3 자간 통신이 가능한 TLS 프로토콜

김인환*, 최형기*
*성균관대학교 정보통신공학부
e-mail:playkih@ece.skku.ac.kr, hkchoi@ece.skku.ac.kr

3-Party Communication Enabled TLS Protocol

In-Hwan Kim*, In-Yong Hwang**, Seok-Joong Kim**, Hyoung-Kee Choi*
*School of Information and Communication Engineering, Sungkyunkwan University

요 약

통신기술의 발달에 따라, 사용자들은 다양한 통신 서비스와 편의를 제공받게 되었지만, 서비스 거부 공격 및 재전송 공격 등의 다양한 위협 역시 존재 하게 되었다. 이에 따라, IETF는 안전한 통신을 위해 Transport Layer Security (TLS)를 제안하였다. 그러나 TLS는 3자간 통신을 보호하려 할 때, Handshake 과정을 여러 번 반복해야하는 비효율성을 지니고 있었다. 따라서 우리는 본 논문을 통해 3자간 통신이 가능한 TLS 프로토콜을 제안한다. 기존 TLS와의 호환성 유지를 위해 최소한의 변형만 이루어졌으며, 객체 간 통신 시간 측정을 통해 Handshake 과정을 효율적으로 수행할 수 있도록 구성하였다.

키워드 : Security, TLS, 3-Party Communication

I. 서론

통신 기술이 발달함에 따라 이제는 누구나 쉽게 인터넷에 접속하고, 이용할 수 있는 환경이 마련되었다. 인터넷에 대한 관심 및 수요가 증가하면서 인터넷 망을 통한 다양한 서비스가 구현되고, 제공되고 있다. 그러나 이러한 서비스들을 사용 시 보안 기법이 적용되지 않는다면, 공격자는 사용자의 서비스를 가로채거나 전송되는 데이터를 위조하는 등의 다양한 공격이 가능하다. 따라서 통신 시 보안 프로토콜의 적용을 통해 사용자 인증을 제공하고, 전송 데이터의 무결성과 기밀성을 보장하여야 한다.

이를 위해 IETF는 전송계층의 보안 프로토콜인 Transport Layer Security (TLS) [1] 를 발표하였다. TLS는 인증서를 이용한 server와 client간 상호인증, 데이터의 압축 및 암호화 방식 및 키 분배 및 협상 방식을 제공하고 있다. TLS는 가장 대표적인 보안 프로토콜로서 client와 server 통신 구조를 이루는 환경에서 광범위하게 사용되고 있다.

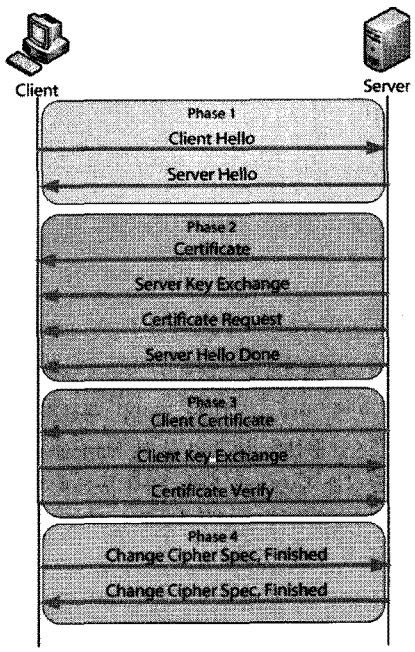
그러나 TLS는 client와 server간의 1:1 통신 상황만을 가정하고 구성되어있다. 이에 따라, 현재 TLS에서 3자간의 통신을 보호하기 위해서는 각 객체 별로 각각의 TLS 세션을 맺어야 하고, 동일한 메시지를 다른 객체에게 전송하려면, 수신자별로 다른 키를 사용하여 암호화과정을 수행해야 한다. 즉, TLS를 통해 3자간 통신을 보호하게 되면, 긴 지연시간과 통신 자원을 비효율적으로 소모하는 단점을 지니게 되었다.

따라서 본 논문에서는 TLS의 Handshake의 과정을 개선하여, 3 자간 암호 통신이 가능한 효율적인 TLS를 제안한다. 본 논문의 2장은 TLS의 전반에 대해서 분석하고 있으며, 3장은 본 논문이 제안하는 새로운 TLS의 자세한 프로토콜을 제시하고 있다. 4장은 논문의 결론에 대해서 설명하고 있다.

II. TLS

TLS는 전송계층의 보안 프로토콜로서 HTTP 등과 같이 다양한 application계층에서 전달되는 데이터에 대한 압축 및 암호화 기능을 제공한다.

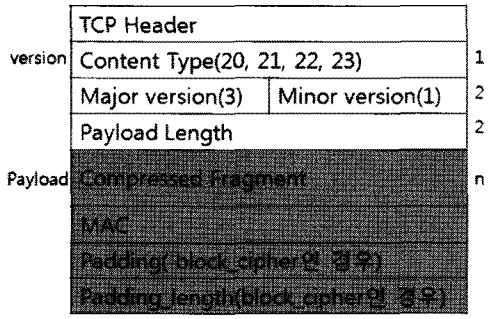
TLS는 Record 프로토콜, Handshake 프로토콜과 Alert 프로토콜로 구성되어 있다. Record 프로토콜은 Handshake, Alert등의 제어 메시지와 application계층에서 전달되는 메시지를 수납하는 역할을 한다. Handshake 프로토콜은 server와 client간의 상호인증을 수행하고, 암호화 및 압축 알고리즘을 협상하고, 세션키 생성을 위해 사용된다. Alert 프로토콜은 Handshake과정에서 상대방이 제시한 암호화 방식을 지원할 수 없는 경우 이에 대한 오류 정보를 알릴 때 사용된다.



〈그림 1〉 TLS의 Handshake 절차

〈그림 1〉은 데이터를 전송하기 전에 TLS 세션을 맺기 위한 Handshake 과정을 보여주고 있다. TLS는 압축, 암호화 및 Message Authentication Code (MAC) 생성 방법을 하나로 규정하고 있지 않다. 이에 따라, server와 client는 자신들이 지원할 수 있는 압축, 암호화 및 MAC 생성 방법을 서로 교환하여 그중 가능한 방법을 선택한다. 이러한 협상정보는 Client Hello와 Server Hello 메시지를 통해 전달된다. 이후, client와 server는 인증서 교환을 통해 상호인증을 수행한다. 상호인증이 정상적으로 종료하면, Key Exchange 메시지를 통해 세션키 생성을 위한 정보를 교환하고 세션키를 생성한다. 그 후, Change Cipher Spec 메시지를 통해 양 쪽이 협상한 압축, 암호화와 MAC생성 방식을 다시 한 번 확인한 후에 Handshake과정을 종료한다.

Handshake과정은 키 생성방식에 따라 RSA 공개키 방식과 Diffie-Hellman 방식으로 구별된다. RSA 공개키 방식은 키 생성에 필요한 pre-master값을 client가 임의로 선정한 후, 이를 server의 공개키 값으로 암호화하여 Client Key Exchange 메시지에 포함시켜 보내준다. Diffie-Hellman 방식은 RSA와 다르게 server와 client간의 키 협상과정이 이루어진다. server는 임의의 값 a를 설정하고, $g^a \bmod p$를 계산하여, 이를 Certificate 메시지에 포함시켜 전송한다. 이를 받은 client 역시 임의의 값 b를 설정하고, $g^b \bmod p$를 계산하여, 이를 Key Exchange 메시지에 포함시켜 전송한다. server와 client 모두 $g^{ab} \bmod p$를 계산할 수 있게 되고, 이를 pre-master 값으로 설정한다.



〈그림 2〉 Record Protocol의 메시지 형식

Handshake 이후의 통신은 record 프로토콜을 이용해서 전송되며, Handshake과정에서 생성된 세션키를 통해 기밀성과 무결성을 보장 받는다. Record 프로토콜의 동작은 다음과 같이 이루어진다. Application 계층에서 보내지는 데이터는 전송을 위해 적절한 크기로 분할되고, 압축된다. 이후, 압축된 데이터에 해당하는 MAC을 생성하고, 암호화 과정이 이루어진다. 이 모든 과정은 Handshake 과정에서 협상된 압축, 암호, MAC 생성 방식을 그대로 사용한다.

〈그림 2〉는 TLS record 패킷의 기본 포맷을 보여주고 있다. Content type은 수납되는 Alert, Handshake와 같은 상위계층 프로토콜의 종류를 표시 한다. Version은 Secure Socket Layer (SSL) 과 호환성을 위하여, major는 3, minor는 1로 설정된다. Payload length는 암호화된 영역의 길이로서, 압축된 fragment의 길이와 MAC의 길이를 합친 것과 같다.

III. 제안하는 3-Party TLS

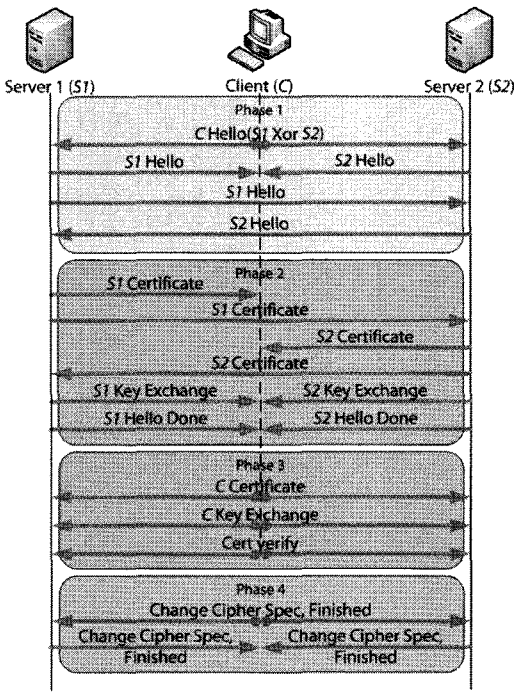
3 Party TLS는 기존 TLS의 Handshake과정을 변화시켜 3개의 객체가 동일한 master secret을 공유할 수 있도록 한다. 우리는 기존 TLS와의 호환성을 유지하기 위해 기존 TLS의 포맷에 3-Party TLS를 위한 5 가지 메시지 타입만을 새롭게 정의하였다. 기존 TLS header의 content type에 새롭게 24, 25, 26, 27, 28이 추가되었다. 각 번호가 지칭하는 바는 〈표 1〉에 나타나 있다. 또한, Default 3-Party TLS 외에 Handshake 메시지를 추가적으로 정의해서 통신비용을 최소화하는 Optimized 3-Party TLS도 함께 정의 하였다

〈표 1〉 3-Party TLS를 위해 추가된 Content Type

Number	Content Type
24	Change Cipher Spec in 3-Party TLS
25	Alert in 3-Party TLS
26	Default Handshake in 3-Party TLS
27	Optimized Handshake lin 3-Party TLS
28	Application Data in 3-Party TLS

1. Default 3 Party TLS

3 Party TLS는 기존의 TLS와 동일하게 하나의 client가 Hello 메시지를 보내면서 Handshake과정이 시작된다. 〈그림 3〉는 3 Party TLS의 Handshake과정을 보여주고 있다.



〈그림 3〉 Default 3-Party TLS

2개의 server는 서로를 인식하고 있지 못하기 때문에 client는 Hello 메시지 내에 두 server의 ID를 ex-or하여 보낸다. 이를 받은 server들은 자신들의 ID로 동일하게 ex-or을 수행하여 서로를 인식하게 한다. 이후, server들은 Hello 메시지를 전송하여 서로가 지원할 수 있는 압축 및 암호화 MAC 생성 방법을 알게 된다. 이 후의 과정은 기존의 TLS와 거의 유사하다. 그러나 기존의 TLS와 달리 TLS세션을 맺기 위해 Handshake과정을 여러 번 수행 할 필요 없이, 양쪽이 동일하게 원하는 메시지는 브로드캐스트하여 중복적인 메시지 생성 및 전달 과정을 제거하였다.

