SSL VPN의 보안 강화와 WLCAPT 가상 서버 설계☆

Security-intensified SSL VPN and WLCAPT Virtual Server Design

정 은 희* 최 은 실** 이 병 관***
Eun-Hee Jeong Eun-Sil Choi Byung-Kwan Lee

요 약

본 논문은 크게 두 분분으로 나누어 기존의 SSL VPN의 보안 강화와 처리속도 향상을 위하여 타원곡선 암호화($GF(2^m)$) 기법과 가상 서버(virtual server)의 보안 강화을 위하여 기존의 LSNAT 대신에 WLCAPT(Weighted Least Connection Address Port Translation)를 제안하고 있다. 일반적으로, 기업이 VPN 구축을 위해 SSL 프로토콜을 사용할 때, SSL의 인증과 기밀성에 필요한 키 생성을 처리 속도와 보안상에 문제가 있는 RSA 알고리즘을 사용한다. 본 논문에서는 RSA 대신에 $GF(2^m)$상의 타원곡선을 사용하여 ECSPK(Elliptic Curve Based Shared Private Key) 알고리즘으로 공유 비밀키를 생성하여 처리 속도와 보안을 강화시키고, 또한 서버측에 가상 서버를 두고, 이 가상 서버에 WLCAPT기법을 적용하여 주소 변환 후, 본사의 실제 서버와 통신이 이루어지는 보안이 향상된 시스템 설계를 목적으로 한다.

Abstract

This paper consists of two parts. One is the ECC($GF(2^m)$) algorithm to improve the security strength and the processing time of SSL VPN and the other is the WLCAPT algorithm instead of LSNAT for the security strength of virtual server. In general when corporates use SSL protocol in order to build VPN, they use RSA algorithm with the problem of security and processing time about authentication and confidentiality. In this paper, a shared public key is generated with ECSPK which uses ECC($GF(2^m)$) algorithm to improve the security and processing time instead of RSA. In addition, WLCAPT algorithm proposed in this paper is applied to virtual server which resides in the server side and then after NAT translation, the actual server of headquarter is securely communicated with it.

Keyword : SSL, VPN, ECC, WLCAPT, ECSPK

1. 서 론

최근 기업의 근무 환경이 많은 정보를 외부의 사용자와 공유하게 됨에 따라 근로자들의 근무 위치가 사무실에 국한되지 않고 집이나 업무 현장으로까지 넓혀짐으로 인하여 재택 근무자 또는 이동하는 근무자(mobile worker)가 증가되었다. 이러한 변화로 기업은 네트워크를 확대한 지사(branch office)와의 인트라넷(intranet)이나, 협력사(business partner)를 위한 익스트라넷(extranet)이 필요하게 되었다. 이와 같은 필요성으로 기업은 본사와 지사, 본사와 협력사, 본사와 이동 근무자 사이에 보안 프로토콜을 적용한 VPN(Virtual Private Network)을 구축하게 되었다.

보안 프로토콜 중 최근 가장 범용적으로 사용되고 있는 SSL은 표준 웹 브라우저에 내장되어 있기 때문에 VPN 구축에 SSL을 사용할 경우 추가의 하드웨어나 소프트웨어 비용이 들지 않는 장점을 얻을 수 있게 된다[1,2].

본 논문에서는 기업이 VPN 구축을 위해 SSL 프로토콜을 사용하여 보안 통신을 할 때, SSL의 인증과 암호 알고리즘에 필요한 키 생성을 기존의 RSA 알고리즘 대신 $GF(2^m)$상의 타원곡선을

* 정 회 원 : 삼척대학교 전자상경제학 전임강사
jeh@samcheok.ac.kr(제 1저자)
** 종신회원 : 관동대학교 멀티미디어공학전공 교수
bklee@kwandong.ac.kr(공동저자)
*** 정 회 원 : 관동대학교 전자계산공학과 석사과정
tosil17@hanmail.net(공동저자)
☆ 본 연구는 삼척대학교 2004 신임교수 교내 학술 연구비 지원에 의해 수행되었습니다.

사용하고, master_secret 키 생성에 필요한 공유 비밀키 pre_master_secret 키 생성에 ECSPK(Elliptic Curve based Shared Private Key) 알고리즘을 이용하여 성능을 향상시키는 것을 목적으로 한다.

또한 대부분의 기업들이 한정된 IP 주소의 적절한 사용을 위해 사설 IP 주소공간을 사용한 네트워크를 구성하고 있다. 이러한 사설 IP 주소는 내부망의 클라이언트끼리는 통신이 가능하지만 외부망과는 직접 통신을 할 수 없다. 이러한 특징을 이용하여 본 논문에서는 본사의 실제 서버에 사설 IP를 부여하여 실제 서버가 외부망에 노출되지 않게 하고, 외부 공중망을 통한 협력사나 지사 클라이언트들과의 직접 통신은 공인 IP 주소를 가진 가상 서버가 담당하게 하여 실제 서버는 가상 서버와만 통신하게 된다.

본 논문에서 제안한 WLCAPT기법은 공인 IP 주소를 가진 가상 서버가 사설 IP 주소를 가진 실제 서버와 통신하기 위한 주소 변환 기법으로 기존의 LSNAT가 가진 단점을 보안하면서 보안성을 높이도록 하였다.

또한 서버측에 가상 서버를 두고, 이 가상 서버에 WLCAPT(Weighted Least Connection Address Port Transaction) 기법을 적용하여 주소 변환 후, 본사의 실제 서버와 통신이 이루어지는 보안이 향상된 시스템 설계를 목적으로 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로써 SSL과 VPN을 살펴보고, 3장에서는 SSL에 적용된 타원곡선과 공유 비밀키를 생성하는 ECSPK 알고리즘을 설명하고, 4장에서는 제안된 VPN 가상 서버 설계를 구체적으로 설명한다. 그리고 5장에서는 실험을 통해 제안된 방법을 평가하고 6장은 결론을 내린다.

