프로토콜 기반 웹 클라이언트-서버 보안 모듈 구현

장 승 주[†] · 한 수 환[††]

요 약

본 논문은 웹 시스템 환경에서 안전한 데이터 전송을 만족하는 Protocol-Based Security Module 구조의 제안과 이를 구현한 내용에 대해서 언급한다. Protocol-Based Security Module 구조는 크게 두 개의 모듈로 구현된다. 하나는 웹 서버에서 동작하는 Web Server Security Module 이고, 다른 하나는 클라이언트에서 동작하는 WinSock Client Security Module 이다. 웹 서버 보안 모듈은 암호된 메시지를 클라이언트에게 전송하고 클라이언트 보안 모듈은 서버로부터 받은 암호화된 메시지를 정상적인 메시지로 변환하여 웹 브라우저에 나타나게 한다. 웹 서버 보안 모듈은 HTML 파일에 대한 암호화 기능과 클라이언트 모듈에서 보낸 암호화된 메시지를 복호화하는 기능을 가지고 있다. 본 논문에서 제안하는 보안 구조는 클라이언트와 서버간에 간단한 모듈의 설치로 안전한 데이터 전송을 보장한다. 본 논문에서 제안하는 클라이언트, 서버 모듈의 구현 내용을 실험하였다.

Implementation of the Secure Web Server-Client Module Based on Protocol Architecture

Seung Ju Jang[†] · Soo Whan Han[††]

ABSTRACT

We implement the PBSM (Protocol-Based Security Module) system which guarantees the secure data transmission under web circumstances. There are two modules to implement for the PBSM architecture. One is Web Server Security Module (WSSM) which is working on a web server, the other is the WinSock Client Security Module (WSCSM) which is working on a client. The WSCSM security module decrypts the encrypted HTML document that is received from the security web server. The decrypted HTML document is displayed on the screen of a client. The WSSM module contains the encryption part for HTML file and the decryption part for CGI (Common Gateway Interface). We also implement the proposed idea at the web system.

키워드 : Protocol-Based Security module, WSCSM, WSSM, 웹 보안 모듈(Web Security Module)

1. 서 론

최근에 많은 사람들이 웹 시스템을 이용하여 중요한 데이터를 전달하고 있다. 웹 환경은 편리성이라는 측면에서 많은 사람들이 쉽게 접근하여 사용하고 있다. 그러나 이러한 편리성의 이면에는 여러 가지 보안상 문제점을 가지고 있다. 편리한 측면으로 인하여 불순한 의도를 가진 사람이 쉽게 시스템에 접근할 수 있는 가능성이 높다는 것이다. 특히 최근에는 많은 금융 시스템들이 웹과 연결되어 사용자들에게 편리한 업무 서비스를 제공하고 있다.

웹 보안과 관련한 연구는 여러 가지 형태로 이루어지고 있다[1, 8, 9, 11]. 그 중에 하나인 네트워크 자체의 보안에 대한 연구가 주류를 이루고 있다[2, 16, 18, 19]. 또한 IPSec 프로토콜과 같은 보안 네트워크 프로토콜의 구현에 대한 연구가 활발히 진행되고 있다. 네트워크 보안으로 많이 사용되고 있는 기법이 방화벽(firewall)이다. 방화벽과 같은 기법은 네트워크를 통한 시스템의 접근을 어렵게 만드는 수단일 뿐 근본적인 보안 해결책으로는 완전한 해법이 아니다[14, 15]. 따라서 최근 들어 네트워크 보안 만으로 부족하다는 인식이 확산되면서 데이터 보안에 대한 연구가 진행되고 있다. 데이터 보안은 주로 네트워크로 흘러가는 데이터에 대한 보안에 초점을 두고 있다. 이러한 데이터 보안 기법의 방법으로 SSL(Secure Socket Layer) 프로토콜이나 S-HTTP 프로토콜을 이용한 웹 시스템이 늘어나고 있다. 그러나 SSL 프로토콜이나 S-HTTP 프로토콜의 단점이 서버에서 클라이언트로 받는 자료에 대한 보안에만 주력하기 때문에 클라이언트에서 서버로 전송되는 자료에 대한 보안

※ 이 논문은 한국과학재단의 해외 Post-doc. 연수지원에 의하여 연구되었음.
† 정 회 원 : 동의대학교 컴퓨터공학과 교수
†† 정 회 원 : 동의대학교 멀티미디어공학과 교수
논문접수 : 2002년 5월 3일, 심사완료 : 2002년 8월 22일

이 취약하다는 것이다[7, 17]. 또한 시스템 환경을 구축하는 데 복잡함과 보안 키(key) 관리에 대한 문제점 등이 따른다 [1, 4, 8, 10, 12]. 이와 같이 현재까지 진행되고 있는 대부분의 연구들이 단편적인 문제 해결로 그치고 있다. 즉 통합 환경에 적절한 시스템 관점에서의 접근은 부족한 실정이다. 본 논문은 이러한 웹 시스템 환경에서 안전한 데이터 전송을 만족하는 Protocol-Based Security Module(PBSM) 구조를 제안한다. PBSM 구조는 분산 시스템 구조에서 프로토콜 중심의 보안 환경을 제공한다. PBSM 구조는 크게 두개의 모듈로 구성된다. 하나는 웹 서버에서 동작하는 Web Server Security Module(WSSM)이고, 다른 하나는 클라이언트에서 동작하는 WinSock Client Security Module(WSCSM) 이다.

