

PictureCrypt

1.3.0

Generated by Doxygen 1.8.14

Contents

1	PictureCrypt	1
1.1	The idea of the project	1
1.2	Realisation	1
1.3	How can someone use it?	1
1.4	Structure of the project.	2
1.5	External use	2
1.6	JPHS use	3
1.7	License	4
1.8	Contact us	4
2	Contributor Covenant Code of Conduct	5
3	PictureCrypt	7
4	Namespace Index	11
4.1	Namespace List	11
5	Hierarchical Index	13
5.1	Class Hierarchy	13
6	Class Index	15
6.1	Class List	15
7	File Index	17
7.1	File List	17

8 Namespace Documentation	19
8.1 ErrorsDictSetup Namespace Reference	19
8.1.1 Variable Documentation	19
8.1.1.1 data	19
8.1.1.2 f	19
8.1.1.3 filename	20
8.1.1.4 indent	20
8.1.1.5 input_data	20
8.1.1.6 key	20
8.1.1.7 raw	20
8.1.1.8 value	20
8.2 tests-setup Namespace Reference	21
8.2.1 Variable Documentation	21
8.2.1.1 arr	21
8.2.1.2 bitsUsed	21
8.2.1.3 data	21
8.2.1.4 expect	22
8.2.1.5 f	22
8.2.1.6 filename	22
8.2.1.7 image	22
8.2.1.8 indent	22
8.2.1.9 input_data	22
8.2.1.10 js	23
8.2.1.11 key	23
8.2.1.12 mode	23
8.2.1.13 obj	23
8.2.1.14 raw	23
8.2.1.15 sep	23
8.3 Ui Namespace Reference	23

9	Class Documentation	25
9.1	AboutPC Class Reference	25
9.1.1	Detailed Description	26
9.1.2	Constructor & Destructor Documentation	26
9.1.2.1	AboutPC()	26
9.1.2.2	~AboutPC()	26
9.1.3	Member Function Documentation	26
9.1.3.1	setVersion()	26
9.2	ControllerPC Class Reference	27
9.2.1	Detailed Description	28
9.2.2	Constructor & Destructor Documentation	28
9.2.2.1	ControllerPC()	29
9.2.3	Member Function Documentation	29
9.2.3.1	abortCircuit	29
9.2.3.2	setBitsUsed	29
9.2.3.3	setJPHSDir	30
9.2.4	Member Data Documentation	30
9.2.4.1	version	31
9.2.4.2	versionString	31
9.3	EncryptDialog Class Reference	31
9.3.1	Detailed Description	32
9.3.2	Constructor & Destructor Documentation	33
9.3.2.1	EncryptDialog()	33
9.3.2.2	~EncryptDialog()	33
9.3.3	Member Function Documentation	33
9.3.3.1	on_buttonBox_accepted	33
9.3.3.2	on_buttonBox_rejected	34
9.3.3.3	on_fileButton_clicked	34
9.3.3.4	on_horizontalSlider_valueChanged	34
9.3.3.5	zip()	34

9.3.4	Member Data Documentation	35
9.3.4.1	bitsUsed	35
9.3.4.2	compr_data	36
9.3.4.3	data	36
9.3.4.4	goodPercentage	36
9.3.4.5	image	36
9.3.4.6	inputFileName	36
9.3.4.7	key	37
9.3.4.8	size	37
9.3.4.9	success	37
9.3.4.10	val	37
9.4	ModelPC Class Reference	38
9.4.1	Detailed Description	40
9.4.2	Constructor & Destructor Documentation	40
9.4.2.1	ModelPC()	40
9.4.3	Member Function Documentation	40
9.4.3.1	alert()	40
9.4.3.2	alertView	41
9.4.3.3	circuit()	41
9.4.3.4	decrypt	42
9.4.3.5	encrypt	43
9.4.3.6	fail	44
9.4.3.7	jphs()	45
9.4.3.8	processPixel()	46
9.4.3.9	saveData	47
9.4.3.10	savelImage	47
9.4.3.11	setProgress	47
9.4.3.12	start	48
9.4.3.13	unzip()	49
9.4.3.14	zip()	50

