DATA ENCRYPTION AND DECRYPTION

SRM SYSTEMS AND SOFTWARE LIMITED

P - 1076

PROJECT REPORT

Submitted in partial fulfillment of the
Requirements for the award of the Degree of
Master Of Science In Applied Science - Software Engineering
Of Bharathiar University, Coimbatore.

Submitted By

Mr. N.Rajesh

Reg.No: 0037S0096

Guided By
Mrs. S.Devaki, B.E., M.S.
Asst. Professor, Kumaraguru college of Technology,
Coimbatore - 641 006.

Mr. S. Kamesh
Software engineer
SRM Systems And Software Limited
Chennai.

Department of computer Science and Engineering

Kumaraguru College of technology

Coimbatore - 641 006.

CERTIFICATE

Department of Computer Science and Engineering

Kumaraguru College of technology

Coimbatore – 641 006.

This is to certify that the project work entitled

" Data Encryption and Decryption "

Done By

Mr. N.Rajesh

Reg.No: 0037S0096

Submitted to the partial fulfillment of the requirements for the award of the Degree of Master of Science in Applied Science – Software Engineering of Bharathiar University, Coimbatore.

During the Academic Year 2003 - 2004

[illegible]
Signature of Guide

[illegible]
Head of the Department

Certified that the candidate was examined by us in the project in the Project Work Viva Voce Examination held on 29/9 and the University Register Number was **<u>0037S0096.</u>**

[illegible] 29/9
Internal Examiner

[illegible]
External Examiner
29/9

M SYSTEMS AND SOFTWARE LIMITED
G.N. Chetty Road, T.Nagar, Chennai - 600 017.
1 - 44 - 8250771, 8258757, 8269471 Fax : 91 - 44 - 8283359
il : srm@srmsoft.co.in Web Site : http://www.srmsoft.com
. Off : 2, Veerasamy St., West Mambalam, Chennai - 600 033.

SRM
SYSTEMS AND SOFTWARE

23.09.2003

CERTIFICATE

This is to certify that the project work entitled "**DATA ENCRYPTION & DECRYPTION**" was Analyzed, Designed and Developed by **Mr.N.RAJESH** of **KUMARAGURU COLLEGE OF TECHNOLOGY (CBE)**, submitted in partial fulfillment of the requirements of degree of 4th **Year M.Sc (S.E)** has been carried out in our organization from June 2003 to Sep 2003.This project has been developed using VC++.

We wish him success in all his future endeavors.

For SRM Systems And Software Limited

[illegible]

Manager-Projects

DECLARATION

I hereby declare that the project entitled "**DATA ENCRYPTION AND DECRYPTION**" submitted to Bharathiar University, Coimbatore, as the project work of Master of Science in Applied Science - Software Engineering, is a record of original work done by me under the supervision and guidance of Mr. Kamesh, Software Engineer, SRM Systems and Software LTD and Mrs. S. Devaki, B.E., M.S., Asst. Professor, Department of Computer Science and Engineering, Kumaraguru college of Technology, Coimbatore. And this project work has not found the basis for the award of any Degree/ Diploma/ Associate ship/ Fellowship or similar title to any candidate of any University.

[illegible]

Signature of the Student

Place: COIMBATORE

Date: 26/09/2003

Countersigned By

Project Guide
(Mrs. S. Devaki)

ACKNOWLEDGEMENT

At the outset, I would like to remember the sacrifices made by two people, who have all along been with me, and who are, mainly responsible for what I am today - **My Parents**.

I would like to express my gratitude to our beloved Principal **Dr. K.K.Padmanabhan**, Kumaraguru college of Technology, Coimbatore, for his constant encouragement throughout my course.

I wish to thank **Prof.S.Thangaswamy**, Prof & Head, Department of Computer Science and Engineering, Kumaraguru college of Technology, Coimbatore, for his invaluable guidance and suggestions that encouraged me to complete this project successfully.

I admit my heart full thanks to my internal project guide, **Mrs. S.Devaki**, B.E., M.S., Asst. Professor, Department of Computer Science and Engineering, Kumaraguru college of technology, Coimbatore, for being supportive throughout the tenure of my project.

I am especially thankful to **Mr. S. Kamesh**, Software Engineer and Project Guide, SRM Systems and Softwares limited, Chennai, for providing me guidance and suggestions throughout the tenure of my project.

I also take this opportunity to extend my sense of gratitude to all the faculty members, non - teaching staffs of the Computer Science & engineering Department, K.C.T, Coimbatore, for their guidance and co - operation rendered throughout my course.

SYNOPSIS

The project **"DATA ENCRYPTION AND DECRYPTION"** aims the protection or security of the data, by using encryption and decryption.

The project aims to provide a more reliable, accurate and fast solution for data encryption and decryption. A through study existing manual system and discussion with the personnel involved to this particular problem revealed the difficulties faced by the user and lead to the development of the proposed project **"DATA ENCRYPTION AND DECRYPTION".**

In the proposed system Visual C++ is used for convert the data into non - readable format or cipher text format and cipher text format into readable format or normal text. In this project the normal text is converted into cipher text within seconds. The cipher text is converted into normal text in no time.

