SSL/TLS와 WTLS의 프로토콜 취약성 분석†

김소진* · 신성한* · 박지환**

1. 서 론

'디지털 혁명'이라는 단어로 요약되는 정보통신기술의 발전은 인류의 삶의 방식에 지대한 변혁을 초래하고 있다. 또한 개인용 컴퓨터와 인터넷의 폭발적 보급과 발전은 삶에 혁신적 변화를 가져오고 있으며, 다른 연결 수단인 무선통신 호출기를 시작으로 곧 상용화될 IMT-2000기반의 무선통신까지 최근 몇 년 사이에 무선통신 환경도 엄청난 속도로 발전하고 있다. 무선 단말기의 발전과 함께 단말기를 통해 가능한 부가 서비스에 대한 모색도 활발히 전개되고 있다.

이동통신의 발전과 더불어 현재 가장 주목받으면서 활발하게 제공되고 있는 서비스는 전자상거래 서비스이며, 이동통신 환경에서도 금융, 증권, 경매 등과 같은 전자상거래 관련 서비스가 주요 서비스 아이템으로 자리잡아갈 것이다. 그러나 무선 인터넷에서 전자상거래를 비롯한 각종 서비스가 성공적으로 제공되기 위해서는 반드시 보안 문제가 해결되어야 한다.

보안기술은 기존의 인터넷에서도 가장 중요한 요소기술 가운데 하나로 취급되고 있다. 특히 음성 위주의 이동통신에서는 도청만이 문제가 되었지만, 단순한 정보 서비스를 뛰어넘어 증권이나 뱅킹과 같은 상거래 활동으로 이어지는 서비스에서는 사용자 인증, 데이터 무결성 보장 등 해결해야 할 문제가 많다. 이에 따라 WAP(Wireless Application Protocol), ME(Mobile Explorer), i- mode 등과 같은 여러 가지 다양한 무선 인터넷 솔루션이 개발되고 있다[1].

본 논문에서는 이러한 여러 솔루션 중에서도 무선 인터넷 업계의 표준으로 자리잡아가고 있는 WAP에 대해 고찰하고자 한다[2]. 특히 WAP의 보안 프로토콜인 WTLS(Wireless Transport Layer Security)를 유선 인터넷 보안을 위한 SSL(Secure Socket Layer)/TLS(Transport Layer Security)와 비교하여 분석하고, 이에 따른 유/무선상의 차이점을 살펴본다. 그리고 각각의 취약성을 검토한 후, SSL/TLS에서 상속된 WTLS의 취약성과 WTLS 자체의 취약성으로 구분하여 분석한다[3-7].

본 논문의 구성은 다음과 같다. 2장에서는 WAP의 목적 및 구조에 대해서 간략하게 설명하고, 3장에서는 유선상의 보안 프로토콜인 SSL/TLS와 무선상의 WTLS의 구조 및 동작과정에 대해서 다룬다. WTLS의 취약성에 대해서 4장에서 상세하게 분석하고 이에 대응하는 해결방안을 제시한다.

† 본연구는 정통부 이동네트워크 정보기술 연구센터의 지원에 의해 수행되었음
* 부경대학교 대학원 전자계산학과
** 부경대학교 전자컴퓨터정보통신공학부

2. WAP(Wireless Application Protocol)

WAP은 1997년 6월에 Phone.com이 주축이 되어 Ericsson, Motorola, Nokia, Unwired Planet 4개사가 공통 규격의 제정을 위해 만든 표준화 단체인 WAP 포럼에서 제정한 무선망과 인터넷 연동을 위한 프로토콜이다. 현재 WAP 포럼에는 전세계 300여개가 넘는 업체가 참여하고 있으며 국내에서는 LG정보통신, 삼성전자, SK텔레콤 등이 참여하고 있다. WAP 방식은 전세계적으로 사용자 면에서 가장 많은 수를 차지하고 있고, 세계적인 표준으로 자리잡기에 가장 유망한 프로토콜이다[1,2].

2.1 목적

WAP은 휴대폰, 호출기, PDA 등의 무선 단말기를 위한 응용 구조와 프로토콜을 정의한다. GSM(Global Standard for Mobiles), TDMA(Time Division Multiple Access), CDMA(Code Division Multiple Access) 등의 서로 다른 망에서 쓰일 수 있는 프로토콜을 정의하고 개발자들이 빠르고 유연하게 더 나은 서비스와 응용 기술을 개발할 수 있도록 한다. WAP포럼에서 추구하는 바는 다음과 같다.

- 계층적이고 확장 가능한 구조를 정의한다.
- 가능한 한 많은 무선 네트워크를 지원한다.
- 좁은 대역의 bearer를 위해서 최적화한다.
- 단말기 자원의 효율적 사용을 고려한다.
- 안전한 응용과 통신을 제공한다. 최대한 유연하게 MMI를 생성할 수 있도록 한다.
- 임시적인 요소를 정의함으로써 많은 개발자들에게 호환성을 제공한다.
- 서비스와의 통합을 위하여 프로그래밍 모델을 제공한다.

2.2 WAP 모델

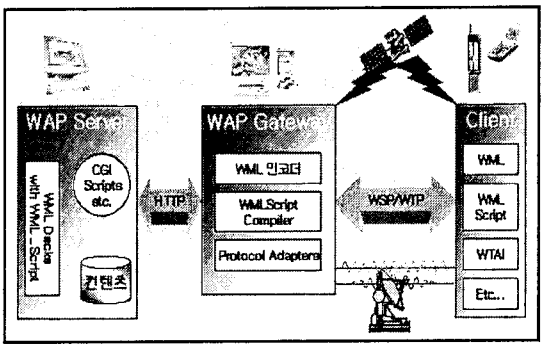


그림 1. WAP 모델

위의 그림과 같이 WAP은 무선망과 WWW(World Wide Web)의 연동을 위하여 Wap Gateway라는 Proxy기능을 사용한다. WAP Gateway의 주요 역할은 WAP 프로토콜과 인터넷 TCP/IP 프로토콜을 중간에서 변환해 주는 것이다. 즉, 모든 휴대 단말기의 인터넷 서비스 요구는 WAP Gateway를 거치도록 되어 있고, Gateway는 WAP 프로토콜에 따라 요청받은 서비스를 기존 인터넷 유선망을 통해 다시 요청한다. 이어서 Gateway가 인터넷 서버로부터 응답을 받고 다시 서비스를 최초 요청했던 휴대 단말기에게 WAP 프로토콜로 전송함으로써 모든 과정이 이루어진다.

이러한 인프라 구조는 사용자가 무선 단말기를 이용하여 WAP 컨텐츠와 어플리케이션을 이용할 수 있도록 하며, 어플리케이션 제작자들이 광대한 무선 통신망에서 사용될 서비스와 어플리케이션을 제작할 수 있도록 한다. WAP Proxy는 컨텐츠와 어플리케이션이 표준 WWW서버 위에서 호스팅되게 하며 CGI 프로그래밍과 같은 검증된 WWW의 기술을 바탕으로 발전하고 있다.

2.3 WAP의 구조

WAP 구조는 빠르고 유연하게 어플리케이션

을 개발할 수 있는 확장된 환경을 제공한다. 이는 전체 프로토콜의 스택이 레이어 기반 구조이기 때문이다. WAP 구조의 레이어는 다른 서비스나 어플리케이션 그리고 상위 레이어와 접속 가능하다. WAP의 레이어에 기반한 구조는 다른 서비스와 어플리케이션이 일련의 잘 정의된 인터페이스를 통해서 WAP 스택의 특징을 잘 이용할 수 있도록 한다. 외부의 어플리케이션은 Session, Transaction, Security, Transport Layer를 직접 연결할 수 있다. WAP 스택은 그림2와 같이 6개의 레이어로 구성되어 있으며 각각은 다음과 같은 역할을 한다.

- WAE(Wireless Application Environment)
  - 일반적, 다목적 응용을 개발하기 위한 응용 환경에 대한 규격을 정의한다.
  - WML(Wireless Markup Language), WML Script 등을 정의하여 휴대 단말기에 적용될 수 있는 소규모 브라우저를 개발할 수 있도록 한다.
- WSP(Wireless Session Protocol)
  - HTTP/1.1에 상응하는 기능을 정의한다.
  - 장기간 활용할 수 있는 세션을 정의하고, 세션 관리를 위해 suspend/resume 기능도 제공한다.
  - 프로토콜 기능에 대한 협상도 가능하게 한다.

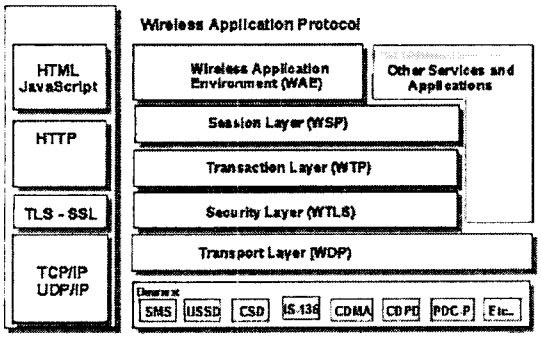


그림 2. WAP의 스택

- WTP(Wireless Transaction Protocol)
  - 트랜잭션 형태의 데이터 전송 기능을 제공
  - 신뢰성 및 비신뢰성 전송 기능을 제공하고 복구를 위해 재전송 기능도 담당한다.
- WTLS(Wireless Transport Layer Security)
  - 인터넷의 TLS를 근간으로 작성된 보안 프로토콜로 인증(Authentication), 부인봉쇄(Non-Repudiation), 무결성, 기밀성(Security) 등의 보안 서비스를 제공한다.
- WDP(Wireless Datagram Protocol)
  - End-To-End 전송을 위해 port 어드레싱을 제공한다.
  - 인터넷의 UDP와 같은 전송 기능을 담당한다.

