP, TELNET, FTP등) 사이에 위치하여 데이터를 송수신하는 두 컴퓨터 사이의 보안 서비스를 제공한다.

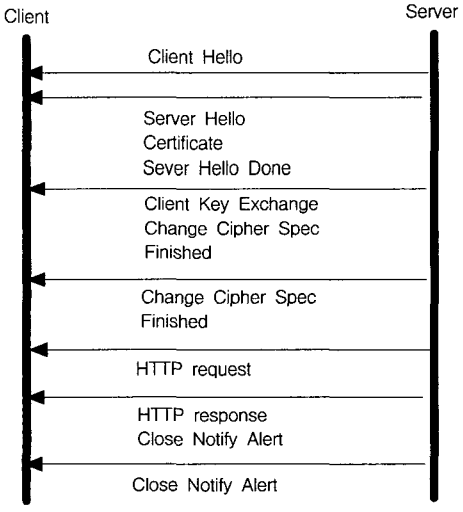


(그림 3-1) ECSSL 프로토콜의 구조

3.1.1 ECSSL Handshake Protocol

ECSSL Handshake Protocol은 한 세션동안 이용되는 암호 매개변수를 생성하고, 한 세션에서 사용되는 비밀 정보를 공유하기 위해 이용된다.

클라이언트와 서버가 처음으로 통신을 시작할때 서로 프로토콜 버전을 확인하고, 암호 알고리즘을 선택하고, 서로 인증하고, 공유할 비밀 정보를 생성하기 위해 공개키 암호 기법을 사용한다.



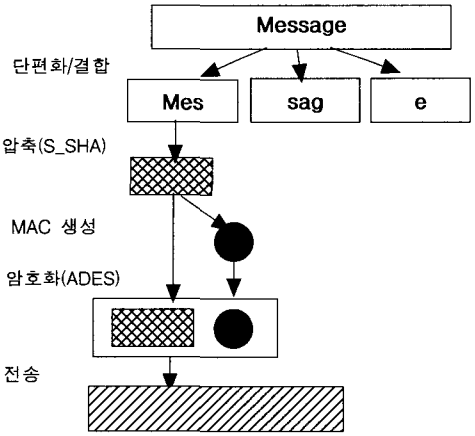
(그림 3-2) ECSSL Handshake Protocol 흐름도

3.1.2 ECSSL Record Protocol

ECSSL Record Protocol은 상위 응용 계층으로부터 임의의 크기의 데이터를 전달받은 후, 전송할 메시지를 ECSSL에서 처리할 수 있는 데이터 블록 크기만큼 단편화하고, 압축을 한 후 암호화를 하여 TCP 계층으로 전달한다.

데이터를 수신한 경우에는 복호화하고, MAC(Message Authentication Code)에 대해 조사한 후 다시 압축 해제를 한 후, 단편화된 데이터 조각을 모아 어플리케이션 계층으로 전달한다.

본 논문에서 구축한 ECSSL은 SHA-1을 변형한 S_SHA를 사용하여 메시지를 압축하였으며, 암호화 기법을 강화한 ADES를 사용하고 공개키 보안을 강화한 ECC를 사용하였다.



(그림 3-3) ECSSL Record Protocol 흐름도

3.2 ADES(Advanced Data Encryption Standard)

```
Ades(){
    static int c0[28], d0[28], c1[28], d1[28] ;
    int i, seed ;
        permu_ip (m, ip_box, L0, R0) ;
        permu_key (pc1_box, key1, key2) ;
        splite_key (key2, c0, d0) ;
        for(i = 0 ; i <= 15 ; i++) {
            shift_key (c0, d0, c1, d1, shift[i]) ;
            merge (c1, d1, k1, pc2_box) ;
        }
        pe_box = random (1, 32, seed) ;
        expand_r (R0, pe_box, k1, e_r) ;
        permu_p (p, s_box(e_r), f) ;
        round (L0, R0, L1, R1, f) ;
}
```

