Volume 2, Issue 12, December 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Security through SSL

Mr. Pradeep Kumar Panwar *
MCA,UPTU
India

Mr. Devendra Kumar
CS,UPTU
India

Abstract*— As Internet has reached every home, so is the increase in the number of users. The studies have shown as the number of internet users is increasing day by day, there is increase in online transactions. As more number of websites are coming up with easy facility of transmitting information it has been seen that it has lead to substantial increase in fraud. Websites are asking for entering sensitive data from the users, like credit card details, which have posed a major threat to the users who make most of their payments online. It is up to the user to see which secure web page is and which is not. Each information which is transmitted over the web has to be traversed to many computer servers before it reach its destination. Considering the scenario SSL is the solution to the problem. The paper gives an insight into the different attacks on the internet, and how the data can be sent securely through SSL.

***Keywords*— Information, Threat, Fraud, Online Payments, Websites**

I. INTRODUCTION

1. SECUIRTY CONCEPTS

Security over the Internet has become a major concern for the modern communication sector. Whether it is financial, personal or business everyone wants to know whom they are communicating with, ensuring that their data can be sent securely, and whether it has reached the destination correctly. Communicating data over a network always implies a possible loss of confidentiality, message integrity or endpoint authentication. These are the three major aspects one has to consider when speaking of data security [1]:

• Confidentiality: the user wants its data to be kept secret from the listeners on the network.
• Message Integrity: The user wants integrity of its data, what is received is what is sent.
• Endpoint Authentication: The user wants to be sure that he is communicating to the right partner.

SSL assures secure data transmission through the use of several security concepts. These concepts consist of one or more cryptographic algorithms, used to provide security.

A. Encryption

The simplest concept of a security algorithm is encryption. It is built on the idea of taking some plaintext data and converting it to so-called cipher text (encrypted data),using a specific key, usually a short random string of 8-24 bytes in length.

The resulting cipher text looks nothing more than random data with no useful information in it. The only thing one could gather out of it without knowing the appropriate key might be the approximate length of the initial plaintext.

However, it is quite difficult to detect, whether the cipher text has been tampered with. Possible effects on the plaintext of tampering with the cipher text depend on the cipher (encryption algorithm) being used for encryption. An encryption algorithm is considered to be efficient when it responds to the following criteria:

• The cipher's security is totally determined by the number of possible keys; the fastest attack should require exhaustive work.
• The cipher's security depends solely on the secrecy of the key, whereas the algorithm itself should not have to be secret to be secure.

The scheme of using the same key for encryption and decryption is often referred to as Secret Key Cryptography (SKC). Some of the most popular ciphers on the market are the Data Encryption Standard (DES) [2], Triple-DES (DES repeated three times) [3], RC2 [4] and RC4 [5]

1) Message Digests:

A message digest is a function that takes a message of arbitrary length as an input and generates a string of fix length as an output. This string is simply a characteristic representation of the initial message's content. Message digests have two important properties:

1. Irreversibility: Computing a message given its digest should be extremely difficult.

2. Collision-Resistance**:** It should be nearly impossible to produce two messages with the same digest.

The main purpose of message digests is the computation of Digital Signatures and Message Authentication Codes (MACs). Message digests generally allow to prove possession of a secret without actually revealing it. Currently, Message Digest 5 (MD5) [6] and Secure Hash Algorithm (SHA-1) [7] are the most widely used message digest algorithms.

2) Message Authentication Codes:

A Message Authentication Code is sort of a digest algorithm with the difference that beside the message, a key is incorporated to the computation as well. Thus a MAC always depends on both the message being treated as well as the key being used for encryption. MACs are usually constructed using digest algorithms. SSLv3 uses a variant of HMAC, while TLS uses HMAC itself. HMAC is a description of how to create MACs with provable security properties based on digest algorithms.