CONTENTS

1. INTRODUCTION

1.1 Data encryption and decryption

The Encryption Project uses a private-key encryption to encrypt files. Private-key encryption algorithms use a single private key to encrypt and decrypt data so it also referred to as symmetric encryption because the same key is used for encryption and decryption. Thus, we need a key and an initialization vector to encrypt and decrypt data. Without an the same input block of plaintext will encrypt to same output block of cipher text, but with the output of two identical plaintext blocks are different and it is hard for unauthorized user to recover the key. The disadvantage of private-key encryption is that it presumes two parties have agreed on a key and communicated their values. Also, the key must be kept secret from unauthorized users. Because of these problems, private-key encryption is often used in conjunction with public-key encryption to privately communicate the values of the key.

1.2 Basics Of Data encryption and Decryption

Cryptography deals with the transformation of ordinary text (plaintext) into a coded form (cipher text) by encryption and the transformation of cipher text into plaintext by decryption. Historically, before the advent of mechanical or electrical computers, the transformation was performed by hand and included, for example, the procedures of substitution and transposition. Whether performed by hand or by computer, these procedures, or transformations, are mathematical in nature. The transformation procedure is known as the cryptographic algorithm.

In a computer environment, the encryption and decryption algorithm uses a cryptographic key to perform these mathematical transformations. The key functions as an input parameter to vary the transformation of plaintext to cipher text and vice versa. The data are more secure to transfer through e -mail or through some other media. The information's are not able to identified or sensed by others.

2. System Study and Analysis

2.1 Feasibility Study

Encryption

Encryption is the transformation of data into a form unreadable by anyone without a secret decryption key. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended, even those who can see the encrypted data. For example, one may wish to encrypt files on a hard disk to prevent an intruder from reading them.

In a multi-user setting, encryption allows secure communication over an insecure channel. The general scenario is as follows: Alice wishes to send a message to Bob so that no one else besides Bob can read it. Alice encrypts the message, which is called the plaintext, with an encryption key; the encrypted message, called the cipher text, is sent to Bob. Bob decrypts the cipher text with the decryption key and reads the message. An attacker, Charlie, may either try to obtain the secret key or to recover the plaintext without using the secret key. In a secure cryptosystem, the plaintext cannot be recovered from the cipher text except by using the decryption key. In a symmetric cryptosystem, a single key serves as both the encryption and decryption keys.

Authentication

Authentication in a digital setting is a process whereby the receiver of a digital message can be confident of the identity of the sender and/or the integrity of the message. Authentication protocols can be based on either conventional secret-key cryptosystems like DES or on public-key systems like RSA; authentication in public-key systems uses digital signatures.

Public Key Cryptography

Traditional cryptography is based on the sender and receiver of a message knowing and using the same secret key: the sender uses the secret key to encrypt the message, and the receiver uses the same secret key to decrypt the message. This method is known as secret-key cryptography. The main problem is getting the sender and receiver to agree on the secret key without anyone else finding out. If they are in separate physical locations, they must trust a courier, or a phone system, or some other transmission system to not disclose the secret key being communicated. Anyone who overhears or intercepts the key in transit can later read all messages encrypted using that key. The generation, transmission and storage of keys is called key management; all cryptosystems must deal with key management issues. Secret-key cryptography often has difficulty providing secure key management.

Whitfield Diffie and Martin Hellman invented public-key cryptography in 1976 in order to solve the key management problem. In the new system, each person gets a pair of keys, called the public key and the private key. Each person's public key is published while the private key is kept secret. The need for sender and receiver to share secret information is eliminated: all communications involve only public keys, and no private key is ever transmitted or shared. No longer is it necessary to trust some communications channel to be secure against eavesdropping or betrayal. Anyone can send a confidential message just using public information, but it can only be decrypted with a private key that is in the sole possession of the intended recipient. Furthermore, public-key cryptography can be used for authentication (digital signatures) as well as for privacy (encryption).

Here's how it works for encryption: when Alice wishes to send a message to Bob, she looks up Bob's public key in a directory, uses it to encrypt the message and sends it off. Bob then uses his private key to decrypt the message and read it. No one listening in can decrypt the message. Anyone can send an encrypted message to Bob but only Bob can read it. Clearly, one requirement is that no one can figure out the private key from the corresponding public key.

Here's how it works for authentication: Alice, to sign a message, does a computation involving both her private key and the message itself; the output is called the digital signature and is attached to the message, which is then sent. Bob, to verify the signature, does some computation involving the message, the purported signature, and Alice's public key. If the results properly hold in a simple mathematical relation, the signature is verified as genuine; otherwise, the signature may be fraudulent or the message altered, and they are discarded.

2.2 Proposed System Study

Encrypting File System (EFS) allows users to store data securely on local computers. EFS does this by encrypting data in selected **NTFS** files and folders. Because EFS is integrated with the file system, it is easy to manage, difficult to attack, and transparent to the user. This is particularly useful for securing data on computers that may be vulnerable to theft, such as mobile computers.

