

**UNIVERSITY OF THE PHILIPPINES MANILA
COLLEGE OF ARTS AND SCIENCES
DEPARTMENT OF PHYSICAL SCIENCES AND MATHEMATICS**

Simulating the Rivest - Shamir - Adleman Algorithm by Creating a Messaging System

A Special Problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

I. INTRODUCTION

A. BACKGROUND OF THE STUDY

Cryptography is the science of making a message unintelligible to anyone but the intended recipient. Cryptography has a long history. One of the first persons to use cryptography was Aeneas Tacticus. On his military manual (360 BC), there is a chapter entitled "On Secret Messages". In this chapter he described how he used different devices to create secret messages [1].

Another famous person to use the principles in cryptography was Julius Caesar. Caesar did not trust his messengers, but he needed to send vital information to his generals in far places. His solution was to code the message by substituting the letters of the message by another letter in with a pattern. For example "a" was changed to "g", "b" is changed to "h" and so on. In effect a message "Use the infantry to attack" would read as "Ayk znk otlgtzxe zu gzzgiq". The reader of the message must know that "a" = "g" to understand the message. Caesar also foresaw the event in which the code would be compromised. In effect he also thought of a way to change the code quickly. Using the same

example, it is seen the “a” is 6 letters away from “g”. Caesar could change 6 to 3. This would imply that “a” = “d”. By using different codes for different messages the chances of compromising the code was lessened. This also implied, that in the event that the message was compromised previous or succeeding messages would not be compromised [2].

Coding and decoding messages were often compared to locking and unlocking. Hence the tools used to lock and unlock were called keys. Just as a certain key will only fit a certain lock, a key in coding will “lock” the message in a certain way, and to “unlock” it one would have to know the key. In relation to the example above, Caesar’s ciphers used a number for his key. Although it tricked a lot of his enemies, the Caesar’s ciphers only had 26 keys, which meant that a person could intercept a message and use all 26 keys and the one message, which is intelligible, can be concluded to be the real message. This shows that the strength of coding and decoding (Cryptographic) system relies on the keys of the system [1].

Cryptography was priceless tool in the military. Knowledge of the messages of the enemy would mean victory. Cryptography was so valuable that during World War II, the allied forces spent a

lot of money to break the axis code and vice versa. Both sides used computers to create and break codes by creating a powerful key and by trying to gain information about the secret keys of the enemy. Cryptography was in fact one of the first functions of the computer [2].

Today, cryptography is used to create privacy. With the growing use of the Internet (which is considered as an insecure medium) as a medium of communication, it is crucial that vital information must be kept a secret during transmission of vital information. This is where cryptography is applied in modern society other than the military [2].

The Rivest - Shamir - Adleman (RSA) Encryption Algorithm is one of the most popular (if not the most popular) encryption algorithms. Systems like PGP and Open PGP use this encryption algorithm [3].

B. STATEMENT OF THE PROBLEM

UP Manila uses online systems. These systems are usually protected by passwords. Also, the data of these programs are transmitted on networks and are stored in databases in its original form. These data are sometimes vital information that is not for everybody to see. Although the systems provide secrecy to the

layman, a trained hacker could easily break into these systems and retrieve data from the database, intercept data during transmission, or even falsify the information on the database. One example of such systems is the Online Student Evaluation of Teachers (OSET). The project leader of OSET admits that a person with knowledge on PHP and MySQL, can break into the system. He also admits that the data of the system should be protected from exposure. Exposure of these data implies the breakdown of the system^a.

C. OBJECTIVES OF THE STUDY

1. To simulate the encryption and decryption of a text file using the Rivest - Shamir - Adleman (RSA) Algorithm
2. To simulate the key generation of the RSA Algorithm.
3. To simulate a message transmission over an insecure medium by sending data to a server where the data is exposed to anybody able to access the server.
4. To simulate an attack by intercepting a cipher and attempt to decrypt the cipher by brute force.
5. To create an application that could be imported by PHP scripts that encrypts and decrypts text files.

^a Conversation with Herbert Eumague, project leader of OSET, on the limitations of OSET.

D. SIGNIFICANCE OF THE STUDY

Since UP Manila uses automated systems, which do not use encryption, it could be easily be broken. With the study, the systems could be modified to encrypt its data before it is stored or transmitted. With data encryption, information is made accessible to and only to the intended person. It would also protect the system from data interception, unauthorized retrieval of data, or even falsification of data. As most of the online systems in UP Manila are created using PHP, the study will also produce an application that could be imported by PHP scripts that encrypts and decrypts text files.

The study will also serve as a guide when creating future online systems that contain vital information, by protecting the information through encryption.

E. SCOPE AND LIMITATIONS

The scope of the study will be on the encryption, decryption, key generation, and transmission using the RSA algorithm. The data to be transmitted over the medium is limited to text messages. Text messages will have variable length will be

composed of characters with an ASCII equivalent. Other aspects of the RSA algorithm such as digital signing will not be included.

II. REVIEW OF RELATED LITERATURE

There are a lot of studies about cryptography. An area in cryptography that is prevalent among studies is the cryptanalysis. Cryptanalysis is the study on the strength of an encryption algorithm and its components. Cryptanalysis' main goal is to break a cipher or deduce the key from the available tools such as a pair of cipher and plaintext, language statistics, etc [4].

In a study by Kilma *et al.* [3], an attack on two of the most used cryptosystem, Pretty Good Privacy (PGP) and Open PGP (an open source version of PGP), has been successful. These two systems use Rivest - Shamir - Adleman (RSA) algorithm and Digital Signature Algorithm (DSA) to encrypt its files. These two systems produce strong ciphers but the point of attack was not on the cipher. The private key file (which was encrypted) was modified, then a signed message intercepted. The researchers were able to deduce the private key from the changes in the statistics, thus compromising the security of the files. With the results, it was then proposed that Open PGP and PGP needed revisions.

In another study Ooi *et al.* [5], a cryptanalysis was done on S - Data Encryption Standard (DES). The first form of attack was brute force, it was then seen that a brute force attack would be feasible if S-

DES had a key length of 56 bits or less. Although brute force is the most primitive form of attack it is becoming more feasible with the increase of computational power. Another form of attack that was used is the differential cryptanalysis. In this differential cryptanalysis involves the analysis of the effect of the plaintext pair difference on the resulting cipher difference. The most common difference utilized is the fixed XORed value of the plaintext pairs. By exploiting these differences, the partial subkey used in the cipher algorithm can be guessed. This guess is done statistically by using a counting procedure for each key in which the key with the highest count is assumed to be the most probable partial subkey. This resulted to 8 bits of an actual 10 bits of the S-DES key. The whole key could be obtained by trying the remaining 2^2 possibilities, which is considered feasible. Another attack was used on S-DES was the linear cryptanalysis. The main idea behind linear cryptanalysis is to obtain an approximation to the block cipher as a whole using a linear expression. The goal is to find the linear expression, which holds with the highest/biggest linear probability bias. It was found out that with the increase of plaintext, the rate of success of approximation also increased.

Although the main objective of cryptanalysis is break a cryptosystem, it does not weaken the system but on the contrary it strengthens the system. Since the attacks are becoming visible with

cryptanalysis, designers of cryptosystems design their systems with response to the attacks stated in the cryptanalysis.

Studies in cryptography also include new means of encrypting data. In a study by Schmidt [6], it states that block sizes should be 128 bits to create a secure cipher but common computers today only have 32-bits of register (64 for high end computers). The proponent of this study developed a code, which encrypted a 256-bit block of data by dividing it into block of 32-bits then encrypting these smaller blocks with sixteen 32-bit independent keys. The block is then rebuilt to produce a 256-bit block and a 516-bit key. This block cipher scheme was resistant to differential cryptanalysis. The proponent recommends a linear cryptanalysis of this scheme.

III. THEORETICAL AND CONCEPTUAL FRAMEWORK

Cryptography

Cryptography comes from the Greek word “kryptos” and “logos”, which when combined means hidden word. The basic idea of cryptography is to make the message unintelligible to anyone else except the intended reader. During encryption, keys are used to assist in creating the cipher. These keys are also used to decrypt the cipher to plaintext form. In the cryptosystem of Julius Caesar, he used integers as the keys. The integer represented the fixed distance between a letter

and its replacement. The use of keys is one of the most important principles of cryptography [2].

Assumptions

It is assumed that M is the plaintext, C is the cipher, (K, L) is the public and private key pair, $e(x, K)$ is the encryption process (or function), and $d(x, L)$ is the decryption process (or function). Therefore, $e(M, K) = C$ and $d(C, L) = M$ and $d(e(M, K), L) = M$ [1].

Symmetric Key Encryption Scheme

Encryption is classified by the use of keys into two, the symmetric key encryption and the asymmetric key encryption. Symmetric keys encryption uses keys in both encryption and decryption. The symmetric key encryption is simple to do, but it is very vulnerable to attacks. One way to break a symmetric key encryption is to randomly select a key then use it to decrypt a message. Repeat this process until the cipher is intelligible. Although this technique will require a lot of man-hours, this simple task could be given to an intelligent agent (computer with AI) and the time to break the system will be much less. Another disadvantage of using a symmetric key encryption is the fact that it requires a single key, which would imply that this key must be transmitted over a

medium. This would again be a point of attack. The attacker could wait for the key to be transmitted and intercept it [2].

An example of a symmetric key encryption is the Linear Transformation. In this encryption technique, the message is separated into blocks and then represented into matrices. To encrypt the message, these matrices are multiplied (matrix multiplication) to a non-singular matrix. To decrypt the message the cipher matrices are multiplied to the inverse of the non-singular matrix. In this scheme the key is the non-singular matrix.

To illustrate the linear transformation, a non-singular key is chosen say,

$$\begin{bmatrix} 3 & 13 \\ 22 & 15 \end{bmatrix}.$$

Its inverse would be

$$\begin{bmatrix} -0.0622 & 0.0539 \\ 0.0913 & 0.0124 \end{bmatrix}.$$

If a message GOOD is to be sent, it is first written as,

$$\text{GOOD} = 7, 15, 15, 4$$

It is then converted to a matrix

$$\begin{bmatrix} 7 & 15 \\ 15 & 4 \end{bmatrix}$$

Then the key and the matrix of the message are multiplied,

$$\begin{bmatrix} 3 & 13 \\ 22 & 15 \end{bmatrix} \begin{bmatrix} 7 & 15 \\ 15 & 4 \end{bmatrix} = \begin{bmatrix} 216 & 97 \\ 379 & 390 \end{bmatrix}$$

The matrix produced is the cipher.

To decrypt the message the cipher is multiplied to the inverse of the key.

$$\begin{bmatrix} -0.0622 & 0.0539 \\ 0.0913 & 0.0124 \end{bmatrix} \begin{bmatrix} 216 & 97 \\ 379 & 390 \end{bmatrix} = \begin{bmatrix} 7 & 15 \\ 15 & 4 \end{bmatrix}$$

The product is converted to its plain text form, which is GOOD [1].

Another example of the symmetric key algorithm is the Data Encryption Standard (DES). The basic idea of DES is to transform a 64-bit block to another 64-bit block with the use of a 56-bit key by means of permutation and substitution. A variant of DES is Triple DES. In this scheme a message is encrypted three times using DES with 3 different key. This makes the key longer thus making it harder to attack.

To illustrate the DES, suppose a message in hexadecimal is to be sent say, "0123456789ABCDEF", and the key to be used is "133457799BBCDFF1". Both the key and the message is converted to binary. Then the message undergoes series of permutation, substitution and XOR's with the key. It produces the cipher "85E813540F0AB405". To decrypt the cipher, both the key and cipher are converted to binary then it goes through the series of permutation, substitution and XOR's with the key in reverse order. Then the message is again converted back to hexadecimal, which is "0123456789ABCDEF" [7].

Asymmetric Key Encryption Scheme

The other classification according to the use of keys is asymmetric key encryption (sometimes called the public-key encryption). This scheme uses two keys, one key for encryption called the public key, and the other key for decryption called the private key. The idea is that public keys are available of any one to use and the corresponding private key is kept secret by the receiver. For example User A wishes to send a message to User B. User A will use User B's public key and encrypts the message. Upon receiving the cipher, User B uses his private key to decrypt the message. This scheme is made possible by trap-door functions, which are complex. Since these trap-door functions are complex, its keys are also complex; this implies that a brute force attack on the cryptosystem will be long even for a computer. Since the private keys are not transmitted through a medium, it could not be intercepted. Asymmetric key encryption, however, are not truly safe. Trap-door functions still have inverses but it is hard to perform at present time. Therefore trap-door functions today may not be trapdoor functions in the future [2].

An example of asymmetric key encryption is Elliptic Curve Encryption. The general form of an elliptic curve is $y^2 + xy = x^3 + ax^2 + b$. The elliptic curve has a special property that when 2 points in the graph it produces a point in the graph. In this scheme the public and private

keys are generated from an elliptic curve where a and b are non zero real numbers. The strength of the elliptic curve encryption is due to the fact that factoring is not an easy process [8].

To illustrate the Elliptic Curve Encryption, first a Galois finite field GF is chosen on an elliptical curve $P(x)$ with a point P lying in GF . Z_p denotes the order of P . GF , $P(x)$, P and Z_p is made public. To generate the keys, a random number $k \in Z_{p-1}$ is generated then $Q=kP$ is computed. Point Q is then made Public. k is made private or secret key. To encrypt a message, suppose Alice sends a message m to Bob. Alice look up Bob's Public Key: Q . Then the message m is represented as a pair of the field elements (m_1, m_2) , $m_1 \in GF$, $m_2 \in GF$. Then a random integer a is selected, such that $a \in Z_{p-1}$. Then the point $(x_1, y_1) = aP$ is computed. Then the point $(x_2, y_2) = aQ$ is computed. The field elements m_1, m_2 with x_2 , and y_2 with some algorithm is combined to give two field elements c_1 and c_2 . After which, the data $m_e = (x_1, y_1, c_1, c_2)$ is transmitted to Bob. To decrypt the cipher Bob computes the point $(x_2, y_2) = k(x_1, y_1)$, using it's private key k . Then m_1 and m_2 are decrypted from m_e [9].

Another asymmetric key encryption is the Knapsack Algorithm. This encryption scheme, used the knapsack problem, which states that, given a set of positive integer $\{a_1, a_2, \dots, a_i, t\}$ is there a subset $J \subseteq \{1, \dots, n\}$

such that $\sum_{i \in J} a_i = t$? During 1978, the knapsack encryption algorithm was considered to be secure. Today however, this scheme has been broken and considered insecure [10].

To illustrate the Knapsack Algorithm, a public key given is an n -tuple $A = (a_1, a_2, \dots, a_n)$ of distinct positive integers, as well as another positive integer k . The question is then which integers a_i sum to equal k . To encrypt the message SECRET, its first converted to binary.

S	E	C	R	E	T
101001	100010	100001	101001	100010	101010
1	1	1	0	1	0

Since each letter corresponds to a binary number with 7 digits, $n = 7$ is set. First a private key made up of n numbers is picked such that the a_i 's are in increasing order, say (1,2,5,11,32,87,141). Next two more numbers: m such that it is greater than the sum of all a_i and w which must have no common factors with m are picked say $w = 901$ and $m = 1234$. These are used to establish our public key by the equation $a_i = w * a_i \text{ mod } m$, where a_i is any single member in the public key and a_i is the corresponding member in the private key. This gives the public key of $A = (901,568,803,39,450,645,1173)$. Next the public key is applied to each letter. Encrypting just the letter S gives:

$$B = 1 \times (901) + 0 \times (568) + 1 \times (803) + 0 \times (39) + 0 \times (450) + 1 \times (645) + 1 \times (1173) = 3522$$

Then the number 3522 is sent to the receiver, he can decrypt that using his private key and the equation $B = B * w^{(-1)} \text{ mod } m$, where $w^{(-1)}$ is the multiplicative inverse of w . This gives $B = 3522 * (901)^{(-1)} \text{ mod } 1234 = 3522 * 1171 \text{ mod } 1234 = 234$. The combination of A that will yield 234 is to be searched. This is easily done since each member of A is larger than all of the members to the right of it added together. In this example we get $141+87+5+1$ or 1010011 which is the same as the S . This solution can always be found without the key by trying all of the subsets of A , but if there are hundreds and hundreds of the numbers a_i , then the problem quickly becomes unmanageable without the key [11].

Another asymmetric key encryption is the RSA (Rivest - Shamir - Adleman) Algorithm. It is often considered a defacto standard. In this scheme the public and private keys are generated from 2 prime numbers. Like the elliptical curve scheme, the strength of RSA lies on the fact that factoring is difficult.

Although Elliptical Curve Algorithm has smaller keys, RSA is faster during signing and decryption. RSA is also significantly faster during key exchange. Elliptical Curve Algorithm is not yet supported by companies since it is a new technology and has not been tested sufficiently [12].

Encryption could also be classified by the ciphers it produces into two, the two-way encryption and the one-way encryption. Two-way encryption produces ciphers that could be decrypted to plaintext. One-way encryption, on the other hand, produces ciphers that could not or would be too hard to decrypt. This kind of encryption used in databases that don't need to be decrypted. An example would be a database of username and passwords. It would be too dangerous to keep passwords in plaintext. With the use of one-way encryption, the passwords in the database could not be decrypted, but could be verified by encrypting the password sent by the user and checking it against the database [2].

Attacks on a Cryptosystem

A person (or an intelligent agent) that is trying to break a cryptosystem is called an attacker. It is assumed that the attacker has full knowledge of the encryption and decryption function. He may also have pair of cipher and plaintext, public keys, language statistics, knowledge of the context.

The attacks are classified by the tools of the attacker. The first is the cipher only attack. In this attack (as the name would suggest), the attacker has a cipher and tries to deduce the plaintext from the cipher.

The second classification is the known-plaintext attack. In this attack, the attacker has a cipher and its corresponding plaintext (may be full or partial). The attacker would then try to deduce the encryption and decryption function and keys from the cipher-plaintext pair through statistics.

The third classification is the chosen plaintext attack. The attacker may have gotten the encryption function, which implies that he has an unlimited number of cipher and plaintext pairs. He could also create ciphers from nonsense messages to check the statistics of language statistics. From the unlimited cipher-plaintext pair, the language statistics the attacker tries to deduce the decryption function and the keys [4].

Digital Signatures

An attacker may disguise himself as a trusted person in the cryptosystem. He could then ask for vital information from other persons in the cryptosystem. This led to the problem of authenticity of a message (or request) for vital information. Digital signatures are the personal mark of a person in the cryptosystem. They are the electronic equivalent of seals and signatures. With a digital signature, it could be verified that a message indeed came from the sender [1].

