ABSTRACT

NON-REPUDIATION SECURE FILE TRANSFER PROTOCOL (NRSFTP)

by
Jerry Chen

Non Repudiation Secure File Transfer Protocol (NRSFTP) is designed to resolve three main concerns for today's electronic file transfer methodology. The three main concerns are Non-Repudiation, Secure, and Non-Real Time file transfer. Non-repudiation is to assure the receiver that the sender of the document is not an imposter. Secure document transfer is to assure the sender that only the intended receiver will be able to read the document. Non-real-time file transfer is to provide convenient and low cost transportability of the encrypted data from one party to another. With the above three concerns addressed, the NRSFTP protocol can be widely accepted by the general public as the method to securely transfer a file.

NON-REPUDIATION SECURE FILE TRANSFER PROTOCOL (NRSFTP)

by
Jerry Chen

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Department of Electrical and Computer Engineering

January 2007

APPROVAL PAGE

NON-REPUDIATION SECURE FILE TRANSFER PROTOCOL (NRSFTP)

Jerry Chen

Dr. Constantine N. Manikopoulos, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Robert Statica, Committee Member — Date
Program Director of Information Technology, NJIT

Dr. Yanchao Zhang, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Jerry Chen

Degree: Master of Science

Date: January 2007

Undergraduate and Graduate Education:

- Master of Science in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2007

- Bachelor of Science in Computer Science,
  Rutgers University, New Brunswick, NJ, 2001

- Bachelor of Science in Civil Engineering,
  Rutgers University, New Brunswick, NJ, 2001

Major: Computer Engineering

To my beloved family

This thesis is dedicated to my family who has supported me throughout the years.

ACKNOWLEDGMENT

I would like to sincerely thank my thesis advisor, Dr. Constantine N. Manikopoulos, for leading me into this exciting field and working with me to produce this fruitful outcome. This thesis and my degree could not have been completed without his enormous support, guidance, and insight. Special thanks to Dr. Robert Statica and Dr. Yanchao Zhang for serving on the committee.

I sincerely appreciate the help from Peter Manikopoulos for working with his father, Dr. Manikopoulos, to provide ideas and refine the NRS program. I also would like to thank Yuedong Wu for answering many different questions that I had about the NRS program using Java. In addition, special thanks to Calin Culianu and Zheng Zhang for providing insights to possibly rebuilding the NRS program using C++.

Lastly and most importantly, I would like to thank my parents, my brother, and my girlfriend. I am forever indebted to their support and compassion. Without them, I would not have gotten this far.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In today's business communications, transfer of documents is a must between two businesses. Both parties will have the need to send out files for the other to see it. Before the age of emails, documents were faxed, mailed, or exchanged in person to the intended recipient. These documents were not easily available for unintended parties to intercept. With the new electronic age of emails being exchanged across public internet, sensitive business documents are very vulnerable for unintended parties to intercept and manipulate.

Due to this fact of unintended parties being able to look at the sensitive business documents being transferred on the internet, two very important characteristic of sending a business document is jeopardized. One, documents should be securely delivered to the recipient without unintended parties being able to look at it. This is call encrypting the document where only the receiver will be able to see the document. Two, the recipient should have the comfort of knowing the document came from the intended sender without unintended parties being able to modify the contents of the document. This is called non-repudiation where the sender can not deny that the document was sent by someone else. Non-Repudiation Secure File Transfer Protocol (NRSFTP) is designed to cover the above two issues.

NRSFTP is also designed to cover a third issue, non-real-time transfer of files. Since transmitting a file attached to an email is considered to be non-real-time, the option of non-real-time file transfer will have to be considered for NRSFTP. Also,

real-time transfer of files requires the sender or the receiver to implement a server. This can be very costly to the individual who has to implement and maintain a server.

There are many different types of security protocols and software available but there were none developed to fit the exact needs for the NRSFTP ideology requirements. The only two protocols that would cover most of the requirements for the need of NRSFTP ideology are PGP and AS2.

PGP does not guarantee non-repudiation. Non-repudiation is one of the necessary requirements for the NRSFTP ideology. PGP fits many different functionality/characteristics for the implementation of NRSFTP. One of the PGP functionality that the NRSFTP will simulate is the ability to securely transfer a file through a non-real time communication medium. PGP is very fast since it uses both public and session key ideology. One can use it to communicate securely with intended recipients with no secure channels needed for prior exchange of keys. PGP has many useful features, and with the hybrid public and session key ideology, it is also very fast. Please refer to [5] for further details.

AS2 does guarantee non-repudiation. But AS2 is designed for real time transmission using Secure Socket Layer (SSL). Therefore, the receiver must maintain a server being up at all times for the sender to send a document to the receiver. In normal business situations, both parties will need to send documents. But AS2 is much more complex to implement and support.

CHAPTER 2

NRSFTP VISION

2.1 Non-Repudiation

When the recipient opens an encrypted document, the recipient should have the comfort of knowing the document came from the intended sender. NRSFTP uses digital signature to allow the recipient to verify the authenticity of the sender. And having a digital signature also allows the recipient to verify that the data was not altered while in transit.

Most file or transmission encryption protocols are not designed for non-repudiation. These protocols simply ensure secure transmission of data by encryption. This is the main reason for NRSFTP to address the issues of providing encrypted files to the receiver with non-repudiation.

2.2 Secure

NRSFTP will guarantee the secure exchange of documents between the sender and the receiver. By securely exchanging documents, both the sender and the receiver will have the comfort of knowing that unintended parties will not be able to view the document being transmitted.

2.3 Non-Real-Time File Transfer

When encrypting data, it is intended for a recipient to decrypt it in the future. Therefore, the encrypted file must be transmitted in some way to the recipient.

Currently most secure file transfer protocol is for real-time transfer of files. This means that either the sender or the receiver of the encrypted file will need to implement a server for the real time transfer. This is not desired for most people since the maintenance of a server is very costly.

With the above point in mind, non-real-time file transfer is a very important criterion for NRSFTP. Non-real-time transmission allows both parties exclude the need to implement a dedicated server. Since the transmission medium can be anything. It's up to the sender and the receiver to decide. Therefore, none-real-time transmission is very cost effective and flexible.

The possibility for non-real-time transmission is very cost effective for businesses. Since a dedicated server is not needed, below are some of the items that a company can take advantage of.

- No need to hire a highly technical personal to maintain a server.
- Save on the cost of buying a dedicated server for real-time transmission, data center to hold the server, and any additional costs to keep the server running at all times.
- Servers typically need a valid public internet IP address for clients to properly identify the location of the server. Public IP address is harder to obtain and it needs to be maintained also.

A major concern for none-real-time encryption and decryption of data is the validity of sender's public key from start to end. Let say that the sender initially digitally signs with the sender's private key, then two weeks later, the receiver decrypts the file. But during the two weeks from the sender initially encrypted the file to the receiver decrypting the file, the sender's digital certificate was revoked (possibly a hacker that got into the system). NRSFTP still addresses this concern.

During decryption, the sender's digital certificate is validated with the certificate authority. Therefore, if the sender revokes its public key with the certificate authority before the receiver decrypts the file, then the receiver will not be able to decrypt the file.

CHAPTER 3

EXAMPLE USES FOR NRSFTP

The real-estate industry can benefit from NRSFTP. Many different documents are exchanged between the buyer, seller, broker, buyer's attorney, seller's attorney, bank, appraiser, and etc. Some of these documents are the binder (agreement between the buyer and the seller's initial price for the estate), the contract (contract between the buyer and the seller's final price and other details for the estate), the financial documents (W-2, pay stubs, and bank statements about the buyer), the appraisal (from the appraiser estimating the price of the estate), the commitment letter (from the bank stating the buyer is qualified for the applied mortgage), and possibly other documents. Most of these documents contain sensitive information about the different parties involved.

Currently, this industry relies heavily on fax as the medium to exchange the documents. Fax, for the most part, is more secure than the exchange over public internet. But with emails gaining popularity, these sensitive documents can be attached within an email exchanged over the public internet more frequently in the future. Therefore, these documents should be securely exchanged between the different parties when transmitted over the public internet. Also, when the recipient opens the document, the recipient should have the comfort of knowing the document came from the intended sender.

Since all parties need the non-real-time functionality of NRSFTP, the NRSFTP client can accommodate the security needs for all parties involved.

NRSFTP even provides non-repudiation so the validity of the sender is guaranteed to the receiver.

For example, the buyer wants to send the bank its financial information document. Since email is the envisioned common transmission medium within the real-estate industry, the buyer can configure the bank's trading partnership to use email as the transmission medium. Once the file is encrypted, then the encrypted file is automatically attached to the email going to the bank's email address specified within the digital certificate. When the bank receives the email with the NRSFTP attachment, the bank can double click on the encrypted file and the NRSFTP client will appear asking the bank to save the decrypted file into a specific directory.