9.4.4	Member Data Documentation	51
9.4.4.1	bitsUsed	51
9.4.4.2	curMode	51
9.4.4.3	defaultJPHSDir	52
9.4.4.4	error	52
9.4.4.5	success	52
9.4.4.6	version	52
9.4.4.7	versionString	52
9.5	QAESEncryption Class Reference	53
9.5.1	Detailed Description	54
9.5.2	Member Enumeration Documentation	54
9.5.2.1	Aes	54
9.5.2.2	Mode	55
9.5.2.3	Padding	55
9.5.3	Constructor & Destructor Documentation	55
9.5.3.1	QAESEncryption()	56
9.5.4	Member Function Documentation	56
9.5.4.1	Crypt()	56
9.5.4.2	decode()	57
9.5.4.3	Decrypt()	58
9.5.4.4	encode()	59
9.5.4.5	ExpandKey()	60
9.5.4.6	expandKey()	61
9.5.4.7	RemovePadding()	61
9.5.4.8	removePadding()	62
9.6	TestPC Class Reference	62
9.6.1	Detailed Description	64
9.6.2	Constructor & Destructor Documentation	64
9.6.2.1	TestPC()	64
9.6.3	Member Function Documentation	64

9.6.3.1	startTest	64
9.6.3.2	test	65
9.7	ViewPC Class Reference	66
9.7.1	Detailed Description	67
9.7.2	Constructor & Destructor Documentation	68
9.7.2.1	ViewPC()	68
9.7.2.2	~ViewPC()	68
9.7.3	Member Function Documentation	68
9.7.3.1	abortCircuit	68
9.7.3.2	abortModel	69
9.7.3.3	alert	69
9.7.3.4	decrypt	70
9.7.3.5	encrypt	71
9.7.3.6	on_actionAbout_triggered	71
9.7.3.7	on_actionHelp_triggered	72
9.7.3.8	on_fileButton_clicked	72
9.7.3.9	on_startButton_clicked	72
9.7.4	Encrypting	72
9.7.5	Decrypting	72
9.7.5.1	saveData	73
9.7.5.2	savelImage	73
9.7.5.3	setBitsUsed	74
9.7.5.4	setEncryptMode	74
9.7.5.5	setJPHSDir	75
9.7.5.6	setProgress	75
9.7.5.7	setVersion	76
9.7.6	Member Data Documentation	76
9.7.6.1	dialog	76
9.7.6.2	errorsDict	76
9.7.6.3	progressDialogClosed	77

10 File Documentation	79
10.1 C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.cpp File Reference	79
10.2 aboutpc.cpp	79
10.3 C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.h File Reference	80
10.4 aboutpc.h	81
10.5 C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.cpp File Reference	81
10.5.1 Function Documentation	81
10.5.1.1 multiply()	82
10.5.1.2 xTime()	82
10.6 qaesencryption.cpp	82
10.7 C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.h File Reference	89
10.8 qaesencryption.h	90
10.9 C:/Users/salex/Documents/GitHub/PictureCrypt/CODE_OF_CONDUCT.md File Reference	92
10.10 C:/Users/salex/Documents/GitHub/PictureCrypt/CODE_OF_CONDUCT.md	92
10.11 C:/Users/salex/Documents/GitHub/PictureCrypt/config/ErrorsDict.json File Reference	92
10.12 ErrorsDict.json	93
10.13 C:/Users/salex/Documents/GitHub/PictureCrypt/config/ErrorsDictSetup.py File Reference	93
10.14 ErrorsDictSetup.py	93
10.15 C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.cpp File Reference	94
10.16 controllerpc.cpp	94
10.17 C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.h File Reference	94
10.17.1 Detailed Description	95
10.18 controllerpc.h	95
10.19 C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.cpp File Reference	96
10.20 encryptdialog.cpp	96
10.21 C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.h File Reference	97
10.22 encryptdialog.h	99
10.23 C:/Users/salex/Documents/GitHub/PictureCrypt/main.cpp File Reference	99
10.23.1 Function Documentation	100
10.23.1.1 main()	100

10.24main.cpp	100
10.25C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.cpp File Reference	100
10.26modelpc.cpp	101
10.27C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.h File Reference	107
10.27.1 Detailed Description	108
10.28modelpc.h	108
10.29C:/Users/salex/Documents/GitHub/PictureCrypt/README.md File Reference	109
10.30C:/Users/salex/Documents/GitHub/PictureCrypt/README.md	109
10.31C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.cpp File Reference	110
10.32testpc.cpp	110
10.33C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.h File Reference	112
10.34testpc.h	113
10.35C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/tests-setup.py File Reference	113
10.36tests-setup.py	114
10.37C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/tests.json File Reference	114
10.38tests.json	114
10.39C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.cpp File Reference	115
10.40viewpc.cpp	115
10.41C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.h File Reference	118
10.41.1 Detailed Description	119
10.42viewpc.h	119
Index	121

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 steganography. 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 communicate. 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 anything. Great example...

