# Chapter 1

# INTRODUCTION

## 1.1 SSL BASICS

The Transmission Control Protocol/Internet Protocol (TCP/IP) governs the transport and routing of data over the Internet. Other protocols such as the Hypertext Transport Protocol (HTTP), Lightweight Directory Access Protocol (LDAP), or Internet Messaging Access Protocol (IMAP), run "on top of" TCP/IP in the sense that they all use TCP/IP to support typical application tasks such as displaying web pages or running email servers.

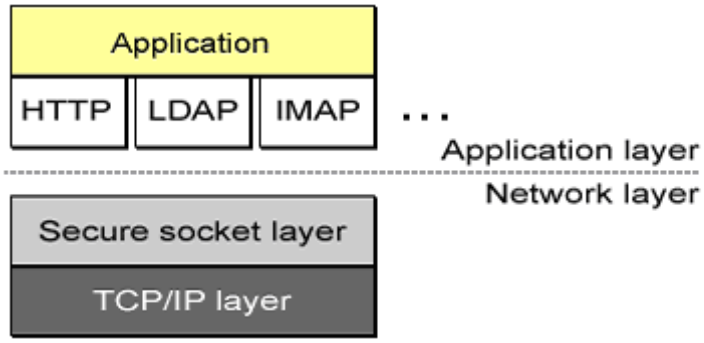



*Fig1.1: SSL runs above TCP/IP and below high-level application protocols*

The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP or IMAP. It uses TCP/IP on behalf of the higher-level protocols and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection. These capabilities address fundamental concerns about communication over the Internet and other TCP/IP network

- **SSL SERVER AUTHENTICATION***: A*llows a user to confirm a server's identity. SSL-enabled client software can use standard techniques of public-key cryptography to check that a server's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the client's list of trusted CA's. This conformation might be important if the user, For example, is sending a credit card number over the network and wants to check the receiving server's identity.

- **SSL CLIENT AUTHENTICATION:** allows a server to confirm a user's identity. Using the same techniques as those used for server authentication, SSL-enabled server software can check that a client's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the server's list of trusted CA's. This confirmation might be

important if the server, for example, is a bank sending confidential financial information to a customer and wants to check the recipient's identity.

- **AN ENCRYPTED SSL CONNECTION:** requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality. Confidentiality is important for both parties to any private transaction. In addition, all data sent over an encrypted SSL connection is protected with a mechanism for detecting tampering--- that is, for automatically determining whether the data has been altered in transit.

The SSL Protocol is composed of two layers. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP), is the SSL Record Protocol. The SSL Record Protocol is used for encapsulation of various higher-level protocols. One such encapsulated protocol, the SSL 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.

## 1.2 INTERNET SECURITY ISSUES

All communication over the internet uses the transmission control protocol (TCP/IP). TCP/IP allows information to be sent from one computer to another through a variety of intermediate computers and separate networks before it reaches its destination.

The great flexibility of TCP/IP has led to its worldwide acceptance as the basic Internet and intranet communications protocol. At the same time, the fact that TCP/IP allows information to pass through intermediate computers makes it possible for a third party to interface with communications in the following ways:

- **EAVESDROPPING:** Information remains intact, but its privacy is compromised. For example, someone could learn your credit card number, record a sensitive conversation, or interact classified information.

- **TAMPERING:** Information in transit is changed or replaced and then sent to the recipient. For example, someone could alter an order for goods or change a person's resume.

- **IMPRESONATION*:*** Information passes to a person who poses as the intended recipient. Impersonation can take two forms:
  - **SPOOFING*:*** A person can pretend to be someone else. For example, a person can pretend to have the email address jdoe@xyz.com or a computer can identify itself as a site called www.xyz.com when it is not. This type of impersonation is known as spoofing.
  - **MISREPRESENTATION:** A person or organization can misrepresent itself. For example, suppose the site www.xyz.com pretends to be a furniture store when it is really just a site that takes credit-card payments but never sends any goods.

### 1.2.1 WEB SECURITY

These general requirements for web security then focus on standardized schemes known as SSL/TLS and SET.

- **WEB SECURITY CONSIDERATION*:*** The World Wide Web is fundamentally a client server application running over the Internet and TCP/IP intranets.

- **WEB SECURITY THREATS*:*** Given table provides a summary of the types of security threats faced in using the web. One way to group these threats is in terms of passive & active attacks. Passive attacks include eaves dropping on network traffic between browser and server and gaining access to information on a website. Active attacks include impersonating another user, attacking messages in transit between client and server and altering information on a website.

| | Threats | Consequences | Countermeasures |
|---|---|---|---|
| Integrity | • Modification of user data<br>• Trojan horse browser<br>• Modification of memory<br>• Modification of message traffic in transit | • Loss of information<br>• Compromise of machine<br>• Vulnerability to all other users | Cryptographic checksums |
| Confidentiality | • Eavesdropping on the Net<br>• Theft of info from server<br>• Theft of data from client<br>• Info about n/w configuration<br>• Info about which client talks to server | • Loss of information<br>• Loss of privacy | Encryption, web proxies |
| Denial of service | • Killing of user threads<br>• Flooding m/c with bogus threats<br>• Filling up disk or memory<br>• Isolating m/c by DNS attacks | • Disruptive<br>• Annoying<br>• Prevent user from work done | Difficult to prevent |
| Authentication | • Impersonation of legitimate users<br>• Data forgery. | • Misrepresentation of user<br>• Belief that false information is valid | Cryptographic techniques |

***Table1.1: -*** *comparison of the threats on the web*

- **WEB TRAFFIC SECURITY APPROACHES : -** There are number of approaches to providing web security. The various approaches that have been considered are similar in the services, but they differ with respect to their scope of applicability and their relative location within the TCP/IP protocol stack.

- One way to provide web security is by using IP security Figure1.2. The advantage of using IPSEC is, it is transparent to end users and applications

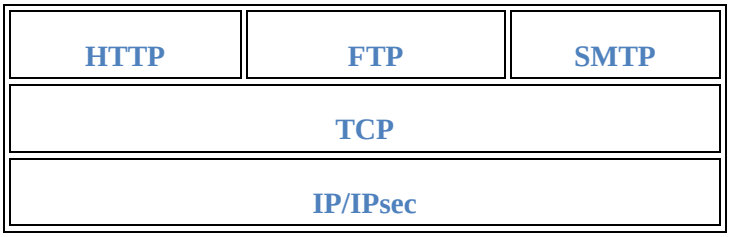



*Fig 1.2: Network level*

- Another way is to implement security just above TCP Figure 1.3. The foremost example of this approach is the security sockets layer (SSL) and the follow-on Internet standard of SSL known as Transport layer security (TLS).

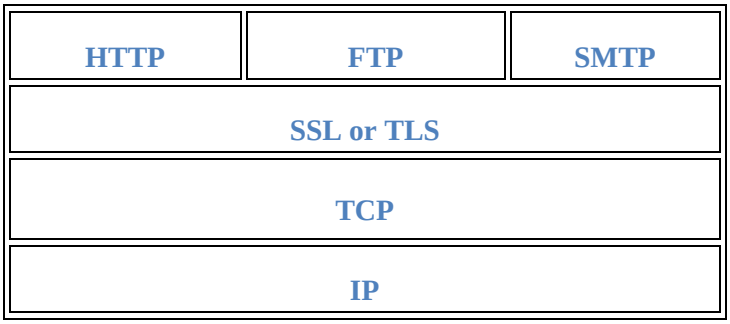



*Fig. 1.3: Transport Level*

- Application specific security services are embedded within the particular application. Fig 1.4. shows examples of this architecture. The advantage of this approach is that the service can be tailored to the specific needs of a given application.

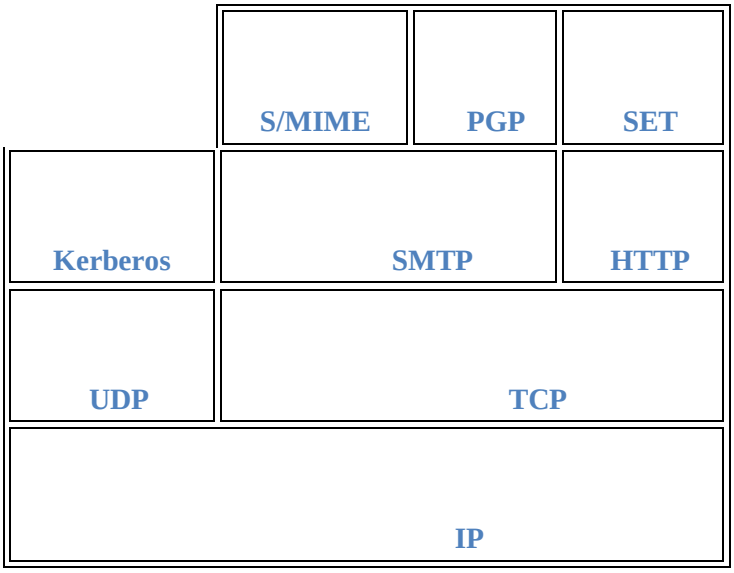



*Fig 1.4: Application level*

## 1.3 ENCRYPTION AND DECRYPTION

Encryption is the process of converting a plaintext message into cipher text, which can be decoded back into the original message. An encryption algorithm along with a key is used in the encryption and decryption of data. There are several types of data encryptions, which form the basis of network security.

- **ENCRYPTION ALGORITHMS**

| **Block Cipher** | | **Stream Cipher** | |
|---|---|---|---|
| *Algorithm* | *Key Size* | *Algorithm* | *Key Size* |
| Idea | 128 | RC4-40 | 40 |
| RC2-40 | 40 | RC4-128 | 128 |
| DES-40 | 40 | | |
| DES | 56 | | |
| 3DES | 168 | | |
| Fortezza | 80 | | |

*Table 1.2: Encryption Algorithms*

Encryption schemes are based on block or stream ciphers**.** The type and length of the keys utilized depend upon the encryption algorithm and the amount of security needed. In conventional **symmetric** encryption a single key is used. With this key, the sender can encrypt a message and a recipient can decrypt the message but the security of the key becomes problematic. In **asymmetric** encryption, the encryption key and the decryption key are different. One is a public key by which the sender can encrypt the message and the other is a private key by which a recipient can decrypt the message.

With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, which is widely known, but on a number called a **key** that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple. Decryption without the correct key is very difficult, and in some cases impossible for all practical purposes.

The sections that follow introduce the use of keys for encryption and decryption.

### 1.3.1 SYMMETRIC-KEY ENCRYPTION

With symmetric-key encryption, the encryption key can be calculated from the decryption key and vice versa. With most symmetric algorithms, the same key is used for both encryption and decryption, as shown in Figure 1.5.

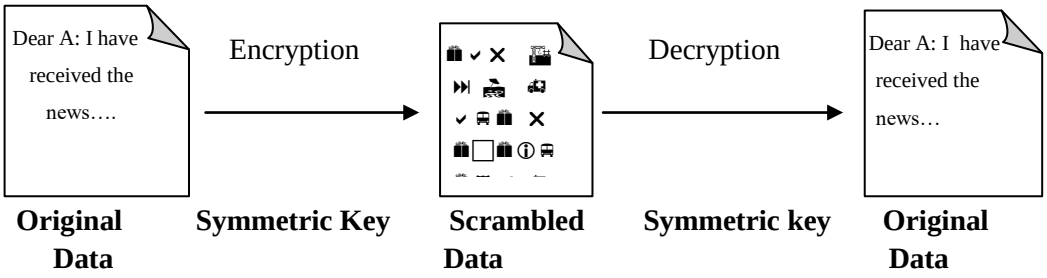



*Fig 1.5: Symmetric-key encryption*

Implementation of symmetric-key encryption can be highly efficient, so that users do not experience any significant time delay as a result of the encryption and decryption. Symmetric-key encryption also provides a degree of authentication, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key. Thus, as long as the symmetric key is kept secret by the two parties using it to encrypt communications, each party can be sure that it is communicating with the other as long as the decrypted messages continue to make sense.