Files and folders cannot be encrypted or decrypted on FAT volumes. Also, EFS is designed to store data securely on local computers. As such, it does not support the sharing of encrypted data.

EFS encryption keys

Once a user has specified that a file be encrypted, the actual process of data encryption and decryption is completely transparent to the user. The user does not need to understand this process. However, the following explanation of how data encryption and decryption works might be useful for administrators.

This explanation only applies to files, not folders. Folders themselves are not encrypted, only the contents of the files within a folder. Like folders, sub-folders are not encrypted; however, they are marked to indicate that they contain encrypted file data.

Encryption of files works as follows:

- Each file has a unique file encryption key, which is later used to decrypt the file's data.
- The file encryption key is in itself encrypted--it is protected by the user's public key corresponding to the user's EFS certificate.
- The public key of an authorized recovery agent also protects the file encryption key.

Decryption of files works as follows:

- To decrypt a file, the file encryption key must first be decrypted. The file encryption key is decrypted when the user has a private key that matches the public key.
- The user is not the only person that can decrypt the file encryption key. A recovery agent can also decrypt the file encryption key, by using the recovery agent's private key.
- Once the file encryption key is decrypted, either the user or the recovery agent to decrypt the data in the file can use it.

Private keys are held in a protective key store, and not in the Security Account Manager (SAM) or in a separate directory.

3. Programming Environment

3.1 Hardware Configuration

Hardware specification of the system is used in the project:

Pentium 300 MHZ

128MB RAM / 20 GB Hard Disk Drive

1.44 MB Floppy Disk Drive

SVGA Monitor

104 keys Keyboard

Platform:

Operating System	: Microsoft Windows98
Software Used	: Visual C++

3.2 Description Of Software's & Tools Used

Visual C++ 6.0

The major purpose of Visual C++ is to support the newest Microsoft technologies, such as the New Active Desktop in Internet Explorer 4.0. With Visual C++ 6.0 we can develop for Internet Explorer 4.0 by accessing the power and flexibility of Dynamic HTML and the new common controls. Internet Explorer 4.0 support enables us to integrate the web directly into Microsoft Foundation Class (MFC) applications.

Visual C++ has supported writing ActiveX documents for some time. Now it supports writing ActiveX document containers, including support for printing, saving, and loading. We can seamlessly get all the functionality of programs such as Excel or Word in our MFC applications with DocObject Containment. You can easily put full – featured charts, graphs, or even Web browsers right in our applications, with full menu merge.

The Visual C++ Integrated Development Environment, or IDE is organized into four distinct areas. They are, Menu and Toolbars, Project view window, code editor, and debug window. The menu items enable we to access different options not only for the IDE but also for our overall project. Using different menu options, we can control everything from the compiler's behavior of the code editor. All the toolbars are fully dockable. This means that they can be docked to any of the four sides of the IDE' s main window. We can also "float" the toolbars anywhere on our desktop.

The debug window displays important during the project building process and while our project is executing within the Visual Studio debugger. Typically, the debug window will display error messages that occur when we compile, or build, our project. We can also write messages directly to the debug window from within our project by using the Trace Macro.

Microsoft Foundation Class (MFC)

MFC is the abbreviation for a collection of C++ classes produced by Microsoft that are called the Microsoft Foundation Classes. MFC provides an object - oriented framework those application developers can use to create Windows applications. MFC is organized as a hierarchy of C++ classes. Several high - level classes provide general functionality while the low - level classes implement more specific behaviors. Each of the low - level classes is derived from a high - level class and, thus, inherits the behaviors of the high - level class.

MFC handles many common Windows - related tasks, such as having to write to the same message-handling loop in every Windows application we develop, MFC implements the message loop for us and provides easy to understand and use member functions, like OnPaint(), which enables us to insert code to handle the window message.

In addition to the class hierarchy, MFC also provides an application development model. This model is called the Document/View model. Document/View, or Doc/View is method of designing an application so that the application's data is separated fro the user interface elements. This allows the two parts of the application to stand on their own, enabling the programmer to make changes to one without having to make drastic changes to the other.

3.3 Cryptography

The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form. An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods is called cipher. The transformed text is called cipher text. Some critical information used by the cipher, known only to the sender and receiver is called key. The key is used for decrypting the data from the cipher text format.

The process of converting plaintext to cipher text using a cipher and a key is called Encipher (Encode). The process of converting cipher text back into plaintext using a cipher and a key is called Decipher (Decode).

The study of principles and methods of transforming an unintelligible message (cipher text) back into an intelligible message (normal text) without knowledge of the key is called cryptanalysis, it also called code breaking. Both cryptography and cryptanalysis is called cryptology.

3.4 Private Key Cryptography

The most ancient and basic problem of cryptography is secure communication over an insecure channel. A wants to send to B a secret message over a communication line that may be tapped by an adversary. The traditional solution to this problem is called private key encryption.

In private key encryption, A and B agree on a pair of encryption and decryption algorithms E and D, and an additional piece of information S to be kept secret. We shall refer to S as the shared secret key. The adversary may know the encryption and decryption algorithms E and D which are being used, but does not know S.

