Encryption Technology on Windows Workstation

by

Zulkhaimi bin Amdan

A dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Technology (Hons)
(Information System)

JUNE 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

t
Z
103
.Z93
2004

1) Cryptography
2) Data encryption (Computer science)
3) IT/IS -- Thesis

CERTIFICATION OF APPROVAL

Encryption Technology on Windows Workstation

by,

Zulkhaimi bin Amdan

A project dissertation submitted to the
Information System Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons.)
(INFORMATION SYSTEM)

Approved by,

[illegible]

(Ms Syarifah Bahiyah Rahayu binti Syed Mansoor)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH PERAK
Jun 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

ZULKHAIMI BIN AMDAN

ABSTRACT

This paper discusses the development of a new method for encrypting text of information in Microsoft Windows environment. The purpose of this study is to develop a new type of encryption method based on RSA technique. In addition, the objective is to tighten the security of Windows platform by implementing a new way of encrypting users' information to prevent access by unauthorized users. This is achieved by trying to combine several techniques of securing and strengthening the RSA algorithm. One of the techniques that will be looked into is 'knocking', where the authorized users will have to perform a series of predetermined actions to validate their requests for the information in addition of the public key they have. This study is done because current encryption method in Windows environment is quite weak (e.g.: Windows password) and easily breakable using program like LC4(@stake LC4, about). Windows NT, 2000 and XP use Security Accounts Manager (SAM) file to store encrypted users passwords using RSA MD4 scheme(Windows NT SAM permission Vulnerability, bulletin), so if the RSA technique is enhanced it will enable a stronger and harder to break workstations. The result of this study will be an enhanced version of current encryption technology that is ready to be implemented or applied by interested parties.

ACKNOWLEDGEMENT

BISMILLAH AR-RAHMANI AR-RAHEEM

In the Name of Allah, the Most Compassionate, the Most Merciful

I thank GOD who gave me the spirit and strength to work and proceed with this project, going through all the problems and finally able to deliver it.

My sincerest thanks go to my supervisor, Ms. Syarifah Bahiyah Rahayu binti Syed Mansoor. Thanks for her supports, patience and ideas in assisting me with this project, her never-ending enthusiasm inspires me.

I also would like to express my gratitude to staff of Universiti Teknologi Petronas, especially IT IS committee and lecturers for providing knowledge and resources and willing to help with full commitment. Your supports and commitments really help me.

Special thanks to my housemates, colleagues, and friends for helping me with this project and for always be at my side whenever I need the helps. Thanks for the courage, the passion and the friendships we shared together.

Credits to those who help and guide me on my project via Internet. Thanks for spending their precious time answering all my questions.

Last but not least, deepest gratitude to my parents and family for their enormous material and spiritual supports, without them this work would not have been possible.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1. BACKGROUND OF DISSERTATION

This research will combine two different techniques to come up with a new method of encrypting documents or files, namely RSA encryption method and networking port-knocking method. Encryption is the transformation of data into a form that is as close to impossible as possible to read without the appropriate knowledge(RSA Laboratories 10). Meanwhile, port-knocking is described as a method of establishing a connection to a networked computer with no open port(PORT KNOCKING, summary). Seeing the lack of method combination in encryption technology and the growing concern of public for the security of their data/information, it is imperative and useful to conduct this dissertation in order to overcome the problem of lack in data security and to enable commercial industry and researchers to improve this dissertation.

2. PROBLEM STATEMENT

2.1. Problem Identification

Nowadays passwords and encrypted documents are easily deciphered or cracked, enabling unauthorized users/person to get the valuable information stored in such medium. This is due to the attitude of the system users to choose simple passwords that are easy to be remembered (birth date, spouse's birth date, IC number/SSN). This lead the users exposed to social engineering, a technique used by hackers to gain information about the system they want to crack by exploiting the weakness of human mind rather than the system itself.

The lack of strong encryption method also leads to the wide number of security breaches in companies' business operations, which cost a lot of money to recover. However strong an encryption method is, it still cannot stand from the brute force attack, where the deciphering program will try every combination possible based on the specified password length and complexity. From an experiment done by the author, it took only 16 hours to determine that the administrator account in a Windows XP workstation was not given any password and less than two days to decipher the password which meets the 8-character and alpha-numeric convention (see Appendix Table 1-1).

Furthermore, most encryption programs nowadays use the Internet and network environment to transfer the encrypted data. This data is prone to sniffing attack by cryptanalysts and crackers. Once they are able to get a hand on the data, the job now is just to crack the encrypted data either using social engineering, brute force attack or other well-known breaking techniques.

2.2. Significant of the Project

The solution to the mentioned problems above is by doing research in encryption technology for password protection method/mechanism. This way, the password and the encrypted data will be more secured and harder to be deciphered. Users especially those who concern about the security of their data will be most relieved if this research is successful. The results can also be used in online environment should an extensive research is done to port the technology into networking environment.

2.3. Objective and Scope of Study

- To enhance the RSA encryption method by combining it with 'knocking' technology.
- To demonstrate the enhanced RSA encryption method in order to get the result whether the enhanced method is better than the current method or not.

CHAPTER 2

LITERATURE REVIEW

Encryption is the transformation of data into a form that is as close to impossible as possible to read without the appropriate knowledge (a key). Its purpose is to ensure privacy by keeping information hidden from anyone for whom it is not intended, even those who have access to the encrypted data(RSA Laboratories 10).

Cryptanalysis refers to the study of ciphers, ciphertext, or cryptosystems (that is, to secret code systems) with a view to finding weaknesses in them that will permit retrieval of the plaintext from the ciphertext, without necessarily knowing the key or the algorithm. This is known as breaking the cipher, ciphertext, or cryptosystem(searchSecurity, cryptanalysis).

One of the fundamental principles for protecting keys is the practice of split knowledge and dual control. The American National Standards (ANSI) X9. 17-1985, Financial Institution Key Management (www.ansi.org), defines split knowledge as: "A condition under which two or more parties separately have key components which, individually, convey no knowledge of the resultant cryptographic key. The resultant key exists only within secure equipment." Dual control is defined in the standard as: "A process of utilizing two or more separate entities (usually persons), operating in concert, to protect sensitive functions or information."(Breithaupt and Merkow 246)

RSA it still vulnerable to several types of attacks although it is dubbed the most secured encryption method. Most of these vulnerabilities are resulting from bad implementation of the encryption, user's carelessness brute force and so on. Bruce Schneier, CTO and founder of Counterpane Internet Security (www.counterpane.com), points out vast numbers of flaws that are introduced into the implementation of cryptography like attacks against cryptographic designs, attacks against implementations, attacks against trust models, attacks on the users, attacks against failure recovery and attacks against the cryptography(Breithaupt and Merkow 248-253).

From the experiment done by the writer (refer Appendices Table 1-1), a malicious user only needs the Windows Security Accounts Manager (SAM) file, a cracker program such as LC4 (formerly LophtCrack) and a dictionary file (usually comes with the cracker program). This statement is supported by a quotation from paragraph from The Complete Guide to Internet Security(qtd. in Breithaupt and Merkow 80) which said, "All that's needed is a copy of the UNIX/etc/passwd file or NT Server SAM file, a cracker program and a dictionary file".

Later on, the issue of brute-force attack was discussed and several samples of cracking programs were introduced, for example LopthCrack (later known as LC4 (@stake LC4)). Brute force attack is an attack that tries all possible combinations possible. It encrypts every entry in a dictionary or a number of dictionaries and then compares the results to the encrypted passwords that are stored. Any successful matches regarding the password file are recorded for later abuses(qtd. in Breithaupt and Merkow 81). Easy to say, brute force attack compares the encrypted password rather than breaking the encryption itself.

