SECURE COMMUNICATION CHANNEL USING RSA ALGORITHM

BY

KAPIL SHARMA - 061249

SWADEEP SINGH - 061299

VIVEK CHAITANYA - 061311

MAY – 2010 (8th SEMESTER)

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

विद्या तत्व ज्योतिसम:

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

UNDERTAKING

This is to certify that the thesis entitled **"Secure Communication channel using RSA Algorithm",** submitted by us to the Jaypee University of Information Technology, Solan for the award of the final year project of B-tech degree by Research is a bonafide record of research work carried out by me under the supervision of Dr.Smriti Agrawal. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: JUIT

Date:

Kapil	Swadeep	Vivek
Signature of Student	**Signature of Student**	**Signature of Student**
Kapil Sharma	**Swadeep Singh**	**Vivek Chaitanya**
061249	**061299**	**061311**

विद्या तत्व ज्योतिसम:

JAYPEE UNIVERSITY OF
INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the work entitled **"Secure Communication channel using RSA Algorithm"** submitted by Kapil Sharma (061249) Swadeep Singh (061299) Vivek Chaitanya (061311) For the award of Bachelor of Technology in Computer Science Engineering of Jaypee University of information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Smriti Agrawal
18/05/10

Dr. Smriti Agrawal
Asst. Professor
Department of Computer Science Engineering and Information Technology
Jaypee University of Information Technology
Waknaghat

ACKNOWLEDGEMENT

"Winning is not everything, but the effort to win is"

Apart from the efforts by us, the success of this project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We would like to show our greatest appreciation to Dr. Smriti Agrawal . We feel motivated and encouraged every time we get her encouragement. For her coherent guidance throughout the tenure of the project, we feel fortunate to be taught by Dr. Smriti Agrawal, who gave us her unwavering support. Besides being our mentor, she has taught us that there's no substitute for hard work.

We owe our heartiest thanks to Brig. (Retd.) S.P.Ghrera(H.O.D-CSE/I.T Department) who've always inspired confidence in us to take initiative. He has always been motivating and encouraging.

Project Group No. 7

061249,061299,061311

B.TECH (CSE)

[illegible] 061299

Kapil 061249

Vivek 061311

Table of Contents

LIST OF FIGURES

LIST OF ABBREVIATIONS

RSA	Rivets Shamir Adelman
DES	Data Encryption Standard
AES	Advanced Encryption Standard
PU	Public Key
PR	Private Key
TCP. /IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
MIT	Massachusetts Institute of Technology
SQL	Structured Query Language
GCD	Greatest Common Divisor
HTTP	Hypertext Transfer Protocol
FTP	File Transfer protocol
DBMS	Database Management System

ABSTRACT

This Report briefly introduces the concept of RSA algorithm, and presents the flaws of other existing cryptographic Algorithm, thereby designs our improved implementation and analyzes the performance of our implementation. We've developed a piece of software for encrypting and decrypting text files.

Using RSA we have generated key pairs (public and private key) and then encryption and decryption of the messages is done. Establishing client server architecture using TCP/IP. Database management system has been used to store the clients information.

In this document we have given an overview of cryptography. We have described the Basic terminologies used in cryptography, Overview of symmetric and asymmetric cryptography. Later we have explained the RSA Algorithm, The steps to generate public and private key and Encryption and Decryption of message.

We have also tried to make it multiple client server, so that at one particular time more than one person can connect to the main server and chat with their desired friends. We have also researched for new techniques of implementing RSA algorithm efficiently.

Softwares which we have used are NetBeans 6.7.1 & IDEMicrosoft SQL 2005.During the year of project work we came across with different subjects to implement the RSA algorithm, which helped us to brush up with our previous subjects and helped us to learn new things, some of them are as follows.

- Network security and cryptography
- Database management system
- Data structures
- Computer networks
- Software engineering
- Fundamentals of algorithm
- Object Oriented Programming Language

CHAPTER 1

Introduction to Cryptography

In the era of information technology, the possibility that the information stored in a person's computer or the information that are being transferred through network of computers or internet being read by other people is very high. This causes a major concern for privacy, identity theft, electronic payments, corporate security, military communications and many others. We need an efficient and simple way of securing the electronic documents from being read or used by people other than who are authorized to do it. Cryptography is a standard way of securing the electronic documents.

Basic idea of cryptography is to mumble-jumble the original message into something that is unreadable or to something that is readable but makes no sense of what the original message is. To retrieve the original message again, we have to transform the mumble-jumbled message back into the original message again.

1.1 Terminologies used in Cryptography:

Data that can be read and understood without any special measures is called ***plaintext*** or ***cleartext***. This is the message or data that has to be secured. The method of disguising plaintext in such a way as to hide its substance is called ***encryption***. Encrypting plaintext results in unreadable gibberish called ***ciphertext***. You use encryption to ensure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting ciphertext to its original plaintext is called ***decryption***.

Cryptography is the science of mathematics to "encrypt" and "decrypt" data. Cryptography enables us to store sensitive information or transmit it across insecure networks like Internet so that no one else other the intended recipient can read it.

Cryptography can be used to provide:

- Confidentiality - ensure data is read only by authorized parties,
- Data integrity - ensure data wasn't altered between sender and recipient,
- Authentication - ensure data originated from a particular party.

Cryptanalysis is the art of breaking Ciphers that is retrieving the original message without knowing the proper key. Cryptography deals with all aspects of secure messaging, authentication, digital signatures, electronic money, and other applications.

1.2 Cryptographic Algorithms:

Cryptographic algorithms are mathematical functions that are used in the encryption and decryption process. A cryptographic algorithms works in combination with a ***key*** (a number, word or phrase), to encrypt the plain text. Same plain text encrypts to different cipher texts for different keys. Strength of a cryptosystems depends on the strength of the algorithm and the secrecy of the key.

1.3 Two Kinds of Cryptography Systems:

There are two kinds of cryptosystems: ***symmetric*** and ***asymmetric***. Symmetric cryptosystems use the same key (the secret key) to encrypt and decrypt a message, and asymmetric cryptosystems use one key (the public key) to encrypt a message and a different key (the private key) to decrypt it. Symmetric cryptosystems are also called as ***private key*** cryptosystems and asymmetric cryptosystems are also called as ***public key*** cryptosystems.

CHAPTER 2

Symmetric and Asymmetric Key Cryptography

2.1 Overview of Symmetric-Key Cryptography

Symmetric-key cryptography *(Private-key cryptography)*, refers to encryption methods in which both the sender and receiver share the same key. A symmetric encryption scheme has five ingredients:

- **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The Secret key is also input to the encryption algorithm. The key is a value independent of the plain text and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact transformations and substitutions performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as the output. It depends on the plaintext and the secret key. For a given message, two different key will produce two different ciphertext. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

There are two requirements for secure use of encryption:

1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertext would be unable to decipher the ciphertext or figure out the key. This requirement is stated in a stronger form: The opponent should be unable to decrypt ciphertext or

discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all the communication using this key is readable.

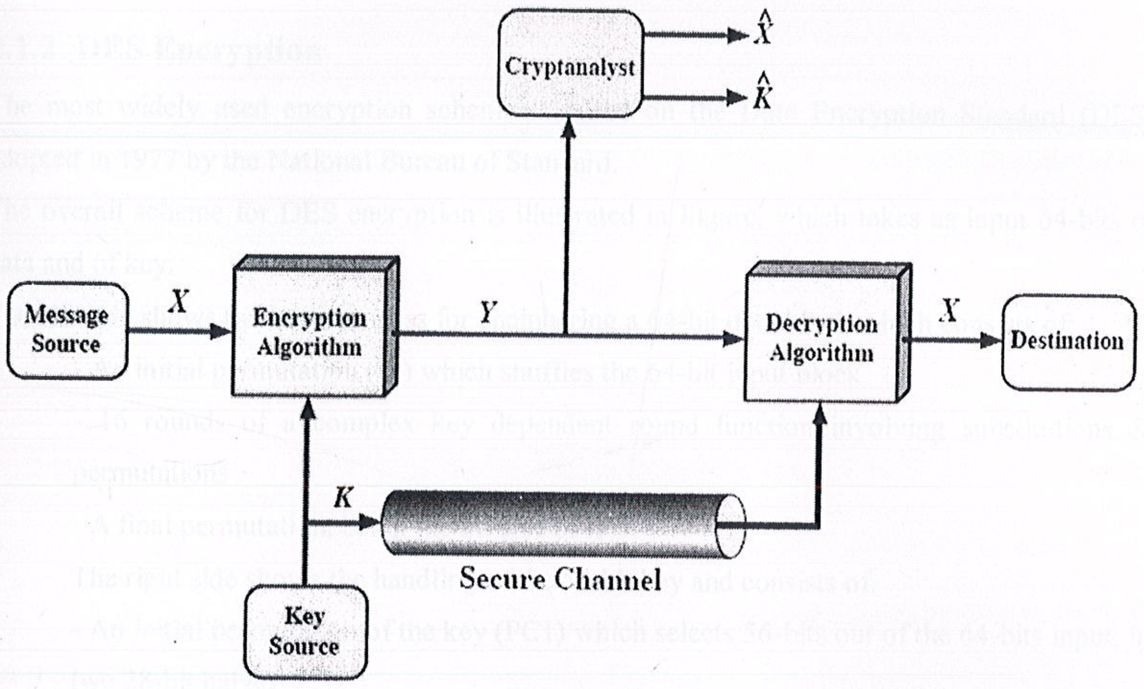


Figure 2.1 Symmetric-key cryptography

2.1.1 Stream Cipher and Block Ciphers

A **Block Cipher** is one in which a block of plain text is treated as a whole and used to produced a ciphertext block of equal length. Typically, a block size of 64 or 124 bit is used.
Most symmetric block encryption algorithms in current use are based on a structure referred to as a Feistel block cipher. A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits. An arbitrary reversible substitution cipher for a large block size is not practical, however, from an implementation and performance point of view. In general, for an *n*-

bit general substitution block cipher, the size of the key is $n \times 2^n$. For a 64-bit block, which is a desirable length to thwart statistical attacks, the key size is $64 \times 2^{64} = 2^{70} = 10^{21}$ bits.

Stream ciphers, in contrast to the 'block' type, create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by-character, somewhat like the one-time pad. In a stream cipher, the output stream is created based on a hidden internal state which changes as the cipher operates.

2.1.2 DES Encryption

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standard.

The overall scheme for DES encryption is illustrated in Figure, which takes as input 64-bits of data and of key.

The left side shows the basic process for enciphering a 64-bit data block which consists of:

- An initial permutation (IP) which shuffles the 64-bit input block
- 16 rounds of a complex key dependent round function involving substitutions & permutations
- A final permutation, being the inverse of IP

The right side shows the handling of the 56-bit key and consists of:

- An initial permutation of the key (PC1) which selects 56-bits out of the 64-bits input, in two 28-bit halves
- 16 stages to generate the 48-bit subkeys using a left circular shift and a permutation of the two 28-bit halves.

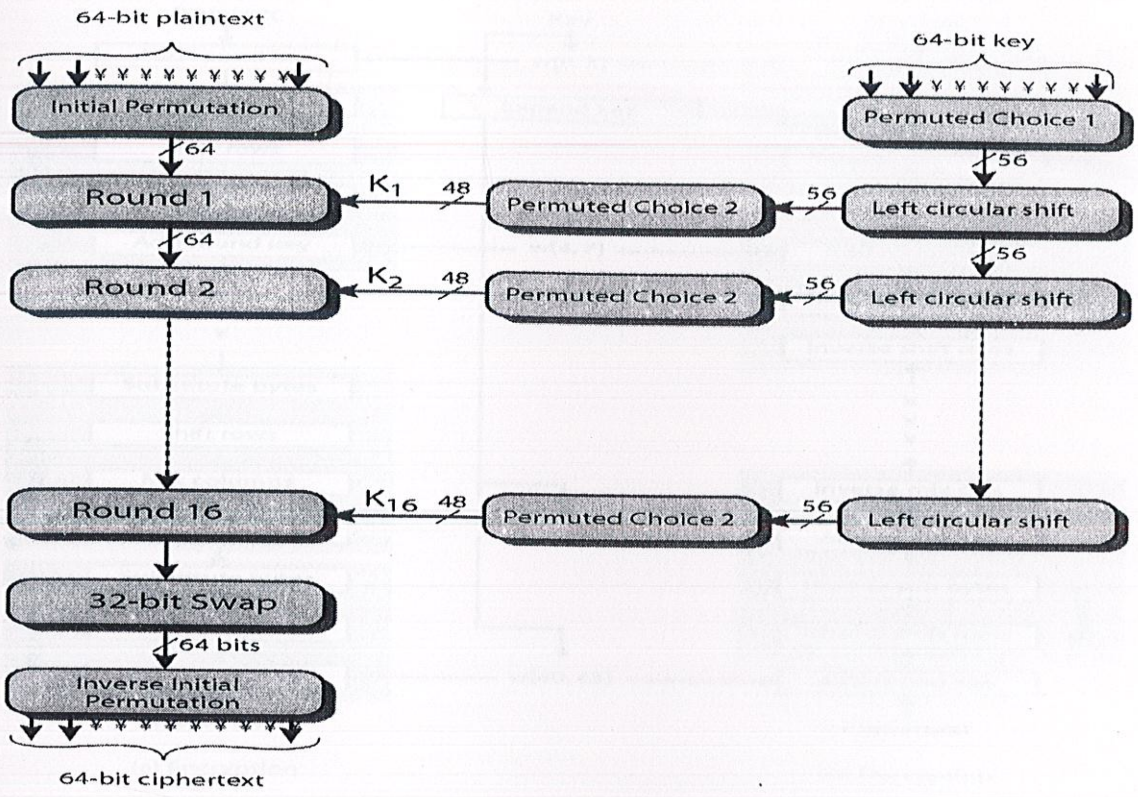


Figure 2.2 General Description of DES Encryption Algorithm

2.1.3 AES Encryption

The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192 or 256 bits. The AES specification uses the same three key size alternatives but limits the block length to 128 bits. A number of AES parameter depends on the key length.

Rijndael was designed to have the following characteristics:

- Resistance against all known attack.
- Speed and code compactness on a wide range of platform.
- Design simplicity

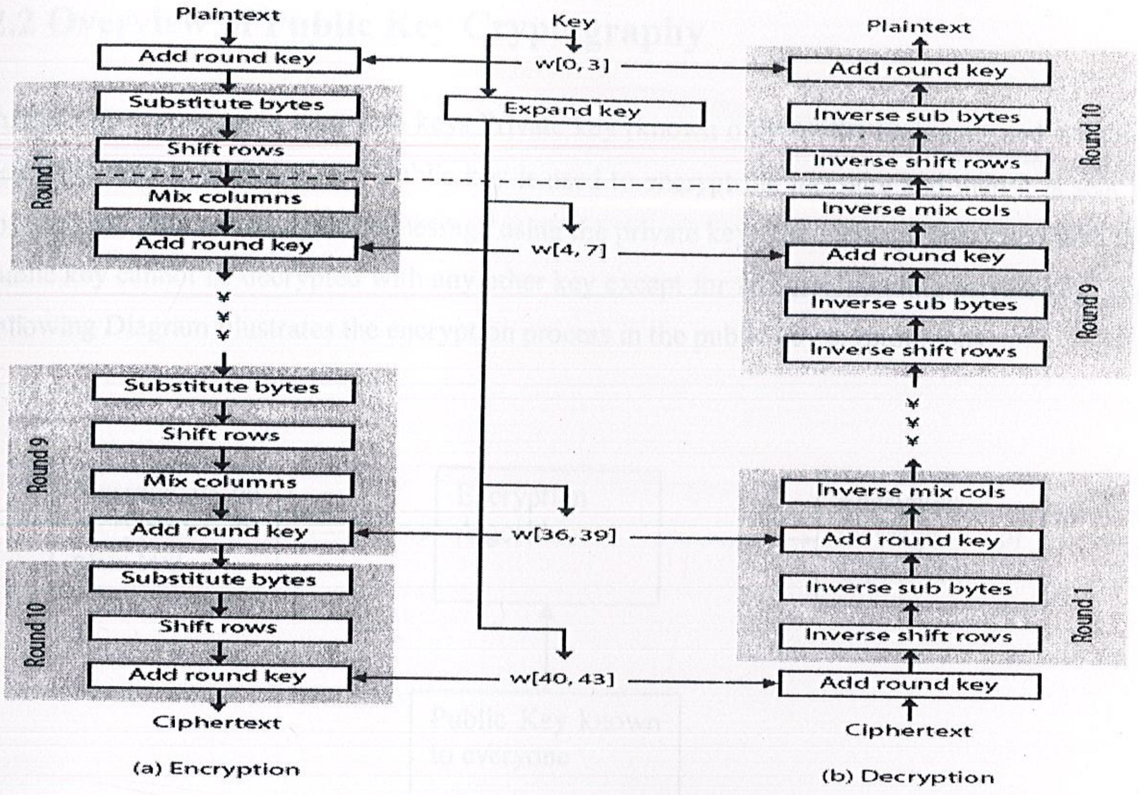


Figure 2.3 AES Encryption and Decryption

1. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinct words (128 bits) serve as a round key for each round; these are indicated in Figure 2.3.
2. Four different stages are used, one of permutation and three of substitution:
   - Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block
   - ShiftRows: A simple permutation
   - MixColumns: A substitution that makes use of arithmetic over GF(2^8)
   - AddRoundKey: A simple bitwise XOR of the current block with a portion of the expanded key
3. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.

