# 무선 인터넷 서비스를 위한 HTTP 기반의 응용 계층 보안 프로토콜

## (An HTTP-Based Application Layer Security Protocol for Wireless Internet Services)

이 동 근 † 김 기 조 †† 임 경 식 †††
(Donggeun Lee) (Kijo Kim) (Kyungshik Lim)

**요 약** 현재 무선 인터넷에서 안전한 서비스를 제공하기 위하여 Secure HyperText Transfer Protocol(S-HTTP), Secure/Multipurpose Internet Mail Extensions(S/MIME), Secure Sockets Layer(SSL)/Transport Layer Security(TLS)와 Wireless TLS(WTLS) 등의 여러 가지 보안 프로토콜이 사용되고 있다. 그러나 S-HTTP와 S/MIME은 특정 응용에 한정적으로 사용 가능하며 SSL/TLS와 WTLS는 채널 보안으로 인하여 자원 낭비가 심할 뿐만 아니라 전자 서명 기능 또한 제공하지 못한다. 본 논문에서는 S-HTTP와 SSL/TLS의 장점을 수용하고 HTTP 기반에서 TLS 보안 메커니즘을 이용한 새로운 형태의 응용 계층 보안 프로토콜인 Application Layer Security(ALS)를 제안한다. ALS는 HTTP 기반에서 동작하므로 다양한 하부 전송망에 독립적이고, 보안을 필요로 하는 응용에 대하여 보안 인터페이스를 제공하는 방법을 통하여 특정 응용에 종속적이지 않는 특성을 가진다. 또한, TLS의 검증된 보안 메커니즘을 적용하여 안전성을 확보하였고, 인증, 기밀성, 무결성, 전자 서명 서비스 및 부분 암호화를 지원함으로써 응용에서 요구하는 다양한 서비스를 제공할 수 있다. 마지막으로 본 논문에서는 ALS를 이용한 Wireless Application Protocol의 단대단 보안 구현 내용을 기술한다.

**키워드** : IP 무선망, 멀티캐스트 핸드오프, 멀티캐스트 트리 확장

***Abstract*** In this paper, we present an application layer protocol to support secure wireless Internet services, called *Application Layer Security(ALS)*. The drawbacks of the two traditional approaches to secure wireless applications motivated the development of *ALS*. One is that in the conventional application-specific security protocol such as Secure HyperText Transfer Protocol(S-HTTP), security mechanism is included in the application itself. This gives a disadvantage that the security services are available only to that particular application. The other is that a separate protocol layer is inserted between the application and transport layers, as in the Secure Sockets Layer(SSL)/Transport Layer Security(TLS). In this case, all channel data are encrypted regardless of the specific application's requirements, resulting in much waste of network resources. To overcome these problems, ALS is proposed to be implemented on top of HTTP so that it is independent of the various transport layer protocols, and provides a common security interface with security applications so that it greatly improves the portability of security applications. In addition, since *ALS* takes advantages of well-known TLS mechanism, it eliminates the danger of malicious attack and provides applications with various security services such as authentication, confidentiality, integrity and digital signature, and partial encryption. We conclude this paper with an example of applying *ALS* to the solution of end-to-end security in a present commercial wireless protocol stack, Wireless Application Protocol.

**Key words** : IP-based wireless network, Multicast Handoff, Multicast Tree Extension

† 비 회 원 : 한국정보보호진흥원 연구원
leedg@ccmc.knu.ac.kr
†† 비 회 원 : 경북대학교 컴퓨터과학과
kijo@ccmc.knu.ac.kr
††† 종신회원 : 경북대학교 컴퓨터과학과 교수
kslim@ccmc.knu.ac.kr
논문접수 : 2002년 8월 20일
심사완료 : 2003년 2월 4일

## 1. 서 론

최근 무선 인터넷이 급성장하면서 멀티미디어 다운로드, 메일 송수신, 인터넷 뱅킹, 증권 거래에서 전자상거래까지 다양한 형태의 서비스가 제공되고 있다. 그러나, 이 서비스들 중에서 인터넷 뱅킹이나 전자상거래의 경우는 사용자의 비밀 정보가 교환되는 민감한 서비스이므로, 이러한 비밀 정보들을 안전하게 전송하기 위한 보안 기술이 필수적이다. 그러나 무선 인터넷 환경은 낮은 대역폭, 단말의 낮은 컴퓨팅 능력, 제한된 메모리와 전력 그리고 전송지연 등의 특징으로 보안 서비스를 제공하는데 많은 제약이 따른다. 그러므로, 이러한 열악한 환경에서 보안 서비스를 제공하기 위하여 다양한 형태의 보안 프로토콜이 사용되고 있다. I-mode[1]와 모바일 익스플로러(Mobile Explorer)는 무선 인터넷에서도 유선망에서 사용하는 SSL[2]을 이용하여 보안 채널을 생성한다. WAP은 초기 1.x 모델에서 연결 분리(split connection) 기법을 사용하기 때문에 유선망은 SSL, 무선망에서는 WTLS[3]를 보안 프로토콜로 적용하고 있다[4]. 2001년에 발표된 WAP 2.0 모델에서는 I-mode나 모바일 익스플로러와 같은 TLS[5] 기반으로 보안 모델을 수정하였다.

본 논문에서는 현재 보편적으로 사용되고 있는 보안 프로토콜들의 문제점을 보완하고 각각의 프로토콜의 장점을 수용하는 보안 프로토콜을 제안한다. 제안된 응용계층 보안 프로토콜인 ALS는 응용 계층에 위치하면서 보안 서비스를 필요로 하는 사용자에게 인터페이스를 제공하는 방식을 통하여 특정 응용에 한정되지 않는 특성을 가지고 HTTP를 전송 매체로 사용함으로써 다양한 전송 계층 독립적이면서 방화벽 제약을 극복하는 특성도 가진다. 또한 부분 암복호화 기능을 통해 불필요한 암복호화를 줄임으로써 무선망 자원을 절약할 수 있는 특성을 가진다.