There is also an article discussing the issue of keystroke recording programs. Keystroke program is a program that launches on system startup and resides in the system's memory. The program will then capture all keys that are pressed by a user when he/she uses the computer and saves the keystrokes in a log file. Breithaupt and Merkow points out that , "When such a program is implanted on a victim's computer system, IDs and passwords can be easily collected and sent to a remote computer for illegal purposes, this saving the time to explore the network or crack password files."(qtd. in Breithaupt and Merkow 85)

Two-factor password system is a password system that takes two factors to identify the authenticated user. As for this dissertation, it tries to implement a two factor system that is not in the same technology as to enhance the current encryption method. Two-factor system examples are auto-teller machine (ATM), web-based email login (username and password) and vehicle registration system (ID card and driver's license). Alexander and Leonhard stressed out the importance of two-factor password system, "Of course, one-factor systems are not as secure as two-factor systems, and two-factor systems are not as secure as three-factor systems."(qtd. in Alexander and Leonhard 48)

As the dissertation looks into defeating techniques crackers used to crack/obtain passwords, it is important to understand the method used by them. One of the methods used by password crackers is by sniffing the network packets transmitted from client to server, for example. Once the packets are captured, these crackers can then analyze them or better, use an analyzer program and let it do all the hard work. Since most of the applications today use client-server architecture, it is common that users will log in at the client and be connected to the server located somewhere remote. If the client program is not well coded, passwords sent over the network can be easily captured as plain text. This makes it important to encrypt the password before it is transmitted over the network(qtd. in Alexander and Leonhard 53).

By using brute force or exhaustive attack, almost all the encrypted key could be broken. One just needs enough resources and time. There are practical limits to the key sizes, which can be cracked by brute-force searching, but since NSA deliberately limited the key size of DES to 56 bits, back in the 1970's when it was designed, DES can be cracked by brute force. Today's technology might not be able to crack other ciphers with 64-bit or 128-bit keys--or it might. Nobody will know until they have tried, and published the details for scientific scrutiny. Most such ciphers have very different internal structure than DES, and it may be possible to eliminate large numbers of possible keys by taking advantage of the structure of the cipher. Some senior cryptographers estimated what key sizes were needed for safety in a 1996 paper; they suggest that to protect against brute force cracking, today's keys should have a minimum of 75 bits, and to protect

information for twenty years, a minimum of 90 bits(Electronic Frontier Foundation and Gilmore).
Port-knocking is a method of establishing connection to a firewall-protected remote server. This is achieved by 'knocking' or trying to connect to the remote server via certain predefined port at a predefined intervals/series. The remote server will then check the firewall log whether the knocking is the same as the determined rule. If the series are indeed correct, another port will be opened and the client will then connect through this opened port. But if somehow the knocking is not the same as the predetermined in the rule, the server will remain silent and no open port will be detected. During the knocking process, the client will not know whether the server responded or not because the server will only check the firewall log and this is done secretly in the firewall program itself. After the client has finished working with the server, it will then generate another sequence of knocking to tell the server to close the connection and return back to the previous state(PORT KNOCKING, summary).

Port-knocking technology gives a stealthy method of authenticating a user of a networked system. This is because during the knocking process, no data is transferred between the client and the server. After the client is authenticated then only data is transferred. But in that case the server can initiate a secure channel to transfer the data such as using 128-bit SSL or SSH connection. The client will also not know whether the server is responding or not, making the server not really vulnerable to brute-force attack. But then although brute-force attack could be mounted to try and guess the port, it will be easily detected by the server through the log file generated by the firewall("Port Knocking", article).

However, there are also disadvantages of this technique. This is because the server will always change the sequence of correct port knocking to minimize the threat of brute-force attack. So, the client will need to generate a script to perform the knocking process. This script is vulnerable and should not be in the wrong hand. This will arise the problem of storing the knocking script and thus brings us back to the old situation where we store our passwords for our systems("Port Knocking", article).

Problem will also arise in the hardware part because the firewall will then have to allocate certain number of ports exclusively for the use of this process. This will disrupt programs that use computer ports to connect to other computers such as Microsoft SQL Server (443), Internet Information Services (80) and Telnet Server (23). The system administrators will have to configure either the knocking program or other programs that also use computer ports so that they will not conflict with each other("Port Knocking", article).

CHAPTER 3

METHODOLOGY

Custom Methodology (Combination and Variation of Different Methodologies)
Due to the limited time frame and extensive researches that need to be done, custom methodology is used to complete this dissertation. This methodology will take a combination and variation of the existing methodologies and apply it to suit the time frame condition and the huge amount of research needed.

The first part of this methodology will be the research and fact-finding process. This part will take the most time of this dissertation's time frame because the nature of this dissertation topic heavily relies on researches and fact-findings. Therefore, duration of two months will be allocated for this part to ensure the smoothness of the project flow.

After all the information and researches have been done, the development process will take place. The testing phase will take place concurrently with the development process to speed up the completion of this dissertation. Basically, in this stage the researches done are mainly in the part of how to integrate or use the codes or modules. Rapid Application Development (RAD) programming language will be used, in this case Microsoft Visual Basic.NET.

Upon the completion of the development process, thorough testing will be done. These involve module testing, module integration testing and multiple system testing. The modules developed in this dissertation's model will be tested individually and later on as integrated modules. The dissertation demo will also be tested in various Windows platforms such as Windows 95, Windows 98, Windows ME, Windows NT, Windows

2000 and Windows XP to ensure the compatibility with these platforms. Debugging tasks will still be continued in this stage to ensure that the prototype will not have major errors or bugs.

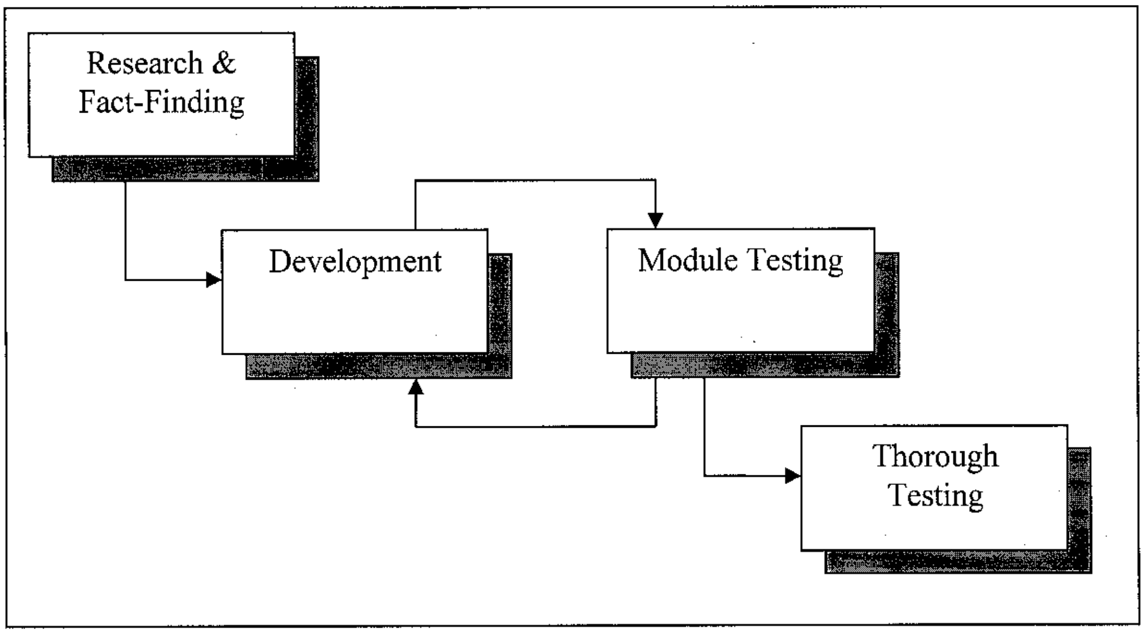


Figure 1-1: Methodology used in this dissertation

CHAPTER 4

RESULTS AND DISCUSSION

Currently, the prototype is using System.Security.Cryptography namespace to implement the RSA encryption algorithm. The prototype has been able to encrypt a text file into unreadable content. This prototype uses DES encryption, which is a subset of RSA method. There are several steps required in applying this encryption method, which are explained below.