3. 보안 프로토콜

지금까지 개발된 보안 프로토콜에는 여러 종류가 있으나, 트랜스포트층의 보안 프로토콜은 SSL, TLS, WTLS 등이다. 이러한 프로토콜은 어플리케이션간의 상호호환성을 확보하고 새로운 알고리즘 추가가 용이하도록 하는 확장성을 제공하며 효율성을 목적으로 한다. 그리고 주요 목적은 기밀성, 무결성, 사용자 인증의 보안 서비스(4가지 요소 중 부인방지는 어플리케이션에서 추가)의 제공이다.

3.1 SSL(Secure Socket Layer)

3.1.1 구조

유선 환경에서는 종점간의 안전한 데이터 전송을 위해 현재 SSL과 TLS를 사용하고 있고, 많은 인터넷 쇼핑몰에서는 SSL을 채택하고 있다.SSL은 트랜스포트층과 응용계층 사이에 위치하며, 크게 상위계층인 Control Protocol과 하위계층 인 Record Protocol로 나뉜다. Control Protocol은 그

림3과 같이 Handshake Protocol, Change Cipher Spec, Alert Protocol로 구분된다[6].

3.1.2 SSL 동작

SSL 보안 프로토콜의 동작은 다음과 같으며 그림4와 같이 요약된다[6].

① 클라이언트는 서버에게 클라이언트의 SSL 버전 번호, 암호 설정, 랜덤 생성 데이터 및 서버가 SSL을 사용하여 클라이언트와 통신할 때 필요한 기타정보 등을 전송한다.(ClientHello)

② 서버는 클라이언트에게 서버의 SSL 버전 번호, 암호 설정, 랜덤 생성 데이터 및 클라이언드가 서버와 SSL을 사용하여 통신하는데 필요한 기타 정보 등을 전송한다. 서버는 또한 자신의 인증서를 전송하고 클라이언트가 서버의 자원을 요구한다면 클라이언트의 인증서에 대한 요구를 전송한다.(ServerHello)

③ 클라이언트는 서버를 인증하기 위해 서버가 보낸 정보를 사용하고 서버가 인증될 수 없으면 암호화 및 인증된 안전한 연결이 수립될 수 없도록 한다. (Server Certificate, Sever Key Exchange) 서버가 인증되었으면 다음과정을 수행한다.

④ Handshake에서 생성된 모든 데이터를 사

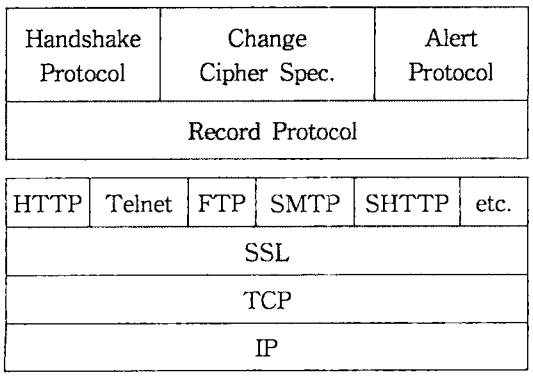
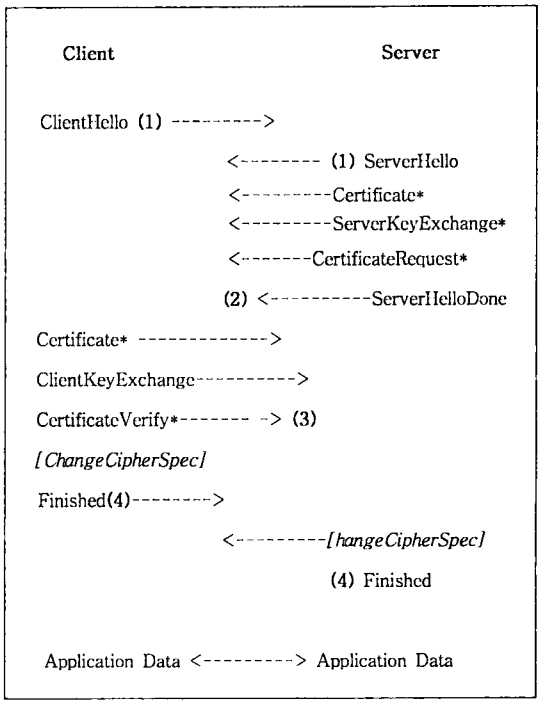


그림 3. SSL 구조



(1) 프로토콜 버전, 세션ID, cipher suite, 압축 method 등의 보안 파라미터 협상과 random값 교환
(2) 선택적(*:Option)으로 서버의 인증서를 전송하고 클라이언트의 인증서 요구
(3) 서버가 인증서를 요청하면 클라이언트의 인증서를 응답으로 전송
(4) Change cipher suite과 finish 메시지 교환으로 Hand shake 과정을 마침

그림 4. SSL의 Full Handshake 동작

용하여 클라이언트는 세션을 위한 premaster secret을 생성하고 이를 서버의 공개키를 사용하여 암호화한 후 서버에게 전송한다.

⑤ 서버가 클라이언트의 인증을 요구하면 클라이언트는 handshake 세션에서 유일하게 알려진 데이터를 서명한다. 클라이언트는 서명한 데이터와 클라이언트의 인증서를 암호화된 premaster secret과 함께 전송한다. (ClientCertificate, Client Key Exchange, Certificate Verify)

⑥ 서버가 클라이언트의 인증서를 요청했다면

서버는 클라이언트의 인증을 시도한다. 클라이언트가 인증되지 않으면 세션은 종료된다. 클라이언트가 인증되었다면 서버는 비밀키를 사용하여 premaster secret을 복호화하고 master secret의 생성을 위한 과정을 수행한다.

⑦ 클라이언트와 서버는 master secret을 사용하여 세션 동안에 교환될 정보의 암호화와 복호화 및 무결성에 사용될 대칭키인 세션키를 생성한다.

⑧ 클라이언트는 서버에게 앞으로의 메시지는 세션키에 의해 암호화 될 것이라는 메시지를 송신하는데 이는 클라이언트 측의 handshake의 종료를 의미한다. (클라이언트의 Change Chipher Spec, Finished)

⑨ 서버는 클라이언트에게 앞으로의 메시지는 세션키에 의해 암호화 될 것이라는 메시지를 송신하는데 이는 서버 측의 handshake의 종료를 의미한다. (서버의 Change Chipher Spec, Finished)

⑩ SSL handshake가 종료되고 SSL 세션이 시작된다. 클라이언트와 서버는 세션키를 사용하여 데이터의 암호화·복호화 및 송·수신을 하며 데이터의 무결성을 검사한다. (Record 프로토콜)

3.2 TLS(Transport Layer Security)

IETF TLS working group에서 SSL을 개정하여 유선상의 보안 프로토콜 표준화가 진행 중이다. 현재 RFC2246(TLS v1.0)이 개발되었으며, 이 TLS를 국제표준 SSL v3.1이라고 부르기도 한다. 이 프로토콜은 서버, 클라이언트 인증을 제공하고 기밀성도 보장한다. 이것의 구조와 동작은 그림2의 SSL과 같으며, Handshake과정은 그림5와 같다[7].

① 클라이언트는 클라이언트 헬로우 메시지를 전송함으로써 서버에게 연결을 요청한다. 이 때 클라이언트가 사용할 관용 알고리즘 목록, 공개

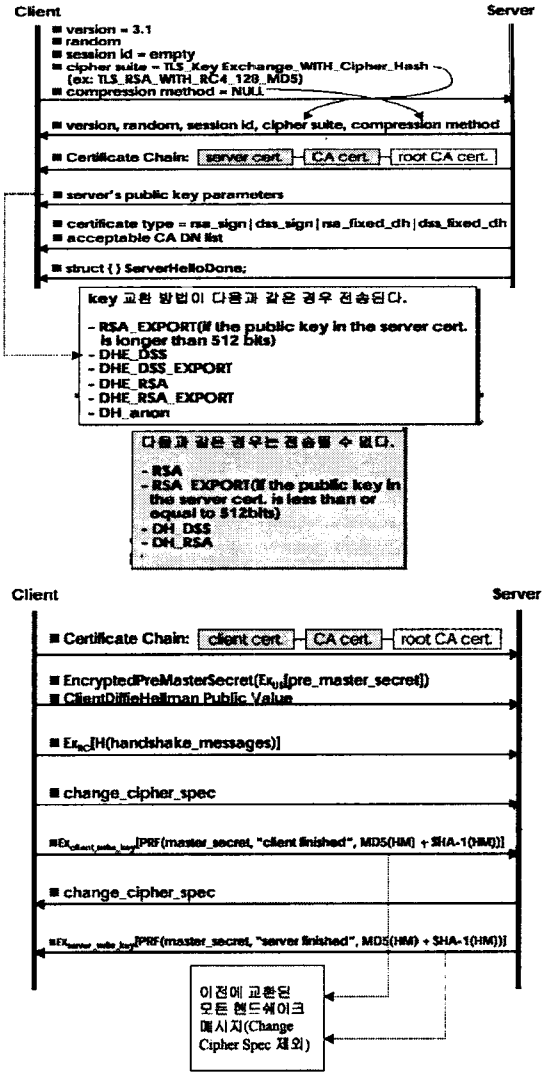


그림 5. TLS Handshake 과정

키 알고리즘의 목록, 압축 방법의 목록을 전송한다.