Rivest - Shamir - Adleman (RSA) Algorithm

RSA is a two-way, asymmetric key encryption scheme. It could also be used for digital signatures. It first generates primes (p and q) and uses it to produce n ($n = pq$). Then a random number d less than n such that the $\text{gcf}(d, (p-1)(q-1)) = 1$, then compute e such that $ed \bmod (p-1)(q-1) = 1$. The public key is set to be the pair (e, n) and the private key is d . The characters of the message are given integral values, then the integers are concatenated to produce a large integer then it is divided into equal size blocks. The blocks (M_i) are encrypted such that $M_i^e \bmod n = C_i$ (where M_i and C_i are of equal block sizes). The cipher would be the concatenation of C_i 's. To decrypt, the cipher is divided into the same block size. The blocks (C_i) are decrypted such that $C_i^d \bmod n = M_i$ (where C_i and M_i are of equal block sizes). Then, the block of M_i 's are concatenated and is reverted to plaintext. (For the Mathematical Proof of RSA see Appendix 1) To digitally sign a message, it is encrypted using the private key and could be verified using the public key. Since the private key and public key is a unique pair, it could be said that the personal mark (which could not be forged with out a private key) is left on the message [1].

RSA Example

In a simple example, let $p = 47$ and $q = 59$, then $n = 2773$. d is then calculated to be 157 and $e = 17$. Then let $A = 01$, $B = 02$, $Z = 26$ and $SPACE = 00$, the message "ITS ALL GREEK TO ME" would be read as 0920190001121200071805051100201500130500. The message is then divided into blocks (4 in this case) $M_1 = 0920$. Then we encrypt the message $C_1 = 920^{17} \bmod 2773 = 948$. Continuing the process on all the blocks the cipher message would be 094823401084144266323900778077402191655. To decrypt the cipher it is again divided into blocks and using the private key, $M_1 = 948^{157} \bmod 2773 = 920$. Continuing on all blocks of the cipher the message is returned to 0920190001121200071805051100201500130500 then by using the look up table we get "ITS ALL GREEK TO ME" [1].

The strength of the RSA algorithm is the fact that factoring a number to a prime components is hard. It is even more difficult to factor a number if it came from the multiplication of 2 prime numbers. It even becomes more difficult if the two prime numbers are large [12].

Definition of Terms

Encryption - the act or process of making a text unintelligible.

Decryption - the act or process of making an encrypted message intelligible.

Plaintext - the message in its original form.

Cipher - the message after encryption

Keys - objects (most of the time numerical in nature) that will assist in the encryption and decryption

One way functions - a function that has no inverse or an inverse that is impossible to solve.

One-way encryption - uses one-way functions to generate the ciphers, in effect the ciphers cannot be decrypted.

Salt - plaintext used in a one-way encryption.

Public-key - Keys that is shared with everyone. Used to encrypt messages.

Private-key - Keys that is kept secret. Used to decrypt messages.

Symmetric Keys - Keys that are used for both encrypting and decrypting messages.

Attack - an attempt to break the cryptosystem.

Successful attack - when the attacker gains knowledge of the keys or could obtain the plaintext from a cipher.

IV. DESIGN AND IMPLEMENTATION

The context diagram of the system is shown in Figure 1. It has system has five main processes as seen in the Top Level Data Flow Diagram (see Figure 2), 1) Create new records, 2) Logging to server, 3) Sending of a message, 4) Receiving messages and 5) Intercept message. It also as 3 databases (stores), 1) Private keys, 2) User information and 3) Message

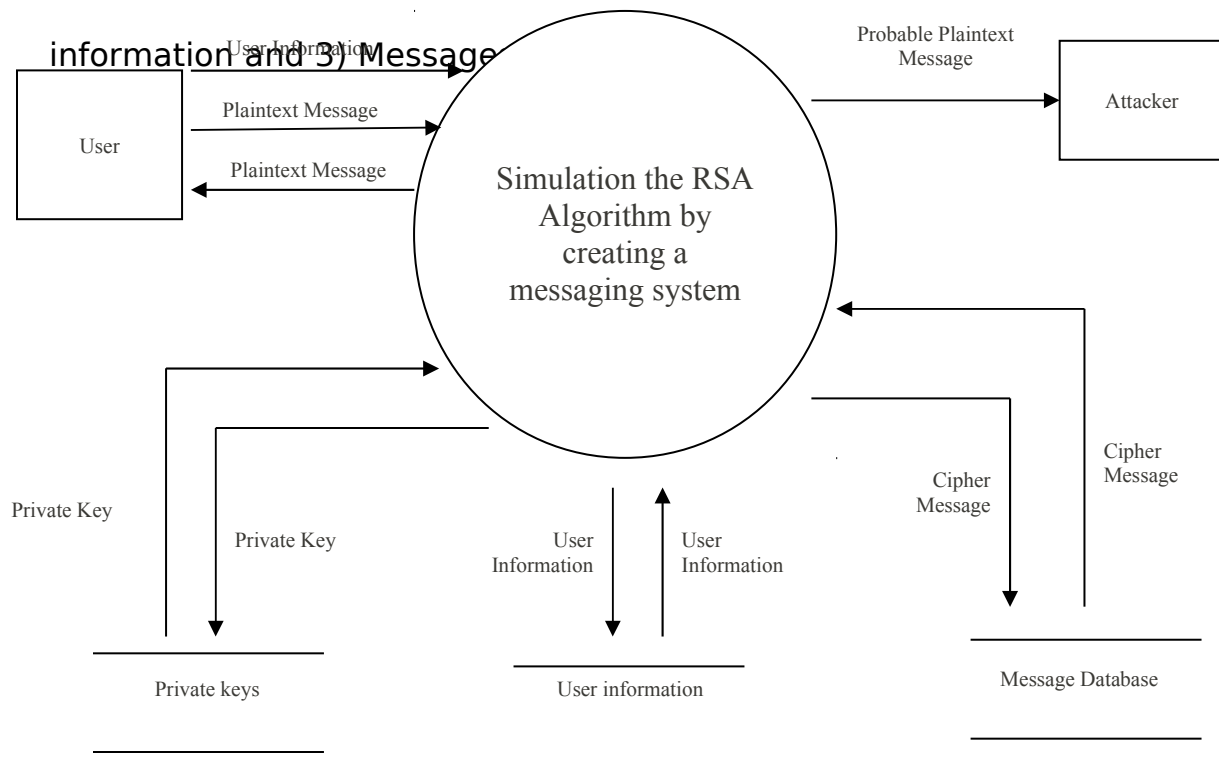


Figure 1 Context Diagram, Simulating the RSA Algorithm by Creating a Messaging System

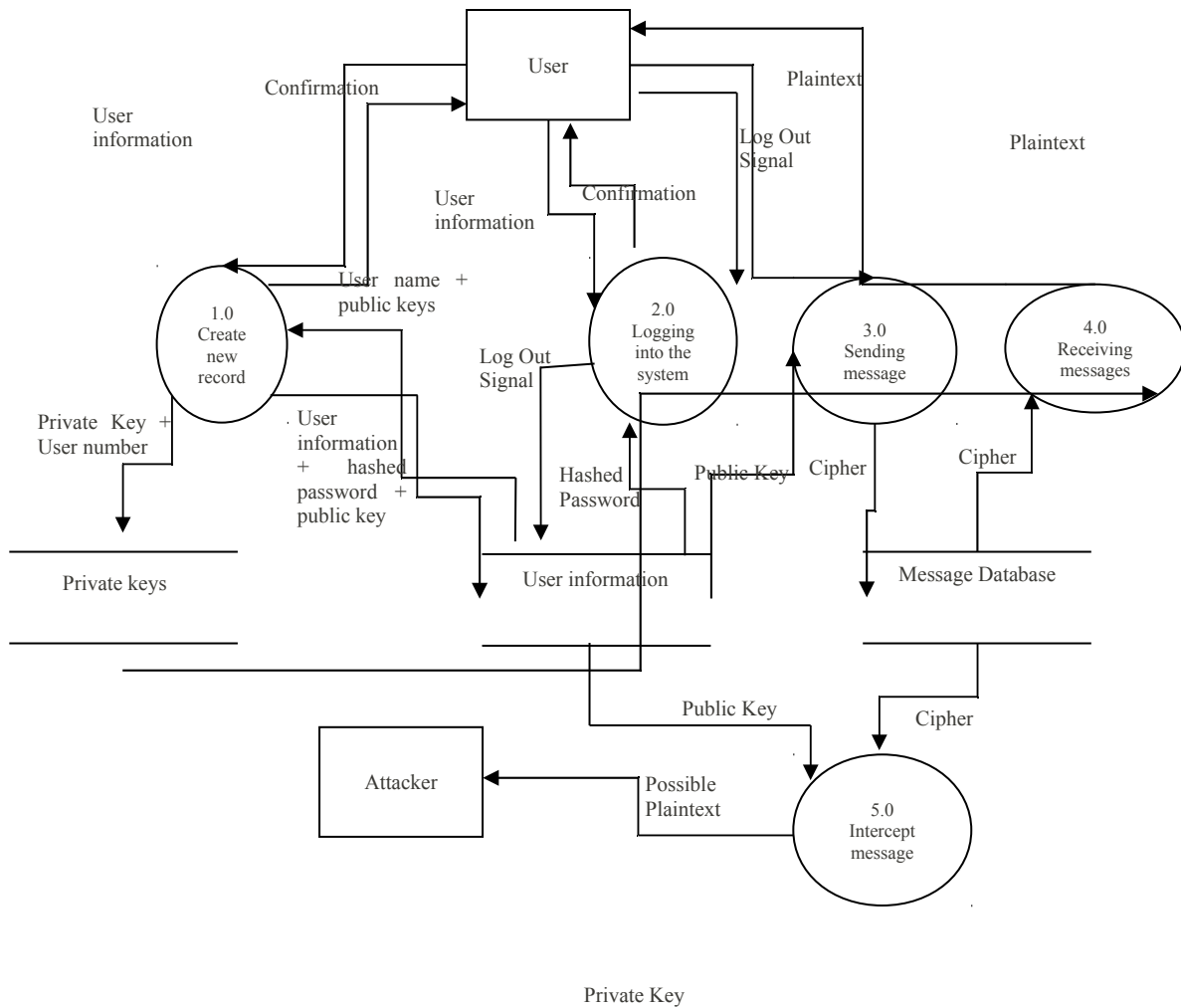


Figure 2 Top Level Data Flow Diagram, Simulation the RSA Algorithm by creating a Messaging system

3.

The databases are located in different sides. The private key database contains the username and his corresponding private key. It is located on the client-side. The user information contains information about the user such as user number, user name, hashed password and his

corresponding public key. It is located on the server-side. The messages database contains the cipher, its sender and its recipient. This store is located on the server side.

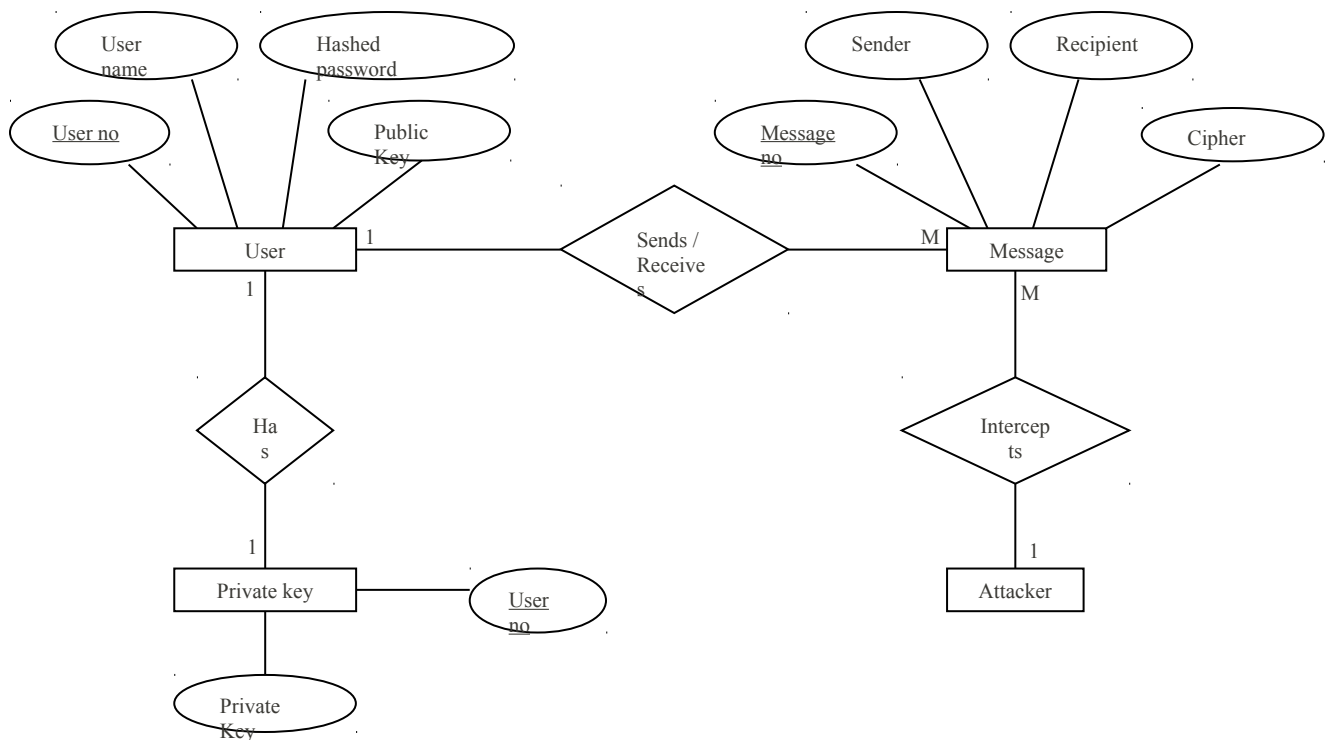


Figure 3 Entity Relationship Diagram, Simulating the RSA Algorithm by Creating a Messaging System

The first process a user will encounter is create new records. This process is used to produce keys and records for the new user. First the user gives a username and password. The user name is then check if it is already in use. If the user name is unique then a unique user number is generated. Then password is hashed so that it is not stored or

transmitted in its raw form. The keys are then generated. The private key is stored in the private keys database and the rest is stored on the User information database. The process then sends a confirmation to the user. (See Figures 4 and 5)

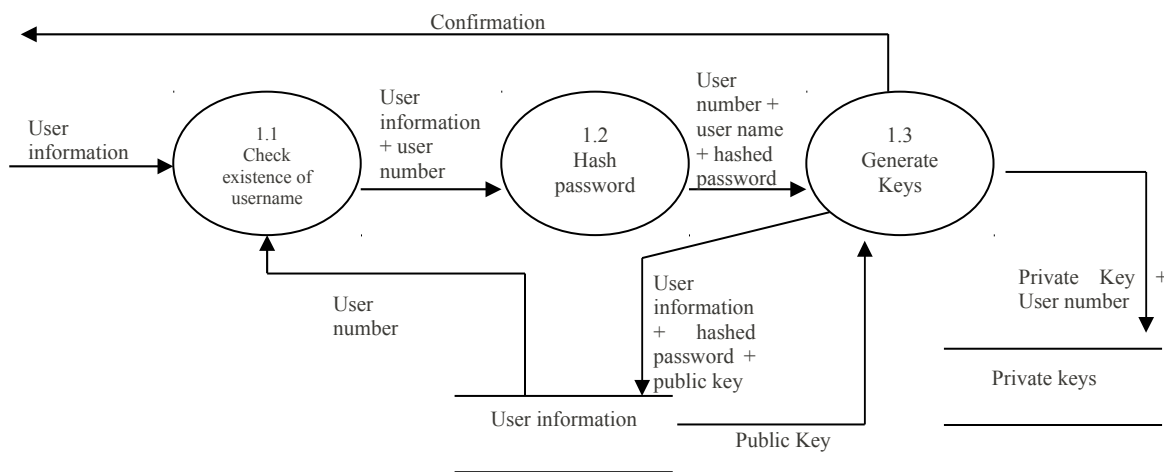


Figure 4 Subexplosion of process, “Create New Records”, Simulating the RSA Algorithm by Creating a Messaging System

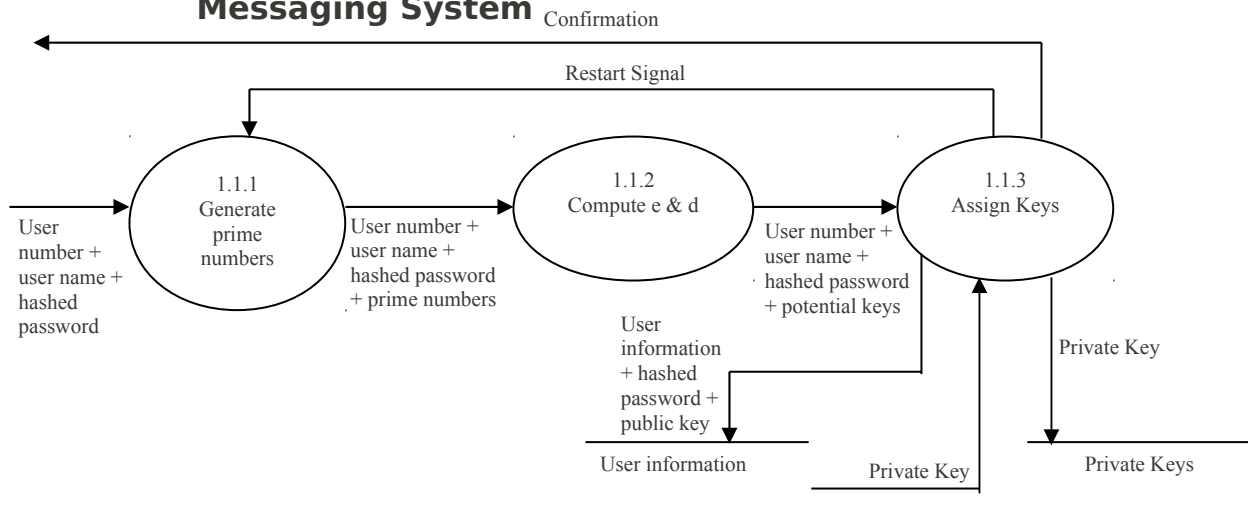


Figure 5 Subexplosion of process, “Generate Keys”, Simulating the RSA Algorithm by Creating a Messaging System