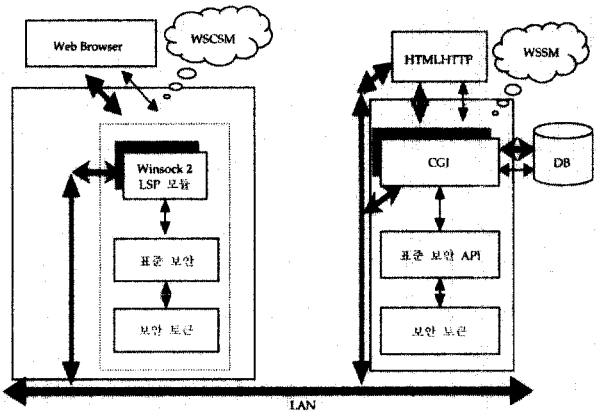
WSSM 모듈은 클라이언트에 암호화된 데이터를 전송하기 위한 모듈과 클라이언트로부터 받은 암호화된 메시지를 받아서 복호화하는 모듈로 나누어진다. 암호화, 복호화 과정에 사용하는 암호 알고리즘은 본 논문에서 제안하는 변형된 m_RSA 알고리즘을 사용한다.

WSCSM 모듈은 WSSM모듈로부터 받은 암호화된 메시지를 복호화 하는 기능, 사용자 개인 신상 정보 등을 암호화하여 서버로 보내는 기능이 있다. WSCSM모듈에서 사용하는 암호화 알고리즘도 WSSM모듈과 동일하게 m_RSA알고리즘을 사용한다. 본 논문에서 제안하는 PBSM구조는 안전한 데이터 전송을 보장하기 위한 웹 환경에 효율적으로 사용이 가능하다.

본 논문의 구성은 제2장 PBSM 보안 웹 시스템 구조, 제3장 WSSM, WSCSM 보안 모듈, 제4장 실험, 제5장 결론의 순으로 언급한다.

2. PBSM 보안 웹 시스템 구조

PBSM 보안 웹 시스템 구조는 다음 (그림 1)과 같다.



(그림 1) PBSM 보안 웹 시스템 구조(굵은 선 : 웹 시스템의 정상적인 자료 흐름, 가는 선 : PBSM 시스템 데이터 흐름)

WSSM 모듈은 웹 서버 모듈과 연동해서 동작하지만 이식의 편리성을 위하여 독립적인 모듈로 존재한다. 웹 서버에서 보안 기능이 필요할 경우는 WSSM모듈을 웹 서버에 설치하는 것으로 충분하다. WSSM 보안 모듈을 적용하기 위해서는 웹 서버에서 제공하는 C API(Application Program Interface)함수를 이용한 모듈의 기능을 이용한다. WSSM 보안 모듈의 구성 요소는 m_RSA알고리즘, m_RSA복호화 알고리즘, SHA-1 해쉬 알고리즘, 전자 서명 기능 등이다.

m_RSA 알고리즘은 RSA 알고리즘을 변형한 알고리즘이다. *m_RSA* 알고리즘은 RSA 알고리즘에서 키 관리 부분을 변경한 알고리즘이다. *m_RSA* 알고리즘은 키 관리 부분을 독립적인 서버를 사용하지 않는다. *m_RSA* 알고리즘은 본 논문에서 RSA알고리즘을 키 관리 부분의 구조를 변경하여 만들었다. *m_RSA* 알고리즘은 키를 웹 서버 자체에서 키 관리 모듈이 있어서 키를 할당하고 관리를 한다. *m_RSA* 알고리즘은 서버에서 클라이언트로 전송되는 HTML 형태의 모든 메시지에 대한 암호화를 적용하기 위한 것이다. *m_RSA* 복호화 알고리즘은 클라이언트에서 CGI 폼에 입력한 암호화된 클라이언트 정보에 대한 복호화를 위한 것이다. SHA-1 해쉬 알고리즘 메시지가 전송 도중 변경되지 않음을 증명하기 위한 것이다. 전자 서명기능은 메시지의 무결성을 보장하기 위한 것이다.

WSCSM 모듈은 Windows Socket2의 Layered Service Provider(LSP)를 기반으로 한다. Windows Sockets2는 network programming interface로서 OSI 계층도에 있어서 전송 계층(transport layer)에 존재한다. 그리고 응용 프로그램 제작자들을 위한 API(Application Program Interface)와 transport stack을 제작하는 제작자들을 위한 SPI(Service Provider Interface)가 제공된다 [11]. WSCSM 모듈은 Winsocket의 Layer Service Provider(LSP) 기능을 이용하여 설계한다.