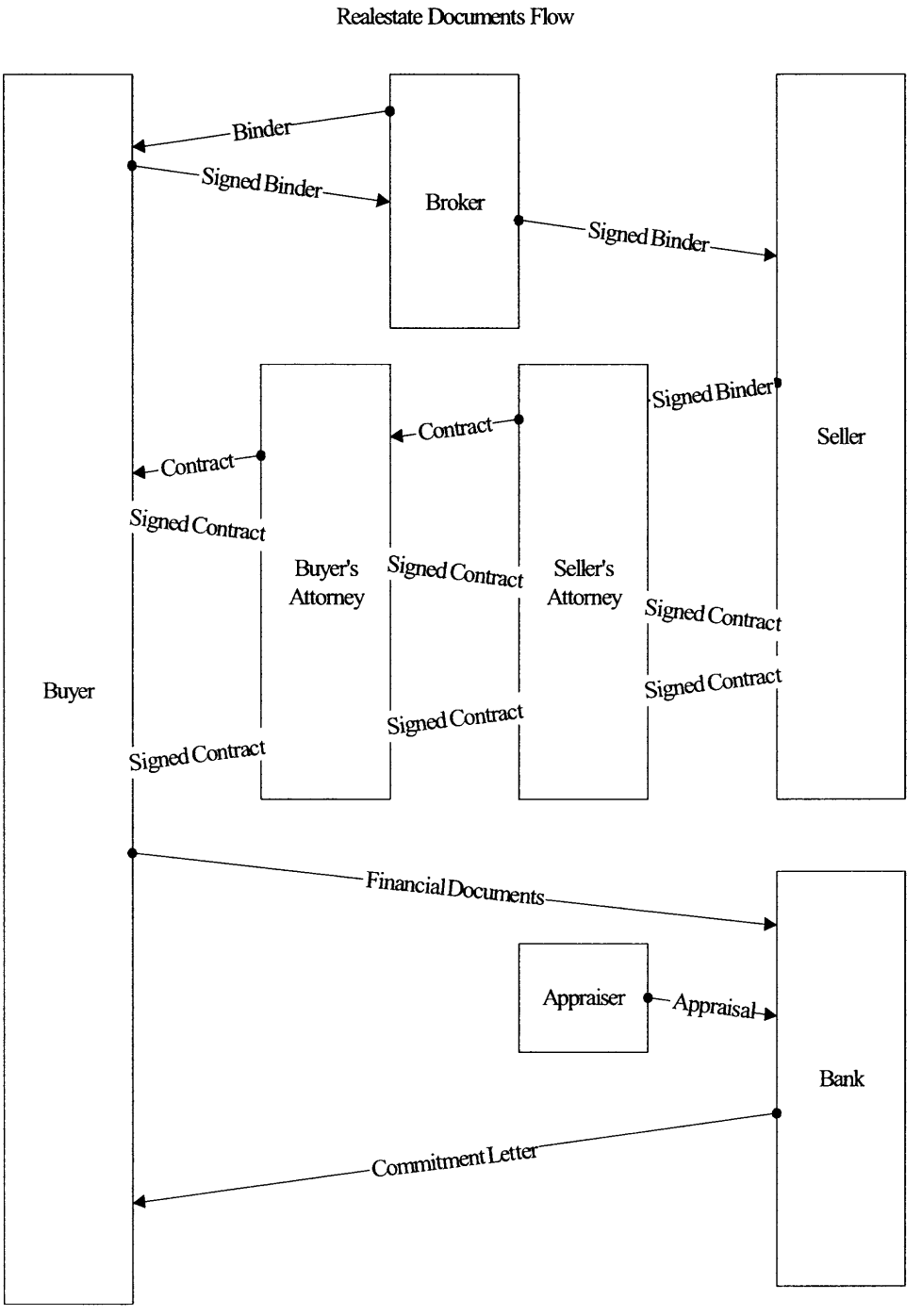

Figure 3.1 Different scenarios to exchange various types of documents from one organization to another within the real-estate industry.

CHAPTER 4

NRSFTP PROTOCOL

4.1 Protocol Objects

The below object/fields are the components for a NRSFTP file. Detailed infrastructure descriptions for some of the objects are within the next chapter.

- Version
- Document generation date
- Session key type
- Compressed object containing all files for transferring encrypted with the session key
- NRSFTP headers

A separate NRSFTP header is generated for each sender receiver pair and included within the NRSFTP file. The below object/fields are the components for a NRSFTP header.

- Sender's digital certificate
- Receiver's digital certificate
- Routing number
- Control number
- Message digest type
- Message digest encrypted with sender's private key
- Session key encrypted with receiver's public key

4.2 Encryption

NRSFTP will encrypt a file using the following steps:

1. Validate all senders' and recipients' digital certificates against the certificate revocation list (CRL) that was generated by the certificate authority.
2. Compress the files to be encrypted with ZIP to generate a zip object.
3. Generate a message digest from the zip object.
4. Encrypt (digitally sign) the message digest against the zipped object with sender's private key. A separate encrypted message digest is generated for each sender/receiver pair
5. Encrypt the compressed file using a random session key. A new random session key is generated for each new NRSFTP file.
6. Encrypt the session key with each receiver's public key.
7. Generate a header object for each sender/receiver pair which contains the sender's digital certificate, receiver's digital certificate, the routing number, the encryption date, control number, recipient's public key encrypted session key object, and the signed message digest object.
8. The receiver will receive the NRSFTP file which contains the session key encrypted compressed object and the list of header objects.

For further clarification of the encryption process, below is a descriptive example to encrypt three files to be sent to two TPs. First, validate all three pairs of sender/receiver digital certificates (total of six) against the CRL. Second, generate a zip object containing the three files. Third, generate a single message digest for the zip object. Fourth, two encrypted message digests are generated using each of the sender's private key. Fifth, a single session key encrypted zip object is generated using a new random session key. Sixth, two encrypted session key objects are generated by encrypting with each receiver's public key. Seventh, two header objects

are created. Lastly, the NRSFTP file will contain the two header objects and the single session key encrypted zip object.

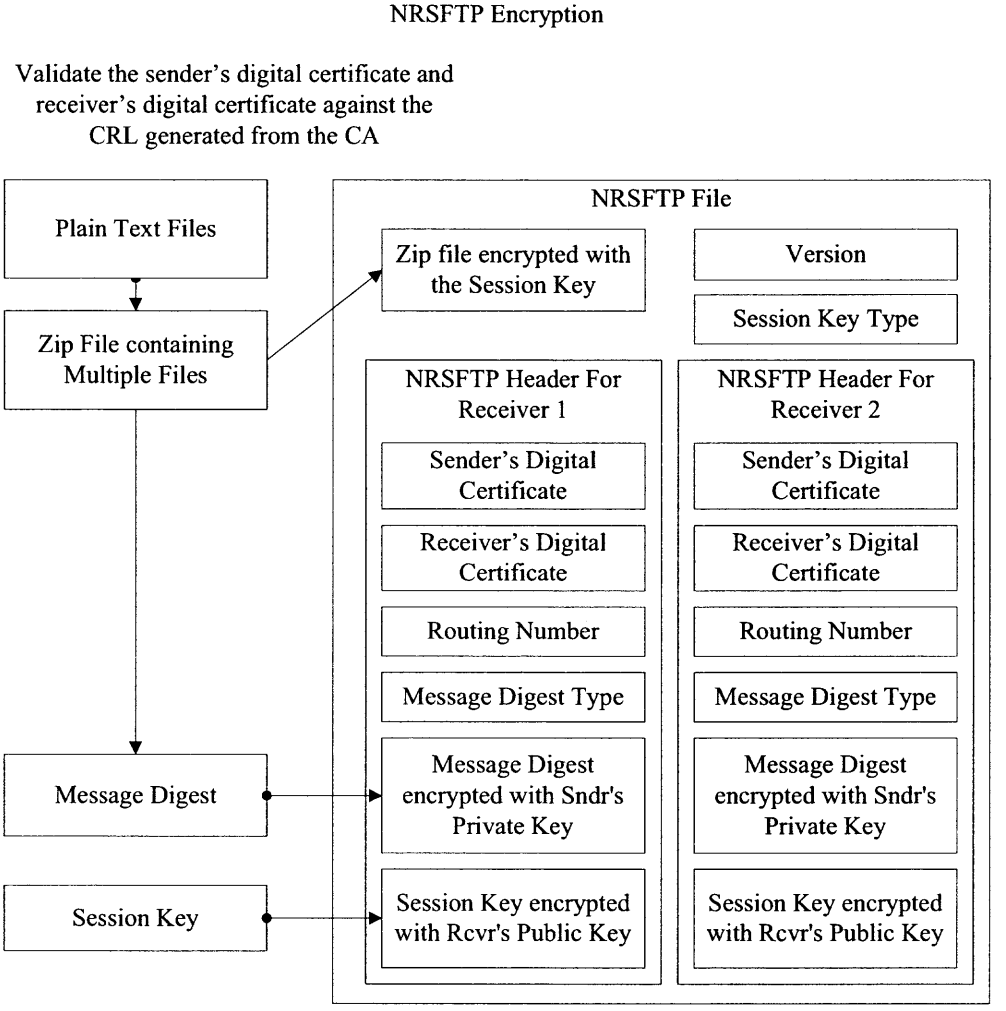

Figure 4.1 Steps to encrypt a document.

