A Security Pattern for the Transport Layer Security (TLS) Protocol

Ajoy Kumar and Eduardo B. Fernandez
Department of Computer and Electrical Eng. and Computer Science
Florida Atlantic University
Boca Raton, FL , USA

Abstract
Transport Layer Security(TLS) is a connection-oriented protocol that provides a secure channel between a client and a server at the transport layer of the network. We present here a pattern that describes the security aspects of this protocol.

1. Introduction

Computer networks have a layered structure and we need to protect these layers against attacks that may compromise their security. Figure 1 shows the layers and the security protocols used at each of the layers. The Application layer has different protocols based on the type of application. The Transport layer uses TLS as the security protocol while the IP layer uses IPSec as the security protocol. Application protocols such as HTTP, LDAP, SOAP need to use the lower layers to support typical application tasks such as displaying web pages or running email services, but they may use their own version of security protocols such as HTTPS, LDAPS (Secure LDAP) and WSS (Web Service Security) respectively.

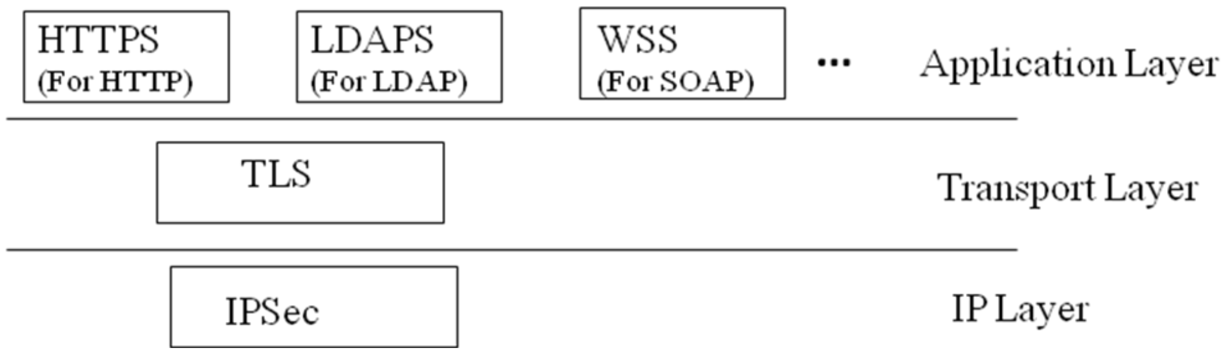

Figure 1. Network Layers and Security Protocols

TLS is a connection-oriented protocol that provides a secure channel between a client and a server at the transport layer. The protocol supports message confidentiality, data integrity, and client/server authentication. The TLS protocol also provides a means for the negotiation of security parameters, such as the encryption algorithms, encryption keys, hashing functions, etc., that are used to transmit data securely. We present here a pattern to describe its security properties. We assume that the reader is familiar with general network concepts and with the POSA template that we use to describe our pattern [Bus96]. This pattern is part of a set of patterns that describe network security mechanisms, which will be part of a catalog of security patterns [Fer13].

2. Transport Layer Security(TLS)

Intent
We need to provide a secure channel between a client and a server where application messages are being communicated over the Transport layer of the Internet. The client and the server need to be mutually authenticated.

Example
A bank customer may want to check his account balance online. The bank uses the Transport layer to transfer its confidential data. We need to protect this communication as this confidential data is vulnerable to attacks. The customer also has to make sure that the transactions are made with the bank and not with an impostor, while the bank may need to verify that this is a legitimate customer.

Context
Users using applications that exchange sensitive information, such as web browsers for e-commerce or similar activities. The **transport layer** provides end-to-end communication services for applications within a layered architecture of network components and protocols. The transport layer provides convenient services such as connection-oriented data stream support, flow control, and multiplexing.

Problem
The messages communicated between applications and servers at the transport layer are vulnerable to attack by intruders who may try to read or modify them. The server and the client may be impostors.

