Spring 2010

Senior Project Report

Computer Science Department

California State University, **Dominguez Hills**

Kryptonize – A Cryptographically Secure Web-Based Messaging System

Prepared by

Eric Flior

In
Partial fulfillment of the requirements

For
Senior Design – CSC 492

Department of Computer Science
California State University Dominguez Hills
Spring 2010

Committee Members/Approval

Dr. Kazimierz Kowalski		
Faculty Advisor	*Signature*	*Date*
Dr. Mohsen Beheshti		
Committee Member	*Signature*	*Date*
Dr. Jianchao 'Jack' Han		
Committee Member	*Signature*	*Date*
Dr. Mohen Beheshti		
Department Chair	*Signature*	*Date*

ABSTRACT

As internet use becomes more widespread, privacy concerns for the average internet user grow. Many organizations already use cryptography for a variety of reasons, including protection of trade secrets and engaging in e-commerce. Despite the availability of numerous symmetric and asymmetric cryptographic schemes, cryptography is not in widespread use among average internet users. Implementing secure communications can be both difficult and time-consuming, requiring a great deal of technical expertise and coordination between all parties intending to communicate. In this paper, we describe Kryptonize, our implementation of a web-based messaging system, similar to e-mail, which allows parties to communicate securely via the internet using the Elgamal asymmetric cryptosystem. In Kryptonize, the use of cryptography is abstracted from the user, providing private communication between parties while transparently removing the technical and logistic pre-requisites inherent in using cryptography.

ACKNOWLEDGEMENT

I would like to acknowledge and thank my faculty advisor, Dr. Kazmierz Kowalski, Professor of Computer Science along with my committee members, Dr. Mohsen Beheshti, and Dr. Jianchao Han, Professors of Computer Science at California State University Dominguez Hills for their guidance, support, and assistance with this project.

In addition, I would like to acknowledge and thank Dr. Alexander Stanoyevitch, Professor of Mathematics at California State University Dominguez Hills for fostering my interest in studying cryptography and mentoring me in the pursuit thereof.

Table of Contents

Index of Figures

1. INTRODUCTION

Computing devices have become a part of everyday life in the United States. The widespread availability of desktop and laptop computers, netbooks, and smart phones, combined with the ubiquity of the Internet means that a large percentage of people's day-to-day communications occur though the use of one of these computing devices.

In turn, as our individual presence on the Internet grows, so too does our need for private and secure communication when using these channels. Many users are shocked and dismayed to discover that the distributed nature of the Internet, along with the fact that a majority of Internet traffic is transmitted in plaintext, means that not only are communications sent to their intended recipient, but are also readable to every network node between the source and destination.

The power of modern computing devices along with advances in cryptography make it possible to communicate privately using public channels, but the coordination and implementation of these cryptographic schemes is often beyond the grasp of the average user. As a result, despite the widespread availability of strong cryptographic algorithms, most current uses of cryptography are limited to corporate, government, or military purposes. Companies with a web presence use cryptography to secure e-commerce transactions, and the government and military use cryptography to communicate privately and prevent espionage, however most consumer grade cryptographic implementations can be unwieldy, requiring software installation and extensive configuration, and therefore are not in widespread use.

We therefore aim to create a method of secure communication via the internet, in which the security of the system is transparent to the end-user. Toward this end, we have developed a

Java based software system, called Kryptonize, which provides secure web-based communication using strong cryptography.

The web-based nature of the system provides ease of use and widespread availability; anyone with an internet connection and a web browser can communicate securely with no requirement to install and configure complicated software. Kryptonize uses public-key cryptography, specifically the Elgamal cryptosystem, in order to facilitate private and secure communication between parties.

2. BACKGROUND

In order to properly implement our Elgamal based cryptosystem to facilitate secure communication, one must first understand a number of concepts. These concepts include asymmetric and symmetric cryptographic methods, computational intractability and discrete logarithms, safe prime numbers and generators, the Elgamal algorithm itself, and methods of large integer computing in Java.

2.1 Asymmetric Cryptography vs. Symmetric Cryptography

Asymmetric cryptography, also known as public-key cryptography, is a method of encryption which uses one-way functions to facilitate secure communication between parties. Prior to the invention of asymmetric cryptography, all methods of cryptography were what are now known as private-key, or symmetric cryptosystems, because they require both parties, the sender and the receiver, to have the same key in order to communicate.

Although symmetric cryptosystems are fast and efficient, there are difficulties in implementing them because both parties have to know the same secret key. At some point in the implementation of a symmetric cryptosystem, there must be a transmission of a secret key from one user to another. If this transmission occurs across an insecure channel, the secret key can be intercepted, compromising the integrity of any subsequent secure communications. This can lead to unauthorized parties eavesdropping on the private message, or masquerading as an authorized sender of messages. (Mackey, 2003)

Public-key cryptography addresses this issue with the use of keypairs, in which a person who wishes to receive secure messages creates two keys, instead of one. One key, known as the private key, is generated and kept secret by the user. The other key, known as the public key, is freely given out to any party who wants to communicate with the holder of the private key. When the communication occurs, the sender of the message uses the recipient's public key in a one-way function to encrypt the message. The recipient of the message takes the resulting ciphertext and uses their private key in order to retrieve the original message.

Despite the advantage of being able to communicate with any party without having to exchange a secret key, public-key cryptosystems have a drawback in that they require more computation than symmetric key cryptosystems. As a result, they can be considered inappropriate for large amounts of data. (Weaver, 2007)

2.2 The Discrete Logarithm Problem and Computational Intractability

There are three types of currently intractable mathematical problems which are widely used in practical cryptographic schemes. They are the Integer Factorization Problem, the Discrete Logarithm Problem, and the Elliptic Curve Discrete Logarithm Problem. These are mathematical problems for which no polynomial time solution is known to exist. (Nichols, 1999)

The discrete logarithm problem is commonly used in public-key cryptosystems to provide security. Recall that the logarithm function provides $\log_b a = n$ if, and only if $a = b^n$. The discrete logarithm problem is an extension of this function, modulo a prime number p. Hence, the discrete logarithm problem can be stated thusly: given integers a and b, $(a > b)$ along with the prime p, find an integer n such that $a \equiv b^n \ (mod\ p)$.

There are a number of algorithms which have been developed to calculate the discrete logarithm. These include exhaustive search, also known as brute force, Baby-step Giant step, Pollard's rho algorithm, Pollard's lambda algorithm, and the Polhig-Hellman algorithm. The brute force algorithm for calculating the discrete logarithm runs in $O(2^{n/2})$ time, while the others run in $O(\sqrt{n})$ time.

At first glance, the fact that some of these algorithms run in $O(\sqrt{n})$ time appears to be contrary to the assertion that there are no polynomial time algorithms for calculating the discrete logarithm. However, the Baby-step Giant step, Pollard's rho algorithm, Pollard's lambda algorithm, and the Polhig-Hellman algorithm run in pseudo-polynomial time, meaning that the running time is polynomial in the numeric value of the input, but exponential in the length of the input. (Garey & Johnson, 1979) Because of this, the discrete logarithm problem is intractable for sufficiently large values of n. Given modern computing power, a 1024 bit-length number is considered sufficient to provide security for the purposes of secure communications. (Bishop, 2003)

2.3 Safe Prime Numbers and Generators

A prime number, p, is susceptible to the Polhig-Hellman algorithm if $p - 1$ is composed of only relatively small prime factors. Therefore, when selecting a prime number for use in a cryptosystem which depends on the difficulty of the discrete logarithm problem, it is important to pick a prime number p such that $p-1$ has at least one large prime factor. A prime number with this property is called a safe prime. (Bishop, 2003)

In addition to relying on safe prime numbers, most cryptosystems which derive their strength from the discrete logarithm problem also use numbers known as generators. Bishop defines a generator as "An integer g such that the prime p does not divide g is called a generator modulo p if $ord(g)(mod\ p) = p-1$." He notes that "… the discrete logarithm problem is most intractable when generators are used as the base… when we have $g^x \equiv b\ (mod\ p)$ for prime p and generator g, the solution… is unique, and, it turns out, harder to find." (Bishop, 2003)

2.4 Elgamal Cryptosystem

The Elgamal Cryptosystem is a public-key cryptosystem which gets its strength from the discrete logarithm problem. There are three separate algorithms that comprise the Elgamal cryptosystem: a key generation algorithm, an encryption algorithm, and a decryption algorithm.

2.4.1 Key Generation

Key Generation for the Elgamal cryptosystem is comprised of four steps.

1. The message recipient chooses a large prime number, p, and a generator g modulo p.
2. The recipient chooses a random integer a, such that $1 < a < p-1$.
3. The recipient computes $r \equiv g^a\ (mod\ p)$.
4. The recipient makes the 3-tuple (p, g, r) public, and keeps their random integer a as their private key.

Note that, in step 3, although r is computed using the recipient's private key, a, and $g, r,$ and p are subsequently made public, determining the recipients private key from these numbers requires the use of an algorithm which solves the discrete logarithm problem, which, as we have discussed, cannot currently be done in polynomial time.

2.4.2 Encryption

Encryption in the Elgamal cryptosystem is performed in five steps.

1. Obtain the public key belonging to the message recipient.
2. Select a plaintext integer m, such that $m < p$.
3. Choose a random integer k, known as an ephemeral key, such that $1 < k < p - 1$.
4. Compute $c \equiv g^k \ (mod\ p)$.
5. Compute $d \equiv mr^k \ (mod\ p)$.

Once these calculations are complete, the sender sends the 2-tuple (c, d) to the message recipient as the ciphertext.

2.4.3 Decryption

Upon receipt of a ciphertext, the message recipient performs the following two steps to recover the original plaintext.

1. The recipient uses their private key,a, to compute $z \ \equiv c^a \ (mod\ p)$.
2. The recipient then computes the plaintext, $m \equiv zd \ (mod\ p)$. (Bishop, 2003)

2.5 Large Integer Computing in Java

Java has a number of primitive data types which are designed to hold integer values. The byte data type is a 8-bit signed two's complement integer, which has a maximum value of 127.

The short data type is a 16-bit signed two's complement integer, which as a maximum value of 32,767. The int data type, which is generally the default data type for storing integers values, is a 32-bit signed two's complement number, which has a maximum value of 2,147,483,647. The largest primitive data type designed to hold integers in Java is the long, which is a 64-bit signed two's complement integer, which has a maximum value of 9,223,372,036,854,775,807. (Oracle Corporation, 2010)

When performing large integer computing in Java, it is not unreasonable to have an integer which is 1024 or 2048 bits in length. Considering that a 1024 bit-length number can have over 300 digits, it is obvious that such a number would clearly overflow even an unsigned long data type by a great deal.

In order to accommodate the needs of large integer computing, Java provides the BigInteger class. This class allows large integers to be represented as immutable arbitrary-precision integers. The Java SE 6 documentation indicates that all operations on the BigInteger class "behave as if BigIntegers were represented in two's-complement notation (like Java's primitive integer types). BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations." (Oracle Corporation, 2009)

The BigInteger classes' ability to not only handle large integers, but also its facility for modular arithmetic, primality testing, and prime generation make this class ideal for cryptographic applications of large integer computing.

3. THE KRYPTONIZE SECURE MESSAGING SYSTEM

The primary aim of the Kryptonize system is to provide user-to-user communication secured with strong cryptography in a manner transparent to the end user. To accomplish this, the system uses the Elgamal cryptosystem to facilitate communication not only between the client and server, but between users of the system as well.

The Kryptonize secure messaging system is implemented as a web-based client/server application, written in Java. Messages between users, as well as users public keys, are stored in encrypted form in a MySQL database.

The client application is accessible as a Java applet embedded in a HTML document which is stored on a webserver. The server is multithreaded, allowing multiple users to connect at once, and is deployable on any server with access to the internet and MySQL server.

3.1 MySQL Database Design

The Kryptonize system requires a MySQL server with a database called “secmail”. The secmail database contains three tables, “users”, “mail”, and “p_keys”. The schema for these tables is represented in Figure 1.

The “users” table holds three fields, a varchar(30) primary key, “username”, and “pass_c” and “pass_d”, both of type text. The ciphertext representing the plaintext password for the user is stored in “pass_c” and “pass_d”.

The “mail” table is related to the “users” table, and has an integer primary key “messageID”, and a varchar(30) foreign key “username”, from the “users” table. The remaining fields in the table are a varchar(30) “fromname”, which holds the username of the message sender.

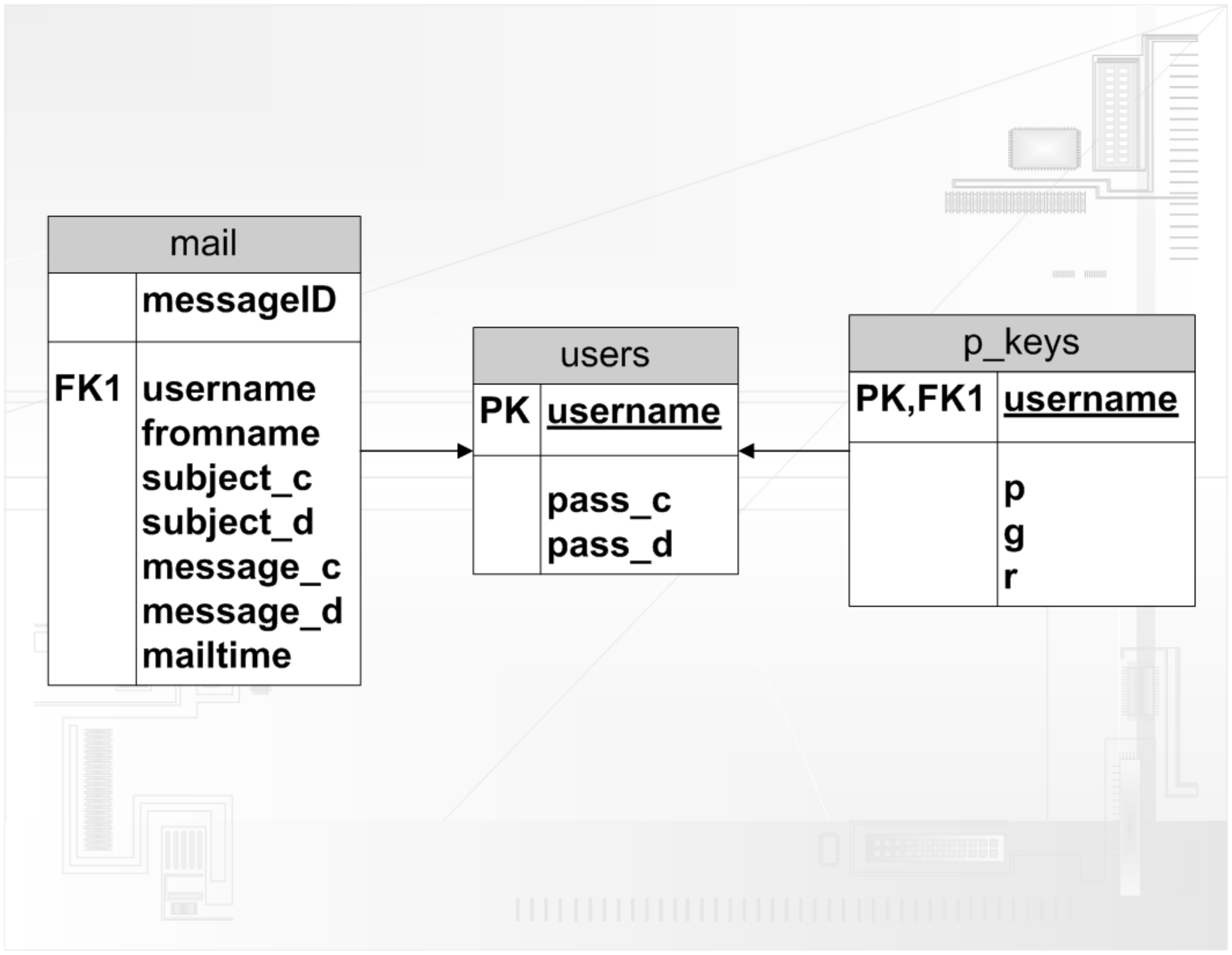


Figure 1

The “subject_c”, “subject_d”, “message_c”, and “message_d” are text fields which hold the ciphertext corresponding to the subject and message which was sent to the user. The “mailtime” field is a timestamp type, and defaults to CURRENT_TIMESTAMP, which is set to the time when the record is added.

The “p_keys” table is also related to the “users” table, and has a varchar(30) primary key, “username” which is also a foreign key from the “users” table. The “p”, “g” and “r” fields are all text types, which hold the public keys for each user.

3.2 Java Class Structure

The Kryptonize system is comprised of six Java classes. Two classes, SafePrimeGenerator.java and Elgamal.java are dedicated to the implementation of the Elgamal cryptosystem. The MessageArray.java class is an abstract data type used by the SecureMessageClient.java class to store messages retrieved from the server. The multithreaded server is implemented in two classes, SecureMessageServer.java and SecureMessageThread.java. We now discuss the structure of these classes.

3.2.1 SafePrimeGenerator.java

SafePrimeGenerator.java (Appendix 6.2) is a class that creates an object to generate a random safe prime and generator for use in the Elgamal cryptosystem. Bishop describes a method of generating large safe primes which the SafePrimeGenerator.java class adapts and uses.

After object instantiation, the .GenerateSafePrime() method returns a BigInteger array which contains a 1024 bit safe prime and associated generator. The method begins by creating a BigInteger, t, which is a 1024 bit prime number. It then finds the first prime of the form $p = 2rt + 1$, using successive values of r, beginning with $r = 1$. This prime, p, is used as the safe prime in the Elgamal key generation.

