WTLS Client/Server 설계 및 구현

정성은*, 염희운*
*동덕여자대학교 전자계산학과
e-mail : sejung@cs4000.dongduk.ac.kr

Design and Implementation of WTLS Client/Server

Sungeun Jung*, Heewoon Yum*
*Dept. of Computer Science, Dongduk Women's University

요 약

WAP 에서 보안 계층에 대한 프로토콜인 WTLS 의 주목적은 두 통신 응용간의 privacy, data integrity, authentication 의 제공이다. 본 논문에서는 WTLS 프로토콜을 토대로 WAP 상에서의 안전한 정보 전달을 위한 WTLS Client/Server 시스템의 설계 및 구현에 대하여 논하고자 한다. 구현된 시스템은 어떤 종류의 온라인 네트웍 응용에서도 WTLS 데이터 암호화를 지원하며, 나아가 다양한 무선 인터넷의 응용에 활용되어질 수 있을 것이다.

1. 서론

최근 무선 통신 시장은 매우 빠르게 성장하고 있으며 이를 기반으로 하는 새로운 서비스들이 제공되고 있다. 무선통신 환경의 응용개발을 지원하기 위해 WAP Forum 에서 정의한 표준인 WAP(Wireless Application Protocol)은 transport, security, transaction, session 및 application 계층의 프로토콜 집합을 정의하며, 계층화된 프로토콜을 나타낸다. 이 WAP 의 구조 가운데 보안(security) 계층 프로토콜을 WTLS(Wireless Transport Layer Security)라 한다.

WTLS 계층은 두 통신 응용간의 기밀성, 무결성, 인증의 제공을 그 목적으로 한다. WTLS 는 TLS 와 유사한 기능을 가지며, 이에 데이터그램의 지원이나, 최적화한 handshake, 동적 키 refresh 같은 새로운 기능이 추가되었다. WTLS 프로토콜은 상대적으로 긴 잠복기간을 갖는 낮은 대역폭의 bearer 통신망에서 최적화 된다.

본 논문에서는 WTLS 프로토콜을 토대로 WAP 상에서의 안전한 정보 전달을 위한 WTLS Client/Server 시스템의 설계 및 구현에 대하여 논하고자 한다. 본 논문에서 설계된 클라이언트 및 서버는 WTLS 에서의 handshake 프로토콜을 실행하고 안전한 정보의 교환을 위한 보안 서비스를 제공한다. 구현된 WTLS 클라이언트 서버의 구조는 그림 1 과 같다.

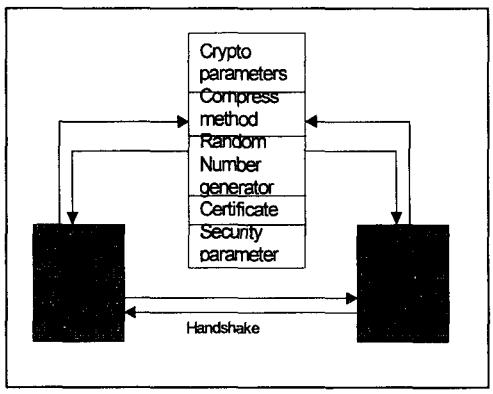

그림 1. WTLS Client/Server 구조

2. WTLS Client/Server Handshake protocol

WTLS 의 Record 계층의 최상위에서 운영되는 handshake 프로토콜(그림 2)은 Secure session 의 cryptographic parameter 들을 제공한다.

WTLS Client/Server Handshake Protocol 은 아래 단계들을 포함한다.:

- 알고리즘 협의와 난수 값 교환을 위한 hello message 교환.

- pre-master secret 상에서 클라이언트와 서버가 협의하는데 필요한 암호화 파라메터들의 교환.
- 클라이언트와 서버가 서로를 인증하기 위한 인증서와 암호화 정보의 교환
- pre-master secret 으로 부터 master secret 을 생성하고 난수 값을 교환.
- 레코드 계층에 보안 파라메터들을 제공
- 클라이언트와 서버가 서로 같은 보안 파라메터들을 계산하고, handshake 가 공격자들의 간섭 없이 안전하게 이루어질 수 있도록 한다.

