안전한 TCP/IP 통신을 위한 128bits SSL-Proxy 설계 및 구현

박성준*, 김창수*, 황수철**
*부경대학교 전자계산학과, **인하공업전문대학 전자계산기과

Design and Implementation of 128bits SSL-Proxy for Secure TCP/IP Communication

Sung-June Park*, Chang-Soo Kim*, Soo-Chul Hwang**
*Dept. of Computer Science, PuKyong Nat'l University
**Dept. of Computer Engineering, Inha T. College

요 약

최근 인터넷 사용자의 증가와 인터넷을 이용한 전자상거래가 활발해지면서 TCP/IP 통신 보안에 대한 많은 연구들이 진행되고 있다. 이러한 연구들 중 SSL(Secure Socket Layer)은 인증 및 보안 프로토콜로서 인터넷을 이용한 통신 보안에 가장 많이 사용되고 있는 방법이다. 본 논문에서는 국내 암호 알고리즘의 표준으로 채택된 SEED알고리즘을 SSL에 적용하여 서버와 클라이언트의 통신을 128bits 암호화할 수 있는 SSL-Proxy 프로그램을 구현하였으며, Windows와 Unix 시스템에 모두 적용 가능하도록 모듈화하였다.

1. 서론

인터넷은 웹 기술을 이용한 브라우저(browser)의 출현으로 멀티미디어 정보를 쉽고 자유롭게 사용할 수 있게 됨으로써 그 사용이 폭발적으로 증가하고 있다. 우리 나라에서도 최근 인터넷 열풍에 힘입어 초기에 주로 정보검색과 학술적 목적에 사용되었던 인터넷이 최근에는 제품에 대한 광고나 판매 등 상업적 용도로 그 사용 영역이 점차 확대되고 있는 추세이다. 이러한 인터넷의 상업적 이용의 대표적인 예가 전자상거래(EC, Electronic Commerce)이다. 사용자층도 다양화되면서 정부에서는 전국민이 인터넷을 사용할 수 있게 한다는 취지 하에서 인터넷 PC보급, 지식 정보 기반구축사업들을 추진하고 있다.

한국전산원의 "'99국가 정보화 백서"[1]의 내용에 따르면 국내 인터넷 호스트수는 년 100%이상의 높은 성장률을 보이고 있으며 최근에는 그 성장 속도가 더욱 빨라지고 있는 상황이다(표 1-1).

표 1-1 인구 1000명당 인터넷 호스트 수

연도	1991	1992	1993	1994	1995	1996	1997	성장율
호스트수	0.03	0.08	0.20	0.40	0.65	1.45	2.65	1.06

인터넷 이용자수의 급격한 확산은 산업 발전과 국자 정보화 발전에 긍정적인 영향을 미치고 있으나, 긍정적인 면과 함께 부정적인 문제들도 많이 발생시키고 있다. 최근의 미국 유명 인터넷 사이트들의 서비스 거부공격 사례와 같은 악의적인 인터넷 사용에 의한 경제적 손실과 개인 정보 유출, 인터넷의 익명성을 악용한 범죄들이 그러한 부정적인 면의 대표적인 것들이다. 특히, 전자상거래가 시간과 공간의 제약을 받지 않는 새로운 개념의 상행위로 인식되면서 여러 벤처 업체들과 대기업들이 앞 다투어 전자상거래 시장에 뛰어 들면서 개인 정보의 유출과 사용자의 부인 행위 등의 사고들이 많이 발생하고 있다. 인터넷은 개방시스템을 기반으로 설계된 TCP/IP 프로토콜을 사용하고 있으며, 대부분의 웹서버는 UNIX기반이기 때문에 웹 기반의 전자상거래는 보안이 취약한 것으로 인식되고 있다. 이는 거래당사자에 대한 거래자의 신

뢰, 거래에 대한 보호, 안전한 지불 등이 무엇보다 중요한 전자상거래에 있어서는 결코 간과할 수 없는 문제이다. 이와 같은 상황에서 최근 많은 보안 제품들이 출시되고있으며 다양한 방법의 보안 알고리즘들이 연구되고 있다. 본 논문에서는 안전한 TCP/IP통신을 구현하기 위하여 SSL프로토콜을 사용하여 128 bits SSL-Proxy프로그램을 설계하고 구현하였다.