The logging to server controls the users using the database. It makes sure that the users are part of the system. First the user is asked for his user name and password. The process verifies the existence of the user name. Then the password is hashed and checked against the hashed password stored on the database. Once verified, the process sends a confirmation to the user and signals the system that the user is logged on. If a user wishes to log out, the user sends a log out signal and the systems signals that the user has logged out. (See Figure 6)

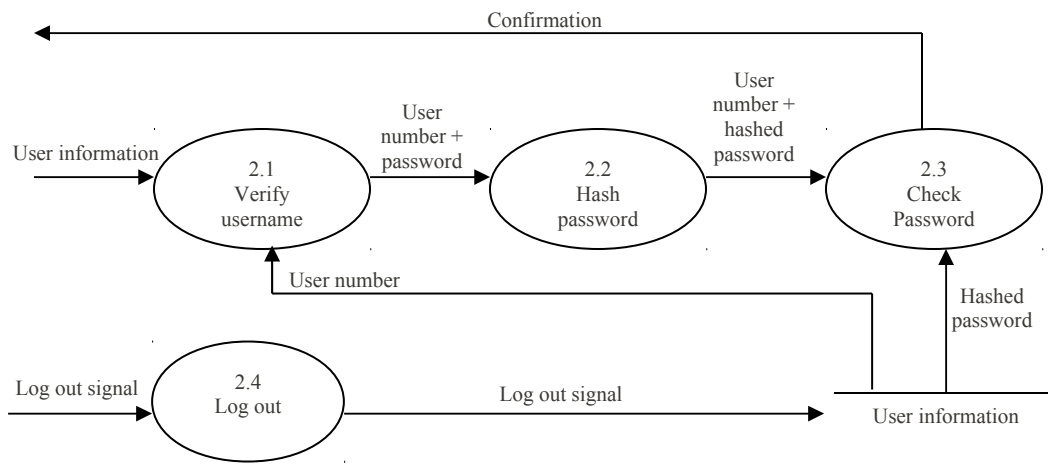


Figure 6 Subexplosion of process, “Logging into the System”, Simulating the RSA Algorithm by Creating a Messaging System

The sending a message process is the encryption process. After a message is requested to be sent the plaintext is changed to a numerical

value with a use of a look up table of ASCII. The plaintext equivalent is encrypted with the use of the public key of the recipient. The cipher is then stored on the message database. (See Figure 7)

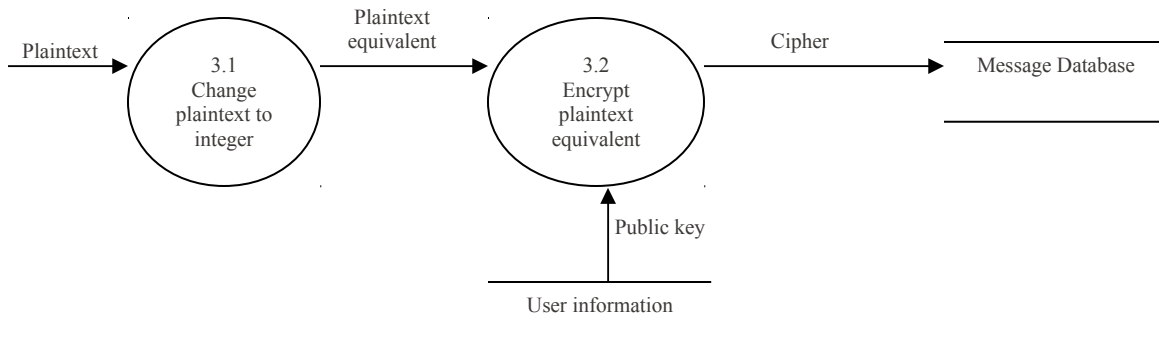


Figure 7 Subexplosion of process, “Sending Message”, Simulating the RSA Algorithm by Creating a Messaging System

The receiving messages process is the decryption process. The process fetches all messages for the user. It then decrypts the cipher to obtain the plaintext equivalent. The plaintext equivalent is converted back to its original form with the use of the ASCII look-up table. This is shown in Figure 8.

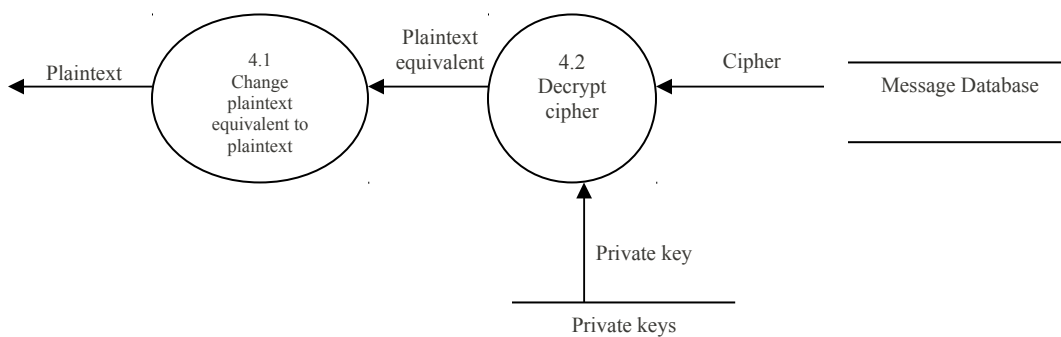


Figure 8 Subexplosion of process, “Receiving Message”, Simulating the RSA Algorithm by Creating a Messaging System

The intercept a message is not a real part of the system. This process simulates the attack by taking a cipher form the message database and decipher it by brute force even with the absence of the proper private key as shown in Figure 9.

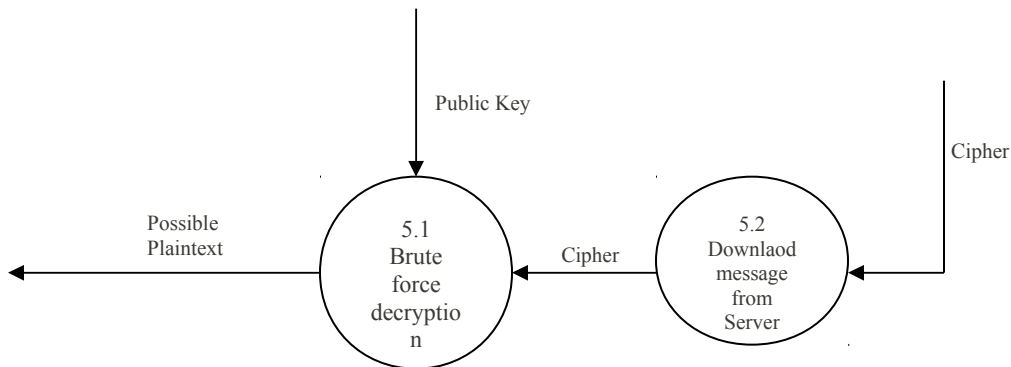


Figure 9 Subexplosion of process, “Intercept Message”, Simulating the RSA Algorithm by Creating a Messaging System

There are three excutables that could be imported by PHP. The first is the Key Generator. It generates a public and private key pair and saves them in 2 separate files. (See Figure 10). The second is the Encryptor. With a text file and the public key file, it creates a cipher file. (See Figure 11) The third is the Decryptor. With the cipher file and a public key file it recreates the text file. (See Figure 12).



Figure 10 Data Flow Diagram of the Key Generator executable, Simulating the RSA Algorithm by Creating a Messaging System

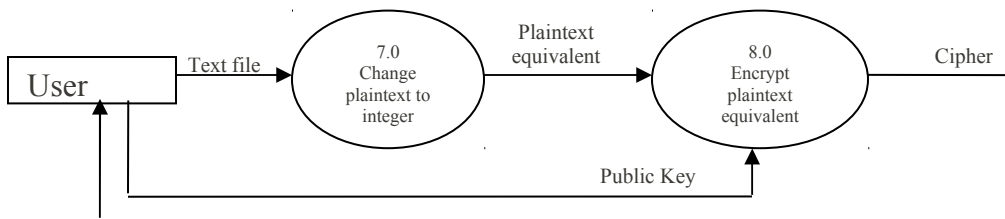


Figure 11 Data Flow Diagram of the Encryptor executable, Simulating the RSA Algorithm by Creating a Messaging System

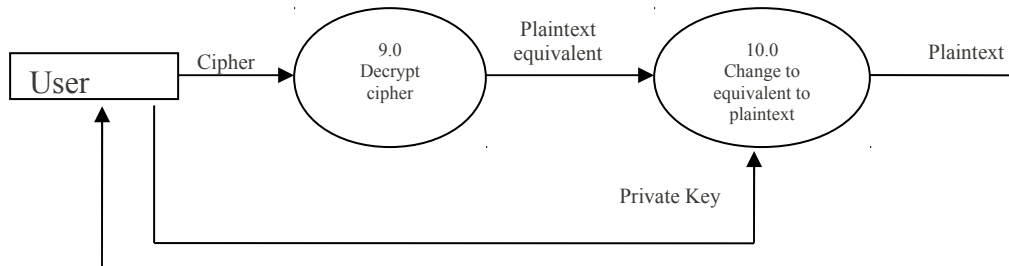


Figure 12 Data Flow Diagram of the Decryptor executable, Simulating the RSA Algorithm by Creating a Messaging System

Data Definition

User Information		
Name	Type	Description
User no	Integer	System generated number that uniquely identifies a user.
User name	Text	Name that uniquely identifies the user.
Hashed Password	Text	Password after it has gone through hashing
Public Key	Integer	Key used to encrypt messages
Messages		
Name	Type	Description
Message no	Integer	Number that uniquely identifies a message
Sender	Integer	The user number of the user who sent the message
Recipient	Integer	The user number to which the message is sent.
Cipher	Integer	Encrypted text

Private keys		
Name	Type	Description
User no	Integer	Number that identifies uniquely a user.
Private Key	Integer	Key used during decryption

TECHNICAL ARCHITECTURE

The will use a client-server architecture. The server and the medium are assumed to be insecure, but the client is assumed to be secure. For this reason, vital information is not transmitted or stored in the server in its raw form. The private key store is located in the client side because the private keys are needed to be kept secret. This also implies that the key generation is done in the client side. The password is another vital information that is needed to be kept secret, however it is needed to be stored in the server. Therefore it is first hashed before it is transmitted and stored on the server. This implies that the hashing process is done on the client side. Messages may contain vital information that is needed to be kept secret. Messages are encrypted and then sent to the server. It is stored in the server if it could not be delivered (recipient is not logged). The encryption is also done in the client side. The decryption process must be done in the client-side; otherwise the message would have been transmitted to client from the server in its raw form, which would have defeated the purpose of the encryption.

V. RESULTS

A. Server Application

The system has three main components. The first of the components is the Server Application. The server is needed for the other 2 components to run. It handles the storing and request for retrieval of non-vital or encrypted information.

B. Client Application

The second component is the Client Application. The client requests the server for data and then processes it. It also stores vital or non-encrypted information.

Connection Screen

This screen asks the user for the IP Address of the server as illustrated in Figure 13. It would then try to connect to the server. If no connection is found an error message is shown. If a connection is found, it screen transfers to the login screen.

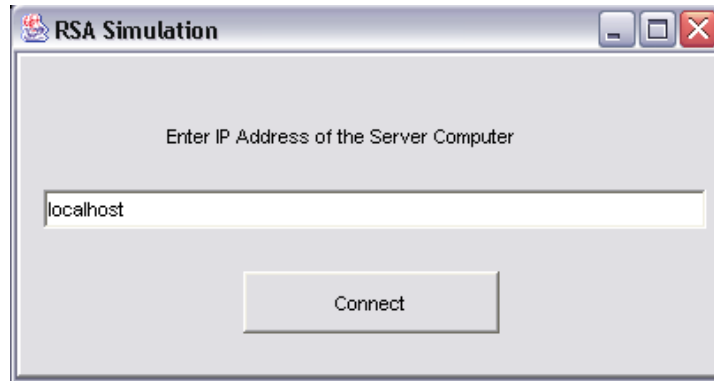


Figure 13 Connection Screen of the Client Application, Simulating the RSA Algorithm by Creating a Messaging System

Login Screen

This screen asks for the user name and password (See Figure 14). Once the Enter button is pressed, first hashes the password. It then requests the server for user information (which includes the hashed password) and it checks if the password is correct. If no user with the specified user name was found, or the password was incorrect it will show an error message. Once the user name and password are verified, the component shows the Main Screen. The exit button is for the User to end the program. The New User button will show the New User Screen.

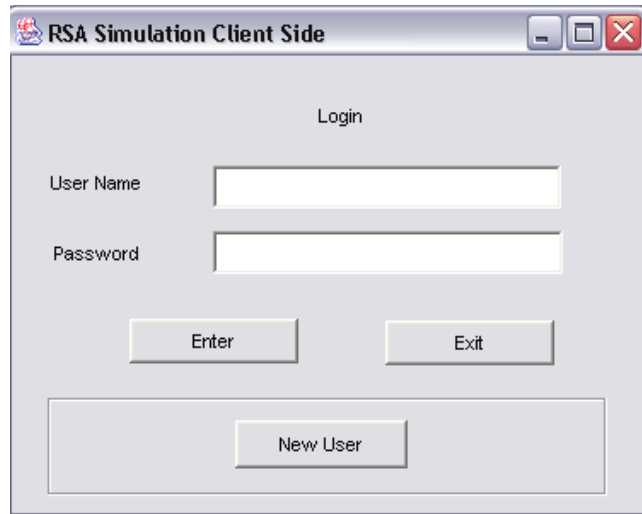


Figure 14 Login Screen of the Client Application, Simulating the RSA Algorithm by Creating a Messaging System

New User Screen

This screen shows allows to user to create a new user in the system as shown in Figure 15. Once the Enter Button is pressed, it requests the server to check if the user name exists; if it exists, it will show an error message. If the user name does not exist, then it hashes the password, creates the public and private keys and stores the private key in the client side while the user name, hashed password and public key will be stored in the server. It then returns the user to the Log in screen. The cancel button returns the user to the log in screen without any operations.

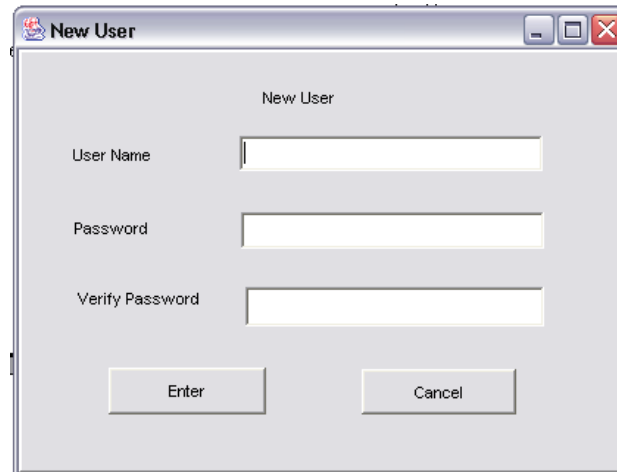


Figure 15 New User Screen of the Client Application, Simulating the RSA Algorithm by Creating a Messaging System

Main Screen

The main screen allows the user to send messages as seen in Figure 16. The combo box lists the all the users in the system. The text area is where the user types his message. Once the send button is pressed, it requests the server for the public key of the recipient of the message. It then encrypts it using the public key of the recipient and stores the message in the server. The Reset button erases the message in the text area without sending it. The Receive messages button shows the Receive messages screen. The Log out button takes the user to the Log in screen where he can log in again or exit the program.

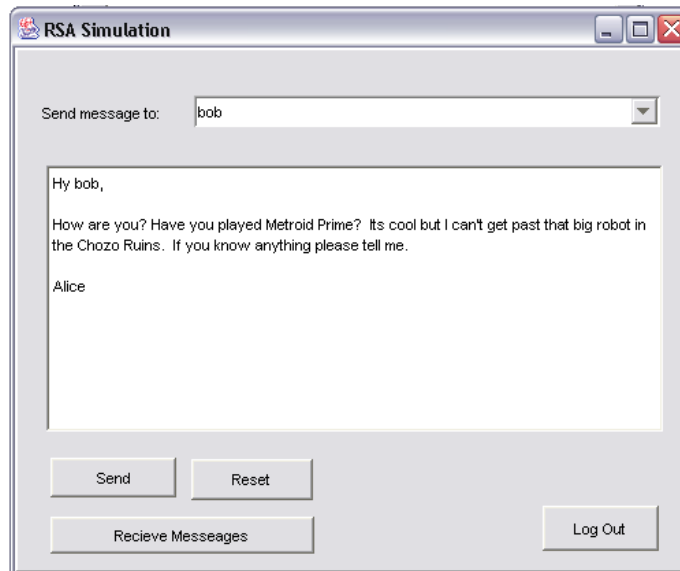


Figure 16 Main Screen of the Client Application, Simulating the RSA Algorithm by Creating a Messaging System

Read Messages Screen

This screen retrieves a message from the server and decrypts it with the private key of the user (See Figure 17). The “read next” button reloads the screen to retrieve the next message. If no new messages was found on the server The “read next” becomes the “back” button that allows the user to go back to the main screen.

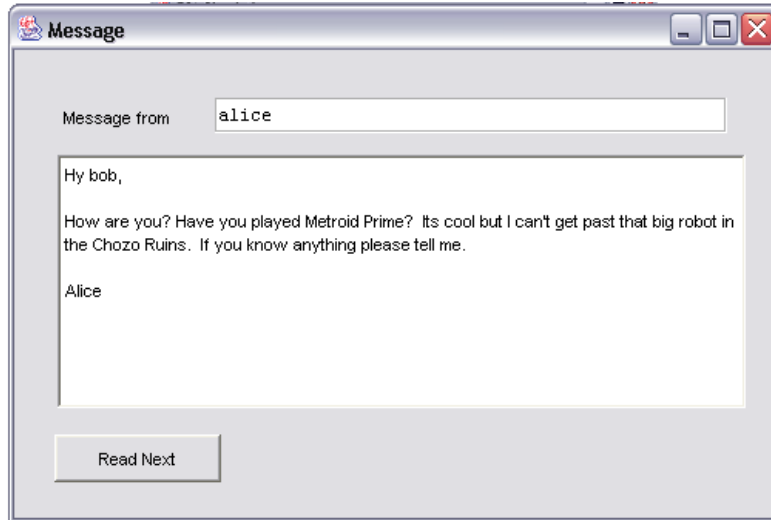


Figure 17 Read New Messages Screen of the Client Application, Simulating the RSA Algorithm by Creating a Messaging System

C. Attacker Component

This component simulates an attacker to the system. Its main intention is read encrypted messages with out the proper private key. This component assumes that the messages is encrypted using the RSA Algorithm, the component can retrieve the public keys of the user and the cipher of the messages.

