제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

패스워드 기반 EAP-TLS

이석준, 양대헌, 정병호, 정교일
한국전자통신연구원 정보보호연구본부
e-mail : {junny, nyang, cbh, kyoil}@etri.re.kr

New EAP-TLS based on Password Authentication

Sokjoon Lee, DaeHun Nyang, ByungHo Chung, Kyoil Chung
Information Security Research Division, ETRI

요 약

EAP(Extensible Authentication Protocol)[3]은 다양한 인증 방법을 제공하기 위한 표준 인증 메커니즘이다. EAP 는 PPP[16], 802.1x[17] 등에서 사용되며, 실제 인증 능력을 가지는 TLS[2] 등과 같은 인증 프로토콜과 결합하여 사용된다.

TLS(Transport Layer Security)는 두 peer 간에 전송계층에서 상호 인증, 무결성, 기밀성을 제공하기 위해 개발되었다. TLS 는 상호 인증을 위하여 공개키 기반 인증서를 사용한다. 그러나, 인증서를 사용하는 것은 대부분의 사용자들이 ID, password 기반의 응용에 익숙하다는 것을 생각하면 일반적인 인증 방법이 아님을 알 수 있다. 따라서, EAP-TLS 와 같은 인증 방법 역시 그런 면에서 PPP 혹은 802.1x 에서 사용하기에 부적합하다고 볼 수 있다.

본 논문에서는 패스워드 기반 인증 및 키교환 프레임웍을 TLS 에 적용하고 이를 다시 EAP 와 결합한다. 패스워드 기반 EAP-TLS 는 인증서에 관련된 복잡한 연산을 수행하지 않으면서도 안전성에 있어 수학적 증명을 가지며, PPP 혹은 802.1x 서비스를 받고자 하는 사용자가 인증서를 가질 필요가 없는 장점을 가지게 된다.

1. 서론

EAP 는 다양한 인증 방법을 제공하는 표준 메커니즘이다. EAP 는 MD5, smart card, TLS 등과 같이 실질적으로 인증을 수행하는 프로토콜과 함께 결합하여 PPP, 802.1x 와 같은 환경에서 사용된다. EAP 의 장점은 새로운 인증 프로토콜을 추가하더라도 NAS(Network Access Server)를 고칠 필요가 없다는 데 있다.

인증 프로토콜로 TLS 를 사용하는 EAP-TLS[14]에서 클라이언트와 서버 사이의 인증은 공개키 기반의 인증서를 통해 이루어진다. 공개키에 기반한 PKI 는 다양한 정보보호 서비스를 제공하는데 유용하다고 알려져 있지만, 인증서 관련 연산의 무거움으로 인하여 일반적인 응용 프로그램에 적용하여 사용하는 것은 쉽지 않다. Telnet, FTP 등을 비롯한 많은 응용 프로그램에서는 사용자 인증을 위해 username 과 패스워드를 이용하고 있기 때문에, 많은 user 들은 대부분 인증서를 사용하는 것보다 패스워드를 직접 타이핑하는 것에 더욱 익숙하다. 이는 PPP 서비스를 통해 인터넷 연결을 하거나, 802.1x 를 통한 무선랜 공중망 서비스를 받고자 하는 사용자에게도 예외는 아닐 것이다.

그러나, 일반적으로 패스워드 자체의 랜덤성의 부족으로 username 과 패스워드에 의한 인증 방법은 위험하다고 알려져 있다. 패스워드를 사용한 Challenge and Response 방식의 프로토콜 역시 오프라인 사전 공격에 취약하다고 알려져 있다. 그러므로 패스워드를 사용한 인증 방식을 설계하는 경우, 패스워드의 취약성에 대한 고려가 있어야 한다.

오프라인 사전 공격에 강한 패스워드 기반의 인증 및 키 교환 프로토콜(Bellovin and Merrit' s EKE[19])이 1992 년에 소개된 이후로, 이 분야에 있어 많은 연구가 있어 왔다(Jablon ' s SPEKE[7], Wu ' s SRP[4] 등). EKE 와 SPEKE 가 패스워드 확인자 모델로 확장되었는데, 이들은 각각 A-EKE, B-SPEKE 이다[8][9]. 2000 년 Bellare 등에 의해 패스워드 기반의 인증 프로토콜에 대한 수학적 접근이 이루어지고[13], 이를 계기로 수학적 증명을 가지는 프로토콜들이 개발되고 있다. 그러나, 이들 프로토콜은 여전히 계산량 등의 측면에 있어 그다지 효율적이지 않다.