Note

`TestPC` class was introduced recently, its use is advised.

```
#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 explanation
                                   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`

Available methods see here: <https://waleko.github.io/PictureCrypt/#external-use> or here [ModelPC](#)

1.6 JPBS use

The newer versions of the app have jpbs support, but they don't have jpbs 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 JPBS, 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 jpbs is on you, that is why we use made only optionally. That means that to use jpbs with our app you will have to download the jpbs yourself and specify the jpbs directory. However we provide link to the site where you can download the supported version of the jpbs: <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](http://contributor-covenant.org/version/1/4), 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 business 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
```

Available 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 characters 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 embbded to an image. image Image to be embbded 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 available [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:

ErrorsDictSetup	19
tests-setup	21
Ui	23

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

QDialog	
AboutPC	25
EncryptDialog	31
QMainWindow	
ViewPC	66
QObject	
ControllerPC	27
ModelPC	38
QAESEncryption	53
TestPC	62

Chapter 6

Class Index

6.1 Class List

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

AboutPC		
	The About Page dialog	25
ControllerPC		
	The ControllerPC class Controller class, which controls View and Model layers	27
EncryptDialog		
	Class to get the image and key to store secret info	31
ModelPC		
	The ModelPC class Model Layer of the app. Controlled by ControllerPC	38
QAESEncryption		
	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	53
TestPC		
	AutoTest for ModelPC Currently used in main.cpp	62
ViewPC		
	View layer of the app. Controls EncryptDialog and ProgressDialog	66

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.cpp	79
C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.h	80
C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.cpp	94
C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.h	94
C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.cpp	96
C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.h	97
C:/Users/salex/Documents/GitHub/PictureCrypt/main.cpp	99
C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.cpp	100
C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.h	107
C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.cpp	115
C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.h	118
C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.cpp	81
C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.h	89
C:/Users/salex/Documents/GitHub/PictureCrypt/config/ErrorsDict.json	92
C:/Users/salex/Documents/GitHub/PictureCrypt/config/ErrorsDictSetup.py	93
C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.cpp	110
C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.h	112
C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/tests-setup.py	113
C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/tests.json	114

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

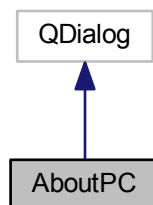
Class Documentation

9.1 AboutPC Class Reference

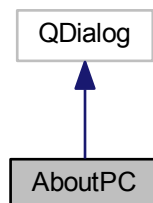
The [AboutPC](#) class The About Page dialog.

```
#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 )
```

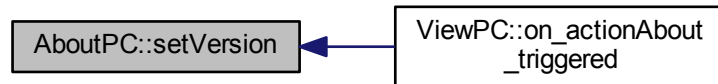
[AboutPC::setVersion](#) Function to set the version display.

Parameters

<i>version</i>	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:

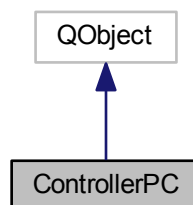
- `C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.h`
- `C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.cpp`