본 논문의 2장에서는 ALS 연구의 기초가 되는 현재 사용중인 보안 프로토콜에 대하여 살펴보고, 3장에서는 구체적인 설계 내용, 동작 과정에 대하여 기술하고 4장에서는 실제 무선 인터넷 프로토콜인 WAP에 적용하여 단대단 보안을 구현한 내용에 대하여 기술한다. 마지막으로 5장에서는 결론 및 향후 과제를 제시한다.

## 2. 사례 연구

S-HTTP, S/MIME, SSL/TLS, 그리고 WAP 보안 프로토콜인 WTLS와 단대단 보안에 대하여 간략히 소개하고, 이들의 특징을 본 논문에서 제안하는 ALS와 비교 분석한다.

### 2.1 S-HTTP

S-HTTP[6]는 HTTP의 보안기능 확장 버전으로 응용 계층의 보안 프로토콜이다. S-HTTP는 Enterprise Integration Technologies에서 개발되었으며 RFC 2660에 등록되어 있다. 이 프로토콜은 메시지 기반으로써 서버와 클라이언트에게 다양한 보안 메커니즘을 제공하며, 암호화 알고리즘, 동작 모드 및 매개변수에 대한 유연성을 가지고 있다. 그리고 S-HTTP가 제공하는 보안 서비스로는 트랜잭션 기밀성, 메시지 무결성, 발신자 인증, 전자 서명 기능을 통한 발신 부인 봉쇄가 있다. 이러한 보안 서비스 제공을 위하여 서버/클라이언트간 협상 기능을 제공한다. PKCS-7, MOSS 등의 암호화 메시지 형식을 지원하며, Secure 확장 메소드를 사용한다. 그리고, 서버/클라이언트가 대칭키 방식으로 동작하는 대칭 세션 키 동작 방식을 지원한다. 또한 S-HTTP를 지원하지 않는 웹 시스템과도 호환된다. 보안 협상 정보 교환은 HTTP 협상 옵션 헤더 또는 서버상의 HTML 문서내의 S-HTTP-Crypto 태그를 사용하여 이루어지며, 키 교환은 공개키 방식인 RSA Enveloping 방식, 대칭키 교환을 위해 Inband 방식, 메시지 전송 채널과 다른 채널로 키교환을 하는 Outband 방식과 Key-Assign 헤더를 통해서 키를 교환하는 Kerberos 방식 등이 있다.

### 2.2 S/MIME

S/MIME[7]은 인터넷 전자메일의 바디부분을 구성하는 Multipurpose Internet Mail Extension(MIME)에 암호화 및 전자서명 기능을 추가시킨 보안 프로토콜이다. X.509 인증서를 사용하여 공개키 방식으로 암호화 및 전자 서명을 제공하도록 설계되어졌다. 전자 서명이 처리된 MIME 메시지를 수신한 수신자는 송신자의 X.509 인증서를 안전한 곳으로부터 획득함으로써 송신자의 공개키를 통하여 메시지 검증이 가능하게 된다. 암호화된 메시지는 송신자가 설정한 비밀키를 이용하여 생성하며 송신자는 사용한 비밀키를 수신자의 X.509 인증서로부터 획득한 공개키로 암호화하여 암호화된 메시지와 함께 전달하게 된다. 수신자는 자신의 공개키로 암호화된 비밀키를 복호화하여 메시지의 암호화에 사용된 비밀키를 획득하게된다. MIME 메시지에 대한 전자 서명 및 암호화 여부는 MIME 타입을 통해서 구분하게 된다.

### 2.3 SSL/TLS

SSL은 넷스케이프사에서 처음으로 제안되어졌으며, 넷스케이프사의 브라우저에 최초로 장착된 보안 프로토콜이다. 현재 계속적인 보완을 거쳐 버전 3.0이 발표되

어 사용되고 있다. TLS는 1996년 ITEF에서 TLS 워킹 그룹 포럼을 구성하여 1999년 발표한 보안 프로토콜로써 SSL을 국제표준으로 구성한 것이다. 현재 인터넷에서는 보안 프로토콜로써 SSL/TLS가 가장 많이 사용되고 있다. 두 보안 프로토콜은 pre_master_secret 정보를 이용해 키를 생성하는 방법에서 차이가 있을 뿐 나머지 동작 과정은 차이가 없다. 두 프로토콜 모두 핸드쉐이크 과정에서 보안 채널을 설정하기 위한 정보들을 교환하며. 전송계층 보안 프로토콜이기 때문에 상위 응용계층에 독립적인 특성을 가진다.

### 2.4 WAP WTLS

WTLS는 TLS를 무선 환경에 맞게 변형한 프로토콜이다. WTLS의 Handshake 프로토콜은 보안 세션을 설정하기 위하여 필요한 정보를 교환하는데 사용한다. 이때, 서버와 클라이언트는 프로토콜 버전과 대칭키 암호 알고리즘을 결정하고, 대칭키를 생성한다. 그리고 인증서 교환을 통하여 상호 인증을 수행한다. Record 프로토콜은 핸드쉐이크 이후 서버와 클라이언트가 합의한 보안 정보를 바탕으로 데이타를 압축하고 인증코드를 첨가한 후에 암호화하여 전송하며, 수신된 데이타는 복호화되고 인증코드 값을 검사하게 된다. Alert 프로토콜은 통신 중에 발생한 이벤트를 알려주는 기능을 한다. Change Cipher Spec 프로토콜은 전송되는 데이타에 보안이 적용되는 시점을 알려주는 기능을 한다. WTLS는 TLS를 기반으로 개발된 프로토콜이기 때문에 구조상의 큰 차이는 없지만 무선 환경에 적합하게 만들기 위하여 관련 인자들의 길이를 줄이고, 전송계층으로 UDP를 사용한다는 차이점이 있다.

