안전한 WWW 통신을 위한 NetCrypt 설계

이 은 성*, 박 현 동*, 류 재 철*
* 충남대학교 컴퓨터 과학과

A Design of NetCrypt for Secure WWW Communication

Eun-Sung Lee , Hyun-Dong Park , Jae-Cheol Ryou
Department of Computer Science, Chungnam National University

요 약

본 논문에서는 WWW 상의 정보를 안전하게 전송하는 보안시스템을 설계하고자 한다. 현재 WWW 상의 모든 정보는 보안서비스를 제공받지 못하는 평문의 형태로 이루어지기 때문에 제삼자가 중간에서 가로챌 수 있다. 이러한 문제를 해결하기 위해서 SSL,S-HTTP 등과 같은 보안서비스가 발표 되었으나, SSL 이나 S-HTTP 는 WWW 상에서 전송되는 모든 정보를 암호화의 대상으로 고려하기 때문에 정보의 전송 시 오버헤드가 크고 보안서비스를 선택하여 제공하기 어렵다. 그러나 본 논문에서 제안하는 보안시스템은 웹브라우저의 확장 기법인 플러그-인 기법을 이용하여 필요한 부분에서만 정보를 암호화 하기 때문에 개발자의 의도에 따라 보안서비스를 선택하여 제공할 수 있고 암호화된 정보 전송시의 오버헤드를 줄일 수 있다.

1. 서 론

인터넷은 세계 최대의 통신 네트워크로서 그 사용이 급격히 증가하고 있는 추세이다. 특히 WWW 사용의 편리성을 제공하는 웹브라우저의 등장으로 일반인들의 인터넷 사용이 급증하고 있다. 이러한 편리성으로 인터넷의 사용자가 더 많아짐에 따라 사용자의 요구사항도 늘어나게 되고, 그에 따라 인터넷상에서의 서비스도 점점 다양해지고 복잡해지고 있다. 서비스

· 본 연구는 한국과학재단 특정기초연구과제(과제번호 : 97-01-00-06-01-3) 연구비 지원에 의해 수행되었음

가 다양해 짐에 따라 처리해야 할 정보의 종류가 점점 더 많아지고 있는데, 이러한 서비스들 가운데는 정보를 전송하는 송신자와 정보를 받는 수신자 이외에서는 그 내용을 알 수 없게 하는 기밀성을 보장해야 하는 서비스도 등장하고 있다.

그러나, WWW 은 인터넷이 가지고 있는 보안상의 문제점들을 그대로 갖고있기 때문에 정보 전송 시 무결성이나 기밀성을 보장 받을 수 없다. 최근 전자상거래가 중요한 인터넷 서비스가 됨에 따라 사용자의 개인정보를 이용하여 서비스를 받는 경우가 많아졌다. 이에 따라 사용자의 측면에서는 자신의 개인정보를 안전하게 보호 받고 싶은 요구사항이 생기게 되었고, 이러한 사용자의 요구사항을 해결해주기 위해서는 암호기법을 이용한 기밀성 및 무결성 서비스가 제공되어야 한다. 이러한 기본적인 보안서비스가 제공되지 않은 상태에서 홈 뱅킹이나 전자상거래의 서비스를 시작하게 되면 정보의 변조나 다른 사용자의 정보를 알아내는 등 많은 보안 사고들이 일어나게 될 것이다. 실제로 현재까지 보고된 많은 해킹 사고들이 보안 서비스를 제공 받지 못하는 정보들을 악용하는 사례가 많았다.

본 논문에서는 이러한 보안 사고를 막고 안전한 인터넷 사용을 위해 보안 프로토콜인 NetCrypt 를 설계하고자 한다. 2 장은 현재 제공되는 보안 프로토콜에 대해 소개하고, 3 장은 본 논문에서 설계하고자 하는 보안 프로토콜인 NetCrypt 의 동작과정에 대해 설명하고, 4 장은 NetCrypt 의 구현에 필요한 핵심기술에 대해 소개하며, 마지막으로 5 장에서는 결론을 기술하였다.

2. 기존의 WWW 보안 프로토콜

1 장에서 언급했던 보안 사고들을 막고 안전한 보안 서비스를 위해 현재 많은 보안 프로토콜들이 여러 회사 또는 단체에서 제공되었고 인터넷에서 많이 쓰이고 있다. 그 보안 프로토콜에 대해 살펴보면 다음과 같다.

