웹 환경에서 안전한 데이타 전송을 보장하는 프로토콜 기반의 보안 모듈에 근거한 보안 시스템

(Security System using Protocol-Based Security Module for Secure Data Transmission in Web Environment)

장 승 주 † 임 동 훈 ††
(Seung Ju Jang) (Dong Hoon Lim)

요 약 본 논문은 웹 시스템 환경에서 안전한 데이타 전송을 만족하는 Protocol-Based Security Module(PBSM) 구조를 제안한다. 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. WSCSM 보안 모듈은 WSSM으로부터 받은 암호된 메시지를 정상적인 메시지로 변환하여 웹 브라우저에 나타나게 한다. WSSM 보안 모듈은 HTML 파일에 대한 암호화(Encryption)기능과 WSCSM 모듈로부터 받은 Common Gateway Interface(CGI) 데이타에 대한 복호화 기능을 가지고 있다. PBSM 보안 시스템의 보안 정확성을 검증하기 위하여 정형화 분석 기법을 이용했다.

키워드 : PBSM 보안 시스템, WSSM 모듈, WSCSM모듈, 웹 보안 모듈

Abstract We propose the PBSM(Protocol-Based Security Module) system which guarantees the secure data transmission under web environments. There are two modules in 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 function for HTML file and the decryption function for CGI(Common Gateway Interface). The formal analysis methodology is imported from formal theory for analyzing the data flow of the PBSM system. The formal analysis methodology is based on the order theory.

Key words : PBSM security system, WSSM module, WSCSM module, web security module

1. 서 론

최근에 많은 사람들이 웹 시스템을 이용하여 중요한 데이타를 전달하고 있다. 웹 환경은 편리성이라는 측면에서 많은 사람들이 쉽게 접근하여 사용하고 있다. 그러나 이러한 편리성에는 여러가지 보안 문제점을 가지고 있다. 편리한 측면이 많다보니 불순한 의도를 가진 사람이 쉽게 시스템에 접근할 수 있는 가능성이 높다는 것이다. 특히 최근에는 많은 금융 시스템들을 웹과 연결하여 사용자들에게 편리한 업무 서비스를 제공하고 있다.

웹 보안과 관련한 연구는 여러 가지 형태로 이루어지고 있다[1,2,3,4]. 그 중의 하나가 네트워크 자체의 보안에 대한 연구가 주류를 이루고 있다[5,6,7,8,9]. 네트워크 보안으로 많이 사용되고 있는 기법이 방화벽(fire wall)이다[10]. 따라서 최근 들어서 네트워크 보안 만으로 부족하다는 인식이 확산되면서 데이타 보안에 대한 연구가 진행되고 있다. 데이타 보안은 주로 네트워크로 흘러가는 데이타에 대한 보안에 초점을 두고 있다. 이러한 데이타 보안 기법의 한 방법으로 SSL(Secure Socket

· 이 논문은 한국과학재단의 해외 Post-doc. 연수지원에 의하여 연구되었음
† 정 회 원 : 동의대학교 컴퓨터공학과 교수
sjjang@dongeui.ac.kr
†† 비 회 원 : 경상대학교 통계정보학과 교수
dhlim@nongae.gsnu.ac.kr
논문접수 : 2002년 2월 8일
심사완료 : 2002년 8월 9일

Layer) 프로토콜[11, 12]이나 S-HTTP 프로토콜[13]을 이용한 웹 시스템이 늘어나고 있다. 현재 웹 보안 시스템 환경을 구축하고자 할 경우에 복잡함과 보안 키(key) 관리에 따른 문제점 등이 있다[1,3,4,14,15, 16]. 또한 최근에는 네트워크 프로토콜 상에서 보안 기능의 구현을 통해서 웹 보안을 만족시키는 연구들이 진행되고 있다. 특히 TCP, UDP 프로토콜 상에서 보안 기능의 구현을 통해서 웹 보안에 주력하고 있다. 또한 IPSec 프로토콜과 같은 보안 네트워크 프로토콜의 구현에 대한 연구가 활발히 진행되고 있다.

그리고 최근에는 웹을 위한 보안 방법에 대한 연구가 활발히 진행되고 있다. 특히 SSL 등을 활용한 보안이 많이 활용되고 있다. 그러나 SSL 프로토콜과 같은 웹 보안은 기존의 특정한 웹 브라우저를 이용한 방식으로 보안 환경에 유연성 등을 제공하는데 한계가 있다.

본 논문은 이러한 웹 시스템 환경에서 안전한 데이타 전송을 만족하는 PBSM(Protocol-Based Security Module) 구조를 제안한다. PBSM 구조는 분산 시스템 구조에서 프로토콜 중심의 보안 환경을 제공한다. PBSM 구조는 크게 두개의 모듈로 구성된다. 하나는 웹 서버에서 동작하는 WSSM(Web Server Security Module)이고, 다른 하나는 클라이언트에서 동작하는 WSCSM(WinSock Client Security Module)이다.