*Symmetric-key encryption* is effective only if the two parties involved keep the symmetric key secret. If anyone else discovers the key, it affects both confidentiality and authentication. A person with an unauthorized symmetric key not only can decrypt messages sent with that key but can encrypt new messages and send them as if they came from one of two parties who were originally using the key.

### 1.3.2 PUBLIC-KEY ENCRYPTION

Public-key encryption (also called asymmetric encryption) involves a pair of keys--- a public key and a private key – associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published, and the corresponding private key is kept secret. Data encrypted with your public key can be decrypted only with your private key. Figure 1.6 shows a simplified view of the way public-key encryption works**.**

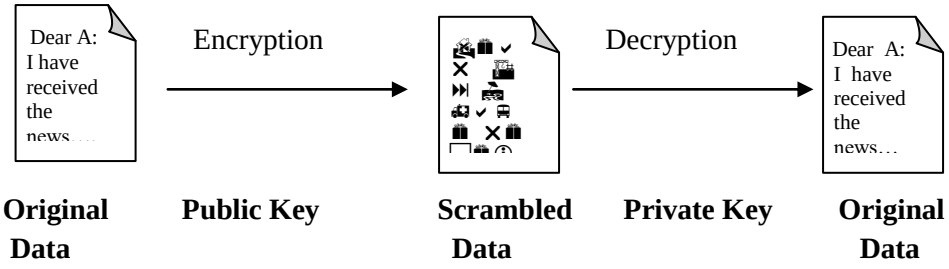



*Fig 1.6: Public-key encryption*

## 1.4 DIGITAL SIGNATURES

Encryption and decryption address the problem of eavesdropping, one of the three Internet security issues mentioned at the beginning of this document. But encryption and decryption, by themselves, do not address the other two problems mentioned in Internet Security Issues: tampering and impersonation.

Tamper detection and related authentication techniques rely on a mathematical function called a one-way hash (also called a message digest). A one-way hash produces a fixed length output with the following characteristics:

- The value of the hash is unique for the hashed data. Any change in the data, even deleting or altering a single character, results in a different value.
- The content of the hashed data cannot, for all practical purposes, be deduced from the hash--which is why it is called "one-way."

Figure 1.7 shows a simplified view of the way a digital signature can be used to validate the integrity of signed data.

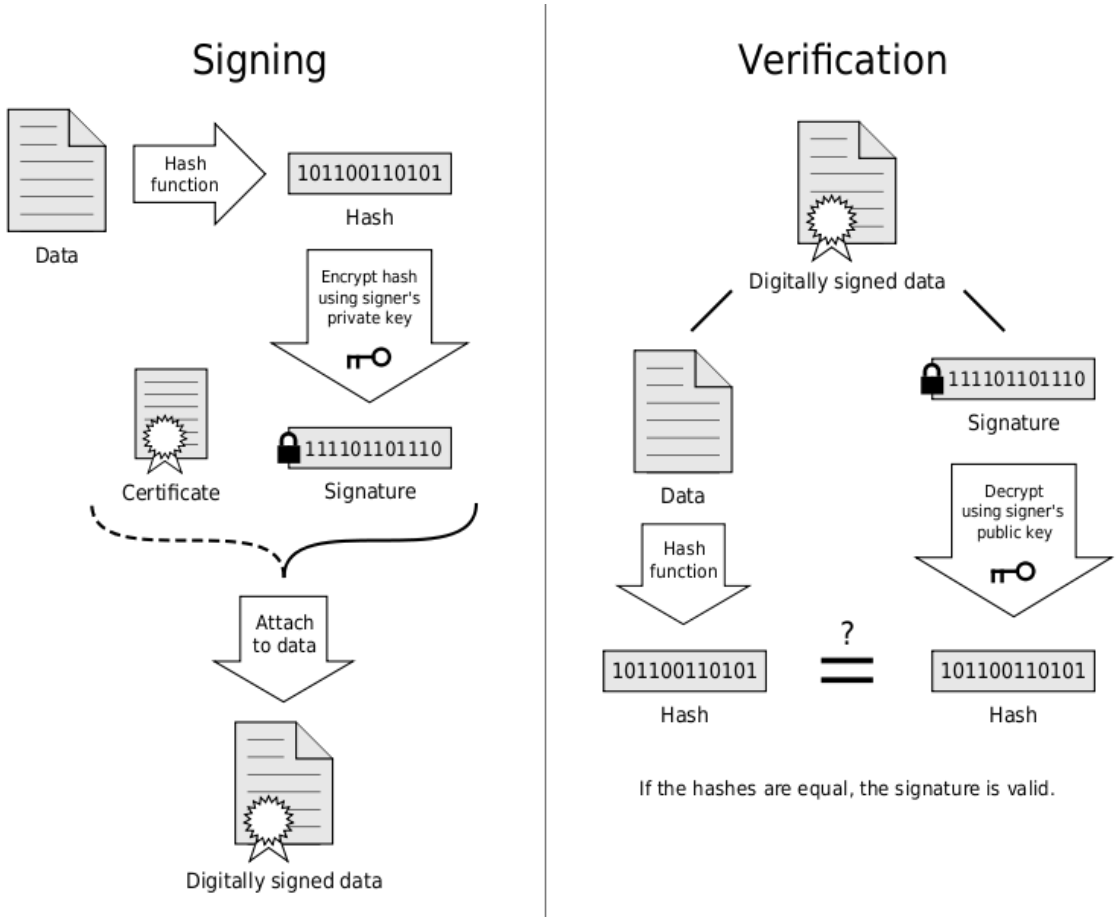



*Fig1.7: Using a digital signature to validate data integrity*

Fig 1.7 shows two items transferred to the recipient of some signed data: the original data and the digital signature, which is basically a one-way hash (of the original data) that has been encrypted with the signer's private key. To validate the integrity of the data, the receiving software first uses the signer's public key to decrypt the hash. It then uses the same hashing algorithm that generated the original hash to generate a new one-way hash of the same data.

## 1.5 CERTIFICATES AND AUTHENTICATION

A **certificate** is an electronic document used to identify an individual, a server, a company, or some other entity and to associate that identity with a public key. Like a driver's license, a passport, or other commonly used personal Ids, a certificate provides generally recognized proof of a person's identity. Public-key cryptography uses certificates to address the problem of impersonation.

**Certificate authorities** (CAs) are entities that validate identities and issue certificates. They can be either independent third parties or organizations running their own certificate-issuing server software. Before issuing the certificate the CA must use its published verification

procedures for that type of certificate to ensure that an entity requesting a certificate is in fact who it claims to be. The certificate issued by the CA binds a particular public key to the name of the entity the certificate identifies (such as the name of an employee or a server). Certificates help prevent the use of fake public keys for impersonation. Only the public key certified by the certificate will work with the corresponding private key possesses by the entity identified by the certificate.

**AUTHENTICATION CONFIRMS AN IDENTITY**

**Authentication** is the process of confirming an identity. In the content of network interactions, authentication involves the confident identification of one party by another party. Authentication over networks can take many forms. Certificates are one way of supporting authentication.

Network interactions typically take place between a client, such as browser software running on a personal computer, and a server, such as the software and hardware used to host a Web site. Client authentication refers to the confident identification of a client by a server (that is, identification of the person assumed to be using the client software). Server authentication refers to the confident identification of a server by a client (that is, identification of the organization for a server at a particular network address).

- **PASSWORD-BASED AUTHENTICATION:** Almost all server software permits client authentication by means of a name and password. For example, a server might require a user to type a name and password before granting access to the server. The server maintains a list of names and passwords; if a particular name is on the list, and the user types the correct password, the server grants access.

- **CERTIFICATE BASED AUTHENTICATION:** Client authentication based on certificates is part of the SSL protocol. The client digitally signs a randomly generated piece of data and sends both the certificate and the signed data across the network. The server uses techniques of public-key cryptography to validate the signature and confirm the validity of the certificate.

# Chapter 2

# Technical Overview

## 2.1 TECHNICAL OVERVIEW OF THE SSL PROTOCOL

The SSL Protocol is designed to provide privacy between two communicating application (a client and a server). Second, the protocol is designed to authenticate the server, and optionally

.

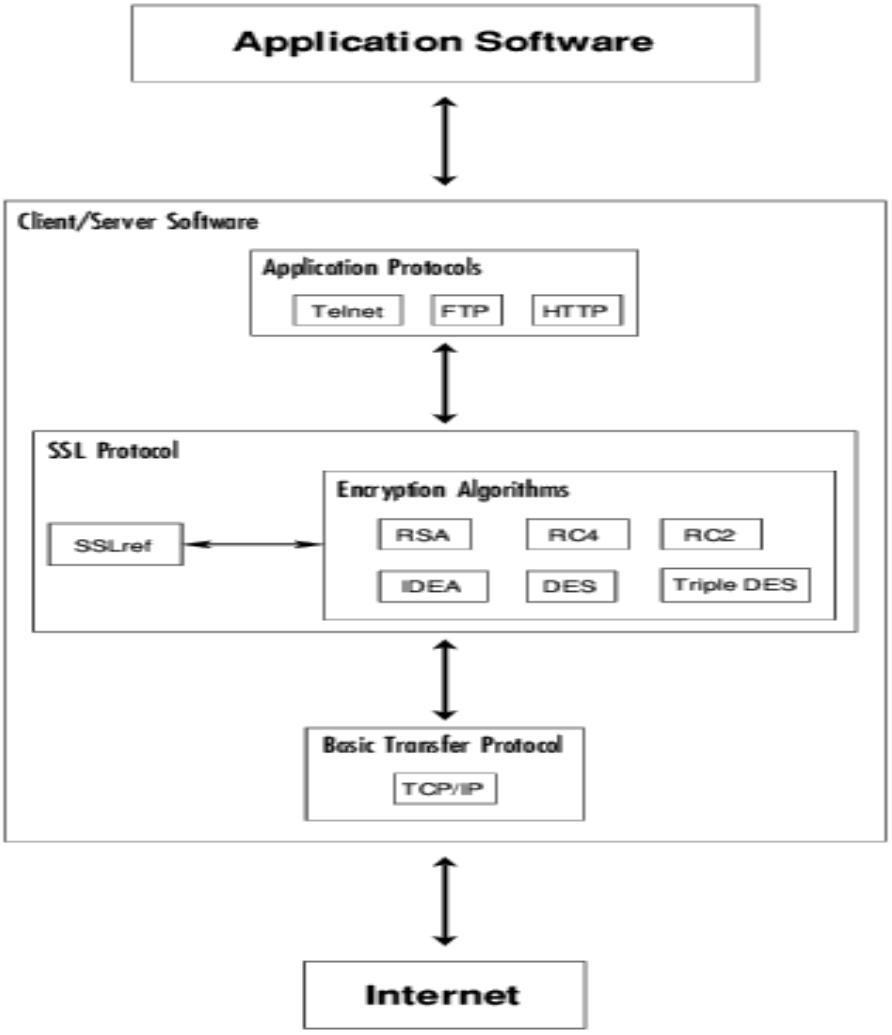



*Fig 2.1: Diagram of relation between SSL, Applications and the Network*

The client SSL uses X.509 certificates for authentication, RS as its public key cipher and one of RC4 -128, RC2-128, DES, Triple DES or IDEA as its bulk symmetric cipher.

## 2.2 SSL OBJECTIVES AND ARCHITECTURE

THE MAIN OBJECTIVE OF SSL ARE

- Authenticating the client and server to each other: The SSL protocol supports the use of standard key cryptographic techniques (public key encryption) to authenticate the communicating parties to each other. Though the most frequent application consists in authenticating the service client on the basis of a certificate, SSL may also use the same methods to authenticate the client.
- Ensuring data integrity: during a session, data cannot be either intentionally or unintentionally tampered with.
- Securing data privacy: data in transport between the client and the server must be protected from interception and be readable only by the intended recipient. This prerequisite is necessary for both the data associated with the protocol itself (securing traffic during negotiations) and the application data that is sent during the session itself.

