PictureCrypt

1.3.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

PictureCrypt

Project made using QT Creator on 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

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.

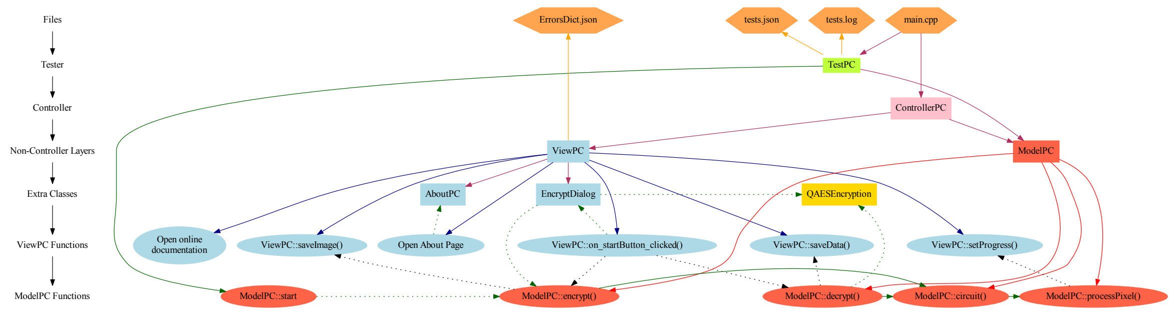
Project is done via MVC Pattern. View and Model layers are totally isolated and run on different threads.

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>

...

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!";
```

See also

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

Avible methods see here: https://waleko.github.io/PictureCrypt/#external-use or here ModelPC

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 our site: http://alex.unaux.com and Github Page: https://waleko.github.io Email me at a.kovrigin0@gmail.com

Author

Alex Kovrigin

Copyright

Alex Kovrigin 2018

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 <QByteArray>
#include <QImage>

...

ModelPC * model = new ModelPC(ver);
// ver is version of the app, used to check the data structure version
// ver is type long and is calculated as if version is "x.y.z" => ver = x * 65536 + y * 256 + z
// Default parameter is 2^17 (2.0.0)

// Connecting signals

// Essential ones

model->saveData(QByteArray data)
// Used to return the retrieved data

model->saveImage(QImage * image)
// Used to return the modified image

// Extra ones

model->alertView(QString message, bool isWarning)
// Used for messages to be shown to users

model->setProgress(int val)
// Used to show user the progress of embedding
// -1 indicates the creation of some kind of progress dialog
// from 0 to 100 shows the progress
// 101 indicates that progress dialog should be closed
```

Avaible methods

Essential ones

start

Used for embedding

Parameters: data Data to be encrypted _image Image to be encrypted into. _bitsUsed Bits per byte, see also ModelPC::bitsUsed key Key, if default (empty), random key of 64 charachters will be generated. mode Mode of encryption

```
model->start(QByteArray data, QImage image, int mode = 0, QString key = "", int _bitsUsed = 8);
```

decrypt

Used for de-embedding

Parameters: image Image to be decrypted.

```
model->decrypt(QImage * image);
```

Extra ones

encrypt

Used for embedding but with data already packed with stuff like version, file size, aes key, etc. Used in PictureCrypt project

Parameters:

encr_data Data to be embbed to an image. image Image to be embbed into. mode Mode of encryption

```
model->encrypt(QByteArray encr_data, QImage * image, int mode = 0);
```

fail

Used for stopping the embedding or de-embedding proccess Parameters:

message Message for user

```
model->fail(QString message);
```

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
- `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

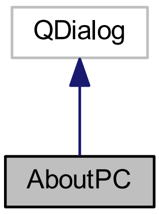
Class Documentation

9.1 AboutPC Class Reference

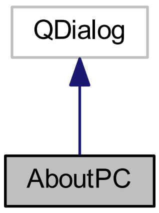
The AboutPC class The About Page dialog.

```
#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 )
```

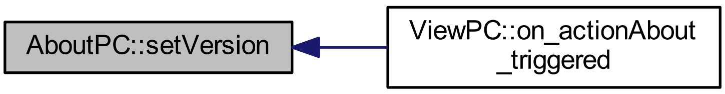
AboutPC::setVersion Function to set the version display.

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:

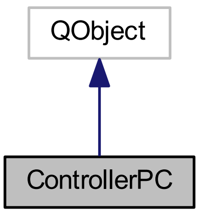
- C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.cpp

9.2 ControllerPC Class Reference

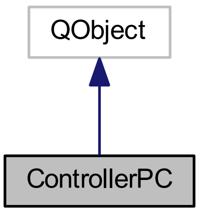
The ControllerPC class Controller class, which controls View and Model layers.

```
#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.

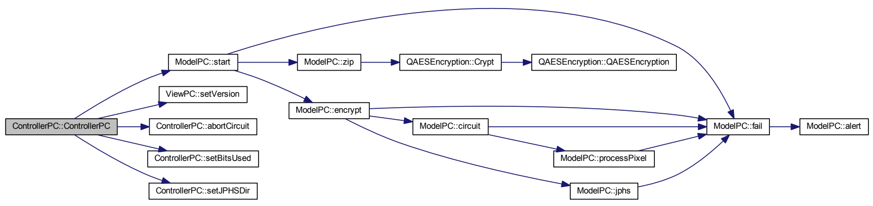
Controller class

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

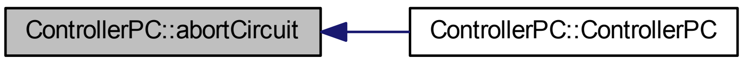
9.2.3.1 abortCircuit

```
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]
```

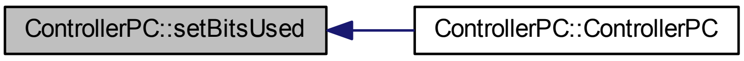
ControllerPC::setBitsUsed Slot to set ModelPC::bitsUsed.

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]
```

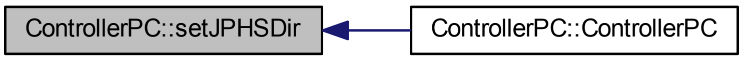
ControllerPC::setJPHSDir Sets JPHS default dir.

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:

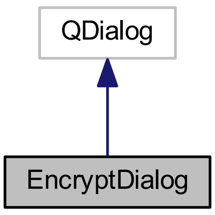
- C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.cpp

9.3 EncryptDialog Class Reference

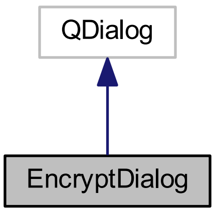
The EncryptDialog class Class to get the image and key to store secret info.

`#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]
```

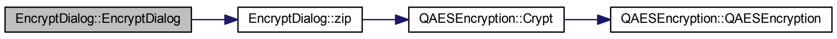
EncryptDialog::EncryptDialog Constructor of the class. Input data is saved here and some variables are set here.

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

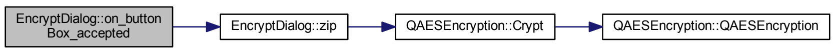
9.3.3.1 on_buttonBox_accepted

```
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:


EncryptDialog::on_button Box_accepted
EncryptDialog::zip
QAESEncryption::Crypt
QAESEncryption::QAESEncryption

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

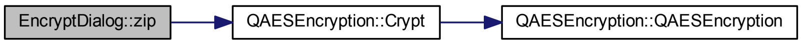
Returns Compressed data.

See also

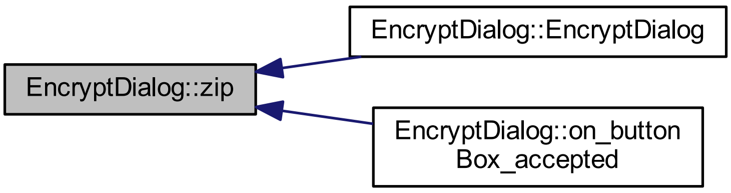
ModelPC::unzip

Definition at line 49 of file encryptdialog.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


EncryptDialog::EncryptDialog
EncryptDialog::zip
EncryptDialog::on_button Box_accepted

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:

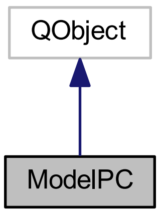
- C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.cpp

9.4 ModelPC Class Reference

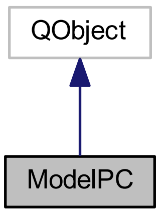
The ModelPC class Model Layer of the app. Controled by ControllerPC.

```
#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

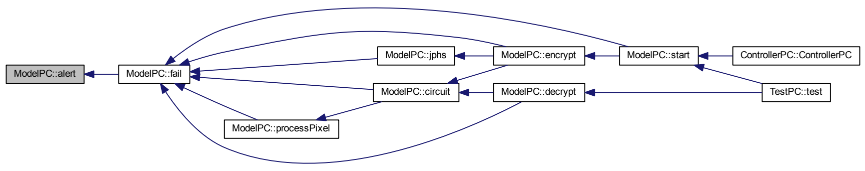
message	Message to be transmitted.
isWarning	Flag if message is critical.

See also

ViewPC::alert

Definition at line 586 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.

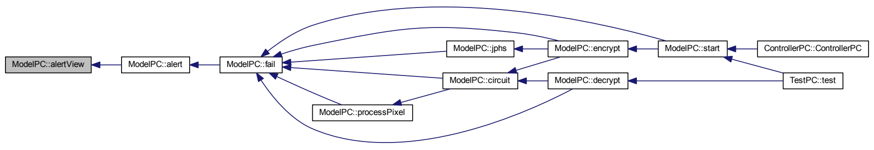
Parameters

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

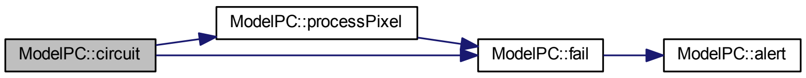
image	Image to be processed.
data	Data to be processed.
countBytes	Number of bytes to be read or written.

See also

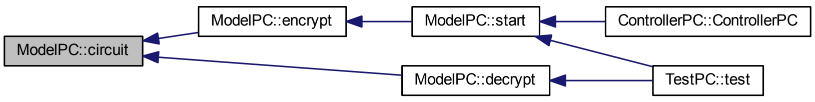
ModelPC::processPixel

Definition at line 290 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

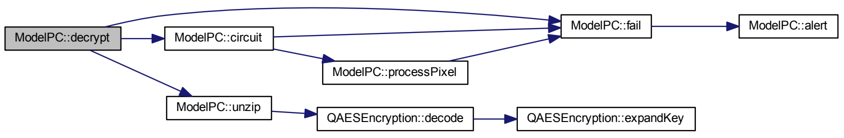
_error	Error output

See also

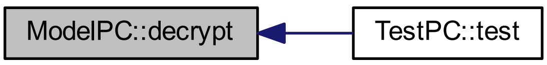
ViewPC::on_startButton_clicked, ModelPC::encrypt, ModelPC::circuit

Definition at line 140 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

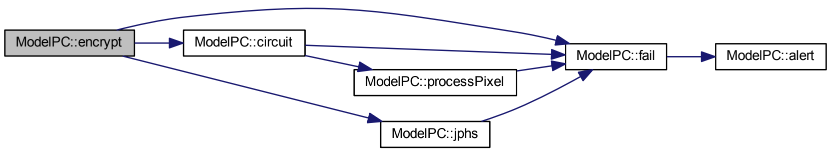
Returns image with embedded data.

See also

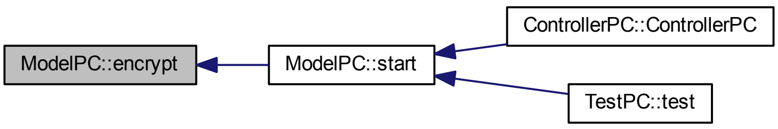
ViewPC::on_startButton_clicked, ModelPC::decrypt, ModelPC::circuit

Definition at line 90 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]
```

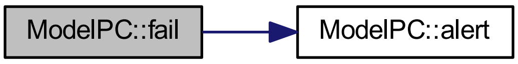
ModelPC::fail Slot to stop execution of cryption.

Parameters

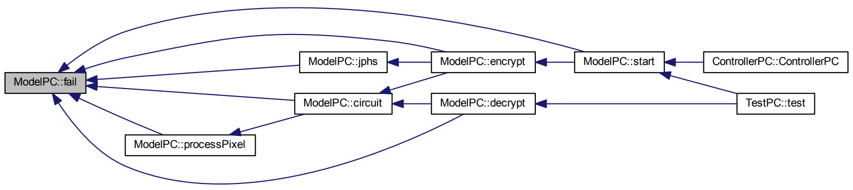
message	Message for user

Definition at line 217 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]
```

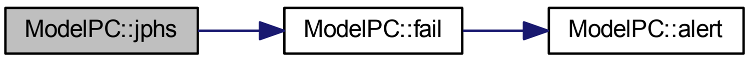
ModelPC::jphs JPHS function to use jphide and jpseek (currently under development)

Parameters

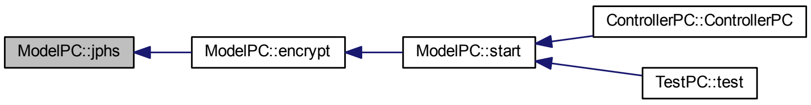
image	Image for embedding
data	Data

Definition at line 229 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]
```

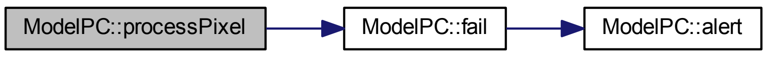
ModelPC::processPixel Processes every pixel. Reads its contains or writes data.

Parameters

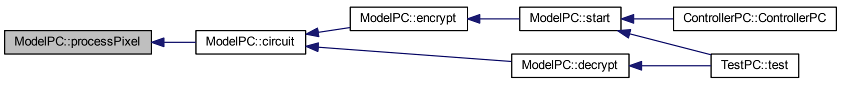
pos	Position of pixel
were	Vector array containing pixels, that were already processed.
isEncrypt	Mode of operation. If true encryption operations will continue, else the decryption ones.