표 1 WSSM, WSCSM모듈에서 사용하는 각 파라메터의 설명

변 수	설 명
Ps(m_RSA, E)	서버에서 클라이언트로 전송되는 HTML형태의 모든 메시지에 대한 암호화를 적용하기 위한 *m_RSA* 암호화 알고리즘
Ps(m_RSA, D)	암호화된 CGI 폼 데이타에 대한 RSA 복호화 알고리즘
Ps(SHA-1)	메시지가 전송 도중 변경되지 않음을 증명하기 위한 SHA-1 해쉬 알고리즘
Ps(Digital Signature), Pc(Digital Signature)	전자 서명기능
Pc(m_RSA, E)	CGI 폼에 입력된 암호화된 사용자 정보의 보안을 위한 *m_RSA* 암호화 알고리즘
Pc(m_RSA,D)	서버에서 클라이언트로 전송되는 HTML 형태의 모든 메시지에 대한 복호화를 위한 *m_RSA* 암호화 알고리즘
Pc(SHA-1)	메시지가 전송 도중 변경되지 않음을 증명하기 위한 SHA-1 해쉬 알고리즘

WSSM 보안 모듈을 SMs 집합으로 표시하면 *SMs={Ps(m_RSA, E), Ps(m_RSA, D), Ps(SHA-1), Ps (Digital Signature)}*이다[17]. WSCSM 모듈은 Win socket의 Layer Service Provider(LSP)기능을 이용한다[18,19,20,21,22,23]. WSCSM 보안 모듈을 *SMc* 집합으로 표시하면 *SMc={Pc(m_RSA, E), Pc(m_RSA, D), Pc (SHA-1), Pc(Digital Signature)}*이다.

본 논문의 구성은 2장 PBSM 시스템 구조, 3장 WSSM, WSCSM 구조, 4장 PBSM 데이타 모델 및 보안 정확성 검증, 5장 결론의 순으로 언급한다.

2. PBSM 시스템 구조

PBSM 시스템의 구조는 다음 그림 1과 같다.

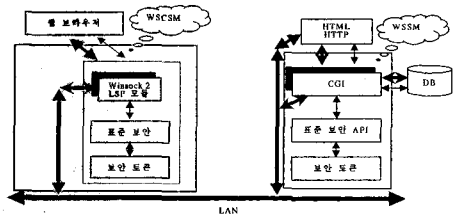

그림 1 PBSM 시스템 구조(굵은 선 : 웹 시스템의 정상적인 자료 흐름, 가는 선 : PBSM 시스템 데이타 흐름)

WSSM 모듈은 웹 모듈에 종속되면서 독립적인 모듈로 존재한다. 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 요소를 *SMs*으로 표시할 수 있다(표 1).

WSCSM 모듈은 Windows Socket2의 LSP(Layered Service Provider)를 기반으로 한다. Windows Sockets2는 네트워크 프로그래밍 인터페이스로서 OSI 계층도에 있어서 전송 계층(transport layer)에 존재한다. 그리고 응용 프로그램 제작자들을 위한 API(Application Program Interface)와 트랜스포트 스택을 제작하는 제작자들을 위한 SPI(Service Provider Interface)가 제공된다[24,25]. WSCSM 모듈은 Winsocket의 LSP기능을 이용한다. 다음 그림 2는 Windows socket 2의 구조를 나타낸다. WSCSM 보안 모듈의 구성 요소를 *SMc* 표시할 수 있다(표 1).

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. WSSM 서버가 서버와 클라이언트에게 키 분배의 역할을 담당한다. 즉, WSSM 서버는 서버와 클라이언트에게 키를 배분하는 역할을 담당한다. 독립적인 키 서버를 두지 않고 WSSM서버가 이 역할을 대신하는 것이다. 이러한 키 배분 방식은 일반적으로 키 서버를 사용하는 웹 시스템의 경우보다 단순하고 키의 관리가 용이하다. 본 논문에서 사용하는 m_RSA 알고리즘은 RSA 알고리즘을 WSSM, WSCSM 모듈에 그대로 사용할 경우에 발생되는 성능 저하 요인을 줄이기 위한 것이다. RSA 알고리즘의 키 관리 부분에 대한 부담을 줄임으로써 성능 저하를 가능하면 줄일 수 있도록 하였다.

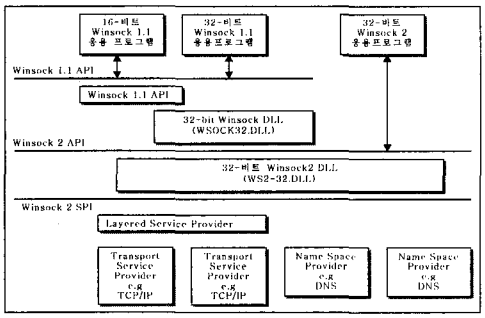

그림 2 Winsock 2 구조

3. WSSM, WSCSM 구조

3.1 기호 표기

본 논문에서 사용하는 기호 표기법은 일반적인 방법을 사용한다. 수학에서 사용하는 집합 기호 등을 범용적으로 인용하여 사용한다. 그리고 각 기호의 의미는 기호가 사용되는 곳에서 간단히 언급한다.