WSCSM 보안 모듈의 구성 요소는 m_RSA 암호화 알고리즘, m_RSA복호화 알고리즘, SHA-1 해쉬 알고리즘, 전자 서명 기능 등이다. *m_RSA* 암호화 알고리즘은 CGI 폼에 입력된 클라이언트 정보의 보안을 위한 것이다. M_RSA 복호화 알고리즘은 서버에서 클라이언트로 전송되는 HTML 형태의 모든 메시지에 대한 복호화를 위한 것이다. SHA-1 해쉬 알고리즘은 메시지가 전송도중 변경되지 않음을 증명하기 위한 것이다. 전자 서명기능은 클라이언트와 서버 간에 메시지 전송도중에 메시지의 무결성을 보장하기 위한 것이다. *m_RSA* 알고리즘은 본 논문에서 제안하는 변형된 RSA 알고리즘이다. *m_RSA* 알고리즘은 RSA 알고리즘에서

키 관리 부분을 키 서버로부터 키를 받지 않고 웹 서버에서 키를 할당하도록 하였다. 즉, *m_RSA* 알고리즘은 키 관리 부분을 위해서 독립적인 키 서버를 사용하지 않는다. WSSM 서버가 웹 서버와 클라이언트에게 키 분배의 역할을 담당한다.

3. WSSM, WSCSM 보안 모듈

3.1 PBSM 시스템 설계

본 논문의 PBSM 시스템 구조에서는 두 가지 종류의 메시지를 다룬다. 하나는 일반 메시지이고 다른 하나는 보안 메시지이다. 일반 메시지는 일반적인 컴퓨터 환경에서 사용하는 ASCII 데이터를 취급하는 경우를 말한다. 보안 메시지는 텍스트 데이터를 이용해서 보안처리 과정을 통해 만들어진 데이터를 말한다. 메시지와 관련된 사항들을 다음 (정의 1)과 같이 정의한다.

(정의 1) 메시지(message)

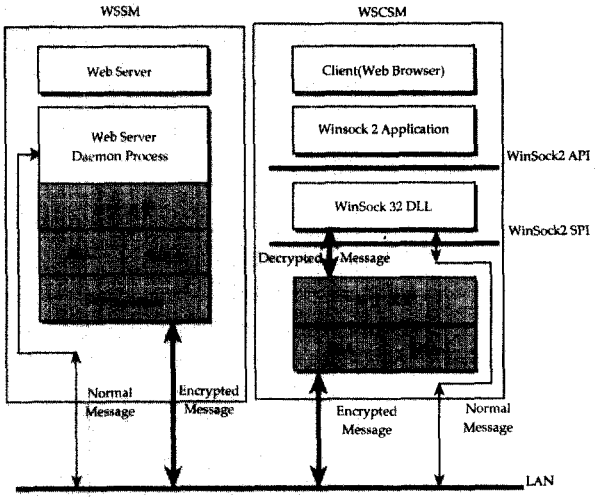
1. Type t를 갖는 기본 메시지는 $m : t$ 로 표기한다.
2. 만약 $m_1 : t_1$ 과 $m_2 : t_2$가 메시지이면 메시지의 연결(concatenation) $m_1 m_2 : t_1 t_2$도 메시지이다.
3. 만약 $f : t_1 \times \cdots \times t_n \; t$가 cryptographic 함수이고 $m_i : t_i$ 가 메시지이면 $m_1, \cdots, m_n$에 대한 f 함수의 적용인 $f(m_1, \cdots, m_n) : t$ 또한 메시지이다.

보안 메시지의 경우는 message equality가 성립한다. 즉, $dec(enc(m, k), k) \cong m$ ($\cong$ 심볼은 변환을 나타내는 기호, dec : 복호화 함수, enc : 암호화 함수, m : 메시지) 따라서, 만약 $m_1 \cong m_2$ 이면 $m_1 = m_2$이다.

WSSM 모듈과 WSCSM 모듈 사이의 데이터 흐름은 (그림 2)와 같다. 클라이언트 사용자가 웹 브라우저를 이용하여 웹 서버에 접속할 경우, WSCSM 모듈에서 Winsock 2 application을 실행하게 된다. WSCSM 모듈에서 Winsock 2 모듈은 Winsock 32 DLL을 실행하게 된다. Winsock 32 DLL 모듈은 Winsock 2 SPI 인터페이스를 통해서 실제적인 보안 모듈을 구동하게 된다. 사용자가 요청한 자료나 명령어 등이 보안 모듈에서 보안 자료 또는 보안 명령어로 변환되어 WSSM 모듈로 전달된다.

WSCSM 모듈로부터 자료를 넘겨받은 WSSM 모듈은 보안 모듈에서 사용자가 요청한 내용을 분석하기 전에 복호화 작업을 수행한다. 복호화 작업을 통해서 정상적인 데이터로 변환이된 후에 web server 모듈은 정상적인 문자 처리 과정에 의하여 사용자의 요구사항을 처리한다. 사용자가 요청한 자료를 WSCSM 모듈로 전송하기전에 WSSM 모듈에서 보안처리 과정을 거치게 된다. WSSM 보안 모듈에서 보안처리된 자료는 WSCSM 시스템으로 전송된다. WSSM 모듈로부터 자료를 넘겨 받은 WSCSM 모듈은 복호화 과정을 통해서 정상적인 메시지로 만든다. 정상적인 메시지는 Winsock 2 SPI 인터페이스를 통해서 Winsock 2 DLL모듈로 전달된다. Winsock 2 DLL 모듈은 정상적인 메시지를 받아서 사용자의 웹 브라우저로 데이터를 출력하게 된다.

WSSM 모듈과 WSCSM 모듈 내의 *m_RSA* 암호화 알고리즘에 사용되는 공개키와 비 공개키의 길이는 512비트로 고정하여 사용한다. WSSM 모듈과 WSCSM 모듈 간의 데이터 흐름을 그림으로 표시하면 (그림 2)와 같다.