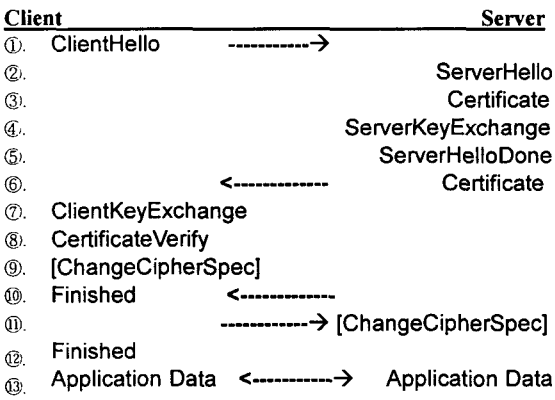

그림 2: Client/Server Handshake Protocol

3. WTLS client/Server System

WTLS client/Server 는 구현되는 시스템에서 제공하는 기존의 네트웍 연결을 사용하며, WAP 스택의 경우에는 WDP 를 의미한다. 즉, connectionless 환경에서 사용되도록 설계되었다.

그림 2 에서 보여지는 WTLS handshake 에서, client 는 handshake 를 초기화할 수 있다. System 은 최초의 handshake 데이터그램을 생성하고, 전송을 위해 그 것을 WDP 계층에 나타낸다. 그리고 나서 이 세션은 응답을 생성하고, 클라이언트에게 응답하기 위해, 데이터그램을 WDP 계층에 나타낸다. handshake 가 모두 이루어진 후에, 사용자는 동배간의 인증서, 선택된 암호화 알고리즘 등, 보안 세부사항에 대한 session 을 query 할 수 있다. 그리고 나서 데이터는 원격 호스트와 안전하게 통신할 수 있다.

TLS 의 경우와는 달리 WTLS 의 데이터 그램 환경하에서는 handshake 메시지는 분실, 중복, 손실 등의 우려가 있다. 이런 환경에서 신용 있는 handshake 를 구성하려면, handshake 메시지는 단 방향 이어야 한다.

4. Client/Server 시스템의 구현

그림 2 에서 나타나는 Client/Server 의 handshake protocol 을 구현하는 단계별 구현은 다음과 같다.

① Hello Messages

client hello A → B : A,Na, Sid, Pa
server hello B → A: Nb, Sid, Pb

여기서 Na, Nb 는 A 와 B 에 대한 random number 를 의미하며, Sid 는 session identifier 를, Pa, Pb 는 A 와 B 의 각각에 대한 암호화 선택 사항들을 나타낸다. 이 hello 메시지는 클라이언트와 서버간에 사용되는 보안 파라메터들을 협의 하는데 사용되어 지며, 이에 대한 요청은 클라이언트가 클라이언트 hello 메세지를 전송하는 것으로 협의를 시작한다는 간단한 통지이다.

- Client Hello

클라이언트가 처음 서버에 연결될 때 클라이언트 hello 메세지의 전송이 최초로 요구된다. 클라이언트는 또한 요청에 대한 응답으로 클라이언트 hello 를 전송할 수 있으며, 현재의 secure connection 내의 security parameter 를 위해 자신을 초기화할 수 있다.

키 교환 목록은 클라이언트가 지원하는 암호화 키 교환 알고리즘을 포함한다. 또한, 각각의 entry 는 인증서 또는 클라이언트가 사용하고자 하는 공개 키를 정의하며, 서버는 이 중 하나를 선택하거나, handshake_failure alert 을 전송하고 secure connection 을 종료할 수 있다.

클라이언트가 서버에게 보낸 클라이언트 hello 메세지 내에 포함된 CipherSuite 목록은 동기 암호화 알고리즘(symmetric cryptographic algorithm)을 포함한다. 개 개의 CipherSuit 은 다양한 암호화 알고리즘 및 MAC 알고리즘을 정의한다. 서버는 암호화 알고리즘을 선택하거나 또는 alert 을 발생한다.

- Server Hello

서버는 적절한 알고리즘 set 을 발견하면 클라이언트의 hello 메시지에 대한 응답으로 이 메세지를 전송한다. 만약 그렇지 못할 경우 handshake_failure alert 을 응답으로 보낸다.

② Server Certificate

server certificate B → A: certificate(B,Kb)

Kb 는 인증서와 함께 전송된 서버 B 의 공개 키를 나타내며 이 메시지는 항상 server hello 메세지에 선행한다. 인증서 형식은 최적화된 WTLS 인증서 크기를 제공하는 X.509v3 인증서 또는, X9.68 인증서를 사용한다. 이는 반드시 키 교환 알고리즘에 적합한 키를 포함해야 한다. 새로운 키 교환 방법을

명기하는 KeyExchangeSuite 는 인증서 포맷을 제공하고 encode 된 키 정보를 포함하는 WTLS 프로토콜에 대해서도 명기한다.