4.3 Decryption

NRSFTP will decrypt a file using the following steps:

1. First identify the sender's digital certificate and receiver's digital certificate to see if it's configured within its system or software.

2. Validate the specific pair of sender and receiver's digital certificates against the certificate revocation list (CRL) generated by the certificate authority.

3. The receiver then uses its private key to decrypt and obtain the session key.

4. Then use the session key to decrypt and obtain the compressed file.

5. Run a hash algorithm on the compressed file to obtain the message digest.

6. Decrypt the sender's signed message digest object by using the sender's public key.

7. Compare the sender's message digest with the message digest that was generated from the compressed file.

8. If the message digests matches, then uncompress the data to get the files.

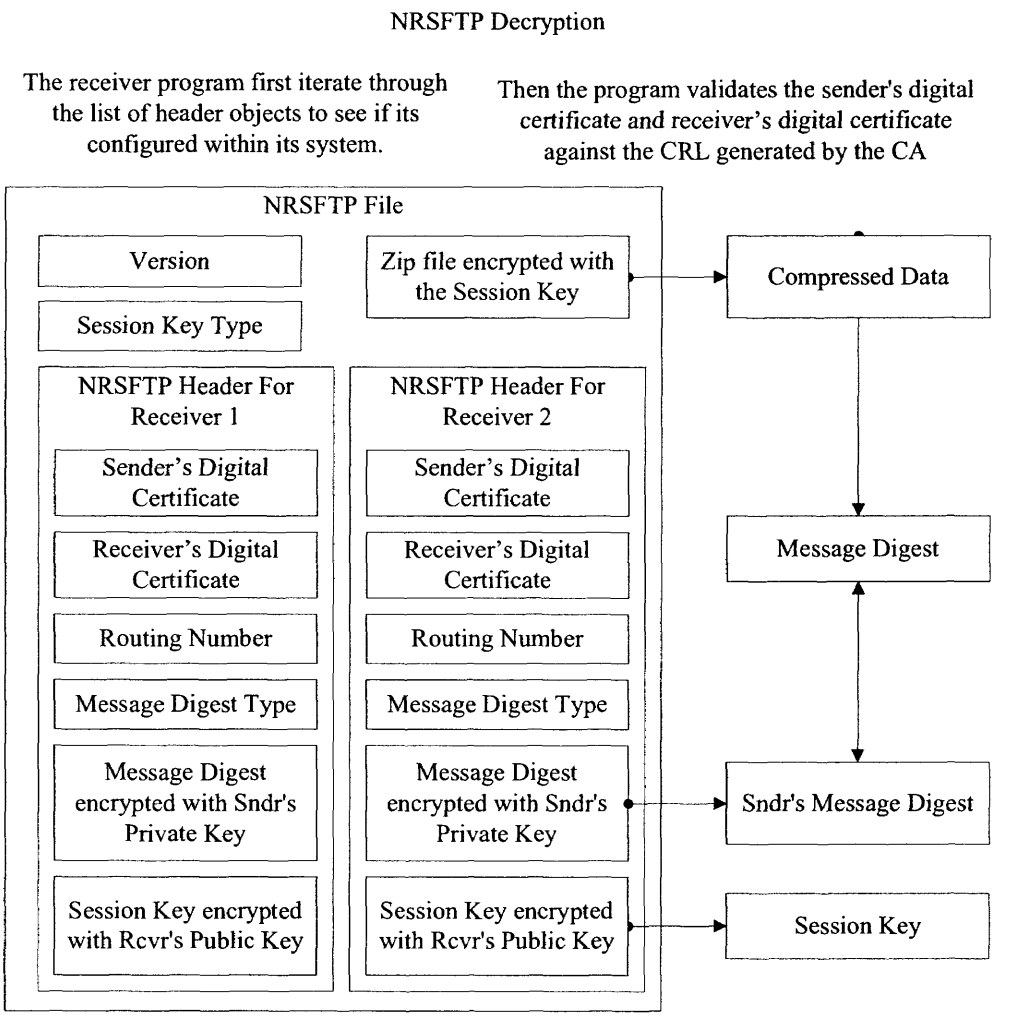

Figure 4.2 Steps to decrypt a document.

CHAPTER 5

NRSFTP INFRASTRUCTURE

5.1 Digital Certificate

NRSFTP will use digital certificates using the X.509 standards since it is the industry standard and most commonly used. X.509 digital certificate is also a part of OATH's approach to standardize the encryption industry [4].

Digital certificate is the core building block for NRSFTP. The public key technology associated with digital certificates is very powerful and useful in the security world. Below are the different functions that NRSFTP utilizes digital certificates for.

- When encrypting a file with NRSFTP, the digital certificate's public key from the receiver is used to securely exchange the session key. The public key is not used to encrypt the data. The public key is actually used to encrypt the session key. The session key is actually the one that encrypts the data.

- When encrypting a file with NRSFTP, digital certificate's public key from the sender is used to confirm for non-repudiation.

- Both the digital certificate for the sender and the receiver are within the NRSFTP encrypted file (unencrypted header object portion). This is done so the receiver's system can recognize the proper configuration to decrypt and process the file.

- By allowing the receiver's digital certificates within the header object, an intermediate party who represents multiple senders and receivers to transmit documents can identify the proper receiver of the file.

- The X.509 digital certificate can validated with a certificate authority to certify the validity of the digital certificate.

- Since the digital certificate is advertised publicly, the digital certificate does not contain the private key. The private key must be kept secret at all times

to ensure the non-repudiation integrity. Please refer to the possible attacks chapter for additional information.

5.2 Session Key

The public and private keys could be used to encrypt data. However the algorithm requires significantly more computational operations than a session key (also called a symmetric key).

With the above deficiency on public and private key cryptography in mind, a symmetric key can be used along with the public key and private key infrastructure. The symmetric key can be encrypted by the sender with the receiver's public key. Then the encrypted symmetric key is sent back to the receiver with the encrypted data, which the receiver uses its private key to decrypt and obtain the symmetric key. Thus the symmetric key can decrypt the actual data.

The symmetric key in NRSFTP is a one-time-only key. It is created for each new set of data to be encrypted.

5.3 Compression

Another nice function that NRSFTP performs is to compress the data with ZIP before encrypting. This additional function reduces the data set by about 50%. Therefore, although encrypting the file will increase the file size, the file being sent out by NRSFTP most likely to be smaller than the original file.

Also, the most useful purpose for compressing the data is to strengthen the cryptographic security. Most cryptanalysis techniques search for patterns found in

the data to crack the key. Since compressing the data decrease the patterns in the data to be encrypted, therefore, it will be harder for cryptanalysis to crack the key.

5.4 Message Digest

A message digest is used as a building block to implement non-repudiation. NRSFTP uses a hash algorithm to generate the message digest, and the message digest is encrypted with the sender's private key. Since no one else has the sender's private key, the encrypted file is a signature from the sender.

As long as a secure hash function is used, there is no way to take someone's signature from one document and attach it to another, or to alter a signed message in any way. The slightest change to a signed document will cause the digital signature verification process to fail.

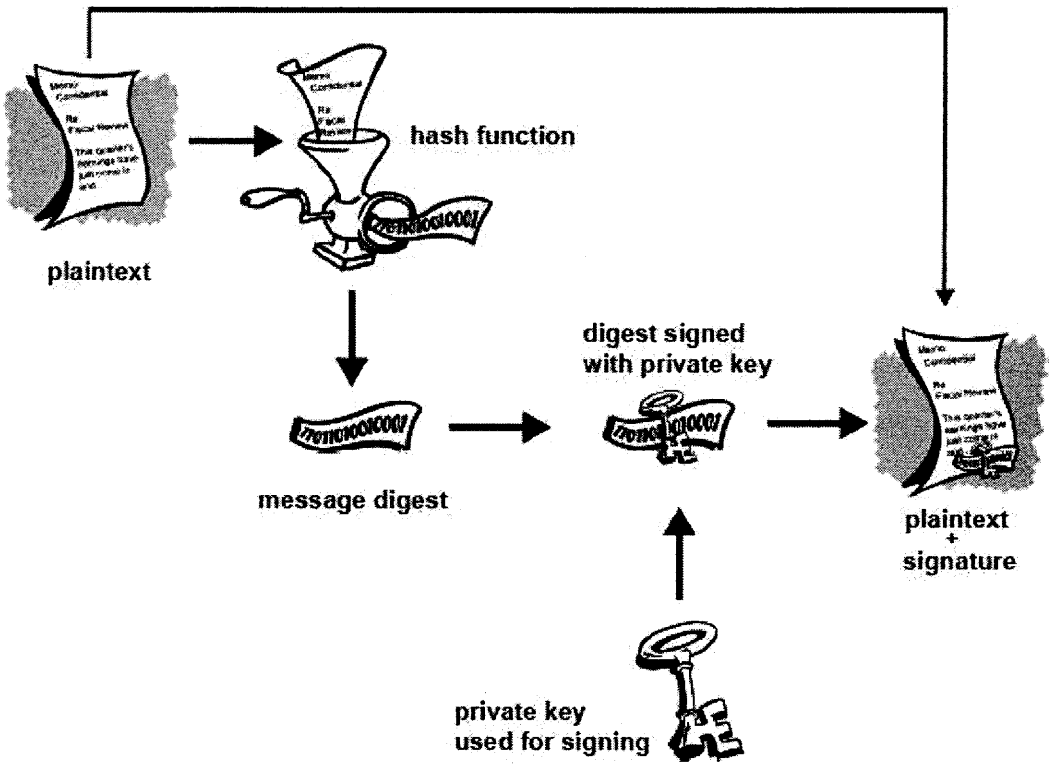

Figure 4.3 How message digest works from [9].