Connection Screen

This screen allows the attacker to connect to the server (See Figure 18). If no connection was found, an error message appears. If a connection was found, the attacker is taken to the main screen.

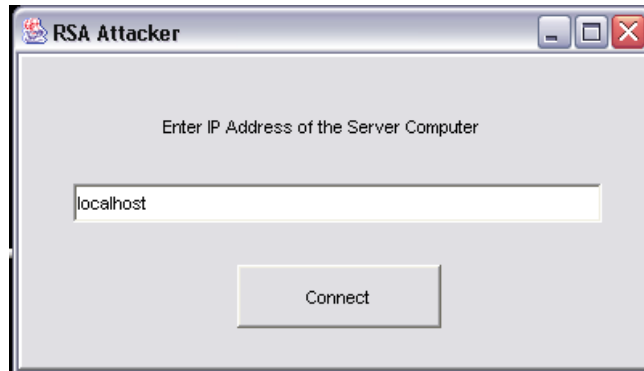


Figure 18 Connection Screen of the Attacker Component, Simulating the RSA Algorithm by Creating a Messaging System

Main Screen

The main screen shown in Figure 19 (left side) allows the attacker to randomly get a message and its corresponding information. If the message has been read then it has been deleted from the server and therefore cannot be attacked as shown on Figure 19 (right side). The exit button allows the attacker to exit the component.

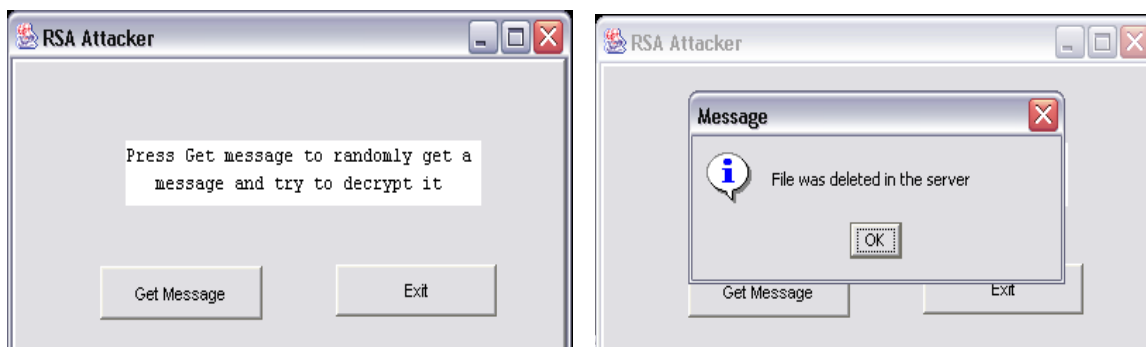


Figure 19 Main Screen of the Attacker Component, Simulating the RSA Algorithm by Creating a Messaging System

Read Messages Screen

On loading the screen, it first tries to deduce the private key of the recipient through his public key by brute force. Then it decrypts cipher using the deduced private key and a block size of one (the real block size is 6). Then it is saved to a file. It then continues to decrypt the cipher while increasing the block size. This is done until the message is decrypted using the block size of ten. The “read next” button opens the file of the probable plaintext with a corresponding increase in block size. The “read previous” button opens the file of the probable plaintext with a corresponding decrease in block size (See Figure 20).



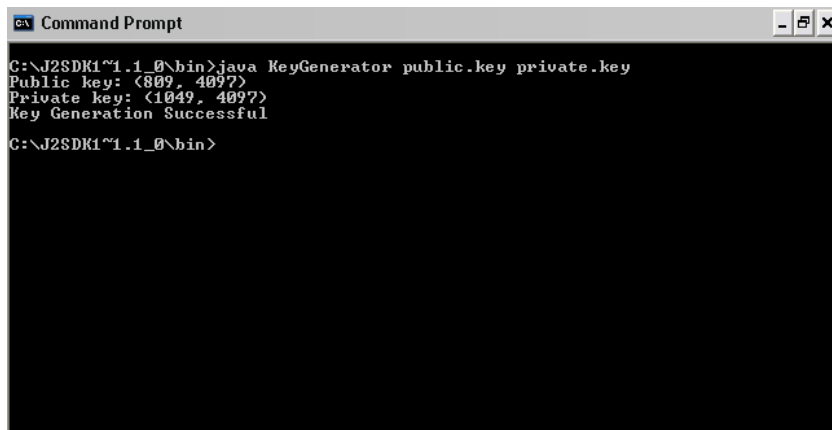
Figure 20 Read Messages Screen of the Attacker Component, Simulating the RSA Algorithm by Creating a Messaging System

D. Executables

There are three executables, Key Generator, Encryptor, Decryptor. All of these executables can be run on a command line.

Key Generator

This command creates a pair of public and private keys. It also saves it in different files as specified by the user (See Figure 21).

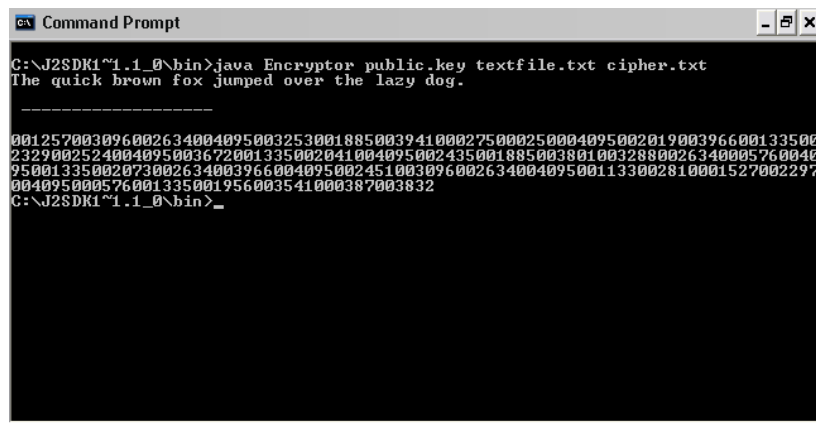


```
Command Prompt
C:\J2SDK1~1\1_0\bin>java KeyGenerator public.key private.key
Public key: <809, 4097>
Private key: <1049, 4097>
Key Generation Successful
C:\J2SDK1~1\1_0\bin>
```

Figure 21 Using the Key Generator in a command line, Simulating the RSA Algorithm by Creating a Messaging System

Encryptor

This command creates a cipher from a text file and a public key file. It then saves the cipher file as specified by the user. (See Figure 22).



```
Command Prompt
C:\J2SDK1~1\1_0\bin>java Encryptor public_key textfile.txt cipher.txt
The quick brown fox jumped over the lazy dog.
-----
00125700309600263400409500325300188500394100027500025000409500201900396600133500
23290025240040950036720013350020410040950024350018850038010032880026340005760040
95001335002073002634003966004095002451003096002634004095001133002810001527002297
004095000576001335001956003541000387003832
C:\J2SDK1~1\1_0\bin>
```

Figure 22 Using the Encryptor in a command line, Simulating the RSA Algorithm by Creating a Messaging System

Decryptor

This command recreates the text from the corresponding private key and cipher file. It then saves the text file as specified by the user (See Figure 23).

```
Command Prompt
C:\J2SDK1~1.1_0\bin>java Decryptor private.key cipher.txt decipher.txt
00008400010400010100003200011300011700010500009900010700003200009800011400011100
01190001100000320001020001110001200000320001060001170001090001120001010001000000
32000111000118000101000114000032000116000104000101000032000108000097000122000121
000032000100000111000103000046000013000010
The quick brown fox jumped over the lazy dog.
C:\J2SDK1~1.1_0\bin>
```

Figure 23 Using the Decryptor in a command line, Simulating the RSA Algorithm by Creating a Messaging System

VI. DISCUSSION

The simulation showed how the RSA algorithm encrypts, decrypts, and generates its keys. The length of the keys determines the speed of the encryption and decryption processes. It also determines the strength of the encryption. As the key sizes increases, the speed of encryption and decryption becomes slower, but the strength of the cipher increases.

The computational time of the key generation is $O(n^4)$ where n is the size of the key [12]. For simulation purposes the length of the keys was only 20-bits. However commercially available cryptosystems using the RSA Algorithm use keys with length of at least 768-bits.

Both the encryption and decryption have a computational time of $O(n^3)$ where n is the size of the key [12]. Likewise, both encryption and decryption is involved in raising a number to a power then getting the modulo with another number. To handle the large numbers a property of modulo was used. It states that if A, B, C are integers then

$$AB \bmod C = (A \bmod C)(B \bmod C) \bmod C$$

This implied that if an integer was multiplied to itself and its modulo was taken and multiplied to itself again for a number of times, then it would

have the same effect as raising a number to a power then take its modulo (For the proof of the property see Appendix 2).

The simulation shows how a message can be transmitted, securely. Since the system uses RSA algorithm (which is an asymmetric key encryption), public keys are stored in the server and private keys are kept secret. This eliminates the need for sharing secret keys [2]. However the connection between public key and private key is hidden with a trap-door function. Trap-door functions are mathematical functions are significantly easier to compute in one direction (the forward direction) than the inverse direction (opposite direction). However the opposite direction can be computed with the knowledge of some value (trap-door value). This means that the private key may be deduced with the use of the public key, by obtaining this trap-door value. With the increase of computational power the trap-door value may be easier to obtain [12].

To simulate the interception of a cipher, the attacker component randomly retrieves a cipher form the server, along with its corresponding information. To simulate the attack, it is assumed that the attacker has a cipher; the public key used to encrypt the cipher and the knowledge that the encryption algorithm used is RSA. Armed with these assumptions the private key needed to decrypt the cipher is

deduced by brute force. Without the knowledge of the block size used to encrypt the message, the attacker decrypts the data again and again using block sizes from one to ten.

To deduce the private key, the attacker must be able to find the prime factors of the public key (trap-door values). The attack uses a crude way of finding the prime factors of a number. This is done by checking if n was divisible by the a number, then incrementing the number until a factor is found. Since the key sizes were small, brute force was enough to get the prime factors. The strength of RSA lies on the fact that factoring is hard. The two best factoring algorithms are Number Field Sieve and Multiple Polynomial Quadratic Seive, which has a computational time of $O(e^{1.9(\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}})$ and $O(e^{(\ln n)^{\frac{1}{2}}(\ln \ln n)^{\frac{1}{2}}})$ respectively. Both of these algorithms has an exponential time complexity. However key generation has a polynomial time complexity [12]. Hence, it is faster to produce keys than to break keys. Therefore it would be impractical to break a key without good factoring algorithms as keys could be changed easily.

Using the processes in the simulation, three Java programs that can run on a command line (or dos) are produced. The first generates a public key, a private key and creates two files to store each key

separately. The second program encrypts a text file using a public key file. The third program decrypts a cipher using a private key file. These three programs could be imported by PHP with the use of the `exec()` and the `passthru()` functions. .

VII. CONCLUSION

A system “Simulating of the Rivest – Shamir – Adleman Algorithm by Creating a Messaging System” was created with the following functionalities:

- 1) simulates the encryption and decryption processes of the algorithm.
- 2) simulates the key generation of the algorithm.
- 3) shows the secure transmission of data over an insecure medium by sending data to a server where the data is exposed to anybody able to access the server.
- 4) simulates an attack by intercepting a cipher and attempt to decrypt it by brute force.

The study also produced three java executables programs that could be runned from command lines (or dos). These executables can be imported by PHP and other scripting languages through their dos shell commands.

VIII. RECOMMENDATIONS

As it was seen a 20-bit key offered some protection, but not total protection. It is recommended for future studies on RSA to have larger key sizes (768-bits).

The study also only showed a brute force attack. However there are many kinds of attacks. It is recommended for future studies to have other kinds of attacks.

Although RSA is one of the most used algorithms, it is not used exclusively. Different encryption algorithms are used in combination to produce a stronger cipher than algorithms used exclusively. One such combination is the RSA-DES. It is recommended for future studies to include combination of algorithms.

The study was limited to the encryption, decryption, key generation, and transmission. It did not include digital signing and verification. It is recommended for future studies to include other aspects of RSA algorithm.

IX. BIBLIOGRAPHY

- [1] Welsh, Dominic. **Codes and Cryptography**. Oxford: Clarendon Press, 1988.
- [2] Data Encryption Page, [<http://www.anusjseth.com/crypto/basics.html>]
- [3] Klima, Vlastimil., Rosa, Tomas. "Attack on Private Signature Keys of the OpenPGP Format, PGP(TM) Programs and Other Applications Compatible with OpenPGP", Cryptology ePrint Archive: Report 2002/076, [<http://eprint.iarc.org/2002/076/>]
- [4] Pfleeger, Charles. **Security in Computing**. New Jersey: Prentice-Hall, 1989.
- [5] Ooi, K.S., Vito, Brain Chin. "Cryptanalysis of S-DES", Cryptology ePrint Archive. [<http://eprint.iacr.org/2002/045/>]
- [6] Schmidt, Dieter. "ABC - A Block Cipher", Cryptology ePrint Archive. [<http://eprint.iacr.org/2002/062/>]
- [7] May, Mike. DES - Example, [<http://www.adeptscience.co.uk/products/mathsim/maple/powertools/cryptography/HTML/DES-Example.html>]
- [8] Elliptic Curve Cryptography, [<http://world.std.com/~dpj/elliptic.html>]
- [9] Gathani, Arnit. "Implementation of Elliptical Curve Cryptography in an Embedded System", [<http://www.cs.rit.edu/~ang6829/cryptography/paper.pdf>]
- [10] Encryption Algorithms, [http://www.mycrypto.net/encryption/crypto_algorithms.html]
- [11] History of Cryptography, [<http://www-cse.stanford.edu/classes/sophomore-college/projects-97/cryptography/history.html>]
- [12] RSA Laboratories Cryptography FAQs Section Index, [<http://www.rsasecurity.com/rsalabs/faq/sections.html>]

Appendix 1

Proof of RSA

To show that an encrypted message with the use a public key can be decrypted with its corresponding private key mathematically, the proof is as follows: If

$$C = M^e \bmod n ,$$

then

$$M = C^d \bmod n .$$

By the substituting C with $M^e \bmod n$

$$M = (M^e \bmod n)^d \bmod n$$

By the definition of mod

$$M = (M^e - sn)^d \bmod n$$

where s is a non-negative integer. By the binomial expansion,

$$M = \left(\sum_{i=0}^d \binom{d}{i} M^{e(d-i)} (-sn)^i \right) \bmod n$$

In rewriting,

$$M = \left(M^{ed} + \sum_{i=1}^d \binom{d}{i} M^{e(d-i)} (-sn)^i \right) \bmod n$$

By the properties of mod

$$M = M^{ed} \bmod n + \sum_{i=1}^d \left(M^{e(d-i)} (-sn)^i \bmod n \right) .$$

Since a mod n = 0 if a is divisible by n and $M^{e(d-i)} (-sn)^i$ is divisible by n,

$$M = M^{ed} \bmod n .$$

Since $ed \bmod (p-1)(q-1) = 1$,

$$(1) M = M^{t(p-1)(q-1)+1} \pmod{n} .$$

By Euler-Fermat identity that states for any prime number p and any positive integer M ,

$$M^{p-1} \pmod{p} = 1 .$$

Therefore,

$$M^{p-1} = rp + 1 .$$

where r is some non-negative integer. By raising both sides by $t(q-1)$ then multiplying M ,

$$M^{(p-1)t(q-1)+1} = (rp + 1)^{t(q-1)} M .$$

By the binomial expansion,

$$M^{t(p-1)(q-1)+1} = M \left(\sum_{i=0}^{t(q-1)} \binom{t(q-1)}{i} (rp)^i \right) .$$

In rewriting,

$$M^{t(p-1)(q-1)+1} = M \left(1 + \sum_{i=0}^{t(q-1)} \binom{t(q-1)}{i} (rp)^i \right)$$

taking the mod p of both sides and using the definition of mod

$$M^{t(p-1)(q-1)+1} \pmod{p} = M \pmod{p} .$$

By the definition of mod,

$$M^{t(p-1)(q-1)+1} - hp = M - jp$$

where h and j are non-negative integers and $h > j$. In rewriting,

$$M^{t(p-1)(q-1)+1} - (h - j)p = M$$

Since $(h-j)$ is a non negative integer and by the definition of mod,

$$M^{t(p-1)(q-1)+1} \pmod{p} = M$$

By the same arguments,

$$M^{t(p-1)(q-1)} \bmod q = M .$$

By the Lemma that states, for any pair of positive integers (x, u) , if $x^u \bmod p = x$ and $x^u \bmod q = x$ then $x^u \bmod pq = x$,

$$M^{t(p-1)(q-1)+1} \bmod pq = M .$$

Since $n = pq$,

$$M^{t(p-1)(q-1)+1} \bmod n = M$$

Substituting this to (1),

$$M = M .$$

Its is now seen that for a public key (e, n) and a private key d the message M could be encrypted using the public key and be decrypted using the corresponding private key.