In order to determine a generator for p, the method next finds the prime factors of $p - 1 = 2rt$. Two trivial factors are 2, and t, and are stored in a Vector, factors, for later retrieval. The method then tests r for primality using the .isProbablePrime() method of the BigInteger class. If r is prime, it is added to the vector, otherwise all potential prime factors are checked sequentially beginning with 3 and continuing to $\sqrt{r}$.

Next the method finds a generator for p, by generating a random integer x, where $1 \leq x \leq p - 2$. In order to determine if x is a generator, we divide $p - 1$ by each element of the Vector factors, and raise x to each of these values modulo p. If any of these powers of x are congruent to 1 $(mod\ p)$, x is not a generator. In this situation, a new random integer is generated, and the process repeats until a generator is found. (Bishop, 2003)

3.2.2 Elgamal.java

The Elgamal.java class (Appendix 6.3) consists of three static methods: .GenerateKeys(), used for generation of asymmetric key pairs; .ElgamalEncipher(), which performs the Elgamal encryption algorithm; and .ElgamalDecipher(), which performs the Elgamal decryption algorithm. Due to the fact that all three methods are static, the Elgamal.java methods can be called from another class without having to instantiate an Elgamal object. This provides a great deal of flexibility and ease of use to the calling classes.

The .GenerateKeys() method returns a BigInteger array which contains the public key (p, g, r) and the private key a. It begins by instantiating a SafePrimeGenerator object, and calling the .GenerateSafePrime() method to obtain a safe prime p, and a generator g. It then selects a random integer a, where $2 \leq a \leq p - 1$ and computes $r \equiv g^a\ (mod\ p)$.

The .ElgamalEncipher() returns a BigInteger array which contains $C = (c, d)$,the ciphertext, and takes four arguments: a plaintext String, and three BigIntegers, p, g and r, the public key of the message recipient. Since the Elgamal algorithm requires the plaintext message to be represented as an integer m, the plaintext String argument is converted into its corresponding Byte array. This byte array is then used to construct a new BigInteger m, suitable for encryption. The method then performs the Elgamal encryption algorithm by generating a random BigInteger k, where $1 \leq k \leq p - 1$, and uses it to compute the ciphertext $c \equiv g^k \ (mod\ p)$ and $d \equiv mr^k \ (mod\ p)$.

To decrypt a ciphertext, the .ElgamalDecipher() method takes four BigInteger arguments, c and d, the ciphertext, along with a and p, the message recipient's private key and safe prime. The method computes $z \equiv \ (c^a)' \ (mod\ p)$, and then recovers the BigInteger representation of the ciphertext by computing $m \equiv zd \ (mod\ p)$. In order to turn this back into the plaintext String, the method then recovers the Byte array and uses it to create a String object, the original plaintext.

3.2.3 MessageArray.java

The MessageArray.java class (Appendix 6.4) is an abstract data type which allows the SecureMessageClient.java class to store and access deciphered plaintext messages after their retrieval from the server. It contains a constructor, two mutator methods, .Addmail() and .DeleteMail(), as well as three accessor methods, .GetSize(), .GetMail(), and .GetMessageNumber().

Each secure message retrieved from the server contains seven fields: the messageid assigned by the MySQL database; the username of the message recipient; the username of the message sender; the date the message was sent; the time the message was sent; the subject line; and the actual plaintext message. Each of these fields is stored as a String and indexed in a String array.

The .AddMail() method takes a message in the form String array as an argument and adds it to the ArrayList. When adding a message, the method first determines if the ArrayList is full, and doubles the capacity of the ArrayList prior to adding the message.

The .DeleteMail() method takes the index of the message to be deleted, and removes it from the ArrayList.

The .GetSize() accessor method returns the number of String array elements in the ArrayList.

The .GetMail() accessor method takes the index of the message to be retrieved and returns the String array containing the message.

Finally, the .GetMessageNumber() accessor method returns the messageid of the desired message.

3.2.4 SecureMessageServer.java

The SecureMessageServer.java (Appendix 6.5) is an executable class which functions as the server which listens for connections from the SecureMessageClient.java class.

The class begins by opening a ServerSocket object listening at a pre-determined port. The server then enters into a loop which waits indefinitely while listening for a client connection. When the server receives a connection attempt, it constructs a new Thread object defined in SecureMessageThread.java, passes it the information from the client connection, and then starts the thread.

3.2.5 SecureMessageThread.java

When the SecureMessageThread.java (Appendix 6.6) thread is started, it creates a PrintStream and InputStreamReader for input and output to the client. It also creates a connection to the MySQL database using MySQL Connector/J, “a native Java driver that converts JDBC (Java Database Connectivity) calls into the network protocol used by the MySQL database,” (Oracle Corporation, 2010) downloaded from http://dev.mysql.com/downloads/connector/j/.

SecureMessageThread.java recognizes seven commands from the client application, each of which has an associated method. The recognized commands are: Login; RegisterUser; RegisterPublicKey; GetMessages; CreateMessage; DeleteMessage; and Logout.

When the Login command is received, SecureMessageThread.java queries the users table and retrieves the user’s stored password for comparison with the password received from the client. If the passwords match, the system provides the user with a login confirmation. The thread then queries the database for the user’s stored public key and transmits it to the client, allowing the client to decrypt transmissions from the server.

The RegisterUser command performs a "SELECT count(*) FROM users WHERE username='<username>’"query on the users table to make sure the client’s desired user name does not already exist. If the query returns 0, the username does not exist and the table is updated with the new username and password.

The RegisterPublicKey command uses the following query, "INSERT INTO p_keys (username, p, g, r) VALUES ('"<username>"', '"<p>"', '"<g>"', '"<r>"')" to update the p_keys table with the public key for the user, provided by the client application.

When the GetMessages command is received, the thread performs a SELECT query on the mail table and retrieves all records which contain the client's username. These records are sent to the client application for processing.

Upon invocation of the CreateMessage command, the thread updates the mail table with an INSERT query, and adds the recipient's username, the sender's username, and the ciphertext for the subject and message.

The DeleteMessage command updates the mail table with a DELETE query to remove the message with the specified messageID.

Finally, the Logout command cleanly releases the ClientSocket and closes the thread.

3.2.6 SecureMessageClient.java

The SecureMessageClient.java (Appendix 6.7) class is a Java applet intended to be embedded in an HTML page. This class provides a Graphical User Interface designed using Swing which allows the user to access the server.

The GUI is implemented as a series of jPanels and jDialogs organized in a CardLayout layout manager. The CardLayout layout manager allows each jPanel to be displayed sequentially as the user progresses through the applet. The CardLayout layout manager is so named because it treats each jPanel much like a playing card in a deck. Only the jPanel at the top of the deck is displayed at a given time, and the jPanels can be displayed in any order by placing them at the top of the deck.

The SecureMessageClient.java class incorporates six jPanels to create the GUI. The purpose of the primary jPanel is to hold the layout manager. The five child jPanels provide: a login and user registration interface; a public and private keypair generation and registration panel; an interface to load the private key file; a message reading interface; and a message creation interface.

3.3 Kryptonize Operation

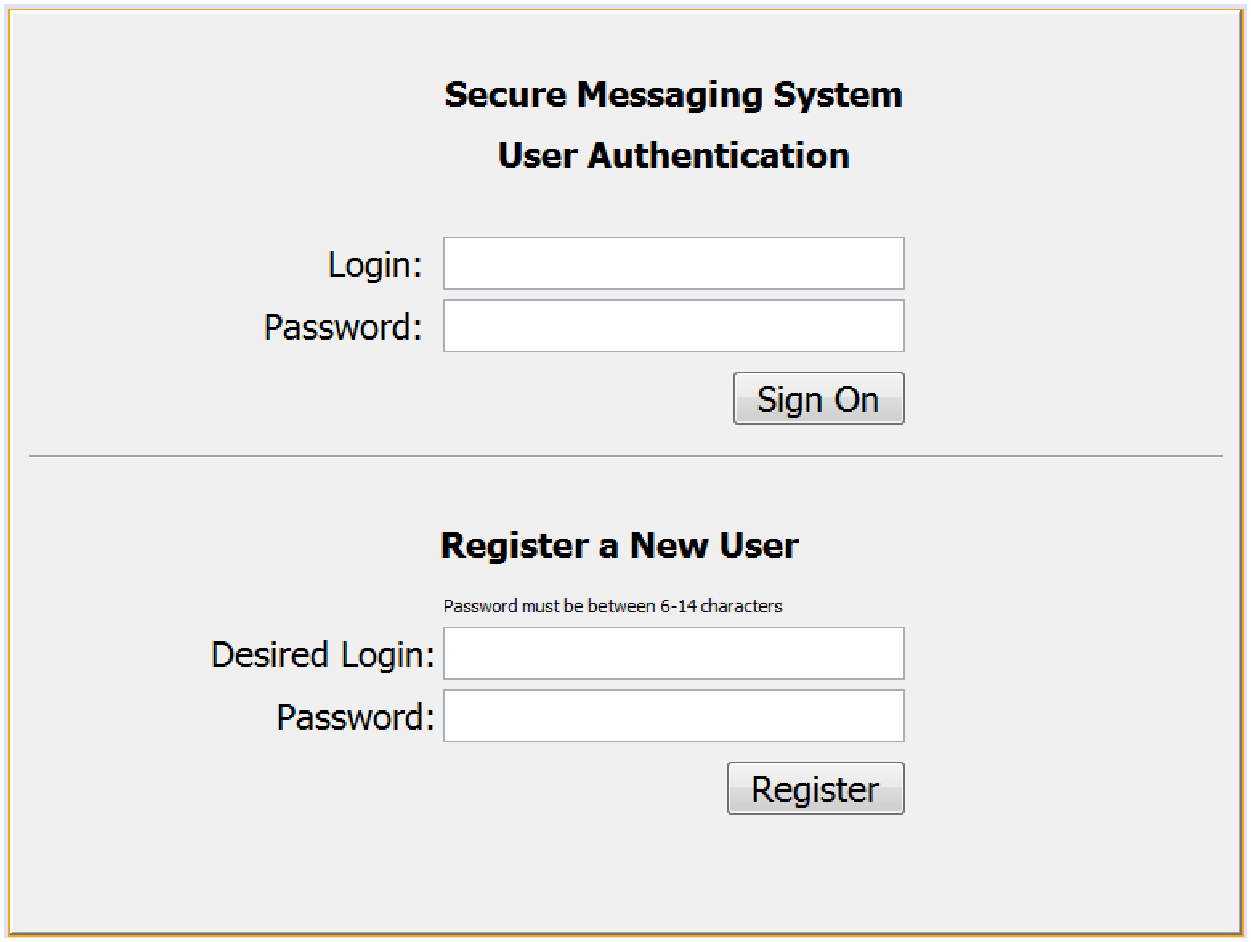


Figure 2

Operation of the Kryptonize secure messaging system requires the SecureMessagingServer.java class to be running on a system which is also running a MySQL server and has access to the internet. The SecureMessagingClient.java class is embedded in a HTML based webpage, which can be hosted on any webserver. Upon loading the webpage containing the Kryptonize applet, the user is presented with a GUI requiring the user to either login or register as a new user (Figure 2).

3.3.1 Existing User Login

Assuming the user already has an account on the system, the user can enter their existing login and password in the corresponding text fields shown in Figure 2. Clicking the "Sign On" button initiates a connection to the SecureMessageServer, and sends the command "Login <username> <password>" to the server, encrypted using the server's public key.

The server then retrieves the user's encrypted password, stored in the users table of the secmail database. The server decrypts the password using its private key, and compares the stored password with the one provided by the client. If the passwords match, the server retrieves the public key associated with the provided username from the p_keys table of the secmail database, and sends the public key to the client so that it can decrypt messages sent from the server. Once public key transmission is complete, the server directs the client applet to display the private key loading screen.

Figure 3 shows a UML Use-Case diagram outlining the Login procedure. All other client/server interactions follow a similar pattern.

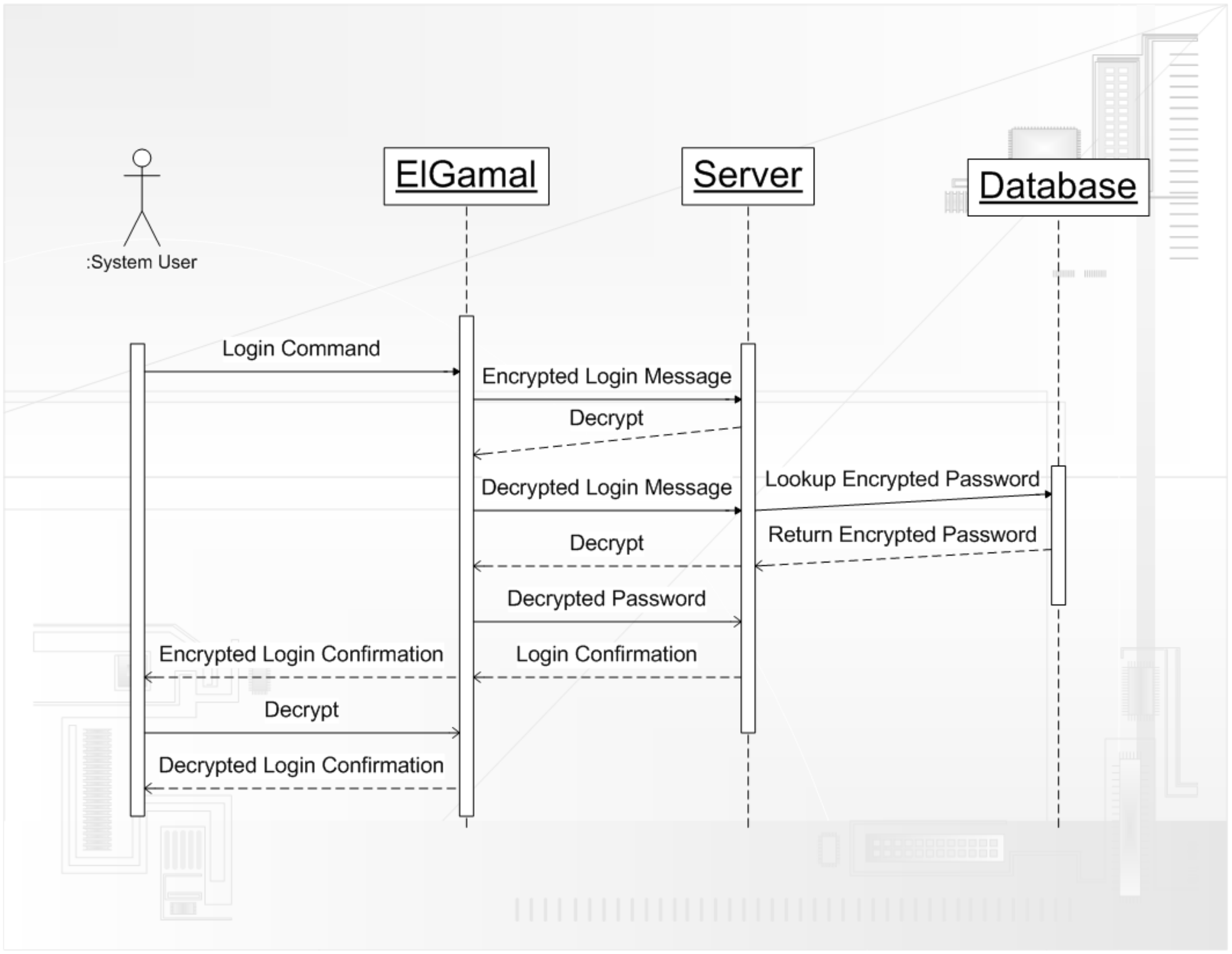


Figure 3

3.3.2 New User Registration

If a user does not have an account on the system, they can enter their desired username and password into the "Register a New User" textfields. Clicking on the "Register" button initiates a connection to the SecureMessageServer, and sends the command "RegisterUser <username> <password>" to the server, encrypted using the server's public key. Providing the username is not already taken, the server creates the new account and directs the client applet to proceed to key generation.

3.3.3 Key Generation

Prior to key generation, the user is directed to select a directory where they would like to save their private key file, which is stored as a data file named ”sms.pkey” (Figure 4).

