

IMPLEMENTING THE CLIENT/SERVER COMMUNICATION SECURITY

A DISSERTATION

*Submitted in partial fulfilment of the
requirements for the award of the degree*

of

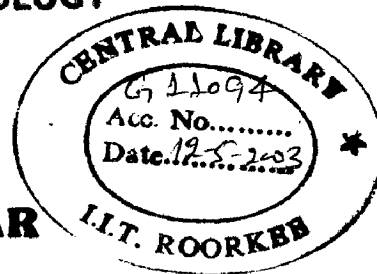
MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

By

NAVEEN KUMAR



**ER & DCI
NOIDA**

**IIT Roorkee-ER&DCI, Noida
C-56/1, "Anusandhan Bhawan"
Sector 62, Noida-201 307**

FEBRUARY, 2003

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation titled “**IMPLEMENTING CLIENT SERVER COMMUNICATION SECURITY**”, in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Information Technology**, submitted in **IIT, Roorkee – ER&DCI Campus, Noida**, is an authentic record of my own work carried out during the period from August 2002 to February, 2003 under the supervision of **Mr. M.K. Bhattacharya**, Senior Project Manager, Electronics Research and Development Centre of India, Noida.

I have not submitted the matter embodied in this dissertation for the award of any other degree.

Date: 25-2-03

Place: Noida


Naveen
(Naveen Kumar)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 25-02-03

Place: Noida


(Mr. M. K. Bhattacharya)
Senior Project Manager
ER&DCI, Noida

ACKNOWLEDGEMENT

I hereby take the privilege to express my deepest sense of gratitude to **Dr. Prem Vrat**, Director, Indian Institute of Technology, Roorkee, and **Mr. R.K. Verma**, Executive Director, ER&DCI, Noida for providing me with this valuable opportunity to carry out this work. I am also very grateful to **Dr. A.K. Awasthi**, our Programme Coordinator and **Dr. R.P. Agrawal**, our course coordinator for providing the best of the facilities for the completion of this work and constant encouragement towards the goal. I have no words to thank my guide, **Mr. M. K. Bhattacharya**, ER&DCI, Noida for his guidance and invaluable suggestions during the entire course of this work.

My sincere thanks are due to **Dr. P.R. Gupta** for the continuous inspiration and support she provided me with throughout the course of this dissertation. I am also grateful to **Mr. Munish Kumar** for the cooperation extended by him in the successful completion of this work.

It is impossible to mention the names of all those persons who have been involved, directly or indirectly, with this work and I extend my gratitude to all of them. However, I feel, I owe special thanks to all my friends who have helped me formulate my ideas and have been a constant support. I find myself short of words to thank my father, mother and sister who have always been by my side throughout my life.

(Naveen Kumar)

Enrolment No. 019026

CONTENTS

CANDIDATE'S DECLARATION	(i)
ACKNOWLEDGEMENT	(ii)
ABSTRACT	1
1. INTRODUCTION	3
1.1 Project Objective	3
1.2 Background	3
1.3 Need of System	4
1.4 Scope of Project	5
1.5 Organization of Dissertation	5
2. LITERATURE SURVEY	7
2.1 Client Server communication: Definition	7
2.2 Client Server Architecture	7
2.3 Methods of communications	8
2.4 NSFDC Client/Server Architectures	9
2.5 Security Attacks	10
2.6 Classification of Attacks	11
3. INTRODUCTION TO CRYPTOGRAPHY	13
3.1 Definition	13
3.2 Types Of Cryptographic Algorithms	14
3.3 Secret Key Cryptography	15
3.4 Public-Key Cryptography	16
3.5 Hash Functions	17
4. ANALYSIS OF THE PROBLEM	19
4.1 Authentication	19
4.2 Data Protection	19
4.3 Data integrity	22
4.4 Advantages/Disadvantages of client server architecture	24
4.5 Methods for General Data Transfer in Networks	25

5. PROBLEM DESIGN	29
5.1 Encryption	29
5.2 Algorithms	30
5.3 Key Databases	32
5.4 File hashing	34
5.5 Authentication	38
5.6 File Transfer	38
6. IMPLEMENTATION ASPECTS	41
6.1 Encrypting And Decrypting Data	41
6.2 Generating Keys	43
6.3 File Hashing	45
6.4 File Transfer Using TFTP	45
6.5 Authentication	47
7. INTERFACE AND DESCRIPTION	49
8. CONCLUSION	53
REFERENCES	55
APPENDIX A	57

ABSTRACT

The project entitled "Implementing Client/Server Communication Security "is concerned with end-to-end security mechanisms for file transfer using the client/server infrastructure. The application allows client and server to communicate in a way that data cannot be eavesdropped and Server always authenticates the clients before any data transfer.

The software has been developed as one of the modules of "Loan Accounting & Management Information System" for NSFDC being developed at ER&DCI, Noida. The application provides "security" which has three basic properties: Data confidentiality, Authentication and Data integrity. The Application is providing the sophisticated way to user to browse the file from the available storage devices, simple way to encrypting /decrypting and authenticating the files. It permits the end user to authenticate on server and send the files from one location to another in the Network. The objective is to develop a professional and general Application that is attractive to businesses and consumers.

The project “**Implementing Client/Server Communication Security** ”is concerned with end-to-end security mechanisms for file transfer using the client/server infrastructure which can be added to the main project for achieving the communication security goal.

1.3 Need Of System

Network and information security is likely to become a key factor in the development of the information society as networking plays a larger role in economic and social life. Security of electronic networks and information systems have been growing along with the rapid increase in the number of network users and the value of their transactions. Security has now reached a critical point where it represents a prerequisite for the growth of electronic businesses and the functioning of the whole economy.

When user transfer a information between client server type environment there exist different kinds of attacks on information and network. Four general categories of attacks on networks are:

1. Interruption.
2. Interception.
3. Modification.
4. Fabrication.

In presence of these attacks the information is not at all is safe thus system is needed to provide a secure communication and which will be able to handle the attacks in client server communication and provide data confidentiality, integrity and authenticity.

1.4 Scope of Project

1.4.1 Methods for Client/Server security

1. Use logging for each user.
2. Care with passwords.
3. Use handshaking.
4. Encrypt sensitive data
5. Use file signing.

1.4.2 Security Mechanisms

1. To provide data integrity, a message digest algorithm is required. A digest is calculated over an appropriate portion of a message and included as part of the message sent to the recipient.
2. To ensure authentication, each user requires logging facility and each user should have unique user name and password, which can authenticate the client on server.
3. For data confidentiality, a symmetric encryption algorithm is required. An appropriate portion of the message is encrypted prior to being transmitted to its recipient.

1.5 Organization of Dissertation

This Dissertation report is organised in a manner so as to follow the waterfall model of software development life cycle. One can easily understand the problems in existing system, theoretical and practical concepts, technology and its solutions. Chapter one is the introduction, with the background of project, need of the system, scope and the main objectives. Chapter two is the literature survey, containing the theoretical concepts that is basically needed to start the project. It covers client server communication and security threats. Chapter three deals with the cryptography and security solution available, which can be used for developing the software proposed.

Chapter four is the analysis of problem, which covers the available methods and algorithms, and also covers the advantages/disadvantages of client server model. Chapter five is problem design in which the techniques proposed have been specified. In chapter six, the implementation aspects are discussed, in which some important codes that are necessary to understand implementation are explained. Chapter seven describes the interfaces of the software. Finally, chapter eight concludes the work, discussing some enhancements that may be incorporated in the software in future.

LITERATURE SURVEY

2.1 Client Server communication: Definition

The client/server model is based on the concept that each application consists of two parts: one that initiates communication or requests for information (**the client**) and another that responds or services the client with the requested information (**the server**).

2.1.1 Server

1. Servers are powerful computers dedicated to managing specific tasks such as: disk drives (file servers), printers (print server), or network traffic (network server i.e. traffic cop)
2. Typically network servers are tied to a specific purpose (i.e., web server, print server, file server, application server)
3. Generally large capacity for memory and speed (workhorse)

2.1.2 Client

1. A PC or workstation that is able to run applications. Clients rely on servers to perform specific tasks, such as retrieving files, using a particular device such as a fax machine, receive and deliver email, processing power, and printing.
2. Primarily displays information (high quality monitor is a consideration)
3. Point of user interface with the network

2.2 Client Server Architecture

2.2.1 Two Major Architecture types

1. Two-tier - each client connects directly to the data server, data is gathered from the server and processed by the client.
2. Three-tier - each client communicates with the application server, and the application server communicates with the data server.

2.2.2 Management Considerations

1. Budget
2. Existing architecture
3. Training
4. Communication channels
5. Future expansion
6. Functionality- currently being used in conjunction with E-Business and web based sites.

2.3 Methods of communications

Client server communication can be basically defined in terms of request and reply, which is shown in figure 2.1. Client will send a message request type to server will accept the request and reply to client accordingly. Some of methods of communication are following.

1. Message passing - send and receive primitives
2. Synchronous or asynchronous
3. Blocking or non-blocking
4. Mechanisms of message passing - channels, sockets, ports
5. Client-server communication model
6. Group multicast communication model

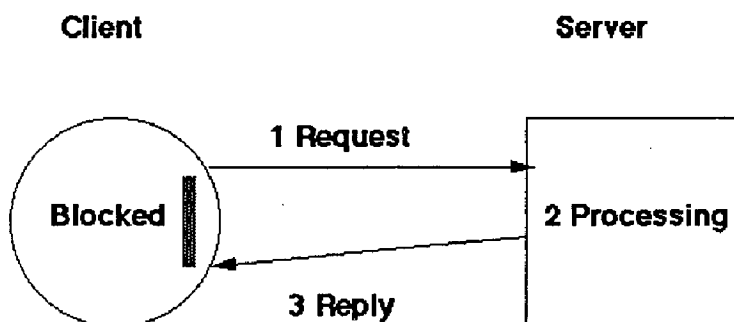


Figure 2.1: Client-server communication model

2.4 NSFDC Client/Server Architectures [1]

At NSFDC, for the physical transfer of information (like movement of papers) considerable amount of manpower time is wasted. Also there may not be direct interaction between any two officers.

Hence it is proposed that the network like local area network spread all over the Apex Corporation need to be designed and deployed, which will connect all the computers and gets connected to a driver server for information transfer between concerned officers and departments.

It is proposed that for this kind of network, Client/Server Technology be adopted. Such a topology is usually a Star topology. Star topology is preferred for its less number of hops, driver and standard minimum time consideration. Such a star topology is modifiable, extendable and is flexible so that modifications if any in future can directly be incorporated. It does not follow directly token ring but a logical ring which behaves like a ring so that network management could be done from a single location and that location need not to be physically fixed. That is, physical address of each client is transparent to the users. They are taken care by NMSP (Network Management Software Package). Such topology is also suitable for making the network as Intranet so that such Intranet can become a part and parcel of a bigger network. The topology is also suitable for physical formulations of object-oriented technology.

2.4.1 Communication links between various Departments

The various departments of NSFDC will be inter-linked through LAN connections. Common data will be entered only once and will be available through all required departments. Each department will have certain users defined who can access the data relevant to them. Each user will have a password so that he or she can be uniquely identified by the system and the system will make visible only those data for which the user is authorised. This will maintain the integrity of data as well as remove duplication of data due to separate entries at individual departments.

2.5 Security Attacks

2.5.1 Definition [2]

Any action that compromises the security of information owned by an organization. Broadly these actions can be of following forms.

1. **Interception of communications:** Electronic communication can be intercepted and data copied or modified. Interception can be undertaken in a number of ways. These include the physical accessing of network lines, e.g. wire tapping, and monitoring radio transmissions. The most critical points for the interception of communication traffic are the network management and concentration points, such as routers, gateways, switches and network operation servers.
2. **Unauthorized access into computers and computer networks:** Unauthorised access to a computer or network of computers is usually done with malicious intent to copy, modify or destroy data. Technically this is called intrusion and can be done in many ways including exploiting inside information, dictionary attacks, brute force attacks (exploiting people's tendency to use predictable passwords), social engineering (exploiting people's tendency to disclose information to seemingly trustworthy people) and password interception. It is often performed from within the organisations (inside attacks).
3. **Potential solutions:** The most common methods of protecting against unauthorised access are password controls and installation of firewalls. Network disruption: Networks are now largely digitized and controlled by computers. In the past a common reason for network disruption was a failure in the computer system that controls the network and attacks on networks were mainly directed towards these computers. Nowadays, the most disrupting attacks tend to exploit the weaknesses and vulnerabilities of network components (operating systems, routers, switches, name servers, etc.).

4. **Execution of malicious software that modifies or destroys data:** Computers run with software. Software can unfortunately also be used to disable a computer, to delete or modify data. As the above descriptions show, if such a computer is part of the network management its malfunctioning can have far-reaching effects. A virus is one form of malicious software. It is a program that reproduces its own code by attaching itself to other programs in such a way that the virus code is executed when the infected computer program is executed.

2.6 Classification of Attacks

There exist four general categories attacks on networks [3]:

1. **Interruption:** An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability of services. This is a Physical attacks such as destruction of hardware; cutting of communication line. Denial of service comes under it.
2. **Interception:** An unauthorized party gains access to an asset. This is an attack on confidentiality. Confidentiality means Message content is being kept secret from illegitimate listeners.
3. **Modification:** An unauthorized party not only gains access to but tampers with an asset. This is an attack on integrity. Proof that messages are sent and received untampered with without duplication, insertion, modification, reordering, or replays.
4. **Fabrication:** An unauthorized party inserts counterfeit objects into the system. This is an attack on authenticity.

Figure 2.2 represent above categories diagrammatically.

These general forms of attacks may be categorized in passive and active attacks. Passive attacks deals with confidentiality and only consists interception. Active attacks can (usually) be detected but hard to prevent. Active attacks are broader and may deal with interruption (availability), modification (integrity) and fabrication (also integrity).

These active attacks may be divided into the following categories:

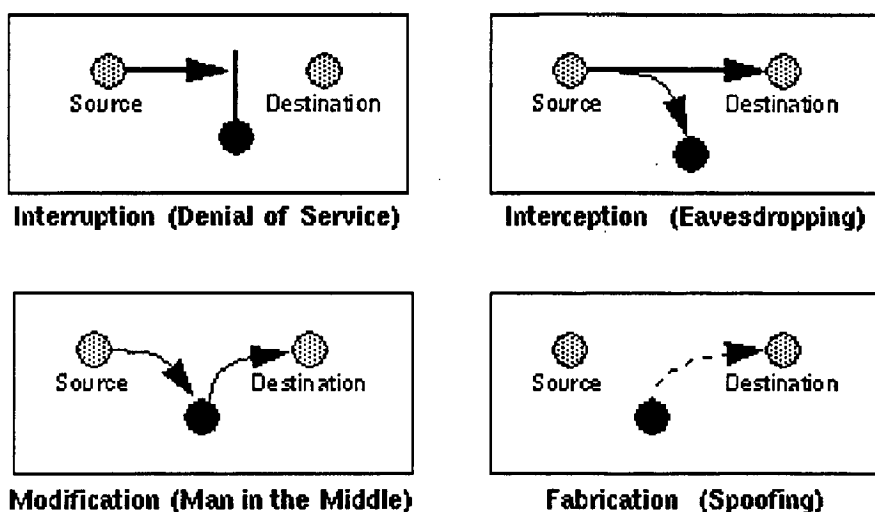


Figure 2.2: Information in enterprise networks can be compromised through interruption, interception, modification, and fabrication.

1. **Masquerade**: This means that one pretends to be someone else. To do that, in the protocol, one tries to fake the sender's signature. If the system is secure, one needs the sender's secret PGP key to do this successfully.
2. **Replay**: With replay, one captures data and makes a retransmission to produce an unauthorized effect.
3. **Modification of messages**: Modification of messages, mean alter of a message, or that messages are delayed or reordered. First some one tried to alter a delegation certificate so that it became the recipient. Unlike masquerade intruder did not try to be someone else, but he tried to get an access that was not intended for him.
4. **Denial of service**: Denial of service, mean prevent the normal use or management of communications facilities. This could be preventing any communication at all, or just making sure that a message (in the protocol) is not delivered to the recipient.

INTRODUCTION TO CRYPTOGRAPHY

3.1 Definition

Cryptography is a collection of techniques for keeping information secure. Using cryptography, you can transform written words and other kinds of messages so that they are unintelligible to unauthorized recipients. An authorized recipient can then transform the words or messages back into a message that is perfectly understandable.

For example, here is a message that you might want to encrypt:

“It is nice to develop cryptographic protocol”

And here is the message after it has been encrypted:

Ç'^@%[ÈFÇ<<\$TÞPÂ|xÀEÛóõÑ0/00B+ö~ÖaÛýB-->uâw

Even better, with cryptography you can transform this gibberish back into the original easily understood message.

Modern cryptographic systems consist of two complementary processes:

3.1.1 Encryption

A process by which a message (the plain text) is transformed into a second message (the cipher text) using a complex function (the encryption algorithm) and a special encryption key .

3.1.2 Decryption

The reverse process, in which the cipher text is transformed back into the original plaintext using a second complex function and a decryption key . With some encryption systems, the encryption key and the decryption key are the same. With others, they are different.

Cryptography is the science of writing in secret code. Within the context of any application-to-application communication, there are some specific security requirements, including:

1. **Authentication:** The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)
2. **Privacy/confidentiality:** Ensuring that no one can read the message except the intended receiver.
3. **Integrity:** Assuring the receiver that the received message has not been altered in any way from the original.
4. **Non-repudiation:** A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below.

In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into ciphertext, which will in turn (usually) be decrypted into usable plaintext.

In many of the descriptions below, two communicating parties will be referred to as Alice and Bob; this is the common nomenclature in the crypto field and literature. If there is a third or fourth party to the communication, they will be referred to as Carol and Dave. Mallory is a malicious party and Eve is an eavesdropper.

3.2 Types Of Cryptographic Algorithms

There are several ways of classifying cryptographic algorithms. For purposes of this paper, they will be categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms that will be discussed are

1. **Secret Key Cryptography:** Uses a single key for both encryption and decryption
2. **Public Key Cryptography:** Uses one key for encryption and another for decryption

3. Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information.

3.3 Secret Key Cryptography

In secret key cryptography, a single key is used for both encryption and decryption. The receiver applies the same key (or rule set) to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called symmetric encryption. With this form of cryptography, it is obvious that both the sender and the receiver must know the key; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key.

There are several widely used secret key cryptography schemes and they are generally categorized as being either stream ciphers or block ciphers. Stream ciphers operate on a single bit, byte, or (computer) word at a time, and implement some form of feedback mechanism so that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will always encrypt to different ciphertext in a stream cipher.

Stream ciphers come in several flavors but two are worth mentioning here. Self-synchronizing stream ciphers calculate each bit in the key stream as a function of the previous n bits in the key stream. It is termed "self-synchronizing" because the decryption process can stay synchronized with the encryption process merely by knowing how far into the n -bit key stream it is.

One problem is error propagation; a garbled bit in transmission will result in n garbled bits at the receiving side. Synchronous stream ciphers generate the key stream in a fashion independent of the message stream but by using the same key stream generation function at sender and receiver. While stream ciphers do not propagate transmission errors, they are, by their nature, periodic so that the key stream will eventually repeat.

There are a number of other secret-key cryptography algorithms that are currently in use.

1. CAST-128: CAST is not an acronym but its name is derived from the initials of its inventors, Carlisle Adams and Stafford Tavares of Nortel), a DES-like substitution-permutation crypto algorithm, employing a 128-bit key operating on a 64-bit block. CAST is internationally available.
2. International Data Encryption Algorithm (IDEA): another DES-like 64-bit block cipher using 128-bit keys. Also internationally available.
3. Rivest Cipher 2 (RC2): named for its inventor Ron Rivest (thus, "RC" is also sometimes expanded as "Ron's Code"), a 64-bit block cipher using variable-sized keys designed to replace DES. Its code has not been made public although many companies have licensed RC2 for use in their products.
4. RC4: a stream cipher using variable-sized keys; it is widely used in commercial cryptography products, although it can only be exported using keys that are 40 bits or less in length.
5. RC5: a block-cipher supporting a variety of block sizes, key sizes, and number of encryption passes over the data.
6. Blowfish: a symmetric 64-bit block cipher invented by Bruce Schneier; optimized for 32-bit processors with large data caches, it is significantly faster than DES on a Pentium/PowerPC-class machine. Key lengths can vary from 32 to 448 bits in length.
7. Twofish: a 128-bit block cipher using 128-, 192-, or 256-bit keys. Designed to be highly secure and highly flexible, well suited for large microprocessors, 8-bit smart card microprocessors, and dedicated hardware.

3.4 Public-Key Cryptography

Public-key cryptography (PKC) has been said to be the most significant new development in cryptography in the last 300-400 years. Modern PKC was first described publicly by Stanford University professor Martin Hellman and graduate student Whitfield Diffie in 1976. Their paper described a two-key crypto system in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key.

The first, and still most common, PKC implementation is RSA, named for the three MIT mathematicians who developed it — Ronald Rivest, Adi Shamir, and Leonard Adleman. RSA today is used in hundreds of software products and can be used for key exchange, digital signatures, or encryption of small blocks of data. RSA uses a variable size encryption block and a variable size key.

The key-pair is derived from a very large number, n , that is the product of two prime numbers chosen according to special rules; these primes may be 100 or more digits in length each, yielding an n with roughly twice as many digits as the prime factors. The public key information includes n and a derivative of one of the factors of n ; an attacker cannot determine the prime factors of n (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secure. (Some descriptions of PKC erroneously state that RSA's safety is due to the difficulty in factoring large prime numbers.

3.5 Hash Functions

Hash functions, also called message digests and one-way encryption, are algorithms that, in some sense, use no key. Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. Furthermore, there is an almost zero probability that two different plaintext messages will yield the same hash value.

Hash algorithms are typically used to provide a digital fingerprint of a file's contents, often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords.

Among the most common hash functions in use today in commercial cryptographic applications are a family of Message Digest (MD) algorithms, all of which are byte-oriented schemes that produce a 128-bit hash value from an arbitrary-length message. MD2 is well suited for systems with limited memory, such as smart cards. MD4 developed by Rivest, is similar to MD2 but designed specifically for fast processing in software.

MD5, also developed by Rivest, came about after potential weaknesses were reported in MD4; this scheme is similar to MD4 but is slower because more manipulation is made to the original data. The Secure Hash Algorithm (SHA), proposed by NIST for their Secure Hash Standard (SHS), is seeing increased use in commercial products today. SHA-1 produces a 160-bit hash value.

ANALYSIS OF THE PROBLEM

Security is about controlling access to a variety of resources, such as application components, data, and hardware. There are two concepts upon which most security measures are based:

1. Authentication
2. Data Protection

4.1 Authentication

Authentication is the process of identity confirmation, which is the one layer of security control. Before an application can authorize access to a resource, it must confirm the identity of the requestor. The requestor establishes an identity by providing some form of credentials, which is known only to the requestor and the authenticating host. In some circumstances, the requestor may want to verify the identity of the authenticating host, which is called mutual authentication.

4.1.1 Types of authentication

1. What you know (username and password)
2. What you have (token, smart card)
3. What you are (biometrics)
4. Where you are (location security)

4.2 Data Protection

Data protection is the process of providing data confidentiality, integrity, and non-reputability. Data requires protection not only while in transit but also while it is stored. Regardless of what the data's form, once data enters unsecured communication channels it becomes vulnerable to attack.

Encrypting the data provides data confidentiality. Data encryption uses a crypto algorithm in conjunction with a crypto key to render data useless to someone lacking both the correct algorithm and key to decrypt the data.

4.2.1 Choosing a Cipher [3]

Selecting between the competing ciphers is not easy. Apart from slightly different currently known security levels, questions of speed, code size, and any patent or licencing issues need to be considered. As a general rule, do rely on ciphers that are known, have been publicly analyzed, and in some use. Different applications are likely to require different tradeoffs and hence choices. Table 4.1 explains the private, public algorithms available and small description about algorithms.

Algorithm	Description
Private Key Algorithms:	
ROT13	Keyless text scrambler; very weak.
Crypt	Variable key length stream cipher; very weak.[12]
DES	56-bit block cipher; patented, but freely usable (but not exportable).
RC2	Variable key length block cipher; proprietary.
RC4	Variable key length stream cipher; proprietary.
RC5	Variable key length block cipher; proprietary.
IDEA	128-bit block cipher; patented.
Skipjack	80-bit stream cipher; classified.
Public Key Algorithms:	
Diffie-Hellman	Key exchange protocol; patented.
RSA	Public key encryption and digital signatures; patented
ElGamal	Public key encryption and digital signatures; patented.
DSA	Digital signatures only; patented.
Table 4.1: Commonly Used Private and Public Key Cryptography Algorithms	

4.2.2 Comparative Speeds [4]

These algorithms have been implemented in c++ (using the Cryptix library). In one run, interpreted on a Pentium2/266 system, the following comparative times were obtained shown in table 4.2.

Algorithm	Time (ms)	Rate (Kbps)	key 1000 pairs (ms)
Blowfish	20506	409	79290
CAST5	23772	352	976
DES	48629	172	519
TripleDES	160807	52	1790
IDEA	43409	193	734
LOKI91	31071	269	76
RC2	43329	193	790
RC4 (*)	12945	648	2382
SAFER	41442	202	2219
Square	29610	283	2166

(*) RC4 is stream cipher.

Table 4.2: Table of Comparative Block Ciphers Timings.

The crypto key is an additional variable used in the algorithm. A crypto key contains a numeric value that is limited by the number of bits the key contains. Although a 40-bit key contains 240, or 1,099,511,627,776, possible key values, a typical PC could try an exhaustive search of every possible key value in approximately one week. However, if the crypto key consists of 128 bits, a brute force attack would need to try up to 2128, or 3.4×10^{38} , values. Each additional bit doubles the possible number of values. Crypto keys enable a public algorithm to be used by multiple parties without compromising data encrypted with the algorithm.

Since the crypto key determines the strength of the encryption, all encryption algorithms are vulnerable to brute force attacks. A brute force attack is the systematic attempt to decrypt data using every possible key.

If the crypto key used to encrypt data consisted of only four bits, a brute force attack would only need to try up to sixteen crypto key values to compromise data.

4.3 Data integrity

Data integrity is achieved through the use of hash algorithms, digital signatures, and message authentication codes. To ensure the integrity of data, a hash of that data can be sent to accompany it. The receiver can then compare a hash that it computes on the received data with the hash that accompanied the received data. If the two match, the received data must be the same as the data from which the received hash was created. A hash is a fixed-length string of numbers and characters. It is computed using a hashing algorithm, such as Message Digest 5 (MD5) or Secure Hash Algorithm (SHA-1). Hashing is a one-way operation that cannot be reversed to recreate the original data.

A digital signature takes hashing a step further by encrypting the computed hash using a private key. This extra step can prevent an attacker from intercepting data and its accompanying hash, modifying the data, and then simply re-computing the new hash for the modified data. Since a digital signature is an encrypted hash, an attacker would need access to the original private key that was used to create the original digital signature. On the receiving end, digital signatures can be verified using the associated public key. Digital signatures can be used to enforce non-repudiation, which can later be used to prove the origin, contents, and timestamp of the data. Message authentication codes (MACs) are used by technologies such as SSL/TLS to verify that data has not been altered while in transit. However, since MACs use a common key for encryption and verification, they cannot be used to enforce non-repudiation.

4.3.1 Comparison of some hash Algorithms

Message digest functions distill the information contained in a file (small or large) into a single large number, typically between 128 and 256 bits in length. Every bit of the message digest function is influenced by every bit of the function's input.

If any given bit of the function's input is changed, every output bit has a 50 percent chance of changing. Given an input file and its corresponding message digest, it should be computationally infeasible to find another file with the same message digest value. Message digests are also called one-way hash functions because they produce values that are difficult to invert, resistant to attack, mostly unique, and widely distributed. Many message digest functions have been proposed and are currently in use. Some of them are following:

1. **HMAC**: The Hashed Message Authentication Code, a technique that uses a secret key and a message digest function to create a secret message authentication code. The HMAC method strengthens an existing message digest function to make it resistant to external attack, even if the message digest function itself is somehow compromised.
2. **MD2**: Message Digest #2, developed by Ronald Rivest. This message digest is the most secure of Rivest's message digest functions, but takes the longest to compute. It produces a 128-bit digest.
3. **MD4**: Message Digest #4, also developed by Ronald Rivest. This message digest algorithm was developed as a fast alternative to MD2. Subsequently, MD4 has been shown to be insecure. That is, it is possible to find two files that produce the same MD4 codes without requiring a brute force search. MD4 produces a 128-bit digest.
4. **MD5**: Message Digest #5, also developed by Ronald Rivest. MD5 is a modification of MD4 that includes techniques designed to make it more secure. Although widely used, in the summer of 1996 a few flaws were discovered in MD5 that allowed some kinds of collisions to be calculated. As a result, MD5 is slowly falling out of favor. MD5 produces a 128-bit digest.
5. **SHA**: The Secure Hash Algorithm, developed by the NSA and designed for use with the National Institute for Standards and Technology's Digital Signature Standard (NIST's DSS). Shortly after the publication of the SHA, NIST announced that it was not suitable for use without a small change. SHA produces a 160-bit digest.

6. **SHA-1:** The revised Secure Hash Algorithm, also developed by the NSA and designed for use with the NSA's DSS. SHA-1 incorporates minor changes from SHA. It is not known if these changes make SHA-1 more secure than SHA, although some people believe that it does. SHA-1 produces a 160-bit digest.

The following Table 4.3 gives an idea of the performance of the different MD4-like hash functions. The implementations are written in 80x86 assembly language and are optimized for the Pentium processor. It is assumed that both code and data resides in the on-chip caches. Under these conditions the cycle figures are independent of the clock speed, and the throughput figures scale with the clock speed.

Algorithm	Cycles	Mbit/sec	Mbyte/sec	Relative performance
MD4	241	191.2	23.90	1.00
MD5	337	136.7	17.09	0.72
RIPMD	480	96.0	12.00	0.50
RIPMD-128	592	77.8	9.73	0.41
SHA-1	837	55.1	6.88	0.29
RIPMD-160	1013	45.5	5.68	0.24

Table 4.3: Performance of optimized assembly language implementations of MD4 like hash functions on a 90 MHz Pentium using a 32-bit flat memory model.

4.4 Advantages/Disadvantages of client server architecture

4.4.1 Advantages

1. Initial investment: The initial cost of a client / server setup can be significantly cheaper than the initial outlay to buy a mainframe computer. [Software for a client server is typically cheaper than for a mainframe.]
2. System flexibility: Client server allows for easy expansion of a system. This can be accomplished by gradually adding on to the network. This may include adding more servers, clients, or even network printers.

3. Remote access: Allows remote access to the system so employees can access the system from virtually anywhere in the world. Provides a means for distributed computing.
4. Expandability: If you need to add more users to the network, simply obtain the components necessary for a network connection and connect them. If you need more processing power, or the system needs changes, solutions can be as simple as adding more clients or servers to the network.
5. Reliability: There are more pieces to the system, so if one piece fails, it is possible for one of the remaining components to “cover” until the “down” piece is fixed.

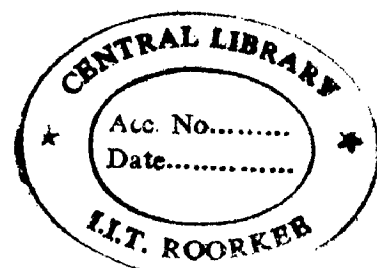
4.4.2 Disadvantages

1. High maintenance: Security, System upgrades, User profiles, Hardware upgrades, Compatibility issues, Communication channels between each network, Employing technical people.
2. Security: many more security issues exist due to the fact there are more access points to secured data and processes.
3. Reliability: There are more pieces to the system, so there is a lot more energy put into backup plans. The complexity of those plans increases considerably as well as taking ownership of the system.

4.5 Methods for General Data Transfer in Networks [5]

4.5.1 File Transfer Protocol (FTP)

A widely used protocol for transferring files on the network, is File Transfer Protocol (FTP). An FTP client program, normally operated by a human user, connects to a server using TCP. The user may send, receive and delete files, create and remove directories, and perform other file operations across the network. Unrestricted use of FTP normally requires the user to have an account on the server host. The FTP session is then initiated by the user providing a user name and a password.



A popular way of distributing publicly available files on the Internet, is using anonymous FTP services, where the user may log in to a public area without having an account on the server host. Users logging in anonymously, are normally restricted to doing downloads only.

To operate any file transfer by these methods, user need to run a client program on the system which initiates the connection, and the system at the other end (the host) needs to be running a server program which can accept incoming connections and process requests from client. Once connected, you can cause files to be transferred in either direction (host to client or client to host). It is possible to run the same servers on same machine but this is not necessary for most users, who only need to run client programs, and there are major security issues to be taken into account if user do want to run the own server.

4.5.2 Hypertext Transfer Protocol (HTTP)

Even though the WWW is designed to envelope existing protocols, a new protocol was defined for it. The Hypertext Transfer Protocol (HTTP) allows the Web to surmount the problems of different data types using negotiation of data representation. In contrast to FTP, which operates directly on the server file system using file- and directory names, HTTP identifies documents using Uniform Resource Locators (URL).

HTTP is a "one-shot" protocol: The client opens a TCP-connection to the server, normally on port 80, and sends it's request. The server in turn sends it's response, and closes the connection. Several requests to the same server, requires establishing new connections. The repeated reconnectioning that frequently occurs when fetching Web pages, puts an unnecessary load on both the client and the server host, along with the network itself. New versions of HTTP will probably allow a connection to be kept open as long as needed. The data type negotiation is done using MIME-like headers in both the request and the response.

Although mainly being used for transferring data from the server to the client by request, the HTTP standard also defines methods for sending data to the server, used for instance in fill-out forms embedded in HTML-documents

4.5.3 Trivial File Transfer Protocol

TFTP is a simple protocol to transfer files, and therefore was named the Trivial File Transfer Protocol or TFTP.

It has been implemented on top of the Internet User Datagram protocol (UDP or Datagram) so it may be used to move files between machines on different networks implementing UDP. (This should not exclude the possibility of implementing TFTP on top of other datagram protocols.) It is designed to be small and easy to implement. Therefore, it lacks most of the features of a regular FTP. The only thing it can do is read and write files (or mail) from/to a remote server. It cannot list directories, and currently has no provisions for user authentication. In common with other Internet protocols, it passes 8 bit bytes of data.

PROBLEM DESIGN

In order to design system, various available methods are studied which can handle the security threat on client server network, the system has been design and implemented for following features.

1. **Data integrity:** A digest is calculated over an appropriate portion of a message and included as part of the message or excluded sent to the recipient.
2. **Authentication:** Each user require logging facility and each user should have separate user name and password, which can authenticate the client on server.
3. **Data confidentiality:** An appropriate portion of the message is encrypted prior to being transmitted to its recipient.

5.1 Encryption

In using data encryption, a plain-text message can be encoded so it appears as completely random binary data that is very difficult (if not impossible) to transform back to the original message without a secret key. In this article, the following definitions apply:

1. **Message** is used to refer to any piece of data. A message can consist of ASCII text, a database file, or any data you want to store or transmit securely.
2. **Plain text** is used to refer to data that has not been encrypted.
3. **Cipher text** refers to data that has been encrypted.

Once a message has been encrypted, it can be stored on nonsecure media or transmitted on a nonsecure network and still remain secret. Later, the message can be decrypted into its original form. This process is shown in Figure 5.1.

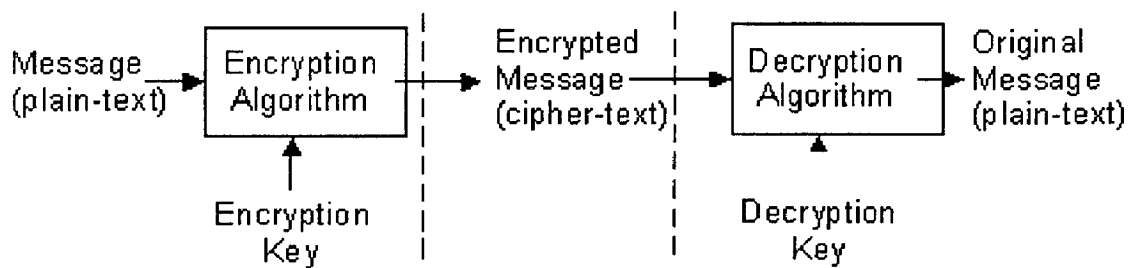


Figure 5.1: Encrypting and decrypting a message

When a message is encrypted, an encryption key is used. This is analogous to the physical key that is used to lock a padlock. To decrypt the message, the corresponding decryption key must be used. It is very important to properly restrict access to the decryption key, because anyone who possesses it will be able to decrypt all messages that were encrypted with the matching encryption key.

5.2 Algorithms

Symmetric algorithms are the most common type of encryption algorithm. They are known as "symmetric" because the same key is used for both encryption and decryption. Unlike the keys used with public-key algorithms, symmetric keys are frequently changed. For this reason, they are referred to here as session keys. Compared to public-key algorithms, symmetric algorithms are very fast and thus are preferred when encrypting large amounts of data. Some of the more common symmetric algorithms are RC2, RC4 that are best suited to provide data confidentiality and both algorithms are widely accepted. Details of algorithms are available in Appendix A

5.2.1 Rivest Cipher 2 (RC2)

Rivest Cipher 2 (RC2), named for its inventor Ron Rivest (thus, "RC" is also sometimes expanded as "Ron's Code"), a 64-bit block cipher using variable-sized keys designed to replace DES. Its code has not been made public although many companies have licensed RC2 for use in their products.

There are three separate algorithms involved:

1. **Key expansion.** This takes a (variable-length) input key and produces an expanded key consisting of 64 words $K[0], \dots, K[63]$. The purpose of the key-expansion algorithm is to modify the key buffer so that each bit of the expanded key depends in a complicated way on every bit of the supplied input key.
2. **Encryption.** This takes a 64-bit input quantity stored in words $R[0], \dots, R[3]$ and encrypts it "in place" (the result is left in $R[0], \dots, R[3]$). The encryption operation is defined in terms of primitive "mix" and "mash" operations. The entire encryption operation can now be described as follows. Here j is a global integer variable, which is affected by the mixing operations.
 1. Initialize words $R[0], \dots, R[3]$ to contain the 64-bit input value.
 2. Expand the key, so that words $K[0], \dots, K[63]$ become defined.
 3. Initialize j to zero.
 4. Perform five mixing rounds.
 5. Perform one mashing round.
 6. Perform six mixing rounds.
 7. Perform one mashing round.
 8. Perform five mixing rounds.
3. **Decryption.** The inverse operation to encryption. The decryption operation is defined in terms of primitive operations that undo the "mix" and "mash" operations of the encryption algorithm. They are named "r-mix" and "r-mash" (r -denotes the reverse operation).

5.2.2 Rivest Cipher 4 (RC4)

RC4 uses a variable length key from 1 to 256 bytes to initialize a 256-byte state table. The state table is used for subsequent generation of pseudo-random bytes and then to generate a pseudo-random stream, which is XORed with the plaintext to give the ciphertext. Each element in the state table is swapped at least once. The RC4 algorithm works in two phases, key setup and ciphering. Key setup is the first and most difficult phase of this algorithm.

During a N-bit key setup (N being key length), the encryption key is used to generate an encrypting variable using two arrays, state and key, and N-number of mixing operations.

These mixing operations consist of swapping bytes, modulo operations, and other formulas. A modulo operation is the process of yielding a remainder from division. For example, 11/4 is 2 remainder 3; therefore eleven mod four would be equal to three.

Once the encrypting variable is produced from the key setup, it enters the ciphering phase, where it is XORed with the plain text message to create an encrypted message. XOR is the logical operation of comparing two binary bits. If the bits are different, the result is 1. If the bits are the same, the result is 0. Once the receiver gets the encrypted message, he decrypts it by XORing the encrypted message with the same encrypting variable.

5.3 Key Databases

The Cryptography API contains functions that allow applications to encrypt or digitally sign data in a flexible manner, while providing protection for the user's sensitive private key data. All cryptographic operations performed by independent modules known as cryptographic service providers (CSPs).

Each CSP has a key database in which it stores its persistent cryptographic keys. Each key database contains one or more key containers, each of which contains all the key pairs belonging to a specific user (or Cryptography API client). Each key container is given a unique name, which applications provide to the CryptAcquireContext function when acquiring a handle to the key container. Figure 5.2 is an illustration of the contents of a key database:

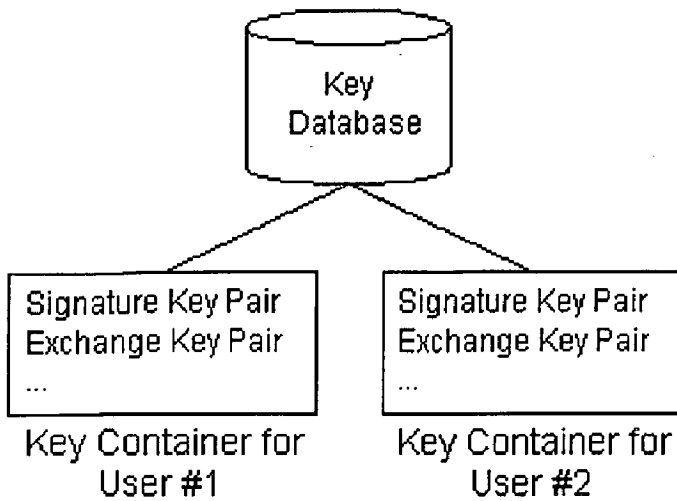


Figure 5.2: Contents of a key database

The CSP stores each key container from session to session, including all the public/private key pairs it contains. However, session keys are not preserved from session to session. Generally, a default key container is created for each user. This key container takes the user's logon name as its own name, which is then used by any number of applications. It is also possible for an application to create its own key container (and key pairs), which it usually names after itself.

5.3.1 Keys

Session Keys: Session keys are used when encrypting and decrypting data. They are created by applications using either the `CryptGenKey` or the `CryptDeriveKey` function. These keys are kept inside the CSP for safekeeping.

Unlike the key pairs, session keys are volatile. Applications can save these keys for later use or transmission to other users by exporting them from the CSP into application space in the form of an encrypted key binary large object or key blob using the `CryptExportKey` function.

Public or Private Key Pairs: Each user generally has two public or private key pairs. One key pair is used to encrypt session keys and the other to create digital signatures. These are known as the key exchange key pair and the signature key pair, respectively.

5.3.2 Key distribution

Diffie-Hellman allows two parties — the ubiquitous Alice and Bob — to generate a secret key; they need to exchange some information over an insecure communications channel to perform the calculation but an eavesdropper cannot determine the shared key based upon this information.

Diffie-Hellman works like this. Alice and Bob start by agreeing on a large prime number, n . They also have to choose some number g so that $g < n$. There is actually another constraint on g , specifically that it must be primitive with respect to n . Primitive is a definition that is a little beyond the scope of discussion but basically g is primitive to n if user can find integers i so that $g^i = j \pmod n$ for all values of j from 1 to $n-1$. As an example, 2 is not primitive to 7 because the set of powers of 2 from 1 to 6, mod 7 = {2,4,1,2,4,1}. On the other hand, 3 is primitive to 7 because the set of powers of 3 from 1 to 6, mod 7 = {3,2,6,4,5,1}.

Alice or Bob selects n and g ; they then tell the other party what the values are. Alice and Bob then work independently:

Alice...Choose a large random number, x

Send to Bob: $X = g^x \pmod n$

Compute: $K_A = Y^x \pmod n$

Bob...Choose a large random number, y

Send to Alice: $Y = g^y \pmod n$

Compute: $K_B = X^y \pmod n$

x and y are kept secret while X and Y are openly shared; these are the private and public keys, respectively. Based on their own private key and the public key learned from the other party, Alice and Bob have computed their secret keys, K_A and K_B , respectively, which are equal to $g^{xy} \pmod n$. Diffie-Hellman can also be used to allow key sharing amongst multiple users. Note again that the Diffie-Hellman algorithm is used to generate secret keys, not to encrypt and decrypt messages.

5.4 File hashing

To ensure the integrity of data, as shown in Figure 5.3 a hash of that data can be sent to accompany it. The receiver can then compare a hash that it computes on the received data with the hash that accompanied the received data.

If the two match, the received data must be the same as the data from which the received hash was created. A hash is a fixed-length string of numbers and characters. It is computed using a hashing algorithm, such as Message Digest 5 (MD5) or Secure Hash Algorithm (SHA-1). Hashing is a one-way operation that cannot be reversed to recreate the original data.

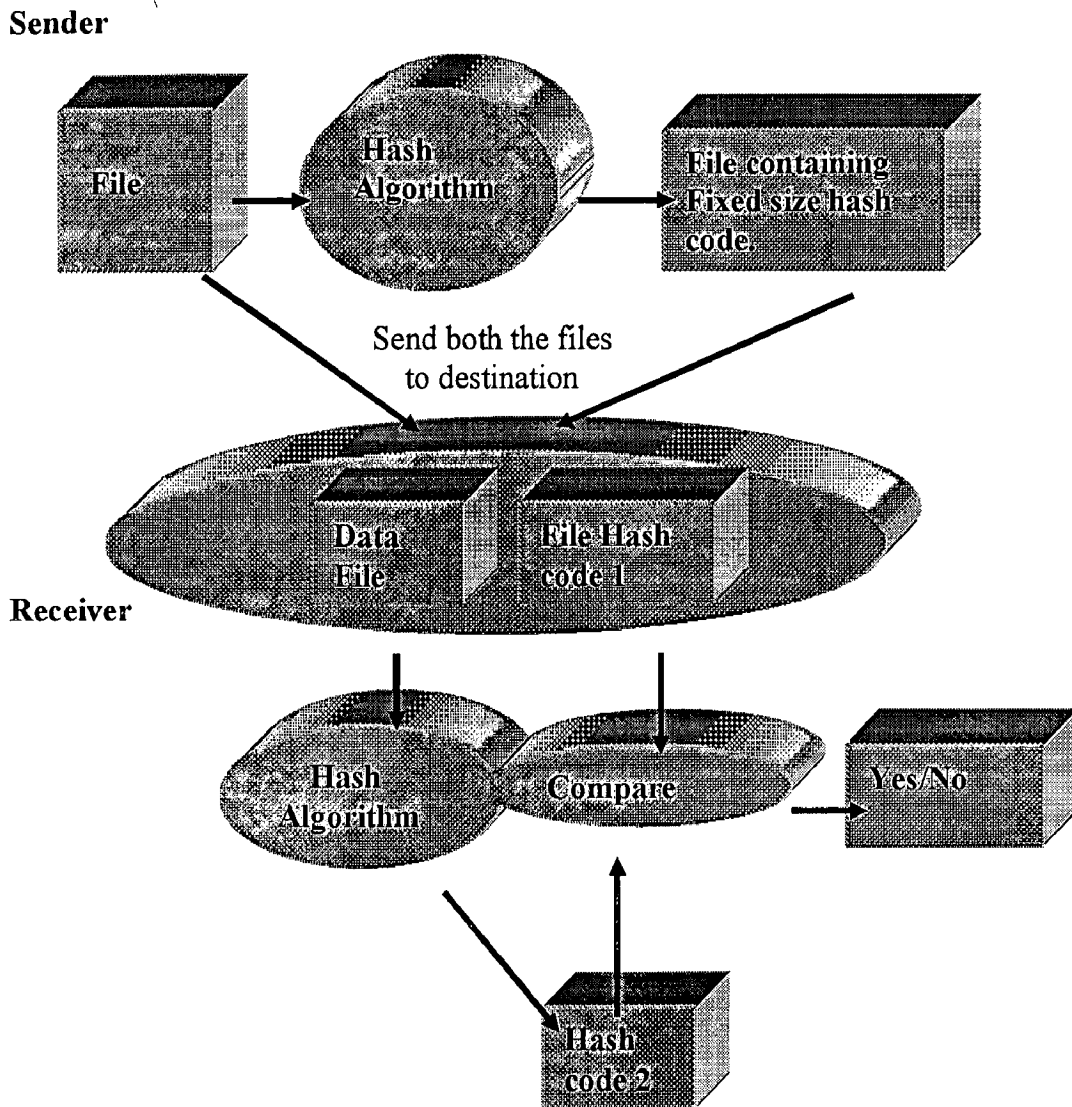


Figure 5.3: Figure is showing the process to check data integrity of data.

5.4.1 Message Digest#5 (MD5)

User have a b-bit message as input, and user wish to find its message digest. The following five steps are performed to compute the message digest of the message.

Step 1. Append Padding Bits

The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512.

Step 2. Append Length

A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^{64} , then only the low-order 64 bits of b are used.