최근 secret coin tossing 이라는 개념이 적용된 영지식 증명을 이용하여 설계된 패스워드 기반의 인증 및 키 교환 프레임웍이 발표되었다[1]. 이 프레임웍은 단지 보안 강도에 대한 수학적 증명 뿐 아니라 효율성까지 가지고 있으며, 다른 인증 프로토콜에 쉽게 적용할 수 있다는 장점도 가지고 있다.

본 논문에서는 이 프레임웍을 이용한 새로운 패스워드 기반의 TLS 프로토콜과 EAP-TLS 프로토콜을 설계한다. 이 프로토콜들은 공개키 인증서를 교환하고 처리할 필요가 없으므로 매우 가벼운 프로토콜이 된다.

2. 패스워드 기반 인증 및 키 교환 프레임웍

이 장에서는 [1]에서 정의하고 있는 프로토콜에 대해 간략히 살펴본다. 이 프로토콜에 대한 자세한 내용 및 증명은 [1]을 참조하기 바란다.

제안하는 프레임웍은 크게 [1]의 변형된 EKE 와 secret coin tossing 을 하는 대화식 영지식 증명의 두 부분으로 이루어져있다. 여기서 쓰이는 H()와 h()는 모두 일방향 해쉬함수이며, 특히 h()는 해쉬값의 상위 k 비트만을 돌려준다 (k 는 계산효율과 보안강도의 trade-off 의 정도가 된다).

2.1 시스템 설정

사용자의 공개키 또는 패스워드 확인자 V = OWF(f(S))는 서버에 비밀리에 보관된다.

2.2 인증 절차

a. 변형된 EKE 를 수행한다.
   a.1 사용자는 x 를 임의로 선택해서 시험수 A=OWF(r)과 함께 ID_{User}, $X=V(g^x)$를 서버에 전송한다. 여기서 $x \in G$.
   a.2 서버는 $y \in G$ 를 이용해서 다음을 계산하고 (auth = H(K' ‖ 1), Y)를 사용자에게 전송한다.

$K = [V^{-1}(X)]^y$, $Y = V(g^y)$
$K' = H(K \| g^x \| g^y \| ID_{User} \| ID_{Server})$
a.3 사용자는 다음을 계산해서 auth = H(K' ‖ 1)인지 확인한다. 맞는다면 서버가 V 를 알고있다고 확신할 수 있다.
$K = [V^{-1}(Y)]^x$
$K' = H(K \| g^x \| g^y \| ID_{User} \| ID_{Server})$
a.4 이 변형된 EKE 가 성공적으로 끝나면, 각각은 공유하는 비밀키 TSK=H(K'‖0). 아니라면 프로토콜 수행을 중단한다.

b. 사용자는 목격자수 B 를 c, r 그리고 자신이 가지고 있는 비밀수 S 를 이용해서 서버에게 보낸다. 여기서 c = h(TSK ‖ A)이고 c 의 길이는 안전강도와 계산효율성의 trade-off 파라미터가 된다.

c. 서버는 c = h(TSK ‖ A)를 계산하고 사용자의 목격자수를 A, V, c 를 이용해서 검증한다. 성공하면 사용자 인증이 완료된다.

d. 프로토콜 종료 후, 교환된 키는 다음과 같다.
SK = H(K'‖A‖B‖2)

여기서 시험수 A 가 영지식 증명 부분이 아닌 변형된 EKE 부분에서 전송됨을 주목해 보면, 사용자가 시험수를 결정할 당시에는 질문수 를 전혀 예측할 수 없음을 알 수 있다. 이 불예측성은 서버가 랜덤하게 질문수를 선택해서 사용자에게 전송하는 것과 같은 효과를 가진다.

이 프로토콜 프레임웍은 [1]에 언급된대로 여러 가지 대화식 영지식 증명에 기초한 인증 프로토콜을 패스워드에 기초한 인증 및 키 교환 프로토콜로 변환할 수 있도록 해준다. 따라서 위의 프레임웍에 Guillous-Quisquater, Feige-Fiat-Shamir, Schnorr 등의 프로토콜[10, 11, 12]을 적용함으로써 패스워드 인증 프로토콜을 구현할 수 있다.

