What is a session key? | Session keys and TLS handshakes

The SSL/TLS protocol uses both public key and symmetric encryption, and new keys for symmetric encryption have to be generated for each communication session. Such keys are called "session keys."

Session Key Learning Objectives

After reading this article you will be able to:

- Learn what a session is, what a key is, and when new session keys have to be created
- Understand the differences between asymmetric and symmetric encryption
- Learn how the SSL/TLS encryption protocol uses both kinds of encryption
- Learn the 4 types of session keys used for symmetric encryption in SSL/TLS

Related Content

What is SSL?

SSL Handshake

SSL Certificate Types

Keyless SSL

How SSL Works

What is a session key?

A session key is any encryption key used to symmetrically encrypt one communication session only. In other words, it's a temporary key that is only used once, during one stretch of time, for encrypting and decrypting data; future conversations between the two parties would be encrypted with different session keys. A session key is like a password that someone resets every time they log in.

In SSL/TLS, the two communicating parties (the client and the server) generate 4 session keys at the start of any communication session, during the TLS handshake. The official RFC for TLS does not actually call these keys "session keys", but functionally that's exactly what they are.

What is a session?

A session is essentially a conversation. A session takes place over a network, and it begins when two devices acknowledge each other and open a virtual connection. It ends when the two devices have obtained the information they need from each other and send "finished" messages, terminating the connection, much like if two people are texting each other, and they close the conversation by saying, "Talk to you later." The connection can also time out due to inactivity, like if two people are texting and simply stop responding to each other.

A session can either be a set period of time, or it can last for as long as the two parties are communicating. If the former, the session will expire after a certain amount of time; in the context of TLS encryption, the two devices would then have to exchange information and generate new session keys to reopen the connection.

What is a cryptographic key?

In encryption, a key is a string of data that is used to alter messages so that they become encrypted – in other words, so that the data appears randomized or scrambled. A key is also used for decrypting the data, or translating it from its scrambled form to its original form. (See What is a cryptographic key? to learn more.)

What is symmetric encryption? What is asymmetric encryption?

In symmetric encryption, the exact same key is used on both sides of a conversation, for both encrypting and decrypting. In a session that uses symmetric encryption, multiple keys can be used, but a message that is encrypted with one key is decrypted with that same key.

In asymmetric encryption, there are two keys, and data that is encrypted with one key can only be decrypted with the other key – unlike in symmetric encryption, when the same key both encrypts and decrypts. This is also known as public key encryption, because one of the keys is shared publicly.

Does HTTPS use symmetric or asymmetric encryption?

HTTPS, which is HTTP with the TLS encryption protocol, uses both types of encryption. All communications over TLS start with a TLS handshake. Asymmetric encryption is crucial for making the TLS handshake work.

During the course of a TLS handshake, the two communicating devices will establish the four session keys, and these will be used for symmetric encryption for the rest of the session. Usually, the two communicating devices are a client, or a user device like a laptop or a smartphone, and a server, which is any web server that hosts a website. (For more, see What is the client-server model?)

How does a TLS handshake work?

During a TLS handshake, both client and server send each other random data, which they use to make calculations separately and then derive the same session keys. Three kinds of randomly generated data are sent from one side to the other:

- The "client random": This is a random string of bytes that the client sends to the server.
- The "server random": This is similar to the client random, except that the server sends it to the client.
- The "premaster secret": This is yet another string of data. In some versions of the TLS handshake, the client generates this and sends it to the server encrypted with the public key; in other versions, the client and server generate the premaster secret on their own, using agreed-upon algorithm parameters to arrive at the same result.

The TLS handshake uses asymmetric encryption either to hide the server random from attackers (by encrypting it with a private key), or for allowing the server to digitally "sign" one of its messages so that the client knows the server is who it claims to be (just as a signature helps verify someone's identity in real life). The server encrypts some data with the private key, and the client uses the public key to decrypt it, proving that the server has the correct key and is legitimate.

What is the 'master secret' in a TLS handshake?

The master secret is the final result from combining the client random, the server random, and the premaster secret via an algorithm. The client and the server each have those three messages, so they should arrive at the same result for the master secret.

The client and server then use the master secret to calculate several session keys for use in that session only – 4 session keys, to be precise.

What 4 session keys are generated from the master secret in a TLS handshake?

The 4 kinds of session keys created in each TLS handshake are:

- The "client write key"
- The "server write key"
- The "client write MAC key"
- The "server write MAC key"

The client write key is the key that the client uses to encrypt its messages. The client write key is a symmetric key, and both the client and the server have it. This enables the server to decrypt messages from the client using the same key.

The server write key is just like the client write key, except on the server side. To summarize: Messages from client to server are encrypted with the client write key, and the server uses the client write key to decrypt them. Messages from server to client are encrypted with the server write key, and the client uses the server write key to decrypt them. (This whole process is handled by the client device or browser; the users themselves don't have to do any of this encrypting or decrypting.)

The MAC, or message authentication code, keys are used to digitally sign messages. The server signs its messages with the server write MAC key, and when the client receives the message, it can check the MAC key used against its own record of the server MAC key to make sure it's legitimate. The client signs its messages with the client write MAC key.

A set of 4 completely new session keys gets created with every new communication session and new TLS handshake. There will be a different client write key, server write key, and so on, but those 4 types of keys get created every time.