Definition at line 432 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

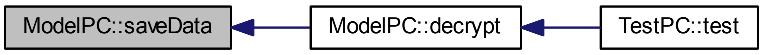
```
ModelPC::saveData (
             QByteArray data )  [signal]
```

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

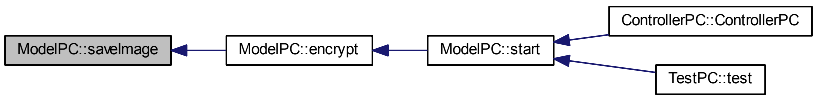
```
ModelPC::saveImage (
             QImage * image )  [signal]
```

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.

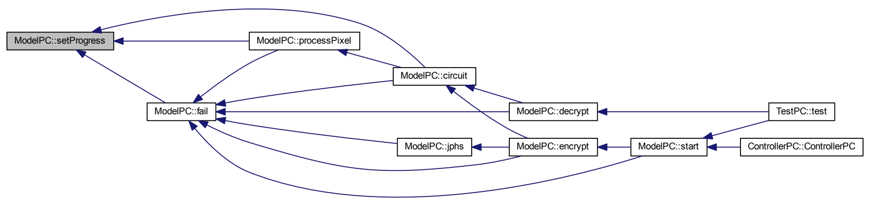
Parameters

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

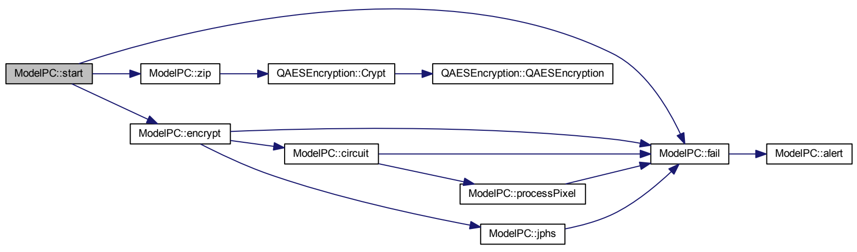
data	Data for embedding
image	Image for embedding
mode	Mode for embedding
key	Key for extra encryption (if empty, key will be auto-generated)
_bitsUsed	Bits per byte (see ModelPC::bitsUsed)
_error	Error output

Returns

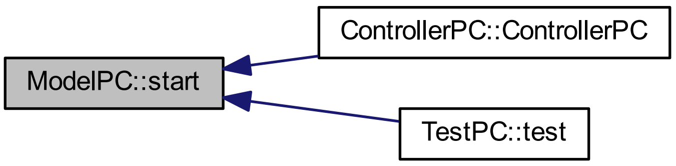
Returns image with embedded data

Definition at line 34 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

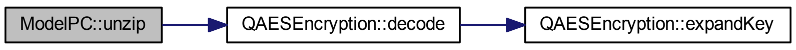
Returns data

See also

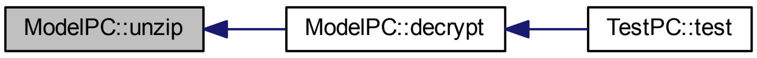
EncryptDialog::zip, ModelPC::decrypt, ModelPC::zip

Definition at line 525 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

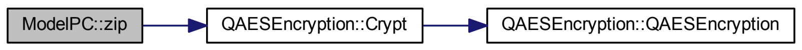
Returns decrypted data

See also

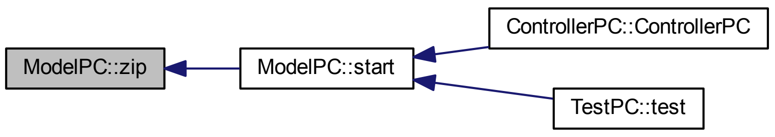
ModelPC::start, ModelPC::encrypt, ModelPC::unzip

Definition at line 542 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:

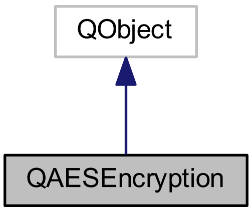
- C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.cpp

9.5 QAESEncryption Class Reference

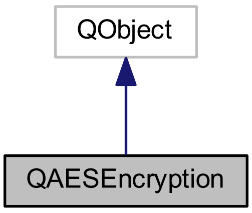
The QAESEncryption class Small and portable AES encryption class for Qt. Supports all key sizes - 128/192/256 bits - ECB, CBC, CFB and OFB modes. Class made entirely by bricke. Github: https://github.↵com/bricke/Qt-AES.

```
#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 QByte↩ Array &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 Q↩ ByteArray &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.

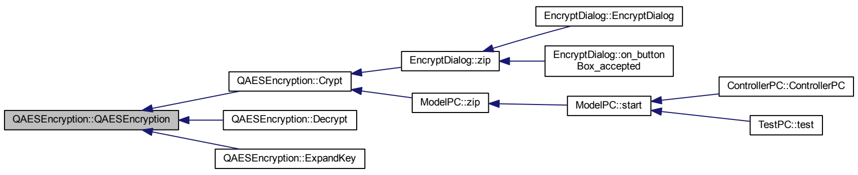
9.5.3 Constructor & Destructor Documentation

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:


EncryptDialog::EncryptDialog
EncryptDialog::zip
EncryptDialog::on_button Box_accepted
QAESEncryption::Crypt
ControllerPC::ControllerPC
ModelPC::zip
ModelPC::start
QAESEncryption::QAESEncryption
QAESEncryption::Decrypt
TestPC::test
QAESEncryption::ExpandKey

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

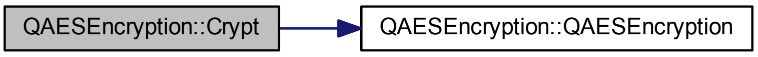
Returns encrypted data

See also

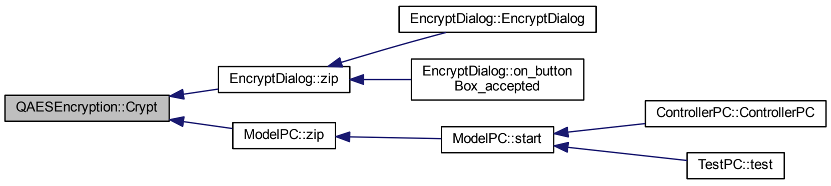
QAESEncryption::encode, QAESEncryption::Decrypt

Definition at line 6 of file qaesencryption.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


EncryptDialog::EncryptDialog
EncryptDialog::zip
EncryptDialog::on_button Box_accepted
QAESEncryption::Crypt
ControllerPC::ControllerPC
ModelPC::zip
ModelPC::start
TestPC::test

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

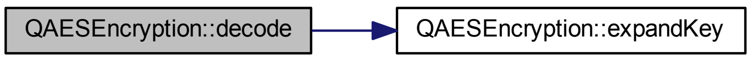
Returns decoded data

See also

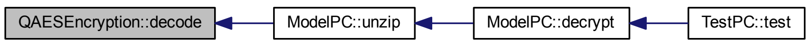
QAESEncryption::Decrypt, QAESEncryption::encode

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

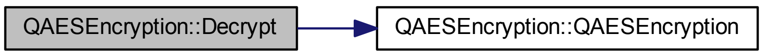
Returns Decrypted data

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

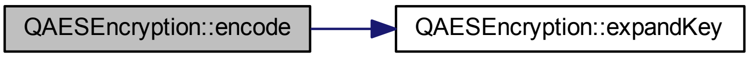
Returns encoded data

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

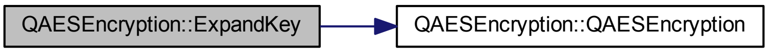
Returns expanded key (I guess)

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

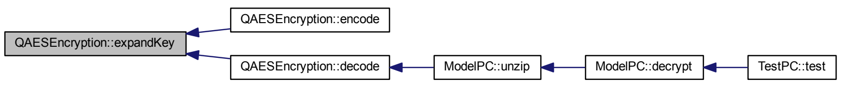
Returns expanded key (I guess)

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:

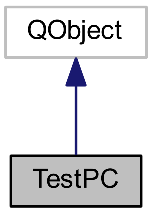
- C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.cpp

9.6 TestPC Class Reference

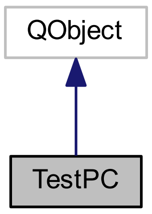
The TestPC class AutoTest for ModelPC Currently used in main.cpp.

```
#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 23 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

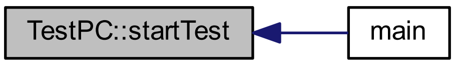
Returns success of all tests

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

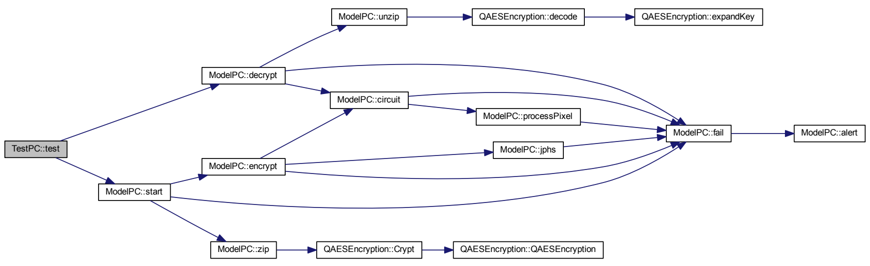
Returns if test is successful

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:

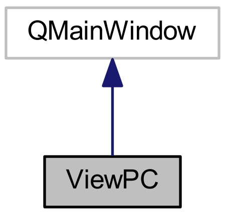
- C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.cpp

9.7 ViewPC Class Reference

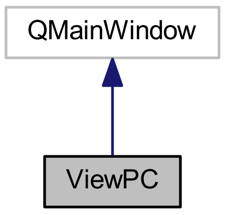
The ViewPC class View layer of the app. Controls EncryptDialog and ProgressDialog.

```
#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.

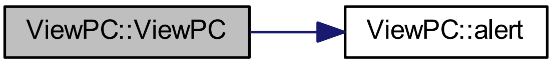
9.7.2 Constructor & Destructor Documentation

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 27 of file viewpc.cpp.

9.7.3 Member Function Documentation

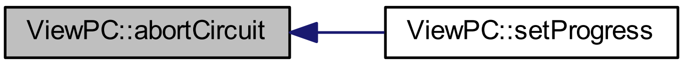
9.7.3.1 abortCircuit

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

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

Definition at line 220 of file viewpc.cpp.

Here is the caller graph for this function:

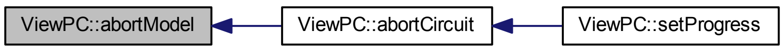


9.7.3.2 abortModel