5.5 Routing Number

The purpose of the routing number is an optional identifier that the sender and receiver can agree to utilize this field. By using the routing number, the receiver can easily identify the type of document is within the NRSFTP file without going through the encryption process. The routing number can also allow automated post processing on the decrypted files for the receiver.

5.6 Control Number

The control number can be optionally utilized for many reasons. One, the sender and the receiver can use this number as the common reference to identify a specific NRSFTP file. Two, document acknowledgement can be sent from the receiver back to the sender stating a specific control numbered NRSFTP file was received. Three, the control number can be used for the receiver to check for duplicates and not process the same NRSFTP document twice. Four, the receiver can check the sequence of the control numbers to ensure documents are processed consecutively without documents missing due to email lost and etc.

CHAPTER 6

POSSIBLE ATTACKS

6.1 Securing the Private Key

The security of NRSFTP relies heavily upon the security of the private key. If the private key is obtained by the wrong person, the potential for exploitation is high. And with "non-repudiation", it is very difficult to deny any messages that may have been created fraudulently. This is a key limitation of public key cryptography from NRSFTP technology over handwritten signature. Unlike the human hand, which can not be compromised, the private key can be stolen even without the owner being aware of it being compromised.

6.2 Digital Certificate Authentication

The key issue that could potentially limit the practical usefulness of digital certificate is that the information to create the digital certificate may have been falsely inputted. This is the classic false identification attack. For example, one may register a digital certificate with a certificate authority and circulate the corresponding public key as belonging to someone else. If another person relies upon such a false digital certificate's public key, communications may actually be happening with an electronic impersonator and not the real person. This can seriously jeopardize people's trust on digital certificates. Therefore, the certificate authority has to ensure the user's identity is valid.

6.3 False Document Flooding

A hacker can snoop on the traffic to and from the sender and receiver. Since the header object contains information in plaintext format, the hacker can obtain both the sender and receiver's digital certificate information. Then the hacker can possibly use the same header object and encrypted digital digest object from the original NRSFTP file. Also, the hacker can generate false encrypted session key object and the encrypted compressed data object. With the four objects, the hacker can then generate a NRSFTP file that appears to be valid. The hacker can then send the false file to the receiver for processing. The false file will be processed by the receiver executing every step until the very last step (checking for differences in the message digests) and the decryption process will fail. But the steps taken to determine that this file is an invalid file are very costly. A lot of processing time is wasted on decrypting for the session key, compressed data, message digest, validating the sender's digital certificate, and uncompressing the data. If the hacker sends multiples like hundreds or even thousands of these documents to the recipient, then the recipient's system will be flooded and appears to be down.

CHAPTER 7

NRSFTP CLIENT SOFTWARE

7.1 Software Description and Structure

The NRSFTP Client consists of multiple programs with different functionalities. The many different programs are building blocks for the entire program and processing. These building blocks fall under different categories like applets for user friendly GUI screens (Graphical User Interface), actual programs to perform specific functionalities, and programming objects that other programs can easily work with and save implementation time.

The NRSFTP client follows through a series of programs to achieve a user friendly encryption or decryption environment. To start, The NRSFTP client starts with the NrsClient program initiating the NrsClient GUI applet through the NrsClientFrame program. The NrsClientFrame program allows the NrsClient GUI applet to display as a normal application within the Windows environment. The NrsClient is the main console that allows the users to initiate other specific NRSFTP encryption functionalities.

Before encrypting or decrypting a file, three different main components need to be identified, the TP's (Trading Partner) digital certificate, your digital certificate, and your PKCS12 file that holds the private key to your digital certificate. These components plus other useful information is saved and constructs a TP configuration file for simplified encryption or decryption process. The NrsClientTpEdit is the GUI applet that organizes and works with these TP configurations files within the

$NRS_ROOT/Tp directory. The NrsClientTpEdit GUI applet is initiated from the NrsClient main console. The NrsClientTpEdit displays the available TP's for the user to use. From the NrsClientTpEdit applet, the user can have the option to create a new TP configuration or edit an existing TP configuration. Both options are handled by the NrsClientTpEdit GUI applet.

Since the TP configuration files are created, the decryption or encryption process is simplified. The NrsClientCrypt is the GUI applet that allows the user to encrypt or decrypt a file. The actual file encryption or decryption transformation processing is done by the NrsCrypt program, which is initiated by the NrsClientCrypt.

NrsFile and NrsTp are very important programming objects to the above overall structure. The NrsFile has multiple fields and parts just like the designed NRSFTP encrypted file. Many of the programs and applets can referred to the NrsFile object and easily utilize the different pieces of information. It'll be a nightmare for the programs and applets to tract these different pieces of information related to a NRSFTP encrypted file. The same concept is also true for the NrsTp object. Also, both of these objects have functionalities related to the object it self. This makes the programs and applets that utilize these objects easy to initiate these functionalities.

The NrsLog is a universal functionality for all of the NRSFTP programs and applets to generate error messages into the Temp directory. The NrsLog is the centralized location to modify the look and feel of how the log files will look like.

The below sections provides more details for each of the components that makes up the NRSFTP Client program.

7.2 NrsClient

The NrsClient program is just a few lines of simple codes to initiate the NrsClientMain GUI applet through the NrsClientFrame program. This is how the NrsClientMain GUI gets started.

7.3 NrsClientDecrypt

NrsClientDecrypt is a GUI applet that allows the users to have an easy to use interface to decrypt a file. The different useable fields within this GUI are very self explanatory. Since the decryption process can only decrypt a single input file at a time and it finds the TP configuration automatically, only the input and output file to be used is needed. After executing with all the above parameters entered, and if the encryption or decryption process did not fail, then various information about the NRS file is displayed within the pop-up panel.

7.4 NrsClientEncrypt

NrsClientEncrypt is a GUI applet that allows the users to have an easy to use interface to encrypt a file. The different useable fields within this GUI are very self explanatory. The user first chooses to encrypt or to decrypt a file. Then choose the TP configuration file that holds the necessary information to perform the data transformation. Then select the input and output file to be used. After executing with all the above parameters entered, and if the encryption process did not fail, then various information about the NRS file is displayed within the pop-up panel.

7.5 NrsClientFrame

Since the NrsClientMain GUI is an applet, it can only be displayed within a web browser. One issue that was encountered with an applet is that tighter security is placed on the applet program and file access to the local pc is denied.

This NrsClientFrame program allows an applet to be executed as an application. An application can have normal access to files and looks like normal windows programming. This omits the need to change the Java security policy on the user's local pc. Changing the Java security policy for applets to have file access rights can be a dangerous thing where other illicit program can take advantage of.

7.6 NrsClientMain

The NrsClientMain is the main GUI (Graphical User Interface) for the NRSFTP client. It is just a main panel to lunch other applet like NrsClientTpEdit and NrsClientCrypt. The NrsClient extends from the Java applet class. Therefore the NrsClient is designed to start from a web browser. This functionality is designed so implementation using a web browser remotely is possible. A very important function is the automated update of CRL (certificate revocation list) files over the internet. The CRL file is automatically updated when the NrsClient applet is started. The CRL file validates the digital certificates to ensure the digital certificate is not revoked. Therefore, the CRL file is updated here since users will have to enter into this screen before using the other GUI screen functionalities.

7.7 NrsClientSerialCode

This is the serial code input applet panel that allows the users to input the serial code.

