ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

Gaining Intuition of SSL / TLS Protocol Version 1.2 for Building Libraries

Aishwarya Radhakrishnan[1]

BE Student, Department of Information Technology, Mumbai University, Vivekanand Education Society's Institute Of Technology, Mumbai, India [1]

ABSTRACT: This paper defines TLS protocol that comprises two layers: the TLS record and the TLS handshake protocols. It also illustrates the messages that go across network when these protocols are implemented. Transport Layer Security (TLS) – and its predecessor, Secure Sockets Layer (SSL), which is now deprecated by the Internet Engineering Task Force (IETF) – are cryptographic protocols that provide communications security over a computer network.

KEYWORDS: Transport Layer Security (TLS), Secure Sockets Layer (SSL), Advanced Encryption Standard (AES), Rivest Cipher 4 (RC4), RSA (Rivest-Shamir-Adleman), DSA (digital signature algorithm), MAC (Media Access Control), SHA-1 (Secure Hash Algorithm 1) , Certificate authority (CA), Request for Comments (RFC), External Data Representation (XDR), Datum (The singular form of data), Virtual private network (VPN), Voice over Internet Protocol (voice over IP), Greenwich Mean Time (GMT), Coordinated Universal Time (UTC), Pseudo Random Function (PRF), Cipher-Block-Chaining(CBC)

I. INTRODUCTION

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL) are both frequently referred to as "SSL". SSL 1.0 has Internal Netscape design but it got lost in the mists of time. SSL 2.0 was again published by Netscape in November 1994 was badly broken. SSL 3.0 was designed by Netscape and Paul Kocher in November 1996. TLS 1.0 is Internet standard based on SSL 3.0 developed in January 1999 but TLS 1.0 is not interoperable with SSL 3.0 as it has same protocol design but different algorithms. TLS 1.0 is deployed in nearly every web browser.

The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications. The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP, is the TLS Record Protocol. The TLS Record Protocol provides connection security that has two basic properties: One is, the TLS Record Protocol is used for encapsulation of various higher-level protocols. Other is that the connection is private. Symmetric cryptography is used for data encryption (e.g., AES, RC4, etc.). The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol). The Record Protocol can also be used without encryption. The TLS Handshake Protocol provides connection security that has three basic properties. First, the peer's identity can be authenticated using asymmetric, or public key, cryptography (e.g., RSA, DSA, etc.).This authentication can be made optional, but is generally required for at least one of the peers. Second, the negotiation of a shared secret is secure: the negotiated secret is unavailable to eavesdroppers, and for any authenticated connection the secret cannot be obtained, even by an attacker who can place himself in the middle of the connection.Third, the negotiation is reliable: no attacker can modify the negotiation communication without being detected by the parties to the communication. One advantage of TLS is that it is application protocol independent. Higher-level protocols can layer on top of the TLS protocol transparently. The TLS standard, however, does not specify how protocols add security with TLS; the decisions on how to initiate TLS handshaking and how to interpret the authentication certificates exchanged are left to the judgment of the designers and implementors of protocols that run on top of TLS.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

II. THE GOAL OF THIS PAPER

This paper describes the two protocol that makes the TLS Protocol and illustrates the messages that go across network when these protocols are implemented. This document is intended primarily for readers who will be implementing the protocol and for those doing cryptographic analysis of it. This paper will help you gain a better intuition of the TLS Protocol but to implement TLS Protocol comprehensively refer RFC 5246.

III. PRESENTATION LANGUAGE

This document deals with the formatting of data in an external representation. The following very basic and somewhat casually defined presentation syntax will be used. The syntax draws from several sources in its structure. Although it resembles the programming language "C" in its syntax and XDR in both its syntax and intent, it would be risky to draw too many parallels.