3.2 PBSM 시스템 설계

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

[정의 1] 메시지(message)

1. Type t를 갖는 기본 메시지는 $m{:}t$로 표기한다.
   1. 만약 $m_1{:}t_1$ 과 $m_2{:}t_2$가 메시지이면 메시지의 연결(concatenation) $m_1m_2{:}t_1t_2$도 메시지이다.
   3. 만약 $f : t_1 x \cdots x\ t_n \rightarrow t$가 cryptographic 함수이고 $m_i{:}t_i$가 메시지이면 $m_1, \cdot, m_n$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 기호로 사용한다.) 따라서, 만약 $m_1 \cong m_2$이면 $m_1 = m_2$이다.

그리고 본 논문은 오토마타로써 프로세스가 동작하는 과정을 상태(state)로 정의한다. PBSM 시스템 동작 과정은 [정의 2]와 같다.

[정의 2] ρ를 프로세스로 두고, ρ는 <Σ, E, r,Σ0>의 tuple로 둔다.

Σ : 가능한 모든 상태의 집합

E : event의 집합

여러가지 event의 타입이 있다.

-*SENDi(ρ, m)* : i 번째 트랜잭션에서 ρ프로세스가 m 모듈을 통해서 메시지를 전송

-*RECEIVEi(ρ, m)* : i 번째 트랜잭션에서 ρ 프로세스가 m 모듈을 통해서 메시지 수신

-*ACTIVE(Ti)* : i 번째 트랜잭션을 활성화

-*RESULTi(m)* : i 번째 트랜잭션에서 ←의 오른쪽 실행 결과를 m 모듈에 저장

-*ACKi(MSG, m)* : 메시지 MSG를 m 모듈로부터 기다린다.

-*MSG* : 두가지 종류 메시지 (HEAD, 메시지 body)

r : type Σ x E x Σ transition relation

Σ0⊆Σ : 초기상태의 집합

WSSM 모듈과 WSCSM 모듈 사이의 데이타 흐름은 그림 4와 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 전달이 된다.

WSCSM 모듈로부터 자료를 넘겨받은 WSSM 모듈은 보안 모듈에서 사용자가 요청한 내용을 분석하기 전에 복호화 작업을 수행한다. 복호화 작업을 통해서 정상

적인 데이타로 변환이 된후에 web server모듈은 정상적인 문자 처리 과정에 의하여 사용자의 요구 사항을 처리한다. 사용자가 요청한 자료를 WSCSM모듈로 전송하기 전에 WSSM 모듈에서 보안 처리 과정을 거치게 된다. WSSM 보안 모듈에서 보안 처리된 자료는 WSCSM 시스템으로 전송된다. WSSM모듈로부터 자료를 넘겨 받은 WSCSM모듈은 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 데이타를 출력하게 된다.

WSSM모듈과 WSCSM 모듈 내의 m_RSA 암호화 알고리즘에 사용되는 공개키와 비 공개키의 길이는 512 비트로 고정하여 사용한다. 아래의 WSSM, WSCSM 시스템에 대한 데이타 흐름을 표시하기 위해서 Ω 표기법을 사용한다. Ω 표기법은 데이타의 흐름을 표시하는 기호로 사용한다. $\Omega_i(x)$는 아이템 x에서의 어떤 연산 Ω_i를 나타낸다. Γ는 트랜잭션의 동작을 표현하는 기호로 사용한다. Γ_i는 i번째 연산(Oi)에 대한 트랜잭션이다. 연산 $\Omega \in \{r(x), w(x)\}$ 으로 읽기 연산(r(x)) 과 쓰기 연산(w(x))으로 구성되어 있다. 그림 4에서 WSSM, WSCSM 모듈 사이의 데이타 흐름을 오토마타 event 타입을 이용해서 나타내면 다음 그림 3과 같다.

- (1)-(2) : 원래 메시지를 단방향 해쉬 함수를 거쳐서 새로운 메시지를 만든다.
- (3)-(5) : (1), (2) 결과 만들어진 메시지를 암호화시킨다. 이때 메시지를 보내는 사람의 개인 키를 추가하여 디지털 서명을 한다.
- (6) : 디지털 서명된 정보를 원본 메시지에 추가시킨다.
- (7)-(8) : (6)에서 생성된 메시지를 받는 사람의 공공키를 가지고 암호화시킨다.
- (9) : 메시지를 인터넷으로 전송
- (10)-(11) : 메시지를 받은 사람은 받은 메시지를 가지고 자신의 개인 키를 이용하여 복호화시킨다.
- (12) : 원본 메시지와 디지털 서명된 메시지를 분리시킨다.
- (13)-(16) : 단방향 해쉬 함수를 사용하여 만들어진 메시지와 디지털 서명 메시지를 보낸 사람의 공공 키를 가지고 복호한다. 복호된 메시지와 해쉬 함수를 사용하여 복호된 메시지 간에 상호 비교하여 틀린 부분이 없는지를 점검한다.