② 클라이언트의 헬로우 메시지를 수신한 서버는 응답으로 서버 헬로우 메시지를 전송한다. 이 때 서버는 클라이언트가 전송한 암호 매개변수의 목록 가운데 세션에서 사용할 것을 결정해서 전송한다.

③ 서버의 인증을 위해 서버 인증서를 클라이언트로 전송한다(인증서는 Root 인증기관(CA)

이나 하위 인증기관에서 발행하는 공인인증 문서이다). 서버 인증서의 public key parameters는 그림5와 같은 key 교환 방법일 경우에만 전송된다.

④ 클라이언트의 인증서를 요청한다(클라이언트는 서버가 허용할 경우 Null 인증서도 가능하다).

⑤ 위의 과정을 통해 상호 인증하고 필요한 암호 매개변수를 생성한 클라이언트와 서버는 Finished 메시지를 보내 Handshake 과정을 종료하고 실제 데이터를 교환한다. 이때 Change_cipher_spec 메시지를 보내는 동시에 Finished 메시지를 전송한다. 이 Finished 메시지는 의사난수함수(PRF: Pseudo Random Function)를 이용한 해쉬함수 값이다. 그리고 의사난수함수는 WTLS에서도 사용되지만, SSL은 사용하지 않는다.

3.3 WTLS(Wireless Transport Layer Security)

3.3.1 개요

WAP 포럼에서 TCP/IP와는 별도의 무선환경에 적합한 프로토콜을 정의하고 있는 보안 프로토콜이 WTLS이다[8].

WTLS는 인터넷 보안 메커니즘으로 잘 알려져 있는 SSL/TLS에 기반해서 작성되었다. 이것은 현재 버전 1.2까지 갱신되었고, WAP 프로토콜과 함께 사용되며 협대역 통신 채널을 위해 최적화되어 있다. 또한 단말기간의 통신 보안을 위해 사용되는 것으로 보안 요청과 네트워크의 특징에 따라 WTLS 기능을 선택적으로 사용할 수 있다. 그리고 WTP(Wireless Transaction Protocol)와 WDP(Wireless Datagram Protocol) 사이에서 수행되기 때문에 특정 응용프로그램에 종속되지 않고, WAP을 사용하는 모든 응용프로그램들을 지원한다. WTLS는 앞에서 언급했듯이 다음과 같은 보안서비스를 제공한다[1].

- 두 응용간의 기밀성 서비스
  DES, IDEA 등과 같은 관용 암호화 방식을 사용하여 기밀성을 제공하며, 어플리케이션 각 각에 대해 다양한 암호 알고리즘을 제공한다.
- 클라이언트와 서버의 상호 인증 서비스
  연결 설정 과정에서 서로간에 신뢰할 수 있도록 클라이언트와 서버가 서로에 대해 인증(RSA와 같은 공개키 암호방식과 X.509 인증서 사용) 서비스를 제공한다.
- 무결성 서비스
  내부적으로 누군가 데이터 전송을 방해할 수 없도록 하거나 재전송 공격에 이용할 수 없도록 MAC(Message Authentication Code)으로 데이터의 무결성을 제공한다.

3.3.2 구조와 프로토콜의 기능

WTLS은 그림6과 같이 SSL/TLS를 기반으로 하기 때문에 기존 유선 프로토콜과 같다고 보면 된다[7,8].

Handshake 프로토콜, Alert 프로토콜, Change Cipher spec 프로토콜은 WTLS의 동작에 대한 관리를 위해 사용되며, 실질적인 보안 서비스는 Record프로토콜에서 제공된다.

(1) WTLS의 상위계층(Control Protocol)

▶ Handshake Protocol

Record 계층의 상위에서 클라이언트와 서버를

<table>
<tr><td colspan="4">Transaction Protocol(WTP) Application</td></tr>
<tr><td>Handshake Protocol</td><td>Alert Protocol</td><td>Change Cipher Spec Protocol</td><td>Application Data Protocol</td></tr>
<tr><td colspan="4">Record Protocol</td></tr>
<tr><td colspan="4">Datagram Protocol(WDP/UDP)</td></tr>
<tr><td colspan="4">Bearer Networks</td></tr>
</table>

그림 6. WTLS 구조

상호 인증하고 암호 알고리즘, 암호키, MAC알고리즘 등의 세션 상태의 보안 속성을 협상하게 한다. 여기서 협상된 것은 사이퍼 슈트로 묶여지고, 키 교환 알고리즘을 이용하여 premaster secret를 생성한다. premaster secret는 master secret를 생성하게 하고 master secret를 이용하여 session keys를 생성한다.

▶ Change Cipher Spec. Protocol

클라이언트와 서버의 협상된 Cipher 규격의 암호키를 이용하여 추후의 레코드 계층의 메시지를 보호할 것을 명령하고, 클라이언트/서버 양쪽에서 전송된다. 이때 두 개의 상태(pending state, current state)가 존재하는데, 이 메시지를 받은 후에 Record계층은 'Change Cipher Spec.'으로 메시지 전송 후에는 'Write pending state'를 'Write current state'로 복사하고, 'Change Cipher Spec.' 메시지 수신 후에는 'Read pending state'를 'Read current state'로 복사한다. 이때 클라이언트/서버 간의 비밀통신을 보장받게 된다.

▶ Alert Protocol

보안 프로토콜의 전 구간에 걸쳐 다양한 에러 메시지를 전달한다. 이것은 경고일 수도 치명적일 수도 있다. 치명적인 에러인 경우는 즉각 연결(connection)을 닫아야하고, 이 연결과 관련된 세션 ID, keys, secret을 버려야 한다. 반면에, 경고 에러는 MAC 검증이 실패된 경우의 에러로 수신 측에서는 수신된 레코드를 무시하면 된다. WTLS에는 SSL/TLS에 없는 Critical error가 있다. 이 때에도 연결을 즉각 닫아야 하지만, 세션ID는 그대로 보존된다.

▶ Application Data Protocol

어플리케이션 데이터는 Record Layer로 전송되고 "current connection state"일 때 분할, 압축, 암호화가 된다. 이러한 Record Layer의 데이터를 transparent data라 한다.

(2) WTLS의 하위계층(Record Protocol)

WTLS 연결이 수립된 후, 실질적인 보안 서비스를 제공하는 층이다. 강력한 암호 알고리즘으로 기밀성을 제공하고 MAC을 사용해서 데이터의 무결성을 제공하도록 한다. 따라서 응용계층에서 전달되는 데이터는 단편화, 압축, MAC 삽입, 암호화 과정을 거쳐 상대측으로 전송되고, 상대측에서 수신된 데이터를 복호화하고 재조립하여 원시 데이터를 응용계층에 전달하는 역할을 수행한다.

3.3.3 동작

WAP에서 클라이언트와 서버가 WTLS를 통해서 연결을 할 경우, 먼저 Handshake 프로토콜을 수행하여 한 세션 동안 보안 서비스 제공에 사용되는 세션키, 암호 알고리즘, 인증서 등과 같은 암호 매개변수를 서로 공유하게 된다. 여기서 생성된 세션 정보는 Record 프로토콜에서 보안 서비스를 제공하는데 이용된다[8].

그림7은 WTLS 프로토콜의 전체적인 흐름을 나타내고 있다. 이것은 보안 서비스 제공에 필요한 세션을 생성하는 단계인 'Session State'와 이

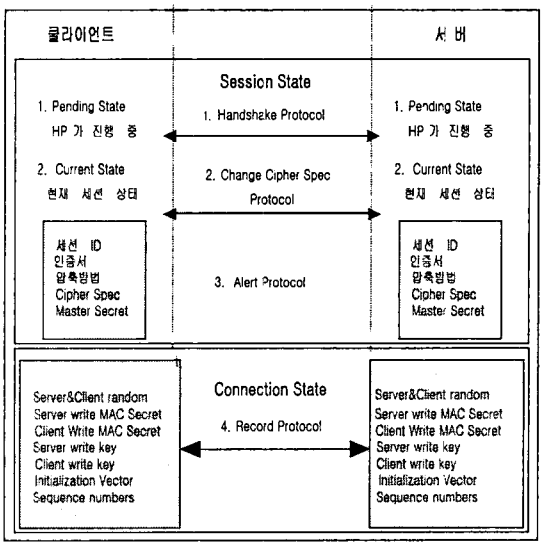


그림 7. WTLS 프로토콜의 개요

세션 정보를 이용해 클라이언트와 서버가 메시지를 주고받는 'Connection State'로 구분할 수 있다.

앞에서 살펴본 바와 같이 WTLS로 구현된 클라이언트가 서버에 연결을 시도하는 시점에서 'Session State'가 시작되며, 세션정보는 Handshake 프로토콜이 진행중인 상태인 'Pending State'와 Handshake 프로토콜이 완료되는 시점에서 Change Cipher Spec. 프로토콜을 통하여 이루어지는 'Current State'단계를 거쳐 생성된다.

