ABSTRACT

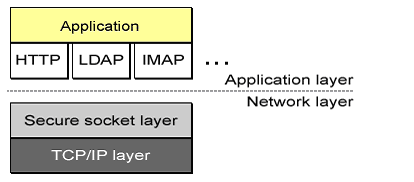
This project deals with the Secure Sockets Layer Protocol. Today, we rely heavily on the Internet for electronic data transfer, financial transactions, communication, and information retrieval and so on. Be it the success and smooth operation of a business enterprise or the convenience of the general public, the Internet has become the answer for many of our needs. But any breach in the confidentiality and integrity of the data transferred over the Internet can nullify the advantages offered and cause great loss and inconvenience to the user. Thus, the growing use of Internet necessitates the development of a protocol that would make ‘secure and reliable transfer of data’ possible and at the same time, would be flexible enough to incorporate changes as the need arises.

Secure Sockets Layer (SSL) is one such protocol that has gained widespread popularity because of its various attractions like efficiency, cryptographic security, interoperability, and extensibility.

This project deals with the implementation of the Secure Sockets Layer protocol. Here we are mainly concerned with the implementation of the client and server entities and the SSL transaction between them. This transaction comprises of the authentication, key exchange and bulk data transfer. Our implementation guarantees secure and reliable communication (message exchange) and data transfer (files) between the two entities.

INTRODUCTION

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.



# Figure:: 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.

# **SSL SERVER AUTHENTICATION**::allows 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**.TheSSL 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.

**\*\*OBJECTIVES::**

1. 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.
2. Ensuring data integrity: during a session, data cannot be either intentionally or unintentionally tampered with.
3. 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.

**\*\*MAJOR FUNCTIONS::**

**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 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.

A **cryptographic algorithm,** also called a **cipher**, is a mathematical function used for encryption or decryption. In most cases, two related functions are employed, one for encryption and the other for decryption.

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 with out the correct key is very difficult, and in some cases impossible for all practical purposes.

**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.

Original Data Encryption Original Data



Dear A: I have received the news….

Dear A: I have received the news…

Decryption

Symmetric Key Scrambled Data Symmetric Key

**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.

**CRYPTOGRAPHIC SECURITY:-**

SSL should be used to establish a secure connection between two parties.

**INTEROPERABILITY:-**

Independent programmers should be able to develop applications utilizing SSL 3.0 that will then be able to successfully exchange cryptographic parameters without knowledge of one another's code.

**Note:** It is not the case that all instances of SSL (even in the same application domain) will be able to successfully connect. For instance, if the server supports a particular hardware token, and the client does not have access to such a token, then the connection will not succeed.

**EXTENSIBILITY:-**

SSL seeks to provide a framework into which new public key and bulk encryption methods can be incorporated as necessary. This will also accomplish two sub-goals: to prevent the need to create a new protocol (and risking the introduction of possible new weaknesses) and to avoid the need to implement an entire new security library.

**RELATIVE EFFICIENCY:-**

Cryptographic operations tend to be highly CPU intensive, particularly public key operations. For this reason, the SSL protocol has incorporated an optional session caching scheme to reduce the number of connections that need to be established from scratch. Additionally, care has been taken to reduce network activity.

##### THE 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.

**THE 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.

**THE 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.

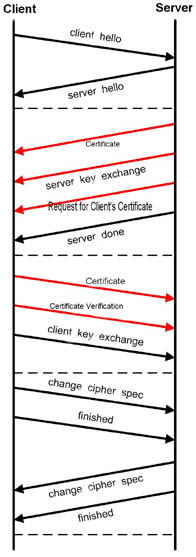
The process of negotiations between the client and the server is illustrated in Figure. It can be divided into 4 phases separated with horizontal broken lines. During the first phase, a logical connection must be initiated between the client and the server followed by the negotiation on the connection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

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

Version: The highest SSL version supported by the client,

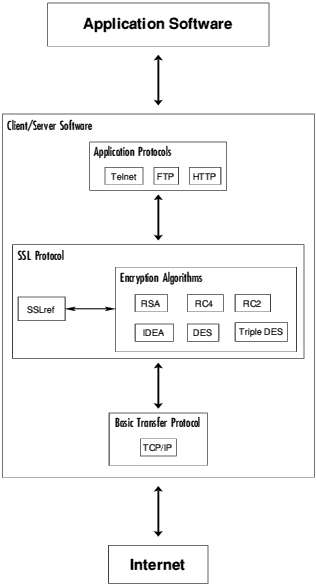
Random: data consisting of a 32-bit time stamp and 28 bytes of randomly generated data. This data is used to protect the key exchange session between the parties of the connection.

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.



\*\*TECHNICAL CONSTRAINTS::

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 the client.



**Diagram of relation between SSL, Applications and the Network**

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.

PROJECT ESTIMATES

\*\*HISTORY::

* Need for secure web communication.
* Netscape

i]Worried especially about credit card transaction over the web

ii]Also worried about ease of implementation since they wanted this to iii]be industry-standard, not proprietary

iv]SSLv1 – 1994

**SSLv2**

* SSLv2 also released in 1994

=>SSLv1 wasn’t widely implemented

* Rules for establishing secure connection.
* Rules for public key encryption.
* Optional certificate-based authentication for servers and even clients.
* Flexible

=>No specifically required encryption, compression, or key generation algorithm.

The first version of SSL was never released because of problems regarding protection of credit card transactions on the Web.

In 1994, Netscape created SSLv2, which made it possible to keep credit card numbers confidential and also authenticate the Web server with the use of encryption and digital certificates. The first public version of SSL, version 2, suffered from a number of security flaws, which have been fixed in SSLv3. As browsers nowadays still support SSLv2, and as it is still in use in some systems, we briefly sum up its security problems:

* SSLv2 has a weak MAC construction and relies solely on the MD5 hash function;
* SSLv2 does not have any protection for the handshake, so that a person-in-the-middle attack cannot be detected;
* finally, a truncation attack is possible, as SSLv2 simply uses the TCP connection close to indicate the end of data, so that the attacker can simply forge the TCP FINs and the recipient cannot tell that it is not a legitimate end of data.

Netscape strengthened the cryptographic algorithms and resolved many of the security problems in SSLv2 with the release of SSLv3. SSLv3 now supports more security algorithms than SSLv2.

Some minor modifications were made to increase security:-

1. The way cryptographic keys are expanded from the initially exchanged secret was improved;
2. The MAC construction was slightly modified into an HMAC;
3. And, implementations were required to include support for the Diffie-Hellman key agreement, the Digital Signature Standard, and Triple-DES encryption.

