**CSCI 5531.01 Advanced Operating Systems**

**Project Documentation**

**Programmer:** Sumanjali Tirunagaru

**Course:** CSCI 5531.01

**Date:** Apr 14, 2018

**Project:** A Distributed Message Delivery System using Encrypted Secure Channel

**Environment:** Java

**Files Included:** TirunagaruP3Sender.java, TirunagaruP3RelayServer.java, userlist.txt, TirunagaruP3Receiver.java, TirunagaruP3Interface.java,TirunagaruP3Tea.java, receiverslist.txt.

**Purpose:** TCP iterative client-server interaction using socket interface in Java programming language using a Cryptographic technique to provide a secure channel for communication.

**Input:** IP and Port Address of Relay Server and Receiver to create a socket and connection, series of strings to find the longest common substring and shared Secret key

**Preconditions:** we need a list of users and receivers, port address and IP address to connect to Relay Server and Receiver.

**Output:** Successful TCP connection establishment and secure communication between Sender, Relay Server and Receiver, and output from receiver to sender with a longest common substring.

**Algorithm:**

***Sender:***

* IP address and port number through command line arguments is required.
* Connect to the Relay server using the Input data.
* Create DataInputStream and DataOutputStream for the connection.
* After successful connection with a Success message with “Relay Server”, Send your username.
* Receive the Secret Key sent from Relay Server as a first Message in the channel and use the key to Decrypt and Encrypt every message before sending/reading data to/from Relay server
* If exists, and successful message
  + send password.
* Else send your username again.
* If password matched
  + Send the Receiver Name to connect with.
* If Password did not match
  + Retype the password.
* If Receiver Name exists, and success message from the “Relay server” about the connection of “Relay Server” with “Receiver”.
  + Send the strings to process to the “Relay Server”.
* Print the processed input from the “Relay server”.
* If wish to send the input again
  + Send the Strings
* Else
  + Quit the “Sender” and close the Streams of the connections and socket connection.

***Relay Server:***

* An input port number and secret key is required through command line argument.
* Load the “userlist.txt”, “receiverlist.txt” into a class object to authenticate users and receivers.
* Start the Server by opening the Relay Server socket using port number taken as input, for listening for the connections.
* After accepting the connection from “Sender”, create a new socket which serve the connection.
* Start a thread which serves, this connection and create DataInputStream and DataOutputStream for this connection.
* After connecting with “Sender” successfully, send a success message along with the Secret Key first Message in the channel, and use the same key to Decrypt and Encrypt every message before sending/reading data to/from Sender.
* Receive a username
* If username is present in the “userList.txt”
  + Send a message as User Name exists and ask for password.
* Else
  + Send a message for the correct User Name again
* If password received, matches with the username specified,
  + Send a success message and ask for the receiver name.
* Else
  + Send a message as password did not match and send a new password.
* If receiver name is received and does not present in the “receiverslist.txt”.
  + Ask for the correct receiver name
* Else
  + Get the IP address and port number from the “receiverlist.txt” using the receiver name as “Sender” specified.
  + Connect to the “Receiver” and create data streams for communication.
  + Send the Secret key sent to Receiver as a first Message in the channel and use the key to Decrypt and Encrypt every message before sending/reading data to/from Receiver.
  + If connection was unsuccessful connection and success message is not received by “Receiver”.
    - send the unsuccessful message to “Sender”.
  + Else
    - Send the success message to the “Sender” and ask for the Strings to be processed by “Receiver”.
    - The Strings sent by the “Sender” are to be transmitted to the receiver through the connection established between “Relay server and Receiver”.
    - Receive the data sent by the “Receiver” after processing from the “Receiver”.
    - Send the processed data to the “Sender”.
    - If Sender sends data again
      * Send strings to Receiver and communicate back.
    - Else if the message says to Quit or Exit.
      * Send the Quit Message to “Receiver” and close all the connections and data streams with “Sender and Receiver”.

***Relay Server:***

* An input port number is required through command line argument.
* Start the Server by opening the “Receiver” socket using port number taken as input, for listening for the connections.
* After accepting the connection from “Relay Server”, create a new socket which serve the connection.
* Start a thread which serves, this connection and create DataInputStream and DataOutputStream for this connection.
* After connecting with “Relay Sever” successfully, Receive the Secret Key sent from Relay Server as a first Message in the channel, and use the Key to Decrypt and Encrypt every message before sending/reading data to/from Relay server
* send a success message
* “Relay Server” sends the array of strings.
* Call a method defined earlier which process the given strings.
* If longest common substring exists.
  + returns the substring
* Else
  + Return the message with “No substrings”
* Send the return message, to the “Relay Server”.
* If the Message received from “Relay Server” contains “EXIT” or “QUIT”.
  + Close all the connections and data streams with “|Relay Server”.
* Else
  + Listen for more array strings to process.