세션은 여러 개 생성될 수 있으며, 또한 세션을 재사용을 할 수 있도록 함으로써 Handshake프로토콜에서 주고받는 메시지를 줄여 효율적으로 동작할 수 있도록 하고 있다. 그리고 세션동안 보안 서비스 제공에 사용되는 세션키, 암호키, 암호 알고리즘, 인증서 등과 같은 암호 매개변수를 서로 결정해야 하는데, 이는 Handshake 프로토콜을 이용해서 이루어진다. 이를 위해서 교환되는 정보는 표1과 같다.

Compression Method, Cipher Spec., Master Secret와 같은 암호 매개변수를 결정하는 Handshake 과정은 크게 Full Handshake, Abbreviated Handshake, Optimized Full Handshake로 구분할 수 있다. 이 가운데 Full Handshake, Abbreviated Handshake는 SSL/TLS에서도 사용되는 방법으로 Full Handshake는 새로운 세션을 시작할 때 사용되는 것이며, Abbreviated Handshake는 기존의 세션을 재개해서 다시 이용할 경우에 사용된다.

▶ Full Handshake

Full Handshake과정은 그림4의 SSL 경우와 같다. 클라이언트는 Client Hello 메시지를 전송함으로서 서버에게 연결을 요청한다. 이때 클라이언트가 사용할 수 있는 관용 암호 알고리즘 목록, 공개키 알고리즘의 목록, 압축 방법의 목록들을 전송한다. Client Hello 메시지를 수신한 서버는

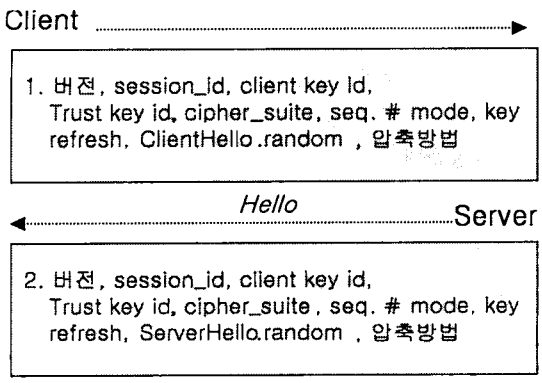
표 1. Handshake에서 교환되는 정보

정 보	설 명
Session Identifier	서버가 세션을 식별하는데 사용하는 임의의 수
Protocol Version	WTLS 버전
Peer Certificate	서버 및 클라이언트 인증서
Compression Method	데이터 암호화에 앞서 사용되는 압축 방법
Cipher Spec	사용되는 관용 알고리즘 및 MAC 알고리즘
Master Secret	클라이언트와 서버에 의해서 공유되는 20바이트의 비밀정보
Sequence Number Mode	현재 세션에서 사용되는 일련번호 사용방법 (off, implicit, explicit)
Key Refresh	보안 서비스 제공에 사용되는 정보(암호키, MAC 정보, IV)등의 교체 주기
Is Resummable	현재 세션이 새로운 세션을 시작하는데 사용될 수 있는지 여부를 나타내는 표시자

이에 대한 응답으로 Server Hello 메시지를 전송한다. 다만, SSL/TLS과 다르게 Hello 메시지를 주고받을 때 교환되는 정보는 그림8과 같다. 이때 서버는 클라이언트가 전송한 암호매개변수들의 목록에서 세션에서 사용할 것을 결정하여 전송하며, 서버 인증을 위해서 클라이언트 인증서를 요청한다. 이 과정을 통해서 서로를 인증하고 필요한 암호 매개변수를 생성한 클라이언트와 서버는 Finished 메시지를 보내서 Handshake 과정을 종료하고, 실제 데이터를 교환하게 된다.

▶ Abbreviated Handshake

Abbreviated Handshake에서는 그림9와 같이 이전 세션 정보를 이용하여 시작하기 때문에 서버와 클라이언트 인증을 위해 필요한 정보는 교환되지 않으며, 이전 세션에서 사용한 암호 매개변수로부터 새로운 세션에서 사용될 매개변수를 생성한다.



Client Key ids	클라이언트가 지원하는 암호 Key exchange option과 identities 목록
Trust Key ids	클라이언트의 신뢰받는 인증서 목록 With theclient's first preference
Cipher suite	Struct{ BulkCipher(Algorithmbulk_Cipher_algorithm; MACAlgorithmmac_algorithm; }
Key exchange suit	Key exchange 알고리즘 목록
key refresh	몇 개의 메시지가 전송될 때마다 암호키, MAC secret, IV를 업데이트할 지를 결정
Seq. # mode	Sequence numbering을 어떻게 할 것인가의 정보

그림 8. SSL과 다른 WTLS의 협상 정보

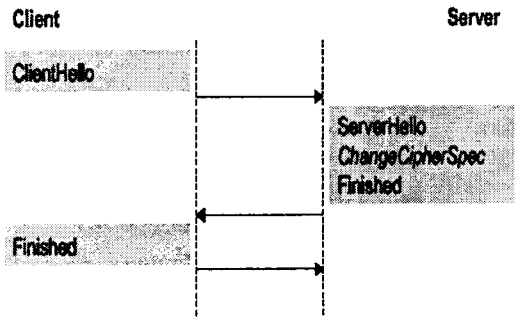


그림 9. Abbreviated Handshake

▶ Optimized Full Handshake

Optimized Full Handshake는 WTLS에서 새롭게 추가된 것으로 그림10과 같이 서버는 클라이언트 인증을 위해 클라이언트의 인증서를 요청하지 않고, 서버내에 보관하거나 저장소를 통해 제공되는 클라이언트 인증을 수행한다.

이렇게 Handshake 과정을 마치게 되면, 그림11의 Record 프로토콜 층에서 데이터를 압축하고, 해쉬 및 암호화를 수행하여 데이터를 전송하며, 수신측에서는 이 데이터를 복호화 및 검사를 한다. 그리도 데이터의 단편화는 하지 않고, SSL과 마찬가지로 압축도 정의는 되어 있지만 현재는 사용하지 않는다[1,8].

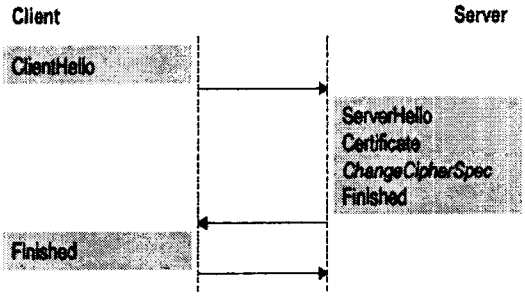


그림 10. Optimized Handshake

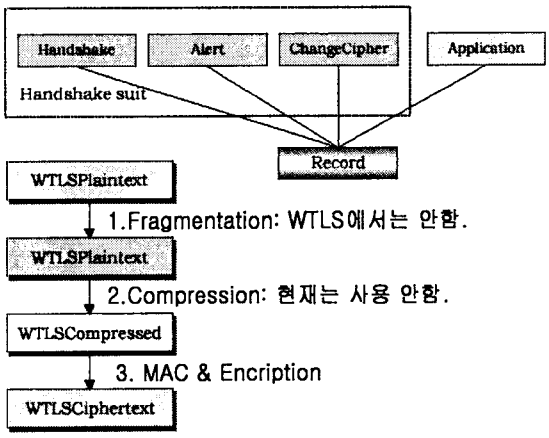


그림 11. Record 프로토콜의 동작 과정

3.4 SSL/TLS와 WTLS 비교

여기에서는 SSL/TLS와 WTLS를 여러 가지 측면에서 비교한다[6-8].

3.4.1 개요

SSL	1994년 Netscape사에서 처음으로 제안, 현재 SSL v.3
TLS	IETF TLS W/G에서 SSL을 개정하여 표준화가 진행 (RFC 2246: TLS v.1)
WTLS	TLS를 바탕으로 WAP포럼에서 표준화 중 (WAP v.2)

3.4.2 목적

- SSL/TLS

보안 서비스 (Security)	• 기밀성 • 무결성 • 사용자 인증
상호호환성 (Interoperability)	• TLS/SSL을 이용한 어플리케이션간의 상호호환성 확보
확장성 (Extensibility)	• 새로운 알고리즘 추가의 용이성
효율성 (Efficiency)	• connection의 수를 줄이기 위한 세션 캐시 스킴

- WTLS

- 보안 서비스(기밀성, 무결성, 사용자 인증) 제공
- 제한된 Processing Power, Limited Capacity 고려
- 다양한 무선환경에서 동작(Datagram transport 지원)

3.4.3 Handshake

- SSL/TLS

- Full Handshake: 새로운 세션을 시작할 때
- Abbreviated Handshake: 이전 세션을 resume하여 사용함

- WTSL

- Full Handshake, Abbreviated Handshake
- Optimized Full Handshake: 서버는 클라이언트의 인증서를 요청하지 않음 (premaster secret와 master secret가 자체적으로 계산 가능)

3.4.4 Alert 프로토콜

다음은 SSL과 WTLS가 유일하게 가지는 에러 메시지이다.

- SSL:

no_certificate(41)

- WTLS:

no_connection(5),
time_required(11),
unknown_key_id(52),
disabled_key_id(53),
key_exchange_disabled(54),
session_not_ready(55),
unknown_parameter_index(56),
duplicate_finished_received(57),
no_certificate(41)

표2는 SSL, TLS와 WTLS의 에러 메시지를 비교한 것이다.

3.4.5 Confidentiality

▶ session keys

connection마다 master secret(shared secret)로 생성한다.