2.2 Overview of Public Key Cryptography

Public Key cryptography uses two keys Private key (known only by the recipient) and a Public key (known to everybody). The public key is used to encrypt the message and then it is sent to the recipient who can decrypt the message using the private key. The message encrypted with the public key cannot be decrypted with any other key except for its corresponding private key. The following Diagram illustrates the encryption process in the public key cryptography

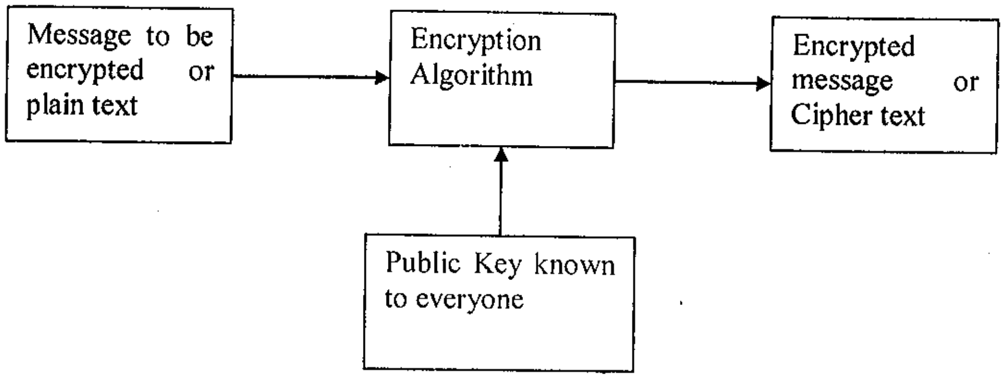

Message to be encrypted or plain text
Encryption Algorithm
Encrypted message or Cipher text
Public Key known to everyone

Figure 2.4 Public Key Encryption

The following diagram illustrates the decryption process in the public key cryptography:

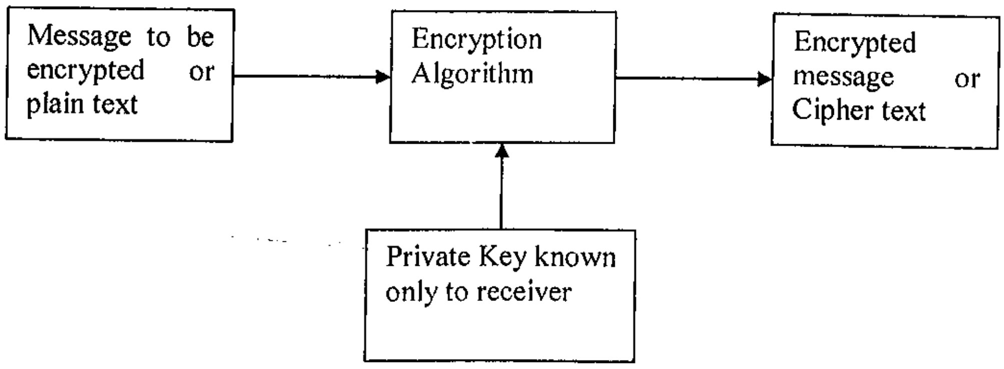

Message to be encrypted or plain text
Encryption Algorithm
Encrypted message or Cipher text
Private Key known only to receiver

Figure 2.5 Public Key Decryption

The public-key algorithm uses a one-way function to translate plaintext to ciphertext. Then, without the private key, it is very difficult for anyone (including the sender) to reverse the process (i.e., translate the ciphertext back to plaintext). A one-way function is a function that is easy to apply, but extremely difficult to invert. The most common one-way function used in public-key cryptography involves factoring very large numbers. The idea is that it is relatively easy to multiply numbers, even large ones, with a computer; however, it is very difficult to factor large numbers. The only known algorithms basically have to do a sort of exhaustive search (Does 2 go in to? Does 3? 4? 5? 6? and so on). With numbers 128 bits long, such a search requires performing as many tests as there are particles in the universe.

For instance, someone wishing to receive encrypted messages can multiply two very large numbers together. He keeps the two original numbers a secret, but sends the product to anyone who wishes to send him a message. The encryption/decryption algorithm is based upon combining the public number with the plaintext. Because it is a one-way function, the only way to reverse the process is to use one of the two original numbers. However, assuming the two original numbers are very large, their product is even bigger; it would be impractical for an adversary to try every possibility to determine what the two original numbers were.

Let us take a closer look at the essential element of a public-key encryption scheme. There is some source A that produces s message in plaintext, $X=[X_1, X_2... X_M]$. The M elements of X are some finite alphabet. The message is intended for destination B. B generates a related pair of keys: A public key PU_b, and a private key PR_b, PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.

With the message X and the encryption key PU_b as input, A forms the ciphertext $Y=[Y_1, Y_2... Y_M]$.

$$Y=E\ (PU_b,X)$$

The intended receiver, in procession of the matching private key, is able to invert the transformation:

$$X=D\ (PR_b,Y)$$

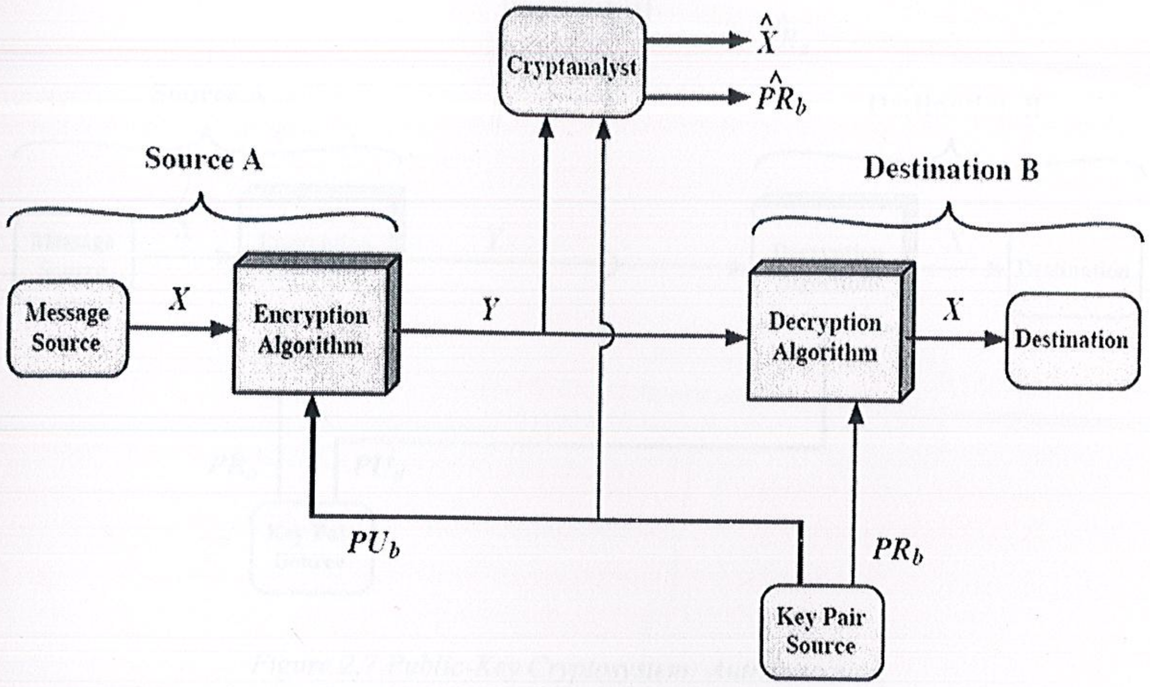


Figure 2.6 Public-Key Cryptosystem: secrecy

An adversary, observing Y and having access to PU_b, but not having access to PR_b *or* X, must attempt to recover X and/or PR_b. It is assumed that the adversary does have knowledge of the encryption (E) and the decryption (D) algorithms. If the adversary is interested only in this particular message, then the focus of effort is to recover X, by generating a plaintext estimate $\hat{}X$. Often, however, the adversary is interested in being able to read future message as well, in which case an attempt is made to recover PR_b by generating an estimate $\hat{}PR_b$.

We mentioned earlier that either of the two related keys can be used for encryption, with the other being used for decryption. This enables a rather different cryptographic scheme to be implemented. Whereas the scheme illustrated in figure provides confidentiality.

$$Y=E\ (PR_b,X)$$

$$X=D\ (PU_b,Y)$$

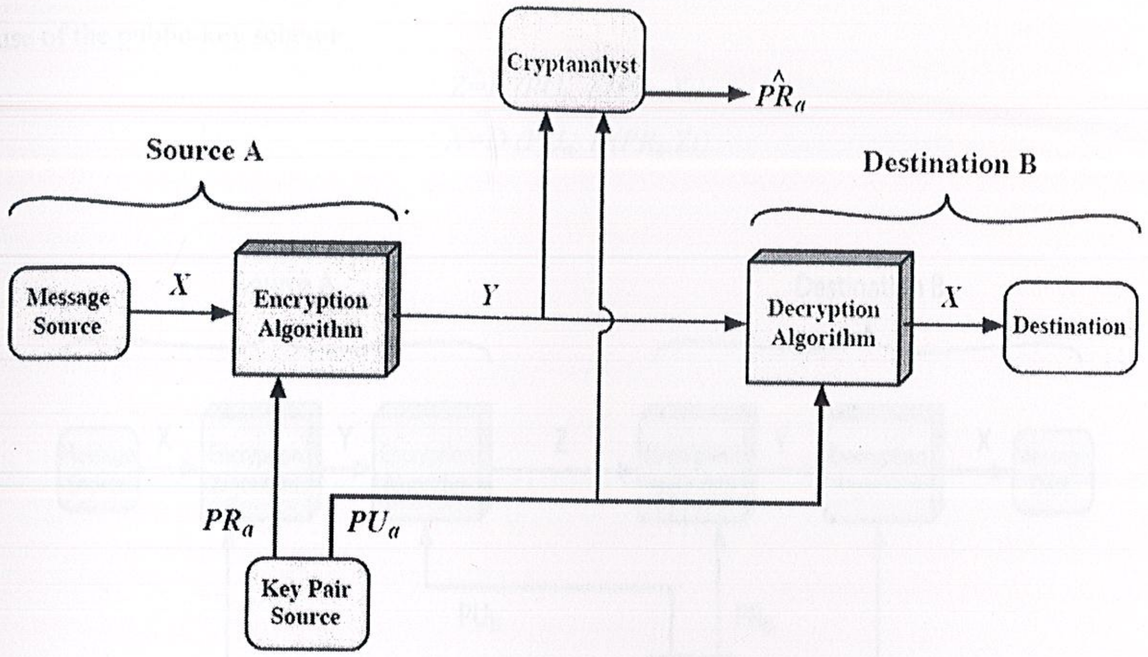


Figure 2.7 Public-Key Cryptosystem: Authentication

In this case, A prepares a message to B and encrypt it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, Only A could have prepared the message. Therefore, the entire encrypted message serves as a *digital signature,* In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in term of source and in term of data integrity.

In the preceding scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document must be kept in plaintext to be used for practical purpose. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same result is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the sender's private key, it serves as a signature that verifies origin, content, and sequencing.

It is possible to provide both the authentication function and confidentiality by a double use of the public-key scheme.

$$Z=E\ (PU_b,\ E(PR_a,X))$$
$$X=D\ (PU_a,\ D(PR_b,Z))$$

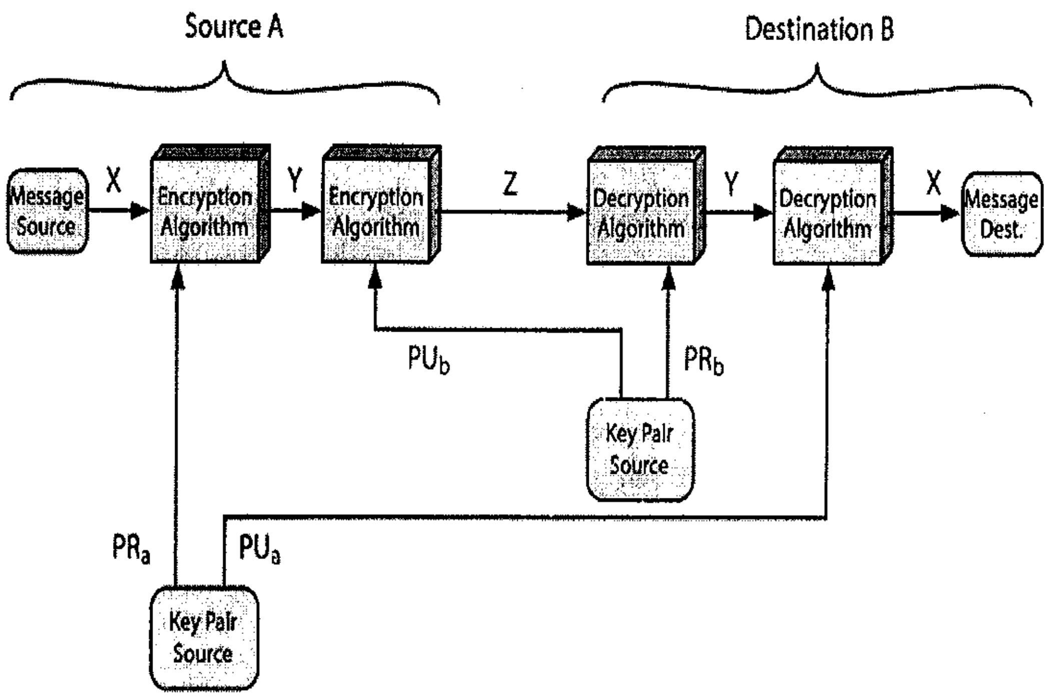


Figure 2.8 Public-Key Cryptosystem: Authentication and Secrecy

In this case, we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus confidentiality is provided.

The disadvantage of this approach is that the public key algorithm, which is complex, must be exercised four times rather than two time in each communication.

CHAPTER 3

RSA – Public Key Cryptography Algorithm

3.1 Introduction to RSA Algorithm:

RSA is one of the most popular and successful public key cryptography algorithms. The algorithm has been implemented in many commercial applications. It is named after its inventor's Ronald L. Rivest, Adi Shamir, and Leonard Adleman. They invented this algorithm in the year 1977. They utilized the fact that when prime numbers are chosen as a modulus, operations behave "conveniently". They found that if we use a prime for the modulus, then raising a number to the power (prime - 1) is 1.

RSA algorithm simply capitalizes on the fact that there is no efficient way to factor very large integers. The security of the whole algorithm relies on that fact. If someone comes up with an easy way of factoring a large number, then that's the end of the RSA algorithm. Then any message encrypted with the RSA algorithm is no more secure.

3.2RSA Algorithm

The encryption and decryption in the RSA algorithm is done as follows. Before encryption and decryption is done, we have to generate the key pair and then those keys are used for encryption and decryption.

3.2.1 Key Generation:

The first step in RSA encryption is to generate a key pair. Two keys are generated of which one is used as the public key and the other is used as the private key. The keys are generated with the help of two large prime numbers. The keys are generated as follows

1. Generate two large random primes p and q.
2. Compute n which is equal to product of those two prime numbers, n = pq
3. Compute φ(n) = (p-1)(q-1).

4. Choose an integer e, $1 < e < \varphi(n)$, such that $\gcd(e, \varphi(n)) = 1$.

5. Compute the secret exponent d, $1 < d < \varphi(n)$, such that $ed \equiv 1 \pmod{\varphi(n)}$.

6. The public key is (n, e) and the private key is (n, d). The values of p, q, and $\varphi(n)$ should also be kept secret.

- n is known as the *modulus*.
- e is known as the *public exponent* or *encryption exponent*.
- d is known as the *secret exponent* or *decryption exponent*.

3.2.2 *Encryption:*

Encryption is done using the public key component e and the modulus n. To whomever we need to send the message, we encrypt the message with their public key (e,n). Encryption is done by taking an exponentiation of the message m with the public key e and then taking a modulus of it. The following steps are done in encryption.

1. Obtain the recipient's public key (n,e)
2. Represent the plaintext message as a positive integer m < n
3. Compute the ciphertext $c = m^{\wedge e} \bmod n$.
4. Send the ciphertext c to the recipient.

3.2.3 *Decryption:*

Decryption is done using the Private key. The person who is receiving the encrypted message uses his own private key to decrypt the message. Decryption is similar to the encryption except that the keys used are different.

1. Recipient uses his private key (n,d) to compute $m = c^{\wedge d} \bmod n$.
2. Extract the plaintext from the integer representative m.

The RSA algorithm has been implemented in many applications and it is currently one of the most popularly used encryption algorithm.

3.2.4 *Digital signing*

Digital signatures are always computed with private key. This makes them easily verifiable publicly with the public key.

The raw message *m* is never signed directly. Instead it is usually hashed with hash function and the message digest is signed. This condition usually also means that the message *m* in fact is not secret to the parties so that each party can compute the message digest separately. It is also possible to use so called redundancy function instead of hash function. This function is reversible which makes it possible to sign secret messages since the message can be retrieved by the party verifying the signature. In practice hash function is often used.

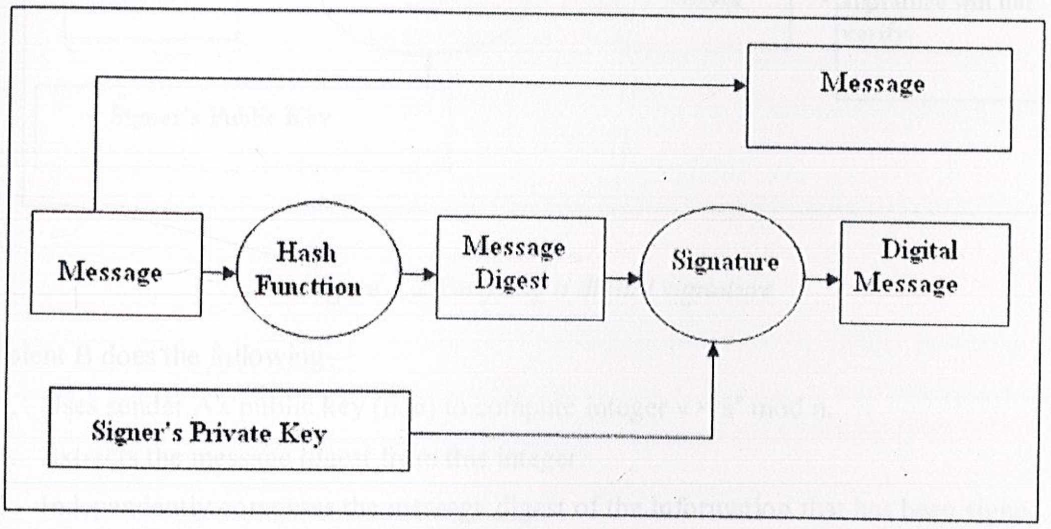


Figure 3.1 Creating a digital signature

Sender A does the following:-

1 Creates a *message digest* of the information to be sent.

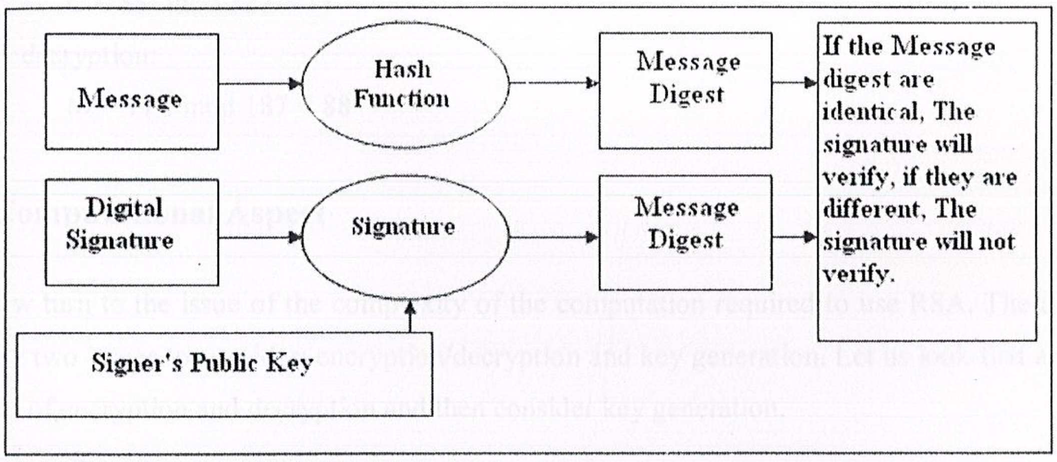
2 Represents this digest as an integer *m* between 0 and *n*-1.