9.2 ControllerPC Class Reference

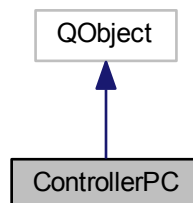
The [ControllerPC](#) class Controller class, which controls View and Model layers.

```
#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 **ModelIPC**, creates Model Class (**ModelIPC**) and View Class (**ViewPC**). All signals and slots are connected here.

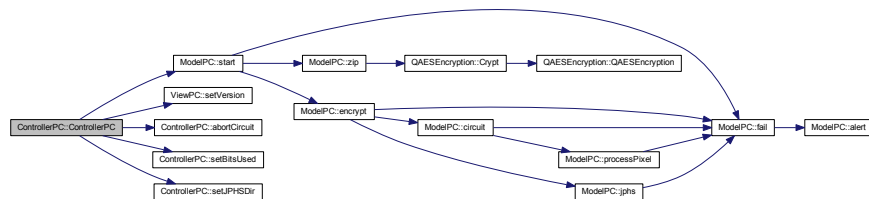
Controller class

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 **ModelIPC** 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 **ModelIPC::bitsUsed**.

Parameters

<i>bitsUsed</i>	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

<i>dir</i>	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:

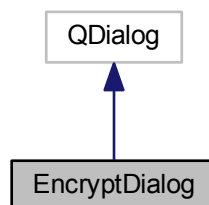
- C:/Users/salex/Documents/GitHub/PictureCrypt/[controllerpc.h](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/[controllerpc.cpp](#)

9.3 EncryptDialog Class Reference

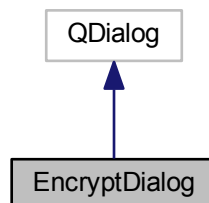
The [EncryptDialog](#) class Class to get the image and key to store secret info.

```
#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 *EncryptDialog::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

<i>_data</i>	Input data.
<i>parent</i>	Parent (not in use)

Definition at line 9 of file [encryptdialog.cpp](#).

Here is the call graph for this function:



9.3.2.2 ~EncryptDialog()

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

Definition at line 29 of file [encryptdialog.cpp](#).

9.3.3 Member Function Documentation

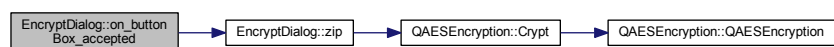
9.3.3.1 on_buttonBox_accepted

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

[EncryptDialog::on_buttonBox_accepted](#) Slot to start the encryption. Successful closing of the app.

Definition at line 85 of file [encryptdialog.cpp](#).

Here is the call graph for this function:



9.3.3.2 on_buttonBox_rejected

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

[EncryptDialog::on_buttonBox_rejected](#) Slot to reject the encryption.

Definition at line 100 of file [encryptdialog.cpp](#).

9.3.3.3 on_fileButton_clicked

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

[EncryptDialog::on_fileButton_clicked](#) Slot to select the image.

Definition at line 60 of file [encryptdialog.cpp](#).

9.3.3.4 on_horizontalSlider_valueChanged

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

[EncryptDialog::on_horizontalSlider_valueChanged](#) Slot if value of the slider is changed. Key is generated here.

Parameters

<i>value</i>	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

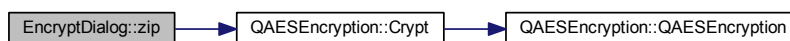
Returns Compressed data.

See also

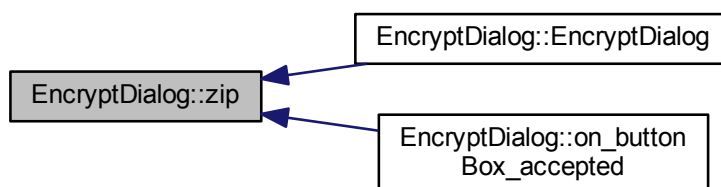
[ModelIPC::unzip](#)

Definition at line 49 of file [encryptdialog.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.4 Member Data Documentation

9.3.4.1 bitsUsed

```
int EncryptDialog::bitsUsed
```

bitsUsed Bits used per byte of pixel.

See also

[ModelIPC::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 EncryptDialog::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:

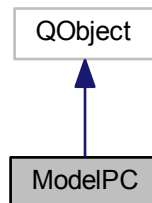
- C:/Users/salex/Documents/GitHub/PictureCrypt/[encryptdialog.h](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/[encryptdialog.cpp](#)