First, we need to declare the namespace that we use for our encryption.

```
Imports System.Security.Cryptography
```

Then, generate the key and internal vector for the crypto service provider class.

```
Dim objDESCSP As New DESCryptoServiceProvider
objDESCSP.GenerateIV()
objDESCSP.GenerateKey()
```

After that, create an instance of the CryptoStream class using the cryptographic provider to obtain an encryption object (CreateEncrypter) and the existing output FileStream object.

```
Dim objCryptoStream As New CryptoStream(fsEncrypted, objDESCSP.CreateEncryptor(),
CryptoStreamMode.Write)
```

Lastly, we do the actual encryption process.

```
While totalBytesWritten < totalFileLength
        packageSize = fin.Read(storage, 0, 4096)
        crStream.Write(storage, 0, packageSize)
        totalBytesWritten = Convert.ToInt32(totalBytesWritten + packageSize /
        des.BlockSize * des.BlockSize)
End While
```

Currently, the prototype is able to encrypt and decrypt files using the DES encryption method. After this, client/server code will be implemented to enable the prototype to make use of the knocking method.

The server code uses TCPListener namespace in order to initiate a server program that will listen for any connection on a specified port. Meanwhile the client code uses TCPClient namespace to connect to the specified server. There will be 2 server codes being implemented in this product, where the first code is used to manage the knocking sequence from the client and the second code is used to accept data from the client if the knocking process is successful.

Basically the prototype has been able to initiate a secure connection between the client and server after the knocking process is establish. A simulated cryptanalyst had not been able to determine the time and workstations that involved in the process of transferring the encrypted data. Below are the achieved results from the testing done using the prototype.

Current Encryption Process	Enhanced Encryption Process
Cryptanalyst can guess which workstations are sending data.	Cryptanalyst has no clue whether the data transfer process has occurred or not.
The encrypted data is able to be sniffed and cracked using widely available tools (LC4, EtherDetect HTTP Sniffer).	The encrypted data is sent after the server is notified, meaning there is no open port indicating the server is waiting any data.
Data is transferred using HTTP protocol.	Data is transferred using SSH protocol.
Data transfer process is done openly without the protection of firewall to avoid conflict with the transferring process.	Server is protected behind firewall thus making it harder for cryptanalyst to run port scan on the server.

Table 1-1: Differences between current encryption process and enhanced encryption process.

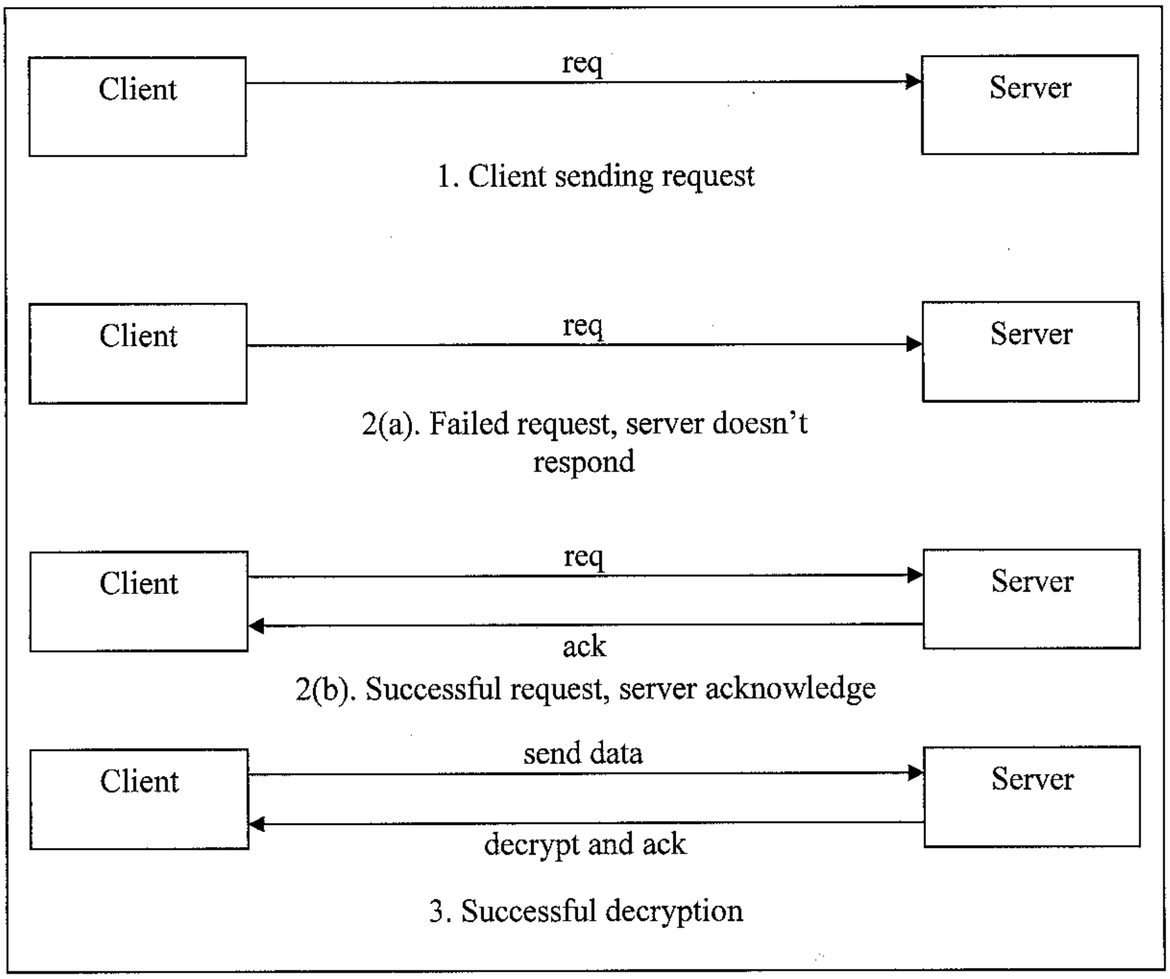

Client
req
Server
1. Client sending request
Client
req
Server
2(a). Failed request, server doesn't respond
Client
req
Server
ack
2(b). Successful request, server acknowledge
Client
send data
Server
decrypt and ack
3. Successful decryption

Figure 1-2: Workflow diagram on how the knocking process is done.

CHAPTER 5

CONCLUSION

The combination of 'port knocking' and RSA encryption algorithm has strengthen the encryption process by making it difficult for cryptanalyst to get their hands on the encrypted data. This is due to the fact that the knocking process makes it difficult for cryptanalyst to guess the time and network port that will be used to transfer the encrypted data. In addition, since the attacker doesn't know the algorithm used to encrypt the data it adds to the work that has to be done by the cryptanalyst before the sensitive information is being able to be cracked. This research therefore has been able to show that the encryption process is enhanced by the combination of two different techniques.

This research will show that it is important to have a secure and hard to crack passwords. For achieving the intended results, two parts are combined together which are strong encryption method and users' awareness on selecting hard-to-guess passwords. Cryptanalyst and cracker will find it harder to crack this type of encryption because they don't know when and which client transmits the data. Furthermore, after the server acknowledges the request a secure connection such as SSL or SSH can be initiated to further reduce risk of eavesdropping.

Due to the time constraints, the end product will be just a demo version of the research intended to show the audience of the functionality of the research. This is because the demo product will just show the audience how the technology works in encrypting a document or text and ensuring that the audience gets the understanding from the demo. However, basic functionalities and ideas of the research will be shown in the demonstration. For example, the connection between the client and server will be done on simple TCP connection while for production or commercial use, a more secure connection method such as SSL or SSH can be used.