표 2. Alert 메시지 비교

SSL	TLS	WTLS
총 12개	- 총 13개 - no-certificate 제외한 SSL3.0의 모든 Alert 메시지 지원	- 총 32개 - TLS의 모든Alert 메시지 지원

- 서버 write MAC secret
- 클라이언트 write MAC secret
- 서버 write 키
- 클라이언트 write 키
- 클라이언트/서버 초기벡터

SSL_Master_secret(48byte)=
MD5(pre_master_secret+**SHA**(A+pre_master_secret+ClientHello.random+ServerHello.random)) +**MD5**(premaster_secret+**SHA**(BB+pre_master_secret+ClientHello.random+ServerHello.random))+**MD5**(pre_master_secret+**SHA**(CCC+pre_master_secret+ClientHello.random+ServerHello. random));

▶ 의사난수함수(PRF: Pseudo Random function)

입력으로 secret, seed, identifying label을 가지며, 임의 길이의 출력값을 생성한다. SSL은 의사난수함수를 사용하지 않는다.

- **TLS_PRF**(secret, label, seed) =P_**MD5**(s1, label+seed)**XOR** P_**SHA-1**(s2, label+seed)
- **WTLS_PRF**(secret, label, seed) =P_**hash**(secret, label+ seed);

▶메시지 인증 (MAC Calculation)

능동적인 공격에 대응하여 통신보안의 취약성

P_hash (secret, seed)=
HMAC_hash(secret, A(1)+seed) +
HMAC_hash(secret, A(2)+seed) +
HMAC_hash(secret, A(3)+seed)....
A(0) = seed
A(i)= HMAC_hash(secret, A(i-1))

- **SSL_HMAC**=
hash(MAC_write_secret+pad_2
hash(MAC_write_secret+pad_1+seq_num+SSLCompreesed.type+SSLCompressed.length+SSLCompressed.frgment));
- **WTLS_HMAC**=
HMAC_hash(MAC_secret, seq_number+WTLSCompressed.record_type+WTLSCompressed. length+WTLSCompressed.fragment));

이 점차 커짐에 따라 강력한 메시지 인증이 요구된다. 따라서 cryptographic MAC을 사용하여 데이터 무결성을 보호한다. HMAC은 다양한 서로 다른 해쉬 알고리즘이 사용될 수 있다. (SHA-1, MD5...)

표3은 키 생성, 표4는 Cipher Suites의 측면에서 SSL/TLS와 WTLS를 비교한 것이다.

3.4.5 정리

▶ SSL/TLS

- SSL/TLS는 종점간의 보안기능을 제공한다.
- Netscape와 Microsoft사가 지원하는 널리 이용되는 표준이다.
- SSL에 바탕을 두고 부인방지 기능을 갖춘 새로운 지불프로토콜이 개발될 예정이다.
- 고속 동작이 가능하고 실현이 용이하며, 유수의 인증기관이 SSL용 인증서를 발행하고 있다.

표 3. SSL/TLS와 WTLS의 키 생성 비교

비교	SSL	TLS	WTLS
master secret	MD5를 3번 더해서 (48byte)	PRF= P_MD5 XOR P_SHA-1(48byte)	PRF 함수 (20byte)
Key block	MD5를 계속 더함	PFR 함수 사용	PRF 함수 (입력에 seq_num)

표 4. Cipher Suites 비교

비교	SSL	TLS	WTLS
Key exchange	DH, RSA, DSA, Fortezza	DH, RSA, DSA	DH, RSA, ECDH
Symmetric	IDEA, RC2, RC4, DES, 3DES, Fortezza	IDEA, RC2, RC4, DES, 3DES	IDEA, RC5, DES, 3DES
Hash	MD5, SHA	MD5,SHA	SHA, SHA_XOR, MD5

- 단점: 부인방지 서비스가 없고, 서버상에 저장된 데이터에 대한 보호 기능이 없다.(세션 정보가 서버에 저장) 그리고 암호키와 인증키가 동일한 마스터키로부터 생성되어 키 관리의 허점이 보인다.

▶ WTLS

- WAP의 통신을 위한 안전한 end-to-end connection을 제공하는 첫 번째 시도이다. (TLS/SSL기반, 무선환경에 맞게 변형)
- Record layer는 선택적으로 sequence numbering을 사용하지만, 데이터그램 전송을 위해서는 반드시 필요하다.
- Man-in-the-middle attack을 방지하기 위해서는 익명 인증을 거부해야 한다.
- 무선환경에서의 공개키 기반구조 확립이 필요하다.

4. WTLS 프로토콜의 취약성 분석

4.1 SSL/TLS에서 상속된 취약성

WTLS는 SSL/TLS를 기반으로 개발되었고, 동작 과정도 기존 유선 프로토콜과 유사하여, 각 프로토콜이 가지는 결함 역시 내포되게 된다. 따라서, 유선에서 일어나는 공격이 무선에서도 가능한 여러 문제점이 예상된다[3,5].

4.1.1 SSL/TLS의 Handshake 취약성

SSL의 Handshake에서 발생하는 공격 중 대표적인 것으로 Ciphersuite rollback attack과 Key exchange algorithm rollback 그리고 Dropping the change cipher spec message attack 등이 있다[3].

① Ciphersuite rollback attack

클라이언트와 서버간에 새로운 세션을 설정하고자 할 때, 그림12에서 보인 바와 같이 먼저 Hello 메세지 (1), (2)로 보안연결 협상을 시작한다. 그리고 클라이언트가 Hello 메시지에 있는 cipher_suite의 목록을 서버에 전송하면 서버는 그 목록 중 하나로 Cipher suite을 결정하게 된다. SSL/TLS의 Cipher suite의 목록에는 데이터 전송시 필요한 암호 규격과 Key exchange 알고리즘의 정보를 가지지만, WTLS는 Cipher suite와 Key exchange의 목록이 분리되어 있어 각각의 정보를 가진다.

이때 각 Hello 메시지는 평문으로 전송됨으로 제3자에 의해 공격을 받게 된다. 공격자는 Hello

```
struct { ProtocolVersion client_version;
  Random random;
  SessionID session_id;
  CipherSuite cipher_suites<2.. 2^16-1>;
CompressionMethodcompression_methods
<1..2^8-1>;
  } ClientHello;
struct { ProtocolVersion server_version;
  Random random; SessionID session_id
 CipherSuite cipher_suite;CompressionMethod
 compression_method;
  } ServerHello;
```

```
ClientHello (1)---------->
            <--------- (2) ServerHello
             <------------ Certificate*
             <-------- ServerKeyExchange*
             <---------CertificateRequest*
             <------- ServerHelloDone
Certificate* ---------->
ClientKeyExchange ----->
CertificateVerify* ------->
[ChangeCipherSpec](3)
Finished--------->
                  (4)[ChangeCipherSpec]
                  <--------- Finished
Application Data <------->Application Data
```

그림 12. Full Handshake

메시지를 중간에서 가로채어 클라이언트가 전송한 cipher_suites의 목록을 수정하여 취약한 암호 규격을 사용하게끔 한다.

② Key exchange algorithm rollback

Ciphersuite rollback attack으로 선택된 Key-exchange 알고리즘을 변경하는 공격이다. 이것은 Hello 메시지가 평문으로 전송되기 때문에 가능한 공격이다. 그림13은 Key exchange algorithm rollback의 예이다. 클라이언트가 RSA key exchange 알고리즘을 사용할 때, 서버는 Diffe-Hellman key exchange 알고리즘을 사용하도록 하는 것으로 당사자들은 그 사실을 알 수 없게 된다. C(클라이언트)는 RSA_.... 암호목록을 전송하지만, 중간에서 M(공격자)이 이 정보를 가로채어 DHE_RSA_... 로 변경한다. 그리고 다시 변경된 정보를 S(서버)에게 전송한다. 서버는 이 사실을 알아채지 못하고, 이 정보를 가지고 모든 데이터를 암호화하고 복호화하게 된다. 따라서 당사자들은 서로 다른 규격의 암호를 사용하게 된다.

이것은 Hello 메시지가 평문으로 전송되기 때문에 가능한 공격이다. 그리고 이러한 공격은 어떤 종류의 KeyExchange 알고리즘을 사용하는지에 대한 filed를 서명하지 않으므로 그 field가 보호받지 못하게 되고 공격을 받게 된다. 그림14를 보면

[client hello:]
1. $C \rightarrow M$: **SSL_RSA_...**
1'. $M \rightarrow S$: **SSL_DHE_RSA_...**
[server hello:]
2. $S \rightarrow M$: **SSL_DHE_RSA_...**
2'. $M \rightarrow C$: **SSL_RSA_...**
[server key exchange:]
3. $S \rightarrow M$: $\{p, g, y\}_{K_S}$
3'. $M \rightarrow C$: $\{p, g, y\}_{K_S}$
[client key exchange:]
4. $C \rightarrow M$: $k^g \bmod p$
4'. $M \rightarrow S$: $g^x \bmod p$
...

그림 13. KeyExchange 알고리즘 변경

KeyExchange 알고리즘이 선택되고 그때 서명을 함으로써 그 공격을 방지할 수 있다.

③ Dropping the change cipher spec message attack그림12의 Change Cipher Spec message (3), (4)는 협상된 cipher spec으로 암호키를 이용하여 추후의 레코드 계층의 메시지를 보호할 것을 명령하는 것으로 pending state가 current state로 갱신하게 한다. 이것을 공격자가 가로채고 제거하여 클라이언트와 서버가 current ciphersuite을 갱신하지 못하도록 하면 데이터의 무결성을 보호받지 못하게 한다.

