# 인터넷에서 사용되는 여러 기술 : HTTPS 8

**조인준**
KBS 미디어기술연구소 차장

지난 편에서 인터넷상에서 데이터를 암호화하여 전송할 때 사용되는 프로토콜인 SSL/TLS의 변화에 대해 설명하며 TLS 1.2 대비 TLS 1.3의 가장 큰 변화 중 하나인 핸드셰이크 관련 내용을 살펴보았습니다. 이번 편에서는 또 다른 하나의 큰 변화인 암호화 스위트(Cipher Suite)에 대해 다루겠습니다. 암호화 스위트는 클라이언트와 서버가 안전한 통신을 하기 위해 사용하는 알고리즘들의 집합을 말하며 TLS 통신에서 사용되는 암호화 방법들을 정의합니다. 암호화 스위트는 [표 1]과 같이 키 교환(Key Exchange), 인증(Authentication), 암호화(Encryption), 해시(Hash)와 관련된 알고리즘들로 구성됩니다. 인증을 위한 서명과 암호화 등에 관한 내용은 연재의 앞선 내용에서 다루었으므로 이번 편에서는 간략한 언급만 하도록 하겠습니다. 구체적 내용이 궁금하시면 '인터넷에서 사용되는 여러 기술 : HTTPS 1~7'을 참고하세요.

**키 교환 알고리즘**
(Key Exchange Algorithm)
클라이언트와 서버 간에 안전하게 비밀 키를 교환하기 위한 알고리즘

**인증 알고리즘**
(Authentication Algorithm)
주로 서버의 인증에 사용되며, 클라이언트의 인증이 필요한 경우에도 사용될 수 있는 알고리즘으로 서버나 클라이언트가 자신이 서명한 인증서를 상대에게 전달하여 그 서명을 확인하는 알고리즘

**암호화 알고리즘**
(Encryption Algorithm)
메시지의 데이터 암호화에 사용되는 알고리즘

**해시 알고리즘**
(Hash Algorithm)
클라이언트와 서버가 메시지를 주고받을 때 중간에 누군가에 의해 조작되거나 전송 오류로 인해 메시지가 손상될 수 있습니다. 메시지의 조작이나 손상 여부를 파악하기 위해서 사용하는 기술을 MAC(Message Authentication Code: 메시지 인증 코드)이라 하며 TLS에서는 HMAC(Hash MAC)이라는 기술을 사용하는데 송신 측에서 메시지를 보낼 때 수신자와 사전에 합의된 해시(Hash)함수를 이용해서 해시 값을 만들고 메시지에 해시 값을 덧붙여 전송하면 수신 측에서 메시지의 해시 값을 자체적으로 다시 계산한 후 이를 받은 해시 값과 비교해 이 둘이 같으면 메시지에 조작이나 오류가 없다는 것을 검증할 수 있는 방법입니다. 메시지에서 해시 값을 계산하는 방식은 암호와 같이 제 3자가 알 수 있는 것이 아니라서 중간에 임의로 메시지를 조작하고 그에 맞는 해시 값을 같이 생성해서 수신 측을 속이는 것이 사실상 불가능합니다.

| 키 교환 (Key Exchange) | 인증 (Authentification) | 암호화 (Encryption) | 해시 (Hash) |
|---|---|---|---|
| RSA<br>DH<br>DHE<br>ECDH<br>PSK 등 | RSA<br>ECDSA<br>DSS<br>PSK 등 | AES-128-CBC<br>AES-256-CBC<br>AES-128-GCM<br>AES-256-GCM<br>3DES-CBC<br>DES-CBC<br>RC4-128 등 | SHA-1<br>SHA256<br>SHA384<br>MD5 등 |

**표 1. TLS 암호화 스위트 알고리즘 예 (TLS 1.2)**

암호화 스위트는 TLS 1.2에서 TLS 1.3으로 넘어가며 핸드셰이킹처럼 큰 변화를 겪습니다. 이 변화에 대해 알아보기 위해 우선 TLS 1.2의 암호화 스위트를 간략히 설명하고 TLS 1.3의 암호화 스위트가 TLS 1.2와 어떻게 다른지 비교해보도록 하겠습니다.

## TLS 1.2 암호화 스위트

TLS 1.2에서 암호화 스위트는 [표 1]의 요소별 알고리즘을 조합하여 [그림 1]과 같은 형식으로 표시됩니다. 예를 들어, 키 교환에 ECDH, 인증에 ECDSA, 암호화에 AES_128_CBC, 해시에 SHA256을 사용하는 경우의 암호화 스위트는 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256로 표시됩니다.

TLS_[키 교환]_[인증]_WITH_[암호화]_[해시]

**그림 1. 암호화 스위트 형식**

[그림 1]의 형식으로 표시된 암호화 스위트는 '인터넷에서 사용되는 여러 기술 : HTTPS 4'의 핸드셰이킹 과정에서 설명한 [그림 2] Client Hello의 Cipher Suites를 통해 서버로 전달되고 이를 통해 서버는 클라이언트가 지원하는 암호화 스위트의 리스트를 알 수 있습니다.