2. 관련연구

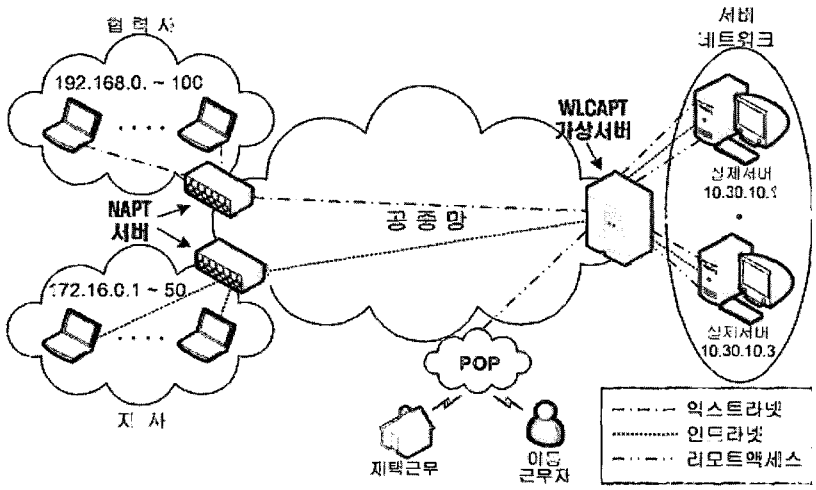
2.1 SSL(Secure Socket Layer)

SSL은 넷스케이프사에서 개발한 보안 프로토콜로서, TCP/IP 프로토콜의 어플리케이션층과 TCP층 사이에 안전한 통신을 위한 표준이다. 넷스케이프사에서 개발된 SSL은 대부분의 웹서버 및 웹 브라우저에 기본으로 내장되어 있으며, 상호 인증, 데이터 무결성, 기밀성 등을 제공함으로써 클라이언트와 서버 사이에 안전한 통신을 제공한다[3].

SSL 3.0이 TLS(Transport Layer Security) 1.0으로 개선된 후 IETF(Internet Engineering Task Force) 표준화가 진행 중이며, SSL은 인증기관의 사용에서 융통성 부여, 구축의 용이성 그리고 브라우저에서 자체적으로 지원되는 솔루션도 있어서 구축과 사용이 편리하다. 반면, 가상 상점에서 고객의 지불 정보를 볼 수 있어서 고객의 사생활 보장문제 및 고객의 정보를 오용할 소지가 있으나, 고객과 가상 상점 사이, 가상 상점과 지불 서버 사이는 SSL로 암호 및 인증 수행을 한다는 점으로 보면 가상 상점을 신뢰한다는 가정하에서는 안전한 시스템이라 할 수 있다[4].

2.2 VPN(Virtual Private Network)

VPN이란 기업이 네트워크를 구성할 때 전용임대 회선 대신에 공중망(public network)을 이용하여 사설망(private network)처럼 직접 운용 관리하는 것으로, 컴퓨터 시스템과 터널링 프로토콜을 사용하여 개별 기업의 목적에 맞게 구성한 데이터 네트워크이다. 공중망이란 전화망이나 인터넷처럼 상호 자유롭게 정보를 주고받는 사람들이 공동으로 사용하는 네트워크를 일컬으며, 공중망 사용자는 통신할 수 있는 사용자 중에서 일부와만 통신하나 잠재적으로는 모든 공중망 사용자와 통신이 가능하다. 반면에 사설망은 어떤 한 조직이 소유한 네트워크로 해당 조직에 관련된 사용자가 터널링 프로토콜을 사용하여 통신하기 때문에 사설망에서 주고받는 정보는 해당 조직과 관련된 사용자 외에는 아무도 볼 수 없는 특징을 가지고 있으며 IP 주소 체계 등의 네트워킹 관련



〈그림 1〉 SSL VPN 흐름도

규칙을 해당조직이 임의로 조정하여 결정할 수 있다. 또한 VPN은 공중망을 이용하기 때문에 네트워크 구축비용을 줄일 수 있으며 회선 관리도 공중망의 운용자가 맡게 되어 전체적인 비용 절감 및 인력 절감 효과를 가질 수 있다[5-7].

그림 1은 SSL 터널링 프로토콜 사용한 VPN의 다양한 형태를 보여주고 있다.

3. SSL VPN 성능향상

3.1 타원곡선 알고리즘($GF(2^m)$상의 ECC)

$GF(2^m)$상의 타원곡선을 정의하는 Weierstrass 방정식은 다음과 같다.

$$E : y^2 + xy = x^3 + ax^2 + b,\ a, b \in GF(2^m), b \neq 0$$

$GF(2^m)$상의 타원곡선 E 위의 방정식을 만족하는 $GF(2^m)$상의 모든 점 (x, y)와 무한 원점 0으로 구성되고, 이들에 대해 아래와 같은 덧셈 법칙을 만족한다[8,9].

(1) 0 + 0 = 0

(2) 모든 점 P∈E에 대해 0 + P = P + 0 = P를 만족한다.

(3) $P=(x_1,y_1)$ 일때, $-P=(x_1,x_1+y_1)$이고 $P+(-P)=0$다.

(4) $P=(x_1,y_1), Q=(x_2,y_2)$ 라고 할때, $P+Q=(x_3,y_3)$ 라고 하면 다음과 같다.

· $L = \begin{cases} (x_1+x_2)^{-1}.(y_1+y_2) & P \neq Q \\ x_1^{-1}.y_1 + x_1 & P = Q \end{cases}$

· $x_3 = L^2 + L + (x_1 + x_2) + \alpha$

· $y_3 = \begin{cases} L.(x_1+x_3)+x_3+y_1 & P \neq Q \\ x_1^2+(L+1).x_3 & P = Q \end{cases}$

$GF(2^m)$의 다항식 방식은 m차 감축 불가 다항식 f(x)에 의해 유일하게 결정되는 다음과 같은 형태의 집합이다.

$$B = \{x^{m-1}, x^{m-2}, \ldots, x, 1\}$$

감축 불가 다항식으로는 구현상의 효율을 위해 3개항으로 구성된 $f(x)=x^m+x^k+1$ (0 < k < m)형태의 삼항 다항식을 사용하는데, 본 논문에서는 감축 불가 다항식 $f(x)=x^4+x+1$을 사용하였다.

```
Equal_addition(x1, y1)
{
        L= x1 + y1 * Multiply_inverse(x1)
        x3 = L^2 + L + a;
        y3=x1^2+ (L + 1) * x3;
        return(x3, y3);
}
```

〈그림 2〉 점 P=Q의 알고리즘

그림 2는 타원곡선 상의 점에서 P점을 두 번 더한 값 2P 즉, P=Q일 때 알고리즘이다.