3. 패스워드 기반 TLS

이 장에서는 user 와 서버의 상호 인증 및 키 교환을 하기 위해 인증서가 아닌 패스워드를 이용한 새로운 TLS 를 소개한다. 이와 비슷한 연구로 TLS 에 SRP 인증 방법을 사용한 방법[4][5][6]이 IETF 인터넷 드래프트로 올라와 있다.

앞장에서 설명한 프레임웍은 그 자체가 프로토콜은 아니며, 영지식 증명 프로토콜도 다소 추상적이다. 따라서 이 프레임웍을 직접 TLS 에 적용하는 것은 무리가 있으며, 이를 해결하기 위해 Guillous-QuisQuater protocol[10], Feige-Fiat-Shamir[11], Schnorr' s scheme[12] 등과 같은 잘 알려진 영지식 증명 프로토콜을 사용한다. Nyang 은 [1]의 full paper 에서 Guillous-QuisQuater 프로토콜에 프레임웍을 적용했으나, 본 논문에서는 생략하고 이를 TLS 에 적용한 방법만을 소개하기로 한다. 이를 위해, TLS 를 다소 수정해야 하며, 수정된 TLS 를 PTLS(Password-based TLS)라 부르기로 한다

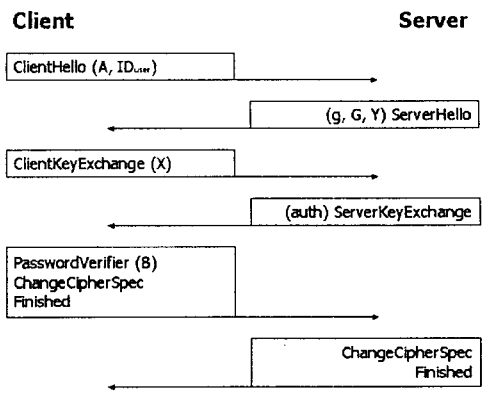
3.1 PTLS 시스템 설정

수정된 프로토콜이 정상적으로 동작할 수 있도록 클라이언트가 서버로부터 패스워드를 발급받고 나면 다음과 같은 상황이 이루어짐을 가정한다. 단, 패스워드를 발급받는 절차에 대해서는 이 논문에서 다루지 않는다.

- 클라이언트는 패스워드 S 를 발급받으면 이를 로컬에 저장하지 않으며, 로컬에는 ID_{User}와 e, n, f()을 저장하고 있다. 클라이언트는 서버와 보안 세션을 맺으려 할 때, 패스워드 S 를 입력하는 절차를 거친다.
- 서버는 클라이언트가 만든 패스워드 S 를 알지 못하며, ID_{User}, e, n, V(= $f(S)^e \bmod n$)를 저장한다.
- n 은 RSA modulus 이며, e 는 2 가 아닌 적당히 작은 홀수이다. 클라이언트가 저장하는 f()는 패스워드의 길이를 적당히 늘이는 어떤 함수이다.

3.2 PTLS 핸드쉐이크 메시지 순서

PTLS 프로토콜은 TLS 규격에 정의된 핸드쉐이크 메시지 순서를 그대로 사용할 수 없다. 여기서는 PTLS 프로토콜에서 사용할 새로운 핸드쉐이크 메시지 순서를 (그림 1)에 제시하고자 한다. 단, Full 핸드쉐이크에서만 언급하며, 보안 세션 재사용을 위한 Abbreivated 핸드쉐이크의 메시지 순서는 기존의 TLS 와 같으므로 생략한다.



(그림 1) PTLS 핸드쉐이크 메시지 순서

3.2.1 C→S : ClientHello

클라이언트는 랜덤한 수 $r \in Z^*_n$ 을 택하여 $A = r^e \bmod n$ 을 계산한다. 그런 후 ClientHello 를 보내며, ClientHello 메시지를 확장하여 A 와 ID_{User}를 추가적으로 보낸다.