ANALYSIS

**PROBLEM DEFNITION:-**

The primary goal of the SSL Protocol is to provide privacy and data integrity between two communicating applications. The protocol is composed of two layers: the SSL Record Protocol and the SSL Handshake Protocol. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP), is the SSL Record Protocol. The SSL Record Protocol provides connection security that has two basic properties.

* **THE CONNECTION IS PRIVATE:** Symmetric cryptography is used for data encryption (e.g., DES, RC4, etc.) The keys for this symmetric encryption are generated uniquelyfor each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol). The record Protocol can also be used without encryption.
* **THE CONNECTION IS RELABLE:**  Message transport includes a message integrity check using a keyed MAC. Secure hash functions (e.g., SHA, MD5, etc.) are used for MAC computations. The Record Protocol can operate without a MAC, but is generally only used in this mod while another protocol is using the Record Protocol as a transport for negotiating security parameters.

Systems need to be represented as separate functional and physical models.

* + System functions change slowly with time while physical elements may change.
  + Dramatically
* The model provides evidence of how well the target is understood.
* The model is reverse-engineered from multi-source, sampled data.
* Total analysis is never completed

System Analysis is a process of examining the current system in order to develop the software design tool through better procedures and methods. It is the process of gathering and interpreting facts, diagnosing the problems and using the information to recommend approaches in the system.

System analysis is done to understand the problem, which the new system is going to solve. Such analysis typically requires a thorough understanding of existing system, and the part, which must be automated. Understanding the existing system is usually the starting activity in problem analysis. The goal of this activity is to understand the requirement of the new system that is to be developed.

**REQUIREMENT ANALYSIS:-**

Requirement analysis is to understand the problem and collect the requirements to solve the problem. It results in a system description and a set of requirements to solve the problem. Requirement analysis is a communication intensive activity. If the communication fails, even the rest technical approach will fail short.

Analysis is used to gain an understanding of an existing system and what is acquired of it. It is very important in the development of project life cycle. It helps us to develop a new package, which satisfies all the necessary goals. The fundamental activities of the system analysis phases are:-

i] Understanding the system clearly.

ii]Analysis the needs and creating a design for the new system.

iii]Difficulties in the existing system.

iv]Propose new system.

System analysis is an important activity that takes place when we are building new information or changing the existing one. It is therefore necessary to spend considerable time to properly understand the system and its problems.

**FEASIBILITY STUDY:-**

Once the problem is clearly understood, the next step is to conduct the feasibility study, which is a high level capsule version of the entire system analysis and design process. The objective is to determine whether the proposed system is feasible.

The feasibility study documents the analysis of a business problem, including the determination of whether it can be solved effectively. The operational (will it work?), economical (costs and benefits) and technical (can it be built?) aspects are part of the study. Results of the study determine whether the solution should be implemented.

The purpose of the feasibility study was two fold. Its first objective was to develop a verification methodology that bridged the gap between formal verification and simulation in a way that would integrate into an existing design flow. The second objective was to gather some quantitative evidence to support the idea that such a methodology would offer advantages over our existing simulation based verification techniques. This selection reflected the importance we place on the functional verification of leading edge processors; however the methodology is applicable to most designs. The choice of the decoder made it possible to investigate the problems of modeling and test specification while simplifying other tasks, in particular the translation from abstract to concrete tests.

The Four tests of feasibility have been carried out

1. **TECHNICAL FEASIBILITY::** The technical issues usually raised during the feasibility stage of the investigation include these:-
2. Does the necessary technology exist to do what is suggested (and Can it be Acquired)?
3. Does the proposed equipment have the technical capacity to hold the data required using the new system?
4. Will the propose system provide adequate responses to inquiries, regardless of the number or location of users?
5. Can the system be expanded if developed?
6. Are there technical guarantees of accuracy, reliability, ease of access, and data security?
7. **ECONOMICAL FEASIBILITY::** A system that can be developed technically and that will be used if installed must still be a good investment for the organization. Financial benefits must equal or exceed the costs. The financial and economic questions rose by analysts during the preliminary investigation a for the purpose of estimating the following:

1. The cost to conduct a full systems investigation

2. The cost of hardware and software for the class of application being considered 3. The benefits in the form of reduced costs or fewer costly errors 4. The cost if nothing changes (i.e., the proposed system is not developed)

1. **OPERATIONAL FEASIBILITY::** Proposed projects are beneficial only if they can be turned into information systems that will mettle organization’s operation requirements.

i] Is their sufficient support for the project from management? From users? If the current system is well liked and used to the extent that persons will not be able to see reasons for a change, there may be resistance.

ii] Are current business methods acceptable to the users? If they are not users may welcome a change that will bring about a more operational and useful system.

iii] Have the users been involved in the planning and development of the project? Early involvement reduced the chances of resistance to the system and change in general and increases the likelihood of successful projects.

iv] Will the proposed system cause harm? Will it produce poorer results in any respect or area? Will loss of control result in any area will accessibility of information be lost? Will individual customers be affected in an undesirable way will the system slow performance in any areas?

v] Issues that appear to be relatively minor in the beginning have way of growing into major problems after implementation therefore all-operational aspects must be considered carefully.

1. **SCHEDULED FEASIBILITY::** The time schedule required for the development of the project is important, since more development time affects machine time and cost of delay in development of other systems. As our goals, the requirements also change with time and technology; one should be first in incorporating them in the allocated schedule.

PROJECT RESOURCES

**SOFTWARE REQUIREMENT SPECIFICATION:-**

OPERATING SYSTEM: WINDOWS XP

LANGUAGE: JAVA2SDK 1.5.0. VERSION

DOCUMENTATION TOOL: MS-WORD

**HARDWARE REQUIREMENT SPECIFICATION:-**

PROCESSOR: PENTIUM

SPEED: 250 MHZ TO 833MHZ

RAM: 512 MB

HARD DISK: 40 GB

NETWORK: LAN

LAN SPEED: 100Mbps

**DEVELOPED BY:-**

CREATED BY: NETSCAPE (SSLv2)

YEAR: 1994

**TESTING**

**Software testing** is the process used to measure the quality of developed computer software. Usually, quality is constrained to correctness completeness, security, but can also include more technical requirements such as capability, reliability, efficiency, portability, maintainability, compatibility, and usability. Testing is a technical investigation that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate.

Any engineered product can be tested in one of two ways.

* Black-box testing
* White-box testing

**BLACK - BOX TESTING:-** Black-box testing, also called Behavioral testing, focuses on the functional requirements of the software. It enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a problem.