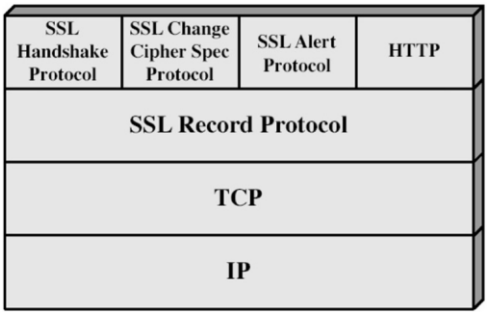
The SSL (Secure Socket Layer) protocol provides means for achieving these goals.

II. SSL OVERVIEW

SSL is a protocol developed by Netscape for transmitting private documents via the Internet. SSL works by using a public key to encrypt data that's transferred over the SSL Connection. Both Netscape Navigator and Internet Explorer support SSL and many Web sites use the protocol to safely transmit confidential information, such as credit card numbers. A number of approaches to providing Web security are possible. The various approaches are similar in many ways but may differ with respect to their scope of applicability and relative location within the TCP/IP protocol stack. For example we can have security at the IP level making it transparent to end users and applications. However another relatively general-purpose solution is to implement security just above TCP. The foremost example of this approach is the Secure Sockets Layer (SSL) [8].

The Secure Sockets Layer (SSL) provides security for web based applications. It uses s TCP to provide end –to-end secure services.SSL is not a single protocol but rather two layers of protocols. It can be seen that one layer makes use of TCP directly. This layer is known as the SSL Record Protocol and it provides basic security services to various higher layer protocols. An independent protocol that makes use of the record protocol is the Hypertext Markup Language (HTTP) protocol. Another three higher level protocols that also make use of this layer are part of the SSL stack. They are used in the management of SSL exchanges and are as follows [9]:

1. Handshake Protocol.
2. Change Cipher Spec Protocol.
3. Alert Protocol.



A. SSL Record Protocol [10]

This protocol provides two services for SSL connections:

1. Confidentiality - using conventional encryption.
2. Message Authentication - using a Message Authentication Code (MAC).

1) Confidentiality: eavesdropping:

The SSL protocol encrypts all application-layer data with a cipher and short-term session key negotiated by the handshake protocol. A wide variety of strong algorithms used in standard modes is available to suit local preferences; reasonable applications should be able to find an encryption algorithm meeting the required level of security.

Key-management is handled well: short-term session keys are generated by hashing random per connection salts and a strong shared secret. Independent keys are used for each direction of a connection as well as for each different instance of a connection. SSL will provide a lot of known plaintext to the eavesdropper, but there seems to be no better alternative; since the encryption algorithm is required to be strong against known-plaintext attacks [11].

2) Confidentiality: traffic analysis:

The Secure Socket Layer (SSL, also known as TLS for Transport Layer Security) was introduced primarily to provide private web access. HTTP requests and replies are encrypted and authenticated between clients and servers, to prevent information from leaking out.

When the standard attacks fail, a cryptanalyst will turn to more obscure ones. Though often maligned, traffic analysis is another passive attack worth considering. Traffic analysis aims to recover confidential information about protection sessions by examining unencrypted packet fields and unprotected packet attributes. For example, by examining the unencrypted IP source and destination addresses (and even TCP ports), or examining the volume of network traffic flow, a traffic analyst can determine what parties are interacting, what type of services are in use, and even sometimes recover information about business or personal relationships [12].

However, there are some more subtle threats posed by traffic analysis in the SSL architecture. Bennet Yee has noted that examination of cipher text lengths can reveal information about URL requests in SSL or SSL-encrypted Web traffic [Yee96]. SSL includes support for random padding for the block cipher modes, but not for the stream cipher modes. We believe that SSL should at the minimum support the usage of random-length padding for all cipher modes, and should also strongly consider requiring it for certain applications.

3) Confidentiality: Active attacks:

It is important that SSL securely protect confidential data even against active attacks. Of course, the underlying encryption algorithm should be secure against adaptive chosen-plaintext/chosen-cipher text attacks, but this is not enough on its own. Recent research motivated by the IETF ipsec (IP security) working group has revealed that sophisticated active attacks on a record layer can breach a system's confidentiality even when the underlying cipher is strong [Bel96].

