A Message Passing Interface with Enhanced Security through TCP protocol

Ajitesh Praveen (15MIS1104)
Mansij Trivedi (15MIS1107)
M.Tech - Software Engineering (5Years)
School of Information Technology and Engineering
Vellore Institute of Technology, Chennai Campus

***Abstract*-**Encryption is process of turning a plaintext to cipher text or the method of changing confidential file to jargon in order prevent unauthorized persons to gain access to confidential message. Message is the transfer of information from the sender to the receiver through a particular medium. Encryption is the most effective process for achieving data security. The process of Encryption hides the contents of a message in a way that the original information is recovered only through a decryption process. To address this security issue, we developed a Message Passing Interface (MPI) implementation to preserve confidentiality of messages communicated among nodes of clusters in an unsecured network. This paper presents an Encryption/Decryption application of messages on python. The method of encryption of message in this paper is dual method 1.Rail fence encryption and 2.Onetime padding/mono alphabetic encryption where the same key that is used to encrypt is used to decrypt. The Encryption key is entered into the code text field by the admin. The same encryption key is also used to decrypt the encrypted binary file.

Index Terms- Encryption, Decryption, plain text, Rail fence, onetime padding, cipher key

I. INTRODUCTION

CRYPTOGRAPHY

Cryptography or cryptology (from "hidden, secret"; and graphene , "to write")is the practice and study of techniques for secure communication in the presence of third parties called adversaries. More generally, cryptography is about constructing and analzying protocols that prevent third parties or the public from reading private messages; various aspects information security such as data confidentiality, data integrity , authentication, and non-repudiation are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics , computer science, electrical engineering, communication science, and physics. Applications of cryptography include electronic commerce, chip-based payment cards, digital currencies, computer passwords, and military communications.

1. Rail Fence Cryptology

The rail fence cipher is a very simple, easy to crack cipher. It is a transposition cipher that follows a simple rule for mixing up the characters in the plaintext to form the cipher text. The rail fence cipher offers essentially no communication security, and it will be shown that it can be easily broken even by hand. Although weak on its own, it can be combined with other ciphers, such as a substitution cipher, the combination of which is more difficult to break than either cipher on it is own.

2. One-time Padding Cryptology

In cryptography, the one-time pad (OTP) is an encryption technique that cannot be cracked, but requires the use of a one-time pre-shared key the same size as, or longer than, the message being sent. In this technique, a plaintext is paired with a random secret key (also referred to as a one-time pad).

II. CRYTOGRAPHIC PRINCIPLES

A. Redundancy

Cryptographic principle 1: The first principle is that all encrypted messages must contain some redundancy, that is, information not needed to understand the message. Messages must contain some redundancy.

B. Freshness

Cryptographic principle 2: Some method is needed to foil replay attacks. One such measure is including in every message a timestamp valid only for, say, 10 seconds. The receiver can then just keep messages around for 10 seconds, to compare newly arrived messages to previous ones to filter out duplicates. Messages older than 10 seconds can be thrown out, since any replays sent more than 10 seconds later will be rejected as too old.

III. CRYPTOSYSTEM TYPES

In general cryptosystems are taxonomies into two classes, symmetric or asymmetric, depending only on whether the keys at the transmitter and receiver are easily computed from each other. In asymmetric cryptography algorithm a different key is used for encryption and decryption. In the symmetric encryption, Alice and Bob can share the same key (K), which is unknown to the attacker, and uses it to encrypt and decrypt their communications channel.

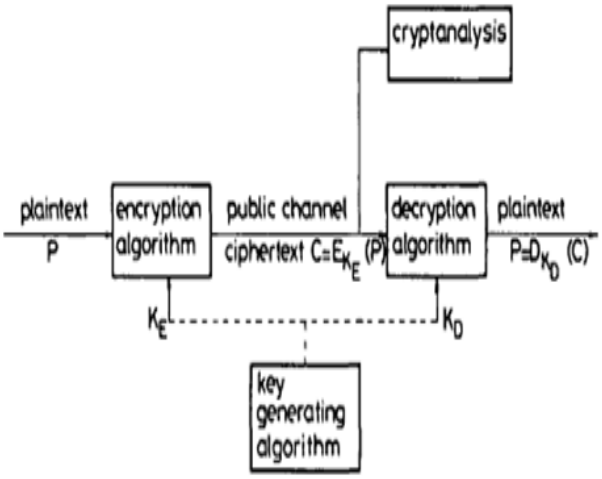


Fig. 1 General secrecy system

keys can be made public without compromising the other.