Black box testing attempts to find errors in the following categories:

* Incorrect or missing functions.
* Interface errors.
* Errors in data structures or external data base access.
* Behavior or performance errors.
* Initialization and termination errors.

**WHITE - BOX TESTING:-** White box testing, also called Glass-box testing is a test case design philosophy that uses the control structure described as part of component-level design to derive test cases.

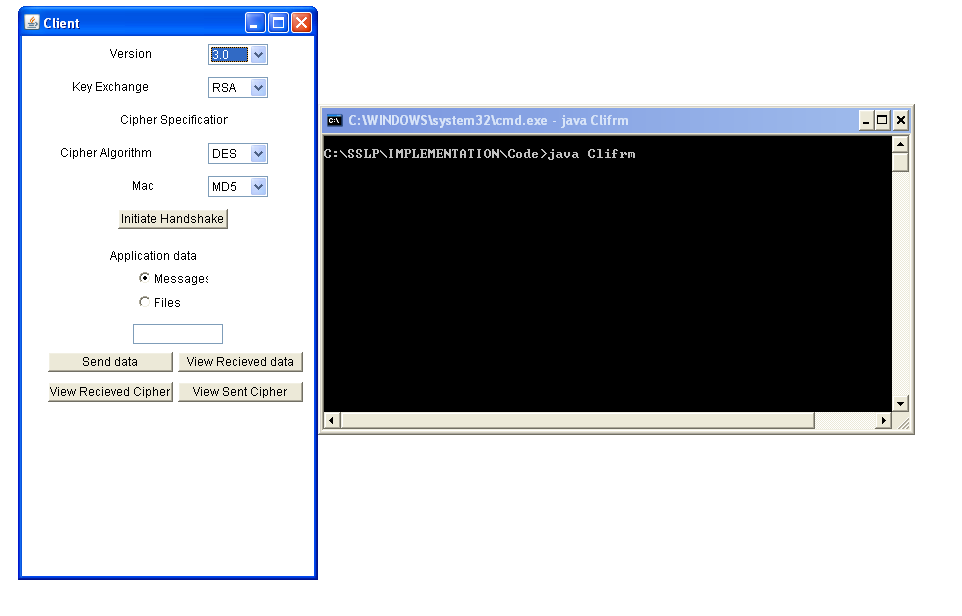
Using white box testing test cases can be derived that:

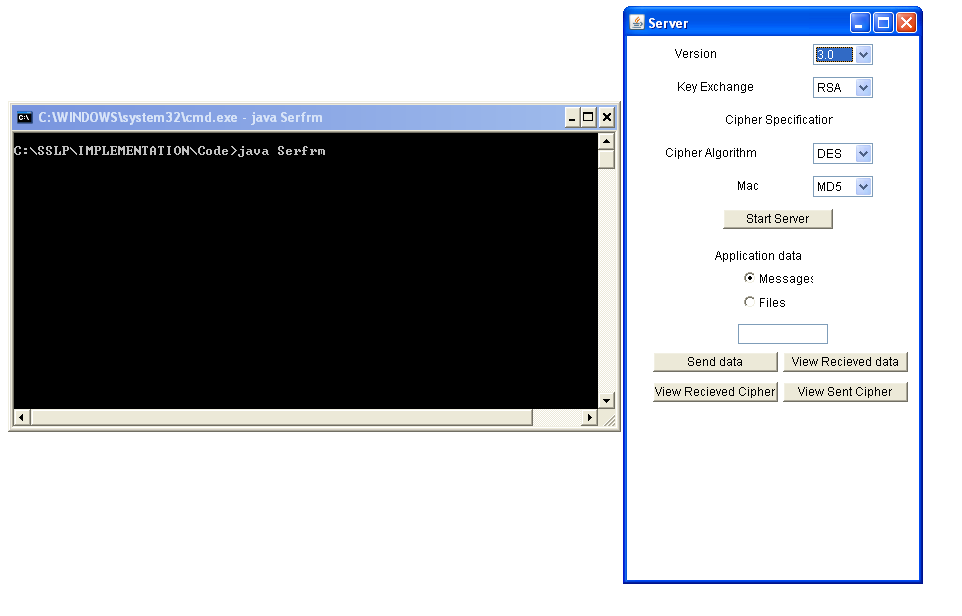
* Guarantee that all independent parts within a module have been exercised at least once.
* Exercise all logical decisions on their true and false sides.
* Execute all loops at their boundaries and within their operational bounds
* Exercise internal data structures to ensure their validity.

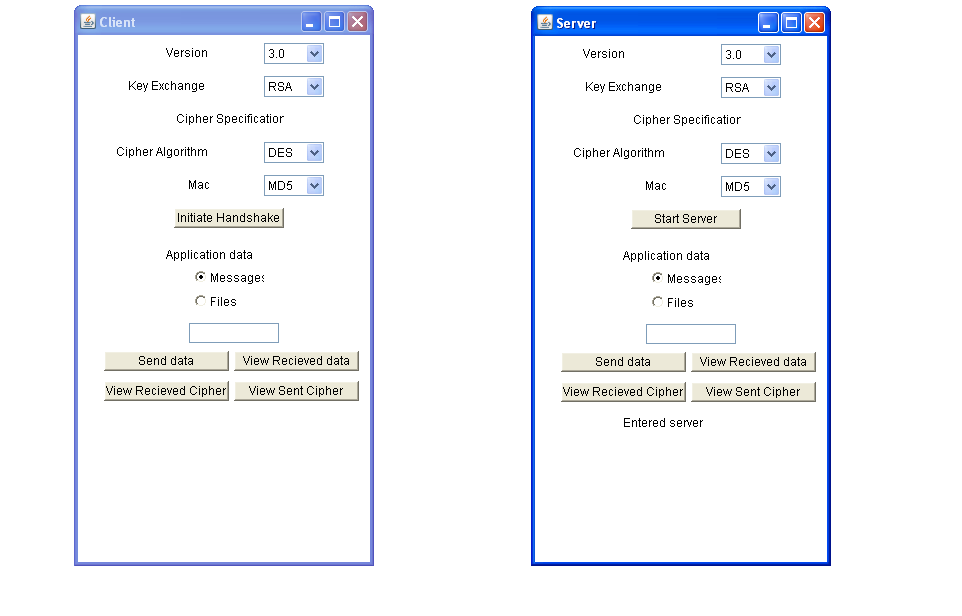
TEST CASE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No.** | **Server side** | **Client side** | **Message/file** | **Handshake Status** | **Result** |
| **1.** | Starts the server | Initiates Handshake | Types the message  In the text box. | Not yet connected | Warning (which shows that Handshake is not yet established) |
| **2.** | Starts the Server | Initiates the handshake | Types the message in the text box | Connection has been established | Message is successfully sent. |
| **3.** | Starts the message | Initiates the handshake | Types the file path of a file(eg: for .jpg, .bmp) | Connection has been established | Data will not be sent. |
| **4.** | Starts the message | Initiates the handshake | Types the file path( eg: for .txt, .doc) | Connection has been established | File is sent successfully |