4.2 WTLS의 Handshake 취약성

앞에서 WTLS 프로토콜은 기존 유선 SSL/T LS 프로토콜을 기반하고 있으므로 WTLS Handshake의 설계 결함 또한 유사하다고 언급하였다[3,5].

4.2.1 Cipher suite와 KeyExchange 알고리즘 선택방식

Hello 메세지에서 cipher suite와 key exchange 알고리즘의 선택 방법은 Client Hello에서 클라이언트가 자신이 소유한 암호방식과 키 교환 방식을

```
enum { rsa, diffie_hellman, fortezza_kea }
KeyExchangeAlgorithm

struct{
 select (KeyExchangeAlgorithm)
  {case diffie_hellman:
          ServerDHParams params;
          Signature signed_params;
          case rsa:
          ServerRSAParams params;
          Signature signed_params;
          case fortezza_kea:
          ServerFortezzaParams params;};
              } ServerKeyExchange;
```

그림 14. KeyExchange 알고리즘의 선택

서버에게 제시하면 서버는 수신된 목록들 중 하나를 선택하여 그것의 순서 번호를 클라이언트에게 보낸다. 서버는 일반적으로 목록들 중 우선순위가 높은 것을 선택하여 보다 강력한 암호를 사용하고자 한다. 하지만, Hello 메시지가 평문으로 전송됨으로 공격자가 cipher suite을 Null 혹은 취약한 수출용 모드로 바꾸거나, KeyExchange 방식을 SSL처럼 rollback 공격으로 변경시킬 우려가 있다.

특히 공격자는 ClientHello 메시지에서 제안하는 KeyExchange 방식 리스트를 anonymous로 모두 대치시킴으로서 사용되는 KeyExhange 방식을 anonymous의 형태가 되도록 만들 수 있다. 그렇게 되면 자신이 클라이언트와 서버의 중간에서 Parallel session을 설립한 후, 클라이언트와 서버를 동시에 위장하는 공격에 성공할 수 있다. 그림15는 공격 시나리오의 한 예를 보여준다.

위의 공격 시나리오에서 주목해야할 점은 클라이언트, 서버, 그리고 공격자가 같은 premaster_secret와 master_secret을 공유하지만, 클라이언트와 서버는 그 사실을 알 수 없다는 것이다.

4.2.2 man-in-the middle attack

서버가 인증이 될 때마다, 채널은 man-in- the middle attack에 대하여 안전해야 된다. 하지만, 익명 세션은 그러한 공격들에게 선천적으로 취약하다. 왜냐하면 익명 서버들은 클라이언트를 인증할 수 없기 때문이다. 그리고 메시지를 검증하는 인증서 내의 클라이언트 서명은 특정 서버를 연관시키기 위해서 서버의 인증서를 요구한다. 따라서 익명 연결은 단지 수동적인 도청에 대한 방지만을 제공한다.

4.2.3 RSA_anon, DH_anon, ECDH_anon의 취약성

RSA_anon 형태의 Handshake는 공격자가 Server

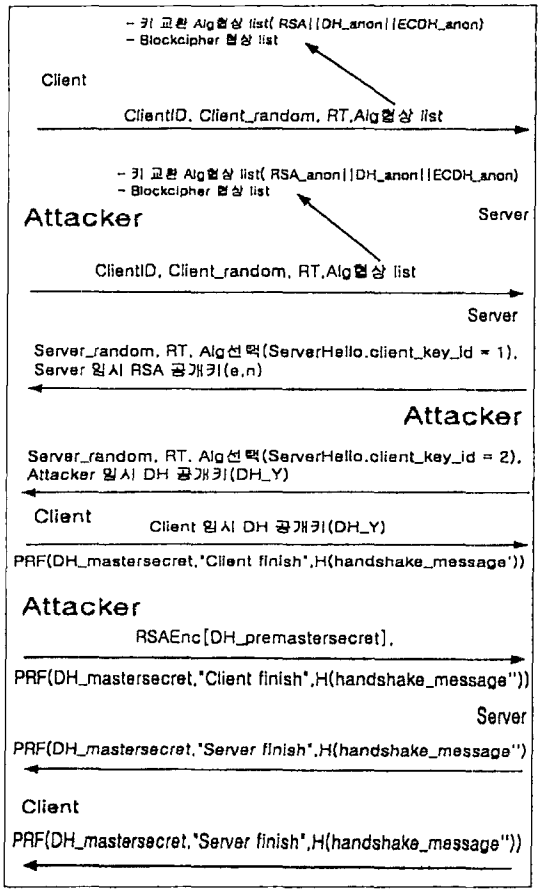


그림 15. Full handshake에 대한 공격

Hello 메시지를 수신한 후에 20 바이트의 비밀값인 위장된 premaster_secret을 생성하여, 수신한 서버의 공개키로 암호화해서 서버로 전송함으로써 클라이언트로 위장하는 것이 가능하다. 또한 공격자가 Server Hello내의 공개키를 자신이 생성한 공개키로 변경함으로써 클라이언트가 생성한 premaster_secret값을 알아낼 수 있다.

4.2.4 RSA with Server Certificate 취약점

RSA with Certificate 형태의 Full Handshake 프로토콜은 Hello 메시지 교환에서 RSA가 선택되고 서버가 인증서를 보낸 후 클라이언트의 인증서를 요구할 때, 클라이언트가 자신의 RSA 인증서를 가지고 있지 않을 경우에 이루어질 수 있다.

그리고 서버가 클라이언트 인증서를 요구하는 메시지를 보내지 않을 경우도 이루어질 수 있는 프로토콜이다. 능동적 공격자에 의한 클라이언트 위장이 가능함을 의미하고 또한 Handshake 프로토콜의 목적인 인증된 키 설립을 달성하지 못한다는 것을 의미한다. 따라서 그림16과 같은 시나리오를 가지는 클라이언트 위장 공격이 가능하게 된다.

공격자는 단계1, 단계2의 메시지를 도청하여 교환된 메시지를 확인한다. 그리고 나서 클라이언트가 서버에게 보내는 메시지를 자신이 생성한 메시지로 대체하여 단계3을 수행한다. 서버는 단계3'에서 바뀐 메시지를 공격자가 아닌 클라이언트가 보낸 것으로 믿고 premaster secret'로부터 master- secret' 및 Finished 메시지를 계산하여 단계4를 진행한다. 이때 서버는 클라이언트와 master_secret'를 공유하고 있다고 믿는다. 단계4'는 선택적인 것으로 클라이언트가 이 메시지를 수신할 경우 Finished 증명이 불가능함으로 프로토콜 수행을 중지하게 되고 공격자가 이 메시지를 보내지 않으면 클라이언트는 정해진 시간만큼 기다리다가 연결요청을 철회하는 DOS(Denial of Service)공격이 가능하게 된다.

1. Client->Server : ClientID, Client_random, RT, Alg협상 list
2. Server->Client : Server_random, RT, Alg선택, Server_RSACert
3. Client->Attacker : RSAEnc[premaster_secret],
   PRF(mastersecret,"Client finish", H(handshake_message))
3'. Attacker->Server : RSAEnc[premaster_secret'],
   PRF(mastersecret',"Client finish",H(handshake_message'))
4. Server->Attacker :
   PRF(mastersecret',"Server finish",H(handshake_message'))
4'. Attacker->Client :
   PRF(mastersecret',"Server finish",H(handshake_message'))

* 3', 4': 클라이언트와 서버 중간에서 클라이언트와 서버로 위장하여 통신

그림 16. RSAwithServerCertificate 공격 시나리오

4.2.5 Dropping the Change Cipher Spec Message

WTLS의 이 메시지 역할도 서로 협상한 암호 규격으로 송/수신될 데이터의 무결성을 보장하고 인증하기 위해 pending cipher를 current cipher로 갱신하도록 한다. 그러나, 제3자가 중간에서 이 메시지를 변경하거나 삭제를 해버리면 그림17과 같이 current state로 전환하지 못하고 pending state로 보호받지 못한 메시지를 주고받게 된다.

4.2.6 그 밖의 취약성

① Master secret 보호

모든 세션키는 master_secret로 생성되고, Handshake 프로토콜에 대한 무결성 보호도 master_secret에 의존함으로 이것의 보호가 매우 중요하다. Abbreviate Handshake에서 클라이언트가 이전에 사용된 세션을 재 사용하고자 하는 경우, master_ secret는 이전의 master_secret를 그대로 사용하고 클라이언트와 서버가 생성하는 난수값만이 변한다. 공격자는 master_secret와 어떤 공개 평문과의 관계를 알아내는데 이러한 점을 이용한다. 즉, 클라이언트를 가장한 공격자가 많은 수의 세션 재사용을 요청할 경우, 서버는 각각에 대해서 Finished 메시지를 전송한다는 점을 이용하는 것이다. 이것은 결국 Finished 메시지를 통해 많은 공개 평문을 알아내게 하고, 이를 master_secret를 알아내기 위하여 사용하는 것을 의미한다.