### 2.5 WAP 단대단 보안

연결 분리 기법을 사용하여 유선망과 무선망을 연결하는 WAP 1.x 모델에서는 단대단 보안을 제공하는데 문제점이 있다. 게이트웨이가 유선망에서 SSL 방식으로 암호화된 데이타를 WTLS 방식으로 암호화하기 위해서 SSL 방식으로 암호화된 내용을 복호화하여 변환하기 때문에 데이타의 원본이 노출되어 버리는 것이다. 이러한 보안 문제를 해결하기 위하여 WAP 게이트웨이를 콘텐츠 제공자의 보안 영역 안으로 포함시켜 게이트웨이의 신뢰성을 높이는 방법과 WMLScript Crypto Library[8]를 확장하여 응용 계층에서 단대단 보안을 제공하는 방법이 있다. 현재 WMLScript Crypto Library는 전자서명과 관련된 signText 응용 프로그램 인터페이스(API)만이 표준화 되어있다. 최근 WAP 포럼에서 제안된 확장 API로는 보다폰, 텔스타, 셀티콤이 제안한 암호화 API인 encryptText, encrypt[9], 엔트러스트에서 제안한 복호화 API인 decrypt[10], 그리고 베리사인에서 제안한 보안 정보 설정 API인 initContext, initContextFinal, closeContext[11]가 있다[12]. WAP 2.0 모델에서는 유무선이 같은 프로토콜을 사용하기 때문에 게이트웨이에서 프로토콜 변환이 발생하지 않는다. 게이트웨이는 프로토콜의 옵션을 조정하는 역할을 한다. 따라서 프로토콜 자체적으로 단대단 보안 제공이 가능하다. 서버와 클라이언트 사이의 보안 채널은 TLS 터널링을 통하여 제공한다. 터널링 방식은 HTTP의 확장 메소드를 통하여 TLS의 보안 설정을 요청하는 HTTP Upgrade to TLS[13]를 사용한다.

### 2.6 ALS와의 비교 분석

S-HTTP는 응용 계층 보안 프로토콜로써 전자 서명 기능을 포함한 다양한 보안 서비스 제공이 가능한 반면 HTTP자체의 확장으로써 기존의 HTTP 프로토콜[14]과는 다른 프로토콜을 수용하는 응용 프로그램을 구현해야하는 문제점을 가지고 있고, S/MIME은 S-HTTP와 마찬가지로 전자 서명 등 다양한 보안 서비스를 제공하는 반면 MIME 메시지라는 전자 메일을 위하여 사용하는 메시지에 한정적으로 적용 가능하다는 문제점과 메시지 전체에 대한 보안 처리만이 가능하여 부분 암호화를 위하여 사용자가 메시지를 임의로 나누어 설정해야하는 문제점이 있다. SSL/TLS는 응용 계층에 종속적이지 않지만 전송 계층 채널 보안으로 자원 낭비가 심하고 전자 서명 서비스가 제공되지 않는 문제점이 있다. WAP 1.x 모델에서 WTLS는 무선 단말과 WAP 게이트웨이 사이에서만 적용이 가능하고 단대단 보안은 응용 계층에서 스크립트 기반으로 제공하는 방법이 있으나 표준화가 진행 중에 있으며 함수의 인터페이스가 매우 복잡하고 암복호화를 위하여 해당 트랜잭션이 사용될 때마다 키 정보를 교환하는 문제점을 가지고 있다. 이러한 문제점을 해결하고자, S-HTTP와 TLS의 장점을 수용하여 HTTP 기반에서 TLS의 보안 메커니즘을 이용한 응용 계층 보안 프로토콜인 ALS를 제안한다. 그림 1은 기존의 보안 적용 프로토콜 스택에 관한 것으로 응용 계층에 위치하는 HTTP의 보안 기능 확장 모델인 S-HTTP의 프로토콜 스택의 위치와 전송 계층 보안 프로토콜인 SSL/TLS의 프로토콜 스택의 의치를 나타낸다. 그림 2는 ALS가 적용된 프로토콜 스택을 나타낸다. 그림 1과 그림 2를 비교해 볼 때 HTTP를 전송매체로 사용하기 위하여 프로토콜 스택에서 HTTP 상위에 위치하며 따라서 하부 전송 계층에 상관없이 동작 가능하다. 또한, 독립된 프로토콜 계층을 가지면서 보안을 필요로 하는 곳에 특정 인터페이스만을 제공하

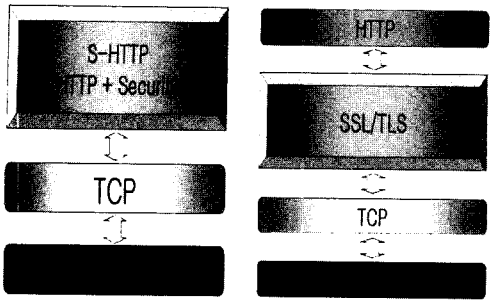


그림 1 기존 보안적용 프로토콜 스택

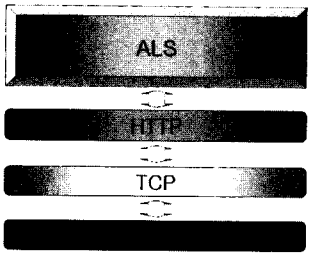


그림 2 ALS 적용 프로토콜 스택

는 방법을 통해서 응용과의 종속관계를 가지지 않는다. ALS는 TLS에 사용하는 메시지와 같은 형태의 메시지를 사용하고 동작 과정을 따르기 때문에 전송 매체를 TCP로 바꿀 경우 TLS와 호환성을 가질 수가 있다. 표 1은 전자 서명 기능 제공, 부분 암호화 기능 제공, 응용 독립 여부 그리고 TLS와의 호환성 여부에 관하여 ALS와 기존 보안 프로토콜과 비교하여 나타낸 것이다.

표 1 ALS와 기존 보안 프로토콜 비교

| 기능 \ 보안프로토콜 | S-HTTP | S/MIME | SSL/TLS | WAP WTLS | WAP 단대단 보안 | ALS |
|---|---|---|---|---|---|---|
| 전자 서명 기능 | O | O | × | × | O | O |
| 부분 암호화 기능 | O | × | × | × | O | O |
| 응용 독립 여부 | × | × | O | O | O | O |
| TLS와의 호환성 | × | × | O | × | × | O |

## 3. HTTP기반 ALS

본 단락에서는 HTTP 기반에서 TLS의 보안 메커니즘을 이용한 응용 계층 보안 프로토콜인 ALS의 구조, 세부 설계 및 동작과정 그리고 특징에 대하여 기술한다.