At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits.

Step 3. Initialize MD Buffer

A four-word buffer (A, B, C, D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

Word A: 01 23 45 67

Word B: 89 ab cd ef

Word C: fe dc ba 98

Word D: 76 54 32 10

Step 4. Process Message in 16-Word Blocks

Algorithm first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

$$F(X,Y,Z) = XY \vee \text{not}(X) Z$$

$$G(X,Y,Z) = XZ \vee Y \text{not}(Z)$$

$$H(X,Y,Z) = X \text{ xor } Y \text{ xor } Z$$

$$I(X,Y,Z) = Y \text{ xor } (X \vee \text{not}(Z))$$

In each bit position F acts as a conditional: if X then Y else Z. The function F could have been defined using + instead of \vee since XY and not (X)Z will never have 1's in the same bit position. The functions G, H, and I are similar to the function F.

Step 5. Output

The message digest produced as output is A, B, C, D. That begins with the low-order byte of A, and end with the high-order byte of D.

5.4.2 Secure Hash Algorithm (SHA)

SHA-1 and SHA-256, one begins by converting the message to a unique representation of the message that is a multiple of 512 bits in length, without loss of information about its exact original length in bits, as follows: append a 1 to the message. Then add as many zeroes as necessary to reach the target length, which is the next possible length that is 64 bits less than a whole multiple of 512 bits. Finally, as a 64-bit binary number, append the original length of the message in bits.

SHA-1 Expands each block of 512, when it becomes time to use it, into a source of 80 32-bit subkeys as follows: the first 16 subkeys are the block itself. All remaining subkeys are generated as follows: subkey N is the exclusive OR of subkeys N-3, N-8, N-14, and N-16, subjected to a circular left shift of one place. (This is the mysterious circular left shift added after the original version of SHA was released.). Starting from the 160-bit block value (in hexadecimal) 67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0 as input for the processing of the first 512-bit block of the modified message, for each message block, do the following:

Encipher the starting value using the 80 subkeys for the current message block. Add each of the 32-bit pieces of the ciphertext result to the starting value, modulo 2^{32} , of course, and use that result as the starting value for handling the next message block. The starting value created at the end of handling the last block is the hash value, which is 160 bits long.

5.5 Authentication

For authenticating the client on server client need a to transfer the user name and password to the server this can be confirm by server on give authentication to client this is a first layer of security in the application then user can select the file from explorer and sign/encrypt that and send this at the client side this can be verify/decrypt using the key, which can be evaluated using some information from client.

5.6 File Transfer

After analyzing different data transfer methods like FTP, HTTP, and TFTP. TFTP is a simple protocol to transfer files, and therefore was named the Trivial File Transfer Protocol or TFTP. It has been implemented on top of the Internet User Datagram protocol (UDP or Datagram) so it may be used to move files between machines on different networks implementing UDP. It is designed to be small and easy to implement. TFTP will be used for file transfer.

Any transfer begins with a request to read or write a file, which also serves to request a connection. If the server grants the request, the connection is opened and the file is sent in fixed length blocks of 512 bytes. Each data packet contains one block of data, and must be acknowledged by an acknowledgment packet before the next packet can be sent. A data packet of less than 512 bytes signals termination of a transfer. If a packet gets lost in the network, the intended recipient will timeout and may retransmit his last packet (which may be data or an acknowledgment), thus causing the sender of the lost packet to retransmit that lost packet. The sender has to keep just one packet on hand for retransmission, since the lock step acknowledgment guarantees that all older packets have been received. Both machines involved in a transfer are considered senders and receivers.

One sends data and receives acknowledgments, the other sends acknowledgments and receives data. Most errors cause termination of the connection. An error is signaled by sending an error packet. This packet is not acknowledged, and not retransmitted (i.e., a TFTP server or user may terminate after sending an error message), so the other end of the connection may not get it. Therefore timeouts are used to detect such a termination when the error packet has been lost.

Errors are caused by three types of events: not being able to satisfy the request (e.g., file not found, access violation, or no such user), receiving a packet which cannot be explained by a delay or duplication in the network (e.g., an incorrectly formed packet), and losing access to a necessary resource (e.g., disk full or access denied during a transfer). TFTP recognizes only one error condition that does not cause termination, the source port of a received packet being incorrect.

IMPLEMENTATION ASPECTS

6.1 Encrypting and Decrypting Data

To provide the data confidentiality user need to encrypt the data before sending it to the communication channel. Encryption algorithms can be use to provide the system security means to protect the data on own system to avoid the illegal access if computer is connected to the network. Once the data is encrypted no one to get it without knowing the key, to retrieve the data again user need to decrypt it using same key. Following steps are explaining encryption /decryption for system and for networks.

Encryption /decryption for system: User have to select the algorithm RC2 or RC4 from the cipher choice and then put the password which can be use as key to encrypt the file the same password will be needed to decrypt the file. user have to remember the password from its own. If user will not put any password then system will generate the password from its own and use for encryption and decryption.

Encryption /decryption for Network: when users are sending the encrypted data from one computer to another in network it is necessary to have the same key on both side else data will never be retrieve back. When user will send file first user set the password which will use negotition protocol to decide the pair of key for sender and receiver. These key will use by them for encryption and decryption. For encryption and decryption crypto A PI are used that are explain below.

Cryptography API revolves around these two functions—the encrypting (CryptEncrypt) and decrypting (CryptDecrypt) of data.

These two functions are extremely useful but require some explanation about their parameters.

1. The first six parameters of each function are the same.
2. The first two parameters are simply handles to the key and an optional hash object.

3. The third parameter is a Boolean that remains FALSE until the last block of data, at which point it must be set to TRUE so that the function can do some special processing for the last block of data.
4. The fourth and fifth parameters are simply a flag value and a pointer to the data to be encrypted or decrypted.
5. The sixth parameter is the number of characters in the buffer to be encrypted.
6. The seventh parameter is usually the same as the sixth parameter in that it specifies how long the block is. This is because for many algorithms the resulting encrypted data is the same length as the decrypted data. However, certain algorithms may increase the length of the encrypted data. In those cases the buffer pointed to by the fifth parameter must be long enough to handle the extra data.

The problem of the longer buffer can be alleviated by using the CryptEncrypt function itself to return the size of the required buffer prior to encryption.

In this sample code , certain values are assumed to have been obtained earlier, and only want to encrypt one buffer of data pointed to by pData, which is dwDataLen bytes in length.

```
//start of code //
BOOL bResult;
PBYTE pBuffer;
DWORD dwSize;
// Set variable to length of data in buffer.
dwSize = dwDataLen;
// Have API return us the required buffer size.
bResult = CryptEncrypt(
    hKey,          // Key obtained earlier
    0,            // No hashing of data
    TRUE,         // Final or only buffer of data
    0,           // Must be zero
    NULL,        // No data yet, simply return size
```

```

        &dwSize,    // Size of data
        dwSize);  // Size of block
// now have a size for the output buffer, so create buffer.
pBuffer = new char[dwSize];
// Now encrypt data.
bResult = CryptEncrypt(
    hKey,        // Key obtained earlier
    0,          // No hashing of data
    TRUE,       // Final or only buffer of data
    0,         // Must be zero
    pBuffer,    // Data buffer
    &dwSize,   // Size of data
    dwSize);   // Size of block
//end of code//

```

6.2 Generating Keys

These three functions are the ones used to generate handles to keys:

1. The `CryptDeriveKey` function is used to generate a key from a specified password.
2. The `CryptGenKey` function is used to generate a key from random generated data.
3. The `CryptDestroyKey` function is used to release the handle to the key object.

If the `CryptGenKey` function is used, it is recommended that the `CRYPT_EXPORTABLE` parameter be used to create an exportable session key. This creates a value that can be moved from one computer to another. Without this parameter the value returned is only valid on that particular computer/session.

Following is an example of how to use the `CryptDeriveKey` function, assuming that `pPassword` points to a user-defined password and `dwPasswordLength` contains the length of the password.

```

// start of sample code    //
#include <wincrypt.h>    // CryptoAPI definitions
BOOL bResult;
HCRYPTHASH hHash;
HCRYPTKEY hKey;
// Obtain handle to hash object.
bResult = CryptCreateHash(
    hProv,        // Handle to CSP obtained earlier
    CALG_MD5,    // Hashing algorithm
    0,           // Non-keyed hash
    0,           // Should be zero
    &hHash);     // Variable to hold hash object handle
// Hash data.
bResult = CryptHashData(
    hHash,        // Handle to hash object
    pPassword,   // Pointer to password
    dwPasswordLength, // Length of data
    0);          // No special flags
// Create key from specified password.
bResult = CryptDeriveKey(
    hProv,        // Handle to CSP obtained earlier.
    CALG_RC4,    // Use a stream cipher.
    hHash,        // Handle to hashed password.
    CRYPT_EXPORTABLE, // Make key exportable.
    &hKey);     // Variable to hold handle of key.
Use key to do something.
// Release hash object.
CryptDestroyHash(hHash);
// Release key object.
CryptDestroyKey(hKey);    //end of sample code//

```


6.3 File Hashing

To ensure the integrity of data, a hash of that data can be sent to accompany it. The receiver can then compare a hash that it computes on the received data with the hash that accompanied the received data. If the two match, the received data must be the same as the data from which the received hash was created. A hash is a fixed-length string of numbers and characters. It is computed using a hashing algorithm, such as Message Digest 5 (MD5), SHA and RIPEMD. Following steps are explaining the steps of file hashing and its comparison. Sender will calculate the hash code1

1. Choose the hash algorithm from the available algorithms MD5, SHA, RIPEMD.
2. Locate or Open the file to calculate the hash code.
3. Calculate the hash code say hash code1 .
4. Save the hash code 1 into the file with desired name and location.

Sender will send the hash code 1 file and original file to the destination. Verify the file or data integrity as follow

1. Locate and load the hash code 1 file.
2. Locate the file for which user want to check the data integrity.
3. Calculate the hash code 2 using the same algorithm.
4. Compare the hash code 1 and hash code 2.