Vectors

A vector (single-dimensioned array) is a stream of homogeneous data elements. The size of the vector may be specified at documentation time or left unspecified until runtime. In either case, the length declares the number of bytes, not the number of elements, in the vector. The syntax for specifying a new type, T', that is a fixed- length vector of type T is

```
T T'[n];
```

Here, T' occupies n bytes in the data stream, where n is a multiple of the size of T. The length of the vector is not included in the encoded stream. In the following example, Datum is defined to be three consecutive bytes that the protocol does not interpret, while Data is three consecutive Datum, consuming a total of nine bytes.

```
opaque Datum[3];/* three uninterpreted bytes */
Datum Data[9]; /* 3 consecutive 3 byte vectors */
```

Constructed Types

Structure types may be constructed from primitive types for convenience. Each specification declares a new, unique type. The syntax for definition is much like that of C.

```
struct {
    T1 f1;
    T2 f2;
    ...
    Tn fn;
} [[T]];
```

Enumerateds

A field of type enum can only assume the values declared in the definition. Only enumerateds of the same type may be assigned or compared.

```
enum { red(3), blue(5), white(7) } Color;
```

The names of the elements of an enumeration are scoped within the defined type. In this example, a fully qualified reference to the second element of the enumeration would be Color.blue. Such qualification is not required if the target of the assignment is wellspecified.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

```
Color color = Color.blue;     /* overspecified, legal */
Color color = blue;           /* correct, type implicit */
```

Miscellaneous
Comments begin with "/*" and end with "*/".
The basic numeric data type is an unsigned byte (uint8).

Table

Table used throughout this document are of the following format and used to define each message passed during TLS Handshake Protocol and TLS Record Protocol and give description about it. This is a generic format and cells in the table are optional.

Message Name
Meaning of this message
Pseudocode
Description of the pseudocode or message.

IV.THEORETICAL FOUNDATIONS

The Transport Layer Security (TLS) protocol evolved from SSL protocol and SSL is often used to refer to what is actually TLS. The combination of SSL/TLS is the most widely deployed security protocol used today and is found in applications such as Web browsers, email and basically any situation where data needs to be securely exchanged over a network, like file transfers, VPN connections, instant messaging and voice over IP. SSL is designed to establish encryption and identity assurance. It enables encrypted communication between a web server and a web browser. SSL ensures that all data passed between the web server and browser remains private and secure.

The TLS Record Protocol provides connection security that has two basic properties:

1. The connection is private. Symmetric cryptography is used for data encryption (e.g., AES, RC4,etc.). The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol). The Record Protocol can also be used without encryption.
2. The connection is reliable. Message transport includes a message integrity check using a keyed MAC. Secure hash functions (e.g., SHA-1, etc.) are used for MAC computations. The Record Protocol can operate without a MAC, but is generally only used in this mode while another protocol is using the Record Protocol as a transport for negotiating security parameters.

The TLS Handshake Protocol provides connection security that has three basic properties:

1. The peer's identity can be authenticated using asymmetric, or public key, cryptography (e.g., RSA, DSA, etc.). This authentication can be made optional, but is generally required for at least one of the peers.
2. The negotiation of a shared secret is secure: the negotiated secret is unavailable to eavesdroppers, and for any authenticated connection the secret cannot be obtained, even by an attacker who can place himself in the middle of the connection.
3. The negotiation is reliable: no attacker can modify the negotiation communication without being detected by the parties to the communication.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

TLS Record Protocol is used for encapsulation of various higher-level protocols. One such encapsulated protocol, the TLS Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data.

V. TLS HANDSHAKE PROTOCOL

The cryptographic parameters of the session state are produced by the TLS Handshake Protocol, which operates on top of the TLS record layer. When a TLS client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key encryption techniques to generate shared secrets.The TLS Handshake Protocol involves the following steps:

1. Exchange hello messages to agree on algorithms, exchange random values, and check for session resumption.
2. Exchange the necessary cryptographic parameters to allow the client and server to agree on a premaster secret.
3. Exchange certificates and cryptographic information to allow the client and server to authenticate themselves.
4. Generate a master secret from the premaster secret and exchanged random values.
5. Provide security parameters to the record layer.
6. Allow the client and server to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

The client sends a ClientHello message to which the server must respond with a ServerHello message, or else a fatal error will occur and the connection will fail. The ClientHello and ServerHello are used to establish security enhancement capabilities between client and server. The ClientHello and ServerHello establish the following attributes: Protocol Version, Session ID, Cipher Suite, and Compression Method. Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello random. The actual key exchange uses up to four messages: the server Certificate, the ServerKeyExchange, the client Certificate, and the ClientKeyExchange. New key exchange methods can be created by specifying a format for these messages and by defining the use of the messages to allow the client and server to agree upon a shared secret. This secret MUST be quite long; currently defined key exchange methods exchange secrets that range from 46 bytes upwards. Following the hello messages, the server will send its certificate in a Certificate message if it is to be authenticated. Additionally, a ServerKeyExchange message may be sent, if it is required (e.g., if the server has no certificate, or if its certificate is for signing only). If the server is authenticated, it may request a certificate from the client, if that is appropriate to the cipher suite selected. Next, the server will send the ServerHelloDone message, indicating that the hello-message phase of the handshake is complete. The server will then wait for a client response. If the server has sent a CertificateRequest message, the client MUST send the Certificate message. The ClientKeyExchange message is now sent, and the content of that message will depend on the public key algorithm selected between the ClientHello and the ServerHello. If the client has sent a certificate with signing ability, a digitally-signed CertificateVerify message is sent to explicitly verify possession of the private key in the certificate.At this point, a ChangeCipherSpec message is sent by the client, and the client copies the pending Cipher Spec into the current Cipher Spec.The client then immediately sends the Finished message under the new algorithms, keys, and secrets. In response, the server will send its own ChangeCipherSpec message, transfer the pending to the current Cipher Spec, and send its Finished message under the new Cipher Spec. At this point, the handshake is complete, and the client and server may begin to exchange application layer data. (See flow chart below.) Application data MUST NOT be sent prior to the completion of the first handshake.

When the client and server decide to resume a previous session or duplicate an existing session (instead of negotiating new security parameters), the message flow is as follows: The client sends a ClientHello using the Session ID of the session to be resumed. The server then checks its session cache for a match. If a match is found, and the server is willing to re-establish the connection under the specified session state, it will send a ServerHello with the same Session ID value. At this point, both client and server MUST send ChangeCipherSpec messages and proceed directly to Finished messages. Once the re-establishment is complete, the client and server MAY begin to exchange application layer data. (See flow chart below.) If a Session ID match is not found, the server generates a new session ID, and the TLS client and server perform a full handshake.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