### 2.2.1 SSL ARCHITECTURE

SSL is in fact not a single protocol but rather a set of protocols that can additionally be further divided in two layers:

1. The protocol to ensure data security and integrity: this layer is composed of the SSL **Record Protocol**,
2. The protocols that are designed to establish an SSL connection: three protocols are used in this layer: the **SSL Handshake Protocol**, the **SSL Change Cipher Spec Protocol** and the **SSL Alert Protocol.**

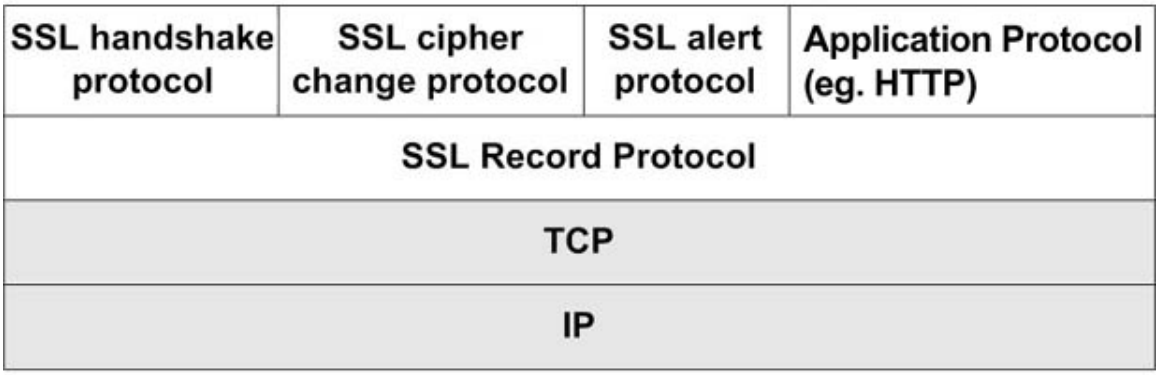



*Fig 2.2: The SSL Protocol stack*

SSL uses these protocols to address the tasks as described above. The SSL record protocol is responsible for data encryption and integrity. As can be seen in Figure, it is also used to encapsulate data sent by other SSL protocols, and therefore, it is also involved in the tasks associated with the SSL check data. The other three protocols cover the areas of session management, cryptographic parameter

management and transfer of SSL messages between the client and the server. Prior to going into a more detailed discussion of the role of individual protocols and their functions let us describe two fundamental concepts related to the use of SSL.

### 2.2.1 SSL SESSION AND CONNECTION

The concepts as mentioned above are fundamental for a connection between the client and the server, and they also encompass a series of attributes which are described below :

- **Connection**: This is a logical client/server link, associated with the provision of a suitable type of service. In SSL terms, it must be a peer-to-peer connection with two network nodes.
- **Session**: This is an association between a client and a server that defines a set of parameters such as algorithms used session number etc.:
- **Session identifier**: this is an identifier 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 termed Cipher Spec**: specifies the bulk data encryption algorithm (for example DES) and the hash algorithm (for example MD5) used during the session.
- **Master secret**: 48-byte data being a secret shared between the client and server,
- **"Is resumable"**: this is a flag indicating whether the session can be used to initiate new connections. According to the specification, the SSL connection state is defined by the following parameters:
- **Server and client random**: random data generated by both the client and server for each connection,

The abbreviation MAC used in the above definitions means **Message Authentication Code** that is used for transmission of data during the SSL session. The role of MAC will be explained further when discussing the record protocols. A brief description of the terms was necessary to be able to explain the next issues connected with the functioning of the SSL protocol, namely the SSl record protocol.

### 2.2.2 SSL Record Protocol

The SSL record protocol involves using SSL in a secure manner and with message integrity ensured. To this end it is used by upper layer SSL protocols. The purpose of the SSL record protocol is to take an application message to be transmitted, fragment the data which needs to be sent, encapsulate it with appropriate headers and create an object just called a record, which is encrypted and can be forwarded for sending under the TCP protocol.

The first step in the preparation of transmission of the application data consists in its fragmentation i.e. breaking up the data stream to be transmitted into 16Kb (or smaller) data fragments followed by the process of their conversion in a record.

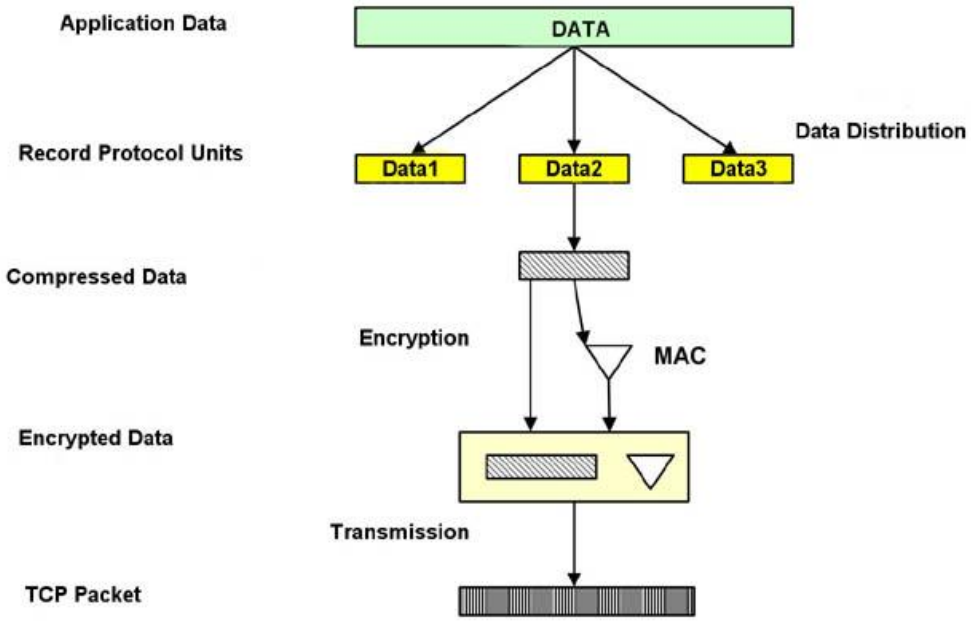



*Fig 2.3: Creating a packet under SSL record protocol*

The SSL record protocol is used to transfer any data within a session - both messages and other SSL protocols (for example the handshake protocol), as well as for any application data.

The SSL Record Protocol provides two services for SSL connections:

- **Confidentiality**: The handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
- **Message Integrity**: The Handshake protocol also defines a shared secret key that is used to form a message authentication code.

Figure 2.3 indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled and then delivered to higher-level users.

The first step is Fragmentation. Each upper layer message is fragmented into blocks of 2^14 bytes or less. Next Compression is optionally applied. Compression must be loss less and may not increase the content length by more than 1024 bytes. In SSLv3 no compression algorithm is specified, so the default compression algorithm is null. The next step is to compute a message authentication code over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as follows:

**Hash (MAC_write_secret || pad_2 ||**

**Hash (MAC_write_secret || pad_1 || seq_num || SSLCompressed.type ||**

**SSLCompressed.length || SSLCompressed.fragment))**

Where,

| | |
|---|---|
| || | = Concatenation |
| MAC_write_secret | = shared secret key |
| Hash | = cryptographic hash algorithm SHA-1 |
| Pad_1 | = the byte 0x36 repeated 40 times for SHA-1 |
| Pad_2 | = the byte 0x5c repeated 40 times for SHA-1 |
| Seq_num | = the sequence number for this message |
| SSLCompressed.type | = the higher level protocol used to process this Fragment |
| SSLCompressed.length | = length of compressed fragment |
| SSLCompressed.fragment | = compressed fragment |

The compressed message plus the MAC are encrypted using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes, so the total length may not exceed 2^14+2048.

### 2.2.3 SSL ALERT PROTOCOL

Parties to convey session messages associated with data exchange and functioning of the protocol use the Alert Protocol. Each message in the alert protocol consists of two bytes. The first byte always takes a value, “warning” (1) or “fatal” (2), that determines the severity of the message sent. Sending a message having a „fatal” status by either party will result in an immediate termination of the SSL session. The next byte of the message contains one of the defined error codes, which may occur during an SSL communication session.

### 2.2.4 CHANGE CIPHER SPEC PROTOCOL

This protocol is the simplest SSL protocol. It consists of a single message that carries the value of 1. The sole purpose of this message is to cause the pending session state to be established as a fixed state, which results, for example, in defining the used set of protocols. The client must send this type of

message to the server and vice versa. After exchange of messages, the session state is considered agreed. This message and any other SSL messages are transferred using the SSL record protocol.

### 2.2.5 SSL HANDSHAKE PROTOCOL

The Handshake protocol constitutes the most complex part of the SSL protocol. It is used to initiate a session between the server and the client. Within the message of this protocol, various components such as algorithms and keys used for data encryption are negotiated. Due to this protocol, it is possible to authenticate the parties to each other and negotiate appropriate parameters of the session between them.

The process of negotiations between the client and the server is illustrated in Figure 2.4. It can be divided into 4 phases separated with horizontal broken lines. During the first phase, a logical connection must be initiated between the client and the server followed by the negotiation on the connection parameters.

The client sends the server a client _hello message containing data such as:

- **Version**: The highest SSL version supported by the client,
- **Random**: data consisting of a 32-bit time stamp and 28 bytes of randomly generated data. This data is used to protect the key exchange session between the parties of the connection.
- **Session ID**: A number that defines the session identifier. A nonzero value of this field indicates that the client wishes to update the parameters of an existing connection or establish a new connection on this session.

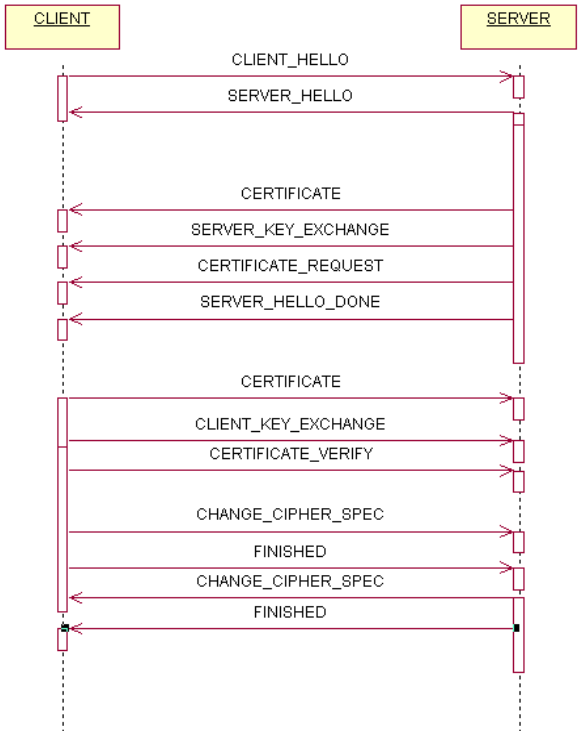



*Fig 2.4: SSL Handshake Protocol Activity Diagram*

## 2.3 ALGORITHMS USED

### 2.3.1 SHA-1 ALGORITHM

The SHA-1 (Secure Hash Algorithm) may be used with the DSA (Digital Signature Algorithm) in electronic mail, electronic funds transfer, software distribution, data storage, and other applications which require **data integrity assurance and data origin authentication** [10]. SHA-1 may also be used whenever it is necessary to generate a condensed version of a message. The algorithm for SHA-1 which is used in this project is explained in detail below.

A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value h (that is, h = H (m)). Hash functions with just this property have a variety of general computational uses, but when employed in cryptography, the hash functions are usually chosen to have some additional properties.

**Various steps involved in SHA-1 algorithm are**:

1. Pad message so its length is 448 mod 512
2. Append a 64-bit length value to message
3. Initialize 5-word (160-bit) buffer (A,B,C,D,E) to (67452301, efcdab89, 98badcfe, 10325476, c3d2e1f0)
4. Process message in 16-word (512-bit) chunks:
   - expand 16 words into 80 words by mixing & shifting
   - use 4 rounds of 20 bit operations on message block & buffer
   - add output to input to form new buffer value
5. Output hash value is the final buffer value.