2. 인터넷 보안

(1) 인터넷 보안을 위한 방법

TCP/IP통신, 특히 웹에서의 안전하고 신뢰성 있는 통신을 위해 제시된 것으로는 SSL(Secure Socket Layer)과 S-HTTP(Secure HTTP)가 있다. S-HTTP는 현재의 WWW 프로토콜인 HTTP(HyperText Transfer Protocol)에 암호 알고리즘을 추가함으로써 기밀성 및 무결성, 인증을 보장하는 것이고, SSL은 어플리케이션과 TCP/IP 계층사이에 존재하는 것으로서 S-HTTP와 SSL은 상호 배타적인 구조가 아니다. EIT(Enterprise Integration Technologies)와 Netscape사는 공동연구를 통해 S-HTTP와 SSL를 통합하는 방법을 개발하여 공식 표준화에 노력하고 있다. 국내에서는 S-HTTP는 많이 사용되고 있지 않으며, 전자상거래 사이트의 보안을 위해 대부분이 SSL을 채택하여 사용하고 있다.

S-HTTP는 1994년 EIT에서 발표한 것으로, 용어에서 쉽게 알 수 있듯이 기존의 HTTP 트랜잭션의 형태를 그대로 보존하며 보안기능을 추가시킨 프로토콜이다. 이 프로토콜은 응용수준에서 메시지의 암호화를 통해 기밀성을 보장하는 것으로 RSA Data Security사의 공개키 암호화 알고리즘을 이용한다. 이 프로토콜의 특징은 클라이언트와 서버에서 사용되는 암호화 처리과정이 동일하며, PGP(Pretty Good Privacy), PEM(Privacy Enhanced Mail)등 여러 암호 메카니즘에 사용되는 다양한 암호문 형태를 지원할 수 있으며, 기밀성 요구정도에 따라 선택적 협상(option negotiation)을 통한 다양한 암호 알고리즘, 모드, 매개변수에 융통성을 제공한다. 또한 서명이나 인증, 암호화 동작이 상호 독립적으로 수행되며 키는 암호문에 추가해서 보내거나 독립적 채널을 통한 전송할 수 있다. 메시지 무결성을 위해서는 메시지 인증코드(MAC, Message Authentication Code)를 사용한다. S-HTTP는 그 용도가 HTTP에 제한되어 있으므로 다양한 종류의 TCP/IP통신에 적용할 수 없다는 단점을 가지고 있다[5].

SSL은 Netscape Communications 사에서 개발한 것으로 데이터의 암호화 및 서버 인증, 메시지 무결성을 제공하며 선택적으로 클라이언트에 대한 인증도 지원한다. 클라이언트와 서버간의 TCP/IP 연결을 위해 handshake 프로토콜을 수행하며, URL, 접근인증 자료 등과 같은 HTTP request와 response에 포함되는 모든 정보들이 암호화되어 전송된다. 데이터 암호화는 handshake 프로토콜 단계에서 비밀키를 결정한 후 암호화하며, 공개키 암호화 알고리즘을 사용하여 상대방을 인증한다. 또한 MD5, SHA 등의 해쉬 함수를 이용하여 생성한 메시지 인증코드로 메시지 무결성을 제공한다. SSL은 TCP프로토콜의 상위에 위치하여 HTTP뿐만 아니라 다양한 형태의 TCP/IP통신에 적용할 수 있으며 선택적으로 클라이언트에 대한 인증기능을 제공하고 있으므로 전자상거래에 필요한 부인방지 기능을 제공할 수 있다[2][3][4].

(2) SSL 프로토콜을 사용한 기존 제품들

SSL 프로토콜은 이미 많은 제품과 웹사이트에서 사용되고 있으며, Proxy형태의 제품도 국내·외에 다수 출시되어 있다. 그러나 최근까지 미국의 수출제한 법률에 의해 기능적으로 많은 제약을 받아오고 있으며 HTTP 이외의 TCP/IP 통신에 적용시키기 어려운 단점을 가지고 있다.