### 3.1 ALS 구조 및 설계

ALS는 TLS의 동작 과정을 따르며, ALS Handshake, ALS ChangeCipherSpec, ALS Alert, ALS Record 프로토콜 그리고 ALS Interface로 구성된다. 네 가지 동작 관련 프로토콜은 HTTP의 요구/응답 방식의 트랜잭션 환경에서 동작할 수 있도록 하였다. 그림 3은 ALS 프로토콜의 구조를 나타낸 것으로 TLS의 프로토콜 구조를 확장한 것이다. ALS Interface는 보안 서비스를 사용할 응용과의 통신 채널 역할을 한다. ALS Handshake 프로토콜은 서버와 클라이언트 사이에서 보안 서비스를 제공하기 위한 세션 정보 설정을 위해 사용되고 ALS ChangeCipherSpec 프로토콜은 보안 서비스가 시작되는 시점을 알려주기 위해 사용한다. ALS Record 프로토콜은 보안이 적용된 데이타를 전달할 때 사용하고 ALS Alert 프로토콜은 서버와 클라이언트 사이의 보안 서비스를 유지하는 과정에서 문제가 발생할 경우 사용한다. TLS는 인증, 기밀성과 무결성 서비스만을 지원하지만, ALS에서는 프로토콜 구조에서도 알 수 있듯이 ALS Interface를 통하여 전자 서명 서비스도 추가적으로 지원한다. HTTP를 단순히 전송 매체로 사용하고 특정 인터페이스를 통해서 보안 서비스를 제공함으로써 전송 계층에 독립적이고 특정 응용에 한정적이지 않는 독립적인 구조를 가진다.

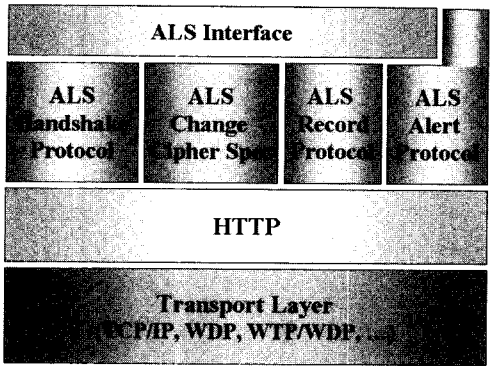


그림 3 ALS 프로토콜 구조

### 3.2 ALS 동작 과정

ALS의 동작 과정은 TLS의 동작 과정을 기반으로 하고 있다. 클라이언트와 서버는 ALS 메시지를 명시하기 위하여 HTTP 헤더의 "Content-Type"의 Media 필드 값을 새롭게 정의하여 사용한다. 그러므로, 클라이언트와 서버 보안 모듈은 메시지의 형식을 명시적으로 파악할 수 있다.

표 2는 ALS 메시지별 Media 필드 값을 나타낸 것이다. Handshake 과정 및 보안 세션 설정 이후에 서버와 클라이언트가 교환하는 메시지의 구분을 위하여 기존

표 2 ALS 메시지별 Media 필드 값

| 메시지 종류 | HTTP 확장 Content Type |
|---|---|
| ClientHello | ALS_HS_CLIENTHELLO |
| ServerHello<br>Certificate<br>ServerKeyExchange<br>CertifcateRequest<br>ServerHelloDone | ALS_HS_SERVERHELLO |
| Certificate<br>ClientKeyExchage<br>CertificateVerify<br>ChangeCipherSpec<br>Finished | ALS_HS_CLIENTFINISHED |
| ChangeCipherSpec<br>Finished | ALS_HS_SERVERFINISHED |
| 암호화 | ALS_RECORD |
| Alert | ALS_ALERT |
| 전자서명 | ALS_DSIGN |

HTTP 변형 없이 확장 Content-Type의 값을 해당 메시지별로 정한 것이다. Media 값과 메시지의 내용이 다를 경우 ALS Alert 메시지를 발생한다. 이 때 사용되는 값은 ALS_ALERT이다. 보안 세션 설정 과정에 사용되는 메시지들은 ALS_HS_로 시작하는 값을 사용한다. 암호화 메시지는 ALS_RECORD 값을 전자 서명에는 ALS_DSIGN을 사용한다. 메시지 전달 방식은 HTTP POST 메소드 요구/응답을 사용한다. 표 3은 ALS에서 사용하는 메시지의 종류와 설명을 나타낸 것이다. 기반 동작 모델이 TLS를 따르기 때문에 사용하는 메시지가 포함하는 내용 및 기능은 TLS의 메시지와 같다.

그림 4는 ALS Handshake 동작 과정을 나타낸 것으로 구체적인 내용은 다음과 같다. 클라이언트는 HTTP를 이용하여 자신이 생성한 난수 값과 세션 ID 값, 보안 세션 정보를 설정하기 위한 Cipher Suite 리스트를