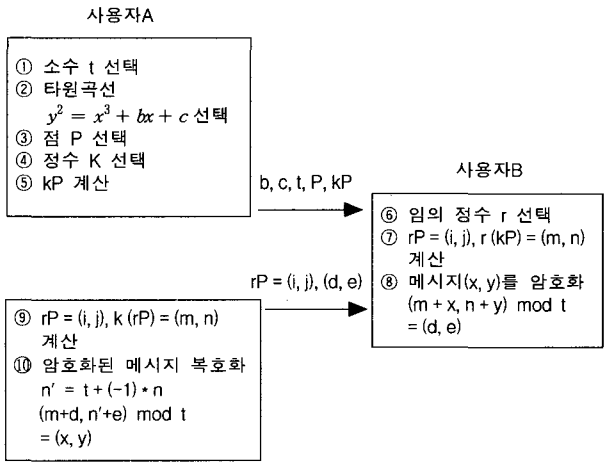
(그림 3-4) ADES 알고리즘

ADES는 기존의 암호화 알고리즘인 DES를 변형시킨 것으로, 암호화 하고자하는 메시지와 암호화시키는 키의 값을 각각 결합하는 과정에서 기존의 DES는 하나의 고정된 PE (Permutation) 박스를 기본 16라운드 순환할때마다 같은 PE 박스를 사용하였는데, ADES는 각 라운드마다 PE 박스를 각각 다르게 설정하여 비공개로 함으로써, 도청자에 의해 대칭키가 도청 당하더라도 PE 박스가 공개되어 있지 않으므로 전체의 암호문 혹은 복호문을 알 수 없기때문에 보다 강력한 암호화 알고리즘이다. 또한 기존의 DES와 본 논문에서 제안한 ADES의 실행 시간을 비교해보면, PE 박스를 각 라운드마다 다르게 설정하더라도 차이가 없다.

3.3 ECC(Elliptic Curve Cryptography)

본 절에서는 ECC 암호 알고리즘을 이용해 사용자 A, B가 메시지를 암호화하여 보내는 과정을 (그림 3-5)에서 보여준다. 여기서 메시지는 소수 t보다 작은 정수의 쌍이다. 랜덤한 정수 즉, 비밀키 k로 메시지를 덧셈 연산하여 사용자 B에게 전송하고 사용자 B는 자신의 비밀키 r로 메시지를 복호화한다.

ECC 알고리즘의 구현 과정은 (그림 3-5)와 같다.



(그림 3-5) ECC 암 · 복호화 과정

타원곡선 알고리즘의 암호화 · 복호화 과정은 다음과 같다.

[단계 1] 사용자A : 소수 t의 값을 1에서 m 사이에서 찾는다.

```
select_t ( ){
    int is_prime, i, sqrn ;
    unsigned int t ;
    while (1) {
        is_prime = 1 ; t = rand ( ) ;
        if ( t <= m ) {
            sqrn = (int) sqrt (t) ;
            for (i = 2 ; i <= sqrn ; i++)
            if (p % i == 0) is_prime = 0 ; }
        else
            continue ;
        if (is_prime == 1) return t ;
    }
```

(그림 3-6) 소수 알고리즘

[단계 2] 사용자 A : 타원곡선 상에서 $y^2 = x^3 + bx + c$ 인 수식을 선택하고 상수 b, c를 선택한다. 사용자 A는 타원곡선 상의 점 P를 찾는다.

[단계 3] 사용자 A : 정수 k를 선택한 후, kP를 계산한다.

① $P = Q$인 경우

타원곡선상의 점에서 같은 P점을 두 번 더한 값 $2P$ 즉, $P = Q$일때 의사코드는 (그림 3-7)과 같다.

```
Addition (P = Q)
PointEC (x1, y1, b, c) + PointEC (x1, y1, b, c)
        := Module (L, IdentityECQ, x3, y3),
           IdentityECQ = True ;
           L = (3x1^2 + b) / 2y1 ;
           x3 = L^2 - 2x1 ;
           y3 = L (x1 - x3) - y1 ;
           Return (PointEC (x3, y3, b, c))
```

(그림 3-7) 점 $P = Q$의 알고리즘

같은 점을 덧셈연산을 하는 경우 즉, 한 점 $P = Q$일때, $P = (x_1, y_1), 2P = 2Q = (x_3, y_3), x_3 = L^2 - 2x_1, y_3 = L(x_1 - x_3) - y_1$이므로 기울기 (L) = $\frac{3x_1^2 + b}{2y_1}$ 이다.