The solution to this problem is affected by the following **forces:**

- *Confidentiality and integrity:* The data transferred in the transport layer between the client and the server could be intercepted and read or modified illegally.
- *Authenticity:* Either the server or the client could be an impostor, which may allow security breaches. A Man in the Middle attack is also possible where an attacker poses as the client to the server and as the server to the client.
- *Flexibility:* The security protocol should be flexible and configurable to be able to handle new attacks.
- *Transparency.* The security measures of the protocol should be transparent to the users.
- *Configurability:* The protocol should allow the users to select different algorithms to provide different degrees of security.

Solution
Establish a cryptographic secure channel between the client and the server. Provide means for client and server to authenticate each other. The client and server can negotiate what cryptographic algorithms they will use.

Structure

Figure 2 shows a class diagram for the basic architecture of the TLS pattern. A **Client** requests some **Service** from the **Server**. The **TLS Protocol** conveys this request using an **Authenticator** to authenticate both the Server and the Client and creating a **Secure Channel** between the client and the server. The Authenticator and the Secure Channel are known patterns (See Related Patterns).

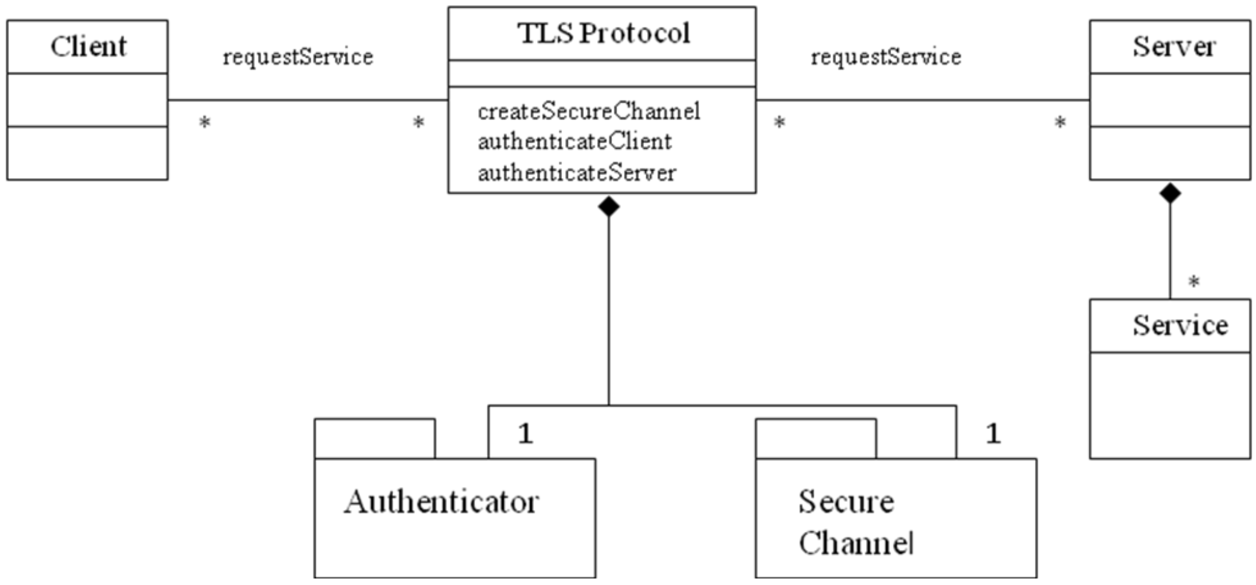

Figure 2. Class diagram for the TLS Protocol pattern

Dynamics

We describe the dynamic aspects of the TLS Pattern using a sequence diagram for the following use case:

Request a service (Figure 3):

Summary: A client requests a service and the TLS protocol authenticates the request and creates a secure channel.

Actors: Client, Server.

Precondition: The security parameters of the secure exchange have been predefined.