***Tea Algorithm:***

A secret key of length 16 Byte is used to convert a text into cipher text is used and is initialised through the constructor. And if the key length is less than 16, an error should be thrown. The key is shifted to create a cipher for encryption and Decryption. It contains two methods Encrypt and Decrypt the text. And a constant key “0x9E3779B9” is also used to add to data while performing encryption and decryption. The Encryption operation like left shifting the bits by 4, XOR using key, adding with the constant key repeatedly on the data sequentially are done. And Decryption is performed in reverse way to change from Cipher to Original Text using the same Secret Key.

***Summary:***

In this project TCP iterative Client-Server interaction with a secure channel is being done using socket interface and symmetric Cryptographic algorithm TEA using Java. A simple and secure message relay system is developed.

The ***Sender*** process approves user-entered "messages" and sends the messages indirectly to ***Receiver*** through ***Relay Server*** process. The receiver process and performs some computation after the message is received by decrypting the data and sends back the data to ***Sender*** through ***Relay Sever*** by encrypting the data***.***

All the communications are being done using TCP sockets and through **cipher texts**. There are two application-level messages defined: a DATA message (containing the text of a message) and a CLOSE message, which tells the ***Relay Server*** and ***Receiver*** to shut down.

Symmetric Cryptographic algorithm TEA is used to provide encryption and decryption mechanisms to provide secure and protect the data.

There is a synchronization mechanism used to send and receive the messages. Used locks to achieve the synchronization.

Used Hash map to store the userlist.txt data and receiverlist.txt data for comparing. Hash map is a collection of key and value pair. So, it is easy to compare the data. The structure of the program is multi-threaded, and synchronization is done using locking.

There are three files named TirunagaruP3Sender.java, TirunagaruP3RelayServer.java TirunagaruP3Receiver.java, TirunagaruP3Interface.java, TirunagaruP3Tea.java

***Steps to run the program:***

* + - * open the command prompt, compile ***TirunagaruP3Tea*** program by typing
        + *java TirunagaruP3Tea*
      * open the command prompt, compile and run the ***Receiver*** program by typing
        + *java TirunagaruP3Receiver portnumber*
      * open the command prompt, compile and run the ***Relay*** ***Server*** program by typing
        + *java TirunagaruP3RelayServer portnumber secretKey*
      * open the command prompt, compile and run the ***Sender*** program by typing
        + *java TirunagaruP3Sender IPaddress portnumber*

***Classes Included in the project:***

* *TirunagaruP3Sender.java*
* *TirunagaruP3RelayServer.java*
* *SenderRelayServer.java*

Starts a thread which serves the connection and communication between ***Sender and Relay Server***

* *RecieverRelayServer.java*

Starts a thread which serves the connection and communication between ***Receiver and Relay Server.***

* *TirunagaruP3Tea.java*

Holds the operation to perform Encryption and Decryption.

* *NetworkDetails.java:*

Holds Receivers IP address and port address as an object paired with receiver name in the receiver list Hashmap.

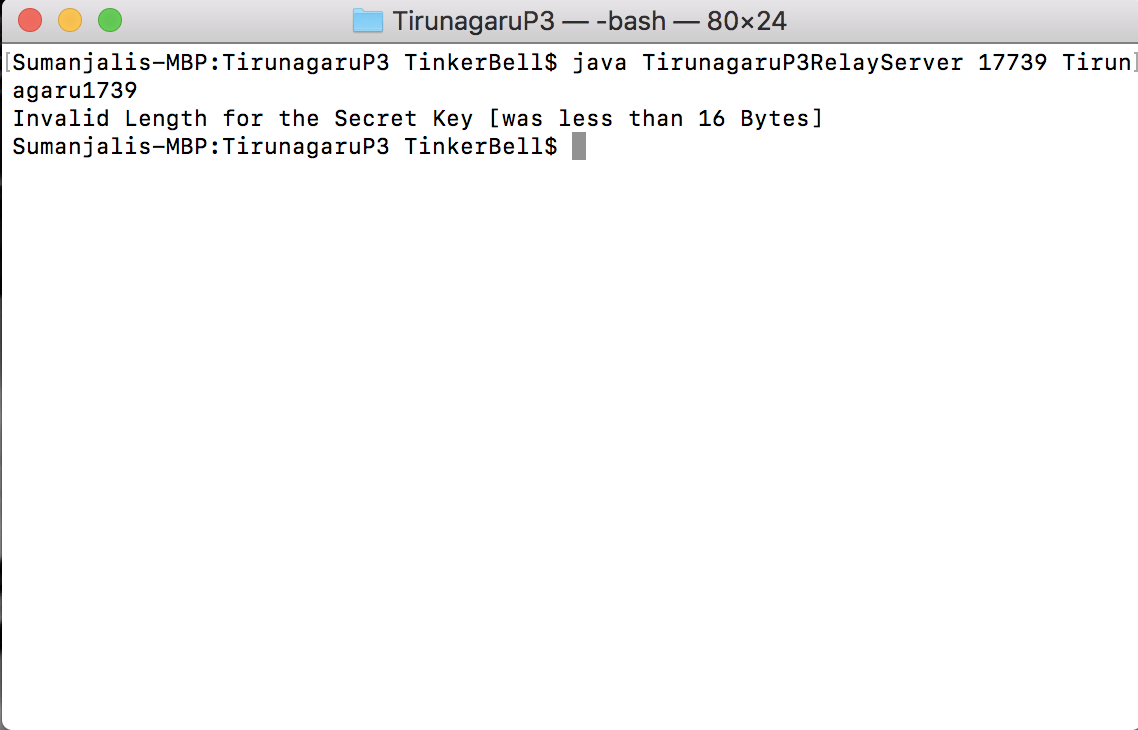
* *TirunagaruP3Receiver.java*
* *ReceiverHelper.java:*