**Compression function in SHA-1 involves the following steps**:

Each round has 20 steps which replaces the 5 buffer words thus:

**(A, B, C, D, E) <-( E+ f ( t, B, C, D )+ ( A<< 5 ) + Wt +Kt ), A, (B<< 30), C, D)**

where,

A, B, C, D refers to the 4 words of the buffer

t is the step number

f (t, B, C, D) is nonlinear function for round

Wt is derived from the message block

Kt is a constant value whose value varies with the step; different values of it are (5A827999, 6ED9EBA1, 8F1B8CDC, CA62C1D6)

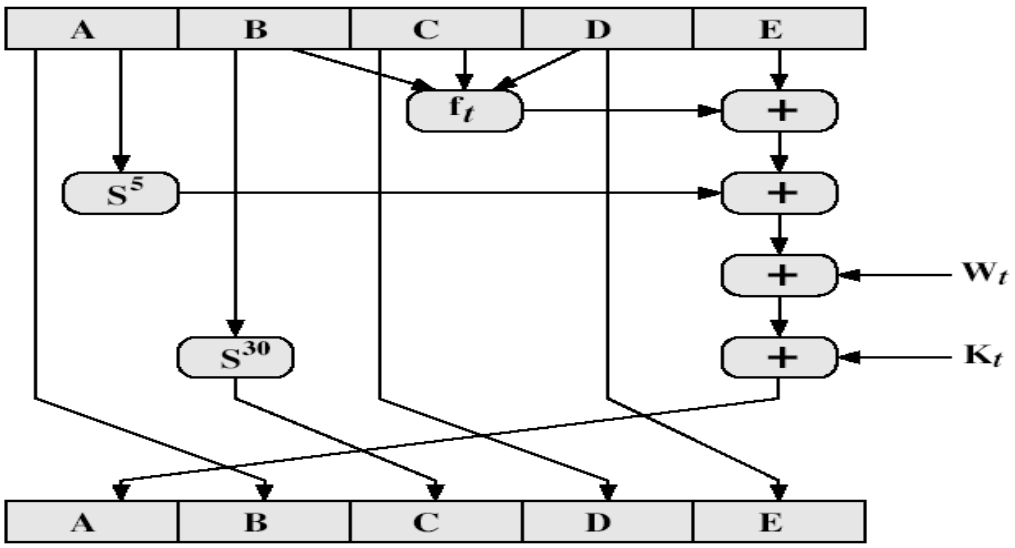



*Fig 2.5 SHA-1 Algorithm*

### 2.3.2 RSA ALGORITHM

[Rivest, Shamir and Adleman 1978] invented this algorithm. In contrast to DES, RSA's security relies on a solid mathematical background. Although current RSA implementations are much slower than symmetric cryptosystems, RSA is not only used for key exchange. Increasing performance of computers could make RSA to a real alternative to symmetric cryptosystems for numerous applications.

A Simple explanation of RSA Algorithm in view to computer:

- Select two large primes at random - p, q
- Compute their system modulus N=p.q
- note ø(N)=(p-1)(q-1)
- Select at random the encryption key e
- where 1<e<ø(N), gcd(e, ø(N))=1
- solve following equation to find decryption key d
- e.d=1 mod ø(N) and 0≤d≤N
- Publish the public encryption key: KU={e, N}
- keep secret private decryption key: KR={d, p, q}
- To encrypt a message M the sender:
- Obtains public key of recipient KU={e, N}
- Computes: $C=M^e \bmod N$, where 0≤M<N

- To decrypt the ciphertext C the owner:
  Use the private key KR={d,p,q}
  Compute: $M=C^d \bmod N$

### 2.3.3 DES ALGORITHM

DES, an acronym for the Data Encryption Standard, is the name of the Federal Information Processing Standard (FIPS) 46-3, which describes the data encryption algorithm (DEA). The DEA is also defined in the ANSI standard X3.92.

The algorithm's overall structure is shown in Fig 2.6.There are 16 identical stages of processing, termed rounds. There is also an initial and final permutation, termed IP and FP, which are inverses (IP "undoes" the action of FP, and vice versa). IP and FP have almost no cryptographic significance, but were apparently included in order to facilitate loading blocks in and out of mid-1970s hardware.

Before the main rounds, the block is divided into two 32-bit halves and processed alternately; this criss-crossing is known as the Feistel scheme. The Feistel structure ensures that decryption and encryption are very similar processes — the only difference is that the subkeys are applied in the reverse order when decrypting. The rest of the algorithm is identical. This greatly simplifies implementation, particularly in hardware, as there is no need for separate encryption and decryption algorithms.

The $\oplus$ symbol denotes the exclusive-OR (XOR) operation. The F-function scrambles half a block together with some of the key. The output from the F-function is then combined with the other half of the block, and the halves are swapped before the next round. After the final round, the halves are not swapped; this is a feature of the Feistel structure which makes encryption and decryption similar processes.

#### 2.3.3.1 Key Schedule Of DES

Fig. 2.6 illustrates the key schedule for encryption — the algorithm which generates the sub keys. Initially, 56 bits of the key are selected from the initial 64 by Permuted Choice 1 (PC-1) — the remaining eight bits are either discarded or used as parity check bits. The 56 bits are then divided into two 28-bit halves; each half is thereafter treated separately. In successive rounds, both halves are rotated left by one or two bits (specified for each round), and then 48 subkey bits are selected by Permuted Choice 2 (PC-2) — 24 bits from the left half, and 24 from the right. The rotations (denoted by "<<<" in the diagram) mean that a different set of bits is used in each subkey; each bit is used in approximately 14 out of the 16 subkeys.

The key schedule for decryption is similar — the subkeys are in reverse order compared to encryption. Apart from that change, the process is the same as for encryption. The same 28 bits are passed to all rotation boxes.

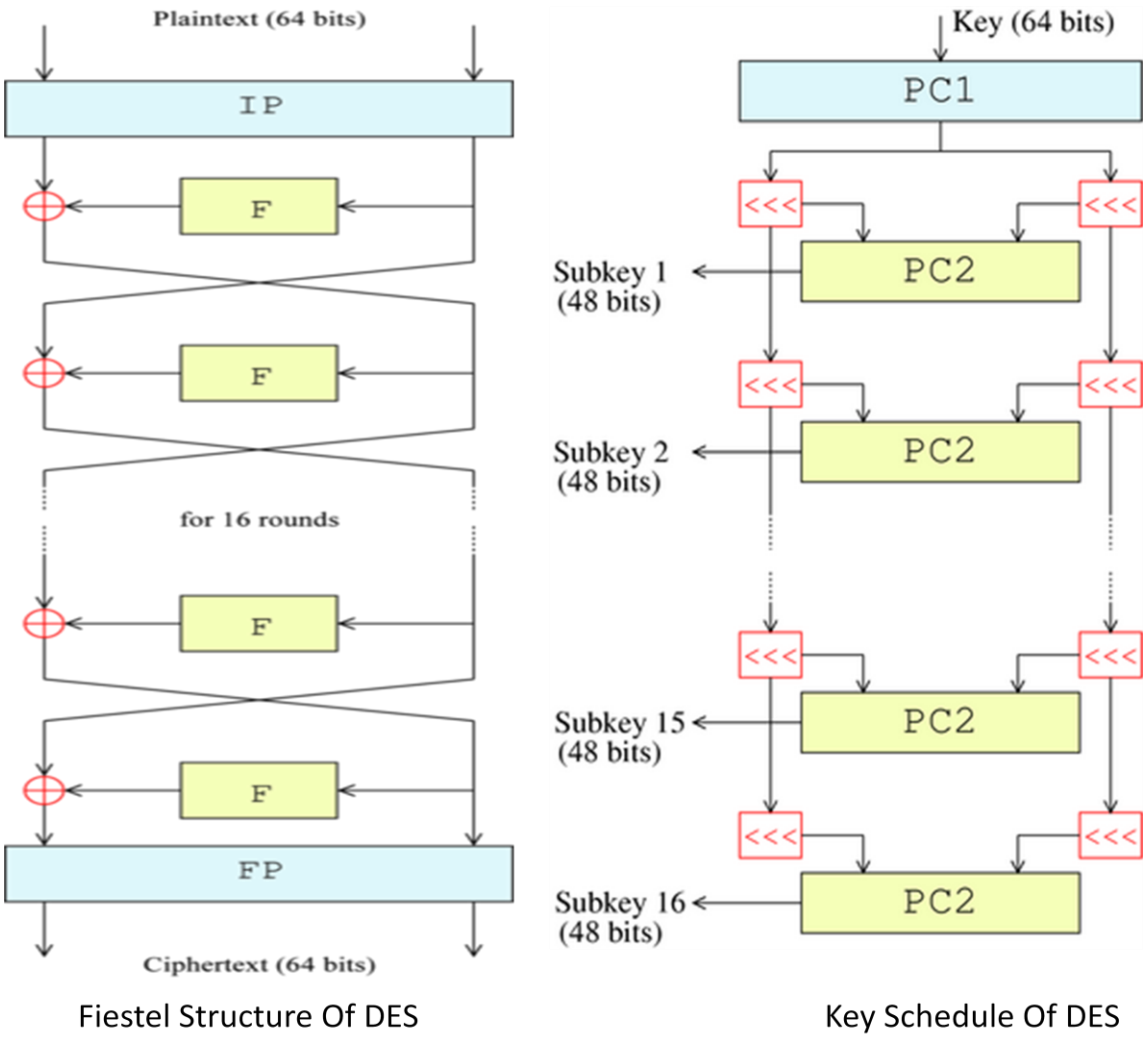



*Fig 2.6 Diagram Showing Fiestel Structure & Key Schedule Of Des*

### 2.3.4 MD5 Algorithm

The **MD5** Message-Digest Algorithm is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value, MD5 has been employed in a wide variety of security applications, and is also commonly used to check data integrity. MD5 was designed by Ron Rivest in 1991 to replace an earlier hash function, MD4. An MD5 hash is typically expressed as a 32-digit hexadecimal number.

MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit little endian integers) the message is padded so that its length is divisible by 512. The padding works as follows:

First a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits fewer than a multiple of 512. The

remaining bits are filled up with a 64-bit big endian integer representing the length of the original message, in bits, modulo 264. The bytes in each 32-bit block are big endian, but the 32-bit blocks are arranged in little endian format.

The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, denoted A, B, C and D. These are initialized to certain fixed constants. The main algorithm then operates on each 512-bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed rounds; each round is composed of 16 similar operations based on a non-linear function F, modular addition, and left rotation. Fig 2.6 illustrates one operation within a round. There are four possible functions F; a different one is used in each round:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

$\oplus, \wedge, \vee, \neg$ denote the XOR, AND, OR and NOT operations respectively.