3.2.2 S→C : ServerHello

서버는 클라이언트가 보낸 ClientHello 메시지를 통해 해쉬알고리즘, 대칭키알고리즘 등을 선택한다. 그리고 어떤 그룹 G 와 그에 대한 generator g 를 택하고, 랜덤한 수 $y \in G$ 를 택해서 $Y = V(g^y)$를 계산하여 클라이언트에게 ServerHello 와 함께 보낸다. 단 V(a)는 V 를 대칭키로 활용하여, ServerHello 에서 선택한 대칭키 암호 알고리즘으로 a 를 암호화하는 것을 뜻한다.

3.2.3 C→S : ClientKeyExchange

클라이언트는 랜덤한 수 $x \in G$ 를 선택하여 $X = V(gx)$를 계산하고 ClientKeyExchange 메시지에 실어 보낸다. 그 후 다음을 계산한다.
$K = [V\text{-}1(Y)]^x$, $K' = H(K \| g^x \| g^y \| ID_{User} \| ID_{Server})$

3.2.4 S→C : ServerKeyExchange

서버는 받은 메시지를 바탕으로 다음을 계산한다.
$K = [V\text{-}1(X)]^y$, $K' = H(K \| g^x \| g^y \| ID_{User} \| ID_{Server})$
서버는 auth = H (K' ‖ 1)을 계산하여 ServerKeyExchange 메시지에 실어 보낸다. 단, H()는 ServerHello 에서 결정된 Hash 알고리즘을 사용한다.

3.2.5 C→S : PasswordVerifier, ChangeCipherSpec, Finished

클라이언트는 H(K' ‖ 1) = auth 인지를 검증한다. 이것이 같지 않으면 서버 인증에 실패하고 종료한다. 같을 경우 클라이언트와 서버는 각각 TSK = H(K'‖0), c = H(TSK‖A)를 계산한다.

이미 처음에 택했던 r, 방금 계산이 된 c, 그리고 사용자의 패스워드 S 를 이용하여 $B = r * f(S)^e$ 를 계산하여 PasswordVerifier 메시지를 만들고, ChangeCipherSpec, Finished 메시지를 만들어 보낸다. 세션키를 만들어내기 위해 필요한 premaster secret 은 SK = H(K' ‖ A ‖ B ‖ 2)로 하며, 이를 이용하여 master secret, key block 을 생성하고 Finished 메시지를 암호화한다.

3.2.6 S→C : ChangeCipherSpec, Finished

서버는 받은 메시지를 바탕으로 다음을 검증한다.
$A = B^e * V^c \bmod n$
이 식이 같으면 서버는 클라이언트 인증에 성공하게 되고 ChangeCipherSpec, Finished 메시지를 만들어 보낸다. 클라이언트는 Finished 메시지를 검증해보고 성공하면 핸드쉐이크를 종료하여 secure session 을 통한 암호화된 통신을 진행한다.

이 식이 같지 않으면 서버는 클라이언트 인증에 실패하게 되고 Alert 메시지를 만들어 보내 연결을 끊는다.

3.3 수정된/추가된 핸드쉐이크 메시지

기존의 TLS 핸드쉐이크 메시지로는 패스워드를 이용한 인증에 필요한 정보들을 주고 받는 것이 불가능하므로 몇몇 메시지의 내용을 수정하거나 추가하도록 한다. 다음은 각 메시지별로 수정 혹은 추가될 내용을 담고 있으며, 각 notation 은 TLS 규격에서 제시된 notation 표현방식과 동일하다.

3.3.1 ClientHello, ServerHello

ClientHello 와 ServerHello 에는 추가되는 내용 중 A, g, G, Y 를 위해 다음의 필드가 추가되도록 한다.

```
struct {
   ProtocolVersion client_version;
   Random random;
   SessionID session_id;
   CipherSuite cipher_suites<2..2^16-1>;
   CompressionMethod compression_methods<1..2^8-1>;
   PTLSExtension PTLS_extension;
} ClientHello;

struct {
   ProtocolVersion server_version;
   Random random;
   SessionID session_id;
   CipherSuite cipher_suite;
   CompressionMethod compression_method;
   PTLSExtension PTLS_extension;
} ServerHello;

enum {client, server} ClientOrServerExtension;

struct {
   select (ClientOrServerExtension) {
      case client :
         opaque PTLS_A<1..2^8-1>;
         opaque PTLS_A<1..2^8-1>;
      case server :
         opaque PTLS_g<1..2^16-1>;
         opaque PTLS_G<1..2^16-1>;
         opaque PTLS_Y<1..2^8-1>
   }
} PTLSExtension
```