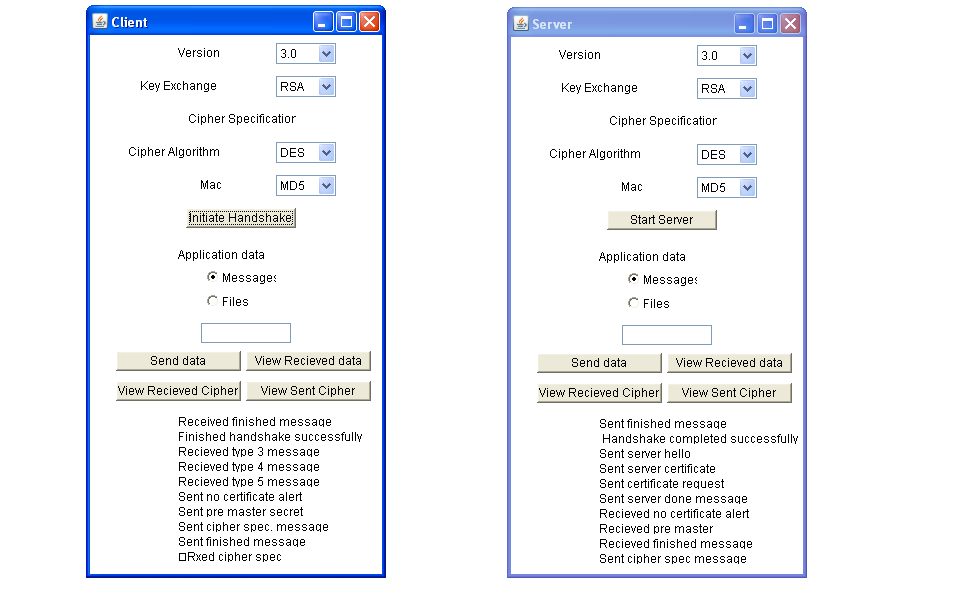
SCREEN SHOTS

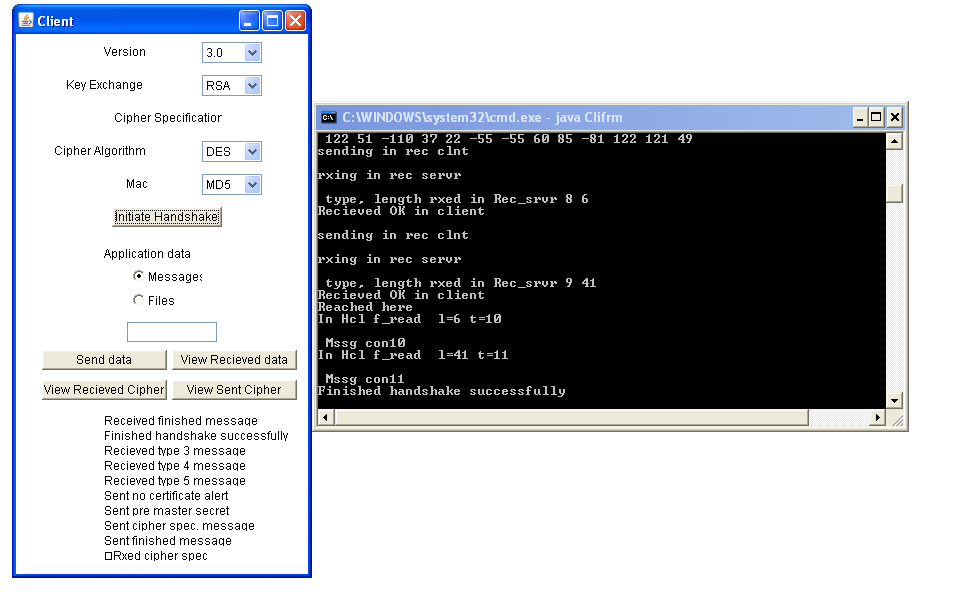
**CLIENT FRAME**

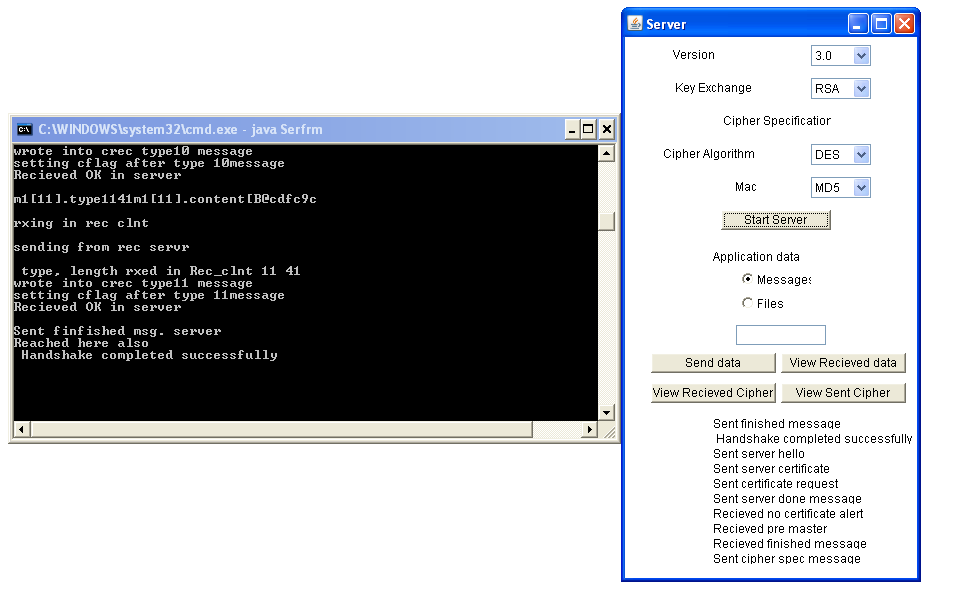
****  **SERVER FRAME**

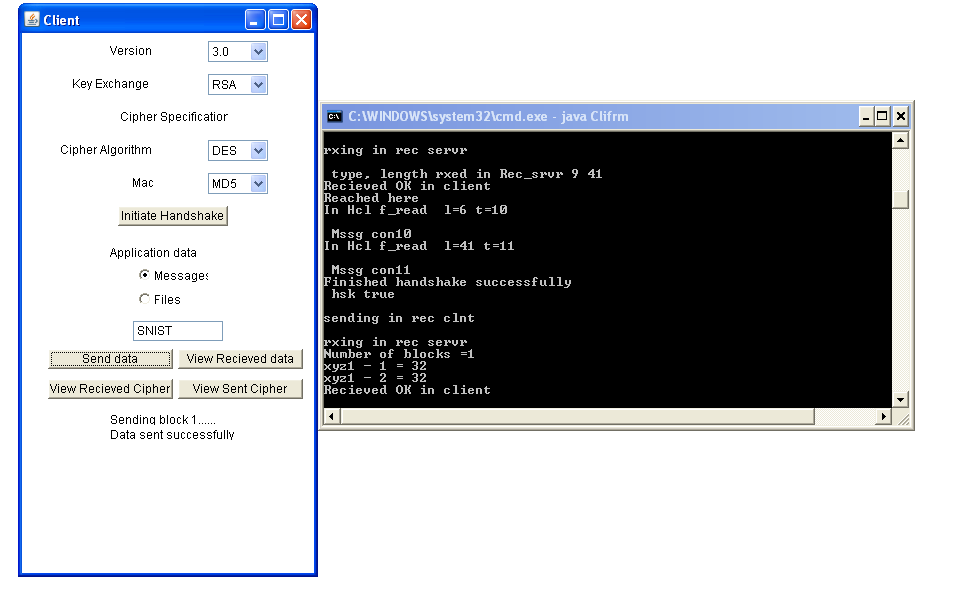