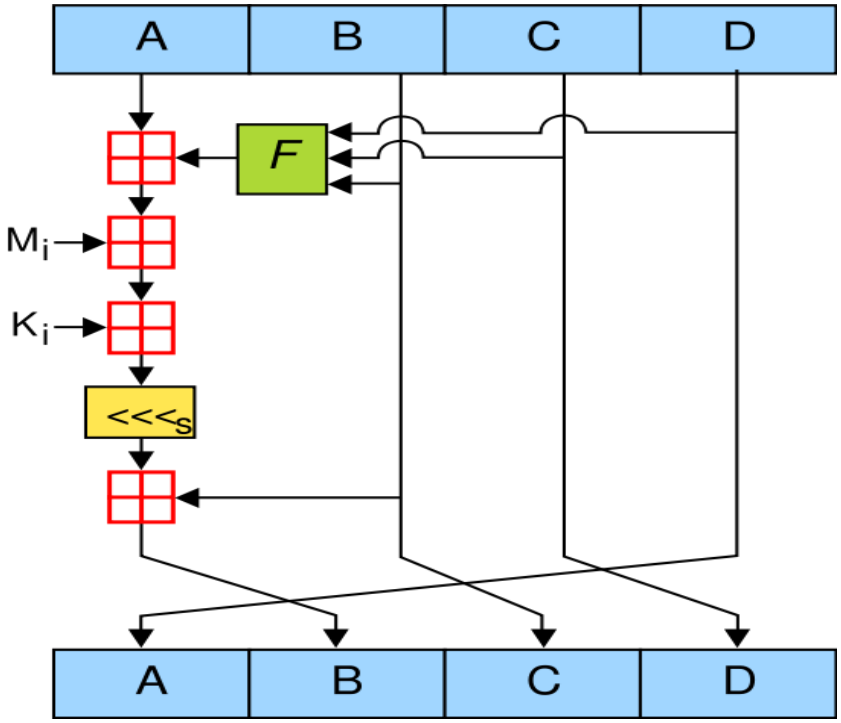



*Fig 2.7 A MD5 Operation { MD5 consists of 64 of these operations, grouped in four rounds of 16 operations}*

# Chapter 3
# SYSTEM DESIGN

Design is first and foremost an intellectual process. Contrary to popular belief, designers are not artists. They employ artistic methods to visualize thinking and process, but, unlike artists, they work to solve a client's problem, not present their own view of the world. If a design project, however, is to be considered successful – and that would be the true measure of quality – it will not only solve the problem at hand, but also add an aesthetic dimension beyond the pragmatic issues.

Design could be viewed as an activity that translates an idea into a blueprint for something useful, whether it's a car, a building, a graphic, a service or a process. The important part is the translation of the idea, though design's ability to spark the idea in the first place shouldn't be overlooked.

## 3.1 FLOW CHARTS

A flow chart is a graphical or symbolic representation of a process. Each step in the process flow is represented by a different symbol and contains a short text description of the process step in the flow chart symbol. The flow chart symbols are linked together with arrow connectors (also known as flow lines).

The step by step procedure for successfully accessing this system is represented by the system flow diagram in the following way:

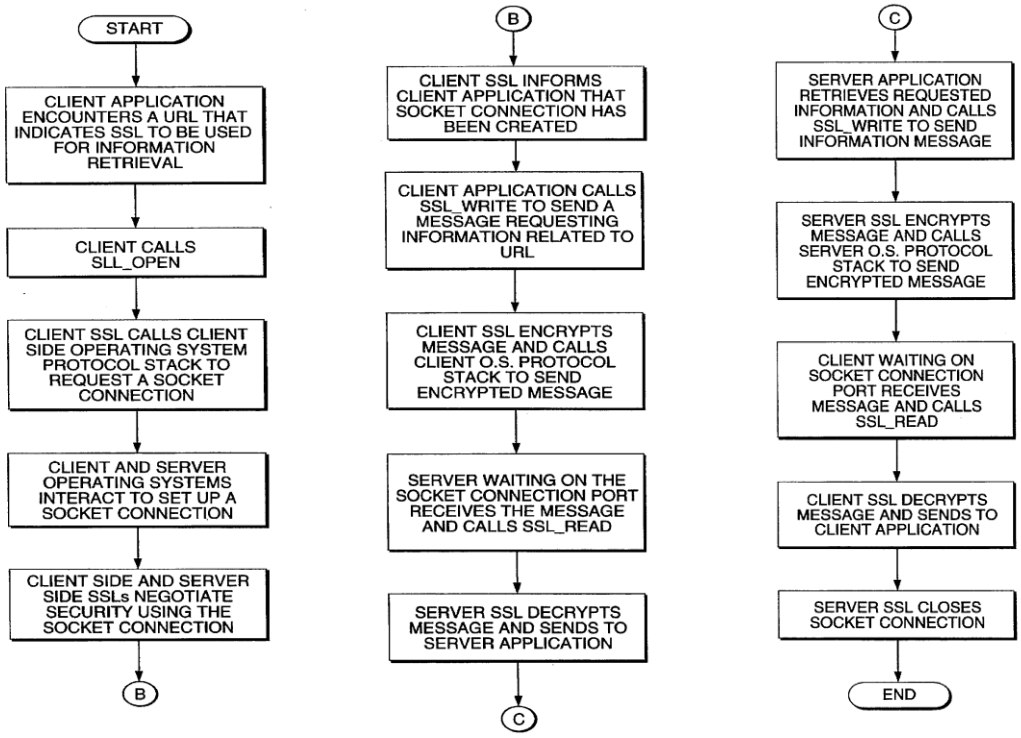



*FIG 3.1: SYSTEM FLOW DIAGRAM*

## 3.2 SOFTWARE MODULES

### 3.2.1 MODULES

There are 5 modules in this project:

1. **CLIENT MODULE**: This module contains the code for the client frame. This module also includes the code for sending and receiving and storing the applications on client side.
2. **SERVER MODULE**: This module contains the code for the server frame. This module also includes the code for sending and receiving and storing the applications on server side.
3. **HANDSHAKE MODULE**: This module contains the code for the implementation of the ssl handshake protocol on both client and server side. This module implements RSA key exchange algorithm.
4. **RECORD PROTOCOL MODULE**: This module implements ssl record protocol. For this, it implements SHA1, MD5 and DES encryption-decryption algorithms.
5. **ALERT PROTOCOL MODULE**: This module contains the code for implementing ssl alert protocol.

The use case diagram for the system is shown in the fig 3.2.

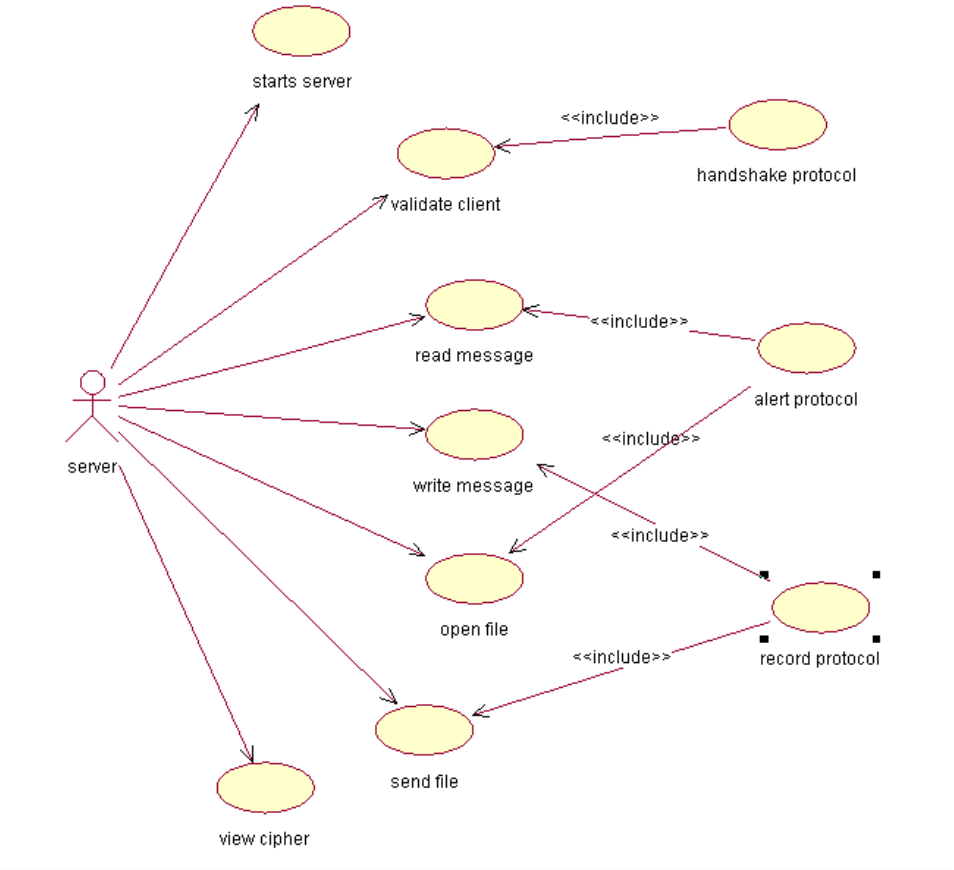



*Fig 3.2 Usecase diagram for ssl protocol*

### 3.2.2 Simulation model overview

It is appropriate to discuss the GUI and the system overview simultaneously because the user's familiarity of the system comes from the GUI. The GUI presents the user with enough information to effectively use system, but hides the complexity of the underlying architecture.

The effective run of the simulation model requires the user to have some background knowledge of ssl protocol. This simulation only shows the end results i.e the output being generated when a transfer occurs.

To start the simulation client just click on the jar file of client .This will start the client .In the similar fashion start the server. Another way to start the the simulation client and server is to use command prompt. Open the command prompt and browse to the folder where class binaries of the project is stored. Now execute the "java Clifrm" to start the client .Similarly, for server the command is "java Serfrm". To start the transferring of files click on start server. First, complete the handshake process before sending and message or files. Handshake is required to exchange the security parameters every time a session is initiated.

### 3.2.3 The Client Module :

The main component of this module is : **Clifrm.java**

Client module contains the client frame for representing a client. The top label represents the current version of ssl i.e. v3 currently being in use. All the exchange parameters used in this simulation module are as per the ssl v3.0. The key exchange algorithm used here is only RSA and is shown in the key exchange label. The symmetric encryption algorithm used in this model is DES. Next, it has two hashing algorithm which lets users to choose which hashing algorithm he/she wants to use for generating message digest. Initiatiate handshake will initiate the handshake with the server. After completing the handshaking user may send a message/file by typing the message for the former or the file name(file has to be in the same directory as that of Clifrm.class) which he/she wants to transfer to the server. The received message can be viewed by clicking the view received message button. Similarly the received cipher text can be seen by clicking on view received cipher button.

### 3.2.4 The Server Module

The main component of this module is : **Serfrm.java**

The server module contains the frame for server which is very similar to the client frame. The first thing user needs to do is to start the server. After starting server and sooner the handshake is completed server can exchange message or file. Similar to the client server can choose the message digest algorithm to use for the given session

### 3.2.5 The handshake Protocol module

The main components of this module are**: Hcl.java , Hsr.java, Rsa.java**

The use case diagram of the protocol is shown in fig.

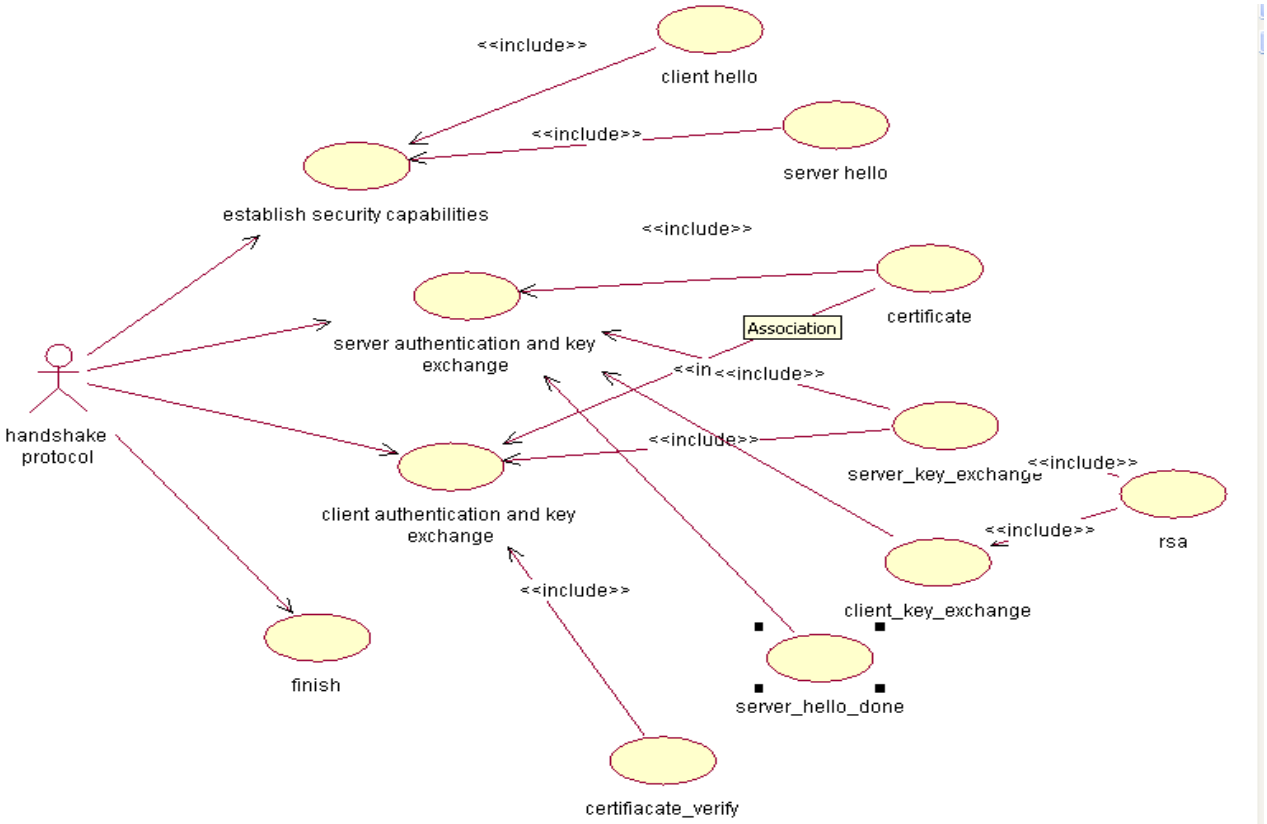



*Fig 3.3Usecase diagram for ssl handshake*.