Description:

a) The client makes a service request to the server.

b) The TLS protocol authenticates the server to the client and the client to the server.

c) The TLS protocol creates a secure channel between the server and the client.

Alternate Flows:

1) The authentication can fail.
2) The creation of a secure channel can fail.

Postcondition: The server accepts the request and grants the service.

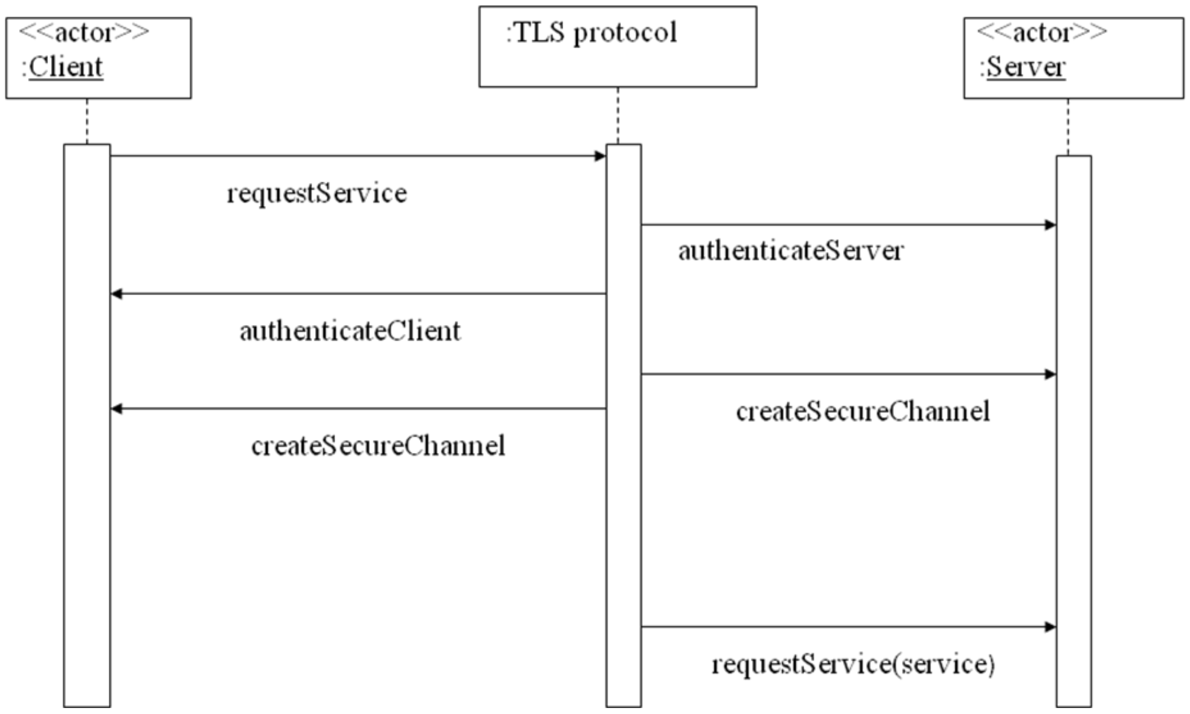

Figure 3: Sequence diagram for UC: Request a Service

Implementation

One of the protocols that is dominant today for providing security at the transport layer is the Secure Sockets Layer (SSL) Protocol. The SSL protocol is a transport layer security protocol that was developed and proposed by Netscape Communications in the 1990s. The Transport Layer Security (TLS) Protocol is an IETF version of the SSL protocol, which has become a standard [Yas04]. A good amount of implementation advice can be found in [Sel12].

The TLS protocol is mainly partitioned into two protocol layers, the *TLS Record Protocol* and the *TLS Handshake Protocol,* executing above the TCP (Transport Layer) protocol as shown in Figure 4 [Elg06, Sta12]. There are other minor protocols at the handshake protocol layer such as the *Cipher Change Protocol*, *Alert Protocol*, and *Application Protocol*.