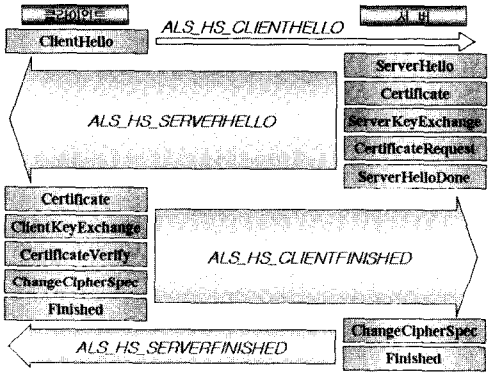


그림 4 ALS Handshake 동작 과정

표 3 ALS 메시지 설명

| 메시지 종류 | 메시지 설명 |
|---|---|
| Client Hello | 클라이언트가 서버로 전달하는 메시지이며 클라이언트가 생성한 난수 값, 세션 ID, 압축 방법 그리고 Cipher suite 리스트를 포함하고 있다. Cipher suite는 키 교환 알고리즘, 암호 알고리즘, 해쉬 알고리즘을 명시한다. |
| ServerHello | 서버가 클라이언트로 전달하는 메시지이며 서버가 생성한 난수 값, 세션 ID 그리고 클라이언트가 보낸 압축 방법 및 Cipher suite 리스트에서 선택된 값을 포함하고 있다. |
| Certificate | 클라이언트 또는 서버가 사용할 수 있는 메시지이며 각자의 X.509 공개키 인증서를 포함하고 있다. |
| ServerKeyExchange | 서버가 클라이언트로 전달하는 메시지이며 서버가 생성한 키 정보를 포함하고 있다. 키 정보는 클라이언트가 서버로 전달하는 키 생성 값을 암호화하는데 사용한다. |
| CertifcateRequest | 서버가 클라이언트의 인증서를 요청할 경우 사용되는 메시지이다. |
| ServerHelloDone | 서버가 클라이언트로 모든 메시지를 전달했음을 나타내는 메시지이다. |
| ClientKeyExchage | 클라이언트가 서버로 전달하는 메시지이며 세션 키를 생성하는데 이용되는 값을 공개키 방식으로 암호화하여 포함하고 있다. |
| CertificateVerify | 클라이언트가 서버로 전달하는 메시지이며 서버가 전달받은 클라이언트의 인증서를 인증하는데 사용하는 메시지이다. |
| ChangeCipherSpec | 클라이언트와 서버 모두가 사용할 수 있는 메시지이며 보안 세션이 사용되는 시점을 나타낸다. |
| Finished | 클라이언트와 서버 모두가 사용할 수 있는 메시지이며 Handshake 과정에서 사용된 모든 메시지에 대한 점검을 할 수 있게 하며 보안세션 설정 과정을 마무리하는데 사용한다. |
| 암호화 | 보안 세션 설정 이후 사용되는 메시지이며 암호화된 데이타를 포함하고 있다. |
| Alert | 보안 세션 설정 과정 또는 이후에 서버와 클라이언트사이에 문제가 발생할 경우 이를 상호 인지하기 위하여 사용하는 메시지이다. |
| 전자서명 | 보안 세션 설정 이후 사용되는 메시지이며 전자 서명된 데이타를 포함하고 있다. |

포함한 ClientHello 메시지를 서버로 전달한다. 서버는 생성한 난수 값과 세션 ID, 그리고 클라이언트가 보낸 Cipher Suite 리스트에서 서버가 선택한 값을 포함한ServerHello 메시지를 클라이언트로 전달한다. 서버의 Certificate 메시지는 서버의 X.509 인증서를 전달하고, ServerKeyExchange 메시지는 대칭키 교환을 위한 인자 값들을 전달한다. CertificateRequest 메시지는 서버가 클라이언트의 인증서를 요구할 경우 전달하는 메시지이며, ServerHelloDone 메시지는 서버가 더 이상 전달할 메시지가 없음을 의미한다. 이때 서버에서 클라이언트로 전달하는 다섯 가지 메시지들은 하나의 데이타 세그먼트 형태로 클라이언트로 전송된다. 클라이언트는 서버로부터 인증서 요청이 있을 경우 ClientCertificate 메시지를 이용해서 자신의 인증서를 전달하고, 대칭 키 생성 인자인 pre_master_secret 값을 포함하는 ClientKeyExchange 메시지를 공개키 알고리즘으로 서버로 전달한다. CertificateVerify 메시지로 서버가 클라이언트의 인증서를 확인하는데 필요한 정보를 전달하고, ChangeCipherSpec 메시지로 암호화가 시작됨을 서버에게 알린다. 마지막으로 Finished 메시지를 통해서 협상된 알고리즘이 적용된 최초의 암호화 전송이 이루어진다. 이때에도 역시 클라이언트는 다섯 가지 메시지를 하나의 세그먼트 형태로 서버로 전달한다. 서버는 클라이언트로부터 수신한 모든 메시지를 확인하고 ChangeCipherSpec 메시지와 Finished 메시지를 하나의 세그먼트 형태로 클라이언트에게 전달하고 전체 과정을 끝낸다. 이후에 전송되는 데이타는 ALS Interface를 통해서 ALS Record 프로토콜로 전달되어 암복호화 된다. 그림 5는 ALS Record 메시지가 생성 및 처리되는 과정을 나타내는 것이다. 그림에서 사용자 데이타는 보안을 적용시키고자 하는 부분이 포함된 전체 데이타를 의미하며 XML, HTML, WML 등의 마크업 언어가 될 수 있다. 이러한 데이타에 대하여 ALS는 암복호화 서비스를 제공한다. 암호화 처리 과정은 다음과 같다. 사용자가 암복호화 모듈로 보안 적용 부분 데이타를 전달하면 이를 처리하여 암호화된 데이타를 반환하고 원본 사용자 데이타 내의 평문 데이타와 치환하게 된다. 그리고 최종적으로 전달되는 메시지는 Message Authentication Code(MAC) 생성 검증 모듈을 통하여 생성한 값을 첨가하고 HTTP 헤더를 첨부하여 완성된다. 복호화 과정은 먼저 전체 메시지에서 MAC 값을 분리하여 MAC 생성 검증 모듈로 전달하여 무결성을 검사하고 확인된 메시지에서 보안 적용된 부분의 데이타를 암복호화 모듈로 전달하여 사용자 데이타의 암호화된 부분과 치환하여 원본 데이타를 복원한다. 사용자가 단지 각 모듈의 ALS Interface를 호출하는 것만으로 보안 서비스를 제공받을 수 있는 것은 ALS가 특정 응용에 한정되지 않는 독립적인 특성을 가지고 있기 때문이다. 그림 6은 ALS Alert 메시지의 흐름이다. 서버와 클라이언트가 Handshake 과정이거나 보안 세션 설정이후 보안 서비스 제공이 이루어지고 있는 가운데 문제가 발생했을 경우 메시지가 상황에 따라서 어떻게 전달되는지 나타낸 것이다. 먼저 서버 측에서 문제 발생을 인지한 경우는 서버에서 메시지를 생성하여 준비하고 클라이언트의 Request 메시지가 전송되어 오면 응답으로 준비된 메시지를 전달하여 상호 문제 발생을 인지하고 보안 서비스를 중단하게 된다. 클라이언트 측에서 문제 발생을 먼저 인지한 경우는 서버 측으로 POST 방식으로 메시지를 보내면서 상호 문제 발생을 인지하게 된다. 이 때 메시지가 명확히 전달되어 서버와 클라이언트가 모두 문제 발생을 인지하면 양쪽 모두 보안 서비스 중단을 위한 처리를 수행하게 된다[15].