This module is mainly responsible for exchanging ssl parameters. The type field decides what procedure is to follow next. There are 10 types defined in Hcl.java which are in accordance with the type field of ssl architecture. When type is it sets the message parameter required during client hello i.e. version(3), session id(a random number generated through random class),key exchange algo, cipher type(stream/block) , hash size (20 bytes for sha-1).During the client hello phase Rec_client.java is called which is responsible for creating connection with the server. It checks first whether the handshake is completed b/w the client or server or not. It then commit the handshake phase. Hsr.java on the other hand works on server side. It generates the server parameters and send them along with the server digital signature.

### 3.2.6 The Record Protocol Module

The main component of this module are : **Rec_clnt.java , Rec_ser.java, Msgbfc.java**

The record protocol is mainly responsible for creating an ssl payload. Rec_clnt creates the socket connection with the server and check whether a handshake has been completed between the client and server. It then takes an application message to be transmitted, fragments it into blocks of size 2^14 bits compresses the data if applicable[compression of data is not present in ssl v3.0]. It then calculates the MAC and add it along with the message blocks. For stream encryption, the compressed message plus the

MAC are encrypted ,the MAC is computed before encryption takes place and that the MAC is then encrypted along with the plaintext or compressed plaintext.

For block encryption, padding may be added after the MAC prior to encryption. The padding is in the form of a number of padding bytes followed by a one-byte indication of the length of the padding. The total amount of padding is the smallest amount such that the total size of the data to be encrypted (plaintext plus MAC plus padding) is a multiple of the cipher's block length.

Finally a header is appended 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. In this simulation 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 plaintext fragment (or compressed fragment if compression is used). The maximum value is 214 + 2048.Usecase diagram for the record protocol is shown in the fig 3.4.

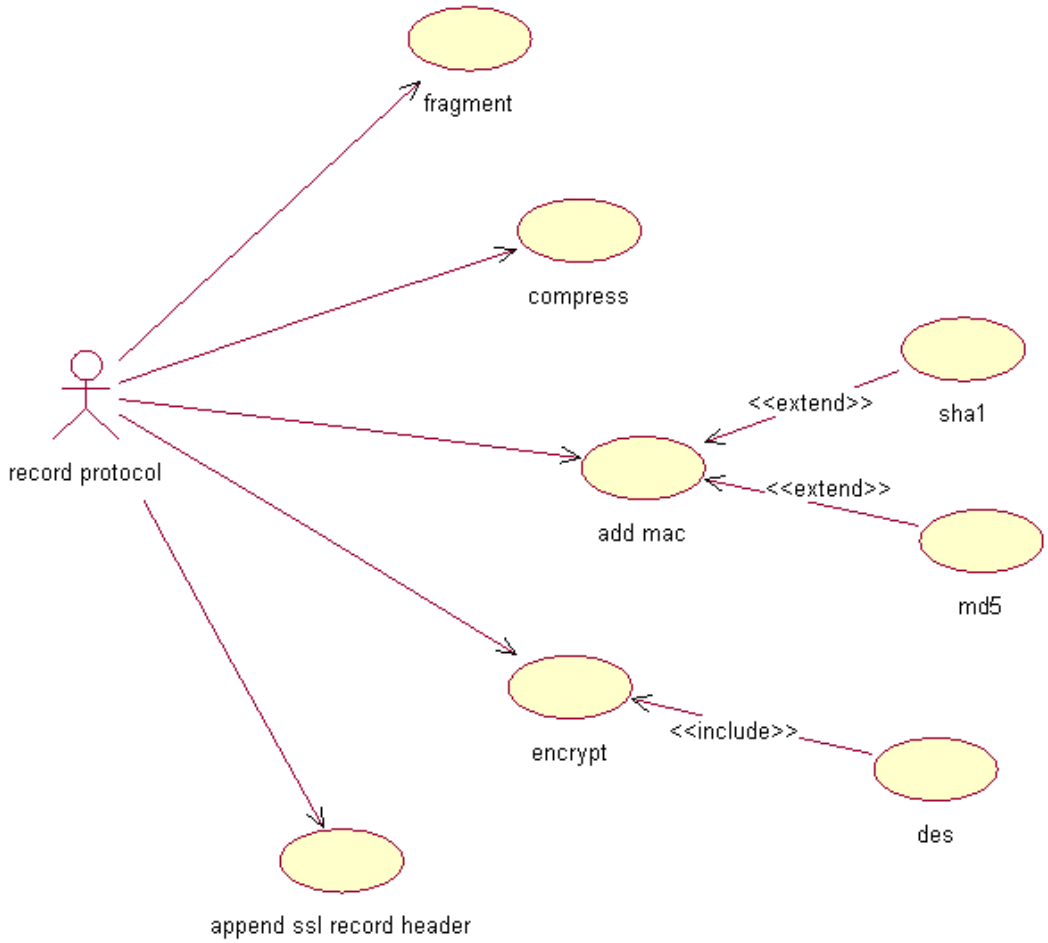



*Fig 3.4 Usecase diagram for record protocol*

### 3.2.7 The Alert Protocol Module

The main component of this module is: **Alert.Java**

The alert protocol is used to convey ssl related alerts to the peers entity. As wth the other application that uses ssl, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol in this protocol consists of two bytes. The first byte takes the value warning(1) or fatal(2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert. First, we list those alerts that are always fatal (definitions from the SSL specification).

● **unexpected_message:** An inappropriate message was received.

● **bad_record_mac:** An incorrect MAC was received.

●**decompression_failure:** The decompression function received improper input (e.g., unable todecompress or decompress to greater than maximum allowable length).

● **handshake_failure:** Sender was unable to negotiate an acceptable set of security parameters given the options available.

● **illegal parameter:** A field in a handshake message was out of range or inconsistent with other fields.The remainder of the alerts are the following:

● **close_notify:** Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a close_notify alert before closing the write side of a connection.

● **no_certificate:** May be sent in response to a certificate request if no appropriate certificate is available.

● **bad certificate:** A received certificate was corrupt (e.g., contained a signature that did not verify).

- **unsupported_certificate:** The type of the received certificate is not supported.
- **certificate_revoked:** A certificate has been revoked by its signer.
- **certificate_expired:** A certificate has expired.
- **certificate_unknown:** Some other unspecified issue arose in processing the certificate, rendering it unacceptable.

# Chapter 4

# System Specifications

Following are the software and hardware requirements for the proper functioning of the system**.**

## Software Requirements**:**

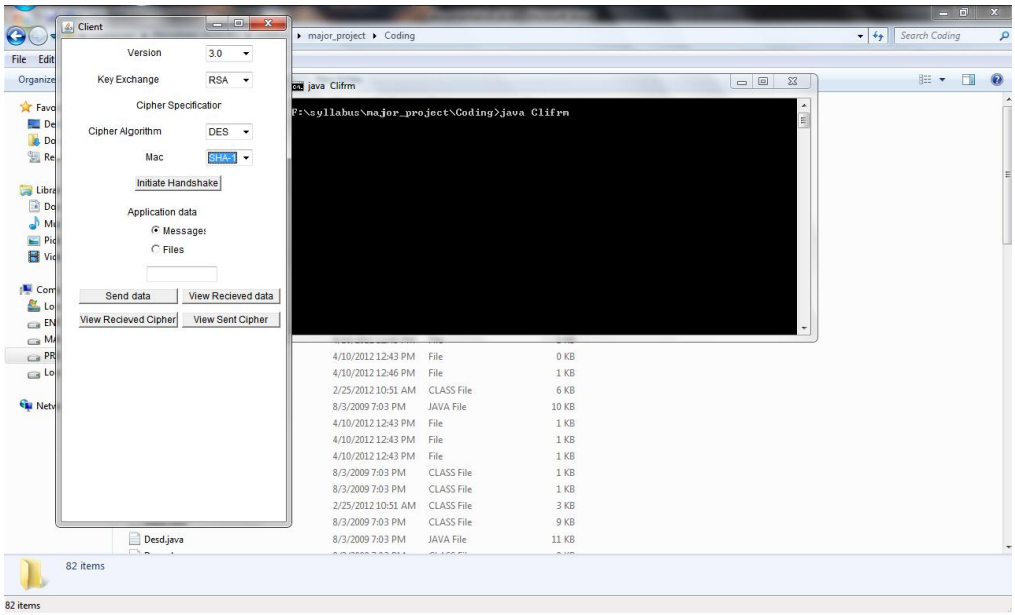
- Java Runtime Environment Version 1.5 and above
- Any System with the support to java run time**.**

## Hardware Requirements:

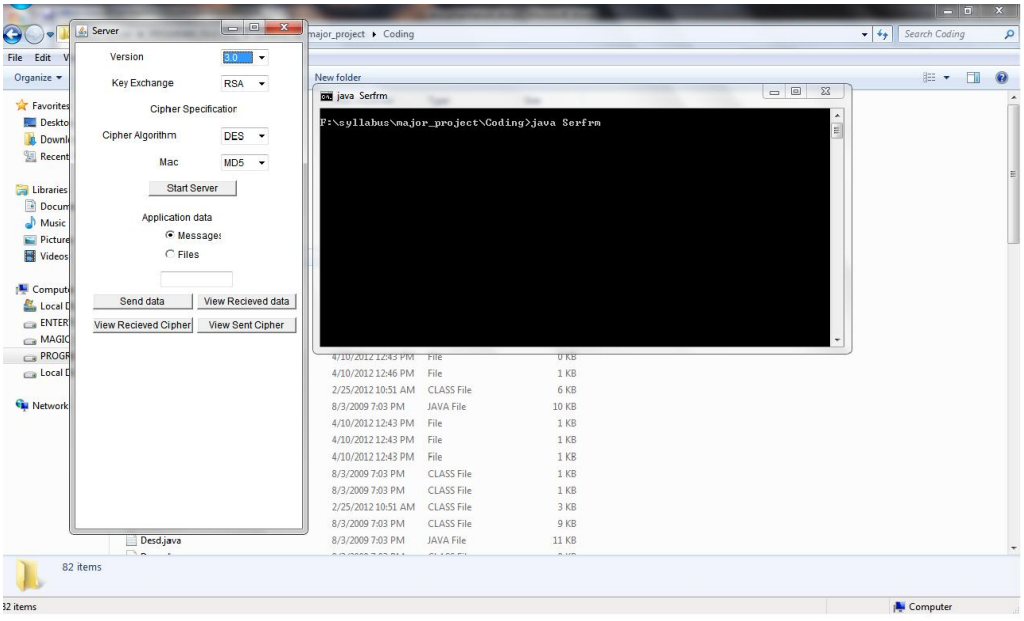
- 200 MB of available disk space
- 256 MB of RAM [Mainly to run java runtime environment].
- Network Interface Card ( Ethernet Card).
- Any processor that can run Java runtime binaries.