```
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]
```

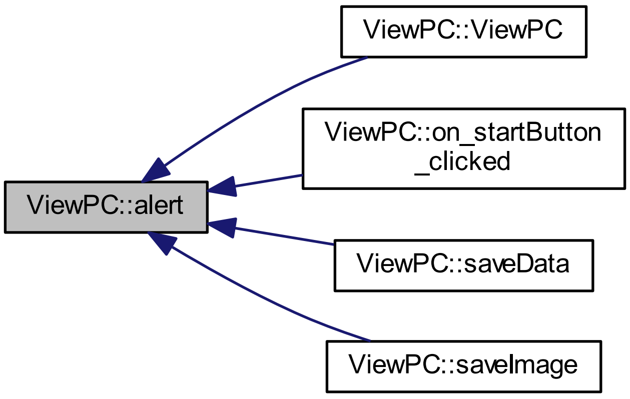
ViewPC::alert Slot to create QMessageBox with message.

Parameters

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

Definition at line 134 of file viewpc.cpp.

Here is the caller graph for this function:



9.7.3.4 decrypt

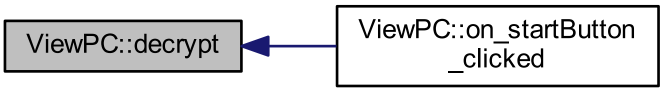
```
ViewPC::decrypt (
            QImage * _image )  [signal]
```

decrypt Signal calling ModelPC::decrypt

Parameters

_image	Image for decryption

Here is the caller graph for this function:



9.7.3.5 encrypt

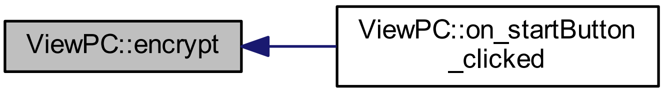
```
ViewPC::encrypt (
             QByteArray data,
             QImage * image,
             int mode )  [signal]
```

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:



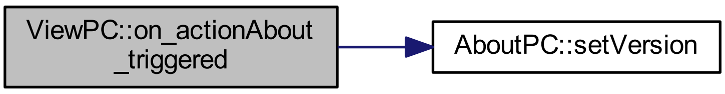
9.7.3.6 on_actionAbout_triggered

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

ViewPC::on_actionAbout_triggered Opens about page.

Definition at line 255 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 265 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 46 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

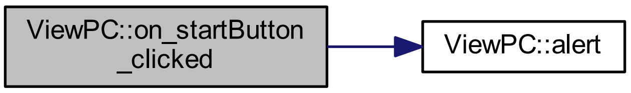
Then the EncryptDialog opens and image and key is selected. Then the ViewPC::encrypt signal is called to start ModelPC::encrypt

9.7.5 Decrypting

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

Definition at line 68 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

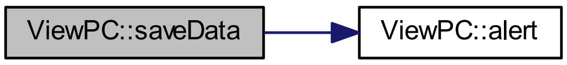
Edata	Encrypted data to be saved.

See also

ModelPC::encrypt

Definition at line 155 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

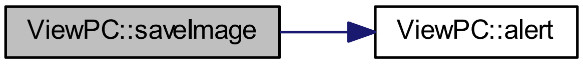
image	Image to be saved.

See also

ModelPC::decrypt

Definition at line 176 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

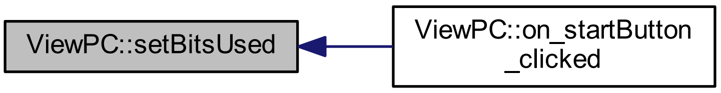
Parameters

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 233 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

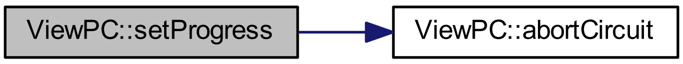
val	New value of the dialog. If -1, creates ProgressDialog, if 101 closes the dialog.

See also

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

Definition at line 194 of file viewpc.cpp.

Here is the call graph for this function:



9.7.5.7 setVersion

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

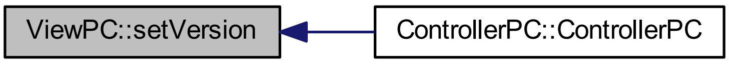
ViewPC::setVersion Set the version of the app from ControllerPC.

Parameters

version	Version as QString

Definition at line 242 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/viewpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.cpp

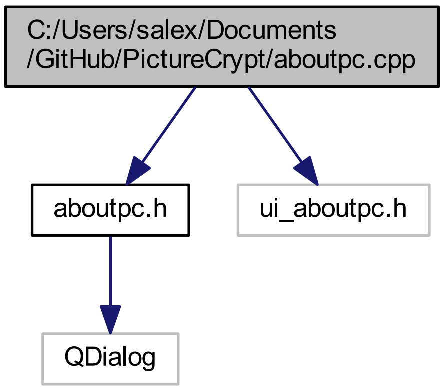
Chapter 10

File Documentation

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

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

Include dependency graph for aboutpc.cpp:

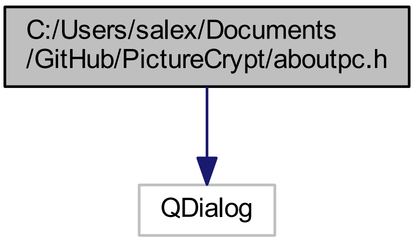


10.2 aboutpc.cpp

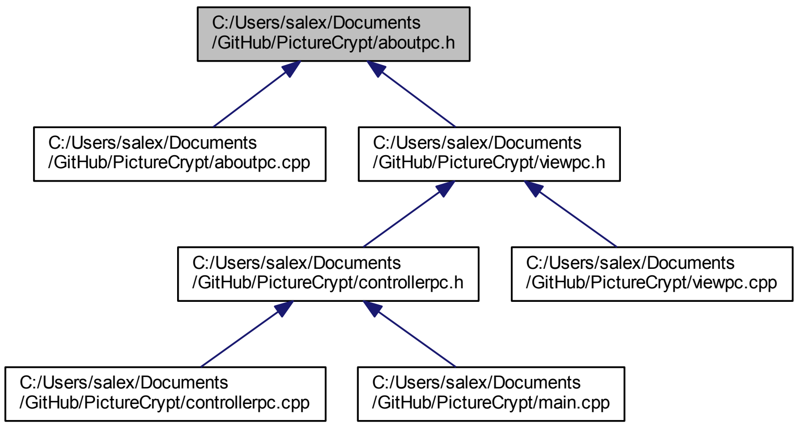
```
#include "aboutpc.h"
#include "ui_aboutpc.h"

AboutPC::AboutPC(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::AboutPC)
{
    ui->setupUi(this);
}

AboutPC::~AboutPC()
{
    delete ui;
}
void AboutPC::setVersion(QString version)
{
    ui->versionLabel->setText("Version " + version);
}
```

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

```
#include <QDialog>
```
Include dependency graph for aboutpc.h:



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


C:/Users/salex/Documents
/GitHub/PictureCrypt/aboutpc.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/aboutpc.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/viewpc.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/controllerpc.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/viewpc.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/controllerpc.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/main.cpp

Classes

- class AboutPC

  The AboutPC class The About Page dialog.

Namespaces

- Ui

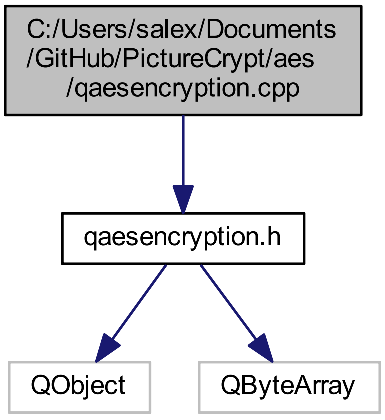
10.4 aboutpc.h

```
#ifndef ABOUTPC_H
#define ABOUTPC_H

#include <QDialog>

namespace Ui {
class AboutPC;
}
class AboutPC : public QDialog
{
    Q_OBJECT

public:
    explicit AboutPC(QWidget *parent = 0);
    ~AboutPC();
    void setVersion(QString version);

private:
    Ui::AboutPC *ui;
};

#endif // ABOUTPC_H
```

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

`#include "qaesencryption.h"`

Include dependency graph for qaesencryption.cpp:


C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.cpp
qaesencryption.h
QObject
QByteArray

Functions

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

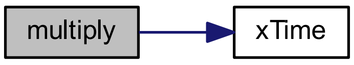
10.5.1 Function Documentation

10.5.1.1 multiply()

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

Definition at line 57 of file qaesencryption.cpp.

Here is the call graph for this function:

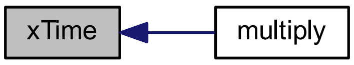


10.5.1.2 xTime()

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

Definition at line 53 of file qaesencryption.cpp.

Here is the caller graph for this function:



10.6 qaesencryption.cpp

```
#include "qaesencryption.h"

/*
 * Static Functions
 * */
QByteArray QAESEncryption::Crypt(QAESEncryption::Aes level,
     QAESEncryption::Mode mode, const QByteArray &rawText,
                                  const QByteArray &key, const QByteArray &iv,
     QAESEncryption::Padding padding)
{
    return QAESEncryption(level, mode, padding).encode(rawText,
     key, iv);
}
```

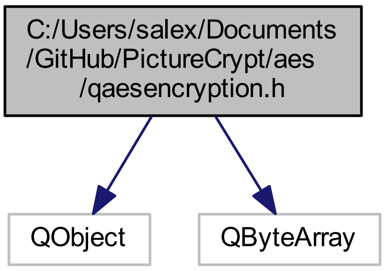
```

QByteArray QAESEncryption::Decrypt(QAESEncryption::Aes level,
QAESEncryption::Mode mode, const QByteArray &rawText,
                                  const QByteArray &key, const QByteArray &iv,
QAESEncryption::Padding padding)
{
     return QAESEncryption(level, mode, padding).decode(rawText,
key, iv);
}

QByteArray QAESEncryption::ExpandKey(
QAESEncryption::Aes level, QAESEncryption::Mode
mode, const QByteArray &key)
{
     return QAESEncryption(level, mode).expandKey(key);
}

QByteArray QAESEncryption::RemovePadding(const QByteArray &rawText,
QAESEncryption::Padding padding)
{
    QByteArray ret(rawText);
    switch (padding)
    {
    case Padding::ZERO:
        //Works only if the last byte of the decoded array is not zero
        while (ret.at(ret.length()-1) == 0x00)
            ret.remove(ret.length()-1, 1);
        break;
    case Padding::PKCS7:
        ret.remove(ret.length() - ret.at(ret.length()-1), ret.at(ret.length()-1));
        break;
    case Padding::ISO:
        ret.truncate(ret.lastIndexOf(0x80));
        break;
    default:
        //do nothing
        break;
    }
    return ret;
}
/*
 * End Static function declarations
 * */

/*
 * Inline Functions
 * */

inline quint8 xTime(quint8 x){
  return ((x<<1) ^ (((x>>7) & 1) * 0x1b));
}

inline quint8 multiply(quint8 x, quint8 y){
  return (((y & 1) * x) ^ ((y>>1 & 1) * xTime(x)) ^ ((y>>2 & 1) * xTime(
xTime(x))) ^ ((y>>3 & 1)
            * xTime(xTime(xTime(x)))) ^ ((y>>4 & 1) * xTime(
xTime(xTime(xTime(x))))));
}

/*
 * End Inline functions
 * */


QAESEncryption::QAESEncryption(Aes level, Mode
mode,
                               Padding padding)
    : m_nb(4), m_blocklen(16), m_level(level), m_mode(mode), m_padding(padding)
{
    m_state = NULL;

    switch (level)
    {
    case AES_128: {
        AES128 aes;
        m_nk = aes.nk;
        m_keyLen = aes.keylen;
        m_nr = aes.nr;
        m_expandedKey = aes.expandedKey;
        }
        break;
    case AES_192: {
        AES192 aes;
        m_nk = aes.nk;
        m_keyLen = aes.keylen;
        m_nr = aes.nr;
        m_expandedKey = aes.expandedKey;
        }
        break;
    case AES_256: {
        AES256 aes;
        m_nk = aes.nk;
        m_keyLen = aes.keylen;
        m_nr = aes.nr;
        m_expandedKey = aes.expandedKey;
        }
        break;
    default: {
        AES128 aes;
        m_nk = aes.nk;
        m_keyLen = aes.keylen;
        m_nr = aes.nr;
        m_expandedKey = aes.expandedKey;
        }
        break;
    }

}
QByteArray QAESEncryption::getPadding(int currSize, int alignment)
{
    int size = (alignment - currSize % alignment) % alignment;
    if (size == 0) return QByteArray();
    switch(m_padding)
    {
    case Padding::ZERO:
        return QByteArray(size, 0x00);
        break;
    case Padding::PKCS7:
        return QByteArray(size,size);
        break;
    case Padding::ISO:
        return QByteArray (size-1, 0x00).prepend(0x80);
        break;
    default:
        return QByteArray(size, 0x00);
        break;
    }
    return QByteArray(size, 0x00);
}

QByteArray QAESEncryption::expandKey(const QByteArray &
key)
{
  int i, k;
  quint8 tempa[4]; // Used for the column/row operations
  QByteArray roundKey(key);

  // The first round key is the key itself.
  // ...

  // All other round keys are found from the previous round keys.
  //i == Nk
  for(i = m_nk; i < m_nb * (m_nr + 1); i++)
  {
    tempa[0] = (quint8) roundKey.at((i-1) * 4 + 0);
    tempa[1] = (quint8) roundKey.at((i-1) * 4 + 1);
    tempa[2] = (quint8) roundKey.at((i-1) * 4 + 2);
    tempa[3] = (quint8) roundKey.at((i-1) * 4 + 3);

    if (i % m_nk == 0)
    {
        // This function shifts the 4 bytes in a word to the left once.
        // [a0,a1,a2,a3] becomes [a1,a2,a3,a0]

        // Function RotWord()
        k = tempa[0];
        tempa[0] = tempa[1];
        tempa[1] = tempa[2];
        tempa[2] = tempa[3];
        tempa[3] = k;

        // Function Subword()
        tempa[0] = getSBoxValue(tempa[0]);
        tempa[1] = getSBoxValue(tempa[1]);
        tempa[2] = getSBoxValue(tempa[2]);
        tempa[3] = getSBoxValue(tempa[3]);

        tempa[0] =  tempa[0] ^ Rcon[i/m_nk];
    }
    if (m_level == AES_256 && i % m_nk == 4)
    {
        // Function Subword()
        tempa[0] = getSBoxValue(tempa[0]);
        tempa[1] = getSBoxValue(tempa[1]);
```

```
      tempa[2] = getSBoxValue(tempa[2]);
      tempa[3] = getSBoxValue(tempa[3]);
    }
    roundKey.insert(i * 4 + 0, (quint8) roundKey.at((i - m_nk) * 4 + 0) ^ tempa[0]);
    roundKey.insert(i * 4 + 1, (quint8) roundKey.at((i - m_nk) * 4 + 1) ^ tempa[1]);
    roundKey.insert(i * 4 + 2, (quint8) roundKey.at((i - m_nk) * 4 + 2) ^ tempa[2]);
    roundKey.insert(i * 4 + 3, (quint8) roundKey.at((i - m_nk) * 4 + 3) ^ tempa[3]);
  }
  return roundKey;
}

// This function adds the round key to state.
// The round key is added to the state by an XOR function.
void QAESEncryption::addRoundKey(const quint8 round, const QByteArray expKey)
{
  QByteArray::iterator it = m_state->begin();
  for(int i=0; i < 16; ++i)
      it[i] = (quint8) it[i] ^ (quint8) expKey.at(round * m_nb * 4 + (i/4) * m_nb + (i%4));
}

// The SubBytes Function Substitutes the values in the
// state matrix with values in an S-box.
void QAESEncryption::subBytes()
{
  QByteArray::iterator it = m_state->begin();
  for(int i = 0; i < 16; i++)
    it[i] = getSBoxValue((quint8) it[i]);
}

// The ShiftRows() function shifts the rows in the state to the left.
// Each row is shifted with different offset.
// Offset = Row number. So the first row is not shifted.
void QAESEncryption::shiftRows()
{
    QByteArray::iterator it = m_state->begin();
    quint8 temp;
    //Keep in mind that QByteArray is column-driven!!

     //Shift 1 to left
    temp   = (quint8)it[1];
    it[1]  = (quint8)it[5];
    it[5]  = (quint8)it[9];
    it[9]  = (quint8)it[13];
    it[13] = (quint8)temp;

    //Shift 2 to left
    temp   = (quint8)it[2];
    it[2]  = (quint8)it[10];
    it[10] = (quint8)temp;
    temp   = (quint8)it[6];
    it[6]  = (quint8)it[14];
    it[14] = (quint8)temp;

    //Shift 3 to left
    temp   = (quint8)it[3];
    it[3]  = (quint8)it[15];
    it[15] = (quint8)it[11];
    it[11] = (quint8)it[7];
    it[7]  = (quint8)temp;
}

// MixColumns function mixes the columns of the state matrix
//optimized!!
void QAESEncryption::mixColumns()
{
  QByteArray::iterator it = m_state->begin();
  quint8 tmp, tm, t;

  for(int i = 0; i < 16; i += 4){
    t       = (quint8)it[i];
    tmp     =  (quint8)it[i] ^ (quint8)it[i+1] ^ (quint8)it[i+2] ^ (quint8)it[i+3] ;

    tm      = xTime( (quint8)it[i] ^ (quint8)it[i+1] );
    it[i]   = (quint8)it[i] ^ (quint8)tm ^ (quint8)tmp;

    tm      = xTime( (quint8)it[i+1] ^ (quint8)it[i+2]);
    it[i+1] = (quint8)it[i+1] ^ (quint8)tm ^ (quint8)tmp;

    tm      = xTime( (quint8)it[i+2] ^ (quint8)it[i+3]);
    it[i+2] =(quint8)it[i+2] ^ (quint8)tm ^ (quint8)tmp;

    tm      = xTime((quint8)it[i+3] ^ (quint8)t);
    it[i+3] =(quint8)it[i+3] ^ (quint8)tm ^ (quint8)tmp;
  }
}

// MixColumns function mixes the columns of the state matrix.
// The method used to multiply may be difficult to understand for the inexperienced.
// Please use the references to gain more information.
void QAESEncryption::invMixColumns()
{
  QByteArray::iterator it = m_state->begin();
  quint8 a,b,c,d;
  for(int i = 0; i < 16; i+=4){
    a = (quint8) it[i];
    b = (quint8) it[i+1];
    c = (quint8) it[i+2];
    d = (quint8) it[i+3];

    it[i]   = (quint8) (multiply(a, 0x0e) ^ multiply(b, 0x0b) ^
multiply(c, 0x0d) ^ multiply(d, 0x09));
    it[i+1] = (quint8) (multiply(a, 0x09) ^ multiply(b, 0x0e) ^
multiply(c, 0x0b) ^ multiply(d, 0x0d));
    it[i+2] = (quint8) (multiply(a, 0x0d) ^ multiply(b, 0x09) ^
multiply(c, 0x0e) ^ multiply(d, 0x0b));
    it[i+3] = (quint8) (multiply(a, 0x0b) ^ multiply(b, 0x0d) ^
multiply(c, 0x09) ^ multiply(d, 0x0e));
  }
}

// The SubBytes Function Substitutes the values in the
// state matrix with values in an S-box.
void QAESEncryption::invSubBytes()
{
    QByteArray::iterator it = m_state->begin();
    for(int i = 0; i < 16; ++i)
        it[i] = getSBoxInvert((quint8) it[i]);
}

void QAESEncryption::invShiftRows()
{
    QByteArray::iterator it = m_state->begin();
    uint8_t temp;

    //Keep in mind that QByteArray is column-driven!!

    //Shift 1 to right
    temp   = (quint8)it[13];
    it[13] = (quint8)it[9];
    it[9]  = (quint8)it[5];
    it[5]  = (quint8)it[1];
    it[1]  = (quint8)temp;

    //Shift 2
    temp   = (quint8)it[10];
    it[10] = (quint8)it[2];
    it[2]  = (quint8)temp;
    temp   = (quint8)it[14];
    it[14] = (quint8)it[6];
    it[6]  = (quint8)temp;

    //Shift 3
    temp   = (quint8)it[15];
    it[15] = (quint8)it[3];
    it[3]  = (quint8)it[7];
    it[7]  = (quint8)it[11];
    it[11] = (quint8)temp;
}

QByteArray QAESEncryption::byteXor(const QByteArray &a, const QByteArray &b)
{
  QByteArray::const_iterator it_a = a.begin();
  QByteArray::const_iterator it_b = b.begin();
  QByteArray ret;

  //for(int i = 0; i < m_blocklen; i++)
  for(int i = 0; i < std::min(a.size(), b.size()); i++)
      ret.insert(i,it_a[i] ^ it_b[i]);

  return ret;
}

// Cipher is the main function that encrypts the PlainText.
QByteArray QAESEncryption::cipher(const QByteArray &expKey, const QByteArray &in)
{

  //m_state is the input buffer...
  QByteArray output(in);
  m_state = &output;

  // Add the First round key to the state before starting the rounds.
  addRoundKey(0, expKey);

  // There will be Nr rounds.
  // The first Nr-1 rounds are identical.
  // These Nr-1 rounds are executed in the loop below.
  for(quint8 round = 1; round < m_nr; ++round){
    subBytes();
    shiftRows();
    mixColumns();
    addRoundKey(round, expKey);
  }

  // The last round is given below.
  // The MixColumns function is not here in the last round.
  subBytes();
  shiftRows();
  addRoundKey(m_nr, expKey);

  return output;
}

QByteArray QAESEncryption::invCipher(const QByteArray &expKey, const QByteArray &in)
{
    //m_state is the input buffer.... handle it!
    QByteArray output(in);
    m_state = &output;

    // Add the First round key to the state before starting the rounds.
    addRoundKey(m_nr, expKey);

    // There will be Nr rounds.
    // The first Nr-1 rounds are identical.
    // These Nr-1 rounds are executed in the loop below.
    for(quint8 round=m_nr-1; round>0 ; round--){
        invShiftRows();
        invSubBytes();
        addRoundKey(round, expKey);
        invMixColumns();
    }

    // The last round is given below.
    // The MixColumns function is not here in the last round.
    invShiftRows();
    invSubBytes();
    addRoundKey(0, expKey);

    return output;
}

QByteArray QAESEncryption::encode(const QByteArray &rawText, const QByteArray &
key, const QByteArray &iv)
{
    if (m_mode >= CBC && (iv.isNull() || iv.size() != m_blocklen))
       return QByteArray();

    QByteArray ret;
    QByteArray expandedKey = expandKey(key);
    QByteArray alignedText(rawText);

    //Fill array with padding
    alignedText.append(getPadding(rawText.size(), m_blocklen));

    switch(m_mode)
    {
    case ECB:
        for(int i=0; i < alignedText.size(); i+= m_blocklen)
            ret.append(cipher(expandedKey, alignedText.mid(i, m_blocklen)));
        break;
    case CBC: {
            QByteArray ivTemp(iv);
            for(int i=0; i < alignedText.size(); i+= m_blocklen) {
                alignedText.replace(i, m_blocklen, byteXor(alignedText.mid(i, m_blocklen),ivTemp));
                ret.append(cipher(expandedKey, alignedText.mid(i, m_blocklen)));
                ivTemp = ret.mid(i, m_blocklen);
            }
        }
        break;
    case CFB: {
            ret.append(byteXor(alignedText.left(m_blocklen), cipher(expandedKey, iv)));
            for(int i=0; i < alignedText.size(); i+= m_blocklen) {
                if (i+m_blocklen < alignedText.size())
                    ret.append(byteXor(alignedText.mid(i+m_blocklen, m_blocklen),
                                       cipher(expandedKey, ret.mid(i, m_blocklen))));
            }
        }
        break;
    case OFB: {
            QByteArray ofbTemp;
            ofbTemp.append(cipher(expandedKey, iv));
            for (int i=m_blocklen; i < alignedText.size(); i += m_blocklen){
                ofbTemp.append(cipher(expandedKey, ofbTemp.right(m_blocklen)));
            }
            ret.append(byteXor(alignedText, ofbTemp));
        }
        break;
    default: break;
    }
    return ret;
}

QByteArray QAESEncryption::decode(const QByteArray &rawText, const QByteArray &key, const QByteArray &iv)
{
    if (m_mode >= CBC && (iv.isNull() || iv.size() != m_blocklen))
       return QByteArray();

    QByteArray ret;
    QByteArray expandedKey = expandKey(key);

    switch(m_mode)
    {
    case ECB:
        for(int i=0; i < rawText.size(); i+= m_blocklen)
            ret.append(invCipher(expandedKey, rawText.mid(i, m_blocklen)));
        break;
    case CBC: {
            QByteArray ivTemp(iv);
            for(int i=0; i < rawText.size(); i+= m_blocklen){
                ret.append(invCipher(expandedKey, rawText.mid(i, m_blocklen)));
                ret.replace(i, m_blocklen, byteXor(ret.mid(i, m_blocklen),ivTemp));
                ivTemp = rawText.mid(i, m_blocklen);
            }
        }
        break;
    case CFB: {
            ret.append(byteXor(rawText.mid(0, m_blocklen), cipher(expandedKey, iv)));
            for(int i=0; i < rawText.size(); i+= m_blocklen){
                if (i+m_blocklen < rawText.size()) {
                    ret.append(byteXor(rawText.mid(i+m_blocklen, m_blocklen),
                                       cipher(expandedKey, rawText.mid(i, m_blocklen))));
                }
            }
        }
        break;
    case OFB: {
        QByteArray ofbTemp;
        ofbTemp.append(cipher(expandedKey, iv));
        for (int i=m_blocklen; i < rawText.size(); i += m_blocklen){
            ofbTemp.append(cipher(expandedKey, ofbTemp.right(m_blocklen)));
        }
        ret.append(byteXor(rawText, ofbTemp));
    }
        break;
    default:
        //do nothing
        break;
    }
    return ret;
}

QByteArray QAESEncryption::removePadding(const QByteArray &rawText)
{
    QByteArray ret(rawText);
    switch (m_padding)
    {
    case Padding::ZERO:
        //Works only if the last byte of the decoded array is not zero
        while (ret.at(ret.length()-1) == 0x00)
            ret.remove(ret.length()-1, 1);
        break;
    case Padding::PKCS7:
        ret.remove(ret.length() - ret.at(ret.length()-1), ret.at(ret.length()-1));
        break;
    case Padding::ISO:
        ret.truncate(ret.lastIndexOf(0x80));
        break;
    default:
        //do nothing
        break;
    }
    return ret;
}
```

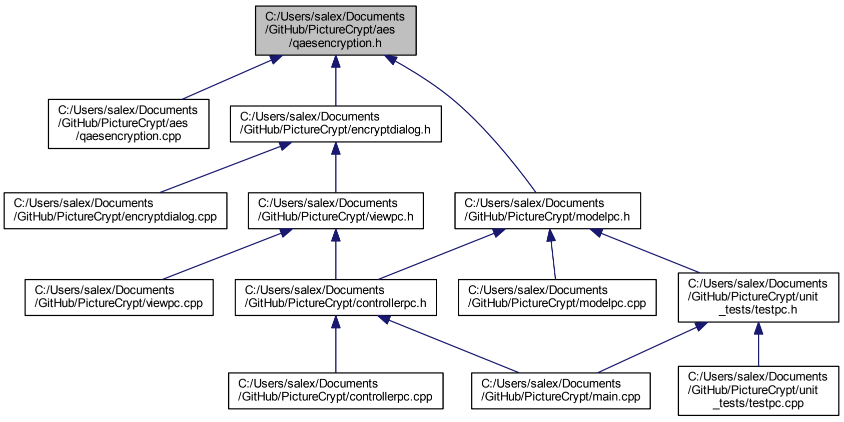
10.7 C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.h File Reference

```
#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`.

10.8 qaesencryption.h

```
#ifndef QAESENCRYPTION_H
#define QAESENCRYPTION_H

#include <QObject>
#include <QByteArray>

class QAESEncryption : public QObject
{
    Q_OBJECT
public:
    enum Aes {
        AES_128,
        AES_192,
        AES_256
    };
    enum Mode {
        ECB,
        CBC,
        CFB,
        OFB
    };

    enum Padding {
      ZERO,
      PKCS7,
      ISO
    };
    static QByteArray Crypt(QAESEncryption::Aes level, QAESEncryption::Mode mode, const QByteArray &rawText, const QByteArray &key,
                            const QByteArray &iv = NULL, QAESEncryption::Padding padding = QAESEncryption::ISO);
    static QByteArray Decrypt(QAESEncryption::Aes level, QAESEncryption::Mode mode, const QByteArray &rawText, const QByteArray &key,
                              const QByteArray &iv = NULL, QAESEncryption::Padding padding = QAESEncryption::ISO);
    static QByteArray ExpandKey(QAESEncryption::Aes level, QAESEncryption::Mode mode, const QByteArray &key);
    static QByteArray RemovePadding(const QByteArray &rawText, QAESEncryption::Padding padding);

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

signals:

public slots:

private:
    int m_nb;
    int m_blocklen;
    int m_level;
    int m_mode;
    int m_nk;
    int m_keyLen;
    int m_nr;
    int m_expandedKey;
    int m_padding;
    QByteArray* m_state;

    struct AES256{
        int nk = 8;
        int keylen = 32;
        int nr = 14;
        int expandedKey = 240;
    };

    struct AES192{
        int nk = 6;
        int keylen = 24;
        int nr = 12;
        int expandedKey = 209;
    };

    struct AES128{
        int nk = 4;
        int keylen = 16;
        int nr = 10;
        int expandedKey = 176;
    };

    quint8 getSBoxValue(quint8 num){return sbox[num];}
    quint8 getSBoxInvert(quint8 num){return rsbox[num];}

    void addRoundKey(const quint8 round, const QByteArray expKey);
    void subBytes();
    void shiftRows();
    void mixColumns();
    void invMixColumns();
    void invSubBytes();
    void invShiftRows();
    QByteArray getPadding(int currSize, int alignment);
    QByteArray cipher(const QByteArray &expKey, const QByteArray &plainText);
    QByteArray invCipher(const QByteArray &expKey, const QByteArray &plainText);
    QByteArray byteXor(const QByteArray &in, const QByteArray &iv);

    const quint8 sbox[256] =   {
      //0     1    2      3     4    5     6     7      8    9     A      B    C     D     E     F
      0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
      0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
      0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
      0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
      0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
      0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
      0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
      0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
      0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
      0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
      0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
      0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
      0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
      0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
      0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
      0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 };

    const quint8 rsbox[256] =
    { 0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
      0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
      0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
      0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
      0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
      0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
      0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
      0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
      0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
      0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
      0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
      0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
      0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
      0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
      0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
      0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d };

    // The round constant word array, Rcon[i], contains the values given by
    // x to th e power (i-1) being powers of x (x is denoted as {02}) in the field GF(2^8)
    // Only the first 14 elements are needed
    const quint8 Rcon[256] = {
        0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab/*, 0x4d, 0x9a,
        0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
        0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,
        0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,
        0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,
        0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,
        0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
        0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
        0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
        0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
        0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
        0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
        0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
        0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,
        0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,
            0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d
*/};
};

#endif // QAESENCRYPTION_H
```

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

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

```
# 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][homepage], version 1.4, available at
 [http://contributor-covenant.org/version/1/4][version]

[homepage]: http://contributor-covenant.org
[version]: http://contributor-covenant.org/version/1/4/
```

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

10.12 ErrorsDict.json

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

10.13 C:/Users/salex/Documents/GitHub/PictureCrypt/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

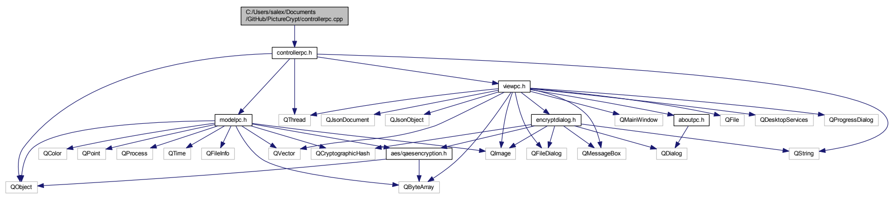
10.14 ErrorsDictSetup.py

```python
import json
filename = ’ErrorsDict.json’

raw = open(filename, ’r’)

data = json.load(raw)
print(’Existing data:’)
for key, value in data.items():
    print(key, value)

print(’------------’)
print(’Type new data’)

input_data = input()

while len(input_data):
    key, value = map(str, input_data.split(’-’))
    data[key] = value
    input_data = input()

with open(filename, ’w’) as f:
    json.dump(data, f, indent=4)
```

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

```
#include "controllerpc.h"
```

Include dependency graph for controllerpc.cpp:



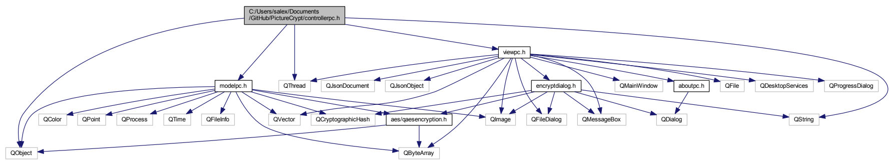
10.16 controllerpc.cpp

```
#include "controllerpc.h"

ControllerPC::ControllerPC()
{
    // Layer creation
    view = new ViewPC();
    model = new ModelPC();
    QThread * modelThread = new QThread();
    model->moveToThread(modelThread);
    modelThread->start();

    view->setVersion(model->versionString);
    view->show();
    // Layer Connection
    connect(view, SIGNAL(encrypt(QByteArray,QImage*,int)), model, SLOT(encrypt(QByteArray,QImage*,int)));
    connect(view, SIGNAL(decrypt(QImage*)), model, SLOT(decrypt(QImage*)));
    connect(view, SIGNAL(abortModel()), this, SLOT(abortCircuit()));
    connect(view, SIGNAL(setBitsUsed(int)), this, SLOT(setBitsUsed(int)));
    connect(view, SIGNAL(setJPHSDir(QString)), this, SLOT(setJPHSDir(QString)));

    connect(model, SIGNAL(alertView(QString,bool)), view, SLOT(alert(QString,bool)));
    connect(model, SIGNAL(saveData(QByteArray)), view, SLOT(saveData(QByteArray)));
    connect(model, SIGNAL(saveImage(QImage*)), view, SLOT(saveImage(QImage*)));
    connect(model, SIGNAL(setProgress(int)), view, SLOT(setProgress(int)));
}
void ControllerPC::abortCircuit()
{
    model->success = false;
}
void ControllerPC::setBitsUsed(int bitsUsed)
{
    model->bitsUsed = bitsUsed;
}
void ControllerPC::setJPHSDir(QString dir)
{
    model->defaultJPHSDir = dir;
}
```

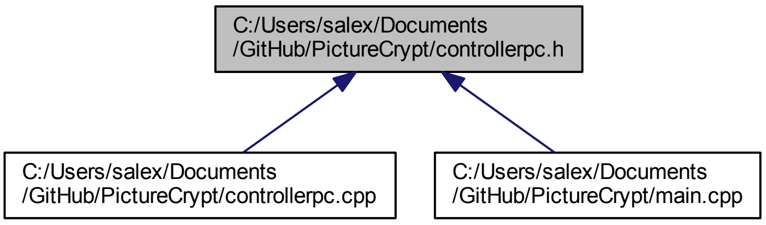
10.17 C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.h File Reference

```
#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:


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

Classes

- class ControllerPC

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

10.17.1 Detailed Description

Header of ControllerPC class

See also

ControllerPC, ModelPC, ViewPC

Definition in file controllerpc.h.

10.18 controllerpc.h

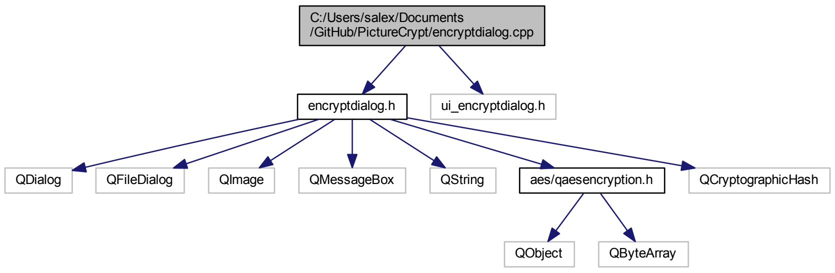
```
#ifndef CONTROLLERPC_H
#define CONTROLLERPC_H

#include <QObject>
#include <QString>
#include <QThread>

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

```
class ControllerPC : public QObject
{
    Q_OBJECT
public:
    ControllerPC();
    long int version;
    QString versionString;
public slots:
    void abortCircuit();
    void setBitsUsed(int bitsUsed);
    void setJPHSDir(QString dir);
private:
    ViewPC * view;
    ModelPC * model;
};

#endif // CONTROLLERPC_H
```

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

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

Include dependency graph for encryptdialog.cpp:



10.20 encryptdialog.cpp

```
#include "encryptdialog.h"
#include "ui_encryptdialog.h"
EncryptDialog::EncryptDialog(QByteArray _data, QWidget *parent) :
    QDialog(parent),
    ui(new Ui::EncryptDialog)
{
    ui->setupUi(this);
    data = _data;
    success = false;
    // UI setup
    ui->totalBytes->setText(QString::number(data.size()));
    key.clear();
    for(int i = 0; i < 24; i++)
        key.append(48 + qrand() % 75);
    val = 24;
    compr_data = zip();
    long long int compr_data_size = compr_data.size();
    ui->zippedBytes->setText(QString::number(compr_data_size));
    goodPercentage = false;
    bitsUsed = 8;
}

EncryptDialog::~EncryptDialog()
{
    delete ui;
}

void EncryptDialog::alert(QString text)
```

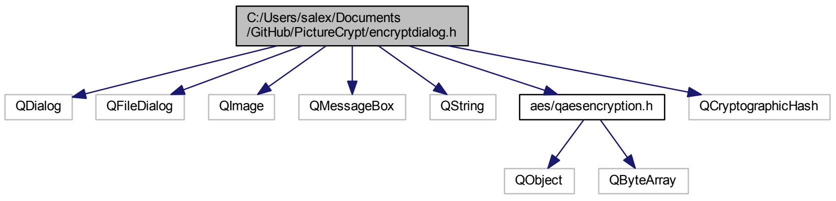
```
{
    QMessageBox t;
    t.setWindowTitle("Message");
    t.setIcon(QMessageBox::Warning);
    t.setWindowIcon(QIcon(":/mail.png"));
    t.setText(text);
    t.exec();
}
QByteArray EncryptDialog::zip()
{
    // Zip
    QByteArray c_data = qCompress(data, 9);
    // Encryption
    QByteArray hashKey = QCryptographicHash::hash(key.toUtf8(), QCryptographicHash::Sha256);
    return QAESEncryption::Crypt(QAESEncryption::AES_256,
QAESEncryption::ECB, c_data, hashKey);
}
void EncryptDialog::on_fileButton_clicked()
{
    // Selet file
    inputFileName = QFileDialog::getOpenFileName(this, tr("Open File"), "/", tr("Images (*.png
 *.xpm *.jpg *.jpeg)"));
    ui->fileLabel->setText(inputFileName);
    // Open image
    QImage img(inputFileName);
    image = img;
    // Get size
    size = img.width() * img.height();
    // UI setup
    long long int compr_data_size = compr_data.size();
    ui->zippedBytes->setText(QString::number(compr_data_size));
    if(inputFileName.isEmpty()) {
        ui->percentage->setText("");
        return;
    }
    double perc = (compr_data_size + 14 + val) * 100 / (size * 3) *
bitsUsed / 8;
    ui->percentage->setText(QString::number(perc) + "%");
    goodPercentage = perc < 70;
}
void EncryptDialog::on_buttonBox_accepted()
{
    if(!goodPercentage) {
        alert("Your encoding percentage is over 70% which is a bit ambiguous :(");
        success = false;
        return;
    }
    // Final zip
    compr_data = zip();
    success = true;
    close();
}
void EncryptDialog::on_buttonBox_rejected()
{
    success = false;
    close();
}
void EncryptDialog::on_horizontalSlider_valueChanged(int
value)
{
    // Key generator with value of charachters
    key.clear();
    for(int i = 0; i < value; i++)
        key.append(48 + qrand() % 75);
    val = value;
    ui->keyLabel->setText(QString::number(value));
}
void EncryptDialog::on_bitsSlider_valueChanged(int value)
{
    bitsUsed = value;
    ui->bitsUsedLbl->setText(QString::number(value));
    if(ui->percentage->text().isEmpty())
        return;
    double perc = (compr_data.size() + 14 + val) * 100 / (size * 3) * 8 /
bitsUsed;
    ui->percentage->setText(QString::number(perc) + "%");
}
```

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

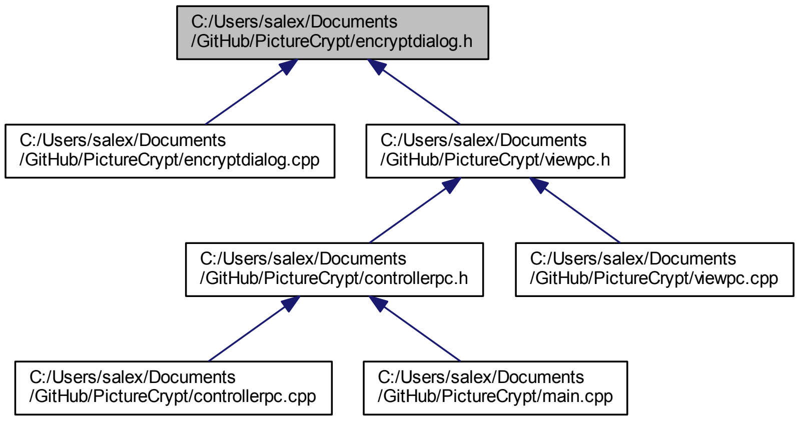
```
#include <QDialog>
#include <QFileDialog>
```

```
#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:


C:/Users/salex/Documents
/GitHub/PictureCrypt/encryptdialog.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/encryptdialog.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/viewpc.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/controllerpc.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/viewpc.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/controllerpc.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/main.cpp

Classes

- class EncryptDialog

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

Namespaces

- Ui

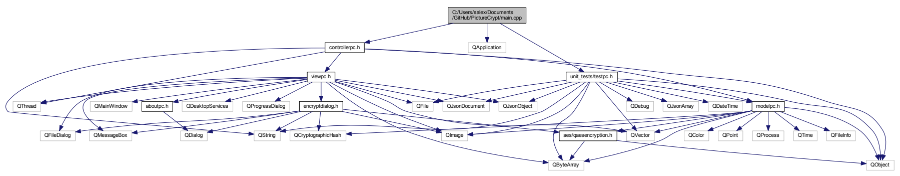
10.22 encryptdialog.h

```
#ifndef ENCRYPTDIALOG_H
#define ENCRYPTDIALOG_H

#include <QDialog>
#include <QFileDialog>
#include <QImage>
#include <QMessageBox>
#include <QString>

#include <aes/qaesencryption.h>
#include <QCryptographicHash>

namespace Ui {
class EncryptDialog;
}
class EncryptDialog : public QDialog
{
    Q_OBJECT

public:
    explicit EncryptDialog(QByteArray _data, QWidget *parent = 0);
    ~EncryptDialog();

public slots:
    void on_fileButton_clicked();

    void on_buttonBox_accepted();

    void on_buttonBox_rejected();

    void on_horizontalSlider_valueChanged(int
value);

public:
    QByteArray data;
    bool success;
    QByteArray compr_data;
    QString inputFileName;
    long long int size;
    QString key;
    bool goodPercentage;
    int val;
    int bitsUsed;
    QImage image;
    QByteArray zip();
private slots:
    void on_bitsSlider_valueChanged(int value);

private:
    Ui::EncryptDialog *ui;
    void alert(QString text);
};

#endif // ENCRYPTDIALOG_H
```

10.23 C:/Users/salex/Documents/GitHub/PictureCrypt/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[ ])

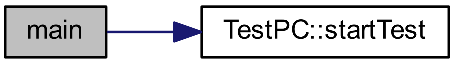
10.23.1 Function Documentation

10.23.1.1 main()

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

Definition at line 110 of file main.cpp.

Here is the call graph for this function:



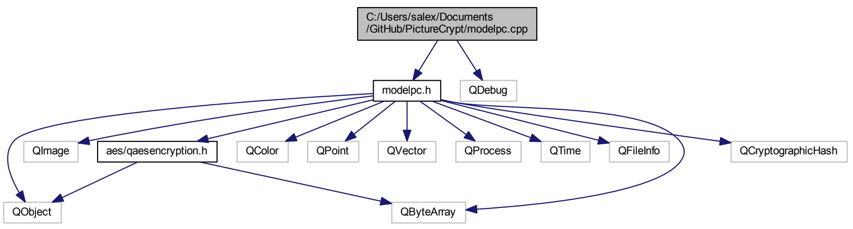
10.24 main.cpp

```
00001 #include "controllerpc.h"
00002 #include <QApplication>
00003 #include <unit_tests/testpc.h>
00110 int main(int argc, char *argv[])
00111 {
00112     QApplication a(argc, argv);
00113     TestPC test;
00114     bool success = test.startTest();
00115     if(success)
00116         ControllerPC w;
00117
00118     return a.exec();
00119 }
```

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

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

Include dependency graph for modelpc.cpp:



10.26 modelpc.cpp

```
#include "modelpc.h"
#include <QDebug>
ModelPC::ModelPC()
{
    // Version control
    versionString = "1.3.0";

    auto ver = versionString.split(".");
    version = ver[0].toInt() * pow(2, 16) + ver[1].toInt() * pow(2, 8) + ver[2].toInt();

    ver_byte = bytes(ver[0].toInt()) +
            bytes(ver[1].toInt()) +
            bytes(ver[2].toInt());
    // Random seed
    qsrand(randSeed());
}
QImage * ModelPC::start(QByteArray data, QImage * image, int
mode, QString key, int _bitsUsed, QString *_error)
{
    // Error management
    *_error = "ok";
    error = _error;

    if(data.isEmpty()) {
        fail("nodata");
        return nullptr;
    }
    if(image == nullptr || image->isNull()) {
        fail("nullimage");
        return nullptr;
    }
    if(_bitsUsed < 1 || _bitsUsed > 8) {
        fail("bitsWrong");
        return nullptr;
    }
    if(key.isEmpty()) {
        qsrand(randSeed());
        for(int i = 0; i < 32; i++)
            key.append(48 + qrand() % 75);
    }
    else if(key.size() > 255) {
        fail("bigkey");
        return nullptr;
    }
    long long usedBytes = data.size() + 14 + key.size();
    long long size = image->width() * image->height();
    if(usedBytes * 100 / (size * 3) * 8 / _bitsUsed > 70) {
        fail("muchdata");
        return nullptr;
    }

    curMode = mode;
    bitsUsed = _bitsUsed;

    QByteArray key_data = key.toUtf8();
    QByteArray zipped_data = zip(data, key_data);
    QByteArray encr_data = bytes(key_data.size()) + key_data + zipped_data;

    if(*error == "ok")
        return encrypt(encr_data, image, curMode, error);
    else
        return nullptr;
}

QImage * ModelPC::encrypt(QByteArray encr_data, QImage * image, int
mode, QString *_error)
{
    // Error management
    *_error = "ok";
    error = _error;

    // TODO Remove debug mode = 0
    mode = 0;

    if(encr_data.isEmpty()) {
        fail("nodata");
        return nullptr;
    }
    if(image == nullptr || image->isNull()) {
        fail("nullimage");
        return nullptr;
    }

    encr_data = ver_byte + encr_data;
    long long int countBytes = encr_data.size();
    curMode = mode;
    switch(curMode)
    {
    case 0:
        circuit(image, &encr_data, countBytes);
        break;
    case 1:
        jphs(image, &encr_data);
        break;
    default:
        fail("wrongmode");
        return nullptr;
    }


    // Saving
    if(success) {
        emit saveImage(image);
        return image;
    }
    else
        return nullptr;
}
QByteArray ModelPC::decrypt(QImage * image, QString *_error)
{
    // Error management
    *_error = "ok";
    error = _error;
    if(image == nullptr || image->isNull()) {
        fail("nullimage");
        return nullptr;
    }
    // Image opening
    int w = image->width();
    int h = image->height();

    // Getting corner pixels
    QColor colUL = image->pixelColor(0, 0).toRgb();
    QColor colUR = image->pixelColor(w - 1, 0).toRgb();
    QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();

    // Getting verification code
    int verifCode = (((colUR.green() % 2) << 5) + colUR.blue() % 32) << 2;
    verifCode += colDR.blue() % 4;
    if(verifCode != 166){
        fail("veriffail");
        return nullptr;
    }
    // Getting number of bytes
    long long int countBytes = (colUL.blue() % 32 + ((colUL.green() % 32) << 5) + ((colUL.red() % 32) << 10
)) << 9;
    countBytes += ((colUR.red() % 32) << 4) + (colUR.green() >> 1) % 16;

    bitsUsed = (colDR.blue() >> 2) % 8 + 1;
    curMode = colDR.green() % 32;

    // Start of the circuit
    QByteArray data;
    circuit(image, &data, countBytes);

    // Check if circuit was successful
    if(!success)
        return nullptr;
    if(data.isEmpty())
    {
        fail("noreaddata");
        return nullptr;

    }
    // Version check
    long long int _ver = mod(data.at(0) * pow(2, 16));
    _ver += mod(data.at(1) * pow(2, 8));
    _ver += mod(data.at(2));
    data.remove(0, 3);
    if(_ver > version) {
        fail("Picture's app version is newer than yours. Image version is "
             + generateVersionString(_ver) + ", yours is "
             + generateVersionString(version) + ".");
        return nullptr;
    }
    else if(_ver < version) {
        fail("Picture's app version is older than yours. Image version is "
             + generateVersionString(_ver) + ", yours is "
             + generateVersionString(version) + ".");
        return nullptr;
    }

    // Obtain the key
    int key_size = mod(data.at(0));
    QByteArray key = data.mid(1, key_size);
    data.remove(0, key_size + 1);

    // Unzip
    QByteArray unzipped_data = unzip(data, key);
    emit saveData(unzipped_data);
    return unzipped_data;
}
void ModelPC::fail(QString message)
{
    *error = message;
    alert(message, true);
    success = false;
    emit setProgress(101);
}
void ModelPC::jphs(QImage *image, QByteArray *data)
{
    // Under Development
    return;

    // Dead code

    success = true;
    bool isEncrypt = !data->isEmpty();
    QString targetEXE = defaultJPHSDir + (isEncrypt ? "/jphide.exe" : "/jpseek.exe");
    if(!fileExists(targetEXE))
    {
        fail("nojphs");
        return;
    }

    QString randomFileName = defaultJPHSDir + "/";
    qsrand(randSeed());
    for(int i = 0; i < 10; i++)
        randomFileName.append(97 + qrand() % 25);
    image->save(randomFileName + ".jpg");
    if(isEncrypt) {
        QFile file(randomFileName + ".pc");
        if(!file.open(QFile::WriteOnly)) {
            fail("savefilefail");
            return;
        }
        file.write(*data);
        file.close();

        QStringList args;
        args << (randomFileName + ".jpg") << (randomFileName + "_out.jpg") << (randomFileName + ".pc");
        QProcess prog(this);
        prog.start(targetEXE, args);
        prog.waitForStarted();
        prog.write("test\n");
        prog.waitForBytesWritten();
        prog.write("test\n");
        prog.waitForBytesWritten();
        prog.waitForReadyRead();
        QByteArray bytes = prog.readAll();
        prog.waitForFinished();
        //QByteArray readData = prog.readAll();
        prog.close();
        // Cleaning - Deleting temp files

    }
    else {

    }

}

void ModelPC::circuit(QImage *image, QByteArray *data, long long countBytes)
{

    // Some flags and creation of the ProgressDialog
    success = true;
    emit setProgress(-1);
    bool isEncrypt = !data->isEmpty();

    // Image setup
    int w = image->width();
    int h = image->height();

    // Visited pixels array
    QVector <QPoint> were;
    were.push_back(QPoint(0, 0));
    were.push_back(QPoint(0, h - 1));
    were.push_back(QPoint(w - 1, 0));
    were.push_back(QPoint(w - 1, h - 1));

    long long int offset = 0;

    // Pre-start Cleaning
    circuitData = data;
    circuitImage = image;
    circuitCountBytes = countBytes;
    cur = 0;
    bitsBuffer.clear();

    // Writing Top-Left to Bottom-Left
    for(int i = 1; i < h - 1 && mustGoOn(isEncrypt); i++) {
        QPoint pos(0, i);
        processPixel(pos, &were, isEncrypt);
    }
    // Writing Bottom-Right to Top-Right
    if(mustGoOn(isEncrypt))
    {
        for(int i = h - 2; i >= 1 && mustGoOn(isEncrypt); i--){
            QPoint pos(w - 1, i);
            processPixel(pos, &were, isEncrypt);
        }
    }
    // Main cycle
    // Strong is considered as actual corner pixel and weak as pixel near it like (1, 0) or (0, 1)
    while(mustGoOn(isEncrypt))
    {
        // Strong Top-Right to Strong Bottom-Right
        for(int i = offset; i < h - offset && mustGoOn(isEncrypt); i++){
            QPoint pos(w - offset - 2, i);
            processPixel(pos, &were, isEncrypt);
        }
        // Strong Top-Left to Weak Top-Right
        for(int i = offset + 1; i < w - offset - 2 && mustGoOn(isEncrypt); i++){
            QPoint pos(i, offset);
            processPixel(pos, &were, isEncrypt);
        }
        // Weak Bottom-Right to Weak Bottom-Left
        for(int i = w - 3 - offset; i >= offset + 2 && mustGoOn(isEncrypt); i--){
            QPoint pos(i, h - offset - 1);
            processPixel(pos, &were, isEncrypt);
        }
        // Weak Top-Left to Strong Bottom-Left
        for(int i = offset + 1; i < h - offset && mustGoOn(isEncrypt); i++){
            QPoint pos(offset + 1, i);
            processPixel(pos, &were, isEncrypt);
        }
        offset++;
    }
    // Extra writing
    if(!success)
        return;
    if(isEncrypt)
    {
        // Getting past colors
        QColor colUL = image->pixelColor(0, 0).toRgb();
        QColor colUR = image->pixelColor(w - 1, 0).toRgb();
        QColor colDL = image->pixelColor(0, h - 1).toRgb();
        QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();
        int red = 0;
        int green = 0;
        int blue = 0;

        // Writing Upper Left
        red = (colUL.red() & 224) + (countBytes >> 19);
        green = (colUL.green() & 224) + (countBytes >> 14) % 32;
        blue = (colUL.blue() & 224) + (countBytes >> 9) % 32;
        image->setPixelColor(0, 0, QColor(red, green, blue));

        // Writing Upper Right
        red = (colUR.red() & 224) + (countBytes >> 4) % 32;
        green = (colUR.green() & 224) + ((countBytes % 16) << 1) + 1;
        blue = (colUR.blue() & 224) + 9;
        image->setPixelColor(w - 1, 0, QColor(red, green, blue));

        // Getting extra bytes if left
        while(cur < countBytes)
            push(mod(circuitData->at(cur++)), 8);
        if(bitsBuffer.size() > 20) {
            fail("bitsBufferFail");
            return;
        }
        // Getting extra data as long.
        long extraData = pop(-2);

        // Writing Down Left
        red = (colDL.red() & 224) + (extraData >> 15);
        green = (colDL.green() & 224) + (extraData >> 10) % 32;
        blue = (colDL.blue() & 224) + (extraData >> 5) % 32;
        image->setPixelColor(0, h - 1, QColor(red, green, blue));

        // Writing Down Right
        red = (colDR.red() & 224) + extraData % 32;
        green = (colDR.green() & 224);
        blue = (colDR.blue() & 224) + ((bitsUsed - 1) << 2) + 2;
        image->setPixelColor(w - 1, h - 1, QColor(red, green, blue));
    }
    else
    {
        // Read the past pixels
        QColor colDL = image->pixelColor(0, h - 1).toRgb();
        QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();

        // Read extra data
        long extraData = ((colDL.red() % 32) << 15) + ((colDL.green() % 32) << 10);
        extraData += ((colDL.blue() % 32) << 5) + colDR.red() % 32;

        // Add extra data to the bitsBuffer
        push(extraData, (countBytes - cur) * 8 - bitsBuffer.size());

        // Move bits from bitsBuffer to the QByteArray
        while(!bitsBuffer.isEmpty())
            data->append(pop(8));
    }
    emit setProgress(101);
}

void ModelPC::processPixel(QPoint pos, QVector<QPoint> *were, bool isEncrypt)
{
    if(!success)
        return;
    // Check if point was already visited
    if(were->contains(pos)){
        fail("Point (" + QString::number(pos.x()) + "," + QString::number(pos.y()) + ") was visited twice! Error code 2");
        return;
    }
    else
        were->push_back(pos);
    if(isEncrypt)
    {
        // Make sure that there are enough bits in bitsBuffer to write
        while(bitsBuffer.size() < 3 * bitsUsed)
            push(mod(circuitData->at(cur++)), 8);
        // Read past contains
        QColor pixelColor = circuitImage->pixelColor(pos);
        int red = pixelColor.red();
        int green = pixelColor.green();
        int blue = pixelColor.blue();

        // Write new data in last bitsUsed pixels
        red += pop() - red % (int) pow(2, bitsUsed);
        green += pop() - green % (int) pow(2, bitsUsed);
        blue += pop() - blue % (int) pow(2, bitsUsed);

        circuitImage->setPixelColor(pos, QColor(red, green, blue));
    }
    else
    {
        QColor read_color = circuitImage->pixelColor(pos).toRgb();
        // Reading the pixel
        int red = read_color.red();
        int green = read_color.green();
        int blue = read_color.blue();

        // Reading the last bitsUsed pixels
        red %= (int) pow(2, bitsUsed);
        green %= (int) pow(2, bitsUsed);
        blue %= (int) pow(2, bitsUsed);

        // Getting the data in the bitsBuffer.
        push(red);
        push(green);
        push(blue);

        // Getting data to QByteArray
        while(bitsBuffer.size() >= 8) {
            circuitData->append(pop(8));
            cur++;
        }
    }
    emit setProgress(100 * cur / circuitCountBytes);
```

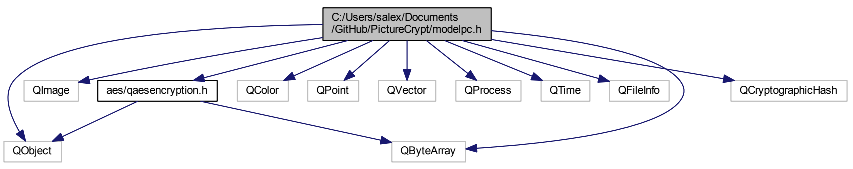
```
}

long ModelPC::pop(int bits)
{
    // Hard to say
    long res = 0;
    int poppedBits = bits == -1 ? bitsUsed : bits;
    if(bits == -2)
        poppedBits = bitsBuffer.size();
    for(int i = 0; i < poppedBits; i++)
        res += bitsBuffer[i] * pow(2, poppedBits - i - 1);
    bitsBuffer.remove(0, poppedBits);
    return res;
}

void ModelPC::push(int data, int bits)
{
    // That's easier, but also hard
    int buf_size = bitsBuffer.size();
    int extraSize = bits == -1 ? bitsUsed : bits;
    bitsBuffer.resize(buf_size + extraSize);
    for(int i = bitsBuffer.size() - 1; i >= buf_size; i--, data >>= 1)
        bitsBuffer[i] = data % 2;
}

bool ModelPC::mustGoOn(bool isEncrypt)
{
    return success && (isEncrypt ? (circuitCountBytes - cur) * 8 + bitsBuffer.size() >= bitsUsed * 3 :
                                   circuitData->size() * 8 + bitsBuffer.size() <
                                   circuitCountBytes * 8 - (circuitCountBytes * 8)% (bitsUsed * 3));
}
QByteArray ModelPC::unzip(QByteArray data, QByteArray key)
{
    // Decryption
    QByteArray hashKey = QCryptographicHash::hash(key, QCryptographicHash::Sha256);
    QAESEncryption encryption(QAESEncryption::AES_256, QAESEncryption::ECB);
    QByteArray new_data = encryption.decode(data, hashKey);
    // Decompressing
    return qUncompress(new_data);
}
QByteArray ModelPC::zip(QByteArray data, QByteArray key)
{
    // Zip
    QByteArray c_data = qCompress(data, 9);
    // Encryption
    QByteArray hashKey = QCryptographicHash::hash(key, QCryptographicHash::Sha256);
    return QAESEncryption::Crypt(QAESEncryption::AES_256, QAESEncryption::ECB, c_data, hashKey);
}

bool ModelPC::fileExists(QString path)
{
    QFileInfo check_file(path);
    return check_file.exists() && check_file.isFile();
}

QByteArray ModelPC::bytes(long long n)
{
    return QByteArray::fromHex(QByteArray::number(n, 16));
}
unsigned int ModelPC::mod(int input)
{
    if(input < 0)
        return (unsigned int) (256 + input);
    else
        return (unsigned int) input;
}
void ModelPC::alert(QString message, bool isWarning)
{
    emit alertView(message, isWarning);
}
QColor ModelPC::RGBbytes(long long byte)
{
    int blue = byte % 256;
    int green = (byte / 256) % 256;
    int red = byte / pow(2, 16);
    return QColor(red, green, blue);
}

QString ModelPC::generateVersionString(long ver)
{
    return QString::number((int)( ver / pow(2, 16))) + "." + QString::number(((int) (ver / 256)) % 256) + "." + QString::number(ver % 256);
}
```

```

uint ModelPC::randSeed()
{
    QTime time = QTime::currentTime();
    uint randSeed = time.msecsSinceStartOfDay() % 65536 + time.minute() * 21 + time.second() * 2;
    return randSeed;
}
```

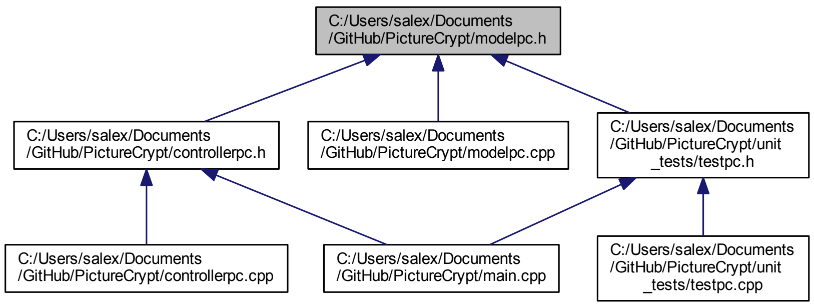
10.27 C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.h File Reference

```
#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:


C:/Users/salex/Documents
/GitHub/PictureCrypt/modelpc.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/controllerpc.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/modelpc.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/unit
_tests/testpc.h
C:/Users/salex/Documents
/GitHub/PictureCrypt/controllerpc.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/main.cpp
C:/Users/salex/Documents
/GitHub/PictureCrypt/unit
_tests/testpc.cpp

Classes

- class ModelPC

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

10.27.1 Detailed Description

Header of ModelPC class

See also

ControllerPC, ModelPC, ViewPC

Definition in file modelpc.h.

10.28 modelpc.h

```
#ifndef MODELPC_H
#define MODELPC_H

#include <QObject>
#include <QImage>
#include <QByteArray>
#include <QColor>
#include <QPoint>
#include <QVector>
#include <QProcess>
#include <QTime>
#include <QFileInfo>

#include <aes/qaesencryption.h>
#include <QCryptographicHash>

class ModelPC : public QObject
{
    Q_OBJECT
public:
    ModelPC();

signals:
    alertView(QString messageCode, bool isWarning);
    saveData(QByteArray data);
    saveImage(QImage *image);
    setProgress(int val);

public slots:
    QImage *start(QByteArray data, QImage *image, int mode = 0, QString key = "", int _bitsUsed = 8, QString *_error = nullptr);
    QImage *encrypt(QByteArray encr_data, QImage * image, int mode = 0, QString *_error = nullptr);
    QByteArray decrypt(QImage * image, QString *_error = nullptr);
    void fail(QString message);

public:
    bool success;
    long version;
    QString versionString;
    int curMode;
    int bitsUsed;
    QString defaultJPHSDir;
    QString * error;
    QByteArray unzip(QByteArray data, QByteArray key);
    void alert(QString message, bool isWarning = false);
protected:
    void circuit(QImage * image, QByteArray * data, long long int countBytes);
    void jphs(QImage * image, QByteArray * data);
    void processPixel(QPoint pos, QVector<QPoint> *were, bool isEncrypt);
    QByteArray zip(QByteArray data, QByteArray key);
private:
    bool fileExists(QString path);
    QByteArray bytes(long long n);
    unsigned int mod(int input);
    QByteArray ver_byte;
    QColor RGBbytes(long long byte);
    QString generateVersionString(long ver);
    uint randSeed();

    QByteArray * circuitData;
    QImage * circuitImage;
    long long circuitCountBytes;
    long cur;
    bool mustGoOn(bool isEncrypt);

    QVector <bool> bitsBuffer;
    long pop(int bits = -1);
    void push(int data, int bits = -1);

    void setError(QString word);
};

#endif // MODELPC_H
```

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

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

```
# PictureCrypt
Make your pictures crypted.


## About
Project is made only using QT.
[QAESEncryption](http://github.com/bricke/Qt-AES) 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 <QByteArray>
#include <QImage>

...

ModelPC * model = new ModelPC(ver);
// ver is version of the app, used to check the data structure version
// ver is type long and is calculated as if version is "x.y.z" => ver = x * 65536 + y * 256 + z
// Default parameter is 2^17 (2.0.0)

// Connecting signals

// Essential ones

model->saveData(QByteArray data)
// Used to return the retrieved data

model->saveImage(QImage * image)
// Used to return the modified image

// Extra ones

model->alertView(QString message, bool isWarning)
// Used for messages to be shown to users

model->setProgress(int val)
// Used to show user the progress of embedding
// -1 indicates the creation of some kind of progress dialog
// from 0 to 100 shows the progress
// 101 indicates that progress dialog should be closed

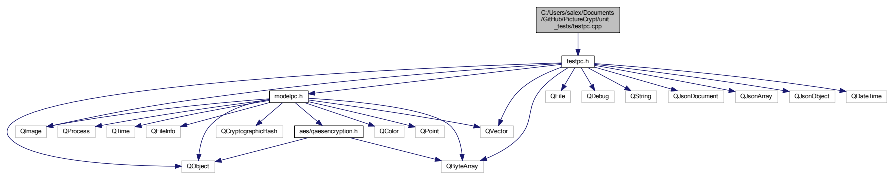
```

## Avaible methods
### Essential ones
#### start
Used for embedding

Parameters:
data   Data to be encrypted
_image Image to be encrypted into.
_bitsUsed  Bits per byte, see also ModelPC::bitsUsed
key    Key, if default (empty), random key of 64 charachters will be generated.
mode   Mode of encryption
```
model->start(QByteArray data, QImage image, int mode = 0, QString key = "", int _bitsUsed = 8);
```

#### decrypt
```

```
Used for de-embedding

Parameters:
image  Image to be decrypted.

```
model->decrypt(QImage * image);
```
### Extra ones
#### encrypt
Used for embedding but with data already packed with stuff like version, file size, aes key, etc.
Used in PictureCrypt project

Parameters:

encr_data  Data to be embbed to an image.
image  Image to be embbed into.
mode   Mode of encryption

```
model->encrypt(QByteArray encr_data, QImage * image, int mode = 0);
```
#### fail
Used for stopping the embedding or de-embedding proccess
Parameters:

message    Message for user
```
model->fail(QString message);
```

## 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](https://waleko.github.io/doc/picturecrypt)


## Dependencies
* qtcore
* [QAESEncryption](https://github.com/bricke/Qt-AES) 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](http://unlicense.org/)
```

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

```
#include "testpc.h"
```

Include dependency graph for testpc.cpp:



10.32 testpc.cpp

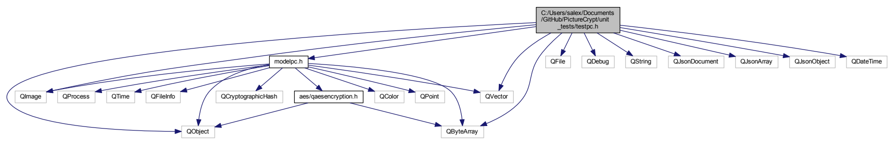
```
#include "testpc.h"
TestPC::TestPC()
```

```
{ }
bool TestPC::test(QByteArray data, QImage rImage, QString expectedOutput, int mode, QString key, int bitsUsed)
{
    // Error outputs
    QString error1, error2;
    // Embedding
    QImage * retImage = model->start(data, &rImage, mode, key, bitsUsed, &error1);
    // De-embedding
    QByteArray output = model->decrypt(retImage, &error2);

    // Success of error outputs
    bool er1 = error1 == expectedOutput;
    bool er2 = error2 == expectedOutput;
    if(expectedOutput == "ok")
        return er1 && er2 && data == output;
    else
        return er1 || er2;
}
int TestPC::startTest()
{
    qDebug() << "Testing started...\n";
    model = new ModelPC();

    // Long text open
    QFile file(":/unit_tests/longtext.txt");
    if(!file.open(QFile::ReadOnly))
        return false;
    text = file.readAll();
    file.close();

    // Big picture open
    image = QImage(":/unit_tests/bigpicture.jpg");
    if(image.isNull())
        return false;

    // JSON tests list open
    QFile json_file(":/unit_tests/tests.json");
    QJsonDocument doc;
    if(!json_file.open(QFile::ReadOnly | QFile::Text))
        return false;
    QByteArray readData = json_file.readAll();
    json_file.close();
    doc = QJsonDocument::fromJson(readData);
    // Testing
    return autoTest(doc);
}
bool TestPC::autoTest(QJsonDocument doc)
{
    // Opening the tests array
    QJsonObject o = doc.object();
    QJsonArray arr = o["tests"].toArray();
    int sum = 0;

    // Info about tests
    QString extraText;
    for(int i = 0; i < arr.size(); i++) {
        // Reading the data
        QJsonObject obj = arr[i].toObject();

        QString t = obj["data"].toString();
        if(t == "/text/")
            t = text;
        QByteArray data = t.toUtf8();

        QString im = obj["image"].toString();
        QImage img(":/unit_tests/" + im);

        QString expect = obj["expectation"].toString();

        int mode = obj["mode"].toInt();

        QString key = obj["key"].toString();

        int bitsUsed = obj["bitsUsed"].toInt();