Private Key Algorithms are also known as Symmetric Algorithms.

Block Ciphers

Symmetric-key block ciphers are the most prominent and important elements in many cryptographic systems. Individually, they provide confidentiality. As a fundamental building block, their versatility allows construction of pseudorandom number generators, stream ciphers, MACs, and hash functions. They serve as a central component in message authentication techniques, data integrity mechanisms, entity authentication protocols, and (symmetric-key) digital signature schemes.

No block cipher is ideally suited for all applications, even one offering a high level of security. This is a result of inevitable tradeoffs required in practical applications, including those arising from, for example, speed requirements and memory limitations (e.g., code size, data size, cache memory), constraints imposed by implementation platforms (e.g., hardware, software, chip cards), and differing tolerances of applications to properties of various modes of operation. In addition, efficiency must typically be traded off against security.

Stream Ciphers

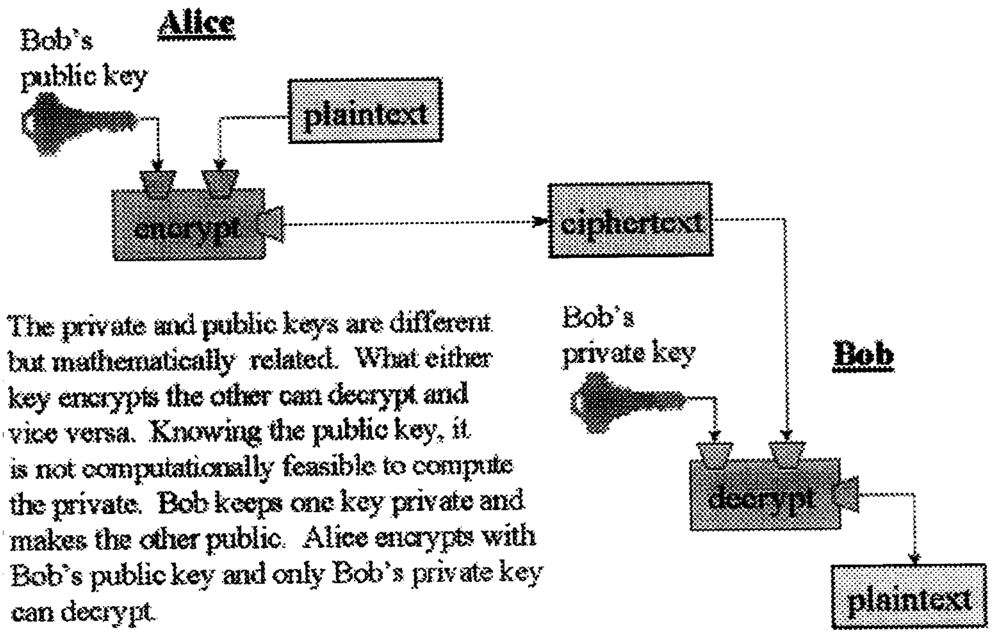
Stream ciphers are an important class of encryption algorithms. They encrypt individual characters (usually binary digits) of a plaintext message one at a time, using an encryption transformation, which varies with time. By contrast, block ciphers tend to simultaneously encrypt groups of characters of a plaintext message using a fixed encryption transformation. Stream ciphers are generally faster than block ciphers in hardware, and have less complex hardware circuitry. They are also more appropriate, and in some cases mandatory (e.g., in some telecommunications applications), when buffering is limited or when characters must be limited or no error propagation, stream ciphers may also be advantageous in situations where transmission errors are highly probable.

3.5 Public Key Cryptography

The setup of a public-key cryptosystem is of a network of users rather than a single pair of users. Each user in the network has a pair of keys associated with him, the public key, which is published under the users name in a *public directory* accessible for everyone to read, and the private-key, which is known only to the user. Running a key-generation algorithm generates the pair of keys. To send a secret message to a user everyone in the network uses the same exact method, which involves looking up the public key from the public directory, encrypting the message using the public key, and sending the resulting cipher text to the user. Upon receiving the cipher text, the receiver can decrypt by looking up his private key.

A particular public-key cryptosystem is thus defined by a triplet of public algorithms (G, E, D), the key generation, encryption, and decryption algorithms. The asymmetric key system does not have the disadvantages of a symmetric key system because the public key is made widely available so that anyone can possess it. In this system only the private key needs to be kept private. Each entity can retrieve another entity's freely available public key, thus removing key distribution management complexity.

Exhibit 3-2 shows the public key cryptography's use of the public and private keys.