② $P \neq Q$인 경우

타원곡선상의 점 P에서 다른 점 Q를 더하는 의사코드는 (그림 3-8)과 같다.

```
Addition (P ≠ Q)
PointEC (x1, y1, b, c) + PointEC (x2, y2, b, c)
        := Module (L, IdentityECQ, x3, y3),
           IdentityECQ = False ;
           L = (y2 - y1) / (x2 - x1) ;
           x3 = L^2 - x1 - x2 ;
           y3 = -y1 + L (x1 - x3) ;
           Return (PointEC (x3, y3, b, c))
```

(그림 3-8) 점 $P \neq Q$의 알고리즘

$y^2 = x^3 + bx + c$ 타원방정식에서 $P \neq Q$일때의 계산방식은 다음과 같다.

$P = (x_1, y_1), Q = (x_2, y_2)$ $P + Q = (x_3, y_3)$이다.

$x_3 = L^2 - x_1 - x_2, y_3 = L(x_1 - x_3) - y_1$ 이므로,

기울기(L) = $\frac{y_2 - y_1}{x_2 - x_1}$ 이다.

[단계 4] 사용자 A : 사용자 B에게 b, c, t, P, kP의 값을 보낸다.

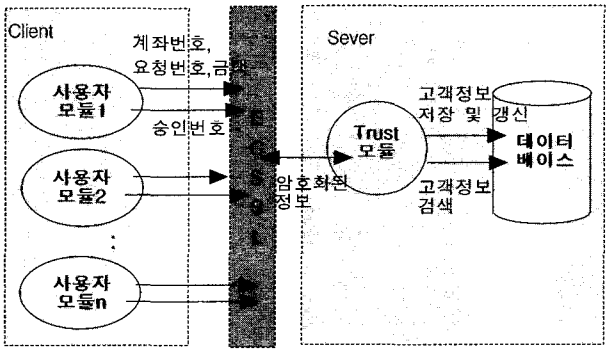
사용자 B : 임의의 정수 r을 비밀키로 선택한 후, rP와 $r(kP)$를 계산한다.

[단계 5] 사용자 B : $rP = (i, j)$, $r(kP) = (m, n)$를 이용해 메시지 (x, y)를 $d = m + x \bmod t$, $e = n + y \bmod t$을 계산하여 메시지를 암호화한 후 $rP = (i, j), (d, e)$를 사용자 A에게 보낸다.

[단계 6] 사용자 A : $rP = (i, j)$를 이용해 사용자 A는 $r(kP) = (m, n)$를 찾고, $n' = t + (-1) * n$을 이용해 $m + d = x \pmod t$, $n' + e = y \pmod t$을 풀어 메시지 (x, y)를 복호화시킨다.

4. DIT 에이전트 설계

DIT 에이전트는 크게 사용자 모듈, Trust 모듈, 데이터베이스로 구성되며, DIT 에이전트의 구성도를 대략 살펴보면 (그림 4-1)과 같다.



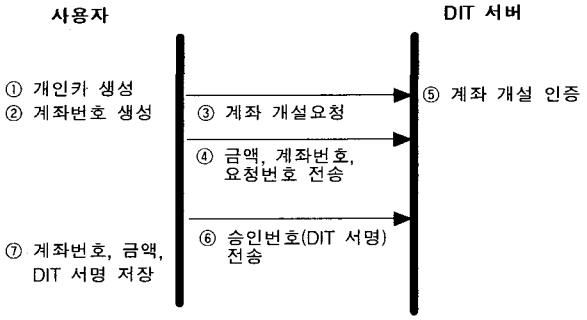
(그림 4-1) DIT 에이전트 구성도

4.1 사용자 모듈

사용자 모듈은 Trust 서비스에 접근하려는 일반적인 웹 브라우저와 관련된 작업을 하며, 고객번호를 생성하고 저장한다.

4.1.1 디지털 투자 신탁 계좌 생성

디지털 투자 신탁 계좌를 개설하는 것은 계좌에 현금을 입금시키는 것만큼 간단하다. 고객은 익명이므로 디지털 투자 신탁은 고객 정보가 필요하지 않으며, 계좌 번호는 고객의 사용자 모듈에 의해 생성되어, Trust 모듈에 의해 예금 기록과 함께 디지털 투자 신탁 데이터베이스에 저장된다.