One important active attack on ipsec is Bellovin's cut-and-paste attack [Bel96]. Recall that, to achieve confidentiality, link encryption is not enough, the receiving endpoint must also guard the sensitive data from inadvertent disclosure. The cut-and-paste attack exploits the principle that most endpoint applications will treat inbound encrypted data differently depending on the context, protecting it more assiduously when it appears in some forms than in others.

The cut-and-paste attack also takes advantage of a basic property of the cipher-block chaining mode: it recovers from errors within one block, so transplanting a few consecutive ciphertext blocks between locations within a ciphertext stream results in a corresponding transfer of plaintext blocks, except for a one-block error at the beginning of the splice. In more detail, Bellovin's cut-and-paste attack cuts an encrypted ciphertext from some packet containing sensitive data, and splices it into the ciphertext of another packet which is carefully chosen so that the receiving endpoint will be likely to inadvertently leak its plaintext after decryption.

The SSL record layer format is rather similar to the old vulnerable ipsec layout, so it is admittedly conceivable that a modified version of the attack might work against SSL. In any case, standard SSL-encrypting Web servers probably would not be threatened by a short-block type of attack, since they do not typically encrypt short blocks [13].

B. Message authentication

In addition to protecting the confidentiality of application data, SSL cryptographically authenticates sensitive communications. On the Internet, active attacks are getting easier to launch every day. We are aware of at least two commercially available software packages to implement active attacks such as IP spoofing and TCP session hijacking, and they even sport a user-friendly graphical interface. Moreover, the financial incentive for exploiting communications security vulnerabilities is growing rapidly. This calls for strong message authentication. SSL protects the integrity of application data by using a cryptographic MAC. The SSL designers have chosen to use HMAC, a simple, fast hash-based construction with some strong theoretical evidence for its security [14]. In an area where several initial ad-hoc proposals for MACs have been crypt analyzed, these provable security results are very attractive.

HMAC is rapidly becoming the gold standard of message authentication, and it is an excellent choice for SSL. The SSL MAC keys contain at least 128 bits of entropy, even in export-weakened modes, which should provide excellent security for both export-weakened and domestic-grade implementations. Independent keys are used for each direction of each connection and for each new incarnation of a connection. The choice of HMAC should stop cryptanalytic attacks.

SSL does not provide non-repudiation services, and it seems reasonable to deliberately leave that to special higher-level application-layer protocols. In order to operate on data the protocol performs the following actions:

• It takes an application message to be transmitted and fragments it into manageable blocks. These blocks are 214 = 16, 384 bytes or less.

• These blocks are then optionally compressed which must be lossless and may not increase the content length by more than 1024 bytes.

• A message authentication code is then computed over the compressed data using a shared secret key. This is then appended to the compressed (or plaintext) block.

MAC Calculation:

hash (MAC_write_secret ||pad_2||
hash(MAC_write_secret||pad_1||seq_num||
SSLCompresses.type||SSLCompressed.Length||SSLCompressed.fragment))