If in the way any bit if the data is change both code hash1 and hash code 2 will show miss match and give result unsuccessful else successful.

6.4 File Transfer using TFTP

TFTP is designed to be implemented on top of the Datagram protocol (UDP). Since Datagram is implemented on the Internet protocol, packets will have an Internet header, a Datagram header, and a TFTP header.

A transfer is established by sending a request (WRQ to write onto a foreign file system, or RRQ to read from it), and receiving a positive reply, an acknowledgment packet for write, or the first data packet for read. In general an acknowledgment packet will contain the block number of the data packet being acknowledged.

The following shows the steps used to establish a connection to write a file. WRQ, ACK, and DATA are the names of the write request, acknowledgment, and data types of packets respectively.

1. Host A sends a "WRQ" to host B with source= A's TID, destination= 69.
2. Host B sends a "ACK" (with block number= 0) to host A with Source= B's TID, destination= A's TID.

At this point the connection has been established and the first data packet can be sent by Host A with a sequence number of 1. In the next step, and in all succeeding steps, the hosts should make sure that the source TID matches the value that was agreed on in steps 1 and 2. If a source TID does not match, the packet should be discarded as erroneously sent from somewhere else. An error packet should be sent to the source of the incorrect packet, while not disturbing the transfer.

Initial Connection Protocol for reading a file

1. Host A sends a "RRQ" to host B with source= A's TID, destination= 69.
2. Host B sends a "DATA" (with block number= 1) to host A with source= B's TID, destination= A's TID.

6.4.1 TFTP Packets

TFTP supports five types of packets, all of which have been mentioned above:

Opcode	Operation
1	Read request (RRQ)
2	Write request (WRQ)
3	Data (DATA)
4	Acknowledgment (ACK)
5	Error (ERROR)

6.5 Authentication

To authenticate the client, server will keep the records of user names and password of the users. Client will need to transfer the user name (ID) and password to the server which can be confirmed by server. Authentication is a first layer of security in the application. Server will contain the username and password of all the authorized users, according to need. Server will add the user or remove the users from its list. This function will only provide protection from external threats from accessing the server. User name and password are alphanumerical characters and using sockets user can exchange the information between client and server. Server is always listening the request from client and will connect and bind the client to a specific port. Then client will type its username and password which will give the authentication to client else break the connection with error message.

INTERFACE AND DESCRIPTION

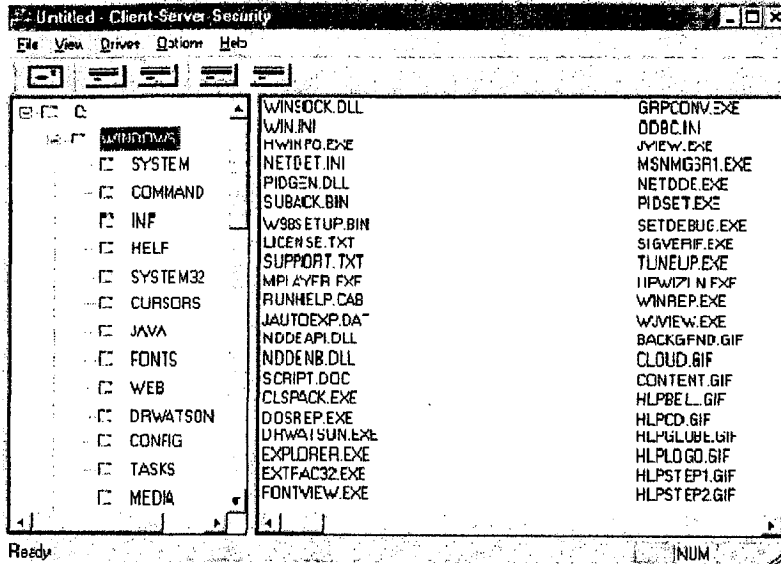


Figure 7.1: Main interface of application.

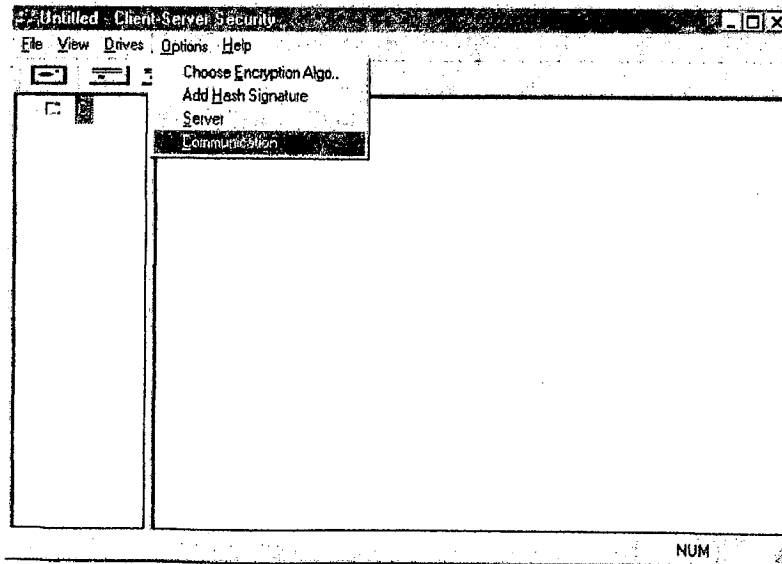


Figure 7.2: Option Bar

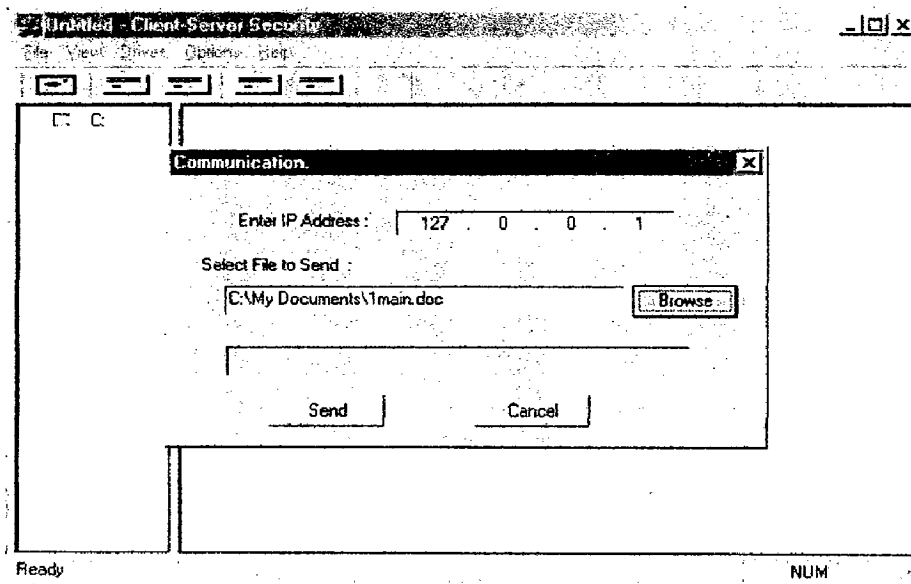


Figure 7.3: Communication Button

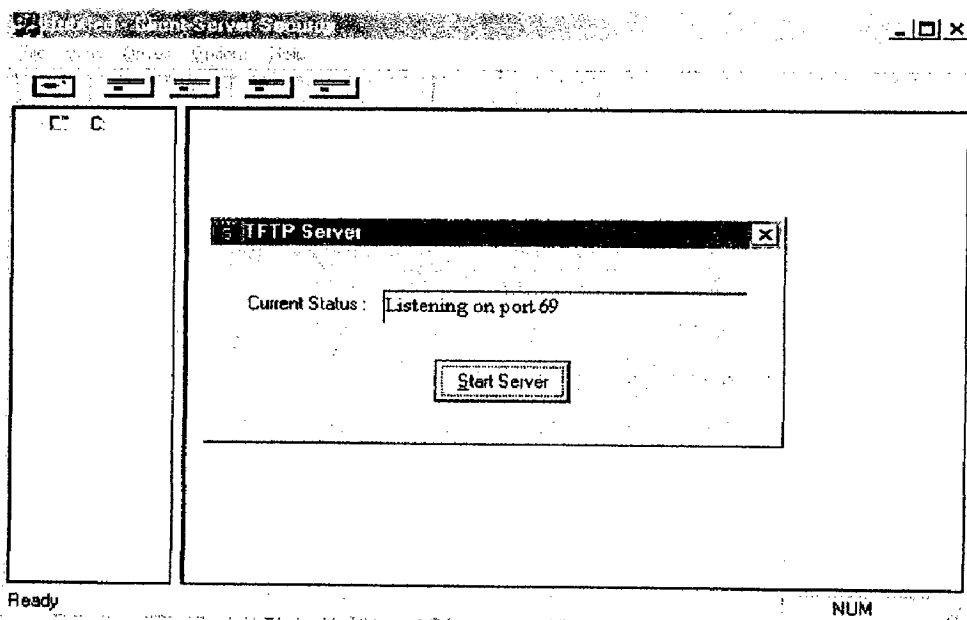
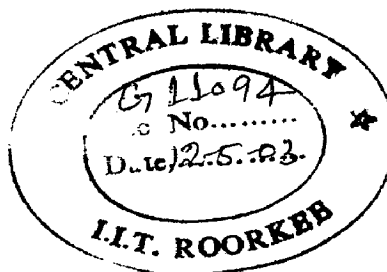


Figure 7.4: TFTP Server.