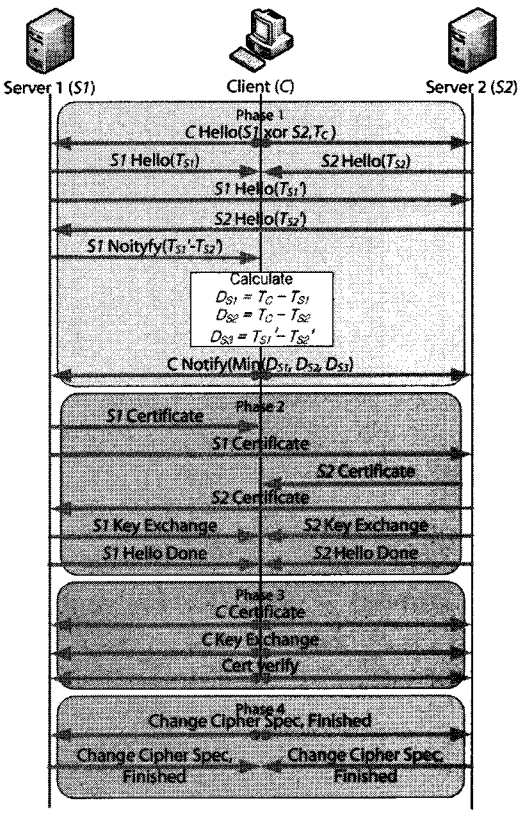
3-Party TLS는 TLS에 참여하는 개체가 2개에서 3개로 확장되었기 때문에 키 생성 방식이 변형되어야 한다. RSA를 사용하여 키 생성을 하는 경우는 기존의 TLS와 동일하게 client가 임의의 pre-master 값을 생성하여, 양 server의 공개키로 암호화하여 전송한다. Diffie-Hellman의 키 협상 방법을 사용하는 경우, 기존방식을 확장한 Group Diffie-Hellman [2]의 방식을 따른다. 우선 server 1은 임의의 값 a를 생성하고, server 2는 임의의 값 c를 생성한다. server 1은 $g^a \bmod p$를 계산하고, server 2는 $g^c \bmod p$를 계산하고, 각 계산된 값은 각자의 Key Certificate 메시지를 통해 전달한다. 양 server는 동일하게 $g^{ac} \bmod p$를 공유하게 된다. server들은 계산한 $g^{ac} \bmod p$를 Key Exchange 메시지에 포함시켜 client에게 전달해준다. 이후, client는 임의의 값 b를 설정하고, $g^{ab} \bmod p$와 $g^{bc} \bmod p$를 계산한다. 이는 server들의 Ceritificate 메시지를 통해서 $g^a \bmod p$와 $g^c \bmod p$이 전달되었기 때문이다. 또한, server들의 Key Exchange 메시지를 통해 $g^{ac} \bmod p$를 전달받았기 때문에, $g^{abc} \bmod p$역시 계산할 수 있게 된다. 이후, client는 Certificate 메시지를 통해 계산한 $g^{ab} \bmod p$와 $g^{bc} \bmod p$를 양 server에게 전달하고, 이를 받은 server들은 $g^{abc} \bmod p$를 계산하수 있게 된다. 이 값이 3개의 객체 사이에 pre-master값으로 설정된다.

2. Optimized 3-Party TLS

Optimized 3-Party TLS의 Handshake과정은 전체적으로 Default 3-Party TLS와 동일하다. 그러나 Handshake 과정의 초기에 client 및 server들 간의 통신 시간을 계산해서 그 뒤의 Handshake과정은 효율적으로 이루어지게 된다. 〈그림 4〉가 Optimized 3-Party TLS의 Handshake 과정을 보여주고 있다.

시작은 Default 3-Party TLS와 동일하게 client가 Hello 메시지를 보내면서 시작된다. 이 때 Hello 메시지에는 client가 메시지를 생성한 timestamp T_C가 포함된다. 이에 대한 응답으로 server들도 Hello 메시지를 생성하고 timestamp 값 T_{S1}과 T_{S2}를 측정하여 메시지에 포함시켜 응답한다. 이때 사용한 timestamp를 이용하여, client와 server 1사이의 통신시간 D_{S1}과 client와 server 2사이의 통신시간 D_{S2}가 각각 측정된다. Client의 Hello 메시지를 통해 server들은 TLS Handshake에 참여하는 모든 객체들을 알 수 있었다. 서로를 인식한 두 server들중 작은 ID 값을 가진 server가 먼저 timestamp가 포함된 Hello 메시지를 다른 server에게 전송한다. 본 논문에서는 server 1이 더 작은 ID 값을 갖고 있기 때문에 server 1이 먼저 Hello 메시지를 server 2에게 보낸다. 이를 받은 server 2는 timestamp가 포함된 Hello 메시로 응답한다. 두 server가 주고받은 timestamp를 통해 통해 두 server 간의 통신거리 D_{S3}가 측정된다. 그 후, 작은 ID를 가진 server는 Notify 메시지를 통