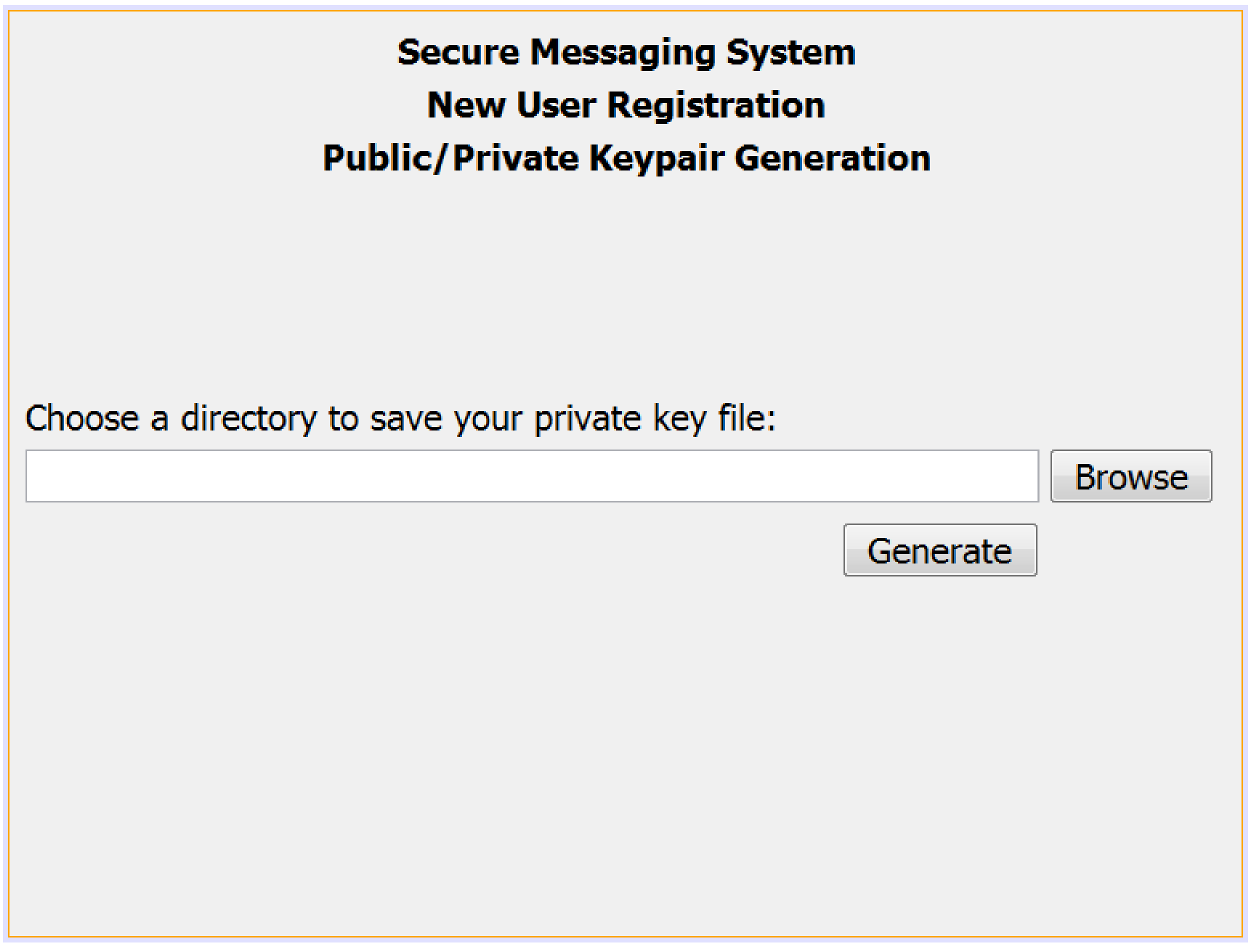

Secure Messaging System
New User Registration
Public/Private Keypair Generation
Choose a directory to save your private key file:
Browse
Generate

Figure 4

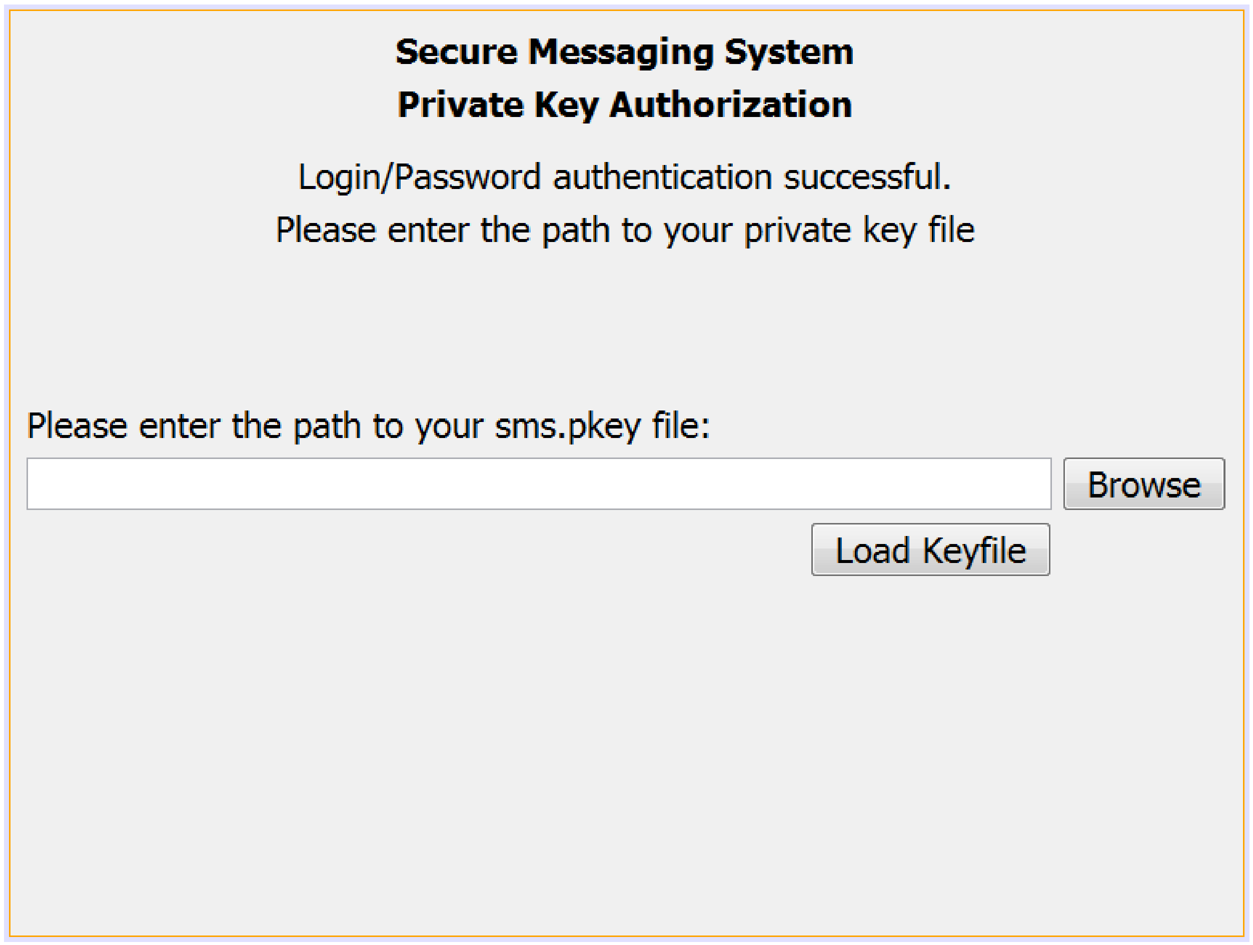
Once a directory has been selected, either by typing the directory into the text box or by clicking the “Browse” button to open a file dialog, clicking the “Generate” button calls the Elgamal.GenerateKeys() method to generate a 1024 bit public/private keypair. The public key is sent to the server, which stores the user's public key in the secmail database, and the private key is written to the disk. The user is instructed to keep their private key secure, as the private key cannot be regenerated if it is corrupted or lost. Once key generation is complete, the server directs the client applet to proceed to the message screen, discussed in section 3.3.5.

3.3.4 Private Key Authorization

After successful login, the user is presented with a screen requesting that the user provide the client application with their previously stored "sms.pkey" file (Figure 5). This is required in order to properly decrypt any messages retrieved from the server, as well as to decrypt encrypted server messages.

After the path to the file has been entered, either by typing the path to the "sms.pkey" file, or clicking the "Browse" button to open a file dialog, clicking the "Load Keyfile" button loads the private key into the client's memory. Once the private key has been loaded, the client sends the command "GetMessages <username>" to the server, encrypted using the server's public key.

The server then retrieves all messages from the mail table of the secmail database which have been sent to the user. For each retrieved message, the server encrypts the recipient's username, sender's username, mail date and mail time using the client's public key, and sends this information to the client along with the ciphertext for the subject and message. The client receives this data, decrypts it, and populates the MessageArray with any received messages. Once the server has sent the final message, it directs the client applet to proceed to the messages screen.



Secure Messaging System
Private Key Authorization

Login/Password authentication successful.
Please enter the path to your private key file

Please enter the path to your sms.pkey file:

Figure 5

3.3.5 Messages Panel

The messages panel lies at the center of the Kryptonize client applet. This panel (Figure 6) provides the user with an interface to select and display received messages, as well as buttons which allow the user to check for recently received messages, send new messages,reply to received messages, forward messages to other users, delete received messages, and logout from the system.

When the messages panel is displayed, the client retrieves all messages from the MessageArray, and dynamically populates the jTable which holds the date/time, sender, and subject of each received message.

After the table has been populated, clicking on any row of the table fires an event which determines the selected row and retrieves the appropriate message from the MessageArray. Once the message has been retrieved, the jTextArea is updated with the contents of the message.

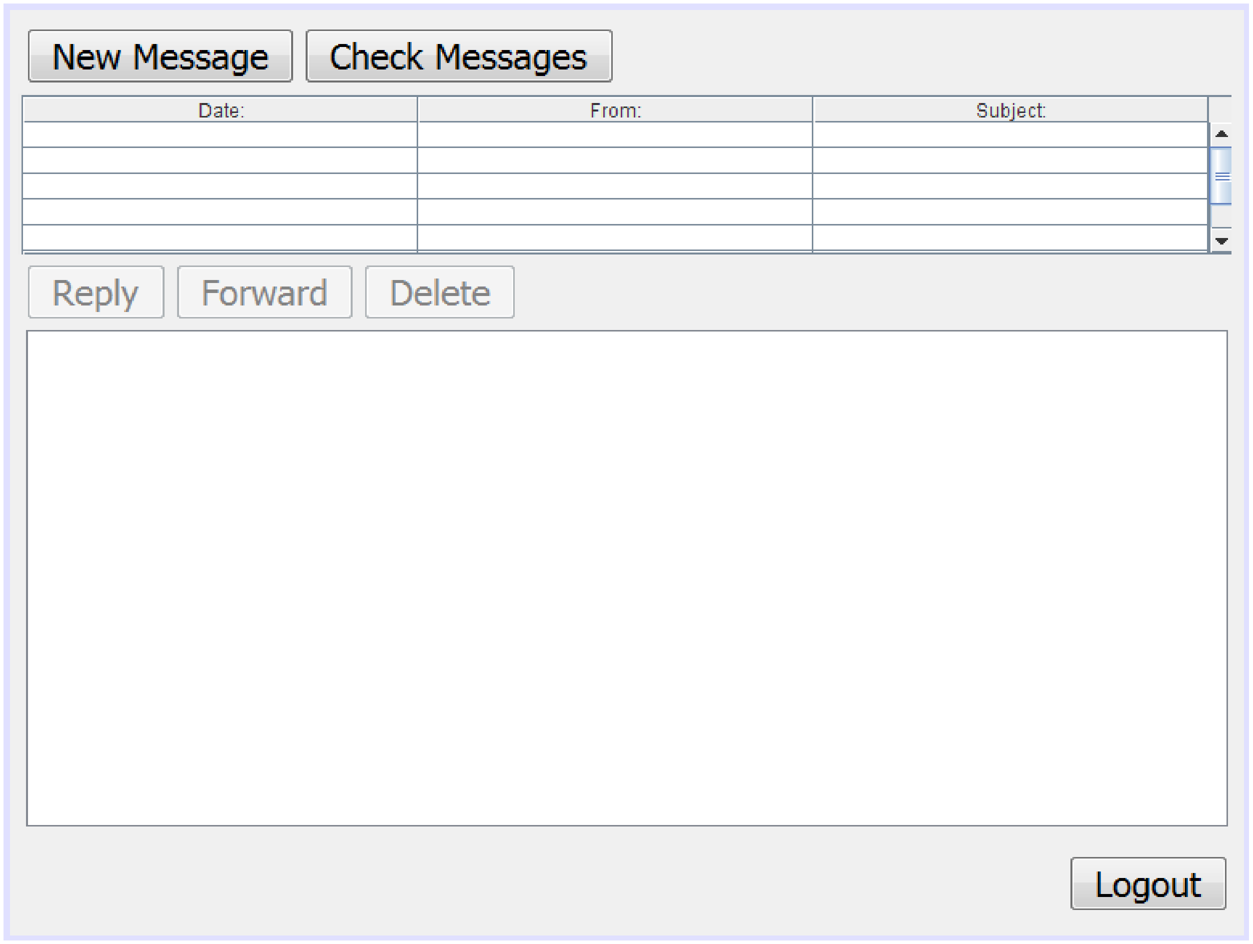


Figure 6

Clicking the “Check Messages” button retrieves all messages from the server, and adds any newly retrieved messages to the MessageArray. If any new messages have been retrieved, the client clears and re-populates the jTable which holds the message information.

When a message is selected, the “Reply”, “Forward”, and “Delete” buttons are enabled, allowing the user to reply to the message, forward the message to another user, or delete the message from the server, working in a similar fashion to e-mail.

If a message has been selected, and the “Delete” button is pressed, the client applet determines the messageid of the selected message, removes the message from the MessageArray structure, and sends a command to the server to delete the record from the mail table of the secmail database which contains the messageid of the selected message.

If the “New Message” button is pressed, the client applet displays the NewMessage panel (Figure 7), with a blank “To:”, “Subject” and “Message” fields.

If a message has been selected, and the “Reply” button is pressed, the client applet displays the NewMessage panel and pre-populates the “To:”, “Subject”, and “Message” fields with the username of the user who sent the selected message, “Re: <subject>”, where <subject> is the subject of the selected message, and “<username> said:” followed by the selected message, respectively.

If a message has been selected, and the “Forward button is pressed, the client applet displays the NewMessage panel and pre-populates the “Subject”, and “Message” fields with “Re: <subject>”, where <subject> is the subject of the selected message, and “<username> said:” followed by the selected message, respectively. The “To:” field is left blank so that the user can enter the username of the user to forward the message to.

If the “Logout” button is pressed, the client applet sends the “logout” command to the server, unloads the user’s private key file, and returns to the login screen.

3.3.6 NewMessage Panel

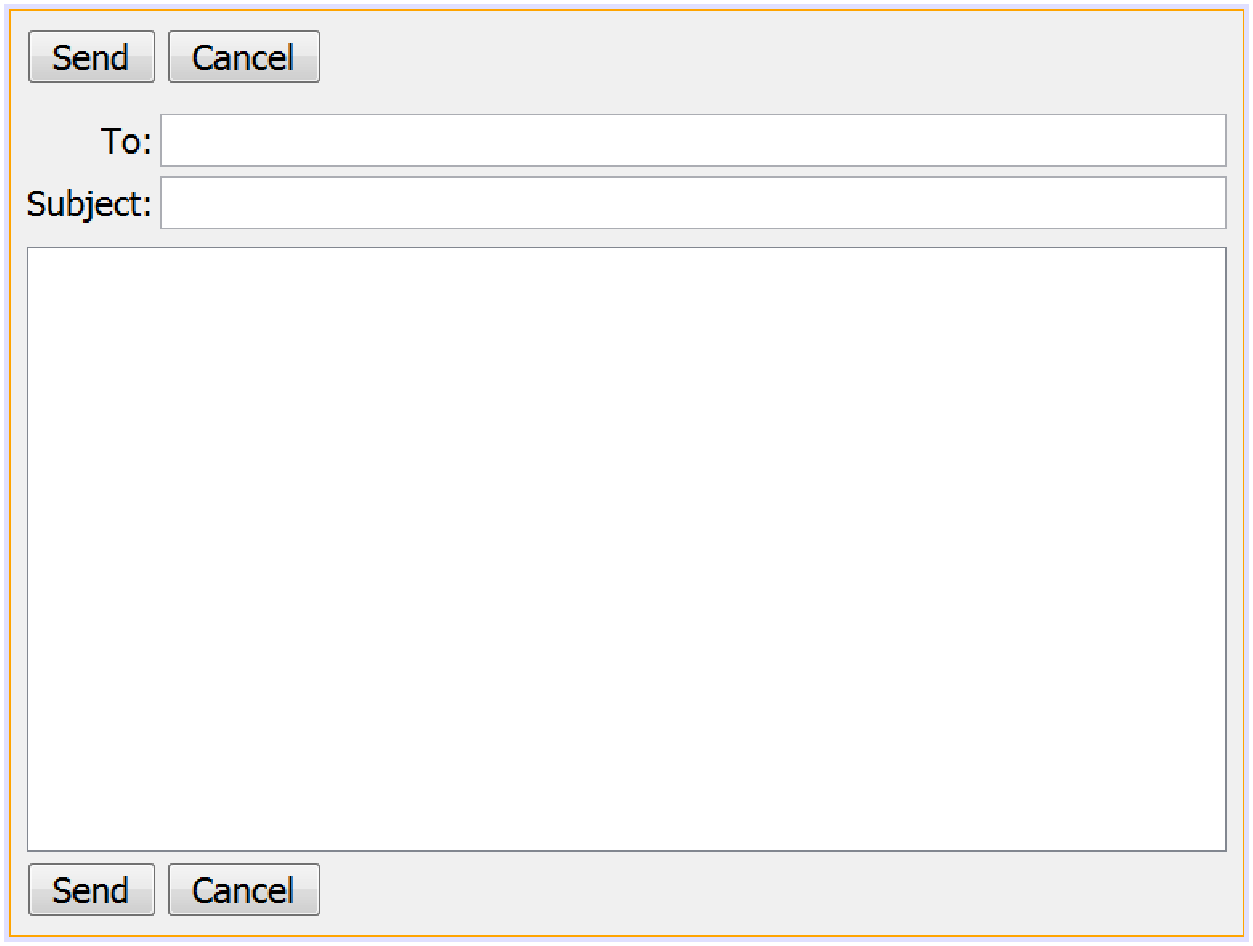


Figure 7

This panel (Figure 7) provides an interface for the user to send a message to another user. The user enters the username of the Recipient, the message subject, and enters the message. When either send button is selected, the client encrypts the command “NewMessage <messagerecipient>” using the server’s public key, and sends it to the server. The server decrypts the command, and determines if the intended recipient is a registered user in the system. If the recipient is a registered user, the server retrieves the recipient's public key from the p_keys table of the secmail database, and sends it to the client. The client uses the recipient's public key and encrypts the subject and message using the recipient's public key. The ciphertext is then sent to the server, which inserts the message into the mail table for later retrieval by the recipient.

