이기종 시스템에서 안전한 데이타 전송을 보장하는 웹 보안 모듈의 설계 및 구현

(Design and Implementation of Web Security Module for a Safe Data Transmission in Heterogeneous Systems)

김 기 성 † 김 광 †† 허 신 †††
(Ki Sung Kim) (Kwang Kim) (Shin Heu)

요 약 본 논문은 이종의 시스템에서 안전한 데이타 전송을 만족하는 웹 보안 모듈을 구현한 내용에 대해서 언급한다. 웹 시스템을 통해 사용자에게 사용의 편리함과 많은 정보를 주었고 웹 서비스 사업이 활발해졌으나 웹 시스템의 한계로 인해 보안상의 취약점인 데이타 유출에 의한 손실을 가져오게 되었다. 이에 많은 대응들이 있지만 좀더 효과적이고 편리한 데이타 보안을 위해 서버 클라이언트 보안 모듈을 구현하였다. 제안한 보안 모듈은 크게 두 개의 모듈로 구현된다. 하나는 웹 서버에서 동작하는 서버 보안 모듈이고, 다른 하나는 클라이언트에서 동작하는 클라이언트 보안 모듈이다. 본 논문에서 제안하는 보안 구조는 클라이언트와 서버간의 간단한 모듈의 설치로 안전한 데이타 전송을 보장한다. 속도에 민감한 웹 서버와 브라우저 사이의 빠른 암호화 통신을 위한 대칭 암호화 방식인 Triple-DES를 사용하였으며 키 관리 문제를 해결하기 위한 Diffie-Hellman 키(key) 교환 알고리즘을 적용하였다. 이렇게 함으로써 대칭 암호화에서 발생하는 키 분배 및 관리문제와 속도문제를 해결했다. 또한 DH방식의 알고리즘을 사용함으로써 서버와 클라이언트 사이의 안전한 데이타 보안을 위한 인증문제를 해결하였다.

키워드 : 웹 보안 모듈, Diffie-Hellman의 키 교환 알고리즘, 데이타 보안

Abstract This thesis is written with web security module for safe data transmission between heterogeneous systems(ex. OS). Web system has allowed users to have great convenience and a lot of information. Though web service business has been progressed much, because of the limitation of it's own system, lots of loss, derived from data spillage which is the weakest point of security, has also followed. Suggested security module is realized by two module. One for server security module for web server, the other is client security module for client. The security structure, suggested on this thesis guarantee safe data transmission by only simple installation of modules in clients and servers. For speed sensitive transmission between web server and browser, Triple-DES, symmetric encryption system suitable for fast encryption communication, is adapted. To solve problems caused from key management, Diffie-Hellman's key exchange algorithm is adapted. By this method, all symmetric encryption troubles from key distribution and management, speed could be work out a solution. And Diffie-Hellman type algorithm secures Authentication for safe data protection.

Key words : Web security module, Diffie-Hellman's key exchange algorithm, Data security

1. 서 론

최근 인터넷 사용이 보편화 되고, 웹을 이용한 광고, 전자출판, 사이버 쇼핑, 인터넷 뱅킹 등 다양한 서비스가 네트워크를 통해 제공되면서 웹 보안에 대한 필요성이 증가하고 있다. 현재 대부분의 인터넷 쇼핑몰을 이용하는 사용자는 구매 물품에 대한 지불을 위해 신용카드를 사용하게 되는데, 이때 신용카드 번호 및 개인 정보가 불법적인 제3자에게 유출되거나 위 변조될 가능성이 점차 커지고 있다. 이러한 문제들은 인터넷의 근간이 되는 TCP/IP와 웹 프로토콜인 HTTP가 데이타에 대한 보안 서비스를 제공하지 않는데서 발생된다고 볼 수 있다. 이

† 정 회 원 : 한양대학교 컴퓨터공학과
bkchoi@cse.hanyang.ac.kr
†† 정 회 원 : (주)스포티누스
kkim@cse.hanyang.ac.kr
††† 종신회원 : 한양대학교 컴퓨터공학과 교수
shinheu@cse.hanyang.ac.kr
논문접수 : 2005년 9월 1일
심사완료 : 2005년 10월 11일

에 따라 인터넷을 통해 전송되는 모든 데이타에 대해 무결성, 기밀성, 사용자 인증, 부인봉쇄, 접근 통제, 보안 감사 등의 보안 서비스를 제공하여 안전하고 편리한 인터넷 환경을 구축하기 위한 노력이 활발하게 진행되고 있다. 보안 서비스를 효과적으로 제공하기 위한 인터넷 보안(Security)은 매우 포괄적인 의미를 지니는데, 일반적으로 시스템 보안과 네트워크 보안 크게 두 부분으로 나뉘어 다양한 보안 솔루션들이 개발되고 있는 상황이다.

웹 보안과 관련한 연구는 여러 가지 형태로 이루어지고 있다[1,2]. 이와 더불어, IPSec 프로토콜과 같은 보안 네트워크 프로토콜의 구현에 대한 연구가 활발히 진행되고 있다. 네트워크 보안으로 많이 사용되고 있는 기법이 바로 방화벽(firewall)이다. 방화벽과 같은 기법은 네트워크를 통한 시스템의 접근을 어렵게 만드는 수단일 뿐 근본적인 보안 해결책으로는 완전한 해법이 아니다[3,4]. 따라서, 최근 들어 데이타 보안에 대한 연구가 진행되고 있다. 데이타 보안은 주로 네트워크로 흘러가는 데이타에 대한 보안에 초점을 두고 있다. 이러한 데이타 보안기법 중에서 SSL(Secure Socker 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 대해서는 보안이 취약하다[5]. 이와 더불어, 시스템 환경을 구축하는데 복잡함과 키(key) 관리에 대해 문제점 등이 따른다[1,6]. 웹 보안 문제를 해결하기 위해서 암호화 기법을 이용한 여러 방법들이 제안되고 있다. 본 논문에서는 웹 브라우저에 로드 되는 웹 페이지에 자바애플릿이 같이 로드되도록 만들고, 웹 서버에서 각 웹 어플리케이션의 출력부분에 자바 암호화 서버를 연동함으로써, 중요 데이타가 암호화 되어 전송될 수 있도록 하였다.