- Apache-SSL Web-Server[12]
  : Apache work group의 공개 웹 서버로 SSLeay 라이브러리를 사용하는 SSL 사용 웹 서버. 현재 SSL을 사용하는 웹 서버 중 시장 점유율이 가장 높은 제품으로 웹 서버 자체적으로 SSL 기능을 제공한다.
- SafePassage[13]
  : C2NET사의 대표적인 SSL-Proxy 프로그램으로 북미 이외의 지역에서 128bit 암호화를 제공하기 위하여 개발된 프로그램으로 HTTP프로토콜에 사용할 수 있다.
- KSSL[14]
  : KT에서 개발한 한국형 SSL 라이브러리로 SEED 암호 알고리즘을 채택하여 개발되었으며, 현재 뱅크타운의 전자지갑에 사용되어 인터넷 금융부분에서 많이 사용되고 있다.
- SSLProxy [15]
  : 장 미디어의 암호화 연구소에서 개발한 소프트웨어로 자바언어를 기반으로 작성되었으며, 128bit와 40bit의 암호화를 선택적으로 적용할 수 있다.

3. SSL 프로토콜

(1) SSL의 개요

SSL(Secure Socket Layer) 프로토콜[2][3]은 Netscape사가 1996년 3월 Version 3을 발표한 보안 프로토콜로 통신을 하는 양측에 인증 및 기밀성을 제공한다. SSL 프로토콜은 Netscape사에서 발표한 이후 대표적인 인터넷 보안 프로토콜로 인식되고 있으며, 인터넷 익스플로러와 넷스케이프에 기본으로 포함되어 현재 보안이 필요한 인터넷 사이트들이 가장 많이 사용하고 있는 보안 프로토콜이다.

SSL 프로토콜은 SSL Handshake 계층과 SSL Record 계층으로 구성되어 있다. SSL Handshake 계층은 두 계층 중 상위에 위치하는 계층으로 서버와 클라이언트간 통신을 수행하기 전에 상호 인증하고, 암호화된 통신을 위해 키 교환을 수행한다. SSL Record 계층은 TCP 프로토콜 위에 위치하며 상위계층의 데이터를 받아서 캡슐화(encapsulation, 암호화)하는 기능을 담당한다. SSL 프로토콜의 장점은 애플리케이션 프로토콜에 독립적이며, 통신 연결의 보안을 위하여 다음과 같은 세 가지 특성을 가진다.

① 연결 설정단계에서 비밀키가 결정되고 통신과정의 data들은 대칭키 암호화 알고리즘(DES, RC4 등)으로 암호화되어 전송된다.

② 인터넷 사용자의 신원을 인증하기 위해 비대칭키 또는 공개키 암호화 알고리즘(RSA, DSS 등)이 사용된다.

③ 전송되는 메시지들은 keyed MAC(SHA, MD5 등)을 이용한 무결성 검증 코드들을 포함하여 전송된다.

(2) SSL Handshake 계층

SSL Handshake 계층의 역할은 서버와 클라이언트의 안전한 통신을 위한 준비와 상호 인증기능이다. 즉, 서버와 클라이언트가 암호화된 통신을 하기 위해 인증을 거쳐 연결을 설정하고 비밀키를 교환하여 양쪽의 통신을 위한 가상 통로(pipe)를 만드는 것이다. 이러한 일련의 과정에는 RSA나 DSS와 같은 공개키 기반의 암호와 인증 알고리즘이 적용된다[10]. SSL Handshake 프로토콜의 동작은 그림 3-1과 같이 수행된다.