****

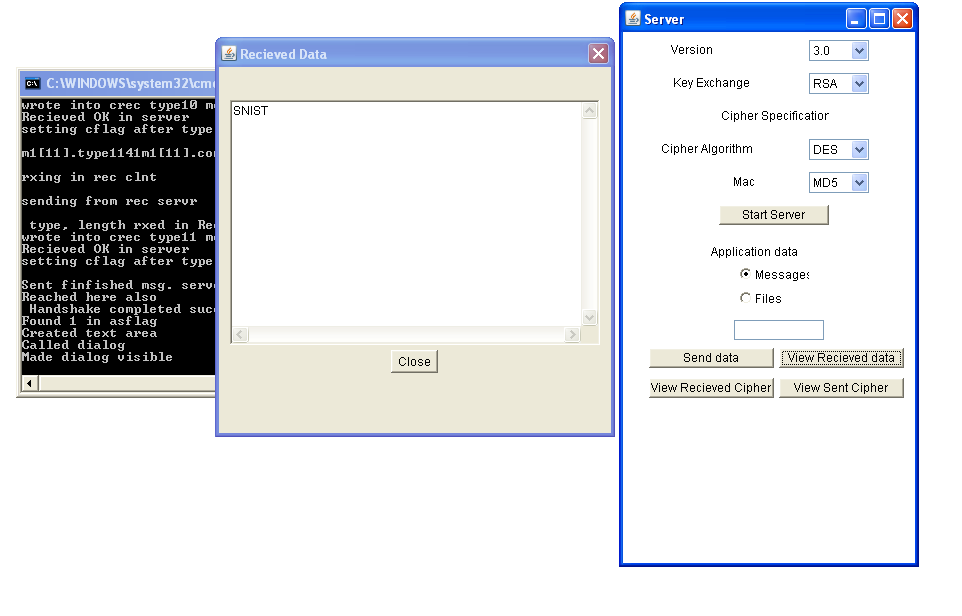
**SERVER ENTERED**

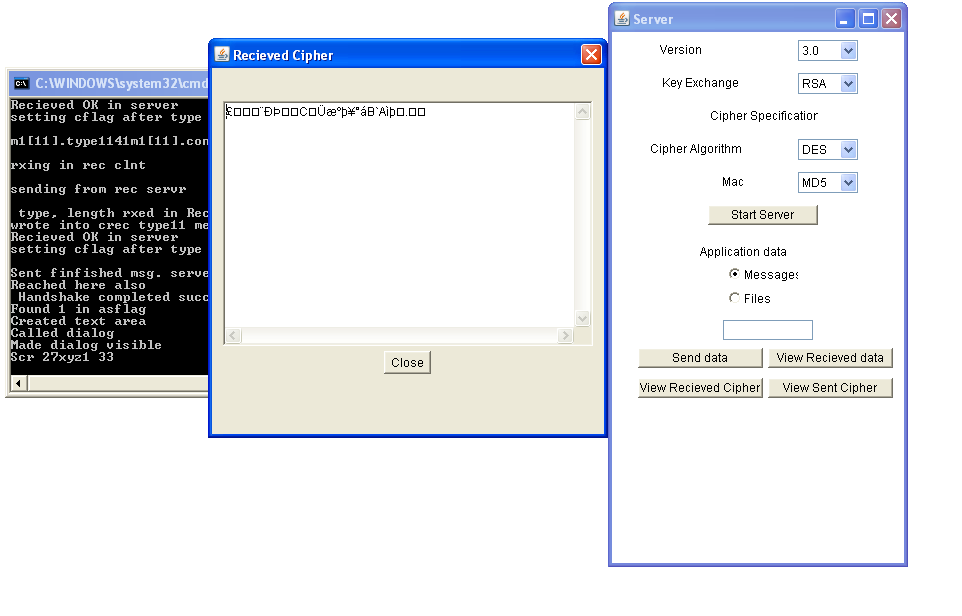
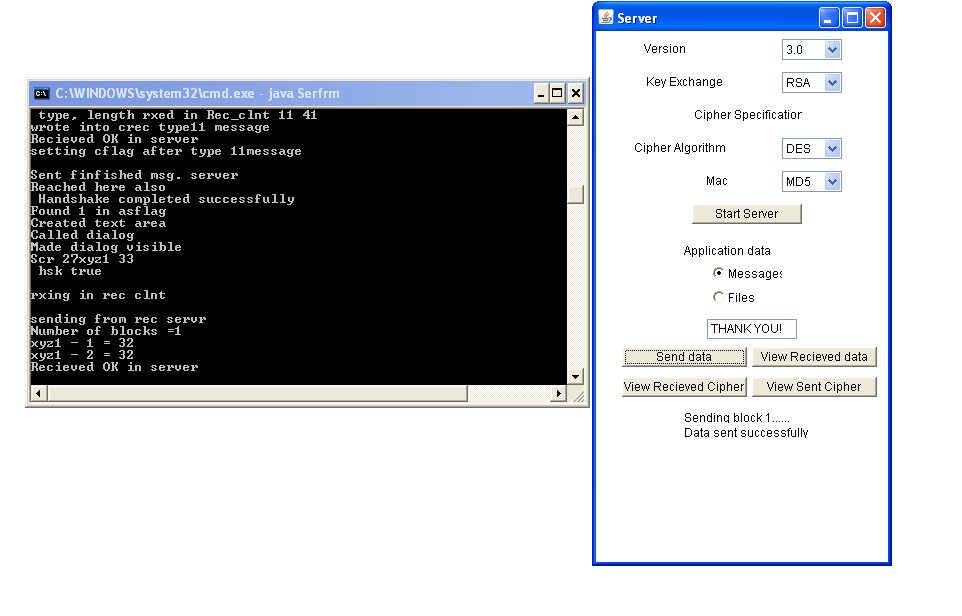
**** **CLIENT INITIATES HANDSHAKE**