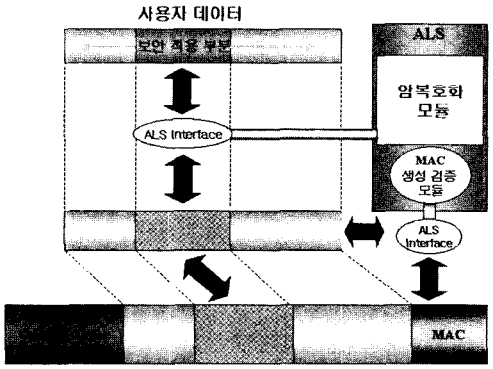


그림 5 ALS Record 메시지 생성 및 처리 과정

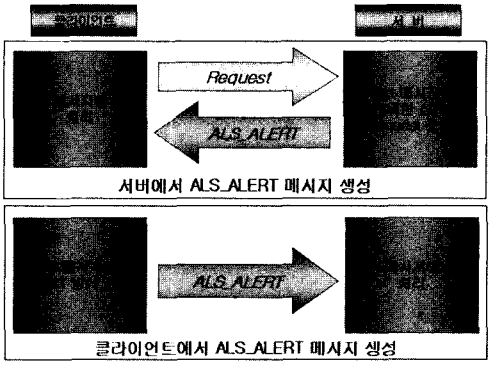


그림 6 ALS Alert 메시지 흐름

### 3.3 ALS 제공 보안 서비스

ALS에서 인증 서비스는 전자 서명 또는 Handshake 과정에서 이루어진다. 먼저, 클라이언트를 인증할 때에는 서버가 CertificateRequest 메시지를 사용하여 클라이언트에게 인증서를 요청하고, 클라이언트로부터 전송된 Certificate 메시지를 바탕으로 클라이언트 인증을 수행한다. 클라이언트는 ServerHello 메시지와 함께 전달된 Certificate 메시지로 서버 인증서를 수신하고 검증 과정을 거쳐 서버를 인증한다. 기밀성 서비스는 보안 정보 협상 과정에서 교환된 세션 정보를 사용하여 ALS Record 프로토콜이 암복호화를 수행함으로써 제공된다. 무결성 서비스는 전송되는 메시지의 뒷부분에 128 비트 MAC을 포함하여 제공된다. 이때 HTTP 헤더의 컨텐츠 길이 값을 변경하여 전달한다[16]. 수신측은 메시지의 마지막 128 비트를 MAC으로 인식하고, 나머지 부분에 대하여 무결성을 검사한다. 그림 7은 전자 서명 메시지의 생성 및 처리에 관한 것으로 서명이 포함된 데이타를 생성하는 과정과 서명된 데이타에 대한 검증을 처리하는 과정을 나타낸다. 그림에서 사용자 데이타는 전자 서명의 원본 데이타이다. 전자 서명 처리된 메시지를 생성하는 과정은 다음과 같다. 사용자는 ALS의 전자 서명 생성 검증 모듈로 원본 데이타를 전달하고 이에 대한 서명 처리 결과를 받게 된다. 이때 원본 데이타와 서명 데이타를 포함한 메시지가 완성된다. 다음으로 MAC 생성 검증 모듈을 통하여 해당 메시지에 대한 값을 생성하여 서명이 포함된 데이타와 결합하게 되고 최종적으로 전자 서명 처리된 메시지를 생성하게 된다. 전자 서명의 검증 과정은 다음과 같다. 전체 메시지의 마지막 128 비트 값을 분리하여 MAC 생성 검증 모듈로 전달하여 무결성을 검사하고 확인된 데이타를 ALS의 전자 서명 생성 검증 모듈로 전달하여 서명 검증을 하게된다. 전자 서명 서비스도 암복호화 서비스에서처럼 사용자가 단지 각 모듈의 ALS Interface를 호출하는 것만으로 서비스를 직접 사용 할 수 있게 된다.

## 4. WAP 단대단 보안을 위한 ALS 구현 사례

본 단락에서는 WAP 1.x[17] 모델에서 단대단 보안을 제공하기 위하여 ALS 방식을 적용한 구현 사례를 소개한다. 이 모델은 WAP 게이트웨이를 중심으로 유선망과 무선망에 서로 다른 프로토콜을 사용하고 있기 때문에 보안 프로토콜에 있어서도 유선망은 SSL/TLS, 무선망은 WTLS를 사용하고 있어 단대단 보안을 제공하지 못하는 문제점을 가지고 있다. 그러나, 상위 응용 계층은 HTTP를 공통적으로 지원하고 있기 때문에 ALS를 적용하여 보안 모델을 구성할 수 있다. 구현 모델은 그림 8과 같다. 클라이언트는 기본적인 WMLScript 라이브러리와 확장된 Crypto 라이브러리[18]를 추가적으로 지원하는 인터프리터를 사용한다. 기본 라이브러리는 WAP 1.x에서 지원하는 것이며 확장된 Crypto 라이브러리는 내부적으로 ALS Interface를 사용하여 구현된 handshake, signtext, encrypt, decrypt API를 포함한다. 사용자에게 편리한 보안 서비스를 제공하기 위하여 가능하다면 복잡성을 줄이는 것이 중요하기 때문에[19] 사용자에게 익숙한 인터페이스 형태로 제공되도록 하였다. 표 4는 서버와 클라이언트에서 사용하는 인터페이스에 대한 설명이다. 서버 측의 API들은 윈도우즈 운영체제의 인터넷 서비스인 IIS를 기반으로 동작하는 Crypto ASP 컴포넌트[20]내에 구현되어 있다. handshake, verifysigntext, encrypt, decrypt API 들은 모두 ALS Interface의 기능을 포함한다. handshake는 ALS Handshake 프로토콜을 사용하여 동작하며, 보안 세션이 설정된 이 후는 encrypt와 decrypt를 통하여 암복호