Starts a thread which serves the connection and communication between ***Receiver and Relay Server***

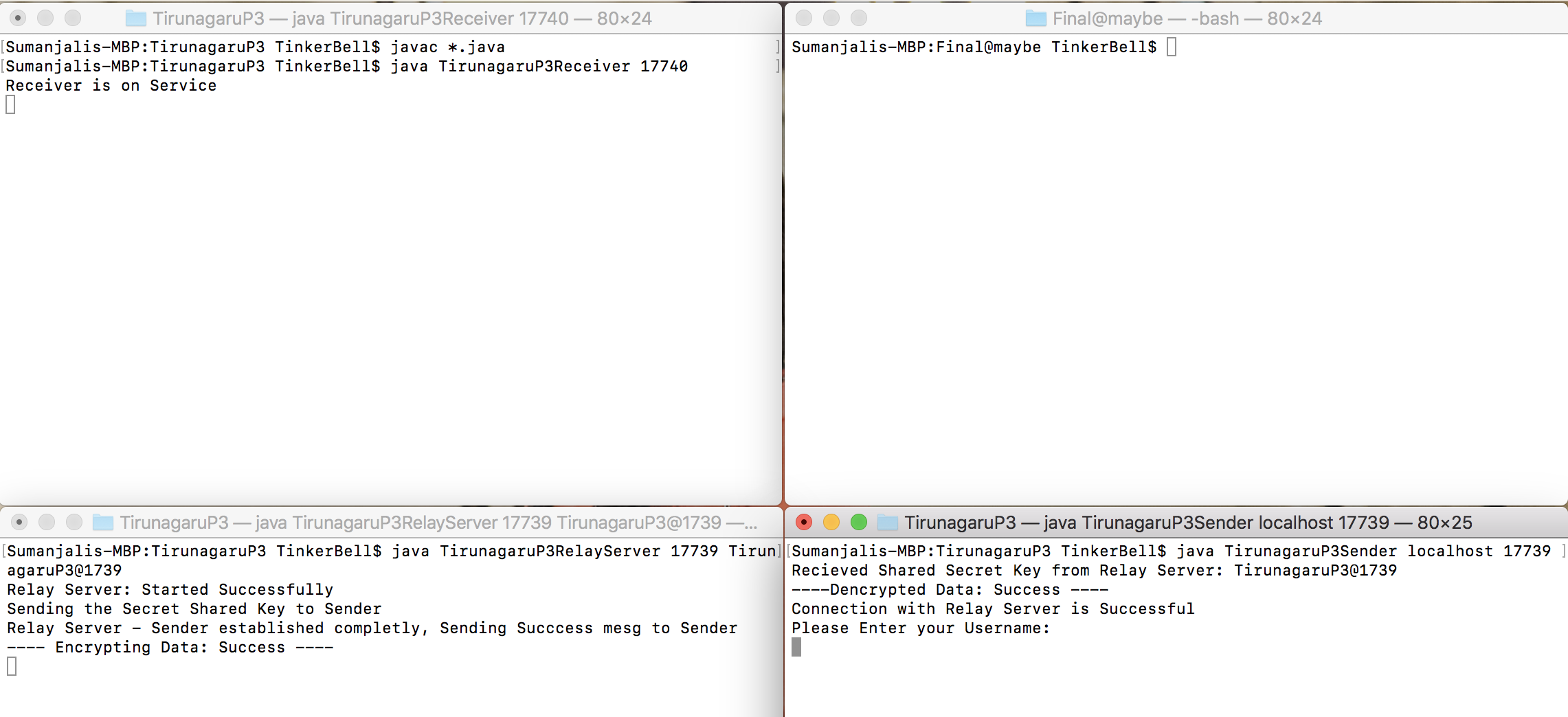
* *TirunagaruP3Interface.java:*

Contains an abstract method declaration “CommonSubstring (String [] array)” which is implemented by “ReceiverHelper.java” class to find longest common substring from a set of strings.