2.1 S-HTTP

EIT 사에서는 Secure Hypertext Transfer Protocol(S-HTTP)이라는 새로운 형식의 HTTP 프로토콜을 제안하였다.

S-HTTP 는 WWW 의 기반 프로토콜인 HTTP 상에 암호화 모듈을 첨가함으로써 정보의 기밀성과 무결성을 보장한다. 그러나 S-HTTP 의 경우에 새로운 프로토콜이라는 특성을 가지고 있기 때문에 새로운 웹서버와 웹브라우저가 필요하다는 단점을 가지고 있다[1].

2.2 SSL

WWW 상의 안전한 데이터 송수신을 위해 Netscape 사에서 Secure Socket Layer(SSL)라는 프로토콜을 제안함으로써 WWW 보안 문제 해결에 새로운 전환점을 맞이하게 되었다. SSL 의 경우 HTTP 의 기반이 되는 TCP/IP 레벨의 암호화 모듈을 제공함으로써 정보보호를 시도하였다. Handshake 프로토콜 단계에서 암호화에 사용될 키를 공유하고, 송수신하는 모든 정보를 암호화의 대상으로 한다. 그러나 SSL 의 경우에는 프로토콜상의 특성으로 사용자의 인증을 위한 전자서명만을 제공하고 메시지에 대한 전자서명을 지원해주지 않는 문제점이 있다. 또한, 미국의 정책에 따라 자국 이외에서는 암호화에 사용하는 키 크기의 제한으로 보안 서비스에 대한 신뢰도가 떨어지고, 채널 자체를 암호화의 대상으로 하기 때문에 서버와 클라이언트와의 정보 전달 시 오버헤드가 크다는 단점이 있다[2].

이러한 단점들 중 미국 이외의 국가에서 암호/복호동작 시 사용하는 키 크기의 문제를 해결한 보안 시스템이 국내에서 개발되었다. 이 보안 시스템은 사용자의 시스템에 프락시 프로그램의 형태로 구동하여 웹브라우저와 SSL 서버사이에서 동작한다. 좀 더 자세히 설명하면 다음과 같다. 웹브라우저에서 40 비트의 키로 정보를 암호화해서 보내면 프락시 프로그램에서 이를 가로챈다. 이 가로챈 정보를 복호화하여 128 비트의 키크기로 다시 암호화해서 SSL 서버에게 보내준다. 또한 서버에서 128 비트의 키크기로 암호화한 정보를 프락시 프로그램에서 가로채서 복호화 하여 40 비트의 키크기로 다시 암호화하여 브라우저에게 전송한다. 그러므로 실제로 WWW 상의 정보는 128 비트의 키크기로 암호화한 정보이다. 그러나, 두번 암호/복호화 동작을 통한 오버헤드가 크고, 기존의 SSL 서버에서는 사용이 안되고 특정한 SSL 서버 즉, 프락시 프로그램과 약속이 되있는 서버에서만 동작하는 호환성의 문제점이 있다.

2.3 Socket 을 이용한 방법

WWW 서버가 수행되는 컴퓨터내에 소켓 서버를 구동하고 사용자의 컴퓨터내에 소켓 클라이언트 프로그램을 두어서 실제로 필요할 때마다 소켓 서버와 연결을 맺어 보안서비스를 제공하는 형태이다. 이 방식은 전자상거래 시스템중에 지불에 관련하여 제공되는 경우가 많다. 즉 상품에 대한 정보는 웹서버와 웹브라우저간의 통신을 통해서 제공 받고 실제 지불단계에서는 소켓서버와 클라이언트 프로그램을 이용하는 방법이다. 그러나 이 방식은 HTTP 상의 정보에 대해서는 보안서비스를 제공해주지 못하는 단점이 있다.