이에 대하여 2장에서는 본 논문의 기술이론들 및 관련 연구를, 3장에서는 제안된 보안 시스템 모듈에 관해 기술한다. 그리고, 4장에서는 제안된 보안 시스템 모듈을 기반으로 실험 및 분석에 관해 보이며, 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

2.1 인터넷 보안

웹은 인터넷을 통해 제공되는 가장 대표적인 서비스이다. 이러한 웹은 전자상거래분야와 밀접한 관련이 있기 때문에, 웹 보안은 안전한 전자상거래를 실현하기 위해 반드시 제공되어야 한다. 먼저 보안 서비스를 제공하기 위한 범위를 정의할 필요가 있다. 웹 서비스는 그림 1에서 보는 바와 같이 HTTP 프로토콜을 이용하는 클라이언트(웹 브라우저)와 웹 서버 어플리케이션을 통해 제공되는데, 이러한 어플리케이션들은 TCP/IP 네트워크를 기반으로 동작한다. 이것은 크게, 서비스를 제공하는 어플리케이션 부분과 데이타 전송 기능을 하는 TCP/IP 네트워크 부분으로 구분할 수 있다.

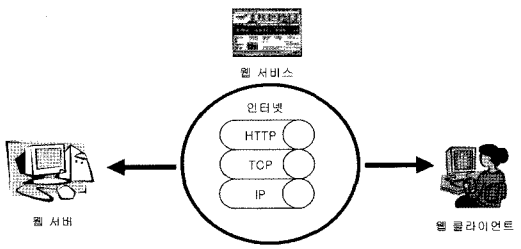

그림 1 인터넷을 이용한 웹 서비스 환경

대부분의 보안 기술은 어플리케이션 또는 네트워크 부분에서 보안 서비스를 제공하기 위한 접근 방식을 취하고 있다. 즉, 네트워크를 안전하게 만들거나 또는 어플리케이션을 안전하게 만드는 것이다.

2.2 관련기술

안전한 HTTP 통신을 위해 많은 방법들이 제안 되었으며 실제로 웹 환경에서 사용되어 지고 있다. 표 1에서 볼 수 있는 내용과 같이 HTTP 통신 방법들을 분류한다.

네트워크 계층 보안과 응용계층 보안 과연 어느 것이 더 우수한 것인가를 단언하기는 매우 어렵지만 시스템

표 1 안전한 HTTP 통신방법

적용계층	적용방법	적용예제
Application Protocol Layer	HTTP 프로토콜이 키 관리, 암호화, 서명 등의 작업을 할 수 있도록 프로토콜의 헤더를 수정한다.	S-HTTP
Session Layer	수정된 트랜스포트 API의 상위에서 구현되며, 양단간 인증 및 기밀성을 보장한다.	Secure Socket Layer(SSL)
Transport Layer	IPv6, PPTP, SSH와 같은 터널링 프로토콜을 사용한 TCP 접속에 대한 안정성을 제공한다.	PPTP SSH VPN
Application Layer	TCP 레벨의 소켓 API나 HTTP 프로토콜을 변경하지 않으면서 다른 어플리케이션 프로그램을 사용하여 안전한 TCP/IP 통신을 수행한다.	PGP

의 사용목적에 따라, 어떤 방식이 더 우수한 것이지 살펴 볼 수 있다. 일반적으로 '우수하다'고 하는 것은 시스템 차원의 절대 적인 보안 평가 보다는 사용자 입장의 상대적인 편이성, 구현의 용이성, 성능 및 사양 등을 의미하는 것이다. 그러나 웹 응용 프로그램 측면에서 응용계층 보안을 탑재한 브라우저가 사용하기 더 쉽고, 저비용과 높은 사양을 제공하므로 응용 계층 보안이 아직은 더 우수한 방식이라 할 수 있다. 즉 시간 대 비용절감 측면에서 보면, 응용 계층 보안이 시스템과 네트워크 관리자 입장에서 좀 더 우수한 방식이라 볼 수 있다. 응용계층 보안의 경우, 사용자는 운영체제 자체를 재설치 하거나 업그레이드 할 필요 없이, 보안 기능을 갖는 브라우저를 다운로드하여 설치하기만 하면 된다. 실제로, 설치하기 어려운 보안은 소수 사용자들만이 설치할 수 있기 때문에 안전성이 떨어진다고 할 수 있다.

2.2.1 암호를 이용한 웹 보안

강력한 암호를 이용한 웹 보안에 대한 요구가 늘어나면서 암호 알고리즘을 사용하게 되었다. 이러한 보안 방법들이 나오기 시작하면서, 암호를 이용한 보안 방법들은 아래와 같은 보안 요구사항들을 만족시켜 주기 위해 구성되었다.

1. 사용자 인증 :현재의 웹은 클라이언트와 서버에 대한 사용자 인증이 이루어지지 않고 있다. 2. 초기부터 HTTP에서 지원한 basic authentication과 IP filtering은 각각이 가지고 있는 약점 때문에 사용자 인증을 제대로 구현해 놓지 못했다. 3. 서버에 대한 인증도 이루어지지 않아 클라이언트는 자신이 접속한 서버가 의도했던 서버인지를 확인할 수 있는 방법이 없었다 [문제 1].

위에 대한 사용자 인증이라는 요구사항을 만족시키기 위하여, 공개키 암호 알고리즘을 기반으로 request와 response에 전자서명을 붙이게 함으로써 서버와 클라이언트에 대한 사용자 인증을 수행하고, 또한 기밀성을 위하여 웹에서 오가는 request 메시지와 response 메시지를 암호화함으로써 서버나 클라이언트만이 그 내용을 알아볼 수 있게 하였다. 특히, 웹이 전자상거래에 본격적으로 이용될 경우에는 웹 통신의 내용이 개인이나 기업의 경제 정보가 주를 이룰 것으로 보이기 때문에 기밀성의 구현은 필수적이 되었다. 메시지 무결성을 구현하기 위해서는 송신자가 송신한 내용과 수신자가 수신한 내용이 일치하는가를 검사해 줄 수 있어야 한다. 그렇지 않을 경우, 내용이 전송 도중에 불법적으로 변경되더라도 서버와 클라이언트는 이를 모를 수가 있게 된다.

2.2.2 Message digest authentication

사용자가 로긴 네임과 패스워드를 입력하면 타임스탬프가 함께 추가되어 로긴 네임, 패스워드, 타임스탬프가 해쉬 함수에 입력하면서 해쉬 함수의 결과로 나온 값에 평문의 로긴 네임과 타임스탬프를 추가하여 서버로 전송한다. 서버는 로긴 네임과 타임스탬프를 읽고, 자신이 보관하고 있는 파일에서 패스워드를 읽어 로긴 네임, 패스워드, 타임스탬프를 생성하게 되는데 이를 다시, 해쉬 함수의 입력으로 주어 나온 결과가 클라이언트가 보낸 값과 일치하면 정당한 사용자라고 인식한다. 타임스탬프는 replay공격을 방지한다. 이와 함께, 서버에 저장되는 사용자들의 패스워드가 서버 관리자 이외의 사용자들도 읽을 수 있는 형태로 저장되어서는 안 된다.

암호를 웹 보안에 이용하는 형태를 나누어 보면 아래와 같이 나눌 수 있다.

1. Kerberos나 PGP와 같이 HTTP의 상위에서 보안을 구현해 주는 방식을 내용기반 방식이라고 하며, 보통의 HTTP 메세지 전체를 암호화하는 형태로 나타낸다.

2. SHTTP는 웹 보안을 위한 새로운 형태의 프로토콜로서 이렇게 HTTP 메시지의 내용(Content)부분 암호화하여 사용하는 방식을 메시지기반 방식이라고 한다.

3. SSL과 같이 HTTP계층과 TCP/IP계층 사이에 존재하며 TCP/IP계층 자체를 암호화함으로써 모든 HTTP 메시지를 암호화하는 방식을 채널기반 방식이라고 한다.

2.2.3 Secure HTTP

1994년 EIT사에서 HTTP 보안요소를 첨가한 확장판으로 구현 범용으로 사용될 수 있도록 설계 되었으며 통신의 기밀성, 인증, 무결성 등을 지원하는데 응용 레벨에서의 메시지 암호화를 통해 안전한 통신을 보장해 주고, RSA사의 공개키 암호 알고리즘을 이용하여 서버와 클라이언트가 공유하여야 하는 정보(비밀키)등을 암호화하여 전송한다. 암호 관련 알고리즘으로는 DES, RC2, RC4, IDEA, RSA, DSS, MD2, MD5, SHA 등을 사용하며, 사용하고자 하는 암호 알고리즘을 서버와 클라이언트가 암호화 통신 이전에 협의 가능 하다 SHTTP는 현재 사용되고 있는 HTML에는 별다른 수정을 요구하지 않지만, HTTP와는 전혀 새로운 프로토콜이기 때문에 이 SHTTP를 수용하는 웹 서버와 브라우저 프로그램의 개발이 필요하게 된다[문제 2].

2.2.4 SSL(Secure Socket Layer)

넷스케이프 사에서 개발한 웹 보안 프로토콜로 응용레벨과 TCP/IP 사이에 위치하며, 내용의 암호화, 서버의 인증, 내용의 무결성을 제공하는데 서버에 대한 인증은 필수적으로 수행하고, 클라이언트에 대한 인증은 선택적으로 수행하게 된다. 서버와 클라이언트 쪽의 TCP/IP 연결을 위해서 핸드쉐이크(handshake) 프로토콜을 수행하면, 양쪽은 암호화 통신에 합의하고, 암호화

통신과 인증에 필요한 값들을 준비한다. 핸드쉐이크 프로토콜을 통해 앞으로 사용될 키(server_write_key, server_read_key, client_write_key, client_read_key)들을 공유하게 된 서버와 클라이언트는 전송되는 모든 데이타를 암호화하여 전송 가능하다. 미국에서 외국으로 수출했던 40비트 SSL제품이 해독된 적이 있는 데, 그 경위는 다음과 같다. 그 제품은 RC4라는 관용암호 알고리즘을 사용하는 데, RC4는 본래 128비트의 키를 사용한다. 하지만, 미국 정부에서 외국 수출용은 이 128비트 중 40비트만 암호화해서 전송하고 나머지 88비트는 평문으로 전송하게 되어 있다. 그리고 RC4는 소스코드가 공개되어 있지 않은 상태였으므로 결국 그 제품은 40비트의 암호화 된 키와 RC4의 비공개성으로 보안성을 구현한 것이 되었다. 그러나, brute-force 공격으로 암호화되었던 40비트의 키가 노출이 되었고, reverse engineering으로 RC4의 구현원리가 밝혀짐에 따라 SSL이 해독된 것이다[문제 3].

2.2.5 외부 프로그램을 이용한 방법

기존의 웹시스템에는 수정을 전혀 요구하지 않으면서 웹보안을 구현해 주는 방법이 있다. 암호를 담당하는 외부 프로그램을 서버와 브라우저에 연결시킴으로써 HTTP request메시지와 response메시지의 암호화와 복호화에 이용한다. 전자우편 보안도구인 PGP를 암호모듈로 사용했고 서버와 외부 프로그램과의 통신은 기존의 CGI 프로그래밍 기법을 사용하여 구현하며, 외부 프로그램과 브라우저와의 통신은 여러 가지 방법이 있는데, 브라우저가 넷스케이프일 경우에는 plug-in, external viewer기법을 이용하고, 모자익일 경우에는 NCSA에서 제공하는 CCI라이브러리를 이용하여 구현했다. 이 방법의 장점으로는 우선 앞에서 언급했던 것처럼 기존의 웹시스템에 아무런 수정을 요구하지 않는다는 점이다. 많은 보안 프로토콜들이 기존의 시스템에 많은 수정을 요구하는 이유로 많이 사용되지 않고 있다는 점을 고려하면 이 장점은 매우 강력한 요소로 작용된 것이다. 또한, PGP 등 암호 모듈로 사용되는 프로그램들은 웹에 국한되지 않고 여러 응용 분야에 사용될 수 있기 때문에 한 사용자의 시스템에 여러 암호 프로그램이 설치될 필요가 없게 되었다. 이 방법의 단점은 암호 모듈이 웹시스템의 외부에 위치한다는 점에서 취약점을 나타낸다. 결국, request 메시지와 response 메시지들을 암호화/복호화 하기 위해서 외부 프로그램과 암호모듈, 외부 프로그램과 웹프로그램 사이에 상당한 양의 통신이 필요하게 되었는데 이러한 통신양의 증가로 인하여 결국 수행시간이 길어진다는 단점이 있다[문제 4].

2.2.6 Kerberizing 웹

Kerberos는 분산 컴퓨팅 환경에서의 사용자 인증과 키 분배를 목적으로 MIT에서 개발한 보안 시스템으로 Kerberos를 이용하여 웹클라이언트와 서버 사이에 상호 인증을 지원하고, NCSA의 httpd_1.5R 버전에서 V4, V5 버전을 지원했다. 클라이언트는 ticket을 얻어 이를 이용하여 자신을 증명하는데 ticket은 인증 서버에서만 생성할 수 있고, 읽을 수 있는 구조를 가지고 있다. ticket은 클라이언트가 최근에 인증 받았음을 의미하는 것으로 Authenticator는 클라이언트에서 생성하고, 서버에서 읽을 수 있는 구조를 하고 있다. 이는 클라이언트의 신원을 증명하고, 서버와 통신하기 위한 currency를 증명해 주는 역할을 한다.

Kerberos의 인증이 실패하는 경우는 인증을 받고자 하는 서버의 이름이 빠져 있는 경우, ticket의 유효기간이 지났을 경우, ticket에 포함되어 있는 주소와 패킷내의 주소가 일치하지 않을 경우, ticket에 대한 replay 공격이 감지될 경우 등과 같이 나타낸다[문제 5].

앞 절에서 언급한 것과 같이 관련 웹 보안을 하는 데 있어서의 문제점들로 [문제 1~5]이 발견되었다. 이러한 문제들을 해결하기위해 본 논문에서는 제3장 시스템 동작 및 시나리오의 3.1 개요에서 문제점을 해결하기 위한 방법들을 제시한다.

2.3 관용 암호화 기법(대칭키 암호화 기법)

관용 암호화 방식은 공개키 암호화 방식에 비해서 상대적으로 처리 속도가 빠르고(1000배)[16], 데이타를 암호화 하는데 사용하는 키와 복호화 하는데 사용하는 키가 동일하다는 점이 가장 큰 특징이다. 따라서, 데이타를 송수신하기 위해서 통신하는 양쪽이 동일한 키를 공유해야 한다. 이러한 특징으로 인해 통신하는 양쪽은 키 분배 및 관리 하는데 있어서 어려움이 따른다[5]. 공개키 암호화 방식은 데이타를 공개키를 이용해서 암호화하고 비밀키를 이용해서 복호화 하는 방식이기 때문에 관용암호화 방식에 비해 키 분배 문제가 쉬워지기는 하지만, 처리 속도가 느려서 웹 환경에서는 관용 암호화 방식과 비교해 볼 때 상대적으로 부적합하다. 또한 공개키 분배에 있어서 소유자에 대한 신뢰성 문제가 여전히 남아 있다. 관용 암호화 방식을 기반으로 한 알고리즘에는 DES(Data Encryption Standard), Triple DES, Feistel, NDS(New data Seal), Lucifer, FEAL, IDEA (International Data Encryption Algorithm)등이 있다.

본 논문에서는 관용 암호화 방식(Triple-DES)을 효과적으로 결합한 Diffie-Hellman 알고리즘을 사용하였다.

2.4 Diffie-Hellman 키 교환 알고리즘

온라인 키 서버(on-line key server)를 사용하는 것을 원하지 않을 경우에 비밀 키를 교환하기 위하여 키

공유 프로토콜(key agreement protocol)을 사용한다. 가장 잘 알려진 키 공유 프로토콜은 Diffie- Hellman 키 교환 (key Exchange) 이다. p가 소수, α는 Zp의 원시원이고 p와 α는 공개적으로 알려져 있다고 가정한다. 이 프로토콜은 Diffie-Hellman키 사전분배와 매우 비슷하다. 차이점은 사용자 U와 V의 각각의 지수 α_U와 α_V가 이 프로토콜이 실행될 때마다 고정되는 것 대신에 새롭게 선택되는 것(프로그램에서는 세션 키)이다. 이와 더불어, 이 프로토콜에서는 U와 V 모두가 키의 새로운 key-freshness를 확신한다. 여기서, 세션키는 두 개의 무작위지수 α_U와 α_V에 의존하기 때문이다.

Diffie-Hellman key Exchange 알고리즘은 표 2와 같다[7].

표 2 Diffie-Hellm an Key Exchange

1. U chooses α_u at random, $0 \le \alpha_u \le \rho-2$
2. U computes α_u mod ρ and sends it to V.
3. V chooses α_v at random, $0 \le \alpha_v \le \rho-2$
4. V computes $\alpha^{\alpha v}$ mod ρ and sends it to U.
5. U computes

$$K = (\alpha^{\alpha v})^{\alpha u} \bmod \rho$$

and V computes

$$K = K = (\alpha^{\alpha u})^{\alpha v} \bmod \rho.$$

3. 시스템 동작 시나리오

3.1 개요

기존 웹 환경의 시스템에서 제공하고 있는 기능은 Basic Authentication, Network Domain Access Control과 같은 기본적인 보안 기능만을 가지고 있었다. 이러한 것들은 네트워크 도청을 통한 사용자의 정보를 쉽게 가로챌 수 있는 취약성을 나타내므로 안전성이 보장되지 않는 보안 기술들이라 말할 수 있다. 보안기술의 향상과 안전한 데이타 전송을 위한 방법들이 제안 되거나 구현되어져 왔지만 문제점들이 나타나게 되었다. 제2장에서 다루었던 문제점을 가지고 해결방안을 알아본다.

[문제 1] 클라이언트와 서버에 대한 사용자 인증이 이루어지지 않는 문제점과 메시지 무결성을 구현하기 위한 어려움이 나타났다. 본 논문에서는 클라이언트와 서버사이의 인증을 위한 방법으로 Diffie-Hellman 키 교환 알고리즘으로 세션 키를 생성함과 기본 메세지를 확인함으로써 서로의 상호인증과 안전한 데이타 전송을 위한 메커니즘을 확립하였다. [문제 2] 메시지 전체를 암호화함으로써 나타나는 불필요한 오버헤드를 줄이기 위한 방법으로 서버와 클라이언트의 모듈화, 또 새로운 프로토콜을 사용함으로써 발생하는 서버와 브라우저 프로그램의 개발에 따른 불편함을 해결하도록 하였다. [문제 3, 4, 5] SSL에서 클라이언트에 대한 인증은 선택적으로 수행을 하기 때문에 보다 안전한 데이타 전송을 보장하지 않는다. 외부 프로그램을 이용한 방법의 단점은 통신양의 증가로 인하여 결국 수행시간이 길어진다는 것이다. Kerberos는 인증을 위해 ticket을 사용하여 수행하게 되는데 이는 인증에 대한 복잡함과 실패확률이 높아지는 결과를 초래하게 된다. 본 논문에서는 Java 암호화 알고리즘을 이용하여 서버 클라이언트의 보안 모듈로 설계, 이종의 시스템에서도 통신이 가능하도록 하였다.

3.2 동작 시나리오

클라이언트와 서버 사이에 Diffie-Hellman Algorithm을 이용해서 만든 세션 키를 공유함으로써, 안전한 정보 전송을 수행함과 동시에 두 시스템간의 상 인증을 통하여 신뢰성을 향상시켰다.

다음은 일반적인 클라이언트 서버의 Diffie-Hellman Algorithm 사용한 세션 키 생성과정에 관한 내용이다. 클라이언트가 서버에 접속하면 Diffie-Hellman 방식의 키 교환 알고리즘을 수행한다. 수행 방식은 그림 2와 같다.

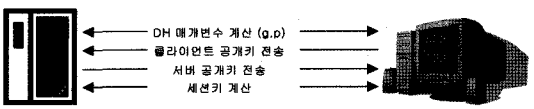

그림 2 Diffie-Hellman key 교환 알고리즘에 의한 세션 키 생성과정

Diffie-Hellman의 알고리즘에 의해 공개키와 개인키가 생성이 되고 생성이 된 키로 클라이언트는 서버에게 공개키를 전송한다. 서버는 클라이언트에게 공개키를 주어 클라이언트에서 키 일치를 하여 세션 키를 생성하게 되는 것이다.

클라이언트와 서버가 세션 키를 공유하게 되면 상호 인증을 수행한다.

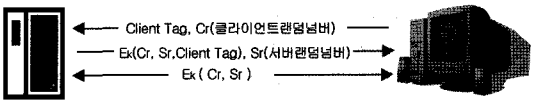

그림 3 클라이언트와 서버 상호 인증

클라이언트와 서버의 상호 인증을 위해 클라이언트는 자기 자신의 랜덤 넘버와 데이타를 서버에 전달하고 서버에서는 클라이언트에서 전달 받은 내용을 세션 키로 암호화한 것과 자신의 서버 랜덤 넘버를 클라이언트에 보내게 된다. 이렇게 보내진 데이타는 클라이언트의 세션 키를 이용하여 복호화하여 클라이언트가 보낸 데이

타와 비교하여 일치하면 서로의 상호 인증이 성립이 된다.