Appendix 2

Proof of $AB \bmod C = (A \bmod C)(B \bmod C) \bmod C$

Let A, B, C be integers then,

$$(A \bmod C)(B \bmod C) \bmod C$$

using the property of modulus we can rewrite the statement as

$$(A - kC)(B - jC) \bmod C$$

where j, c are positive integers. By performing the multiplication we get

$$(AB - kBC - jAC + kjC^2) \bmod C$$

then we can cancel the addends which are divisible by C. We arrive at the expression

$$AB \bmod C .$$

Appendix 3

Code of Server application

```
import java.io.*;
import java.net.*;
public class Server_Connection
{
    public static void main(String args[])
    {
        Thread t;
        int port = 9001;
        try
        {
            ServerSocket SS = new ServerSocket(port);
            System.out.println("Server Started");
            System.out.println(SS.getLocalSocketAddress());
            while (true)
            {
                Socket s = SS.accept();
                System.out.println("Connected Server");
                Server_Connection_Thread SCT = new
Server_Connection_Thread(s);
                t = new Thread(SCT);
                t.start();
            }
        }
        catch(Exception e)
        {}
    }
}

import java.io.*;
import java.net.*;

public class Server_Connection_Thread implements Runnable
{
    private Socket s;
    private InputStream sis;
    private OutputStream sos;
    private ObjectInputStream sois;
    private ObjectOutputStream soos;

    Server_Connection_Thread(Socket a)
    {
        try
        {
            s = a;
            sis = a.getInputStream();
            sois = new ObjectInputStream(sis);
            sos = a.getOutputStream();
            soos = new ObjectOutputStream(sos);
            System.out.println("New Thread Created");
        }
        catch(Exception e){System.out.print(e);}
    }

    public void Check_If_Key_Exists()
    {
        boolean found = false;
        try
        {
            int n = sois.readInt();
            File users = new File("User_Information.db");
            FileInputStream fis = new FileInputStream(users);
            ObjectInputStream ois = new ObjectInputStream(fis);
```

```

        boolean check = true;
        while(check && !found)
        {
            try
            {
                User_Info temp = (User_Info) ois.readObject();
                if (temp.Public_Key.n == n)
                {
                    found = true;
                }
            }
            catch(Exception e){check = false;}
        }
        ois.close();
        fis.close();
        soos.writeBoolean(found);
        soos.flush();
    }
    catch(Exception e){}
}

public User_Info Find_User(String name)
{
    boolean found = false;
    User_Info ui = new User_Info();
    try
    {
        File no_users = new File("Number_of_Users.num");
        File user_infomration = new File("User_Information.db");
        FileInputStream fis = new FileInputStream(no_users);
        ObjectInputStream ois = new ObjectInputStream(fis);
        int num = ois.readInt();
        ois.close();
        fis.close();
        fis = new FileInputStream(user_infomration);
        ois = new ObjectInputStream(fis);
        int x = 0;
        while ((x != num) && !found)
        {
            ui = (User_Info) ois.readObject();
            if (ui.User_Name.compareTo(name) == 0)
            {
                found = true;
            }
            x++;
        }
        ois.close();
        fis.close();
    }
    catch(Exception e){}
    if (!found)
    {
        ui = new User_Info(0, " ", " ", null);
    }
    return ui;
}

public void Request_User_Info()
{
    try
    {
        String name = (String) sois.readObject();
        System.out.println("User Info of: "+ name+" has been requested");
        User_Info ui = Find_User(name);
        soos.writeObject(ui);
        soos.flush();
    }
    catch(Exception e){}
}
}

```

```

public void Save_User_Info()
{
    try
    {
        User_Info ui = (User_Info) sois.readObject();
        System.out.print("User Number: "+ ui.User_Number);
        System.out.print(" Username: "+ui.User_Name+" Password: ");
        System.out.print(ui.Hashed_Password+" Public Key: ");
        System.out.println(ui.Public_Key.d+ " ", "+ui.Public_Key.n);
        User_Info pl = Find_User(ui.User_Name);
        if (pl.User_Number == 0)
        {
            File old_no_users = new File("Number_of_Users.num");
            File new_no_users = new File("templ.jsp");
            FileInputStream fis = new FileInputStream(old_no_users);
            ObjectInputStream ois = new ObjectInputStream(fis);
            FileOutputStream fos = new FileOutputStream(new_no_users);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            ui.User_Number = ois.readInt() + 1;
            oos.writeInt(ui.User_Number);
            oos.close();
            fos.close();
            ois.close();
            fis.close();
            old_no_users.delete();
            new_no_users.renameTo(new File("Number_of_Users.num"));
            File oldFile = new File("User_Information.db");
            File newFile = new File("Temp2.jsp");
            fis = new FileInputStream(oldFile);
            ois = new ObjectInputStream(fis);
            fos = new FileOutputStream(newFile);
            oos = new ObjectOutputStream(fos);
            for (int counter = 1; counter != ui.User_Number; counter++)
            {
                User_Info tempnode = (User_Info) ois.readObject();
                oos.writeObject(tempnode);
            }
            oos.writeObject(ui);
            oos.close();
            ois.close();
            fis.close();
            fos.close();
            oldFile.delete();
            newFile.renameTo(new File("User_Information.db"));
            soos.writeBoolean(true);
            soos.flush();
        }
        else
        {
            soos.writeBoolean(false);
            soos.flush();
        }
    }
    catch(Exception e){System.out.println(e);}
}

public void Save_Message()
{
    try
    {
        Message_Info mi = (Message_Info)sois.readObject();
        File oldnm = new File("Number_of_Messages.num");
        File newnm = new File("temp3.jsp");
        FileInputStream fis = new FileInputStream(oldnm);
        ObjectInputStream ois = new ObjectInputStream(fis);
        FileOutputStream fos = new FileOutputStream(newnm);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        mi.Message_Number = ois.readInt() +1;
        oos.writeInt(mi.Message_Number);
        oos.close();
        ois.close();
    }
}

```



```

        fis.close();
        fos.close();
        oldnm.delete();
        newnm.renameTo(new File("Number_of_Messages.num"));
        System.out.println("Message Number: "+mi.Message_Number);
        File oldFile = new File("Messages.db");
        File newFile = new File ("temp5.jmp");
        File archive = new File ("archive.jmp");
        FileOutputStream fos2 = new FileOutputStream(archive);
        ObjectOutputStream oos2 = new ObjectOutputStream(fos2);
        fis = new FileInputStream(oldFile);
        fos = new FileOutputStream(newFile);
        ois = new ObjectInputStream(fis);
        oos = new ObjectOutputStream(fos);
        boolean check = true;
        while(check)
        {
            try
            {
                Message_Info tempnode = (Message_Info)
ois.readObject();

                oos.writeObject(tempnode);
                oos2.writeObject(tempnode);
            }
            catch(Exception e){check = false;}
        }
        oos.writeObject(mi);
        oos2.writeObject(mi);
        oos2.close();
        fos2.close();
        oos.close();
        ois.close();
        fis.close();
        fos.close();
        oldFile.delete();
        newFile.renameTo(new File("Messages.db"));
        File unsent_message = new File(String.valueOf(mi.Message_Number)
+"mes");

        fos = new FileOutputStream(unsent_message);
        PrintStream ps = new PrintStream(fos);
        char c = 'a';
        check = true;
        while(check)
        {
            try
            {
                c = sois.readChar();
                System.out.print(c);
                ps.print(c);
            }
            catch(Exception e){check = false;}
        }
        ps.close();
        fos.close();
    }
    catch(Exception e)
    {
        System.out.println(e);
        System.out.print(e.getMessage());
        System.out.print(e.getLocalizedMessage());
    }
}

public void Request_Message()
{
    try
    {
        Message_Info mi = new Message_Info();
        int reciever = sois.readInt();
        File messages = new File("Messages.db");
        File newMes = new File("temp6.jmp");
    }
}

```

```

FileInputStream fis = new FileInputStream(messages);
ObjectInputStream ois = new ObjectInputStream(fis);
FileOutputStream fos = new FileOutputStream(newMes);
ObjectOutputStream oos = new ObjectOutputStream(fos);
boolean check = true, found = false;
while(check && !found)
{
    try
    {
        Message_Info temp = (Message_Info) ois.readObject();
        if(temp.Reciever == reciever)
        {
            mi = temp;
            found = true;
        }
        else
        {
            oos.writeObject(temp);
        }
    }
    catch(Exception e){check = false;}
}
check = true;
while(check)
{
    try
    {
        Message_Info temp = (Message_Info) ois.readObject();
        oos.writeObject(temp);
    }
    catch(Exception e){check = false;}
}
ois.close();
fis.close();
oos.close();
fos.close();
soos.writeBoolean(found);
soos.flush();
if (found)
{
    soos.writeObject(mi);
    soos.flush();
    char[] c = new char[1];
    File mes = new File(String.valueOf(mi.Message_Number)
+ ".mes");

    FileReader fr = new FileReader(mes);
    BufferedReader br = new BufferedReader(fr);
    while(br.read(c, 0, 1) != -1)
    {
        soos.writeChar(c[0]);
        soos.flush();
    }
    br.close();
    fr.close();
    mes.delete();
    messages.delete();
    newMes.renameTo(new File("Messages.db"));
}
else
{
    newMes.delete();
}
}
catch(Exception e){System.out.println(e);}
}

public void Request_All_Users()
{
    try
    {
        File num = new File("Number_of_Users.num");

```

```

        FileInputStream fis = new FileInputStream(num);
        ObjectInputStream ois = new ObjectInputStream(fis);
        int x = ois.readInt();
        ois.close();
        fis.close();
        soos.writeInt(x);
        soos.flush();
        File users = new File("User_Information.db");
        fis = new FileInputStream(users);
        ois = new ObjectInputStream(fis);
        for(int y = 0; y != x; y++)
        {
            User_Info temp = (User_Info) ois.readObject();
            soos.writeObject(temp);
            soos.flush();
        }
        ois.close();
        fis.close();
    }
    catch(Exception e){}
}

public void get_Archive()
{
    try
    {
        File num = new File("Number_of_Messages.num");
        FileInputStream fis = new FileInputStream(num);
        ObjectInputStream ois = new ObjectInputStream(fis);
        int x = ois.readInt();
        ois.close();
        fis.close();
        soos.writeInt(x);
        soos.flush();
        File mes = new File("Archive.jmp");
        fis = new FileInputStream(mes);
        ois = new ObjectInputStream(fis);
        for(int y = 0; y!= x; y++)
        {
            Message_Info mi = (Message_Info) ois.readObject();
            soos.writeObject(mi);
            soos.flush();
        }
        ois.close();
        fis.close();
    }
    catch(Exception e){}
}

public void get_Message_from_no()
{
    try
    {
        int message_no = sois.readInt();
        File message = new File(String.valueOf(message_no)+".mes");
        FileReader fr = new FileReader(message);
        BufferedReader br = new BufferedReader(fr);
        char[] c = new char[1];
        soos.writeBoolean(true);
        while(br.read(c, 0, 1) != -1)
        {
            soos.writeChar(c[0]);
            soos.flush();
        }
        br.close();
        fr.close();
    }
    catch(Exception e)
    {
        try

```

```

        {
            soos.writeBoolean(false);
            soos.flush();
        }
        catch(Exception ex){}
    }
}

public void Get_User_Name()
{
    try
    {
        int x = sois.readInt();
        boolean check = true, found = false;
        String name = " ";
        File users = new File("User_Information.db");
        FileInputStream fis = new FileInputStream(users);
        ObjectInputStream ois = new ObjectInputStream(fis);
        while(check && !found)
        {
            try
            {
                User_Info ui = (User_Info) ois.readObject();
                if (x == ui.User_Number)
                {
                    name = ui.User_Name;
                    found = true;
                }
            }
            catch(Exception e){check = false;}
        }
        ois.close();
        fis.close();
        soos.writeObject(name);
        soos.flush();
    }
    catch(Exception e){}
}

public void run()
{
    try
    {
        System.out.println("Connected");
        int command = sois.readInt();
        switch(command)
        {
            case 0:
            {
                System.out.println(sois.readUTF() +" Connected");
                break;
            }
            case 1:
            {
                System.out.println("Save User Info");
                boolean bool = true;
                while(bool)
                {
                    Check_If_Key_Exists();
                    bool = sois.readBoolean();
                }
                Save_User_Info();
                break;
            }
            case 2:
            {
                System.out.println("Request Message");
                Request_Message();
                break;
            }
        }
    }
}

```

```

        }
        case 3:
        {
                Request_User_Info();
                break;
        }
        case 4:
        {
                System.out.println("Request all Users Information");
                Request_All_Users();
                break;
        }
        case 5:
        {
                System.out.println("Save Message");
                Save_Message();
                break;
        }
        case 6:
        {
                System.out.println("Get User Name");
                Get_User_Name();
                break;
        }
        case 7:
        {
                System.out.println("Getting archive");
                Request_All_Users();
                get_Archive();
                break;
        }
        case 8:
        {
                get_Message_from_no();
                break;
        }
        }
        sois.close();
        soos.close();
        sis.close();
        sos.close();
        s.close();
        System.out.println("Connection Ended");
    }
    catch(Exception e){}
}

import java.io.*;
public class Key implements Serializable
{
    public int d, n;
    public boolean type;
    public Key (String a, int x, int y)
    {
        type = (a == "public"); // TRUE = PUBLIC, FALSE = PRIVATE
        d = x;
        n = y;
    }
}

import java.io.*;
public class Message_Info implements Serializable
{
    int Message_Number;
    int Sender;
    int Reciever;
    Message_Info(){}
    Message_Info(int a, int b, int c)
    {

```

```
        Message_Number = a;
        Sender = b;
        Reciever = c;
    }
}

import java.io.*;
import java.lang.*;
public class User_Info implements Serializable
{
    int User_Number;
    String User_Name;
    String Hashed_Password;
    Key Public_Key;
    User_Info(int a, String b, String c, Key d)
    {
        User_Name = b;
        Hashed_Password = c;
        Public_Key = d;
        User_Number = a;
    }
    User_Info(){}
}
```

Appendix 4

Code of the Client Application

```
import java.io.*;
public class Key implements Serializable
{
    public int d, n;
    public boolean type;
    public Key (String a, int x, int y)
    {
        type = (a == "public"); // TRUE = PUBLIC, FALSE = PRIVATE
        d = x;
        n = y;
    }
}

import java.io.*;
public class Message_Info implements Serializable
{
    int Message_Number;
    int Sender;
    int Reciever;
    Message_Info(){}
    Message_Info(int a, int b, int c)
    {
        Message_Number = a;
        Sender = b;
        Reciever = c;
    }
}

import java.io.*;
import java.lang.*;
public class User_Info implements Serializable
{
    int User_Number;
    String User_Name;
    String Hashed_Password;
    Key Public_Key;
    User_Info(int a, String b, String c, Key d)
    {
        User_Name = b;
        Hashed_Password = c;
        Public_Key = d;
        User_Number = a;
    }
    User_Info(){}
}

import javax.swing.UIManager;
import java.awt.*;

public class Connection_Screen
{
    private boolean packFrame = false;

    //Construct the application
    public Connection_Screen()
    {
        Connection_Frame frame = new Connection_Frame();
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else
        {
            frame.validate();
        }
    }
}
```

```

//Center the window
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
Dimension frameSize = frame.getSize();
if (frameSize.height > screenSize.height)
{
    frameSize.height = screenSize.height;
}
if (frameSize.width > screenSize.width)
{
    frameSize.width = screenSize.width;
}
frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
frame.setVisible(true);
}
//Main method
public static void main(String[] args)
{
    try
    {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    new Connection_Screen();
}
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.lang.*;
import java.net.*;
import java.io.*;
public class Connection_Frame extends JFrame
{
    private JPanel contentPane;
    private JButton Button_Connect = new JButton();
    private JTextField TextField_IPAddress = new JTextField();
    private JLabel Label_IPAddress = new JLabel();

    //Construct the frame
    public Connection_Frame()
    {
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try
        {
            jbInit();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
    //Component initialization
    private void jbInit() throws Exception
    {
        //setIconImage(Toolkit.getDefaultToolkit().createImage(Connection_Frame.class.getResource("[Your Icon]")));
        contentPane = (JPanel) this.getContentPane();
        Button_Connect.setBounds(new Rectangle(128, 123, 145, 35));
        Button_Connect.setText("Connect");
        Button_Connect.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                Button_Connect_actionPerformed(e);
            }
        });
    }
}

```



```

contentPane.setLayout(null);
this.setSize(new Dimension(406, 217));
this.setTitle("RSA Simulation");
contentPane.setBorder(BorderFactory.createRaisedBevelBorder());
TextField_IPAddress.setText("localhost");
TextField_IPAddress.setBounds(new Rectangle(15, 77, 375, 22));
Label_IPAddress.setText("Enter IP Address of the Server Computer");
Label_IPAddress.setBounds(new Rectangle(85, 34, 228, 24));
contentPane.add(TextField_IPAddress, null);
contentPane.add(Label_IPAddress, null);
contentPane.add(Button_Connect, null);
}
//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        System.exit(0);
    }
}
public void Button_Connect_actionPerformed(ActionEvent e)
{
    try
    {
        String[] str = new String[1];
        str[0] = TextField_IPAddress.getText();
        Socket s = new Socket(str[0], 9001);
        OutputStream os = s.getOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(os);
        InputStream is = s.getInputStream();
        ObjectInputStream ois = new ObjectInputStream(is);
        String utf = s.getInetAddress().getCanonicalHostName();
        oos.writeInt(0);
        oos.flush();
        oos.writeUTF(utf);
        oos.flush();
        oos.close();
        ois.close();
        os.close();
        is.close();
        s.close();
        Login_Screen.main(str);
        this.dispose();
    }
    catch(Exception ex)
    {
        JOptionPane.showMessageDialog(null, "No Connection of that address");
        TextField_IPAddress.setText("localhost");
    }
}
}

import javax.swing.UIManager;
import java.awt.*;
import java.lang.*;
public class Login_Screen
{
    private boolean packFrame = false;

    //Construct the application
    public Login_Screen(String IPAddress)
    {
        Login_Frame frame = new Login_Frame(IPAddress);
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else

```

```

    {
        frame.validate();
    }
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
    {
        frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width)
    {
        frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
}
//Main method
public static void main(String[] args)
{
    try
    {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    new Login_Screen(args[0]);
}
}

```

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.io.*;

```

```

public class Login_Frame extends JFrame
{
    private JPanel contentPane;
    private JLabel Label_Login = new JLabel();
    private JLabel Label_Username = new JLabel();
    private JLabel Label_Password = new JLabel();
    private JPanel Panell = new JPanel();
    private JButton Button_Exit = new JButton();
    private JButton Button_Enter = new JButton();
    private JButton Button_NewUser = new JButton();
    private JTextField TextField_Username = new JTextField();
    private JPasswordField PasswordField_Password = new JPasswordField();
    private String IPAddress;
    //Construct the frame
    public Login_Frame(String str)
    {
        IPAddress = str;
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try
        {
            jbInit();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
    //Component initialization
    private void jbInit() throws Exception
    {

```