본 논문에서 제안하는 NetCrypt는 위에서 설명한 보안 프로토콜들이 가지고 있는 약점들을 보완하였다. 우선 사용하는 관용 암호 알고리즘의 키의 크기를 128 비트 이상으로 제한 없이 사용할 수 있다. 그리고, SSL을 사용하면 전송되는 모든 정보들을 암호화하는 데에 반해, NetCrypt는 웹서버 관리자가 보안 기능이 필요하다고 판단되는 정보에 대해서만 보안 서비스를 제공할 수 있다. 또한, SSL은 정보에 대해 기밀성만을 제공해 주지만, NetCrypt는 웹 서버 관리자가 기밀성, 무결성, 전자서명을 필요한 대로 선택하여 사용할 수 있다. 즉, 어떤 문서는 관용 암호 알고리즘만을 사용하여 기밀성만을 제공해 주고, 어떤 문서는 전자서명만을 또는 기밀성과 전자서명을 모두 제공해 줄 수 있게끔 한다. 이는 SSL을 사용할 경우, 모든 정보에 대해 암호화 동작을 수행하는 데에 필요한 부담을 줄여 줄 수 있다는 장점도 가지고 있다.

3. NetCrypt의 설계

본 논문에서 설계하는 NetCrypt의 기능은 브라우저와 웹 서버 사이에 전송되는 특정 데이터를 암호화하거나 전자서명을 생성하여 기밀성, 무결성, 송신자 인증을 제공하는 것이다. 이를 위해서 폼페이지를 사용하게 되는 데, 폼페이지에 브라우저 사용자는 암호화 동작에 필요한 값들, 즉 자신의 공개키 ID, 비밀키를 사용 가능케 하는 패스프레이즈를 입력한다. 공개키 ID는 웹 서버로 전송되어 웹 서버가 브라우저 사용자의 공개키를 구별할 수 있도록 해 주는 것이고, 패스프레이즈는 사용자가 자신의 비밀키를 이용하는 데에 사용하는 것이므로 웹 서버로 전송되지 않는다.

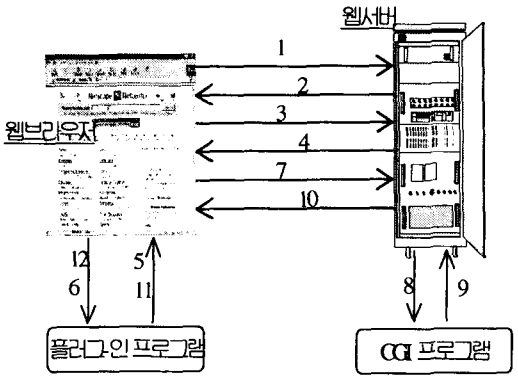
폼페이지를 이용하는 이유는 다음과 같다. 일반적인 HTTP는 웹 서버에서 전송한 정보를 브라우저가 받아 보여 주기만 하는 형태이다. 하지만, NetCrypt에서는 브라우저에서 먼저 요청 메시지를 암호화하여 웹 서버에게 전송해 줌으로서 암호화 통신이 시작된다. 이 때 브라우저 쪽에서 암호화를 수행하는 데 필요한 정보를 사용자가 입력해 주는 과정이 필요한 데, 이 역할을 폼페이지를 이용해서 수행해 주는 것이다. 또한, 암호화 동작을 위해 브라우저 사용자가 웹 서버에게 전송해 주어야 하는 정보를 폼페이지에서 입력 받는다. 또한 폼페이지는 일반적으로 웹 서버의 특정 CGI 프로그램을 구동시키는 데, 이 폼페이지와 연결되어 있는 CGI 프로그램에서 요청 메시지의 복호화와 응답 메시지의 암호화를 담당한다.

서버와 클라이언트의 외부프로그램에서는 암호/복호화시 관용 암호

방식과 공개키 암호방식을 같이 사용한다. 정보의 기밀성 서비스는 암호화 하고자 하는 메시지를 관용 암호방식인 IDEA를 이용해서 암호화 하고 이때 사용된 세션키를 공개키 암호방식인 RSA를 이용하여 상대방의 공개키로 암호화한다. 또한 무결성 서비스와 전자서명 서비스는 공개키 암호방식인 RSA와 해쉬함수인 MD5를 이용해서 제공한다. 공개키와 비밀키쌍은 서버와 클라이언트에서 각각 생성하며, 세션키는 정보를 암호화할 때 마다 매번 생성한다.

3.1 전체적인 동작 과정

본 논문에서 제안하는 NetCrypt의 전체 동작과정을 살펴보면 <그림 1>과 같다. 먼저 클라이언트는 서버에게 초기 웹페이지를 요구하는 1번의 단계를 거친다. 그러면 2번의 단계에서 서버는 클라이언트에게 1번의 단계에서 선택한 웹페이지를 전송하게 된다.