메시지의 구조는 아래와 같다.:

```
enum { WTLSCert(1), X509Cert(2), X968Cert(3),
(255) } CertificateFormat;
struct {
  select (PublicKeyType) {
      case ecdh: ECPublicKey;
      case ecdsa: ECPublicKey;
      case rsa: RSAPublicKey;
      } PublicKey;
struct {
    uint8 certificate_version;
    SignatureAlgorithm signature_algorithm;
    Identifier issuer;
    uint32 valid_not_before;
    uint32 valid_not_after;
    Identifier subject;
    PublicKeyType public_key_type;
    ParameterSpecifier parameter_specifier;
    PublicKey public_key;
} ToBeSignedCertificate;
```

여기서 해쉬 값과 서명은 CertificateSignatureAlgorithm 을 사용하는 ToBeSignedCertificate 에 의해 계산된다. 서버 인증서는 CA 공개키에 의해 인증되는 인증서 하나만을 가질 수 있다. 클라이언트 인증서 체인은 여러 개의 인증서를 포함한다. 또한 서버는 인증서 분산 서비스로 부터 클라이언트 인증서를 획득한다. 인증서 체인 내에서 모든 인증서는 선택된 키 교환 알고리즘에 적절한 알고리즘을 사용해야만 한다.(즉, 예를 들어 RSA 에 대해 인증서는 RSA 가 서명한 RSA 키를 갖는다.)

③. *Server Key Exchange Message*

서버 키 교환 메시지(Server Key Exchange Message)는 서버인증서 메시지 전송 후 바로 전송된다. 서버의 인증서 메시지가 클라이언트가 pre-master secret 을 교환할 충분한 데이터를 포함하지 않을 경우에만 서버는 서버 키 교환 메시지를 전송하며, ECDH_anon, RSA_anon, DH_anon 의 경우에만 허용되며, ECDH_ECDSA (고정 파라메터) 및 RSA 알고리즘에 대해서는 전송되어질 수 없다.

이 메시지는 클라이언트가 암호화 하기위한 RSA 공개키 뿐 아니라, 클라이언트가 키 교환을 완성하도록 하는 ECDH 파라메터를 포함하는 pre-master secret 과 통신하도록 해주는 암호화 정보를 전달한다.

④. Certificate Request

서버는 선택된 암호 알고리즘이 적절하다면, 클라이언트에게 인증서를 요청할 수 있다. 이 메시지가 전송되면, 서버는 즉시 서버인증서 메시지와 서버 키 교환 메시지를 전송할 수 있다.

⑤. Hello Done

server hello done 메시지는 server hello 의 마지막을 나타내고 메시지와 연관된 서버에 의해 전송되어진다. 메시지 전송 후에 서버는 클라이언트의 응답을 기다린다.

이 메시지는 서버가 키 교환을 지원하는 메시지의 전송을 완료 했음과, 클라이언트가 키 교환 구문을 처리할 수 있음을 나타낸다.

⑥. Client Certificate

client certificate $A \rightarrow B$: certificate(A,Ka)
client key exchange $A \rightarrow B : \{PMS\}_{Kb}$
certificate verify $A \rightarrow B: \{Hash\{Nb,B,PMS\}\}_{Ka-1}$

$\{X\}_K$ 는 K 로 암호화되고 서명 된 메시지 X 를 나타내며, PMS 는 평문을 나타낸다. Client certificate 와 certificate verify 는 WTLS 의 최적화된 handshake 시에는 생략될 수 있다.

이 메세지는 서버의 hello done 메세지 수신 후 클라이언트가 전송할 수 있으며, 서버가 인증서를 요청한 경우에만 전송된다 클라이언트 인증서는 서버 인증서에서 정의된 인증서 구조를 사용해 전송되어진다. 만약 서버가 클라이언트 인증서를 알고 있다면, 이 메세지는 인증서를 포함하지 않는다.