Ω_i(wscsm) : ACTIVE(Γ_i)$\in\Omega_i$(wscsm)
RESULT$_i$(wsock-api) ←ACTIVE(Γ_i(wsock-api))
RESULT$_i$(wsock-spi)←ACTIVE(Γ_i(RESULT$_i$(wsock-api))
RESULT$_i$(SMc) ←ACTIVE(Γi(SM$_c$))
SEND$_i$(RESULT$_i$(SM$_c$), WSSM)

Ω_i(WSSM) : ACTIVE(Γ_i)$\in\Omega_i$(WSSM)
RECEIVEi(RESULT$_i$(SM$_c$))
RESULT$_i$(MSG(E)) ←ACTIVE(Γi(RESULTi(SM$_c$)))
RESULT$_i$(MSG(T)) ←ACTIVE(Γi(SM$_s$, MSG(E)))
ACK$_i$(HTTP$_{HEAD}$, wscsm)

그림 3 WSCSM과 WSSM내의 데이타 흐름

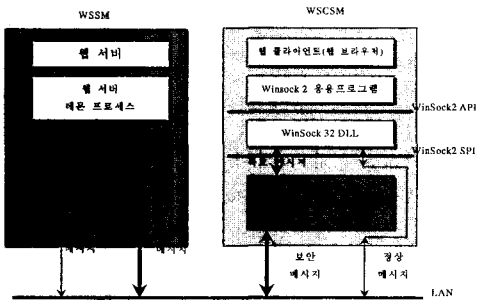

그림 4 WSSM, WSCSM 모듈 내부 구조(가는 선 : 웹 시스템의 정상적인 자료 흐름, 굵은 선 : PBSM 시스템 데이타 흐름)

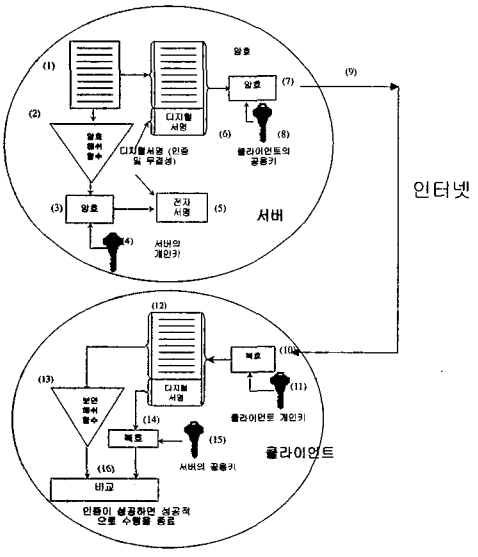

그림 5 PBSM 시스템의 논리적인 동작 구조

3.3 WSSM 보안 모듈 설계

WSSM모듈은 WSCSM모듈의 데이타 요청이 발생하면 이 요청에 대한 응답을 보낸다. 서버의 WSSM 보안 모듈은 WSCSM모듈에서 요청한 HTML(HyperText Markup Language)파일을 WSSM 보안 모듈 내에서 보안 처리하여 WSCSM모듈로 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 데이타 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. 이 과정을 함수 관계로 표시하면 다음 수식 (1)과 같다.

$$\left.\begin{array}{l} E_{WSSM}(file(x))=m\_RSA_{WSSM}(file(x),\ E)\notin H_{http}(x) \\ D_{WSSM}(file(x))=m\_RSA_{WSSM}(file(y),\ D)\in H_{http}(y) \end{array}\right\} \quad (1)$$

알고리즘 1은 WSSM 보안 모듈의 동작 과정을 보여준다.

3.4 클라이언트 WSCSM 보안 모듈 설계

클라이언트(WSCSM)는 웹 브라우저를 이용해서 웹 서버(WSSM)에게 HTML 문서를 요청한다. 보안 모듈이 설치되어 있지 않을 경우 클라이언트의 웹 브라우저에 암호화된 데이타($E_{WSSM}(file(x))$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.

```
Input : MSG_IM = HTTP_HEAD + m_RSA(E(m1:t)),
                 HTTP_HEAD ∈ WSSM
                 HTTP_HEAD + m1:t, HTTP_HEAD ∉ WSSM
Output : MSG_TM = HTTP_HEAD + E(m2:t),
                  HTTP_HEAD ∈ WSSM
                  HTTP_HEAD + m2:t, HTTP_HEAD ∉ WSSM
                  HTTP_HEAD, HTTP_HEAD ∉ WSSM,
                  CGI 데이타인 경우
Algorithm
Begin
(1) MSG_IM 중에서 HTTP_HEAD를 분리
(2) If HTTP_HEAD ∈ WSSM module begin
(3)       m1:t := m_RSA (E(m1:t), D)
(4)       m1:t 내용을 분석
(5)       if m1:t ∈ CGI begin
(6)             write(m1:t);
(7)             Ack(HTTP_HEAD, wscsm), return;
(8)       end
(9)       m2:t:=read(m1:t)
(10)      E(m2:t) := m_RSA (m2:t)
(11)      send(HTTP_HEAD + E(m2:t))
(12)      return;
(13) end
(14) m4:t:=read(m3:t)
(15) send(HTTP_HEAD + m4:t)
End.
```

알고리즘 1 WSSM 보안 모듈의 동작

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 알수 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 문서에 대한 복호화는 클라이언트의 비공개 키 값($key(i)$)으로 수행된다. 복호화된 HTTP 문서는 API를 통하여 브라우저에 정상적