<그림 1>전체 구성도

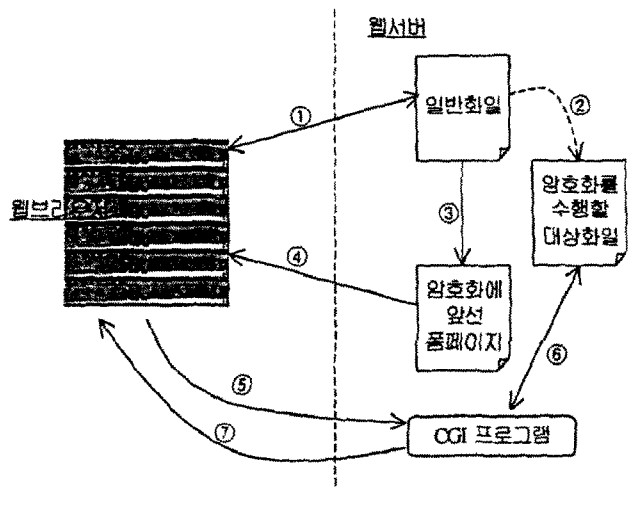
이 웹페이지에는 사용자가 암호화해서 받고자 하는 화일의 목록들이 링크되어 있다. 3번의 단계에서 클라이언트가 암호화해서 받고자 하는 화일을 선택하면 바로 암호화된 페이지가 전송이 되는 것이 아니라 선택한 화일을 암호문으로 전송받기 위한 폼페이지가 서버로부터 전송이 된다.이 단계가 4번째 단계이다. 4번째 단계에서 전송되어진 폼페이지에 암호동작시 필요한 아이디와 패스프레이즈를 입력하고 submit 버튼을 누르게 되면, 일반 폼페이지에서는 서버로 폼필드의 값이 바로 전송된다. 그러나 본 논문에서 설계한 NetCrypt는 <그림 1>의 5번째 단계에서처럼 라이브 코넥트 기법을 이용하여 클라이언트에 위치한 외부 프로그램으로 폼필드의 값을 보낸다 [3]. 그리고 입력 받은 폼필드의 값 즉, 사용자의 아이디와 패스프레이즈를 이용해서 사용자가 서버로부터 전송받고자 하는 화일명과 사용자의 아이디를 외부 프로그램에서 암호화한 후에 6,7번의 단계를 거쳐 서버에게 전송한다. 8번 단계에서 서버는 전송된 암호문을 CGI 프로그램으로 넘겨준다. 9번의 단계에서는 넘겨받은 암호문을 복호화하여 사용자의 공개키 아이디와 화일명을 파싱한다. 그리고, 클라이언트가 요구하는 화일을 찾아, 그 화일에 대해서 암호화 동작을 한다. 10번의 단계에서는 클라이언트로 전송된

암호문은 웹브라우저가 화면에 보여주기 전에 외부 프로그램으로 넘겨주는 11번의 단계를 가진다. 마지막으로 외부 프로그램에서 복호화 작업을 거친 후 해당화일을 다시 클라이언트의 웹브라우저에 보여주는 동작이 12번의 동작이다.

서버와 클라이언트의 동작과정을 자세히 살펴보면 다음과 같다.

3.2 서버의 동작 과정

본 논문에서 제안한 NetCrypt에서는 화일의 암호화 전송을 지원하는데, 이를 위해서 대상 화일의 전송에 앞서 폼페이지를 이용한다. 이에 대한 설명은 <그림 2>와 같다. <그림 2>에서 WWW 서버에 접속하여 문서들을 검색하다가 보안서비스를 이용하여 다운로드 받을 수 있는 웹페이지 또는 화일의 목차를 보는 것이 1번 단계이다. 그리고, 위의 목차들은 "일반화일"에서 "암호화를 수행할 대상화일"로 가는 링크로 되어있다. 이 링크를 클릭하면 2번 단계와 같이 대상화일이 전송되는 것이 아니라 먼저 대상화일의 암호화를 지원하는 폼페이지가 웹브라우저로 전송되는 3번과 4번 단계가 실행된다. 이 폼페이지에는 대상화일의 위치, WWW 서버의 공개키 ID, 암호화한 화일을 넘겨줄 서버의 외부 프로그램의 주소, 그리고 대상화일 이름이 내부적으로 미리 지정되어 있다. 그래서, 웹브라우저 화면에는 위의 내용들이 보이지 않지만 웹브라우저쪽에 위치해 있는 외부 프로그램에서는 이를 이용하여 정보를 전송할 서버에 대한 암호화 및 복호화 동작을 수행할 수 있다. 그러므로 암호화를 요구하는 모든 문서마다 각각의 폼페이지가 필요하게 된다.