7.8 NrsClientTpEdit

The NrsClientTpEdit program is a GUI applet that allows the users to have an easy to use interface to create a new TP configuration or edit an existing TP configuration file. Creating a new TP and editing a TP were two separate GUI applets. But since both GUI applets functions similarly, this NrsClientTpEdit was modified to handle both functionalities. One can create new or edit the TP configuration file by using the TP's digital certificate file, your own digital certificate file, and your PKCS12 private key file. This method is typically used for a sender creating a TP to encrypt and send out a file. The different useable field within this GUI is fairly self explanatory. The user first chooses to create a TP by using the actual digital certificate files. Then the user chooses the TP code to be created. The PKCS12 file that contains the private key also has to be selected. After executing with all the above parameters entered, and if the digital certificate are not invalid or revoked, then various information about the TP configuration is displayed within the

7.9 NrsCrypt

NrsCrypt is the most important program for NRSFTP. It contains the core functionalities for the actual file encryption and decryption process. Before encryption or decryption, both the sender and the receiver's digital certificates are validated to make sure they're valid. Doesn't make sense to try and process data if

the digital certificates aren't valid. For encryption, this program first generates the message digest for the receiver to compare with. Then this program encrypts the message digest with the sender's private key to generate a digital signature stating that the message was from the intended sender. The actual data is then compressed (to reduce file size and make the encryption stronger) and encrypted with a random symmetric key. The symmetric key is then encrypted with the recipient's public key so only the intended recipient can decrypt and have access to the data. For decryption, the recipient first uses its private key to decrypt and obtain the symmetric key. Then the symmetric key is used to decrypt and obtain the compressed data. The compressed data is then uncompressed and the original data is obtained. Before the original data is saved into the destination file, the original message digest is decrypted using the sender's public key. Then a new message digest is created from the original data. The two message digests are compared to see if they're the same. Only if they are the same then the file can be created. This program is also designed for users to execute and encrypt or decrypt within a command line environment. This can be very helpful since large corporation will need to encrypt and decrypt at the command line for automated processing.

7.10 NrsFile

NrsFile functions like an object which contains the fields for the NRSFTP protocol. This is the core object that programs will use when referring to the NRSFTP encrypted file. The layout of the fields for this object is exactly the same as the documentation explaining about the NRSFTP protocol.

The NrsFile object contains multiple NrsFileHeader objects and the symmetric key encrypted data. More detailed explanations about the headers are within the NrsFileHeader section.

7.11 NrsFileHeader

The NrsFileHeader is designed to accommodate the capability to encrypt a single NRSFTP file for multiple receivers. Each header object contains the necessary information for the individual trading partner NRSFTP encryption/decryption. The actual payload data is still within the NrsFile object. Therefore, the multiple headers will not contribute to a large file size. Since the sender may want to choose different digital certificates to the various receivers, this is why both the sender's and the receiver's information is contained within the header.

7.12 NrsLog

NrsLog is designed to be a universal log generation utility for other NRSFTP programs. The main reason for creating NrsLog is due to the fact that when executing Java programs through a jar or executable, there are no command windows to capture the error messages. Therefore, I created this universal program to generate all error messages into log files within the temp directory. Having this universal log mechanism will save lots of development time since all error messages and log functionalities are centralized within one function. Each of the programs creates an NrsLog object at the beginning. When error messages generate, it is written into the Temp/programName.log file.

7.13 NrsSerialCode

This object is to implement a serial code. The serial code is needed to encrypt a file, but a serial code is not needed when decrypting a file. With this in place, the owner of the NRSFTP software can provide the end users with free trial versions of this software for decryption purposes. This will help in advertising the software to others.

7.14 NrsTp

NrsTp is a very important object that the rest of the user interfaces and functionalities depend on. This is the core object that programs will be used to properly identify the certificates, the type of encryption encoding, and other properties when encrypting or decrypting a file. Think of the NrsTp object as the object for effortless encryption and decryption. Other than just the configuration properties for file encryption during encryption or decryption, the NrsTp object also handles the loading and saving the configuration properties into a file format. The most important function for NrsTp is the digital certificate validation function. The digital certificate validation function authenticates the digital certificate with the downloaded CRL (Certificate Revocation List) file to see if the digital certificate was revoked. This function is currently used at three different places. One, this function is used before the NrsTp is created. If either the sender's digital certificate or the receiver's digital certificates are revoked, it's doesn't make sense to generate a TP with revoked certificates. Two and three, this function is used before both encrypting and decrypting a NRSFTP file.

CHAPTER 8

TESTING AND VALIDATION

8.1 Secure

The NRSFTP encrypted file was tested to ensure the contents of the file was secure and not viewable by anyone else. Various methods were performed to check if the file was readable by other programs like text editors, Microsoft Office programs, document image readers, zip/compression software, and etc. For sample screen shots of the above tests performed, please refer to Figures 8.1 to 8.5. Since the file is compressed and encrypted, there were no visible patterns or readable characters within the encrypted file. Please note that there are readable characters within the NRSFTP file, but the characters are related to the NRSFTP file header information which includes the digital certificate information. For further validation, the content of the encrypted NRSFTP file was displayed in front of other users many times previously and no one was able to point out any patterns or possible flaws in the encrypted file.

Figure 8.1 Contents of a NRSFTP encrypted file using a text editor.

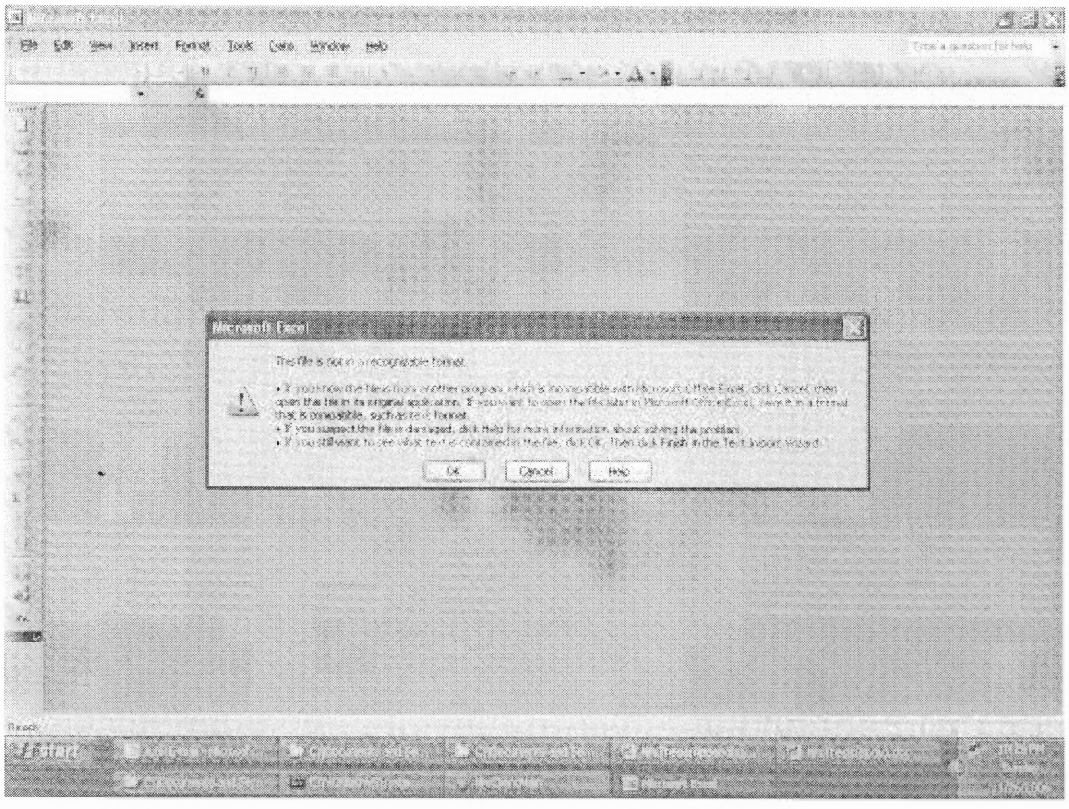

Figure 8.2 Microsoft Excel program is not able to open an encrypted NRSFTP file.

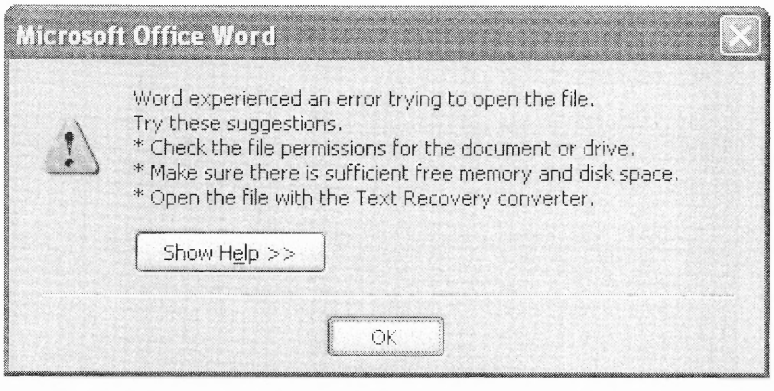

Figure 8.3 Microsoft Word program is not able to open an encrypted NRSFTP file.