인 메시지로 보여진다. 만일 HTML 문서를 복호화 한 후 메시지의 내용이 깨어지거나 이상한 문자가 나타날 경우는 메시지가 네트워크를 통한 전송과정 중 이상이 발생했음을 의미한다. 수식 (2)는 이러한 과정을 수식으로 표현한 것이다.

$$\left.\begin{aligned} &SHA\text{-}1_{wscsm}(x,\ key(i)),\ x\in HTTP_{head},\ x\in HTTP_{body} \\ &E_{wscsm}(x) = m\_RSA_{wscsm}(x,\ E),\ x\notin HTTP_{head},\ x\in HTTP_{body} \\ &D_{wscsm}(x) = m\_RSA_{wscsm}(x,\ D),\ D_{wscsm}(x)\in\{HTTP_{head},\ HTTP_{body}\} \end{aligned}\right\} \quad (2)$$

그리고 클라이언트 보안 모듈(WSCSM)은 보안 웹 서버(WSSM)와 일반 서버로 데이타가 각각 전달될 수 있도록 URL 구분기능을 갖는다. 즉, 보안 사이트와 일반 사이트의 연결은 URL DB 기능을 이용하여 수행된다. 일반 데이타가 보안 모듈을 거치는 경우나 보안 데이타가 일반 모듈을 거치는 경우를 방지하기 위해서 클라이언트에서 서버로 데이타를 보내기 전에 HTTP 헤더의 URL과 WSSM 서버로 등록된 URL(URL DB)을 비교한다. 클라이언트에서 서버로 보내고자 하는 데이타가 WSSM 서버로 URL DB에 등록된 경우에는 클라이언트의 보안 모듈(WSCSM)을 거치도록 하며, URL이 WSSM 서버로 URL DB에 등록되지 않은 경우는 일반적인 데이타 전송 과정을 거친다. 이 과정은 그림 1, 그림 4에 나타나 있다. 그리고 WSSM 서버 URL인지 일반 서버 URL인지를 구분하는 알고리즘은 다음 그림 6과 같다.

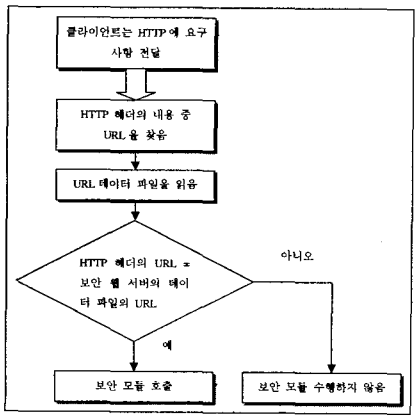

그림 6 URL 필터링 처리 과정

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 데이타를 입력받아서 서버(WSSM)의 CGI 처리 모듈로 전송된다.

클라이언트 WSCSM 모듈은 사용자가 폼에 관련 데이타를 입력하고 확인 버튼을 누르면 데이타가 네트워크로 전송되기 전에 암호화(m_RSAwscsm)를 수행하여 암호화 된 데이타(Ewscsm(file(x))가 서버(WSSM)로 전송된다. 사용자 인증 정보를 서버로 전송하는 경우에 보안 데이타 처리를 함으로써 안전한 데이타 전송을 보장한다. 알고리즘 2는 WSCSM 보안 모듈의 동작 과정을 보여준다.

Input : MSG_{IM} = $HTTP_{HEAD}$ + m_1:t, $HTTP_{HEAD}\notin$ {WSSM, WSCSM}
$HTTP_{HEAD}$ + *m_RSA*(E(m_1:t)), $HTTP_{HEAD}\in$ {WSSM, WSCSM}
Output : MSG_{TM} = $HTTP_{HEAD}$ + E(m_2:t), $HTTP_{HEAD}\in$WSSM
$HTTP_{HEAD}$ + m_2:t, $HTTP_{HEAD}\notin$ WSSM

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

알고리즘 2 WSCSM 보안 모듈의 동작

4. PBSM 데이타 모델의 보안 정확성 검증

PBSM 시스템의 데이타 흐름을 분석하기 위하여 정형화 분석(formal analysis) 기법을 도입한다. 정형화 분석 기법은 순서 이론(order theory)을 바탕으로 하고 있다. 정형화 기법을 이용하여 PBSM 시스템에 대한 데이타 흐름을 모델화시킬 경우 두 가지 데이타 모델이 가능하다. 하나는 클라이언트에서 서버로의 데이타 흐름 모델이고, 다른 하나는 서버에서 클라이언트로의 데이타

먼저 클라이언트에서 서버로의 데이타 흐름 모델을 오토마타(automata) 형식으로 표현하면 다음 그림 7과 같다.

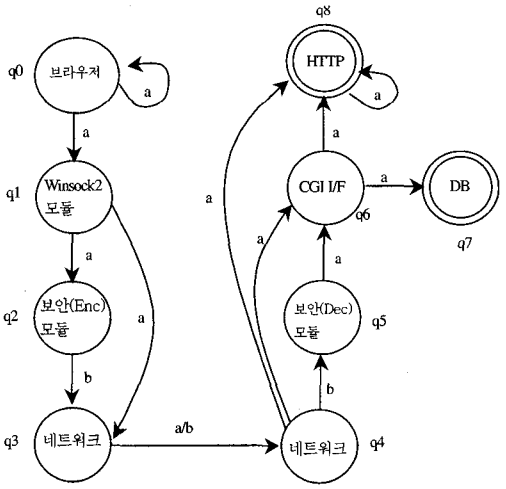

그림 7 서버에서 클라이언트로의 오토마타
(a, b는 데이타)

이 그림에 대해서 데이타 흐름을 정형화하면 다음과 같이 정의할 수 있다.