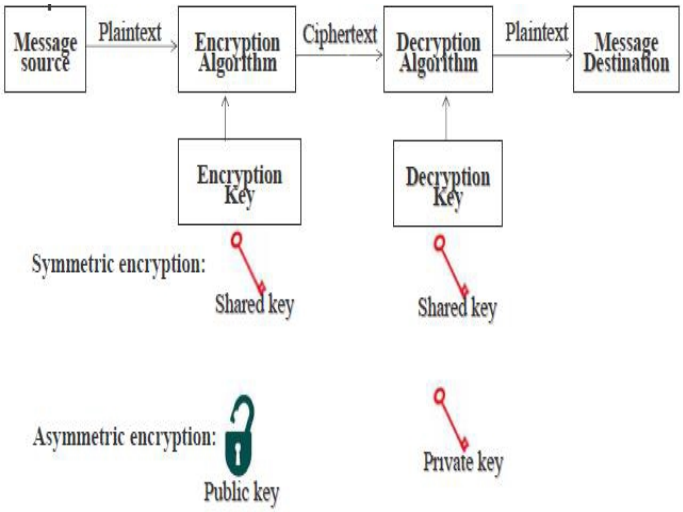
Multiple encryption is the process of encrypting an already encrypted message one or more times, either using the same or a different algorithm. It is also known as cascade encryption, cascade ciphering, multiple encryption and superencipherment. Superencryption refers to the outer-level encryption of a multiple encryption.

IV. CRYPTOGRAPHIC MODEL and ALGORITHM

A. *Encryption model*

There are two encryption models namely they are as follows:

Symmetric encryption and Asymmetric encryption. In Symmetric encryption, Encryption key = Decryption key. In Asymmetric encryption, Encryption key Decryption key.



V. IMPLEMENTATION

The following features have been implemented in the V-HUB:

Text Messaging with IC, transmitting or sending messages is faster, secure and can be done easily. The messages that are sent are encrypted and sent securely. If a message has been sent to a user who is not online, the user will receive the message as soon he logins into his account.

Starting a chat session:

1. Select the contact from the contact list.

2. Type the message in the text box.

3. Click on the “send” button or press Enter.

Message history

The chat sessions are stored locally on the client computer. The user can easily review the messages that are exchanged by using the chat history.

Offline messages

Messages can be sent to even to users who are offline. The user will receive the message as soon as he/she logins into their account.

VI. TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP/IP, or the Transmission Control Protocol/Internet Protocol, is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP can also be used as a communications protocol in a private network (an intranet or an extranet).

The two main protocols in the internet protocol suite serve specific functions. TCP defines how applications can create channels of communication across a network. It also manages how a message is assembled into smaller packets before they are then transmitted over the internet and reassembled in the right order at the destination address.

Socket Programming

In socket programming, there is a buffer containing data sent and received through the TCP socket channel. It demonstrates the encryption and decryption process in ESMPICH2. ES-MPICH2 encrypt the plaintext in the buffer of the sending node then decrypt it on the receiving node because the plaintext and cipher text are identical in length in block cipher algorithms, the sizes of the buffers in both the sending and receiving nodes remain unchanged after the encryption and decryption processes.

VII. FUTURE ENHANCEMENTS

V-HUB can be further implemented in the area of voice and video calling and can be used in the large organizations by hosting the application. IC can by further implemented by providing security algorithms to encrypt the data transfer without the size restrictions for a larger scale.

VIII. CONCLUSION

It has dealt with implementing a chatting application on small scale using latest technology in web development. It has features such as text messaging, group chat, data transfer. The main objective of IC was to develop an application which provides the data transfers without the size restrictions and hence has been implemented. We have studied various cryptographic techniques to increase the security of network. Cryptography, together with suitable communication protocols, can provide a high degree of protection in digital communications .