⑦. Client Key Exchange Message

클라이언트 인증서 메시지 바로 뒤이어 전송되거나, 서버의 hello done 메세지를 수신한 후 클라이언트에 의해 우선적으로 전송된다. 여기서 pre-master secret 이 설정되고, RSA-encrypted secret 의 직접적 전송뿐 아니라, 키 교환 방법에 따라 서버의 인증서와 부합되는 파라메터를 갖는 인증서와 함께 응답할 수 있다.

메시지의 구조는 아래와 같으며, 선택된 키 교환 알고리즘에 종속적이다.

```
struct {
  select (KeyExchangeAlgorithm) {
    case rsa: RSAEncryptedSecret param;
    case rsa_anon: RSAEncryptedSecret param;
    case dh_anon: DHPublicKey param;
          /* client public value*/
    case ecdh_anon: ECPublicKey param;
          /* client public value */
    case ecdh_ecdsa: ECPublicKey param;
          /* client public value */
    } exchange_keys;
} ClientKeyExchange;
```

예를 들어 만약 RSA가 키 agreement 와 authentication에 사용 되어 진다면, 클라이언트는 20 바이트 secret을 생성하고, 서버 인증서로부터의 공개 키를 사용하여 암호화 하며, 암호화된 secret 메시지를 결과로 전송하게 된다.

⑧. Certificate Verify

인증서 증명 메시지(Certificate Verify Message)는 클라이언트 인증서를 확인하며, 서명 가능한 인증서에 명기된 클라이언트에 의해 전송되며, 전송 후 즉시 클라이언트 키 교환 메시지가 수행된다.

⑨. Finished

client finished $A \rightarrow B : \{Finisned\}_{CLIENTK(Na,Nb,M)}$
server finished $A \rightarrow B : \{Finisned\}_{SERVERK(Na,Nb,M)}$

클라이언트와 서버 양자는 PMS, Na, Nb 로 부터 master-secret 인 M 을 연산하고, Sid, M, Na, Pa, A, Nb, Pb, B 의 해쉬로 부터 *Finished* 를 연산한다. 이 Finished 메시지는 항상 handshake 의 마지막에 키 교환 및 인증이 성공적으로 수행 되었음을 확인하고 정확하다는 것이 검증되면 자신의 종료 메세지를 보내고, 상대방의 종료 메세지를 수신하고 검증되면, secure connection 상에서 application data 를 송수신한다.

만약 종료 메시지가 handshake 내에서 적절한 시점에서 change cipher spec 에 의해 처리 되지 않는다면 critical 또는 fatal error 를 발생한다.

5. 결론 및 향후 연구 계획

본 논문에서 구현된 클라이언트/서버 시스템은 WTLS 프로토콜을 토대로 WAP 상에서 클라이언트와 서버 양자 간의 기밀성, 무결성, 인증 을 제공하도록 설계되고 구현 되었다. 또한 구현된 클라이언트 및 서버는 WTLS 에서의 handshake 프로토콜을 실행하고 안전한 정보의 교환을 위한 보안 서비스를 제공한다 본 시스템에서는 유선 인터넷 망에서 지원되는 기존의 TLS 의 기능 이외에 데이터그램의 지원 및 최적화(optimised) handshake, 동적 키 refresh 같은 새로운 기능들이 추가되었다.

구현된 WTLS 클라이언트/서버 시스템은 TCP/IP 네트워크의 UDP 를 기반으로 구현되었다.

앞으로 WTLS 클라이언트/서버 시스템을 WAP 프로토콜 스택에 포팅하고 단말기에 포팅하는 작업이 필요하다. 포팅된 시스템은 다양한 무선 인터넷 응용에서 보안 서비스를 제공하는데 사용될 수 있을 것이다.

참고문헌

[1] WAP Architecture Specification, WAP Forum, 30-April-1998.
URL: http://www.wapforum.org/

[2] WTLS : Wireless Transport Layer Security Specification V5, WAP Forum, Nov-1999.
URL: http://www.wapforum.org/

[3] WPKI : Wireless Public Key Infrastructure, draft, WAP Forum, Mar-2000.
URL: http://www.wapforum.org/

[4] Lawrence C. Palson. "Inductive Analysis of the Internet Protocol TLS", ACM Transaction on Information and System Security, Vol.2, No.3, Aug. 1999

[5] Wagner,D. and Schneier, B. "Analysis of the SSL 3.0 protocol : In proceedings of the USENIX Conference on Electronic Commerce." USENIX Assoc, Berkely, CA, 29-40, 1996.