(그림 2) WSSM, WSCSM 모듈 내부 구조(가는 선 : 웹 시스템의 정상적인 자료 흐름, 굵은 선 : PBSM 시스템 데이터 흐름)

3.2 WSSM 보안 모듈

WSSM 모듈은 WSCSM 모듈의 데이터 요청이 발생하면 이 요청에 대한 응답을 보낸다. 서버의 WSSM 보안 모듈은 WSCSM 모듈에서 요청한 HyperText Markup Language (HTML) 파일을 WSSM 보안 모듈 내에서 보안 처리하여 WSCSM 모듈로 전송한다. WSSM 보안 모듈은 HTML 파일에 대한 요청이 발생하면 이 파일의 보안 처리를 위하여 웹 서버의 API 함수를 거치도록 한다.

WSSM 보안 모듈은 HTML 파일에 대한 암호화(Encryption)기능($E_{WSSM}(file(x))$과 WSCSM 모듈에서의 암호화된 메시지에 대한 복호화 기능($D_{WSSM}(file(x))$을 가지고 있다. WSSM 보안 모듈의 암호화 기능은 *m_RSA* 암호화 알고리즘을 사용하여 WSCSM 모듈의 데이터 요청에 대한 암호화된 응답($m_RSA_{WSSM}(file(x), key(i))$과 디지털 서명기능

($Dig_{WSSM}(file(x))$)을 갖는다. WSSM모듈의 *m_RSA* 암호화 알고리즘($E_{WSSM}file(x) = m\_RSA_{WSSM}(file(x), key(i))$)은 HTML 파일의 내용을 한 문자 단위로 읽어서 키 값으로 암호화 후 암호화된 내용($E_{WSSM}file(x)$)을 클라이언트의 WSCSM 모듈로 전송한다. 암호화된 메시지($E_{WSSM}file(x)$)는 네트워크를 통해 클라이언트로 전송되는 중간에 공격자(attacker)에 의해 가로채기 당하더라도 메시지의 내용 자체가 보안 처리되어 있으므로 복호화가 어렵다. 또한 공격자가 메시지를 변조하여 전송할 경우에 디지털 서명 알고리즘($Dig_{WSSM}(file(x))$)을 이용하여 메시지 변조 여부를 확인할 수 있다. 그리고 WSCSM 모듈을 통해서 보낸 메시지에 대해 공개/비공개 키를 이용하여 부인하지 못하도록 한다.

그리고 WSSM 모듈은 클라이언트로부터 CGI(Common Gateway Interface) 인터페이스를 이용해서 받은 메시지를 복호화($D_{WSSM}(file(x))$) 후 CGI 프로그램 처리가 정상적으로 이루어지도록 설계한다. (알고리즘 1)은 WSSM 보안 모듈의 동작 과정을 보여준다.

$$\text{Input}: MSG_{IM} = \begin{cases} HTTP_{HEAD} + m\_RSA\ (E(m_1 : t)), \\ \quad HTTP_{HEAD} \in WSSM \\ HTTP_{HEAD} + m_1 : t,\ HTTP_{HEAD} \notin WSSM \end{cases}$$

$$\text{Output}: MSG_{TM} = \begin{cases} HTTP_{HEAD} + E(m_2 : t), \\ \quad HTTP_{HEAD} \in WSSM \\ HTTP_{HEAD} + m_2 : t,\ HTTP_{HEAD} \notin WSSM \\ HTTP_{HEAD},\ HTTP_{HEAD} \notin WSSM, \\ \quad \text{CGI 데이터인 경우} \end{cases}$$

Algorithm
begin
WSSM 1 : MSG_{IM} 중에서 $HTTP_{HEAD}$를 분리
WSSM 2 : If $HTTP_{HEAD} \in$ WSSM module **begin**
WSSM 3 : $m_1 : t := m\_RSA$ (E(m1 : t), D)
WSSM 4 : $m_1 : t$ 내용을 분석
WSSM 5 : **if** $m_1 : t \in$ CGI **begin**
WSSM 6 : write($m_1 : t$) ;
WSSM 7 : Ack($HTTP_{HEAD}$, wscsm), return ;
WSSM 8 : **end**
WSSM 9 : $m_2 : t := read(m_1 : t)$
WSSM 10 : $E(m_2 : t) := m\_RSA\ (m_2 : t)$
WSSM 11 : send($HTTP_{HEAD} + E(m_2 : t)$)
WSSM 12 : return ;
WSSM 13 : **end**
WSSM 14 : $m_4 : t := read(m_3 : t)$
WSSM 15 : send ($HTTP_{HEAD} + m_4 : t$)
end.

(알고리즘 1) WSSM 보안 모듈의 동작

3.3 WSCSM 보안 모듈

클라이언트(WSCSM)는 웹 브라우저를 이용해서 웹 서버(WSSM)에게 HTML 문서를 요청한다. 보안 모듈이 설치되어 있지 않을 경우 클라이언트의 웹 브라우저에 암호화된 데이터($E_{WSSM}(file(x))$)가 보여진다. 클라이언트(WSCSM) 보안 모듈은 보안 웹 서버(WSSM)로부터 받은 HTML 문서에 대한 복호화 기능을 갖는다. WSCSM 보안 모듈은 WSSM으로부터 받은 암호된 메시지($E_{WSSM}(file(x))$)를 정상적인 메시지($D_{wscsm}(E_{WSSM}(file(x)))$)로 변환하여 웹 브라우저에 나타나게 한다. WSCSM 보안 모듈은 암호화된 HTML 문서를 복호화한다.