...
1. $C \rightarrow M$: **[change cipher spec]**
2. $C \rightarrow M$: **[finished:]** $\{a\}_k$
2'. $M \rightarrow S$: **[finished:]** a
3. $S \rightarrow M$: **[change cipher spec]**
4. $S \rightarrow M$: **[finished:]** $\{a\}_k$
4'. $M \rightarrow C$: **[finished:]** a
5. $C \rightarrow M$: $\{m\}_k$
5'. $M \rightarrow S$: m

그림 17. Change Cipher Spec 메시지의 유실

② Replay, Delay, deleted data attack 등

공격자가 데이터를 가로채어 상대방에게 재전송을 한다든지, 그 데이터를 즉각 보내지 않고 지연을 시킬 수 있다. 또한 가로챈 데이터를 제거하여 전송을 못하도록 할 수도 있다. 이러한 경우는 잘못된 데이터로 인해서 당사자들에게 혼란을 초래하게 된다.

4.3 WTLS 자체의 취약성

4.3.1 예상 가능한 IV(Initial Vector)의 사용

예상 가능한 초기 벡터를 사용하게 되면 선택 평문공격에 취약하게 된다. 왜냐하면 IV는 CBC 모드의 블록 암호를 사용함으로써 엔트로피(entropy)를 생성하기 때문이다. 또한, 낮은 엔트로피는 CBC 모드 블록 암호에 사용되는 대칭 키를 보호하는데 필요하다.

통상 IV없이 원래의 평문은 master key로서 암호화된다. 이는 shared secret를 발견하기 위해서 brute force를 사용할 가능성을 제공하게 된다.

또한 WTLS가 데이터그램이 분실되거나, 반복 또는 재정렬되는 상황에서 신뢰할 수 없는 데이터그램을 지원하기 때문에 CBC 모드는 각각의 패킷들을 암호화하기 위해서 새로운 IV를 필요로 한다. 사용되는 IV는 패킷의 순차번호와 키 생성시에 유도된 원래의 IV를 XOR함으로써 구해진다. 따라서 패킷의 첫 번째 평문블록은 계산된 IV와 XOR된다. 원래의 IV는 handshake동안 받은 값에 기반해서 계산된다. 이러한 모든 값들(client_random, server_random, sequence num- ber)은 암호화 없이 전송되므로 도청될 수 있다. 또한 예상 가능한 IV는 낮은 엔트로피의 secret에 대한 chosen-plaintextattack을 가능하게 한다. 이 안전성 문제는 기밀성에 영향을 끼치는 취약성을 갖게 된다.

4.3.2 XOR MAC과 stream ciphers

WTLS는 40비트 XOR MAC을 지원하기 때문에 메시지를 zero로 패딩하고 5바이트 블록으로 나누어 이러한 블록들을 함께 XOR시킨다. 이때 스트림 암호가 사용되면, 키 길이에 상관없이 XOR MAC은 메시지 무결성 보호를 제공하지 못한다. 또한 inverting이 MAC에서 수행된다면 암호문에서 어떤 한 비트는 invert될 수 있다. 따라서, 컨텐츠가 변경이 되었더라도 무결성 체크는 성공할 수 있다. 그러므로 이 안전성 문제는 무결성에 영향을 끼치게 된다.

4.3.3 35bit DES encryption

DES 키는 각 바이트에 한 개의 패리티 비트를 포함하고 있다. 40 비트의 키를 사용하면, 실제로 DES 암호의 키 길이는 5*7=35비트이다. 40비트 DES에서 확장된 키 길이가 8바이트임에도 불구하고, 실제 키 길이는 단지 5바이트이므로 기밀성에 영향을 주게 된다.

4.4.4 The PKCS # 1 attack

PKCS #1 version 1.5는 주어진 패킷이 올바른 PKCS #1 version 1.5패딩을 가지는지를 알려주는 oracle을 포함하는 프로토콜과 사용될 때 안전성 문제를 가진다. 만약 어떠한 방법으로든 그 시스템이 침입자에게 사용된 키가 올바른가를 알려준다면, 그 시스템은 oracle을 가진다고 한다. 이 oracle을 사용하여 침입자는 모든 가능성을 시도하고 그 시스템의 응답을 체크하여 올바른 키를 구하려고 시도할 수 있다. RSA 서명과 암호화는 WTLS내의 PKCS #1version 1.5에 준하여 수행되기 때문에 RSA 메시지들은 대략 2^{20}개의 chosen ciphertext queries로서 복호할 수 있다. WTLS에서 bad_certificate와 decode_error 메시지들은 불법적인 복호를 위해서 사용될 oracle을 제공할 수

있다. 이러한 문제는 인증에 영향을 미치게 된다.

4.3.5 Undefined subgroup order on Diffie-Hellman key agreement

WTLS specification은 Diffie-Hellman 연산에서 사용되는 변수를 위한 미리 정의된 값들을 포함하고 있다. 그러나, multiplicative subgroup의 grouporder는 빠져 있다. group order가 빠짐으로써 주어진 public 값이 올바른 multiplicative subgroup에 속하는가를 검사할 수 없게 된다. 이것은 사소한 문제일 수도 있지만, 인증에 영향을 미칠지도 모른다.

4.3.6 Design of PRNGs

PRNGs(Pseudo Random Number Generator)를 설계하고 초기 값 설정시에 주의해야 한다. 안전한 해쉬 연산에 기반한 PRNGs는 대부분 MD5나 SHA로 대치할 수 있지만, randomnumber generator state의 길이보다 더 나은 안전성을 제공할 수 없다.

4.3.7 The length of temporary RSA key

512비트 RSA 키들은 high-value transaction이나 long-term 안전성을 요구하는 어플리케이션에 대해서는 충분히 안전하지 못하다. 인증서내의 공개키가 암호화에 사용될 수 없을 때, 서버는 임시 RSA 키를 서명하고 나서 교환된다. 임시 RSA 키는 허용되는 최대 길이를 가져야 한다. (512 bits).

4.4 취약성 해결 방안 제시

4.4.1 CipherSuite Rollback attack

그림18과 같이 Handshake의 마지막 단계인 finished message에서 모든 Handshake 프로토콜과 관련된 메시지를 해쉬함수로 계산하여, 그 결과 값을 다시 한번 서로 주고받아 메시지를 인증

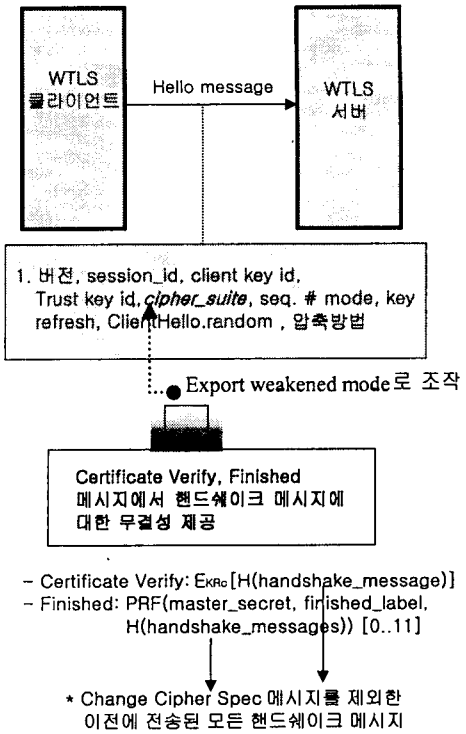


그림 18. Chiper Suite Rollback attack

한다. 따라서 Hello 메시지의 cipher suite의 목록을 변경하여 당사자들의 암호 규격을 취약하게 만드는 공격이 있는지를 알 수 있다.

4.4.2 Key_Exchange 알고리즘 Rollback attack

서버는 Key_Exchange message로 public key parameters를 전송하는데, 이에 대한 설명은 그림19와 같이 어떠한 타입의 Key_Exchange 알고리즘을 선택하였는지를 열거한 filed는 보장하지 않는다. 따라서 public parameters뿐 아니라 이와 관련된 모든 데이터를 서명하도록 WTLS를 갱신해야한다.

4.4.3 Dropping Change Cipher Spec

Change Cipher Spec 메시지는 다른 메시지와는 달리 finished 메시지에서 인증을 보장하지 못한다. 왜냐하면 Change Cipher Spec 자체가 하나의 프로토콜로서 Handshake 프로토콜과는 다른

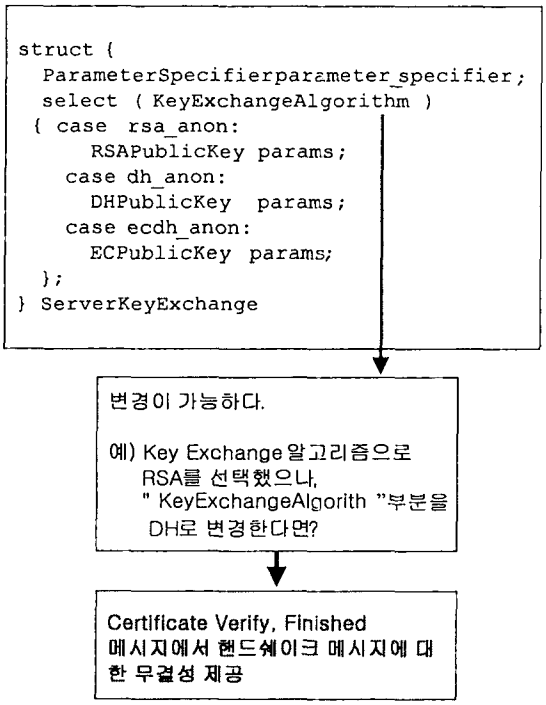


그림 19. Key_Exchange 알고리즘

Alert 프로토콜과 같은 독립된 프로토콜이다. 그러므로 이것의 인증을 보장받기 위해서는 Hand shake과정의 일부로 추가해야한다.