3 Uses her *private* key (n, d) to compute the signature s = m^d mod n.

4 Sends this signature *s* to the recipient, B.

3.2.5 Signature verification

Same issue of authenticity of public key with public key encryption applies also to signature verification. Since the signatures are always verified with public key the public key must be obtained and verified before the signature can be reliably verified.



Message
Hash Function
Message Digest
If the Message digest are identical, The signature will verify, if they are different, The signature will not verify.
Digital Signature
Signature
Message Digest
Signer's Public Key

Figure 3.2 Verifying a digital signature

Recipient B does the following—

1. Uses sender A's public key (n, e) to compute integer $v = s^e \bmod n$.
2. Extracts the message digest from this integer.
3. Independently computes the message digest of the information that has been signed.
4. If both message digests are identical, the signature is valid.

Example of RSA

1. Select two primes numbers : p=17 & q=11
2. Compute $n = pq$ =17 x 11=187
3. Compute $ø(n)$=(p–1)(q-1)=16 x 10=160
4. Select e such that e is relatively prime to $ø(n)$=160 and less than $ø(n)$: gcd(e,160)=1; we choose e=7
5. Determine d such that de=1(mod 160) and d < 160

   Value is d=23 because 23*7=161= 10*16+1

Publish public key PU= {7,187}

Keep secret private key PR= {23,187}

RSA encryption/decryption:

Given message M = 88 (88<187)

1. encryption:

   $C = 88^7 \bmod 187 = 11$

2. decryption:

   $M = 11^{23} \bmod 187 = 88$

3.3 Computational Aspect

We now turn to the issue of the complexity of the computation required to use RSA. There are actually two issues to consider: encryption/decryption and key generation. Let us look first at the process of encryption and decryption and then consider key generation.

3.3.1 Exponentiation in Modular Arithmetic

Both encryption and decryption in RSA involve raising an integer to an integer power, mod n. If the exponentiation is done over the integers and then reduced modulo n, the intermediate values would be gargantuan. Fortunately, as the preceding example shows, we can make use of a property of modular arithmetic:

$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Thus, we can reduce intermediate results modulo n. This makes the calculation practical.

Another consideration is the efficiency of exponentiation, because with RSA we are dealing with potentially large exponents. To see how efficiency might be increased, consider that we wish to compute x^{16}. A straightforward approach requires 15 multiplications:

$$x^{16} = x * x * x * x * x * x * x * x * x * x * x * x * x * x * x * x$$

However, we can achieve the same final result with only four multiplications if we repeatedly take the square of each partial result, successively forming x^2, x^4, x^8, x^{16}. As another example, suppose we wish to calculate x^{11} mod n for some integers x and n. Observe that $x^{11} = x^{1+2+8} = (x)(x^2)(x^8)$. In this case we compute x mod n, x^2 mod n, x^4 mod n, and x^8 mod n and then calculate [(x mod n) x (x^2 mod n) x (x^8 mod n) mod n.

To solve this Problem efficiently we suggest procedure called—"Exponentiation by repeated squaring and multiplication."

— *Convert the exponential value into binary digit.*

— *Start with a "squares" value (s) equal x and an "accumulated" value (a) equal 1.*

— *Reading from least significant bit to most significant,*

— *When there is a 1 in the binary notation, multiply a by s. square s.*

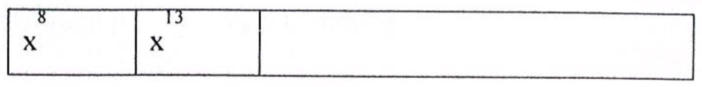
— *When there is a 0 in the binary notation, don't multiply a by s. Keep squaring s.*

<u>Example:</u>

To find x^{13}, Write the exponent in binary notation.

13=1101

S	**a**	
x^1	1	Least significant bit of exponent is 1, so multiply a = a * s
x^1	x^1	Square s
x^2	x^1	Next bit is 0, so don't multiply
x^2	x^1	Square s
x^4	x^1	Next bit is 1, so multiply
x^4	x^5	Square s
x^8	x^5	Highest bit is 1, so multiply



$$x^{13} = x^{1101} = x^{8} \quad * x^{4} \quad * x^{1}$$

3.3.2 *Efficient Operation Using the Public Key*

To speed up the operation of the RSA algorithm using the public key, a specific choice of e is usually made. The most common choice is 65537 (2^{16} 1); two other popular choices are 3 and 17. Each of these choices has only two 1 bits and so the number of multiplications required to perform exponentiation is minimized.

However, with a very small public key, such as e = 3, RSA becomes vulnerable to a simple attack. Suppose we have three different RSA users who all use the value e = 3 but have unique values of n, namely n_1, n_2, n_3. If user A sends the same encrypted message M to all three users, then the three ciphertexts are $C_1 = M^3 \bmod n_1$; $C_2 = M^3 \bmod n_2$; $C_3 = M^3 \bmod n_3$. It is likely that n_1, n_2, and n_3 are pair wise relatively prime. Therefore, one can use the Chinese remainder theorem (CRT) to compute $M^3 \bmod (n_1n_2n_3)$. By the rules of the RSA algorithm, M is less than each of the n_i therefore $M^3 < n_1n_2n_3$. Accordingly, the attacker need only compute the cube root of M^3. This attack can be countered by adding a unique pseudorandom bit string as padding to each instance of M to be encrypted. This approach is discussed subsequently.

The reader may have noted that the definition of the RSA algorithm requires that during key generation the user selects a value of e that is relatively prime to f(n). Thus, for example, if a user has preselected e = 65537 and then generated primes p and q, it may turn out that gcd(f(n),e)= 1, Thus, the user must reject any value of p or q that is not congruent to 1 (mod 65537).

3.3.3 *Efficient Operation Using the Private Key*

We cannot similarly choose a small constant value of d for efficient operation. A small value of d is vulnerable to a brute-force attack and to other forms of cryptanalysis. However, there is a way to speed up computation using the CRT. We wish to compute the value $M = C^d \bmod n$. Let us define the following intermediate results:

$V_p = C^d \bmod p \qquad V_q = C^d \bmod q$

$X_p = q \times (q^1 \bmod p) \qquad X_q = p \times (p^1 \bmod q)$

$M = (V_pX_p + V_qX_q) \bmod n$

Further, we can simplify the calculation of V_p and V_q using Fermat's theorem, which states that $a^{pl} = 1 \pmod p$ if p and a are relatively prime. Some thought should convince you that the following are valid:

$$V_p = C^d \bmod p = C^{d \bmod (p1)} \bmod p \qquad V_q = C^d \bmod q = C^{d \bmod (q1)} \bmod q$$

The quantities d mod (P1) and d mod (q1) can be precalculated. The end result is that the calculation is approximately four times as fast as evaluating $M = C^d \bmod n$ directly.

3.3.4 Key Generation

Before the application of the public-key cryptosystem, each participant must generate a pair of keys. This involves the following tasks:

- Determining two prime numbers, p and q
- Selecting either e or d and calculating the other

The procedure for picking a prime number is as follows.

1. Pick an odd integer n at random (e.g., using a pseudorandom number generator).
2. Pick an integer a < n at random.
3. Perform the probabilistic primality test, such as Miller-Rabin, with a as a parameter. If n fails the test, reject the value n and go to step 1.
4. If n has passed a sufficient number of tests, accept n; otherwise, go to step 2.

Input: $n > 3$, an odd integer to be tested for primality.
Input: *k*, a parameter that determines the accuracy of the test
Output: *composite* if *n* is composite, otherwise *probably prime*

```
write $n-1$ as $2^s \cdot d$ with $d$ odd by factoring powers of 2 from $n-1$
LOOP: repeat $k$ times:
    pick $a$ randomly in the range $[2, n-2]$
    $x \leftarrow a^d \bmod n$
    if $x = 1$ or $x = n-1$ then do next LOOP
    for $r = 1 .. s-1$
        $x \leftarrow x^2 \bmod n$
        If $x = 1$ then return composite
        if $x = n-1$ then do next LOOP
    return composite
return probably prime
```

This algorithm is $O(k \log^3 n)$, where k is the number of different values of a we test.

3.4 Key management

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard:

- The distribution of public keys
- The use of public-key encryption to distribute secret keys

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated in Figure. The scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. The following steps (matched by number to Figure) occur

1. A sends a times tamped message to the public-key authority containing a request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth} Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:
    - B's Public Key, PU_b which A can use to encrypt messages destined for B.
    - The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority.
    - The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key.

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.
5. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable.
6. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.
7. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.

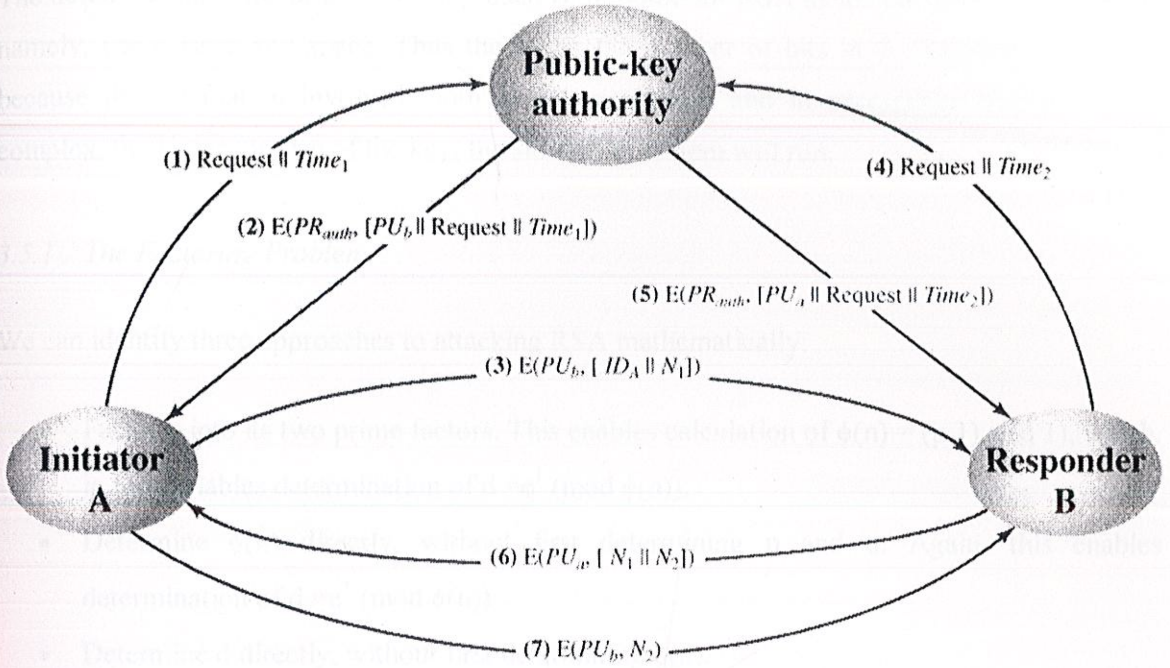

Public-key authority
(1) Request ‖ $Time_1$
(2) E(PR_{auth}, [PU_b ‖ Request ‖ $Time_1$])
(4) Request ‖ $Time_2$
(5) E(PR_{auth}, [PU_a ‖ Request ‖ $Time_2$])
(3) E(PU_b, [ID_A ‖ N_1])
Initiator A
Responder B
(6) E(PU_a, [N_1 ‖ N_2])
(7) E(PU_b, N_2)

Figure 3.3 Public-Key Distribution Scenario

Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use, a echnique known as caching. Periodically, a user should request fresh copies of the public keys its correspondents to ensure currency.

5 The Security of RSA

ur possible approaches to attacking the RSA algorithm are as follow:

ute Force: This involves trying all possible private keys.

athematical Attacks: There are several approaches, all equivalent in effort to actoring the product of two primes.

iming Attacks: These depend on the running time of the decryption algorithm.

hosen Ciphertext Attacks: This type of attack exploits properties of the RSA lgorithm.

The defense against the brute-force approach is the same for RSA as for the other cryptosystem, namely, use a large key space. Thus the larger the number of bits in d, the better. However, because the calculation involved, both in key generation and in encryption/ decryption, are complex, the larger the size of the key, the slower the system will run.

3.5.1 The Factoring Problem

We can identify three approaches to attacking RSA mathematically:

- Factor n into its two prime factors. This enables calculation of $\phi(n) = (p\ 1) \times (q\ 1)$, which, in turn, enables determination of $d = e^1 \pmod{\phi(n)}$.
- Determine $\phi(n)$ directly, without first determining p and q. Again, this enables determination of $d = e^1 \pmod{\phi(n)}$.
- Determine d directly, without first determining $\phi(n)$.

Most discussions of the cryptanalysis of RSA have focused on the task of factoring n into its two prime factors. Determining $\phi(n)$ given n is equivalent to factoring n. With presently known algorithms, determining d given e and n appears to be at least as time-consuming as the factoring problem.

For a large n with large prime factors, factoring is a hard problem, but not as hard as it used to be. A striking illustration of this is the following.

Number of Decimal Digits	**Approximate Number of Bits**	**Date Achieved**	**MIPS-years**	**Algorithm**
100	332	April 1991	7	Quadratic sieve
110	365	April 1992	75	Quadratic sieve
120	398	June 1993	830	Quadratic sieve
129	428	April 1994	5000	Quadratic sieve
130	431	April 1996	1000	Generalized number field sieve
140	465	February 1999	2000	Generalized number field sieve

Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
155	512	August 1999	8000	Generalized number field sieve
160	530	April 2003		Lattice sieve
174	576	December 2003		Lattice sieve
200	663	May 2005		Lattice sieve

The threat to larger key sizes is twofold: the continuing increase in computing power, and the continuing refinement of factoring algorithms. Thus, we need to be careful in choosing a key size for RSA. For the near future, a key size in the range of 1024 to 2048 bits seems reasonable

3.5.2 *Timing Attack*

Had a new category of attacks developed by Paul Kocher in mid-1990's, based on observing how long it takes to compute the cryptographic operations. Timing attacks are applicable not just to RSA, but to other public-key cryptography systems. This attack is alarming for two reasons: It comes from a completely unexpected direction and it is a ciphertext only attack. A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number.

Although the timing attack is a serious threat, there are simple countermeasures that can be used, including the following:

- **Constant exponentiation time:** Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.
- **Random delay:** Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack. Kocher points out that if defenders don't add enough noise; attackers could still succeed by collecting additional measurements to compensate for the random delays.
- **Blinding:** Multiply the ciphertext by a random number before performing exponentiation. This process prevents the attacker from knowing what ciphertext bits are

being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

3.5.3 Chosen Ciphertext Attack and Optimal Asymmetric Encryption Padding

The basic RSA algorithm is vulnerable to a chosen ciphertext attack (CCA). CCA is defined as an attack in which adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key. Thus, the adversary could select a plaintext, encrypt it with the target's public key and then be able to get the plaintext back by having it decrypted with the private key. Clearly, this provides the adversary with no new information. Instead, the adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, yield information needed for cryptanalysis.

3.6 Java Cryptography

Cryptography is a field looking at techniques for "encoding and verifying things securely". As we touched on in our cryptography introduction, **encryption** is the technique of encoding a message (or series of bytes) so that it can only be read by a party that knows some "secret" about how it's been encoded. We assume for now that they can't get the secret by directly observe the encoding/decoding process or by having access to the code in any way. For example, imagine a communication between a client and a server, where an attacker can freely observe any point in the network *between* the two machines, but not the machines themselves: they just see the bytes flowing to and fro.

3.6.1 Creating an RSA key pair in Java

RSA encryption and decryption are essentially mathematical operations. They are what are termed *exponentiation, modulo* a particular number. Because of this, **RSA keys** actually consist of numbers involved in this calculation, as follows:

- The **public key** consists of the **modulus** and a **public exponent;**
- The **private key** consists of that same **modulus** plus a **private exponent.**

Creating an RSA key pair essentially consists of picking a modulus, which is based on two random primes intended to be unique to that key pair, picking a public exponent (in practice, the common exponent 65537 is often used), then calculating the corresponding private exponent given the modulus and public exponent. Java provides the *KeyPairGenerator* class for performing this task. The essential idea is as follows:

- We create an instance of *KeyPairGenerator* suitable for generating RSA keys;
- We initialise the generator, telling it the bit length of the modulus that we require;
- We call *genKeyPair(),* which eventually returns a `KeyPair` object;
- We call *getPublic() and getPrivate()* on the latter to pull out the public and private keys.