TLS Handshake Protocol	TLS Cipher change Protocol	Alert Protocol	Application Protocol
TLS Record Protocol			
TCP			
IP			

Figure 4. TLS layers

Record Protocol
The TLS Record Protocol provides encryption and message authentication for each message. A connection is created using symmetric cryptography data encryption. 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). Messages include a message integrity check using a keyed MAC, computed using hash functions [Sta03].

Handshake Protocol
A TLS *handshake* supplies the authentication and key exchange operations for the TLS protocol. The security state agreed upon in the handshake is used by the TLS Record Protocol to provide session security. This 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 any data. The TLS Handshake Protocol provides connection security where the peers' identities can be authenticated using asymmetric cryptography. This authentication can be made optional, but is generally required for at least one of the peers.

A TLS session is an association between a client and a server, created by the handshake protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

A session state is defined by the following parameters:

- *Session identifier*. It is generated by the server to identify a session with a chosen client.
- *Peer certificate*. X.509 certificate of the peer.
- *Compression method*. A method used to compress data prior to encryption.
- *Algorithm specification or CipherSpec*. Specifies the encryption algorithm that encrypts the data and the hash algorithm used during the session.
- *Master secret*: 48-byte data being a secret shared between the client and server, "it *is resumable*": a flag indicating whether the session can be used to initiate new connections

The Handshake protocol consists of the following four phases:

a) In the first phase, an initial connection is established which starts the negotiation. The client and server exchange hello messages that are used to establish security parameters (as defined above) used in the TLS session and settings used during the handshake, such as the key exchange algorithm.

b) During the second phase (authentication), the server sends a Certificate message to the client that may include a server certificate when an RSA key exchange is used, or Diffie-Hellman parameters when a Diffie-Hellman key exchange is used. The server may also request a certificate from the client, using the certificateRequest message.

c) During the third phase, the client, if asked, may send its certificate to the server in a Certificate message along with a certificateVerify message so that the server can verify certificate ownership if the server requested a client certificate during the second phase. This phase includes the establishment of the security parameters such as the encryption key. The client must send either a pre-master secret encrypted using the server's public key, or public Diffie-Hellman parameters in the clientKeyExchange message so that the client and server can compute a shared master secret.

d) In the fourth phase, the client and server finish the handshake, which implies that the client and server are mutually authenticated and have completed the required key exchange operations.

Structure and Dynamics of the Handshake Protocol

We describe the structure of the Handshake protocol using the class diagram in Figure 5. The **Client** requests for a service from the **Server** at the Transport layer. The TLS **Handshake Protocol** uses **Certificate(s)** to mutually authenticate the Client and the Server and does the clientKeyExchange once the authentication is done.

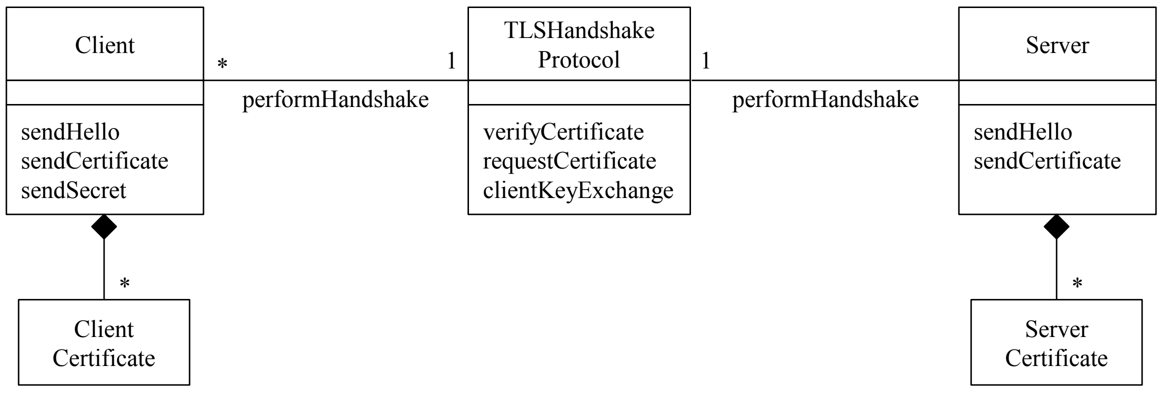

Figure 5: Class diagram for the TLS handshake protocol.

