SSL/TLS: Encryption

Michael Purcell

01 June 2021

Australian National University

Software Innovation Institute

SSL/TLS: Encryption

Cipher Suites

The Pseudorandom Function

Key Calculation

Record Payload Protection
- Legacy Cipher Suites
- AEAD Cipher Suites

References

Cipher Suites

A cipher suite is a set of cryptographic algorithms that can be used to secure a SSL/TLS session. An SSL/TLS cipher suite consists of:

- A key exchange algorithm
- An authentication algorithm
- A bulk encryption algorithm
- A message authentication code (MAC) algorithm

The primary purpose of the SSL/TLS Handshake is to establish which cipher suite will be used during the subsequent session.

Example Cipher Suites

Modern cipher suites that are believed to be secure:

- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256`

Legacy cipher suites that may be insecure:

- `TLS_RSA_WITH_RC4_128_MD5`
  - Both `RC4` and `MD5` are known to be insecure and should not be used in modern applications.
- `TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA`
  - The small block size of `3DES` (64 bits) makes it unsuitable to handle high-volume traffic as as demonstrated by the "Sweet32" attack.

SSL/TLS: Encryption

Cipher Suites

The Pseudorandom Function

Key Calculation

Record Payload Protection
- Legacy Cipher Suites
- AEAD Cipher Suites

References

HMAC

Given a hash function `hash`, we define

$$\texttt{HMAC\_hash}(\texttt{k}, \texttt{m}) = \texttt{hash}\left((\texttt{k'} \oplus \texttt{opad}) \parallel \texttt{hash}\left((\texttt{k'} \oplus \texttt{ipad}) \parallel \texttt{m}\right)\right)$$

where $\texttt{opad} = \texttt{0x5c5c...5c}$, $\texttt{ipad} = \texttt{0x3636...36}$, and $\texttt{k'}$ is a block-sized key derived from $\texttt{k}$. That is

$$\texttt{k'} = \begin{cases} \texttt{hash}(\texttt{k}) & \text{if } \texttt{k} \text{ is larger than the block size;} \\ \texttt{k} & \text{otherwise.} \end{cases}$$

The Pseudorandom Function

We have

$$\texttt{PRF}(\texttt{secret}, \texttt{label}, \texttt{seed}) = \texttt{P\_hash}(\texttt{secret}, \texttt{label} \parallel \texttt{seed})$$

where

$$\begin{aligned}\texttt{P\_hash}(\texttt{secret}, \texttt{seed}) = \texttt{HMAC\_hash}(\texttt{secret}, \texttt{A}(1) \parallel \texttt{seed}) \\ \parallel \texttt{HMAC\_hash}(\texttt{secret}, \texttt{A}(2) \parallel \texttt{seed}) \\ \parallel \texttt{HMAC\_hash}(\texttt{secret}, \texttt{A}(3) \parallel \texttt{seed}) \\ \vdots\end{aligned}$$

SSL/TLS: Encryption

Cipher Suites

The Pseudorandom Function

Key Calculation

Record Payload Protection
- Legacy Cipher Suites
- AEAD Cipher Suites

References

Compute the Master Secret

Let `cs_random` be the concatenation of the random bytes supplied by the client and server during the TLS Handshake. That is

$$\texttt{cs\_random} = \texttt{client\_random} \,\|\, \texttt{server\_random}.$$

Then we compute the value of `master_secret` via

$$\begin{aligned}\texttt{master\_secret} = \texttt{PRF}(&\texttt{pre\_master\_secret},\\ &\texttt{"master secret"},\\ &\texttt{cs\_random})[0 \ldots 47].\end{aligned}$$

Notice that `master_secret` is always 48 bytes long.

Key Calculation

Let `sc_random` be the concatenation of the random bytes supplied by the server and client during the TLS Handshake. Notice that the order is reversed form that used in the computation of `master_secret`. That is

$$\texttt{sc\_random} = \texttt{server\_random} \,\|\, \texttt{client\_random}.$$

Then we compute the value of `key_block` via

$$\begin{aligned}\texttt{key\_block} = \texttt{PRF}(&\texttt{master\_secret},\\ &\texttt{"key expansion"},\\ &\texttt{sc\_random})[0 \ldots \texttt{n-1}]\end{aligned}$$

where `n` is the required number of bytes of key material.