Where

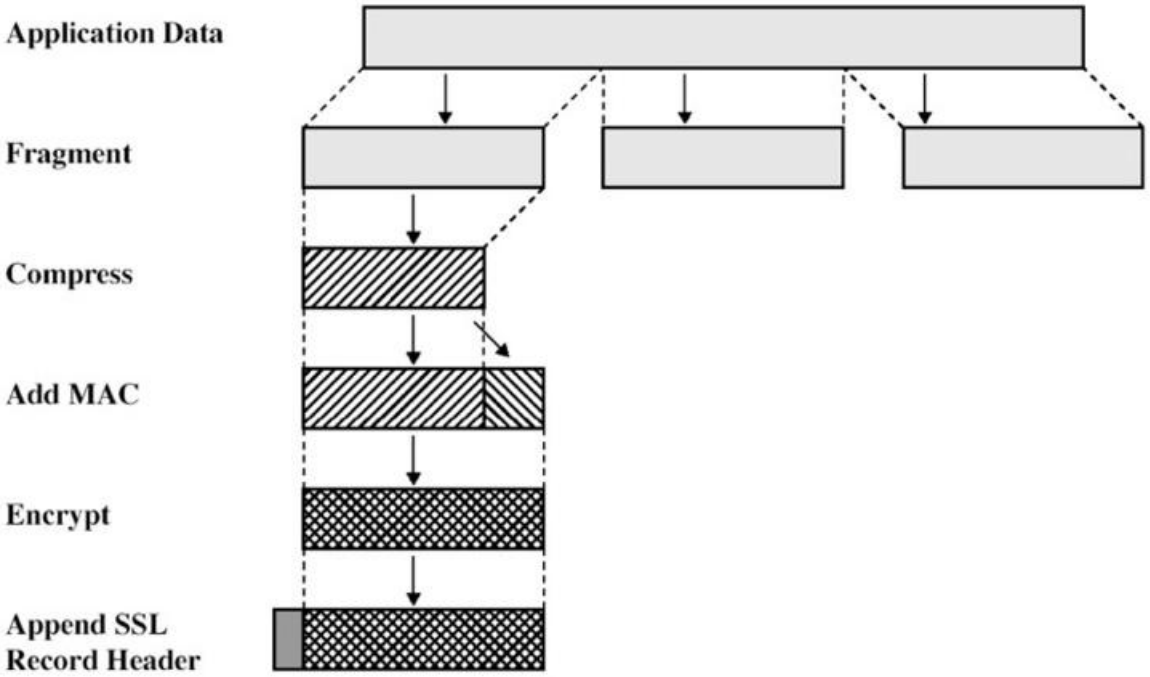
|| = concatenation

MAC_write_secret = shared secret key
Hash = cryptographic hash algorithm: either MD5 or SHA-1
Pad_1 = the byte 0x36 repeated 48 times for MD5
Pad_2 = the byte 0x5C repeated 48 times for MD5
Seq_num =the sequence number for this message
SSLCompressed.type =the higher-level protocol used to process this fragment
SSLCompressed.length =the length of the compressed fragment
SSLCompressed.fragment =the compressed fragment (if compression is not used, the plain text fragment)

• The compressed message plus MAC are then encrypted using symmetric encryption.

- Encryption may not increase the content length by more than 1024

bytes, so that the total length may not exceed 214 + 2048. A number of different encryption algorithms are permitted.
• The final step is to prepend a header, consisting of the following fields:



– Content type (8 bits) - The higher layer protocol used to process the enclosed fragment.
– Major Version (8 bits) - Indicates major version of SSL in use. For SSLv3, the value is 3.
– Minor Version (8 bits) - Indicates minor version in use. For SSLv3, the value is 0.
– Compressed Length (16 bits) - The length in bytes of the compressed (or plaintext) fragment.
The "content type" above is one of four types; the three higher level protocols given above that make use of the SSL record, and a fourth known as "application data". The first three are described next as they are SSL specific protocols.

III. The Key Exchange Protocol

A. The SSL Handshake Protocol

The SSL handshake-protocol message flow involves client and server negotiating a common cipher suite acceptable to both parties, exchanging random nonces, and the client sending an encrypted master secret. Then each verifies that their protocol runs match by authenticating all messages with the master secret, and assuming that the check succeeds, both generate session keys from the master secret and proceed to send application data.

1) Handshake protocol:

The handshake protocol is responsible for selecting a CipherSpec and generating a MasterSecret, which together comprise the primary cryptographic parameters associated with a secure session. The handshake protocol can also optionally authenticate parties who have certificates signed by a trusted certificate authority [14].

a) Authentication and key exchange

SSL supports three authentication modes: authentication of both parties, server authentication with an unauthenticated client, and total anonymity. Whenever the server is authenticated, the channel should be secure against man-in-the-middle attacks, but completely anonymous sessions are inherently vulnerable to such attacks. Anonymous servers cannot authenticate clients, since the client signature in the certificate verify message may require a server certificate to bind the signature to a particular server.