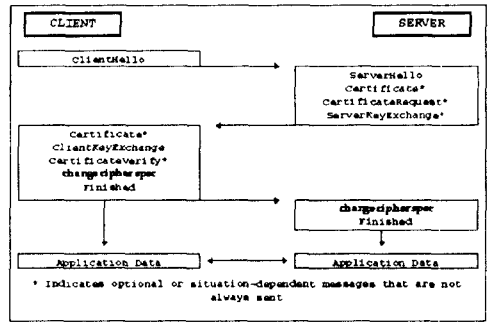

그림 3-1 Handshake 프로토콜

◆ Hello Message 처리 : 클라이언트는 서버와 연결을 설립하기 위해 먼저 hello 메시지를 보내고 서버측에서 hello 응답을 보내면 보안 능력이 있는 연결이 설립된다. 클라이언트와 서버의 hello 메시지에 포함되어 있는 내용은 프로토콜 버전, 세션 ID, 암호 조합, 압축방법, 랜덤 값으로 구성되어 있다.

◆ 인증서 및 키 교환 : 클라이언트와 서버의 hello 메시지가 교환된 후, 서버의 인증서가 클라이언트에 보내지고, 클라이언트는 비밀키를 생성하여 서버의 공개키를 사용해서 암호화하여 서버로 전송한다.

◆ Handshake 종료 : 암호화 알고리즘의 선택과 키 교환이 모두 끝나면 Handshake 단계가 종료되고, 본격적인 서버와 클라이언트의 통신을 위해 Record 계층의 통신이 시작된다.

(3) SSL Record 계층

데이터 송신의 경우 SSL Record 계층은 상위 계층의 가변적 크기를 가진 불규칙적인 데이터를 받아서 일정 크기로 분할(fragmentation)하는 기능을 수행한다. 이렇게 분할된 데이터를 Handshake 과정에서 선택된 내용으로 압축, 암호화, 메시지 인증코드 추가 등의 과정을 거쳐 SSLCiphertext 형태의 정형화된 형태로 구성한다. 이와같이 정형화된 데이터들을 일반적인 소켓의 사용방법과 동일하게 TCP계층으로 보내진다.

데이터 수신의 경우는 앞의 과정과 역으로 처리되며, TCP 계층에서 일정한 크기로 수신되는 SSLCiphertext 데이터를 받아서 메시지 인증코드를 계산하여 수신된

데이터의 무결성을 검증하고, 복호화한 후, 평문으로 변환하여 상위의 응용 계층에 전달한다[11].

SSL Handshake 계층과 Record 계층에서 사용되는 기본적인 암호화 알고리즘의 조합은 표 3-1과 같으며, 표 3-1에 나타난 표기는 SSL_'키 교환 알고리즘'_'암호 알고리즘'_'해쉬 알고리즘'의 형태로 표현되어있다[6].

표 3-1 SSL CipherSuit

CipherSuite
SSL_NULL_WITH_NULL_NULL
SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
SSL_RSA_WITH_IDEA_CBC_SHA
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_DSS_WITH_DES_CBC_SHA
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA
SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_RSA_WITH_DES_CBC_SHA
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
SSL_DH_anon_WITH_RC4_128_MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_anon_WITH_DES_CBC_SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
SSL_FORTEZZA_DMS_WITH_NULL_SHA
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA

4. SEED 알고리즘

암호 알고리즘은 암·복호화에 사용되는 키의 특성에 따라 암·복호화 키가 같은 대칭키 암호 알고리즘과 암·복호화 키가 서로 다른 공개키 암호 알고리즘으로 구분할 수 있으며, 대칭키 암호 알고리즘은 데이터 처리 형식에 따라 스트림 암호 알고리즘과 블록 암호알고리즘으로 나눌 수 있다.

SEED는 대칭키 암호알고리즘이며, 블록 단위로 메시지를 처리하는 블록 암호알고리즘이다. 대부분의 블록알고리즘과 같이 SEED도 Feistel 구조(그림 4-1)로 설계되어 있으며, 128bit 단위의 블록 처리와, 128bit 길이의 비밀키를 사용한다[9].