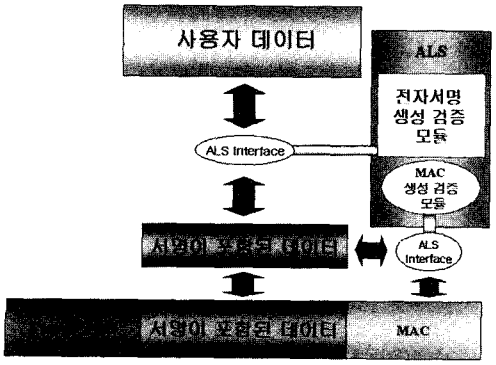


그림 7 전자 서명 생성 및 처리 과정

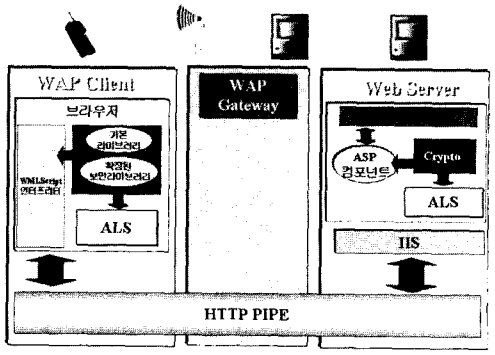


그림 8 WAP 1.x 모델 단대단 보완 구조

표 4 ALS Interface

| 구 분 | Name | Description |
|---|---|---|
| 서 버 | HRESULT<br>handshake() | 클라이언트와 웹 서버 사이에서 보안 정보를 교환하기 위한 협상을 수행한다. |
| | HRESULT<br>encrypt(BSTR plaintext, BSTR *rciphertext) | 암호화를 수행한다. |
| | HRESULT<br>decrypt(BSTR ciphertext, BSTR *rplaintext) | 복호화를 수행한다. |
| | HRESULT<br>verifysigntext(BSTR signin, BSTR indata,<br>BSTR url) | 전자 서명 검증을 수행한다. |
| 클라이언트 | Bool<br>handshake(String handshakeURL,<br>String nextpageURL) | 클라이언트와 웹 서버 사이에서 보안 정보를 교환하기 위한 협상을 수행한다. |
| | String<br>encrypt(string ToEncrypt) | 암호화를 수행한다. |
| | String<br>decrypt(string ToDecrypt) | 복호화를 수행한다. |
| | String<br>signtext(string ToSign, int options, int KeyidType,<br>String Keyid) | 전자 서명을 수행한다. |

화 서비스를 제공한다. plaintext와 rplaintext 인자는 원본 평문 데이타를 나타내고 rciphertext와 ciphertext 인자는 보안 적용된 데이타를 나타낸다. API 내부적으로 ALS Record 프로토콜로 암호화와 복호화를 지원한다. 서버의 verifysigntext는 클라이언트가 signText를 사용하여 서명한 내용을 검증하는 역할을 한다. 클라이언트는 WMLScript API 형태로 구현되어 있다. handshake는 서버에 ALS Handshake 프로토콜 메시지를 전달하여 보안 세션 설정 과정이 시작되도록 하는 역할을 하며 수행 후 보안 세션이 적용된 WML 페이지로 이동하도록 nextpageURL 인자가 설정되어 있다. encrypt와 decrypt는 서버의 API와 기능이 같으며 singtext는 원본 데이타를 입력받아서 서명 데이타를 생성하는 역할을 한다. 다음은 클라이언트가 암호화된 데이타를 서버로 전달하는 WML[21] 페이지 예제이다. WML 페이지에서 사용자가 특정 데이타를 입력할 경우 accountbalance.wmls 라는 스크립트 파일에 구현된 afnc()를 호출하게 되고 함수 내에 포함된 ALS Interface를 호출하여 암호화 데이타를 생성하여 POST 방식으로 서버로 전달하게 된다.

다음은 Crypto API를 호출하여 암호화를 수행하는 스크립트 코드가 포함된 WML 스크립트[22] 파일 예제이다. WMLScript Crypto 라이브러리에 정의된 encrypt()를 사용한다 k3 =Crypto.encrypt(k1)에서 입력 인자는 WML 페이지에서 사용자가 입력한 평문 문자열 데이타이며 반환되는 값은 암호화 처리된 문자열 값이다.

```
<wml>
<card id="main" title="E2E Secure Internet Banking">
<p>
  Account Num :[ ]
  <input name="accnum" />
  <do type="accept" label="go">
      <go href="#card2">
      </go>
  </do>
</p>
</card>
<card id="card2" title="E2E Secure Internet Banking">
<p>
Account Num : [ $(accnum) ]
<br/>
Passwd :[ ]
   <input name="passwd" />
   <do type="accept" label="go">
      <go href="http://(사이트주소)/e2esbank/Security1/
            account/accountbalance.wmls#afnc()">
      </go>
  </do>
</p>
</card>
<card id="result" title="E2E Secure Internet Banking">
<p>
  <do type="accept" label="go">
    <go href="http://(사이트주소)/E2ESBank/Security1/
        account/result_ab.asp" method="post"
      enctype="ALS_RECORD">
      <postfield name="accnum" value="$eaccnum"/>
      <postfield name="passwd" value="$epasswd"/>
    </go>
  </do>
</p>
</card>
</wml>
```

```
extern function afnc()
{
var k1;
var k2;
var k3;
var k4;

k1 = WMLBrowser.getVar("accnum");
//글로벌 변수 accnum에 있는 값을 가져온다.
k2 = WMLBrowser.getVar("passwd");
//글로벌 변수 passwd에 있는 값을 가져온다.

k3 = Crypto.encrypt(k1);
k4 = Crypto.encrypt(k2);

WMLBrowser.setVar("eaccnum",k3);
//글로벌 변수 eaccnum에 값을 설정한다.
WMLBrowser.setVar("epasswd",k4);
//글로벌 변수 epasswd에 값을 설정한다.

WMLBrowser.go("#result");
};
```