3.3.2 ClientKeyExchange

ClientKeyExchange 메시지는 X 의 전달을 위해 다음과 같이 구성되도록 한다.

```
enum {rsa, diffie_hellman, PTLS}
KeyExchangeAlgorithm;

struct {
   select (KeyExchangeAlgorithm) {
      case rsa: EncryptedPreMasterSecret;
      case diffie_hellman:
         ClientDiffieHellmanPublic;
      case PTLS: PTLSClientKey;
   } exchange_keys;
} ClientKeyExchange;

struct {
   opaque PTLS_X<1..2^16-1>;
} PTLSClientKey;
```

3.3.3 ServerKeyExchange

auth 의 전달을 위해 다음과 같이 구성한다.

```
struct {
   select (KeyExchangeAlgorithm) {
      case diffie_hellman:
         ServerDHParams params;
         Signature signed_params;
      case rsa:
         ServerRSAParams params;
         Signature signed_params;
      Case PTLS:
         PTLSServerParams params;
   }
} ServerKeyExchange;

struct {
   opaque PTLS_auth<1..2^8-1>;
} PTLSServerParams;
```

3.3.4 PasswordVerify

핸드쉐이크 메시지 종류에 password_verify(17)을 추가하고 새로운 핸드쉐이크 메시지를 정의한다.

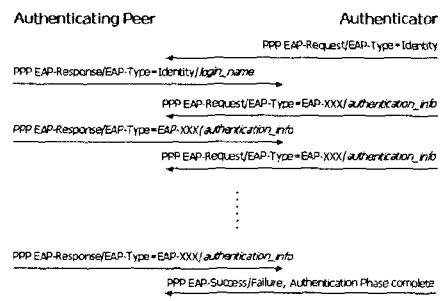
```
enum {
   hello_request(0), client_hello(1),
   server_hello(2), certificate(11),
   server_key_exchange(12), certificate_request(13),
   server_hello_done(14), certificate_verify(15),
   Client_key_exchange(16), password_verifier(17),
   finished(20), (255)
} HandshakeType;

struct {
   HandshakeType msg_type;
   uint24 length;
   select (HandshakeType) {
      case hello_request: HelloRequest;
      case client_hello: ClientHello;
      case server_hello: ServerHello;
      case certificate: Certificates;
      case server_key_exchange: ServerKeyExchange;
      case certificate_request: CertificateRequest;
      case server_hello_done: ServerHelloDone;
      case certificate_verify: CertificateVerify;
      case client_key_exchange: ClientKeyExchange;
      case password_verifier: PasswordVerifier;
      case finished: Finished;
   } body;
} Handshake;

struct {
   opaque PTLS_B<0..2^8-1>;
} PasswordVerifier;
```

4. EAP-PTLS

EAP 는 앞서 말한 바와 같이 NAS 의 변경없이 다양한 인증 프로토콜을 지원하기 위한 매커니즘이며, 원래 PPP 서비스에서의 인증을 위해 만들어 졌으나, 현재 802.1x 에서도 표준 인증 프로토콜로 등록되어 있다. EAP 인증 흐름은 (그림 3)과 같으며 EAP 프로토콜의 자세한 내용은 [3]을 참조하도록 한다.



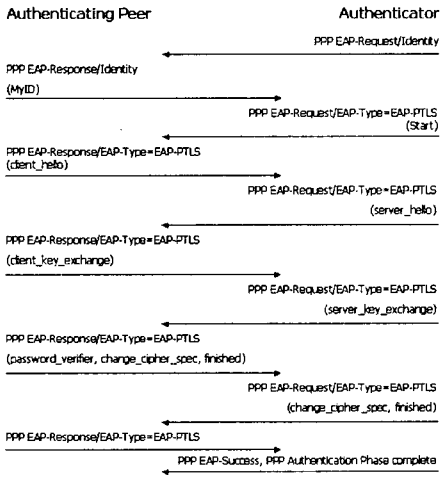
(그림 2) 일반적인 EAP 인증 흐름도

4.1 EAP-PTLS

PPP 나 802.1x 에서는 네트웍 서비스를 제공하기 위하여 사용자를 인증할 수 있어야 하며, 특히 무선랜 서비스와 결합된 802.1x 에서, Link 계층 보안을 위한 세션키 교환은 필수적이다. EAP-MD5 를 비롯한 일부 EAP 프로토콜에서는 사용자와 인증 서버사이에 세션키를 교환하는 것이 불가능하다.