REFERENCES

[1] DENNING, D., and DENNING, P.J.: 'Data security', *ACM Comput. Surveys,* 1979, 11, pp. 227-250

[2] A Role-Based Trusted Network Provides Pervasive Security and Compliance - interview with Jayshree Ullal, senior VP of Cisco.

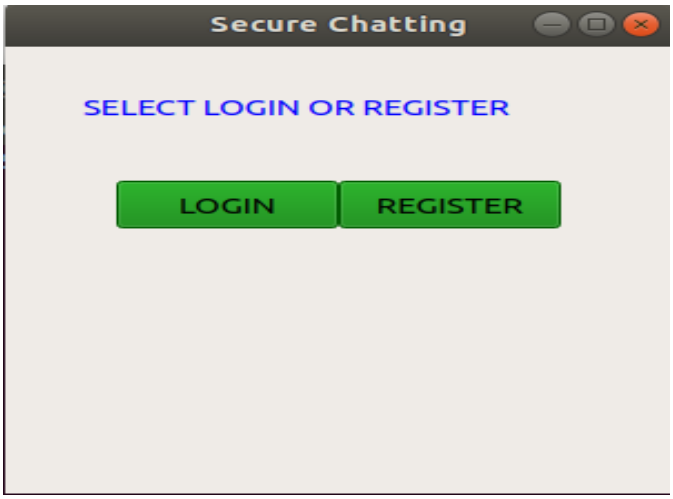
[3] Dave Dittrich, Network monitoring/Intrusion Detection Systems (IDS), University of Washington.

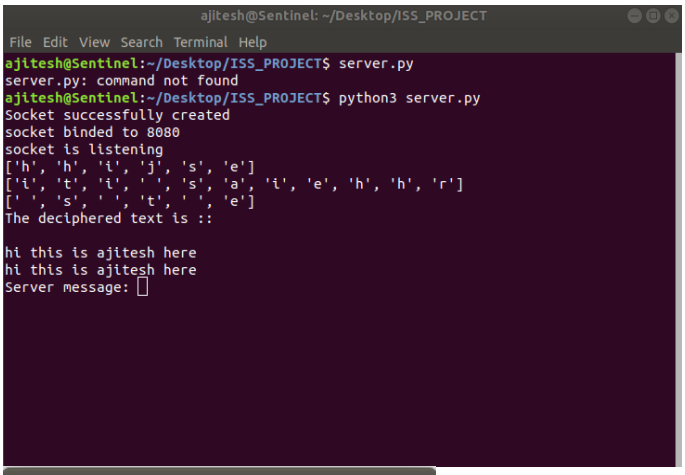
[4] 'Data encyption standard', FIPS PUB 46, National Bureau of Standards,Washington, DC Jan. 1977

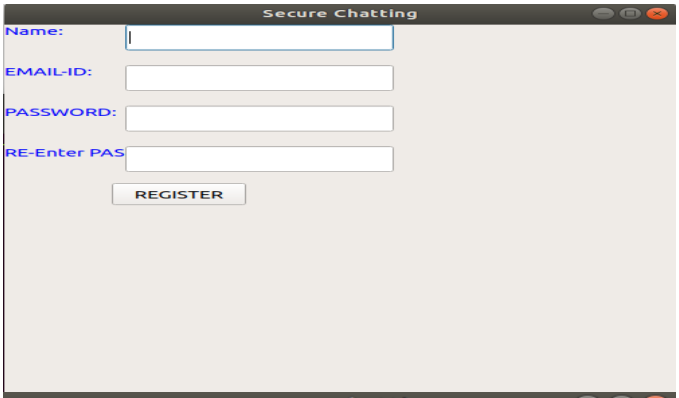
[5] Murat Fiskiran , Ruby B. Lee, —Workload Characterization of Elliptic Curve Cryptography and other Network Security Algorithms for Constrained Environments‖, IEEE International Workshop on Workload Characterization, 2002. WWC-5. 2002.