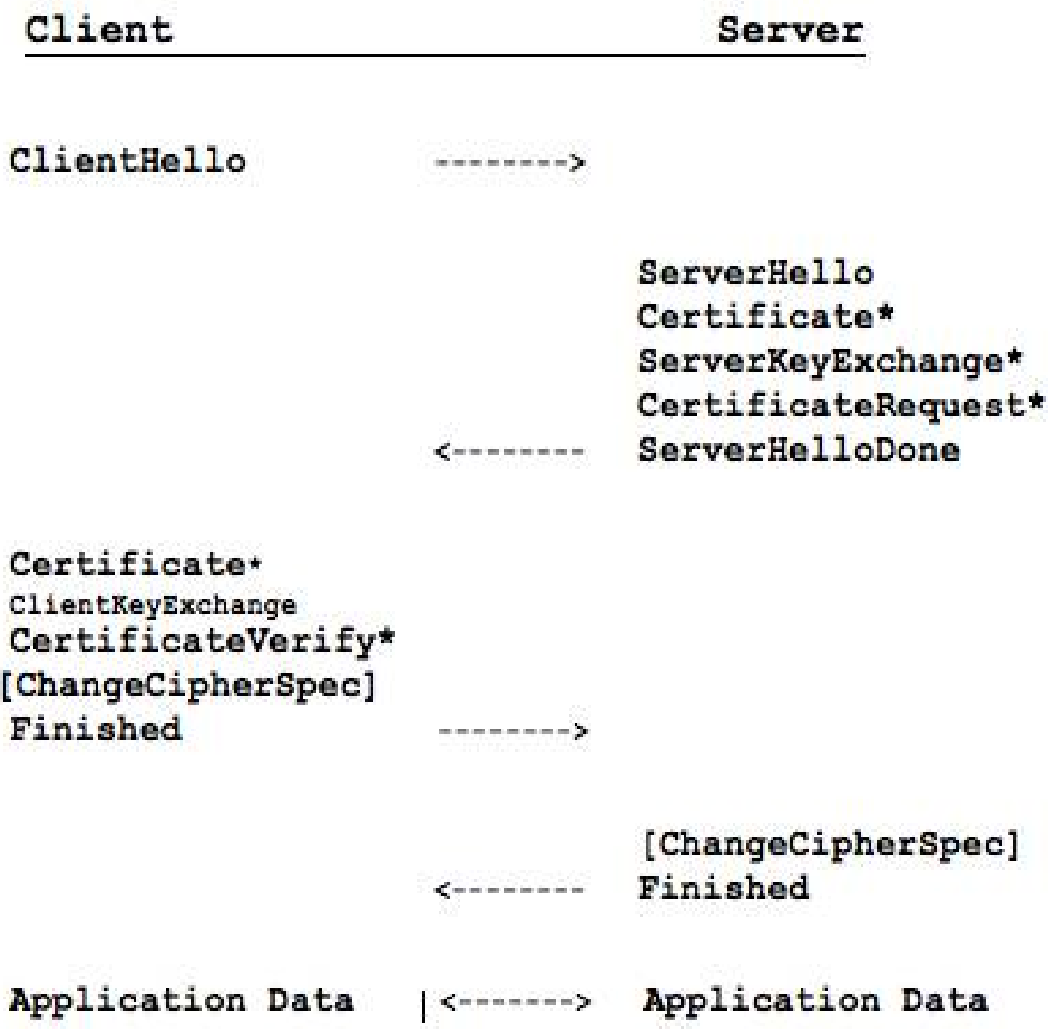

Figure 1. Message flow for a full handshake

In figure 1, * Indicates optional or situation-dependent messages that are not always sent.

Following are the description and pseudocode of the messages that are sent by the client and server during TLS Handshake protocol:

CLIENT HELLO
Meaning of this message: Initiate a new negotiation. It includes a random structure, which is used later in the protocol.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

Pseudocode:

```
struct{

         uint32   gmt_unix_time;     /* using GMT for
historical reasons, the predecessor of the current worldwide
time base, UTC. */

         SessionID    session_id;     /* from an earlier or
active connection */

         CipherSuite    cipher_suites;    /* list or from
previous session */

         CompressionMethod compression_methods;

         select (extensions_present) {
                case false: struct {};
                case true: Extension extensions;
          };

 } ClientHello;
```

SIGNATURE ALGORITHMS EXTENSION

Meaning of this message: Indicate to the server which signature/hash algorithm pairs may be used in digital signatures.

Pseudocode:

```
enum {
     md5(1), sha1(2), sha224(3), sha256(4)
 }  HashAlgorithm;
 enum {
    rsa(1), dsa(2)
}  SignatureAlgorithm;
 struct {
     HashAlgorithm hash;
     SignatureAlgorithm signature;
} SignatureAndHashAlgorithm;

SignatureAndHashAlgorithm  supported_signature_algorithms;
```

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

Description:

1. Each SignatureAndHashAlgorithm value lists a single hash/signature pair that the client is willing to verify.
2. The values are indicated in descending order of preference.
3. If the client does not send the signature_algorithms extension, the server uses preconfigured pairs like {sha1,rsa} , {sha1,dsa} etc.

