Handshaking Mechanism in E-Business Applications

D. Madhavi
Asst Professor
VR Siddhartha Engineering
College, Kanuru, Vijayawada

M. Ashok Kumar
Asst Professor
VR Siddhartha Engineering
College, Kanuru, Vijayawada

C.Narasimham
Asst Professor
VR Siddhartha Engineering
College, Kanuru, Vijayawada

ABSTRACT

Secure Session Layer (SSL) and Transport Layer Security (TLS) are the two secure layer protocols in all of current web applications on a network. This paper focuses on SSL, TLS and how handshaking mechanism has been implemented in both SSL and TLS. Further, describes about the generation of keys and certificates.

Keywords

Secure socket layer, Transport layer security, Key Exchange, Certificate Request, Change cipher spec.

1. INTRODUCTION

Most users spend their online time performing PIM (Personal Information Management) functions such as Email, Calendar and Intranet access. As many specialist applications are now available web enabled, SSL is an attractive solution. Web enabled applications such as Email (Outlook and Notes), CRM (Customer Relationship Management) and ERP (Employee Resource Planning) are available to browser-based clients including suppliers and customers. What is needed is some form of access control and logging in order to restrict users to the resources they are allowed to access. If this can also be included, it becomes possible to provide secure connectivity to all forms of remote users as long as they have a device with a secure browser that meets certain criteria.

1.1 SSL (SECURE SOCKET LAYER)

SSL was developed by Netscape as a mechanism to provide secure communication over an insecure medium such as the Internet. Netscape published the SSL specification and provided support in both their browsers and server software [3]. SSL became the defacto tool for secure web transactions. The layered diagram of SSL and TLS operated on top of TCP/IP and operates at Presentation layer as shown in the below figure 1.This protocol to transmit information privately, ensures message integrity, and guarantees the server identity. SSL works mainly through using public/private key encryption on data. SSL is now an IETF (Internet Engineering Task Force) standard and is referenced as Transport Layer Security or TLS and is defined in RFC 2246 (http://www.ietf.org/rfc/rfc2246.txt). SSL uses TCP to provide a reliable end-to-end service [2].

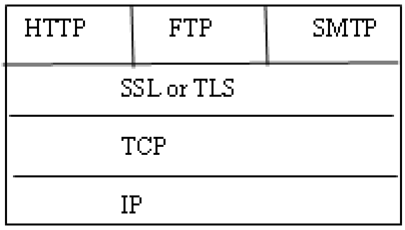


Figure 1: Trasport Level

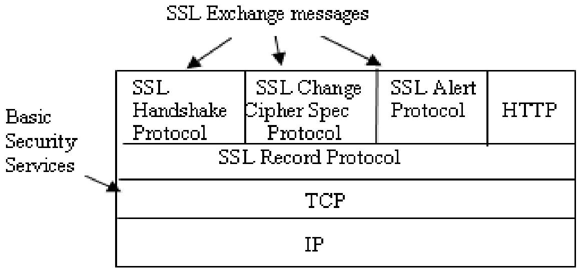


Figure 2: SSL Protocol Stack

1.2 TLS (TRASPORT LAYER SECURITY)

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide communication security over the Internet [4]. TLS and SSL encrypt the segments of network connections above the Transport Layer, using asymmetric cryptography for privacy and a keyed message authentication code for message reliability. TLS is an IETF (Internet Engineering Task Force) standard standards track protocol, last updated in RFC 5246 and is based on the earlier SSL specifications developed by Netscape Communications, but there are enough differences between them in order to be considered as different protocols [11]. The initial versions of SSL are SSL 1.0, 2.0 and 3.0.

TLS 1.0 =SSL 3.1, TLS 1.1 =SSL 3.2 and TLS 1.2 =SSL 3.3[14].