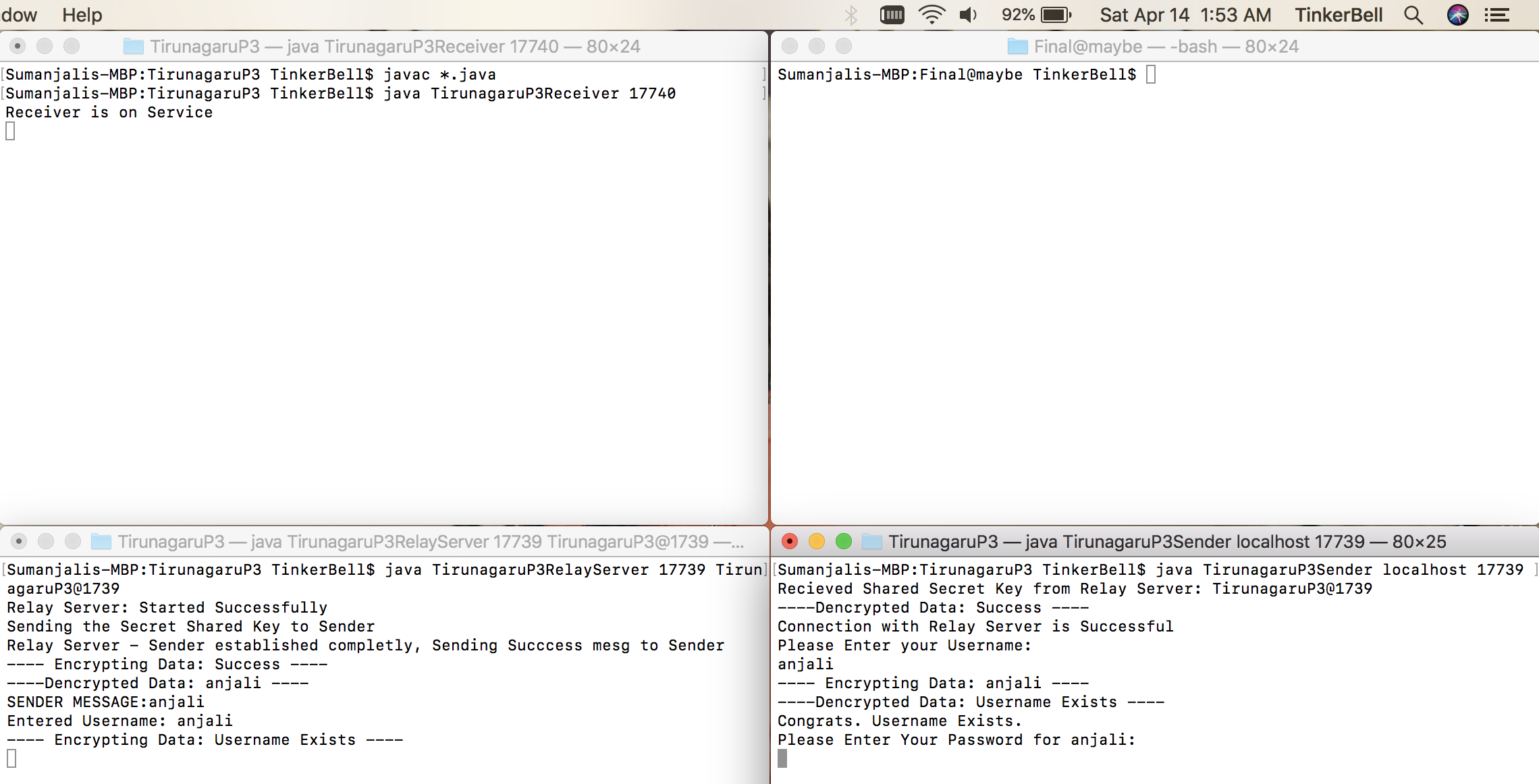
***Output Screenshots:***

******

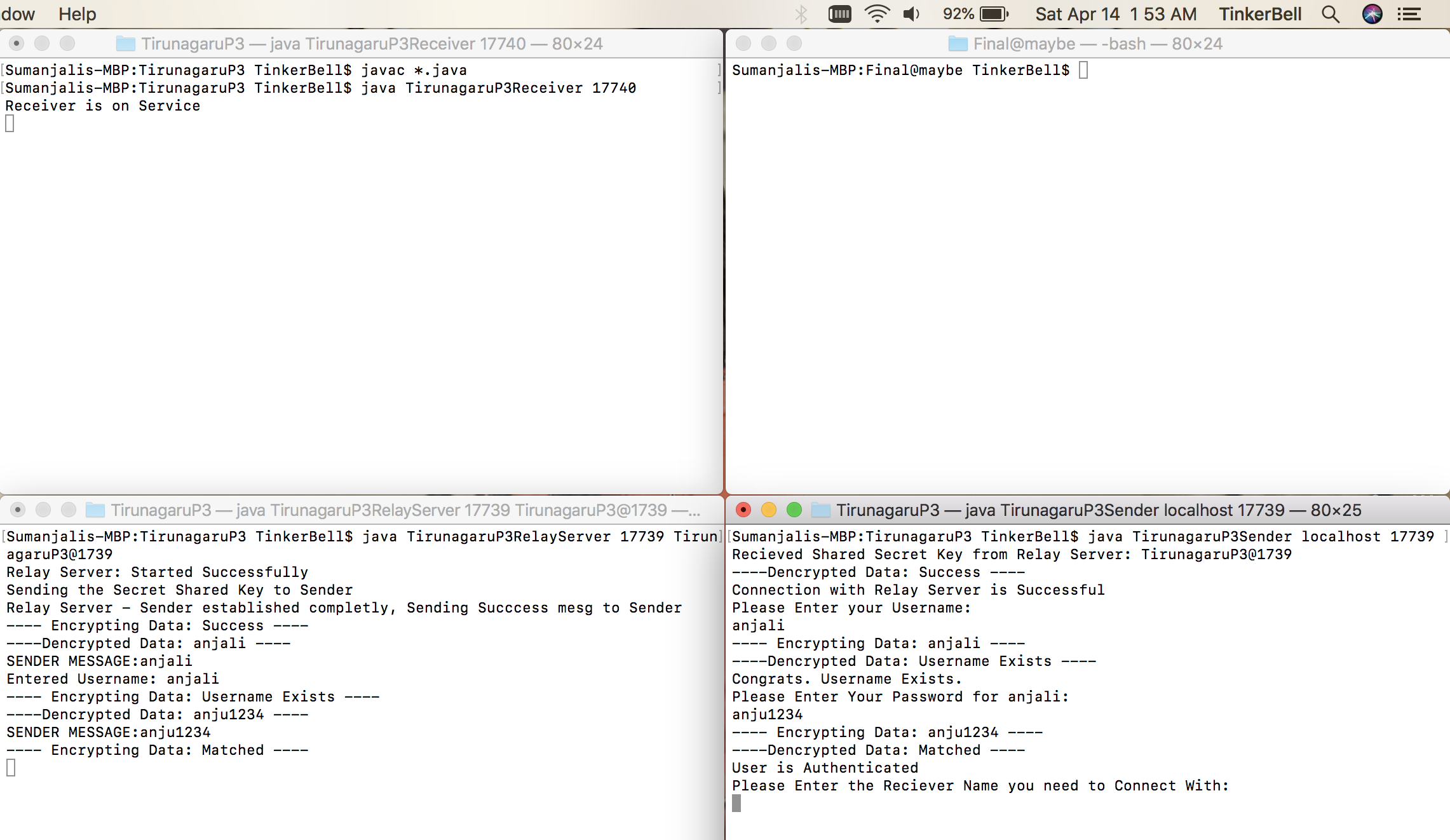
***fig 2. Invalid length for Secret Key***