In addition, a more secure encryption method can be implemented in this research. The use of public key technology should be fine, as the technique is just the same as the password used by this prototype. This will make the research to be more valuable as the users don't have to transfer their password using traditional ways that are prone to attacks by cryptanalyst or crackers.

Further enhancements can be added for market entry where companies can use this technology to commercialize their software. This will make the technology more secured and trusted by the users.

REFERENCES

"@stake LC 4 – The Award Winning Password Recovery and Auditing Tool". Feb 23, 2004. <http://www.atstake.com/products/lc>

"Windows NT SAM permission Vulnerability". Feb 23, 2004. <http://www.ciac.org/ciac/bulletins/h-45.shtml>

RSA Laboratories. RSA Laboratories' Frequently Asked Question about Today's Cryptography, Version 4.1. RSA Security Inc., 2000

"PORT KNOCKING – A system for stealthy authentication across closed port. : ABOUT : summary". Feb 25, 2004 <http://www.portknocking.org>

"cryptanalysis – a searchSecurity definition". Feb 25, 2004. <http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci214432,00.html>

Jim Breithaupt and Mark S. Merkow. The Complete Guide to Internet Security. Amacom Book Division, July 2000

Michael Alexander and Woody Leonhard. The Underground Guide to Computer Security. Addison-Wesley Pub Co., 1995

Electronic Frontier Foundation and John Gilmore. Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design, May 1998

"Port Knocking". Feb 28, 2004. <http://www.linuxjournal.com/article.php?sid=6811>

APPENDICES

Table 1-2

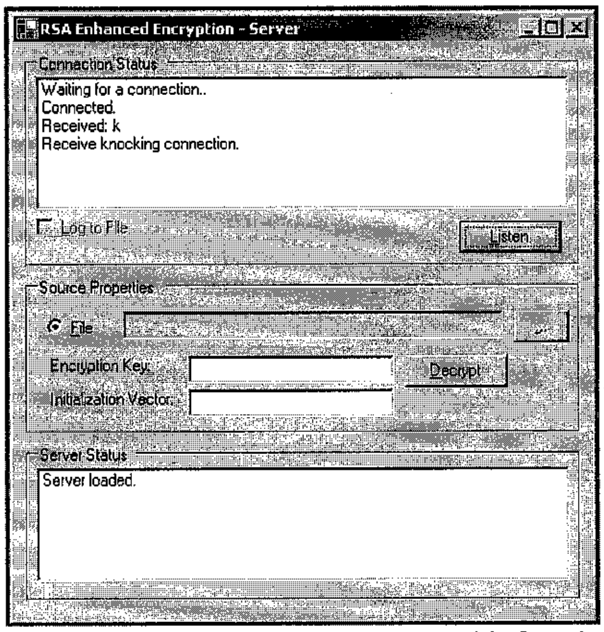
Password Length and Complexity	Time Taken to Decipher
0 character, simple variation	16 hours
8 characters, simple variation	26 hours
8 characters, complex variation	Less than 48 hours
>8 characters, simple variation	Less than 48 hours
>8 characters, complex variation	More than 48 hours

Note: All operations were done on a machine with 6 user accounts, Pentium 3 500 MHz processors, 256MB of Random Access Memory (RAM).

Table 1-2: Information on the password cracking process.

Appendix A: Prototype Screenshots

Figure 1-3

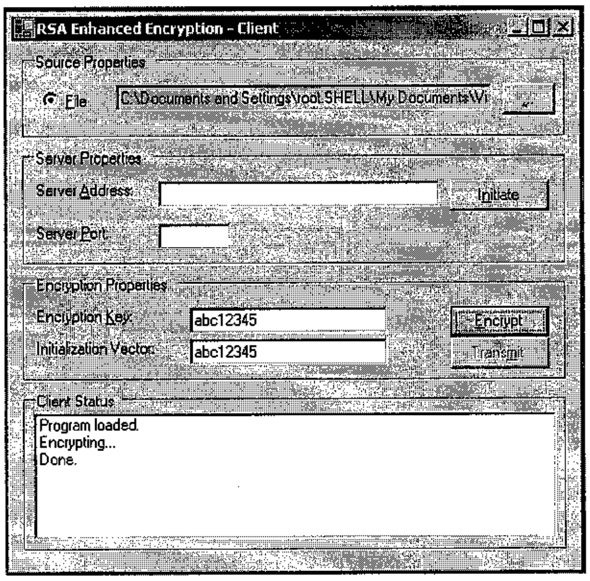


Waiting for a connection..
Connected.
Received: k
Receive knocking connection.

Server loaded.

Figure 1-3: Server acknowledges knocking sequence and informing client to send data

Figure 1-4



File: C:\Documents and Settings\root.SHELL\My Documents\Wi

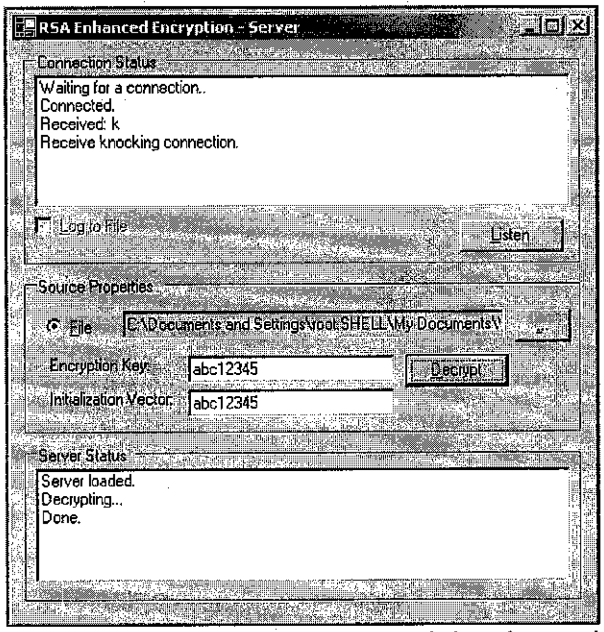
Encryption Key: abc12345

Initialization Vector: abc12345

Program loaded.
Encrypting...
Done.

Figure 1-4: Client encrypting sensitive data

Figure 1-5



```
Waiting for a connection..
Connected.
Received: k
Receive knocking connection.
```