A = a*b∪ab*∪a*b*
Q = {q0, q1, q2, q3, q4, q5, q6, q7, q8}
F = {q7, q8}
S = q0
Σ = {a, b}
△ = (Q X Σ) X Q

△	A	b
q0	{q0, q1,q3}	Φ
q1	{q2}	Φ
q2	Φ	{q3}
q3	{q4}	{q4}
q4	{q8}	{q5}
q5	{q6}	Φ
q6	{q7, q8}	Φ
q7	Φ	Φ
q8	{q8}	Φ

서버에서 클라이언트로의 데이타 흐름 모델을 오토마타(automata) 형식으로 표현하면 다음 그림 8과 같다.

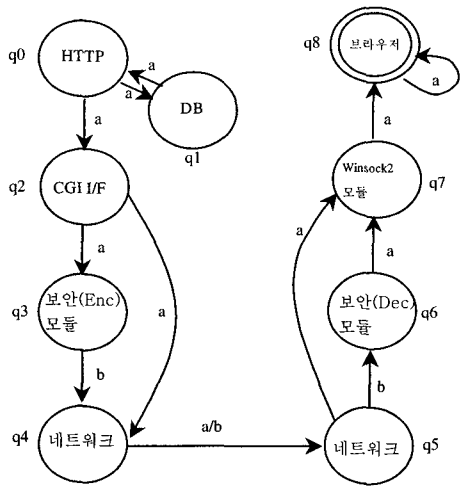

그림 8 서버에서 클라이언트로의 오토마타
(a, b는 데이타)

이 그림에 대해서 아래와 같이 데이타 흐름을 정형화 할 수 있다.

A = a*b∪ab*∪a*b*
Q = {q0, q1, q2, q3, q4, q5, q6, q7, q8}
F = {q8}
S = q0
Σ = {a, b}
△ = (Q X Σ) X Q

△	A	b
q0	{q1, q2,q4}	Φ
q1	{q0}	Φ
q2	{q3, q4}	{q4}
q3	Φ	{q5}
q4	Φ	{q6}
q5	{q7}	Φ
q6	{q7}	Φ
q7	{q8}	Φ
q8	{q8}	Φ

4.1 PBSM 시스템의 정확성(correctness)

PBSM 시스템은 웹에서 클라이언트와 서버 간에 데이타를 주고 받는 경우에 안전한 데이타 전송을 보장하기 위해서 클라이언트(WSCSM)와 서버(WSSM) 보안 모듈을 가지고 있다. WSCSM과 WSSM보안 모듈에서 핵심적인 기능은 m_RSA_{WSSM}과 m_RSA_{wscsm} 모듈이다.

PBSM 시스템의 보안 정확성(correctness) 검증은 *m_RSA* 모듈의 보안 정확성 검증으로 충분하다. 따라서 본 논문에서 사용하는 *m_RSA* 보안 모듈을 중심으로 안정성을 검증한다.

m_RSA 알고리즘은 수식 (3)의 트랩 도어 함수를 사용한다.

$T_{i,n}(x) = x^i \ MOD \ n$

(여기서 i, n, x는 정수형 변수) (3)

i와 n은 public key를 형성한다. Secret key는 n과 다른 수 j에 의하여 구성된다. Secrect key는 다음의 (4) 수식과 같이 생성된다. 수식 (4)에서 (n,i,j)는 키의 쌍이다.

$x^{ij} \equiv x \ MOD \ n$ (4)

키의 생성을 위하여 다음의 과정을 거친다. (n,i,j)쌍의 값은 $x^{ij} \equiv x \ MOD \ n$를 만족하는 값을 선택한다. (n,i,j) 키의 쌍은 p, q 두 솟수(prime number)를 찾는 것으로 가능하다. 그리고 두 솟수를 곱해서 $n(n=pq)$을 만들 수 있다. 그리고 일반적으로 i는 3을 선택한다. 그리고 j는 다음의 확장 유클리드 알고리즘 (4), (5)에 의해서 구해진다.

이 알고리즘은 두 정수 x, y의 GCD(Greatest Common Divisor) h를 계산하고 두 정수 s와 t를 찾는다.

$sx + ty = h$ (5)

우리는 이 함수 (3)에 $x=i$와 $y=(p-1)(q-1)$를 적용한다. 만약 GCD h가 1이 아니면 우리는 다음 수 i를 찾아야 한다. h=1인 경우 s와 t에 대한 함수를 다음 수식 (4)와 같이 표현할 수 있다.

$si + t(p-1)(q-1) = 1$ (6)

또한 우리는 $j=s$를 취할 수 있다.

[정리Theorem] Fermat's Little 정리

어떤 솟수 p와 어떤 수 $a \neq 0$에 대해서 다음의 수식을 만족한다.