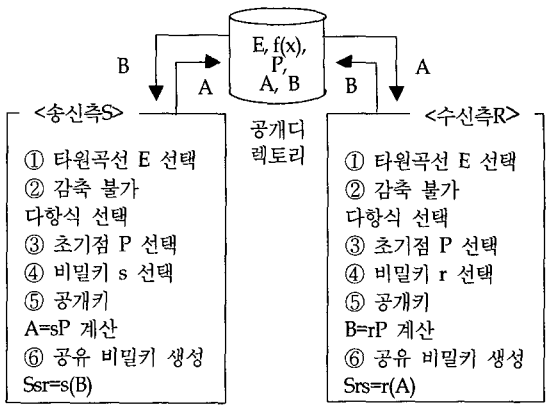
그림 3은 타원곡선 상의 점P에서 다른 점 Q를 더하는 알고리즘이다.

```
Not_Equal_addition(x1, y1, x2, y2)
{
   L= (y2 + y1) * Multiply_inverse(x2 + x1)
   x3 = L^2 + L + x1 + x2 + a;
   y3 = L * ( x1+ x3) + x3 + y1;
   return(x3, y3);
}
```

〈그림 3〉 점 P≠Q의 알고리즘

본 논문은 $GF(2^m)$상의 타원곡선을 사용하여 공유 비밀키를 생성한다. 이것은 상대방의 공개키를 이용하여 공유 비밀키(shared secret key)를 생성한다. 즉, 수신자가 타원곡선 알고리즘을 이용하여 비밀키와 공개키를 생성한 후, 인증기관의 공개 디렉토리에 공개키를 등록하면, 송신자는 등록된 수신자의 공개키를 이용하여 자신의 비밀키와 덧셈 연산하여 공유 비밀키를 생성한다. 송신자도 역시 본인의 공개키를 공개 디렉토리에 등록하게 되고, 수신자는 수신자의 비밀키와 송신자의 공개키를 덧셈 연산하여 공유 비밀키를 생성하게 된다.

ECSPK 알고리즘을 이용하여 생성된 공유 비밀키를 master_Secret 키 생성에 필요한 pre_master_secret 키 값으로 이용된다. 기존의 SSL에서 사용되는 알고리즘을 변경하기 위해, SSL Handshake



〈그림 4〉 ECSPK를 이용한 공유 비밀키 생성 과정

과정에서 Client Hello 메시지를 통해 클라이언트 SSL 버전, 클라이언트에서 생성된 임의의 난수, 세션 식별자, Cipher Suit 리스트, 클라이언트가 지원하는 압축방법 리스트 등의 정보를 서버에 전송하는데, 이때 제안된 알고리즘 정보로 SSL_ECSPK_WITH_ECC_EC-HMAC인 형태의 cipher_suites 리스트를 서버에 전송한다.

그림 4는 타원곡선을 이용한 공유 비밀키 생성 과정(ECSPK)을 나타낸다.

[단계 1] 송·수신측 S, R : 타원곡선 $E: y^2+xy=x^3+ax^2+1$을 선택한다.

[단계 2] 송·수신측 S, R : 감축 불가 다항식 $f(x)=x^4+x+1$을 선택한다.

[단계 3] 송·수신측 S, R : $GF(2^m)$ 타원곡선 상의 초기점 P를 정한다.

[단계 4] 송신측 S : 비밀키 s를 선택한 후, 공개 디렉토리의 초기점 P를 이용해 공개키 A를 계산한 후 공개 디렉토리에 등록한다.

[단계 5] 수신측 R : 비밀키 r를 선택한 후, 공개 디렉토리의 초기점 P를 이용해 공개키 B를 계산한 후, 공개 디렉토리에 등록한다.

[단계 6] 송신측 S : 수신측의 공개키 B에 자신의 비밀키 s만큼 덧셈 연산을 하여

공유 비밀키 Ssr를 생성한다.

[단계 7] 수신측 R : 송신측의 공개키 A에 자신의 비밀키 r만큼 덧셈 연산을 하여 공유 비밀키 Srs를 생성한다.

3.2 EC-HMAC 인증 알고리즘

본 논문에서 제안된 EC-HMAC은 $GF(2^m)$상의 타원곡선 알고리즘으로 공유 비밀키를 사용하는 Keyed HMAC로서 송신자와 수신자만이 알고 있는 공유 비밀키를 사용해 메시지 인증 코드를 생성한다. 제안된 EC-HMAC는 SSL 레코드 프로토콜에서 사용하는 메시지 인증 코드를 대신하여 사용하면서, 해쉬 함수가 단지 무결성 만을 확인하는데 비해 EC-HMAC 알고리즘은 공유 비밀키를 모르면 메시지를 인증할 수 없으므로 메시지의 인증은 물론 상대방의 신원 확인 및 전자서명에도 사용할 수 있다.

송신자는 본인의 공유 비밀키와 메시지를 결합하여 메시지 인증 코드를 생성하여 상대방에게 전송한다. 수신자는 전송된 메시지에 송신자와 같은 인증 알고리즘을 이용해 메시지 인증 코드(MAC*)를 생성한 후 전송된 MAC와 비교한다. 만약 두 값이 같다면 전송 중 메시지에 변조가 없다는 것을 검증할 수 있다.

그림 5는 EC-HMAC 인증 알고리즘의 검증 흐름을 설명한 것이다.

EC-HAMC를 이용한 메시지 인증 코드 생성과정은 아래와 같다.

① $GF(2^m)$상의 타원곡선으로 생성된 공유 비밀키 Sx, Sy를 각각 바이트 변환, 정수 변환하여 그 값을 연결(concatenation)하여 S_K 변수에 저장한다.

② 공유 비밀키 S_K의 길이를 검사한다.

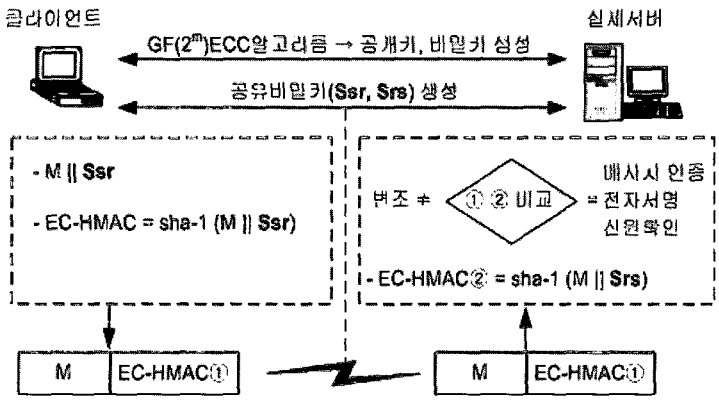
③ 키의 길이가 64바이트이면, 공유 비밀키를 S_K_0에 기억시키고, ⑥로 이동한다.