```

        //setIconImage(Toolkit.getDefaultToolkit().createImage(Login_Framw.class.getResource("[
Your Icon"]));
        contentPane = (JPanel) this.getContentPane();
        Label_Login.setText("Login");
        Label_Login.setBounds(new Rectangle(180, 26, 36, 17));
        contentPane.setLayout(null);
        this.setSize(new Dimension(379, 300));
        this.setTitle("RSA Simulation Client Side");
        Label_Username.setText("User Name");
        Label_Username.setBounds(new Rectangle(23, 65, 70, 16));
        Label_Password.setText("Password");
        Label_Password.setBounds(new Rectangle(25, 102, 70, 25));
        Panell.setBorder(BorderFactory.createEtchedBorder());
        Panell.setBounds(new Rectangle(21, 199, 328, 57));
        Panell.setLayout(null);
        Button_Exit.setBounds(new Rectangle(219, 154, 99, 26));
        Button_Exit.setText("Exit");
        Button_Exit.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                Button_Exit_actionPerformed(e);
            }
        });
        Button_Enter.setBounds(new Rectangle(69, 153, 99, 26));
        Button_Enter.setText("Enter");
        Button_Enter.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                Button_Enter_actionPerformed(e);
            }
        });
        Button_NewUser.setBounds(new Rectangle(110, 13, 101, 28));
        Button_NewUser.setText("New User");
        Button_NewUser.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                Button_NewUser_actionPerformed(e);
            }
        });
        TextField_Username.setBounds(new Rectangle(118, 64, 204, 25));
        PasswordField_Password.setBounds(new Rectangle(118, 102, 205, 25));
        contentPane.add(Label_Login, null);
        contentPane.add(Button_Exit, null);
        contentPane.add(Button_Enter, null);
        contentPane.add(Label_Username, null);
        contentPane.add(Label_Password, null);
        contentPane.add(Panell, null);
        Panell.add(Button_NewUser, null);
        contentPane.add(PasswordField_Password, null);
        contentPane.add(TextField_Username, null);
    }
    //Overridden so we can exit when window is closed
    protected void processWindowEvent(WindowEvent e)
    {
        super.processWindowEvent(e);
        if (e.getID() == WindowEvent.WINDOW_CLOSING)
        {
            System.exit(0);
        }
    }

    void Button_Exit_actionPerformed(ActionEvent e)
    {
        System.exit(0);
    }

    void Button_NewUser_actionPerformed(ActionEvent e)
    {

```

```

        String[] str = new String[1];
        str[0] = IPAddress;
        NewUser_Screen.main(str);
        this.dispose();
    }

    void Button_Enter_actionPerformed(ActionEvent e)
    {
        String Username = TextField_Username.getText();
        String Password = new String>PasswordField_Password.getPassword();
        if ((Username.compareTo("") == 0) || (Password.compareTo("") == 0))
        {
            JOptionPane.showMessageDialog(null, "Please fill the fields correctly");
            System.out.print("Error");
        }
        else
        {
            try
            {
                String Hashed = Math_Functions.hash_function>Password);
                Socket s = new Socket(IPAddress, 9001);
                System.out.println("Connected");
                OutputStream os = s.getOutputStream();
                InputStream is = s.getInputStream();
                ObjectOutputStream oos = new ObjectOutputStream(os);
                ObjectInputStream ois = new ObjectInputStream(is);
                oos.writeInt(3);
                oos.flush();
                oos.writeObject(Username);
                oos.flush();
                User_Info ui = (User_Info) ois.readObject();
                if (ui.Hashed_Password.compareTo(Hashed) == 0)
                {
                    String[] str = new String[2];
                    str[0] = String.valueOf(ui.User_Number);
                    str[1] = IPAddress;
                    Main_Screen.main(str);
                    this.dispose();
                }
                else
                {
                    JOptionPane.showMessageDialog(null, "Incorrect Password\n
or \nDoes Not Exist");
                    System.out.print("Incorrect");
                }
                oos.close();
                os.close();
                ois.close();
                is.close();
                s.close();
            }
            catch(Exception el){System.out.print(el);}
        }
    }
}

import javax.swing.UIManager;
import java.awt.*;

public class Main_Screen
{
    private boolean packFrame = false;

    //Construct the application
    public Main_Screen(int i, String str)
    {
        Main_Frame frame = new Main_Frame(i, str);
    }
}

```

```

//Validate frames that have preset sizes
//Pack frames that have useful preferred size info, e.g. from their layout
if (packFrame)
{
    frame.pack();
}
else
{
    frame.validate();
}
//Center the window
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
Dimension frameSize = frame.getSize();
if (frameSize.height > screenSize.height)
{
    frameSize.height = screenSize.height;
}
if (frameSize.width > screenSize.width)
{
    frameSize.width = screenSize.width;
}
frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
frame.setVisible(true);
}
//Main method
public static void main(String[] args)
{
    Integer x = new Integer(args[0]);

    try
    {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    new Main_Screen(x.intValue(), args[1]);
}
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import java.io.*;
import java.net.*;

```

```

public class Main_Frame extends JFrame
{
    private JPanel contentPane;
    private JComboBox ComboBox_Reciever = new JComboBox();
    private JLabel Label_Send_Message = new JLabel();
    private TitledBorder titledBorder1;
    private JButton Button_Send = new JButton();
    private JButton Button_Reset = new JButton();
    private JButton Button_Exit = new JButton();
    private JButton Button_ReceiveMessages = new JButton();
    private JScrollPane ScrollPane = new JScrollPane();
    private JEditorPane EditorPane_Message = new JEditorPane();
    private int SenderNumber;
    private String IPAddress;
    //Construct the frame
    public Main_Frame(int x, String str)
    {
        SenderNumber = x;
        IPAddress = str;
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try

```

```

    {
        jbInit();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
//Component initialization
private void Fill_Combo_Box()
{
    try
    {
        Socket s = new Socket(IPAddress, 9001);
        OutputStream os = s.getOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(os);
        InputStream is= s.getInputStream();
        ObjectInputStream ois = new ObjectInputStream(is);
        oos.writeInt(4);
        oos.flush();
        int x = ois.readInt();
        for(int y = 0; y != x; y++)
        {
            User_Info ui = (User_Info) ois.readObject();
            if (SenderNumber != ui.User_Number)
            {
                ComboBox_Reciever.addItem(ui.User_Name);
            }
        }
        oos.close();
        ois.close();
        os.close();
        is.close();
    }
    catch(Exception ex) {}
}

private void jbInit() throws Exception
{
    //setIconImage(Toolkit.getDefaultToolkit().createImage(Main_Frame.class.getResource("Y
our Icon"))));
    contentPane = (JPanel) this.getContentPane();
    titledBorder1 = new TitledBorder("");
    contentPane.setLayout(null);
    this.setSize(new Dimension(501, 417));
    this.setTitle("RSA Simulation");
    ComboBox_Reciever.setBorder(BorderFactory.createLoweredBevelBorder());
    ComboBox_Reciever.setBounds(new Rectangle(131, 35, 341, 23));
    Label_Send_Message.setText("Send message to:");
    Label_Send_Message.setBounds(new Rectangle(20, 36, 104, 22));
    contentPane.setBorder(BorderFactory.createRaisedBevelBorder());
    Button_Send.setBounds(new Rectangle(26, 299, 92, 29));
    Button_Send.setText("Send");
    Button_Send.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            Button_Send_actionPerformed(e);
        }
    });
    Button_Reset.setBounds(new Rectangle(129, 300, 89, 30));
    Button_Reset.setText("Reset");
    Button_Reset.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            Button_Reset_actionPerformed(e);
        }
    });
    Button_Exit.setBounds(new Rectangle(386, 334, 85, 33));
}

```

```

Button_Exit.setText("Log Out");
Button_Exit.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        Button_Exit_actionPerformed(e);
    }
});
Button_ReceiveMessages.setBounds(new Rectangle(26, 342, 193, 27));
Button_ReceiveMessages.setText("Recieve Messeages");
Button_ReceiveMessages.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        Button_ReceiveMessages_actionPerformed(e);
    }
});
ScrollPane.setBounds(new Rectangle(23, 86, 451, 194));
contentPane.add(ComboBox_Reciever, null);
contentPane.add(Label_Send_Message, null);
contentPane.add(Button_Send, null);
contentPane.add(Button_Reset, null);
contentPane.add(Button_ReceiveMessages, null);
contentPane.add(Button_Exit, null);
contentPane.add(ScrollPane, null);
ScrollPane.getViewport().add(EditorPane_Message, null);
Fill_Combo_Box();
}
//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        System.exit(0);
    }
}

void Button_Exit_actionPerformed(ActionEvent e)
{
    String[] str = new String[1];
    str[0] = IPAddress;
    Login_Screen.main(str);
    this.dispose();
}

void Button_Reset_actionPerformed(ActionEvent e)
{
    EditorPane_Message.setText("");
}

void Button_Send_actionPerformed(ActionEvent e)
{
    try
    {
        String reciever = (String)ComboBox_Reciever.getSelectedItem();
        String body = EditorPane_Message.getText();
        File PTFile = new File("plaintext.txt");
        FileOutputStream fos = new FileOutputStream(PTFile);
        PrintStream ps = new PrintStream(fos);
        ps.print(body);
        ps.close();
        fos.close();
        Socket s = new Socket(IPAddress, 9001);
        System.out.println("connected");
        OutputStream os = s.getOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(os);
        InputStream is = s.getInputStream();
        ObjectInputStream ois = new ObjectInputStream(is);
        oos.writeInt(3);
        oos.flush();
        oos.writeObject(reciever);
    }
}

```

```

        oos.flush();
        User_Info ui = (User_Info) ois.readObject();
        oos.close();
        os.close();
        ois.close();
        is.close();
        s.close();
        Message_Info mi = new Message_Info(0, SenderNumber, ui.User_Number);
        RSA_Functions.Encryptor(ui.Public_Key, "plaintext.txt", "cipher.jmp");
        File cipher = new File("cipher.jmp");
        FileReader fr = new FileReader(cipher);
        BufferedReader br = new BufferedReader(fr);
        //encryption and sending
        boolean check = true;
        char[] c = new char[1];
        s = new Socket(IPAddress, 9001);
        os = s.getOutputStream();
        oos = new ObjectOutputStream(os);
        is = s.getInputStream();
        ois = new ObjectInputStream(is);
        oos.writeInt(5);
        oos.flush();

        oos.writeObject(mi);
        oos.flush();
        while (br.read(c, 0, 1) != -1)
        {
            oos.writeChar(c[0]);
            oos.flush();
        }
        br.close();
        fr.close();
        ois.close();
        is.close();
        oos.close();
        os.close();
        s.close();
        EditorPane_Message.setText("");
    }
    catch(Exception ex){System.out.println(ex);}
}
void Button_ReceiveMessages_actionPerformed(ActionEvent e)
{
    String[] str = new String[2];
    str[0] = String.valueOf(SenderNumber);
    str[1] = IPAddress;
    Read_Messages_Screen.main(str);
    this.dispose();
}
}

```

```

import javax.swing.UIManager;
import java.awt.*;

```

```

public class NewUser_Screen
{
    private boolean packFrame = false;

    //Construct the application
    public NewUser_Screen(String str)
    {
        NewUser_Frame frame = new NewUser_Frame(str);
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else {

```



```

        frame.validate();
    }
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
    {
        frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width)
    {
        frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
}
//Main method
public static void main(String[] args)
{
    try
    {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    new NewUser_Screen(args[0]);
}
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.io.*;

public class NewUser_Frame extends JFrame
{
    private JPanel contentPane;
    private JLabel Label_NewUser = new JLabel();
    private JTextField TextField_Username = new JTextField();
    private JPasswordField PasswordField_Password = new JPasswordField();
    private JPasswordField PasswordField_Verify = new JPasswordField();
    private JButton Button_Enter = new JButton();
    private JButton Button_Cancel = new JButton();
    private JLabel Label_Username = new JLabel();
    private JLabel Label_Password = new JLabel();
    private JLabel Label_VerifyPassword = new JLabel();
    private String IPAddress;
    //Construct the frame
    public NewUser_Frame(String str)
    {
        IPAddress = str;
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try
        {
            jbInit();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
    //Component initialization
    private void jbInit() throws Exception
    {
        //setIconImage(Toolkit.getDefaultToolkit().createImage(NewUser_Screen.class.getResource
("[Your Icon]"));
        contentPane = (JPanel) this.getContentPane();
    }
}

```

```

Label_NewUser.setText("New User");
Label_NewUser.setBounds(new Rectangle(166, 18, 60, 28));
contentPane.setBorder(BorderFactory.createRaisedBevelBorder());
contentPane.setLayout(null);
this.setSize(new Dimension(420, 323));
this.setTitle("New User");
TextField_Username.setBounds(new Rectangle(150, 59, 206, 24));
PasswordField_Password.setBounds(new Rectangle(151, 111, 206, 25));
PasswordField_Verify.setBounds(new Rectangle(154, 162, 203, 27));
Button_Enter.setBounds(new Rectangle(61, 217, 107, 32));
Button_Enter.setText("Enter");
Button_Enter.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        Button_Enter_actionPerformed(e);
    }
});
Button_Cancel.setBounds(new Rectangle(233, 218, 105, 31));
Button_Cancel.setText("Cancel");
Button_Cancel.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        Button_Cancel_actionPerformed(e);
    }
});
Label_Username.setText("User Name");
Label_Username.setBounds(new Rectangle(37, 60, 75, 23));
Label_Password.setText("Password");
Label_Password.setBounds(new Rectangle(38, 110, 78, 23));
Label_VerifyPassword.setText("Verify Password");
Label_VerifyPassword.setBounds(new Rectangle(40, 156, 97, 26));
contentPane.add(TextField_Username, null);
contentPane.add(PasswordField_Password, null);
contentPane.add(Button_Enter, null);
contentPane.add(Button_Cancel, null);
contentPane.add(PasswordField_Verify, null);
contentPane.add(Label_VerifyPassword, null);
contentPane.add(Label_Password, null);
contentPane.add(Label_Username, null);
contentPane.add(Label_NewUser, null);
}
//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        System.exit(0);
    }
}

void Button_Enter_actionPerformed(ActionEvent e)
{
    boolean b = false;
    String Username = TextField_Username.getText();
    String Password = new String(PasswordField_Password.getPassword());
    String Verify = new String(PasswordField_Verify.getPassword());
    if (Username.compareTo("") == 0 || Password.compareTo("") == 0)
    {
        b = true;
    }
    if (Verify.compareTo>Password) != 0)
    {
        b = true;
    }
    if (!b)
    {
        try
        {

```

```

String Hashed = Math_Functions.hash_function(Password);
Socket s = new Socket(IPAddress, 9001);
OutputStream os = s.getOutputStream();
ObjectOutputStream oos = new ObjectOutputStream(os);
InputStream is = s.getInputStream();
ObjectInputStream ois = new ObjectInputStream(is);
b = true;
Key[] k = new Key[2];
oos.writeInt(1);
oos.flush();
while(b)
{
    k = RSA_Functions.Pair_Key_generator();
    oos.writeInt(k[0].n);
    oos.flush();
    b = ois.readBoolean();
    oos.writeBoolean(b);
    oos.flush();
}
User_Info ui = new User_Info(0, Username, Hashed, k[0]);
oos.writeObject(ui);
oos.flush();
b = ois.readBoolean();
oos.close();
ois.close();
is.close();
os.close();
s.close();
if (b)
{
    File PKFile = new File(Username+".pri");
    FileOutputStream fos = new FileOutputStream(PKFile);
    oos = new ObjectOutputStream(fos);
    oos.writeObject(k[1]);
    oos.close();
    fos.close();
    JOptionPane.showMessageDialog(null, "New User Created");
    System.out.println("New User Created");
    String[] str = new String[1];
    str[0] = IPAddress;
    Login_Screen.main(str);
    this.dispose();
}
else
{
    JOptionPane.showMessageDialog(null, "User already exists");
    System.out.println("Error! User already Exists");
    TextField_Username.setText("");
    PasswordField_Password.setText("");
    PasswordField_Verify.setText("");
}
}
catch(Exception ex){System.out.println(ex);}
}
else
{
    JOptionPane.showMessageDialog(null, "Please fill in fields correctly");
    System.out.println("Error! Fill in fields correctly");
    TextField_Username.setText("");
    PasswordField_Password.setText("");
    PasswordField_Verify.setText("");
}
}

void Button_Cancel_actionPerformed(ActionEvent e)
{
    String[] str = new String[1];
    str[0] = IPAddress;
    Login_Screen.main(str);
    this.dispose();
}

```

```

    }
}

import javax.swing.UIManager;
import java.awt.*;

public class Read_Messages_Screen
{
    private boolean packFrame = false;

    //Construct the application
    public Read_Messages_Screen(int i, String str)
    {
        Read_Messages_Frame frame = new Read_Messages_Frame(i, str);
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else
        {
            frame.validate();
        }
        //Center the window
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();
        if (frameSize.height > screenSize.height)
        {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width)
        {
            frameSize.width = screenSize.width;
        }
        frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
        frame.setVisible(true);
    }
    //Main method
    public static void main(String[] args)
    {
        Integer x = new Integer(args[0]);
        try
        {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        new Read_Messages_Screen(x.intValue(), args[1]);
    }
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import java.io.*;
import java.net.*;

public class Read_Messages_Frame extends JFrame
{
    private JPanel contentPane;
    private JLabel Label_MessageFrom = new JLabel();
    private JTextArea TextArea_Sender = new JTextArea();
    private JButton Button_ReadNext = new JButton();
    private TitledBorder titledBorder1;
    private JScrollPane ScrollPane = new JScrollPane();

```

```

private JTextPane TextPane_Message = new JTextPane();
private int UserNumber;
private boolean bool;
private String IPAddress;
//Construct the frame
public Read_Messages_Frame(int x, String str)
{
    UserNumber = x;
    IPAddress = str;
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try
    {
        jbInit();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
//Component initialization
private void Get_Message()
{
    boolean found = false;
    //get Username
    try
    {
        Socket s = new Socket(IPAddress, 9001);
        OutputStream os = s.getOutputStream();
        InputStream is = s.getInputStream();
        ObjectOutputStream soos = new ObjectOutputStream(os);
        ObjectInputStream sois = new ObjectInputStream(is);
        soos.writeInt(6);
        soos.flush();
        soos.writeInt(UserNumber);
        soos.flush();
        String UserName = (String) sois.readObject();
        soos.close();
        sois.close();
        is.close();
        os.close();
        //get message
        s = new Socket(IPAddress, 9001);
        os = s.getOutputStream();
        is = s.getInputStream();
        soos = new ObjectOutputStream(os);
        sois = new ObjectInputStream(is);
        soos.writeInt(2);
        soos.flush();
        soos.writeInt(UserNumber);
        soos.flush();
        found = sois.readBoolean();
        if (found)
        {
            Message_Info mi = (Message_Info) sois.readObject();
            File PKFile = new File(UserName+".pri");
            FileInputStream fis = new FileInputStream(PKFile);
            ObjectInputStream ois = new ObjectInputStream(fis);
            Key Private_Key = (Key) ois.readObject();
            ois.close();
            fis.close();
            File cipher = new File("temp_cipher.jmp");
            FileOutputStream fos = new FileOutputStream(cipher);
            PrintStream ps = new PrintStream(fos);
            boolean check = true;
            char[] c = new char[1];
            while(check)
            {
                try
                {
                    c[0] = sois.readChar();
                    ps.print(c[0]);
                }
            }
        }
    }
}

```