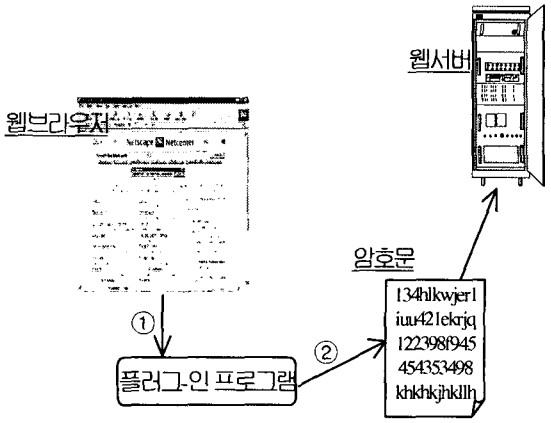
<그림 2> 서버 쪽의 NetCrypt 설계도

그러나, 암호화/복호화를 담당하는 CGI 프로그램은 하나만 존재하면 된다[4]. 웹브라우저로 폼페이지가 전송되면 클라이언트는 폼페이지에 자신의 공개키 ID와 비밀키를 사용하기 위한 패스프레이즈를 입력한다. 이 정보들은 웹브라우저에서의 암호화를 위한 부분으로 이 부분에 대한 설명은 3.3에서 설명하도록 한다. 5번 단계에서 요청 메시지를 암호화해서 서버로 전송하면 이를 받아 들이는 CGI 프로그램에서 복호화하여 대상화일의 위치

와 클라이언트의 공개키 ID 를 얻어 낸다. 대상화일의 위치는 4 번 단계에서 폼페이지를 전송할 때, 내부적으로 웹페이지에 포함된 값이며 공개키의 아이디는 폼필드에 사용자가 입력한 값이다. 6 번 단계에서 CGI 프로그램은 대상화일에 접근하여 화일의 내용을 읽어 이를 암호화한다. 암호화된 내용을 7 번 단계에서 웹브라우저로 다시 전송하게 된다.

3.3 브라우저의 동작 과정.

웹브라우저의 외부 프로그램의 역할을 살펴보면 다음과 같다. 폼페이지의 폼필드에 사용자가 입력한 값을 서버에게 바로 전송하지 않고 그 값을 이용해서 정보를 암호화한 후 전송하는 역할과 서버에서 전송되는 암호화된 화일을 복호화해서 웹브라우저에 출력하는 역할이다. 이 두개의 역할은 하나의 플러그-인 프로그램으로 구현할 수 있다. 이 두개의 역할을 자세하게 살펴보면 다음과 같다. 첫번째로 암호화에 앞선 폼페이지의 폼필드에 입력된 값을 이용하여 메시지를 암호화하여 서버에게 전송하는 역할은 <그림 3>과 같이 2 가지의 단계로 나눌 수 있다. 서버에서 전송된 폼페이지의 폼필드에 사용자가 입력한 값을 클라이언트의 외부프로그램으로 넘겨주는 단계가 .번 단계이다. 사용자가 입력해야 하는 값들은 클라이언트에서 암호화 동작시 사용될 공개키의 아이디와 비밀키를 사용하기 위한 패스프레이즈이다. 그러나 실제로 외부 프로그램으로 넘겨지는 값들은 위의 두 값들만은 아니다. 사용자가 암호화해서 받고자 하는 화일이름, 서버의 공개키 아이디, 암호문을 전송해줄 때 필요한 서버의 주소가 내부적으로 넘어온다. 이 방법은 라이브 코넥트 기법을 이용하여 구현할 수 있다. 라이브 코넥트 기법에 대해서는 4 장에서 설명하도록 한다. 다음으로 1 번 단계를 통해 넘어온 위의 값들을 이용해서 외부 프로그램에서 암호동작을 수행한 후, 서버로 전송해주는 단계가 2 번 단계이다. 2 번 단계를 좀더 자세히 설명하면 다음과 같다.