## 5. 결론 및 향후 과제

본 논문은 무선 인터넷 서비스를 위한 보안 서비스 제공을 위하여 응용 종속적인 S-HTTP와 채널 보안으로 자원 낭비가 심할 뿐만 아니라 전자 서명 기능 또한 제공되지 않는 TLS의 문제점을 해결하고 응용 독립적이면서 다양한 보안 서비스를 효과적으로 제공하는 보안 프로토콜을 설계하였다. ALS는 전송 계층 보안 프로토콜로 안전성이 인정된 TLS의 동작 메커니즘을 사용하고 무선 인터넷 서비스에 보편적으로 사용되고 있는 HTTP를 전송매체로 사용하여 동작하도록 하였다. 그리고 응용 계층 보안 프로토콜이면서 응용에 종속적이 않고 보안 서비스를 제공할 수 있도록 인터페이스 제공 형태의 모델 구조이며, TLS에서 프로토콜 자체적으로 지원할 수 없었던 전자 서명 보안 서비스를 제공한다. 또한, 전체 채널내의 모든 데이타에 대하여 보안을 적용하는 것이 아니라 보안을 필요로 하는 특정 부분의 데이타에 대해서만 부분 암호화를 제공할 수 있다. 이러한 선택적 암복호화는 ALS Interface를 통해서 이루어지며 무선 인터넷 망 자원을 효율적으로 사용할 수 있게 해준다.

현재 무선 인터넷은 분산 모바일 환경으로 변하고 있다. 모바일 단말에 Java 2 Micro Edition(J2ME)[23]이 장착되면서 모바일 코드가 보편화되고 있으며 분산환경의 전송매체로써 SOAP[24]이 주목을 받고 있는 상황에서 SOAP과 마찬가지로 HTTP를 사용하는 ALS 프로토콜은 최적의 보안 서비스를 제공할 것으로 기대되어지며 향후 컴포넌트화하여 모바일 코드형태로써 사용이 가능하도록 연구가 필요할 것이다.

## 참 고 문 헌


[1] i-mode, "DoCoMo i-mode", NTT, November 1999.

[2] Alan O. Freier, Philip Karlton, Paul C. Kocher, "The SSL Protocol version 3.0, Internet-Draft," 1996, http://home.netscape.com/eng/ssl3/.

[3] WTLS, "Wireless Transport Layer Security Protocol Specification," WAP Forum, November 8, 1999, http://www.wapforum.org/.

[4] 원유재, "무선 응용 프로토콜 보안 기술", 한국정보과학회 정보통신연구회 정보통신기술지, 14권, 1호, pp. 34-35, 2000년 5월.

[5] T. Dierks, C. Allen, "The TLS Protocol," January 1999, http://www.ietf.org/rfc/rfc2246.txt.

[6] E. Rescorla, A.Schiffman, "The Secure HyperText Transfer Protocol," August 1999, http://www.ietf.org/rfc/rfc2660.txt.

[7] B. Ramsdell, "S/MIME Version 3 Message Specification," June 1999, http://www.ietf.org/rfc/rfc2633.txt.

[8] WMLScript Crypto, "WMLScript Crypto API Library," WAP Forum, November 1999, http://www.wapforum.org/.

[9] Vodafone, Telstar, Certicom, "Change Request WMLScript Crypto API," WAP Forum, June 2001, http://www.wapforum.org/.

[10] Entrust, "Change Request WMLScript Crypto Specification," June 2001, http://www.wapforum.org/.

[11] VeriSign, "Change Request WMLScript Crypto Library Specification," August 2001, http://www.wapforum.org/.

[12] 이동근, 김기조, 임경식, 이석준, 정병호, "무선 응용 프로토콜 보안기술", 한국정보과학회 정보과학회학회지, 제20권, 제4호, pp. 58-65, 2002년 4월.

[13] S. Lawrence, "Upgrading to TLS Within HTTP/1.1," May 2000, http://www.ietf.org/rfc/rfc2817.txt.

[14] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "HyperText Transfer Protocol-HTTP 1.1," June 1999, http://www.ietf.org/rfc/rfc2616.txt.

[15] Stephen A. Thomas, "SSL & TLS Essentials Securing the Web," ISBN:0-471-38354-6, Wiley Computer Publishing, 2000.

[16] J. Franks, S. Lawrence, "HTTP Authentication: Basic and Digest Access Authentication," http://www.ietf.org/rfc/rfc2617.txt.

[17] WAP, "Wireless Application Protocol Architecture,"

WAP Forum, April 1998, http://www.wapforum.org/.
[18] WMLScript Crypto, "WMLScript Crypto API Library," WAP Forum, November 1999, http://www.wapforum.org/.
[19] Miguel Soriano, Diego Ponce, "A Security and Usability Proposal for Mobile Electronic Commerce," IEEE Communication Magazine, August 2002.
[20] COM, "The Component Object Model Specification," Microsoft Corporation, April 1999, http://www.microsoft.com/com/resources/comdocs.asp.
[21] WML, "Wireless Markup Language," WAP Forum, November 8, 1999, http://www.wapforum.org/.
[22] WMLScript, "Wireless Markup Language Script," WAP Forum, November 8, 1999, http://www.wapforum.org/.
[23] M. Hardee, "Why Wireless Needs JavaTM Technology," 2000 JavaOne SM Dev. Conf., July 2000.
[24] Jepsen, T., "SOAP Cleans up Interoperability Problems on the Web," IT Professional, Volume: 3, Issue: 1, January-February 2001.

이 동 근

2001년 경북대학교 컴퓨터과학과(이학사) 2003년 경북대학교 컴퓨터과학과(이학석사). 2003년~현재 한국정보보호진흥원(KISA) 연구원. 관심분야는 무선 인터넷, 네트워크 보안, 컴퓨터통신

김 기 조

1999년 경북대학교 컴퓨터과학과(이학사) 2002년 경북대학교 컴퓨터과학과(이학석사). 2002년~현재 경북대학교 컴퓨터과학과 박사과정. 관심분야는 컴퓨터 통신, 이동 컴퓨팅, 네트워크 보안, 액티브 네트워크

임 경 식

1982년 경북대학교 전자공학과(공학사) 1985년 한국과학기술원 전산학과(공학석사). 1994년 University of Florida 전산학과(공학박사). 1985년~1998년 한국전자통신연구원 책임연구원, 실장. 1998년~현재 경북대학교 컴퓨터과학과 조교수. 관심분야는 이동 컴퓨팅, 무선 인터넷, 홈 네트워킹, 컴퓨터 통신