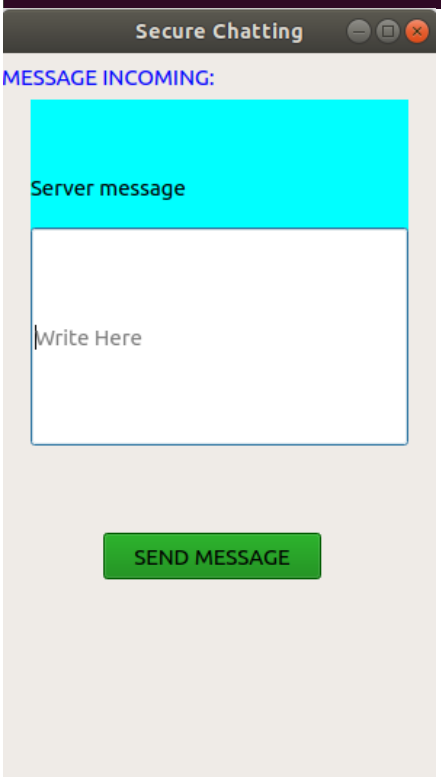
[6] RIVEST, R.L., SHAMIR, A., and ADLEMAN, L: 'A method for obtaining digital signatures and public-key cryptosystems', *CACM,* 1978, **21,** pp. 120-126

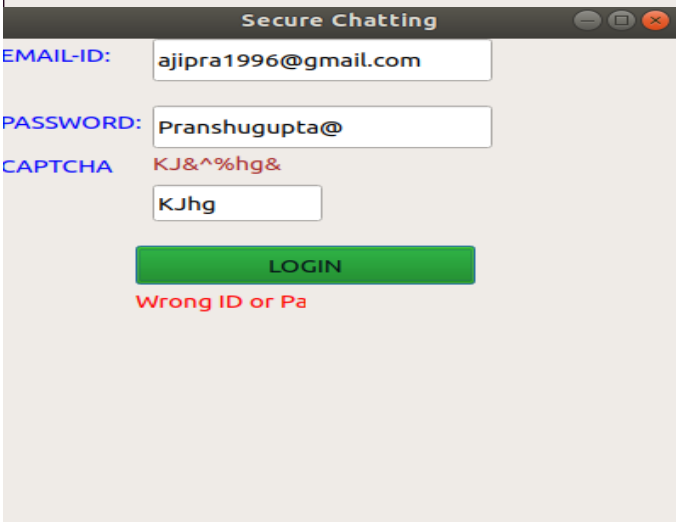
[7] Algorithms: http://www.cryptographyworld.com/algo.htm


Secure Chatting
SELECT LOGIN OR REGISTER
LOGIN
REGISTER

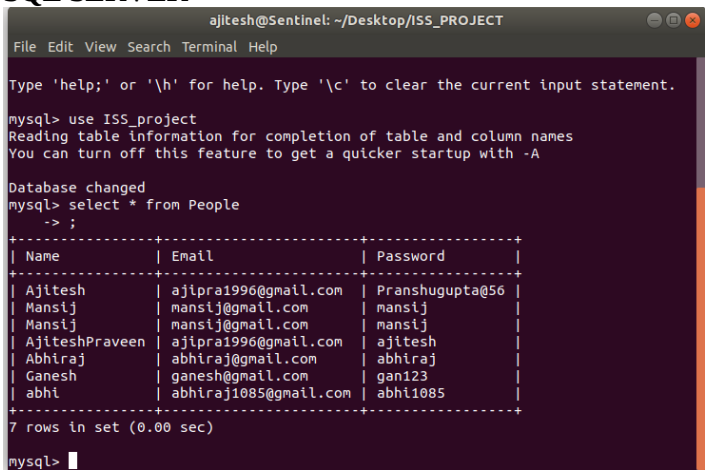

ajitesh@Sentinel: ~/Desktop/ISS_PROJECT
File Edit View Search Terminal Help
ajitesh@Sentinel:~/Desktop/ISS_PROJECT$ server.py
server.py: command not found
ajitesh@Sentinel:~/Desktop/ISS_PROJECT$ python3 server.py
Socket successfully created
socket binded to 8080
socket is listening
['h', 'h', 'i', 'j', 's', 'e']
['i', 't', 'i', ' ', 's', 'a', 'i', 'e', 'h', 'h', 'r']
[' ', 's', ' ', 't', ' ', 'e']
The deciphered text is ::
hi this is ajitesh here
hi this is ajitesh here
Server message:


Secure Chatting
Name:
EMAIL-ID:
PASSWORD:
RE-Enter PAS
REGISTER


Secure Chatting
MESSAGE INCOMING:
Server message
Write Here
SEND MESSAGE


Secure Chatting
EMAIL-ID:
ajipra1996@gmail.com
PASSWORD:
Pranshugupta@
CAPTCHA
KJ&^%hg&
KJhg
LOGIN
Wrong ID or Pa

SQL SERVER


ajitesh@Sentinel: ~/Desktop/ISS_PROJECT
File Edit View Search Terminal Help
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use ISS_project
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> select * from People
-> ;
Name	Email	Password
Ajitesh	ajipra1996@gmail.com	Pranshugupta@56
Mansij	mansij@gmail.com	mansij
Mansij	mansij@gmail.com	mansij
AjiteshPraveen	ajipra1996@gmail.com	ajitesh
Abhiraj	abhiraj@gmail.com	abhiraj
Ganesh	ganesh@gmail.com	gan123
abhi	abhiraj1085@gmail.com	abhi1085
7 rows in set (0.00 sec)
mysql>

CODE:

IMPLEMENTATION OF RAIL FENCE CIPHER:
ENCRYPTION:

```
file_plain_text = open("plain_text.txt","r")
p_text = file_plain_text.read()
file_plain_text.close()
depth = 3
text_modified = ""

for each_letter in p_text:
        text_modified = text_modified + each_letter

if len(text_modified) == 2:
        text_modified = text_modified + "x" + "x"

print ("The modified string is :  ",text_modified)

length_modified_text = len(text_modified)

arr = []

#Converting modified string into array

for i in text_modified:
        arr.append(i)

arr_1 = []
arr_2 = []
arr_3 = []
length_arr = len(arr)

i = 0
j = 1
k = 2
check  = 0

#Main Algorithm  -- Rail Fence Implementation
while i <= length_arr:
        if i < length_arr:
                arr_1.append(arr[i])
        i = i + 4

while j <= length_arr:
        if j < length_arr:
                arr_2.append(arr[j])
        j = j + 2

while k < length_arr:
        if k <= length_arr:
                arr_3.append(arr[k])
        k = k + 4
```

```
print(arr_1)
print("\n")
print(arr_2)
print("\n")
print(arr_3)


# Arranging the encrypted text
ciphered_text = ""

for i in arr_1:
        ciphered_text = ciphered_text + i


for i in arr_2:
        ciphered_text = ciphered_text + i


for i in arr_3:
        ciphered_text = ciphered_text + i


#Printing the ciphered text and writing it into a file
file_cipher = open("encryption_file.txt","w")
file_cipher.write(ciphered_text)
file_cipher.close()
print("The ciphered text is : \n")
print(ciphered_text)
```

DECRYPTION:

```
file_cipher_text = open("encryption_file.txt","r")
ciphered_text = file_cipher_text.read()
file_cipher_text.close()
##Decipher algortihm
d_cipher = []
decipher_text = ""
a = 0
b = 1
c = 2
count = 0

for i in ciphered_text:
        d_cipher.append(i)

length_d_cipher = len(d_cipher)
d_arr1 = [None] * length_d_cipher
d_arr2 = [None] * length_d_cipher
d_arr3 = [None] * length_d_cipher
count = 0

for j in d_cipher:
        if a < length_d_cipher:
```

```
                d_arr1[a] = j
                count = count + 1
        a = a + 4


for j in range(length_d_cipher):
        if b < length_d_cipher and count < length_d_cipher:
                d_arr2[b] = d_cipher[count]
                b = b + 2
                count = count + 1


for j in range(length_d_cipher):
        if c < length_d_cipher and count < length_d_cipher:
                d_arr3[c] = d_cipher[count]
                c = c + 4
                count = count + 1

d_arr11 = []
d_arr22 = []
d_arr33 = []

## Use these for further proceedings
for i in d_arr1:
        if i != None:
                d_arr11.append(i)

for j in d_arr2:
        if j != None:
                d_arr22.append(j)

for k in d_arr3:
        if k != None:
                d_arr33.append(k)



print(d_arr11)
print(d_arr22)
print(d_arr33)
length_d_arr11 = len(d_arr11)
length_d_arr22 = len(d_arr22)
length_d_arr33 = len(d_arr33)

mid_row = 0
for i in range(length_d_cipher):
        if i < length_d_arr11:
                decipher_text = decipher_text + d_arr11[i]

        if mid_row < length_d_arr22:
                decipher_text = decipher_text + d_arr22[mid_row]

        if i < length_d_arr33:
                decipher_text = decipher_text + d_arr33[i]

        mid_row = mid_row + 1
```

```
        if mid_row < length_d_arr22:
                decipher_text = decipher_text + d_arr22[mid_row]

        mid_row = mid_row + 1



print("The deciphered text is :: \n")
print(decipher_text)

file_cipher = open("decoded_text.txt","w")
file_cipher.write(decipher_text)
file_cipher.close()
```