4. CONCLUSION / FUTURE WORK

We believe that by providing users with an easy-to-use, widely available means of communicating securely on the Internet, people will take advantage of the opportunity to do so. Kryptonize is an effective and transparent method of securing communications between end-users, and can be deployed in a number of situations.

The system will work well when it is installed and running on a private server, for instance, for use by a small company or private user. In addition, due to the fact that the client application does a large portion of the message processing, along with the multithreaded nature of the server means that it is capable of handling large volumes of traffic and would also work well in a large scale deployment on a publicly available server on the Internet.

Because the Elgamal cryptosystem can only encrypt messages up to the size of the key, we currently have to split messages larger than 1024 bits into a series of 1024 bit blocks prior to encryption. As a result, in the future, we plan to implement a symmetric cryptosystem, such as Rijndael, to provide the enciphering and deciphering of the secure messages, and use Elgamal to provide security for our session communications and symmetric key passing.

In addition, we would like to explore applications for Kryptonize which will allow us to extend the functionality and availability of the system. We believe that Kryptonize may be ideal for securely communicating while using cloud computing. We plan to investigate the potential for creating a trusted publicly available key repository, allowing users of both private and public versions of Kryptonize to communicate with each other. We are also excited about the prospect of extending our system to facilitate real-time chat style communication between users.

In conclusion, we believe that there is currently both a need and desire among Internet users to be able to communicate securely with each other. As the details of people's day-to-day life becomes increasingly digitized, there is an imperative that data and communication needs to be protected in order to maintain individual privacy. This must be balanced, however, with the ability for users to access and use tools that allow them to accomplish this. A simple, widespread, and easy to use solution to this problem is needed, and we believe that the Kryptonize secure messaging system accomplishes this goal.

5. REFERENCES

Bishop, D. (2003). *Introduction to Cryptography with Java Applets.* Sudbury: Jones and Barlett Publishers.

Garey, M., & Johnson, D. S. (1979). *Computers and Intractibility: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company.

Mackey, D. (2003). *Web Security for Network and System Administrators.* Canada: Thomson Course Technology.

Mollin, R. (2007). *An Introduction to Cryptography.* Boca Raton: Chapman & Hall/CRC.

Nichols, R. K. (1999). *ICSA Guide to Cryptography.* McGraw-Hill.

Oracle Corporation. (2009). *BigInteger (Java Platform SE 6).* Retrieved May 1, 2010, from Java Platform SE 6 Documentation: http://java.sun.com/javase/6/docs/api/java/math/BigInteger.html

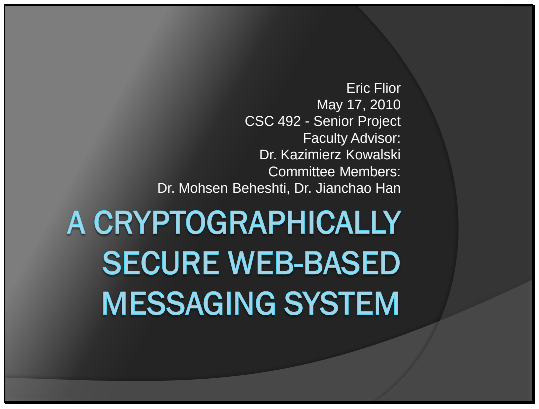
Oracle Corporation. (2010, January 12). *Primitive Data Types*. Retrieved May 1, 2010, from The Java Tutorials: http://java.sun.com/docs/books/tutorial/java/nutsandbolts/datatypes.html

Oracle Corporation. (2010). *Using MySQL with Java*. Retrieved May 01, 2010, from MySQL Developer Zone: http://dev.mysql.com/usingmysql/java/

Weaver, R. (2007). *Guide to Network Defense and Countermeasures.* Boston: Cengage Course Technology.

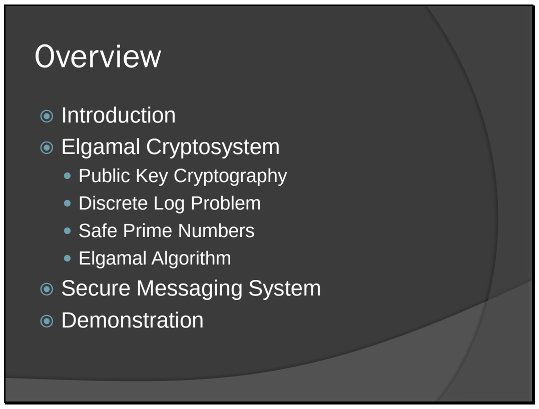
6. APPENDIX

6.1 Project Final Presentation



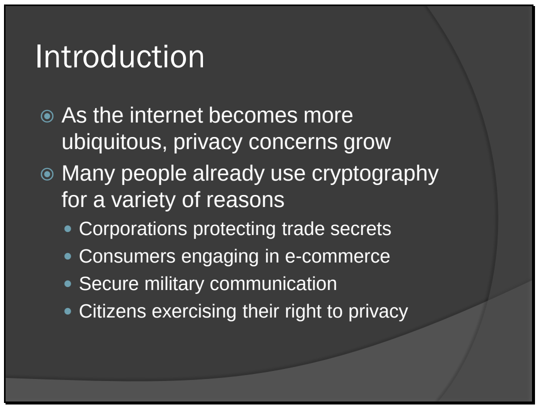
Eric Flior
May 17, 2010
CSC 492 - Senior Project
Faculty Advisor:
Dr. Kazimierz Kowalski
Committee Members:
Dr. Mohsen Beheshti, Dr. Jianchao Han

A CRYPTOGRAPHICALLY SECURE WEB-BASED MESSAGING SYSTEM



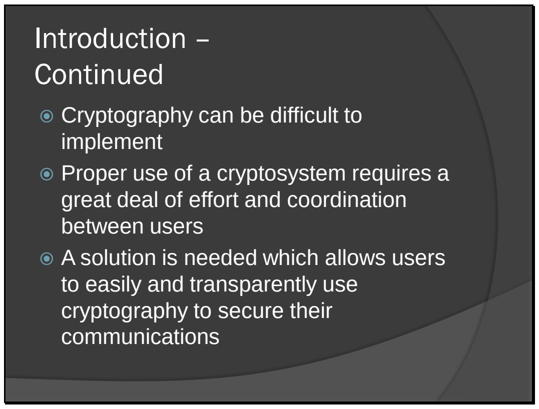
Overview

- Introduction
- Elgamal Cryptosystem
  - Public Key Cryptography
  - Discrete Log Problem
  - Safe Prime Numbers
  - Elgamal Algorithm
- Secure Messaging System
- Demonstration



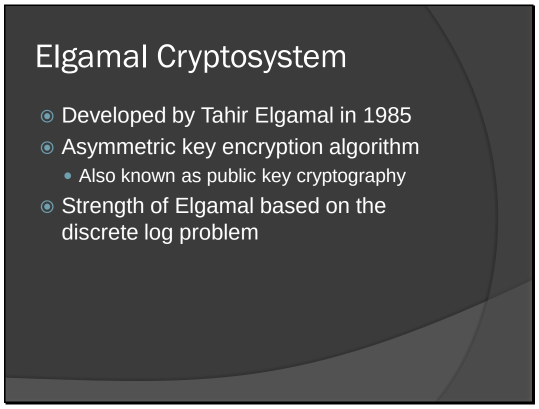
Introduction

- As the internet becomes more ubiquitous, privacy concerns grow
- Many people already use cryptography for a variety of reasons
  - Corporations protecting trade secrets
  - Consumers engaging in e-commerce
  - Secure military communication
  - Citizens exercising their right to privacy



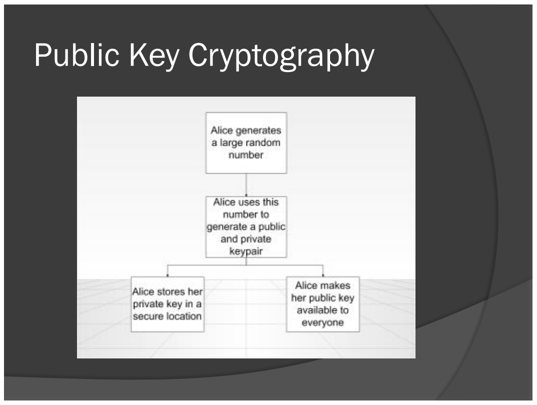
Introduction – Continued

- Cryptography can be difficult to implement
- Proper use of a cryptosystem requires a great deal of effort and coordination between users
- A solution is needed which allows users to easily and transparently use cryptography to secure their communications



Elgamal Cryptosystem

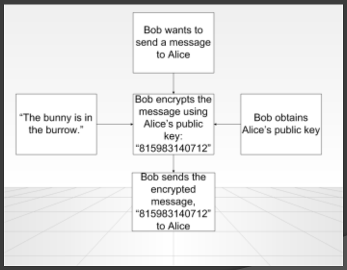
- Developed by Tahir Elgamal in 1985
- Asymmetric key encryption algorithm
  - Also known as public key cryptography
- Strength of Elgamal based on the discrete log problem



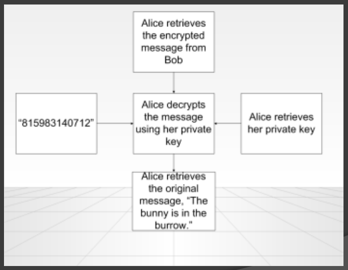
Public Key Cryptography

Alice generates a large random number
Alice uses this number to generate a public and private keypair
Alice stores her private key in a secure location
Alice makes her public key available to everyone

Public Key Cryptography


Bob wants to send a message to Alice
"The bunny is in the burrow."
Bob encrypts the message using Alice's public key: "815983140712"
Bob obtains Alice's public key
Bob sends the encrypted message, "815983140712" to Alice

Public Key Cryptography


Alice retrieves the encrypted message from Bob
"815983140712"
Alice decrypts the message using her private key
Alice retrieves her private key
Alice retrieves the original message, "The bunny is in the burrow."

Discrete Log Problem

- $log_b(a)=n$ if and only if $a=b^n$
- The log_b function solves the problem:
  - Given a base b and a power a of b, find an exponent n such that $a=b^n$
- The discrete log problem is the same, taken modulo p
  - Given a base $b \ (mod\ p)$ and $a \ (mod\ p)$, find an exponent n such that $a=b^n$

Discrete Log Problem

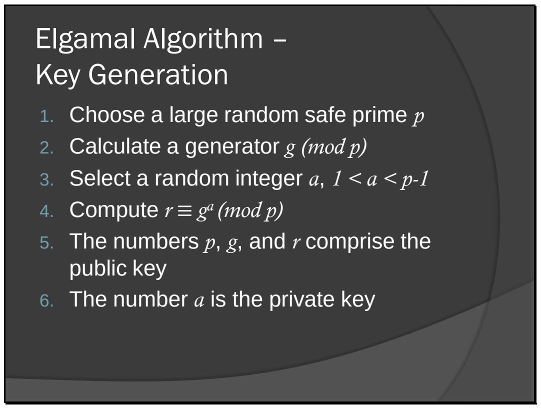
- Brute Force
  - $O(2^{(n/2)})$
- Baby-Step Giant Step
- Pollard's rho algorithm
- Pollard's lambda algorithm
- Pohlig-Hellman algorithm
- Index Calculus Algorithm
- Number Field Sieve
- Function Field Sieve

Safe Prime Numbers

- Most cryptosystems that depend on the difficulty of the discrete logarithm problem use generators
- A generator is:
  - an integer g, such that a prime p does not divide g, if $ord(g)\ p = p\ -1$
  - In other words, $p\ -1$ is the smallest integer such that $g^{p-1} \equiv 1\ (mod\ p)$, and $p/g \notin \mathbb{Z}$

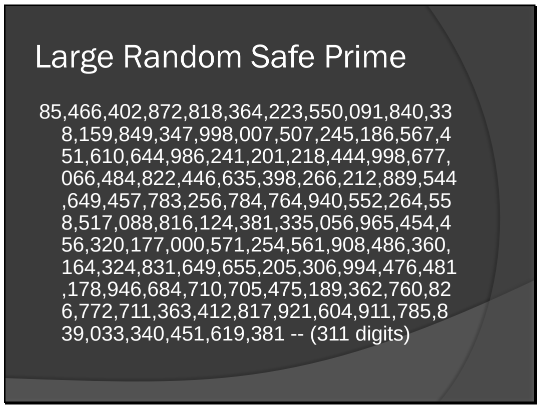
Safe Prime Numbers

- If $p\ -1$ is composed of only small factors, then p is susceptible to discrete log problem algorithms
- We must therefore choose prime numbers such that $p-1$ has at least one large factor
- Prime numbers with this property are considered safe primes



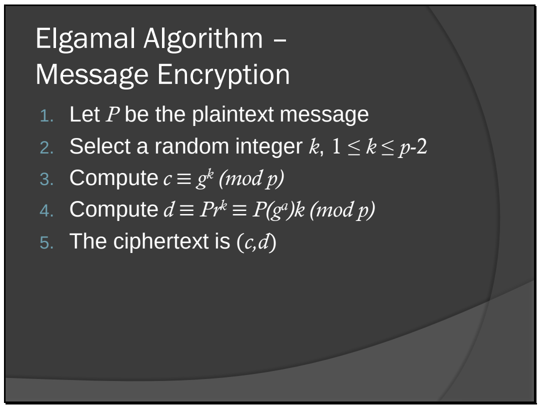
Elgamal Algorithm – Key Generation

1. Choose a large random safe prime p
2. Calculate a generator $g \ (mod\ p)$
3. Select a random integer a, $1 < a < p\text{-}1$
4. Compute $r \equiv g^a \ (mod\ p)$
5. The numbers p, g, and r comprise the public key
6. The number a is the private key



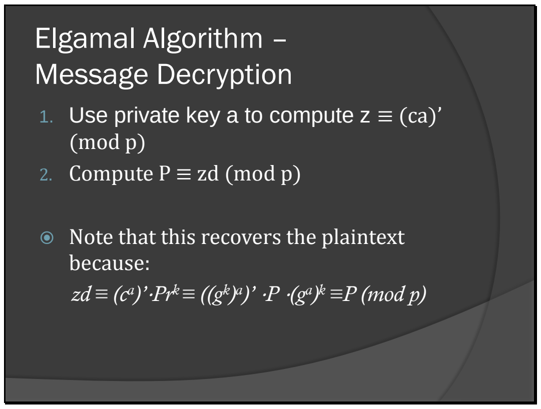
Large Random Safe Prime

85,466,402,872,818,364,223,550,091,840,33
8,159,849,347,998,007,507,245,186,567,4
51,610,644,986,241,201,218,444,998,677,
066,484,822,446,635,398,266,212,889,544
,649,457,783,256,784,764,940,552,264,55
8,517,088,816,124,381,335,056,965,454,4
56,320,177,000,571,254,561,908,486,360,
164,324,831,649,655,205,306,994,476,481
,178,946,684,710,705,475,189,362,760,82
6,772,711,363,412,817,921,604,911,785,8
39,033,340,451,619,381 -- (311 digits)



Elgamal Algorithm – Message Encryption

1. Let P be the plaintext message
2. Select a random integer k, $1 \le k \le p\text{-}2$
3. Compute $c \equiv g^k \ (mod\ p)$
4. Compute $d \equiv Pr^k \equiv P(g^a)k \ (mod\ p)$
5. The ciphertext is (c,d)

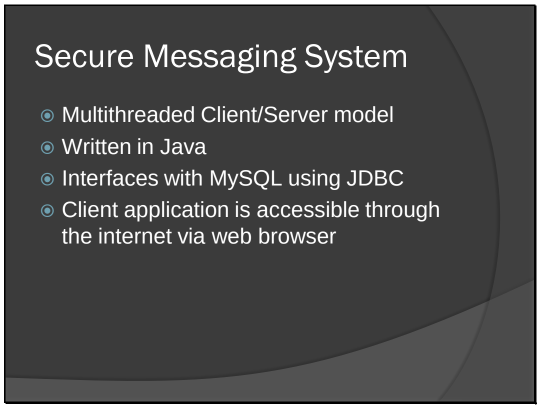


Elgamal Algorithm – Message Decryption

1. Use private key a to compute z ≡ (ca)’ (mod p)
2. Compute P ≡ zd (mod p)

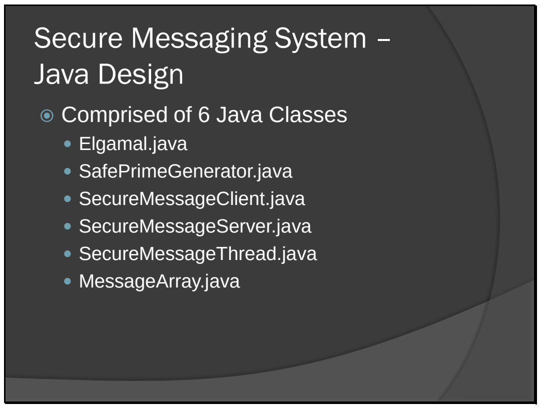
- Note that this recovers the plaintext because:

  $zd \equiv (c^a)'\cdot Pr^k \equiv ((g^k)^a)' \cdot P \cdot (g^a)^k \equiv P \ (mod\ p)$



Secure Messaging System

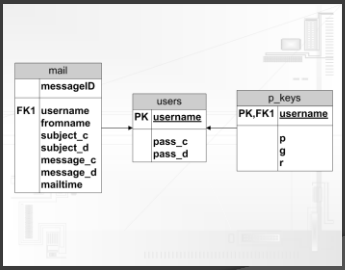
- Multithreaded Client/Server model
- Written in Java
- Interfaces with MySQL using JDBC
- Client application is accessible through the internet via web browser