EAP-TLS[14]에서는 세션키 교환은 가능하지만 복잡한 인증서 연산을 필요로 하며, EAP-TTLS[15]에서는 서버 인증 후 이루어진 보안 세션을 통해 서버에게 사용자가 패스워드를 전달하는 형식을 취하고 있다. 그러나, EAP-TTLS 에서도 여전히 서버 인증서를 처리하는 모듈은 필요하다.

3 장 설계된 PTLS 는 클라이언트와 서버 사이에 패스워드 기반의 상호 인증 및 키 교환 방법을 제공했다. PTLS 환경에서는 user 와 인증 서버 모두 인증서 관련 연산을 수행할 필요가 없다. 이러한 장점을 바탕으로 우리는 PTLS 를 EAP 에 적용하고자 한다. 그 결과는 (그림 4)와 같다.



(그림 3) EAP-PTLS 인증 흐름도

4.2 다른 EAP 프로토콜과의 비교

EAP 종류	인증토큰	사용자 인증 방법	세션키 생성 여부	기타
EAP-MD5	패스워드	MD5	No	단방향 인증
EAP-TLS	인증서	PKI	Yes	PKI 연산 필요
EAP-TTLS [15]	패스워드 인증서	PAP, CHAP, MSCHAP	Yes	PKI 연산 필요
EAP-SRP [18]	패스워드	패스워드 확인자	Yes	드래프트 논의 중단중
EAP-PTLS	패스워드	패스워드 확인자	Yes	안전성 증명 가능

(표 1) 다른 EAP 프로토콜과의 비교

5. 안정성 증명

5.1 보안성 분석

PTLS 는 기본적으로 패스워드라는 약한 키를 사용하면서도 오프라인 사전 공격에 강하다. 이는 공격을 하기 위해 필요한 c 값이 네트워크상으로 전송되는 것이 아니라, 사전에 약속한 규칙에 의해서 따로 생성하게 되며, 이를 위해 필요한 g^x 와 g^y 가 V 에 의해서 암호화되어 전송되기 때문이다. 그러므로 전송되는 값만을 가지고는 예측한 값이 S 와 같은지를 공격자는 전혀 알 수 없다.

또한, 어떤 active 공격자가 적법한 클라이언트로 위장을 하는 경우에는 f(S)를 계산할 수 없으며, 적법한 서버로 위장을 하더라도 V 를 알지 못하므로 상대방과 같은 K 를 구할 수 없어 공격에 성공할 수 없다.

그리고 서버가 각 클라이언트에 대한 V 값을 저장해 놓은 디렉토리를 공격자가 해킹하여 알아내더라도 이를 바탕으로 S 를 추측하거나 적법한 사용자로 위장하는 것은 불가능하다. $V = f(S)^e \bmod n$ 인데 V 와 e, n 를 안다고 해서 f(S)를 구할 수 없기 때문이다.

PTLS 는 기본적으로 [1]에서 사용한 프레임웍을 바탕으로 설계되었으며, 이 프레임웍의 안전성에 대한 수학적인 증명은 [1]에 있으므로 생략한다.

5.2 효율성 분석

PTLS 와 EAP-PTLS 는 TLS, EAP-TLS 에 비해 핸드쉐이크 메시지를 주고받는 횟수가 1 번 더 있다. 그러나, PTLS 에서 인증서 관련 연산이 없음을 고려한다면 PTLS 가 TLS 에 비하여 더 비효율적이다라고 할 수는 없다.

핸드쉐이크 도중 필요한 연산에 있어서 가장 무거운 공개키 연산만을 비교한다면, 클라이언트의 경우 PTLS 는 DH 공개키 연산 1 번, RSA 공개키 연산 수준의 지수 연산 2 번이 있으며, TLS 는 클라이언트 인증을 하는 class 3 핸드쉐이크의 경우 서버인증서 검증시 전자서명 검증 1 회 이상, 키 교환시 1 회, 클라이언트 인증서 검증 메시지 생성시 1 회 공개키 연산이 필요하므로 큰 차이가 없다고 볼 수 있다.