④ 공유 비밀키 S_K의 길이가 64바이트 보다 작으면, 공유 비밀키의 길이가 64바이트가 되도록 S_K의 끝에 0을 채운 후에 S_K_0에 기억시키고 ⑥로 이동한다.

⑤ 공유 비밀키 S_K의 길이가 64바이트보다 큰 경우에는 공유 비밀키 S_K를 SHA-1 함수로 다이제스트 한다. 공유 비밀키 S_K를 다이제스트 하면, 항상 20바이트가 출력되므로 나머지 44바이트는 0으로 채운 후에 S_K_0에 기억시킨다.

⑥ S_K_0와 ipad 값인 0x36을 Exclusive-OR 연산을 한다. 즉, S_K_0 ⊕ 0x36을 한다.

⑦ ⑥의 결과 끝에 메시지를 연결시킨 후, SHA-1 함수로 다시 한번 다이제스트를 한다. 즉,



〈그림 5〉 EC-HMAC 검증 개략도

```
EC-HMAC(Sx, Sy, message)
{
  Key_byte(Sx, Sy, Ix, Iy);
  Key_int(Ix, Iy, S_K);
  if (length(S_K== 64))
          S_K0=S_K;
  else if (length(S_K) < 64)
       pad_zeroes(S_K); S_K0=S_K;
  else if (length(S_K) > 64)
  {
       S_K=sha-1(S_K);
       pad_zeroes(S_K); S_K0=S_K;
  }
  EC-HMAC=sha-1((S_K0⊕opad)||
          sha-1((S_K0⊕ipad)||
          message;
  return EC-HMAC;
}
```

〈그림 6〉 메시지 인증 코드 생성 알고리즘

SHA-1((S_K_0⊕ipad) || Message)를 한다.

⑧ S_K_0와 opad 값인 0x5C와 Exclusive-OR 연산을 한다. 즉, S_K_0 ⊕ 0x5C을 한다.

⑨ ⑦의 결과와 ⑧의 결과를 연결한 후 SHA-1 함수로 다이제스트 한 결과 20바이트 값이 된다. 즉, EC-HMAC=SHA-1((S_K_0⊕opad) || SHA-1(S_K_0⊕ipad || Message))이다.

20 바이트 전체를 메시지 인증 코드로 사용하는 경우도 있고, 송·수신자간에 길이를 정해 20 바이트 전체를 사용하지 않고 처음부터 정해진 길이만큼 잘라서 메시지 인증 코드 값으로 사용하기도 한다[10].

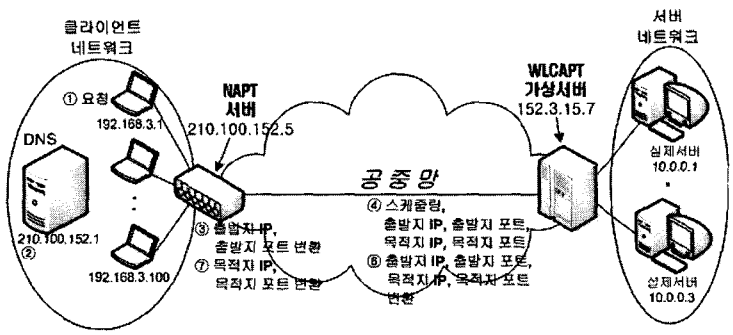
그림 6은 EC-HMAC 메시지 인증 코드 생성 알고리즘이다.

4. VPN 가상 서버 설계

4.1 NAPT(Network Address Port Translation)기법

사설 IP주소는 내부망에서는 통신이 가능하지만 외부망으로는 라우팅이 안 된다. 따라서 내부망의 사설 IP 주소를 외부망의 공인 IP주소로 바꾸어 주는 NAT가 필요하게 된다. 이러한 NAT기법중의 하나인 NAPT는 내부의 사설망에서 외부의 공중망으로 보내지는 패킷이 있을 때, NAPT 서버가 포트 번호를 임의로 할당하여 이것을 출발지 포트 번호로 사용하고, NAPT서버의 공인 주소를 출발지 주소로 변환한다. 내부로 들어오는 패킷에 대해서는 원래의 보내진 포트 번호로 되돌리고 IP 주소도 내부의 것으로 변환한 다음 전송한다[11,12].

NAPT를 사용하는 클라이언트 서버 구조에서 클라이언트는 임의로 선택되는 포트 번호로 구분하는데, TCP/IP 프로토콜에서 포트 번호는 0에서 65,535사이의 정수이며, 1~1,023까지는 IANA (Internet Assigned Numbers Authority)가 특정 서비



〈그림 7〉 SSL VPN 패킷 흐름도

스에 대한 포트 번호로 이미 할당하였다[13]. 따라서 서버는 해당 서비스들에 대해 IANA가 미리 할당해 놓은 포트 번호를 사용한다.

그림 7은 클라이언트가 HTTP 서비스를 받기 위해 서버에 요청 패킷을 보낼 때, 패킷이 NAPT 서버에서 주소 변환된 후, 외부 네트워크로 송신되고, 다시 응답 패킷이 수신되는 과정을 보여주고 있다.

< 클라이언트 측 >

요청 패킷의 처리

(1) 클라이언트가 HTTP 서비스를 받기 위해 먼저 EC-HMAC 알고리즘으로 인증 값을 계산하고, 암호화 알고리즘으로 요청 패킷을 암호화한다.

(2) 해당 서버로 패킷을 전송하기 위해 자신의 PC에 등록된 DNS(Domain Name Server) 서버에 DNS query를 보내어 자신이 접속하기를 원하는 서버의 주소를 알아낸다. 그 후 패킷의 출발지 주소와 포트번호, 목적지 주소와 포트 번호 각각을 src:192.168.3.1:1257,dst:152.3.15.7:80로 캡슐화 하여 전송한다.

(3) 클라이언트 측이 사설 IP 주소를 사용하므로 NAPT 서버는 기록 테이블에 패킷에 대한 정보를 기록한 후, 요청 패킷의 출발지 주소를 NAPT 서버의 공인 IP 주소로 하며, 출발지 포트 번호는 NAPT 서버가 임의로 할당한 번호로 변환되어 표 1의 형태로 전송된다. 변환된 요청 패킷을 역시 기록 테이블에 남긴다. 그림 8은 요청 패킷의 출발지 주소와 포트를 변환하는 알고리즘이다.

〈표 1〉 요청 패킷의 NAPT 변환

	출발지 주소, 포트	목적지 주소, 포트
수신된 패킷 IP주소	192.168.3.1:1257	152.3.15.7:80
NAPT변환	210.100.152:63451	152.3.15.7:80

```
client_source_napt(struct  pak_form
*req_data)
{
    for(i=1; i<=req_cnt; i++)
    {
        *bakup_req++=*req_data++;
         new_port=rand_port();
        *req_data.src_port=new_port;
         *req_data.source=napt_addr;
         *bak_trans_req++=*request_
          data++;
    }
}
```

〈그림 8〉 출발지 주소와 포트 변환 알고리즘

응답 패킷의 처리

(4) 응답 패킷을 수신한 NAPT 서버는 기록 테이블에서 응답 패킷에 대한 정보를 검색하여 요청에 대한 응답 패킷인지의 여부를 검사하고, 응답 패킷의 목적지 주소를 요청을 보낸 클라이언트의 사설 IP 주소와 포트 번호로 변환하여 표 2 형태로 클라이언트에 전송한다. 만약 기록 테이블에 응답 패킷에 대한 정보가 없으면 패킷을 폐기한다. 그림 9는 응답 패킷 목적지 주소와 포트를 변환하는 알고리즘이다.

〈표 2〉 응답 패킷의 주소 변환

	출발지 주소,포트	목적지 주소,포트
수신된 패킷 IP 주소	152.3.15.7:80	210.100.152.5:63451
주소변환	152.3.15.7:80	192.168.3.1:1257

(5) 클라이언트는 복호화 알고리즘으로 수신된 응답 패킷을 복호화하고, EC-HMAC 알고리즘으로 패킷의 무결성과 신원을 검사하여 변조되었으면 폐기 한다.

```
client_dest_napt(struct pak_form *temp_res_data)
{
 for(i=1; i<=res_cnt; i++)
 {// 백업배열에 주소변환된 요청패킷의 출발지 주소와 포트번호가
  // 응답패킷의 목적지 주소, 포트번호와 일치하면 주소변환을 수행
   for(j=1; j<=trans_cnt; j++)
  { bak_trans_req=temp_bak_trans_req;
    res_data=temp_res_data;
     if (*bak_trans_req.source==*res_data.dest &&
       *bak_trans_req.src_port==*res_data.dest_port)
    { for(k=1; k<=bak_cnt; k++)
      { bakup_req=temp_bakup_req;
        if (*res_data.src_port==*bakup_req.dst_port
          && *res_data.source==*backup_req.dest)
        {  *res_data.dest_port=*bakup_req.src_port;
           *res_data.dest=*bakup_req.source; }
         else bakup_req++;
      }
     else {bak_trans_req++;   res_data++;}
     }
   }
 }
}
```

〈그림 9〉 응답패킷 목적지 주소와 포트변환 알고리즘

4.2 WLCAPT(Weighted Least Connection Address Port Translation)기법

LSNAT는 트래픽 분산(load sharing)기능과 주소변환(address translation) 기능을 함께 가지고 있는 기술로서 서버 측에 적용하여 수천의 클라이언트 요청을 스케줄링 알고리즘을 이용하여 각 서버로 분산시킨다. 트래픽 분산 기능은 웹 사이트 구축에 필수적으로 사용되는 기술로 여러 대의 웹 서버를 운영하여 짧은 시간에 사용자의 요청을 처리할 수 있게 하는 목적으로 사용된다[14].

본 논문에서 제안하는 WLCAPT는 트래픽 분산을 위해 가중 최소 연결(weighted least connection)알고리즘과 주소 변환 기법중의 하나인 NAPT 기능을 결합시킨 것으로, 가상 서버에 적용하여 기존의 LSNAT의 단점을 보완하였다. 표 3은 WLCAPT 가상 서버에 정의된 규칙을 보여주고 있으며, 실제 서버는 HTTP 서비스를 수행한다.

〈표 3〉 WLCAPT 가상 서버에 정의된 규칙

Protocol	Virtual IP Address	Port	Real IP Address	Port	Weight
TCP	152.3.15.7	80	10.0.0.1	8000	3
TCP	152.3.15.7	80	10.0.0.2	9001	2
TCP	152.3.15.7	80	10.0.0.3	9003	1

표 3과 같은 규칙을 가상 서버에 설정하기 위해 리눅스의 kernal 2.4.x를 사용하여 patch하는 부분은 아래와 같다.

가상 서버가 패킷 마스커레이딩(NAPT)를 처리하도록 하기 위해 다음과 같이 설정한다.

```
echo 1> /proc/sys/net/ipv4/ip_forward
ipchains -A forward -j MASQ -s 10.0.0.0/20 -d
    0.0.0.0/0
```

웹 서버를 설정하는 경우, 가상 서버의 80번 포트 번호에 대해 가중 최소 연결 스케줄링 알고리즘을 설정한다.

```
ipvsadm -A -t 152.3.15.7:80 -s wlc
```

가상 서버에서 실제 서버를 다음과 같이 추가한다.

```
ipvsadm -a -t 152.3.15.7:80 -r 10.0.0.1:8000
-m -w 3
ipvsadm -a -t 152.3.15.7:80 -r 10.0.0.2:9001
-m -w 2
ipvsadm -a -t 152.3.15.7:80 -r 10.0.0.3:9003
-m
```

사설 IP 주소를 사용하는 네트워크의 클라이언트가 HTTP 서비스를 받기 위해 서버에 요청 패킷을 보낼 때, 요청 패킷이 서버측의 WLCAPT 가상 서버를 경유하고, 사설 IP 주소를 사용하는 해당 서버에서 서비스를 받은 후, 응답 패킷을 보내는 과정을 그림 7과 함께 살펴보면 아래와 같다.

< 서버 측 >

요청 패킷에 대한 처리

(1) 가상 서버의 실제 서버 선택 : 가상 서버는 수신된 패킷의 목적지 주소와 포트 번호를 검사하고 기록 테이블에 패킷에 대한 정보를 기록한 후, 가중 최소 연결 알고리즘을 실행하여 실제 서버를 선택한다.

(2) 가상 서버의 주소 변환 : 실제 서버가 선택되면 요청 패킷의 출발지 주소는 가상 서버의 주소로 하며, 출발지 포트 번호는 가상 서버가 임의로 할당한 번호로 변환한다. 목적지 주소는 선택된 실제 서버의 사설 IP 주소로, 포트 번호는 실제 서버 관리자가 해당 서비스에 대해 지정한 HTTP 포트 번호로 변환된다.