# Chapter 5

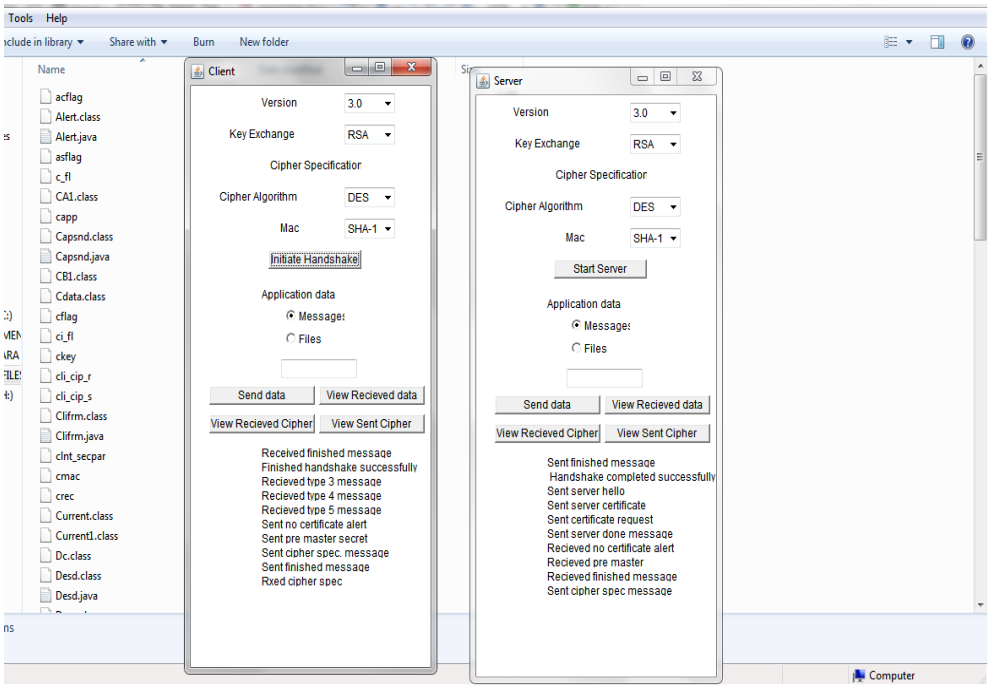
## SCREEN SHOTS



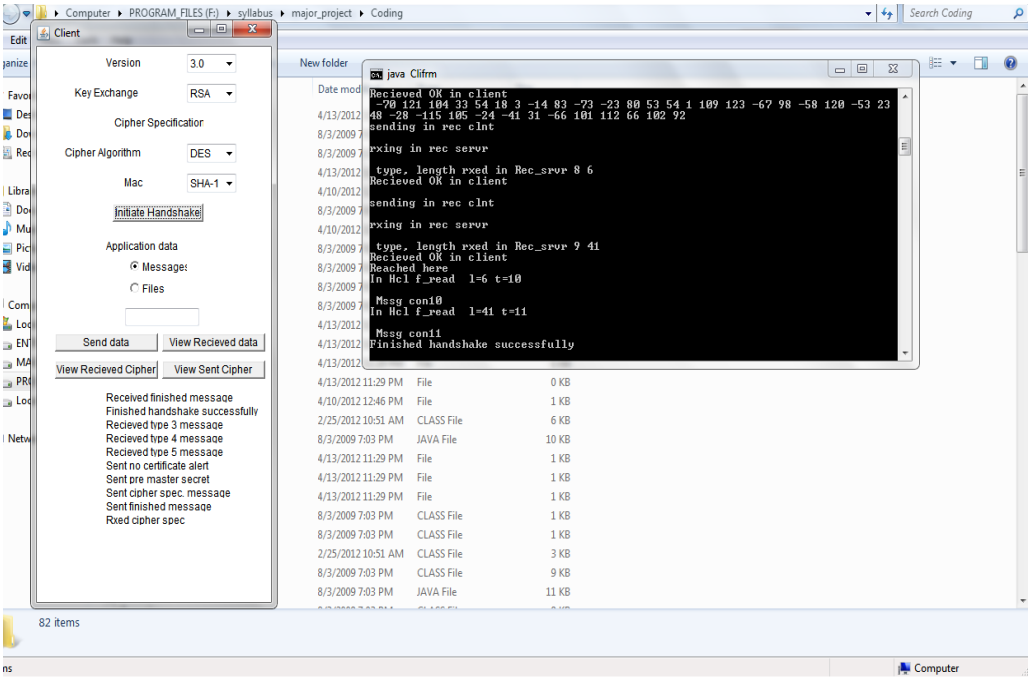

*5.1 CLIENT FRAME*



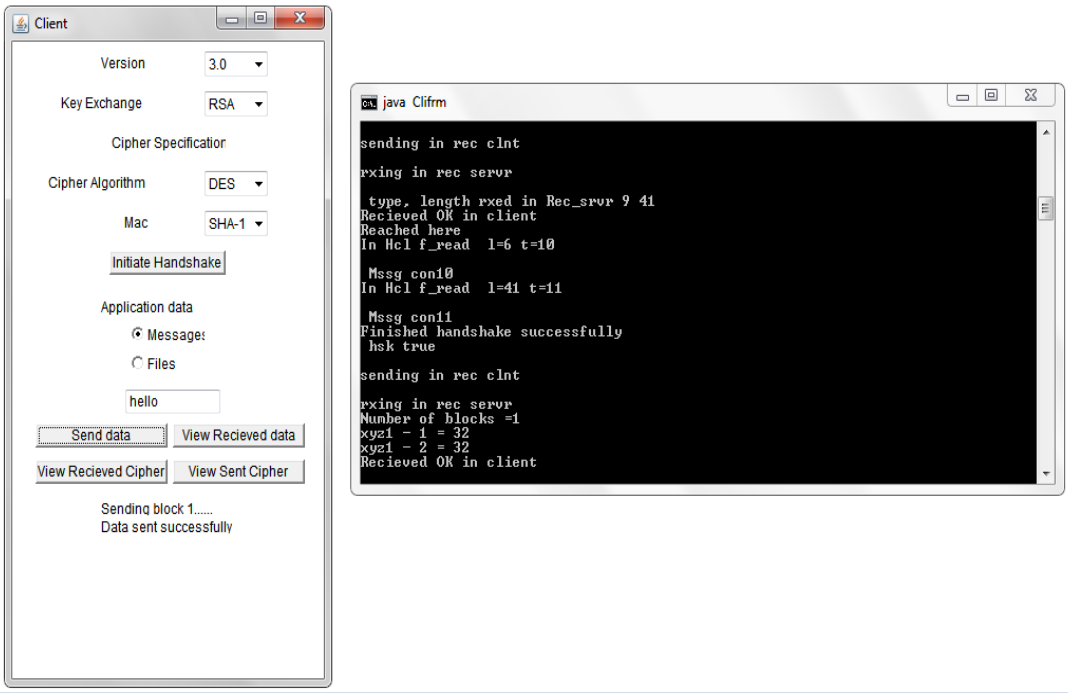

*5.2 SERVER FRAME*



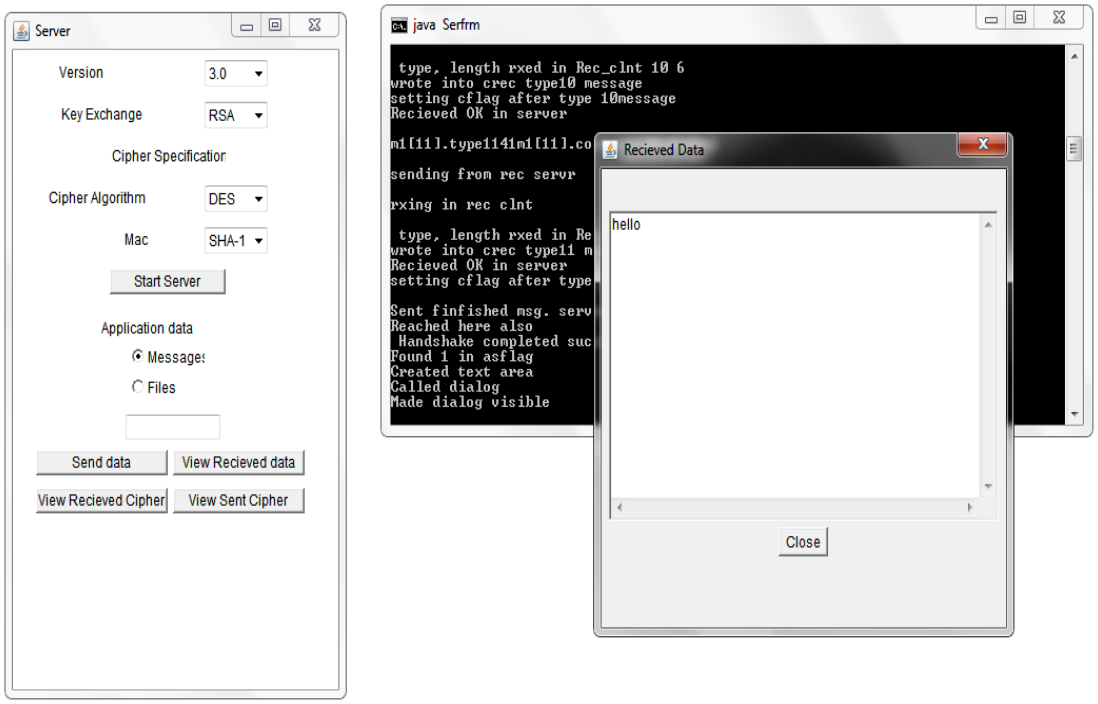

*5.3 CLIENT INITIATES HANDSHAKE*



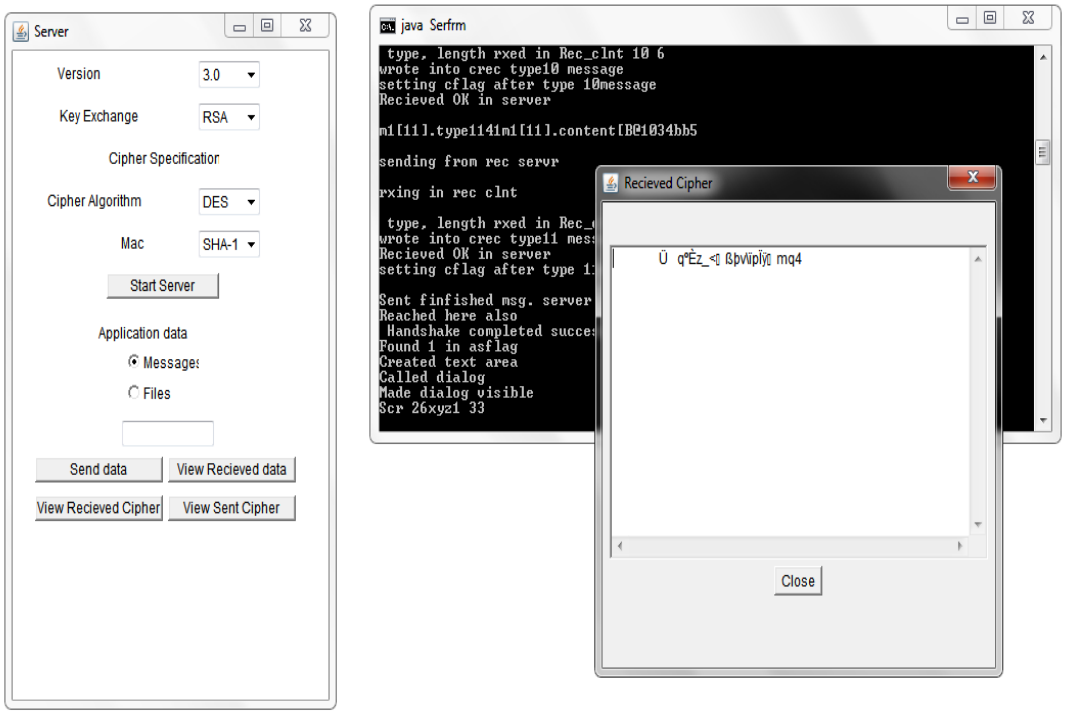

*5.4 HANDSHAKE SUCCESSFUL*



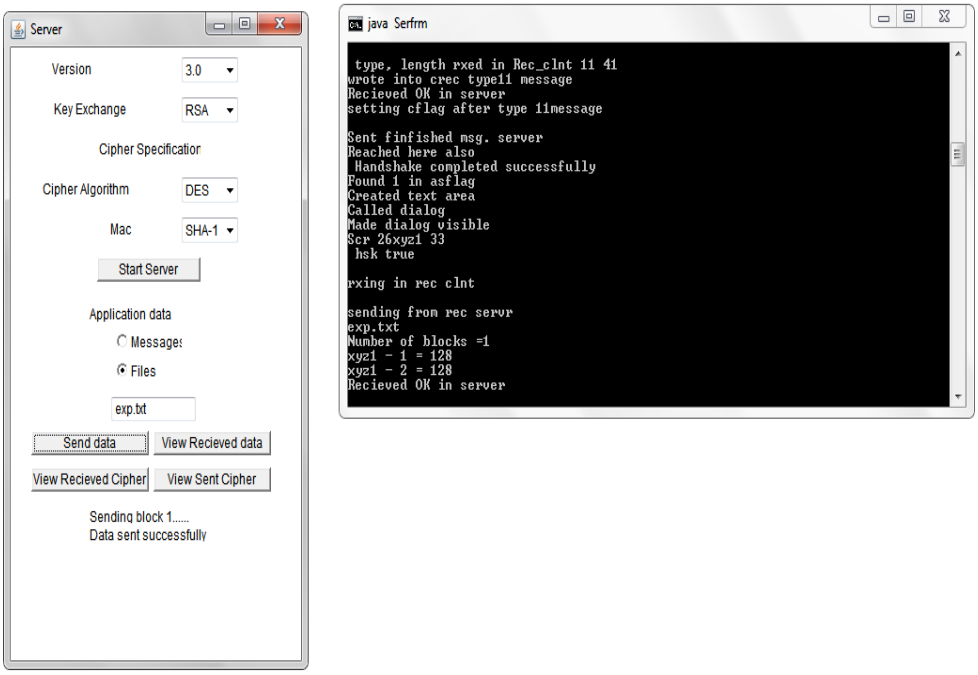

*5.5 CLIENT SENDING A MESSAGE*



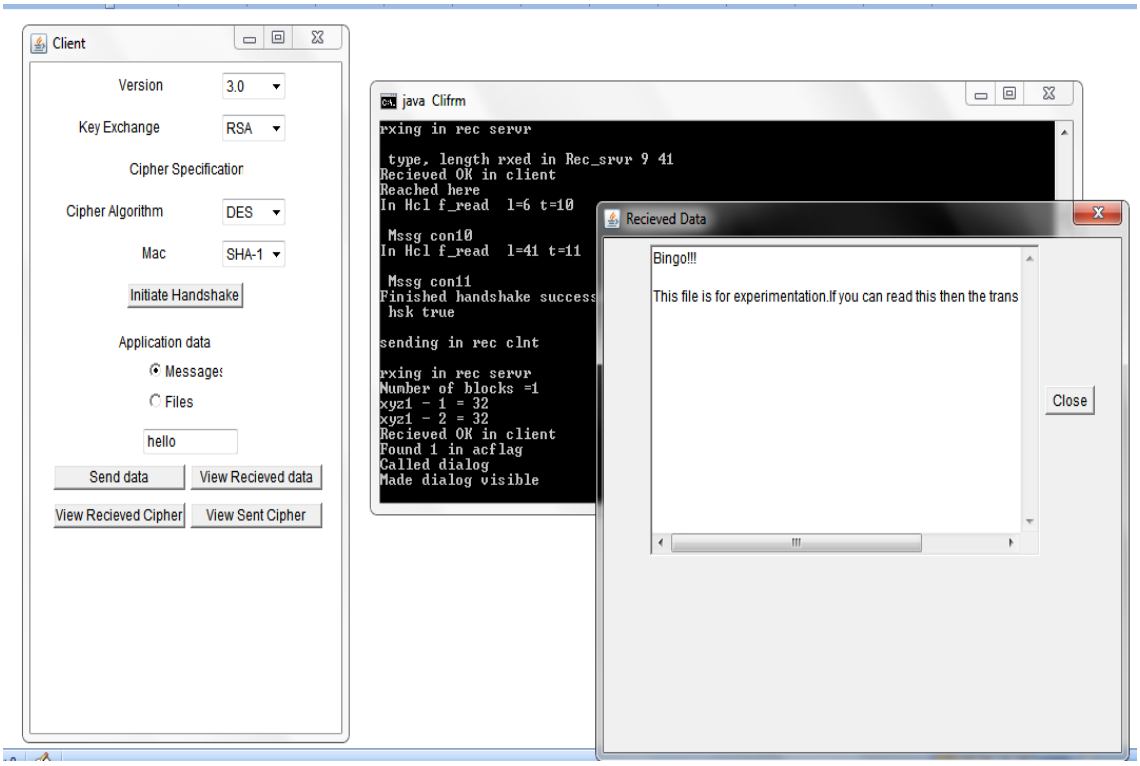

*5.6 SERVER VIEWING THE RECEIVED MESSAGE*



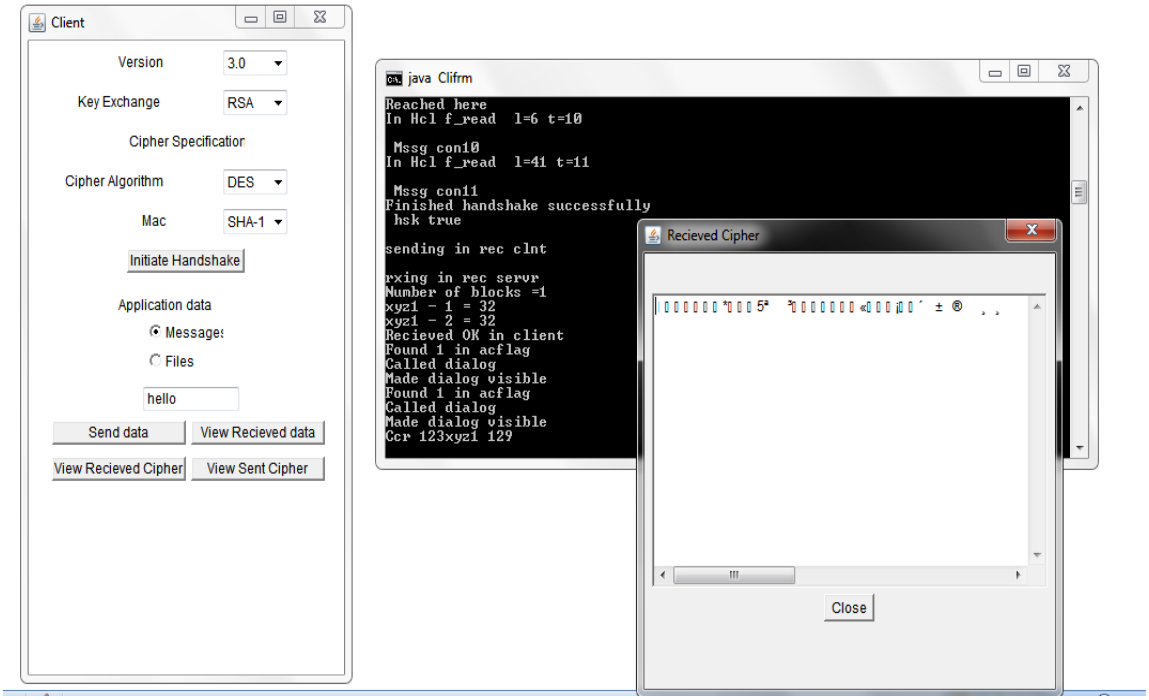

*5.7 SERVER VIEWING THE RECEIVED CIPHER*




5.*8 SERVER SENDING A FILE*




*5.9 CLIENT OPENIG THE RECEIVED FILE*




*5.10 Client Viewing the receiver cipher for the file*

# Chapter 6

# CONCLUSION

## 6.1 CONCLUSION

SSL is a very effective way of providing secure and reliable communication. The advantage offered by the establishment of a secure connection more than justifies the cost of encryption incurred at the server side and saves us from various kinds of attacks, which can result in loss and inconvenience.

The whole process can be viewed as – three SSL specific protocols (handshake, alert, change_cipher_spec) calling the SSL record protocol, which is responsible for passing the application data / messages after placing them in an encrypted payload. After successful completion of the handshake, both the client and the server become secure enough to pass their secret data to the record layer for fragmentation, encryption and transmission. The alert protocol is responsible for raising alerts (mentioning the severity of the failure) which when occur; decide the future of the connection (whether to be resumed or terminated) and any other necessary actions. The change_cipher_spec protocol updates the cipher-suite being used on the current connection. Thus, through an interleaved operation of these protocols, we achieve the overall intended effect of a 'secure tunnel' for communication, which is protected from attacks done for interception and alteration.