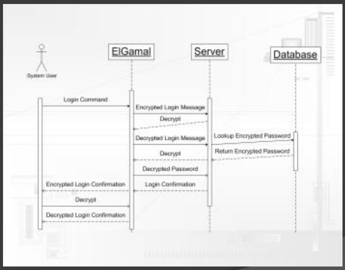
Secure Messaging System – Java Design

- Comprised of 6 Java Classes
  - Elgamal.java
  - SafePrimeGenerator.java
  - SecureMessageClient.java
  - SecureMessageServer.java
  - SecureMessageThread.java
  - MessageArray.java

Secure Messaging System – Database Schema



Secure Messaging System – Sample Use Case



Demonstration

6.2 SecurePrimeGenerator.java

```
package SecureMessaging;

import java.security.SecureRandom;
import java.math.BigInteger;
import java.util.Vector;

public class SecurePrimeGenerator {

    private int bitLength;
    private int certainty;
    private SecureRandom rnd;

    public SecurePrimeGenerator(int bitLength, int certainty) {
        this.bitLength = bitLength;
        this.certainty = certainty;
        rnd = new SecureRandom();
    }

    public BigInteger[] generateSafePrime() {
        // Based on (Bishop, 2003)
        // returns BigInteger[] safePrime where safePrime[0] is the prime
        // and safePrime[1] is the generator for the safe prime.
        BigInteger[] safePrime = new BigInteger[2];
        BigInteger t = new BigInteger(bitLength, certainty, rnd);
        BigInteger r = BigInteger.ONE;
        BigInteger p;
        do {
            p = r.multiply(t).multiply(BigInteger.valueOf((long)
2)).add(BigInteger.ONE);
            r = r.add(BigInteger.ONE);
        } while (!p.isProbablePrime(certainty));
        safePrime[0] = p;
        Vector<BigInteger> factorsVec = new Vector<BigInteger>();
        factorsVec.add(BigInteger.valueOf((long) 2));
        factorsVec.add(t);
        if (r.isProbablePrime(certainty)) {
            factorsVec.add(r);
        } else {
            BigInteger factor = BigInteger.valueOf((long) 3);
            BigInteger factor2 = factor.multiply(factor);
            while (factor2.compareTo(r) <= 0) {
                if (r.mod(factor).compareTo(BigInteger.ZERO) == 0) {
                    factorsVec.addElement(factor);
                }
                while (r.mod(factor).compareTo(BigInteger.ZERO) == 0) {
                    r = r.divide(factor);
                }
                factor = factor.add(BigInteger.ONE);
                factor2 = factor.multiply(factor);
            }
        }
        factorsVec.trimToSize();
        int factorsVecSize = factorsVec.size();
        BigInteger x = new BigInteger(p.bitLength() - 2, certainty, rnd);
        BigInteger z, y;
        boolean isGenerator;
        do {
            isGenerator = true;
            for (int i = 0; i < factorsVecSize; i++) {
                z = p.divide(factorsVec.get(i));
                y = x.modPow(z, p);
                if (y.compareTo(BigInteger.ONE) == 0) {
                    isGenerator = false;
                    break;
                }
            }
            if (isGenerator) {
                safePrime[1] = x;
            } else {
                x = new BigInteger(p.bitLength() - 2, certainty, rnd);
            }
        } while (!isGenerator);
        return safePrime;
    }
}
```

6.3 Elgamal.java

```
package SecureMessaging;

import java.math.BigInteger;
import java.security.SecureRandom;

public class ElGamal {

    private static SecurePrimeGenerator spg;
    private static SecureRandom sr = new SecureRandom();

    public static BigInteger[] GenerateKeys() {
        // returns a BigInteger[] keys:
        //
        // keys[0] is p
        // keys[1] is g
        // keys[2] is r
        // keys[3] is the private key, a
        spg = new SecurePrimeGenerator(1024, 10);
        BigInteger[] keys = new BigInteger[4];
        BigInteger[] safePrime = spg.generateSafePrime();
        keys[0] = safePrime[0];
        keys[1] = safePrime[1];
        BigInteger a = new BigInteger(safePrime[0].bitLength() - 2, sr);
        keys[3] = a;
        BigInteger r = safePrime[1].modPow(a, safePrime[0]);
        keys[2] = r;
        return keys;
    }

    public static BigInteger[] ElGamalEncipher(String plaintext, BigInteger
p, BigInteger g, BigInteger r) {
        // returns a BigInteger[] cipherText
        // cipherText[0] is c
        // cipherText[1] is d
        BigInteger[] cipherText = new BigInteger[2];
        BigInteger pText = new BigInteger(plaintext.getBytes());
        BigInteger k = new BigInteger(p.bitLength() - 2, sr);
        BigInteger c = g.modPow(k, p);
        BigInteger d = pText.multiply(r.modPow(k, p)).mod(p);
        cipherText[0] = c;
        cipherText[1] = d;
        return cipherText;
    }
    public static String ElGamalDecipher(BigInteger c, BigInteger d,
BigInteger a, BigInteger p) {
        //returns the plaintext enciphered as (c,d)
        BigInteger z = c.modPow(a, p).modInverse(p);
        BigInteger P = z.multiply(d).mod(p);
        byte[] plainTextArray = P.toByteArray();
        String output = null;
        try {
            output = new String(plainTextArray, "UTF8");
        } catch (Exception e) {
        }
        return output;
    }
}
```

6.4 MessageArray.java

```
package SecureMessaging;

import java.util.*;

public class MessageArray {

    private ArrayList<String[]> messageArray;
    int capacity = 10;
    public MessageArray(){
        messageArray = new ArrayList<String[]>();
    }

    public void AddMail(String[] newMessage) {
        if (messageArray.size() == capacity) {
            messageArray.ensureCapacity(capacity*2);
            capacity *= 2;
        }
        messageArray.add(newMessage);
    }

    public void DeleteMail(int messageNumber) {
        messageArray.remove(messageNumber);
    }

    public int GetSize() {
        return messageArray.size();
    }

    public String[] GetMail(int messageNumber) {
        return messageArray.get(messageNumber);
    }

    public int GetMessageNumber(int messageNumber) {
        String[] tempArray = messageArray.get(messageNumber);
        return Integer.parseInt(tempArray[0]);
    }
}
```

6.5 SecureMessageServer.java

```
package SecureMessaging;

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

public class SecureMessageServer {

    public static void main(String[] args) throws IOException {
        int backlog = 5;
        int port = 31338;
        Socket clientSocket;

        ServerSocket serverSocket = new ServerSocket(port, backlog);
        System.out.println("Server is listening at: LOCALHOST:31338");

        while(true) {
            clientSocket = serverSocket.accept();  // wait for a client
            Thread newThread = new SecureMessageThread(clientSocket);
            newThread.start();
        }
    }
}
```

6.6 SecureMessageThread.java

```
package SecureMessaging;

import java.net.*;
import java.io.*;
import java.sql.*;
import java.math.BigInteger;

public class SecureMessageThread extends Thread {

    private Socket clientSocket;
    private String fromClient, clientUserName;
    private String clientCipherText;
    private String[] clientCipherArray;
    private String[] command;
    private boolean connected = true;
    private BigInteger[] clientCipherArrayBI;
    private BigInteger serverP = new
BigInteger("5043949472997068537015530l3896" +

"84280477042304215811641505360353286967364198255023568384386734094O" +

"96620475360098340547978845551443379627987850672322183583603275O512" +

"56064402840568109160621306356783046519220580715395991539901100O034" +

"55252682094935566965755038130975718718489887989387570170497127441 5" +
            "14478399783476333");
    private BigInteger serverG = new BigInteger(
"21992770754050853445359770 3003" +

"4295591427563559222079728583447014940857777710245388849713731003 83" +

"638414367464245191449370361757076493997857350001863069289771154021" +
"55497831093655759134947008817034757081932271387048431981470714986" +

"255113622514268041310186649973342897548585372347791480265538749088" +
            "928678775311353929");
    private BigInteger serverR = new BigInteger(
"395829495611063339596417642930" +

"116687985783890496695957813019000735558728233291263189199316374164" +

"297960599021961482833246827217244492133944941558372494264084464852" +

"060934023375273551181446828561390852080197456090501729147733642867" +

"492810391722677240584807164852911123743422055571943067489962115889" +
            "05622393094065778");
    private BigInteger serverA = new BigInteger(
"883999618426692443397863190790" +

"683071104632948696809723397431886747062052417587003462580670246660" +

"750693677294292578233415186282800507096515012297886808067714204704" +

"318843553633763244669726886175740828966356566050798551786703600535" +

"950184698690993523758629991224646886865461665729613945647246604599" +
            "3196663106559783");
    private BigInteger clientP, clientG, clientR;
    ResultSet result;
    PreparedStatement statement;
    Statement stmt;
    Connection con;
    PrintStream out;
    InputStreamReader isr;
    BufferedReader in;

    public SecureMessageThread(Socket socket) {
        clientSocket = socket;
    }

    public void run() {
        try {
            out = new PrintStream(clientSocket.getOutputStream());
            isr = new InputStreamReader(clientSocket.getInputStream());
            in = new BufferedReader(isr);

            Class.forName("com.mysql.jdbc.Driver");
            // This needs to be changed based on your db info
            con = DriverManager.getConnection(
                          "jdbc:mysql://localhost:3306/secmail", "secmail",
      "blupaint");

            while (connected) {
                clientCipherText = new String();
                clientCipherArray = new String[2];
                command = new String[5];
                fromClient = new String();
                clientCipherText = in.readLine();
                clientCipherArray = clientCipherText.split("\\W");
                command = fromClient.split("\\W");

                if (command[0].equals("login")) {
                    Login(command);
                } else if (command[0].equals("register")) {
                    Register(command);
                } else if (command[0].equals("pubkey")) {
                    RegisterPKey(command);
                } else if (command[0].equals("getMessages")) {
                    GetMessages(command);
                } else if (command[0].equals("message")) {
                    SendMessage(command);
                } else if (command[0].equals("delete")) {
                    Delete(command);
                } else if (command[0].equals("logout")) {
                    break;
                } else {
                    break;
                }
            }
            clientSocket.close();
        } catch (IOException ioe) {
            // fail silently
        } catch (ClassNotFoundException cnfe) {
            // fail silently
        } catch (SQLException se) {
            // fail silently
        }
    }

    public void Login(String[] command) {
        try {
            // command[1] is username
            // command[2] is password
            statement = con.prepareStatement("SELECT count(*) FROM users " +
                    "WHERE username='" + command[1] + "'");
            result = statement.executeQuery();
            result.next();
            int isUser = result.getInt(1);
            if (isUser == 1) {
                // username found, verify password
                statement = con.prepareStatement("SELECT * FROM users " +
                        "WHERE username='" + command[1] + "'");
                result = statement.executeQuery();
                result.next();
                BigInteger storedPass_c = new BigInteger(
      result.getString(2));
                BigInteger storedPass_d = new BigInteger(
      result.getString(3));
                String storedPass = ElGamal.ElGamalDecipher(
                        storedPass_c, storedPass_d, serverA, serverP);
                if (storedPass.equals(command[2])) {
                    statement = con.prepareStatement("SELECT * FROM p_keys "
+
                            "WHERE username='" + command[1] + "'");
                    result = statement.executeQuery();
                    result.next();

                    clientP = new BigInteger(result.getString(2));
                    clientG = new BigInteger(result.getString(3));
                    clientR = new BigInteger(result.getString(4));
                    clientUserName = command[1];

                    out.println("1");
                    out.flush();
                    out.println(clientP.toString());
                    out.flush();
                    out.println(clientG.toString());
                    out.flush();
                    out.println(clientR.toString());
                    out.flush();
                } else {
                    out.println("2");
                    out.flush();
                }
            }
        } catch (SQLException se) {
// fail silently
        }
    }

    public void Register(String[] command) {
        try {
            // command[1] is username
            // command[2] is password
            statement = con.prepareStatement("SELECT count(*) FROM users " +
                    "WHERE username='" + command[1] + "'");
            result = statement.executeQuery();
            result.next();
            int isUser = result.getInt(1);
            if (isUser == 0) {
                // username not found, create new user
                // encrypt password with server's public key
                BigInteger[] encryptedPassword = ElGamal.ElGamalEncipher(
                        command[2], serverP, serverG, serverR);
                stmt =
                        con.createStatement();
                stmt.executeUpdate("INSERT INTO users (username, pass_c," +
                        " pass_d) VALUES ('" + command[1] + "', '" +
                        encryptedPassword[0].toString() + "', '" +
                        encryptedPassword[1].toString() + "')");
                out.println("1");
                clientUserName = command[1];
            } else {
                out.println("2");
                out.flush();
            }
        } catch (SQLException se) {
//              fail silently
        }

    }

    public void RegisterPKey(String[] command) {
        // command[1] is username
        // command[2] is p
        // command[3] is g
        // command[4] is r
        try {
            out.println("1");
            String publicKeys = in.readLine();
            String[] publicKeysArray = publicKeys.split("\\W");
            clientP = new BigInteger(publicKeysArray[0]);
            clientG = new BigInteger(publicKeysArray[1]);
            clientR = new BigInteger(publicKeysArray[2]);
            stmt = con.createStatement();
            stmt.executeUpdate("INSERT INTO p_keys (username, p, g, r) VALUES
('" + command[1] + "', '" + publicKeysArray[0] + "', '" + publicKeysArray[1]
+ "', '" + publicKeysArray[2] + "')");
        } catch (SQLException se) {
            System.out.println(se);
        } catch (IOException ioe) {
            System.out.println(ioe);
        }
    }

    public void SendMessage(String[] command) {
        try {
            statement = con.prepareStatement("SELECT count(*) FROM users " +
                    "WHERE username='" + command[1] + "'");
            result = statement.executeQuery();
            result.next();
            int isUser = result.getInt(1);
            if (isUser == 1) {
                out.println("1");
                out.flush();
                statement = con.prepareStatement("SELECT * FROM p_keys " +
                        "WHERE username='" + command[1] + "'");
                result = statement.executeQuery();
                result.next();
                out.println(result.getString(2) + " " + result.getString(3) +
                        " " + result.getString(4));
                out.flush();
                fromClient = in.readLine();
                String[] subject = fromClient.split(" ");
                fromClient = in.readLine();
                String[] message = fromClient.split(" ");
                stmt = con.createStatement();
                stmt.executeUpdate("INSERT INTO mail (`messageID`, " +
                        "`username`, `fromname`, `subject_c`, `subject_d`,
`message_c`, " + "`message_d`, `mailtime`) VALUES " +
"(NULL, '" + command[1] + "', '" + clientUserName + "', '" +
subject[0] + "', '" + subject[1] + "', '" + message[0] +
"', '" + message[1] + "', CURRENT_TIMESTAMP)");
                out.println("1");
                out.flush();
            } else {
                out.println("2");
                out.flush();
            }
        } catch (SQLException sqle) {
            // fail silently
        } catch (IOException ioe) {
            // fail silently
        }
    }

    public void GetMessages(String[] command) {
        try {
            statement = con.prepareStatement("SELECT count(*) FROM mail " +
                    "WHERE username='" + command[1] + "'");
            result = statement.executeQuery();
            result.next();
            int numMessages = result.getInt(1);
            out.println(numMessages);
            out.flush();
            if (numMessages != 0) {
                statement = con.prepareStatement("SELECT * FROM mail " +
                        "WHERE username='" + command[1] + "' ORDER BY
mailtime DESC");
                result = statement.executeQuery();
                while (result.next()) {
                    clientCipherArrayBI = ElGamal.ElGamalEncipher(
                            result.getString(1) + " " + result.getString(2) +
                            " " + result.getString(3) + " " +
                            result.getString(8), clientP, clientG, clientR);
                    out.println(
clientCipherArrayBI[0] + " " +  clientCipherArrayBI[1] + " " +
result.getString(4) +" " + result.getString(5) + " " + result.getString +
" " + result.getString(7));
                out.flush();
                }
            }
        } catch (SQLException sqle) {
            // fail silently
        }
    }

    public void Delete(String[] command) {
        try {
            stmt = con.createStatement();
            stmt.executeUpdate("DELETE FROM mail WHERE messageID = " +
                    command[1] + " LIMIT 1");
        } catch (SQLException sqle) {
        }

    }
}
```