****

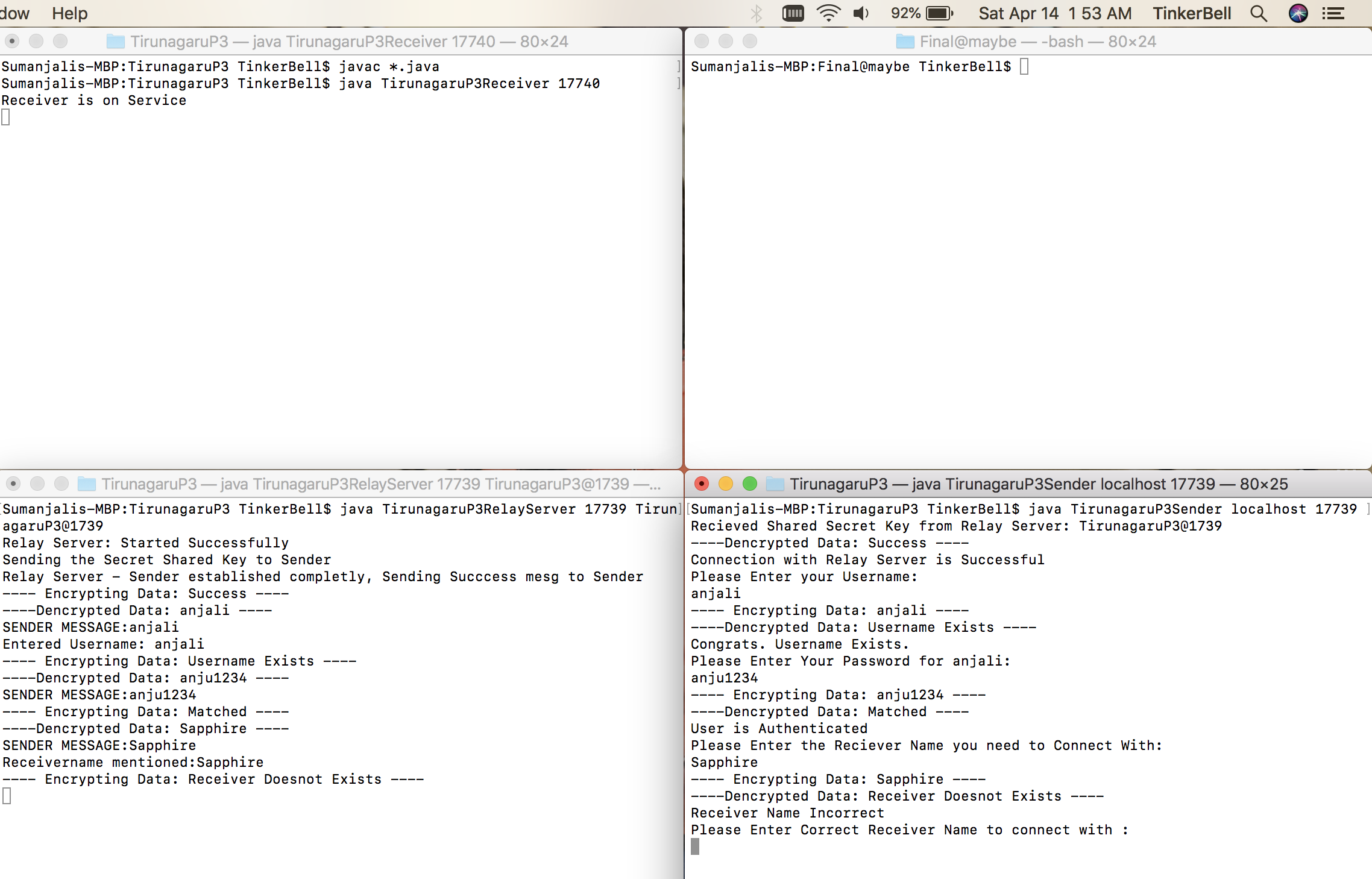
***fig 3. Relay Server sends Secret Key to Sender after Connecting to Relay Server***