If the server is authenticated, its certificate message must provide a valid certificate chain leading to an acceptable certificate authority. Similarly, authenticated clients must supply an acceptable certificate to the server. Each party is responsible for verifying that the other's certificate is valid and has not expired or been revoked.

The general goal of the key exchange process is to create a pre_master_secret known to the communicating parties and not to attackers. The pre_master_secret will be used to generate the master_secret.The master_secret is required to generate the finished messages, encryption keys, and MAC secrets. By sending a correct finished message, parties thus prove that they know the correct pre_master_secret [15].

b) Anonymous key exchange

Completely anonymous sessions can be established using RSA, Diffie-Hellman, or Fortezza for key exchange.

With anonymous RSA, the client encrypts a pre_master_secret with the server's uncertified public key extracted from the server key exchange message. The result is sent in a client key exchange message. Since eavesdroppers do not know the server's private key, it will be infeasible for them to decode the pre_master_secret.

With Diffie-Hellman or Fortezza, the server's public parameters are contained in the server key exchange message and the client's are sent in the client key exchange message. Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (*i.e.* the pre_master_secret) or the Fortezza token encryption key (TEK).

Warning: Completely anonymous connections *only* provide protection against passive eavesdropping. Unless an independent tamper-proof channel is used to verify that the finished messages were not replaced by an attacker, server authentication is required in environments where active man-in-the-middle attacks are a concern [16].

c) RSA key exchange and authentication

With RSA, key exchange and server authentication are combined. The public key may be either contained in the server's certificate or may be a temporary RSA key sent in a server key exchange message. When temporary RSA keys are used, they are signed by the server's RSA or DSS certificate. The signature includes the current ClientHello.random, so old signatures and temporary keys cannot be replayed. Servers may use a single temporary RSA key for multiple negotiation sessions.

After verifying the server's certificate, the client encrypts a pre_master_secret with the server's public key. By successfully decoding the pre_master_secret and producing a correct finished message, the server demonstrates that it knows the private key corresponding to the server certificate.

When RSA is used for key exchange, clients are authenticated using the certificate verify message. The client signs a value derived from the master_secret and all preceding handshake messages. These handshake messages include the server certificate, which binds the signature to the server, and ServerHello.random, which binds the signature to the current handshake process [17].

d) Diffie-Hellman key exchange with authentication

When Diffie-Hellman key exchange is used, the server can either supply a certificate containing fixed Diffie-Hellman parameters or can use the client key exchange message to send a set of temporary Diffie-Hellman parameters signed with a DSS or RSA certificate. Temporary parameters are hashed with the hello.random values before signing to ensure that attackers do not replay old parameters. In either case, the client can verify the certificate or signature to ensure that the parameters belong to the server.

If the client has a certificate containing fixed Diffie-Hellman parameters, its certificate contains the information required to complete the key exchange. Note that in this case the client and server will generate the same Diffie-Hellman result (*i.e.*, pre_master_secret) every time they communicate. To prevent the pre_master_secret from staying in memory any longer than necessary, it should be converted into the master_secret as soon as possible. Client Diffie-Hellman parameters must be compatible with those supplied by the server for the key exchange to work.

If the client has a standard DSS or RSA certificate or is unauthenticated, it sends a set of temporary parameters to the server in the client key exchange message, then optionally uses a certificate verify message to authenticate itself [18].

e) Fortezza

Fortezza's design is classified, but at the protocol level it is similar to Diffie-Hellman with fixed public values contained in certificates. The result of the key exchange process is the token encryption key (TEK), which is used to wrap data encryption keys, client write key, server write key, and master secret encryption key. The data encryption keys are not derived from the pre_master_secret because unwrapped keys are not accessible outside the token. The encrypted pre_master_secret is sent to the server in a client key exchange message.

The asymmetric algorithms are used in the handshake protocol to authenticate parties and to generate shared keys and secrets.