6.7 SecureMailClient.java

```
package SecureMessaging;

import java.io.*;
import java.math.BigInteger;
import java.net.*;

public class SecureMessageClient extends javax.swing.JApplet {

    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {

                public void run() {
                    initComponents();
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

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

        directoryFileChooser = new javax.swing.JFileChooser();
        pKeyFileChooser = new javax.swing.JFileChooser();
        noDirectoryDialog = new javax.swing.JDialog();
        jLabel15 = new javax.swing.JLabel();
        noDirectoryOKButton = new javax.swing.JButton();
        writeSuccessDialog = new javax.swing.JDialog();
        jLabel16 = new javax.swing.JLabel();
        jLabel17 = new javax.swing.JLabel();
        writeSuccessButton = new javax.swing.JButton();
        noKeyfileDialog = new javax.swing.JDialog();
        jLabel18 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        ioErrorDialog = new javax.swing.JDialog();
```

```
jLabel19 = new javax.swing.JLabel();
jButton2 = new javax.swing.JButton();
loadSuccessDialog = new javax.swing.JDialog();
jLabel20 = new javax.swing.JLabel();
jButton3 = new javax.swing.JButton();
serverConnectionErrorDialog = new javax.swing.JDialog();
jLabel21 = new javax.swing.JLabel();
jButton4 = new javax.swing.JButton();
loginSuccessDialog = new javax.swing.JDialog();
jLabel22 = new javax.swing.JLabel();
jButton5 = new javax.swing.JButton();
registrationSuccessDialog = new javax.swing.JDialog();
jLabel23 = new javax.swing.JLabel();
jLabel24 = new javax.swing.JLabel();
jButton6 = new javax.swing.JButton();
registrationUserNameFailedDialog = new javax.swing.JDialog();
jLabel25 = new javax.swing.JLabel();
jButton7 = new javax.swing.JButton();
toUserNotFoundDialog = new javax.swing.JDialog();
jLabel26 = new javax.swing.JLabel();
jButton8 = new javax.swing.JButton();
emptyMessageDialog = new javax.swing.JDialog();
jLabel27 = new javax.swing.JLabel();
jButton9 = new javax.swing.JButton();
userPassFailureDialog = new javax.swing.JDialog();
jLabel28 = new javax.swing.JLabel();
jButton12 = new javax.swing.JButton();
jPanel1 = new javax.swing.JPanel();
loginPanel = new javax.swing.JPanel();
loginLabel1 = new javax.swing.JLabel();
loginLabel2 = new javax.swing.JLabel();
loginLabel3 = new javax.swing.JLabel();
loginTextField1 = new javax.swing.JTextField();
loginPasswordLabel1 = new javax.swing.JLabel();
loginSignOnButton = new javax.swing.JButton();
loginPasswordField1 = new javax.swing.JPasswordField();
jLabel5 = new javax.swing.JLabel();
loginErrorLabel2 = new javax.swing.JLabel();
jLabel6 = new javax.swing.JLabel();
loginTextField2 = new javax.swing.JTextField();
jLabel7 = new javax.swing.JLabel();
loginPasswordField2 = new javax.swing.JPasswordField();
loginRegisterButton = new javax.swing.JButton();
jSeparator1 = new javax.swing.JSeparator();
registrationPanel = new javax.swing.JPanel();
jLabel8 = new javax.swing.JLabel();
jLabel9 = new javax.swing.JLabel();
jLabel10 = new javax.swing.JLabel();
jLabel11 = new javax.swing.JLabel();
registrationTextField1 = new javax.swing.JTextField();
registrationBrowseButton = new javax.swing.JButton();
registrationGenerateButton = new javax.swing.JButton();
privateKeyPanel = new javax.swing.JPanel();
jLabel12 = new javax.swing.JLabel();
jLabel13 = new javax.swing.JLabel();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jLabel14 = new javax.swing.JLabel();
privateKeyTextField1 = new javax.swing.JTextField();
privateKeyBrowseButton1 = new javax.swing.JButton();
privateKeyfileLoadButton = new javax.swing.JButton();
messagePanel = new javax.swing.JPanel();
emailNewMessageButton = new javax.swing.JButton();
emailCheckMessagesButton = new javax.swing.JButton();
```

```
        emailReplyButton = new javax.swing.JButton();
        emailForwardButton = new javax.swing.JButton();
        emailDeleteButton = new javax.swing.JButton();
        jScrollPane2 = new javax.swing.JScrollPane();
        emailTextArea = new javax.swing.JTextArea();
        emailLogoutButton = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        mailTable1 = new javax.swing.JTable();
        newMessagePanel = new javax.swing.JPanel();
        emailNewMessageButton1 = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
        toTextField = new javax.swing.JTextField();
        jLabel4 = new javax.swing.JLabel();
        subjectTextField = new javax.swing.JTextField();
        jScrollPane3 = new javax.swing.JScrollPane();
        messageTextArea = new javax.swing.JTextArea();
        emailNewMessageButton2 = new javax.swing.JButton();
        jButton10 = new javax.swing.JButton();
        jButton11 = new javax.swing.JButton();

        directoryFileChooser.setCurrentDirectory(new
java.io.File("C:\\Program Files\\NetBeans 6.7.1\\null"));

directoryFileChooser.setFileSelectionMode(javax.swing.JFileChooser.DIRECTORIE
S_ONLY);

        pKeyFileChooser.setCurrentDirectory(new java.io.File("C:\\Program
Files\\NetBeans 6.7.1\\null"));
        pKeyFileChooser.setFileFilter(null);


noDirectoryDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOS
E_ON_CLOSE);
        noDirectoryDialog.setTitle("ERROR!");
        noDirectoryDialog.setModal(true);

        jLabel15.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel15.setText("Error: Please enter the directory in which you wish
to save your private key");

        noDirectoryOKButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        noDirectoryOKButton.setText("OK");
        noDirectoryOKButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                noDirectoryOKButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout noDirectoryDialogLayout = new
javax.swing.GroupLayout(noDirectoryDialog.getContentPane());

noDirectoryDialog.getContentPane().setLayout(noDirectoryDialogLayout);
        noDirectoryDialogLayout.setHorizontalGroup(

noDirectoryDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(noDirectoryDialogLayout.createSequentialGroup()

.addGroup(noDirectoryDialogLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)
                    .addGroup(noDirectoryDialogLayout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel15))
                    .addGroup(noDirectoryDialogLayout.createSequentialGroup()
                        .addGap(332, 332, 332)
                        .addComponent(noDirectoryOKButton)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        noDirectoryDialogLayout.setVerticalGroup(

noDirectoryDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(noDirectoryDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel15)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(noDirectoryOKButton)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );


writeSuccessDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPO
SE_ON_CLOSE);
        writeSuccessDialog.setTitle("Successful Key Generation");
        writeSuccessDialog.setModal(true);

        jLabel16.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel16.setText("Your private key has been successfully generated
and saved.");

        jLabel17.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel17.setText("Please make a secure backup of your sms.pkey file,
it cannot be recovered or regenerated. ");

        writeSuccessButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        writeSuccessButton.setText("OK");
        writeSuccessButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                writeSuccessButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout writeSuccessDialogLayout = new
javax.swing.GroupLayout(writeSuccessDialog.getContentPane());

writeSuccessDialog.getContentPane().setLayout(writeSuccessDialogLayout);
        writeSuccessDialogLayout.setHorizontalGroup(

writeSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(writeSuccessDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(writeSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.CENTER)
                    .addComponent(jLabel16)
                    .addComponent(jLabel17)
                    .addComponent(writeSuccessButton))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        writeSuccessDialogLayout.setVerticalGroup(
writeSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(writeSuccessDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel16)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel17)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 11,
Short.MAX_VALUE)
                .addComponent(writeSuccessButton)
                .addContainerGap())
        );


noKeyfileDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_
ON_CLOSE);
        noKeyfileDialog.setTitle("ERROR");
        noKeyfileDialog.setModal(true);

        jLabel18.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel18.setText("Please enter the path to your sms.pkey file");

        jButton1.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton1.setText("OK");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout noKeyfileDialogLayout = new
javax.swing.GroupLayout(noKeyfileDialog.getContentPane());
        noKeyfileDialog.getContentPane().setLayout(noKeyfileDialogLayout);
        noKeyfileDialogLayout.setHorizontalGroup(

noKeyfileDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(noKeyfileDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(noKeyfileDialogLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.CENTER)
                    .addComponent(jLabel18)
                    .addComponent(jButton1))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        noKeyfileDialogLayout.setVerticalGroup(

noKeyfileDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(noKeyfileDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel18)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton1)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
```

```
ioErrorDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON
_CLOSE);
        ioErrorDialog.setTitle("ERROR");
        ioErrorDialog.setModal(true);

        jLabel19.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel19.setText("I/O Error Detected!  Please check your path!");

        jButton2.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton2.setText("OK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout ioErrorDialogLayout = new
javax.swing.GroupLayout(ioErrorDialog.getContentPane());
        ioErrorDialog.getContentPane().setLayout(ioErrorDialogLayout);
        ioErrorDialogLayout.setHorizontalGroup(

ioErrorDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
            .addGroup(ioErrorDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(ioErrorDialogLayout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.CENTER)
                    .addComponent(jLabel19)
                    .addComponent(jButton2))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        ioErrorDialogLayout.setVerticalGroup(

ioErrorDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
            .addGroup(ioErrorDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel19)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton2)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );


loadSuccessDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOS
E_ON_CLOSE);
        loadSuccessDialog.setTitle("Private Key Loaded");
        loadSuccessDialog.setModal(true);

        jLabel20.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel20.setText("Private keyfile sms.pkey successfully loaded");

        jButton3.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton3.setText("OK");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });
        javax.swing.GroupLayout loadSuccessDialogLayout = new
javax.swing.GroupLayout(loadSuccessDialog.getContentPane());

loadSuccessDialog.getContentPane().setLayout(loadSuccessDialogLayout);
        loadSuccessDialogLayout.setHorizontalGroup(

loadSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(loadSuccessDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(loadSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.CENTER)
                    .addComponent(jLabel20)
                    .addComponent(jButton3))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        loadSuccessDialogLayout.setVerticalGroup(

loadSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(loadSuccessDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel20)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton3)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );


serverConnectionErrorDialog.setDefaultCloseOperation(javax.swing.WindowConsta
nts.DISPOSE_ON_CLOSE);
        serverConnectionErrorDialog.setTitle("ERROR");
        serverConnectionErrorDialog.setModal(true);

        jLabel21.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel21.setText("Error!  Could not connect to server!");

        jButton4.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton4.setText("OK");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout serverConnectionErrorDialogLayout = new
javax.swing.GroupLayout(serverConnectionErrorDialog.getContentPane());

serverConnectionErrorDialog.getContentPane().setLayout(serverConnectionErrorD
ialogLayout);
        serverConnectionErrorDialogLayout.setHorizontalGroup(

serverConnectionErrorDialogLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)

.addGroup(serverConnectionErrorDialogLayout.createSequentialGroup()
                .addContainerGap()
```

```
.addGroup(serverConnectionErrorDialogLayout.createParallelGroup(javax.swing.G
roupLayout.Alignment.CENTER)
                    .addComponent(jLabel21)
                    .addComponent(jButton4))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        serverConnectionErrorDialogLayout.setVerticalGroup(

serverConnectionErrorDialogLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)

.addGroup(serverConnectionErrorDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel21)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton4)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );


loginSuccessDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPO
SE_ON_CLOSE);
        loginSuccessDialog.setTitle("LOGIN SUCCESSFUL");
        loginSuccessDialog.setModal(true);

        jLabel22.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel22.setText("Successful login.  Please provide your sms.pkey
file.");

        jButton5.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton5.setText("OK");
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout loginSuccessDialogLayout = new
javax.swing.GroupLayout(loginSuccessDialog.getContentPane());

loginSuccessDialog.getContentPane().setLayout(loginSuccessDialogLayout);
        loginSuccessDialogLayout.setHorizontalGroup(

loginSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(loginSuccessDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(loginSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.CENTER)
                    .addComponent(jLabel22)
                    .addComponent(jButton5))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        loginSuccessDialogLayout.setVerticalGroup(

loginSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(loginSuccessDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel22)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton5)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );


registrationSuccessDialog.setDefaultCloseOperation(javax.swing.WindowConstant
s.DISPOSE_ON_CLOSE);
        registrationSuccessDialog.setTitle("Registration Succesful");
        registrationSuccessDialog.setModal(true);

        jLabel23.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel23.setText("You have successfully registered your username and
password.");

        jLabel24.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel24.setText("You will now be directed to generate your
public/private keypair.");

        jButton6.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton6.setText("OK");
        jButton6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton6ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout registrationSuccessDialogLayout = new
javax.swing.GroupLayout(registrationSuccessDialog.getContentPane());

registrationSuccessDialog.getContentPane().setLayout(registrationSuccessDialo
gLayout);
        registrationSuccessDialogLayout.setHorizontalGroup(

registrationSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
            .addGroup(registrationSuccessDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(registrationSuccessDialogLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.CENTER)
                    .addComponent(jLabel23)
                    .addComponent(jLabel24)
                    .addComponent(jButton6))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        registrationSuccessDialogLayout.setVerticalGroup(

registrationSuccessDialogLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
            .addGroup(registrationSuccessDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel23)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel24)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton6)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, 
Short.MAX_VALUE))
        );


registrationUserNameFailedDialog.setDefaultCloseOperation(javax.swing.WindowC
onstants.DISPOSE_ON_CLOSE);
        registrationUserNameFailedDialog.setTitle("ERROR");
        registrationUserNameFailedDialog.setModal(true);

        jLabel25.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel25.setText("ERROR: Username already exists.  Please choose a 
different username.");

        jButton7.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton7.setText("OK");
        jButton7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout registrationUserNameFailedDialogLayout = new 
javax.swing.GroupLayout(registrationUserNameFailedDialog.getContentPane());

registrationUserNameFailedDialog.getContentPane().setLayout(registrationUserN
ameFailedDialogLayout);
        registrationUserNameFailedDialogLayout.setHorizontalGroup(

registrationUserNameFailedDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(registrationUserNameFailedDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(registrationUserNameFailedDialogLayout.createParallelGroup(javax.sw
ing.GroupLayout.Alignment.CENTER)
                    .addComponent(jLabel25)
                    .addComponent(jButton7))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, 
Short.MAX_VALUE))
        );
        registrationUserNameFailedDialogLayout.setVerticalGroup(

registrationUserNameFailedDialogLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(registrationUserNameFailedDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel25)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton7)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, 
Short.MAX_VALUE))
        );


toUserNotFoundDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DIS
POSE_ON_CLOSE);
        toUserNotFoundDialog.setTitle("ERROR");
        toUserNotFoundDialog.setModal(true);

        jLabel26.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel26.setText("Username not found!  Please enter another 
username.");

        jButton8.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton8.setText("OK");
        jButton8.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton8ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout toUserNotFoundDialogLayout = new 
javax.swing.GroupLayout(toUserNotFoundDialog.getContentPane());

toUserNotFoundDialog.getContentPane().setLayout(toUserNotFoundDialogLayout);
        toUserNotFoundDialogLayout.setHorizontalGroup(

toUserNotFoundDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
            .addGroup(toUserNotFoundDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(toUserNotFoundDialogLayout.createParallelGroup(javax.swing.GroupLay
out.Alignment.CENTER)
                    .addComponent(jLabel26)
                    .addComponent(jButton8))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, 
Short.MAX_VALUE))
        );
        toUserNotFoundDialogLayout.setVerticalGroup(

toUserNotFoundDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
            .addGroup(toUserNotFoundDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel26)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton8)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, 
Short.MAX_VALUE))
        );


emptyMessageDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPO
SE_ON_CLOSE);
        emptyMessageDialog.setTitle("ERROR");
        emptyMessageDialog.setModal(true);

        jLabel27.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel27.setText("The username, subject, and message fields can not 
be empty.");

        jButton9.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton9.setText("OK");
        jButton9.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton9ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout emptyMessageDialogLayout = new 
javax.swing.GroupLayout(emptyMessageDialog.getContentPane());
```

```
emptyMessageDialog.getContentPane().setLayout(emptyMessageDialogLayout);
        emptyMessageDialogLayout.setHorizontalGroup(

emptyMessageDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(emptyMessageDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(emptyMessageDialogLayout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.CENTER)
                    .addComponent(jButton9)
                    .addComponent(jLabel27))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        emptyMessageDialogLayout.setVerticalGroup(

emptyMessageDialogLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(emptyMessageDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel27)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton9)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );


userPassFailureDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DI
SPOSE_ON_CLOSE);
        userPassFailureDialog.setTitle("ERROR");
        userPassFailureDialog.setModal(true);

        jLabel28.setFont(new java.awt.Font("Tahoma", 0, 22)); // NOI18N
        jLabel28.setText("Invalid Username / Password combination.  Please
try again.");

        jButton12.setFont(new java.awt.Font("Tahoma", 0, 22)); // NOI18N
        jButton12.setText("OK");
        jButton12.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton12ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout userPassFailureDialogLayout = new
javax.swing.GroupLayout(userPassFailureDialog.getContentPane());

userPassFailureDialog.getContentPane().setLayout(userPassFailureDialogLayout)
;
        userPassFailureDialogLayout.setHorizontalGroup(

userPassFailureDialogLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(userPassFailureDialogLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(userPassFailureDialogLayout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.CENTER)
                    .addComponent(jLabel28)
                    .addComponent(jButton12))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        userPassFailureDialogLayout.setVerticalGroup(

userPassFailureDialogLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
            .addGroup(userPassFailureDialogLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel28)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton12)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        jPanel1.setLayout(new java.awt.CardLayout());


loginPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.
border.BevelBorder.RAISED));
        loginPanel.setMinimumSize(new java.awt.Dimension(768, 576));
        loginPanel.setName("login"); // NOI18N