Bob's public key
Alice
plaintext
encrypt
ciphertext
The private and public keys are different but mathematically related. What either key encrypts the other can decrypt and vice versa. Knowing the public key, it is not computationally feasible to compute the private. Bob keeps one key private and makes the other public. Alice encrypts with Bob's public key and only Bob's private key can decrypt.
Bob's private key
Bob
decrypt
plaintext

Advantages of public-key cryptography

1. Only the private key must be kept secret (authenticity of public keys must, however, be guaranteed).
2. The administration of keys on a network requires the presence of only a functionally trusted TTP as opposed to an unconditionally trusted TTP. Depending on the mode of usage, the TTP might only be required in an *offline* manner, as

3. Depending on the mode of usage, a private key/public key pair may remain unchanged for considerable periods of time, e.g., many sessions (even several years).
4. Many public-key schemes yield relatively efficient digital signature mechanisms. The key used to describe the public verification function is typically much smaller than for the symmetric-key counterpart.
5. In a large network, the number of keys necessary may be considerably smaller than in the symmetric-key scenario.

Disadvantages of public-key encryption

1. Throughput rates for the most popular public-key encryption methods are several orders of magnitude slower than the best-known symmetric-key schemes.
2. Key sizes are typically much larger than those required for symmetric-key encryption, and the size of public-key signatures is larger than that of tags providing data origin authentication from symmetric-key techniques.
3. No public-key scheme has been proven to be secure (the same can be said for block ciphers). The most effective public-key encryption schemes found to date have their security based on the presumed difficulty of a small set of number-theoretic problems.

4. SYSTEM DESIGN AND DEVELOPMENT

System Analysis and design comprise of the design, file design and output design phases. All these phases are related to one another in some manner. So they will not be designed in separate ways. Hence this will be done only in an integrated way.

Another thing is requirement of user. Each user has same type of requirements. Hence design of the system completely depends on the requirements of user. In this project, the user can work easily. In this project the user give only the input file during encryption and give input file (.enc file) and the key (generated during encryption) for decryption.

4.1 INPUT DESIGN

The input design to the Data encryption and decryption contains the path of the input fie and the destination for both output file and the key during encryption and decryption. So input screens have been designed according to these details. All these information are inside a single window.

The screens are well laid without any cramping of input fields.
[illegible] available wherever possible, so that the user can select

input files from the explore. Thus, the screen is designed to be very user - friendly.

Validation at the screen design level is simple but it restricts the unauthorized person. The software has a password, it protect the software. This software encrypts the input file (Normal data) into the cipher text (.enc file) within few seconds. And it decrypts the input file (.enc file) into normal data format in no time. The cipher text is not understandable and this is not possible to decrypt into normal data format without using key.

4.2 OUTPUT DESIGN

Output design is a very important phase in the designing of a system. The important objective of any system is in its capability of producing high quality outputs.

This system has three outputs. They are,

1. Encrypted data (cipher text) from normal data format after Encryption.
2. The Key generated during Encryption for protect the encrypted data.
3. Normal data from Cipher text after Decryption.

In this software, after encryption creates two outputs one is encrypted format (cipher text) of the input file and the second one is key for decrypt. After decryption it generates the original format i.e., normal text format from the cipher text.

5. SYSTEM IMPLEMENTATION AND TESTING

5.1 System Implementation

The project undergoes a versioning and release management before it is delivered to the Users. It is a process of identifying and keeping track of different versions and releases of the software. And the released product usually includes Configuration files defining how the release should be configured for particular installations. Data files needed for successful operations. An installation program, which is used to help install the system on the target hardware. Electronic and paper documentation describing the system. All these information are made available on a medium, which can be read and understood by the customer for the software.

Operational Documentation

Properly produced and maintained system documentation is a tremendous aid to maintenance engineers. The system documentation includes all of the documents describing the implementation of the system from the requirements specification to the final acceptance rest plan.

A complete set of Operational Documentation was prepared for the User or Administrator, which included the features of this procedure was included in the documentation for data encryption, data decryption and the use of key generated after the encryption for decrypt the data. The documentation is prepared keeping in mind users who have little or no knowledge of computers.

The operational documentation includes a document describing the overall architecture, a maintenance guide, a user manual for operations like how to data encrypted and decrypted, and how to use the key and maintain the key. The purpose of input controls and the validations for the same are explained diagrammatically. A clear picture of the system and its functionalities are thus provided.

System Maintenance:

The process of changing a system after it has been delivered and is in use is called software maintenance. The changes may involve simple changes to correct coding errors, more extensive changes to correct design errors, or significant enhancements to accommodate new requirements. It is the process of changing the system to maintain its ability to survive.

Owing to scope creeps or new requirements in the future there could be possibilities for changes in the system in future. The will make it more adaptable to changes. The application domain is clearly defined which will help in making changes to a new domain.

5.2 System Testing

Periodical tests were conducted during the design and implementation phases of development. Tests were conducted as per test plans, which were scheduled according to the company's policies. A detailed report on various tests conducted is given below. A Bottom - up testing methodology was adapted to test the system developed. A bottom - up test strategy starts with the fundamental components and works upwards.

While conceiving the Architectural Design of the design phase in development decomposing of the entire project into modules, the relationship between the normal text and the data they encrypted into cipher text thoroughly analyzed. In formulating the detail design during the design phase an analysis was conducted on the algorithm specification to implement functions, decision on data structures to represent data, and the decision of design techniques to be followed.