해 server간 측정된 통신시간을 client에게 알려준다. Client는 각 객체들 간의 통신시간을 모두 알 수 있게 되었고, 가장 짧은 통신 시간을 가지는 경로를 Notify 메시지를 통해 server들에게 알려준다. TLS에 참여하는 객체들은 최적화된 통신경로를 알게 되었기 때문에, 이후의 Handshake 과정은 통신시간이 가장 짧은 객체가 중심에 위치하여 진행하게 된다. 중심에 위치하는 객체는 다른 객체들과의 통신시간이 가장 짧은 객체가 되고, 이에 따라 가장 많은 메시지를 전송하게 된다. 〈그림 4〉에서는 server 1, client, server 2 순으로 배열되어 서로 메시지를 전송하고 있으나, 이는 $D_{S1} + D_{S2}$가 가장 작은 값이 나왔을 경우이다. $D_{S2} + D_{S3}$가 가장 작은 값일 경우에는 client, server 2, server 1순으로 배치되고, $D_{S3} + D_{S1}$이 가장 작은 값일 경우에는 client, server 1, server 2순으로 배열된다. Phase 2부터 이후의 과정은 객체간의 위치만 변경될 뿐, Default 3-Party TLS와 동일하다.



〈그림 4〉 Optimized 3-Party TLS

IV. 평가 및 분석

1:1 통신 상황만을 가정하고 설계된 기존의 TLS를 확장해서 3-Party TLS를 구성하였기 때문에, 키 생성 과정에서 공격자도 키를 유추할 수 있는 위협이 존재할 수 있다.

기존 TLS에서는 pre-master값을 server의 공개키로 암호화해서 전달하거나, Diffie-Hellman의 키 생성 방식을 이용하여, 키 유출을 방지하였다. 3-Party TLS에서도 이는 동일하게 적용된다. RSA 키 교환방식의 경우, client가 설정한 pre-master값을 각 server들의 공개키로 암호화하여 전달하기 때문에, 해당 개인키를 갖고 있지 않은 이상 이를 풀어볼 수 없다. 또한, Diffie-Hellman의 키 교환 방식을 이용하는 경우, group Diffie-Hellman과 동일한 방식을 사용하기 때문에 각 객체가 설정하는 값을 모두 알지 않는 이상, pre-master값을 유추할 수 없다.

3-Party TLS는 기존의 TLS와 비교하여 훨씬 더 적은 통신비용을 소모한다. 기존의 TLS는 3자간에 세션을 맺기 위해서는 3번의 Handshake과정이 필요하며, 총 33개의 메시지를 주고받아야 한다. 그 외에도 동일한 메시지를 다른 객체들에게 보내려할 때 메시지를 수신하는 객체에 따라 다른 키를 사용하여 암호화 과정을 수행해야하는 비효율성을 지니고 있었다. 그러나 3-Party TLS는 한 번의 Handshake과정을 통해서 통신을 보호할 수 있는 키를 생성하며, 이 때 필요한 메시지수도 19개이다. 또한, 세 개의 객체가 공유하고 있는 세션키는 동일하므로 한 번의 암호화 과정만으로 메시지를 객체들에게 전달 할 수 있게 된다.

V. 결론

인터넷 망을 통해 다양한 서비스가 제공되게 되었지만, 이와 비례하여 여러 공격들 역시 존재하게 되었다. 이에 따라, 안전한 통신을 위해 TLS가 개발되었지만, 1:1 통신 상황만을 가정하고 만들어져 3자간 TLS 세션을 맺으려고 하는 경우, 효율성이 떨어지는 단점이 존재 하였다.

따라서 우리는 3자간 TLS 세션을 맺고 통신이 가능한 3-Party TLS를 제안하였다. 제안한 프로토콜은 TLS를 통해 3자간 통신을 보호하려 할 때, Handshake과정을 여러 번 수행해야 하는 비효율성을 제거하였다. 또한, TLS 세션을 맺으려는 각 객체 간 최적 통신경로를 찾고, 통신비용을 최소화 할 수 있는 Optimized 3-Party TLS 역시 제안 하였다.

참고문헌

[1] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008.

[2] M. Steiner et al, "Diffie-Hellman Key Distribution Extended to Group Communication", In Proc. of ACM CCS, pp.31-37, (1996)