4.4.4 man-in-the-middle attack

이러한 공격에 대처하기 위한 방법이 현재의 규격에는 존재하지 않으므로 WTLS를 구현할 경우, 그림20에 있는 class2, class3을 사용하여 an onymous 형태의 키 교환 방식을 서비스하지 않도록 해야한다. 그리고 man-in-the-middle attack을 방지하기 위해서 익명 인증은 최소한 서버측에서 거부되어야 한다.

4.4.5 RSA with Server Certificate attack

서버의 클라이언트 인증서 요청에도 불구하고 클라이언트가 인증서를 보내지 않아 Handshake 프로토콜이 RSA with Server Certificate 과정을 수행하게 된다면 앞에서 언급한 클라이언트 위장

Feature	Class 1	Class 2	Class 3
Public-key exchange	M	M	M
서버 인증서	O	M	M
클라이언트 인증서	O	O	M
Shared-secret handshake	O	O	O
Compression	-	O	O
Encryption	M	M	M
MAC	M	M	M
Smart card interface	-	O	O

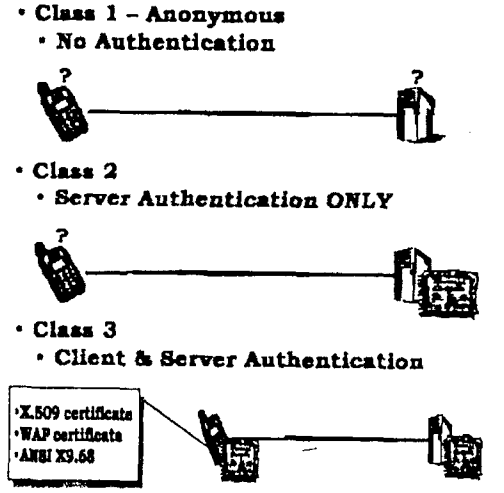


그림 20. WTLS 구현 클래스

공격을 받을 가능성이 존재한다. 따라서 서버가 클라이언트 인증서를 요청한 후 이를 획득하지 못한다면 fatal_alert 메시지와 함께 연결을 종료하도록 구현되는 것이 바람직하다.

4.4.6 Master secret 보호

Secret key를 찾는데 공개 평문이 이용됨으로 Handshake 프로토콜은 한정된 공개 평문을 사용해야 하고, 또 master_secret는 premaster_secret로부터 생성됨으로 premaster_secret의 안전을 위해 클라이언트의 random nonces를 서버의 random nonces와 함께 계산하여 premaster_ secret를 생성하는 것이 바람직하다.

4.4.7 Replay, Delay, Deleted Data attack

MAC이 함께 전송되는 데이터는 항상 필수적

으로 순차번호를 가지도록 함으로써 안전성을 보장할 수 있다. 데이터가 순서 없이 전송되더라도 순차번호를 가지고 수신자는 데이터를 차례로 확인할 수 있기 때문이다. 따라서 데이터가 재전송되었는지, 손실되었는지를 알 수 있다. 이러한 순차번호(Sequence Number)는 64bytes의 크기로 각 connection에서 데이터가 전송될 때마다 개별적으로 유지가 된다.

4.4.8 기타

업데이트가 요구되는 부분으로 현재의 PKCS #1 버전 1.5를 버전 2.0으로 바꾸어 몇 개의 중요한 안전성 문제를 보완하는 것이다. WTLS에서는 인증과 무결성을 제공하기 위하여 RSA가 사용된다. 다른 주요 단점은 블록 암호에서의 Initial Vector이다. 계산에 사용되는 모든 값은 도청자에게 알려진다. 더 높은 안전성을 제공하기 위해서IV-calculations에서는 어떤 유일한 비밀정보가 사용되어야 한다. 그러나, 그 비밀정보는 블록 암호에서 암호키로 사용된 정보와 동일해서는 안된다.

4.5 WTLS의 최신동향

올해 8월 WAP 포럼에서는 WAP 2.0 spec.을 발표하였다. 2.0 버전에서는 이제까지 다루었던 WAP의 보안 취약성에 대해 어느 정도 유동적으로 대처하였다. 본 장에서는 업데이트가 된 주된 내용을 살펴보겠다.

- Diffie-Hellman 알고리즘에서 사용되는 파라미터 들 중 Sub-group의 order, 즉 소인수 q의 값을 정의하였다. 이로써 앞에서 다루었던 소인수에 대한 인증 취약성을 해결하였다.
- Key exchange나 cipher suite가 null일 때 man-in-the-middle 공격에 대한 취약성을 명시하고 있다.
- Spec에서 정의된 타원곡선의 파라미터들 중에 field size가 160비트보다 작은 것은 ECDSA (Elliptic Curve Digital Signature Algorithm)에서 사용될 수 없으므로 ECC basic curve에서 제외되었다. Basic curve라는 것은 모든 서버들이 기본적으로 지원해야 하는 규격을 말한다.
- 40비트 암호는 전탐색 공격에 매우 취약하기때문에 이것들을 사용하지 않기를 권고하고 있다. 또한, 추가로 RC5와 IDEA를 위한 64비트 암호를 제시하였다.
- 소한의 요구조건으로 서버는 암호에 대해서 C5_CBC, RC5_CBC_64, RC5_CBC_56을 지원할 것을, 그리고 해쉬함수에 대해서는 SHA보다는 SHA_80을 지원할 것을 권고하고 있다.
- 해쉬함수의 무결성에 취약점을 가진 SHA_XOR_40 알고리즘은 삭제되었다.
- 마지막으로 RSA 알고리즘이 사용될 경우에는 최소한 1024 모듈러 연산을 지원할 것을 권고하고 있다.

5. 결 론

21세기 디지털 시대의 도래와 함께 무선 인터넷의 보급이 급속하게 확산되고 있다. 이에 따라 무선에서도 유선과 마찬가지로 전자상거래 등을 위해 안전성 확보가 중요한 과제이다.

따라서 본 논문에서는 무선 인터넷 중 현재 가장 큰 관심과 연구의 대상이 되고 있는 WAP의 보안 프로토콜인 WTLS에 대해 분석을 하였다. 그리고 SSL/TLS를 함께 비교하여 각각의 구조와 기능, 동작 등을 살펴보았고, 이것을 기초하여 WTLS의 취약성을 분석하였다.

향후 과제로 WAP 서버와 Gateway 등을 직접 시뮬레이션하여 실제 구현상황에서 어떠한 취약성이 있는지를 살펴볼 필요가 있다. 이를 위해서

서버와 Gateway의 기본 구조와 동작 원리에 대한 연구가 병행되어야 할 것이다.

참 고 문 헌

[1] 무선인터넷백서편찬위원회, "무선인터넷 백 서 2001", (주)소프트뱅크미디어, 2000.9

[2] "Wireless Application Protocol", WAP Forum, 1999, available at http://www.wap- forum.org

[3] D. Wagner, B. Schneier, "Analysis of the SSL 3.0 Protocol", Proceedings of the Second USENIX Workshop on Electronic Commerce, USENIX Press, pp. 2-40, 1996

[4] Marku-Juhani Saarinen, "Attacks Against The WAP WTLS Protocol", Proceedings of Communication & Multimedia Security (CMS'99), available at http://www.cc.jyu.fi- /~mjos/wtls.pdf

[5] 문종철, 원유재, 윤이중, "WTLS handshake 프로토콜의 분석", Proceedings of The 12th Workshop on Information Security and Cryptography(WISC2000), 2000.9

[6] A. Freier, P. Karlton, and P. Kocher, "The SSL Protocol Version 3.0", Internet Draft, work in progress, 1996

[7] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", IETF, RFC2246, 1999.

[8] "Wireless Application Protocol Wireless Transport Layer Security Specification, version1.2", WAP Forum, 1999, available athttp://www.wapforum.org

김 소 진

- 2001년 동명정보대학교 정보통신공학과 졸업(학사)
- 2001년 3월~현재 부경대학교 전자계산학과 석사과정
- 관심분야: WAP, DRM 등
- e-mail: sojin97@hanmail.net

신 성 한

- 2000년 부경대학교 전자계산학과 졸업(학사)
- 2000년 3월~현재 부경대학교 전자계산학과 석사과정
- 관심분야: 정보보호, 암호학
- e-mail: shinsh@mail1.pknu.ac.kr

박 지 환

- 1990년 3월 일본 요코하마국립대학 전자정보공학과 졸업 (공학박사)
- 1994년 9월~1995년 3월 일본 동경대학 생산기술연구소 방문연구
- 1998년 1월~1998년 2월 일본 전기통신대학 방문연구
- 1999년 7월~1999년 8월 Monash University, Austrailia, Visiting Research
- 2001년 2월~2001년 3월 STA Fellowship, Communication Research Laboratory, Japan
- 1996년 4월~현재 동경대학 생산기술연구소 협력연구원
- 1990년 3월~현재 부경대학교 전자컴퓨터정보통신공학부 교수
- 1997년 3월~현재 한국통신학회 부호 및 정보이론 연구회 운영위원
- 1997년 3월~현재 한국통신정보보호학회 이사 및 영남지부 부지부장
- 1998년 12월~현재 한국멀티미디어학회 총무이사
- 1999년 3월~현재 한국정보처리학회 논문지 편집위원
- 관심분야: 멀티미디어 압축 및 응용, 정보보호 및 암호학
- e-mail: jpark@pknu.ac.kr