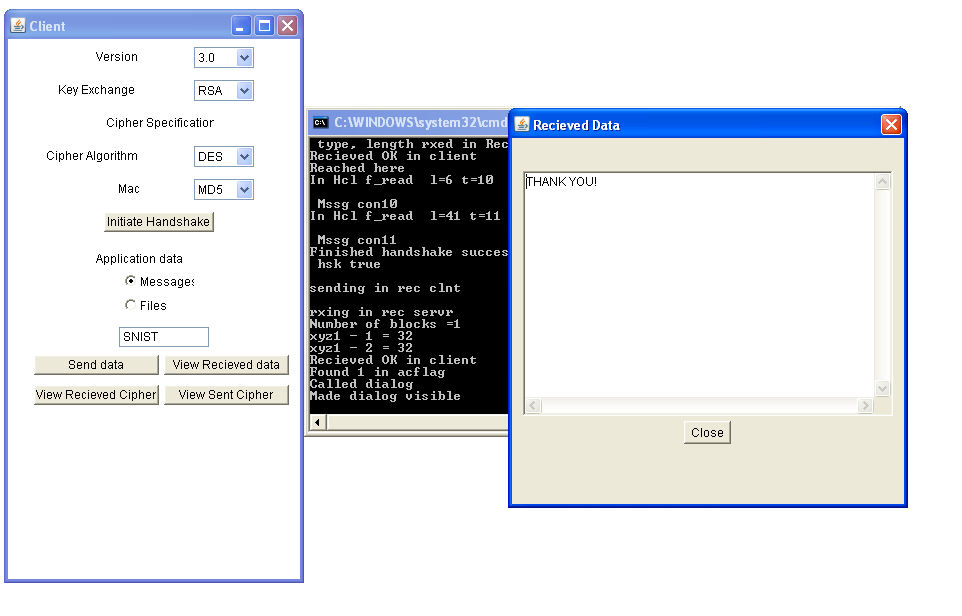
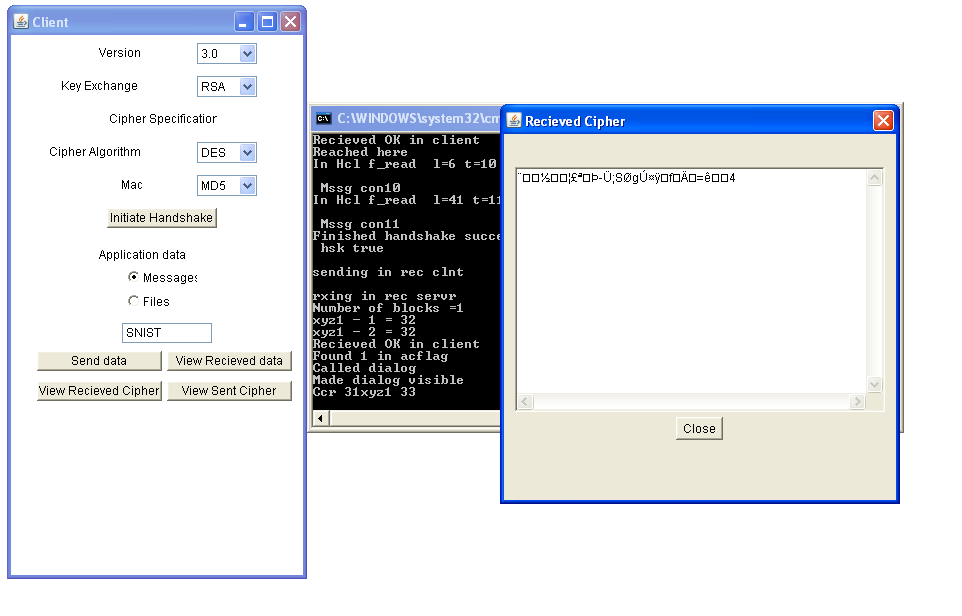
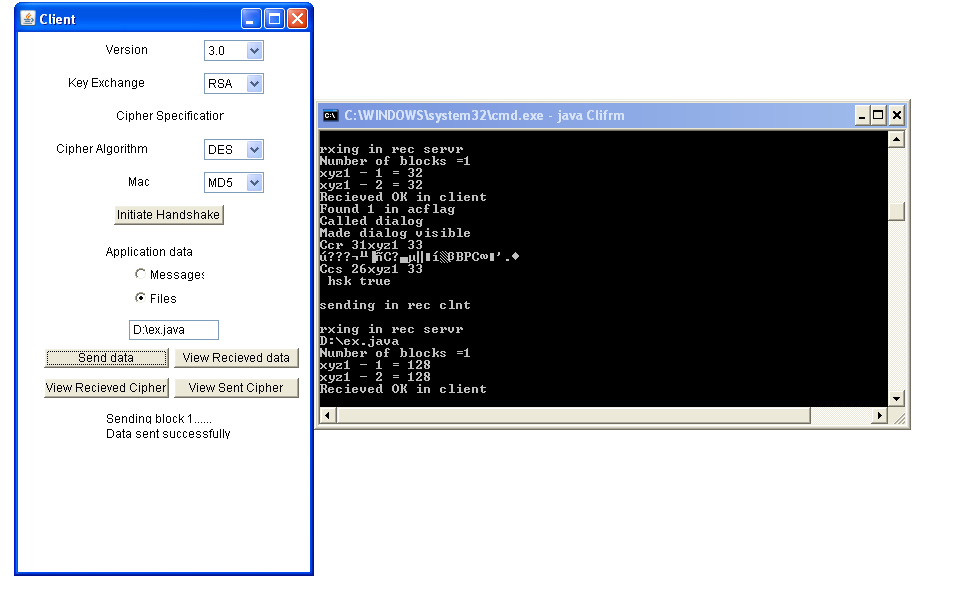
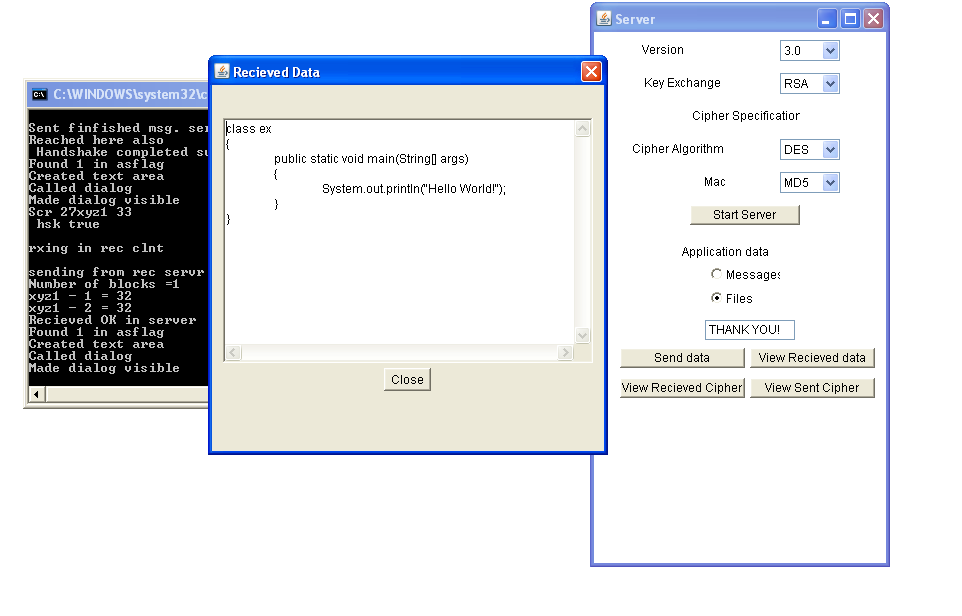
** HANDSHAKE SUCCESSFUL ON CLIENT SIDE**