The code looks as follows:

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
kpg.initialize(2048);
KeyPair kp = kpg.genKeyPair();
Key publicKey = kp.getPublic();
Key privateKey = kp.getPrivate();
```

Notice that we specify a **key length** of 2048 bits. Common values are 1024 or 2048. Choosing an RSA key length is a tradeoff between security and performance.

3.6.2 Saving the public and private key

In practice, we need to store the public and private keys somewhere. Typically, the private key will be placed on our server, and the public key distributed to clients. To store the key, we simply need to pull out the modulus and the public and private exponents, then write these numbers to some file (or put in whatever convenient place).

There also exist "key specification" classes— RSAPublicKeySpec and RSAPrivateKeySpec in this case— with transparent methods for pulling out the parameters that make up the key. Then, a

KeyFactory allows us to translate between Keys and their corresponding specification. It's a bit clumsy, but the code ends up as follows:

```
KeyFactory fact = KeyFactory.getInstance("RSA");
RSAPublicKeySpec pub = fact.getKeySpec(kp.getPublic(),
  RSAPublicKeySpec.class);
RSAPrivateKeySpec priv = fact.getKeySpec(kp.getPrivate(),
  RSAPrivateKeySpec.class);
saveToFile("public.key", pub.getModulus(),
  pub.getPublicExponent());
saveToFile("private.key", priv.getModulus(),
```

modulus and exponents are just *BigInteger* objects:

```
public void saveToFile(String fileName,
  BigInteger mod, BigInteger exp) throws IOException {
  ObjectOutputStream oout = new ObjectOutputStream(
    new BufferedOutputStream(new FileOutputStream(fileName)));
  try {
    oout.writeObject(mod);
    oout.writeObject(exp);
  } catch (Exception e) {
    throw new IOException("Unexpected error", e);
  } finally {
    oout.close();
  }
}
```

We end up with two files: public.key, which is distributed without clients; meanwhile, private.key, is kept secret on our server. Of course, we needn't even bother saving the keys to a file: they're just numbers, and we could embed them as constant strings in our code, then pass those strings to the constructor of BigInteger. Saving the keys to file simply makes it a bit easier to manage keys in some cases.

3.6.3 How to set the key size in Java

If we use the KeyGenerator class, then we can set the key size in bits via the init() method. The following example shows how to generate a 256-bit AES key:

```
KeyGenerator gen = KeyGenerator.getInstance("AES/CTR/PKCS5PADDING");
gen.init(256);
SecretKey k = gen.generateKey();
Cipher ciph = Cipher.getInstance("AES");
ciph.init(Cipher.ENCRYPT_MODE, k);
```

CHAPTER 4

RSA Chat Messenger

Chat servers today are readily available and very useful in conversing with people that might be close by or far away. Internet chat services like Gmail, Yahoo messenger, AOL provide the convenience of conversing with people in real time. This service provides a host of possibilities for work, school, and connectivity. Unfortunately these widely available chat services do not provide protection/privacy of what is being sent through the chat servers.

The objective of this project is to build a secure chat server utilizing Public Key encryption to send secure chat messages across the internet

Benefits of Chat Service

- Allows for "instant" communications between people.
- Use of the real time chat over the Internet can eliminate costly long distance charges.
- Ability to stay in contact with people who you normally never see.
- Allows for quick questions and quick responses.

Negatives of Chat Service

- Potential security/privacy problems of these Instant Messaging programs.
- Chat in most instances are routed through a server system where the service is provided and that is a single point where all messages can be intercepted.
- Chat programs can provide an open avenue of attack for hackers, crackers, spies and thieves.
  - Eavesdrop: intercept messages
  - Actively insert messages into connection
  - Removing sender or receiver, inserting himself in place

4.1 Implemented Solution

Encryption

– Utilize public key encryption to securely transmit messages between users.

– Encrypt each message with public key of target recipient.

– Since chat messages are normally not long, it requires less processing time for the program to encrypt the message.

– Digitally sign every message

Positives

- Very secure message exchange

- Interception can happen but the interceptor can not decipher the message.

- Insertion of data can happen but the digital signature ensures that message is authentic.

Negatives

- Much slower then symmetric encryption methods, however since messages are relatively short. The difference between the times is negligible.

- Much more hassle to encrypt and decrypt messages.

- Can be subject to man in the middle attack

4.2 Secure Chat Protocol

• 2 Clients connect to a server

• Once connected, each *client generates their public and private keys locally.*

• The public key sent to the server and is set so when a user clicks their name any messages sent will be encrypted with that public key.

• *The private key remains only in the client program.*

• When message is sent out, the client program downloads the public key and encrypts the intended message and then applies the digital signature which is created with the private key and then sends the encrypted message out.

• When the packet is received by the specified person, the client program automatically applies the private key on the text and outputs the message so that the user can see it decrypted and then double checks the digital signature with the public key.

• Once completed with each step we have successfully transmitted a secure message.

4.3 Exactly what happens when a user connects to the server

1st user enters name and clicks connect.

Public key generated

Private key generated

Public key is sent to the server

Private key remains on client

2nd user enters name and clicks connect.
Public key generated
Private Key generated
Public key is sent to the server
Private Key remains on client

1st user types a message and clicks send.
Message is displayed encrypted with 2nd user's public key.
Message is also displayed with "Decrypted: (some garbage because the private key applied is not the right key to apply to 2nd users public key.)"

2nd users screen displays encrypted message sent by 1st user.
2nd users screen also displays "Decrypted: (message decrypted with proper key)". Since the message was intended for the 2nd user, the private key of the 2nd user was the correct key applied to the decryption function. So the output is in readable plain text.

2nd user types a message and clicks send.
Message is displayed encrypted with 1st users public key.

Message is also displayed with "Decrypted: (some garbage because the private key applied is not the right key to apply to 1st users public key.)"

1st users screen displays encrypted message sent by 2nd user.
1st users screen also displays "Decrypted: (message decrypted with proper key)". Since the message was intended for the 1st user, the private key of the 1st user was the correct key applied to the decryption function. So the output is in readable plain text.

Success!!! Secure message transaction completed successfully.

4.4 Chat Messenger Architecture

In client-server applications, the server provides some service, such as processing database queries or creating new account. The client uses the service provided by the server, either displaying database query results to the user or validating the user. The communication that occurs between the client and the server must be reliable. That is, no data can be dropped and it must arrive on the client side in the same order in which the server sent it.
TCP provides a reliable, point-to-point communication channel those client-server applications on the Internet use to communicate with each other. To communicate over TCP, a client program and a server program establish a connection to one another. Each program binds a socket to its end of the connection. To communicate, the client and the server each reads from and writes to the socket bound to the connection.

4.4.1 What Is a Socket?

A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. The java.net package provides two classes--Socket and Server Socket-that implement the client side of the connection and the server side of the connection, respectively.

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.

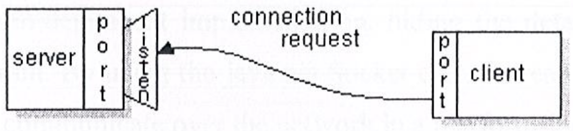


Figure 4.1 Connection Request

If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

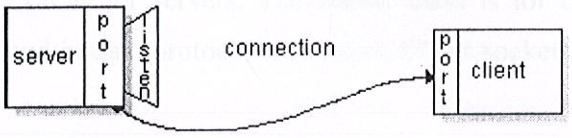


Figure 4.2 Socket Connection Established

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

The client and server can now communicate by writing to or reading from their sockets.

Definition: A *socket* is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.

An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.

The java.net package in the Java platform provides a class, Socket, that implements one side of a two-way connection between our Java program and another program on the network. The Socket class sits on top of a platform-dependent implementation, hiding the details of any particular system from your Java program. By using the java.net.Socket class instead of relying on native code, our Java programs can communicate over the network in a platform-independent fashion.

Additionally, java.net includes the ServerSocket class, which implements a socket that servers can use to listen for and accept connections to clients.

4.4.2 TCP/IP Client Socket

There are two kinds of TCP sockets in JAVA. One is for server, and the other is for clients. The ServerSocket class is designed to be a "listener", which wait for client to connect before doing anything. Thus, *ServerSocket* is for servers. The *Socket* class is for clients. It is designed to connect to server sockets and initiate protocol exchanges. Client sockets are the most commonly used by Java Application

The Creation of a **Socket** object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructor used to create client sockets.

Socket(String *hostname,* int *port)* throws UnknownHostException, IOexception	Create a socket connected to the named host and port.
Socket(InetAddress *IpAddress,* int *port)* throws IOException	Creates a socket using a preexisting **InetAddress** object and a port.

Socket defines several instance methods. For example, a **Socket** can be examined at any time for the address and port information associated with it, by using the following methods:

InetAddress getInetAddress()	Return the **InetAddress** associated with the **Socket** objects. It returns null if the socket is not connected
Int getPort()	Returns the remote port to which the invoking **Socket** objects is connected. It return 0 if the socket is not connected.
Int getLocalPort()	Return the local port to which the invoking **Socket** objects is bound. It return -1 if the socket is not bound.

We can gain access to the input and output streams associated with a *socket* by use of the **getInputStream( )** and **getOutputStream( )** methods. Each can throws an IOException is the socket has been invalidated by a loss of connection. These streams are used exactly like I/O streams to send and receive data.

InputStream getInputStream() throws IOException	Return the **InputStream** associated with the invoking socket.
OutputStream getOutputStream() throws IOException	Return the **OutputStream** associated with the invoking socket.

Several other Methods are available, including **connect( )**, which allows you to specify a new connection; **isConnected( )**, which return true if the socket is connected to the server; **isBound()**, which return true if the socket is bound to an address; and **isClosed( )**, which returns true if the socket is closed.

4.4.3 TCP/IP Server Socket

As mentioned earlier, java has a different socket class that must be used for creating server application. The **ServerSocket** class is used to create servers that listen for either local or remote client program to connect to them on published ports. **ServerSocket** are quite different from normal **Socket.** When we create a **ServerSocket,** it will register itself with the system as having an interested in client connections. The constructors for **ServerSocket** reflect the port number that we want to accept connections on and, optionally, how long we want the queue for said port to be. The queue length tell the system how many client connection it can leave pending before it should simply refuse the connections. The default is 50. The constructor might throw an **IOException** under adverse condition.

ServerSocket(int *port*) throws IOException	Creates server socket on the specific port with a queue length of 50
SereverSocket(int *port,* int *maxQueue)* throws IOException	Creates a server socket on the specific port with a maximum queue length of *maxQueue*
ServerSocket(int *port,* int *maxQueue,* inetAddress *localhost)* throws IOException	Creates a server socket on the specific port with a maximum queue length of *maxQueue*. On a multi homed host, *localaddress* specifies the IP address to which this socket binds.

ServerSocket has a method called **accept( )**, which is a blocking call that will wait for a client to initiate communication and then return with a normal **Socket** is used for communication with the client.

4.4.4 Java Swing

Some terminology: AWT, JFC, Swing

- AWT stands for *Abstract Window Toolkit.* It refers to a collection of basic GUI components, and other features, e.g. layout managers, component printing, etc.
- JFC stands for *Java Foundation Classes*. The JFC contains a much larger set of classes than the AWT set. Mostly it adds to the functionality of the AWT, however, in some cases it replaces the functionality of the AWT. The JFC contains accessibility functions, 2D drawing libraries, pluggable look and feel, etc.
- The *Swing* component set is part of the JFC. It contains a number of GUI components (buttons, textfields, scrollpanes, etc.) that are intended to be direct replacements of the corresponding AWT GUI components. Swing only replaces a subsection of the AWT

(the GUI components), other aspects of the AWT (e.g. layout managers, etc.) remain unchanged.

In AWT all components are termed *heavyweight* components. The reason for this is that an AWT component relies on the underlying operating system (e.g. Windows, Solaris, etc.) to provide the component (i.e. to paint it, redraw it, etc.). For example, an AWT button, when running on MacOS is actually a MacOS button. The term *heavyweight* does not really provide a clear indication of this dependency.

All Swing components are *lightweight* components. A lightweight component does not use a corresponding peer or native component. Instead, it is drawn by the Java VM (and not the underlying OS).

4.4.5 Overview of Swing Component class

JButton	bJCheckBox	JComboBox	JLabel
JList	JRadioButton	JScrollPane	JTabbedPane
JTable	JTextField	JToggleButton	JTree

These components are all lightweight, which mean that they are all derived from **JComponenet.**

Icons and Labels

In Swing, icons are encapsulated by the **ImageIcon** class, which paints an icon from an image. Two of its constructors are shown here:

ImageIcon(String filename)

ImageIcon(URL url)

The first form uses the image in the file named *filename*. The second form uses the image in the resource identified by *url*

The **ImageIcon** class implements the **Icon** interface that declares the methods shown here:

Method	Description
int getIconHeight( )	Returns the height of the icon in pixels.
int getIconWidth( )	Returns the width of the icon in pixels.
void paintIcon(Component *comp*,Graphics *g*,int *x*, int *y*)	Paints the icon at position *x*, *y* on the graphics context *g*. Additional information about the paint operation can be provided in *comp*.

Swing labels are instances of the **JLabel** class, which extends **JComponent.** It can display text and/or an icon. Some of its constructors are shown here:

JLabel(Icon i)

Label(String s)

JLabel(String s, Icon i, int align)

Here, *s* and *i* are the text and icon used for the label. The *align* argument is **LEFT, RIGHT**, or **CENTER**. These constants are defined in the **SwingConstants** interface, along with several others used by the Swing classes.

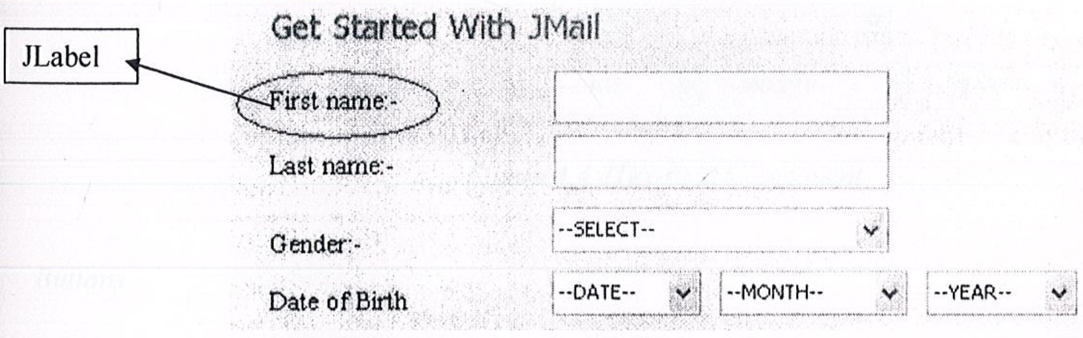


Figure 4.3 JLabel Component

The icon and text associated with the label can be read and written by the following methods:

Icon getIcon( )

String getText( )

void setIcon(Icon i)

void setText(String s)

Text Fields

The Swing text field is encapsulated by the **JTextComponent** class, which extends **JComponent**. It provides functionality that is common to Swing text components. One of its subclasses is **JTextField**, which allows you to edit one line of text. Some of its constructors are shown here:

```
JTextField( )
JTextField(int cols)
JTextField(String s, int cols)
JTextField(String s)
```

Here, *s* is the string to be presented, and *cols* is the number of columns in the text field.

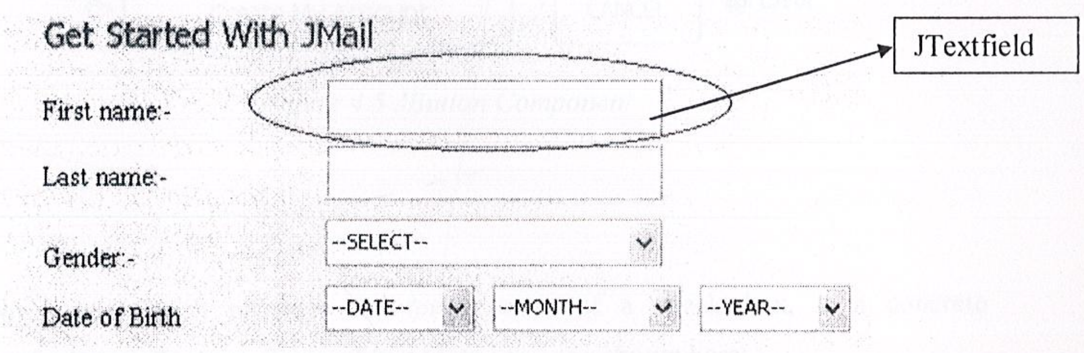


Figure 4.4 JTextfield Component

Buttons

Swing buttons provide features that are not found in the **Button** class defined by the AWT. For example, you can associate an icon with a Swing button. Swing buttons are subclasses of the **AbstractButton** class, which extends **JComponent**. **AbstractButton** contains many methods that allow you to control the behavior of buttons, check boxes, and radio buttons. For example, we can define different icons that are displayed for the component when it is disabled, pressed, or selected. Another icon can be used as a *rollover* icon, which is displayed when the mouse is positioned over that component.

The JButton Class

The **JButton** class provides the functionality of a push button. **JButton** allows an icon, a string, or both to be associated with the push button. Some of its constructors are shown here:

JButton(Icon i)

JButton(String s)

JButton(String s, Icon i)

Here, *s* and *i* are the string and icon used for the button.

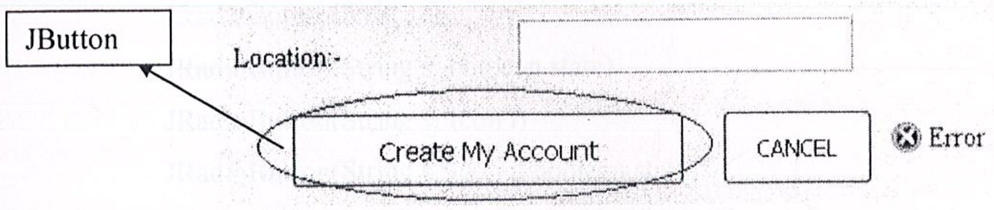


Figure 4.5 JButton Component

Check Boxes

The **JCheckBox** class, which provides the functionality of a check box, is a concrete implementation of **AbstractButton**. Some of its constructors are shown here:

JCheckBox(Icon i)

JCheckBox(Icon i, boolean state)

JCheckBox(String s)

JCheckBox(String s, boolean state)

JCheckBox(String s, Icon i)

JCheckBox(String s, Icon i, boolean state)

Here, *i* is the icon for the button. The text is specified by *s*. If *state* is **true**, the check box is initially selected. Otherwise, it is not.

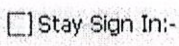


Figure 4.6 JCheckBox Component

The state of the check box can be changed via the following method:

```
void setSelected(boolean state)
```

Here, *state* is **true** if the check box should be checked.

Radio Buttons

Radio buttons are supported by the **JRadioButton** class, which is a concrete implementation of **AbstractButton**. Some of its constructors are shown here:

```
JRadioButton(Icon i)
JRadioButton(Icon i, boolean state)
JRadioButton(String s)
JRadioButton(String s, boolean state)
JRadioButton(String s, Icon i)
JRadioButton(String s, Icon i, boolean state)
```

Here, *i* is the icon for the button. The text is specified by *s*. If *state* is **true**, the button is initially selected. Otherwise, it is not.

Radio buttons must be configured into a group. Only one of the buttons in that group canbe selected at any time.

Combo Boxes

Swing provides a *combo box* (a combination of a text field and a drop-down list) through the **JComboBox** class, which extends **JComponent**. A combo box normally displays one entry. However, it can also display a drop-down list that allows a user to select a different entry. You can also type your selection into the text field. Two of **JComboBox**'s constructors are shown here:

```
JComboBox( )
JComboBox(Vector v)
```

Here, *v* is a vector that initializes the combo box.

Items are added to the list of choices via the **addItem( )** method.

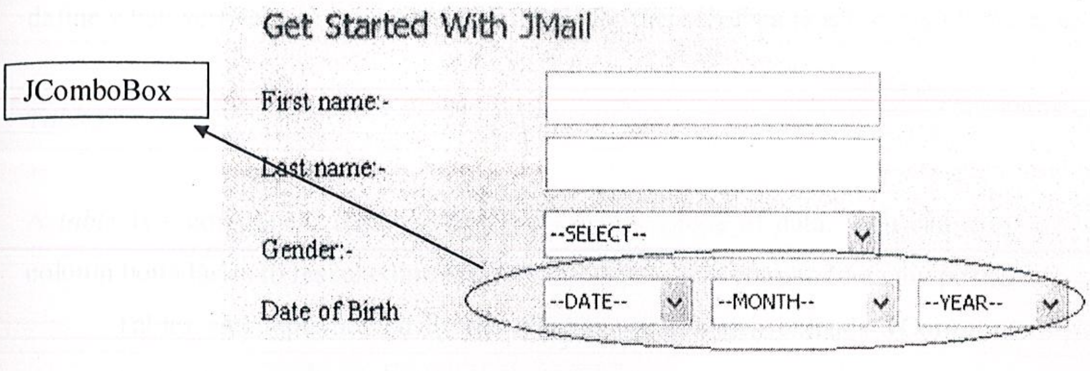


Figure 4.7 JComboBox Component

Tabbed Panes

A *tabbed pane* is a component that appears as a group of folders in a file cabinet. Each folder has a title. When a user selects a folder, its contents become visible. Only one of the folders may be selected at a time. Tabbed panes are commonly used for setting configuration options.

Tabbed panes are encapsulated by the **JTabbedPane** class, which extends **JComponent**. We will use its default constructor. Tabs are defined via the following method:

void addTab(String str, Component comp)

Here, *str* is the title for the tab, and *comp* is the component that should be added to the tab. Typically, a **JPanel** or a subclass of it is added.

Scroll Panes

A *scroll pane* is a component that presents a rectangular area in which a component may be viewed. Horizontal and/or vertical scroll bars may be provided if necessary. Scroll panes are implemented in Swing by the **JScrollPane** class, which extends **JComponent**. Some of its constructors are shown here:

JScrollPane(Component comp)

JScrollPane(int vsb, int hsb)

JScrollPane(Component comp, int vsb, int hsb)

Here, *comp* is the component to be added to the scroll pane. *vsb* and *hsb* are **int** constants that define when vertical and horizontal scroll bars for this scroll pane are shown.

Tables

A *table* is a component that displays rows and columns of data. You can drag the cursor on column boundaries to resize columns. You can also drag a column to a new position.

Tables are implemented by the **JTable** class, which extends **JComponent**. One of its constructors is shown here:

JTable(Object data[ ][ ], Object colHeads[ ])

Here, *data* is a two-dimensional array of the information to be presented, and *colHeads* is a one-dimensional array with the column headings.

Class ArrayList

Resizable- array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list.

Each ArrayList instance has a *capacity*. The capacity is the size of the array used to store the elements in the list. It is always at least as large as the list size. As elements are added to an ArrayList, its capacity grows automatically.

Note that this implementation is not synchronized. If multiple threads access an ArrayList instance concurrently, and at least one of the threads modifies the list structurally, it *must* be synchronized externally. (A structural modification is any operation that adds or deletes one or more elements, or explicitly resizes the backing array; merely setting the value of an element is not a structural modification.) This is typically accomplished by synchronizing on some object that naturally encapsulates the list. If no such object exists, the list should be "wrapped" using the Collections.synchronizedList method. This is best done at creation time, to prevent accidental unsynchronized access to the list:

List list = Collections.synchronizedList(new ArrayList(...));

Constructor Summary

`ArrayList``()`	Constructs an empty list with an initial capacity of ten.
`ArrayList``(Collection<? extends E> c)`	Constructs a list containing the elements of the specified collection, in the order they are returned by the collections iterate.
`ArrayList`` (int initialCapacity)`	Constructs an empty list with the specified initial capacity

Working with Java SQL Package

The java.sql package contains API for the following:

- Making a connection with a database via the DriverManager facility
  - DriverManager class -- makes a connection with a driver
  - SQLPermission class -- provides permission when code running within a Security Manager, such as an applet, attempts to set up a logging stream through the DriverManager
  - Driver interface -- provides the API for registering and connecting drivers based on JDBC technology ("JDBC drivers"); generally used only by the DriverManager class
  - DriverPropertyInfo class -- provides properties for a JDBC driver; not used by the general user
- Sending SQL statements to a database
  - Statement -- used to send basic SQL statements
  - PreparedStatement -- used to send prepared statements or basic SQL statements (derived from Statement)
  - CallableStatement -- used to call database stored procedures (derived from PreparedStatement)
  - Connection interface -- provides methods for creating statements and managing connections and their properties

  - Savepoint -- provides savepoints in a transaction
- Retrieving and updating the results of a query
  - ResultSet interface
- Standard mappings for SQL types to classes and interfaces in the Java programming language
  - Array interface -- mapping for SQL ARRAY
  - Blob interface -- mapping for SQL BLOB
  - Clob interface -- mapping for SQL CLOB
  - Date class -- mapping for SQL DATE
  - Ref interface -- mapping for SQL REF
  - Struct interface -- mapping for SQL STRUCT
  - Time class -- mapping for SQL TIME
  - Timestamp class -- mapping for SQL TIMESTAMP
  - Types class -- provides constants for SQL types
- Custom mapping an SQL user-defined type (UDT) to a class in the Java programming language
  - SQLData interface -- specifies the mapping of a UDT to an instance of this class
  - SQLInput interface -- provides methods for reading UDT attributes from a stream
  - SQLOutput interface -- provides methods for writing UDT attributes back to a stream
- Metadata
  - DatabaseMetaData interface -- provides information about the database
  - ResultSetMetaData interface -- provides information about the columns of a ResultSet object
  - ParameterMetaData interface -- provides information about the parameters to PreparedStatement commands
- Exceptions
  - SQLException -- thrown by most methods when there is a problem accessing data and by some methods for other reasons
  - SQLWarning -- thrown to indicate a warning
  - DataTruncation -- thrown to indicate that data may have been truncated

- BatchUpdateException -- thrown to indicate that not all commands in a batch update executed successfully

4.5 Screen Shot of Jmail Messenger

This section contains the screen shots of Jmail messenger.

(1) In the chat messenger first we need to start the server. IP Address and Port Number is required, so that the client can be connected to the server.

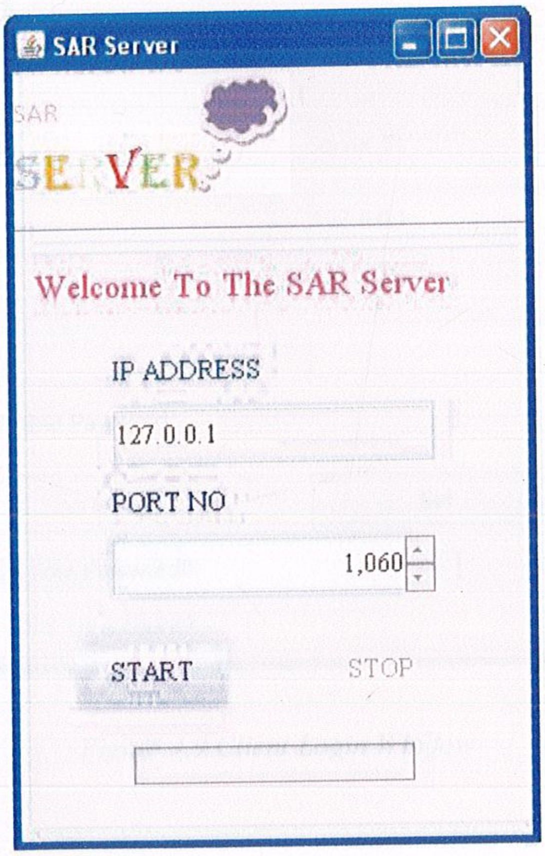
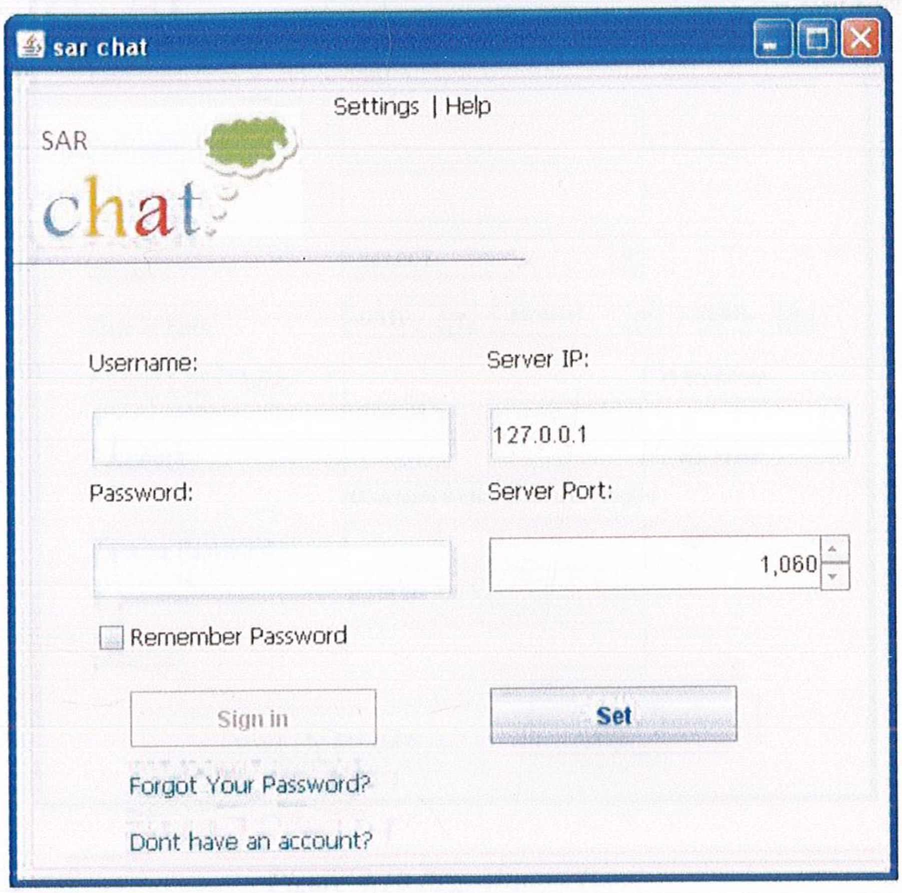


Figure 4.8 Server IP and Port Connection

(2) It takes two parameters - the server IP Address and the port number to connect to the client server. It makes a socket connection and ask for the Username and the Password to login. If the user doesn't have any account, then user can create a new account by clicking on the "don't have any account" link and register yourself.

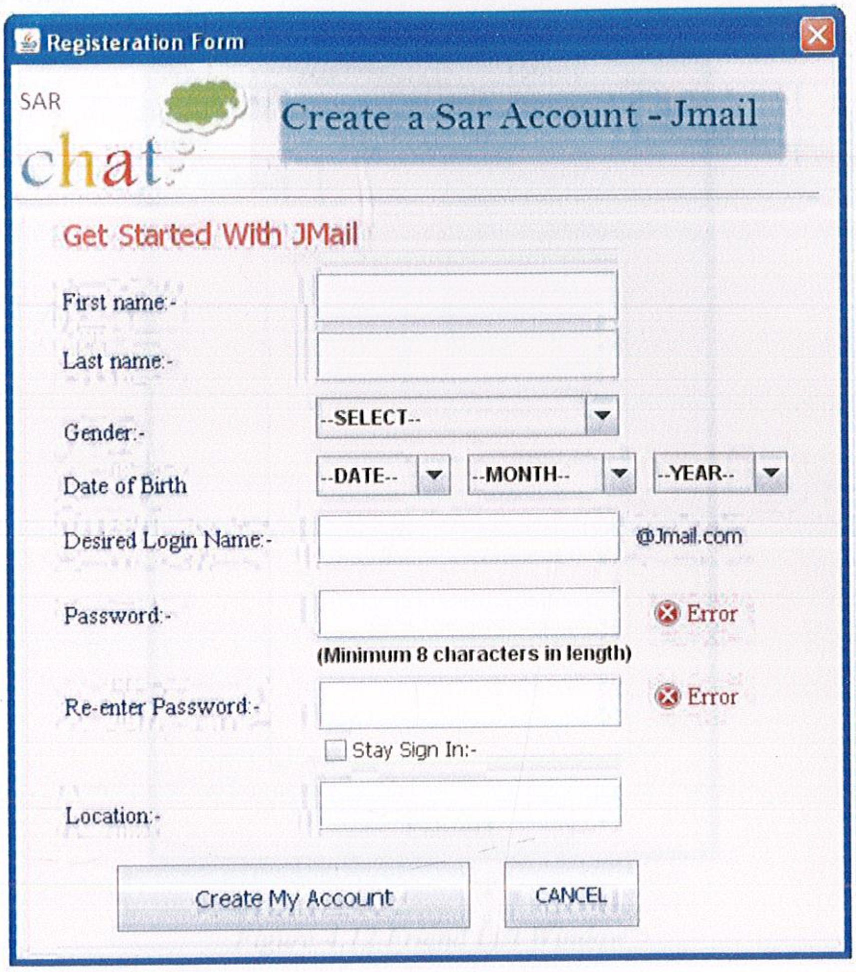


Server IP: 127.0.0.1

Server Port: 1,060

Figure 4.9 Client Login Window

(3) The user can create a Jmail account by filling up the following required field.


Registeration Form

SAR chat

Create a Sar Account - Jmail

Get Started With JMail

First name:-

Last name:-

Gender:- --SELECT--

Date of Birth --DATE-- --MONTH-- --YEAR--

Desired Login Name:- @Jmail.com

Password:- Error

(Minimum 8 characters in length)

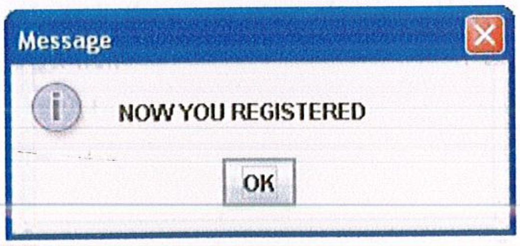
Re-enter Password:- Error

Stay Sign In:-

Location:-

Figure 4.10 Registration Form

As the required fields are filled, the client has successfully created the account.


Message

NOW YOU REGISTERED

Figure 4.11 Confirm Registration

(4) Once the client is connected to the server, the client can select friend from the friend list to chat with.

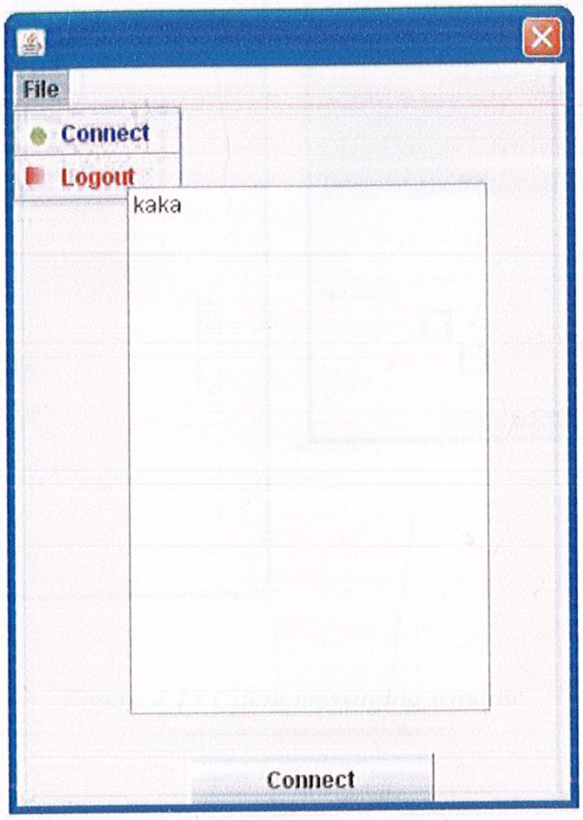
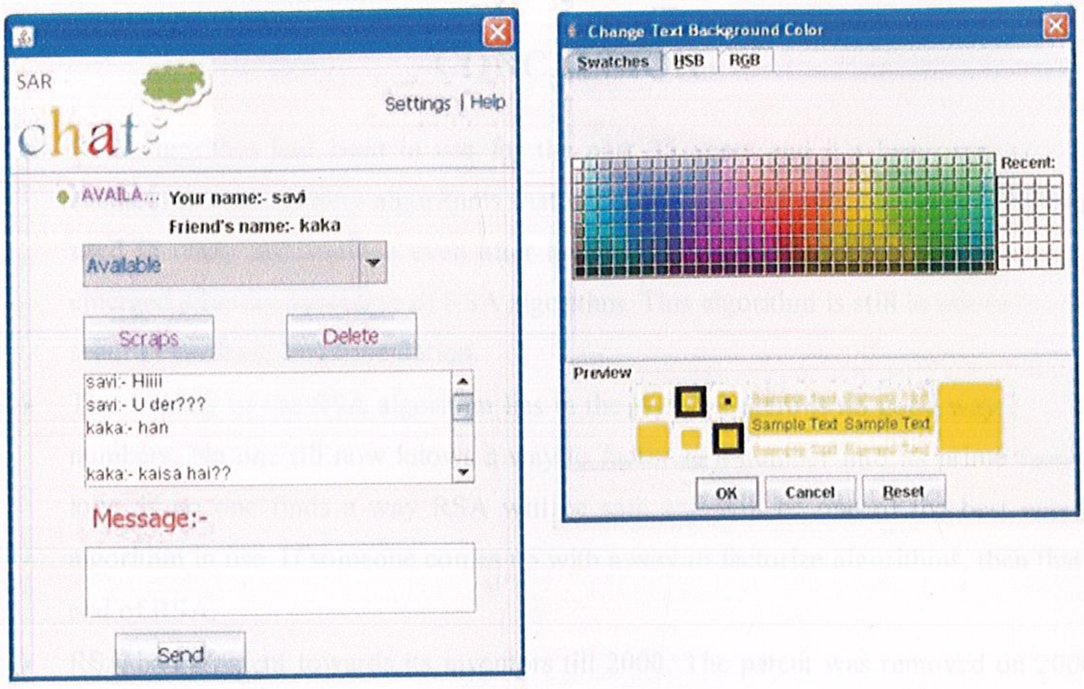


Figure 4.12 Friend List Window

(5) Once the client is selected, they can start chatting. User can also set their status as Available or Busy.



Your name:- savi
Friend's name:- kaka
Available

savi:- Hiiii
savi:- U der???
kaka:- han
kaka:- kaisa hai??

Figure 4.13 Client messaging window

CONCLUSION

- RSA algorithm had been in use for the past 25 years and it's been one of the most successful cryptography algorithms that the security world ever had. This is still widely used in many applications even after hundreds of public key cryptography algorithms emerged after the invention of RSA algorithm. This algorithm is still in use because of its security and easy implementation.
- The security of the RSA algorithm lies in the fact that there is no good way of factoring numbers. No one till now knows a way to factorize a number into its prime factors. As long as no one finds a way RSA will be safe and will be one of the best encryption algorithm in use. If someone comes up with a way to factorize algorithms, then that's the end of RSA.
- RSA had a patent towards its inventors till 2000. The patent was removed on 2000 and now it's open to all. Anybody can use it now. This makes it still easier to study the algorithm in a more detailed manner and it will certainly be reviewed by many other people all over the world and there is a lot of chances that someone will come up with another version of RSA which is lot more sophisticated than this one.
- The RSA Security inc, which held the patent for RSA had announced lot of cash prizes for the people who could come up with factors of few large numbers. The numbers are nearly 200 digits long and it is published in their website.
- Finally, a few days back the inventors of the RSA Ronald Rivest, Adi Shamir and Leonard Adleman received the Turing Award for the year 2002 for their invention of RSA algorithm. This is really great news and obviously these people deserve this award for having created such a wonderful algorithm, which dominated the world of cryptography for about 25 years and is still dominating.

BIBLIOGRAPHY

1) R. L. Rivest, A. Shamir and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, 21 (2), pp. 120-126, February 1978.
2) Rabin, M.O., Probabilistic algorithms. In Algorithms and Complexity, J. F.Traub, Ed., Academic Press, New York, 1976, pp. 21-40.
3) Merkle, R. Secure communications over an insecure channel. Submitted to Comm.ACM.
4) Diffie, W., and Hellman, M. New directions in cryptography. IEEE Trans. Inform.Theory IT-22, (Nov. 1976), 644-654.
5) Knuth, D. E. The Art of Computer Programming, Vol 2: Seminumerical Algorithms. Addison-Wesley, Reading, Mass., 1969.
6) Burt Kaliski,"The Mathematics of the RSA Public-Key Cryptosystem," RSA Laboratories. April 9,2006
7) Potter,R.J.,Electronnic mail. Science 195, (4283, March 1977), 1160-1164
8) Rania Salah EL-Sayed, Moustafa Abd EL-Azmin, Mohammad Ali Gomaa, An Efficient Signatute system using optimized RSA algorithm.
9) William Stallings, "Cryptography and Network Security Principles and practices". Fourth Edition. November 26,2005
10) Herbert schildt, "Java : The Complete reference" Fifth Edition.

PROJECT CODE

Chat Client

Main.java