$a^{p-1} \equiv 1 \ MOD \ p$ (7)

(증명)

오일러 함수를 $\Phi(n)$이라 $a^{\Phi(p)} = 1 (mod \ p)$이다(단 a와 p는 서로소). 여기서 오일러 함수 $\Phi(n)$은 n이하의 자연수 중 n과 서로소인 수들의 갯수를 말한다. 먼저 보조정리를 보임으로써 이 증명을 마치고자 한다.

[Corollary] $A=\{r_1\Phi(1), r_2\Phi(2), \ldots, r_n\Phi(n)\}$이 n의 축소 잉여계일 때 $(a,n)=1$인 a에 대하여 $B=\{ar_1\Phi(1), ar_2\Phi(2), \ldots, ar_n\Phi(n)\}$도 n의 축소 잉여계이다.

(증명) 만약 $ar_1 \equiv ar_2 (mod \ n)$이라 하면 $r_1 \equiv r_2 (mod \ n)$이므로 모순이다. 위에서 B의 원소를 모두 곱하면 $ar_1\Phi(1)*ar_2\Phi(2)*\ldots*ar_n\Phi(n) \equiv r_1\Phi(1)*r_2\Phi(2)*\ldots*r_n\Phi(n) (mod \ p)$, 즉, $a^{\Phi(n)} \equiv 1 (mod \ n)$ (양변을 $r_1\Phi(1)*r_2\Phi(2)*\ldots*r_n\Phi(n)$로 나눈다. p의 축소 잉여계의 원소의 곱과 n와는 서로소이기 때문에) 그런데 n가 소수이면 1부터 $(n-1)$까지는 모두 n와 서로소이다. 즉 $\Phi(n)=n-1$이 된다.

[Correctness of *m_RSA*]

$E(m_1{:}t, PKEY_r) = E(m_1{:}t)$ $D(E(m_1{:}t), SKEY_r) = m_1{:}t$

$(m_1{:}t)^e = E(m_1{:}t)$ $(E(m_1{:}t))^d = m_1{:}t$ (8)

즉, 우리는 $(m_1{:}t)^{ed} = (m_1{:}t) \ (mod \ n)$ for all $(m_1{:}t)$를 증명하는 것이 필요하다.

$n = pq$ and $de = 1 \ mod \ (p-1)(q-1)$ (9)

이것은 $de = k(p-1)(q-1) + 1$를 의미한다.

Case 1: $(m_1{:}t) \sim= 0 \ mod \ p$

$$(m_1{:}t)^{ed} = (m_1{:}t)^{k(p-1)(q-1)+1} \ mod \ p$$
$$= ((m_1{:}t)^{(p-1)\,k\,(q-1)} (m_1{:}t)) \ mod \ p$$
$$= ((1)^{k(q-1)} (m_1{:}t)) \ mod \ p$$

by Format little theorem

$$= m_1{:}t$$

Case 2: $(m_1{:}t) = 0 \ mod \ p$

$(m_1{:}t)^{ed} = (m_1{:}t) \ mod \ p$

$(m_1{:}t)^{ed} = (m_1{:}t) \ mod \ p$ for all $(m_1{:}t)$

Case 1과 2의 경우를 조합하면,

$(m_1{:}t)^{ed} = (m_1{:}t) \ mod \ pq$ for all $(m_1{:}t)$ by Chinese Remainder Theorem (10)

따라서 *m_RSA* 알고리즘의 정확성은 증명이 되었다.

[Correctness of the PBSM]

E_ORD_{wssm} = {Br, Ws_{wscsm}, P_{wscsm}, TCP_{wscsm}, TCP_{wssm}, P_{wssm}, {CGI, DB}, $HTTP_{wssm}$, m1:t} (11)

D_ORD_{wscsm} = {$HTTP_{wssm}$, {DB, CGI}, P_{wssm}, TCP_{wssm}, TCP_{wscsm}, P_{wscsm}, Ws_{wscsm}, Br} (12)

수식 (11), (12)에서 E_ORD_{wssm}는 PBSM 시스템에서 암호화 과정에 대한 순서 집합이다. D_ORD_{wscsm}는 PBSM 시스템에서 복호화 과정에 대한 순서 집합이다. (Br : web browser, Ws : Winsock, TCP : TCP protocol, HTTP : HTTP protocol)

순서 집합, E_ORD_{wssm}과 D_ORD_{wscsm}은 disjoint union set 이다. 그래서 우리는 수식 (11), (12)를 다음과 같이 수정할 수 있다.

$E(m_1{:}t, Br) \times E(m_1{:}t, W_{wscsm}) \times E(m_1{:}t, TCP_{wscsm}) \times E(m_1{:}t, TCP_{wscsm}) \times E(m_1{:}t, TCP_{wssm}) \times E(m_1{:}t, P_{wssm}) \times E(m_1{:}t, \{CGI, DB\}) \times E(m_1{:}t, HTTP_{wssm})$ (13)

우리는 $E(m_1{:}t, P_{wssm})$ 함수를 제외하고 $m_1{:}t$ 결과를 얻을 수 있다. 즉, 모든 함수의 결과는 $E(m_1{:}t, P_{wssm})$ 함수를 제외하고 동일한 집합을 갖는다. 따라서 PBSM 시스템의 정확성을 증명이 된 것이다.