In the implementation phase tests were conducted according to the most widely used two stages testing process. The system

Unit Testing

Unit testing was used to test individual units (i.e., Functions) in the system and ensure that they operate correctly. Alternate logic analysis and screen validations were tested in this phase to ensure optimum efficiency in the system. The procedures and functions used and their association with data were tested.

Module Testing

Module testing was used to ensure that the dependable components in a module work in coordination with one another. Functional testing, performance testing and stress tests were conduct on modules independently to ensure robustness in the system developed. The various functions and their validations in module were analyzed and tested. The procedures and functions common to a module were also tested during module testing.

CONCLUSIO

6. CONCLUSION

Thus the data can be transmitted securely using the Encryption and Decryption. The multiple Encryptions will be very hard to break. The public key and private key encryptions are more secure than the single key encryption because others can also use the single key to decrypt the data. The disadvantage of single-key encryption is that it presumes two parties have agreed on a key and communicated their values. Also, the key must be kept secret from unauthorized users. Because of these problems, single-key encryption is often used in conjunction with public-key encryption to privately communicate the values of the key.

7. BIBLIOGRAPHY

David J. Kruglinski, George Shepherd, Scot Wingo, "Programming Microsoft Visual C++", Microsoft Press; Fifth Edition 1992

Richard C. Leinecker and Archer Tom, "Visual C++ 6 Programming Bible", IDG Books India (P) Ltd; Fifth Edition 1999

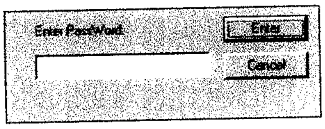
Roger S.Pressman "Software Engineering and Application", McGraw Hill; Fourth Edition 1992

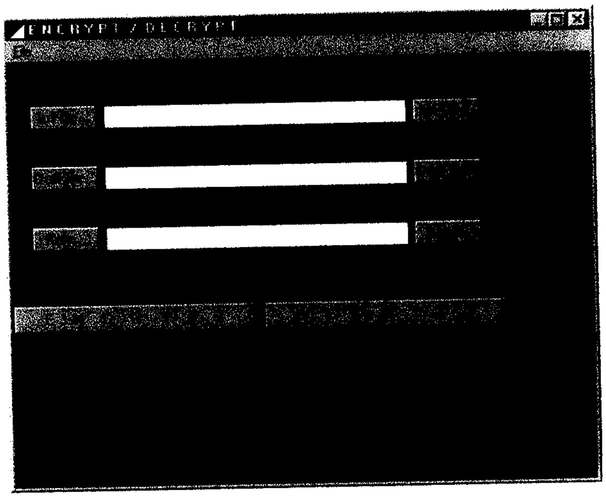
John Paul Mueller "Visual C++ 6 from the Ground up", Tata McGraw Hill, 1998

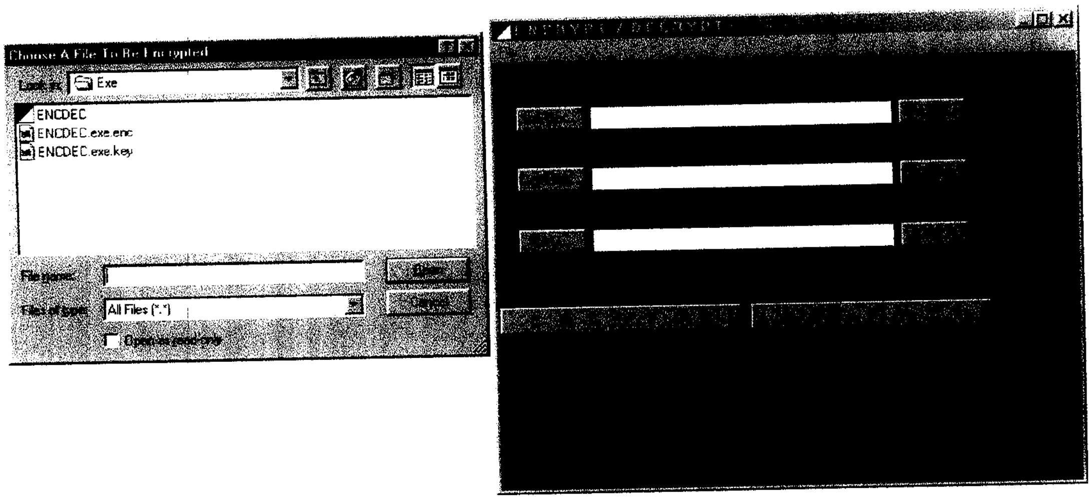
Andrew Tenanbaum " Computer Networks", Tata McGrawHill, 1996