표 4와 같이 주소 변환된 패킷은 기록 테이블에 기록한 후, 실제 서버로 전송된다. 그림 10은 요청 패킷에 대한 가중최소연결과 주소 변환 알고리즘이다.

(3) 실제 서버의 패킷에 대한 주소 검사 : 실제 서버는 패킷이 WLCAPT 가상 서버를 통과하여 지정된 주소로 변환 되었는지의 여부를 판단

〈표 4〉 요청 패킷의 WLCAPT 변환

	출발지 주소,포트	목적지 주소,포트
수신된 패킷 IP 주소	210.100.152.5:63451	152.3.15.7.80
WLCAPT변환	152.3.15.7:5211	10.0.0.1:8000

```
wlc_nat(struct pak_form *req_data, int
w[], int c[])
 {
   for (i=1; i<=3; i++)
        pack_proce[i]=c[i]/w[i];
    // 가장적은 패킷 처리량을 갖는서버와 목적지 주소를 선택
   select_server=min_server(pack_proce);
   dest_addr=select_dest(select_server);
   // 실제 서버의 서비스에 대한 포트번호로 변환

dest_port=select_port(req_data.dest_port)
   // 주소 변환 과정
   for(i=0; i<=req_cnt; i++)
   {  *bakup_req++=*req_data++;
       new_port=rand_port();
       req_data.src_port=new_port;
       *req_data.source=vs_addr;
      req_data.dest_port=dest_port;
      req_data.dest_dest=dest_addr;
        // 변환된 패킷을 기록
     *bak_trans_req++=*req_data++;
   }
   return (select_server)
 }
```

〈그림 10〉 가중최소연결(weighted least connection)과 주소 변환 알고리즘

하기 위해 수신된 패킷의 주소 네 부분을 검사하여 모두 합당하다면 (4)을 수행하고, 그렇지 않으면 폐기한다.

(4) 실제 서버의 복호화, 인증 알고리즘 수행 : 패킷이 암호화 되었다면 복호화 알고리즘으로 복호화하고, EC_HMAC 알고리즘으로 패킷의 무결성과 신원을 확인하여 패킷이 변조되지 않았으면 요청에 대한 서비스를 수행한다. 만약 패킷이 변조되었으면 폐기한다.

응답 패킷에 대한 처리

(5) 실제 서버는 요청에 대한 응답 패킷을 보낼 때, 패킷을 EC-HMAC 알고리즘으로 인증값을 계산하고, 암호화 알고리즘으로 암호화한 후, 패킷을 표 5 형태로 캡슐화하여 다시 가상 서버로 전송한다. 그림 11은 목적지 주소와 포트 변환 알고리즘이다.

(6) 가상 서버의 패킷에 대한 주소 검사 : 가상 서버는 WLCAPT 가상 서버를 통과한 응답 패킷인지의 여부를 확인하기 위해, 기록 테이블에서 응답 패킷에 대한 정보를 검색한다. 만약 기록 정보가 없으면 폐기하고, 그렇지 않으면 (7)번을 수행한다.

(7) 가상 서버의 주소 변환 : 가상 서버는 요청에 대한 서비스를 가상 서버에서 수행한 것처럼 하기 위해, 응답 패킷의 출발지 주소를 가상 서버의 공인 주소로 하고, 출발지 포트 번호는 요청 서비스에 대한 포트 번호로 하며, 목적지 주소와 포트 번호는 가상 서버가 패킷을 수신했을 때의 출발지 주소와 출발지 포트 번호로 변환하여 표 6 형태로 전송된다. 그림 12는 응답 패킷의 출발지 주소와 포트변환 알고리즘이다.

〈표 5〉 실제 서버의 주소 변환

	출발지 주소,포트	목적지 주소,포트
수신된 패킷 IP 주소	152.3.15.7:5211	10.0.0.1:8000
주소변환	10.0.0.1:8000	152.3.15.7:5211

```
real-server_nat(struct pak_form *req_data)
  struct pak_form *res_data)
{
 for(i=1; i<=req_cnt; i++)
 { temp_source=*req_data.source;
 temp_port=*req_data.src_port;
 *res_data.source=*req_data.dest;

*res_data.src_port=*req_data.dest_port
 *res_data.dest=temp_source;
 *res_data.dest_port=temp_port;
 res_data++; req_data++;
 }
}
```

〈그림 11〉 목적지 주소와 포트 변환 알고리즘

〈표 6〉 응답 패킷의 주소 변환

	출발지 주소,포트	목적지 주소,포트
수신된 패킷 IP 주소	10.0.0.1:8000	152.3.15.7:5211
주소변환	152.3.15.7:80	210.100.152.5:63451

```
vs_source_napt(struct pak_form *temp_res_data)
{
 for(i=1; i<=res_cnt; i++)
 {for(j=1; j<=trans_cnt; j++)
  { bak_trans_req=temp_trans_req;
    res-data=temp_res_data;
// 백업 배열에 주소 변환된 요청패킷의 출발지 주소, 포트번호가
// 응답 패킷의 목적지 주소, 포트번호와 일치하면 주소변환을 수행
    if(*bak_trans_req.source==*res_data.dest &&
*bak_trans_req.src_port==*res_data.dest_port)
    { for(k=1; k<=bak_cnt; k++)
     { bakup_request=temp_bakup_req;
          if(*res_data.dest==*bakup_req.dest)
      { *res_data.source=vs_addr;
 *res_data.src_port=*backup_req.dest_port;
           *res_data.dest=*backup_req.source;
  *res-data.dest_port=*bakup_req.src_port;}
      else  bakup_req++; }
   else { bak_trans_req++;  res_data++;}
  }
 }
}
```

〈그림 12〉 출발지 주소와 포트 변환 알고리즘

5. 시뮬레이션 및 평가

5.1 Diffie-Hellman과 ECSPK

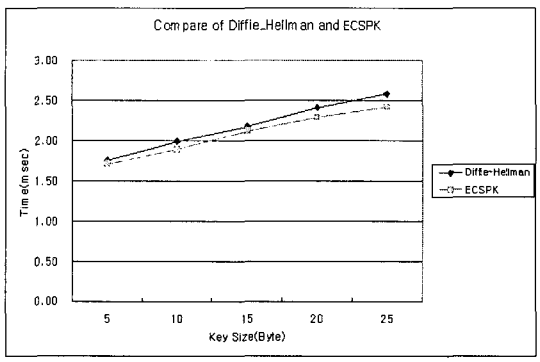
Diffie-Hellman은 송·수신측이 공유 비밀키를 교환하도록 하는 키 교환 알고리즘으로 이산 대수 문제를 기반으로 하고 있다. ECSPK 알고리즘은 Diffie-Hellman 알고리즘을 타원곡선 위로 옮긴 것으로 Diffie-Hellman의 곱셈 연산 대신에 타원곡선의 덧셈 연산으로 공유 비밀키를 생성하므로, 연산 속도가 감소되었다.