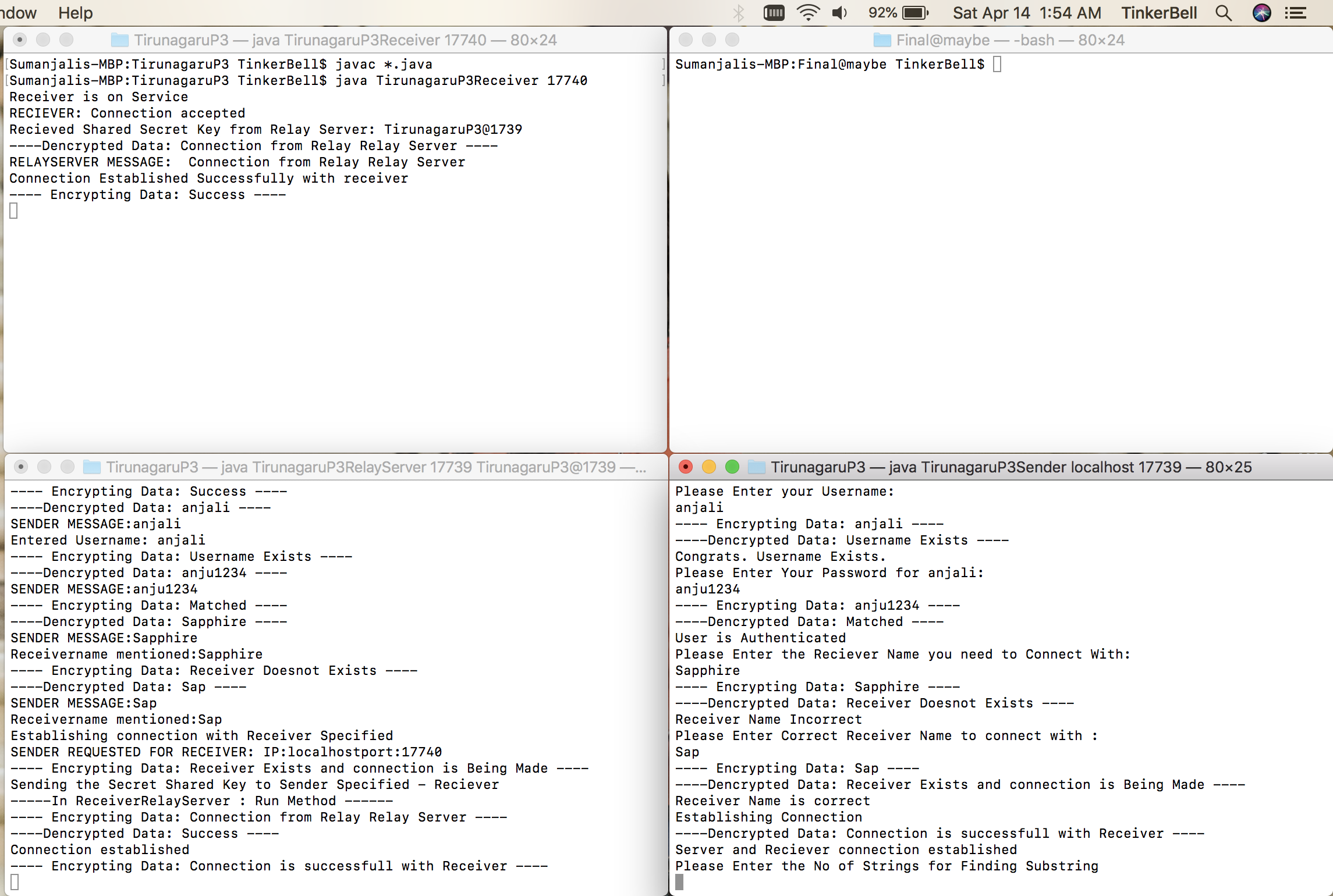
***fig 4. Sender sends username by encrypting to Relay Server, and server Decrypts the message***