For Diffie-Hellman, RSA, and Fortezza, the same algorithm is used to convert the pre_master_secret into the master_secret. The pre_master_secret should be deleted from memory once the master_secret has been computed [19].

```
master_secret =
    MD5(pre_master_secret + SHA('A' + pre_master_secret +
      ClientHello.random + ServerHello.random)) +
    MD5(pre_master_secret + SHA('BB' + pre_master_secret +
      ClientHello.random + ServerHello.random)) +
    MD5(pre_master_secret + SHA('CCC' + pre_master_secret +
      ClientHello.random + ServerHello.random));
```

IV. THE ALERT PROTOCOL

SSL includes a small provision for sending event driven alert messages. Many of these indicate fatal error conditions and instruct the recipient to immediately tear down the session. For instance, the close-notify alert message indicates that the sender is finished sending application data on the connection; since alert messages are normally authenticated, this prevents a truncation attack. As another example, reception of any packet with an incorrect MAC will result in a fatal alert.

V. MAC usage

The SSL handshake protocol uses several adhoc MAC constructions to provide message integrity. The security of these MACs has not been thoroughly evaluated. We believe that SSL should consistently use HMAC whenever a MAC is called for; ad-hoc MACs should be avoided [21].

VI. Conclusion

SSL protocol provides total security to data which is sent over the communication network. The analysis shows that it provides different measures of security against eavesdropping and other passive attacks. The analysis has also revealed several ways in which the robustness of the SSL protocol can be improved. It also discusses the role of key exchange protocol through handshake protocol authenticate parties who have certificates signed by a trusted certificate authority. Thus, the paper describes the usage of SSL in securing the data transmitted through web pages.

REFERENCES

[1] A. O. Freier P. Karlton and P. C. Kocher. The SSL Protocol, Version 3.0. Netscape Communications, 1996, http://wp.netscape.com/eng/ssl3/draft302.txt (2003).

[2] National Institute of Standards and Technologies. Data Encryption Standard. U.S. Dpt. of Commerce, December 1993.

[3] ANSI. American National Standard for Financial Institution Key Management (wholesale). ANSI, 1985.

[4] R. Rivest. A Description of the RC2(r) Encryption Algorithm, RFC 2268. Network Working Group, 1998, ftp://ftp.rfc-editor.org/in-notes/rfc2268.txt (2003).

[5] B. Schneier. Applied Cryptography, 2ed. John Wiley and Sons, 1996.

[6] R. Rivest. The MD-5 Message-Digest Algorithm, RFC 1321. John Wiley and Sons, 1996, ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt (2003).

[7] National Institute of Standards and Technologies. National Institute of Standards and Technologies, and Secure Hash Standard. U.S. Dpt. of Commerce, 1994.

[8] http://www.webopedia.com/TERM/S/SSL.html

[9] [10] http://www.facweb.iitkgp.ernet.in/~sourav/SSL.pdf

[11] [12] [13] [Bel96] S. Bellovin, \Problem Areas for the IP Security Protocols", *Proceedings of the Sixth USENIX Security Symposium*, Usenix Association, 1996, pp. 205-214. ftp://ftp.research.att.com/dist/ smb/badesp.ps.

[14] [BCK96] M. Bellare, R. Canetti, and H. Krawczyk, \Keying Hash Functions for Message Authentication," *Advances in Cryptology|CRYPTO '96 Proceedings*, Springer-Verlag, 1996, pp. 1-15.

[15] [16][FKK96] A. Freier, P. Karlton, and P. Kocher, The SSL Protocol Version 3.0", ftp://ftp.netscape.com/pub/review/ ssl-spec.tar.Z, March 4 1996, Internet Draft, work in progress.

[17] MERKLE, R., AND HELLMAN, M. 1981. On the security of multiple encryptlon *Commun. ACM* 24, 7 (July), 465-467.

[18] [RSA93] RSA Data Security, Inc., \Public-Key Cryptography Standards (PKCS)," Nov 93.

[19] Whitfield Diffe and Susan Landau. Privacy on the Line: The Politics of Wiretapping and Encryption. MIT Press, 1998.

[20] [21]BRANSTAD, D. 1973. Security aspects of computer networks. In *Proc AIAA Computer Network Systems Con[* (Huntsville, Ala., Apr).