그림 4는 안전한 통신 채널을 구성한 것으로 구성요소는 그림과 같다. 클라이언트 보안 모듈에서 자료를 넘겨받은 서버 모듈은 보안 모듈에서 사용자가 요청한 내용을 처리하기 전에 복호화 작업을 수행한다. 복호화 작업을 통해서 정상적인 데이타로 변환이 되면, 웹 서버 모듈은 정상적인 문자 처리 과정에 의하여 사용자의 요구사항을 처리한다. 사용자가 요청한 자료를 클라이언트 전송하기 전에 서버 보안 모듈에서 암호화 과정을 거치게 된다. 서버 보안 모듈에서 암호화된 자료는 클라이언트 보안 모듈 시스템으로 전송되며, 자료를 넘겨받은 클라이언트 보안 모듈은 복호화 과정을 통해서 원본 메시지를 만든다. 이 메시지가 사용자의 웹 브라우저로 출력되게 된다.

그림 5는 제안하는 서버 클라이언트의 구조를 전체적으로 나타낸 것이다. 주된 메커니즘은 서버에서 클라이언트로 또는 클라이언트에서 서버로의 필요한 데이타만을 암호화하여 안전하게 통신한다는 것이다. 이 구조에는 서버 보안 모듈과 클라이언트 보안 모듈이 존재한다. 각각의 보안 모듈은 메시지를 암호화 하거나 전송되어진 암호화된 메시지를 복호화 하는 기능을 가지게 된다. 각 암호화와 복호화가 수행되기 전에, 본 시스템에서는

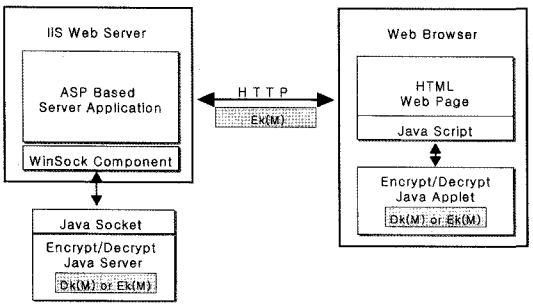

그림 4 데이타 암호화에 의한 안전한 통신 채널 구성

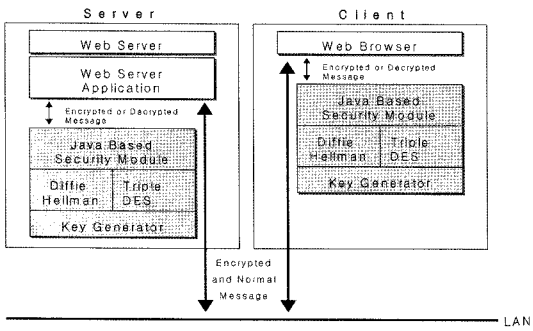

그림 5 서버 클라이언트 보안 모듈 구조

Diffie-Hellman 키 교환 알고리즘을 이용하여 세션 키를 생성 서버와 클라이언트 사이에 공유하게 되고, 이를 통해 상호 인증이 이뤄지면 그 이후부터는 안전한 데이타 전송이 이뤄진다는 것을 보여준다. 주목할 만한 사항은 서버와 클라이언트 간의 인증을 위해 필요한 세션 키를 통신이 이루어질 때마다 새로운 키를 생성하여 키 교환이 이루어지므로 보다 안전한 통신이 이루어진다는 것이다. 다음은 서버 보안 모듈과 클라이언트 보안 모듈의 구조를 자세하게 살펴본다.

3.2.1 서버, 클라이언트 보안 모듈

서버 보안 모듈의 구조는 자바(Java) 암호화 패키지로 구성되어져 있는데, 이는 유닉스, 리눅스, 윈도우 제품 등과 같이 어느 운영체제나 제한 없이 연동하며 이기종간의 통신을 가능하도록 구현한 것이다. 서버 보안 모듈 구조를 살펴보면 안전한 데이타 통신을 위한 암호화 및 복호화 기능이 있다. 암호화 및 복호화 기능은 관용 암호화 방식인 Triple-DES로, 일반 비대칭 암호화 알고리즘이 가지는 속도의 문제점을 극복하고, 관용 암호에서 나타나는 키(key) 관리 문제를 Diffie-Hellman 키 교환 알고리즘으로 해결 하도록 설계하였다. 또한, Asp와 Java 사이의 API가 존재하지 않는 문제로 통신이 불가능한 문제는 내부적인 소켓을 사용하여 문제를 해결 하였다.

그림 7은 클라이언트 보안 모듈을 나타낸다. 웹 브라우저와 애플릿의 연동을 위해 Javascript를 사용하며, 웹 브라우저에 로드되는 HTML의 이벤트를 감지하여 동적인 페이지 구성이 가능하도록 작성되었다.

암호화, 복호화를 수행하는 자바 애플릿의 통신 경로가 브라우저에만 국한되어 있는데, 이로 인하여 서버와 클라이언트 사이에 연결을 위한 소켓이나 어떤 구체적인 통신 채널을 구성하지 않아도 웹만으로 통신이 가능하다. 그리고, 내부 방화벽(Firewall)으로 인한 통신 장애를 일으키지 않는 장점이 있다. 하지만, 서버와 클라