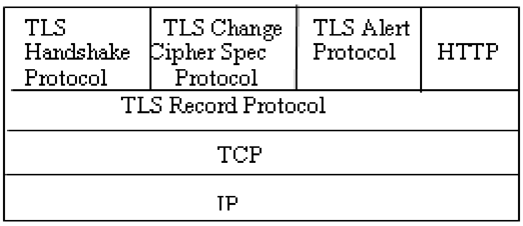


Figure 3: TLS Protocol Stack

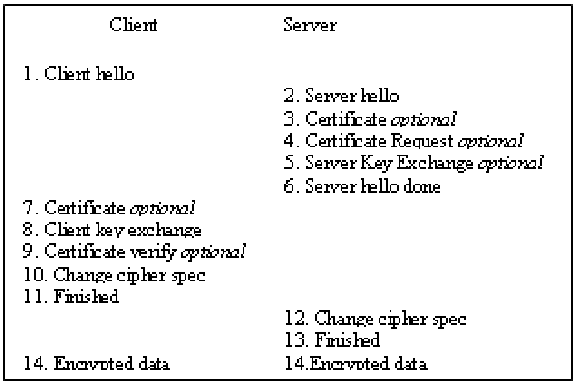
TLS is composed of four sub protocols: Handshake, which allows two agents to establish a session; record, which exchanges application data; Alert, which reports about errors in the session; and Change Cipher Spec, which copies the negotiated settings to the actual settings. In this paper we will focus on TLS Handshake.

2. HANDSHAKING IN NETWORKS

A handshake in networking happens when a computer wants to talk to another computer or when packets of data are exchanged between them. Before anything is sent and received the hand shake takes place. Handshaking is created to be a fast prelude to other network activity; many Internet handshaking protocols are created to be sending out in nanoseconds of PC CPU time and under milliseconds of network time. A connection between two hosts across a network can consist of many sessions over time, each called an incarnation. A connection is synchronized using a connection establishment protocol to allow a reliable exchange of data. A handshake mechanism is employed to establish the connection. A three-way handshake protocol exists for formal model of connection management regardless of the handshake level.

2.1 Handshaking in SSL

SSL handshake is the exchange of information between the client and the server prior to sending the encrypted message [6]. This section provides more detail. The "SSL Messages" figure below shows the sequence of messages that are exchanged in the SSL handshake [5]. Messages that are only sent in certain situations are noted as optional [12]. Each of the SSL messages is described in the following figure 4:



Client	Server
1. Client hello	
	2. Server hello
	3. Certificate *optional*
	4. Certificate Request *optional*
	5. Server Key Exchange *optional*
	6. Server hello done
7. Certificate *optional*	
8. Client key exchange	
9. Certificate verify *optional*	
10. Change cipher spec	
11. Finished	
	12. Change cipher spec
	13. Finished
14. Encrypted data	14.Encrypted data

Figure 4: SSL Handshaking

2.1.1 Protocol description

Client hello - The client sends the server information including the highest version of SSL it supports and a list of the cipher suites it supports. (TLS 1.0 is indicated as SSL 3.1.) The cipher suite information includes cryptographic algorithms and key sizes.

Server hello - The server chooses the highest version of SSL and the best cipher suite that both the client and server support and sends this information to the client.

Certificate - The server sends the client a certificate or a certificate chain. A certificate chain typically begins with the server's public key certificate and ends with the certificate authority's root certificate. This message is optional, but is used whenever server authentication is required [10].

Certificate request - If the server needs to authenticate the client, it sends the client a certificate request. In Internet applications, this message is rarely sent.

Server key exchange - The server sends the client a server key exchange message when the public key information sent in 3) above is not sufficient for key exchange.

Server hello done - The server tells the client that it is finished with its initial negotiation messages.

Certificate - If the server requests a certificate from the client in Message 4, the client sends its certificate chain, just as the server did in Message 3 [13].

Client key exchange - The client generates information used to create a key to use for symmetric encryption. For RSA, the client then encrypts this key information with the server's public key and sends it to the server.

Certificate verify - In internet applications, this message is rarely sent. Its purpose is to allow the server to complete the process of authenticating the client. When this message is used, the client sends information that it digitally signs using a cryptographic hash function. When the server decrypts this information with the client's public key, the server is able to authenticate the client.