File: C:\Documents and Settings\root.SHELL\My Documents\

Encryption Key: abc12345

Initialization Vector: abc12345

```
Server loaded.
Decrypting...
Done.
```

Figure 1-5: Server decrypting the encrypted data into original form

Appendix B: Coding for frmMain.vb (Client Code)

```
Imports System.Text
Imports System.IO
Imports System.Net
Imports System.IO.File
Imports System.Net.Sockets

Public Class frmMain
  Inherits System.Windows.Forms.Form
  Private _ClientSocket As Socket
  Private _HostSocket As Socket
  Private _TcpListener As New TcpListener(Dns.GetHostByName(Dns.GetHostName).AddressList(0), 5124)
  Private _IpEndPoint As New IPEndPoint(Dns.GetHostByName(Dns.GetHostName).AddressList(0), 5124)
  Private _Timer As New Timer

#Region " Windows Form Designer generated code "

  Public Sub New()
    MyBase.New()

    'This call is required by the Windows Form Designer.
    InitializeComponent()

    'Add any initialization after the InitializeComponent() call

  End Sub

  'Form overrides dispose to clean up the component list.
  Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
      If Not (components Is Nothing) Then
        components.Dispose()
      End If
    End If
    MyBase.Dispose(disposing)
  End Sub

  'Required by the Windows Form Designer
  Private components As System.ComponentModel.IContainer

  'NOTE: The following procedure is required by the Windows Form Designer
  'It can be modified using the Windows Form Designer.
  'Do not modify it using the code editor.
  Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
  Friend WithEvents optSourceF As System.Windows.Forms.RadioButton
  Friend WithEvents txtSourceF As System.Windows.Forms.TextBox
  Friend WithEvents cmdSourceF As System.Windows.Forms.Button
  Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
  Friend WithEvents GroupBox3 As System.Windows.Forms.GroupBox
  Friend WithEvents Label2 As System.Windows.Forms.Label
  Friend WithEvents txtSvrAddr As System.Windows.Forms.TextBox
  Friend WithEvents Label3 As System.Windows.Forms.Label
  Friend WithEvents txtSvrPort As System.Windows.Forms.TextBox
  Friend WithEvents OpenFileDialog1 As System.Windows.Forms.OpenFileDialog
  Friend WithEvents GroupBox4 As System.Windows.Forms.GroupBox
  Friend WithEvents cmdTransmit As System.Windows.Forms.Button
  Friend WithEvents cmdEncrypt As System.Windows.Forms.Button
  Friend WithEvents Label4 As System.Windows.Forms.Label
  Friend WithEvents txtEncKey As System.Windows.Forms.TextBox
  Friend WithEvents lstStatus As System.Windows.Forms.ListBox
  Friend WithEvents Label1 As System.Windows.Forms.Label
  Friend WithEvents txtEncIV As System.Windows.Forms.TextBox
  Friend WithEvents cmdInit As System.Windows.Forms.Button
  <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.GroupBox1 = New System.Windows.Forms.GroupBox
    Me.cmdSourceF = New System.Windows.Forms.Button
Me.txtSourceF = New System.Windows.Forms.TextBox
Me.optSourceF = New System.Windows.Forms.RadioButton
Me.GroupBox2 = New System.Windows.Forms.GroupBox
Me.lstStatus = New System.Windows.Forms.ListBox
Me.GroupBox3 = New System.Windows.Forms.GroupBox
Me.cmdInit = New System.Windows.Forms.Button
Me.txtSvrPort = New System.Windows.Forms.TextBox
Me.Label3 = New System.Windows.Forms.Label
Me.txtSvrAddr = New System.Windows.Forms.TextBox
Me.Label2 = New System.Windows.Forms.Label
Me.OpenFileDialog1 = New System.Windows.Forms.OpenFileDialog
Me.GroupBox4 = New System.Windows.Forms.GroupBox
Me.txtEncIV = New System.Windows.Forms.TextBox
Me.Label1 = New System.Windows.Forms.Label
Me.txtEncKey = New System.Windows.Forms.TextBox
Me.Label4 = New System.Windows.Forms.Label
Me.cmdTransmit = New System.Windows.Forms.Button
Me.cmdEncrypt = New System.Windows.Forms.Button
Me.GroupBox1.SuspendLayout()
Me.GroupBox2.SuspendLayout()
Me.GroupBox3.SuspendLayout()
Me.GroupBox4.SuspendLayout()
Me.SuspendLayout()
'
'GroupBox1
'
Me.GroupBox1.Controls.Add(Me.cmdSourceF)
Me.GroupBox1.Controls.Add(Me.txtSourceF)
Me.GroupBox1.Controls.Add(Me.optSourceF)
Me.GroupBox1.Location = New System.Drawing.Point(8, 8)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(416, 64)
Me.GroupBox1.TabIndex = 0
Me.GroupBox1.TabStop = False
Me.GroupBox1.Text = "Source Properties"
'
'cmdSourceF
'
Me.cmdSourceF.Location = New System.Drawing.Point(368, 24)
Me.cmdSourceF.Name = "cmdSourceF"
Me.cmdSourceF.Size = New System.Drawing.Size(40, 23)
Me.cmdSourceF.TabIndex = 4
Me.cmdSourceF.Text = "&..."
'
'txtSourceF
'
Me.txtSourceF.Cursor = System.Windows.Forms.Cursors.Default
Me.txtSourceF.Location = New System.Drawing.Point(72, 24)
Me.txtSourceF.Name = "txtSourceF"
Me.txtSourceF.ReadOnly = True
Me.txtSourceF.Size = New System.Drawing.Size(288, 20)
Me.txtSourceF.TabIndex = 3
Me.txtSourceF.Text = ""
'
'optSourceF
'
Me.optSourceF.Checked = True
Me.optSourceF.Location = New System.Drawing.Point(16, 24)
Me.optSourceF.Name = "optSourceF"
Me.optSourceF.Size = New System.Drawing.Size(56, 24)
Me.optSourceF.TabIndex = 1
Me.optSourceF.TabStop = True
Me.optSourceF.Text = "&File"
'
'GroupBox2
'
Me.GroupBox2.Controls.Add(Me.lstStatus)
Me.GroupBox2.Location = New System.Drawing.Point(8, 264)
Me.GroupBox2.Name = "GroupBox2"
Me.GroupBox2.Size = New System.Drawing.Size(416, 120)
Me.GroupBox2.TabIndex = 1
Me.GroupBox2.TabStop = False
Me.GroupBox2.Text = "Client Status"
'
'lstStatus
'
Me.lstStatus.HorizontalScrollbar = True
Me.lstStatus.Location = New System.Drawing.Point(8, 16)
Me.lstStatus.Name = "lstStatus"
Me.lstStatus.Size = New System.Drawing.Size(400, 95)
Me.lstStatus.TabIndex = 6
'
'GroupBox3
'
Me.GroupBox3.Controls.Add(Me.cmdInit)
Me.GroupBox3.Controls.Add(Me.txtSvrPort)
Me.GroupBox3.Controls.Add(Me.Label3)
Me.GroupBox3.Controls.Add(Me.txtSvrAddr)
Me.GroupBox3.Controls.Add(Me.Label2)
Me.GroupBox3.Location = New System.Drawing.Point(8, 80)
Me.GroupBox3.Name = "GroupBox3"
Me.GroupBox3.Size = New System.Drawing.Size(416, 88)
Me.GroupBox3.TabIndex = 7
Me.GroupBox3.TabStop = False
Me.GroupBox3.Text = "Server Properties"
'
'cmdInit
'
Me.cmdInit.Location = New System.Drawing.Point(328, 24)
Me.cmdInit.Name = "cmdInit"
Me.cmdInit.TabIndex = 4
Me.cmdInit.Text = "I&nitiate"
'
'txtSvrPort
'
Me.txtSvrPort.Location = New System.Drawing.Point(104, 56)
Me.txtSvrPort.Name = "txtSvrPort"
Me.txtSvrPort.Size = New System.Drawing.Size(56, 20)
Me.txtSvrPort.TabIndex = 3
Me.txtSvrPort.Text = ""
'
'Label3
'
Me.Label3.Location = New System.Drawing.Point(8, 56)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(88, 23)
Me.Label3.TabIndex = 2
Me.Label3.Text = "Server &Port:"
'
'txtSvrAddr
'
Me.txtSvrAddr.CausesValidation = False
Me.txtSvrAddr.Location = New System.Drawing.Point(104, 24)
Me.txtSvrAddr.Name = "txtSvrAddr"
Me.txtSvrAddr.Size = New System.Drawing.Size(216, 20)
Me.txtSvrAddr.TabIndex = 1
Me.txtSvrAddr.Text = ""
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(8, 24)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(88, 23)
Me.Label2.TabIndex = 0
Me.Label2.Text = "Server &Address:"
'
'GroupBox4
'
Me.GroupBox4.Controls.Add(Me.txtEncIV)