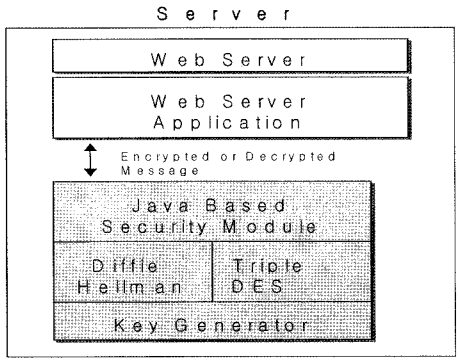

그림 6 서버 보안 모듈

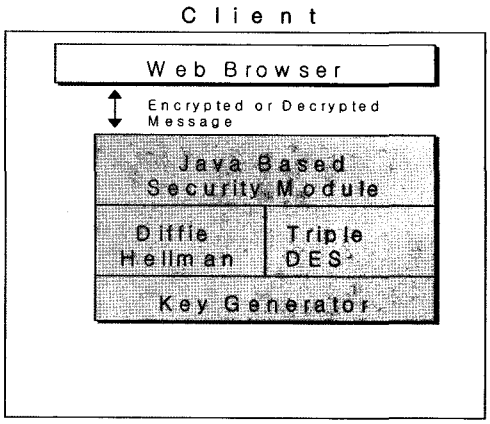

그림 7 클라이언트 보안 모듈

이언트 사이의 다이렉트 연결이 구성되어 있지 않기 때문에 세션 키 생성을 위하여 서버와 클라이언트의 공개키를 전달하는데 있어서 웹을 이용하므로, 전송의 신뢰성이 떨어질 수 있다는 단점이 있다. 이 문제를 해결하기 위하여 본 논문에서 제시한 Diffie-Hellman 키 교환 알고리즘을 이용한다.

4. 보안 모듈 실험

4.1 서버에서 클라이언트로 암호화된 데이타 전송

웹 서버에서 기존의 출력되던 HTML 데이타에 대하여 서버측 보안 모듈을 통해 암호화를 수행한 뒤 이를 클라이언트 브라우저에서 읽어 들여 클라이언트 보안 모듈에 의해 복호화를 수행하는 것으로 그림 8은 정상적인 메시지를 나타낸다. 그림 9는 보안 모듈이 없는 클라이언트인 경우에 화면에 위와 같이 암호화된 알 수 없는 데이타로 보여 지게 된다. 좀더 명확하게 확인하기 위하여 HTML의 소스를 열어본 결과이다.

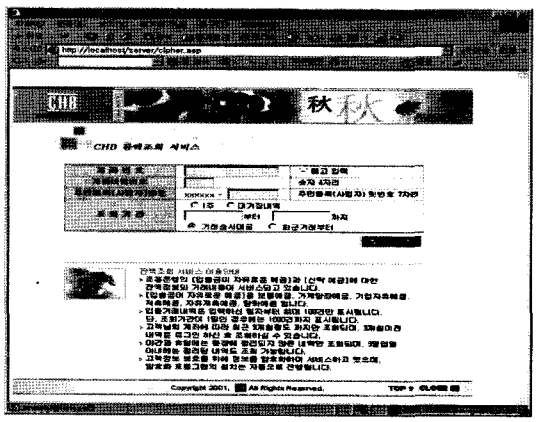

그림 8 정상적인 클라이언트 보안 모듈의 복호화

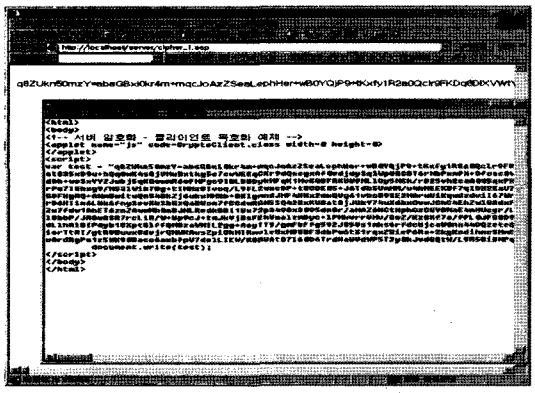

그림 9 보안 모듈이 없는 클라이언트 복호화의 실패

4.2 클라이언트에서 서버로 암호화된 데이타 전송

그림 10과 같은 경우 클라이언트에서 보낼 메시지를 나타낸 것이고 그림 11의 경우 클라이언트에서 서버로 암호화 메시지를 전송하기 전 클라이언트 보안 모듈에 의해 데이타가 암호화된 것을 나타낸 것이다. 그림 12의 경우 클라이언트 보안 모듈에 의해 데이타가 암호화되어 서버로 전송된 데이타가 서버 보안 모듈에 의해 데이타가 복호화 되어 데이타를 정상적으로 처리한 결과이다.

4.1과 4.2의 결과로 본 논문에서 설계한 보안 모듈에 의해 서버에서 클라이언트로 클라이언트에서 서버로의 데이타가 정상적으로 암호화 및 복호화 되어진다는 것을 확인 하였다.