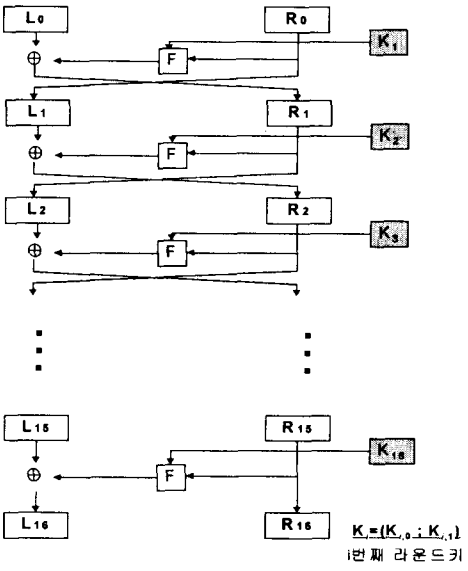

그림 4-1 SEED 알고리즘의 구조

SEED 암호 알고리즘은 KISA(Korea Information Security Agency)에서 국내 표준으로 제정하여 배포하고 있는 국산 암호 알고리즘으로 안전성 및 효율성이 입증되었으며, 차후 전자 상거래 등을 위한 보안 소프트웨어 개발에 많이 적용될 것으로 예상된다.

5. 128bits SSL-Proxy 프로그램 설계 및 구현

본 논문에서 설계하고 구현한 sslproxy 프로그램은 기본적으로 모든 종류의 TCP/IP 통신에 사용될 것을 가정하고 작성되었으며, 테스트를 위하여 HTTP프로토콜에 적용하였다. 그리고 기존의 보안 기능이 없는 TCP/IP 프로토콜 기반의 상위 계층 응용 프로그램들을 수정하기 않고 SSL 보안 기능을 제공하도록 작성되었으며, 국내 표준으로 제정된 SEED 암호 알고리즘을 추가하여 구현하였다.

(1) 시스템 환경

- OS : Windows 9.x, Unix, Linux
- 구현 도구 : Visual C++(Windows), gcc(Unix, Linux)
- 사용 프로토콜 : SSLv2, SSLv3, TLSv1
- 사용 Library : OpenSSL[1], SEED

본 논문에서 구현한 프로그램은 Windows 계열의 OS와 Unix 계열의 OS 양쪽 모두에서 컴파일 할 수 있도록 작성되었으며, 서버 쪽의 프록시 프로그램과 클라이언트 쪽의 프록시 프로그램을 별개로 작성하지 않고 하나의 프로그램에 서버와 클라이언트의 기능을 모두 포함하여 작성하였다.

(2) sslproxy 구성도

본 논문의 프로그램은 TCP/IP 기반의 서버와 클라이언트 양쪽에 설치되어 서버와 클라이언트는 각각의 프록시 프로그램과 통신을 하게 되고 실제 전송로 상의 통신은 서버와 클라이언트에서 실행되고 있는 sslproxy간에 이루어지게 된다(그림 5-1)[8].

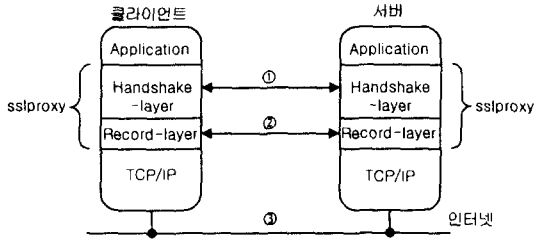

그림 5-1 구현된 sslproxy 구성도

그림 5-1은 sslproxy 프로그램을 사용한 통신의 개념도로 양쪽의 TCP/IP 통신에 SSL의 Handshake 계층과 Record 계층이 추가된 것을 볼 수 있다. 그림에서 sslproxy는 응용프로그램(Application)의 아래쪽에, TCP/IP 프로토콜의 위쪽에 위치하고 있으며, 응용프로그램의 통신을 보안기능과 인증기능을 추가하여 대행하게 된다. 이러한 기능들을 구현하기 위한 실제 통신의 순서는 Handshake 계층의 키 교환 및 상호인증(그림 5-1의 ①)이 먼저 수행되고, Handshake 계층의 수행 결과를 사용하여 양쪽의 통신에 암호화와 무결성 검증 코드를 추가하여(그림 5-1의 ②) 실제 전송로를 사용하여 통신을 수행하게 된다(그림 5-1의 ③).