(그림 4-2) 신탁 계좌 생성 흐름도

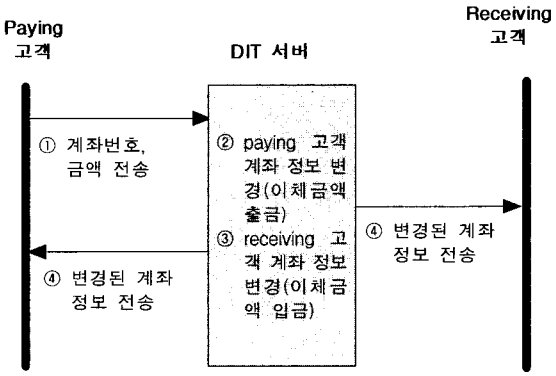
(그림 4-3)는 사용자 모듈이 Trust 모듈에 계좌 생성을 요청하는 알고리즘이다. 계좌 번호는 사용자 모듈에 의해 생성되며, 계좌번호와 금액에 대한 정보는 디지털 투자 신탁의 공개키로 암호화되어 전송된다.

```
User_module ( ) {
  generate_private_key (x) ;
  y = g^x mod p ;
  y1 = SHA_1 (y) ;
  call_deposit (y1, y, C) ;
  if (equal_ECC (y, s1)) save_user (y, C, s) ;
}
```

(그림 4-3) User Module 알고리즘

4.1.2 paying 고객 알고리즘

디지털 투자 신탁 시스템의 고객들끼리 계좌 이체를 하는 경우로, 누구든지 paying 고객 혹은 receiving 고객이 될 수 있다.



(그림 4-4) 고객간 계좌이체 흐름도

(그림 4-5)는 paying 고객에 대한 알고리즘으로 Trust 모듈에 계좌이체에 대한 정보를 전송하여 계좌이체를 한 후, Trust 모듈로부터 계좌 잔액에 대한 정보를 전송 받아 paying 고객의 컴퓨터에 계좌번호와 금액, 디지털 투자 신탁의 서명을 저장한다.

```
pay_module( ) {
   t = concatenate (y, C, y1, T) ;
   send_to_DIT (t) ;
   s = concatenate (C, y) ;
   s1 = sign_ECC (s) ; send_to_DIT (s, s1) ;
   s2 = sign_ECC (c) ; send_to_DIT (c, s2) ;
   if(equal_ECC (c2, s3)) save_to_user (y, C-T, s3) ;
}
```

(그림 4-5) paying 고객 알고리즘

4.2 Trust 모듈

Trust 모듈은 디지털 투자 신탁 시스템을 제어하기 위해 웹서버와 관련된 작업을 하며, Trust 번호를 생성하고 기록을 작성한다. 또한 익명 계좌에 대해 예금 혹은 인출을 관리하며, 디지털 투자 신탁 계좌간의 계좌 이체를 관리한다.

4.2.1 계좌 승인 알고리즘

(그림 4-6)은 고객이 요청한 계좌 개설을 승인하는 알고리즘으로 고객이 전송한 정보중 요청번호가 일치하는지를 검색한 후 요청번호가 일치하면, 디지털 투자 신탁의 서명을 한 후, 고객에게 전송한다. 디지털 투자 신탁의 전자 서명방법으로 ECC를 사용하였다.

```
Trust_module( ) {
   while (!EOF) {
      if (SHA_1 (y) == y1) {
         s1 = sign_ECC (y1) ; send_to_user (y, s1) ;
         save_to_DIT_db (y1, y, C) ; } }
}
```

(그림 4-6) Trust Module 알고리즘-계좌 승인 알고리즘

4.2.2 paying 고객 계좌 관리 알고리즘

(그림 4-7)은 paying 고객이 Trust 모듈에 paying 고객의 계좌 정보를 전송하여, paying 고객의 계좌에서 이체 금액을 뺀 남은 금액으로 디지털 투자 신탁 데이터베이스 정보를 갱신하는 알고리즘이다.

```
Trust_module1 ( ) {
   if (equal_ECC (s, s1)) {
      c = generate_number ( ) ;
      send_to_pay_user (c) ; send_to_receive_user (c) ;
   }
   if (equal_ECC (c, s2)) {
      serach_db (y) ; replace_db (y, C-T) ;
      c1 = concatenate (y, C-T) ;
      c2 = SHA_1 (c1) ; s3 = sign_ECC (c2) ;
      send_to_pay_user (c2, s3) ;
      save_to_DIT_db (c2, y, C-T) ;
   }
}
```

(그림 4-7) Trust Module 알고리즘-1

4.2.3 receiving 고객 계좌 관리 알고리즘

(그림 4-8)은 Trust 모듈이 receiving 고객의 정보를 전송 받아, receiving 고객의 계좌에 이체 금액을 더하여 디지털 투자 신탁 데이터베이스의 receiving 고객의 정보를 갱신하는 알고리즘이다.