5. 결 론

본 논문은 이종의 시스템에서 안전한 데이타 전송을

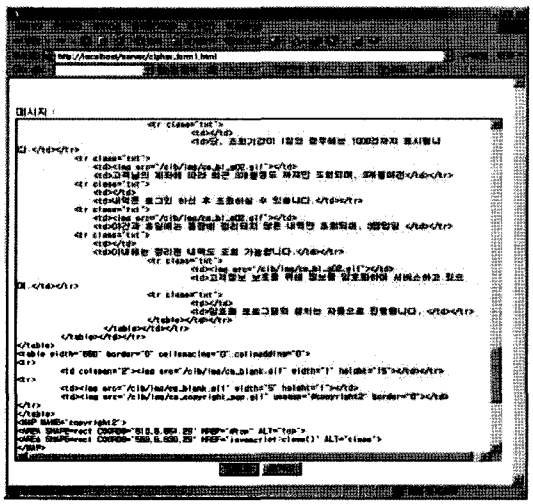

그림 10 클라이언트 전송 메시지

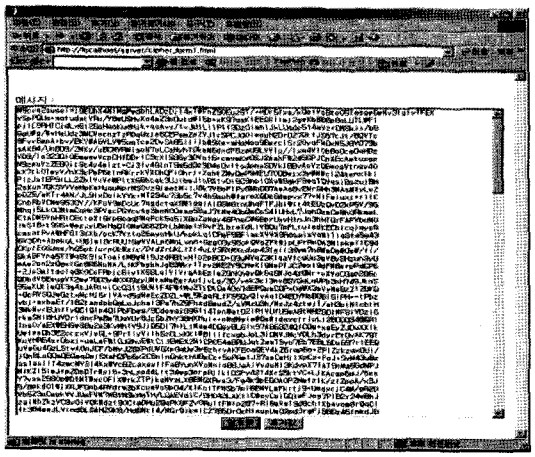

그림 11 클라이언트 전송 메시지 암호화

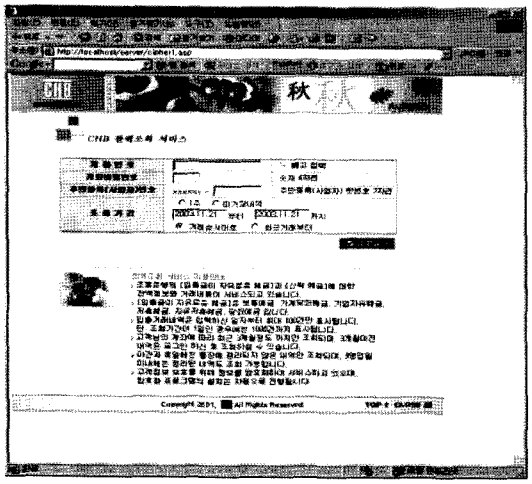

그림 12 서버에서 복호화된 메시지

만족하는 웹 보안 모듈을 구현한 내용에 대해서 언급했다. 기본적인 구조는 크게 두 개의 모듈로 구현된다. 하나는 웹 서버에서 동작하는 서버 보안 모듈이고, 다른 하나는 클라이언트에서 동작하는 클라이언트 보안 모듈이다. 웹 서버 보안 모듈은 암호화된 메시지를 클라이언트에게 전송하고 클라이언트 보안 모듈은 서버로부터 받은 암호화된 메시지를 정상적인 메시지로 복호화하여 웹 브라우저에 나타나게 한다. 각 보안 모듈은 HTML 메시지에 대한 암호화 기능과 암호화된 메시지를 복호화 하는 기능을 가지고 있다.

본 논문에서 제안하는 보안 구조는 클라이언트와 서버간의 간단한 모듈의 설치로 안전한 데이타 전송을 보장한다. 웹 서버와 브라우저 사이의 빠른 암호화 통신을 위한 관용 암호화 방식인 Triple-DES를 사용하였으며, 관용 암호화 방식의 문제인 키 관리 문제를 해결하기 위한 Diffie-Hellman 키(key) 교환 알고리즘을 적용하였다. 이렇게 함으로 인해, 관용 암호화에서 발생하는 키 분배 및 관리문제를 해결되도록 하였다. 또한, 이를 바탕으로 상호인증이 수행되도록 하여, 안전한 데이타 보안을 위한 서버와 클라이언트 사이의 인증문제를 해결하였다. 이 시스템은 이종간의 자유롭고 안전한 통신이 가능하도록 Java로 구현하였다. 이종의 시스템에서 발생하는 API문제를 내부 소켓을 이용하여 API를 구축, 효율적인 보안 모듈을 설계 및 구현하였다. 향후 과제로는 좀더 효율적인 키 관리 알고리즘을 사용한 웹 보안에 대한 연구와 키 교환 알고리즘에 신뢰성 향상에 연구가 필요하다. 그리고 이러한 암호 알고리즘을 이용한 웹 보안에 있어서의 성능 향상 지표에 대한 연구가 필요하겠다.

참 고 문 헌

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

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

[3] 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.

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

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

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

[7] Bruce Schneier, "Applied *Cryptography*," Jon Wiley & Son, Inc.

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

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

[10] Whitefield Diffie, Paul C. van Oorchot and Michael J. Wiener "Authentication and Authenticated Key Exchanges," Designs Codes and Cryptography, 1992, pp.107-125.

[11] F.Bergadano, B. Crispo, M. Eccettuato "Secure www Transactions Using Standard HTTP and Java Applets," 3rd USENIX Workshop on Electronic Commerce, 1998, pp.109-119.
[12] Joris Claessens, Bart Preneel and Joos Vandewalle., "Secure communication for secure agent-based *electronic commerce applications*," LNAI 2033, 2001, pp180-190.
[13] Http://java.sun.com/j2se/1.4.1/guide/securtiy/
[14] Jess barms, Daniel Somerfield "Professional Java Security," Wrox Press Inc. 2001.
[15] Avied D. Rubin, Daniel Geer, Marcus J. Ranum "Web Security Source Book," John Wiley & Sons, Inc. 1997.

김 기 성
2001년 2월 한양대학교 전자컴퓨터공학부 학사 졸업. 2004년 2월 한양대학교 컴퓨터공학과 석사 졸업

김 광
1994년 한양대학교 전자계산학과(공학사). 1996년 한양대학교 전자계산학과(공학석사). 1999년 한양대학교 전자계산학과(박사수료). 2001년~2004년 아이티플러스(주) 정보기술연구소. 2005년~현재 한양대학교 컴퓨터공학과 강사

허 신
1973년 2월 서울대학교 전기공학 학사 졸업. 1979년 5월 미국 University of Southern California 전산학 석사 졸업 1986년 5월 미국 University of South Florida 전산학 박사 졸업. 1980년~1986년 University of South Florida 연구원보. 1986년~1988년 The Catholic University of America 조교수. 1988년~현재 한양대학교 컴퓨터공학과 교수. 관심분야는 Distributed Computing Systems, Fault-Tolerant System, Real-Time Operating Systems