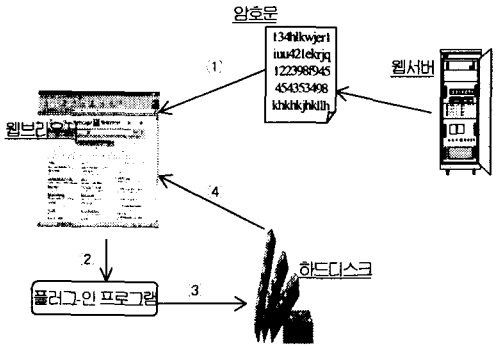


<그림 3> 클라이언트에서 서버로 전송

1 번 단계에서 넘어온 값 중 폼필드에서 입력받은 패스프레이즈와 서버의 공개키 아이디를 이용해서 자신의 공개키 아이디와 받고자 하는 화일이름을 암호화 한다. 위의 암호문을

다시 URL 의 형태로 구성하고 서버로 전송하기 위해 헤더 정보를 붙인다. 헤더 정보는 서버에서 전송된 서버의 주소를 이용한다. 그리고, 그 값을 플러그-인 기법에서 제공하는 모듈을 이용해서 서버의 CGI 프로그램으로 전송한다.

다음으로 서버에서 전송된 암호문을 복호화한 후 브라우저에 출력하는 경우는 <그림 4>와 같이 4 가지의 단계로 나눌 수 있다. 웹서버가 HTTP 를 통해서 웹브라우저에게 암호문을 보내주는 1 번의 단계와 서버에서 전송된 암호문을 웹브라우저의 외부 프로그램으로 넘겨주는 2 번의 단계 그리고, 외부 프로그램에서 복호화 하는 부분과 복호화한 결과를 하드디스크에 저장한 후 웹브라우저에 출력하는 3,4 번의 단계로 나눌 수 있다. 2 번의 단계에서 외부 프로그램의 구동은 서버에서 전송되는 웹페이지에 특정태그를 삽입하여 구현할 수 있고 전송된 화일을 외부 프로그램으로 넘겨주는 것은 플러그-인에서 제공하는 모듈을 통해서 구현할 수 있다. 3 번 단계에서는 넘겨받은 암호문을 자기의 비밀키로 복호화하고 서버의 비밀키로 암호화된 해쉬값을 복호화한 후 원래 메시지의 해쉬값과 비교하여 서명확인과 정보의 무결성을 확인한다



<그림 4> 서버에서 전송된 암호문을 복호화

위의 단계가 정상적으로 동작하면 복호화한 화일을 클라이언트의 하드 디스크에 저장한다. 마지막으로 4 단계에서의 역할은 3 번 단계에서 저장한 결과를 웹브라우저의 화면에 출력한다. 이것은 플러그-인 제공하는 모듈을 통해서 구현할 수 있다.

4. 구현에 필요한 제반 기술

NetCrypt 의 구현에 필요한 기술을 보면 웹브라우저의 외부 프로그램으로 정보를 보내기 위해서 라이브 코넥트 기법이 필요하고, 또한 서버로부터 전송된 정보를 브라우저의 외부 프로그램에서 처리하기 위해 플러그-인 기법이 필요하다. 이에 관한 설명은 다음과 같다.

4.1 플러그-인 기법

본 논문에서는 웹브라우저와 외부 프로그램이 직접적으로 연동하기 위해서 플러그-인 기법이 필요하다[5].

플러그-인기법이란 Netscape 에 의해서 수행되는 C/C++등의 프로그래밍언어로 제작된 동적 프로그램 모듈이다. 이러한 모듈을 개발하기 위해서는 플러그-인 API 라는 일종의 프로그래밍 인터페이스를 사용해야 한다. 이런 API 의 역할은 개발자가 제작한 응용프로그램과 웹브라우저의 핵심기능을 연결시켜 주는 역할을 수행함으로써 웹브라우저의 확장기능을 제공한다. 플러그-인 API 를 이용해 작성된 플러그-인 모듈을 웹브라우저에서 사용하기 위해서는 dll 형태로 프로그래밍되어야 한다. 그리고 만들어진 dll 모듈은 사용자의 하드디스크에 저장된 Netscape 프로그램의 특정부분에 놓여져야 한다. 그 이유는 웹브라우저가 처음 구동될 때 특정 디렉토리 밑의 플러그-인 모듈들만을 인식하기 때문이다.