LSP모듈은 UDP(User Datagram Protocol) 를 통해 들어온 패킷을 읽어들인다. 이 패킷 $P_{lsp}(x)$의 구성은 $HTTP_{head}$(정상적인 HTTP 헤더)와 $E_{WSSM}(file(x))$(암호화된 메시지)로 구성된다. HTTP헤더($HTTP_{head}$)는 암호화 작업없이 WSCSM 보안 모듈로 전송을 한다. 이유는 HTTP헤더($HTTP_{head}$)가 변경될 경우 WSCSM 보안 모듈에서 패킷의 구성 정보를 알 수 없기 때문이다. HTTP 헤더($HTTP_{head}$)를 제외한 메시지 부분($E_{WSSM}(file(x))$은 WSSM 서버에서 *m_RSA* 암호화 알고리즘으로 암호화($m\_RSA_{WSSM}(file(x), key(i))$)되어 있으므로 복호화 과정($m\_RSA_{wscsm}(file(x), key(i))$)을 거치게 된다. LSP 모듈에서 HTML 문서($HTTP_{body}$)에 대한 복호화는 클라이언트의 비공개 키 값($key(i)$)으로 수행된다. 복호화된 HTTP 문서는 API를 통하여 브라우저에 정상적인 메시지로 보여진다. 만일 HTML 문서를 복호화 한 후 메시지의 내용이 깨어지거나 이상한 문자가 나타날 경우는 메시지가 네트워크를 통한 전송과정 중 이상이 발생했음을 의미한다.

그리고 클라이언트 보안 모듈(WSCSM)은 보안 웹 서버(WSSM)와 일반 서버로 데이터가 각각 전달될 수 있도록 URL 구분기능을 갖는다. 즉, 보안 사이트와 일반 사이트의 연결은 URL DB 기능을 이용하여 수행된다. 일반 데이터가 보안 모듈을 거치는 경우나 보안 데이터가 일반 모듈을 거치는 경우를 방지하기 위해서 클라이언트에서 서버로 데이터를 보내기 전에 HTTP 헤더의 URL과 WSSM 서버로 등록된 URL(URL DB)을 비교한다. 클라이언트에서 서버로 보내고자 하는 데이터가 WSSM 서버로 URL DB에 등록된 경우에는 클라이언트의 보안 모듈(WSCSM)을 거치도록 하며, URL이 WSSM 서버로 URL DB에 등록되지 않은 경우는 일반적인 데이터 전송과정을 거친다. 이 과정은 (그림 1)에 나타나 있다.

마지막으로 사용자가 입력한 정보에 대한 암호화 기능이다. 클라이언트(WSCSM)는 웹 서버(WSSM)로부터 일방적으로 정보를 제공받는 경우가 대부분이지만 사용자 인증 관련 정보는 서버로 전송된다. 사용자 인증 정보는 CGI 폼

(FORM) 형식으로 데이터를 입력받아서 서버(WSSM)의 CGI 처리 모듈로 전송된다.

클라이언트 WSCSM 모듈은 사용자가 폼에 관련 데이터를 입력하고 확인 버튼을 누르면 데이터가 네트워크로 전송되기전에 암호화(m_RSA_{WSCSM})를 수행하여 암호화된 데이터($E_{WSCSM}(file(x))$)가 서버(WSSM)로 전송된다. 사용자 인증 정보를 서버로 전송하는 경우에 보안 데이터 처리를 함으로써 안전한 데이터 전송을 보장한다. (알고리즘 2)는 WSCSM 보안 모듈의 동작 과정을 보여준다.

$$\text{Input}: MSG_{IM} = \begin{cases} HTTP_{HEAD} + m_1 : t, & HTTP_{HEAD} \notin \{WSSM, WSCSM\} \\ HTTP_{HEAD} + m\_RSA(E(m_1 : t)), & HTTP_{HEAD} \in \{WSSM, WSCSM\} \end{cases}$$

$$\text{Output}: MSG_{TM} = \begin{cases} HTTP_{HEAD} + E(m_2 : t), & HTTP_{HEAD} \in WSSM \\ HTTP_{HEAD} + m_2 : t, & HTTP_{HEAD} \notin WSSM \end{cases}$$

Algorithm
begin
WSCSM 1 : get(URL)
WSCSM 2 : Flag : = check(URL) ;
WSCSM 3 : **while** ((flag = TRUE) and No More MSG)) **begin**
WSCSM 4 : E (m_1 : t) : = m_RSA (m_1 : t, E)
WSCSM 5 : send ($HTTP_{HEAD}$ + E (m_1 : t))
WSCSM 6 : receive ($HTTP_{HEAD}$ + E (m_2 : t))
WSCSM 7 : m_2 : t : = m_RSA(E (m_2 : t), D)
WSCSM 8 : Display (m_2 : t), **return** ;
WSCSM 9 : **end**
WSCSM 10 : **while** (No More MSG) **begin**
WSCSM 11 : send ($HTTP_{HEAD}$ + m_1 : t)
WSCSM 12 : receive ($HTTP_{HEAD}$ + m_2 : t)
WSCSM 13 : Display (m_2 : t)
WSCSM 14 : **end**
end.

