ENHANCED SECURITY IN SECURE SOCKET LAYER 3.0 SPECIFICATION

Meenu
meenucs@mmmec.net

Prabhat Kumar Pankaj
prabhat.cse.mmmec@gmail.com

Tarkeshwar Nath
tkn_001@gmail.com

Computer Science & Engineering Department.
M.M.M. Engineering. College Gorakhpur-273010 (U P) India

ABSTRACT

We address the issue network security.All the bank provides 128 bit ssl encryption in two way communication between client and server.In this paper we can tell you idea behind 256 ssl encryption.By which two way communicaton is much secure(ie: using of secure socket layer 3.0)

INTRODUCTION

The Secure Sockets Layer was originally developed (1994) by Netscape in order to secure http communications. Version 3 of SSL was released in 1995.

It is what we think of when we say SSL Slight variation became Transport Layer Security (TLS) and was accepted by the IETF in 1999. TLS is backward compatible with SSLv3. TCP provides a reliable end-to-end service. SSL consists of two sub layers:

1. SSL Record Protocol (where all the action takes place)
2. SSL Management: (Handshake/Cipher Change/ Alert Protocols)

An SSL Session is an association between a client and a server (created by the Handshake Protocol). There are a set of security parameters associated with each session

An SSL Connection is a peer-to-peer relationship, and is transient. There may be many connections associated with one session. The same security parameters may apply to many connections.

Session Security Parameters:

1. Session Identifier
2. Peer Certificate: X.509v3 certificate of the peer
3. Compression: Optional algorithm used to compress data
4. Cipher Specs: Encryption Algorithm (3DES, AES, etc.) and hash algorithm (MD5, SHA-1)
5. Master Secret: 48-byte secret shared between client and server.

. **OBJECTIVE**

The Web has become the visible interface of the Internet. Many corporations now use the Web for advertising, marketing and sales. Web servers might be easy to use but…

Complicated to configure correctly and difficult to build without security flaws. They can serve as a security hole by which an adversary might be able to access other data and computer systems.

	Threats	Consequences	Countermeasures
Integrity	**Modification of Data** **Trojan horses**	**Loss of Information** **Compromise of Machine**	**MACs and Hashes**
Confidentiality	**Eavesdropping** **Theft of Information**	**Loss of Information** **Privacy Breach**	**Encryption**
DoS	**Stopping** **Filling up Disks and Resources**	**Stopped Transactions**	
Authentication	**Impersonation** **Data Forgery**	**Misrepresentation of User** **Accept false Data**	**Signatures, MACs**

LANGUAGES AND ALGORITHM USED

LANGUAGES:

1. Java SDK 6.
2. Swing Class in javax.swing.*;
3. Cryptography Class javax.security.*;
4. Net Beans IDE 5.5

ALGORITHM:

This document specifies Version 3.0 of the Secure Sockets Layer (SSL V3.0) protocol, a security protocol that provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

ENCRYPTION:

In the encryption block of the project module that is:

First of all, three files has to be inputed one for encryption, other for key, and the last for decryption.

Then read the key file in order to produce a secret key from Secret Key Class, and then pass the secret key, file to encrypt and the output file to store the cipher text.

In order to complete the above task read key and encrypt functions are used. Read key function is used to convert the inputed key to the key used in Triple DES or DESede.

And then Cipher Output Stream is used to output the cipher data from inputed plain text in encrypt function, TripleDES encryption is used from the Cipher Class of the JDK 1.6.0 in javax.security, and javax.crypto.

Then module of SHA and DSA for Simple Hash Algorithm and Digital Signature Algorithm respectively is used to more secure the cipher text through the process of hashing and digital signing.

This module is done through MakeSignature.java in which file is digitally signed and it's hash value is calculated through the algorithm available in java.security; and hence the sequence of SSL Encryption is completed in this manner.

STRUCTURE OF THE PROGRAM

MODULE DESCRIPTION

There are only five modules in this project:

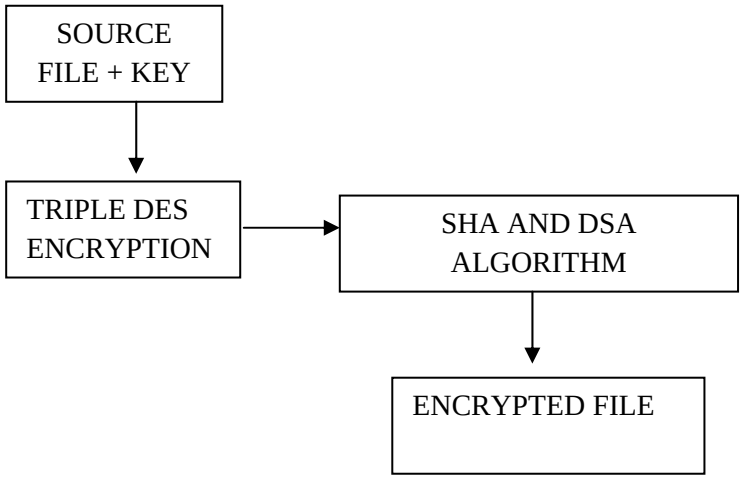
index.java -This is the main module of the project, it is used to display messages while execution, encryption and decryption the file. It will ask the user to choose the file and to input the key for the file for encryption. Then it can also be used in order to decrypt the file in this regard user has to choose the encrypted file and will have to enter the correct key for correct decryption it will be like authentication for decryption. This same form can also be used for transmitting the file in local network as well as on the Internet and can also be used for adding and verifying digital signature to the file.

MakeSignature.java – This is the file which is used to add digital signature to the selected file and for generating public key and private key.

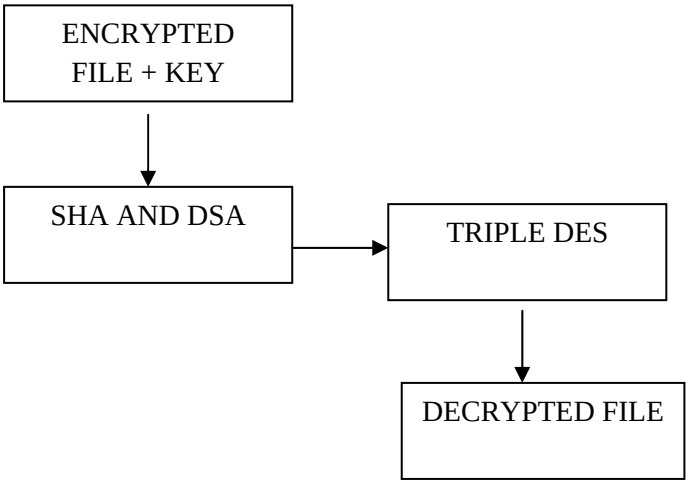
VerifySignature.java – This is the file which is used to verify the signature of the already signed file and after verifying it prints the file report as true or false.

LOGIC DIAGRAM

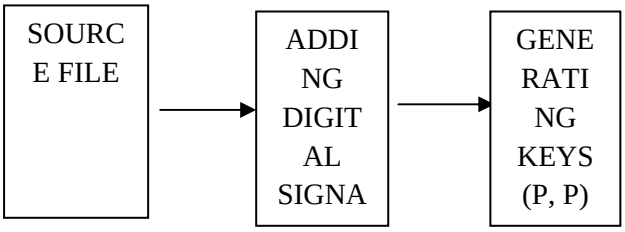
For Encryption:



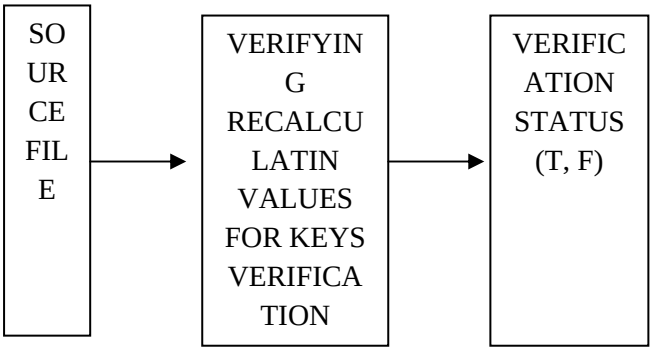
For Decompression:



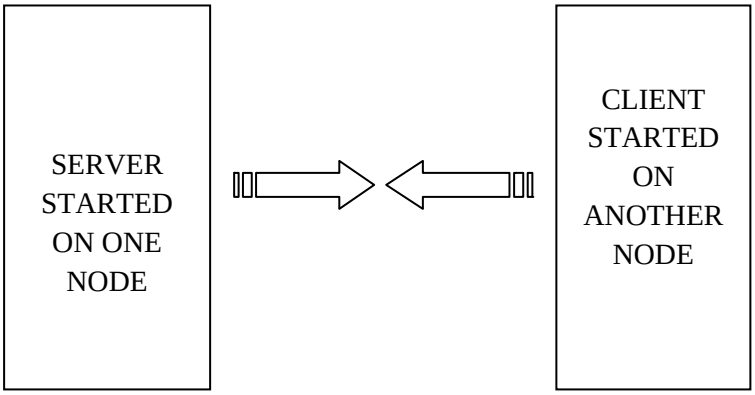
For Adding Digital Signature:



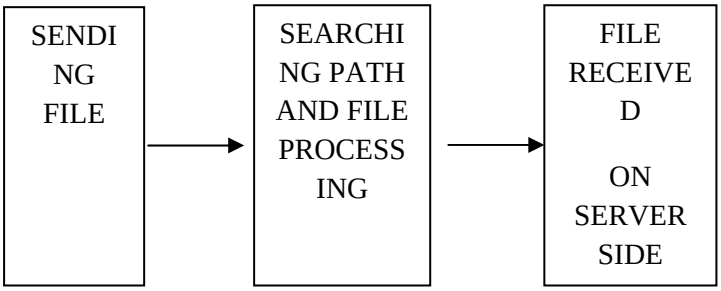
For Digital Signature Verification:



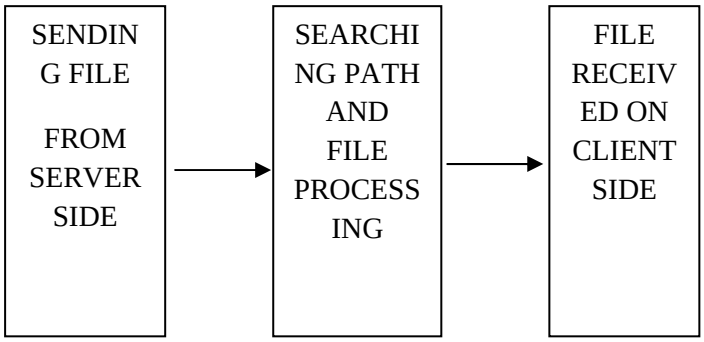
For File Transfer:



For Sending file from Client side:



For Sending file from Server Side:



CONCLUSION

In this paper, we focused upon Security of two way communication in banking. We proposed 128 bit ssl encryption into 256 ssl encryption. That is showed promising compared with the previous schemes.

REFERENCES

[1] [RFC1334] Lloyd, B., and Simpson, W., "PPP Authentication Protocols", RFC 1334, October 1992, http://www.ietf.org/rfc/rfc1334.txt
[2] [RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, http://www.ietf.org/rfc/rfc1945.txt
[3] [RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996, http://www.ietf.org/rfc/rfc1994.txt
[4] [RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, http://www.ietf.org/rfc/rfc2104.txt
[5] [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt
[6] [RFC2284] Blunk, L., and Vollbrecht, J., "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998, http://www.ietf.org/rfc/rfc2284.txt
[7] [RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt
[8] [RFC2716] Aboba, B., and Simon, D., "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999, http://www.ietf.org/rfc/rfc2716.txt
[9] [RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, January 2000, http://www.ietf.org/rfc/rfc2759.txt
[10] [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt
[11] [RFC2965] Kristol, D., and Montulli, L., "HTTP State Management Mechanism", RFC 2965, October 2000, http://www.ietf.org/rfc/rfc2965.txt
[12] [RFC3079] Zorn, G., "Deriving Keys for Use with Microsoft Point-to-Point Encryption (MPPE)", RFC 3079, March 2001, http://www.ietf.org/rfc/rfc3079.txt
[13] [RFC3174] Eastlake III, D., and Jones, P., "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001, http://www.ietf.org/rfc/rfc3174.txt
[14] [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., et al., "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004, http://www.ietf.org/rfc/rfc3748.txt
[15] [SHA256] National Institute of Standards and Technology, "FIPS 180-2, Secure Hash Standard (SHS)", August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf
[16] RFC1334] Lloyd, B., and Simpson, W., "PPP Authentication Protocols", RFC 1334, October 1992, http://www.ietf.org/rfc/rfc1334.txt
[17] [RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol –
[18] HTTP/1.0", RFC 1945, May 1996, http://www.ietf.org/rfc/rfc1945.txt
[19] [RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996, http://www.ietf.org/rfc/rfc1994.txt
[20] [RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, http://www.ietf.org/rfc/rfc2104.txt
[21] [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt
[22] [RFC2284] Blunk, L., and Vollbrecht, J., "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998, http://www.ietf.org/rfc/rfc2284.txt
[23] [RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt
[24] [RFC2716] Aboba, B., and Simon, D., "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999, http://www.ietf.org/rfc/rfc2716.txt
[25] [RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, January 2000, http://www.ietf.org/rfc/rfc2759.txt
[26] [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt
[27] [RFC2965] Kristol, D., and Montulli, L., "HTTP State Management Mechanism", RFC 2965, October 2000, http://www.ietf.org/rfc/rfc2965.txt
[28] [SHA256] National Institute of Standards and Technology, "FIPS 180-2, Secure Hash Standard (SHS)", August 2002, http://www.csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf