PictureCrypt

1.3.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

PictureCrypt

Project made using QT Creator in C++

1.1 The idea of the project

The idea came to me, when I read an article about steganoraphy. I realised, that you can store data in an image in pixels near the border, so noone can see and even if they did, it is practically impossible to decipher the contents.

1.2 Realisation

To create the encrypted image, you need to select any file for encryption, then using EncryptDialog you select the image to store the data. Then output image is generated.

Attention

Output image format available is .PNG, because .jpg isn't lossless, so the pixels containing data would be seriously simplified and the data damaged. .BMP isn't used, because noone really uses it and .PNG is just compressed .BMP (more or less)

Note

JPHS support is under development :D

1.3 How can someone use it?

Well... Anyone who wants to securely commuicate. For example your boss watches your inbox, so you do the work and don't chat with your friends about the bar, they've just visited. Using this app you can send them a photo of your desk, saying it's my new working space, but inside the image there is secret message saying "Wanna get another beer tonight? xD". Boss sees this image, but doesn't spot anyhing. Great example...

1.4 Structure of the project.

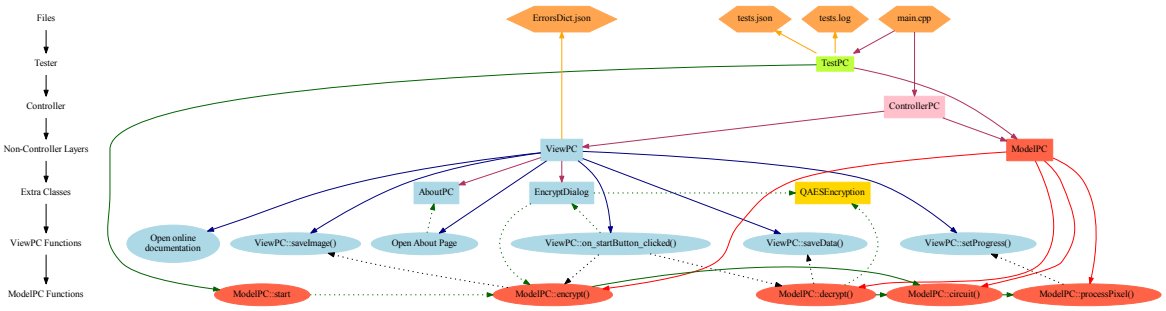
Project is done via MVC Pattern. View and Model layers are totally isolated and run on different threads.

Code from controller.cpp

```
view = new ViewPC();
model = new ModelPC(version);
QThread * modelThread = new QThread();
model->moveToThread(modelThread);
modelThread->start();
```

So when Model is hard-working, View layer is just fine.

Layers also have a ton of functions, so here is a scheme, that I was doing for about 10 hours, which demonstrates the most important functions and classes in the project. And everything is clickable here, so try it out!



Well... I think you didn't quite understand what is happening here... So hop into my "User-friendly" Documentation!

See source on https://github.com/waleko/PictureCrypt

Note

QAESEncryption class done by Bricke

1.5 External use

ModelPC class can be used externally (without UI)

Note

TestPC class was introduced recently, its use is adviced.

```
#include <modelpc.h>
#include <testpc.h>
#include <QByteArray>
#include <QImage>

#include <QDebug> // Just for demonstration use

...

TestPC testing;
if(!testing.startTest())
    return;
ModelPC * model = new ModelPC();

// Embedding
QImage * resultImage = model->start(QByteArray data, // Data to be embedded
                                    QImage *image, // Image for embedding
                                    int mode = 0, // Mode of embedding
                                    QString key = "", // Key for extra-encryption (if empty, key will be
       generated automatically)
                                    int bitsUsed = 8, // Bits per Byte used (better explaination
       ModelPC::bitsUsed)
                                    QString *error = nullptr); // Error output, if everything is ok, error
       will be "ok"
if(*error != "ok")
    return;
// Note *error is just a code of error (like "muchdata", dictionary of error codes is also available on
       github.

// De-embedding
QByteArray output = model->decrypt(QImage * image, // Image with hidden data
                                   QString *_error = nullptr); // Error output
if(data == output)
   qDebug() << "Great success!";
else
   qDebug() << "Fiasco :(";
```

See also

ModelPC, ModelPC::ModelPC, ModelPC::saveData, ModelPC::saveImage, ModelPC::alertView, ModelPC::setProgress

1.6 JPHS use

The newer versions of the app have jphs support, but they don't have jphs built in as it is provided under GNU General Public License v3.0, is "for test purposes only" and is illegal in some countries, so...

Attention

We support JPHS, but we don't use any responsibility for it, we never used or downloaded it, we just used .exe output in the web, and it somehow works by chance. All responsibility for using jphs is on you, that is why we use made only optionally. That means that to use jphs with our app you will have to download the jphs yourself and specify the jphs directory. However we provide link to the site where you can download the supported version of the jphs: http://linux01.gwdg.de/~alatham/stego.html As it's not our site publishing the dangerous zip file, we just put link to that site (Google does that too, so what? Sue Google?), This text is subject to United Nations' Universal Declaration of Human Rights, (see Article 19 http://www.un.org/en/universal-declaration-human-rights):

> Everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers.

And I typed this link randomly, and I'm scared...

1.7 License

This software is provided under the UNLICENSE

1.8 Contact us

Visit Github Page: https://waleko.github.io and our site http://alex.unaux.com

Email me at a.kovrigin0@gmail.com

Author

Alex Kovrigin

Chapter 2

Contributor Covenant Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at a.kovrigin0@gmail.com. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

Chapter 3

PictureCrypt

Make your pictures crypted.

About

Project is made only using QT. QAESEncryption by bricke was also used. MVC pattern used. PictureCrypt project is UI based, the model contains all buisness logic and can work as standalone class.

External use

ModelPC class can be used externally (without UI)

```
#include <modelpc.h>
#include <testpc.h>
#include <QByteArray>
#include <QImage>

...
// Testing the ModelPC
TestPC testing;
if(!testing.startTest())
    return;
ModelPC * model = new ModelPC();

// Embedding
QImage * resultImage = model->start(QByteArray data, // Data to be embedded
                                    QImage *image, // Image for embedding
                                    int mode = 0, // Mode of embedding
                                    QString key = "", // Key for extra-encryption (if empty, key will be
       generated automatically)
                                    int bitsUsed = 8, // Bits per Byte used (better explaination
       ModelPC::bitsUsed)
                                    QString *error = nullptr); // Error output, if everything is ok, error
       will be "ok"
if(*error != "ok")
    return;
// Note *error is just a code of error (like "muchdata", dictionary of error codes is also available on
       github.

// De-embedding
QByteArray output = model->decrypt(QImage * image, // Image with hidden data
                                   QString *_error = nullptr); // Error output
if(data == output)
   qDebug() << "Great success!";
```

Available modes of embedding

- 0 - Standard, created by me
- 1 - JPHS, requires manually installed JPHS and specified directory (not currently available).

Documentation

Doxygen Documentation avaible here

Dependencies

- qtcore

Used works from other people

- QAESEncryption by bricke

Contact

Question or suggestions are welcome! Please use the GitHub issue tracking to report suggestions or issues. Email me a.kovrigin0@gmail.com and visit my site http://alex.unaux.com

License

This software is provided under the UNLICENSE

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Namespace Documentation

8.1 ErrorsDictSetup Namespace Reference

Variables

- string filename = 'ErrorsDict.json'
- raw = open(filename, 'r')
- data = json.load(raw)
- input_data = input()
- key
- value
- f
- indent

8.1.1 Variable Documentation

8.1.1.1 data

```
ErrorsDictSetup.data = json.load(raw)
```

Definition at line 6 of file ErrorsDictSetup.py.

8.1.1.2 f

```
ErrorsDictSetup.f
```

Definition at line 22 of file ErrorsDictSetup.py.

8.1.1.3 filename

```
string ErrorsDictSetup.filename = 'ErrorsDict.json'
```

Definition at line 2 of file ErrorsDictSetup.py.

8.1.1.4 indent

```
ErrorsDictSetup.indent
```

Definition at line 22 of file ErrorsDictSetup.py.

8.1.1.5 input_data

```
ErrorsDictSetup.input_data = input()
```

Definition at line 14 of file ErrorsDictSetup.py.

8.1.1.6 key

```
ErrorsDictSetup.key
```

Definition at line 17 of file ErrorsDictSetup.py.

8.1.1.7 raw

```
ErrorsDictSetup.raw = open(filename, 'r')
```

Definition at line 4 of file ErrorsDictSetup.py.

8.1.1.8 value

```
ErrorsDictSetup.value
```

Definition at line 17 of file ErrorsDictSetup.py.

8.2 tests-setup Namespace Reference

Variables

- string filename = 'tests.json'
- raw = open(filename, 'r')
- js = json.load(raw)
- sep
- input_data = input()
- list arr = [ ]
- data
- image
- expect
- mode
- key
- bitsUsed
- dictionary obj = {'data':data, 'image':image,'expectation':expect,'mode':int(mode),'key':key,'bitsUsed'↩ :int(bitsUsed)}
- f
- indent

8.2.1 Variable Documentation

8.2.1.1 arr

```
list tests-setup.arr = [ ]
```

Definition at line 16 of file tests-setup.py.

8.2.1.2 bitsUsed

```
tests-setup.bitsUsed
```

Definition at line 18 of file tests-setup.py.

8.2.1.3 data

```
tests-setup.data
```

Definition at line 18 of file tests-setup.py.

8.2.1.4 expect

```
tests-setup.expect
```

Definition at line 18 of file tests-setup.py.

8.2.1.5 f

```
tests-setup.f
```

Definition at line 26 of file tests-setup.py.

8.2.1.6 filename

```
string tests-setup.filename = 'tests.json'
```

Definition at line 2 of file tests-setup.py.

8.2.1.7 image

```
tests-setup.image
```

Definition at line 18 of file tests-setup.py.

8.2.1.8 indent

```
tests-setup.indent
```

Definition at line 26 of file tests-setup.py.

8.2.1.9 input_data

```
tests-setup.input_data = input()
```

Definition at line 14 of file tests-setup.py.

8.2.1.10 js

```
tests-setup.js = json.load(raw)
```

Definition at line 6 of file tests-setup.py.

8.2.1.11 key

```
tests-setup.key
```

Definition at line 18 of file tests-setup.py.

8.2.1.12 mode

```
tests-setup.mode
```

Definition at line 18 of file tests-setup.py.

8.2.1.13 obj

```
dictionary tests-setup.obj = {'data':data, 'image':image,'expectation':expect,'mode':int(mode),'key'↩
:key,'bitsUsed':int(bitsUsed)}
```

Definition at line 20 of file tests-setup.py.

8.2.1.14 raw

```
tests-setup.raw = open(filename, 'r')
```

Definition at line 4 of file tests-setup.py.

8.2.1.15 sep

```
tests-setup.sep
```

Definition at line 9 of file tests-setup.py.

8.3 Ui Namespace Reference

Chapter 9

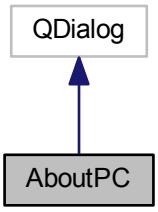
Class Documentation

9.1 AboutPC Class Reference

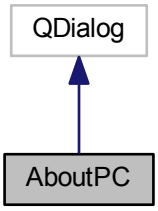
The AboutPC class The About Page dialog.

`#include <aboutpc.h>`

Inheritance diagram for AboutPC:



Collaboration diagram for AboutPC:



Public Member Functions

- AboutPC (QWidget *parent=0)
- ~AboutPC ()
- void setVersion (QString version)

  AboutPC::setVersion Function to set the version display.

9.1.1 Detailed Description

The AboutPC class The About Page dialog.

Definition at line 12 of file aboutpc.h.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 AboutPC()

```
AboutPC::AboutPC (
            QWidget * parent = 0 )  [explicit]
```

Definition at line 4 of file aboutpc.cpp.

9.1.2.2 ~AboutPC()

```
AboutPC::~AboutPC ( )
```

Definition at line 11 of file aboutpc.cpp.

9.1.3 Member Function Documentation

9.1.3.1 setVersion()

```
void AboutPC::setVersion (
            QString version )
```

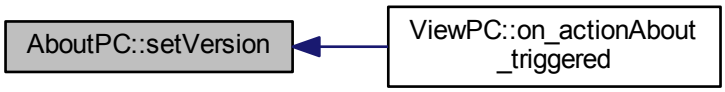
AboutPC::setVersion Function to set the version display.

Parameters

version	Version as QString

Definition at line 19 of file aboutpc.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

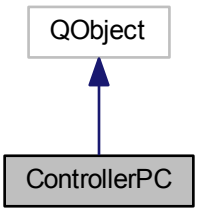
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/aboutpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/aboutpc.cpp

9.2 ControllerPC Class Reference

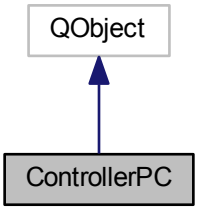
The ControllerPC class Controller class, which controls View and Model layers.

`#include <controllerpc.h>`

Inheritance diagram for ControllerPC:



Collaboration diagram for ControllerPC:



Public Slots

- void abortCircuit ()

  ControllerPC::abortCircuit Slot to be called when ProgressDialog in ViewPC is closed. It flags ModelPC to stop.
- void setBitsUsed (int bitsUsed)

  ControllerPC::setBitsUsed Slot to set ModelPC::bitsUsed.
- void setJPHSDir (QString dir)

  ControllerPC::setJPHSDir Sets JPHS default dir.

Public Member Functions

- ControllerPC ()

  ControllerPC::ControllerPC Constructor of controller Constructor runs auto-test for ModelPC, creates Model Class (ModelPC) and View Class (ViewPC). All signals and slots are connected here.

Public Attributes

- long int version

  version Version of the app
- QString versionString

  versionString Version of the app as QString.

9.2.1 Detailed Description

The ControllerPC class Controller class, which controls View and Model layers.

See also

ViewPC, ModelPC

Definition at line 19 of file controllerpc.h.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 ControllerPC()

```
ControllerPC::ControllerPC ( )
```

ControllerPC::ControllerPC Constructor of controller Constructor runs auto-test for ModelPC, creates Model Class (ModelPC) and View Class (ViewPC). All signals and slots are connected here.

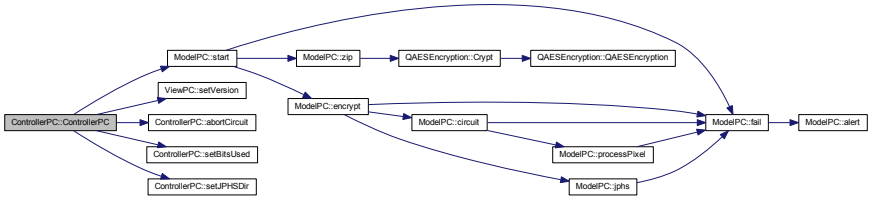
Controller class

Note

Version of the app is specified here.

Definition at line 9 of file controllerpc.cpp.

Here is the call graph for this function:



9.2.3 Member Function Documentation

9.2.3.1 abortCircuit

```
void ControllerPC::abortCircuit ( )  [slot]
```

ControllerPC::abortCircuit Slot to be called when ProgressDialog in ViewPC is closed. It flags ModelPC to stop.

Definition at line 36 of file controllerpc.cpp.

Here is the caller graph for this function:

9.2.3.2 setBitsUsed

```
void ControllerPC::setBitsUsed (
            int bitsUsed )  [slot]
```

ControllerPC::setBitsUsed Slot to set ModelPC::bitsUsed.

Parameters

bitsUsed	Value

Definition at line 44 of file controllerpc.cpp.

Here is the caller graph for this function:

9.2.3.3 setJPHSDir

```
void ControllerPC::setJPHSDir (
            QString dir )  [slot]
```

ControllerPC::setJPHSDir Sets JPHS default dir.

Parameters

dir	Directory

Definition at line 52 of file controllerpc.cpp.

Here is the caller graph for this function:

9.2.4 Member Data Documentation

9.2.4.1 version

```
long int ControllerPC::version
```

version Version of the app

Definition at line 27 of file controllerpc.h.

9.2.4.2 versionString

```
QString ControllerPC::versionString
```

versionString Version of the app as QString.

Definition at line 31 of file controllerpc.h.

The documentation for this class was generated from the following files:

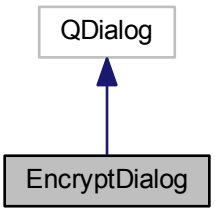
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/controllerpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/controllerpc.cpp

9.3 EncryptDialog Class Reference

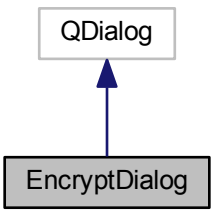
The EncryptDialog class Class to get the image and key to store secret info.

```
#include <encryptdialog.h>
```

Inheritance diagram for EncryptDialog:



Collaboration diagram for EncryptDialog:



Public Slots

- void on_fileButton_clicked ()

  EncryptDialog::on_fileButton_clicked Slot to select the image.
- void on_buttonBox_accepted ()

  EncryptDialog::on_buttonBox_accepted Slot to start the encryption. Successful closing of the app.
- void on_buttonBox_rejected ()

  EncryptDialog::on_buttonBox_rejected Slot to reject the encryption.
- void on_horizontalSlider_valueChanged (int value)

  EncryptDialog::on_horizontalSlider_valueChanged Slot if value of the slider is changed. Key is generated here.

Public Member Functions

- EncryptDialog (QByteArray _data, QWidget ∗parent=0)

  EncryptDialog::EncryptDialog Constructor of the class. Input data is saved here and some variables are set here.
- ∼EncryptDialog ()
- QByteArray zip ()

  EncryptDialog::zip Zipping algorithm It copresses the data and then compresses it using qCompress()

Public Attributes

- QByteArray data

  data Input data
- bool success

  success Flag, if image was successfully selected and data was encrypted.
- QByteArray compr_data

  compr_data Compressed data, aka Output data.
- QString inputFileName

  inputFileName Filename of the image.
- long long int size

  size Size of the image in square pixels
- QString key

  key Key to be used for encryption in EncrytDialog::zip
- bool goodPercentage

  goodPercentage Flag if area of the used data via encryption is less than 70% of the area of the image.
- int val

  val Value of the slider
- int bitsUsed

  bitsUsed Bits used per byte of pixel.
- QImage image

  image Inputted image

9.3.1 Detailed Description

The EncryptDialog class Class to get the image and key to store secret info.

Note

Not the most important and well written class.

See also

ViewPC

Definition at line 21 of file encryptdialog.h.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 EncryptDialog()

```
EncryptDialog::EncryptDialog (
            QByteArray _data,
            QWidget * parent = 0 )  [explicit]
```

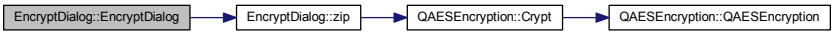
EncryptDialog::EncryptDialog Constructor of the class. Input data is saved here and some variables are set here.

Parameters

_data	Input data.
parent	Parent (not in use)

Definition at line 9 of file encryptdialog.cpp.

Here is the call graph for this function:



9.3.2.2 ~EncryptDialog()

```
EncryptDialog::~EncryptDialog ( )
```

Definition at line 29 of file encryptdialog.cpp.

9.3.3 Member Function Documentation

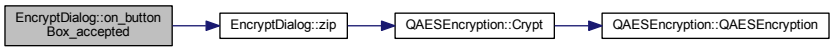
9.3.3.1 on_buttonBox_accepted

```
void EncryptDialog::on_buttonBox_accepted ( )  [slot]
```

EncryptDialog::on_buttonBox_accepted Slot to start the encryption. Successful closing of the app.

Definition at line 85 of file encryptdialog.cpp.

Here is the call graph for this function:



9.3.3.2 on_buttonBox_rejected

```
void EncryptDialog::on_buttonBox_rejected ( )  [slot]
```

EncryptDialog::on_buttonBox_rejected Slot to reject the encryption.

Definition at line 100 of file encryptdialog.cpp.

9.3.3.3 on_fileButton_clicked

```
void EncryptDialog::on_fileButton_clicked ( )  [slot]
```

EncryptDialog::on_fileButton_clicked Slot to select the image.

Definition at line 60 of file encryptdialog.cpp.

9.3.3.4 on_horizontalSlider_valueChanged

```
void EncryptDialog::on_horizontalSlider_valueChanged (
            int value )  [slot]
```

EncryptDialog::on_horizontalSlider_valueChanged Slot if value of the slider is changed. Key is generated here.

Parameters

value	Value of the slider.

Definition at line 110 of file encryptdialog.cpp.

9.3.3.5 zip()

```
QByteArray EncryptDialog::zip ( )
```

EncryptDialog::zip Zipping algorithm It copresses the data and then compresses it using qCompress()

Returns

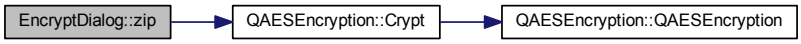
Returns Compressed data.

See also

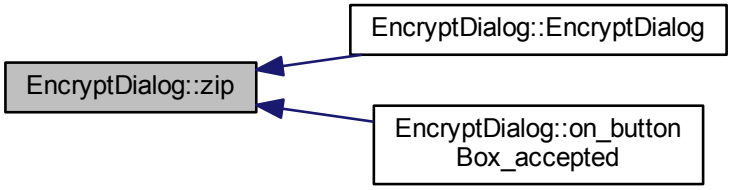
ModelPC::unzip

Definition at line 49 of file encryptdialog.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4 Member Data Documentation

9.3.4.1 bitsUsed

```
int EncryptDialog::bitsUsed
```

bitsUsed Bits used per byte of pixel.

See also

ModelPC::circuit

Definition at line 75 of file encryptdialog.h.

9.3.4.2 compr_data

```
QByteArray EncryptDialog::compr_data
```

compr_data Compressed data, aka Output data.

Definition at line 50 of file encryptdialog.h.

9.3.4.3 data

```
QByteArray EncryptDialog::data
```

data Input data

Definition at line 42 of file encryptdialog.h.

9.3.4.4 goodPercentage

```
bool EncryptDialog::goodPercentage
```

goodPercentage Flag if area of the used data via encryption is less than 70% of the area of the image.

Definition at line 66 of file encryptdialog.h.

9.3.4.5 image

```
QImage EncryptDialog::image
```

image Inputted image

Definition at line 79 of file encryptdialog.h.

9.3.4.6 inputFileName

```
QString EncryptDialog::inputFileName
```

inputFileName Filename of the image.

Definition at line 54 of file encryptdialog.h.

9.3.4.7 key

```
QString EncryptDialog::key
```

key Key to be used for encryption in EncrytDialog::zip

Definition at line 62 of file encryptdialog.h.

9.3.4.8 size

```
long long int EncryptDialog::size
```

size Size of the image in square pixels

Definition at line 58 of file encryptdialog.h.

9.3.4.9 success

```
bool EncryptDialog::success
```

success Flag, if image was successfully selected and data was encrypted.

Definition at line 46 of file encryptdialog.h.

9.3.4.10 val

```
int EncryptDialog::val
```

val Value of the slider

Definition at line 70 of file encryptdialog.h.

The documentation for this class was generated from the following files:

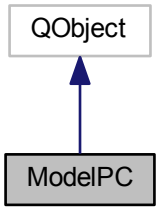
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/encryptdialog.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/encryptdialog.cpp

9.4 ModelPC Class Reference

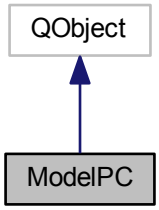
The ModelPC class Model Layer of the app. Controled by ControllerPC.

`#include <modelpc.h>`

Inheritance diagram for ModelPC:



Collaboration diagram for ModelPC:



Public Slots

- QImage * start (QByteArray data, QImage *image, int mode=0, QString key="", int _bitsUsed=8, QString *_error=nullptr)

  ModelPC::start Slot to zip and encrypt data and provide it with some extra stuff After completion start standard ModelPC::encrypt Isn't used in PictureCrypt, but used can be used in other - custom projects.
- QImage * encrypt (QByteArray encr_data, QImage *image, int mode=0, QString *_error=nullptr)

  ModelPC::encrypt Slot to be called when encrypt mode in ViewPC is selected and started.
- QByteArray decrypt (QImage *image, QString *_error=nullptr)

  ModelPC::decrypt Slot to be called when decrypt mode in ViewPC is selected and started.
- void fail (QString message)

  ModelPC::fail Slot to stop execution of cryption.

Signals

- alertView (QString messageCode, bool isWarning)

  alertView Signal to be called to create MessageBox.
- saveData (QByteArray data)

  saveData Signal to be called to save data from ModelPC::decrypt.
- saveImage (QImage *image)

  saveImage Signal to be called to save image from ModelPC::encrypt.
- setProgress (int val)

  setProgress Signal to be called to set progress of ProgressDialog.

Public Member Functions

- ModelPC ()

  ModelPC::ModelPC Constructor Unit tests are run here.
- QByteArray unzip (QByteArray data, QByteArray key)

  ModelPC::unzip Unzip data from ModelPC::decrypt. Just mirrored EncryptDialog::zip.
- void alert (QString message, bool isWarning=false)

  ModelPC::alert Function emits signal ModelPC::alertView and calls ViewPC::alert.

Public Attributes

- bool success

  success Flag that true by default, but in case of error or cancelling of ProgressDialog it turns to false, which stops execution of ModelPC::circuit
- long version

  version Version of the class
- QString versionString

  versionString Version as string
- int curMode

  curMode Mode of en- or decryption
- int bitsUsed

  bitsUsed Bits per byte used in pixel
- QString defaultJPHSDir

  defaultJPHSDir Default JPHS directory
- QString * error

  error Current error

Protected Member Functions

- void circuit (QImage *image, QByteArray *data, long long int countBytes)

  ModelPC::circuit The brain of the app. Via special circuit stores data in image.
- void jphs (QImage *image, QByteArray *data)

  ModelPC::jphs JPHS function to use jphide and jpseek (currently under development)
- void processPixel (QPoint pos, QVector< QPoint > *were, bool isEncrypt)

  ModelPC::processPixel Processes every pixel. Reads its contains or writes data.
- QByteArray zip (QByteArray data, QByteArray key)

  ModelPC::zip Zip function, copy of EncryptDialog::zip Used for ModelPC in custom projects, other than PictureCrypt.

9.4.1 Detailed Description

The ModelPC class Model Layer of the app. Controled by ControllerPC.

See also

ViewPC, ControllerPC

Definition at line 27 of file modelpc.h.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 ModelPC()

```
ModelPC::ModelPC ( )
```

ModelPC::ModelPC Constructor Unit tests are run here.

See also

ControllerPC, ViewPC

Definition at line 8 of file modelpc.cpp.

9.4.3 Member Function Documentation

9.4.3.1 alert()

```
void ModelPC::alert (
            QString message,
            bool isWarning = false )
```

ModelPC::alert Function emits signal ModelPC::alertView and calls ViewPC::alert.

Parameters

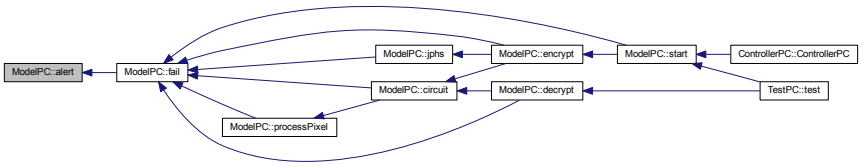
message	Message to be transmitted.
isWarning	Flag if message is critical.

See also

ViewPC::alert

Definition at line 593 of file modelpc.cpp.

Here is the caller graph for this function:



9.4.3.2 alertView

```
ModelPC::alertView (
            QString messageCode,
            bool isWarning )  [signal]
```

alertView Signal to be called to create MessageBox.

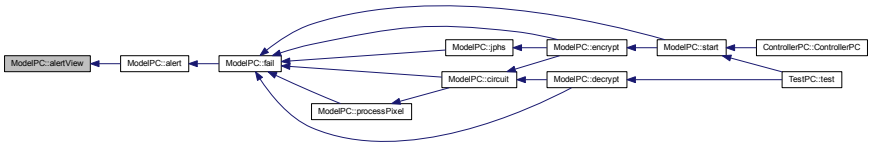
Parameters

messageCode	Message Code to be shown.
isWarning	Flag if message is critical.

See also

ModelPC::alert, ViewPC::alert

Here is the caller graph for this function:



9.4.3.3 circuit()

```
void ModelPC::circuit (
            QImage * image,
            QByteArray * data,
            long long int countBytes )  [protected]
```

ModelPC::circuit The brain of the app. Via special circuit stores data in image.

The circuit itself can be found in documentation or in commentaries in source.

Parameters

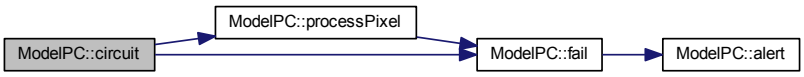
image	Image to be processed.
data	Data to be processed.
countBytes	Number of bytes to be read or written.

See also

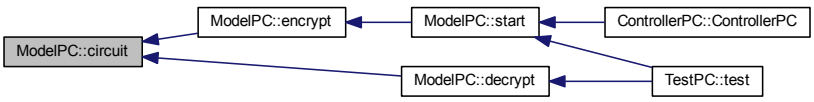
ModelPC::processPixel

Definition at line 297 of file modelpc.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.4.3.4 decrypt

```
QByteArray ModelPC::decrypt (
            QImage * image,
            QString * _error = nullptr )  [slot]
```

ModelPC::decrypt Slot to be called when decrypt mode in ViewPC is selected and started.

Parameters

image	Image to be decrypted.

Returns

Returns decrypted data

Parameters

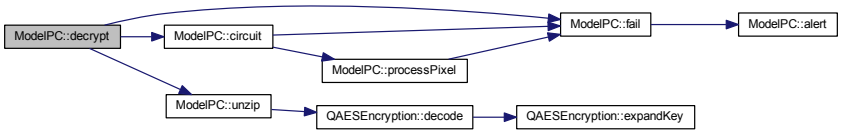
_error	Error output

See also

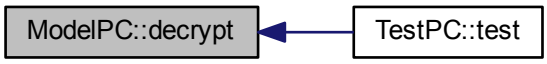
ViewPC::on_startButton_clicked, ModelPC::encrypt, ModelPC::circuit

Definition at line 145 of file modelpc.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.4.3.5 encrypt

```
QImage * ModelPC::encrypt (
            QByteArray encr_data,
            QImage * image,
            int mode = 0,
            QString * _error = nullptr )  [slot]
```

ModelPC::encrypt Slot to be called when encrypt mode in ViewPC is selected and started.

Parameters

encr_data	Data to be inserted to an image.
image	Image to be inserted in.
mode	Mode of encryption
_error	Error output

Returns

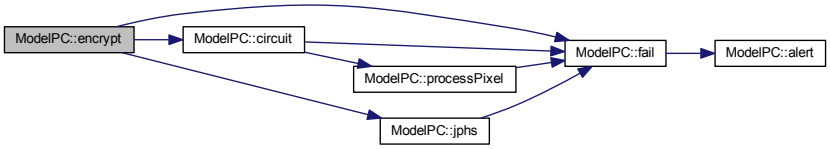
Returns image with embedded data.

See also

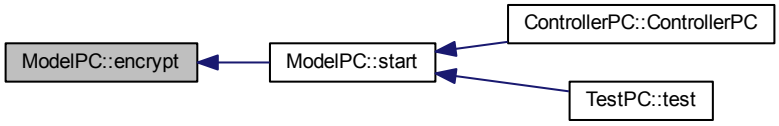
ViewPC::on_startButton_clicked, ModelPC::decrypt, ModelPC::circuit, ModelPC::start

Definition at line 93 of file modelpc.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.4.3.6 fail

```
void ModelPC::fail (
            QString message )  [slot]
```

ModelPC::fail Slot to stop execution of cryption.

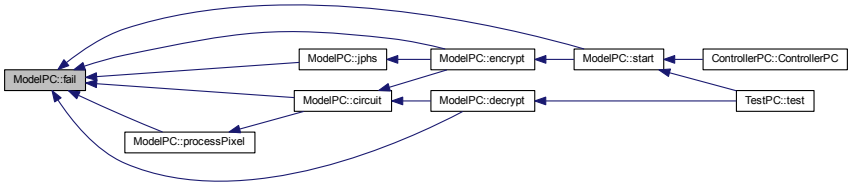
Parameters

message	Message for user

Definition at line 224 of file modelpc.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



9.4.3.7 jphs()

```
void ModelPC::jphs (
            QImage * image,
            QByteArray * data )  [protected]
```

ModelPC::jphs JPHS function to use jphide and jpseek (currently under development)

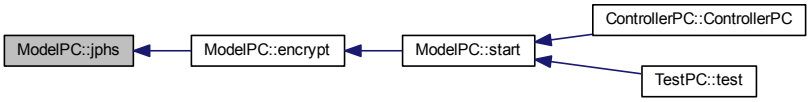
Parameters

image	Image for embedding
data	Data

Definition at line 236 of file modelpc.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



9.4.3.8 processPixel()

```
void ModelPC::processPixel (
            QPoint pos,
            QVector< QPoint > * were,
            bool isEncrypt )  [protected]
```

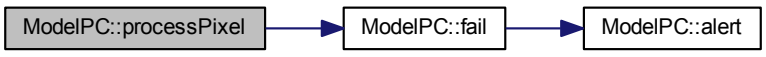
ModelPC::processPixel Processes every pixel. Reads its contains or writes data.

Parameters

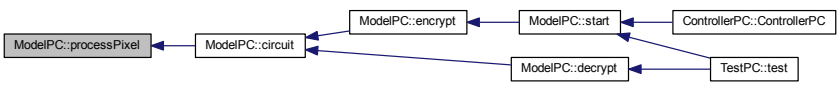
pos	Position of pixel
were	Vector array containing pixels, that were already processed.
isEncrypt	Mode of operation. If true encryption operations will continue, else the decryption ones.

Definition at line 439 of file modelpc.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.4.3.9 saveData

```
ModelPC::saveData (
            QByteArray data )  [signal]
```

saveData Signal to be called to save data from ModelPC::decrypt.

Parameters

data	Data to be saved.

Here is the caller graph for this function:

9.4.3.10 saveImage

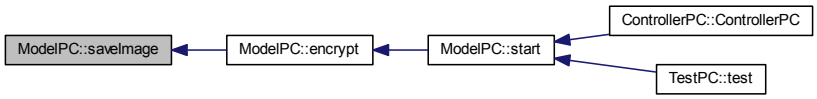
```
ModelPC::saveImage (
            QImage * image )  [signal]
```

saveImage Signal to be called to save image from ModelPC::encrypt.

Parameters

image	Image to be saved.

Here is the caller graph for this function:



9.4.3.11 setProgress

```
ModelPC::setProgress (
            int val )  [signal]
```

setProgress Signal to be called to set progress of ProgressDialog.

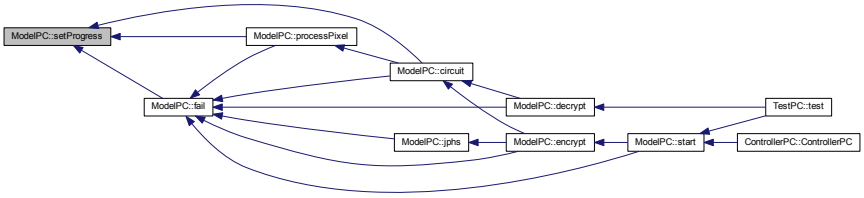
Parameters

val	Value to be set.

See also

ViewPC::setProgress

Here is the caller graph for this function:



9.4.3.12 start

```
QImage * ModelPC::start (
            QByteArray data,
            QImage * image,
            int mode = 0,
            QString key = "",
            int _bitsUsed = 8,
            QString * _error = nullptr )  [slot]
```

ModelPC::start Slot to zip and encrypt data and provide it with some extra stuff After completion start standard ModelPC::encrypt Isn't used in PictureCrypt, but used can be used in other - custom projects.

Parameters

data	Data for embedding
image	Image for embedding
mode	Mode for embedding
key	Key for extra encryption (if empty, key will be auto-generated)
_bitsUsed	Bits per byte (see ModelPC::bitsUsed)
_error	Error output

Returns

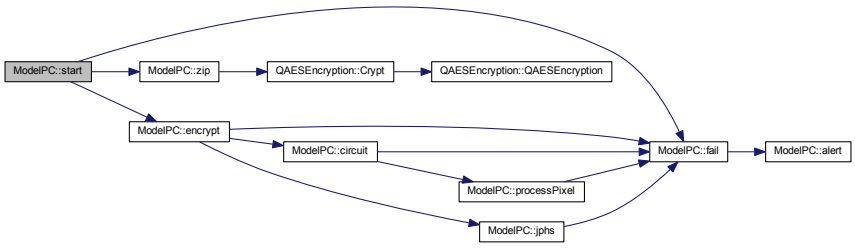
Returns image with embedded data

See also

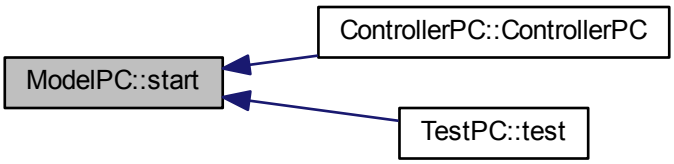
ModelPC::encrypt

Definition at line 35 of file modelpc.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.4.3.13 unzip()

```
QByteArray ModelPC::unzip (
            QByteArray data,
            QByteArray key )
```

ModelPC::unzip Unzip data from ModelPC::decrypt. Just mirrored EncryptDialog::zip.

Parameters

data	Data to be decrypted.
key	Key to decrypt the data.

Returns

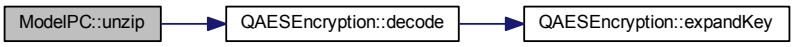
Returns data

See also

EncryptDialog::zip, ModelPC::decrypt, ModelPC::zip

Definition at line 532 of file modelpc.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

9.4.3.14 zip()

```
QByteArray ModelPC::zip (
            QByteArray data,
            QByteArray key )  [protected]
```

ModelPC::zip Zip function, copy of EncryptDialog::zip Used for ModelPC in custom projects, other than PictureCrypt.

Parameters

data	Data to be encrypted
key	Key for encryption

Returns

Returns decrypted data

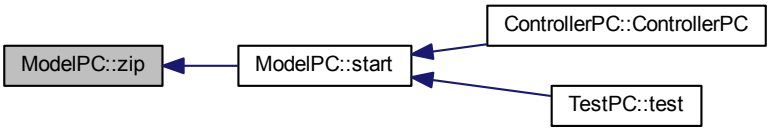
See also

ModelPC::start, ModelPC::encrypt, ModelPC::unzip

Definition at line 549 of file modelpc.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



9.4.4 Member Data Documentation

9.4.4.1 bitsUsed

```
int ModelPC::bitsUsed
```

bitsUsed Bits per byte used in pixel

Definition at line 85 of file modelpc.h.

9.4.4.2 curMode

```
int ModelPC::curMode
```

curMode Mode of en- or decryption

Definition at line 81 of file modelpc.h.

9.4.4.3 defaultJPHSDir

```
QString ModelPC::defaultJPHSDir
```

defaultJPHSDir Default JPHS directory

Definition at line 89 of file modelpc.h.

9.4.4.4 error

```
QString* ModelPC::error
```

error Current error

Definition at line 93 of file modelpc.h.

9.4.4.5 success

```
bool ModelPC::success
```

success Flag that true by default, but in case of error or cancelling of ProgressDialog it turns to false, which stops execution of ModelPC::circuit

Definition at line 69 of file modelpc.h.

9.4.4.6 version

```
long ModelPC::version
```

version Version of the class

Definition at line 73 of file modelpc.h.

9.4.4.7 versionString

```
QString ModelPC::versionString
```

versionString Version as string

Definition at line 77 of file modelpc.h.

The documentation for this class was generated from the following files:

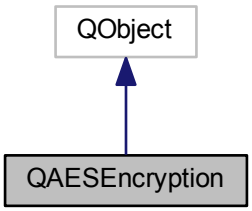
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/modelpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/modelpc.cpp

9.5 QAESEncryption Class Reference

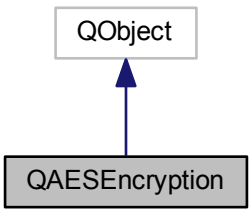
The QAESEncryption class Small and portable AES encryption class for Qt. Supports all key sizes - 128/192/256 bits - ECB, CBC, CFB and OFB modes. Class made entirely by bricke. Github: https://github.com/bricke/Qt-AES.

```
#include <qaesencryption.h>
```

Inheritance diagram for QAESEncryption:



Collaboration diagram for QAESEncryption:



Public Types

- enum Aes { AES_128, AES_192, AES_256 }

  The Aes enum AES Level AES Levels The class supports all AES key lenghts.
- enum Mode { ECB, CBC, CFB, OFB }

  The Mode enum AES Mode The class supports the following operating modes ECB CBC CFB OFB.
- enum Padding { ZERO, PKCS7, ISO }

  The Padding enum Padding By default the padding method is ISO, however, the class supports:

Public Member Functions

- QAESEncryption (QAESEncryption::Aes level, QAESEncryption::Mode mode, QAESEncryption::Padding padding=QAESEncryption::ISO)
- QByteArray encode (const QByteArray &rawText, const QByteArray &key, const QByteArray &iv=NULL)

  encode Encodes data with AES
- QByteArray decode (const QByteArray &rawText, const QByteArray &key, const QByteArray &iv=NULL)

  decode Decodes data with AES
- QByteArray removePadding (const QByteArray &rawText)

  RemovePadding Removes padding.
- QByteArray expandKey (const QByteArray &key)

  ExpandKey Expands the key.

Static Public Member Functions

- static QByteArray Crypt (QAESEncryption::Aes level, QAESEncryption::Mode mode, const QByteArray &rawText, const QByteArray &key, const QByteArray &iv=NULL, QAESEncryption::Padding padding=QAESEncryption::ISO)

  Crypt Static encode function.
- static QByteArray Decrypt (QAESEncryption::Aes level, QAESEncryption::Mode mode, const QByteArray &rawText, const QByteArray &key, const QByteArray &iv=NULL, QAESEncryption::Padding padding=QAESEncryption::ISO)

  Decrypt Static decode function.
- static QByteArray ExpandKey (QAESEncryption::Aes level, QAESEncryption::Mode mode, const QByteArray &key)

  ExpandKey Expands the key.
- static QByteArray RemovePadding (const QByteArray &rawText, QAESEncryption::Padding padding)

  RemovePadding Removes padding.

9.5.1 Detailed Description

The QAESEncryption class Small and portable AES encryption class for Qt. Supports all key sizes - 128/192/256 bits - ECB, CBC, CFB and OFB modes. Class made entirely by bricke. Github: https://github.com/bricke/Qt-AES.

Author

Bricke (Matteo B)

Definition at line 14 of file qaesencryption.h.

9.5.2 Member Enumeration Documentation

9.5.2.1 Aes

`enum QAESEncryption::Aes`

The Aes enum AES Level AES Levels The class supports all AES key lenghts.

AES_128 AES_192 AES_256

Enumerator

AES_128	
AES_192	
AES_256	

Definition at line 27 of file qaesencryption.h.

9.5.2.2 Mode

`enum QAESEncryption::Mode`

The Mode enum AES Mode The class supports the following operating modes ECB CBC CFB OFB.

Enumerator

ECB	
CBC	
CFB	
OFB	

Definition at line 40 of file qaesencryption.h.

9.5.2.3 Padding

`enum QAESEncryption::Padding`

The Padding enum Padding By default the padding method is ISO, however, the class supports:

ZERO PKCS7 ISO

Enumerator

ZERO	
PKCS7	
ISO	

Definition at line 55 of file qaesencryption.h.

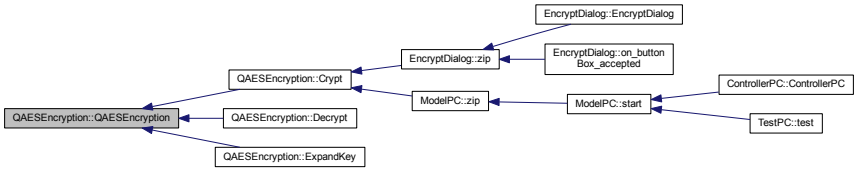
9.5.3 Constructor & Destructor Documentation

9.5.3.1 QAESEncryption()

```
QAESEncryption::QAESEncryption (
            QAESEncryption::Aes level,
            QAESEncryption::Mode mode,
            QAESEncryption::Padding padding = QAESEncryption::ISO )
```

Definition at line 67 of file qaesencryption.cpp.

Here is the caller graph for this function:



9.5.4 Member Function Documentation

9.5.4.1 Crypt()

```
QByteArray QAESEncryption::Crypt (
            QAESEncryption::Aes level,
            QAESEncryption::Mode mode,
            const QByteArray & rawText,
            const QByteArray & key,
            const QByteArray & iv = NULL,
            QAESEncryption::Padding padding = QAESEncryption::ISO )  [static]
```

Crypt Static encode function.

Parameters

level	AES level of encryption
mode	AES mode
rawText	Input data
key	Key for encrytion
iv	IV vector
padding	Padding

Returns

Returns encrypted data

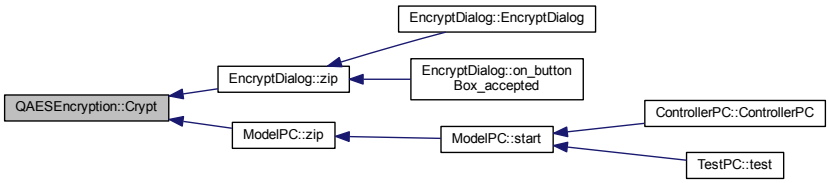
See also

QAESEncryption::encode, QAESEncryption::Decrypt

Definition at line 6 of file qaesencryption.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



9.5.4.2 decode()

```
QByteArray QAESEncryption::decode (
            const QByteArray & rawText,
            const QByteArray & key,
            const QByteArray & iv = NULL )
```

decode Decodes data with AES

Note

Basically the non-static method of QAESEncryption::Decrypt

Parameters

rawText	Input data
key	Key
iv	IV vector

Returns

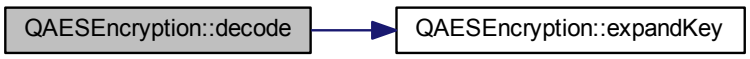
Returns decoded data

See also

QAESEncryption::Decrypt, QAESEncryption::encode

Definition at line 441 of file qaesencryption.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

9.5.4.3 Decrypt()

```
QByteArray QAESEncryption::Decrypt (
            QAESEncryption::Aes level,
            QAESEncryption::Mode mode,
            const QByteArray & rawText,
            const QByteArray & key,
            const QByteArray & iv = NULL,
            QAESEncryption::Padding padding = QAESEncryption::ISO )  [static]
```

Decrypt Static decode function.

Parameters

level	AES level of encryption
mode	AES mode
rawText	Encrypted data
key	Key for encrytion
iv	IV vector
padding	Padding

Returns

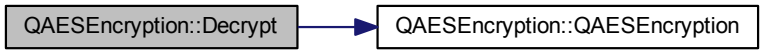
Returns Decrypted data

See also

QAESEncryption::decode, QAESEncryption::Crypt

Definition at line 12 of file qaesencryption.cpp.

Here is the call graph for this function:



9.5.4.4 encode()

```
QByteArray QAESEncryption::encode (
            const QByteArray & rawText,
            const QByteArray & key,
            const QByteArray & iv = NULL )
```

encode Encodes data with AES

Note

Basically the non-static method of QAESEncryption::Crypt

Parameters

rawText	Input data
key	Key
iv	IV vector

Returns

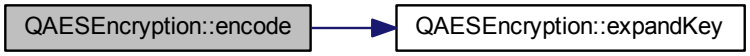
Returns encoded data

See also

QAESEncryption::Crypt, QAESEncryption::decode

Definition at line 391 of file qaesencryption.cpp.

Here is the call graph for this function:



9.5.4.5 ExpandKey()

```
QByteArray QAESEncryption::ExpandKey (
            QAESEncryption::Aes level,
            QAESEncryption::Mode mode,
            const QByteArray & key )  [static]
```

ExpandKey Expands the key.

Parameters

level	AES level
mode	AES Mode
key	key

Returns

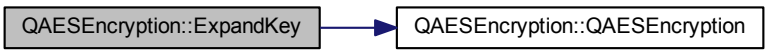
Returns expanded key (I guess)

See also

QAESEncryption::expandKey

Definition at line 18 of file qaesencryption.cpp.

Here is the call graph for this function:



9.5.4.6 expandKey()

```
QByteArray QAESEncryption::expandKey (
            const QByteArray & key )
```

ExpandKey Expands the key.

Note

Basically the non-static method of QAESEncryption::ExpandKey

Parameters

key	key

Returns

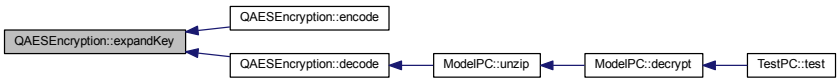
Returns expanded key (I guess)

See also

QAESEncryption::ExpandKey

Definition at line 132 of file qaesencryption.cpp.

Here is the caller graph for this function:



9.5.4.7 RemovePadding()

```
QByteArray QAESEncryption::RemovePadding (
            const QByteArray & rawText,
            QAESEncryption::Padding padding )  [static]
```

RemovePadding Removes padding.

Parameters

rawText	Input data
padding	Padding

Returns

Returns data with removed padding (I guess)

See also

QAESEncryption::removePadding

Definition at line 23 of file qaesencryption.cpp.

9.5.4.8 removePadding()

```
QByteArray QAESEncryption::removePadding (
            const QByteArray & rawText )
```

RemovePadding Removes padding.

Note

Basically the non-static method of QAESEncryption::RemovePadding

Parameters

rawText	Input data

Returns

Returns data with removed padding (I guess)

See also

QAESEncryption::RemovePadding

Definition at line 490 of file qaesencryption.cpp.

The documentation for this class was generated from the following files:

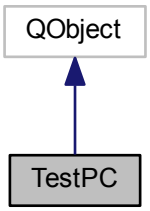
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/aes/qaesencryption.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/aes/qaesencryption.cpp

9.6 TestPC Class Reference

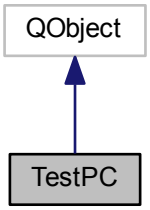
The TestPC class AutoTest for ModelPC Currently used in main.cpp.

```
#include <testpc.h>
```

Inheritance diagram for TestPC:



Collaboration diagram for TestPC:



Public Slots

- int startTest ()

  TestPC::startTest Starts the tests running.

Public Member Functions

- TestPC ()

  TestPC::TestPC Constructor.

Protected Slots

- bool test (QByteArray data, QImage rImage, QString expectedOutput="ok", int mode=0, QString key="", int bitsUsed=8)

  TestPC::test Function calling TestPC::model for tests.

9.6.1 Detailed Description

The TestPC class AutoTest for ModelPC Currently used in main.cpp.

Definition at line 22 of file testpc.h.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 TestPC()

```
TestPC::TestPC ( )
```

TestPC::TestPC Constructor.

Definition at line 5 of file testpc.cpp.

9.6.3 Member Function Documentation

9.6.3.1 startTest

```
int TestPC::startTest ( )  [slot]
```

TestPC::startTest Starts the tests running.

Note

Tests are configured in tests.json

Returns

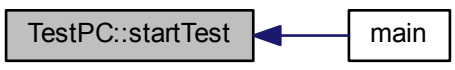
Returns success of all tests

See also

TestPC::autoTests

Definition at line 42 of file testpc.cpp.

Here is the caller graph for this function:



9.6.3.2 test

```
bool TestPC::test (
            QByteArray data,
            QImage rImage,
            QString expectedOutput = "ok",
            int mode = 0,
            QString key = "",
            int bitsUsed = 8 )  [protected], [slot]
```

TestPC::test Function calling TestPC::model for tests.

Parameters

data	Data for test
rImage	Image for test
expectedOutput	Expected output for test ("ok" if everything is well... ok, else errorcode from ErrorsDict.json)
mode	Mode for embedding
key	Key for for test
bitsUsed	Bits Used

Returns

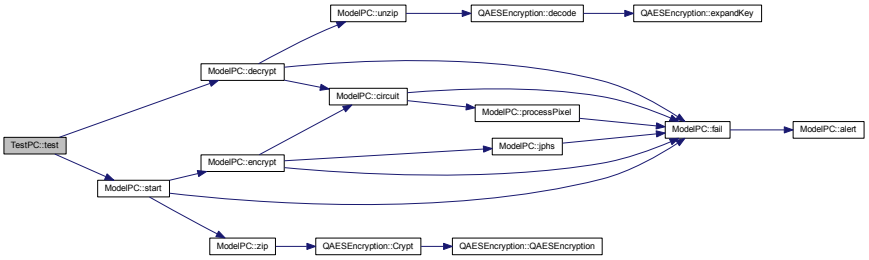
Returns if test is successful

See also

TestPC::autoTest, ModelPC::start, ModelPC::decrypt

Definition at line 18 of file testpc.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

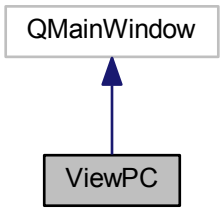
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/unit_tests/testpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/unit_tests/testpc.cpp

9.7 ViewPC Class Reference

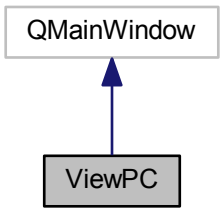
The ViewPC class View layer of the app. Controls EncryptDialog and ProgressDialog.

`#include <viewpc.h>`

Inheritance diagram for ViewPC:



Collaboration diagram for ViewPC:



Public Slots

- void alert (QString message, bool isWarning=false)

  ViewPC::alert Slot to create QMessageBox with message.
- void saveData (QByteArray Edata)

  ViewPC::saveData Slot to be called to save data using QFileDialog.
- void saveImage (QImage ∗image)

  ViewPC::saveImage Slot to be called to save image using QFileDialog.
- void setProgress (int val)

  ViewPC::setProgress Slot to set the value of the ProgressDialog (ViewPC::dialog).
- void abortCircuit ()

  ViewPC::abortCircuit Slot to close ProgressDialog (ViewPC::dialog)
- void setEncryptMode (bool encr)

  ViewPC::setEncryptMode Set the encrpt mode (ViewPC::isEncrypt)
- void setVersion (QString version)

  ViewPC::setVersion Set the version of the app from ControllerPC.

Signals

- encrypt (QByteArray data, QImage *image, int mode)

  encrypt Signal calling ModelPC::encrypt
- decrypt (QImage *_image)

  decrypt Signal calling ModelPC::decrypt
- abortModel ()

  abortModel Signal calling to stop ModelPC::circuit
- setBitsUsed (int bitsUsed)

  setBitsUsed Sets bits used in ModelPC
- setJPHSDir (QString dir)

  setJPHSPath Sets the default JPHS directory

Public Member Functions

- ViewPC (QWidget *parent=nullptr)
- ~ViewPC ()

Public Attributes

- QProgressDialog * dialog

  dialog ProgressDialog used.
- bool progressDialogClosed

  progressDialogClosed Flag, if dialog is closed.
- QJsonObject errorsDict

Protected Slots

- void on_fileButton_clicked ()

  ViewPC::on_fileButton_clicked Slot to be called, when according button is pressed.
- void on_startButton_clicked ()

  ViewPC::on_startButton_clicked Slot to be called, when Start Button is pressed.
- void on_actionAbout_triggered ()

  ViewPC::on_actionAbout_triggered Opens about page.
- void on_actionHelp_triggered ()

  ViewPC::on_actionHelp_triggered Opens online documentation.

9.7.1 Detailed Description

The ViewPC class View layer of the app. Controls EncryptDialog and ProgressDialog.

See also

ControllerPC, ModelPC, EncryptDialog

Definition at line 33 of file viewpc.h.

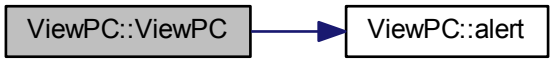
9.7.2 Constructor & Destructor Documentation

9.7.2.1 ViewPC()

```
ViewPC::ViewPC (
            QWidget * parent = nullptr )  [explicit]
```

Definition at line 4 of file viewpc.cpp.

Here is the call graph for this function:



9.7.2.2 ~ViewPC()

```
ViewPC::~ViewPC ( )
```

Definition at line 26 of file viewpc.cpp.

9.7.3 Member Function Documentation

9.7.3.1 abortCircuit

```
void ViewPC::abortCircuit ( )  [slot]
```

ViewPC::abortCircuit Slot to close ProgressDialog (ViewPC::dialog)

Definition at line 219 of file viewpc.cpp.

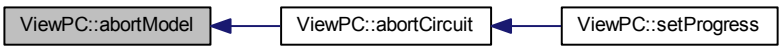
Here is the caller graph for this function:

9.7.3.2 abortModel

```
ViewPC::abortModel ( )  [signal]
```

abortModel Signal calling to stop ModelPC::circuit

Here is the caller graph for this function:



9.7.3.3 alert

```
void ViewPC::alert (
            QString message,
            bool isWarning = false )  [slot]
```

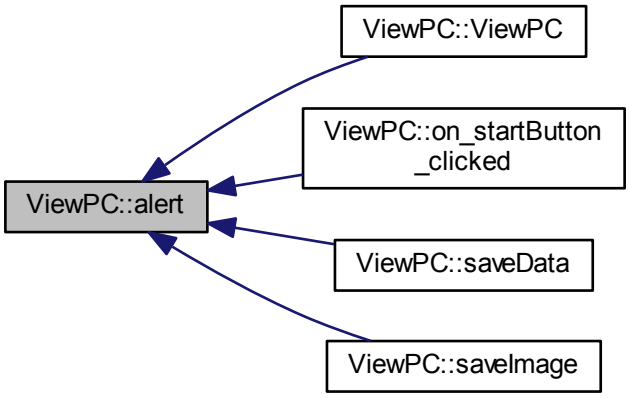
ViewPC::alert Slot to create QMessageBox with message.

Parameters

message	Message to be shown
isWarning	Flag, if message is critical.

Definition at line 133 of file viewpc.cpp.

Here is the caller graph for this function:



9.7.3.4 decrypt

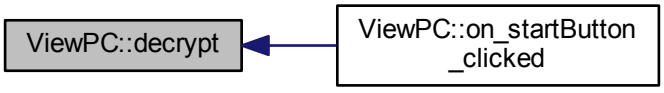
```
ViewPC::decrypt (
            QImage * _image )  [signal]
```

decrypt Signal calling ModelPC::decrypt

Parameters

_image	Image for decryption

Here is the caller graph for this function:



9.7.3.5 encrypt

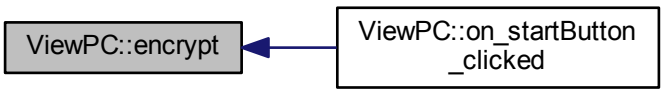
```
ViewPC::encrypt (
            QByteArray data,
            QImage * image,
            int mode )  [signal]
```

encrypt Signal calling ModelPC::encrypt

Parameters

data	Data to write
image	Image to be encrypted into.
mode	Mode of encryption

Here is the caller graph for this function:



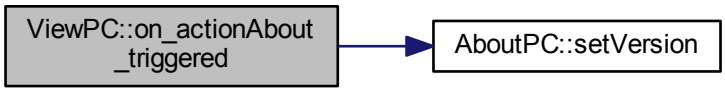
9.7.3.6 on_actionAbout_triggered

```
void ViewPC::on_actionAbout_triggered ( )  [protected], [slot]
```

ViewPC::on_actionAbout_triggered Opens about page.

Definition at line 254 of file viewpc.cpp.

Here is the call graph for this function:



9.7.3.7 on_actionHelp_triggered

```
void ViewPC::on_actionHelp_triggered ( )  [protected], [slot]
```

ViewPC::on_actionHelp_triggered Opens online documentation.

Definition at line 264 of file viewpc.cpp.

9.7.3.8 on_fileButton_clicked

```
void ViewPC::on_fileButton_clicked ( )  [protected], [slot]
```

ViewPC::on_fileButton_clicked Slot to be called, when according button is pressed.

Definition at line 45 of file viewpc.cpp.

9.7.3.9 on_startButton_clicked

```
void ViewPC::on_startButton_clicked ( )  [protected], [slot]
```

ViewPC::on_startButton_clicked Slot to be called, when Start Button is pressed.

9.7.4 Encrypting

If Encrypting mode is active the data from text browser or from file from file selector will be opened and checked in size.

Note

File size limit is 16MB

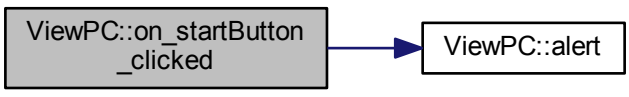
Then the EncryptDialog opens and image and key is selected. Then the ViewPC::encrypt signal is called to start ModelPC::encrypt

9.7.5 Decrypting

Else, the image from file selector is transmitted to ModelPC::decrypt

Definition at line 67 of file viewpc.cpp.

Here is the call graph for this function:



9.7.5.1 saveData

```
void ViewPC::saveData (
            QByteArray Edata )  [slot]
```

ViewPC::saveData Slot to be called to save data using QFileDialog.

Parameters

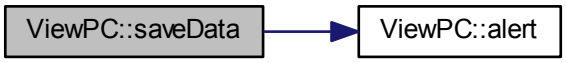
Edata	Encrypted data to be saved.

See also

ModelPC::encrypt

Definition at line 154 of file viewpc.cpp.

Here is the call graph for this function:



9.7.5.2 saveImage

```
void ViewPC::saveImage (
            QImage * image )  [slot]
```

ViewPC::saveImage Slot to be called to save image using QFileDialog.

Parameters

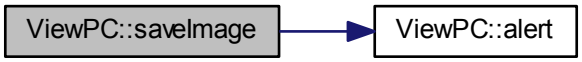
image	Image to be saved.

See also

ModelPC::decrypt

Definition at line 175 of file viewpc.cpp.

Here is the call graph for this function:



9.7.5.3 setBitsUsed

```
ViewPC::setBitsUsed (
            int bitsUsed )  [signal]
```

setBitsUsed Sets bits used in ModelPC

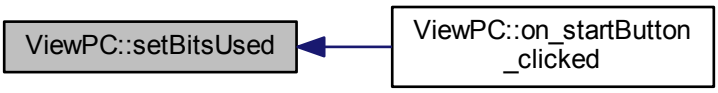
Parameters

bitsUsed	The new value

See also

ModelPC::bitsUsed

Here is the caller graph for this function:



9.7.5.4 setEncryptMode

```
void ViewPC::setEncryptMode (
            bool encr )  [slot]
```

ViewPC::setEncryptMode Set the encrpt mode (ViewPC::isEncrypt)

Parameters

encr	

Definition at line 232 of file viewpc.cpp.

9.7.5.5 setJPHSDir

```
ViewPC::setJPHSDir (
            QString dir )  [signal]
```

setJPHSPath Sets the default JPHS directory

Parameters

dir	Directory

9.7.5.6 setProgress

```
void ViewPC::setProgress (
            int val )  [slot]
```

ViewPC::setProgress Slot to set the value of the ProgressDialog (ViewPC::dialog).

Parameters

val	New value of the dialog. If -1, creates ProgressDialog, if 101 closes the dialog.

See also

ViewPC::abortCircuit(), ModelPC::setProgress()

Definition at line 193 of file viewpc.cpp.

Here is the call graph for this function:

9.7.5.7 setVersion

```
void ViewPC::setVersion (
            QString version )  [slot]
```

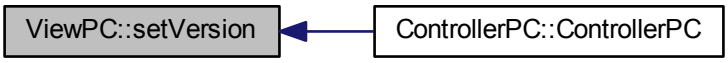
ViewPC::setVersion Set the version of the app from ControllerPC.

Parameters

version	Version as QString

Definition at line 241 of file viewpc.cpp.

Here is the caller graph for this function:



9.7.6 Member Data Documentation

9.7.6.1 dialog

```
QProgressDialog* ViewPC::dialog
```

dialog ProgressDialog used.

See also

ViewPC::setProgress, ViewPC::cancel, ModelPC::setProgress

Definition at line 96 of file viewpc.h.

9.7.6.2 errorsDict

```
QJsonObject ViewPC::errorsDict
```

Definition at line 102 of file viewpc.h.

9.7.6.3 progressDialogClosed

```
bool ViewPC::progressDialogClosed
```

progressDialogClosed Flag, if dialog is closed.

See also

ViewPC::abortCircuit, ViewPC::setProgress

Definition at line 101 of file viewpc.h.

The documentation for this class was generated from the following files:

- C:/Users/salex/Documents/GitHub/PictureCrypt/src/viewpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/src/viewpc.cpp

Chapter 10

File Documentation

10.1 C:/Users/salex/Documents/GitHub/PictureCrypt/CODE_OF_CONDUCT.md File Reference

10.2 C:/Users/salex/Documents/GitHub/PictureCrypt/CODE_OF_CONDUCT.md

```
00001 # Contributor Covenant Code of Conduct
00002
00003 ## Our Pledge
00004
00005 In the interest of fostering an open and welcoming environment, we as contributors and maintainers
       pledge to making participation in our project and our community a harassment-free experience for everyone,
       regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience,
       nationality, personal appearance, race, religion, or sexual identity and orientation.
00006
00007 ## Our Standards
00008
00009 Examples of behavior that contributes to creating a positive environment include:
00010
00011 * Using welcoming and inclusive language
00012 * Being respectful of differing viewpoints and experiences
00013 * Gracefully accepting constructive criticism
00014 * Focusing on what is best for the community
00015 * Showing empathy towards other community members
00016
00017 Examples of unacceptable behavior by participants include:
00018
00019 * The use of sexualized language or imagery and unwelcome sexual attention or advances
00020 * Trolling, insulting/derogatory comments, and personal or political attacks
00021 * Public or private harassment
00022 * Publishing others' private information, such as a physical or electronic address, without explicit
       permission
00023 * Other conduct which could reasonably be considered inappropriate in a professional setting
00024
00025 ## Our Responsibilities
00026
00027 Project maintainers are responsible for clarifying the standards of acceptable behavior and are
       expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.
00028
00029 Project maintainers have the right and responsibility to remove, edit, or reject comments, commits,
       code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban
       temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening,
       offensive, or harmful.
00030
00031 ## Scope
00032
00033 This Code of Conduct applies both within project spaces and in public spaces when an individual is
       representing the project or its community. Examples of representing a project or community include using an
       official project e-mail address, posting via an official social media account, or acting as an appointed
       representative at an online or offline event. Representation of a project may be further defined and clarified by
       project maintainers.
00034
00035 ## Enforcement
00036
00037 Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the
```

```
      project team at a.kovrigin0@gmail.com. The project team will review and investigate all complaints, and will
      respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain
      confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies
      may be posted separately.
00038
00039 Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary
      or permanent repercussions as determined by other members of the project's leadership.
00040
00041 ## Attribution
00042
00043 This Code of Conduct is adapted from the [Contributor Covenant][homepage], version 1.4, available at
      [http://contributor-covenant.org/version/1/4][version]
00044
00045 [homepage]: http://contributor-covenant.org
00046 [version]: http://contributor-covenant.org/version/1/4/
```

10.3 C:/Users/salex/Documents/GitHub/PictureCrypt/README.md File Reference

10.4 C:/Users/salex/Documents/GitHub/PictureCrypt/README.md

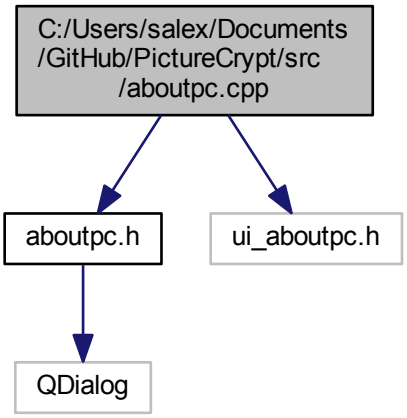
```
00001 # PictureCrypt
00002 Make your pictures crypted.
00003
00004
00005 ## About
00006 Project is made only using QT.
00007 [QAESEncryption](http://github.com/bricke/Qt-AES) by bricke was also used.
00008 MVC pattern used.
00009 PictureCrypt project is UI based, the model contains all buisness logic and can work as standalone
      class.
00010
00011 ## External use
00012 ModelPC class can be used externally (without UI)
00013 ```
00014 #include <modelpc.h>
00015 #include <testpc.h>
00016 #include <QByteArray>
00017 #include <QImage>
00018
00019 ...
00020 // Testing the ModelPC
00021 TestPC testing;
00022 if(!testing.startTest())
00023    return;
00024 ModelPC * model = new ModelPC();
00025
00026 // Embedding
00027 QImage * resultImage = model->start(QByteArray data, // Data to be embedded
00028                                     QImage *image, // Image for embedding
00029                                     int mode = 0, // Mode of embedding
00030                                     QString key = "", // Key for extra-encryption (if empty, key will
      be generated automatically)
00031                                     int bitsUsed = 8, // Bits per Byte used (better explaination
      ModelPC::bitsUsed)
00032                                     QString *error = nullptr); // Error output, if everything is ok,
      error will be "ok"
00033 if(*error != "ok)
00034    return;
00035 // Note *error is just a code of error (like "muchdata", dictionary of error codes is also available
      on github.
00036
00037 // De-embedding
00038 QByteArray output = model->decrypt(QImage * image, // Image with hidden data
00039                                    QString *_error = nullptr); // Error output
00040 if(data == output)
00041    qDebug() << "Great success!";
00042
00043 ```
00044
00045 ## Available modes of embedding
00046 * 0 - Standard, created by me
00047 * 1 - JPHS, requires manually installed JPHS and specified directory (not currently available).
00048
00049 ## Documentation
00050 Doxygen Documentation avaible [here](https://waleko.github.io/PictureCrypt)
00051
00052
00053 ## Dependencies
00054 * qtcore
```

```
00055
00056 ## Used works from other people
00057  * [QAESEncryption](https://github.com/bricke/Qt-AES) by bricke
00058
00059 ## Contact
00060 Question or suggestions are welcome!
00061 Please use the GitHub issue tracking to report suggestions or issues.
00062 Email me a.kovrigin0@gmail.com and visit my site http://alex.unaux.com
00063
00064 ## License
00065 This software is provided under the [UNLICENSE](http://unlicense.org/)
```

10.5 C:/Users/salex/Documents/GitHub/PictureCrypt/src/aboutpc.cpp File Reference

```
#include "aboutpc.h"
#include "ui_aboutpc.h"
```

Include dependency graph for aboutpc.cpp:



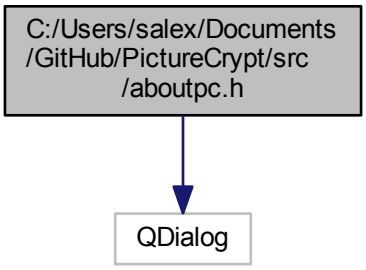
10.6 aboutpc.cpp

```
00001 #include "aboutpc.h"
00002 #include "ui_aboutpc.h"
00003
00004 AboutPC::AboutPC(QWidget *parent) :
00005     QDialog(parent),
00006     ui(new Ui::AboutPC)
00007 {
00008     ui->setupUi(this);
00009 }
00010
00011 AboutPC::~AboutPC()
00012 {
00013     delete ui;
00014 }
00019 void AboutPC::setVersion(QString version)
00020 {
00021     ui->versionLabel->setText("Version " + version);
00022 }
```

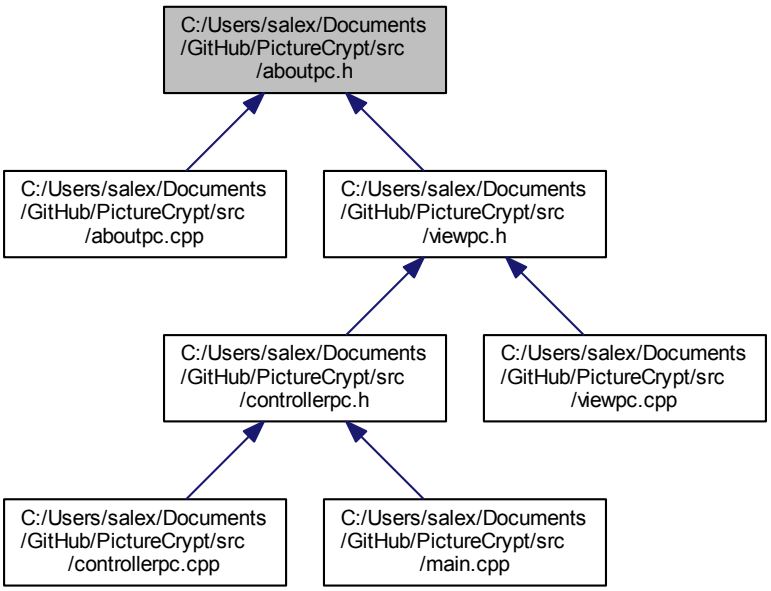
10.7 C:/Users/salex/Documents/GitHub/PictureCrypt/src/aboutpc.h File Reference

```
#include <QDialog>
```

Include dependency graph for aboutpc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class AboutPC

  The AboutPC class The About Page dialog.

Namespaces

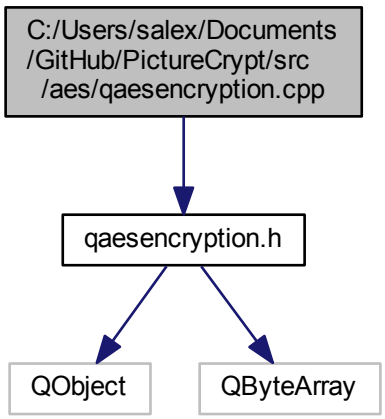
- Ui

10.8 aboutpc.h

```
00001 #ifndef ABOUTPC_H
00002 #define ABOUTPC_H
00003
00004 #include <QDialog>
00005
00006 namespace Ui {
00007 class AboutPC;
00008 }
00012 class AboutPC : public QDialog
00013 {
00014     Q_OBJECT
00015
00016 public:
00017     explicit AboutPC(QWidget *parent = 0);
00018     ~AboutPC();
00019     void setVersion(QString version);
00020
00021 private:
00022     Ui::AboutPC *ui;
00023 };
00024
00025 #endif // ABOUTPC_H
```

10.9 C:/Users/salex/Documents/GitHub/PictureCrypt/src/aes/qaesencryption.cpp File Reference

`#include "qaesencryption.h"`
Include dependency graph for qaesencryption.cpp:



Functions

- quint8 xTime (quint8 x)
- quint8 multiply (quint8 x, quint8 y)

10.9.1 Function Documentation

10.9.1.1 multiply()

```
quint8 multiply (
            quint8 x,
            quint8 y )  [inline]
```

Definition at line 57 of file qaesencryption.cpp.

Here is the call graph for this function:

10.9.1.2 xTime()

```
quint8 xTime (
            quint8 x )  [inline]
```

Definition at line 53 of file qaesencryption.cpp.

Here is the caller graph for this function:

10.10 qaesencryption.cpp

```
00001 #include "qaesencryption.h"
00002
00003 /*
00004  * Static Functions
00005  * */
00006 QByteArray QAESEncryption::Crypt(QAESEncryption::Aes level,
      QAESEncryption::Mode mode, const QByteArray &rawText,
00007                                   const QByteArray &key, const QByteArray &iv,
      QAESEncryption::Padding padding)
00008 {
00009     return QAESEncryption(level, mode, padding).encode(rawText,
      key, iv);
00010 }
00011
00012 QByteArray QAESEncryption::Decrypt(QAESEncryption::Aes level,
      QAESEncryption::Mode mode, const QByteArray &rawText,
00013                                     const QByteArray &key, const QByteArray &iv,
      QAESEncryption::Padding padding)
00014 {
00015      return QAESEncryption(level, mode, padding).decode(rawText,
      key, iv);
00016 }
00017
00018 QByteArray QAESEncryption::ExpandKey(
      QAESEncryption::Aes level, QAESEncryption::Mode
      mode, const QByteArray &key)
00019 {
00020      return QAESEncryption(level, mode).expandKey(key);
00021 }
00022
00023 QByteArray QAESEncryption::RemovePadding(const QByteArray &rawText,
      QAESEncryption::Padding padding)
00024 {
00025     QByteArray ret(rawText);
00026     switch (padding)
00027     {
00028     case Padding::ZERO:
00029         //Works only if the last byte of the decoded array is not zero
00030         while (ret.at(ret.length()-1) == 0x00)
00031             ret.remove(ret.length()-1, 1);
00032         break;
00033     case Padding::PKCS7:
00034         ret.remove(ret.length() - ret.at(ret.length()-1), ret.at(ret.length()-1));
00035         break;
00036     case Padding::ISO:
00037         ret.truncate(ret.lastIndexOf(0x80));
00038         break;
00039     default:
00040         //do nothing
00041         break;
00042     }
00043     return ret;
00044 }
00045 /*
00046  * End Static function declarations
00047  * */
00048
00049 /*
00050  * Inline Functions
00051  * */
00052
00053 inline quint8 xTime(quint8 x){
00054   return ((x<<1) ^ (((x>>7) & 1) * 0x1b));
00055 }
00056
00057 inline quint8 multiply(quint8 x, quint8 y){
00058   return (((y & 1) * x) ^ ((y>>1 & 1) * xTime(x)) ^ ((y>>2 & 1) * xTime(
      xTime(x))) ^ ((y>>3 & 1)
00059             * xTime(xTime(xTime(x)))) ^ ((y>>4 & 1) * xTime(
      xTime(xTime(xTime(x))))));
00060 }
00061
00062 /*
00063  * End Inline functions
00064  * */
00065
00066
00067 QAESEncryption::QAESEncryption(Aes level, Mode
      mode,
00068                                Padding padding)
00069     : m_nb(4), m_blocklen(16), m_level(level), m_mode(mode), m_padding(padding)
00070 {
00071     m_state = NULL;
00072
```

```
00073     switch (level)
00074     {
00075     case AES_128: {
00076         AES128 aes;
00077         m_nk = aes.nk;
00078         m_keyLen = aes.keylen;
00079         m_nr = aes.nr;
00080         m_expandedKey = aes.expandedKey;
00081         }
00082         break;
00083     case AES_192: {
00084         AES192 aes;
00085         m_nk = aes.nk;
00086         m_keyLen = aes.keylen;
00087         m_nr = aes.nr;
00088         m_expandedKey = aes.expandedKey;
00089         }
00090         break;
00091     case AES_256: {
00092         AES256 aes;
00093         m_nk = aes.nk;
00094         m_keyLen = aes.keylen;
00095         m_nr = aes.nr;
00096         m_expandedKey = aes.expandedKey;
00097         }
00098         break;
00099     default: {
00100         AES128 aes;
00101         m_nk = aes.nk;
00102         m_keyLen = aes.keylen;
00103         m_nr = aes.nr;
00104         m_expandedKey = aes.expandedKey;
00105         }
00106         break;
00107     }
00108
00109 }
00110 QByteArray QAESEncryption::getPadding(int currSize, int alignment)
00111 {
00112     int size = (alignment - currSize % alignment) % alignment;
00113     if (size == 0) return QByteArray();
00114     switch(m_padding)
00115     {
00116     case Padding::ZERO:
00117         return QByteArray(size, 0x00);
00118         break;
00119     case Padding::PKCS7:
00120         return QByteArray(size,size);
00121         break;
00122     case Padding::ISO:
00123         return QByteArray (size-1, 0x00).prepend(0x80);
00124         break;
00125     default:
00126         return QByteArray(size, 0x00);
00127         break;
00128     }
00129     return QByteArray(size, 0x00);
00130 }
00131
00132 QByteArray QAESEncryption::expandKey(const QByteArray &
      key)
00133 {
00134   int i, k;
00135   quint8 tempa[4]; // Used for the column/row operations
00136   QByteArray roundKey(key);
00137
00138   // The first round key is the key itself.
00139   // ...
00140
00141   // All other round keys are found from the previous round keys.
00142   //i == Nk
00143   for(i = m_nk; i < m_nb * (m_nr + 1); i++)
00144   {
00145     tempa[0] = (quint8) roundKey.at((i-1) * 4 + 0);
00146     tempa[1] = (quint8) roundKey.at((i-1) * 4 + 1);
00147     tempa[2] = (quint8) roundKey.at((i-1) * 4 + 2);
00148     tempa[3] = (quint8) roundKey.at((i-1) * 4 + 3);
00149
00150     if (i % m_nk == 0)
00151     {
00152         // This function shifts the 4 bytes in a word to the left once.
00153         // [a0,a1,a2,a3] becomes [a1,a2,a3,a0]
00154
00155         // Function RotWord()
00156         k = tempa[0];
00157         tempa[0] = tempa[1];
00158         tempa[1] = tempa[2];
```

```
00159         tempa[2] = tempa[3];
00160         tempa[3] = k;
00161 
00162         // Function Subword()
00163         tempa[0] = getSBoxValue(tempa[0]);
00164         tempa[1] = getSBoxValue(tempa[1]);
00165         tempa[2] = getSBoxValue(tempa[2]);
00166         tempa[3] = getSBoxValue(tempa[3]);
00167 
00168         tempa[0] =  tempa[0] ^ Rcon[i/m_nk];
00169     }
00170     if (m_level == AES_256 && i % m_nk == 4)
00171     {
00172         // Function Subword()
00173         tempa[0] = getSBoxValue(tempa[0]);
00174         tempa[1] = getSBoxValue(tempa[1]);
00175         tempa[2] = getSBoxValue(tempa[2]);
00176         tempa[3] = getSBoxValue(tempa[3]);
00177     }
00178     roundKey.insert(i * 4 + 0, (quint8) roundKey.at((i - m_nk) * 4 + 0) ^ tempa[0]);
00179     roundKey.insert(i * 4 + 1, (quint8) roundKey.at((i - m_nk) * 4 + 1) ^ tempa[1]);
00180     roundKey.insert(i * 4 + 2, (quint8) roundKey.at((i - m_nk) * 4 + 2) ^ tempa[2]);
00181     roundKey.insert(i * 4 + 3, (quint8) roundKey.at((i - m_nk) * 4 + 3) ^ tempa[3]);
00182   }
00183   return roundKey;
00184 }
00185 
00186 // This function adds the round key to state.
00187 // The round key is added to the state by an XOR function.
00188 void QAESEncryption::addRoundKey(const quint8 round, const QByteArray expKey)
00189 {
00190   QByteArray::iterator it = m_state->begin();
00191   for(int i=0; i < 16; ++i)
00192       it[i] = (quint8) it[i] ^ (quint8) expKey.at(round * m_nb * 4 + (i/4) * m_nb + (i%4));
00193 }
00194 
00195 // The SubBytes Function Substitutes the values in the
00196 // state matrix with values in an S-box.
00197 void QAESEncryption::subBytes()
00198 {
00199   QByteArray::iterator it = m_state->begin();
00200   for(int i = 0; i < 16; i++)
00201     it[i] = getSBoxValue((quint8) it[i]);
00202 }
00203 
00204 // The ShiftRows() function shifts the rows in the state to the left.
00205 // Each row is shifted with different offset.
00206 // Offset = Row number. So the first row is not shifted.
00207 void QAESEncryption::shiftRows()
00208 {
00209     QByteArray::iterator it = m_state->begin();
00210     quint8 temp;
00211     //Keep in mind that QByteArray is column-driven!!
00212 
00213      //Shift 1 to left
00214     temp   = (quint8)it[1];
00215     it[1]  = (quint8)it[5];
00216     it[5]  = (quint8)it[9];
00217     it[9]  = (quint8)it[13];
00218     it[13] = (quint8)temp;
00219 
00220     //Shift 2 to left
00221     temp   = (quint8)it[2];
00222     it[2]  = (quint8)it[10];
00223     it[10] = (quint8)temp;
00224     temp   = (quint8)it[6];
00225     it[6]  = (quint8)it[14];
00226     it[14] = (quint8)temp;
00227 
00228     //Shift 3 to left
00229     temp   = (quint8)it[3];
00230     it[3]  = (quint8)it[15];
00231     it[15] = (quint8)it[11];
00232     it[11] = (quint8)it[7];
00233     it[7]  = (quint8)temp;
00234 }
00235 
00236 // MixColumns function mixes the columns of the state matrix
00237 //optimized!!
00238 void QAESEncryption::mixColumns()
00239 {
00240   QByteArray::iterator it = m_state->begin();
00241   quint8 tmp, tm, t;
00242 
00243   for(int i = 0; i < 16; i += 4){
00244     t       = (quint8)it[i];
00245     tmp     =  (quint8)it[i] ^ (quint8)it[i+1] ^ (quint8)it[i+2] ^ (quint8)it[i+3] ;
```

```
00246
00247     tm      = xTime( (quint8)it[i] ^ (quint8)it[i+1] );
00248     it[i]   = (quint8)it[i] ^ (quint8)tm ^ (quint8)tmp;
00249
00250     tm      = xTime( (quint8)it[i+1] ^ (quint8)it[i+2]);
00251     it[i+1] = (quint8)it[i+1] ^ (quint8)tm ^ (quint8)tmp;
00252
00253     tm      = xTime( (quint8)it[i+2] ^ (quint8)it[i+3]);
00254     it[i+2] =(quint8)it[i+2] ^ (quint8)tm ^ (quint8)tmp;
00255
00256     tm      = xTime((quint8)it[i+3] ^ (quint8)t);
00257     it[i+3] =(quint8)it[i+3] ^ (quint8)tm ^ (quint8)tmp;
00258   }
00259 }
00260
00261 // MixColumns function mixes the columns of the state matrix.
00262 // The method used to multiply may be difficult to understand for the inexperienced.
00263 // Please use the references to gain more information.
00264 void QAESEncryption::invMixColumns()
00265 {
00266   QByteArray::iterator it = m_state->begin();
00267   quint8 a,b,c,d;
00268   for(int i = 0; i < 16; i+=4){
00269     a = (quint8) it[i];
00270     b = (quint8) it[i+1];
00271     c = (quint8) it[i+2];
00272     d = (quint8) it[i+3];
00273
00274     it[i]   = (quint8) (multiply(a, 0x0e) ^ multiply(b, 0x0b) ^
      multiply(c, 0x0d) ^ multiply(d, 0x09));
00275     it[i+1] = (quint8) (multiply(a, 0x09) ^ multiply(b, 0x0e) ^
      multiply(c, 0x0b) ^ multiply(d, 0x0d));
00276     it[i+2] = (quint8) (multiply(a, 0x0d) ^ multiply(b, 0x09) ^
      multiply(c, 0x0e) ^ multiply(d, 0x0b));
00277     it[i+3] = (quint8) (multiply(a, 0x0b) ^ multiply(b, 0x0d) ^
      multiply(c, 0x09) ^ multiply(d, 0x0e));
00278   }
00279 }
00280
00281 // The SubBytes Function Substitutes the values in the
00282 // state matrix with values in an S-box.
00283 void QAESEncryption::invSubBytes()
00284 {
00285     QByteArray::iterator it = m_state->begin();
00286     for(int i = 0; i < 16; ++i)
00287         it[i] = getSBoxInvert((quint8) it[i]);
00288 }
00289
00290 void QAESEncryption::invShiftRows()
00291 {
00292     QByteArray::iterator it = m_state->begin();
00293     uint8_t temp;
00294
00295     //Keep in mind that QByteArray is column-driven!!
00296
00297     //Shift 1 to right
00298     temp   = (quint8)it[13];
00299     it[13] = (quint8)it[9];
00300     it[9]  = (quint8)it[5];
00301     it[5]  = (quint8)it[1];
00302     it[1]  = (quint8)temp;
00303
00304     //Shift 2
00305     temp   = (quint8)it[10];
00306     it[10] = (quint8)it[2];
00307     it[2]  = (quint8)temp;
00308     temp   = (quint8)it[14];
00309     it[14] = (quint8)it[6];
00310     it[6]  = (quint8)temp;
00311
00312     //Shift 3
00313     temp   = (quint8)it[15];
00314     it[15] = (quint8)it[3];
00315     it[3]  = (quint8)it[7];
00316     it[7]  = (quint8)it[11];
00317     it[11] = (quint8)temp;
00318 }
00319
00320 QByteArray QAESEncryption::byteXor(const QByteArray &a, const QByteArray &b)
00321 {
00322   QByteArray::const_iterator it_a = a.begin();
00323   QByteArray::const_iterator it_b = b.begin();
00324   QByteArray ret;
00325
00326   //for(int i = 0; i < m_blocklen; i++)
00327   for(int i = 0; i < std::min(a.size(), b.size()); i++)
00328       ret.insert(i,it_a[i] ^ it_b[i]);
```

```
00329
00330   return ret;
00331 }
00332
00333 // Cipher is the main function that encrypts the PlainText.
00334 QByteArray QAESEncryption::cipher(const QByteArray &expKey, const QByteArray &in)
00335 {
00336
00337   //m_state is the input buffer...
00338   QByteArray output(in);
00339   m_state = &output;
00340
00341   // Add the First round key to the state before starting the rounds.
00342   addRoundKey(0, expKey);
00343
00344   // There will be Nr rounds.
00345   // The first Nr-1 rounds are identical.
00346   // These Nr-1 rounds are executed in the loop below.
00347   for(quint8 round = 1; round < m_nr; ++round){
00348     subBytes();
00349     shiftRows();
00350     mixColumns();
00351     addRoundKey(round, expKey);
00352   }
00353
00354   // The last round is given below.
00355   // The MixColumns function is not here in the last round.
00356   subBytes();
00357   shiftRows();
00358   addRoundKey(m_nr, expKey);
00359
00360   return output;
00361 }
00362
00363 QByteArray QAESEncryption::invCipher(const QByteArray &expKey, const QByteArray &in)
00364 {
00365     //m_state is the input buffer.... handle it!
00366     QByteArray output(in);
00367     m_state = &output;
00368
00369     // Add the First round key to the state before starting the rounds.
00370     addRoundKey(m_nr, expKey);
00371
00372     // There will be Nr rounds.
00373     // The first Nr-1 rounds are identical.
00374     // These Nr-1 rounds are executed in the loop below.
00375     for(quint8 round=m_nr-1; round>0 ; round--){
00376         invShiftRows();
00377         invSubBytes();
00378         addRoundKey(round, expKey);
00379         invMixColumns();
00380     }
00381
00382     // The last round is given below.
00383     // The MixColumns function is not here in the last round.
00384     invShiftRows();
00385     invSubBytes();
00386     addRoundKey(0, expKey);
00387
00388     return output;
00389 }
00390
00391 QByteArray QAESEncryption::encode(const QByteArray &rawText, const QByteArray &
      key, const QByteArray &iv)
00392 {
00393     if (m_mode >= CBC && (iv.isNull() || iv.size() != m_blocklen))
00394        return QByteArray();
00395
00396     QByteArray ret;
00397     QByteArray expandedKey = expandKey(key);
00398     QByteArray alignedText(rawText);
00399
00400     //Fill array with padding
00401     alignedText.append(getPadding(rawText.size(), m_blocklen));
00402
00403     switch(m_mode)
00404     {
00405     case ECB:
00406         for(int i=0; i < alignedText.size(); i+= m_blocklen)
00407             ret.append(cipher(expandedKey, alignedText.mid(i, m_blocklen)));
00408         break;
00409     case CBC: {
00410             QByteArray ivTemp(iv);
00411             for(int i=0; i < alignedText.size(); i+= m_blocklen) {
00412                 alignedText.replace(i, m_blocklen, byteXor(alignedText.mid(i, m_blocklen),ivTemp));
00413                 ret.append(cipher(expandedKey, alignedText.mid(i, m_blocklen)));
00414                 ivTemp = ret.mid(i, m_blocklen);
```

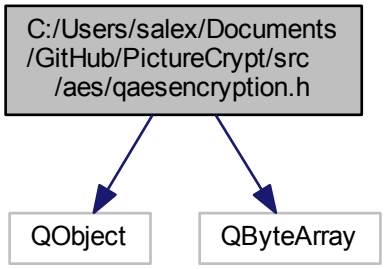
```
00415             }
00416         }
00417         break;
00418     case CFB: {
00419             ret.append(byteXor(alignedText.left(m_blocklen), cipher(expandedKey, iv)));
00420             for(int i=0; i < alignedText.size(); i+= m_blocklen) {
00421                 if (i+m_blocklen < alignedText.size())
00422                     ret.append(byteXor(alignedText.mid(i+m_blocklen, m_blocklen),
00423                                        cipher(expandedKey, ret.mid(i, m_blocklen))));
00424             }
00425         }
00426         break;
00427     case OFB: {
00428             QByteArray ofbTemp;
00429             ofbTemp.append(cipher(expandedKey, iv));
00430             for (int i=m_blocklen; i < alignedText.size(); i += m_blocklen){
00431                 ofbTemp.append(cipher(expandedKey, ofbTemp.right(m_blocklen)));
00432             }
00433             ret.append(byteXor(alignedText, ofbTemp));
00434         }
00435         break;
00436     default: break;
00437     }
00438     return ret;
00439 }
00440
00441 QByteArray QAESEncryption::decode(const QByteArray &rawText, const QByteArray &
      key, const QByteArray &iv)
00442 {
00443     if (m_mode >= CBC && (iv.isNull() || iv.size() != m_blocklen))
00444        return QByteArray();
00445
00446     QByteArray ret;
00447     QByteArray expandedKey = expandKey(key);
00448
00449     switch(m_mode)
00450     {
00451     case ECB:
00452         for(int i=0; i < rawText.size(); i+= m_blocklen)
00453             ret.append(invCipher(expandedKey, rawText.mid(i, m_blocklen)));
00454         break;
00455     case CBC: {
00456             QByteArray ivTemp(iv);
00457             for(int i=0; i < rawText.size(); i+= m_blocklen){
00458                 ret.append(invCipher(expandedKey, rawText.mid(i, m_blocklen)));
00459                 ret.replace(i, m_blocklen, byteXor(ret.mid(i, m_blocklen),ivTemp));
00460                 ivTemp = rawText.mid(i, m_blocklen);
00461             }
00462         }
00463         break;
00464     case CFB: {
00465             ret.append(byteXor(rawText.mid(0, m_blocklen), cipher(expandedKey, iv)));
00466             for(int i=0; i < rawText.size(); i+= m_blocklen){
00467                 if (i+m_blocklen < rawText.size()) {
00468                     ret.append(byteXor(rawText.mid(i+m_blocklen, m_blocklen),
00469                                        cipher(expandedKey, rawText.mid(i, m_blocklen))));
00470                 }
00471             }
00472         }
00473         break;
00474     case OFB: {
00475         QByteArray ofbTemp;
00476         ofbTemp.append(cipher(expandedKey, iv));
00477         for (int i=m_blocklen; i < rawText.size(); i += m_blocklen){
00478             ofbTemp.append(cipher(expandedKey, ofbTemp.right(m_blocklen)));
00479         }
00480         ret.append(byteXor(rawText, ofbTemp));
00481     }
00482         break;
00483     default:
00484         //do nothing
00485         break;
00486     }
00487     return ret;
00488 }
00489
00490 QByteArray QAESEncryption::removePadding(const QByteArray &rawText)
00491 {
00492     QByteArray ret(rawText);
00493     switch (m_padding)
00494     {
00495     case Padding::ZERO:
00496         //Works only if the last byte of the decoded array is not zero
00497         while (ret.at(ret.length()-1) == 0x00)
00498             ret.remove(ret.length()-1, 1);
00499         break;
00500     case Padding::PKCS7:
```

```
00501         ret.remove(ret.length() - ret.at(ret.length()-1), ret.at(ret.length()-1));
00502         break;
00503     case Padding::ISO:
00504         ret.truncate(ret.lastIndexOf(0x80));
00505         break;
00506     default:
00507         //do nothing
00508         break;
00509     }
00510     return ret;
00511 }
```

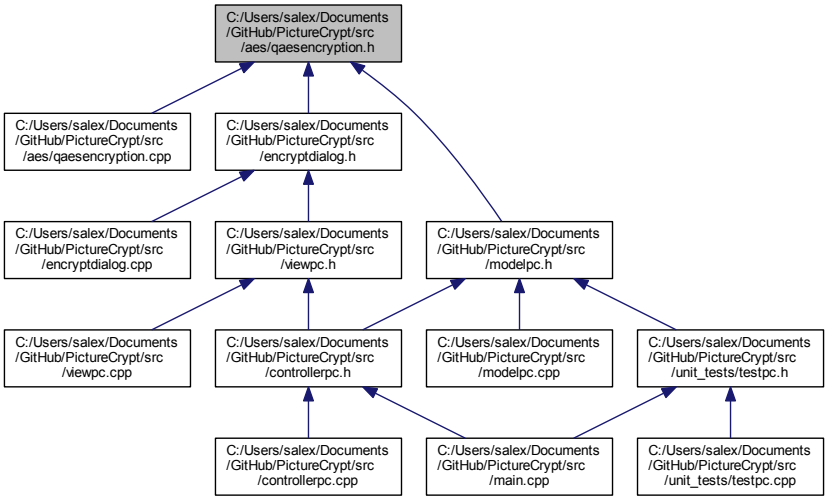
10.11 C:/Users/salex/Documents/GitHub/PictureCrypt/src/aes/qaesencryption.h File Reference

```
#include <QObject>
#include <QByteArray>
```

Include dependency graph for qaesencryption.h:



This graph shows which files directly or indirectly include this file:



Classes

- class QAESEncryption

  The QAESEncryption class Small and portable AES encryption class for Qt. Supports all key sizes - 128/192/256 bits - ECB, CBC, CFB and OFB modes. Class made entirely by bricke. Github: https://github.com/bricke/Qt-AES.

10.12 qaesencryption.h

```
00001 #ifndef QAESENCRYPTION_H
00002 #define QAESENCRYPTION_H
00003
00004 #include <QObject>
00005 #include <QByteArray>
00006
00014 class QAESEncryption : public QObject
00015 {
00016     Q_OBJECT
00017 public:
00027     enum Aes {
00028         AES_128,
00029         AES_192,
00030         AES_256
00031     };
00040     enum Mode {
00041         ECB,
00042         CBC,
00043         CFB,
00044         OFB
00045     };
00046
00055     enum Padding {
00056       ZERO,
00057       PKCS7,
00058       ISO
00059     };
00071     static QByteArray Crypt(QAESEncryption::Aes level,
      QAESEncryption::Mode mode, const QByteArray &rawText, const QByteArray &
      key,
00072                             const QByteArray &iv = NULL, QAESEncryption::Padding
      padding = QAESEncryption::ISO);
00084     static QByteArray Decrypt(QAESEncryption::Aes level,
      QAESEncryption::Mode mode, const QByteArray &rawText, const QByteArray &
      key,
00085                               const QByteArray &iv = NULL,
      QAESEncryption::Padding padding = QAESEncryption::ISO);
00094     static QByteArray ExpandKey(QAESEncryption::Aes level,
      QAESEncryption::Mode mode, const QByteArray &key);
00102     static QByteArray RemovePadding(const QByteArray &rawText,
      QAESEncryption::Padding padding);
00103
00104     QAESEncryption(QAESEncryption::Aes level,
      QAESEncryption::Mode mode,
00105                    QAESEncryption::Padding padding =
      QAESEncryption::ISO);
00116     QByteArray encode(const QByteArray &rawText, const QByteArray &key, const QByteArray &iv =
      NULL);
00127     QByteArray decode(const QByteArray &rawText, const QByteArray &key, const QByteArray &iv =
      NULL);
00136     QByteArray removePadding(const QByteArray &rawText);
00145     QByteArray expandKey(const QByteArray &key);
00146
00147 signals:
00148
00149 public slots:
00150
00151 private:
00152     int m_nb;
00153     int m_blocklen;
00154     int m_level;
00155     int m_mode;
00156     int m_nk;
00157     int m_keyLen;
00158     int m_nr;
00159     int m_expandedKey;
00160     int m_padding;
00161     QByteArray* m_state;
00162
00163     struct AES256{
```

```
00164         int nk = 8;
00165         int keylen = 32;
00166         int nr = 14;
00167         int expandedKey = 240;
00168     };
00169
00170     struct AES192{
00171         int nk = 6;
00172         int keylen = 24;
00173         int nr = 12;
00174         int expandedKey = 209;
00175     };
00176
00177     struct AES128{
00178         int nk = 4;
00179         int keylen = 16;
00180         int nr = 10;
00181         int expandedKey = 176;
00182     };
00183
00184     quint8 getSBoxValue(quint8 num){return sbox[num];}
00185     quint8 getSBoxInvert(quint8 num){return rsbox[num];}
00186
00187     void addRoundKey(const quint8 round, const QByteArray expKey);
00188     void subBytes();
00189     void shiftRows();
00190     void mixColumns();
00191     void invMixColumns();
00192     void invSubBytes();
00193     void invShiftRows();
00194     QByteArray getPadding(int currSize, int alignment);
00195     QByteArray cipher(const QByteArray &expKey, const QByteArray &plainText);
00196     QByteArray invCipher(const QByteArray &expKey, const QByteArray &plainText);
00197     QByteArray byteXor(const QByteArray &in, const QByteArray &iv);
00198
00199     const quint8 sbox[256] =   {
00200       //0     1    2      3     4    5     6     7      8    9     A      B    C     D     E     F
00201       0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
00202       0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
00203       0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
00204       0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
00205       0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
00206       0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
00207       0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
00208       0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
00209       0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
00210       0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
00211       0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
00212       0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
00213       0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
00214       0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
00215       0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
00216       0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 };
00217
00218     const quint8 rsbox[256] =
00219     { 0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
00220       0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
00221       0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
00222       0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
00223       0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
00224       0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
00225       0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
00226       0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
00227       0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
00228       0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
00229       0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
00230       0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
00231       0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
00232       0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
00233       0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
00234       0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d };
00235
00236     // The round constant word array, Rcon[i], contains the values given by
00237     // x to th e power (i-1) being powers of x (x is denoted as {02}) in the field GF(2^8)
00238     // Only the first 14 elements are needed
00239     const quint8 Rcon[256] = {
00240         0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab/*, 0x4d, 0x9a,
00241         0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
00242         0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,
00243         0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,
00244         0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,
00245         0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,
00246         0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
00247         0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
00248         0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
00249         0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
00250         0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
```

```
00251         0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
00252         0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
00253         0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,
00254         0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,
00255             0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d
      */};
00256 };
00257
00258 #endif // QAESENCRYPTION_H
```

10.13 C:/Users/salex/Documents/GitHub/PictureCrypt/src/config/ErrorsDict.json File Reference

10.14 ErrorsDict.json

```
00001 {
00002     "nodata": "No data given!",
00003     "nullimage": "Image not valid!",
00004     "bigkey": "Key is too big, max is 255 bytes!",
00005     "muchdata": "Too much data for this image",
00006     "wrongmode": "Incorrect mode selected",
00007     "wrongimage": "Image wasn't encrypted by this app or is damaged!",
00008     "noreaddata": "Read data is empty!",
00009     "savefilefail": "Cannot save file, wait wut?",
00010     "bitsBufferFail": "Something went very wrong! Error code 1",
00011     "nojphs": "JPHS not installed, installation required!\nSee Menu -> Configure -> JPHS directory"
00012 }
```

10.15 C:/Users/salex/Documents/GitHub/PictureCrypt/src/config/ErrorsDictSetup.py File Reference

Namespaces

- ErrorsDictSetup

Variables

- string ErrorsDictSetup.filename = 'ErrorsDict.json'
- ErrorsDictSetup.raw = open(filename, 'r')
- ErrorsDictSetup.data = json.load(raw)
- ErrorsDictSetup.input_data = input()
- ErrorsDictSetup.key
- ErrorsDictSetup.value
- ErrorsDictSetup.f
- ErrorsDictSetup.indent

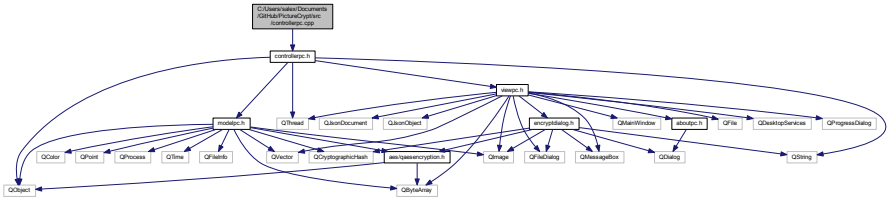
10.16 ErrorsDictSetup.py

```
00001 import json
00002 filename = 'ErrorsDict.json'
00003
00004 raw = open(filename, 'r')
00005
00006 data = json.load(raw)
00007 print('Existing data:')
00008 for key, value in data.items():
00009     print(key, value)
00010
00011 print('------------')
00012 print('Type new data')
00013
00014 input_data = input()
00015
00016 while len(input_data):
00017     key, value = map(str, input_data.split('-'))
00018     data[key] = value
00019     input_data = input()
00020
00021 with open(filename, 'w') as f:
00022     json.dump(data, f, indent=4)
```

10.17 C:/Users/salex/Documents/GitHub/PictureCrypt/src/controllerpc.cpp File Reference

`#include "controllerpc.h"`

Include dependency graph for controllerpc.cpp:



10.18 controllerpc.cpp

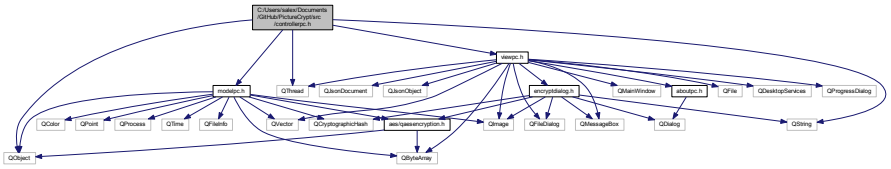
```
00001 #include "controllerpc.h"
00002
00009 ControllerPC::ControllerPC()
00010 {
00011     // Layer creation
00012     view = new ViewPC();
00013     model = new ModelPC();
00014     QThread * modelThread = new QThread();
00015     model->moveToThread(modelThread);
00016     modelThread->start();
00017
00018     view->setVersion(model->versionString);
00019     view->show();
00020     // Layer Connection
00021     connect(view, SIGNAL(encrypt(QByteArray,QImage*,int)), model, SLOT(encrypt(QByteArray,QImage*,int)));
00022     connect(view, SIGNAL(decrypt(QImage*)), model, SLOT(decrypt(QImage*)));
00023     connect(view, SIGNAL(abortModel()), this, SLOT(abortCircuit()));
00024     connect(view, SIGNAL(setBitsUsed(int)), this, SLOT(setBitsUsed(int)));
00025     connect(view, SIGNAL(setJPHSDir(QString)), this, SLOT(setJPHSDir(QString)));
00026
00027     connect(model, SIGNAL(alertView(QString,bool)), view, SLOT(alert(QString,bool)));
00028     connect(model, SIGNAL(saveData(QByteArray)), view, SLOT(saveData(QByteArray)));
00029     connect(model, SIGNAL(saveImage(QImage*)), view, SLOT(saveImage(QImage*)));
00030     connect(model, SIGNAL(setProgress(int)), view, SLOT(setProgress(int)));
00031 }
```

```
00036 void ControllerPC::abortCircuit()
00037 {
00038     model->success = false;
00039 }
00044 void ControllerPC::setBitsUsed(int bitsUsed)
00045 {
00046     model->bitsUsed = bitsUsed;
00047 }
00052 void ControllerPC::setJPHSDir(QString dir)
00053 {
00054     model->defaultJPHSDir = dir;
00055 }
```

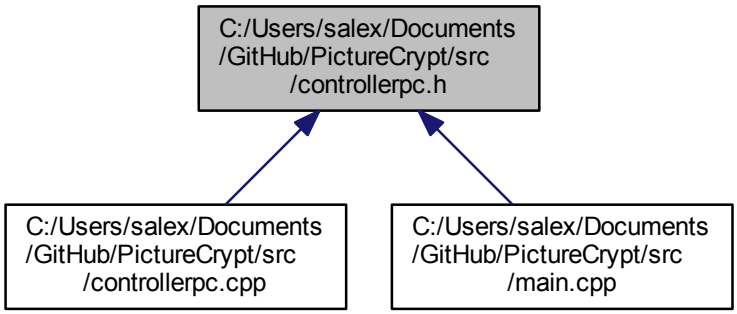
10.19 C:/Users/salex/Documents/GitHub/PictureCrypt/src/controllerpc.h File Reference

```
#include <QObject>
#include <QString>
#include <QThread>
#include <modelpc.h>
#include <viewpc.h>
```

Include dependency graph for controllerpc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ControllerPC

  The ControllerPC class Controller class, which controls View and Model layers.

10.19.1 Detailed Description

Header of ControllerPC class

See also

ControllerPC, ModelPC, ViewPC

Definition in file controllerpc.h.

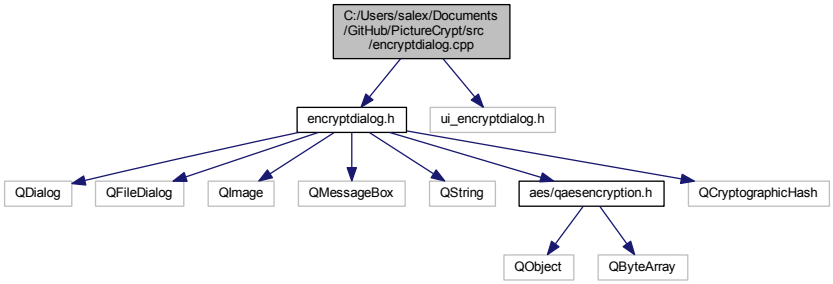
10.20 controllerpc.h

```
00001 #ifndef CONTROLLERPC_H
00002 #define CONTROLLERPC_H
00003
00004 #include <QObject>
00005 #include <QString>
00006 #include <QThread>
00007
00008 #include <modelpc.h>
00009 #include <viewpc.h>
00019 class ControllerPC : public QObject
00020 {
00021     Q_OBJECT
00022 public:
00023     ControllerPC();
00027     long int version;
00031     QString versionString;
00032 public slots:
00033     void abortCircuit();
00034     void setBitsUsed(int bitsUsed);
00035     void setJPHSDir(QString dir);
00036 private:
00037     ViewPC * view;
00038     ModelPC * model;
00039 };
00040
00041 #endif // CONTROLLERPC_H
```

10.21 C:/Users/salex/Documents/GitHub/PictureCrypt/src/encryptdialog.cpp File Reference

```
#include "encryptdialog.h"
#include "ui_encryptdialog.h"
```

Include dependency graph for encryptdialog.cpp:



10.22 encryptdialog.cpp

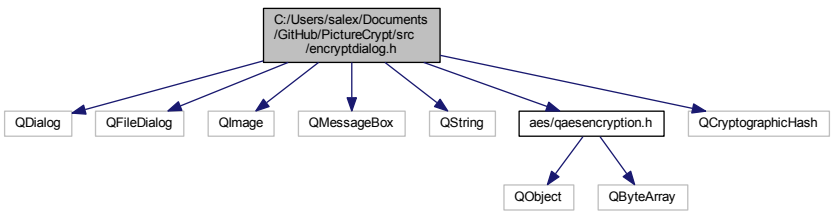
```
00001 #include "encryptdialog.h"
00002 #include "ui_encryptdialog.h"
00009 EncryptDialog::EncryptDialog(QByteArray _data, QWidget *parent) :
00010     QDialog(parent),
00011     ui(new Ui::EncryptDialog)
00012 {
00013     ui->setupUi(this);
00014     data = _data;
00015     success = false;
00016     // UI setup
00017     ui->totalBytes->setText(QString::number(data.size()));
00018     key.clear();
00019     for(int i = 0; i < 24; i++)
00020         key.append(48 + qrand() % 75);
00021     val = 24;
00022     compr_data = zip();
00023     long long int compr_data_size = compr_data.size();
00024     ui->zippedBytes->setText(QString::number(compr_data_size));
00025     goodPercentage = false;
00026     bitsUsed = 8;
00027 }
00028
00029 EncryptDialog::~EncryptDialog()
00030 {
00031     delete ui;
00032 }
00033
00034 void EncryptDialog::alert(QString text)
00035 {
00036     QMessageBox t;
00037     t.setWindowTitle("Message");
00038     t.setIcon(QMessageBox::Warning);
00039     t.setWindowIcon(QIcon(":/mail.png"));
00040     t.setText(text);
00041     t.exec();
00042 }
00049 QByteArray EncryptDialog::zip()
00050 {
00051     // Zip
00052     QByteArray c_data = qCompress(data, 9);
00053     // Encryption
00054     QByteArray hashKey = QCryptographicHash::hash(key.toUtf8(), QCryptographicHash::Sha256);
00055     return QAESEncryption::Crypt(QAESEncryption::AES_256,
      QAESEncryption::ECB, c_data, hashKey);
00056 }
00060 void EncryptDialog::on_fileButton_clicked()
00061 {
00062     // Selet file
00063     inputFileName = QFileDialog::getOpenFileName(this, tr("Open File"), "/", tr("Images (*.png
       *.xpm *.jpg *.jpeg)"));
00064     ui->fileLabel->setText(inputFileName);
00065     // Open image
00066     QImage img(inputFileName);
00067     image = img;
00068     // Get size
00069     size = img.width() * img.height();
00070     // UI setup
00071     long long int compr_data_size = compr_data.size();
00072     ui->zippedBytes->setText(QString::number(compr_data_size));
00073     if(inputFileName.isEmpty()) {
00074         ui->percentage->setText("");
00075         return;
00076     }
00077     double perc = (compr_data_size + 14 + val) * 100 / (size * 3) *
      bitsUsed / 8;
00078     ui->percentage->setText(QString::number(perc) + "%");
00079     goodPercentage = perc < 70;
00080 }
00085 void EncryptDialog::on_buttonBox_accepted()
00086 {
00087     if(!goodPercentage) {
00088         alert("Your encoding percentage is over 70% which is a bit ambiguous :(");
00089         success = false;
00090         return;
00091     }
00092     // Final zip
00093     compr_data = zip();
00094     success = true;
00095     close();
00096 }
00100 void EncryptDialog::on_buttonBox_rejected()
00101 {
00102     success = false;
00103     close();
```

```
00104 }
00110 void EncryptDialog::on_horizontalSlider_valueChanged(int
      value)
00111 {
00112     // Key generator with value of charachters
00113     key.clear();
00114     for(int i = 0; i < value; i++)
00115         key.append(48 + qrand() % 75);
00116     val = value;
00117     ui->keyLabel->setText(QString::number(value));
00118 }
00123 void EncryptDialog::on_bitsSlider_valueChanged(int value)
00124 {
00125     bitsUsed = value;
00126     ui->bitsUsedLbl->setText(QString::number(value));
00127     if(ui->percentage->text().isEmpty())
00128         return;
00129     double perc = (compr_data.size() + 14 + val) * 100 / (size * 3) * 8 /
      bitsUsed;
00130     ui->percentage->setText(QString::number(perc) + "%");
00131 }
```

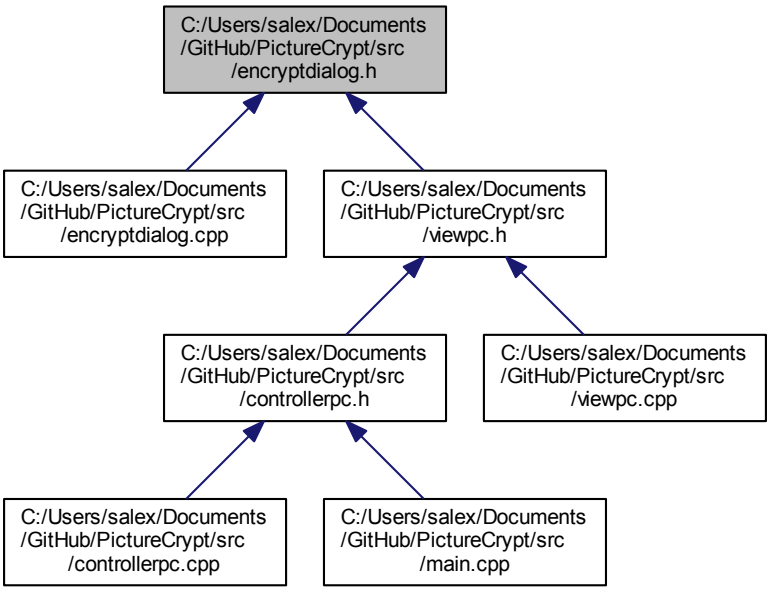
10.23 C:/Users/salex/Documents/GitHub/PictureCrypt/src/encryptdialog.h File Reference

```
#include <QDialog>
#include <QFileDialog>
#include <QImage>
#include <QMessageBox>
#include <QString>
#include <aes/qaesencryption.h>
#include <QCryptographicHash>
```

Include dependency graph for encryptdialog.h:



This graph shows which files directly or indirectly include this file:



Classes

- class EncryptDialog

  The EncryptDialog class Class to get the image and key to store secret info.

Namespaces

- Ui

10.24 encryptdialog.h

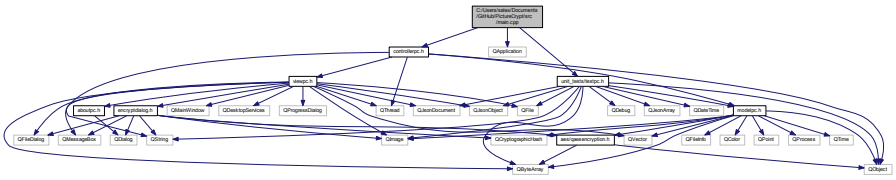
```
00001 #ifndef ENCRYPTDIALOG_H
00002 #define ENCRYPTDIALOG_H
00003
00004 #include <QDialog>
00005 #include <QFileDialog>
00006 #include <QImage>
00007 #include <QMessageBox>
00008 #include <QString>
00009
00010 #include <aes/qaesencryption.h>
00011 #include <QCryptographicHash>
00012
00013 namespace Ui {
00014 class EncryptDialog;
00015 }
00021 class EncryptDialog : public QDialog
00022 {
00023     Q_OBJECT
00024
00025 public:
00026     explicit EncryptDialog(QByteArray _data, QWidget *parent = 0);
```

```
00027     ~EncryptDialog();
00028
00029 public slots:
00030     void on_fileButton_clicked();
00031
00032     void on_buttonBox_accepted();
00033
00034     void on_buttonBox_rejected();
00035
00036     void on_horizontalSlider_valueChanged(int
      value);
00037
00038 public:
00042     QByteArray data;
00046     bool success;
00050     QByteArray compr_data;
00054     QString inputFileName;
00058     long long int size;
00062     QString key;
00066     bool goodPercentage;
00070     int val;
00075     int bitsUsed;
00079     QImage image;
00080     QByteArray zip();
00081 private slots:
00082     void on_bitsSlider_valueChanged(int value);
00083
00084 private:
00085     Ui::EncryptDialog *ui;
00086     void alert(QString text);
00087 };
00088
00089 #endif // ENCRYPTDIALOG_H
```

10.25 C:/Users/salex/Documents/GitHub/PictureCrypt/src/main.cpp File Reference

```
#include "controllerpc.h"
#include <QApplication>
#include <unit_tests/testpc.h>
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char *argv[])

10.25.1 Function Documentation

10.25.1.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 115 of file main.cpp.

Here is the call graph for this function:

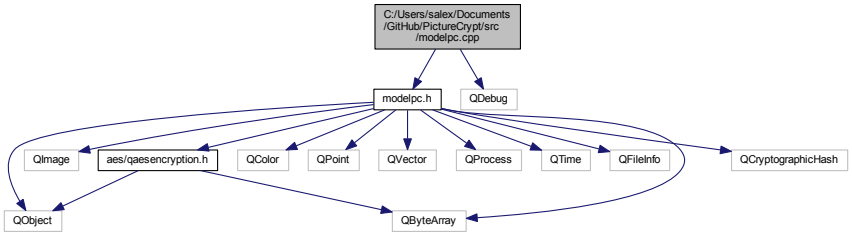
10.26 main.cpp

```
00001 #include "controllerpc.h"
00002 #include <QApplication>
00003 #include <unit_tests/testpc.h>
00115 int main(int argc, char *argv[])
00116 {
00117     QApplication a(argc, argv);
00118     TestPC test;
00119     bool success = test.startTest();
00120     if(success)
00121         ControllerPC w;
00122
00123     return a.exec();
00124 }
```

10.27 C:/Users/salex/Documents/GitHub/PictureCrypt/src/modelpc.cpp File Reference

```
#include "modelpc.h"
#include <QDebug>
```

Include dependency graph for modelpc.cpp:



10.28 modelpc.cpp

```
00001 #include "modelpc.h"
00002 #include <QDebug>
00008 ModelPC::ModelPC()
00009 {
00010     // Version control
00011     versionString = "1.3.0";
00012
00013     auto ver = versionString.split(".");
00014     version = ver[0].toInt() * pow(2, 16) + ver[1].toInt() * pow(2, 8) + ver[2].toInt();
00015
00016     ver_byte = bytes(ver[0].toInt()) +
00017             bytes(ver[1].toInt()) +
00018             bytes(ver[2].toInt());
00019     // Random seed
00020     qsrand(randSeed());
00021 }
00035 QImage * ModelPC::start(QByteArray data, QImage * image, int
      mode, QString key, int _bitsUsed, QString *_error)
00036 {
00037     // Error management
00038     if(_error == nullptr)
00039         _error = new QString();
00040     *_error = "ok";
00041     error = _error;
00042
00043     if(data.isEmpty()) {
00044         fail("nodata");
00045         return nullptr;
00046     }
00047     if(image == nullptr || image->isNull()) {
00048         fail("nullimage");
00049         return nullptr;
00050     }
00051     if(_bitsUsed < 1 || _bitsUsed > 8) {
00052         fail("bitsWrong");
00053         return nullptr;
00054     }
00055     if(key.isEmpty()) {
00056         qsrand(randSeed());
00057         for(int i = 0; i < 32; i++)
00058             key.append(48 + qrand() % 75);
00059     }
00060     else if(key.size() > 255) {
00061         fail("bigkey");
00062         return nullptr;
00063     }
00064     long long usedBytes = data.size() + 14 + key.size();
00065     long long size = image->width() * image->height();
00066     if(usedBytes * 100 / (size * 3) * 8 / _bitsUsed > 70) {
00067         fail("muchdata");
00068         return nullptr;
00069     }
00070
00071     curMode = mode;
00072     bitsUsed = _bitsUsed;
00073
00074     QByteArray key_data = key.toUtf8();
00075     QByteArray zipped_data = zip(data, key_data);
00076     QByteArray encr_data = bytes(key_data.size()) + key_data + zipped_data;
00077
00078     if(*error == "ok")
00079         return encrypt(encr_data, image, curMode, error);
00080     else
00081         return nullptr;
00082 }
00083
00093 QImage * ModelPC::encrypt(QByteArray encr_data, QImage * image, int
      mode, QString *_error)
00094 {
00095     // Error management
00096     if(_error == nullptr)
00097         _error = new QString();
00098     *_error = "ok";
00099     error = _error;
00100
00101     // TODO Remove debug mode = 0
00102     mode = 0;
00103
00104     if(encr_data.isEmpty()) {
00105         fail("nodata");
00106         return nullptr;
00107     }
00108     if(image == nullptr || image->isNull()) {
00109         fail("nullimage");
```

```
00110         return nullptr;
00111     }
00112
00113     encr_data = ver_byte + encr_data;
00114     long long int countBytes = encr_data.size();
00115     curMode = mode;
00116     switch(curMode)
00117     {
00118     case 0:
00119         circuit(image, &encr_data, countBytes);
00120         break;
00121     case 1:
00122         jphs(image, &encr_data);
00123         break;
00124     default:
00125         fail("wrongmode");
00126         return nullptr;
00127     }
00128
00129
00130     // Saving
00131     if(success) {
00132         emit saveImage(image);
00133         return image;
00134     }
00135     else
00136         return nullptr;
00137 }
00145 QByteArray ModelPC::decrypt(QImage * image, QString *_error)
00146 {
00147     // Error management
00148     if(_error == nullptr)
00149         _error = new QString();
00150     *_error = "ok";
00151     error = _error;
00152     if(image == nullptr || image->isNull()) {
00153         fail("nullimage");
00154         return nullptr;
00155     }
00156     // Image opening
00157     int w = image->width();
00158     int h = image->height();
00159
00160     // Getting corner pixels
00161     QColor colUL = image->pixelColor(0, 0).toRgb();
00162     QColor colUR = image->pixelColor(w - 1, 0).toRgb();
00163     QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();
00164
00165     // Getting verification code
00166     int verifCode = (((colUR.green() % 2) << 5) + colUR.blue() % 32) << 2;
00167     verifCode += colDR.blue() % 4;
00168     if(verifCode != 166){
00169         fail("veriffail");
00170         return nullptr;
00171     }
00172     // Getting number of bytes
00173     long long int countBytes = (colUL.blue() % 32 + ((colUL.green() % 32) << 5) + ((colUL.red() % 32) << 10
      )) << 9;
00174     countBytes += ((colUR.red() % 32) << 4) + (colUR.green() >> 1) % 16;
00175
00176     bitsUsed = (colDR.blue() >> 2) % 8 + 1;
00177     curMode = colDR.green() % 32;
00178
00179     // Start of the circuit
00180     QByteArray data;
00181     circuit(image, &data, countBytes);
00182
00183     // Check if circuit was successful
00184     if(!success)
00185         return nullptr;
00186     if(data.isEmpty())
00187     {
00188         fail("noreaddata");
00189         return nullptr;
00190
00191     }
00192     // Version check
00193     long long int _ver = mod(data.at(0) * pow(2, 16));
00194     _ver += mod(data.at(1) * pow(2, 8));
00195     _ver += mod(data.at(2));
00196     data.remove(0, 3);
00197     if(_ver > version) {
00198         fail("Picture's app version is newer than yours. Image version is "
00199               + generateVersionString(_ver) + ", yours is "
00200               + generateVersionString(version) + ".");
00201         return nullptr;
00202     }
```

```
00203     else if(_ver < version) {
00204         fail("Picture's app version is older than yours. Image version is "
00205               + generateVersionString(_ver) + ", yours is "
00206               + generateVersionString(version) + ".");
00207         return nullptr;
00208     }
00209
00210     // Obtain the key
00211     int key_size = mod(data.at(0));
00212     QByteArray key = data.mid(1, key_size);
00213     data.remove(0, key_size + 1);
00214
00215     // Unzip
00216     QByteArray unzipped_data = unzip(data, key);
00217     emit saveData(unzipped_data);
00218     return unzipped_data;
00219 }
00224 void ModelPC::fail(QString message)
00225 {
00226     *error = message;
00227     alert(message, true);
00228     success = false;
00229     emit setProgress(101);
00230 }
00236 void ModelPC::jphs(QImage *image, QByteArray *data)
00237 {
00238     // Under Development
00239     return;
00240
00241     // Dead code
00242
00243     success = true;
00244     bool isEncrypt = !data->isEmpty();
00245     QString targetEXE = defaultJPHSDir + (isEncrypt ? "/jphide.exe" : "/jpseek.exe");
00246     if(!fileExists(targetEXE))
00247     {
00248         fail("nojphs");
00249         return;
00250     }
00251
00252     QString randomFileName = defaultJPHSDir + "/";
00253     qsrand(randSeed());
00254     for(int i = 0; i < 10; i++)
00255         randomFileName.append(97 + qrand() % 25);
00256     image->save(randomFileName + ".jpg");
00257     if(isEncrypt) {
00258         QFile file(randomFileName + ".pc");
00259         if(!file.open(QFile::WriteOnly)) {
00260             fail("savefilefail");
00261             return;
00262         }
00263         file.write(*data);
00264         file.close();
00265
00266         QStringList args;
00267         args << (randomFileName + ".jpg") << (randomFileName + "_out.jpg") << (randomFileName + ".pc");
00268         QProcess prog(this);
00269         prog.start(targetEXE, args);
00270         prog.waitForStarted();
00271         prog.write("test\n");
00272         prog.waitForBytesWritten();
00273         prog.write("test\n");
00274         prog.waitForBytesWritten();
00275         prog.waitForReadyRead();
00276         QByteArray bytes = prog.readAll();
00277         prog.waitForFinished();
00278         //QByteArray readData = prog.readAll();
00279         prog.close();
00280         // Cleaning - Deleting temp files
00281
00282     }
00283     else {
00284
00285     }
00286
00287 }
00288
00297 void ModelPC::circuit(QImage *image, QByteArray *data, long long countBytes)
00298 {
00299
00300     // Some flags and creation of the ProgressDialog
00301     success = true;
00302     emit setProgress(-1);
00303     bool isEncrypt = !data->isEmpty();
00304
00305     // Image setup
00306     int w = image->width();
```

```
00307     int h = image->height();
00308
00309     // Visited pixels array
00310     QVector <QPoint> were;
00311     were.push_back(QPoint(0, 0));
00312     were.push_back(QPoint(0, h - 1));
00313     were.push_back(QPoint(w - 1, 0));
00314     were.push_back(QPoint(w - 1, h - 1));
00315
00316     long long int offset = 0;
00317
00318     // Pre-start Cleaning
00319     circuitData = data;
00320     circuitImage = image;
00321     circuitCountBytes = countBytes;
00322     cur = 0;
00323     bitsBuffer.clear();
00324
00325     // Writing Top-Left to Bottom-Left
00326     for(int i = 1; i < h - 1 && mustGoOn(isEncrypt); i++) {
00327         QPoint pos(0, i);
00328         processPixel(pos, &were, isEncrypt);
00329     }
00330     // Writing Bottom-Right to Top-Right
00331     if(mustGoOn(isEncrypt))
00332     {
00333         for(int i = h - 2; i >= 1 && mustGoOn(isEncrypt); i--){
00334             QPoint pos(w - 1, i);
00335             processPixel(pos, &were, isEncrypt);
00336         }
00337     }
00338     // Main cycle
00339     // Strong is considered as actual corner pixel and weak as pixel near it like (1, 0) or (0, 1)
00340     while(mustGoOn(isEncrypt))
00341     {
00342         // Strong Top-Right to Strong Bottom-Right
00343         for(int i = offset; i < h - offset && mustGoOn(isEncrypt); i++){
00344             QPoint pos(w - offset - 2, i);
00345             processPixel(pos, &were, isEncrypt);
00346         }
00347         // Strong Top-Left to Weak Top-Right
00348         for(int i = offset + 1; i < w - offset - 2 && mustGoOn(isEncrypt); i++){
00349             QPoint pos(i, offset);
00350             processPixel(pos, &were, isEncrypt);
00351         }
00352         // Weak Bottom-Right to Weak Bottom-Left
00353         for(int i = w - 3 - offset; i >= offset + 2 && mustGoOn(isEncrypt); i--){
00354             QPoint pos(i, h - offset - 1);
00355             processPixel(pos, &were, isEncrypt);
00356         }
00357         // Weak Top-Left to Strong Bottom-Left
00358         for(int i = offset + 1; i < h - offset && mustGoOn(isEncrypt); i++){
00359             QPoint pos(offset + 1, i);
00360             processPixel(pos, &were, isEncrypt);
00361         }
00362         offset++;
00363     }
00364     // Extra writing
00365     if(!success)
00366         return;
00367     if(isEncrypt)
00368     {
00369         // Getting past colors
00370         QColor colUL = image->pixelColor(0, 0).toRgb();
00371         QColor colUR = image->pixelColor(w - 1, 0).toRgb();
00372         QColor colDL = image->pixelColor(0, h - 1).toRgb();
00373         QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();
00374         int red = 0;
00375         int green = 0;
00376         int blue = 0;
00377
00378         // Writing Upper Left
00379         red = (colUL.red() & 224) + (countBytes >> 19);
00380         green = (colUL.green() & 224) + (countBytes >> 14) % 32;
00381         blue = (colUL.blue() & 224) + (countBytes >> 9) % 32;
00382         image->setPixelColor(0, 0, QColor(red, green, blue));
00383
00384         // Writing Upper Right
00385         red = (colUR.red() & 224) + (countBytes >> 4) % 32;
00386         green = (colUR.green() & 224) + ((countBytes % 16) << 1) + 1;
00387         blue = (colUR.blue() & 224) + 9;
00388         image->setPixelColor(w - 1, 0, QColor(red, green, blue));
00389
00390         // Getting extra bytes if left
00391         while(cur < countBytes)
00392             push(mod(circuitData->at(cur++)), 8);
00393         if(bitsBuffer.size() > 20) {
```

```
00394             fail("bitsBufferFail");
00395             return;
00396         }
00397         // Getting extra data as long.
00398         long extraData = pop(-2);
00399
00400         // Writing Down Left
00401         red = (colDL.red() & 224) + (extraData >> 15);
00402         green = (colDL.green() & 224) + (extraData >> 10) % 32;
00403         blue = (colDL.blue() & 224) + (extraData >> 5) % 32;
00404         image->setPixelColor(0, h - 1, QColor(red, green, blue));
00405
00406         // Writing Down Right
00407         red = (colDR.red() & 224) + extraData % 32;
00408         green = (colDR.green() & 224);
00409         blue = (colDR.blue() & 224) + ((bitsUsed - 1) << 2) + 2;
00410         image->setPixelColor(w - 1, h - 1, QColor(red, green, blue));
00411     }
00412     else
00413     {
00414         // Read the past pixels
00415         QColor colDL = image->pixelColor(0, h - 1).toRgb();
00416         QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();
00417
00418         // Read extra data
00419         long extraData = ((colDL.red() % 32) << 15) + ((colDL.green() % 32) << 10);
00420         extraData += ((colDL.blue() % 32) << 5) + colDR.red() % 32;
00421
00422         // Add extra data to the bitsBuffer
00423         push(extraData, (countBytes - cur) * 8 - bitsBuffer.size());
00424
00425         // Move bits from bitsBuffer to the QByteArray
00426         while(!bitsBuffer.isEmpty())
00427             data->append(pop(8));
00428     }
00429     emit setProgress(101);
00430 }
00431
00439 void ModelPC::processPixel(QPoint pos, QVector<QPoint> *were, bool isEncrypt)
00440 {
00441     if(!success)
00442         return;
00443     // Check if point was already visited
00444     if(were->contains(pos)){
00445         fail("Point (" + QString::number(pos.x()) + "," + QString::number(pos.y()) + ") was visited
      twice! Error code 2");
00446         return;
00447     }
00448     else
00449         were->push_back(pos);
00450     if(isEncrypt)
00451     {
00452         // Make sure that there are enough bits in bitsBuffer to write
00453         while(bitsBuffer.size() < 3 * bitsUsed)
00454             push(mod(circuitData->at(cur++)), 8);
00455         // Read past contains
00456         QColor pixelColor = circuitImage->pixelColor(pos);
00457         int red = pixelColor.red();
00458         int green = pixelColor.green();
00459         int blue = pixelColor.blue();
00460
00461         // Write new data in last bitsUsed pixels
00462         red += pop() - red % (int) pow(2, bitsUsed);
00463         green += pop() - green % (int) pow(2, bitsUsed);
00464         blue += pop() - blue % (int) pow(2, bitsUsed);
00465
00466         circuitImage->setPixelColor(pos, QColor(red, green, blue));
00467     }
00468     else
00469     {
00470         QColor read_color = circuitImage->pixelColor(pos).toRgb();
00471         // Reading the pixel
00472         int red = read_color.red();
00473         int green = read_color.green();
00474         int blue = read_color.blue();
00475
00476         // Reading the last bitsUsed pixels
00477         red %= (int) pow(2, bitsUsed);
00478         green %= (int) pow(2, bitsUsed);
00479         blue %= (int) pow(2, bitsUsed);
00480
00481         // Getting the data in the bitsBuffer.
00482         push(red);
00483         push(green);
00484         push(blue);
00485
00486         // Getting data to QByteArray
```

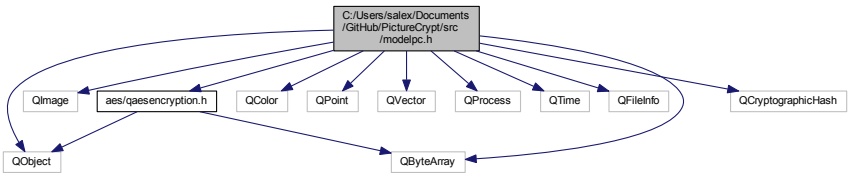
```
00487         while(bitsBuffer.size() >= 8) {
00488             circuitData->append(pop(8));
00489             cur++;
00490         }
00491     }
00492     emit setProgress(100 * cur / circuitCountBytes);
00493 }
00494
00495 long ModelPC::pop(int bits)
00496 {
00497     // Hard to say
00498     long res = 0;
00499     int poppedBits = bits == -1 ? bitsUsed : bits;
00500     if(bits == -2)
00501         poppedBits = bitsBuffer.size();
00502     for(int i = 0; i < poppedBits; i++)
00503         res += bitsBuffer[i] * pow(2, poppedBits - i - 1);
00504     bitsBuffer.remove(0, poppedBits);
00505     return res;
00506 }
00507
00508 void ModelPC::push(int data, int bits)
00509 {
00510     // That's easier, but also hard
00511     int buf_size = bitsBuffer.size();
00512     int extraSize = bits == -1 ? bitsUsed : bits;
00513     bitsBuffer.resize(buf_size + extraSize);
00514     for(int i = bitsBuffer.size() - 1; i >= buf_size; i--, data >>= 1)
00515         bitsBuffer[i] = data % 2;
00516 }
00517
00518 bool ModelPC::mustGoOn(bool isEncrypt)
00519 {
00520     return success && (isEncrypt ? (circuitCountBytes - cur) * 8 + bitsBuffer.size() >=
    bitsUsed * 3 :
00521                                    circuitData->size() * 8 + bitsBuffer.size() <
00522                                    circuitCountBytes * 8 - (circuitCountBytes * 8)% (
    bitsUsed * 3));
00523 }
00532 QByteArray ModelPC::unzip(QByteArray data, QByteArray key)
00533 {
00534     // Decryption
00535     QByteArray hashKey = QCryptographicHash::hash(key, QCryptographicHash::Sha256);
00536     QAESEncryption encryption(QAESEncryption::AES_256,
    QAESEncryption::ECB);
00537     QByteArray new_data = encryption.decode(data, hashKey);
00538     // Decompressing
00539     return qUncompress(new_data);
00540 }
00549 QByteArray ModelPC::zip(QByteArray data, QByteArray key)
00550 {
00551     // Zip
00552     QByteArray c_data = qCompress(data, 9);
00553     // Encryption
00554     QByteArray hashKey = QCryptographicHash::hash(key, QCryptographicHash::Sha256);
00555     return QAESEncryption::Crypt(QAESEncryption::AES_256,
    QAESEncryption::ECB, c_data, hashKey);
00556 }
00557
00558 bool ModelPC::fileExists(QString path)
00559 {
00560     QFileInfo check_file(path);
00561     return check_file.exists() && check_file.isFile();
00562 }
00563
00570 QByteArray ModelPC::bytes(long long n)
00571 {
00572     return QByteArray::fromHex(QByteArray::number(n, 16));
00573 }
00580 unsigned int ModelPC::mod(int input)
00581 {
00582     if(input < 0)
00583         return (unsigned int) (256 + input);
00584     else
00585         return (unsigned int) input;
00586 }
00593 void ModelPC::alert(QString message, bool isWarning)
00594 {
00595     emit alertView(message, isWarning);
00596 }
00602 QColor ModelPC::RGBbytes(long long byte)
00603 {
00604     int blue = byte % 256;
00605     int green = (byte / 256) % 256;
00606     int red = byte / pow(2, 16);
00607     return QColor(red, green, blue);
00608 }
```

```
00609
00610 QString ModelPC::generateVersionString(long ver)
00611 {
00612     return QString::number((int)( ver / pow(2, 16))) + "." + QString::number(((int) (ver / 256)) % 256) + "
      ." + QString::number(ver % 256);
00613 }
00614
00615 uint ModelPC::randSeed()
00616 {
00617     QTime time = QTime::currentTime();
00618     uint randSeed = time.msecsSinceStartOfDay() % 65536 + time.minute() * 21 + time.second() * 2;
00619     return randSeed;
00620 }
```

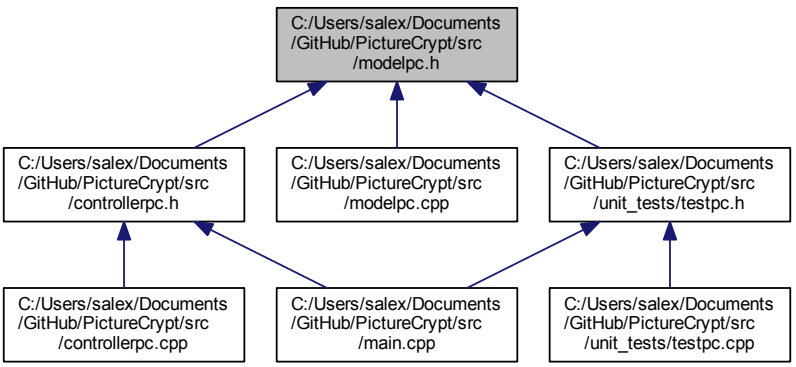
10.29 C:/Users/salex/Documents/GitHub/PictureCrypt/src/modelpc.h File Reference

```
#include <QObject>
#include <QImage>
#include <QByteArray>
#include <QColor>
#include <QPoint>
#include <QVector>
#include <QProcess>
#include <QTime>
#include <QFileInfo>
#include <aes/qaesencryption.h>
#include <QCryptographicHash>
```

Include dependency graph for modelpc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ModelPC

  The ModelPC class Model Layer of the app. Controled by ControllerPC.

10.29.1 Detailed Description

Header of ModelPC class

See also

ControllerPC, ModelPC, ViewPC

Definition in file modelpc.h.

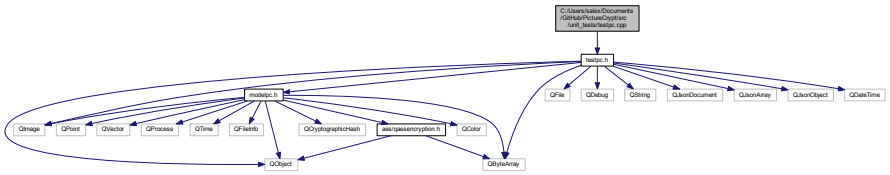
10.30 modelpc.h

```
00001 #ifndef MODELPC_H
00002 #define MODELPC_H
00003
00004 #include <QObject>
00005 #include <QImage>
00006 #include <QByteArray>
00007 #include <QColor>
00008 #include <QPoint>
00009 #include <QVector>
00010 #include <QProcess>
00011 #include <QTime>
00012 #include <QFileInfo>
00013
00014 #include <aes/qaesencryption.h>
00015 #include <QCryptographicHash>
00016
00027 class ModelPC : public QObject
00028 {
00029     Q_OBJECT
00030 public:
00031     ModelPC();
00032
00033 signals:
00040     alertView(QString messageCode, bool isWarning);
00045     saveData(QByteArray data);
00050     saveImage(QImage *image);
00056     setProgress(int val);
00057
00058 public slots:
00059     QImage *start(QByteArray data, QImage *image, int mode = 0, QString
      key = "", int _bitsUsed = 8, QString *_error = nullptr);
00060     QImage *encrypt(QByteArray encr_data, QImage * image, int mode = 0, QString *_error =
      nullptr);
00061     QByteArray decrypt(QImage * image, QString *_error = nullptr);
00062     void fail(QString message);
00063
00064 public:
00069     bool success;
00073     long version;
00077     QString versionString;
00081     int curMode;
00085     int bitsUsed;
00089     QString defaultJPHSDir;
00093     QString * error;
00094     QByteArray unzip(QByteArray data, QByteArray key);
00095     void alert(QString message, bool isWarning = false);
00096 protected:
00097     void circuit(QImage * image, QByteArray * data, long long int countBytes);
00098     void jphs(QImage * image, QByteArray * data);
00099     void processPixel(QPoint pos, QVector<QPoint> *were, bool isEncrypt);
00100     QByteArray zip(QByteArray data, QByteArray key);
00101 private:
00102     bool fileExists(QString path);
00103     QByteArray bytes(long long n);
```

```
00104     unsigned int mod(int input);
00105     QByteArray ver_byte;
00106     QColor RGBbytes(long long byte);
00107     QString generateVersionString(long ver);
00108     uint randSeed();
00109
00110     QByteArray * circuitData;
00111     QImage * circuitImage;
00112     long long circuitCountBytes;
00113     long cur;
00114     bool mustGoOn(bool isEncrypt);
00115
00116     QVector <bool> bitsBuffer;
00117     long pop(int bits = -1);
00118     void push(int data, int bits = -1);
00119
00120     void setError(QString word);
00121 };
00122
00123 #endif // MODELPC_H
```

10.31 C:/Users/salex/Documents/GitHub/PictureCrypt/src/unit_tests/testpc.cpp File Reference

`#include "testpc.h"`
Include dependency graph for testpc.cpp:



10.32 testpc.cpp