9.4 ModelPC Class Reference

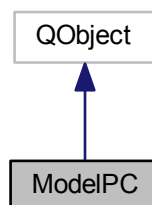
The [ModelPC](#) class Model Layer of the app. Controlled by [ControllerPC](#).

```
#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 crypton.

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

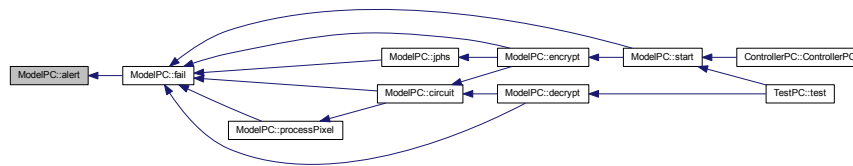
<i>message</i>	Message to be transmitted.
<i>isWarning</i>	Flag if message is critical.

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.

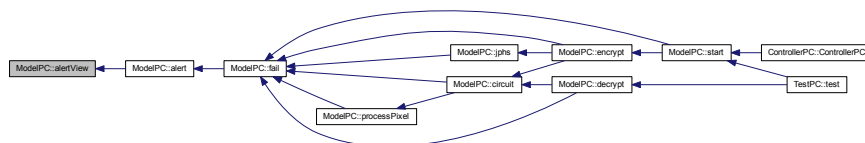
Parameters

<i>messageCode</i>	Message Code to be shown.
<i>isWarning</i>	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

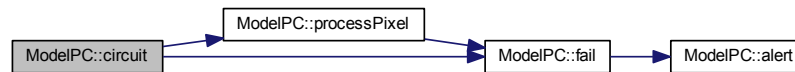
<i>image</i>	Image to be processed.
<i>data</i>	Data to be processed.
<i>countBytes</i>	Number of bytes to be read or written.

See also

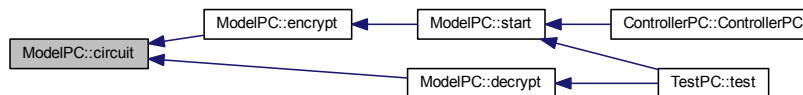
[ModelPC::processPixel](#)

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

<i>image</i>	Image to be decrypted.
--------------	------------------------

Returns

Returns decrypted data

Parameters

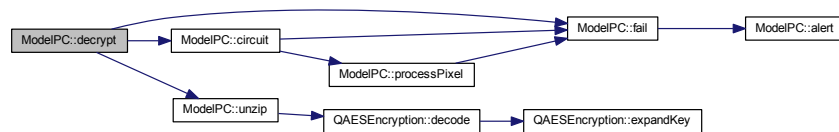
<code>_error</code>	Error output
---------------------	--------------

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

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

Returns

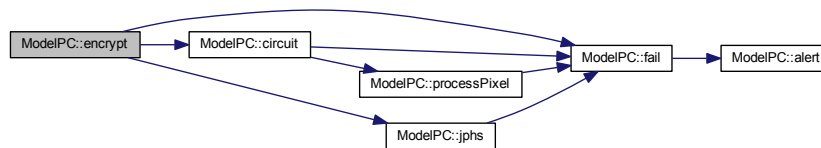
Returns image with embedded data.

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 crypton.

Parameters

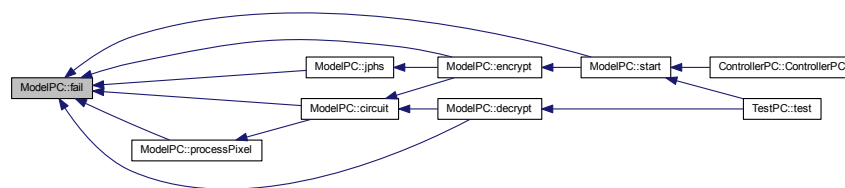
<i>message</i>	Message for user
----------------	------------------

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]
  
```

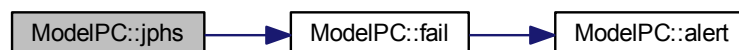
[ModelPC::jphs](#) JPHS function to use jphide and jpseek (currently under development)

Parameters

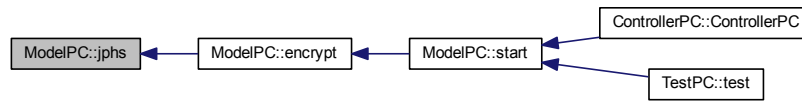
<i>image</i>	Image for embedding
<i>data</i>	Data