그림 13은 Diffie-Hellman과 ECSPK의 공유 비밀키 생성 시간에 대한 비교 그래프이다.

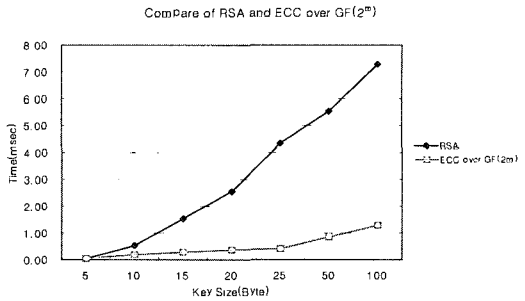
5.2 RSA와 $GF(2^m)$상의 ECC

ECC 알고리즘은 RSA의 키 크기가 1024비트인데 비해 단지 163비트의 크기만을 요구하기 때문에 그만큼 대역폭을 적게 차지하게 되며, 이산 대수문제를 기반으로 덧셈 연산을 수행하므로 곱셈의 멱승으로 키 생성을 하는 RSA보다 키 생성 시간이 10배 이상 빠르다[15,16].

특히, 본 논문에서 사용한 타원곡선은 $GF(2^m)$상의 polynomial basis로 E : $y^2+xy=x^3+ax^2++b, a,b\in GF(2^m), b\neq 0$이고, 구현의 효율을 위해 감축 불가 다항식으로$f(x)=x^4+x+1$를 사용하였다.



〈그림 13〉 Diffie-Hellman과 ECSPK의 공유 비밀키 생성시간 비교



〈그림 14〉 RSA와 ECC의 키 생성시간 비교

그림 14에서 보여 지는 것처럼 키 크기가 커질수록 RSA와 ECC의 키 생성 시간의 차이가 점점 커진다는 것을 알 수 있으며, ECC가 훨씬 빠르다는 것을 알 수 있다.

5.3 스케줄링 알고리즘 성능 비교

스케줄링 알고리즘은 트래픽 분산 기능을 가지고 있는 기술로서 서버에 작업을 균등하게 분산하여 서버가 자신의 성능에 맞게 요청을 처리할 수 있게 하는 목적으로 사용된다.

본 논문에서는 최소 연결(least connection)알고리즘과 가중 최소 연결(weighted least connection) 알고리즘을 4가지 가정 하에 시뮬레이션하여 성능평가 하였다. 첫째, 가상 서버의 패킷 할당 단위는 1sec로 한다. 둘째, 서버1, 서버2, 서버3의 초당 패킷 처리 능력을 각각 15, 10, 5 패킷이라 하고, 각 서버의 가중치는 3, 2, 1이라 한다. 셋째, TCP의 TIME_WAIT 시간을 5초로 하여 클라이언트가 요청 패킷을 보내고 5초 이내에 응답 패킷을 보내지 못하면 패킷을 폐기한다. 넷째, 초당 요청 패킷들의 범위는 20~40 패킷으로 하여 실험하였다.

실험 결과에서 가중 최소 연결 알고리즘을 사용하는 서버가 최대 1.5배의 패킷 처리능력을 보였다. 따라서 가중 최소 연결 알고리즘이 훨씬 효율적으로 트래픽을 분산하므로 더 많은 양의 요청 패킷을 처리한다는 것을 알 수 있다.

5.4 가중 최소 연결 알고리즘기반 LSNAT와 WLCAPT 비교

그림 15와 그림 16의 결과처럼 트래픽을 더욱 효율적으로 분산하는 가중 최소 연결 알고리즘을 기반으로 하는 LSNAT와 WLCAPT를 비교 평가하였다.

표 7은 가상 서버에 수신된 패킷의 출발지 주소, 출발지 포트 번호, 목적지 주소, 목적지 포트 번호가 src:210.100.152.5:63451, dst:152.3.15.7:80와 같을 때, 가상 서버에서 LSNAT와 WLCAPT의 패킷 주소 변환 범위를 비교하여 보여주고 있다.

〈표 7〉 LSNAT와 WLCAPT의 패킷 주소 변환 범위

주소변환 범위 / 주소변환 알고리즘	출발지 주 소	출발지 포 트	목적지 주 소	목적지 포 트
LSNAT	210.100.152.5	63451	10.0.0.1 (변환)	80
WLCAPT	152.3.15.7 (변환)	5211 (변환)	10.0.0.1 (변환)	8000 (변환)

표 7에서 보여 지는 것처럼 LSNAT가 패킷의 목적지 주소만을 변환하는 데에 비해 WLCAPT는 패킷의 주소 네 부분이 모두 변환되어 실제 서버로 전송된다. LSNAT를 가상 서버에 적용하였다면, 실제 서버의 사설 IP 주소를 알고 있는 악의의 침입자가 기록을 남기지 않기 위해 가상 서버를 통과하지 않고 주소 변환하여 실제 서버로 접근할 수 있게 된다. 이것은 LSNAT가 오직 목적지 주소만을 변환하기 때문에 가능하게 된다. 반면, WLCAPT를 가상 서버에 적용하여 패킷을 실제 서버로 전송한다면, 실제 서버는 먼저 패킷의 주소 부분을 검사하고, 네 부분 모두가 지정된 주소와 번호로 변경되지 않았을 경우 패킷을 폐기하게 된다.

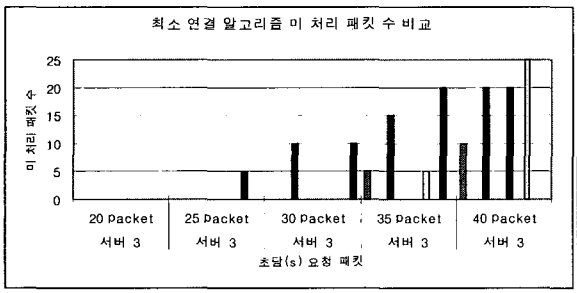
따라서 LSNAT가 가상 서버를 통과하지 않고도 목적지 주소만을 변환하여 실제 서버로 직접 접근할 수 있는 약점을 WLCAPT로 보안할 수 있게 된다.