SERVER HELLO

*Meaning of this message:*Hellophase messages are used to exchange security enhancement capabilities

Pseudocode:

```
struct {
     SessionID     session_id;
     CipherSuite     cipher_suite;
     CompressionMethod    compression_method;
     select (extensions_present) {
            case false: struct {};
            case true: Extension extensions;
      };
} ServerHello;
```

Description:
The single cipher suite and compression algorithm selected by the server from the list in ClientHello.cipher_suites and ClientHello.compression_methods

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

SERVER CERTIFICATE
*Meaning of this message:*Conveys the server's certificate chain to the client.

Pseudocode:
`opaque ASN.1Cert;`
`struct { ASN.1Cert certificate_list; } Certificate;`

Description: 1. The sender's certificate MUST come first in the list. Each following certificate MUST directly certify the one preceding it in certificate hierarchy. 2. Certificate type MUST be X.509v3, unless explicitly negotiated. 3. The "trusted_ca_keys" extensions are used to guide certificate selection. 4. If the client provided a "signature_algorithms" extension, then all certificates provided by the server MUST be signed by a hash/signature algorithm pair that appears in that extension.

SERVER KEY EXCHANGE
*Meaning of this message:*Sent by the server only when the server Certificate message does not contain enough data to allow the client to exchange a Premaster secret: a Diffie-Hellman public key

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

CERTIFICATE REQUEST
*Meaning of this message:*A server can optionally request a certificate from the client, if appropriate for the selected cipher suite

Pseudocode:
 	
`enum {`
`rsa_sign(1), dsa_sign(2), rsa_ephemeral_dh_RESERVED(5)`
`} ClientCertificateType;`
 	
`opaque DistinguishedName;`
 	
`struct {`
`SignatureAndHashAlgorithm supported_signature_algorithms;`
`ClientCertificateType certificate_types;`
`DistinguishedName certificate_authorities;`
`} CertificateRequest`

SERVER HELLO DONE
*Meaning of this message:*Upon receipt of the ServerHelloDone message, the client SHOULD verify that the server provided a valid certificate, if required, and check that the server hello parameters are acceptable.
Pseudocode: `struct { } ServerHelloDone;`

CLIENT CERTIFICATE
Description: This message is only sent if the server requests a certificate. If client doesn't send any certificates or if some aspect of the certificate chain was

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

unacceptable (e.g., it was not signed by a known, trusted CA) the server MAY at its discretion either continue the handshake without client authentication, or respond with a fatal handshake_failure alert.

PREMASTER & MASTER SECRET
Description: Exchange the necessary cryptographic parameters to allow the client and server to agree on a premaster secret.Generate a master secret from the premaster secret and exchanged random values If RSA is used, the client generates a 48-byte master secret, encrypts it using the public key from the server's certificate, and sends the result in an encrypted Master secret message. The master secret is expanded into a sequence of secure bytes, which is then split to : 1. client write MAC key 2. server write MAC key 3. client write encryption key 4. server write encryption key

VI. TLS RECORD PROTOCOL

The TLS Record Protocol is a layered protocol. At each layer, messages may include fields for length, description, and content. The Record Protocol takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result. Received data is decrypted, verified, decompressed, reassembled, and then delivered to higher-level clients.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