We describe the dynamic aspects of the TLS Handshake using a sequence diagram (Figure 6).

Summary: A TLS *handshake* supplies the authentication and key exchange operations for the TLS protocol.

Actors: Client, Server.

Precondition: The client has made a request for a service from the host server and an initial connection has already been established. The client and server need to have a digital certificate, issued by some Certificate Authority,

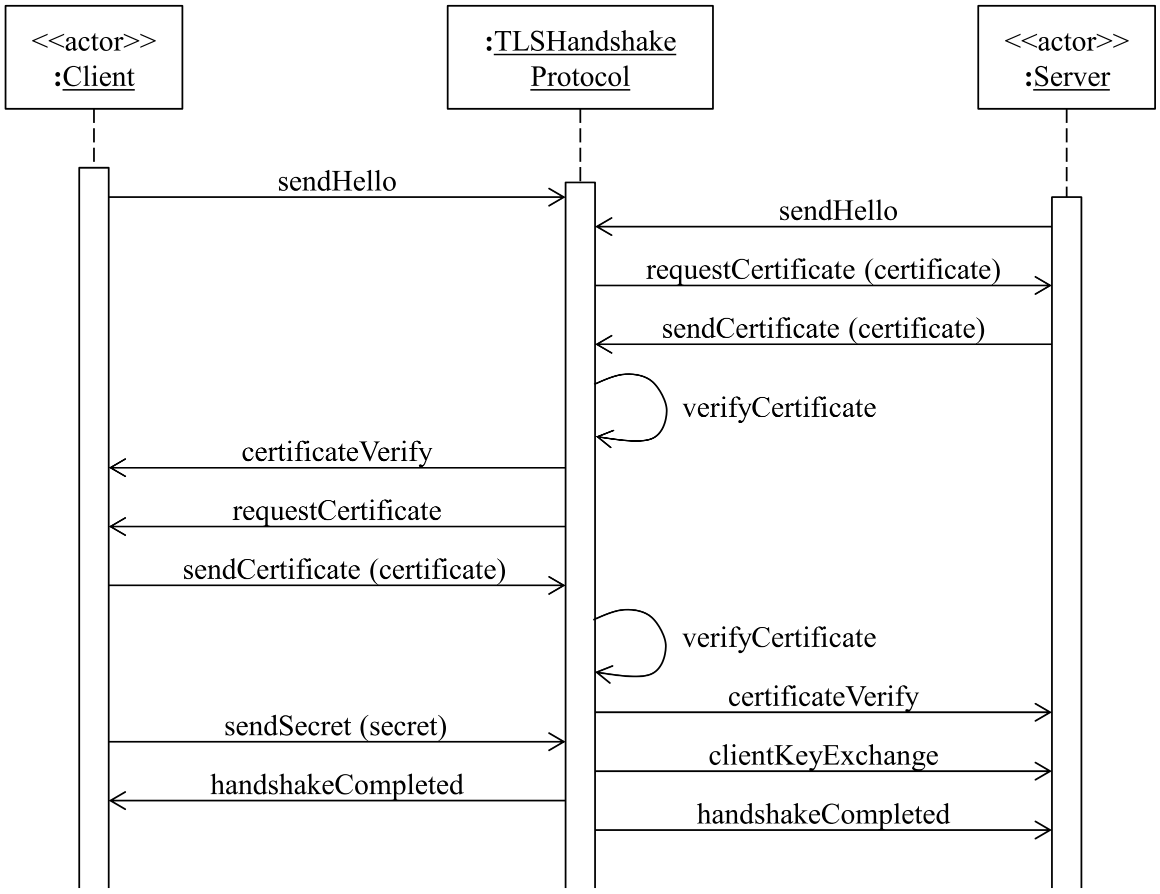

Figure 6: Sequence diagram for the TLS handshake use case

Description:

a) The client and server exchange initial Hello messages.