        loginPanel.setPreferredSize(new java.awt.Dimension(768, 576));

        loginLabel1.setFont(new java.awt.Font("Tahoma", 1, 22));

loginLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        loginLabel1.setText("Secure Messaging System");

loginLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        loginLabel2.setFont(new java.awt.Font("Tahoma", 1, 22));

loginLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        loginLabel2.setText("User Authentication");

loginLabel2.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        loginLabel3.setFont(new java.awt.Font("Tahoma", 0, 22));
        loginLabel3.setText("Login:");

        loginTextField1.setFont(new java.awt.Font("Tahoma", 0, 22));
        loginTextField1.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginTextField1ActionPerformed(evt);
            }
        });

        loginPasswordLabel1.setFont(new java.awt.Font("Tahoma", 0, 22));
        loginPasswordLabel1.setText("Password:");

        loginSignOnButton.setFont(new java.awt.Font("Tahoma", 0, 22)); //
NOI18N
        loginSignOnButton.setText("Sign On");
        loginSignOnButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginSignOnButtonActionPerformed(evt);
            }
        });
        loginPasswordField1.setFont(new java.awt.Font("Tahoma", 0, 22));
        loginPasswordField1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginPasswordField1ActionPerformed(evt);
            }
        });

        jLabel5.setFont(new java.awt.Font("Tahoma", 1, 22));
        jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel5.setText("Register a New User");
        jLabel5.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        loginErrorLabel2.setText("Password must be between 6-14 characters");

        jLabel6.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel6.setText("Desired Login:");

        loginTextField2.setFont(new java.awt.Font("Tahoma", 0, 22));
        loginTextField2.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginTextField2ActionPerformed(evt);
            }
        });

        jLabel7.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel7.setText("Password:");

        loginPasswordField2.setFont(new java.awt.Font("Tahoma", 0, 22));
        loginPasswordField2.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginPasswordField2ActionPerformed(evt);
            }
        });

        loginRegisterButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        loginRegisterButton.setText("Register");
        loginRegisterButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginRegisterButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout loginPanelLayout = new
javax.swing.GroupLayout(loginPanel);
        loginPanel.setLayout(loginPanelLayout);
        loginPanelLayout.setHorizontalGroup(

loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGroup(loginPanelLayout.createSequentialGroup()
                .addGap(123, 123, 123)

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
                    .addGroup(loginPanelLayout.createSequentialGroup()

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)

.addGroup(loginPanelLayout.createSequentialGroup()
                                .addGap(33, 33, 33)

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING)
                                    .addComponent(loginLabel3)
                                    .addComponent(loginPasswordLabel1)))

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING)
                                .addComponent(jLabel6)
                                .addComponent(jLabel7)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.CENTER)
                            .addComponent(loginLabel1)
                            .addComponent(loginLabel2)
                            .addComponent(loginTextField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 288, Short.MAX_VALUE)
                            .addComponent(loginPasswordField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 288, Short.MAX_VALUE)
                            .addComponent(loginTextField2,
javax.swing.GroupLayout.DEFAULT_SIZE, 288, Short.MAX_VALUE)
                            .addComponent(loginPasswordField2,
javax.swing.GroupLayout.DEFAULT_SIZE, 288, Short.MAX_VALUE))
                        .addGap(1, 1, 1))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
loginPanelLayout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 325,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(loginSignOnButton))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
loginPanelLayout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 321,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(loginRegisterButton))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
loginPanelLayout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 143,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel5)
                        .addGap(67, 67, 67))
                    .addGroup(loginPanelLayout.createSequentialGroup()
                        .addGap(146, 146, 146)
                        .addComponent(loginErrorLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 77,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(207, 207, 207))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
loginPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jSeparator1,
javax.swing.GroupLayout.DEFAULT_SIZE, 744, Short.MAX_VALUE)
                .addContainerGap())
        );
        loginPanelLayout.setVerticalGroup(

loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
```

```
            .addGroup(loginPanelLayout.createSequentialGroup()
                .addGap(36, 36, 36)
                .addComponent(loginLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(loginLabel2)
                .addGap(38, 38, 38)

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(loginTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(loginLabel3))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(loginPasswordField1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(loginPasswordLabel1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(loginSignOnButton)
                .addGap(18, 18, 18)
                .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(32, 32, 32)
                .addComponent(jLabel5)
                .addGap(18, 18, 18)
                .addComponent(loginErrorLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(loginTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel6))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(loginPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                    .addComponent(loginPasswordField2,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel7))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(loginRegisterButton)
                .addContainerGap(72, Short.MAX_VALUE))
        );

        jPanel1.add(loginPanel, "login");

        registrationPanel.setName("registration"); // NOI18N
        registrationPanel.setPreferredSize(new java.awt.Dimension(768, 576));

        jLabel8.setFont(new java.awt.Font("Tahoma", 1, 22));
        jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel8.setText("Secure Messaging System");
        jLabel8.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        jLabel9.setFont(new java.awt.Font("Tahoma", 1, 22));
        jLabel9.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel9.setText("New User Registration");
        jLabel9.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        jLabel10.setFont(new java.awt.Font("Tahoma", 1, 22));
        jLabel10.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel10.setText("Public/Private Keypair Generation");

jLabel10.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        jLabel11.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel11.setText("Choose a directory to save your private key
file:");

        registrationTextField1.setFont(new java.awt.Font("Tahoma", 0, 22));
        registrationTextField1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                registrationTextField1ActionPerformed(evt);
            }
        });

        registrationBrowseButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        registrationBrowseButton.setText("Browse");
        registrationBrowseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                registrationBrowseButtonActionPerformed(evt);
            }
        });

        registrationGenerateButton.setFont(new java.awt.Font("Tahoma", 0,
22));
        registrationGenerateButton.setText("Generate");
        registrationGenerateButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                registrationGenerateButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout registrationPanelLayout = new
javax.swing.GroupLayout(registrationPanel);
        registrationPanel.setLayout(registrationPanelLayout);
        registrationPanelLayout.setHorizontalGroup(

registrationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(registrationPanelLayout.createSequentialGroup()

.addGroup(registrationPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)
                    .addGroup(registrationPanelLayout.createSequentialGroup()
                        .addGap(195, 195, 195)

.addGroup(registrationPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.CENTER)
                            .addComponent(jLabel8)
                            .addComponent(jLabel9)
                            .addComponent(jLabel10)))
                    .addGroup(registrationPanelLayout.createSequentialGroup()
                        .addContainerGap()

.addGroup(registrationPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.TRAILING)
                            .addComponent(registrationGenerateButton)
                            .addComponent(registrationTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 632,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(registrationBrowseButton))
                    .addGroup(registrationPanelLayout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel11)))
                .addContainerGap(17, Short.MAX_VALUE))
        );
        registrationPanelLayout.setVerticalGroup(

registrationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(registrationPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel8)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel9)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel10)
                .addGap(135, 135, 135)
                .addComponent(jLabel11)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(registrationPanelLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.BASELINE)
                    .addComponent(registrationTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(registrationBrowseButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(registrationGenerateButton)
                .addContainerGap(223, Short.MAX_VALUE))
        );

        jPanel1.add(registrationPanel, "registration");

        privateKeyPanel.setName("privateKey"); // NOI18N

        jLabel12.setFont(new java.awt.Font("Tahoma", 1, 22));
        jLabel12.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel12.setText("Secure Messaging System");

jLabel12.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        jLabel13.setFont(new java.awt.Font("Tahoma", 1, 22));
        jLabel13.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel13.setText("Private Key Authorization");

jLabel13.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
```

```
        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel1.setText("Login/Password authentication successful.");

        jLabel2.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel2.setText("Please enter the path to your private key file");

        jLabel14.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel14.setText("Please enter the path to your sms.pkey file:");

        privateKeyTextField1.setFont(new java.awt.Font("Tahoma", 0, 22));
        privateKeyTextField1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                privateKeyTextField1ActionPerformed(evt);
            }
        });

        privateKeyBrowseButton1.setFont(new java.awt.Font("Tahoma", 0, 22));
        privateKeyBrowseButton1.setText("Browse");
        privateKeyBrowseButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                privateKeyBrowseButton1ActionPerformed(evt);
            }
        });

        privateKeyfileLoadButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        privateKeyfileLoadButton.setText("Load Keyfile");
        privateKeyfileLoadButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                privateKeyfileLoadButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout privateKeyPanelLayout = new
javax.swing.GroupLayout(privateKeyPanel);
        privateKeyPanel.setLayout(privateKeyPanelLayout);
        privateKeyPanelLayout.setHorizontalGroup(

privateKeyPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(privateKeyPanelLayout.createSequentialGroup()

.addGroup(privateKeyPanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
                    .addGroup(privateKeyPanelLayout.createSequentialGroup()
                        .addGap(165, 165, 165)

.addGroup(privateKeyPanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.CENTER)
                            .addComponent(jLabel2)
                            .addComponent(jLabel1)
                            .addComponent(jLabel13)
                            .addComponent(jLabel12)))
                    .addGroup(privateKeyPanelLayout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel14))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
privateKeyPanelLayout.createSequentialGroup()
                        .addContainerGap()

.addGroup(privateKeyPanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.TRAILING)
```

```
                            .addComponent(privateKeyfileLoadButton)
                            .addComponent(privateKeyTextField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 639, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(privateKeyBrowseButton1)))
                .addContainerGap())
        );
        privateKeyPanelLayout.setVerticalGroup(

privateKeyPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(privateKeyPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel12)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel13)
                .addGap(18, 18, 18)
                .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)
                .addGap(95, 95, 95)
                .addComponent(jLabel14)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(privateKeyPanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                    .addComponent(privateKeyTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(privateKeyBrowseButton1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(privateKeyfileLoadButton)
                .addContainerGap(223, Short.MAX_VALUE))
        );

        jPanel1.add(privateKeyPanel, "privateKey");

        messagePanel.setName("message"); // NOI18N

        emailNewMessageButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        emailNewMessageButton.setText("New Message");
        emailNewMessageButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emailNewMessageButtonActionPerformed(evt);
            }
        });

        emailCheckMessagesButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        emailCheckMessagesButton.setText("Check Messages");
        emailCheckMessagesButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emailCheckMessagesButtonActionPerformed(evt);
            }
        });

        emailReplyButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        emailReplyButton.setText("Reply");
        emailReplyButton.setEnabled(false);
        emailReplyButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emailReplyButtonActionPerformed(evt);
            }
        });

        emailForwardButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        emailForwardButton.setText("Forward");
        emailForwardButton.setEnabled(false);
        emailForwardButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emailForwardButtonActionPerformed(evt);
            }
        });

        emailDeleteButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        emailDeleteButton.setText("Delete");
        emailDeleteButton.setEnabled(false);
        emailDeleteButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emailDeleteButtonActionPerformed(evt);
            }
        });

        emailTextArea.setColumns(20);
        emailTextArea.setRows(5);
        emailTextArea.setName("message"); // NOI18N
        jScrollPane2.setViewportView(emailTextArea);

        emailLogoutButton.setFont(new java.awt.Font("Tahoma", 0, 22));
        emailLogoutButton.setText("Logout");
        emailLogoutButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emailLogoutButtonActionPerformed(evt);
            }
        });

        mailTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null},
                {null, null, null},
                {null, null, null},
                {null, null, null},
                {null, null, null},
                {null, null, null},
                {null, null, null}
            },
            new String [] {
                "Date:", "From:", "Subject:"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class,
java.lang.String.class
            };
            boolean[] canEdit = new boolean [] {
                false, false, false
            };
            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        mailTable1.getSelectionModel().addListSelectionListener(
            new javax.swing.event.ListSelectionListener() {
                public void valueChanged(javax.swing.event.ListSelectionEvent
evt) {
                    mailTable1ValueChanged(evt);
                }
            }
        );
        jScrollPane1.setViewportView(mailTable1);

        javax.swing.GroupLayout messagePanelLayout = new
javax.swing.GroupLayout(messagePanel);
        messagePanel.setLayout(messagePanelLayout);
        messagePanelLayout.setHorizontalGroup(

messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
            .addGroup(messagePanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 748, Short.MAX_VALUE)
                    .addGroup(messagePanelLayout.createSequentialGroup()
                        .addComponent(emailNewMessageButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(emailCheckMessagesButton))
                    .addGroup(messagePanelLayout.createSequentialGroup()
                        .addComponent(emailReplyButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(emailForwardButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(emailDeleteButton))
                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 748, Short.MAX_VALUE)
                    .addComponent(emailLogoutButton,
javax.swing.GroupLayout.Alignment.TRAILING))
                .addContainerGap())
        );
        messagePanelLayout.setVerticalGroup(

messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
            .addGroup(messagePanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE)
                    .addComponent(emailNewMessageButton)
                    .addComponent(emailCheckMessagesButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(messagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE)
                    .addComponent(emailReplyButton)
                    .addComponent(emailForwardButton)
                    .addComponent(emailDeleteButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 309,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(emailLogoutButton)
                .addContainerGap(15, Short.MAX_VALUE))
        );

        jPanel1.add(messagePanel, "message");

        newMessagePanel.setName("newMessage"); // NOI18N

        emailNewMessageButton1.setFont(new java.awt.Font("Tahoma", 0, 22));
        emailNewMessageButton1.setText("Send");
        emailNewMessageButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emailNewMessageButton1ActionPerformed(evt);
            }
        });

        jLabel3.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel3.setText("To:");

        toTextField.setFont(new java.awt.Font("Tahoma", 0, 22));

        jLabel4.setFont(new java.awt.Font("Tahoma", 0, 22));
        jLabel4.setText("Subject:");

        subjectTextField.setFont(new java.awt.Font("Tahoma", 0, 22));

        messageTextArea.setColumns(20);
        messageTextArea.setRows(5);
        jScrollPane3.setViewportView(messageTextArea);

        emailNewMessageButton2.setFont(new java.awt.Font("Tahoma", 0, 22));
        emailNewMessageButton2.setText("Send");
        emailNewMessageButton2.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                emailNewMessageButton2ActionPerformed(evt);
            }
        });

        jButton10.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton10.setText("Cancel");
        jButton10.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton10ActionPerformed(evt);
            }
        });
        jButton11.setFont(new java.awt.Font("Tahoma", 0, 22));
        jButton11.setText("Cancel");
        jButton11.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton11ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout newMessagePanelLayout = new
javax.swing.GroupLayout(newMessagePanel);
        newMessagePanel.setLayout(newMessagePanelLayout);
        newMessagePanelLayout.setHorizontalGroup(

newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(newMessagePanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
                    .addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 748, Short.MAX_VALUE)
                    .addGroup(newMessagePanelLayout.createSequentialGroup()
                        .addComponent(emailNewMessageButton1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jButton11))
                    .addGroup(newMessagePanelLayout.createSequentialGroup()

.addGroup(newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.TRAILING)
                            .addComponent(jLabel3)
                            .addComponent(jLabel4))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
                            .addComponent(toTextField,
javax.swing.GroupLayout.DEFAULT_SIZE, 665, Short.MAX_VALUE)
                            .addComponent(subjectTextField,
javax.swing.GroupLayout.DEFAULT_SIZE, 665, Short.MAX_VALUE)))
                    .addGroup(newMessagePanelLayout.createSequentialGroup()
                        .addComponent(emailNewMessageButton2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jButton10)))
                .addContainerGap())
        );
        newMessagePanelLayout.setVerticalGroup(

newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addGroup(newMessagePanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                    .addComponent(emailNewMessageButton1)
                    .addComponent(jButton11))
                .addGap(18, 18, 18)
.addGroup(newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                    .addComponent(toTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(subjectTextField,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 377, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(newMessagePanelLayout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                    .addComponent(emailNewMessageButton2)
                    .addComponent(jButton10))
                .addContainerGap())
        );

        jPanel1.add(newMessagePanel, "newMessage");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        );