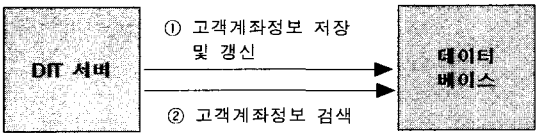
```
Trust_module2(){
   if (equal_ECC(c, s)){
        search_db(y1) ;
      R = add_db(y1, T) ;
      t1 = concatenate(y1, R) ;
      s = SHA_1(t1) ;
      s1 = sign_ECC(s) ;
      send_to_receive_user(t1, s1) ;
      save_to_DIT_db(s, y1, R) ; }
}
```

(그림 4-8) Trust Module 알고리즘-2

4.3 데이터베이스

암호화 알고리즘인 ADES를 이용해 Trust 계좌 정보를 저장하며, ADES에 이용된 대칭 암호키 s를 타원곡선 암호화 기법인 ECC를 이용해 암호화하여 저장한다.

데이터베이스 정보의 흐름은 다음과 같다.



(그림 4-9) 데이터베이스 흐름도

(그림 4-10)은 계좌 개설이 승인된 정보를 디지털 투자 신탁 데이터베이스에 저장하는 알고리즘으로 고객 정보인 금액, 계좌번호, 임의의 수에 대한 보안을 위해 ADES 암호화 기법을 사용하였으며, 대칭키 s를 ECC로 한번 더 암호화하여 보안을 강화시켰다.

```
DIT_db ( ){
   s = generate_ADES_key ( ) ;
   nonce = generate_nonce ( ) ;
   c = concatenate (C, y, nonce) ;
   v = ADES (c) ;
   s1 = ECC (s) ;
   save_to_DIT_database (i, v, s1) ;
}
```

(그림 4-10) 디지털 투자 신탁 데이터베이스 알고리즘

디지털 투자 신탁 데이터베이스 테이블 구조는 다음과 같다.

구 분	index_number	ADES_value	ECC_value
값	long	long	long

(그림 4-11) 데이터베이스 테이블 구조

5. 시뮬레이션

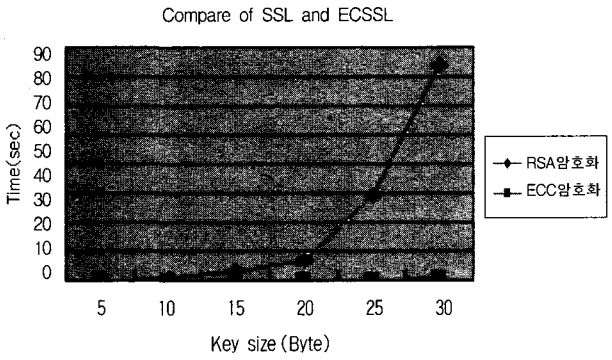
5.1 SSL과 ECSSL

기존의 SSL은 전자서명과 암호화 기법에 SHA-1, RSA와 DES를 사용하지만, 본 논문에서 구축한 ECSSL은 전자서명과 암호화 기법으로 S_SHA, ECC와 ADES를 사용하였다.

RSA 암호방식은 합성수의 소인수 분해 계산의 어려움을 이용하여 백 자리 이상의 두 개의 소수 p, q를 선택하여 공개키와 비밀키를 구성한 후 평문을 암, 복호화 하는 형식을 갖는다. 또 ECC 암호화 방식은 이산대수에서 사용하는 유한체의 곱셈군을 타원곡선군으로 대치한 암호시스템이다.

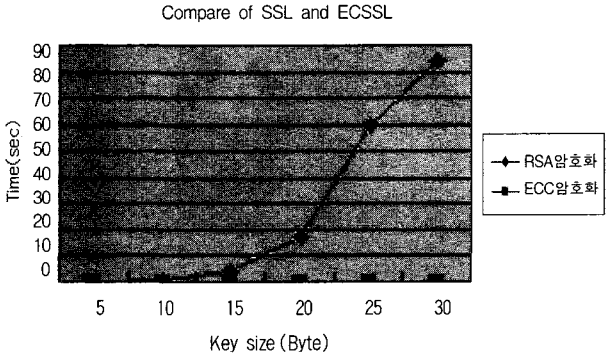
시뮬레이션은 O/S는 Winows XP, CPU는 Pentium Ⅳ 1.50GHz, RAM은 512MB에서 하였다.