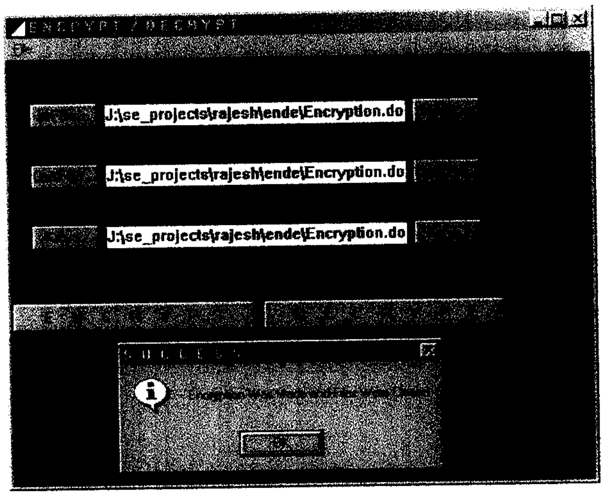
WEB SITES

- www.msdn.microsoft.com/visualc
- www.visionx.com/mfcpro


Enter Password
Enter
Cancel


ENCRYPT / DECRYPT


Choose A File To Be Encrypted
Look in: Exe
ENCDEC
ENCDEC.exe.enc
ENCDEC.exe.key
File name:
Files of type: All Files (*.*)
Open
Cancel
Open as read-only
ENCRYPT / DECRYPT



J:\se_projects\rajesh\ende\Encryption.do

J:\se_projects\rajesh\ende\Encryption.do

J:\se_projects\rajesh\ende\Encryption.do

[illegible]