****  **HANDSHAKE SUCCESSFUL ON SERVER SIDE**

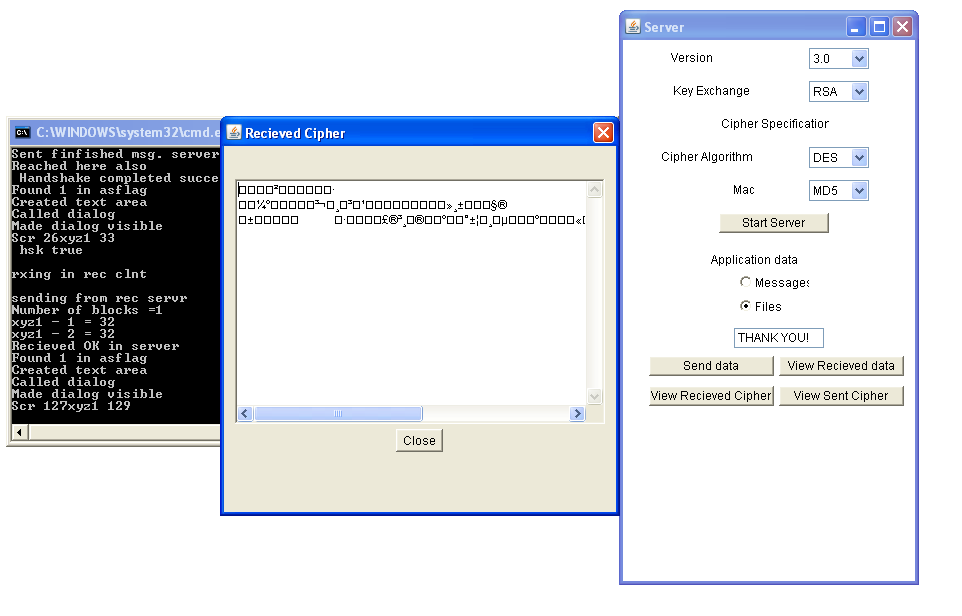
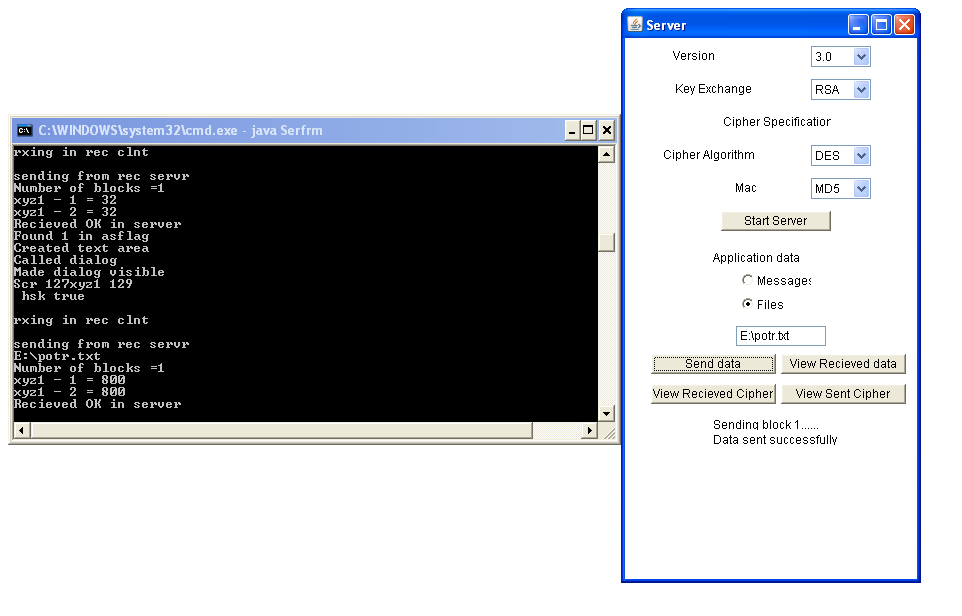
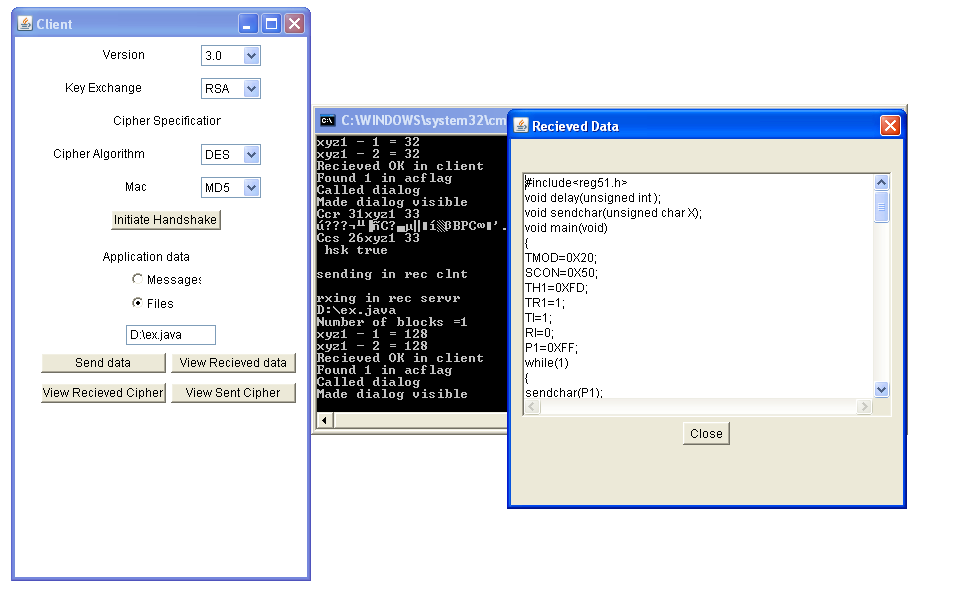
**CLIENT SENDING A MESSAGE**