IMPLEMENTATION OF CHATTING SERVER AND CLIENT:

SERVER:

```
import socket
import subprocess


s = socket.socket()
print ("Socket successfully created")
port = 8080
s.bind(('', port))
print ("socket binded to %s" %(port))
s.listen(5)
print ("socket is listening")
while True:
        c, addr = s.accept()
        cipher_text = (c.recv(1024)).decode("utf-8")
        file_cipher_text = open("encryption_file.txt","w")
        file_cipher_text.write(cipher_text)
        file_cipher_text.close()
        subprocess.check_call(["python3", "rail_fence_decipher.py"])

        file_decode = open("decoded_text.txt","r")
        t = file_decode.read()
        file_decode.close()
        print(t)

        msg = input("Server message: ")
        file_plain = open("plain_text.txt","w")
        file_plain.write(msg)
        file_plain.close()
        subprocess.check_call(["python3", "rail_fence.py"])
        file_cipher = open("encryption_file.txt","r")
        encrypted_text = file_cipher.read()
        file_cipher.close()
        c.send(bytes(encrypted_text,"utf8"))
        c.close()
```

CLIENT:

```
import sys
import subprocess
from PyQt5 import QtCore, QtWidgets
from PyQt5.QtWidgets import QMainWindow, QLabel, QGridLayout, QWidget, QLineEdit
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtCore import QSize
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import socket


class MainWindow(QMainWindow):
        def __init__(self):
                QMainWindow.__init__(self)

                self.setMinimumSize(QSize(300, 500))
                self.setWindowTitle("Secure Chatting")


                self.message_label = QLabel(self)
                self.message_label.setText("MESSAGE INCOMING:")
                self.message_label.setStyleSheet("QLabel {color : blue; }");
                self.message_label2 = QLabel(self)
                self.message_label2.setText("Server message")
                self.message_box = QLineEdit(self)
                self.send_button = QPushButton('SEND MESSAGE', self)
                self.send_button.setStyleSheet("background-color: green")
                self.send_button.clicked.connect(self.send_func)
                self.message_label.move(0, 0)
                self.message_label.resize(150, 32)
                self.message_label2.resize(260,120)
                self.message_label2.move(20, 32)
                self.message_label2.setStyleSheet("background: cyan");
                self.message_box.move(20, 120)
                self.message_box.resize(260, 150)
                self.message_box.setPlaceholderText("Write Here ")
                self.send_button.move(70, 330)
                self.send_button.resize(150,32)



        def send_func(self):
                s = socket.socket()
                port = 8080
                s.connect(('localhost', port))
                msg = str(self.message_box.text())
                file_plain = open("plain_text.txt","w")
                file_plain.write(msg)
                file_plain.close()
```

```
                subprocess.check_call(["python3", "rail_fence.py"])
                file_cipher = open("encryption_file.txt","r")
                cipher_text = file_cipher.read()
                file_cipher.close()
                s.send(bytes(cipher_text,"utf8"))

                incoming_msg = (s.recv(1024)).decode("utf-8")
                file_cipher_text = open("encryption_file.txt","w")
                file_cipher_text.write(incoming_msg)
                file_cipher_text.close()
                subprocess.check_call(["python3", "rail_fence_decipher.py"])

                file_decode = open("decoded_text.txt","r")
                t = file_decode.read()
                file_decode.close()
                self.message_label2.setText(t)
                s.close()



if __name__ == "__main__":
        app = QtWidgets.QApplication(sys.argv)
        mainWin = MainWindow()
        mainWin.show()
        sys.exit( app.exec_() )
```