b) The protocol requests the certificate from the server and the server sends the certificate.

c) The server certificate is verified

d) The server requests the certificate from the client (optional).

e) If asked, client sends the certificate to the server.

f) The client certificate is verified.

g) The client sends the predefined secret encrypted using the server's public key which is the client key exchange.

h) The client and the server complete mutual handshake and the initial encryption parameters.

Alternate Flows:

1) Authentication of the server or client can fail. A certificate can be expired or outdated.

2) The client could lose the encryption key while exchanging with the server.

Postcondition: Client and Server can start exchanging data at the transport layer.

The other minor protocol layers from Figure 4 are discussed below:

Cipher Change Protocol
This protocol signals transitions in cipherSpec which is a session parameter explained above.

Alert Protocol
This protocol raises alerts for the communication. This record should normally not be sent during normal handshaking or application exchanges. However, this message can be sent at any time during the handshake and up to the closure of a TLS session. If this record is used to signal a fatal error, the session will be closed immediately after sending this record. If the alert level is flagged as a warning, the remote partner can decide or not to close the session.

Application Protocol
Now the handshake is completed and the Application protocol is enabled. This marks the start of data exchange between the server and the client.

Variants
WTLS: A modified version of TLS, called WTLS (Wireless TLS protocol) has been used in mobile systems. WTLS is based on TLS and is similar in some aspects [Bad04]. WTLS has been superseded in the WAP (Wireless Aplication Protocol) 2.0 standard by the End-to-end Transport Layer Security Specification.

MultipleTLS (MTLS): This is an application-level protocol running over the TLS Record protocol. The MTLS provides application multiplexing over a single TLS session. Therefore, instead of associating a TLS session with each application, this protocol allows several applications to protect their communication over a single TLS session [Bad09].

Some different versions of TLS are given below:
TLS1.0: TLS 1.0 was first defined in January 1999 as an upgrade to SSL Version 3.0 and is an IETF version of SSL. The differences between this protocol and SSL 3.0 are not large, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate. TLS 1.0 does include a means by which a TLS implementation can downgrade the connection to SSL 3.0, but this weakens security.

TLS1.1: TLS 1.1, defined in April 2006, is an update of TLS version 1.0. Significant differences include:

- Added protection against Cipher Block Chaining (CBC) attacks. (In CBC mode, each block of plaintext is XORed with the previous cipher text block before being encrypted.)
    - The implicit Initialization Vector (IV) was replaced with an explicit IV.
    - Change in handling of padding errors.
- Support for registration of parameters.

TLS1.2: This is a revision of the TLS 1.1 protocol from August 2008, which contains improved flexibility, particularly for negotiation of cryptographic algorithms. The major changes are:

- The MD5/SHA-1 combination in the pseudorandom function (PRF) has been replaced with cipher-suite-specified PRFs. All cipher suites in this document use P_SHA256.
- The MD5/SHA-1 combination in the digitally-signed element has been replaced with a single hash. Signed elements now include a field that explicitly specifies the hash algorithm used.
- Substantial cleanup to the client's and server's ability to specify which hash and signature algorithms they will accept.
- Addition of support for authenticated encryption with additional data modes.
- Tightening up of a number of requirements.
- Verification of data length now depends on the cipher suite (default is still 12) [Wik].

Known Uses

- Mozilla Firefox versions 2 and above, support TLS 1.0 [Moz].
- Internet Explorer(IE) 8 in Windows 7 and Windows Server 2008 support TLS 1.2 [MS].
- Presto 2.2, used in Opera 10, supports TLS [Rut].