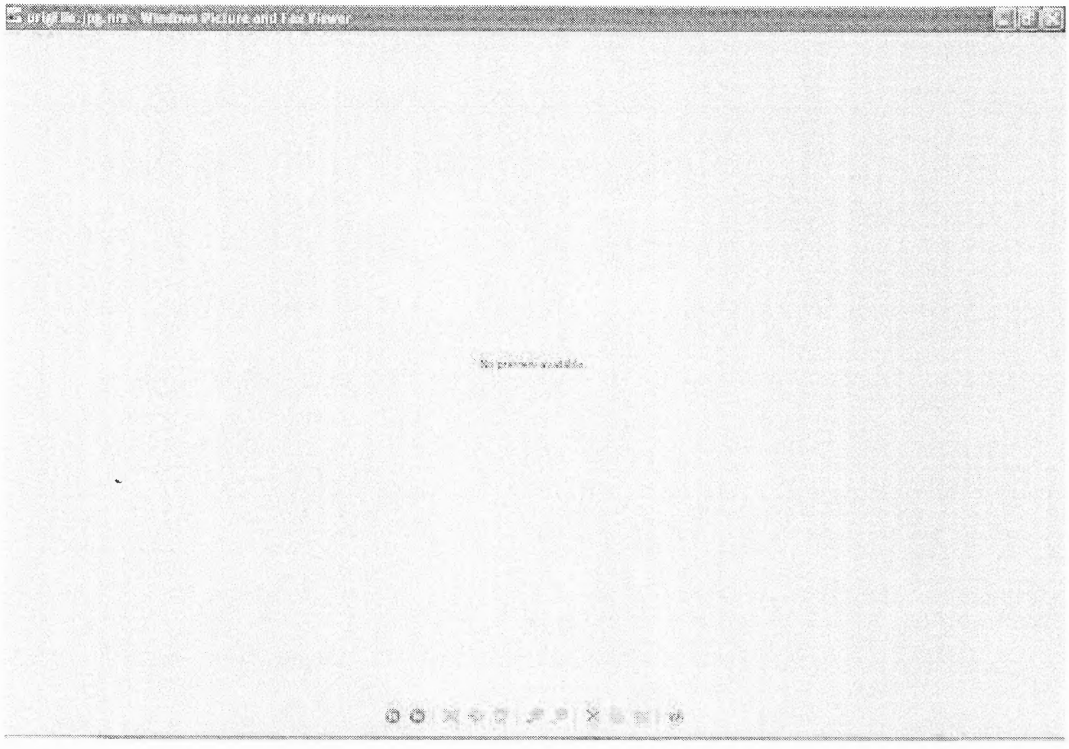

Figure 8.4 Windows Picture and Fax Viewer program is not able to open an encrypted NRSFTP file.

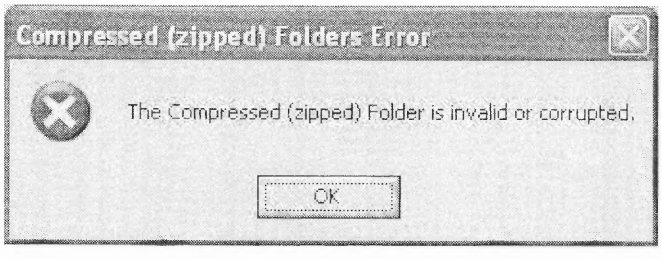

Figure 8.5 Windows zip compression program is not able to open an encrypted NRSFTP file.

8.2 Non-Repudiation

The NRSFTP encrypted file was also tested to ensure the non-repudiation functionality was working properly. Various tests were performed to possibly compromise the NRSFTP encrypted file.

Some of the tests were performed in an attempt to decrypt a file using TP setups that do not contain the proper digital certificate information for either the sender or the receiver. Since the decryption process match and validates both the sender and the receiver digital certificates against the TP setup and the CRL files, it's impossible for an attacker to imitate the sender or receiver digital certificates and decrypt the content of the file. The decryption process with an invalid TP setup was unsuccessful in decrypting a different receiver's NRSFTP file. Please refer to Figure 8.6 for sample screen shot of the test performed.

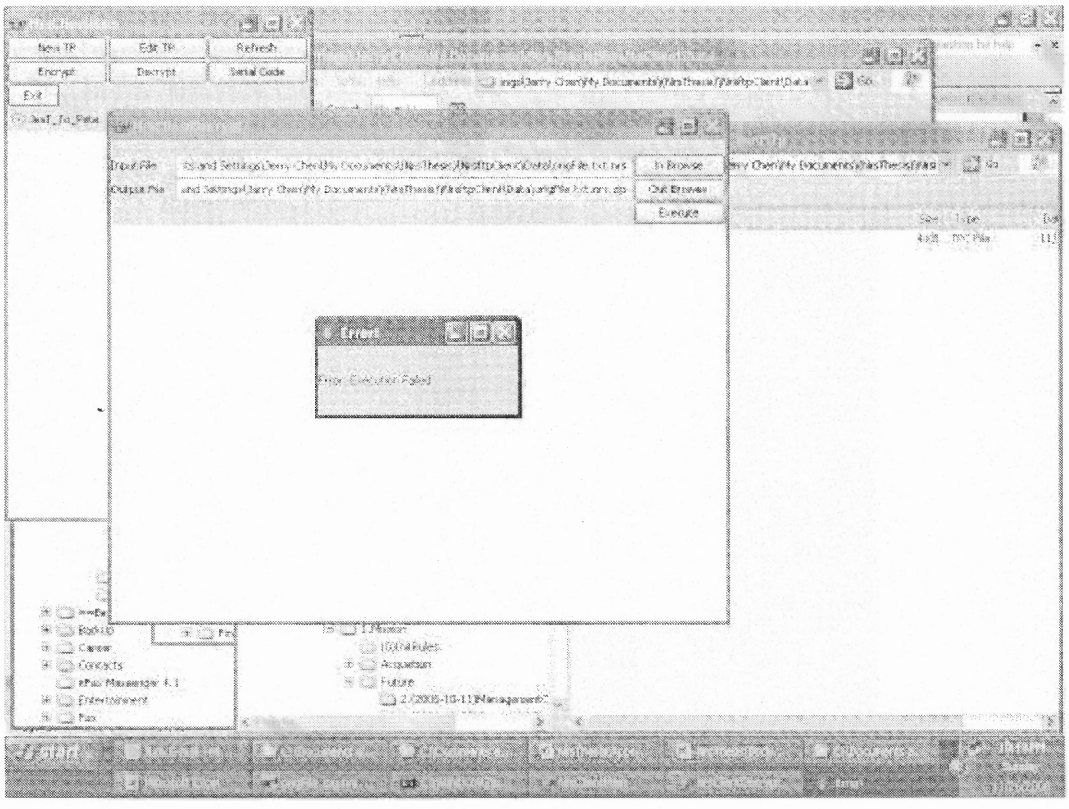

Figure 8.6 Example of the program not allowing the user to decrypt an encrypted NRSFTP file when a TP setup does not match the sender's and the receiver's digital certificates.

Another test that was performed is revoking the sender's digital certificate after generating and sending a NRSFTP file. When the receiver tries to decrypt the NRSFTP file, and if the TP set up for the receiver is set to check against the CRL file, the receiver will not be able to decrypt the file. This test proves that if the sender's private key is jeopardized due to security breach, the sender can still revoke the sender's digital certificate to ensure the receiver will not be able to decrypt the NRSFTP file for non-repudiation purposes. Please refer to Figures 8.7 and 8.8 for sample screen shots of the tests performed.