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]
  
```

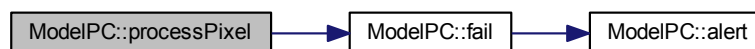
[ModelPC::processPixel](#) Processes every pixel. Reads its contains or writes data.

Parameters

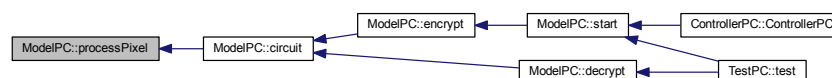
<i>pos</i>	Position of pixel
<i>were</i>	Vector array containing pixels, that were already processed.
<i>isEncrypt</i>	Mode of operation. If true encryption operations will continue, else the decryption ones.

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

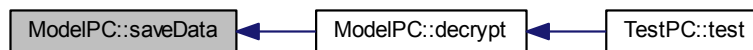
```
ModelPC::saveData (
    QByteArray data ) [signal]
```

saveData Signal to be called to save data from [ModelPC::decrypt](#).

Parameters

<i>data</i>	Data to be saved.
-------------	-------------------

Here is the caller graph for this function:



9.4.3.10 saveImage

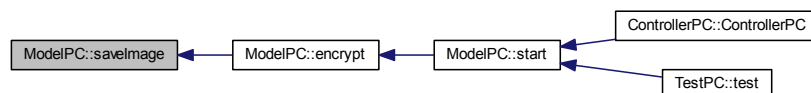
```
ModelPC::saveImage (
    QImage * image ) [signal]
```

saveImage Signal to be called to save image from [ModelPC::encrypt](#).

Parameters

<i>image</i>	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.

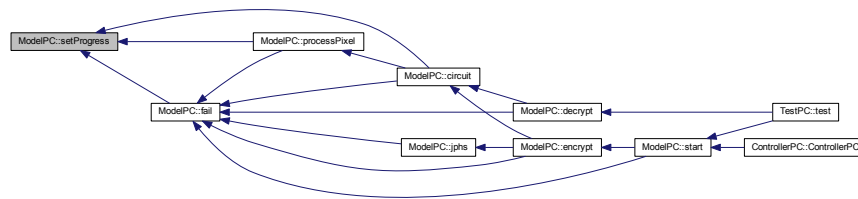
Parameters

<i>val</i>	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

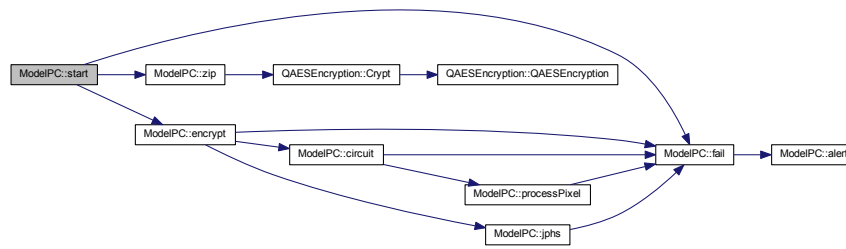
<i>data</i>	Data for embedding
<i>image</i>	Image for embedding
<i>mode</i>	Mode for embedding
<i>key</i>	Key for extra encryption (if empty, key will be auto-generated)
<i>_bitsUsed</i>	Bits per byte (see ModelPC::bitsUsed)
<i>_error</i>	Error output

Returns

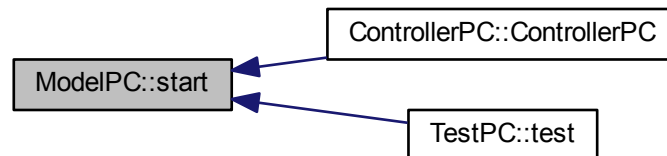
Returns image with embedded data

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

<i>data</i>	Data to be decrypted.
<i>key</i>	Key to decrypt the data.

Returns

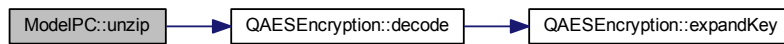
Returns data

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

<i>data</i>	Data to be encrypted
<i>key</i>	Key for encryption

Returns

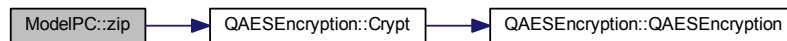
Returns decrypted data

See also

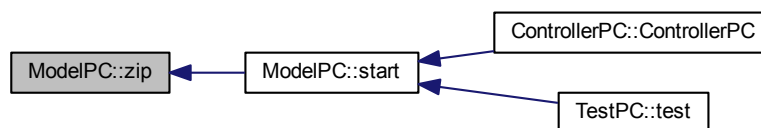
[ModelPC::start](#), [ModelPC::encrypt](#), [ModelPC::unzip](#)

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:

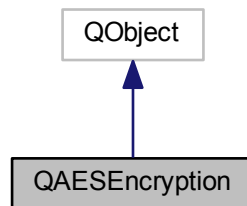
- [C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.h](#)
- [C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.cpp](#)

9.5 QAESEncryption Class Reference

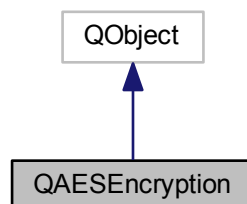
The [QAESEncryption](#) class Small and portable AES encryption class for Qt. Supports all key sizes - 128/192/256 bits - ECB, CBC, CFB and OFB modes. Class made entirely by bricke. Github: <https://github.com/bricke/Qt-AES>.