(알고리즘 2) WSCSM 보안 모듈의 동작

4. 실 험

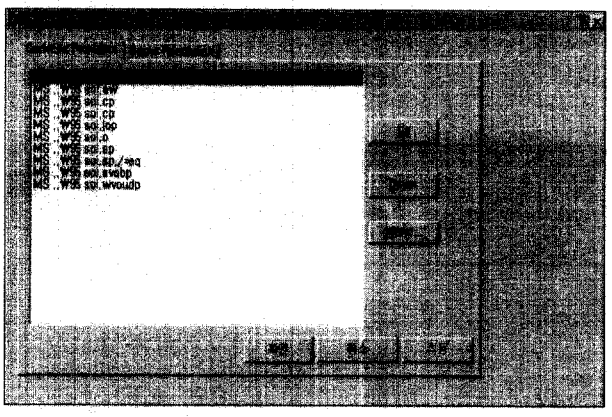
4.1 LSP 보안 모듈의 설치

WSSM과 WSCSM 보안 모듈을 실험하기 위해 클라이언트의 LSP를 Install 한 후 서버에서 클라이언트에게 전송된 암호화된 메시지가 클라이언트의 웹 브라우저에 정상적으로 나타나는지를 살펴본다.

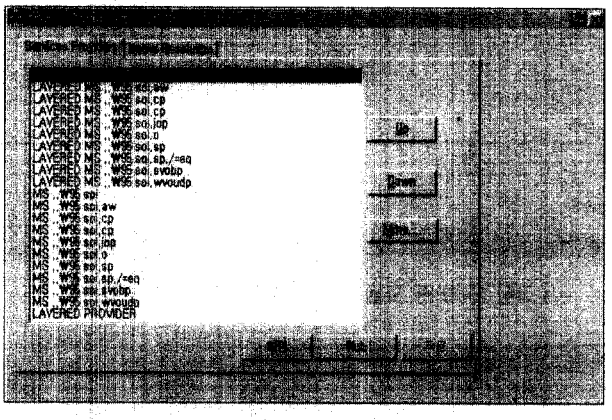
클라이언트에서 현재 LSP가 실행되고 있는지를 살펴보기 위해 (그림 3)과 같이 SPORDER.EXE를 실행한다.

(그림 3)에서는 기존 9개의 base 프로토콜에 LSP체인이 설치된 것을 알 수 있다. 다시 SPORDER.EXE를 실행하면 (그림 4)와 같다. 클라이언트의 보안 모듈을 c:\ 디렉토리에 복사를 한 후 inst_lsp프로그램으로 실행한다.

(그림 4)에서 새로운 체인이 base 프로토콜에 연결되어 보안 모듈의 기능을 수행하도록 설치가 된것을 확인 할 수 있다.



(그림 3) SPORDER.EXE 실행



(그림 4) inst_lsp 실행 후 SPORDER.EXE 실행

4.2 보안 모듈 실험

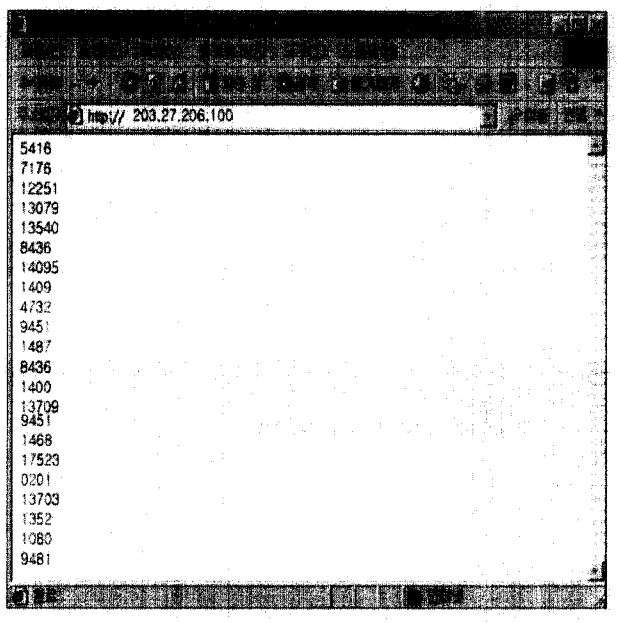
서버와 클라이언트의 보안 모듈 실험은 두 가지의 경우에 대하여 실험을 한다. 첫째, 서버에서 암호화된 데이터가 클라이언트로 전송되는 경우 클라이언트에서 정상적으로 복호화 되어 클라이언트의 웹 브라우저에 복호화 된 데이터가 나타나는 것과, 둘째 CGI 폼에 데이터를 입력한 후 클라이언트에서 서버로 암호화된 데이터를 보내고 서버는 클라이언트로부터 입력된 암호화된 데이터를 복호화 후 그 결과를 CGI프로그램으로 넘겨주는 것을 실험한다.