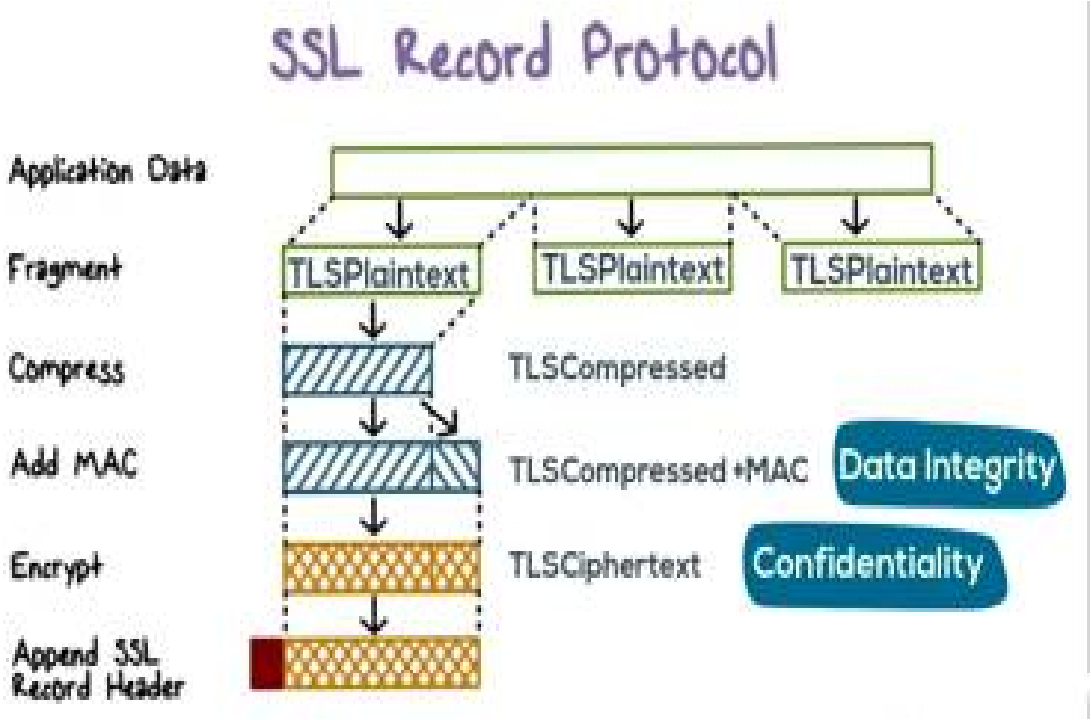

Figure 2. Message flow during Record Protocol

Key Calculation

The Record Protocol requires an algorithm to generate keys required by the current connection state from the security parameters provided by the handshake protocol. The master secret is expanded into a sequence of secure bytes, which is then split to a client write MAC key, a server write MAC key, a client write encryption key, and a server write encryption key. Each of these is generated from the byte sequence in that order. Unused values are empty.The TLS record layer receives uninterpreted data from higher layers in non-empty blocks of arbitrary size.

Master Secret

A 48-byte secret shared between the two peers in the connection.

Pseudocode defining parameters:

```
enum { server, client } ConnectionEnd;

enum { tls_prf_sha256 } PRFAlgorithm;
```

Fragmentation

The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of 2^14 bytes or less. Client message boundaries are not preserved in the record layer (i.e., multiple client messages of the same ContentTypeMAY be coalesced into a single TLSPlaintext record, or a single message MAY be fragmented across several records).

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

Pseudocode:

```
struct {
          uint8 major;
          uint8 minor;
} ProtocolVersion;

enum {
      change_cipher_spec(20), handshake(22),application_data(23)
} ContentType;

struct {
        ContentType type;
        ProtocolVersion version;
        uint16 length;
        opaque fragment[TLSPlaintext.length];
} TLSPlaintext;
```

Record Compression and Decompression

An algorithm to be used for data compression. This specification must include all information the algorithm requires to do compression.

The compression algorithm translates a TLSPlaintext structure into a TLSCompressed structure. Compression must be lossless and may not increase the content length by more than 1024 bytes. If the decompression function encounters a TLSCompressed.fragment that would decompress to a length in excess of 2^14 bytes, it MUST report a fatal decompression failure error.