(그림 5-1)은 RSA와 ECC의 암호화 수행 시간 비교한 것으로 키 크기가 5byte일 때에는 6 : 1, 10일 때에는 18.3 : 1, 15일 때에는 44.1 : 1, 20일 때에는 23.9 : 1, 25일 때에는 76.6 : 1, 30일 때에는 136.9 : 1로 키 크기가 커질수록 ECC가 상당히 빠른 비율을 보였다.



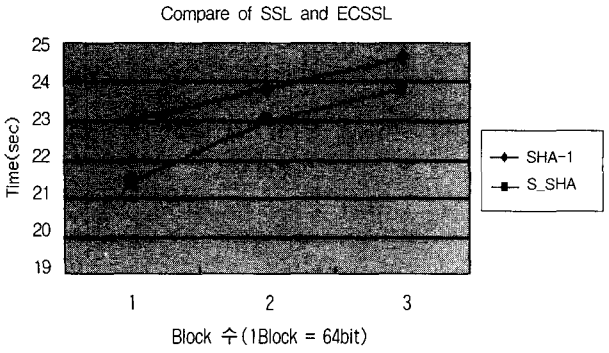
(그림 5-1) RSA와 ECC의 암호화 시간 비교

(그림 5-2)는 RSA와 ECC의 복호화 수행 시간을 비교한 것으로 키 크기가 5일 때에는 1.5 : 1, 10일 때에는 6.3 : 1, 15일 때에는 23.3 : 1, 20일 때에는 47.5 : 1, 25일 때에는 166.3 : 1, 30일 때에는 215.6 : 1의 비율을 나타냈다.



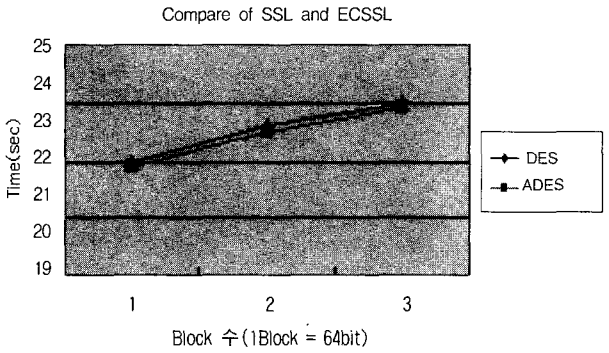
(그림 5-2) RSA와 ECC의 복호화 시간 비교

(그림 5-3)은 기존의 SHA-1과 S_SHA의 메시지 다이제스트 시간을 비교한 것으로 S_SHA의 메시지 다이제스트 수행 시간이 SHA-1보다 평균 1.06초가 빠르다.

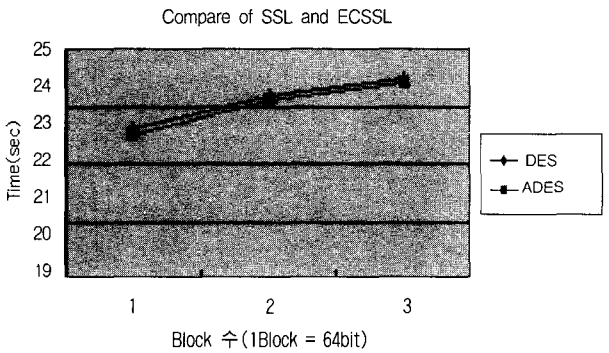


(그림 5-3) SHA-1과 S_SHA의 메시지 다이제스트 수행시간 비교

(그림 5-4)와 (그림 5-5)는 DES와 ADES의 메시지 암·복호화 시간을 메시지 길이 따라 비교한 것이다. 기존의 DES와 본 논문에서 제안한 ADES의 암·복호화 시간은 ADES가 빠르다고는 할 수 없다. 하지만, PE 박스를 다르게 함으로써 키 값이 노출되더라도 메시지를 보호할 수 있다. 또한, 위의 그림에서 볼 수 있듯이, 본 논문에서 제안한 ECSSL의 전체 수행시간이 기존의 SSL보다 월등히 빠르다는 것을 알 수 있으며, 특히, SSL보다 보안 기능이 뛰어나다고 볼 수 있다.