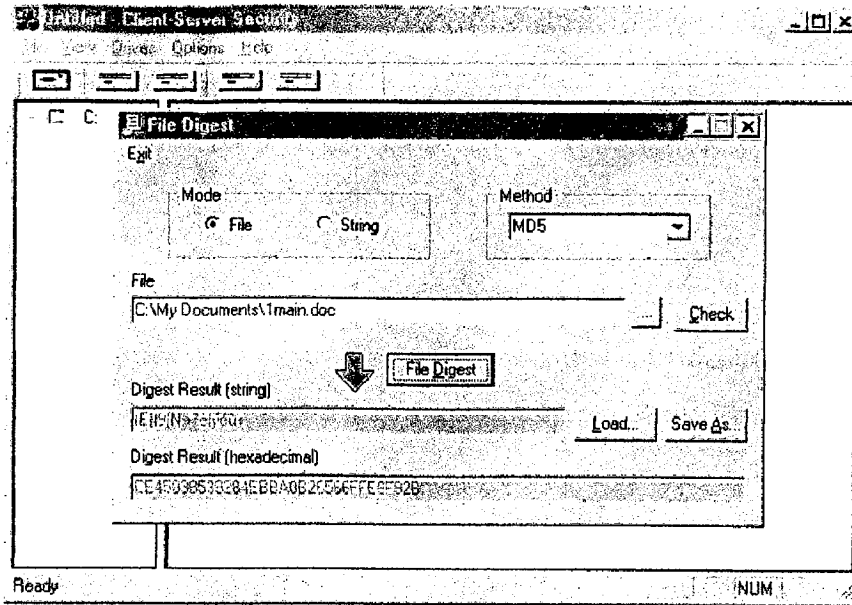


Figure 7.5: Hashing Function,

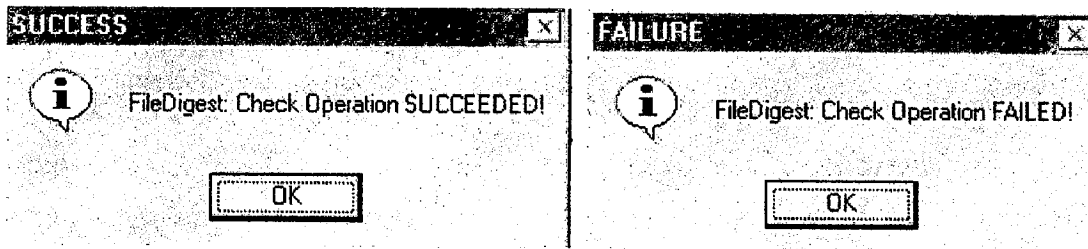


Figure 7.6: Data Integrity Checks.

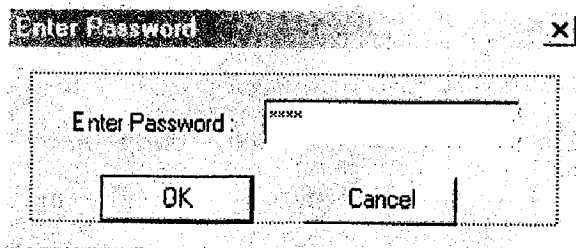


Figure 7.8: Password Option.

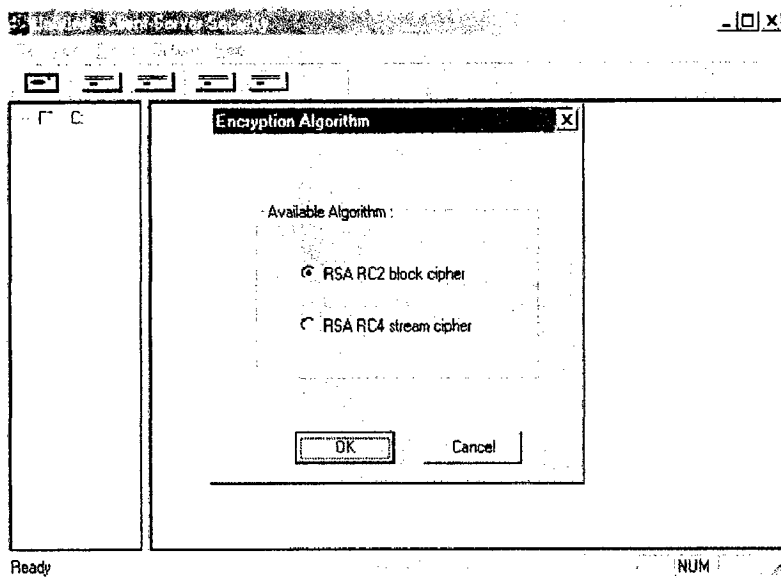


Figure 7.7: Encryption Algorithms Available.

Figure 7.1 shows the main interface of the application, which provides access to all files and drives available in the system and the options in menu provide hashing, encryption /decryption and sending and receiving the files. Figure 7.2 shows the Option Bar with 'choose encryption'; 'add Hash Signature'. 'Server' is 'TFTP' server, which receives the files. The 'communication button' is used to send the file to the user. Figure 7.3 shows the communication interface, which is used to send the file to desired IP address. If there is any error in communication, it will be shown in the dialog box. Figure 7.4 shows the TFTP server which listens on the port number 69. Figure 7.5 shows the hashing function 'File digest', which calculates the hash of the file. User can see the result in string form or Hexadecimal form. The 'save as' button will save the hash code in the file with FDG extension. The 'Load' button will load the hash code in the program for verification. Figure 7.6 shows Data integrity. It verifies the results by comparing the old hash code from the file with the newly calculated hash code. If both are same, then check operation is successful. Else it is failed. Figure 7.7 shows password options for selecting the encryption and decryption keys. Figure 7.8 shows encryption algorithms RC2 block cipher and RC4 stream cipher.

CONCLUSION

“Implementing Client/Server Communication Security ”the application will provide end-to-end security mechanisms for file transfer using the client/server infrastructure. The application allows client and server to communicate in a way that data cannot be eavesdropped and Server always authenticated the clients. The application provides "security" which has three basic properties:

- 1) Data confidentiality
- 2) Authentication
- 3) Data integrity.

The Application is providing the sophisticated way to user to browse the file from the available drives and simple way to encrypting /decrypting , and hashing the files. It provides the end user to authenticate on server and send the files from one location to another in the Network. The intention is to make a professional and general Application that is attractive to businesses and consumers. It has scope for further improvement in the following areas:

File transfer: To transfer the file from one place to another developer can use the File Transfer Protocol, which will provide more functionality and security to the user. Server will have more control and server will monitor the activity in the network.

Signature: Developer can use Public key algorithm and hash algorithm combination to generate the signature of file and after signing, hash file can be packed with original file and then encrypt that file to provide more security to data.

REFERENCES:

1. Electronics Research And Development Center Of India, Customer Requirement Document of NSFDC project, 2002.
2. Bruce Schneier , “Security Pitfalls in Cryptography Paper” , [www. Counterpane.com](http://www.Counterpane.com), by Counterpane Internet Security, Inc.2002.
3. William Stalling, Cryptography and Network Security Second Edition, Prentice Hall Inc., 1999, pp 4-9,93-127,272-298.
4. Bruce Schneier , Applied cryptography Second Edition, John Wiley & Sons, Inc., 2001. pp, 151-185,213-232
5. Andrew S. Tanenbaum ,Computer Networks Third Edition, Prentice Hall of India Private Limited, 1996.pp, 681-695.
6. K. Sollins, “RFC 1350 The TFTP Protocol (Revision 2)”, Network Working Group July, 1992.
7. Eugene Olafsen, Kenn Scribner & K. David White , MFC Programming With Visual C++, Techmedia, 1999.
8. Richard.M.Jones, Introduction to MFC Programming With Visual C++, Pearson Education Asia, 2000.
9. Microsoft Corporation “ Microsoft Developer Network (MSDN) Library” April 2002.

INTRODUCTION TO VISUAL C++

A.1 Visual C++ 6.0

Microsoft Visual C++ is a developer studio member, which provides integrated Development for the development of project.

A.2 The Visual C++ Development Environment

The Visual C++ Development Environment mainly contains three areas-Workspace, Output Pane and Editor Area. Each of these areas has a specific purpose in the developer studio environment.

A.2.1 The Workspace

This is the key area to navigating the various pieces and parts of users development project. The workspace allows user to view the parts of user application in three different ways:

- Class View allow user to navigate and manipulate users source code on a C++ class level
- Resource View allow user to find and edit each of the various resources in the application, including dialog window designs, icons, and menus.
- File View allows user to view and navigate all the files that male up in the user's application.

A.2.2 The Output Pane

The Output Pane is where Developer Studio provides any information that it needs to give the user ; where user can see all the compiler progress statements, warnings, and error messages; and where the Visual C++ debugger displays all the variables with their current values as you step through the user's code.

The above screen shows the different areas in a Visual C++ development environment. The following screen shows the MFC ClassWizard using which the functions, classes, member variables, etc can be added easily.

A.2.3 The Editor Area

This is the area where user can perform all user's editing when user use Visual C++, where the code editor windows display when user edit C++ source code, and where the window painter displays when user design a dialog box. The Editor Area is even where the icon painter displays when user designs the icons for use in the user's application. The editor area is basically the entire Developer Studio area that is not otherwise occupied by panes, menus, or toolbars.

A.3 Visual C++ program

A Visual C++ program consists of C++ code and selected basic framework provided by Visual C++ compiler. These frameworks are to be selected by users according to the type of program he wants to develop.

A.3.1 Kinds of Visual C++ programs

- An MFC program
- An MFC DLL
- An MFC ActiveX control program
- An ActiveX Container program
- An Internet Server(ISAPI) Extension or Filter
- A Win32(non-MFC) program for Windows
- A Win32 DLL
- A Static Library
- A Console program
- A Utility project
- A program that supports a DAO or ODBC database
- An Extended Stored Procedure

A.4 Microsoft Foundation Classes

The Microsoft Foundation Class Library (MFC) is an “application framework” for programming in Microsoft Windows. Written in C++, MFC provides much of the code necessary for managing windows, menus, and dialog boxes; performing basic input/output; storing collections of data objects; and so on. All user needs to do is add user’s application-specific code into this framework. And, given the nature of C++ class programming, it’s easy to extend or override the functionality the MFC framework supplies.