APPENDIX =

Coding

```
#include"Main.h"
#include"Functions.h"

//START WINMAIN//

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE
hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)

{

DialogBox(hInst, MAKEINTRESOURCE(IDD_PASSWORD), NULL,
(DLGPROC)DlgPasswordProc );

        if(!CorrectPassword) return false;

        splash(hInstance);

        App_Init();

//FILL WINDOW CLASS//

//DECLARE WINDOW CLASS

WNDCLASS wndWc;

wndWc.style = CS_OWNDC |CS_HREDRAW | CS_VREDRAW;
wndWc.lpfnWndProc = (WNDPROC) WndProc;
wndWc.cbClsExtra = 0;
wndWc.hInstance =hInstance;
wndWc.hIcon =
LoadIcon(hInstance,MAKEINTRESOURCE(IDI_ICON));
wndWc.hCursor = LoadCursor(0, IDC_ARROW);
wndWc.hbrBackground =
(HBRUSH)GetStockObject(BLACK_BRUSH);
wndWc.lpszMenuName = MAKEINTRESOURCE(IDR_MENU1);
wndWc.lpszClassName = "EncDec";

//REGISTER WINDOW CLASS
RegisterClass(&wndWc);

//FILL WINDOW CLASS//

//CREATE MAIN WINDOW//

hWnd = CreateWindow("EncDec", "E N C R Y P T  /  D E C R Y P
T",WS_OVERLAPPEDWINDOW,rc.left,rc.top, 450,300,NULL, NULL,
hInst, NULL);

//SHOW WINDOW
ShowWindow(hWnd, SW_SHOW);

//CREATE MAIN WINDOW//

//SPLASH TIMER

splashTimer();

//MAIN LOOP//
```

```
{
TranslateMessage(&Msg);
DispatchMessage(&Msg);
}
//MAIN LOOP//

//RELEASE WINDOW

 DestroyWindow(hWnd);

  return Msg.wParam;
 }

 //END WINMAIN//

 //START WNDPROC//
 LRESULT CALLBACK WndProc(HWND hWnd, UINT Msg, WPARAM
 wParam, LPARAM lParam)
 {
  switch(Msg)
   {

 //WM_CREATE//
 case WM_CREATE:
 {
 ENC         =     CreateWindow("Button","E   N   C   R   Y   P
 T",WS_CHILD | WS_VISIBLE |
 BS_PUSHBUTTON,2,220,215,25,hWnd,(HMENU)ID_ENC,hInst,0);

           DEC         =     CreateWindow("Button","D   E   C
 R   Y   P   T",WS_CHILD | WS_VISIBLE |
KEY          =      CreateWindow("Button","Key...",WS_CHILD |
WS_VISIBLE |
BS_PUSHBUTTON,20,150,60,22,hWnd,(HMENU)ID_KEY,hInst,0);
KEY2 =      CreateWindow("Button","Key...",WS_CHILD |
WS_VISIBLE |
BS_PUSHBUTTON,360,150,60,22,hWnd,(HMENU)ID_KEY,hInst,0);

INF          =      CreateWindow("Button","InFile...",WS_CHILD |
WS_VISIBLE |
BS_PUSHBUTTON,20,40,60,22,hWnd,(HMENU)ID_INF,hInst,0);

INF2  =      CreateWindow("Button","InFile...",WS_CHILD |
WS_VISIBLE |
BS_PUSHBUTTON,360,40,60,22,hWnd,(HMENU)ID_INF,hInst,0);

OUF          =      CreateWindow("Button","OutFile...",WS_CHILD |
WS_VISIBLE |
BS_PUSHBUTTON,20,95,60,22,hWnd,(HMENU)ID_OUF,hInst,0);

OUF2 =      CreateWindow("Button","OutFile...",WS_CHILD |
WS_VISIBLE |
BS_PUSHBUTTON,360,95,60,22,hWnd,(HMENU)ID_OUF,hInst,0);

E1           =      CreateWindow("Edit",NULL,WS_CHILD |
WS_VISIBLE | WS_BORDER
,85,40,270,22,hWnd,(HMENU)ID_E1,hInst,0);

E2           =      CreateWindow("Edit",NULL,WS_CHILD |
WS_VISIBLE | WS_BORDER
,85,95,270,22,hWnd,(HMENU)ID_E2,hInst,0);
```

```
E3            =      CreateWindow("Edit",NULL,WS_CHILD |
WS_VISIBLE | WS_BORDER
,85,150,270,22,hWnd,(HMENU)ID_E3,hInst,0);

break;
}
///WM_CREATE//////////////////////////////////

//WM_COMMAND///////////////////////////////////

      case WM_COMMAND:
            {

                  switch(HIWORD(wParam))

                        {

                        case BN_CLICKED:

                              switch(LOWORD(wParam))

                                    {

                                    case ID_ENC:

                                    {

                                          Encrypt();
                                          break;
                                    }

                                    case ID_DEC:

                                    {
                                          Decrypt();
                                          break;
                                    }


                                    case ID_INF:

                                    {
                                          InFileOpen();
```

```
                                                break;
                                        }

                                        case ID_OUF:

                                        {
                                                OutFileOpen();
                                                break;
                                        }

                                        case ID_KEY:
                                        {
                                                KeyFileOpen();
                                                break;
                                        }

                                        case ID_ABOUT:
                                        {
                                                DialogBox(hInst,
MAKEINTRESOURCE(IDD_ABOUT), NULL, (DLGPROC)DlgAboutProc
);

                                                break;
                                        }

                                        case IDM_EXIT:
                                        {
                                                Exit();
                                                break;
                                        }


                                        }//switch(LOWORD(wParam))

                                }//switch(HIWORD(wParam))
                        break;
                }

/////////////////////////////////WM_COMMAND//////////////////////////
//////


/////////////////////////////////WM_DESTROY////////////////////////////
////

        case WM_DESTROY:
                {
                    break;
               }

//////////////////////////////////WM_DESTROY////////////////////////////
////


//////////////////////////////////WM_KEYDOWN////////////////////////////
/////

     case WM_KEYDOWN:
          {

               switch (wParam)
                    {
                         case VK_ESCAPE:
                         {
                         PostQuitMessage(0);
                         break;
                         }
                    }
          break;
          }

//////////////////////////////////WM_KEYDOWN////////////////////////////
/////



//////////////////////////////////WM_TIMER///////////////////////////////
/

     case WM_TIMER:
          {
          ShowWindow(SPL,SW_HIDE);
               KillTimer(hWnd,1);
               break;
          }

//////////////////////////////////WM_TIMER///////////////////////////////
/



     }//switch(Msg)
```

```
 return DefWindowProc(hWnd,Msg,wParam,lParam);

}

/////////////////////////////////////END
WNDPROC/////////////////////////////////////


/////////////////////////////////////START
DLGPROC/////////////////////////////////////

LRESULT CALLBACK DlgPasswordProc( HWND hwnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
        char Password[25]={0};


    switch( message )

    {
             case WM_INITDIALOG:




          return TRUE;

             case WM_COMMAND:


                  switch( LOWORD( wParam ) )

          {
                       case IDOK:
                             {


                             GetDlgItemText(hwnd,
 IDC_PASSWORD, Password, 25);



                             if(!strcmp(Password, "Password") ||
                                        {
                                              CorrectPassword = true;

                                              EndDialog( hwnd, FALSE );

                                        }
                                        else
                                        {

                                              MessageBox(hwnd, "InValid
password!(The Password is 'password')", "E   R   R   O   R   !",
MB_OK);
                                        }


                                        return TRUE;

                                        }
                                  case IDCANCEL:

                                        {
                                        int response;

                                        response=MessageBox(hwnd, "Are
You Sure?", "E   R   R   O   R   !?!", MB_YESNO);

                                        if(response==IDYES)
                                        {
                                        EndDialog( hwnd, FALSE );

                                        }
                                        return TRUE;

                                        }
                            }
                break;

            case WM_CLOSE:

                            {
                            EndDialog( hwnd, FALSE );


                            break;
                            }
                    case WM_DESTROY:
```

```
                    break;

    }

    return FALSE;

}

///////////////////////////////////END
DLGPROC////////////////////////////////////


///////////////////////////////////START
DLGABOUTPROC//////////////////////////////////

LRESULT CALLBACK DlgAboutProc( HWND hwnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
        switch(message)
        {

                case WM_INITDIALOG:


                        return TRUE;


                case WM_COMMAND:
                        {
                                switch( LOWORD( wParam ) )
                                {
                                case IDOK:
                                        EndDialog(hwnd,FALSE);
                                }
                        break;
                        }


                case WM_DESTROY:


                        break;

                case IDOK:
                        {
```

```
                    return TRUE;
                }
        }

return FALSE;
}

/////////////////////////////////END
DLGABOUTPROC/////////////////////////////////
```