```

        }
        catch(Exception e){check = false;}
    }
    ps.close();
    sois.close();
    soos.close();
    is.close();
    os.close();
    fos.close();
    s.close();
    RSA_Functions.Decryptor(Private_Key, "temp_cipher.jmp",
"decipher.jmp");

    File decipher = new File ("decipher.jmp");
    FileReader fr = new FileReader(decipher);
    BufferedReader br = new BufferedReader(fr);
    String text = "";
    while(br.read(c, 0, 1) != -1)
    {
        text = text + new String(c);
    }
    TextPane_Message.setText(text);
    br.close();
    fr.close();
    //get Sender's name
    s = new Socket(IPAddress, 9001);
    os = s.getOutputStream();
    is = s.getInputStream();
    soos = new ObjectOutputStream(os);
    sois = new ObjectInputStream(is);
    soos.writeInt(6);
    soos.flush();
    soos.writeInt(mi.Sender);
    soos.flush();
    String SenderName = (String) sois.readObject();
    TextArea_Sender.setText(SenderName);
    TextArea_Sender.setEditable(false);
    TextPane_Message.setEditable(false);
    soos.close();
    sois.close();
    os.close();
    is.close();
    bool = true;
    }
    else
    {
        TextPane_Message.setText("No New Messages");
        TextPane_Message.setEditable(false);
        TextArea_Sender.setText("No New Messages");
        TextArea_Sender.setEditable(false);
        bool = false;
    }
    }
    catch(Exception e){}
}

private void jbInit() throws Exception
{
    //setIconImage(Toolkit.getDefaultToolkit().createImage(Read_Messages_Frame.class.getResource("[Your Icon]")));
    contentPane = (JPanel) this.getContentPane();
    titledBorder1 = new TitledBorder("");
    Label_MessageFrom.setText("Message from");
    Label_MessageFrom.setBounds(new Rectangle(32, 31, 89, 24));
    contentPane.setLayout(null);
    this.setSize(new Dimension(502, 337));
    this.setTitle("Message");
    TextArea_Sender.setBorder(BorderFactory.createEtchedBorder());
    TextArea_Sender.setBounds(new Rectangle(130, 31, 332, 23));
    Button_ReadNext.setBounds(new Rectangle(26, 249, 108, 31));
    Button_ReadNext.setText("Read Next");
    Button_ReadNext.addActionListener(new java.awt.event.ActionListener()

```

```

    {
        public void actionPerformed(ActionEvent e)
        {
            Button_ReadNext_actionPerformed(e);
        }
    });
ScrollPane.setBounds(new Rectangle(28, 68, 446, 164));
contentPane.add(Label_MessageFrom, null);
contentPane.add(TextArea_Sender, null);
contentPane.add(Button_ReadNext, null);
contentPane.add(ScrollPane, null);
ScrollPane.getViewPort().add(TextPane_Message, null);
Get_Message();
if (!bool)
{
    Button_ReadNext.setText("Back");
}
}

//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        String[] str = new String[2];
        str[0] = String.valueOf(UserNumber);
        str[1] = IPAddress;
        Main_Screen.main(str);
        this.dispose();
    }
}
public void Button_ReadNext_actionPerformed(ActionEvent e)
{
    if (bool)
    {
        String[] str = new String[2];
        str[0] = String.valueOf(UserNumber);
        str[1] = IPAddress;
        Read_Messages_Screen.main(str);
        this.dispose();
    }
    else
    {
        String[] str = new String[2];
        str[0] = String.valueOf(UserNumber);
        str[1] = IPAddress;
        Main_Screen.main(str);
        this.dispose();
    }
}
}

import java.math.*;
public class Math_Functions
{
    public static String hash_function(String password)
    {
        char[] c = password.toCharArray();
        String str = "", rev = "";
        int length = password.length();
        for(int x = 0; x != length; x++)
        {
            str = str+String.valueOf((int)c[x]);
            rev = rev+String.valueOf((int)c[length-x-1]);
        }
        BigInteger a = new BigInteger(str);
        BigInteger b = new BigInteger(rev);
        b = b.and(a);
        return b.toString();
    }
}

```

```

}

public static int power_mod(int base, int power, int mod)
{
    int x = 0;
    long p = 1;
    for (x = 0; x != power; x++)
    {
        p = p*base;
        p = p%mod;
    }
    Integer y = new Integer(String.valueOf(p));
    return y.intValue();
}

public static boolean check_if_prime(int x)
{
    /*if (x == 2)
    {
        return true;
    }
    else
    {
        return (power_mod(2,x,x) == 2);
    }*/
    BigInteger b = new BigInteger(String.valueOf(x));
    return b.isProbablePrime(10000);
}

public static int gcf(int x, int y)
{
    int z = 0;
    if (x<y)
    {
        z = x;
        x = y;
        y = z;
    }
    while (y != 0)
    {
        z = x%y;
        x = y;
        y = z;
    }
    return x;
}

public static int random_number(int x) // random number generator from 1 to x;
{
    int y;
    y = (int) Math.round(Math.random() * x)+1;
    return y;
}

public static int prime_generator(int n)
{
    int x = 0;
    boolean check = false;
    while (!check)
    {
        x = random_number(n);
        check = check_if_prime(x);
    }
    return x;
}

public static String pad(String a, int x)
{
    int y;
    for(y = a.length(); y < x; y++)
    {

```



```

        a = "0"+a;
    }
    return a;
}
}

import java.io.*;
import java.lang.*;
public class RSA_Functions
{
    public static Key[] Pair_Key_generator()
    {
        boolean check = false;
        int p=0, q=0, n=0, x=0;
        Key[] pair;
        pair = new Key[2];
        //public key generator
        while (!check)
        {
            p = Math_Functions.prime_generator(1000);
            q = Math_Functions.prime_generator(1000);
            n = p*q;
            if (n > 1000) check = true;
        }
        boolean bool = false;
        while(!bool)
        {
            check = false;
            while(!check)
            {
                x = Math_Functions.random_number(n);
                if (x != 1)
                {
                    check = (Math_Functions.gcf(x, (p-1)*(q-1))==1);
                }
            }
            pair[0] = new Key("public",x,n);
            //private key generator
            x = 1;
            check = false;
            while(!check)
            {
                x++;
                check = ((x*pair[0].d) % ((p-1)*(q-1)) == 1);
            }
            if ((x < (p-1)*(q-1)) && (pair[0].d < 10000))bool = true;
        }
        pair[1] = new Key("private",x,n);
        return pair;
    }

    public static void Encryptor(Key pub, String src_file, String dest_file)
    {
        try
        {
            File source = new File(src_file);
            File temp = new File("temp.tmp");
            File dest = new File(dest_file);
            int i = 0, counter = 0, y = 0;
            Integer x;
            String str = "";
            //read from source, write to temp
            BufferedReader br = new BufferedReader(new FileReader(source));
            FileOutputStream out = new FileOutputStream(temp);
            PrintStream p = new PrintStream(out);
            char[] c = new char[1];
            while(br.read(c, 0, 1) != -1)
            {
                i = (int) c[0];
                str = String.valueOf(i);
            }
        }
    }
}

```

```

        str = Math_Functions.pad(str, 6);
        p.print(str);
        System.out.print(str);
        counter++;
    }
    p.close();
    br.close();
    out.close();
    System.out.println("\n ----- \n");
    //read from temp write to dest
    br = new BufferedReader(new FileReader(temp));
    out = new FileOutputStream(dest);
    p = new PrintStream(out);
    c = new char[6];
    while(br.read(c, 0, 6) != -1)
    {
        x = new Integer(String.valueOf(c));
        i = x.intValue();
        i = Math_Functions.power_mod(i, pub.d, pub.n);
        str = String.valueOf(i);
        str = Math_Functions.pad(str, 6);
        p.print(str);
        System.out.print(str);
    }
    p.close();
    br.close();
    temp.delete();
}
catch(Exception e)
{}
}

public static void Decryptor (Key pri, String src_file, String dest_file)
{
    try
    {
        Math_Functions a = new Math_Functions();
        File source = new File(src_file);
        File temp = new File("temp.tmp");
        File dest = new File(dest_file);
        FileReader fr = new FileReader(source);
        BufferedReader br = new BufferedReader(fr);
        FileOutputStream out = new FileOutputStream(temp);
        PrintStream p = new PrintStream(out);
        char[] c = new char[6];
        String str="";
        char z= 'a';
        Integer x;
        int i =0;
        while(br.read(c, 0, 6) != -1)
        {
            x = new Integer(String.valueOf(c));
            i = x.intValue();
            i = Math_Functions.power_mod(i, pri.d, pri.n);
            str = String.valueOf(i);
            str = Math_Functions.pad(str, 6);
            p.print(str);
            System.out.print(str);
        }
        p.close();
        br.close();
        fr.close();
        fr = new FileReader(temp);
        br = new BufferedReader(fr);
        out = new FileOutputStream(dest);
        p = new PrintStream(out);
        c = new char[6];
        System.out.println(" ");
        while(br.read(c,0,6) != -1)
        {
            x = new Integer(String.valueOf(c).trim());

```

```
        i = x.intValue();
        z = (char) i;
        p.print(z);
        System.out.print(z);
    }
    p.close();
    out.close();
    br.close();
    fr.close();
}
catch(Exception e)
{System.out.print(e);}
}
}
```

Appendix 5

Code of the Attacker Component

```
import java.io.*;
public class Key implements Serializable
{
    public int d, n;
    public boolean type;
    public Key (String a, int x, int y)
    {
        type = (a == "public"); // TRUE = PUBLIC, FALSE = PRIVATE
        d = x;
        n = y;
    }
}

import java.io.*;
public class Message_Info implements Serializable
{
    int Message_Number;
    int Sender;
    int Reciever;
    Message_Info(){}
    Message_Info(int a, int b, int c)
    {
        Message_Number = a;
        Sender = b;
        Reciever = c;
    }
}

import java.io.*;
import java.lang.*;
public class User_Info implements Serializable
{
    int User_Number;
    String User_Name;
    String Hashed_Password;
    Key Public_Key;
    User_Info(int a, String b, String c, Key d)
    {
        User_Name = b;
        Hashed_Password = c;
        Public_Key = d;
        User_Number = a;
    }
    User_Info(){}
}

import javax.swing.UIManager;
import java.awt.*;

public class Attacker_Connection_Screen
{
    private boolean packFrame = false;

    //Construct the application
    public Attacker_Connection_Screen()
    {
        Attacker_Connection_Frame frame = new Attacker_Connection_Frame();
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else
    }
}
```

```

        {
            frame.validate();
        }
        //Center the window
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();
        if (frameSize.height > screenSize.height)
        {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width)
        {
            frameSize.width = screenSize.width;
        }
        frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
        frame.setVisible(true);
    }
    //Main method
    public static void main(String[] args)
    {
        try
        {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        new Attacker_Connection_Screen();
    }
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.io.*;

public class Attacker_Connection_Frame extends JFrame
{
    private JPanel contentPane;
    private JButton Button_Connect = new JButton();
    private JLabel Label_IPAddress_of_Server = new JLabel();
    private JTextField TextField_IPAddress = new JTextField();

    //Construct the frame
    public Attacker_Connection_Frame()
    {
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try
        {
            jbInit();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
    //Component initialization
    private void jbInit() throws Exception
    {
        //setIconImage(Toolkit.getDefaultToolkit().createImage(Attacker_Connection_Frame.class.
getResource("[Your Icon]"));
        contentPane = (JPanel) this.getContentPane();
        Button_Connect.setBounds(new Rectangle(136, 130, 127, 39));
        Button_Connect.setText("Connect");
        Button_Connect.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {

```

```

        Button_Connect_actionPerformed(e);
    }
});
contentPane.setLayout(null);
this.setSize(new Dimension(400, 229));
this.setTitle("RSA Attacker");
Label_IPAddress_of_Server.setText("Enter IP Address of the Server Computer");
Label_IPAddress_of_Server.setBounds(new Rectangle(90, 33, 229, 23));
TextField_IPAddress.setText("localhost");
TextField_IPAddress.setBounds(new Rectangle(34, 80, 329, 24));
contentPane.setBorder(BorderFactory.createRaisedBevelBorder());
contentPane.add(TextField_IPAddress, null);
contentPane.add(Button_Connect, null);
contentPane.add(Label_IPAddress_of_Server, null);
}
//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        System.exit(0);
    }
}

void Button_Connect_actionPerformed(ActionEvent e)
{
    try
    {
        String str = TextField_IPAddress.getText();
        Socket s = new Socket(str, 9001);
        OutputStream os = s.getOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(os);
        InputStream is = s.getInputStream();
        ObjectInputStream ois = new ObjectInputStream(is);
        String utf = s.getInetAddress().getCanonicalHostName();
        oos.writeInt(0);
        oos.flush();
        oos.writeUTF(utf);
        oos.flush();
        oos.close();
        ois.close();
        os.close();
        is.close();
        s.close();
        User_Info[] ui = new User_Info[1];
        ui[0] = new User_Info(0,"",null);
        Message_Info[] mi = new Message_Info[1];
        mi[0] = new Message_Info(0,0,0);
        Attacker_Main_Screen.main(str, ui, mi, 0);

        this.dispose();
    }
    catch(Exception ex)
    {
        JOptionPane.showMessageDialog(null, "No Connection of that address");
        TextField_IPAddress.setText("localhost");
    }
}
}

import javax.swing.UIManager;
import java.awt.*;

public class Attacker_Main_Screen
{
    private boolean packFrame = false;

    //Construct the application

```

```

public Attacker_Main_Screen(String IP, User_Info[] ui, Message_Info[] mi, int m)
{
    Attacker_Main_Frame frame = new Attacker_Main_Frame(IP, ui, mi, m);
    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from their layout
    if (packFrame)
    {
        frame.pack();
    }
    else
    {
        frame.validate();
    }
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
    {
        frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width)
    {
        frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
}
//Main method
public static void main(String args, User_Info[] uis, Message_Info[] mis, int i)
{
    try
    {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    new Attacker_Main_Screen(args, uis, mis, i);
}
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;
import java.io.*;
import java.math.*;

public class Attacker_Main_Frame extends JFrame
{
    private JPanel contentPane;
    private JButton Button_Decrypt = new JButton();
    private JButton Button_Exit = new JButton();
    private User_Info[] uis;
    private Message_Info[] mis;
    private String IPAddress;
    private JTextArea TextArea_Instructions = new JTextArea();
    private int m;
    //Construct the frame
    public Attacker_Main_Frame(String IP, User_Info[] ui, Message_Info[] mi, int n)
    {
        IPAddress = IP;
        uis = ui;
        mis = mi;
        m = n;
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try
        {
            jbInit();

```

```

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

private void get_UIs_MIs()
{
    try
    {
        Socket s = new Socket(IPAddress, 9001);
        OutputStream os = s.getOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(os);
        InputStream is = s.getInputStream();
        ObjectInputStream ois = new ObjectInputStream(is);
        oos.writeInt(7);
        oos.flush();
        int x = ois.readInt();
        System.out.println(x);
        uis = new User_Info[x];
        for(int y = 0; y != x; y++)
        {
            uis[y] = (User_Info) ois.readObject();
        }
        x = ois.readInt();
        System.out.println(x);
        m = x;
        mis = new Message_Info[x];
        for(int y = 0; y != x; y++)
        {
            mis[y] = (Message_Info) ois.readObject();
        }
        oos.close();
        ois.close();
        os.close();
        is.close();
        s.close();
    }
    catch(Exception e){}
}

//Component initialization
private void jbInit() throws Exception
{
    //setIconImage(Toolkit.getDefaultToolkit().createImage(Attacker_Main_Frame.class.getResource("[Your Icon]")));
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(null);
    this.setSize(new Dimension(400, 215));
    this.setTitle("RSA Attacker");
    contentPane.setBorder(BorderFactory.createRaisedBevelBorder());
    Button_Decrypt.setBounds(new Rectangle(62, 127, 115, 33));
    Button_Decrypt.setText("Get Message");
    Button_Decrypt.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            Button_Decrypt_actionPerformed(e);
        }
    });
    Button_Exit.setBounds(new Rectangle(229, 126, 114, 32));
    Button_Exit.setText("Exit");
    Button_Exit.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            Button_Exit_actionPerformed(e);
        }
    });
    TextArea_Instructions.setToolTipText("");
    TextArea_Instructions.setBounds(new Rectangle(80, 50, 252, 40));
}

```



```

        TextArea_Instructions.setText("Press Get message to randomly get a \n message and try
to decrypt it");
        TextArea_Instructions.setEditable(false);
        contentPane.add(Button_Exit, null);
        contentPane.add(Button_Decrypt, null);
        contentPane.add(TextArea_Instructions, null);
        if (mis[0].Message_Number == 0)
        {
            get_UIs_MIs();
        }
    }
    //Overridden so we can exit when window is closed
    protected void processWindowEvent(WindowEvent e)
    {
        super.processWindowEvent(e);
        if (e.getID() == WindowEvent.WINDOW_CLOSING)
        {
            System.exit(0);
        }
    }

    void Button_Decrypt_actionPerformed(ActionEvent e)
    {
        int y;
        System.out.println(m);
        y = (int) Math.round(Math.random() * (m-1))+1;
        System.out.println(y);
        Attacker_Read_Messages_Screen.main(IPAddress, uis, mis, y, m);
        this.dispose();
    }

    void Button_Exit_actionPerformed(ActionEvent e)
    {
        System.exit(0);
    }
}

import javax.swing.UIManager;
import java.awt.*;

public class Attacker_Read_Messages_Screen
{
    private boolean packFrame = false;

    //Construct the application
    public Attacker_Read_Messages_Screen(String IP, User_Info[] ui, Message_Info[] mi, int
mes_no, int x)
    {
        Attacker_Read_Messages_Frame frame = new Attacker_Read_Messages_Frame(IP, ui, mi,
mes_no, x);
        //Validate frames that have preset sizes
        //Pack frames that have useful preferred size info, e.g. from their layout
        if (packFrame)
        {
            frame.pack();
        }
        else
        {
            frame.validate();
        }
        //Center the window
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();
        if (frameSize.height > screenSize.height)
        {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width)
        {
            frameSize.width = screenSize.width;
        }
    }
}

```