(3) HTTP 프로토콜에 sslproxy 프로그램 적용

대표적인 TCP/IP 사용 프로토콜인 HTTP 프로토콜에 sslproxy 프로그램을 적용한 경우의 구조는 그림 5-2와 같다. 본 논문에서 실시한 테스트에서 클라이언트의 웹 브라우저는 MS의 Internet Explorer5.0을 사용하였으며, 웹 서버는 Apache1.3.9를 사용하였다. MS의 Internet Explorer는 최근까지 미국의 수출 규제 법률에 의해서 40bit의 암호키를 사용하는 SSL기능만을 제공하였으나 최근 Win2000출시와 함께 128bit의 암호 기능을 제공하는 SSL을 사용할 수 있게 패치되었다. 그러나 MS의 웹 브라우저는 소스코드를 공개하지 않고 있으므로 웹 브라우저 자체에 다른 암호 알고리즘을 적용한 SSL기능을 추가하는 것이 어렵고, SSL기능 이외에도 다른 기능을 브라우저에 추가하는 것이 사실상 불가능하다. Apache 웹 서버는 전 세계적으로 가장 많이 사용하고 있는 웹 서버로 SSLeay라이브러리와 Apache-SSL 패치를 사용하여 자체에 SSL 기능을 포함시킬 수 있다. 그림 5-2에 사용된 웹 서버는 일반적인 HTTP 접속만을 허용하는 웹 서버로 well-known port 80을 사용한다.

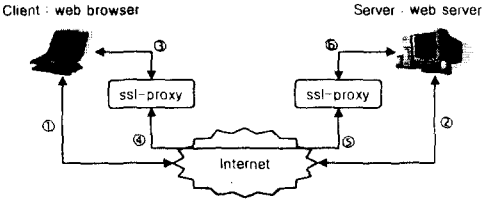

그림 5-2 HTTP 프로토콜에 적용한 구성도

- http 통신 : 일반적인 HTTP 접속의 경로로 웹 브라우저에서 http://~ 로 접속한 경우에 해당되며 보안 기능이 제공되지 않는 TCP/IP를 사용한 통신이 이루어지며, port 80을 사용한다. 그림 5-2의 web-browser↔①↔Internet↔②↔web-server의 경로를 사용하여 통신하는 경우에 해당된다. 이러한 접속에서 sniffing을 사용한 도청을 하였을 경우 사용자가 입력하고 서버가 보내는 모든 정보들을 쉽게 도청할 수 있다.[7]
- https 통신 : 클라이언트의 web browser는 클라이언트측의 sslproxy 프로그램과 연결, 통신을 수행하고, 서버의 web server는 서버측의 sslproxy 프로그램과 연결, 통신을 수행한다. 양측의 sslproxy 프로그램은 인터넷을 통하여 SSL(SSL2, SSL3, TLS1중 선택) 프로토콜을 사용한 암호화된 통신을 실행한다. 그림 5-2의 web-browser↔③↔sslproxy↔④↔Internet↔⑤↔sslproxy↔⑥web-server의 경로를 통하여 통신이 이루어지고, 통신을 위한 port는 7088(③), 443(④,⑤), 80(⑥)번이 사용된다. 이러한 통신을 위하여 sslproxy 프로그램의 클라이언트측에서는 웹 브라우저에서 프록시로 내려오는 데이터를 웹 서버에서 인식 할 수

있는 형태로 http 헤더를 재 조합하는 부가적인 작업이 필요하다. 이러한 방법의 통신에서는 도청자가 전송로 상에서 도청을 수행하는 것이 불가능하고 서버와 클라이언트는 인증서를 사용하여 상호인증하는 것이 가능하다. https 통신을 위하여 sslproxy 프로그램을 사용하였을 때의 실행은 그림 5-3, 4와 같다.