```
#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 lengths.

- enum [Mode](#) { [ECB](#), [CBC](#), [CFB](#), [OFB](#) }

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

- enum [Padding](#) { [ZERO](#), [PKCS7](#), [ISO](#) }

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

Public Member Functions

- [QAESEncryption](#) ([QAESEncryption::Aes](#) level, [QAESEncryption::Mode](#) mode, [QAESEncryption::Padding](#) padding=[QAESEncryption::ISO](#))
- [QByteArray encode](#) (const [QByteArray](#) &rawText, const [QByteArray](#) &key, const [QByteArray](#) &iv=NULL)
encode Encodes data with AES
- [QByteArray decode](#) (const [QByteArray](#) &rawText, const [QByteArray](#) &key, const [QByteArray](#) &iv=NULL)
decode Decodes data with AES
- [QByteArray removePadding](#) (const [QByteArray](#) &rawText)
RemovePadding Removes padding.
- [QByteArray expandKey](#) (const [QByteArray](#) &key)
ExpandKey Expands the key.

Static Public Member Functions

- static [QByteArray Crypt](#) ([QAESEncryption::Aes](#) level, [QAESEncryption::Mode](#) mode, const [QByteArray](#) &rawText, const [QByteArray](#) &key, const [QByteArray](#) &iv=NULL, [QAESEncryption::Padding](#) padding=[QAESEncryption::ISO](#))
Crypt Static encode function.
- static [QByteArray Decrypt](#) ([QAESEncryption::Aes](#) level, [QAESEncryption::Mode](#) mode, const [QByteArray](#) &rawText, const [QByteArray](#) &key, const [QByteArray](#) &iv=NULL, [QAESEncryption::Padding](#) padding=[QAESEncryption::ISO](#))
Decrypt Static decode function.
- static [QByteArray ExpandKey](#) ([QAESEncryption::Aes](#) level, [QAESEncryption::Mode](#) mode, const [QByteArray](#) &key)
ExpandKey Expands the key.
- static [QByteArray RemovePadding](#) (const [QByteArray](#) &rawText, [QAESEncryption::Padding](#) padding)
RemovePadding Removes padding.

9.5.1 Detailed Description

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

Author

Bricke (Matteo B)

Definition at line 14 of file [qaesencryption.h](#).

9.5.2 Member Enumeration Documentation

9.5.2.1 Aes

enum [QAESEncryption::Aes](#)

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

AES_128 AES_192 AES_256

Enumerator

AES_128	
AES_192	
AES_256	

Definition at line 27 of file [qaesencryption.h](#).

9.5.2.2 Mode

enum [QAESEncryption::Mode](#)

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

Enumerator

ECB	
CBC	
CFB	
OFB	

Definition at line 40 of file [qaesencryption.h](#).

9.5.2.3 Padding

enum [QAESEncryption::Padding](#)

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

ZERO PKCS7 ISO

Enumerator

ZERO	
PKCS7	
ISO	

Definition at line 55 of file [qaesencryption.h](#).

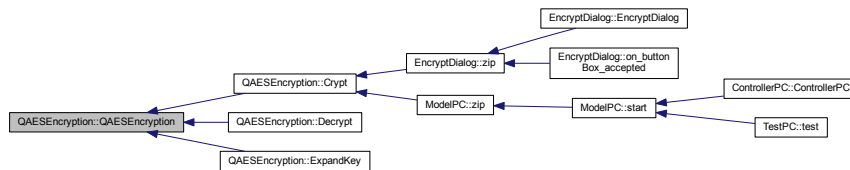
9.5.3 Constructor & Destructor Documentation

9.5.3.1 QAESEncryption()

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

Definition at line 67 of file [qaesencryption.cpp](#).