Change cipher spec - The client sends a message telling the server to change to encrypted mode. Finished - The client tells the server that it is ready for secure data communication to begin.

Change cipher spec - The server sends a message telling the client to change to encrypted mode.

Finished - The server tells the client that it is ready for secure data communication to begin. This is the end of the SSL handshake.

Encrypted data - The client and the server communicate using the symmetric encryption algorithm and the cryptographic hash function negotiated in Messages 1 and 2, and using the secret key that the client sent to the server in Message 8.

If the parameters generated during an SSL session are saved, these parameters can sometimes be re-used for future SSL sessions. Saving SSL session parameters allows encrypted communication to begin much more quickly [8].

2.2 Handshaking in TLS

A relationship is established between two parties in TLS by using a handshake exchange [9]. This involves a series of messages sent between the parties in a specific order, as summarized in the Figure 5. TLS Handshake negotiates the elements that configure a session between a client and a server: a session identifier, the agent’s certificate, a compression method, some secret data (Master Secret) for generating session keys, a cipher mode CipherSpec, and a flag that indicates if the session can be resumable [5].

Consider the case when a client wants to establish a new session with the server, and they have to negotiate the session settings. This situation is represented in Figure 5, where the messages in brackets represent optional messages. The protocol performs the following actions: The agents exchange hello messages in order to negotiate the best combination of settings [1]. These settings are the agent protocol version, a session identifier, a Cipher Suite and a Compression method. Two random values are generated: ClientHello.random and ServerHello.random. The agents exchange some secret data in a secure way. The client generates some secret data that is sent to the server in the ClientKeyExchange message. The agents authenticate each extreme if it is required by the selected Cipher Suite. Then, the agent must send a Certificate message. If a certificate allows a digital sign and the owner is the client, it sends a Certificate Verify message. If the owner is the server, it sends a ServerKeyExchange message. They finish the handshake with a ChangeCipherSpec message and a finished message. The Finished message includes all the previous handshake messages and allows an agent to verify whether the session parameters are correct or not [7].

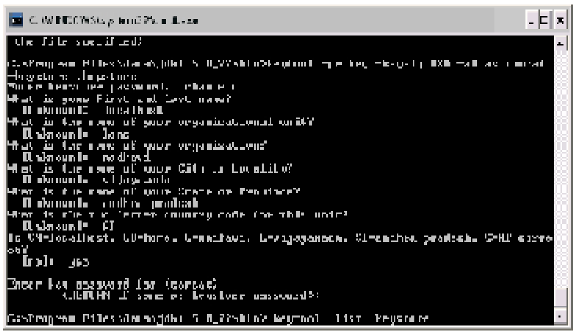
Client	Server
1. Client hello	
	2. Server hello
	3. Certificate *optional*
	4. Server Key Exchange *optional*
	5. Certificate Request *optional*
	6. Server hello done
7. Certificate *optional*	
8. Client key exchange	
9. Certificate verify *optional*	
10. Change cipher spec	
11. Finished	
	12. Change cipher spec
	13. Finished
14. Encrypted data	14.Encrypted data