Me.GroupBox4.Controls.Add(Me.Label1)
Me.GroupBox4.Controls.Add(Me.txtEncKey)
Me.GroupBox4.Controls.Add(Me.Label4)
Me.GroupBox4.Controls.Add(Me.cmdTransmit)
Me.GroupBox4.Controls.Add(Me.cmdEncrypt)
Me.GroupBox4.Location = New System.Drawing.Point(8, 176)
Me.GroupBox4.Name = "GroupBox4"
Me.GroupBox4.Size = New System.Drawing.Size(416, 80)
Me.GroupBox4.TabIndex = 8
Me.GroupBox4.TabStop = False
Me.GroupBox4.Text = "Encryption Properties"
'
'txtEncIV
'
Me.txtEncIV.Location = New System.Drawing.Point(128, 48)
Me.txtEncIV.MaxLength = 8
Me.txtEncIV.Name = "txtEncIV"
Me.txtEncIV.Size = New System.Drawing.Size(152, 20)
Me.txtEncIV.TabIndex = 12
Me.txtEncIV.Text = ""
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(8, 48)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(104, 23)
Me.Label1.TabIndex = 11
Me.Label1.Text = "Initialization Vector:"
'
'txtEncKey
'
Me.txtEncKey.Location = New System.Drawing.Point(128, 24)
Me.txtEncKey.MaxLength = 8
Me.txtEncKey.Name = "txtEncKey"
Me.txtEncKey.Size = New System.Drawing.Size(152, 20)
Me.txtEncKey.TabIndex = 10
Me.txtEncKey.Text = ""
'
'Label4
'
Me.Label4.Location = New System.Drawing.Point(8, 24)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(100, 16)
Me.Label4.TabIndex = 9
Me.Label4.Text = "Encryption &Key:"
'
'cmdTransmit
'
Me.cmdTransmit.Enabled = False
Me.cmdTransmit.Location = New System.Drawing.Point(328, 48)
Me.cmdTransmit.Name = "cmdTransmit"
Me.cmdTransmit.TabIndex = 8
Me.cmdTransmit.Text = "Trans&mit"
'
'cmdEncrypt
'
Me.cmdEncrypt.Location = New System.Drawing.Point(328, 24)
Me.cmdEncrypt.Name = "cmdEncrypt"
Me.cmdEncrypt.TabIndex = 7
Me.cmdEncrypt.Text = "&Encrypt"
'
'frmMain
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.ClientSize = New System.Drawing.Size(432, 397)
Me.Controls.Add(Me.GroupBox4)
Me.Controls.Add(Me.GroupBox2)
Me.Controls.Add(Me.GroupBox1)
Me.Controls.Add(Me.GroupBox3)
Me.Name = "frmMain"
Me.Text = "RSA Enhanced Encryption - Client"
```

```
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox2.ResumeLayout(False)
        Me.GroupBox3.ResumeLayout(False)
        Me.GroupBox4.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub

#End Region

    Private Sub optSourceF_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
optSourceF.CheckedChanged
        EnableControl()
    End Sub

    Private Sub optSourceT_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
        EnableControl()
    End Sub

    Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        EnableControl()
        lstStatus.Items.Add("Program loaded.")
    End Sub

    Private Sub cmdSourceF_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSourceF.Click
        Dim objOFDialog As New OpenFileDialog
        objOFDialog.DefaultExt = "txt"
        objOFDialog.Filter = "Text File (*.txt) |*.txt"
        objOFDialog.InitialDirectory = Application.StartupPath
        objOFDialog.ShowDialog()
        txtSourceF.Text = objOFDialog.FileName
        objOFDialog.Dispose()
    End Sub

    Private Sub EnableControl()
        If optSourceF.Checked Then
            txtSourceF.Enabled = True
            cmdSourceF.Enabled = True
        Else
            txtSourceF.Enabled = False
            cmdSourceF.Enabled = False
        End If
    End Sub

    Private Function CheckEncButton() As Boolean
        Dim blnAllOK As Boolean = True
        If txtSourceF.Text = "" And optSourceF.Checked = True Then
            blnAllOK = False
        End If
        If txtEncKey.Text = "" Or txtEncIV.Text = "" Then
            blnAllOK = False
        End If
        If blnAllOK = False Then
            MsgBox("One or more required fields are empty. Please fill in all required fields and try again.", MsgBoxStyle.Exclamation +
MsgBoxStyle.OKOnly, "Error")
        End If
        CheckEncButton = blnAllOK
    End Function

    Private Sub cmdEncrypt_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdEncrypt.Click
        Dim blnOK As Boolean
        Dim strNewName As String = ChangeFilePath(txtSourceF.Text)
        blnOK = CheckEncButton()
        If blnOK Then
            If optSourceF.Checked = True And Not txtSourceF.Text = "" Then
                lstStatus.Items.Add("Encrypting...")
                modEncProc.EncryptF(txtSourceF.Text, strNewName, txtEncKey.Text, txtEncIV.Text)
                lstStatus.Items.Add("Done.")
            End If
        End If
    End Sub
    Private Function ChangeFilePath(ByVal strFileName As String) As String
        Dim strNewName As String
        If Strings.Right(strFileName, 4) = ".txt" Then
            strNewName = Replace(strFileName, ".txt", ".enc")
        ElseIf Strings.Right(strFileName, 4) = ".enc" Then
            strNewName = Replace(strFileName, ".enc", ".txt")
        End If
        Return strNewName
    End Function

    Private Sub txtSvrAddr_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
txtSvrAddr.TextChanged
        EnableTransmit()
    End Sub

    Public Sub EnableTransmit()
        If Not txtSvrAddr.Text = "" And Not txtSvrPort.Text = "" Then
            cmdTransmit.Enabled = True
        Else
            cmdTransmit.Enabled = False
        End If
    End Sub

    Private Sub txtSvrPort_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
txtSvrPort.TextChanged
        EnableTransmit()
    End Sub

    Private Sub cmdInit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdInit.Click
        ClientCode("localhost", "8002", "k")
    End Sub

    Private Sub ClientCode(ByVal server As [String], ByVal port As String, ByVal message As [String])
        Try
            Dim client As New TcpClient(server, port)
            Dim data As [Byte]() = System.Text.Encoding.ASCII.GetBytes(message)
            Dim stream As NetworkStream = client.GetStream()

            stream.Write(data, 0, data.Length)
            data = New [Byte](256) {}

            Dim responseData As [String] = [String].Empty
            client.Close()

        Catch e As ArgumentNullException
            lstStatus.Items.Add(e)
        Catch e As SocketException
            lstStatus.Items.Add(e)
        End Try
    End Sub

    Private Sub cmdTransmit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdTransmit.Click
        'Try
        '  Dim sr As StreamReader = New StreamReader("testfile.enc")
        '  Dim line As String
        '  Do
        '     line = sr.ReadLine()
        '     TextBox1.Text = TextBox1.Text & line
        '  Loop Until line Is Nothing
        '  sr.Close()
        'Catch ex As Exception
        '  Console.WriteLine("The file could not be read:")
        '  Console.WriteLine(ex.Message)
        'End Try

        'Const FILE_NAME As String = "Testfile.enc"
        'Dim fs As New FileStream(FILE_NAME, FileMode.Open)
        '' Create the reader for data.
        'fs = New FileStream(FILE_NAME, FileMode.Open, FileAccess.Read)
        'Dim r As New BinaryReader(fs)
        '' Read data from Test.data.
        'Dim i As Integer
        'For i = 0 To 10
        '    TextBox1.Text = TextBox1.Text & r.ReadInt32()
        'Next i

    End Sub
End Class
```