    }// </editor-fold>
    private BigInteger[] myKeys = new BigInteger[4];
    BigInteger[] cipherText;
    private String userName;
    private String password;
    private Socket serverSocket;
    private PrintWriter toServer;
    private BufferedReader fromServer;
    private String stringFromServer;
    private BigInteger serverP = new BigInteger(

"504394947299706853701553013896842804770423042158116415053603532 8" +

"6967364198255023568384386734094096620475360098340547978845551443" +

"3796279878506723221835836032750512560644028405681091606213063567" +
"830465192205807153959915399011000034552526820949355669657550381 3" +
            "09757187184898879893875701704971274415144783997834763 33");
    private BigInteger serverG = new BigInteger(

"2199277075405085344535977030034295591427563559222079728583447014" +

"9408577771024538884971373100383638414367464245191449370361757 07" +

"6493997857350001863069289771154021554978310936557591349470088170" +

"3475708193227138704843198147071498625511362251426804131018664997" +
            "3342897548585372347791480265538749088928678775311353929");
    private BigInteger serverR = new
BigInteger("39582949561106333959641764 29" +

"3011668798578389049669595781301900073555872823329126318919931637" +

"4164297960599021961482833246827217244492133944941558372494264084" +

"4648520609340233752735511814468285613908520801974560905017291477" +

"3364286749281039172267724058480716485291112374342205557194306748" +
            "9962115889056223930940657 78");
    private MessageArray mArray;
    String eol = System.getProperty("line.separator");
    javax.swing.ListSelectionModel model;


    private void mailTable1ValueChanged(javax.swing.event.ListSelectionEvent
evt) {
        int selectedRow = mailTable1.getSelectedRow();
//        System.out.println(selectedRow);
        if (selectedRow >= 0) {
            String[] message = mArray.GetMail(selectedRow);
            emailTextArea.setText(message[6]);
            emailReplyButton.setEnabled(true);
            emailForwardButton.setEnabled(true);
            emailDeleteButton.setEnabled(true);
        }
    }

    private boolean ServerLogin() {
        boolean login = true;
        serverSocket = null;
        toServer = null;
        fromServer = null;
        try {
            serverSocket = new Socket("localhost", 31337);
            toServer = new PrintWriter(serverSocket.getOutputStream(), true);
            fromServer = new BufferedReader(new InputStreamReader(
                    serverSocket.getInputStream()));
        } catch (UnknownHostException e) {
            serverConnectionErrorDialog.setBounds(400, 0, 775, 125);
            serverConnectionErrorDialog.setVisible(true);
            login = false;
        } catch (IOException e) {
            serverConnectionErrorDialog.setBounds(400, 0, 775, 125);
            serverConnectionErrorDialog.setVisible(true);
            login = false;
        }
        return login;
    }
    private void LoadMessages() {
//        System.out.println("made it to mesasges");
        mArray = new MessageArray();
        cipherText = ElGamal.ElGamalEncipher(
                "getMessages " + userName, serverP, serverG, serverR);
        toServer.println(cipherText[0] + " " + cipherText[1]);
        String[] mailCipherArray;
        try {
            stringFromServer = fromServer.readLine();
            int numMessages = Integer.parseInt(stringFromServer);
            if (numMessages > 0) {
                for (int i = 0; i < numMessages; i++) {
                    stringFromServer = fromServer.readLine();
                    mailCipherArray = stringFromServer.split(" ");
                    String mail1 = ElGamal.ElGamalDecipher(new BigInteger(
                            mailCipherArray[0]), new BigInteger(
                            mailCipherArray[1]), myKeys[3], myKeys[0]);
                    String mail2 = ElGamal.ElGamalDecipher(new BigInteger(
                            mailCipherArray[2]), new BigInteger(
                            mailCipherArray[3]), myKeys[3], myKeys[0]);
                    String mail3 = ElGamal.ElGamalDecipher(new BigInteger(
                            mailCipherArray[4]), new BigInteger(
                            mailCipherArray[5]), myKeys[3], myKeys[0]);
                    String[] mail1Array = mail1.split(" ");
                    String[] newMessage = new String[7];
                    for (int j = 0; j < 5; j++) {
                        newMessage[j] = mail1Array[j];
                    }
                    newMessage[5] = mail2;
                    newMessage[6] = mail3;
                    mArray.AddMail(newMessage);
                }
            }
        } catch (IOException ioe) {
        }
    }

    private void PopulateMessageTable() {
        int mArraySize = mArray.GetSize();
        if (mArraySize > 0) {
            if (mArraySize > 4) {
                ((javax.swing.table.DefaultTableModel)
mailTable1.getModel()).setNumRows(mArraySize);
            } else {
                ((javax.swing.table.DefaultTableModel)
mailTable1.getModel()).setNumRows(5);
            }
            for (int i = 0; i <
                    mailTable1.getRowCount(); i++) {
                for (int j = 0; j <
                        mailTable1.getColumnCount(); j++) {
                    mailTable1.setValueAt("", i, j);
                }
            }
            for (int i = 0; i <
                    mArraySize; i++) {
                String[] currentMessage = mArray.GetMail(i);
                mailTable1.setValueAt(currentMessage[3] + " " +
currentMessage[4], i, 0);
                mailTable1.setValueAt(currentMessage[2], i, 1);
                mailTable1.setValueAt(currentMessage[5], i, 2);
            }
            model = mailTable1.getSelectionModel();
            model.clearSelection();
            emailTextArea.setText("");
        }
    }

    private void loginTextField1ActionPerformed(java.awt.event.ActionEvent
evt) {
        loginSignOnButtonActionPerformed(evt);
    }

    private void loginSignOnButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        // get username and password
        boolean nextCard = true;
        userName = loginTextField1.getText();
        char[] passwordArray = loginPasswordField1.getPassword();
        password = new String(passwordArray);
        if (!userName.isEmpty() && !password.isEmpty()) {
            nextCard = ServerLogin();
            if (nextCard) {
                cipherText = ElGamal.ElGamalEncipher(
                        "login " + userName + " " +
                        password, serverP, serverG, serverR);
                toServer.println(cipherText[0] + " " + cipherText[1]);
                toServer.flush();
                try {
                    stringFromServer = fromServer.readLine();
                } catch (IOException ioe) {
                    serverConnectionErrorDialog.setBounds(400, 0, 400, 125);
                    serverConnectionErrorDialog.setVisible(true);
                    nextCard = false;
                }

                if (stringFromServer.equals("1")) {
                    try {
                        myKeys[0] = new BigInteger(fromServer.readLine());
                        myKeys[1] = new BigInteger(fromServer.readLine());
                        myKeys[2] = new BigInteger(fromServer.readLine());
                    } catch (IOException ioe) {
                    }
                    loginSuccessDialog.setBounds(400, 0, 575, 125);
                    loginSuccessDialog.setVisible(true);
                    java.awt.CardLayout cl = (java.awt.CardLayout)
(jPanel1.getLayout());
                    cl.show(jPanel1, "privateKey");
                } else {
                    userPassFailureDialog.setBounds(400, 0, 650, 125);
                    userPassFailureDialog.setVisible(true);
                }
            }
        }
    }

    private void loginTextField2ActionPerformed(java.awt.event.ActionEvent
evt) {
        loginRegisterButtonActionPerformed(evt);
    }

    private void
loginRegisterButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        boolean nextCard = true;
        userName = loginTextField2.getText();
        char[] passwordArray = loginPasswordField2.getPassword();
        password = new String(passwordArray);
        if (!userName.isEmpty() && !password.isEmpty()) {
            nextCard = ServerLogin();
            if (nextCard) {
                cipherText = ElGamal.ElGamalEncipher("register " + userName +
" " + password, serverP, serverG, serverR);
                toServer.println(cipherText[0] + " " + cipherText[1]);
                toServer.flush();
                try {
                    stringFromServer = fromServer.readLine();
                } catch (IOException ioe) {
                }
                if (stringFromServer.equals("1")) {
                    registrationSuccessDialog.setBounds(400, 0, 680, 150);
                    registrationSuccessDialog.setVisible(true);
                    java.awt.CardLayout cl = (java.awt.CardLayout)
(jPanel1.getLayout());
                    cl.show(jPanel1, "registration");
                } else if (stringFromServer.equals("2")) {
                    registrationUserNameFailedDialog.setBounds(400, 0, 775,
125);
                    registrationUserNameFailedDialog.setVisible(true);
                }
            }
        }
    }

    private void
registrationTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        java.awt.CardLayout cl = (java.awt.CardLayout) (jPanel1.getLayout());
        cl.show(jPanel1, "message");
    }

    private void
registrationGenerateButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String directory = registrationTextField1.getText();
        PrintWriter fileWriter = null;
        boolean nextCard = true;
        if (!directory.isEmpty()) {
            myKeys = ElGamal.GenerateKeys();
            try {
                fileWriter = new PrintWriter(new
FileOutputStream(directory));
            } catch (FileNotFoundException fnfe) {
                ioErrorDialog.setBounds(400, 0, 775, 125);
                ioErrorDialog.setVisible(true);
                nextCard = false;
            }
            fileWriter.println(myKeys[3].toString());
            fileWriter.close();
            cipherText = null;
            String publicKey = myKeys[0].toString() + " " +
                    myKeys[1].toString() + " " + myKeys[2].toString();

            cipherText = ElGamal.ElGamalEncipher(
                    "pubkey " + userName, serverP, serverG, serverR);

            toServer.println(cipherText[0] + " " + cipherText[1]);
            toServer.flush();
            try {
                stringFromServer = fromServer.readLine();
            } catch (IOException ioe) {
            }
            if (stringFromServer.equals("1")) {
                toServer.println(publicKey);
                toServer.flush();
                writeSuccessDialog.setBounds(400, 0, 925, 175);
                writeSuccessDialog.setVisible(true);
                java.awt.CardLayout cl = (java.awt.CardLayout)
(jPanel1.getLayout());
                cl.show(jPanel1, "message");
            }
        } else {
            noDirectoryDialog.setBounds(400, 0, 775, 125);
            noDirectoryDialog.setVisible(true);
        }
    }

    private void
privateKeyTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        java.awt.CardLayout cl = (java.awt.CardLayout) (jPanel1.getLayout());
        cl.show(jPanel1, "message");
    }

    private void
privateKeyfileLoadButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String directory = privateKeyTextField1.getText();
        boolean nextCard = true;
        if (!directory.isEmpty()) {
            try {
                BufferedReader fileReader = new BufferedReader(new
FileReader(directory));
                String str;
                str = fileReader.readLine();
                myKeys[3] = new BigInteger(str);
            } catch (IOException ioe) {
                ioErrorDialog.setBounds(400, 0, 500, 125);
                ioErrorDialog.setVisible(true);
                nextCard = false;
            }
            if (nextCard) {
                LoadMessages();
                PopulateMessageTable();
                loadSuccessDialog.setBounds(400,0, 475, 125);
                loadSuccessDialog.setVisible(true);
                java.awt.CardLayout cl = (java.awt.CardLayout)
(jPanel1.getLayout());
                cl.show(jPanel1, "message");
            }
        } else {
            noKeyfileDialog.setBounds(400, 0, 475, 125);
            noKeyfileDialog.setVisible(true);
        }

    }

    private void
loginPasswordField1ActionPerformed(java.awt.event.ActionEvent evt) {
        loginSignOnButtonActionPerformed(evt);
    }

    private void
loginPasswordField2ActionPerformed(java.awt.event.ActionEvent evt) {
        loginRegisterButtonActionPerformed(evt);
    }

    private void
emailNewMessageButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
```

```
        toTextField.setText("");
        subjectTextField.setText("");
        messageTextArea.setText("");
        java.awt.CardLayout cl = (java.awt.CardLayout) (jPanel1.getLayout());
        cl.show(jPanel1, "newMessage");
    }

    private void emailReplyButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        int selectedRow = mailTable1.getSelectedRow();
        String[] replyMail = mArray.GetMail(selectedRow);
        toTextField.setText(replyMail[2]);
        subjectTextField.setText("Re: " + replyMail[5]);
        messageTextArea.setText(eol + replyMail[2] + " said: " + eol +
replyMail[6]);
        java.awt.CardLayout cl = (java.awt.CardLayout) (jPanel1.getLayout());
        cl.show(jPanel1, "newMessage");
    }

    private void emailForwardButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        int selectedRow = mailTable1.getSelectedRow();
        String[] replyMail = mArray.GetMail(selectedRow);
        toTextField.setText("");
        subjectTextField.setText("Re: " + replyMail[5]);
        messageTextArea.setText(eol + replyMail[2] + " said: " + eol +
replyMail[6]);
        java.awt.CardLayout cl = (java.awt.CardLayout) (jPanel1.getLayout());
        cl.show(jPanel1, "newMessage");
    }

    private void emailLogoutButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        cipherText = ElGamal.ElGamalEncipher("logout", serverP, serverG,
serverR);
        toServer.println(cipherText[0] + " " + cipherText[1]);
        toServer.flush();
        loginTextField1.setText("");
        loginTextField2.setText("");
        loginPasswordField1.setText("");
        loginPasswordField2.setText("");
        privateKeyTextField1.setText("");
        registrationTextField1.setText("");
        java.awt.CardLayout cl = (java.awt.CardLayout) (jPanel1.getLayout());
        cl.show(jPanel1, "login");
    }

    private void
emailNewMessageButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String to = toTextField.getText();
        String subject = subjectTextField.getText();
        String message = messageTextArea.getText();
        if (!to.isEmpty() && !subject.isEmpty() && !message.isEmpty()) {
            cipherText = ElGamal.ElGamalEncipher("message " + to, serverP,
serverG, serverR);
            toServer.println(cipherText[0] + " " + cipherText[1]);
            toServer.flush();
            try {
                stringFromServer = fromServer.readLine();
                if (stringFromServer.equals("1")) {
                    stringFromServer = fromServer.readLine();
                    String[] cText = stringFromServer.split(" ");
                    cipherText = ElGamal.ElGamalEncipher(subject, new
BigInteger(
                            cText[0]), new BigInteger(
                            cText[1]), new BigInteger(cText[2]));
                    toServer.println(cipherText[0] + " " + cipherText[1]);
                    toServer.flush();
                    cipherText = null;
                    cipherText = ElGamal.ElGamalEncipher(message, new
BigInteger
                            (cText[0]), new BigInteger(
                            cText[1]), new BigInteger(cText[2]));
                    toServer.println(cipherText[0] + " " + cipherText[1]);
                    toServer.flush();
                    stringFromServer = fromServer.readLine();
                    if (stringFromServer.equals("1")) {
                        LoadMessages();
                        PopulateMessageTable();
                        java.awt.CardLayout cl = (java.awt.CardLayout)
(jPanel1.getLayout());
                        cl.show(jPanel1, "message");
                    }
                } else {
                    toUserNotFoundDialog.setBounds(400, 0, 580, 125);
                    toUserNotFoundDialog.setVisible(true);
                }
            } catch (IOException ioe) {
            }
        } else {
            emptyMessageDialog.setBounds(400, 0, 650, 125);
            emptyMessageDialog.setVisible(true);
        }
    }

    private void
emailNewMessageButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        emailNewMessageButton1ActionPerformed(evt);
    }

    private void
registrationBrowseButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int result = directoryFileChooser.showDialog(jPanel1, "Select");
        if (result == directoryFileChooser.APPROVE_OPTION) {
            File file = directoryFileChooser.getSelectedFile();
            registrationTextField1.setText(file.getPath() + "\\sms.pkey");
        }
    }

    private void
privateKeyBrowseButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        int result = pKeyFileChooser.showDialog(jPanel1, "Select");
        if (result == pKeyFileChooser.APPROVE_OPTION) {
            File file = pKeyFileChooser.getSelectedFile();
            privateKeyTextField1.setText(file.toString());
        }
    }

    private void
noDirectoryOKButtonActionPerformed(java.awt.event.ActionEvent evt) {
        noDirectoryDialog.setVisible(false);
    }

    private void writeSuccessButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        writeSuccessDialog.setVisible(false);
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        noKeyfileDialog.setVisible(false);
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        ioErrorDialog.setVisible(false);
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        loadSuccessDialog.setVisible(false);
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        serverConnectionErrorDialog.setVisible(false);
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        loginSuccessDialog.setVisible(false);
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        registrationSuccessDialog.setVisible(false);
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
        registrationUserNameFailedDialog.setVisible(false);
    }

    private void
emailCheckMessagesButtonActionPerformed(java.awt.event.ActionEvent evt) {
        LoadMessages();
        PopulateMessageTable();
    }

    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
        toUserNotFoundDialog.setVisible(false);
    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
        emptyMessageDialog.setVisible(false);
    }

    private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {
        java.awt.CardLayout cl = (java.awt.CardLayout) (jPanel1.getLayout());
        cl.show(jPanel1, "message");
    }

    private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {
        java.awt.CardLayout cl = (java.awt.CardLayout) (jPanel1.getLayout());
        cl.show(jPanel1, "message");
    }

    private void emailDeleteButtonActionPerformed(java.awt.event.ActionEvent
evt) {
        int selectedRow = mailTable1.getSelectedRow();
        int toDelete = mArray.GetMessageNumber(selectedRow);
        mArray.DeleteMail(selectedRow);
        cipherText = ElGamal.ElGamalEncipher("delete " + toDelete, serverP,
serverG, serverR);
        toServer.println(cipherText[0] + " " + cipherText[1]);
        toServer.flush();
        LoadMessages();
        PopulateMessageTable();
    }
```

```
private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
    userPassFailureDialog.setVisible(false);
}

// Variables declaration - do not modify
private javax.swing.JFileChooser directoryFileChooser;
private javax.swing.JButton emailCheckMessagesButton;
private javax.swing.JButton emailDeleteButton;
private javax.swing.JButton emailForwardButton;
private javax.swing.JButton emailLogoutButton;
private javax.swing.JButton emailNewMessageButton;
private javax.swing.JButton emailNewMessageButton1;
private javax.swing.JButton emailNewMessageButton2;
private javax.swing.JButton emailReplyButton;
private javax.swing.JTextArea emailTextArea;
private javax.swing.JDialog emptyMessageDialog;
private javax.swing.JDialog ioErrorDialog;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton10;
private javax.swing.JButton jButton11;
private javax.swing.JButton jButton12;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton6;
private javax.swing.JButton jButton7;
private javax.swing.JButton jButton8;
private javax.swing.JButton jButton9;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel18;
private javax.swing.JLabel jLabel19;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel20;
private javax.swing.JLabel jLabel21;
private javax.swing.JLabel jLabel22;
private javax.swing.JLabel jLabel23;
private javax.swing.JLabel jLabel24;
private javax.swing.JLabel jLabel25;
private javax.swing.JLabel jLabel26;
private javax.swing.JLabel jLabel27;
private javax.swing.JLabel jLabel28;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JDialog loadSuccessDialog;
```

```
    private javax.swing.JLabel loginErrorLabel2;
    private javax.swing.JLabel loginLabel1;
    private javax.swing.JLabel loginLabel2;
    private javax.swing.JLabel loginLabel3;
    private javax.swing.JPanel loginPanel;
    private javax.swing.JPasswordField loginPasswordField1;
    private javax.swing.JPasswordField loginPasswordField2;
    private javax.swing.JLabel loginPasswordLabel1;
    private javax.swing.JButton loginRegisterButton;
    private javax.swing.JButton loginSignOnButton;
    private javax.swing.JDialog loginSuccessDialog;
    private javax.swing.JTextField loginTextField1;
    private javax.swing.JTextField loginTextField2;
    private javax.swing.JTable mailTable1;
    private javax.swing.JPanel messagePanel;
    private javax.swing.JTextArea messageTextArea;
    private javax.swing.JPanel newMessagePanel;
    private javax.swing.JDialog noDirectoryDialog;
    private javax.swing.JButton noDirectoryOKButton;
    private javax.swing.JDialog noKeyfileDialog;
    private javax.swing.JFileChooser pKeyFileChooser;
    private javax.swing.JButton privateKeyBrowseButton1;
    private javax.swing.JPanel privateKeyPanel;
    private javax.swing.JTextField privateKeyTextField1;
    private javax.swing.JButton privateKeyfileLoadButton;
    private javax.swing.JButton registrationBrowseButton;
    private javax.swing.JButton registrationGenerateButton;
    private javax.swing.JPanel registrationPanel;
    private javax.swing.JDialog registrationSuccessDialog;
    private javax.swing.JTextField registrationTextField1;
    private javax.swing.JDialog registrationUserNameFailedDialog;
    private javax.swing.JDialog serverConnectionErrorDialog;
    private javax.swing.JTextField subjectTextField;
    private javax.swing.JTextField toTextField;
    private javax.swing.JDialog toUserNotFoundDialog;
    private javax.swing.JDialog userPassFailureDialog;
    private javax.swing.JButton writeSuccessButton;
    private javax.swing.JDialog writeSuccessDialog;
    // End of variables declaration
}
```