4.2.1 서버에서 클라이언트로 암호화된 데이터 전송

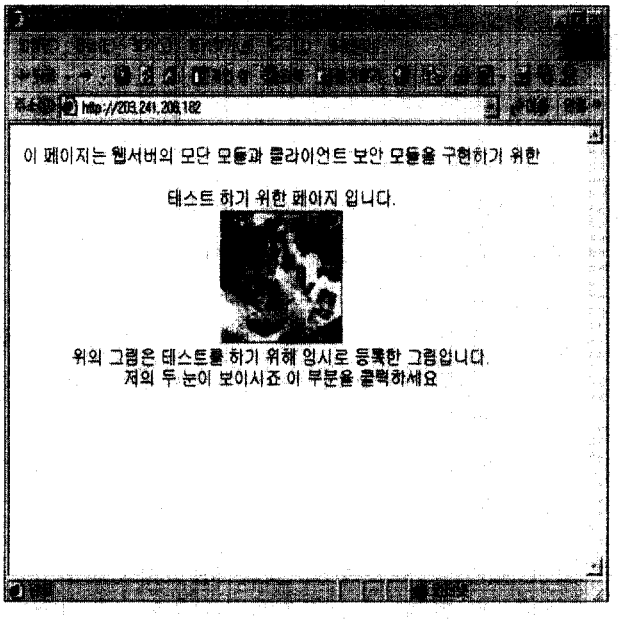
클라이언트 LSP 보안 모듈이 없는 경우에 클라이언트의 웹 브라우저에는 암호화된 데이터가 (그림 5)와 같이 그대로 브라우저에 나타난다.

보안모듈이 정상적으로 설치되면 웹 서버에서 클라이언

트로 전송한 암호화된 메시지는 클라이언트의 브라우저에 복호화 되어 정상적인 메시지로 나타난다. 이미지의 경우 헤더의 내용중 이미지에 관련된 정보를 포함하고 있기 때문에 보안 모듈에서 이미지 부분에 대한 처리를 한 후 정상적인 데이터가 출력이 되도록 한다. (그림 6)은 정상적인 복호화 처리를 하고 난 후에 웹 브라우저에 나타난 메시지이다.



(그림 5) 보안 모듈이 없는 클라이언트 웹 브라우저



(그림 6) 정상적인 클라이언트 보안 모듈의 동작

그리고 URL 필터링 기능으로 클라이언트가 보안 웹 서버에 접속하면 정상적으로 복호화 후 결과를 브라우저에 나타내고 보안 웹 서버가 아닌 경우 서버로 입력된 데이터는 보안 모듈을 거치지 않고 바로 웹 브라우저에 나타나도록 한다.

4.2.2 CGI 폼의 형태로 클라이언트에서 서버로 암호화된 데이터 전송

CGI폼 형태의 데이터 전송은 클라이언트에서 서버로 데이터를 입력하기 위한 폼은 HTML 파일로 작성되어 서버에서 암호화된 데이터로 클라이언트에 넘어온 후 클라이언트의 보안 모듈이 복호화 후 (그림 7)과 같이 브라우저에 정상적인 형태로 나타난다.

(그림 7) CGI 폼 형태의 데이터 입력

(그림 7)에 나타난 폼에 데이터를 입력한 후 확인 버튼을 누르면 폼의 HTML 소스에 링크된 CGI프로그램과 연결되어 데이터가 서버로 전송된다. 이 과정에서 폼에 입력한 데이터는 클라이언트의 보안 모듈을 통해 암호화되어 서버로 전송된다. 서버에서는 암호화된 데이터를 복호화 한 후 입력된 데이터에 해당되는 CGI 변수에 복호화한 값을 넘겨주어 정상적인 CGI프로그램이 동작되도록 하고 클라이언트에게 정상적으로 메시지가 처리되었음을 알리는 메시지를 보내줌으로써 CGI폼의 데이터 처리를 마치게 된다.

5. 결 론

본 논문은 웹 시스템 환경에서 안전한 데이터 전송을 만족하는 Protocol-Based Security Module(PBSM) 구조를 제안한다. PBSM 구조는 분산 시스템 구조에서 프로토콜 중

심의 보안 환경을 제공한다. PBSM 구조는 크게 두 개의 모듈로 구성된다. 하나는 웹 서버에서 동작하는 Web Server Security Module(WSSM)이고, 다른 하나는 클라이언트에서 동작하는 WinSock Client Security Module(WSCSM)이다.

구현된 WSSM, WSCSM 보안 모듈은 공개키를 기반으로 기밀성, 무결성의 기능을 제공한다. WSSM, WSCSM 보안 모듈은 보안 서버 모듈과 보안 클라이언트 모듈의 역할을 수행한다. WSSM 보안 모듈은 WSCSM 보안 모듈에서 데이터 요청이 발생하면 이 요청에 대한 응답을 보낸다. WSSM 보안 서버 모듈은 WSCSM 보안 클라이언트 모듈에서 요청한 HTML 파일이나 데이타를 보안 처리하여 WSCSM 보안 모듈로 전송한다. WSSM 보안 모듈은 HTML 파일에 대한 암호화 기능과 WSCSM 보안 모듈에서의 CGI 응답에 대한 복호화 기능을 갖고 있다.

WSCSM 보안 모듈은 WSSM 보안 모듈로부터 받은 HTML 문서에 대한 복호화 기능을 갖는다. WSCSM 보안 모듈은 WSSM 보안 모듈로부터 받은 암호된 메시지를 정상적인 메시지로 변환하여 웹 브라우저에 나타나게 한다.