Appendix B: Coding for frmMain.vb (Server Code)

```
Imports System.IO
Imports System.Net
Imports System.Net.Sockets

Public Class frmMain
  Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

  Public Sub New()
    MyBase.New()

    'This call is required by the Windows Form Designer.
    InitializeComponent()

    'Add any initialization after the InitializeComponent() call

  End Sub

  'Form overrides dispose to clean up the component list.
  Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
      If Not (components Is Nothing) Then
        components.Dispose()
      End If
    End If
    MyBase.Dispose(disposing)
  End Sub

  'Required by the Windows Form Designer
  Private components As System.ComponentModel.IContainer

  'NOTE: The following procedure is required by the Windows Form Designer
  'It can be modified using the Windows Form Designer.
  'Do not modify it using the code editor.
  Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
  Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
  Friend WithEvents CheckBox1 As System.Windows.Forms.CheckBox
  Friend WithEvents GroupBox3 As System.Windows.Forms.GroupBox
  Friend WithEvents cmdSourceF As System.Windows.Forms.Button
  Friend WithEvents txtSourceF As System.Windows.Forms.TextBox
  Friend WithEvents optSourceF As System.Windows.Forms.RadioButton
  Friend WithEvents OpenFileDialog1 As System.Windows.Forms.OpenFileDialog
  Friend WithEvents lstConnStatus As System.Windows.Forms.ListBox
  Friend WithEvents lstServerStatus As System.Windows.Forms.ListBox
  Friend WithEvents cmdDecrypt As System.Windows.Forms.Button
  Friend WithEvents Label1 As System.Windows.Forms.Label
  Friend WithEvents txtEncKey As System.Windows.Forms.TextBox
  Friend WithEvents Label2 As System.Windows.Forms.Label
  Friend WithEvents txtEncIV As System.Windows.Forms.TextBox
  Friend WithEvents cmdListen As System.Windows.Forms.Button
  <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.GroupBox1 = New System.Windows.Forms.GroupBox
    Me.lstConnStatus = New System.Windows.Forms.ListBox
    Me.CheckBox1 = New System.Windows.Forms.CheckBox
    Me.GroupBox2 = New System.Windows.Forms.GroupBox
    Me.lstServerStatus = New System.Windows.Forms.ListBox
    Me.GroupBox3 = New System.Windows.Forms.GroupBox
    Me.txtEncIV = New System.Windows.Forms.TextBox
    Me.Label2 = New System.Windows.Forms.Label
    Me.txtEncKey = New System.Windows.Forms.TextBox
    Me.Label1 = New System.Windows.Forms.Label
    Me.cmdDecrypt = New System.Windows.Forms.Button
    Me.cmdSourceF = New System.Windows.Forms.Button
    Me.txtSourceF = New System.Windows.Forms.TextBox
Me.optSourceF = New System.Windows.Forms.RadioButton
Me.OpenFileDialog1 = New System.Windows.Forms.OpenFileDialog
Me.cmdListen = New System.Windows.Forms.Button
Me.GroupBox1.SuspendLayout()
Me.GroupBox2.SuspendLayout()
Me.GroupBox3.SuspendLayout()
Me.SuspendLayout()
'
'GroupBox1
'
Me.GroupBox1.Controls.Add(Me.lstConnStatus)
Me.GroupBox1.Controls.Add(Me.CheckBox1)
Me.GroupBox1.Controls.Add(Me.cmdListen)
Me.GroupBox1.Location = New System.Drawing.Point(8, 8)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(408, 152)
Me.GroupBox1.TabIndex = 0
Me.GroupBox1.TabStop = False
Me.GroupBox1.Text = "Connection Status"
'
'lstConnStatus
'
Me.lstConnStatus.HorizontalScrollbar = True
Me.lstConnStatus.Location = New System.Drawing.Point(8, 16)
Me.lstConnStatus.Name = "lstConnStatus"
Me.lstConnStatus.Size = New System.Drawing.Size(392, 95)
Me.lstConnStatus.TabIndex = 2
'
'CheckBox1
'
Me.CheckBox1.Enabled = False
Me.CheckBox1.Location = New System.Drawing.Point(8, 112)
Me.CheckBox1.Name = "CheckBox1"
Me.CheckBox1.TabIndex = 1
Me.CheckBox1.Text = "&Log to File"
'
'GroupBox2
'
Me.GroupBox2.Controls.Add(Me.lstServerStatus)
Me.GroupBox2.Location = New System.Drawing.Point(8, 288)
Me.GroupBox2.Name = "GroupBox2"
Me.GroupBox2.Size = New System.Drawing.Size(408, 112)
Me.GroupBox2.TabIndex = 1
Me.GroupBox2.TabStop = False
Me.GroupBox2.Text = "Server Status"
'
'lstServerStatus
'
Me.lstServerStatus.Location = New System.Drawing.Point(8, 16)
Me.lstServerStatus.Name = "lstServerStatus"
Me.lstServerStatus.Size = New System.Drawing.Size(392, 82)
Me.lstServerStatus.TabIndex = 0
'
'GroupBox3
'
Me.GroupBox3.Controls.Add(Me.txtEncIV)
Me.GroupBox3.Controls.Add(Me.Label2)
Me.GroupBox3.Controls.Add(Me.txtEncKey)
Me.GroupBox3.Controls.Add(Me.Label1)
Me.GroupBox3.Controls.Add(Me.cmdDecrypt)
Me.GroupBox3.Controls.Add(Me.cmdSourceF)
Me.GroupBox3.Controls.Add(Me.txtSourceF)
Me.GroupBox3.Controls.Add(Me.optSourceF)
Me.GroupBox3.Location = New System.Drawing.Point(8, 168)
Me.GroupBox3.Name = "GroupBox3"
Me.GroupBox3.Size = New System.Drawing.Size(408, 112)
Me.GroupBox3.TabIndex = 3
Me.GroupBox3.TabStop = False
Me.GroupBox3.Text = "Source Properties"
'
```

```
'txtEncIV
'
Me.txtEncIV.Location = New System.Drawing.Point(120, 80)
Me.txtEncIV.MaxLength = 8
Me.txtEncIV.Name = "txtEncIV"
Me.txtEncIV.Size = New System.Drawing.Size(152, 20)
Me.txtEncIV.TabIndex = 10
Me.txtEncIV.Text = ""
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(16, 80)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(104, 23)
Me.Label2.TabIndex = 9
Me.Label2.Text = "Initialization Vector:"
'
'txtEncKey
'
Me.txtEncKey.Location = New System.Drawing.Point(120, 56)
Me.txtEncKey.Name = "txtEncKey"
Me.txtEncKey.Size = New System.Drawing.Size(152, 20)
Me.txtEncKey.TabIndex = 8
Me.txtEncKey.Text = ""
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(16, 56)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(88, 23)
Me.Label1.TabIndex = 7
Me.Label1.Text = "Encryption Key:"
'
'cmdDecrypt
'
Me.cmdDecrypt.Location = New System.Drawing.Point(280, 56)
Me.cmdDecrypt.Name = "cmdDecrypt"
Me.cmdDecrypt.TabIndex = 6
Me.cmdDecrypt.Text = "&Decrypt"
'
'cmdSourceF
'
Me.cmdSourceF.Location = New System.Drawing.Point(360, 24)
Me.cmdSourceF.Name = "cmdSourceF"
Me.cmdSourceF.Size = New System.Drawing.Size(40, 23)
Me.cmdSourceF.TabIndex = 4
Me.cmdSourceF.Text = "&..."
'
'txtSourceF
'
Me.txtSourceF.Cursor = System.Windows.Forms.Cursors.Default
Me.txtSourceF.Location = New System.Drawing.Point(72, 24)
Me.txtSourceF.Name = "txtSourceF"
Me.txtSourceF.ReadOnly = True
Me.txtSourceF.Size = New System.Drawing.Size(280, 20)
Me.txtSourceF.TabIndex = 3
Me.txtSourceF.Text = ""
'
'optSourceF
'
Me.optSourceF.Checked = True
Me.optSourceF.Location = New System.Drawing.Point(16, 24)
Me.optSourceF.Name = "optSourceF"
Me.optSourceF.Size = New System.Drawing.Size(56, 24)
Me.optSourceF.TabIndex = 1
Me.optSourceF.TabStop = True
Me.optSourceF.Text = "&File"
'
'cmdListen
'
```

```
        Me.cmdListen.Location = New System.Drawing.Point(320, 120)
        Me.cmdListen.Name = "cmdListen"
        Me.cmdListen.TabIndex = 4
        Me.cmdListen.Text = "&Listen"
        '
        'frmMain
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(424, 413)
        Me.Controls.Add(Me.GroupBox3)
        Me.Controls.Add(Me.GroupBox2)
        Me.Controls.Add(Me.GroupBox1)
        Me.Name = "frmMain"
        Me.Text = "RSA Enhanced Encryption - Server"
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox2.ResumeLayout(False)
        Me.GroupBox3.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub

#End Region

    Private Sub cmdSourceF_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSourceF.Click
        Dim objOFDialog As New OpenFileDialog
        objOFDialog.DefaultExt = "txt"
        objOFDialog.Filter = "Encrypted File (*.enc) |*.enc"
        objOFDialog.InitialDirectory = Application.StartupPath
        objOFDialog.ShowDialog()
        txtSourceF.Text = objOFDialog.FileName
        objOFDialog.Dispose()
    End Sub

    Private Function CheckEncButton() As Boolean
        Dim blnAllOK As Boolean = True
        If txtSourceF.Text = "" And optSourceF.Checked = True Then
            blnAllOK = False
        End If
        If txtEncKey.Text = "" Or txtEncIV.Text = "" Then
            blnAllOK = False
        End If
        If blnAllOK = False Then
            MsgBox("One or more required fields are empty. Please fill in all required fields and try again.", MsgBoxStyle.Exclamation +
MsgBoxStyle.OKOnly, "Error")
        End If
        CheckEncButton = blnAllOK
    End Function

    Private Sub cmdDecrypt_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdDecrypt.Click
        Dim blnOK As Boolean
        blnOK = CheckEncButton()
        Dim strNewName As String = ChangeFilePath(txtSourceF.Text)
        If blnOK Then
            If optSourceF.Checked = True And Not txtSourceF.Text = "" Then
                lstServerStatus.Items.Add("Decrypting...")
                modEncProc.DecryptF(txtSourceF.Text, strNewName, txtEncKey.Text, txtEncIV.Text)
                lstServerStatus.Items.Add("Done.")
            End If
        End If
    End Sub

    Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        lstServerStatus.Items.Add("Server loaded.")
    End Sub

    Private Function ChangeFilePath(ByVal strFileName As String) As String
        Dim strNewName As String
        If Strings.Right(strFileName, 4) = ".txt" Then
            strNewName = Replace(strFileName, ".txt", ".enc")
        ElseIf Strings.Right(strFileName, 4) = ".enc" Then
```

```
            strNewName = Replace(strFileName, ".enc", ".txt")
        End If
        Return strNewName
    End Function

    Private Sub KnockCode()
        Try
            Dim port As Int32 = 8002
            Dim localAddr As IPAddress = IPAddress.Parse("127.0.0.1")
            Dim server As New TcpListener(localAddr, port)

            server.Start()

            Dim bytes(1024) As [Byte]
            Dim data As [String] = Nothing

            While True
                lstConnStatus.Items.Add("Waiting for a connection..")

                Dim client As TcpClient = server.AcceptTcpClient()
                lstConnStatus.Items.Add("Connected.")
                data = Nothing

                ' Get a stream object for reading and writing
                Dim stream As NetworkStream = client.GetStream()
                Dim i As Int32

                ' Loop to receive all the data sent by the client.
                i = stream.Read(bytes, 0, bytes.Length)
                While (i <> 0)
                    ' Translate data bytes to a ASCII string.
                    data = System.Text.Encoding.ASCII.GetString(bytes, 0, i)
                    lstConnStatus.Items.Add([String].Format("Received: {0}", data))
                    If data = "k" Then
                        client.Close()
                        lstConnStatus.Items.Add("Receive knocking connection.")
                        Exit Sub
                    Else
                        client.Close()
                        lstConnStatus.Items.Add("Failed connection on port 8002")
                        Exit Sub
                    End If

                    ' Process the data sent by the client.
                    data = data.ToUpper()

                    Dim msg As [Byte]() = System.Text.Encoding.ASCII.GetBytes(data)

                    ' Send back a response.
                    stream.Write(msg, 0, msg.Length)
                    lstConnStatus.Items.Add([String].Format("Sent: {0}", data))
                    i = stream.Read(bytes, 0, bytes.Length)
                End While

                ' Shutdown and end connection
                client.Close()
            End While
        Catch e As SocketException
            lstConnStatus.Items.Add(e)
        End Try
    End Sub

    Private Sub cmdListen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdListen.Click
        KnockCode()
    End Sub
End Class
```