        // Testing
        bool s = test(data, img, expect, mode, key, bitsUsed);

        sum += s;
        extraText += "\n * Test #" + QString::number(i + 1) + " " + (s ? "completed." : "failed.");
    }
    // Writing log
    QFile file("tests.log");
    bool testsSuc = sum == arr.size();
    if(!file.open(QFile::WriteOnly | QFile::Text))
        return testsSuc;
    QDateTime curTime = QDateTime::currentDateTime();
    QString date = curTime.toString("dd.MM.yyyy HH:mm");
    QString logtext = "############################################\n"
                      "####Log file created at " + date + "####\n"
                      "############################################\n\n"
                      "Status: " + (testsSuc ? "All tests completed" : "Tests failed") + "\n\n"
                      "Tests list:\n";
    logtext += extraText;
    file.write(logtext.toUtf8());
    file.close();
    // Cleaning up
    qDebug() << "Testing completed\n";
    delete model;
    return testsSuc;
}
```

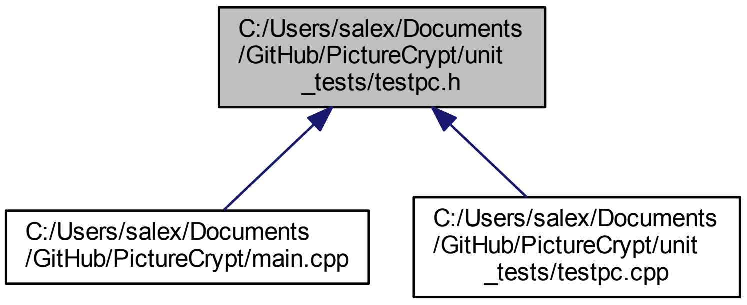
10.33 C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.h File Reference

```
#include <QObject>
#include <modelpc.h>
#include <QFile>
#include <QDebug>
#include <QString>
#include <QImage>
#include <QByteArray>
#include <QVector>
#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

```
#ifndef TESTPC_H
#define TESTPC_H

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

#include <QFile>
#include <QDebug>
#include <QString>
#include <QImage>
#include <QByteArray>
#include <QVector>

#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonObject>

#include <QDateTime>
class TestPC : public QObject
{
    Q_OBJECT
public:
    TestPC();
public slots:
    int startTest();
protected slots:
    bool test(QByteArray data, QImage rImage,
              QString expectedOutput = "ok", int mode = 0,
              QString key = "", int bitsUsed = 8);
private:
    ModelPC * model;
    QByteArray text;
    QImage image;

    bool autoTest(QJsonDocument doc);
};

#endif // TESTPC_H
```

10.35 C:/Users/salex/Documents/GitHub/PictureCrypt/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,'bitsUsed':int(bitsUsed)}
- tests-setup.f
- tests-setup.indent

10.36 tests-setup.py

```python
import json
filename = 'tests.json'

raw = open(filename, 'r')

js = json.load(raw)
print('Existing tests:')
for obj in js['tests']:
    print(obj['data'], obj['image'], obj['expectation'], obj['mode'], obj['key'], obj['bitsUsed'], sep='-')

print('------------')
print('Type new tests')

input_data = input()

arr = []
while len(input_data):
    data, image, expect, mode, key, bitsUsed = map(str, input_data.split('-'))

    obj = {'data':data, 'image':image,'expectation':expect,'mode':int(mode),'key':key,'bitsUsed':int(
bitsUsed)}
    arr.append(obj)
    input_data = input()

js['tests'] += arr
with open(filename, 'w') as f:
    json.dump(js, f, indent=4)
```

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

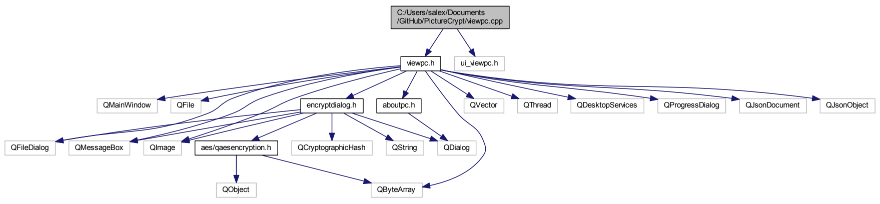
10.38 tests.json

```json
{
    "tests": [
        {
            "data": "/text/",
            "image": "bigpicture.jpg",
            "expectation": "ok",
            "mode": 0,
            "key": "",
            "bitsUsed": 8
        },
        {
            "data": "/text/",
            "image": "bigpicture.jpg",
            "expectation": "ok",
            "mode": 0,
            "key": "",
            "bitsUsed": 7
        },
        {
            "data": "/text/",
            "image": "bigpicture.jpg",
            "expectation": "ok",
            "mode": 0,
            "key": "",
            "bitsUsed": 1
        },
        {
            "data": "/text/",
            "image": "tinypicture.png",
            "expectation": "muchdata",
            "mode": 0,
            "key": "",
            "bitsUsed": 8
        },
        {
            "data": "",
            "image": "bigpicture.jpg",
            "expectation": "nodata",
            "mode": 0,
            "key": "",
            "bitsUsed": 8
        },
        {
            "data": "/text/",
            "image": "invalid.jpg",
            "expectation": "nullimage",
            "mode": 0,
            "key": "",
            "bitsUsed": 8
        },
        {
            "data": "/text/",
            "image": "bigpicture.jpg",
            "expectation": "bitsWrong",
            "mode": 0,
            "key": "",
            "bitsUsed": 12
        }
    ]
}
```

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

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

Include dependency graph for viewpc.cpp:



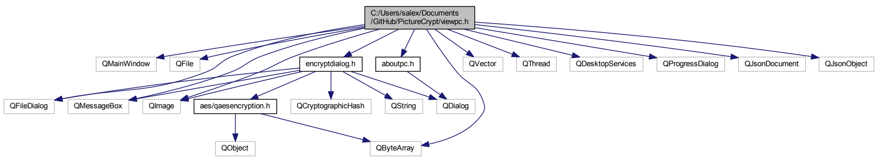
10.40 viewpc.cpp

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

ViewPC::ViewPC(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::ViewPC)
{
    ui->setupUi(this);

    progressDialogClosed = true;

    // Alerts dictionary setup
    // TODO Add relative path
    QFile file(":/config/ErrorsDict.json");
    if(!file.open(QFile::ReadOnly | QFile::Text)) {
        alert("Cannot open config file!");
        return;
    }
    QByteArray readData = file.readAll();
    file.close();

    QJsonParseError error;
    QJsonDocument doc = QJsonDocument::fromJson(readData, &error);
    errorsDict = doc.object();
}

ViewPC::~ViewPC()
{
    delete ui;
}

void ViewPC::on_encryptMode_clicked()
{
    // Encrypt radio button clicked
    setEncryptMode(true);
}

void ViewPC::on_decryptMode_clicked()
{
    // Decrypt radio button clicked
    setEncryptMode(false);
}
void ViewPC::on_fileButton_clicked()
{
    // Opening QFileDialog depending on isEncrypt
    if(isEncrypt)
        inputFileName = QFileDialog::getOpenFileName(this, tr("Select file"), "/untitled.txt", tr("Text
   files (*.txt);;All Files (*)"));
    else
        inputFileName = QFileDialog::getOpenFileName(this, tr("Select file"), "/untitled.png", tr("PNG
   files (*.png);;All Files (*)"));
    // Display the file name
    ui->fileLabel->setText(inputFileName.isEmpty() ? "File not chosen" : inputFileName);
}
void ViewPC::on_startButton_clicked()
{
    if(isEncrypt)
    {
        // Getting the data
        QString text = ui->text->toPlainText();
        QByteArray data;
        if(text.isEmpty()) {
            if(inputFileName.isEmpty()) {
                alert("No input file or text was not given. Cannot continue!", true);
                return;
            }
            // Opening the file
            QFile file(inputFileName);
            if (!file.open(QIODevice::ReadOnly))
            {
                alert("Cannot open file. Cannot continue!", true);
                return;
            }
            // Check the data size
            auto size = file.size();
            if(size > pow(2, 24)) {
                alert("Your file is too big, our systems can handle it, but it requires a lot of time.
   We decline.", true);
                file.close();
                return;
            }
            data = file.readAll();
            file.close();
        }
        else
            data = text.toUtf8();
        // Select image via EncryptDialog
        EncryptDialog * dialog = new EncryptDialog(
data);
        dialog->exec();
        if(!dialog->success)
            return;

        // Get the data
        QByteArray encr_data = dialog->compr_data;

        // Save the key
        QByteArray key_data = dialog->key.toUtf8();

        encr_data = bytes(key_data.size()) + key_data + encr_data;
        // TODO do the mode thing
        emit setBitsUsed(dialog->bitsUsed);
        emit encrypt(encr_data, &dialog->image, 0);
    }
    else
    {
        // Get the filename of the image
        if(!ui->text->toPlainText().isEmpty())
            alert("Obviously, the text browser isn't supported for decryption, use File Dialog instead.");
        if(inputFileName.isEmpty()) {
            alert("File not selected. Cannot continue!", true);
            return;
        }
        QImage * res_image = new QImage(inputFileName);
        emit decrypt(res_image);
    }
}
void ViewPC::alert(QString message, bool isWarning)
{
    // Get message
    if(errorsDict.contains(message))
        message = errorsDict[message].toString();
    // Create message box
    QMessageBox box;
    if(isWarning)
        box.setIcon(QMessageBox::Warning);
    else
        box.setIcon(QMessageBox::Information);
    box.setText(message);
    box.setWindowIcon(QIcon(":/icons/mail.png"));
    box.setWindowTitle("Message");
    box.exec();
}
void ViewPC::saveData(QByteArray Edata)
{
    // Save data using QFileDialog
    QString outputFileName = QFileDialog::getSaveFileName(this, tr("Save File"),
                                "/untitled.txt",
                                tr("Text (*.txt);;All files (*)"));
    QFile writeFile(outputFileName);
    if (!writeFile.open(QIODevice::WriteOnly))
    {
        alert("Cannot access file path. Cannot continue!", true);
        return;
    }
    writeFile.write(Edata);
    writeFile.close();
    alert("Decryption completed!");
}
void ViewPC::saveImage(QImage * image)
{
    // Save image using QFileDialog
    QString outputFileName = QFileDialog::getSaveFileName(this, tr("Save Image"),
                                "/untitled.png",
                                tr("Images (*.png)"));
    if(!image->save(outputFileName)) {
        alert("Cannot save file. Unable to continue!", true);
        return;
    }
    alert("Encryption completed!");
}
void ViewPC::setProgress(int val)
{
    if(val < 0) {
        // Create dialog
        dialog = new QProgressDialog("Cryption in progress.", "Cancel", 0, 100);
        connect(dialog, SIGNAL(canceled()), this, SLOT(abortCircuit()));
        progressDialogClosed = false;
        dialog->setWindowTitle("Processing");
        dialog->setWindowIcon(QIcon(":/icons/loading.png"));
        dialog->show();
    }
    else if(val >= 100 && !progressDialogClosed) {
        // Close dialog
        dialog->setValue(100);
        QThread::msleep(25);
        dialog->close();
        dialog->reset();
        progressDialogClosed = true;
    }
    // Update the progress
    else if(!progressDialogClosed)
        dialog->setValue(val);
}
void ViewPC::abortCircuit()
{
    // Set the flag
    progressDialogClosed = true;
    // Close the dialog
    dialog->close();
    dialog->reset();
    emit abortModel();
}
void ViewPC::setEncryptMode(bool encr)
{
    ui->text->setEnabled(encr);
    isEncrypt = encr;
}
void ViewPC::setVersion(QString version)
{
    // Version setup
    versionString = version;
}

QByteArray ViewPC::bytes(long long n)
{
    return QByteArray::fromHex(QByteArray::number(n, 16));
}
void ViewPC::on_actionAbout_triggered()
{
    AboutPC about;
    about.setVersion(versionString);
    about.exec();
}

void ViewPC::on_actionHelp_triggered()
{
    QUrl docLink("http://doc.alex.unaux.com/picturecrypt");
    QDesktopServices::openUrl(docLink);
}

void ViewPC::on_actionJPHS_path_triggered()
{
    QString dir = QFileDialog::getExistingDirectory(this, tr("Open JPHS folder"),
                                                    "/home",
                                                    QFileDialog::ShowDirsOnly
                                                    | QFileDialog::DontResolveSymlinks);
    emit setJPHSDir(dir);
}
```

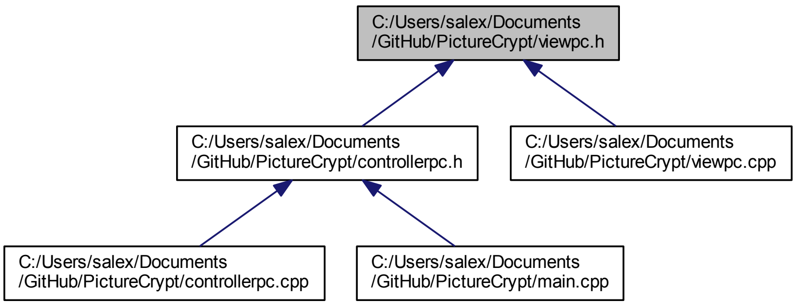
10.41 C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.h File Reference

```
#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:


C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.h
C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.h
C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.cpp
C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.cpp
C:/Users/salex/Documents/GitHub/PictureCrypt/main.cpp

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

```
#ifndef VIEWPC_H
#define VIEWPC_H

#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>

namespace Ui {
class ViewPC;
}
class ViewPC : public QMainWindow
{
    Q_OBJECT

public:
    explicit ViewPC(QWidget *parent = nullptr);
    ~ViewPC();
private slots:
    void on_encryptMode_clicked();

    void on_decryptMode_clicked();

    void on_actionJPHS_path_triggered();

protected slots:
    void on_fileButton_clicked();

    void on_startButton_clicked();

    void on_actionAbout_triggered();

    void on_actionHelp_triggered();
public slots:
    void alert(QString message, bool isWarning = false);
    void saveData(QByteArray Edata);
    void saveImage(QImage *image);
    void setProgress(int val);
    void abortCircuit();
    void setEncryptMode(bool encr);
    void setVersion(QString version);
signals:
    encrypt(QByteArray data, QImage * image, int mode);
    decrypt(QImage * _image);
    abortModel();
    setBitsUsed(int bitsUsed);
    setJPHSDir(QString dir);
public:
    QProgressDialog * dialog;
    bool progressDialogClosed;
    QJsonObject errorsDict;
private:
    Ui::ViewPC *ui;
    bool isEncrypt;
    QString inputFileName;
    QByteArray bytes(long long n);
    QString versionString;
};

#endif // VIEWPC_H
```

Index