```

        frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
        frame.setVisible(true);
    }
    //Main method
    public static void main(String IP, User_Info[] ui, Message_Info[] mi, int mes_no, int i)
    {
        try
        {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        new Attacker_Read_Messages_Screen(IP, ui, mi, mes_no, i);
    }
}

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.net.*;

public class Attacker_Read_Messages_Frame extends JFrame
{
    private JPanel contentPane;
    private JScrollPane ScrollPanel = new JScrollPane();
    private JTextArea TextArea_Message = new JTextArea();
    private JButton Button_ReadNext = new JButton();
    private JButton Button_ReadPrevious = new JButton();
    private JLabel Label_Sender = new JLabel();
    private JLabel Label_To = new JLabel();
    private JLabel Label_Receipient = new JLabel();
    private User_Info[] uis;
    private Message_Info[] mis;
    private String IPAddress;
    private int Message_No;
    private boolean File_Exists;
    private int block_size;
    private Key pri;
    private int jmp;
    //Construct the frame
    public Attacker_Read_Messages_Frame(String IP, User_Info[] ui, Message_Info[] mi, int
mes_no, int j)
    {
        IPAddress = IP;
        uis = ui;
        mis = mi;
        Message_No = mes_no;
        block_size = 1;
        jmp = j;
        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        try
        {
            jbInit();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
    //Component initialization
    private void get_File()
    {
        try
        {
            Socket s = new Socket(IPAddress, 9001);
            OutputStream os = s.getOutputStream();
            ObjectOutputStream soos = new ObjectOutputStream(os);

```

```

InputStream is = s.getInputStream();
ObjectInputStream sois = new ObjectInputStream(is);
sois.writeInt(8);
sois.flush();
sois.writeInt(Message_No);
sois.flush();
File_Exists = sois.readBoolean();
if(File_Exists)
{
    File cipher = new File("cipher.mes");
    FileOutputStream fos = new FileOutputStream(cipher);
    PrintStream ps = new PrintStream(fos);
    boolean check = true;
    char c = 'a';
    while(check)
    {
        try
        {
            c = sois.readChar();
            ps.print(c);
        }
        catch(Exception e){check = false;}
    }
    ps.close();
    fos.close();
}
sois.close();
soos.close();
os.close();
is.close();
s.close();
}
catch(Exception e){}
}

private void Initialize_Message()
{
    if(File_Exists)
    {
        try
        {
            pri = Attack.Crack_Key(uis[uis[Message_No-1].Reciever-1].Public_Key);
            for(int x = 1 ; x != 11; x++)
            {
                Attack.Decryptor(pri, "cipher.mes", String.valueOf(x)+".txt",
x);

                File PLFile = new File("1.txt");
                FileReader fr = new FileReader(PLFile);
                BufferedReader br = new BufferedReader(fr);
                char[] c = new char[1];
                String text = "";
                while(br.read(c, 0,1) != -1)
                {
                    text = text + String.valueOf(c);
                }
                TextArea_Message.setText(text);
                TextArea_Message.setEditable(false);
                br.close();
                fr.close();
                System.out.println("");
            }
        }
        catch(Exception e){}
    }
    else
    {
        JOptionPane.showMessageDialog(null, "File was delete in the server");
        TextArea_Message.setText("File was deleted in the server");
        Button_ReadNext.setText("Back");
        Button_ReadPrevious.setEnabled(false);
    }
}

```

```

        Button_ReadPrevious.setVisible(false);
        TextArea_Message.setEditable(false);
    }
}

private void jbInit() throws Exception
{
    //setIconImage(Toolkit.getDefaultToolkit().createImage(Attacker_Read_Messages_Frame.class.getResource("[Your Icon]")));
    contentPane = (JPanel) this.getContentPane();
    contentPane.setBorder(BorderFactory.createRaisedBevelBorder());
    contentPane.setLayout(null);
    this.setSize(new Dimension(400, 363));
    this.setTitle("RSA Attacker");
    ScrollPanel.setBorder(BorderFactory.createLoweredBevelBorder());
    ScrollPanel.setBounds(new Rectangle(36, 83, 324, 167));
    Button_ReadNext.setBounds(new Rectangle(67, 268, 110, 33));
    Button_ReadNext.setText("Read Next");
    Button_ReadNext.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            Button_ReadNext_actionPerformed(e);
        }
    });
    Button_ReadPrevious.setBounds(new Rectangle(220, 267, 119, 32));
    Button_ReadPrevious.setText("Read Previous");
    Button_ReadPrevious.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            Button_ReadPrevious_actionPerformed(e);
        }
    });
    Label_Sender.setText("<Sender>");
    Label_Sender.setBounds(new Rectangle(61, 40, 86, 19));
    Label_To.setText("to");
    Label_To.setBounds(new Rectangle(181, 39, 18, 20));
    Label_Receipient.setText("<Receipient>");
    Label_Receipient.setBounds(new Rectangle(241, 38, 84, 20));
    contentPane.add(ScrollPanel, null);
    ScrollPanel.getViewport().add(TextArea_Message, null);
    contentPane.add(Button_ReadNext, null);
    contentPane.add(Label_To, null);
    contentPane.add(Label_Sender, null);
    contentPane.add(Label_Receipient, null);
    contentPane.add(Button_ReadPrevious, null);
    Label_Sender.setText(uis[uis[Message_No-1].Sender-1].User_Name);
    Label_Receipient.setText(uis[uis[Message_No-1].Reciever-1].User_Name);
    get_File();
    Initialize_Message();
}
//Overridden so we can exit when window is closed
protected void processWindowEvent(WindowEvent e)
{
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
        Attacker_Main_Screen.main(IPAddress, uis, mis, jmp);
        this.dispose();
    }
}

void Button_ReadNext_actionPerformed(ActionEvent e)
{
    if(File_Exists)
    {
        try
        {
            TextArea_Message.setEditable(true);
            block_size++;
        }
    }
}

```

```

        //Attack.Decryptor(pri, "cipher.mes", "plaintext.txt", block_size);
        File PLFile = new File(String.valueOf(block_size)+".txt");
        FileReader fr = new FileReader(PLFile);
        BufferedReader br = new BufferedReader(fr);
        char[] c = new char[1];
        String text = "";
        while(br.read(c, 0,1) != -1)
        {
            text = text + String.valueOf(c);
        }
        TextArea_Message.setText(text);
        br.close();
        fr.close();
        if (block_size == 10)
        {
            Button_ReadNext.setText("Back");
            File_Exists = false;
        }
        TextArea_Message.setEditable(false);
    }
    catch(Exception ex){}
}
else
{
    Attacker_Main_Screen.main(IPAddress, uis, mis,jmp);
    this.dispose();
}
}

void Button_ReadPrevious_actionPerformed(ActionEvent e)
{
    if(block_size == 1)
    {
        JOptionPane.showMessageDialog(null, "Cannot lower block size anymore");
    }
    else
    {
        try
        {
            TextArea_Message.setEditable(true);
            if (block_size == 10)
            {
                File_Exists = true;
                this.Button_ReadNext.setText("Read Next");
            }
            block_size--;
            //Attack.Decryptor(pri, "cipher.mes", "plaintext.txt", block_size);
            File PLFile = new File(String.valueOf(block_size)+".txt");
            FileReader fr = new FileReader(PLFile);
            BufferedReader br = new BufferedReader(fr);
            char[] c = new char[1];
            String text = "";
            while(br.read(c, 0,1) != -1)
            {
                text = text + String.valueOf(c);
            }
            TextArea_Message.setText(text);
            br.close();
            fr.close();
            TextArea_Message.setEditable(false);
        }
        catch(Exception ex){}
    }
}

import java.io.*;

public class Attack
{
    public static int[] get_PQ(int n)

```

```

{
    int[] PQ = new int[2];
    boolean found = false;
    for(int x = 2; (x != n && !found); x++)
    {
        if (n % x == 0)
        {
            found = true;
            PQ[0] = x;
        }
        System.out.println(x);
    }
    PQ[1] = n / PQ[0];
    return PQ;
}

public static int gcd(int x, int y)
{
    int z = 0;
    if (x<y)
    {
        z = x;
        x = y;
        y = z;
    }
    while (y != 0)
    {
        z = x%y;
        x = y;
        y = z;
    }
    return x;
}

public static int power_mod(int base, int power, int mod)
{
    int x = 0;
    long p = 1;
    for (x = 0; x != power; x++)
    {
        p = (p*base)%mod;
    }
    Integer y = new Integer(String.valueOf(p));
    return y.intValue();
}

public static void Decryptor (Key pri, String src_file, String dest_file, int block)
{
    try
    {
        File source = new File(src_file);
        File dest = new File(dest_file);
        BufferedReader br = new BufferedReader(new FileReader(source));
        FileOutputStream out = new FileOutputStream(dest);
        PrintStream p = new PrintStream(out);
        char[] c = new char[block];
        String str="";
        char z= 'a';
        Integer x;
        int i =0;
        while(br.read(c, 0, block) != -1)
        {
            x = new Integer(String.valueOf(c));
            i = x.intValue();
            i = power_mod(i, pri.d, pri.n);
            str = String.valueOf(i);
            z = (char) i;
            p.print(z);
            System.out.print(z);
        }
        p.close();
    }
}

```

```

        out.close();
        br.close();
    }
    catch(Exception e){System.out.println(e);}
}
public static Key Crack_Key(Key pub)
{
    int[] PQ = get_PQ(pub.n);
    int x = 1;
    boolean check = false;
    while(!check)
    {
        x++;
        check = ((x*pub.d) % ((PQ[0]-1)*(PQ[1]-1)) == 1);
        System.out.println(""+x+ " ", "+pub.n+");
    }
    Key pri = new Key("private", x, pub.n);
    return pri;
}

public void run()
{
    try
    {
        File users = new File ("User_Information.db");
        FileInputStream fis = new FileInputStream(users);
        ObjectInputStream ois = new ObjectInputStream(fis);
        ois.readObject();
        User_Info ui = (User_Info)ois.readObject();
        ois.close();
        fis.close();
        int[] PQ = get_PQ(ui.Public_Key.n);
        int x = 1;
        boolean check = false;
        while(!check)
        {
            x++;
            check = ((x*ui.Public_Key.d) % ((PQ[0]-1)*(PQ[1]-1)) == 1);
            System.out.println(""+x+ " ", "+ui.Public_Key.n+");
        }
        Key PK = new Key("private", x, ui.Public_Key.n);
        Decryptor(PK, "l.mes", "test.txt", 6);
    }
    catch(Exception e){System.out.println(e);}
}
}

```

Appendix 6

Code of the Java programs to be imported by scripting languages

```
import java.io.*;
public class Key implements Serializable
{
    public int d, n;
    public boolean type;
    public Key (String a, int x, int y)
    {
        type = (a == "public"); // TRUE = PUBLIC, FALSE = PRIVATE
        d = x;
        n = y;
    }
}

import java.io.*;
public class KeyGenerator
{
    public static void main(String args[])
    {
        if (args.length == 2)
        {
            try
            {
                File PubKey = new File(args[0]);
                File PriKey = new File(args[1]);
                Key[] k = RSA_Functions.Pair_Key_generator();
                FileOutputStream fos = new FileOutputStream(PubKey);
                ObjectOutputStream oos = new ObjectOutputStream(fos);
                oos.writeObject(k[0]);
                oos.close();
                fos.close();
                fos = new FileOutputStream(PriKey);
                oos = new ObjectOutputStream(fos);
                oos.writeObject(k[1]);
                oos.close();
                fos.close();
                System.out.println("Public Key: (" +k[0].d+", "+k[0].n+"");
                System.out.println("Private Key: (" +k[1].d+", "+k[1].n+"");
                System.out.println("Key Generation Successful");
            }
            catch(Exception e){System.out.println("Error in creating keys");}
        }
        else
        {
            System.out.println("Too few or too many parameters");
        }
    }
}

import java.io.*;
public class Encryptor
{
    public static void main(String args[])
    {
        if (args.length == 3)
        {
            try
            {
                File PKFile = new File(args[0]);
                FileInputStream fis = new FileInputStream(PKFile);
                ObjectInputStream ois = new ObjectInputStream(fis);
                Key pub = (Key) ois.readObject();
                ois.close();
                fis.close();
                RSA_Functions.Encryptor(pub, args[1], args[2]);
            }
        }
    }
}
```



```

        }
        catch(Exception e){System.out.println("Error in Encrypting");}
    }
    else
    {
        System.out.println("Too few or too many parameters");
    }
}

import java.io.*;
public class Decryptor
{
    public static void main(String args[])
    {
        if (args.length == 3)
        {
            try
            {
                File PKFile = new File(args[0]);
                FileInputStream fis = new FileInputStream(PKFile);
                ObjectInputStream ois = new ObjectInputStream(fis);
                Key pri = (Key) ois.readObject();
                ois.close();
                fis.close();
                RSA_Functions.Decryptor(pri, args[1], args[2]);
            }
            catch(Exception e){System.out.println("Error in Decrypting");}
        }
        else
        {
            System.out.println("Too few or too many parameters");
        }
    }
}

import java.math.*;
public class Math_Functions
{
    public static String hash_function(String password)
    {
        char[] c = password.toCharArray();
        String str = "", rev= "";
        int length = password.length();
        for(int x = 0; x != length; x++)
        {
            str = str+String.valueOf((int)c[x]);
            rev = rev+String.valueOf((int)c[length-x-1]);
        }
        BigInteger a = new BigInteger(str);
        BigInteger b = new BigInteger(rev);
        b = b.and(a);
        return b.toString();
    }

    public static int power_mod(int base, int power, int mod)
    {
        int x = 0;
        long p = 1;
        for (x = 0; x != power; x++)
        {
            p = p*base;
            p = p%mod;
        }
        Integer y = new Integer(String.valueOf(p));
        return y.intValue();
    }

    public static boolean check_if_prime(int x)
    {
        /*if (x == 2)

```

```

        {
            return true;
        }
        else
        {
            return (power_mod(2,x,x) == 2);
        }
    }*/
    BigInteger b = new BigInteger(String.valueOf(x));
    return b.isProbablePrime(10000);
}

public static int gcf(int x, int y)
{
    int z = 0;
    if (x<y)
    {
        z = x;
        x = y;
        y = z;
    }
    while (y != 0)
    {
        z = x%y;
        x = y;
        y = z;
    }
    return x;
}

public static int random_number(int x) // random number generator from 1 to x;
{
    int y;
    y = (int) Math.round(Math.random() * x)+1;
    return y;
}

public static int prime_generator(int n)
{
    int x = 0;
    boolean check = false;
    while (!check)
    {
        x = random_number(n);
        check = check_if_prime(x);
    }
    return x;
}

public static String pad(String a, int x)
{
    int y;
    for(y = a.length(); y < x; y++)
    {
        a = "0"+a;
    }
    return a;
}
}

import java.io.*;
import java.lang.*;
public class RSA_Functions
{
    public static Key[] Pair_Key_generator()
    {
        boolean check = false;
        int p=0, q=0, n=0, x=0;
        Key[] pair;
        pair = new Key[2];
        //public key generator
        while (!check)

```

```

    {
        p = Math_Functions.prime_generator(1000);
        q = Math_Functions.prime_generator(1000);
        n = p*q;
        if (n > 1000) check = true;
    }
    boolean bool = false;
    while(!bool)
    {
        check = false;
        while(!check)
        {
            x = Math_Functions.random_number(n);
            if (x != 1)
            {
                check = (Math_Functions.gcf(x, (p-1)*(q-1))==1);
            }
        }
        pair[0] = new Key("public",x,n);
        //private key generator
        x = 1;
        check = false;
        while(!check)
        {
            x++;
            check = ((x*pair[0].d) % ((p-1)*(q-1)) == 1);
        }
        if ((x < (p-1)*(q-1)) && (pair[0].d < 10000))bool = true;
    }
    pair[1] = new Key("private",x,n);
    return pair;
}

public static void Encryptor(Key pub, String src_file, String dest_file)
{
    try
    {
        File source = new File(src_file);
        File temp = new File("temp.tmp");
        File dest = new File(dest_file);
        int i = 0, counter = 0, y = 0;
        Integer x;
        String str = "";
        //read from source, write to temp
        BufferedReader br = new BufferedReader(new FileReader(source));
        FileOutputStream out = new FileOutputStream(temp);
        PrintStream p = new PrintStream(out);
        char[] c = new char[1];
        while(br.read(c, 0, 1) != -1)
        {
            System.out.print(c);
            i = (int) c[0];
            str = String.valueOf(i);
            str = Math_Functions.pad(str, 6);
            p.print(str);
            counter++;
        }
        p.close();
        br.close();
        out.close();
        System.out.println("\n ----- \n");
        //read from temp write to dest
        br = new BufferedReader(new FileReader(temp));
        out = new FileOutputStream(dest);
        p = new PrintStream(out);
        c = new char[6];
        while(br.read(c, 0, 6) != -1)
        {
            x = new Integer(String.valueOf(c));
            i = x.intValue();

```

```

        i = Math_Functions.power_mod(i, pub.d, pub.n);
        str = String.valueOf(i);
        str = Math_Functions.pad(str, 6);
        p.print(str);
        System.out.print(str);
    }
    p.close();
    br.close();
    temp.delete();
}
catch(Exception e)
{}
}

public static void Decryptor (Key pri, String src_file, String dest_file)
{
    try
    {
        Math_Functions a = new Math_Functions();
        File source = new File(src_file);
        File temp = new File("temp.tmp");
        File dest = new File(dest_file);
        FileReader fr = new FileReader(source);
        BufferedReader br = new BufferedReader(fr);
        FileOutputStream out = new FileOutputStream(temp);
        PrintStream p = new PrintStream(out);
        char[] c = new char[6];
        String str="";
        char z= 'a';
        Integer x;
        int i =0;
        while(br.read(c, 0, 6) != -1)
        {
            x = new Integer(String.valueOf(c));
            i = x.intValue();
            i = Math_Functions.power_mod(i, pri.d, pri.n);
            str = String.valueOf(i);
            str = Math_Functions.pad(str, 6);
            p.print(str);
            System.out.print(str);
        }
        p.close();
        br.close();
        fr.close();
        fr = new FileReader(temp);
        br = new BufferedReader(fr);
        out = new FileOutputStream(dest);
        p = new PrintStream(out);
        c = new char[6];
        System.out.println(" ");
        while(br.read(c,0,6) != -1)
        {
            x = new Integer(String.valueOf(c).trim());
            i = x.intValue();
            z = (char) i;
            p.print(z);
            System.out.print(z);
        }
        p.close();
        out.close();
        br.close();
        fr.close();
    }
    catch(Exception e)
    {System.out.print(e);}
}
}

```