In this implementation, a secure file-transfer between the client and server is successfully achieved by performing: authentication of the parties, negotiation of cryptographic parameters, generation of session keys, establishment of sessions and finally encryption of all the communication between the entities. Here, we also provide the flexibility of transferring either a pre-existing file or a block taken as input from the user through a graphical, easy-to-understand interface. The end-user interface also shows the cipher-suite (set of cryptographic algorithms) and the exact sequence of events, leading to a clear insight into the working of the protocol. Thus, the final product is able to provide an effective and user-friendly means of making a transmission truly secure.

## 6.2 FUTURE WORK

In future the rate of the number of transactions that are handled by this system will be increased, with the same amount of authentication and security that which it provides now. Hence it can be successfully applied in the advanced levels.

The present work done in this system is only supported to a level of LAN and this can be further extended to the MAN and WAN levels.

**REFERENCES:**


**[1]** Cryptography and Network Security Principles and Practices, By William Stallings, Prentice Hall Publication

**[2]** Computer Networks , By Andrew S. Tanenbaum, Pressman Publication

**[3]** A Network Perimeter With Secure External Access , Frederick M. Avolio & Marcus J. Ranum, IEEE

**[4]** Complete Reference To Java , By Patrick Norton & Herbert Schild , Tata McGraw Hill Publication

**[5]** *http://en.wikipedia.org/wiki/Secure_Socket_Layer*

**[6]** *http://en.wikipedia.org/wiki/X.509*

**[7]** *http://www.openssl.org/, an open source bsd project*.

**[8]** *http://tools.ietf.org/html/rfc6101 ready for comment document for SSL3.0 specifications*

**[9]** *http://www.ietf.org/rfc/rfc3174.txt , a ready for comment document for SHA-1*

**[10]** *http://en.wikipedia.org/wiki/Data_Encryption_Standard*

**[11]** *http://orlingrabbe.com/des.htm*

**[12]** *http://tools.ietf.org/html/rfc1321 , a ready for comment document for DES*

**[13]** *http://cactus.eas.asu.edu/partha/Teaching/539-CommonFiles/SSL3_0Specification.htm*

**[14]** *http://en.wikipedia.org/wiki/HTTP_Secure*

**[15]** *http://en.wikipedia.org/wiki/Digital_signature*

**[16]** *http://en.wikipedia.org/wiki/Certificate_authority*