Figure 5: TLS Handshaking

3. IMPLEMENTATION

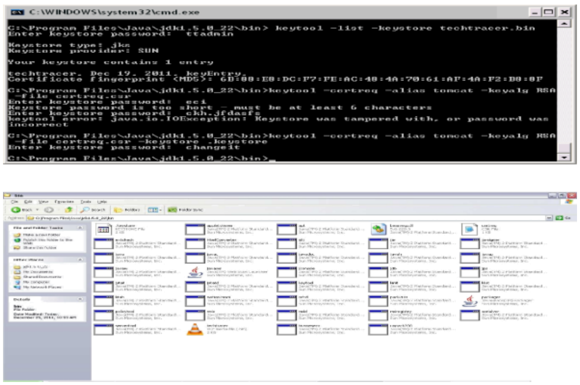
3.1 Secure Session Layer

The key generation and certificates in SSL has been implemented and that has been shown in the following figures 6 and figure 7 respectively.



```
 the file specified)

C:\Program Files\Java\jdk1.5.0_22\bin>keytool -genkey -keyalg RSA -alias tomcat
-keystore .keystore
Enter keystore password:  changeit
What is your first and last name?
  [Unknown]:  localhost
What is the name of your organizational unit?
  [Unknown]:  [illegible]
What is the name of your organization?
  [Unknown]:  madhavi
What is the name of your City or Locality?
  [Unknown]:  vijayawada
What is the name of your State or Province?
  [Unknown]:  andhra pradesh
What is the two-letter country code for this unit?
  [Unknown]:  AP
Is CN=localhost, OU=[illegible], O=madhavi, L=vijayawada, ST=andhra pradesh, C=AP corre
ct?
  [no]:  yes

Enter key password for <tomcat>
        (RETURN if same as keystore password):

C:\Program Files\Java\jdk1.5.0_22\bin>keytool -list -keystore
```

Figure 6 Key generation in SSL

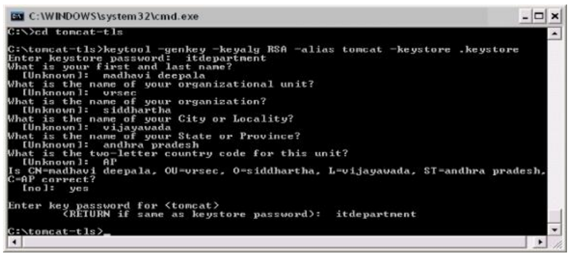


```
C:\Program Files\Java\jdk1.5.0_22\bin> keytool -list -keystore techtracer.bin
Enter keystore password:  ttadmin

Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

techtracer, Dec 19, 2011, keyEntry,
Certificate fingerprint (MD5): 6B:88:E8:DC:F7:FE:AC:48:4A:70:61:AF:4A:F2:B8:8F

C:\Program Files\Java\jdk1.5.0_22\bin>keytool -certreq -alias tomcat -keyalg RSA
 -file certreq.csr
Enter keystore password:  eci
Keystore password is too short - must be at least 6 characters
Enter keystore password:  ckh.jfdasfs
keytool error: java.io.IOException: Keystore was tampered with, or password was
incorrect

C:\Program Files\Java\jdk1.5.0_22\bin>keytool -certreq -alias tomcat -keyalg RSA
 -file certreq.csr -keystore .keystore
Enter keystore password:  changeit

C:\Program Files\Java\jdk1.5.0_22\bin>_
```

Figure 7: Certificate generation in SSL

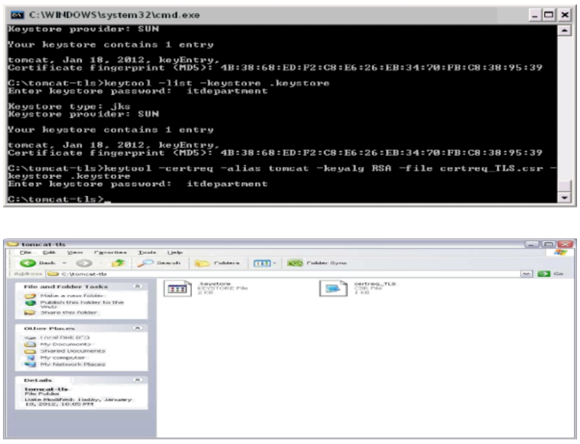
3.2 Transport Layer Security

The key generation and certificates in TLS has been implemented and that have been shown in the following figures 8 and figures 9 respectively.



```
C:\>cd tomcat-tls

C:\tomcat-tls>keytool -genkey -keyalg RSA -alias tomcat -keystore .keystore
Enter keystore password:  itdepartment
What is your first and last name?
  [Unknown]:  madhavi deepala
What is the name of your organizational unit?
  [Unknown]:  vrsec
What is the name of your organization?
  [Unknown]:  siddhartha
What is the name of your City or Locality?
  [Unknown]:  vijayawada
What is the name of your State or Province?
  [Unknown]:  andhra pradesh
What is the two-letter country code for this unit?
  [Unknown]:  AP
Is CN=madhavi deepala, OU=vrsec, O=siddhartha, L=vijayawada, ST=andhra pradesh,
C=AP correct?
  [no]:  yes

Enter key password for <tomcat>
        (RETURN if same as keystore password):  itdepartment

C:\tomcat-tls>_
```