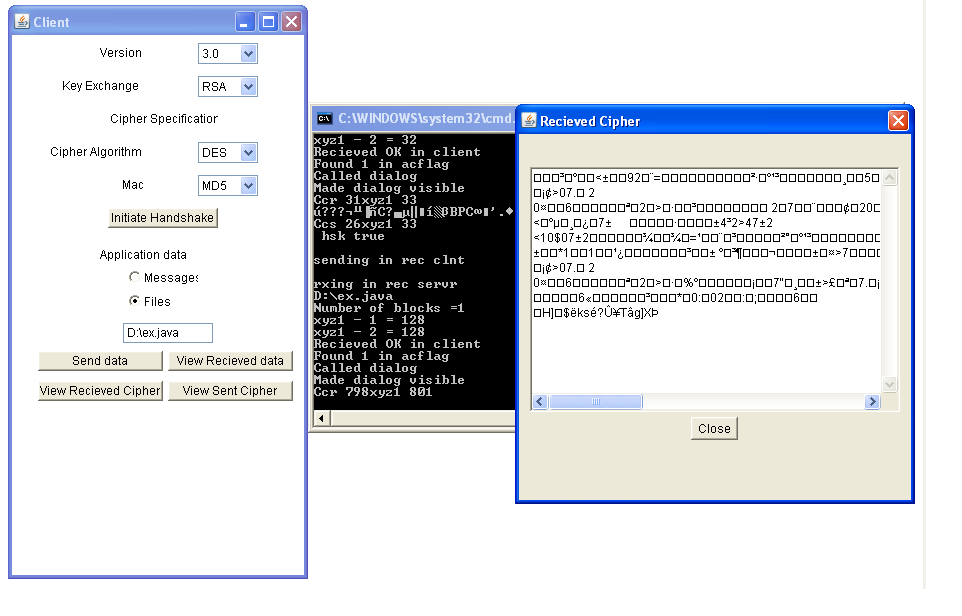
** SERVER VIEWING THE RECEIVED MESSAGE**

**** **SERVER VIEWING THE RECEIVED CIPHER SERVER SENDING A MESSAGE**

** CLIENT VIEWING THE RECEIVED MESSAGE CLIENT VIEWING THE RECEIVED CIPHER CLIENT SENDING A FILE**

**SERVER VIEWING THE RECEIVED FILE**

** SERVER VIEWING THE RECEIVED FILE CIPHER SERVER SENDING A FILE**  **CLIENT OPENIG THE RECEIVED FILE**

****

**CLIENT VIEWING THE FILE CIPHER**

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.

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.

BIBILOGRAPHY

**CRYPTOGRAPHY AND NETWORK SECURITY:-**

**Author:** WILLIAM STALLINGS

**Publisher:** PRESSMAN

**JAVA-2 COMPLETE REFERENCE**

**Author:** PATRICK NORTON AND HERBERT SCHILD

**Publisher:** TATA McGRAW HILL

**OBJECT ORIENTED SYSTEM ANALYSIS AND DESIGN**

**Author:** ALI BAHRAMI

**Publisher:** TATA McGRAW HILL

**IEEE Paper: A NETWORK PRIMETER WITH SECURE EXTERNAL ACCESS**

**Author**: FREDERICK M. AVOLIO AND MARCUS J. RANUM

**COMPUTER NETWORKS**

**Author:** ANDREW S. TANENBAUM

**Publisher:** PRESSMAN

ALGORITHMS USED

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.

The basic requirements for a cryptographic hash function are as follows:

* The input can be of any length.
* The output has a fixed length.
* *H*(*x*) is relatively easy to compute for any given *x*.
* *H*(*x*) is one-way.
* *H*(*x*) is collision-free.

A hash function *H* is said to be *one-way* if it is hard to invert, where ``hard to invert'' means that given a hash value *h*, it is computationally infeasible to find some input *x* such that *H*(*x*) = *h*. If, given a message *x*, it is computationally infeasible to find a message *y* not equal to *x* such that *H*(*x*) = *H*(*y*), then *H* is said to be a *weakly collision-free* hash function. A *strongly collision-free* hash function *H* is one for which it is computationally infeasible to find any two messages *x* and *y* such that *H*(*x*) = *H*(*y*).

1. **RSA ALGORITHM:-**

[Rivest, Shamir and Adleman 1978] invented an **asymmetric cryptosystem** named MIT cryptosystem. However, the name RSA is usually used today instead of MIT cryptosystem. 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.

An "RSA operation," whether encrypting, decrypting, signing, or verifying is essentially a modular exponentiation. This computation is performed by a series of modular multiplications.

1. **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 DEA has a 64-bit block size and uses a 56-bit key during execution (8 parity bits are stripped off from the full 64-bit key). The DEA is a symmetric cryptosystem, specifically a 16-round Feistel cipher and was originally designed for implementation in hardware. When used for communication, both sender and receiver must know the same secret key, which can be used to encrypt and decrypt the message, or to generate and verify a message authentication code (MAC). The DEA can also be used for single-user encryption, such as to store files on a hard disk in encrypted form environment.

DATA FLOW DIAGRAM (DFD)