서버의 경우에도 이와 유사하며, 본 논문에서는 생략한다.

6. 결론

이 논문에서는 패스워드 인증 방식에 기반하여 새로운 TLS 프로토콜을 설계하고 이를 보안성과 효율성 측면에서 개략적으로 분석하였다. 새로 설계한 PTLS 는 보안성 측면에서 약한 키를 사용함에도 불구하고 active 한 공격에 약한 면을 보이지 않으며, 효율성 측면에서도 기존의 TLS 에 비해 핸드쉐이크시 메시지를 한번 더 주고 받아야 하는 점을 제외하고는 다른 약점을 보이지 않았다. 더군다나 인증서 관련 연산 및 복잡한 PKI 관련 정의(CA, RA, 인증서 발급/갱신/취소 절차, CRL, 인증서 타입 등) 등이 필요없음을 생각한다면 큰 장점을 가진 프로토콜이라고 볼 수 있다.

EAP-PTLS 역시 EAP-MD5 혹은 다른 EAP 에 비해 패스워드를 이용한 인증 방법 제공 및 세션키 교환이 가능하면서도 안전성 증명이 가능하다는 좋은 장점을 지니고 있음을 보았다.

본 논문에서 패스워드에 기반한 인증 방법을 논함에 있어서 패스워드 자체를 서버로부터 발급/갱신/취소하는 절차에 대해서는 언급하지 않았다. 이는 추후 연구대상으로 두도록 한다.

참고문헌

[1] D. Nyang, "Armoring password based protocol using zero-knowledge with secret coin tossing", 2001 IEEE International Symposium on Information Theory, pp 139-139, IEEE.
[2] T. Dierks and C. Allen, "The TLS Protocol", IETF RFC 2246, 1999.
[3] L. Blunk and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", IETF RFC 2284, 1998
[4] T. Wu, "The Secure Remote Password Protocol", Internet Society Symposium on Network and Distributed Systems Security, pp. 97-111.
[5] T. Wu, "The SRP Authentication and Key Exchange System", IETF RFC 2945, 2000
[6] D. Taylor, "Using SRP for TLS Authentication", IETF Internet Draft, 2001.
[7] D. Jablon, "Strong password-only authenticated key exchange", ACM Comp. Comm. Review, Vol. 26, No. 5, pp. 5-26.
[8] D. Jablon, "Extended Password Key Exchange Protocols Immune to Dictionary Attacks", Proc. of WET-ICE 97, IEEE Computer Society, Cambridge, MA, pp. 248-255.
[9] S. Bellovin and M. Merrit, Augmented encrypted key exchange: a password based protocol secure against dictionary attacks and password file compromise, ACM Conference on Comp. and Comm. Security, pp.244-250.
[10] L.C. Guillou and J.J. Quisquater, "Protocol fitted to security microprocessor minimizing both transmission and memory", Proceedings of EuroCrypt 88, Springer-Verlag, pp. 123-128.
[11] U. Feige, A. Fiat and A. Shamir, "Zero-knowledge proofs of identity", Journal of Cryptology, Vol. 1, No. 2, pp. 77-94.
[12] C.P. Schnorr, "Efficient Identification and Signatures for Smart cards, Advances in Cryptology", Proceedings of Crypto 89, Springer-Verlag, pp. 239-251.
[13] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attacks", Proceedings of EuroCrypt 2000, Springer-Verlag, pp. 139-155.
[14] B. Aboba and D. Simon, "PPP EAP TLS Authentication Protocol", IETF RFC 2716, 1999
[15] P. Funk and S. Blake-Wilson, "EAP Tunneled TLS Authentication Protocol (EAP-TTLS)", IETF Internet draft, 2002
[16] W. Simpson, "The Point-to-Point Protocol (PPP)", IETF RFC 1661, 1994
[17] "Port-Based Network Access Control", IEEE Standard for Local and Metropolitan Area Networks, 2001.
[18] J. Carlson, "EAP SRP-SHA1 Authentication Protocol", IETF Internet Draft, 2001
[19] S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks", Proceedings of the IEEE Symposium on Research in Security and Privacy, 1992