Appendix B: Coding for modEncProc.vb

```
Imports System.IO
Imports System.Text
Imports System.Security.Cryptography

Module modEncProc

  Sub EncryptF(ByVal inName As String, ByVal outName As String, ByVal strKey As String, ByVal strIV As String)
    Try
      Dim storage(32786) As Byte        'Creates buffer
      Dim totalBytesWritten As Long = 32  'Keeps track of bytes written.
      Dim txtConvert As New ASCIIEncoding
      Dim toEnc() As Byte               'Key for encryption/decryption process
      Dim packageSize As Integer        'Specifies the number of bytes written at one time.

      'Declare the file streams.
      Dim fin As New FileStream(inName, FileMode.Open, FileAccess.Read)
      Dim fout As New FileStream(outName, FileMode.OpenOrCreate, FileAccess.Write)
      fout.SetLength(0)
      Dim totalFileLength As Long = fin.Length   'Specifies the size of the source file.

      'Create the Crypto object
      Dim des As New DESCryptoServiceProvider
      toEnc = txtConvert.GetBytes(strKey)       'Gets the byte value from the key
      des.Key() = toEnc                         'Sets the encryption/decryption key
      toEnc = txtConvert.GetBytes(strIV)        'Gets the byte value from the initialization vector
      des.IV() = toEnc                          'Sets the initialization vector

      Dim crStream As New CryptoStream(fout, des.CreateEncryptor(), CryptoStreamMode.Write)

      'Flow the streams
      While totalBytesWritten < totalFileLength
        packageSize = fin.Read(storage, 0, 32786)
        crStream.Write(storage, 0, packageSize)
        totalBytesWritten = Convert.ToInt32(totalBytesWritten + packageSize / des.BlockSize * des.BlockSize)
      End While
      crStream.Close()

    Catch e As Exception
      MsgBox(e.Message)
    End Try
  End Sub

  Sub DecryptF(ByVal inName As String, ByVal outName As String, ByVal strKey As String, ByVal strIV As String)
    Try
      Dim storage(32786) As Byte        'Creates buffer
      Dim totalBytesWritten As Long = 32  'Keeps track of bytes written.
      Dim txtConvert As New ASCIIEncoding
      Dim toEnc() As Byte               'Key for encryption/decryption process
      Dim packageSize As Integer        'Specifies the number of bytes written at one time.

      'Declare the file streams.
      Dim fin As New FileStream(inName, FileMode.Open, FileAccess.Read)
      Dim fout As New FileStream(outName, FileMode.OpenOrCreate, FileAccess.Write)
      fout.SetLength(0)
      Dim totalFileLength As Long = fin.Length   'Specifies the size of the source file.

      'Create the Crypto object
      Dim des As New DESCryptoServiceProvider
      toEnc = txtConvert.GetBytes(strKey)       'Gets the byte value from the key
      des.Key() = toEnc                         'Sets the encryption/decryption key
      toEnc = txtConvert.GetBytes(strIV)        'Gets the byte value from the initialization vector
      des.IV() = toEnc                          'Sets the initialization vector

      Dim crStream As New CryptoStream(fout, des.CreateDecryptor(), CryptoStreamMode.Write)
```

```
        'Flow the streams
        While totalBytesWritten < totalFileLength
          packageSize = fin.Read(storage, 0, 32786)
          crStream.Write(storage, 0, packageSize)
          totalBytesWritten = Convert.ToInt32(totalBytesWritten + packageSize / des.BlockSize * des.BlockSize)
        End While
        crStream.Close()

      Catch e As Exception
        MsgBox(e.Message)
      End Try
    End Sub

End Module
```