SSL/TLS: Encryption

Cipher Suites

The Pseudorandom Function

Key Calculation

Record Payload Protection
- Legacy Cipher Suites
- AEAD Cipher Suites

References

MAC-then-encrypt

All of the cipher suites defined in the original description of TLSv1.2 use `HMAC_hash` as their MAC algorithm.

The MAC is computed on the unencrypted record data.

The bulk encryption algorithm is then used to encrypt both the record data and MAC.

Stream Ciphers

A *stream cipher* generates a key stream which is XORed with the plaintext to produce the ciphertext.

Prior to 2016, RC4 was the only stream cipher used in SSL/TLS cipher suites.

In 2016, cipher suites which use ChaCha20 were introduced by RFC 7905.

Use of RC4 was prohibited in 2015 by RFC 7465

CBC Block Ciphers

A block cipher applies a key-dependent permutation to a *block* of plaintext to produce a block of ciphertext.

AES is used in many SSL/TLS cipher suites.

A *block cipher mode of operation* is an algorithm that allows a block cipher to be used to encrypt a plaintext that consists of more than one block.

The original specification for TLSv1.2 included algorithms for using block ciphers in CBC (cipher block chaining) mode to encrypt TLS records.

Padding

Block ciphers can only be applied to plaintexts whose length is an integer multiple of the block size.

If a TLS record is not an appropriate length then *padding* must be added.

The padding may be any length up to 255 bytes.

Each byte of padding is filled with the padding length value.

Initialization Vector

Using a block cipher in CBC mode requires an initialization vector (IV).

The IV is not secret and is sent in-the-clear immediately prior to the encrypted contents of its associated TLS record.

Despite being sent in-the-clear, the IV for each record must be unpredictable prior to transmission.

Early versions of TLS did not use explicit IVs. Instead, they used the last ciphertext block of the previous record. This allows an attacker compromise the security of the connection as demonstrated by the so-called ”BEAST” attack.

Authenticated Encryption

Authenticated encryption (AE) combines the functionalities of bulk encryption and message authentication into a single algorithm.

One simple way to create an authenticated encryption algorithm is via the encrypt-then-MAC (EtM) construction.

With EtM, the plaintext is first encrypted and then a MAC is computed (using a different key) for the resulting ciphertext.

This construction is commonly called the MAC-then-Encrypt (MtE) construction.

Additional Data

Authenticated encryption with associated data (AEAD) is an extension of authenticated encryption that allows a sender to include some data that must be authenticated but does not need to be (or cannot be) encrypted.

One common example of associated data is packet routing information. If the destination address and/or ports are encrypted, then the packet cannot be delivered. It can be useful, however, to authenticated the routing information to prevent an attacker from manipulating network traffic flows.

AEAD Algorithms

AEAD algorithms used in SSL/TLS cipher suites include:

- AES-GCM (AES - Galois/Counter Mode)
- AES-CCM (AES - Counter with CBC-MAC)
- ChaCha20-Poly1305

The associated data usually consists of a sequence number and information about the compression algorithms applied as part of the TLS protocol.

SSL/TLS: Encryption

Cipher Suites

The Pseudorandom Function

Key Calculation

Record Payload Protection
- Legacy Cipher Suites
- AEAD Cipher Suites

References

RFCs

- RFC 5246 - ”The Transport Layer Security (TLS) Protocol Version 1.2”
- RFC 2104 - ”HMAC: Keyed-Hashing for Message Authentication”
- RFC 7465 - ”Prohibiting RC4 Cipher Suites”
- RFC 7905 - ”ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)”
- RFC 5116 - ”An Interface and Algorithms for Authenticated Encryption”

Other Resources

- The paper ”On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN” by Karthikeyan Bhargavan and Gaëtan Leurent
- The blog ”Command Line Fanatic” by Jeremy Davies
  - ”An Illustrated Guide to the BEAST Attack”
- The blog ”hashedout” by The SSL Store
  - ”Cipher Suites: Ciphers, Algorithms and Negotiating Security Settings” by Patrick Nohe
- The blog ”Moxie Marlinspike” by Moxie Marlinspike
  - ”The Cryptographic Doom Principle”