******

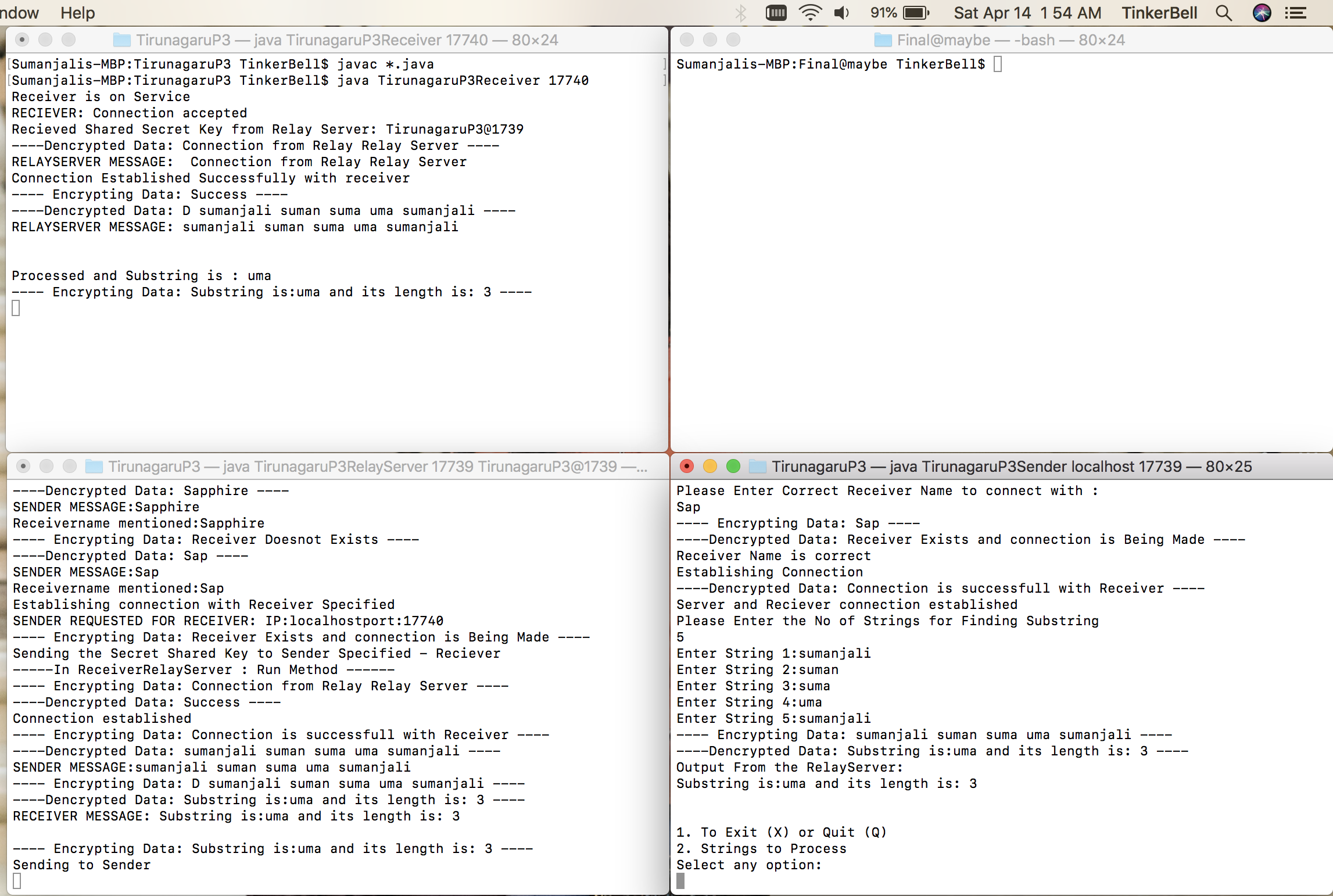
***fig 5. Sender sends password by encrypting to Relay Server, and server Decrypts the message***