4.2 라이브 코넥트

서버에서 전송된 폼페이지에 사용자가 정보를 입력한후 submit 버튼을 눌렀을 경우에 바로 서버에게 입력한 값을 전송하지 않고 먼저 클라이언트의 외부 프로그램 즉 플러그-인 프로그램으로 전송하기 위해서는 라이브 코넥트 기법이 필요하다. 라이브 코넥트 기법이란 플러그-인 기법 중 하나로 자바와 자바스크립, 플러그-인 사이의 상호통신 메커니즘을 의미한다. 자바스크립과 플러그-인 프로그램과는 직접적으로 통신이 되지 않기 때문에 자바를 통해서 통신을 해야한다. 즉 자바와 자바스크립을 이용해서 플러그인을 제어하거나 반대로 플러그-인을 통하여 자바와 자바스크립을 제어 할 수 있는 기능을 제공한다[6]. 라이브 코넥트 기법을 구현하기 위해서는 먼저 자바와 C/C++사이의 상호 인터페이스가 정의되어야 한다. 먼저 자바화일을 만들고 나서 jri 버전의 자바 컴파일러를 이용하여 자바화일을 컴파일하여 자바 클래스를 생성한다. 그리고 jri 에서 제공하는 javah 를 가지고 자바클래스를 다시 컴파일하면 C 화일과 헤더화일이 생성이 된다[7]. 이때 생성된 C 화일과 헤더화일을 가지고 플러그-인 모듈을 이용하여 플러그-인 프로그램을 구현한다. 마지막으로 생성된 플러그-인 프로그램을 브라우저에서 이용하기 위해서는 만들어진 dll 화일과 class 화일을 Nescape 의 특정 디렉토리에 저장해야 한다.

5. 결론

본 논문에서는 기존의 WWW 보안 방법과는 다른 형태의 방법을 제안하고 있다. 제안된 NetCrypt 는 기존의 HTTP 와 WWW 서버, WWW 웹브라우저에 수정을 요구하지 않으면서, 단지 외부 프로그램을 추가하면 사용할 수 있는 WWW 보안 기술이다. 본

논문에서 구현한 외부 프로그램의 주된 역할은 크게 2 가지이다. 하나는 사용자가 입력한 폼필드의 값을 이용해 브라우저에서 정보를 암호화해서 전송하는것이고, 두번째는 서버로부터 전송된 암호문을 복호화해서 브라우저에 전송해주는 것이다. 위의 외부 프로그램을 구현하기 위해 플러그-인 기법이 필요하다. 브라우저에서 전송된 암호문을 복호화해서 그 결과를 이용하여 다시 암호화해서 브라우저에게 보내주는 기법은 CGI 기법을 이용하였다. NetCrypt 의 특징은 기존의 WWW 보안프로토콜들과는 달리 독립적인 암호 모듈을 이용하여 개발자의 의도대로 암호 통신을 설계할 수 있기 때문에 SSL 처럼 통신의 채널을 획일적으로 모두 암호화하는 방식과 구별된다. 보안 서비스를 선택하여 사용할 수 있고, 암호화하려고 하는 정보와 그렇지 않은 정보를 선택할 수 있다는 장점이 있다. 이러한 장점은 곧 전자 상거래에서의 지불 시스템과 같은 WWW 을 기반으로 한 응용 시스템의 구현시에 SSL 과같은 기존의 보안 프로토콜을 사용하는 것보다 더 높은 융통성과 활용도를 제공한다.

참고문헌

[1] S-HTTP Secure Connection "http://epmnas.epm.ornl.gov /examples/shttp_test.html"

[2] Overview of SSL3.0

[3] "http://developer.netscape.com /misc/developer/conference/proceedings/cs2/index.html"

[4] "The LiveConnect/Plug-In Developer's Guide", Netscape corp,

[5] Gundavaram,"CGI PROGRAMMING on the WWW", O'Reilly & Associates,1996

[6] Oliphant, "프로그래밍 넷스케이프 플러그-인" ,인포북., 1997

[7] Richard Wagner, "자바스크립 언리쉬드", 대림, 1998

[8] 황부현, "JAVA 프로그래밍",정익사,1998