```
package ui;

import javax.swing.JOptionPane;

public class Main extends javax.swing.JFrame {

    /** Creates new form Main */
    public Main() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    //    <editor-fold defaultstate="collapsed"    desc="Generated    Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        jSeparator2 = new javax.swing.JSeparator();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jMenuItem2 = new javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        jMenuItem3 = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setMinimumSize(new java.awt.Dimension(395, 345));
        getContentPane().setLayout(null);

        jLabel1.setIcon(new    javax.swing.ImageIcon(getClass().getResource("/images/sar
talk.jpg"))); // NOI18N
        jLabel1.setBorder(javax.swing.BorderFactory.createEtchedBorder(new
java.awt.Color(255, 153, 0), null));
```

```
        getContentPane().add(jLabel1);
        jLabel1.setBounds(0, 0, 140, 80);

        jLabel2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/chat1.JPG"))); // NOI18N
        jLabel2.setBorder(javax.swing.BorderFactory.createEtchedBorder(new
java.awt.Color(255, 204, 51), null));
        getContentPane().add(jLabel2);
        jLabel2.setBounds(0, 80, 260, 208);

        jPanel1.setBackground(new java.awt.Color(0, 204, 141));
        jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder(new
java.awt.Color(255, 204, 51), null));
        jPanel1.setLayout(null);

        jSeparator2.setMinimumSize(new java.awt.Dimension(135, 2));
        jPanel1.add(jSeparator2);
        jSeparator2.setBounds(0, 135, 128, 2);

        jButton1.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        jButton1.setText("LOGIN FORM");
        jButton1.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.LOWERED));
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton1);
        jButton1.setBounds(20, 30, 90, 30);

        jButton2.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        jButton2.setText("REGISTER FORM");

jButton2.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.B
evelBorder.RAISED));
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton2);
        jButton2.setBounds(17, 80, 100, 30);

        jButton3.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        jButton3.setText("EXIT");
```

```
jButton3.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.B
evelBorder.RAISED));
        jPanel1.add(jButton3);
        jButton3.setBounds(30, 150, 70, 40);

        getContentPane().add(jPanel1);
        jPanel1.setBounds(260, 80, 130, 208);

        jMenu1.setText("File");

        jMenuItem1.setText("Login");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem1);

        jMenuItem2.setText("Register");
        jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem2);
        jMenu1.add(jSeparator1);

        jMenuItem3.setFont(new java.awt.Font("Segoe UI", 0, 14));
        jMenuItem3.setForeground(new java.awt.Color(255, 0, 51));
        jMenuItem3.setText("Exit");
        jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem3);

        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);

        pack();
    }// </editor-fold>//GEN-END:initComponents
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        Login l = new Login();
        l.setVisible(true);
        this.setVisible(false);


        if (evt.getSource() == jButton1 && l.isVisible()) {
          this.jButton1.setEnabled(false);

        }
    }//GEN-LAST:event_jButton1ActionPerformed

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton2ActionPerformed
        // TODO add your handling code here:
        Register r = new Register(this);
        r.setVisible(true);
        this.setVisible(false);

        if (evt.getSource() == jButton2 && r.isVisible()) {
          this.jButton2.setEnabled(false);

        }
    }//GEN-LAST:event_jButton2ActionPerformed

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jMenuItem1ActionPerformed
        // TODO add your handling code here:
        Login l = new Login();
        l.setVisible(true);
        if (evt.getSource() == jMenuItem1 && l.isVisible()) {
          this.jButton1.setEnabled(false);
          JOptionPane.showMessageDialog(this, "LOGIN FORM HAS BEEN ALREADY OPEN");
        }
    }//GEN-LAST:event_jMenuItem1ActionPerformed

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jMenuItem2ActionPerformed
        // TODO add your handling code here:
        Register r = new Register(this);
        r.setVisible(true);
        if (evt.getSource() == jMenuItem2 && r.isVisible()) {
            this.jButton1.setEnabled(false);
            JOptionPane.showMessageDialog(this,    "LOGIN    FORM    HAS    BEEN
ALREADY OPEN");
        }
    }//GEN-LAST:event_jMenuItem2ActionPerformed

    private  void  jMenuItem3ActionPerformed(java.awt.event.ActionEvent  evt)  {//GEN-
FIRST:event_jMenuItem3ActionPerformed
        // TODO add your handling code here:
        System.exit(0);
    }//GEN-LAST:event_jMenuItem3ActionPerformed

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new Main().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;
    // End of variables declaration//GEN-END:variables
}
```

Register.java

```
package ui;

import communication.Config;
import java.awt.Color;
import java.text.SimpleDateFormat;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class Register extends javax.swing.JDialog {

  /** Creates new form Register */
  public Register(JFrame parent) {
        super(parent, true);
        initComponents();
        LErr3.setVisible(false);
  }

     @SuppressWarnings("unchecked")
  //    <editor-fold    defaultstate="collapsed"    desc="Generated    Code">//GEN-
BEGIN:initComponents
  private void initComponents() {

     Pic1 = new javax.swing.JLabel();
     Pic2 = new javax.swing.JLabel();
     MLabel = new javax.swing.JLabel();
     Date = new javax.swing.JComboBox();
     Month = new javax.swing.JComboBox();
     Year = new javax.swing.JComboBox();
     Select = new javax.swing.JComboBox();
     Stay = new javax.swing.JCheckBox();
     LFname = new javax.swing.JLabel();
     LLname = new javax.swing.JLabel();
     LGender = new javax.swing.JLabel();
     LDOB = new javax.swing.JLabel();
     LLogin = new javax.swing.JLabel();
     LPassword = new javax.swing.JLabel();
     LRPassword = new javax.swing.JLabel();
     LJMail = new javax.swing.JLabel();
     LMin = new javax.swing.JLabel();
     LLocation = new javax.swing.JLabel();
     TFname = new javax.swing.JTextField();
     TLname = new javax.swing.JTextField();
     TLogin = new javax.swing.JTextField();
     TLocation = new javax.swing.JTextField();
```

```
    MPassword = new javax.swing.JPasswordField();
    SPassword = new javax.swing.JPasswordField();
    Create = new javax.swing.JButton();
    Cancel = new javax.swing.JButton();
    LErr1 = new javax.swing.JLabel();
    LErr2 = new javax.swing.JLabel();
    Sep = new javax.swing.JSeparator();
    LErr3 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setTitle("Registeration Form");
    setMinimumSize(new java.awt.Dimension(500, 570));
    getContentPane().setLayout(null);

    Pic1.setIcon(new      javax.swing.ImageIcon(getClass().getResource("/images/sar
talk.jpg"))); // NOI18N
    getContentPane().add(Pic1);
    Pic1.setBounds(0, 0, 150, 80);

    Pic2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/label.jpg"))); // NOI18N
    getContentPane().add(Pic2);
    Pic2.setBounds(160, 20, 300, 40);

    MLabel.setFont(new java.awt.Font("Tahoma", 0, 18));
    MLabel.setForeground(new java.awt.Color(255, 51, 51));
    MLabel.setText("Get Started With JMail");
    getContentPane().add(MLabel);
    MLabel.setBounds(30, 90, 200, 25);

    Date.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "--DATE-
-", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17",
"18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31" }));
    getContentPane().add(Date);
    Date.setBounds(180, 235, 80, 25);

    Month.setModel(new   javax.swing.DefaultComboBoxModel(new   String[]   {   "--
MONTH--", "JANUARY", "FEBRUARY", "MARCH", "APRIL", "MAY", "JUNE",
"JULY", "AUGUST", "SEPTEMBER", "OCTOBER", "NOVEMBER", "DECEMBER"
}));
    getContentPane().add(Month);
    Month.setBounds(270, 235, 100, 25);

    Year.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "--YEAR-
-", "1980", "1981", "1982", "1983", "1984", "1985", "1986", "1987", "1988", "1989",
"1990", "1991", "1992", "1993", "1994", "1995", "1996", "1997", "1998", "1999",
"2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "2009",
"2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019",
"2020", "2021", "2022", "2023", "2024", "2025", "2026", "2027", "2028", "2029",
"2030", "2031", "2032", "2033", "2034", "2035", "2036", "2037", "2038", "2039",
"2040", "2041", "2042", "2043", "2044", "2045", "2046", "2047", "2048", "2049", "2050"
}));
    Year.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        YearActionPerformed(evt);
      }
    });
    getContentPane().add(Year);
    Year.setBounds(380, 235, 80, 25);

    Select.setModel(new  javax.swing.DefaultComboBoxModel(new  String[]  {  "--
SELECT--", "MALE", "FEMALE" }));
    Select.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        SelectActionPerformed(evt);
      }
    });
    getContentPane().add(Select);
    Select.setBounds(180, 200, 180, 25);

    Stay.setFont(new java.awt.Font("Tahoma", 0, 12));
    Stay.setText("Stay Sign In:-");
    getContentPane().add(Stay);
    Stay.setBounds(180, 400, 180, 25);

    LFname.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LFname.setForeground(new java.awt.Color(0, 0, 255));
    LFname.setText("First name:-");
    getContentPane().add(LFname);
    LFname.setBounds(30, 130, 68, 25);

    LLname.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LLname.setForeground(new java.awt.Color(0, 0, 255));
    LLname.setText("Last name:-");
    getContentPane().add(LLname);
    LLname.setBounds(30, 165, 66, 25);

    LGender.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LGender.setForeground(new java.awt.Color(0, 0, 255));
    LGender.setText("Gender:-");
    getContentPane().add(LGender);
    LGender.setBounds(30, 205, 70, 30);
    LDOB.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LDOB.setForeground(new java.awt.Color(0, 0, 255));
    LDOB.setText("Date of Birth");
    getContentPane().add(LDOB);
    LDOB.setBounds(30, 240, 73, 25);

    LLogin.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LLogin.setForeground(new java.awt.Color(0, 0, 255));
    LLogin.setText("Desired Login Name:-");
    getContentPane().add(LLogin);
    LLogin.setBounds(30, 265, 124, 40);

    LPassword.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LPassword.setForeground(new java.awt.Color(0, 0, 255));
    LPassword.setText("Password:- ");
    getContentPane().add(LPassword);
    LPassword.setBounds(30, 310, 68, 40);

    LRPassword.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LRPassword.setForeground(new java.awt.Color(0, 0, 255));
    LRPassword.setText("Re-enter Password:-");
    getContentPane().add(LRPassword);
    LRPassword.setBounds(30, 370, 116, 30);

    LJMail.setFont(new java.awt.Font("Tahoma", 0, 12));
    LJMail.setForeground(new java.awt.Color(0, 0, 255));
    LJMail.setText("@Jmail.com");
    getContentPane().add(LJMail);
    LJMail.setBounds(370, 270, 63, 30);

    LMin.setText("(Minimum 8 characters in length)");
    getContentPane().add(LMin);
    LMin.setBounds(180, 345, 190, 20);

    LLocation.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LLocation.setForeground(new java.awt.Color(0, 0, 255));
    LLocation.setText("Location:-");
    getContentPane().add(LLocation);
    LLocation.setBounds(30, 430, 57, 40);

TFname.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border.E
tchedBorder.RAISED, null, new java.awt.Color(102, 153, 255)));
    getContentPane().add(TFname);
    TFname.setBounds(180, 125, 180, 30);
```

```
TLname.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border.E
tchedBorder.RAISED, null, new java.awt.Color(102, 153, 255)));
      getContentPane().add(TLname);
      TLname.setBounds(180, 160, 180, 30);

TLogin.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border.Et
chedBorder.RAISED, null, new java.awt.Color(102, 153, 255)));
      getContentPane().add(TLogin);
      TLogin.setBounds(180, 270, 180, 30);

TLocation.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border.
EtchedBorder.RAISED, null, new java.awt.Color(102, 153, 255)));
      getContentPane().add(TLocation);
      TLocation.setBounds(180, 430, 180, 30);

MPassword.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.borde
r.EtchedBorder.RAISED, null, new java.awt.Color(102, 153, 255)));
      MPassword.addKeyListener(new java.awt.event.KeyAdapter() {
         public void keyReleased(java.awt.event.KeyEvent evt) {
            MPasswordKeyReleased(evt);
         }
      });
      getContentPane().add(MPassword);
      MPassword.setBounds(180, 315, 180, 30);

SPassword.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border
.EtchedBorder.RAISED, null, new java.awt.Color(102, 153, 255)));
      SPassword.addKeyListener(new java.awt.event.KeyAdapter() {
         public void keyReleased(java.awt.event.KeyEvent evt) {
            SPasswordKeyReleased(evt);
         }
      });
      getContentPane().add(SPassword);
      SPassword.setBounds(180, 370, 180, 30);

      Create.setFont(new java.awt.Font("Tahoma", 0, 14));
      Create.setForeground(new java.awt.Color(0, 0, 255));
      Create.setText("Create My Account");
      Create.addMouseListener(new java.awt.event.MouseAdapter() {
         public void mouseClicked(java.awt.event.MouseEvent evt) {
            CreateMouseClicked(evt);
        }
        public void mouseEntered(java.awt.event.MouseEvent evt) {
            CreateMouseEntered(evt);
        }
        public void mouseExited(java.awt.event.MouseEvent evt) {
            CreateMouseExited(evt);
        }
    });
    Create.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            CreateActionPerformed(evt);
        }
    });
    getContentPane().add(Create);
    Create.setBounds(60, 480, 210, 40);