```
00001 #include "testpc.h"
00005 TestPC::TestPC()
00006 { }
00018 bool TestPC::test(QByteArray data, QImage rImage, QString expectedOutput, int
      mode, QString key, int bitsUsed)
00019 {
00020     // Error outputs
00021     QString error1, error2;
00022     // Embedding
00023     QImage * retImage = model->start(data, &rImage, mode, key,
      bitsUsed, &error1);
00024     // De-embedding
00025     QByteArray output = model->decrypt(retImage, &error2);
00026
00027     // Success of error outputs
00028     bool er1 = error1 == expectedOutput;
00029     bool er2 = error2 == expectedOutput;
00030     if(expectedOutput == "ok")
00031         return er1 && er2 && data == output;
00032     else
00033         return er1 || er2;
00034 }
00042 int TestPC::startTest()
00043 {
00044     qDebug() << "Testing started...\n";
00045     model = new ModelPC();
00046
00047     // Long text open
00048     QFile file(":/unit_tests/longtext.txt");
00049     if(!file.open(QFile::ReadOnly))
00050         return false;
00051     text = file.readAll();
```

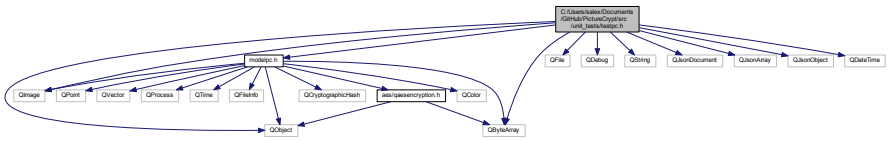
```
00052     file.close();
00053
00054     // Big picture open
00055     image = QImage(":/unit_tests/bigpicture.jpg");
00056     if(image.isNull())
00057         return false;
00058
00059     // JSON tests list open
00060     QFile json_file(":/unit_tests/tests.json");
00061     QJsonDocument doc;
00062     if(!json_file.open(QFile::ReadOnly | QFile::Text))
00063         return false;
00064     QByteArray readData = json_file.readAll();
00065     json_file.close();
00066     doc = QJsonDocument::fromJson(readData);
00067     // Testing
00068     return autoTest(doc);
00069 }
00077 bool TestPC::autoTest(QJsonDocument doc)
00078 {
00079     // Opening the tests array
00080     QJsonObject o = doc.object();
00081     QJsonArray arr = o["tests"].toArray();
00082     int sum = 0;
00083
00084     // Info about tests
00085     QString extraText;
00086     for(int i = 0; i < arr.size(); i++) {
00087         // Reading the data
00088         QJsonObject obj = arr[i].toObject();
00089
00090         QString t = obj["data"].toString();
00091         if(t == "/text/")
00092             t = text;
00093         QByteArray data = t.toUtf8();
00094
00095         QString im = obj["image"].toString();
00096         QImage img(":/unit_tests/" + im);
00097
00098         QString expect = obj["expectation"].toString();
00099
00100         int mode = obj["mode"].toInt();
00101
00102         QString key = obj["key"].toString();
00103
00104         int bitsUsed = obj["bitsUsed"].toInt();
00105
00106         // Testing
00107         bool s = test(data, img, expect, mode, key,
      bitsUsed);
00108
00109         sum += s;
00110         extraText += "\n * Test #" + QString::number(i + 1) + " " + (s ? "completed." : "failed.");
00111     }
00112     // Writing log
00113     QFile file("tests.log");
00114     bool testsSuc = sum == arr.size();
00115     if(!file.open(QFile::WriteOnly | QFile::Text))
00116         return testsSuc;
00117     QDateTime curTime = QDateTime::currentDateTime();
00118     QString date = curTime.toString("dd.MM.yyyy HH:mm");
00119     QString logtext = "##########################################\n"
00120                       "####Log file created at " + date + "####\n"
00121                       "##########################################\n\n"
00122                       "Status: " + (testsSuc ? "All tests completed" : "Tests failed") + "\n\n"
00123                       "Tests list:\n";
00124     logtext += extraText;
00125     file.write(logtext.toUtf8());
00126     file.close();
00127     // Cleaning up
00128     qDebug() << "Testing completed\n";
00129     delete model;
00130     return testsSuc;
00131 }
```

10.33 C:/Users/salex/Documents/GitHub/PictureCrypt/src/unit_tests/testpc.h File Reference

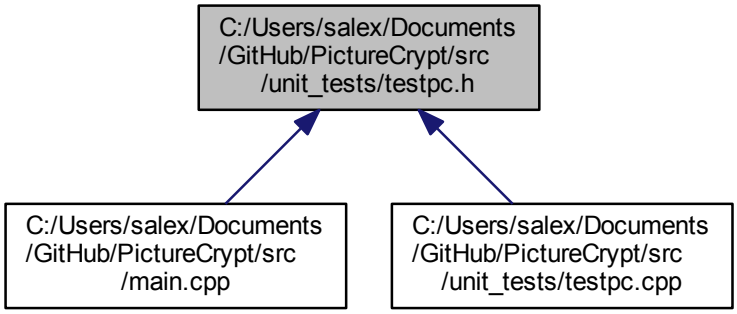
```
#include <QObject>
#include <modelpc.h>
```

```
#include <QFile>
#include <QDebug>
#include <QString>
#include <QImage>
#include <QByteArray>
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonObject>
#include <QDateTime>
```

Include dependency graph for testpc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TestPC

  The TestPC class AutoTest for ModelPC Currently used in main.cpp.

10.34 testpc.h

```
00001 #ifndef TESTPC_H
00002 #define TESTPC_H
00003
00004 #include <QObject>
00005 #include <modelpc.h>
00006
00007 #include <QFile>
00008 #include <QDebug>
00009 #include <QString>
00010 #include <QImage>
00011 #include <QByteArray>
```

```
00012
00013 #include <QJsonDocument>
00014 #include <QJsonArray>
00015 #include <QJsonObject>
00016
00017 #include <QDateTime>
00022 class TestPC : public QObject
00023 {
00024     Q_OBJECT
00025 public:
00026     TestPC();
00027 public slots:
00028     int startTest();
00029 protected slots:
00030     bool test(QByteArray data, QImage rImage,
00031               QString expectedOutput = "ok", int mode = 0,
00032               QString key = "", int bitsUsed = 8);
00033 private:
00037     ModelPC * model;
00041     QByteArray text;
00045     QImage image;
00046
00047     bool autoTest(QJsonDocument doc);
00048 };
00049
00050 #endif // TESTPC_H
```

10.35 C:/Users/salex/Documents/GitHub/PictureCrypt/src/unit_tests/tests-setup.py File Reference

Namespaces

- tests-setup

Variables

- string tests-setup.filename = 'tests.json'
- tests-setup.raw = open(filename, 'r')
- tests-setup.js = json.load(raw)
- tests-setup.sep
- tests-setup.input_data = input()
- list tests-setup.arr = [ ]
- tests-setup.data
- tests-setup.image
- tests-setup.expect
- tests-setup.mode
- tests-setup.key
- tests-setup.bitsUsed
- dictionary tests-setup.obj = {'data':data, 'image':image,'expectation':expect,'mode':int(mode),'key':key,'bits↩ Used':int(bitsUsed)}
- tests-setup.f
- tests-setup.indent

10.36 tests-setup.py

```
00001 import json
00002 filename = 'tests.json'
00003
00004 raw = open(filename, 'r')
00005
00006 js = json.load(raw)
00007 print('Existing tests:')
00008 for obj in js['tests']:
00009     print(obj['data'], obj['image'], obj['expectation'], obj['mode'], obj['key'], obj['bitsUsed'], sep='-')
00010
00011 print('------------')
00012 print('Type new tests')
00013
00014 input_data = input()
00015
00016 arr = []
00017 while len(input_data):
00018     data, image, expect, mode, key, bitsUsed = map(str, input_data.split('-'))
00019
00020     obj = {'data':data, 'image':image,'expectation':expect,'mode':int(mode),'key':key,'bitsUsed':int(
      bitsUsed)}
00021     arr.append(obj)
00022     input_data = input()
00023
00024 js['tests'] += arr
00025 with open(filename, 'w') as f:
00026     json.dump(js, f, indent=4)
```

10.37 C:/Users/salex/Documents/GitHub/PictureCrypt/src/unit_tests/tests.json File Reference

10.38 tests.json

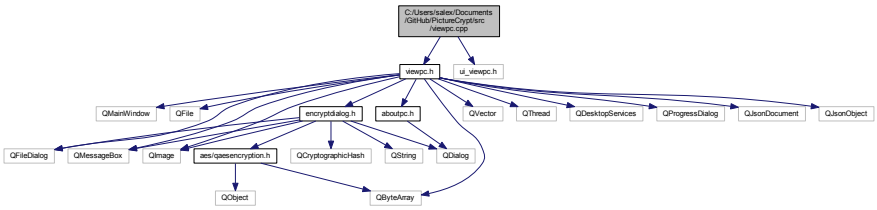
```
00001 {
00002     "tests": [
00003         {
00004             "data": "/text/",
00005             "image": "bigpicture.jpg",
00006             "expectation": "ok",
00007             "mode": 0,
00008             "key": "",
00009             "bitsUsed": 8
00010         },
00011         {
00012             "data": "/text/",
00013             "image": "bigpicture.jpg",
00014             "expectation": "ok",
00015             "mode": 0,
00016             "key": "",
00017             "bitsUsed": 7
00018         },
00019         {
00020             "data": "/text/",
00021             "image": "bigpicture.jpg",
00022             "expectation": "ok",
00023             "mode": 0,
00024             "key": "",
00025             "bitsUsed": 1
00026         },
00027         {
00028             "data": "/text/",
00029             "image": "tinypicture.png",
00030             "expectation": "muchdata",
00031             "mode": 0,
00032             "key": "",
00033             "bitsUsed": 8
00034         },
00035         {
00036             "data": "",
00037             "image": "bigpicture.jpg",
00038             "expectation": "nodata",
00039             "mode": 0,
00040             "key": "",
00041             "bitsUsed": 8
```

```
00042         },
00043         {
00044             "data": "/text/",
00045             "image": "invalid.jpg",
00046             "expectation": "nullimage",
00047             "mode": 0,
00048             "key": "",
00049             "bitsUsed": 8
00050         },
00051         {
00052             "data": "/text/",
00053             "image": "bigpicture.jpg",
00054             "expectation": "bitsWrong",
00055             "mode": 0,
00056             "key": "",
00057             "bitsUsed": 12
00058         }
00059     ]
00060 }
```

10.39 C:/Users/salex/Documents/GitHub/PictureCrypt/src/viewpc.cpp File Reference

```
#include "viewpc.h"
#include "ui_viewpc.h"
```

Include dependency graph for viewpc.cpp:



10.40 viewpc.cpp

```
00001 #include "viewpc.h"
00002 #include "ui_viewpc.h"
00003
00004 ViewPC::ViewPC(QWidget *parent) :
00005     QMainWindow(parent),
00006     ui(new Ui::ViewPC)
00007 {
00008     ui->setupUi(this);
00009
00010     progressDialogClosed = true;
00011
00012     // Alerts dictionary setup
00013     QFile file(":/config/ErrorsDict.json");
00014     if(!file.open(QFile::ReadOnly | QFile::Text)) {
00015         alert("Cannot open config file!");
00016         return;
00017     }
00018     QByteArray readData = file.readAll();
00019     file.close();
00020
00021     QJsonParseError error;
00022     QJsonDocument doc = QJsonDocument::fromJson(readData, &error);
00023     errorsDict = doc.object();
00024 }
00025
00026 ViewPC::~ViewPC()
00027 {
00028     delete ui;
00029 }
00030
00031 void ViewPC::on_encryptMode_clicked()
```

```
00032 {
00033     // Encrypt radio button clicked
00034     setEncryptMode(true);
00035 }
00036
00037 void ViewPC::on_decryptMode_clicked()
00038 {
00039     // Decrypt radio button clicked
00040     setEncryptMode(false);
00041 }
00045 void ViewPC::on_fileButton_clicked()
00046 {
00047     // Opening QFileDialog depending on isEncrypt
00048     if(isEncrypt)
00049         inputFileName = QFileDialog::getOpenFileName(this, tr("Select file"), "/untitled.txt", tr("Text
       files (*.txt);;All Files (*)"));
00050     else
00051         inputFileName = QFileDialog::getOpenFileName(this, tr("Select file"), "/untitled.png", tr("PNG
       files (*.png);;All Files (*)"));
00052     // Display the file name
00053     ui->fileLabel->setText(inputFileName.isEmpty() ? "File not chosen" : inputFileName);
00054 }
00067 void ViewPC::on_startButton_clicked()
00068 {
00069     if(isEncrypt)
00070     {
00071         // Getting the data
00072         QString text = ui->text->toPlainText();
00073         QByteArray data;
00074         if(text.isEmpty()) {
00075             if(inputFileName.isEmpty()) {
00076                 alert("No input file or text was not given. Cannot continue!", true);
00077                 return;
00078             }
00079             // Opening the file
00080             QFile file(inputFileName);
00081             if (!file.open(QIODevice::ReadOnly))
00082             {
00083                 alert("Cannot open file. Cannot continue!", true);
00084                 return;
00085             }
00086             // Check the data size
00087             auto size = file.size();
00088             if(size > pow(2, 24)) {
00089                 alert("Your file is too big, our systems can handle it, but it requires a lot of time.
       We decline.", true);
00090                 file.close();
00091                 return;
00092             }
00093             data = file.readAll();
00094             file.close();
00095         }
00096         else
00097             data = text.toUtf8();
00098         // Select image via EncryptDialog
00099         EncryptDialog * dialog = new EncryptDialog(
      data);
00100         dialog->exec();
00101         if(!dialog->success)
00102             return;
00103
00104         // Get the data
00105         QByteArray encr_data = dialog->compr_data;
00106
00107         // Save the key
00108         QByteArray key_data = dialog->key.toUtf8();
00109
00110         encr_data = bytes(key_data.size()) + key_data + encr_data;
00111         // TODO do the mode thing
00112         emit setBitsUsed(dialog->bitsUsed);
00113         emit encrypt(encr_data, &dialog->image, 0);
00114     }
00115     else
00116     {
00117         // Get the filename of the image
00118         if(!ui->text->toPlainText().isEmpty())
00119             alert("Obviously, the text browser isn't supported for decryption, use File Dialog
       instead.");
00120         if(inputFileName.isEmpty()) {
00121             alert("File not selected. Cannot continue!", true);
00122             return;
00123         }
00124         QImage * res_image = new QImage(inputFileName);
00125         emit decrypt(res_image);
00126     }
00127 }
00133 void ViewPC::alert(QString message, bool isWarning)
```

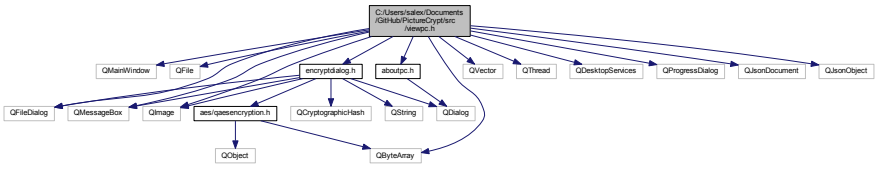
```
00134 {
00135     // Get message
00136     if(errorsDict.contains(message))
00137         message = errorsDict[message].toString();
00138     // Create message box
00139     QMessageBox box;
00140     if(isWarning)
00141         box.setIcon(QMessageBox::Warning);
00142     else
00143         box.setIcon(QMessageBox::Information);
00144     box.setText(message);
00145     box.setWindowIcon(QIcon(":/icons/mail.png"));
00146     box.setWindowTitle("Message");
00147     box.exec();
00148 }
00154 void ViewPC::saveData(QByteArray Edata)
00155 {
00156     // Save data using QFileDialog
00157     QString outputFileName = QFileDialog::getSaveFileName(this, tr("Save File"),
00158                                 "/untitled.txt",
00159                                 tr("Text(*.txt);;All files (*)"));
00160     QFile writeFile(outputFileName);
00161     if (!writeFile.open(QIODevice::WriteOnly))
00162     {
00163         alert("Cannot access file path. Cannot continue!", true);
00164         return;
00165     }
00166     writeFile.write(Edata);
00167     writeFile.close();
00168     alert("Decryption completed!");
00169 }
00175 void ViewPC::saveImage(QImage * image)
00176 {
00177     // Save image using QFileDialog
00178     QString outputFileName = QFileDialog::getSaveFileName(this, tr("Save Image"),
00179                                 "/untitled.png",
00180                                 tr("Images(*.png)"));
00181     if(!image->save(outputFileName)) {
00182         alert("Cannot save file. Unable to continue!", true);
00183         return;
00184     }
00185     alert("Encryption completed!");
00186 }
00193 void ViewPC::setProgress(int val)
00194 {
00195     if(val < 0) {
00196         // Create dialog
00197         dialog = new QProgressDialog("Cryption in progress.", "Cancel", 0, 100);
00198         connect(dialog, SIGNAL(canceled()), this, SLOT(abortCircuit()));
00199         progressDialogClosed = false;
00200         dialog->setWindowTitle("Processing");
00201         dialog->setWindowIcon(QIcon(":/icons/loading.png"));
00202         dialog->show();
00203     }
00204     else if(val >= 100 && !progressDialogClosed) {
00205         // Close dialog
00206         dialog->setValue(100);
00207         QThread::msleep(25);
00208         dialog->close();
00209         dialog->reset();
00210         progressDialogClosed = true;
00211     }
00212     // Update the progress
00213     else if(!progressDialogClosed)
00214         dialog->setValue(val);
00215 }
00219 void ViewPC::abortCircuit()
00220 {
00221     // Set the flag
00222     progressDialogClosed = true;
00223     // Close the dialog
00224     dialog->close();
00225     dialog->reset();
00226     emit abortModel();
00227 }
00232 void ViewPC::setEncryptMode(bool encr)
00233 {
00234     ui->text->setEnabled(encr);
00235     isEncrypt = encr;
00236 }
00241 void ViewPC::setVersion(QString version)
00242 {
00243     // Version setup
00244     versionString = version;
00245 }
00246
00247 QByteArray ViewPC::bytes(long long n)
```

```
00248 {
00249     return QByteArray::fromHex(QByteArray::number(n, 16));
00250 }
00254 void ViewPC::on_actionAbout_triggered()
00255 {
00256     AboutPC about;
00257     about.setVersion(versionString);
00258     about.exec();
00259 }
00260
00264 void ViewPC::on_actionHelp_triggered()
00265 {
00266     QUrl docLink("http://doc.alex.unaux.com/picturecrypt");
00267     QDesktopServices::openUrl(docLink);
00268 }
00269
00270 void ViewPC::on_actionJPHS_path_triggered()
00271 {
00272     QString dir = QFileDialog::getExistingDirectory(this, tr("Open JPHS folder"),
00273                                                     "/home",
00274                                                     QFileDialog::ShowDirsOnly
00275                                                     | QFileDialog::DontResolveSymlinks);
00276     emit setJPHSDir(dir);
00277 }
```

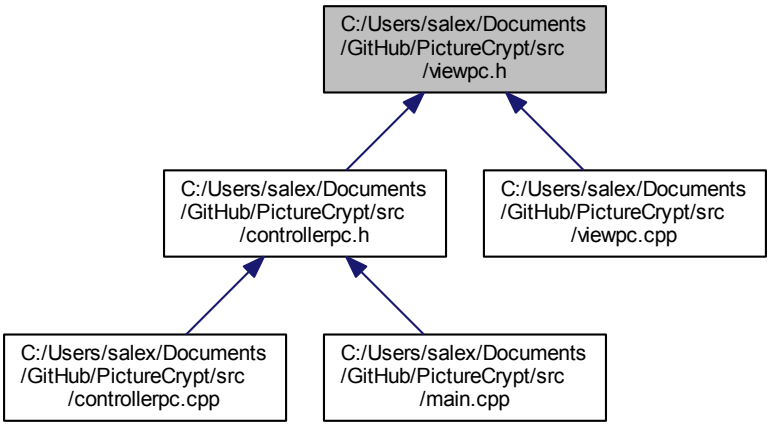
10.41 C:/Users/salex/Documents/GitHub/PictureCrypt/src/viewpc.h File Reference

```
#include <QMainWindow>
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include <QImage>
#include <QByteArray>
#include <QVector>
#include <QThread>
#include <QDesktopServices>
#include <encryptdialog.h>
#include <QProgressDialog>
#include <aboutpc.h>
#include <QJsonDocument>
#include <QJsonObject>
```

Include dependency graph for viewpc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ViewPC

  The ViewPC class View layer of the app. Controls EncryptDialog and ProgressDialog.

Namespaces

- Ui

10.41.1 Detailed Description

Header of ViewPC class

See also

ControllerPC, ModelPC, ViewPC

Definition in file viewpc.h.

10.42 viewpc.h

```
00001 #ifndef VIEWPC_H
00002 #define VIEWPC_H
00003
00004 #include <QMainWindow>
00005 #include <QFile>
00006 #include <QFileDialog>
00007 #include <QMessageBox>
00008 #include <QImage>
00009 #include <QByteArray>
00010 #include <QVector>
```

```
00011 #include <QThread>
00012 #include <QDesktopServices>
00013
00014 #include <encryptdialog.h>
00015 #include <QProgressDialog>
00016 #include <aboutpc.h>
00017
00018 #include <QJsonDocument>
00019 #include <QJsonObject>
00020
00021 namespace Ui {
00022 class ViewPC;
00023 }
00033 class ViewPC : public QMainWindow
00034 {
00035     Q_OBJECT
00036
00037 public:
00038     explicit ViewPC(QWidget *parent = nullptr);
00039     ~ViewPC();
00040 private slots:
00041     void on_encryptMode_clicked();
00042
00043     void on_decryptMode_clicked();
00044
00045     void on_actionJPHS_path_triggered();
00046
00047 protected slots:
00048     void on_fileButton_clicked();
00049
00050     void on_startButton_clicked();
00051
00052     void on_actionAbout_triggered();
00053
00054     void on_actionHelp_triggered();
00055 public slots:
00056     void alert(QString message, bool isWarning = false);
00057     void saveData(QByteArray Edata);
00058     void saveImage(QImage *image);
00059     void setProgress(int val);
00060     void abortCircuit();
00061     void setEncryptMode(bool encr);
00062     void setVersion(QString version);
00063 signals:
00070     encrypt(QByteArray data, QImage * image, int mode);
00075     decrypt(QImage * _image);
00079     abortModel();
00085     setBitsUsed(int bitsUsed);
00090     setJPHSDir(QString dir);
00091 public:
00096     QProgressDialog * dialog;
00101     bool progressDialogClosed;
00102     QJsonObject errorsDict;
00103 private:
00104     Ui::ViewPC *ui;
00105     bool isEncrypt;
00106     QString inputFileName;
00107     QByteArray bytes(long long n);
00108     QString versionString;
00109 };
00110
00111 #endif // VIEWPC_H
```

Index