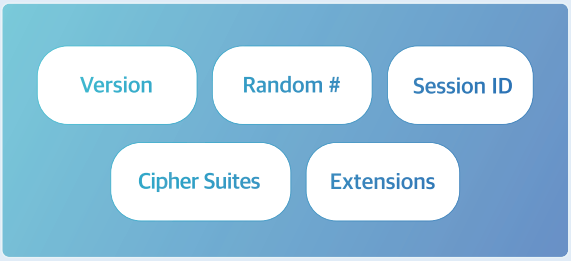


**그림 2. Client Hello**

클라이언트가 지원하는 알고리즘들을 조합하여 [표 2]와 같은 암호화 스위트의 지원이 가능하다면 [그림 2] Client Hello의 Cipher Suites를 통해 [표 2]의 리스트가 서버로 전달됩니다. 서버는 Client Hello의 응답인 Server Hello의 Cipher Suites에 클라이언트가 지원하는 암호화 스위트 중 하나를 선택하여 이후의 통신에 사용할 암호화 스위트를 클라이언트에 알려줍니다.

TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_PSK_WITH_AES_128_CBC_SHA256
TLS_PSK_WITH_AES_128_CBC_SHA256
TLS_DH_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

**표 2. 클라이언트 지원 암호화 스위트 리스트 예**

TLS 1.2의 암호화 스위트에서 사용되는 주요 알고리즘의 간략한 특징은 다음과 같습니다.

## 키 교환 알고리즘 (Key Exchange Algorithm)

### RSA(Rivest-Shamir-Adleman)

- RSA 방식은 누구나 접근 가능한 공개키와 소유자만이 알고 있는 사설키의 쌍을 통한 비대칭키 암호화 방식
- 비대칭키 암호화를 사용하여 이후의 데이터 암호화를 위한 세션키를 교환

### DH(Diffie-Hellman)

- 두 호스트가 사전에 공유된 정보 없이도 공용 비밀키를 생성할 수 있도록 하는 알고리즘
- 생성된 공용 비밀키는 대칭키 암호화 알고리즘을 사용하여 데이터를 암호화하는 데 사용할 수 있으며 대칭키 암호화는 비대칭키 암호화보다 연산양이 적고 속도가 빠름

### DHE(Diffie-Hellman Ephemeral)

- DHE(Diffie-Hellman Ephemeral) 암호화는 Ephemeral (수명이 짧은, 덧없는)이 의미하는 바와 같이 DH 키 교환 방식을 기반으로 매 세션(Session, 호스트 간 논리적 연결)마다 새로운 DH 키 쌍을 생성하여 사용함으로써 보안 수준을 높인 키 교환 알고리즘

### ECDH(Elliptic Curve Diffie-Hellman)

- 타원 곡선 암호(Elliptic Curve Cryptography, ECC)를 사용하여 DH 키 교환 알고리즘을 구현한 방식
- ECDH는 더 짧은 키로도 높은 보안성을 제공하여 효율성과 보안성에서 이점을 가짐
- ECDH는 공개키가 변하지 않는 방식이며, ECDHE(ECDH Ephemeral)는 연결마다 공개키가 바뀌는 방식

### PSK(Pre-Shared Key)

- PSK 키 교환 알고리즘은 사전에 공유된 비밀키를 사용하여 간단하고 효율적으로 통신 데이터를 보호하는 방법
- 사전에 공유된 비밀키를 데이터 암호화를 위한 세션키로 사용

  ※ TLS 핸드셰이크 이전에 이미 세션키는 서버와 클라이언트 모두 가지고 있고 핸드셰이크에서 사용할 세션키를 상호 확인하는 방식이며, 비밀키를 공유하는 메커니즘이 별도로 존재
- 설정이 간단하고 성능이 우수하지만 키 관리와 확장성 측면에서 제한이 있을 수 있음
- 주로 소규모 네트워크나 제한된 환경에서 사용

## 인증 알고리즘 (Authentication Algorithm)

### RSA(Rivest-Shamir-Adleman)

- 앞서 설명된 비대칭키 암호화를 통해 서명 및 인증

### ECDSA (Elliptic Curve Digital Signature Algorithm)

- 타원 곡선 암호(ECC)를 기반으로 한 디지털 서명 알고리즘이며 메시지의 무결성과 인증을 보장하기 위해 사용
- RSA 기반의 디지털 서명보다 더 짧은 키 길이로 높은 보안 수준을 제공

### DSS(Digital Signature Standard)

- 미국 국립표준기술연구소(NIST, National Institute of Standards and Technology)에서 제정한 디지털 서명 표준
- DSA(Digital Signature Algorithm)를 기반으로 하며 여러 암호화 알고리즘(DSA, RSA, ECDSA)을 지원