(그림 5-4) DES와 ADES의 암호화



(그림 5-5) DES와 ADES의 복호화

6. 결 론

본 논문에서 제시한 DIT 에이전트는 익명 계좌 생성 및 계좌 이체와 투자 개념이 도입된 인터넷 기반 은행 업무 시스템이다.

DIT 에이전트에선 현재 프로토콜로 널리 이용되는 기존의 SSL(Secure Socket Layer)의 보안 기법에 공개키 보안을 강화한 ECC와 기존의 비밀키에 암호 기능을 강화시킨 ADES(Advanced DES)를 사용한 ECSSL(Elliptic Curve SSL)을 사용하였으며, ECSSL를 사용하였을 경우에 기존의 SSL을 사용하였을 때보다 암호화 시간과 복호화 시간이 현저히 감소되는 알 수 있었다.

DIT 에이전트에선, 고객 모듈과 Trust 모듈 사이에서 데이터를 전송할때, 공개키 암호화 방식을 사용하였으며, 특히, 디지털 투자 신탁 데이터베이스에 고객 정보를 저장할때 비밀키 암호화 방식을 보안한 ADES를 사용하였고, 대칭키를 다시 한번 ECC로 암호화시킴으로써 고객 정보 보안을 강화시킴으로서 제 3자로부터 고객 정보를 보호하였다. 또한, DIT 에이전트는 계좌 생성 및 계좌 이체를 익명으로 처리하므로, 고객의 정보와 자산을 보호할 수 있는 차세대 금융 관리 시스템이라고 볼 수 있다.

향후 과제로는 본 논문에서 제안한 DIT 에이전트에 자산 관리 기능을 추가시키는 것이다.

참 고 문 헌

[1] 남기범, 이건명 "전자상거래 에이전트", 정보과학회논문지, 제18권 제5호, pp41-47, 2000.

[2] 이은철, "뉴밀레니엄 시대의 전자화폐(상)", 지식재산21, 통권 제59호, March, 2000.

[3] 이은철, "뉴밀레니엄 시대의 전자화폐(하)", 지식재산21, 통권 제60호, May, 2000년.

[4] 이만영외 5인 공저, "전자상거래 보안기술", 생능출판사.

[5] 이병관, "전자상거래 보안", 남두도서.

[6] 양승해, 이병관 "ECSET 설계를 위한 타원곡선 알고리즘", 정보처리학회 추계학술발표논문집, 제8권 제2호, pp.843-846, 2001.

[7] 정은희, 이병관 "DIT 시스템 설계를 위한 계좌관리", 정보처리학회 춘계학술발표논문집, 제9권 제1호, pp.951-954, 2002.

[8] Dabid Pointchebal and Jacques Stern, "Security Proofs for Signature Schemes," Advances in Cryptology-Proceedings of EUROCRYPT'96, pp.387-398.

[9] 배움닷컴, http://www.baeoom.com

[10] "PKCS#1 : RSA Encryption Standard,"

[11] SSL 3.0 Specification, http://www.netscape.com/eng/ssl3/draft302.txt.

[12] SSL 3.0 Implementation Assistance, http://www.netscape.com/eng/ssl3/traces/index.html.

정 은 희

e-mail : jeongnala@hanmail.net

1991년 강릉대학교 통계학과(이학사)

1998년 관동대학교 대학원 전자계산공학과 (공학석사)

2002년 관동대학교 대학원 전자계산공학과 박사과정 수료

1998년~현재 관동대학교 컴퓨터공학과 강사

2000년~현재 경동대학교 정보통신공학부 강사

관심분야 : 전자상거래, 네트워크 보안, 멀티미디어

이 병 관

e-mail : bklee@kwandong.ac.kr

1979년 부산대학교(공학사)

1986년 중앙대학교 대학원 전자계산학과 (이학석사)

1990년 중앙대학교 대학원 전자계산학과 (이학박사)

1988년~현재 관동대학교 컴퓨터공학과 교수

2000년~2002년 Visiting Professor of Saginaw Vally State University

관심분야 : 전자상거래, 컴퓨터 네트워크, 네트워크 보안