Example Resolved
When a request is made to the bank server by the online client at the transport layer the bank server is authenticated to the customer, the customer is authenticated to the server and a secure channel is created between them. Now the client knows that the online bank transactions are secure.

Consequences
This pattern has the following advantages:

- *Authentication:* Both client and server can be mutually authenticated. Man in the middle attacks can be prevented by mutual authentication.

- *Security:* A secure channel is established between the server and the client, which can provide data confidentiality and integrity for the messages sent. We could add a logging system for the client at its end point for future audits.
- *Flexibility:* We can easily change encryption algorithms and authentication protocols.
- *Transparency.* The users don't need to perform any operation to have a secure channel established.

This pattern has the following disadvantages:

- The protocol adds some overhead to the communication.
- SSL/TLS is a two-party protocol, thus it is not designed to handle multiple parties. However, as described earlier, the MTLS variant can handle multiple parties.

Related Patterns

- The Authenticator describes how to mutually authenticate a Client and a Server [Bro99].
- The Secure Channel describes a cryptographic channel used to communicate secure data [Bra98]

Acknowledgements

We thank our shepherd Jaap Kabbedijk for his useful comments that significantly improved the paper.

References

[Bad04] Mohamad Badra, Ahmed Serhrouchni and Pascal Urien. "A lightweight identity authentication protocol for wireless networks" **,** *Computer Communications* Volume 27, Issue 17, 1 November 2004.

[Bad09] Mohamad Badra and Hajjeh, Internet-Draft, (D)TLS Multiplexing, April 2009 http://tools.ietf.org/html/draft-badra-hajjeh-mtls-05

[Bra98] A. Braga, C. Rubira, and R. Dahab, "Tropyc: A pattern language for cryptographic object-oriented software". *Chapter 16 in Pattern Languages of Program Design 4* (N. Harrison, B. Foote, and H. Rohnert, Eds.). 1998.

[Bro99] F.L. Brown and E.B. Fernandez, "The Authenticator pattern", *Procs. of PLoP99.*

[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal., *Pattern-oriented software architecture,* Wiley 1996.

[Elg06] Ashraf Elgohary, Tarek S. Sobh, M. Zaki. "Design of an enhancement for SSL/TLS protocols**"** *Computers & Security* Volume 25, Issue 4, June 2006

[Fer13] E.B.Fernandez, "*Security patterns in practice: Building secure architectures using software patterns*", to appear in the Wiley Series on Software Design Patterns.

[Kum10] Ajoy Kumar and E.B.Fernandez, "Security patterns for Virtual Private Networks", *8th Latin American Conference on Pattern Languages of Programs (SugarLoafPLoP 2010)*, Salvador, Bahia, Brazil, Sept 23-26, 2010

[Moz] Mozilla *Newsgroup: mozilla.dev.tech.crypto*
http://www.mozilla.org/projects/security/pki/nss/ssl/

[MS] Microsoft Forums: What is TLS/SSL
http://technet.microsoft.com/en-us/library/cc784450(v=WS.10).aspx

[Rut] Wikipedia: GNU Free Documentation License, 2009
http://mjrutherford.org/files/2009-Spring-COMP-4704-TLS-Wikipedia.pdf\

[Sel12] L. Seltzer, "Best practices and applications of TLS/SSL", white paper, Symantec Corp., 2012.

[Sta12] W. Stallings and L. Brown, *Computer security: Principles and practice* (2nd Ed.), Pearson 2012.

[Wik] Proposed Standard: The Transport Layer Security (TLS) Protocol Version 1.2
http://wiki.tools.ietf.org/html/rfc5246

[Yas04] Alec Yasinsac and Justin Childs. "Formal analysis of modern security protocols" *Information Sciences* Volume 171, Issues 1–3, 4 March 2005