Figure 8: Key Generation in TLS



```
Keystore provider: SUN

Your keystore contains 1 entry

tomcat, Jan 18, 2012, keyEntry,
Certificate fingerprint (MD5): 4B:38:68:ED:F2:C8:E6:26:EB:34:70:FB:C8:38:95:39

C:\tomcat-tls>keytool -list -keystore .keystore
Enter keystore password:  itdepartment

Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

tomcat, Jan 18, 2012, keyEntry,
Certificate fingerprint (MD5): 4B:38:68:ED:F2:C8:E6:26:EB:34:70:FB:C8:38:95:39

C:\tomcat-tls>keytool -certreq -alias tomcat -keyalg RSA -file certreq_TLS.csr -
keystore .keystore
Enter keystore password:  itdepartment

C:\tomcat-tls>_
```

Figure 9: Certificate generation in TLS

4. CONCLUSION

Finally, our conclusions are made based on the observations that the working principle of handshaking mechanism in both SSL and TLS is similar. They differ in the way the connection has been established. In the connection establishment process, we experimented the generation of keys and certificates for their secure communication establishment in E-Business.

5. REFERENCES

[1] Chris Crall, Mike Danseglio, and David Mowers, “SSL/TLS in Windows Server 2003” Microsoft Corporation publication 31 July, doi: cc781800 (v=ws.10).

[2] Dierks T. and Allen, C. “The TLS Protocol Version 1.0”, Internet Engineering Task Force publication of January 1999, ietf/rfc2246.

[3] D. Wagner and B. Bruce Schneier, “Analysis of the SSL 3.0 Protocol”, in the Second USENIX Workshop on Electronic Commerce Proceedings (University of California, USA, 1996); Berkeley: USENIX Press. 29-40.

[4] Erik Kangas, “SSL versus TLS“; LuxSci secure form 10 November, 2008: FYI blog.

[5] Holly Lynne McKinley, “SSL and TLS: A Beginners Guide” as part of Information Security Reading Room ©SANS Institute 2003, GSEC Practical v.1.4b.

[6] Llanos Tobarra, Diego Cazorla, Fernando Cuartero and Gregorio Díaz (2008) Formal verification of SSL Protocol , in International Conference on Enterprise Information Systems (Barcelona,spain,2008); Barcelona: fi_1269425980-tobarra_iceis08 .

[7] Llanos Tobarra, Diego Cazorla, Fernando Cuartero and Gregorio Díaz, “Formal verification of TLS handshake and extensions for wireless networks”, in IADIS International Conference Applied Computing (*Albacete*, Spain, 2009); *Albacete*: ISBN: 972-8924-09-7.

[8] Nitinpai, Thawte, “Setting up SSL on Tomcat in 3 easy steps: Free SSL Product and Technical Guide”, 12th September 2007. Techtracker 2.0.

[9] Paulson, L. C. 1999. Inductive Analysis of the Internet Protocol TLS, ACM Transactions on Information and System Security, 2:332–351

[10] Peter Birk, Keys Botzum. SSL, certificate, and key management enhancements for even stronger security in Web Sphere Application Server V6.1, Technical *journal of IBM Web Sphere* Developer publication 6 December, 2006. doi: 0612_birk

[11] Slake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and Wright, T. 2003. Transport Layer Security Extensions, Internet Engineering Task Force Technical Report, ietf/rfc3546

[12] Symantec. 1995. SSL information Center-About SSL Certificate Licensing; USA: VeriSign organization [WWW document]

[13] VeriSign; Beginners guide to SSL Certificates; SSL Information Center: VeriSign organization [WWW document, October 2011)

[14] Wikipedia. 2008. Transport_Layer_Security, Historic document in RFC, IETF publication 1 November 2008.