******

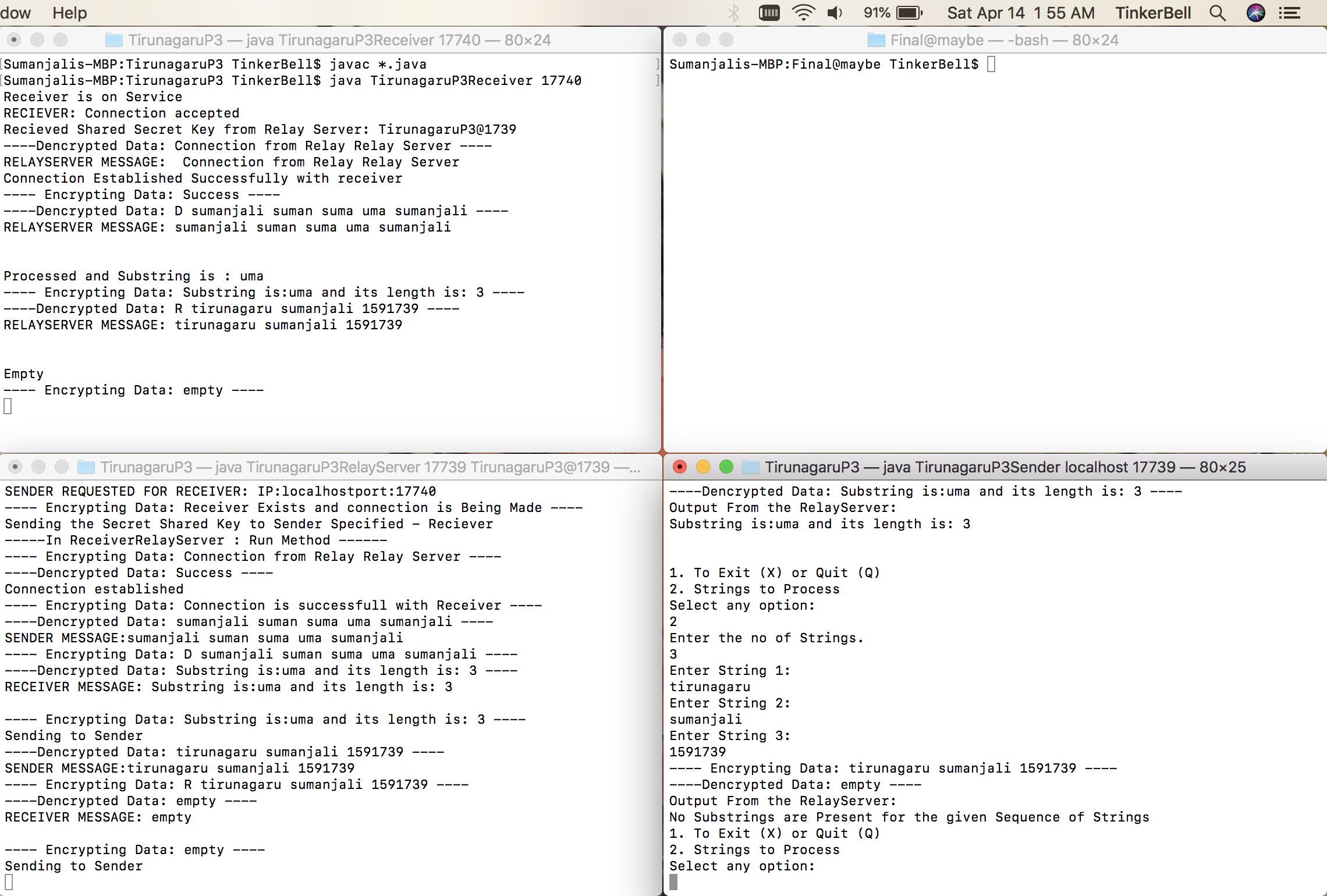
***fig 6. Sender sends Receiver name but not Present by encrypting to Relay Server, and server Decrypts the message***



***fig 7. Sender sends correct Receiver name but not Present by encrypting to Relay Server, and server Decrypts***



***fig 8. Sender sends Strings to Process Receiver by encrypting to Relay Server, and server, receiver Decrypts***



***fig 9. Sender sends Strings to Process Receiver by encrypting to Relay Server, and server, receiver Decrypts, but there are No Substrings***