### PSK (Pre-Shared Key)

- 앞서 설명된 사전에 공유된 비밀키를 통해 서명 및 인증

## 암호화 알고리즘 (Encryption Algorithm)

### AES-128-CBC & AES-256-CBC

- AES(Advanced Encryption Standard)는 미국 국립표준기술연구소(NIST)에서 제정한 대칭키 암호화 표준
- AES는 다양한 키 길이(128, 192, 256비트)를 지원하며 키 길이에 따라 AES-128, AES-256 등으로 구별
- CBC(Cipher Block Chaining) 모드는 암호화 모드 중 하나로, 각 블록이 암호화될 때 이전 블록의 암호문과 XOR 연산을 통해 암호화하여 동일한 데이터가 반복되더라도 다르게 암호화하여 패턴을 숨김
- 데이터가 암호화 블록 크기의 배수가 아닌 경우 패딩이 필요함

### AES-128-GCM & AES-256-GCM (Galois/Counter Mode)

- AES(Advanced Encryption Standard)를 기반으로

CBC(Cipher Block Chaining) 모드 대신 GCM 모드를 채용하여 다음과 같은 특징을 제공

- 각 블록이 독립적으로 처리되므로 암호화 및 복호화 시 병렬 처리가 가능
- 데이터의 길이가 암호화 블록 크기의 배수가 아닌 경우에도 데이터 패딩이 필요하지 않음

### 3DES-CBC

- 3DES(Data Encryption Standard)는 DES의 보안성을 향상시키기 위해 제안된 방법으로, 세 번의 DES 연산을 수행하여 더 강력한 암호화를 제공

  ※ DES : IBM에서 개발한 대칭키 알고리즘으로, 56비트 키를 사용하며 키 길이의 한계와 공격 기법의 발전으로 보안이 취약해졌음
- 키 길이는 168비트(3개의 56비트 키)
- 암호화된 각 블록은 이전 블록의 암호문과 XOR 연산을 수행하여 다음 블록을 암호화하는 CBC 모드를 사용

### DES-CBC

- 위 3DES-CBC 이전에 개발된 방식으로 DES 연산을 한 번만 사용하는 방식
- 초기에는 안전성이 보장되는 알고리즘이었으나 현재는 비교적 키 길이가 짧아 보안 취약점이 발생하였고 추후 이의 해결을 위해 키 길이를 늘인 3DES가 사용되고 있음

### RC4-128

- RC4은 1987년에 Ronald Rivest에 의해 개발되었으며 데이터를 스트림으로 변환한 후에 키 스트림과 XOR 연산을 수행하여 암호문을 생성

  ※ 데이터를 일정한 크기로 암호화하는 블럭 암호화와 비교하여 스트림 암호화는 비트 또는 바이트 단위로 암호화하는 방식
- 현재는 보안 결함이 발견되어 안전하지 않은 것으로 알려져 있음

## 해시 알고리즘 (Hash Algorithm)

### SHA-1, SHA256, SHA384

- SHA(Secure Hash Algorithm)는 주어진 데이터의 고정 길이 해시 값을 생성하는 알고리즘
- 해시 값은 입력 데이터의 고유한 디지털 지문으로 사용됨
- SHA는 단방향 해시 함수로 주어진 입력에 대해 해시 값을 생성하며 입력으로부터 해시 값을 계산하는 것은 쉽지만, 해시 값을 이용하여 원본 입력을 복원하는 것은 매우 어렵거나 불가능
- SHA-1은 160비트 해시 값을 생성하는 가장 오래된 SHA 알고리즘 중 하나이며 현재는 안전하지 않은 것으로 여겨져서 2013년 이후 NIST에서 사용중지를 권고
- SHA-256, SHA-384는 각각 256비트, 384비트의 해시 값을 생성하는 SHA-2 계열의 알고리즘으로 SHA-1 대신 권고되고 있음

### MD5

- MD5(Message Digest Algorithm 5)는 앞선 RC4를 개발한 Ronald Rivest가 개발한 128비트 길이의 해시 값을 생성하는 해시 함수
- RC4와 마찬가지로 이전에는 널리 사용되었으나 현재는 보안상의 취약점으로 인해 RC4와 같이 안전하지 않은 것으로 알려져 있음

**지금까지 TLS 1.2의 암호화 스위트에 관해 간략히 정리해보았습니다. 다음 편에서는 TLS 1.3의 암호화 스위트가 TLS 1.2 대비 어떻게 변화되었는지 알아보겠습니다.**

P.S.

C군이 여러분께 전하는 내용 중 전문적 성격이 짙은 것은 엄밀한 언어를 사용하여 설명하기에는 한계가 있습니다.
본 내용은 설명하는 대상에 대한 전체적 맥락의 이해에만 이용하시고, 그 이상은 권위 있는 전문자료를 참고하시기 바랍니다.