Assignment 4

Soumyadeep Basu, Kumar Utkarsh, Rohit Haolader, Raushan Raj, Suryasen Singh

V Semester BTech, Department of Information Technology,

Indian Institute of Information Technology, Allahabad, India.

Q:9 A) When a session is resumed with a new connection, SSL does not require the full handshaking process. Show the messages that need to be exchanged in a partial handshaking.

Solution:

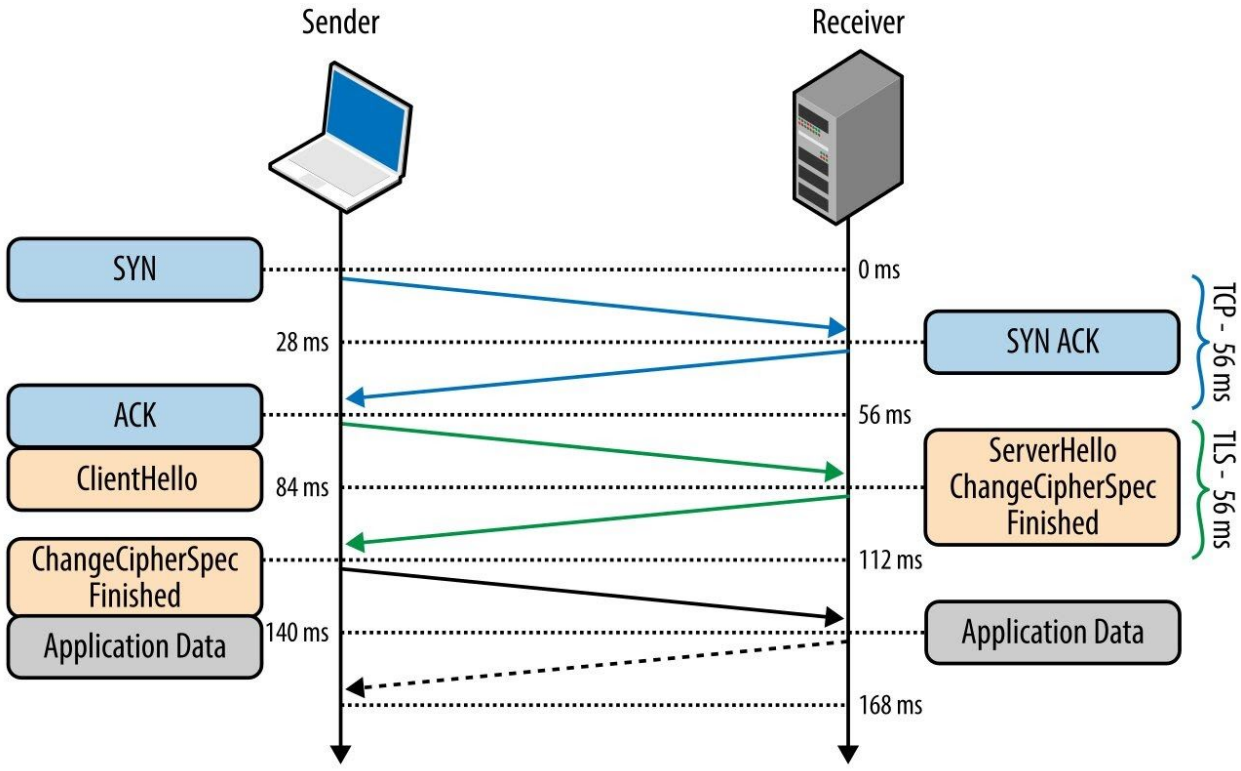
When a SSL/TLS handshake session take place, the client and the server share a symmetric secret-key which can be used to establish new SSL connections, thus avoiding heavy public-key operations which were required for the session establishment earlier because the additional latency and computational costs of the full SSL/TLS handshake inflict a significant performance penalty on all applications that need secure communication. To reduce some of these costs, TLS provides a mechanism to resume or share the identical negotiated secret key data between multiple connections.

MECHANISM:

I. **Session Identifiers**

RFC 5246 was the first Session Identifiers resumption mechanism introduced in SSL 2.0. It allowed the server to create and send a session ID consisting of 32 byte along with ServerHello message during the full SSL/TLS handshaking. Using that same session ID, both the client and server can retrieve the previously negotiated session data and reuse them for a subsequent session.

The client attaches the session ID to the ClientHello message and sends it to the server to specify that it is the same client with which the server has communicated before and it still remembers the negotiated session parameters (cipher suite and keys) from the previously negotiated session and can use them again. To that , the server searches the session datas related to that ID in its cache and if it finds, then a partial handshaking takes place. Otherwise, a new full SSL/TLS handshaking session is evoked to generate a new session ID.



ADVANTAGES:

It optimizes deployments as it reduces computational costs of the full SSL/TLS handshaking process.

It allows establishment of quick and secure connections without the loss of data and security.

LIMITATIONS:

Servers are required to create and maintain a session cache for every client which is not feasible because there are millions of unique connections every day which results in several problems on the server.
A requirement of cache for a session ID, memory consumption for every open handshaking session, and eviction policies.

Sites having multiple servers face challenges of deployment of non trivial tasks.

II.Session Tickets

RFC 5077 was the "Session Ticket" replacement mechanism introduced to address the problem of deployment of TLS session caches at the server side. It allowed the server to not keep per-client session state but add and maintain a new record for the session ticket, which contains all the information regarding the negotiated session and is encrypted with a secret key belonging only to the server.

The session ticket generated during the session gets stored with the client. When a new session is to be established with the same server, then the session ticket is attached within the ClientHello message and sent to the server.

The main improvement done in this mechanism is the removal of session cache at the server-side, which helps in reducing the cost of deployment as the client provides the session ticket every time on a new connection to the server till the ticket gets expired.

Q9 B)

I) Hash calculation for CertificateVerify message in TLS

Solution:

Actually the client sends **three** messages:

- Certificate contains its cert, with chain cert(s) if applicable which it usually is.
- ClientKeyExchange; and CertificateVerify contains a signature of the transcript so far using the client (private)key.
- The cert itself is verified in the standard X.509/PKIX way, and the CertVerify is verified using the key in the cert.

II) Hash calculation for Finished message in TLS

Solution:

We are using TLS_RSA_WITH_AES_256_GCM_SHA384 cipher suite. We tracked all handshake messages and successfully can decrypt the Client Finished message.

We collected all messages (in my case Client_Hello, Server_Hello, Certificate, Server_Done, Client_Key_Exchange) and then use the following PRF(master_secret, finished_label, Hash(handshake_messages)) finished_label = "client finished"

When doing Hash(handshake_messages) what we are doing here is digest using SHA384.

III) Calculating MAC in TLS:

Solution:

Message authentication code (MAC) is a method that is used to check the authenticity as well as the integrity of a message. It accepts two input parameters: a secret key and a message of arbitrary length.

Formula to calculate MAC in TLS -

HMAC_hash(MAC_write_secret, seq_num + TLSCompressed.type + TLSCompressed.version + TLSCompressed.length + TLSCompressed.fragment));