Pseudocode defining these parameters:

```
enum { null(0), (255) } CompressionMethod;
```

Pseudocode:

```
struct {
     ContentType type;  /* same as TLSPlaintext.type */
     ProtocolVersion version;/* same as TLSPlaintext.version */
     uint16 length;
     opaque fragment[TLSCompressed.length];
} TLSCompressed;
```

MAC algorithm

An algorithm to be used for message authentication. This specification includes the size of the value returned by the MAC algorithm.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

Pseudocode defining these parameters:

```
enum {
      null, hmac_md5, hmac_sha1, hmac_sha256, hmac_sha384,
hmac_sha512
} MACAlgorithm;
```

Pseudocode for MAC generation:

```
MAC(MAC_write_key, seq_num
                   + TLSCompressed.type
                   + TLSCompressed.version
                   +TLSCompressed.length
                   + TLSCompressed.fragment);
```

```
   where "+" denotes concatenation.
```

Bulk encryption algorithm

An algorithm to be used for bulk encryption. This specification includes the key size of this algorithm, whether it is a block, stream, or AEAD cipher, the block size of the cipher (if appropriate), and the lengths of explicit and implicit initialization vectors (or nonces).

Pseudocode defining these parameters:

```
enum { null, rc4, 3des, aes }BulkCipherAlgorithm;

enum { stream, block, aead } CipherType;
```

Standard Stream Cipher

Stream ciphers convert TLSCompressed.fragment structures to and from stream TLSCiphertext.fragment structures. Note that the MAC is computed before encryption. The stream cipher encrypts the entire block, including the MAC.

Pseudocode:

```
stream-ciphered struct {
    opaque content[TLSCompressed.length];
    opaque MAC[SecurityParameters.mac_length];
} GenericStreamCipher;
```

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

CBC Block Cipher

For block ciphers (such as 3DES or AES), the encryption and MAC functions convert TLSCompressed.fragment structures to and from block TLSCiphertext.fragment structures.

Pseudocode:

```
block-ciphered struct {

     opaque content[TLSCompressed.length];
     opaque MAC[SecurityParameters.mac_length];
      uint8 padding[GenericBlockCipher.padding_length];
      uint8 padding_length;

 } GenericBlockCipher;
```

Description:

Padding that is added to force the length of the plaintext to bean integral multiple of the block cipher's block length.

The algorithms specified in Compression Method, PRF Algorithm, Bulk Cipher Algorithm, and MAC Algorithm may be added to.

Pseudocode defining security parameters:

```
struct {

 ConnectionEnd   entity;
    PRFAlgorithm    prf_algorithm;
    BulkCipherAlgorithm    bulk_cipher_algorithm;
CipherType    cipher_type;
    uint8         enc_key_length;
    uint8         block_length;
    uint8 mac_algorithm;
uint8  mac_length;
    uint8  mac_key_length;
    CompressionMethod      compression_algorithm;
   opaque  master_secret[48];

} SecurityParameters;
```

VII. CONCLUSION

For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors. The system is only as strong as the

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 9, September 2018

weakest key exchange and authentication algorithm supported, and only trustworthy cryptographic functions should be used. Short public keys and anonymous servers should be used with great caution. Implementations and users must be careful when deciding which certificates and certificate authorities are acceptable; a dishonest certificate authority can do tremendous damage.

REFERENCES

1. Tim Dierks and Eric Rescorla, Request for Comments: 5246 "The TLS Protocol Version 1.2", August 2008
2. TLS/SSL - Stanford CS Theory
3. Jelena Ćurguz, VULNERABILITIES OF THE SSL/TLS PROTOCOL
4. Wikipedia, Transport Layer Security
5. National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)" FIPS 197. November 26, 2001.
6. National Institute of Standards and Technology, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", NIST Special Publication 800-67, May 2004.
7. NIST FIPS PUB 186-2, "Digital Signature Standard", National Institute of Standards and Technology, U.S. Department of Commerce, 2000.
8. Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed- Hashing for Message Authentication", RFC 2104, February 1997.
9. Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
10. Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.
11. Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.
12. B. Schneier. "Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed.", Published by John Wiley & Sons, Inc. 1996.
13. NIST FIPS PUB 180-2, "Secure Hash Standard", National Institute of Standards and Technology, U.S. Department of Commerce, August 2002.
14. Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
15. Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.
16. McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.
17. Kent, S., "IP Authentication Header", RFC 4302, December 2005.