5. 결 론

본 논문은 웹 환경에서 안전한 데이타 전송을 위한 WSSM, WSCSM 보안 모듈을 설계하였다. WSSM, WSCSM 보안 모듈은 공개키를 기반으로 기밀성, 무결성의 기능을 제공한다. PBSM 시스템 내의 보안 모듈은 서버와 클라이언트로 구분을 한다. WSSM모듈은 WSCSM모듈의 데이타 요청이 발생하면 이 요청에 대한 응답을 보낸다. 서버의 WSSM 보안 모듈은 WSCSM모듈에서 요청한 HTML(HyperText Markup Language) 파일 이나 데이타를 보안 처리하여 WSCSM모듈로 전송한다. WSSM 보안 모듈은 HTML 파일에 대한 암호화 기능과 WSCSM 모듈에서의 CGI 응답에 대한 복호화 기능을 갖고 있다.

클라이언트(WSCSM) 보안 모듈은 보안 웹 서버(WSSM)로부터 받은 HTML 문서에 대한 복호화 기능을 갖는다. WSCSM 보안 모듈은 WSSM으로부터 받은 암호된 메시지를 정상적인 메시지로 변환하여 웹 브라우저에 나타나게 한다. WSCSM 보안 모듈은 암호화된 HTML 문서를 복호화하기 위하여 API의 하부 레벨인 SPI에서 동작하는 LSP기능을 이용한다.

그리고 클라이언트 보안 모듈(WSCSM)은 보안 웹 서버(WSSM)와 일반 서버로 데이타가 각각 전달될 수 있도록 URL 구분기능을 갖는다. 즉, 보안 사이트와 일반 사이트에 대한 URL DB 구축을 통해서 정상적인 동작이 가능하도록 한다.

PBSM 시스템의 데이타 흐름을 분석하기 위하여 정형화 분석(formal analysis) 기법을 도입했다. 그리고 이 방법을 이용하여 PBSM 시스템의 정확성을 증명했다.

참 고 문 헌

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

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

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

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

[5] 박일환, 장청룡, 원동호, "증명이 가능한 전자서명기법", 한국통신정보보호학회 논문지, 제4권/1호, pp.41-50, 1994.

[6] Lala, C.; Panda, B., "Evaluating damage from cyber attacks: a model and analysis Systems, Man and Cybernetics," Part A, IEEE Transactions on, Volume: 31 Issue: 4, Page(s): 300-310, July 2001.

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

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

[9] Younglove, R.W., "IP security: what makes it work?," Computing & Control Engineering Journal, Volume: 12 Issue: 1 , Page(s): 44-46, Feb 2001.

[10] Rubin, A.D.; Geer, D.E., Jr., "A survey of Web security," Computer, Volume: 31 Issue: 9, Page(s): 34-41, Sept. 1998.

[11] A.O. Frier, P. Karlton, and P.C. Kocher, The SSL protocol version 3.0, draft-ietf-tls-ssl-version3-00.txt, November 18, 1996.

[12] 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, Page(s): 779-792, 2001.

[13] Gutzmann, K., "Access control and session management in the HTTP environment," IEEE Internet Computing, Volume:5 Issue:1, Page(s): 26-35, Jan.-Feb. 2001.

[14] Debaty, P.; Caswell, D., "Uniform Web presence architecture for people, places, and things," IEEE Personal Communications, Volume: 8 Issue: 4, Page(s): 46-51, Aug. 2001.

[15] Anup K. Ghosh, E-Commerce Security: Weak Links, Best Defenses, John Wiley & Sons, 1998.

[16] Liu, S.; Sullivan, J.; Ormaner, J., "A practical approach to enterprise it security," IT Professional, Volume: 3 Issue: 5 , Page(s): 35-42, Sep/Oct 2001.

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

[18] 염흥렬, "전자 서명 방식 고찰," 한국통신정보보호학회 학회지, 제3권/2호, pp.7-18, 1993.

[19] R.C. Merkle and M. E. Hellman "Hiding

Information and signatures in trap-door knapsacks," IEEE Trans. On Information Theory IT-24, No.5 pp.525-530, 1978.

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

[21] Ronald L. Rivest, Adi Shamir, Len Adelman, "On Digital Signatures and Public Key Crypto systems," MIT Labatory for Computer Science Technical Memorandum 82, 1972.

[22] 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.

[23] 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.

[24] Mohammed J. Kabir, "Apache Server Bible," IDG Books Worldwide, 1998.

[25] Bob Quinn, Dave Shute, Windows Sockets Network Programming, Addison-Wesley, 1995.

장 승 주

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 관심분야는 운영체제, 분산시스템, Active Network, 시스템 보안

임 동 훈

1987년 부산대학교 계산통계학과 졸업(이학사). 1989년 부산대학교 대학원 계산통계학과 졸업(이학석사). 1993년 부산대학교 통계학과 졸업(이학박사). 1994년 ~ 현재 경상대학교 통계정보학과 부교수. 관심분야는 이미지 프로세싱, 컴퓨터 통계