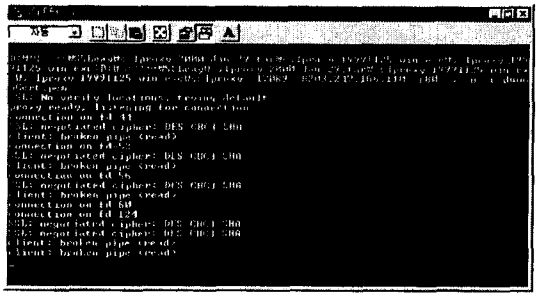

그림 5-3 sslproxy 프로그램의 서버측 동작

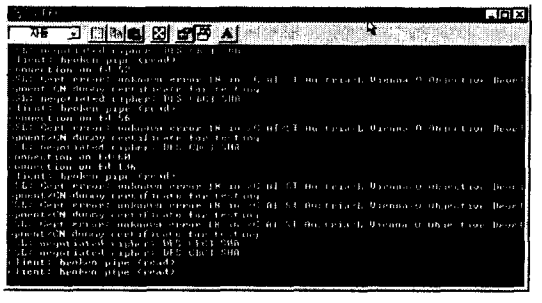

그림 5-4 sslproxy 프로그램의 클라이언트측 동작

6. 결론

국내 인터넷 사용자의 증가와, 전자 상거래 시장의 성장, 그리고 인터넷 뱅킹 서비스의 활성화 등 TCP/IP 프로토콜을 사용한 통신의 증가와 대중화는 주지의 사실이다. 이와 함께 TCP/IP 통신의 보안 문제 해결은 중요한 문제가 되었다. 본 논문에서는 이러한 TCP/IP 통신의 보안 문제 해결의 한 방법인 SSL 프로토콜에 SEED 암호 알고리즘을 적용하여 Proxy 형태의 프로그램으로 구현하였다. 구현 결과 서버와 클라이언트의 통신에서 상호인증과 보안기능을 제공할 수 있었다. 그러나, 구현 결과물인 sslproxy 프로그램은 효율성 향상을 위해 서버쪽 동작에서 Session change 기능 보강 등의 개선이 필요하고 사용자 편의를 위한 인터페이스의 추가도 필요할 것으로 생각된다. 이와 같은 드러난 문제점은 차후의 연구과제로 수행할 계획이다.

본 논문의 결과물인 sslproxy 프로그램은 자체 웹 브라우저 개발, 인터넷 주식 거래 등 전자상거래를 위한 다양한 프로그램에 적용될 수 있으며 TCP/IP 통신에 보안기능을 추가하기 위해서 쉽게 사용될 수 있을 것이다. 본 연구결과가 국내 전자상거래 도입 및 통신을 사용하는 프로그램에 보안기능을 추가하기 위한 참고자료로 활용되기를 바란다.

[참고문헌]

[1] 한국 전산원, "99' 국가 정보화 백서" http://www.nca.or.kr, 1999

[2] [SSL3] A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.

[3] [SSL2] Hickman, Kipp, "The SSL Protocol", Netscape Communications Corp., Feb 9, 1995

[4] [TLS] Tim Dierks, C. Allen. "The TLS Protocol", RFC2246, January 1999.

[5] [SHTTP] E. Rescorla, A. Schiffman, "The Secure HyperText Transfer Protocol", RFC2660, August 1999.

[6] The OpenSSL Project Group, "OpenSSL 0.9.5", 28 Feb 2000

[7] [HTTP] Berners-Lee, T., Fielding, R. and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[8] [TCP] Postel, J., "Transmission Control Protocol," STD 7, RFC 793, September 1981.

[9] KISA, "128비트 블록 암호알고리즘 (SEED) 개발 및 분석 보고서", Dec, 1998

[10] [X509] CCITT. Recommendation X.509: "The Directory - Authentication Framework". 1988.

[11] [HMAC] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, February 1997.

[12] The Apache Project Group, "Apache Web Server 1.3.9", June 1997

[13] C2NET, "SafePassage", http://www.c2net.com

[14] KT, "KSSL", http://www.kt.co.kr

[15] Jang Media Interactive, "SSLProxy", http://crypto.jmi.co.kr