6. 결 론

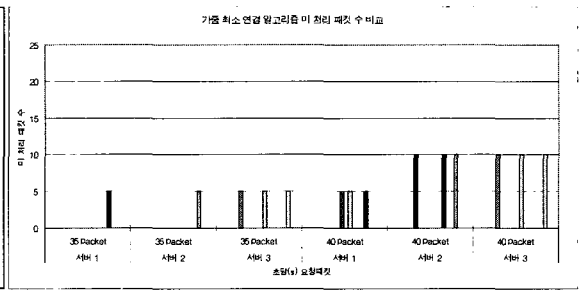
본 논문에서 제안하는 시스템은 실험 결과에서처럼 네 가지 측면에서 전체적으로 성능과 보안이 향상되었다.

첫째, $GF(2^m)$상의 ECC 알고리즘으로 공개키와 비밀키를 생성하고, ECSPK 알고리즘으로 공유 비밀키를 생성하므로 SSL 프로토콜의 보안과 처리속도를 향상하였다.

둘째, 실제 서버 앞에 가상 서버를 두고, 이 가상 서버의 주소를 DNS에 등록하였다. 따라서 실제 서비스를 수행하는 실제 서버는 외부망에 노출되지 않을 뿐만 아니라, 사설 IP 주소를 사용하기 때문에 외부와 직접 통신이 불가능하게 되어 보안을 높이는 효과를 얻을 수 있다.



〈그림 15〉 최소 연결 알고리즘 미 처리 패킷 수 비교



〈그림 16〉 가중최소연결 알고리즘 미 처리 패킷 수 비교

셋째, 트래픽 분산을 가장 효율적으로 수행하는 가중 최소 연결 알고리즘을 가상 서버에 적용하여 각 서버가 더 많은 양의 요청 패킷을 처리할 수 있게 하였다.

넷째, WLCAPT는 패킷의 주소 네 부분이 모두 변환되어 실제 서버로 전송된다. 그렇기 때문에 실제 서버는 패킷의 주소 네 부분이 합당하게 변환되지 않은 패킷을 폐기하게 된다.

따라서 목적지 주소만을 변환하는 기존의 LSNAT는 실제 서버의 사설 IP 주소를 알고 있으면 가상 서버를 통과하지 않고도 임의로 주소 변환하여 실제 서버로 접근할 수 있게 되는데, 이러한 LSNAT의 단점을 WLCAPT로 보완할 수 있게 되어 보안성을 높일 수 있다.

참 고 문 헌

[1] Hacienda Avenue Compbell, "SSL VPN vs. IPSec VPN", 2002. pp.3~6,

[2] Andrew Conry-Murray, "SSL VPNs: Remote Access for the Masses", Network Magazine, Oct.6.2003.

[3] SSL 3.0 Specification, http://www.nescaple.com/endg/ss3/draft302.txt

[4] 한진희, 하경주, 정교일, 손승원, "인터넷 전자상거래를 위한 보안 기술", 『한국인터넷정보학회 추계학술발표논문집』 제 1권 2호, 2000. pp. 33~43

[5] 이만영, 손승원, 조현숙, 정태명, 채기준. "차세대 네트워크 보안 기술", 생능출판사, 2002. pp. 249~255

[6] 주식회사 니츠, "인터넷 보안 기술(I)", 동서출판사, 2000. pp.251~256

[7] 최은실, 정은희, 이병관, "ECC기반 VPN 터널링 프로토콜 설계", 『한국정보처리학회 추계학술발표논문집』, Vol.10, No.2, 2003. pp. 2001~2004

[8] 한국정보통신기술협회, "타원곡선을 이용한 인증서 기반 전자서명 알고리즘", 2001. pp. 8~11, pp. 14

[9] 정은희, "타원곡선 보안 소켓층 프로토콜 설계 및 평가", 관동대학교, 2003. pp. 47~49

[10] H. Krawczyk, M.Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, Februry. 1997. pp. 1~3

[11] P. Srisuresh, M. Holdrege, "IP Network Address Translator(NAT) Terminology and Considerations", RFC 2663, August. 1999. pp. 1~6

[12] 김광호, 임채훈. "인터넷 보안 기술 개론 -NAT(Network Address Translator)", FS-TROO-09, 2000. pp. 1~6

[13] 김병철 외 5인 공역, "TCP/IP 프로토콜", 미래컴, 2001. pp. 337~341

[14] P. Srisuresh, D. Gan. "Load Sharing using IP Network Address Translation (LSNAT)", RFC 2301, August. 1998.

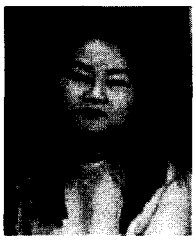
[15] 이용기, 이정규. "타원곡선 공개키 알고리즘의 효율성", Journal of Engineering & Technology Hanyang University 공학기술논문집, Vol. 7, No.1, 1998. pp. 295~310

[16] 정은희, 이병관, "ECSSL(Elliptic Curve SSL) 기반 DIT(Digital Investment Trust) 에이전트", 『정보처리학회 논문지』, Vol.9-B, No.5, 2002. pp. 599~608

◐ 저 자 소 개 ◑

정 은 희

1991. 2 강릉대학교 통계학과 이학사
1998. 2 관동대학교 전자계산공학과 공학석사
2003. 2 관동대학교 전자계산공학과 공학박사
2003. 9～현재 삼척대학교 전자상경제학전임강사
관심분야 : 네트워크 보안, 전자상거래, 웹 프로그래밍
E-mail : jeh@samcheok.ac.kr
Tel:+82-33-570-6646
Fax:+82-33-574-6640



최 은 실

2000. 2 삼척대학교 컴퓨터공학과 공학사
2002. ～현재 관동대학교 전자계산공학과 석사과정
관심분야 : 네트워크보안, 데이터통신, 전자상거래
E-mail : tosil17@hanmail.net

이 병 관

1975. 2 부산대학교 기계설계학과 학사
1986. 2 중앙대학교 전자계산공학과 석사
1990. 2 중앙대학교 전자계산공학과 박사
1988. 3～현재 관동대학교 멀티미디어공학전공 교수
관심분야 : 네트워크 보안, 전자상거래, 컴퓨터 네트워크
E-mail : bklee@kwandong.ac.kr
Tel:+82-33-670-3373
Fax:+82-33-670-3370