그리고 WSCSM 보안 클라이언트 모듈은 WSSM 보안 웹 서버 모듈과 일반 웹 서버로 데이터가 각각 전달될 수 있도록 URL에 대해서 보안 웹 서버와 일반 웹 서버를 구분할 수 있도록 한다. 즉, 보안 사이트와 일반 사이트에 대한 URL DB를 구축해서 정상적인 데이터 전송이 가능하도록 한다. 본 논문에서 제안하는 보안 구조는 클라이언트와 서버 간에 간단한 모듈의 설치로 안전한 데이터 전송을 보장한다. 본 논문에서 제안하는 WSSM, WSCSM 보안 모듈을 실제 웹 시스템에서 구축하여 실험하였다.

참 고 문 헌

[1] 김병천, 이경호, 박성준, 원동호, "전자 서명 방식의 구현 및 성능분석", 제4회 통신정보 합동학술대회논문집, pp.662-666, 1994.

[2] 박일환, 장청룡, 원동호, "중명이 가능한 전자서명기법", 한국통신정보보호학회논문지, 제4권 제1호, pp.41-50, 1994.

[3] 염흥렬, "전자 서명 방식 고찰", 한국통신정보보호학회 학회지, 제3권 제2호, pp.7-18, 1993.

[4] Debaty, P., Caswell, D., "Uniform Web presence architecture for people, places, and things," IEEE Personal Communications, Vol.8, Issue.4, pp.46-51, Aug., 2001.

[5] W. Diffie and M. E. Hellman, "New directions in cryptography," IEEE Trans. on Information Theory IT-22, No.6, pp. 644-654, 1976.

[6] D. L. Dill., "The Murpi verification system," In Computer Aided Verification 8th International Conference, pp.390-403, 1996.

[7] A. O. Frier, P. Karlton, and P. C. Kocher, The SSL protocol version 3.0, draft-ietf-tls-ssl-version3-00.txt, November, 1996.

[8] Warwick Ford, Michael S. Baum, Secure Electronic Commerce : Building the Infrastructure for Digital Signatures and Encryption, Prentice Hall, 2000.

[9] Gutzmann, K., "Access control and session management in the HTTP environment," IEEE Internet Computing, Vol.5 Issue.1, pp.26-35, Jan.~Feb., 2001.

[10] Niemeyer, R. E., "Using Web technologies in two MLS environments : a security analysis," Computer Security Applications Conference, 1997. Proceedings., 13th Annual, pp. 205-214, 1997.

[11] K. Nyberg and R. A. Rueppel, "Message recovery for signature scheme based on the discrete logarithm problem," Eurocrypt'94 Proceedings, Springer-Verlag, 1995.

[12] Liu, S., Sullivan, J., Ormaner, J., "A practical approach to enterprise it security," IT Professional, Vol.3, Issue.5, pp.35-42, Sep.~Oct., 2001.

[13] S. C. Pohlig and M. E. Hellman, "An improved algorithm for computing logarithm over GF(p) and its cryptographic significance," IEEE Trans. on Information Theory IT-24, No.5, pp.106-110, 1978.

[14] R. L. Rivest, A. Shamir and L. Adleman, "A method of obtaining digital signature and public key cryptosystem," ACM Communication 21, No.2, pp.120-126, 1978.

[15] Rubin, A. D., Geer, D. E., Jr., "A survey of Web security," Computer, Vol.31, Issue.9, pp.34-41, Sept., 1998.

[16] Lincoln D. Stein, Web Security : A Step-by-Step Reference Guide, Addison-Wesley, 1999.

[17] Wangham, M. S., Lung, L. C., Westphall, C. M., Fraga, J. S., "Integrating SSL to the JaCoWeb security framework : project and implementation," Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on, pp.779-792, 2001.

[18] Donna Woouteiler, Web Security ; A Matter of Trust, O'Reilly & Associates, 1997.

[19] Younglove, R. W., "IP security : what makes it work?," Computing & Control Engineering Journal, Vol.12, Issue.1, pp.44-46, Feb., 2001.

장 승 주

e-mail : sjjang@dongeui.ac.kr

1985년 부산대학교 계산통계학과(전산학) 학사

1991년 부산대학교 계산통계학과(전산학) 석사

1996년 부산대학교 컴퓨터공학과 박사

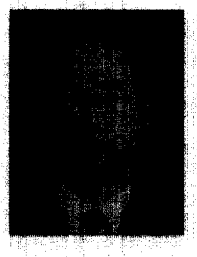
1987년~1996년 한국전자통신연구원 시스템 S/W연구실

1993년~1996년 부산대학교 시간강사

2000년~2002년 University of Missouri at Kansas City, visiting professor

1996년 현재 동의대학교 컴퓨터공학과 부교수

관심분야 : 운영체제, 분산시스템, Active Network, 시스템 보안



한 수 환

e-mail : swhan@dongeui.ac.kr

1986년 연세대학교 전자공학과 졸업 (공학사)

1990년 미 Florida Institute of Technology 전기전자공학과(석사)

1993년 미 Florida Institute of Technology 전기전자공학과(박사)

1994년~1997년 관동대학교 컴퓨터공학과 조교수

1997년~현재 동의대학교 멀티미디어공학과 부교수

관심분야 : Digital Signal & Image Processing, Pattern Recognition, Fuzzy Logic & Neural Networks