    Cancel.setFont(new java.awt.Font("Tahoma", 0, 12));
    Cancel.setForeground(new java.awt.Color(0, 0, 255));
    Cancel.setText("CANCEL");
    Cancel.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            CancelMouseClicked(evt);
        }
        public void mouseEntered(java.awt.event.MouseEvent evt) {
            CancelMouseEntered(evt);
        }
        public void mouseExited(java.awt.event.MouseEvent evt) {
            CancelMouseExited(evt);
        }
    });
    Cancel.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            CancelActionPerformed(evt);
        }
    });
    getContentPane().add(Cancel);
    Cancel.setBounds(290, 480, 80, 40);

    LErr1.setFont(new java.awt.Font("Times New Roman", 0, 14));
    LErr1.setForeground(new java.awt.Color(255, 0, 0));
    LErr1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/alert.gif"))); // NOI18N
    LErr1.setText("Error");
    getContentPane().add(LErr1);
    LErr1.setBounds(380, 320, 70, 25);
```

```
        LErr2.setFont(new java.awt.Font("Times New Roman", 0, 14));
        LErr2.setForeground(new java.awt.Color(255, 0, 0));
        LErr2.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/alert.gif"))); // NOI18N
        LErr2.setText("Error");
        LErr2.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                LErr2KeyReleased(evt);
            }
        });
        getContentPane().add(LErr2);
        LErr2.setBounds(380, 370, 70, 25);
        getContentPane().add(Sep);
        Sep.setBounds(0, 80, 480, 2);

        LErr3.setFont(new java.awt.Font("Times New Roman", 0, 14));
        LErr3.setForeground(new java.awt.Color(255, 0, 0));
        LErr3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/alert.gif"))); // NOI18N
        LErr3.setText("Error");
        getContentPane().add(LErr3);
        LErr3.setBounds(380, 480, 60, 30);

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void SelectActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_SelectActionPerformed
}//GEN-LAST:event_SelectActionPerformed

    private void YearActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_YearActionPerformed
    }//GEN-LAST:event_YearActionPerformed

    private void CancelMouseEntered(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_CancelMouseEntered
        this.Cancel.setForeground(Color.red);
    }//GEN-LAST:event_CancelMouseEntered

    private void MPasswordKeyReleased(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_MPasswordKeyReleased
        String s1 = new String();
        s1 = this.MPassword.getText();
        if (s1.length() >= 8) {
            this.LErr1.setVisible(false);
        } else {
            this.LErr1.setVisible(true);
        }
    }//GEN-LAST:event_MPasswordKeyReleased

    private    void    LErr2KeyReleased(java.awt.event.KeyEvent    evt)    {//GEN-
FIRST:event_LErr2KeyReleased
    }//GEN-LAST:event_LErr2KeyReleased

    private    void    SPasswordKeyReleased(java.awt.event.KeyEvent    evt)    {//GEN-
FIRST:event_SPasswordKeyReleased
        String s1 = new String();
        String s2 = new String();
        s2 = this.SPassword.getText();
        s1 = this.MPassword.getText();
        if (s2.length() >= 8 & s1.equals(s2)) {
            this.LErr2.setVisible(false);
        } else {
            this.LErr2.setVisible(true);
        }
    }//GEN-LAST:event_SPasswordKeyReleased

    private    void    CancelActionPerformed(java.awt.event.ActionEvent    evt)    {//GEN-
FIRST:event_CancelActionPerformed
        this.setVisible(false);
        Main m = new Main();
        m.setVisible(true);
    }//GEN-LAST:event_CancelActionPerformed

    private    void    CreateActionPerformed(java.awt.event.ActionEvent    evt)    {//GEN-
FIRST:event_CreateActionPerformed
        if (this.LErr1.isVisible() || this.LErr2.isVisible()) {
            this.LErr3.setVisible(true);
        } else {
            this.LErr3.setVisible(false);
            try {
                SimpleDateFormat df = new SimpleDateFormat("ddMMMMyyyy");
                String username = this.TLogin.getText();
                String password = this.MPassword.getText();
                String name = this.TFname.getText() + " " + this.TLname.getText();
                String gender = (String) this.Select.getSelectedItem();
                String  date_of_birth  =  (String)  this.Date.getSelectedItem()  +  (String)
this.Month.getSelectedItem() + (String) this.Year.getSelectedItem();
                java.util.Date date = df.parse(date_of_birth);
                String location = this.TLocation.getText();
                String s = Config.current.relay.registerUser(username, password, name,
gender, date, location);
                if (s.equals("")) {
                    JOptionPane.showMessageDialog(this, "NOW YOU REGISTERED");
                    this.setVisible(false);
                } else {
                    JOptionPane.showMessageDialog(this, s);
                }
            } catch (Exception e) {
                JOptionPane.showMessageDialog(this, e);
            }
        }
    }//GEN-LAST:event_CreateActionPerformed

    private void CreateMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_CreateMouseClicked
    }//GEN-LAST:event_CreateMouseClicked

    private void CreateMouseEntered(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_CreateMouseEntered
    }//GEN-LAST:event_CreateMouseEntered

    private void CreateMouseExited(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_CreateMouseExited
    }//GEN-LAST:event_CreateMouseExited

    private void CancelMouseExited(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_CancelMouseExited
        this.Cancel.setForeground(Color.blue);
    }//GEN-LAST:event_CancelMouseExited

    private void CancelMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_CancelMouseClicked
        Login obj = new Login();
        this.setVisible(false);
        obj.setVisible(true);

    }//GEN-LAST:event_CancelMouseClicked

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new Register(null).setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton Cancel;
    private javax.swing.JButton Create;
    private javax.swing.JComboBox Date;
    private javax.swing.JLabel LDOB;
    private javax.swing.JLabel LErr1;
    private javax.swing.JLabel LErr2;
    private javax.swing.JLabel LErr3;
    private javax.swing.JLabel LFname;
    private javax.swing.JLabel LGender;
    private javax.swing.JLabel LJMail;
    private javax.swing.JLabel LLname;
    private javax.swing.JLabel LLocation;
    private javax.swing.JLabel LLogin;
    private javax.swing.JLabel LMin;
    private javax.swing.JLabel LPassword;
    private javax.swing.JLabel LRPassword;
    private javax.swing.JLabel MLabel;
    private javax.swing.JPasswordField MPassword;
    private javax.swing.JComboBox Month;
    private javax.swing.JLabel Pic1;
    private javax.swing.JLabel Pic2;
    private javax.swing.JPasswordField SPassword;
    private javax.swing.JComboBox Select;
    private javax.swing.JSeparator Sep;
    private javax.swing.JCheckBox Stay;
    private javax.swing.JTextField TFname;
    private javax.swing.JTextField TLname;
    private javax.swing.JTextField TLocation;
    private javax.swing.JTextField TLogin;
    private javax.swing.JComboBox Year;
    // End of variables declaration//GEN-END:variables
}
```

Login.java

```
package ui;

import communication.*;
import java.awt.Color;
import java.net.Socket;
import java.security.KeyPair;
import javax.swing.JOptionPane;
import javax.swing.JColorChooser;
public class Login extends javax.swing.JFrame {

  /** Creates new form Login */
  public Login() {
       initComponents();
       this.txtPort.setValue(1060);
       this.jButton1.setEnabled(false);
  }


  @SuppressWarnings("unchecked")
  //    <editor-fold defaultstate="collapsed"  desc="Generated  Code">//GEN-
BEGIN:initComponents
  private void initComponents() {

    jLabel6 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();
    username = new javax.swing.JLabel();
    password = new javax.swing.JLabel();
    forgot = new javax.swing.JLabel();
    picture = new javax.swing.JLabel();
    create = new javax.swing.JLabel();
    settings = new javax.swing.JLabel();
    help = new javax.swing.JLabel();
    jusername = new javax.swing.JTextField();
    jPassword = new javax.swing.JPasswordField();
    jCheckBox1 = new javax.swing.JCheckBox();
    jButton1 = new javax.swing.JButton();
    username1 = new javax.swing.JLabel();
    username2 = new javax.swing.JLabel();
    txtIP = new javax.swing.JTextField();
    txtPort = new javax.swing.JSpinner();
    jButton2 = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();

    jLabel6.setText("jLabel6");
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("sar chat");
    setBackground(new java.awt.Color(255, 255, 255));
    setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    setForeground(new java.awt.Color(51, 0, 255));
    addPropertyChangeListener(new java.beans.PropertyChangeListener() {
      public void propertyChange(java.beans.PropertyChangeEvent evt) {
        formPropertyChange(evt);
      }
    });
    getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
    getContentPane().add(jSeparator1,                                          new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 102, 270, -1));

    username.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
    username.setText("Server Port:");
    getContentPane().add(username,                                             new
org.netbeans.lib.awtextra.AbsoluteConstraints(250, 210, 90, 30));

    password.setFont(new java.awt.Font("Tahoma", 0, 12));
    password.setText("Password:");
    getContentPane().add(password,                                             new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 214, 90, 20));

    forgot.setFont(new java.awt.Font("Tahoma", 0, 12));
    forgot.setForeground(new java.awt.Color(0, 102, 204));
    forgot.setText("Forgot Your Password?");
    forgot.addMouseListener(new java.awt.event.MouseAdapter() {
      public void mouseEntered(java.awt.event.MouseEvent evt) {
        forgotMouseEntered(evt);
      }
      public void mouseExited(java.awt.event.MouseEvent evt) {
        forgotMouseExited(evt);
      }
    });
    getContentPane().add(forgot,                                               new
org.netbeans.lib.awtextra.AbsoluteConstraints(60, 370, 130, 20));

    picture.setIcon(new     javax.swing.ImageIcon(getClass().getResource("/images/sar
talk.jpg"))); // NOI18N
    getContentPane().add(picture,                                              new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 20, -1, 70));

    create.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
    create.setForeground(new java.awt.Color(0, 102, 204));
    create.setText("Dont have an account?");
create.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      createMouseClicked(evt);
   }
   public void mouseEntered(java.awt.event.MouseEvent evt) {
      createMouseEntered(evt);
   }
   public void mouseExited(java.awt.event.MouseEvent evt) {
      createMouseExited(evt);
   }
});
getContentPane().add(create,                                          new
org.netbeans.lib.awtextra.AbsoluteConstraints(60, 400, 130, 20));

settings.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
settings.setText("Settings");
settings.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      settingsMouseClicked(evt);
   }
   public void mouseEntered(java.awt.event.MouseEvent evt) {
      settingsMouseEntered(evt);
   }
   public void mouseExited(java.awt.event.MouseEvent evt) {
      settingsMouseExited(evt);
   }
});
getContentPane().add(settings,                                        new
org.netbeans.lib.awtextra.AbsoluteConstraints(170, 10, 50, 20));

help.setFont(new java.awt.Font("Tahoma", 0, 12));
help.setText("| Help");
help.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      helpMouseClicked(evt);
   }
   public void mouseEntered(java.awt.event.MouseEvent evt) {
      helpMouseEntered(evt);
   }
   public void mouseExited(java.awt.event.MouseEvent evt) {
      helpMouseExited(evt);
   }
});
getContentPane().add(help, new org.netbeans.lib.awtextra.AbsoluteConstraints(220,
10, 40, 20));
```

```
jusername.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border
.EtchedBorder.RAISED, null, new java.awt.Color(153, 204, 255)));
    jusername.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyReleased(java.awt.event.KeyEvent evt) {
        jusernameKeyReleased(evt);
      }
    });
    getContentPane().add(jusername,                                          new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 180, 190, 30));


jPassword.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border
.EtchedBorder.RAISED, null, new java.awt.Color(153, 204, 255)));
    jPassword.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyPressed(java.awt.event.KeyEvent evt) {
        jPasswordKeyPressed(evt);
      }
      public void keyReleased(java.awt.event.KeyEvent evt) {
        jPasswordKeyReleased(evt);
      }
    });
    getContentPane().add(jPassword,                                          new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 250, 190, 30));

    jCheckBox1.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
    jCheckBox1.setLabel("Remember Password");
    getContentPane().add(jCheckBox1,                                         new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 290, 150, -1));

    jButton1.setForeground(new java.awt.Color(0, 51, 255));
    jButton1.setText("Sign in");
    jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
      public void mouseClicked(java.awt.event.MouseEvent evt) {
        jButton1MouseClicked(evt);
      }
    });
    jButton1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
      }
    });
    getContentPane().add(jButton1,                                           new
org.netbeans.lib.awtextra.AbsoluteConstraints(60, 330, 130, 30));

    username1.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
```

```
        username1.setText("Username:");
        getContentPane().add(username1,                                            new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 140, 90, 30));

        username2.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        username2.setText("Server IP:");
        getContentPane().add(username2,                                            new
org.netbeans.lib.awtextra.AbsoluteConstraints(250, 140, 90, 30));

        txtIP.setText("127.0.0.1");

txtIP.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border.Etch
edBorder.RAISED, null, new java.awt.Color(153, 204, 255)));
        txtIP.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                txtIPKeyReleased(evt);
            }
        });
        getContentPane().add(txtIP,                                                new
org.netbeans.lib.awtextra.AbsoluteConstraints(250, 180, 190, 30));
        getContentPane().add(txtPort,                                              new
org.netbeans.lib.awtextra.AbsoluteConstraints(250, 250, 190, 30));

        jButton2.setForeground(new java.awt.Color(0, 51, 255));
        jButton2.setText("Set");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton2,                                             new
org.netbeans.lib.awtextra.AbsoluteConstraints(250, 330, 130, 30));
        getContentPane().add(jLabel1,                                              new
org.netbeans.lib.awtextra.AbsoluteConstraints(440, 400, 20, 30));

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void formPropertyChange(java.beans.PropertyChangeEvent evt) {//GEN-
FIRST:event_formPropertyChange
        // TODO add your handling code here:
    }//GEN-LAST:event_formPropertyChange

    private void createMouseEntered(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_createMouseEntered
        this.create.setForeground(Color.red);
    }//GEN-LAST:event_createMouseEntered

    private void forgotMouseExited(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_forgotMouseExited
        this.forgot.setForeground(Color.black);         // TODO add your handling code here:
    }//GEN-LAST:event_forgotMouseExited

    private void forgotMouseEntered(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_forgotMouseEntered
        this.forgot.setForeground(Color.RED);         // TODO add your handling code here:
    }//GEN-LAST:event_forgotMouseEntered

    private void createMouseExited(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_createMouseExited
        this.create.setForeground(Color.black);         // TODO add your handling code here:
    }//GEN-LAST:event_createMouseExited

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
        try {
            String msg = Config.current.relay.doLogin(this.jusername.getText(), this.jPassword.getText());
            if (msg.equals("")) {
                Config.current.username = this.jusername.getText();

                SelectFriend f = new SelectFriend();
                f.setVisible(true);
                this.setVisible(false);
            } else {
                JOptionPane.showMessageDialog(this, msg);
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(rootPane, ex);
        }
    }//GEN-LAST:event_jButton1ActionPerformed

    private void createMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_createMouseClicked
        Register f = new Register(this);
        this.setVisible(false);
        f.setVisible(true);

    }//GEN-LAST:event_createMouseClicked
```

```
  private    void    jPasswordKeyPressed(java.awt.event.KeyEvent     evt)     {//GEN-
FIRST:event_jPasswordKeyPressed
   }//GEN-LAST:event_jPasswordKeyPressed

  private    void    jPasswordKeyReleased(java.awt.event.KeyEvent     evt)     {//GEN-
FIRST:event_jPasswordKeyReleased
        String n2 = new String();
        String n = new String();
        n = jusername.getText();
        n2 = jPassword.getText();
        if (n2.length() > 0 && n.length() > 0) {
           this.jButton1.setEnabled(true);
        } else {
           this.jButton1.setEnabled(false);
        }
   }//GEN-LAST:event_jPasswordKeyReleased

  private    void    jButton1MouseClicked(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_jButton1MouseClicked
        this.setVisible(false);
   }//GEN-LAST:event_jButton1MouseClicked

  private    void    settingsMouseClicked(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_settingsMouseClicked
        Color  c = JColorChooser.showDialog(this, "Change Text Background Color",
Color.yellow);
        this.jusername.setBackground(c);
   }//GEN-LAST:event_settingsMouseClicked

  private    void    helpMouseClicked(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_helpMouseClicked
        JOptionPane.showMessageDialog(null, "This is a simple Chat application built
using Java.",
              "About Help",
              JOptionPane.INFORMATION_MESSAGE);
   }//GEN-LAST:event_helpMouseClicked

  private    void    settingsMouseEntered(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_settingsMouseEntered
        this.setForeground(Color.red);
   }//GEN-LAST:event_settingsMouseEntered

  private    void    settingsMouseExited(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_settingsMouseExited
        // TODO add your handling code here:
```

```
        this.setForeground(Color.blue);
    }//GEN-LAST:event_settingsMouseExited

    private    void    helpMouseEntered(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_helpMouseEntered
        // TODO add your handling code here:
        this.setForeground(Color.red);
    }//GEN-LAST:event_helpMouseEntered

    private    void    helpMouseExited(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_helpMouseExited
        // TODO add your handling code here:
        this.setForeground(Color.blue);
    }//GEN-LAST:event_helpMouseExited

    private    void    jusernameKeyReleased(java.awt.event.KeyEvent    evt)    {//GEN-
FIRST:event_jusernameKeyReleased
        // TODO add your handling code here:
    }//GEN-LAST:event_jusernameKeyReleased

    private    void    txtIPKeyReleased(java.awt.event.KeyEvent    evt)    {//GEN-
FIRST:event_txtIPKeyReleased
        // TODO add your handling code here:
    }//GEN-LAST:event_txtIPKeyReleased

    private  void  jButton2ActionPerformed(java.awt.event.ActionEvent  evt)  {//GEN-
FIRST:event_jButton2ActionPerformed
        try {
            Socket s = new Socket(this.txtIP.getText(), (Integer) this.txtPort.getValue());
            Config.current = new ConnectionInfo();
            Config.current.relay = new ClientRelay(s);

            KeyPair k = RSAEncryptUtil.generateKey();
            Config.priKey = k.getPrivate();
            Config.current.pubKey = k.getPublic();
            Config.current.relay.sendKey(Config.current.pubKey.getEncoded());

            this.jButton2.setEnabled(false);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex);
        }
    }//GEN-LAST:event_jButton2ActionPerformed