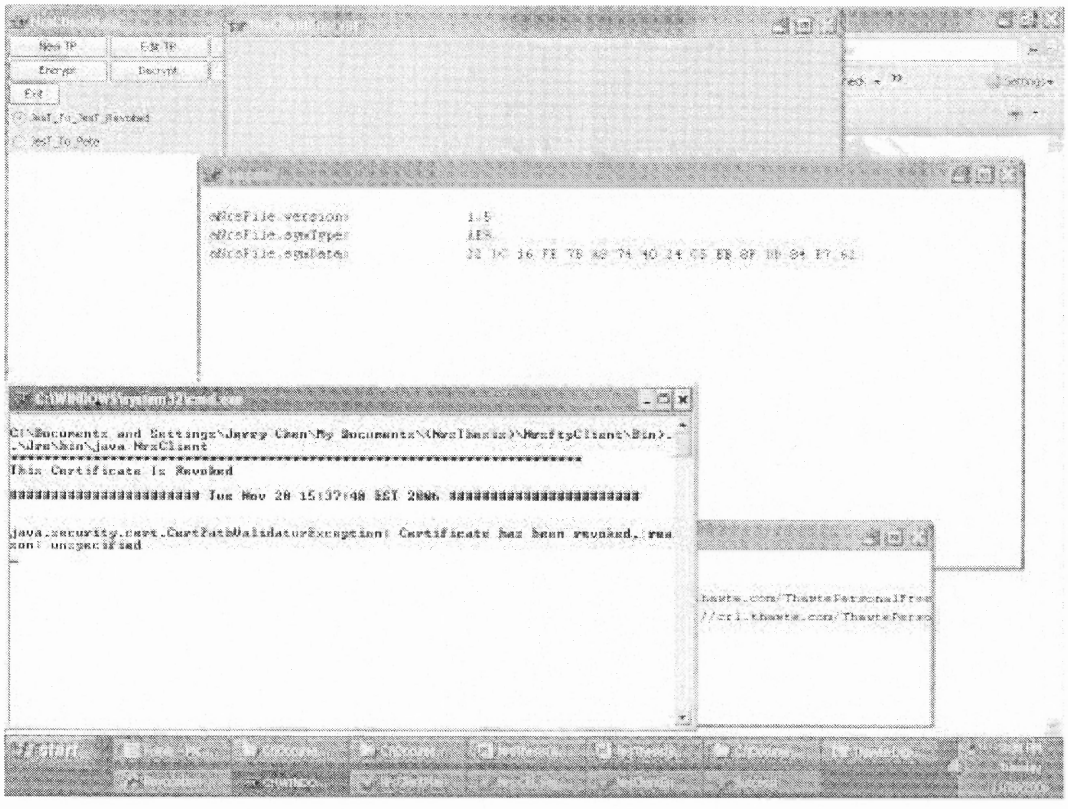

Figure 8.7 Example of the program not allowing the user to encrypt a file since the digital certificate (sender or receiver) was revoked.

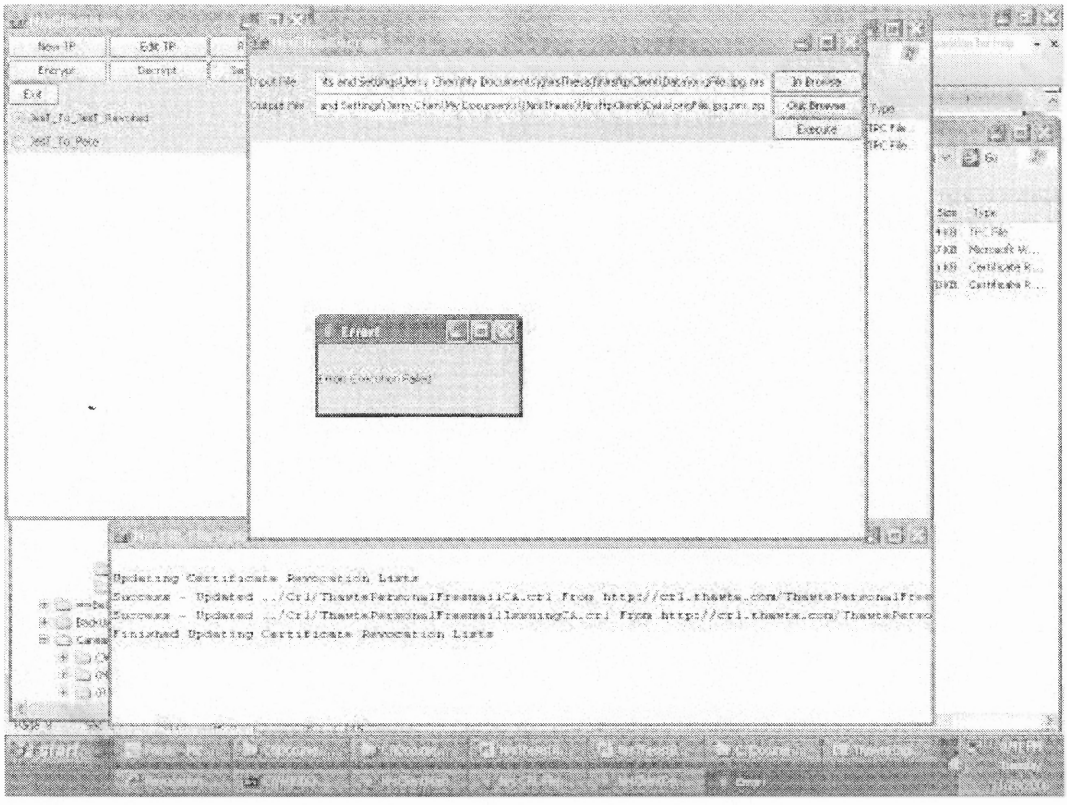

Figure 8.8 Example of the program not allowing the user to decrypt a file since the digital certificate (sender or receiver) was revoked.

Lastly, tests were performed with incorrect TP setups for either the sender or the receiver. Some of the incorrect fields include the CRL files information, digital certificates, and private key files. The NRSFTP program checks and validates all these components for both the encryption and decryption process. For example, if the private key does not match the digital certificate, then the NRSFTP program does not perform the encryption/decryption process and generates an error message. Also, if any of the CRL files does not belong to the proper intermediate or root digital certificates, the NRSFTP program will not allow the user to encrypt/decrypt. Lastly, if the intermediate or root digital certificate does not match the chain of authority for

the sender/receiver digital certificate, the NRSTP program will not allow the encrypt/decrypt process. Please refer to Figure 8.9 for sample screen shot of the test performed.

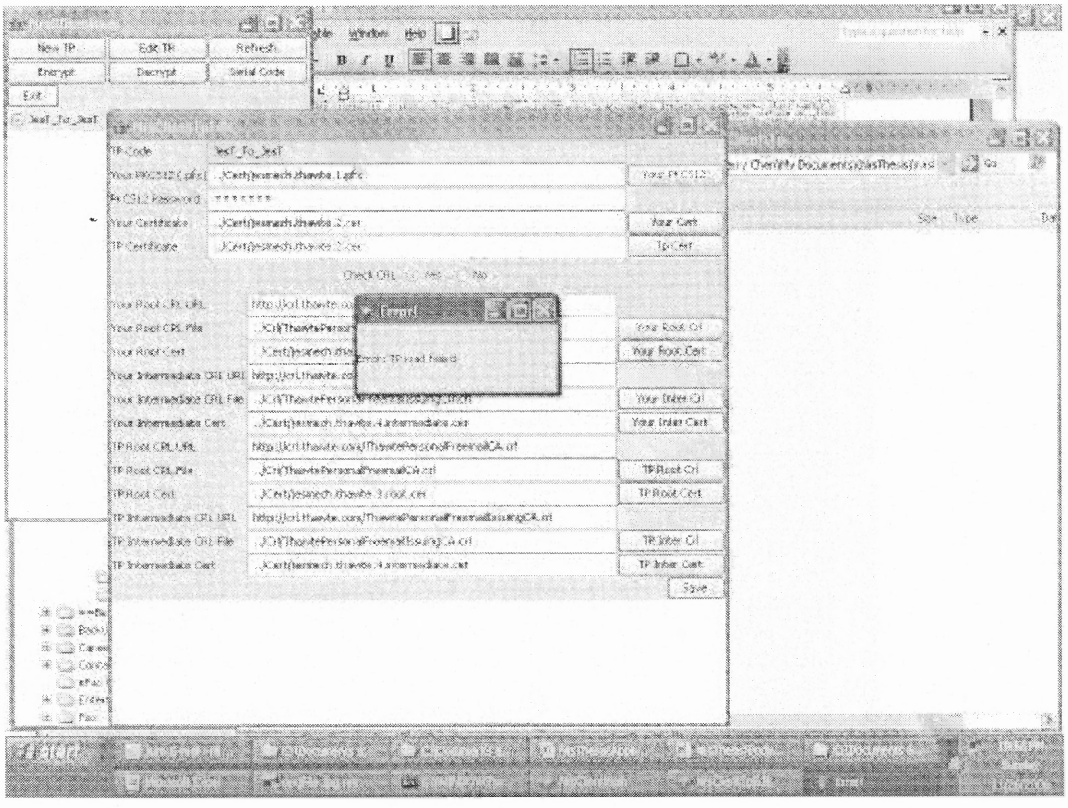

Figure 8.9 Example of the program not allowing the user to setup a TP when the CRL file does not match the digital certificate (sender or receiver).

8.3 Non-Real Time File Transfer

The last important aspect of NRSFTP is the capability of non-real time file transfer. This functionality was also tested to ensure the non-real time transfer will not have any affect or corrupt the NRSFTP encrypted file in any way. Some of the non-real time transfer test methods were FTP, email, and USB file transfer. In all the above transfer methods, the NRSFTP file was decrypted without any issues. In addition, the

NRSFTP file was left in transit for over a period of time and the NRSFTP file was still able to decrypt with validating against the CRL. Please refer to Figure 8.10 for sample screen shot of the test performed.

