// please run this command on linux before running code

//sudo apt-get install libcrypto++-utils libcrypto++8 libcrypto++-dev libcrypto++-doc

**Includes:**

#include <QApplication>

#include <QPushButton>

#include <QTextEdit>

#include <QWidget>

#include <QVBoxLayout>

#include <QInputDialog>

#include <QLabel>

#include <QMessageBox>

#include <QFileDialog>

#include "cryptopp/modes.h"

#include "cryptopp/aes.h"

#include "cryptopp/blowfish.h"

#include "cryptopp/filters.h"

#include "cryptopp/osrng.h"

#include "cryptopp/files.h"

These are the necessary header files for the Qt framework and Crypto++ library. The Qt headers are for building the graphical user interface (GUI), and the Crypto++ headers are for cryptographic operations.

**Namespace Declarations:**

class CryptoApp : public QWidget {

public:

CryptoApp(QWidget \*parent = 0) : QWidget(parent) {

// Constructor for the main application class

// GUI setup and initialization of member variables

}

private slots:

// Definition of private slots (event-handling functions)

GUI setups:

QVBoxLayout \*layout = new QVBoxLayout(this);

inputText = new QTextEdit(this);

outputText = new QTextEdit(this);

outputText->setReadOnly(true);

// ... (similar setup for other GUI elements)

These lines initialize the GUI layout and various widgets such as text edits, labels, and buttons. The layout is organized in a vertical box (**QVBoxLayout**), and several buttons for encryption/decryption are created.

**WelcomeLabel:**

QLabel \*welcomeLabel = new QLabel("Welcome to the uncrackable encryption tool!", this);

welcomeLabel->setAlignment(Qt::AlignCenter);

QColor redColor(255, 0, 0);

QPalette palette;

palette.setColor(QPalette::WindowText, redColor);

welcomeLabel->setPalette(palette);

QFont boldFont;

boldFont.setBold(true);

welcomeLabel->setFont(boldFont);

layout->addWidget(welcomeLabel);

**Button Connections:**

connect(aesEncryptButton, &QPushButton::clicked, this, &CryptoApp::encryptAes);

// ... (similar connections for other buttons)

These lines establish connections between button clicks and corresponding slots (functions). For example, when the "AES Encrypt" button is clicked, the **encryptAes** function will be called.

**File Encryption Buttons:**

QPushButton \*fileEncryptAesButton = new QPushButton("File AES Encrypt", this);

// ... (similar setup for other file encryption buttons)

connect(fileEncryptAesButton, &QPushButton::clicked, this, &CryptoApp::fileEncryptAes);

// ... (similar connections for other file encryption buttons)

These buttons handle file encryption and decryption. When clicked, they prompt the user to select a file, choose an output file, and perform the encryption or decryption using AES or Blowfish.

**Private Slots Implementation:**

void encryptAes() { /\* ... \*/ }

// ... (similar implementation for other private slots)

These are the implementation details for the private slots. For instance, **encryptAes** handles the encryption of the entered text using AES, and similar functions handle decryption, file encryption, and file decryption.

**Main Function:**

int main(int argc, char \*argv[]) {

QApplication app(argc, argv);

CryptoApp cryptoApp;

cryptoApp.show();

return app.exec();

}

The main function sets up the Qt application, creates an instance of the **CryptoApp** class, shows the main window, and enters the Qt event loop.

This code creates a basic cryptographic application with a GUI for text and file encryption/decryption using AES and Blowfish algorithms. The GUI includes buttons, text input fields, and labels for user interaction.