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Login().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel create;
    private javax.swing.JLabel forgot;
    private javax.swing.JLabel help;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JCheckBox jCheckBox1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPasswordField jPassword;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JTextField jusername;
    private javax.swing.JLabel password;
    private javax.swing.JLabel picture;
    private javax.swing.JLabel settings;
    private javax.swing.JTextField txtIP;
    private javax.swing.JSpinner txtPort;
    private javax.swing.JLabel username;
    private javax.swing.JLabel username1;
    private javax.swing.JLabel username2;
    // End of variables declaration//GEN-END:variables
}
```

Chat.java

```
package ui;

import communication.Config;
import ui.Login;
import java.awt.*;
import java.net.*;
import javax.swing.JColorChooser;
import javax.swing.JDialog;
import javax.swing.JOptionPane;

public class Chat extends javax.swing.JDialog {

    public String withUser;

    /** Creates new form Chat */
    public Chat(JDialog parent, String withUser) {
        super(parent, true);
        this.withUser = withUser;

        initComponents();

        this.jLabel4.setVisible(false);

        this.jLabel8.setVisible(false);

        this.lblYourName.setText("Your name:- " + Config.current.username);
        this.lblFriendName.setText("Friend's name:- " + withUser);
    }

    @SuppressWarnings("unchecked")
    //    <editor-fold    defaultstate="collapsed"    desc="Generated    Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jComboBox1 = new javax.swing.JComboBox();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        txtChat = new javax.swing.JTextArea();
        jScrollPane2 = new javax.swing.JScrollPane();
        txtMessage = new javax.swing.JTextPane();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jButton3 = new javax.swing.JButton();
        jSeparator1 = new javax.swing.JSeparator();
        jLabel4 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        lblYourName = new javax.swing.JLabel();
        lblFriendName = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        addWindowListener(new java.awt.event.WindowAdapter() {
          public void windowClosing(java.awt.event.WindowEvent evt) {
            formWindowClosing(evt);
          }
        });
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jLabel1.setIcon(new     javax.swing.ImageIcon(getClass().getResource("/images/sar
talk.jpg"))); // NOI18N
        getContentPane().add(jLabel1,                                          new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 140, 70));

        jLabel2.setFont(new java.awt.Font("Tahoma", 0, 12));
        jLabel2.setForeground(new java.awt.Color(0, 0, 255));
        jLabel2.setText("Settings");
        jLabel2.addMouseListener(new java.awt.event.MouseAdapter() {
          public void mouseClicked(java.awt.event.MouseEvent evt) {
            jLabel2MouseClicked(evt);
          }
          public void mouseEntered(java.awt.event.MouseEvent evt) {
            jLabel2MouseEntered(evt);
          }
          public void mouseExited(java.awt.event.MouseEvent evt) {
            jLabel2MouseExited(evt);
          }
        });
        getContentPane().add(jLabel2,                                          new
org.netbeans.lib.awtextra.AbsoluteConstraints(260, 20, 50, 30));

        jComboBox1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jComboBox1.setForeground(new java.awt.Color(0, 51, 255));
        jComboBox1.setModel(new  javax.swing.DefaultComboBoxModel(new  String[]  {
"Set your status message here", "Available", "Busy", "Logout" }));
        jComboBox1.addMouseListener(new java.awt.event.MouseAdapter() {
          public void mouseEntered(java.awt.event.MouseEvent evt) {
            jComboBox1MouseEntered(evt);
          }
```

```
            public void mouseExited(java.awt.event.MouseEvent evt) {
                jComboBox1MouseExited(evt);
            }
        });
        jComboBox1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jComboBox1ActionPerformed(evt);
            }
        });
        getContentPane().add(jComboBox1,                                         new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 130, 210, 30));

        jButton1.setFont(new java.awt.Font("Tahoma", 0, 14));
        jButton1.setForeground(new java.awt.Color(0, 0, 255));
        jButton1.setText("Send");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton1,                                           new
org.netbeans.lib.awtextra.AbsoluteConstraints(70, 400, 90, 30));

        jButton2.setFont(new java.awt.Font("Tahoma", 0, 14));
        jButton2.setForeground(new java.awt.Color(255, 0, 51));
        jButton2.setText("Delete");
        jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButton2MouseClicked(evt);
            }
        });
        getContentPane().add(jButton2,                                           new
org.netbeans.lib.awtextra.AbsoluteConstraints(190, 180, 90, 30));

        txtChat.setColumns(20);
        txtChat.setEditable(false);
        txtChat.setLineWrap(true);
        txtChat.setRows(5);
        jScrollPane1.setViewportView(txtChat);

        getContentPane().add(jScrollPane1,                                       new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 220, 270, 80));

        jScrollPane2.setViewportView(txtMessage);
```

```
    getContentPane().add(jScrollPane2,                                    new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 340, 270, 50));

    jLabel5.setFont(new java.awt.Font("Tahoma", 0, 18));
    jLabel5.setForeground(new java.awt.Color(255, 51, 51));
    jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    jLabel5.setText("Message:-");
    getContentPane().add(jLabel5,                                         new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 310, 90, 20));

    jLabel6.setFont(new java.awt.Font("Tahoma", 0, 12));
    jLabel6.setForeground(new java.awt.Color(0, 51, 255));
    jLabel6.setText("| Help");
    jLabel6.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jLabel6MouseClicked(evt);
        }
        public void mouseEntered(java.awt.event.MouseEvent evt) {
            jLabel6MouseEntered(evt);
        }
        public void mouseExited(java.awt.event.MouseEvent evt) {
            jLabel6MouseExited(evt);
        }
    });
    getContentPane().add(jLabel6,                                         new
org.netbeans.lib.awtextra.AbsoluteConstraints(310, 24, 40, 20));

    jButton3.setFont(new java.awt.Font("Tahoma", 0, 14));
    jButton3.setForeground(new java.awt.Color(204, 0, 255));
    jButton3.setText("Scraps");
    jButton3.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton3MouseClicked(evt);
        }
    });
    getContentPane().add(jButton3,                                        new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 180, 90, 30));
    getContentPane().add(jSeparator1,                                     new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 80, 350, -1));

    jLabel4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/green.jpg"))); // NOI18N
    getContentPane().add(jLabel4,                                         new
org.netbeans.lib.awtextra.AbsoluteConstraints(30, 90, -1, -1));

    jLabel8.setFont(new java.awt.Font("Tahoma", 0, 12));
```

```
        jLabel8.setForeground(new java.awt.Color(204, 0, 255));
        jLabel8.setText("Available");
        getContentPane().add(jLabel8,                                            new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 80, 60, 30));
        getContentPane().add(lblYourName,                                        new
org.netbeans.lib.awtextra.AbsoluteConstraints(110, 90, -1, -1));
        getContentPane().add(lblFriendName,                                      new
org.netbeans.lib.awtextra.AbsoluteConstraints(110, 110, -1, -1));

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jComboBox1MouseEntered(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jComboBox1MouseEntered
        this.jComboBox1.setBackground(Color.WHITE);
    }//GEN-LAST:event_jComboBox1MouseEntered

    private void jButton3MouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jButton3MouseClicked
        //this.jscrap.setVisible(true);
    }//GEN-LAST:event_jButton3MouseClicked

    private void jComboBox1MouseExited(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jComboBox1MouseExited
        this.jComboBox1.setBackground(Color.lightGray);
    }//GEN-LAST:event_jComboBox1MouseExited

    private void jLabel2MouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jLabel2MouseClicked
    Color c=JColorChooser.showDialog(this,"SET FOREGROUND COLOR",
Color.yellow);
      this.txtChat.setForeground(c);
    }//GEN-LAST:event_jLabel2MouseClicked

    private void jLabel2MouseEntered(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jLabel2MouseEntered
        this.jLabel2.setForeground(Color.red);
    }//GEN-LAST:event_jLabel2MouseEntered

    private void jLabel2MouseExited(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jLabel2MouseExited
        this.jLabel2.setForeground(Color.blue);
    }//GEN-LAST:event_jLabel2MouseExited

    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jComboBox1ActionPerformed
```

```
        switch (this.jComboBox1.getSelectedIndex()) {
            case 1:
                this.jLabel4.setVisible(true);
                this.jLabel8.setVisible(true);
            jLabel4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/green.jpg")));
                this.jLabel8.setText("AVAILABLE");
            break;
            case 2:
                this.jLabel4.setVisible(true);
                this.jLabel8.setVisible(true);
            jLabel4.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/red.jpg")));
                this.jLabel8.setText("BUSY");
                break;
            case 3:
                this.setVisible(false);
                Login f= new Login();
                f.setVisible(true);
                break;
            default:
            this.jLabel8.setVisible(false);
            this.jLabel4.setText("");
        }
    }//GEN-LAST:event_jComboBox1ActionPerformed

    private void jButton2MouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jButton2MouseClicked
        //this.jscrap.setText(null);
    }//GEN-LAST:event_jButton2MouseClicked

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton1ActionPerformed
        try {
            Config.current.relay.sendChatMessage(this.withUser,
this.txtMessage.getText());
            this.txtChat.setText(this.txtChat.getText() + Config.current.username + ":- " +
this.txtMessage.getText() + "\n");
            this.txtMessage.setText("");
            this.txtChat.setSelectionStart(this.txtChat.getText().length() - 1);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, ex);
        }
    }//GEN-LAST:event_jButton1ActionPerformed
```

```
   private    void    formWindowClosing(java.awt.event.WindowEvent    evt)    {//GEN-
FIRST:event_formWindowClosing
        Config.windows.remove(this);
   }//GEN-LAST:event_formWindowClosing

   private    void    jLabel6MouseClicked(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_jLabel6MouseClicked
      // TODO add your handling code here:
      JOptionPane.showMessageDialog(null, "This is a simple Chatting Frame built using
Java.",
                              "About Help",
                              JOptionPane.INFORMATION_MESSAGE);
   }//GEN-LAST:event_jLabel6MouseClicked

   private    void    jLabel6MouseEntered(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_jLabel6MouseEntered
      // TODO add your handling code here:
      this.setForeground(Color.red);
   }//GEN-LAST:event_jLabel6MouseEntered

   private    void    jLabel6MouseExited(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_jLabel6MouseExited
      // TODO add your handling code here:
      this.setForeground(Color.blue);
   }//GEN-LAST:event_jLabel6MouseExited

   public void receiveMessage(String msg) {
        this.txtChat.setText(this.txtChat.getText() + this.withUser + ":- " + msg + "\n");
        this.txtChat.setSelectionStart(this.txtChat.getText().length() - 1);
   }
   // Variables declaration - do not modify//GEN-BEGIN:variables
   private javax.swing.JButton jButton1;
   private javax.swing.JButton jButton2;
   private javax.swing.JButton jButton3;
   private javax.swing.JComboBox jComboBox1;
   private javax.swing.JLabel jLabel1;
   private javax.swing.JLabel jLabel2;
   private javax.swing.JLabel jLabel4;
   private javax.swing.JLabel jLabel5;
   private javax.swing.JLabel jLabel6;
   private javax.swing.JLabel jLabel8;
   private javax.swing.JScrollPane jScrollPane1;
   private javax.swing.JScrollPane jScrollPane2;
   private javax.swing.JSeparator jSeparator1;
   private javax.swing.JLabel lblFriendName;
   private javax.swing.JLabel lblYourName;
```

```
    private javax.swing.JTextArea txtChat;
    private javax.swing.JTextPane txtMessage;
    // End of variables declaration//GEN-END:variables
}
```

Chat Server

ServerForm.java

```
package ui;

import communication.ServerRelay;
import javax.swing.*;
import java.net.*;

public class ServerForm extends javax.swing.JFrame {

    ServerSocket server;

    /** Creates new form ServerForm */
    public ServerForm() {
        initComponents();
        this.TPort.setValue(1060);
    }

    @SuppressWarnings("unchecked")
    //    <editor-fold    defaultstate="collapsed"    desc="Generated    Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        Start = new javax.swing.JButton();
        Stop = new javax.swing.JButton();
        TIP = new javax.swing.JTextField();
        LIP = new javax.swing.JLabel();
        LPort = new javax.swing.JLabel();
        Progress = new javax.swing.JProgressBar();
        Sep = new javax.swing.JSeparator();
        Pic = new javax.swing.JLabel();
        LWelcome = new javax.swing.JLabel();
        TPort = new javax.swing.JSpinner();
        Hack = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("SAR Server");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        Start.setFont(new java.awt.Font("Times New Roman", 0, 14));
        Start.setForeground(new java.awt.Color(51, 0, 255));
        Start.setText("START");
        Start.setBorder(null);
        Start.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                StartMouseEntered(evt);
            }
            public void mouseExited(java.awt.event.MouseEvent evt) {
                StartMouseExited(evt);
            }
        });
        Start.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                StartActionPerformed(evt);
            }
        });
        getContentPane().add(Start, new org.netbeans.lib.awtextra.AbsoluteConstraints(30,
300, 80, 40));

        Stop.setFont(new java.awt.Font("Times New Roman", 0, 14));
        Stop.setForeground(new java.awt.Color(255, 51, 51));
        Stop.setText("STOP");
        Stop.setBorder(null);
        Stop.setEnabled(false);
        Stop.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                StopMouseEntered(evt);
            }
            public void mouseExited(java.awt.event.MouseEvent evt) {
                StopMouseExited(evt);
            }
        });
        Stop.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                StopActionPerformed(evt);
            }
        });
```

```
getContentPane().add(Stop, new
org.netbeans.lib.awtextra.AbsoluteConstraints(150, 300, 80, 40));

TIP.setEditable(false);
TIP.setFont(new java.awt.Font("Times New Roman", 0, 14));
TIP.setText("127.0.0.1");

TIP.setBorder(javax.swing.BorderFactory.createEtchedBorder(javax.swing.border.Etche
dBorder.RAISED, null, new java.awt.Color(153, 204, 255)));
getContentPane().add(TIP, new org.netbeans.lib.awtextra.AbsoluteConstraints(50,
180, 170, 30));

LIP.setFont(new java.awt.Font("Times New Roman", 0, 14));
LIP.setForeground(new java.awt.Color(0, 51, 255));
LIP.setText("IP ADDRESS");
getContentPane().add(LIP, new org.netbeans.lib.awtextra.AbsoluteConstraints(50,
150, 90, 20));

LPort.setFont(new java.awt.Font("Times New Roman", 0, 14));
LPort.setForeground(new java.awt.Color(0, 0, 255));
LPort.setText("PORT NO");
getContentPane().add(LPort, new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 220, 80, 20));
getContentPane().add(Progress, new
org.netbeans.lib.awtextra.AbsoluteConstraints(60, 360, -1, 20));
getContentPane().add(Sep, new org.netbeans.lib.awtextra.AbsoluteConstraints(0,
84, 270, -1));

Pic.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/server.jpg"))); // NOI18N
getContentPane().add(Pic, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0,
-1, 70));

LWelcome.setFont(new java.awt.Font("Times New Roman", 0, 18));
LWelcome.setForeground(new java.awt.Color(255, 51, 102));
LWelcome.setText("Welcome To The SAR Server");
getContentPane().add(LWelcome, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 100, 250, 30));

TPort.setFont(new java.awt.Font("Times New Roman", 0, 14)); // NOI18N
TPort.setBorder(javax.swing.BorderFactory.createEtchedBorder(null, new
java.awt.Color(153, 204, 255)));
getContentPane().add(TPort, new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 250, 170, 30));

Hack.setFont(new java.awt.Font("Times New Roman", 0, 14));
```

```
        Hack.setForeground(new java.awt.Color(0, 255, 0));
        getContentPane().add(Hack, new org.netbeans.lib.awtextra.AbsoluteConstraints(75,
390, 120, 20));

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void formWindowClosing(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_formWindowClosing
        this.stopServer();
    }//GEN-LAST:event_formWindowClosing

    private void StopActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_StopActionPerformed
        this.stopServer();
    }//GEN-LAST:event_StopActionPerformed

    private void StartActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_StartActionPerformed
        Runnable r = new Runnable() {

            public void run() {
                startServer();
            }
        };

        Thread t = new Thread(r);
        t.start();
    }//GEN-LAST:event_StartActionPerformed

    private void StartMouseExited(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_StartMouseExited
        this.Start.setBorder(null);
    }//GEN-LAST:event_StartMouseExited

    private void StartMouseEntered(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_StartMouseEntered
        this.Start.setBorder(javax.swing.BorderFactory.createEtchedBorder(null, new
java.awt.Color(153, 204, 255)));
    }//GEN-LAST:event_StartMouseEntered

    private void StopMouseEntered(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_StopMouseEntered
        this.Stop.setBorder(javax.swing.BorderFactory.createEtchedBorder(null, new
java.awt.Color(153, 204, 255)));
    }//GEN-LAST:event_StopMouseEntered
```

```
  private    void    StopMouseExited(java.awt.event.MouseEvent    evt)    {//GEN-
FIRST:event_StopMouseExited
        this.Stop.setBorder(null);
  }//GEN-LAST:event_StopMouseExited

  private void startServer() {
        int port = (Integer) this.TPort.getValue();
        try {
            server = new ServerSocket(port);

            this.Progress.setIndeterminate(true);
            this.Start.setEnabled(false);
            this.Stop.setEnabled(true);
            this.Hack.setText("Server Running");

            while (true) {
                final Socket client = server.accept();
                Runnable r = new Runnable() {

                    public void run() {
                        new ServerRelay(client).handleCommunication();
                    }
                };

                Thread t = new Thread(r);
                t.start();
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
}

private void stopServer() {
        try {
            if (this.server != null) {
                this.Start.setEnabled(true);
                this.Stop.setEnabled(false);
                this.Progress.setIndeterminate(false);
                this.Hack.setText("Server stoped");
                this.server.close();
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
}
```

```
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new ServerForm().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel Hack;
    private javax.swing.JLabel LIP;
    private javax.swing.JLabel LPort;
    private javax.swing.JLabel LWelcome;
    private javax.swing.JLabel Pic;
    private javax.swing.JProgressBar Progress;
    private javax.swing.JSeparator Sep;
    private javax.swing.JButton Start;
    private javax.swing.JButton Stop;
    private javax.swing.JTextField TIP;
    private javax.swing.JSpinner TPort;
    // End of variables declaration//GEN-END:variables
}
```