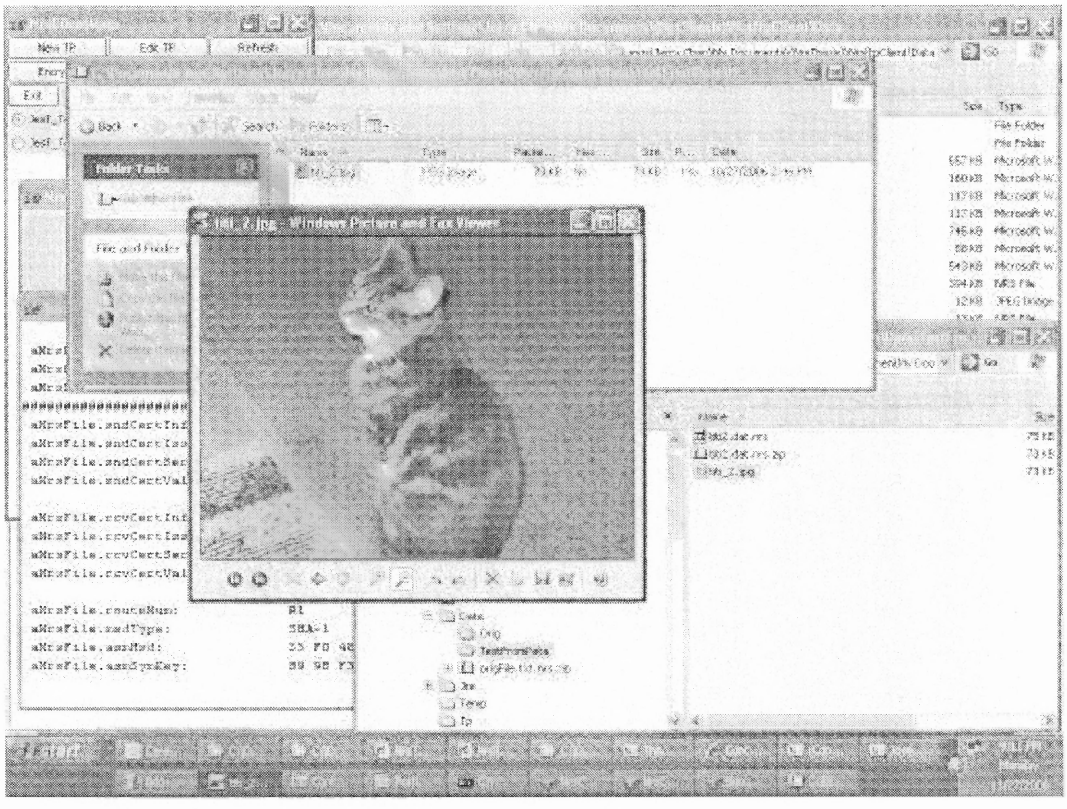

Figure 8.10 The NRSFTP file can be transported in non-real time and the NRSFTP file can still be decrypted after a period of time being left in transit. The picture shown above is from another user using email to transfer the NRSFTP encrypted file and the decryption process took place at another location.

8.4 User Functionality Testing

To better accommodate the usability for testing purposes, many additional functionalities are enhanced to the NRSFTP program. To ensure the functionalities are working properly, the below tests and validations were performed.

- No other third party binaries other than just Java
- Using Java SDK 5.0
- Incorporate the TP screen into the main screen
- Automated CRL file update
- Serial Code Validation for only Encryption
- TP Creation with Check CRL
- TP Creation without Check CRL
- Refresh the TP screen
- Edit TP
- TP Creation with expired TP certificate
- TP Creation with expired Your certificate
- TP Creation with revoked TP certificate
- TP Creation with revoked Your certificate
- TP Creation with non-matching Root Cert and CRL
- TP Creation with non-matching Intermediate Cert and CRL
- Encryption with Check CRL TP
- Encryption with Non-Check CRL TP
- Encryption for multiple TP
- Encryption specifying duplicate TP
- Encryption for multiple input files
- Encryption with bad TP set up (expired TP certificate)
- Encryption with bad TP set up (expired Your certificate)
- Encryption with bad TP set up (revoked TP certificate)
- Encryption with bad TP set up (revoked Your certificate)

- Encryption with bad TP set up (non-matching Root Cert and CRL)
- Encryption with bad TP set up (non-matching Intermediate Cert and CRL)
- Program memorizing last execution parameters
- Display Certificate Information After Encryption
- Decryption with Check CRL TP
- Decryption with Non-Check CRL TP
- Decryption from a multiple TP/Header NRS file
- Decryption for multiple input files
- Decryption with bad TP set up (expired TP certificate)
- Decryption with bad TP set up (expired Your certificate)
- Decryption with bad TP set up (revoked TP certificate)
- Decryption with bad TP set up (revoked Your certificate)
- Decryption with bad TP set up (non-matching Root Cert and CRL)
- Decryption with bad TP set up (non-matching Intermediate Cert and CRL)
- Display Certificate Information After Decryption
- Command Line Encryption with Check CRL TP
- Command Line Encryption with Non-Check CRL TP
- Command Line Encryption with bad TP set up (expired TP certificate)
- Command Line Encryption with bad TP set up (expired Your certificate)
- Command Line Encryption with bad TP set up (revoked TP certificate)
- Command Line Encryption with bad TP set up (revoked Your certificate)
- Command Line Encryption with bad TP set up (non-matching Root Cert and CRL)
- Command Line Encryption with bad TP set up (non-matching Intermediate Cert and CRL)

- Command Line Decryption with Check CRL TP
- Command Line Decryption with Non-Check CRL TP
- Command Line Decryption with bad TP set up (expired TP certificate)
- Command Line Decryption with bad TP set up (expired Your certificate)
- Command Line Decryption with bad TP set up (revoked TP certificate)
- Command Line Decryption with bad TP set up (revoked Your certificate)
- Command Line Decryption with bad TP set up (non-matching Root Cert and CRL)
- Command Line Decryption with bad TP set up (non-matching Intermediate Cert and CRL)
- When closing window, close only current window
- Automated log information when error occur

CHAPTER 9

FUTURE IMPROVEMENTS

9.1 NRSFTP Server

Although NRSFTP client is designed for non-real-time processing, the protocol itself can also be used for real time processing. One of the great advantages for this protocol is that a NRSFTP server can be implemented without the need for a user ID and password authentication taken place. Due to the non-repudiation and secure nature of the NRSFTP file, the server can accept an NRSFTP document without questioning its authenticity.

The NRSFTP server can also be implemented as a Value Added Network (VAN) which can route the NRSFTP files securely and without uncertainty to the intended receiver. The sender and receiver digital certificates embedded within the NRSFTP file are the identifiers for the routing sequence necessary for the server.

9.2 File Transfer Methods for NRSFTP Client

When the sender has a need to encrypt a file intended only for the receiver to see, there must be a method developed to exchange the file between the sender and the receiver. Therefore, as part of NRSFTP, different file transfer methods are addressed. Since one of NRSFTP main focus is to allow non-real-time transfer of documents, NRSFTP client should be able to handle any form of transmission method. The file transfer methods addressed by NRSFTP client are.

- Save into directory (up to the user to transmit the file)
- Email
- FTP
- AS2
- NRSFTP

CHAPTER 10

CONCLUSION

In summary, NRSFTP is designed to cover the following core functionalities.

- Secure transfer of document to the intended recipient.
- Provide non-repudiation to the document.
- Be able to transfer the file non-real time.

NRSFTP is designed to accommodate a wide area of security and transmission needs. Documents can be securely exchanged with non-repudiation without having to have the document exchanged through a real-time environment.

- The NRSFTP client software will have to provide the following functionalities to ensure wide overall acceptance.
- Encrypt and decrypt using different trading partnership setups.
- Automated import and export of digital certificates to notify other trading partners about your digital certificate.
- Command line execution to accommodate the need of automated environments for ecommerce.
- Compatible with other current open protocols like FTP, PGP, and AS2.

REFERENCES

1. Charlie Kaufman, Radia Perlman and Mike Speciner, *Network Security*, Prentice Hall PTR, Upper Saddle River, NJ, 2002

2. Kent Cearley and Lindsay Winsor, "Securing IT Resources with Digital Certificates and LDAP" in Proceedings of the 1997 CAUSE annual Conference August 2004, http://www.educause.edu/ir/library/html/cnc9707/cnc9707.html.

3. Charlene O'Hanlon, CRN, "VeriSign, Microsoft to Deliver Authentication via Windows Server 2003" September 2004, http://www.crn.com.

4. VeriSign, "Oath – initiative for open authentication" September 2004, http://www.openauthentication.org.

5. PGP Corporation, "An Introduction to Cryptography" September 2004, http://www.pgp.com.

6. Dave H. "FTP File Transfer Protocol" September 2004, http://www.cs.rpi.edu/courses/fall96/netprog/lectures/html/ftp/.

7. N Software Inc. "FTPS Componenet" September 2004, http://www.nsoftware.com/products/controls/?ctl=FTPS&prod=ipsssl.

8. John Radko, Chief Architect for Global eXchange Services (GXS), "Global eXchange Services: AS2 White Paper" September 2004, http://www.gxs.com/gxs/as2wprequest.