Here is the caller graph for this function:



9.5.4 Member Function Documentation

9.5.4.1 Crypt()

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

Crypt Static encode function.

Parameters

<i>level</i>	AES level of encryption
<i>mode</i>	AES mode
<i>rawText</i>	Input data
<i>key</i>	Key for encryption
<i>iv</i>	IV vector
<i>padding</i>	Padding

Returns

Returns encrypted data

See also

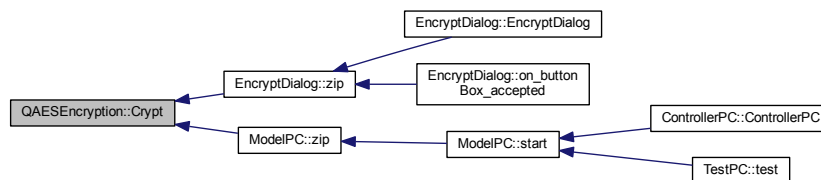
[QAESEncryption::encode](#), [QAESEncryption::Decrypt](#)

Definition at line 6 of file [qaesencryption.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



9.5.4.2 decode()

```

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

decode Decodes data with AES

Note

Basically the non-static method of [QAESEncryption::Decrypt](#)

Parameters

<i>rawText</i>	Input data
<i>key</i>	Key
<i>iv</i>	IV vector

Returns

Returns decoded data

See also

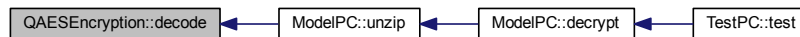
[QAESEncryption::Decrypt](#), [QAESEncryption::encode](#)

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

<i>level</i>	AES level of encryption
<i>mode</i>	AES mode
<i>rawText</i>	Encrypted data
<i>key</i>	Key for encrytion
<i>iv</i>	IV vector
<i>padding</i>	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

<i>rawText</i>	Input data
<i>key</i>	Key
<i>iv</i>	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

<i>level</i>	AES level
<i>mode</i>	AES Mode
<i>key</i>	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

<i>key</i>	key
------------	-----

Returns

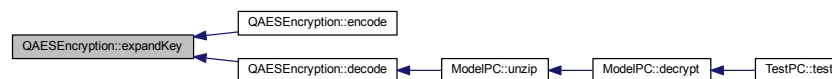
Returns expanded key (I guess)

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

<i>rawText</i>	Input data
<i>padding</i>	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

<i>rawText</i>	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:

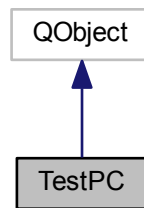
- C:/Users/salex/Documents/GitHub/PictureCrypt/aes/[qaesencryption.h](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/aes/[qaesencryption.cpp](#)

9.6 TestPC Class Reference

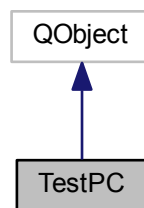
The [TestPC](#) class AutoTest for [ModelPC](#) Currently used in [main.cpp](#).

```
#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

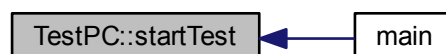
Returns success of all tests

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

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

Returns

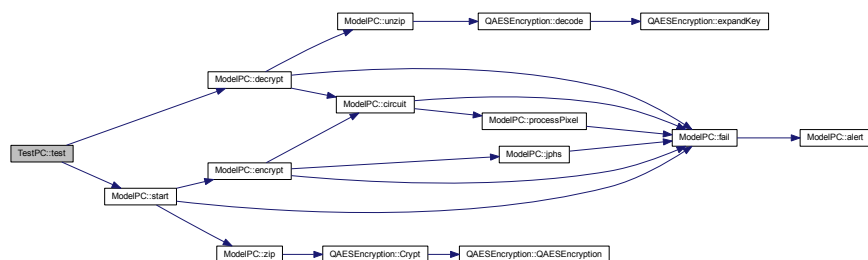
Returns if test is successful

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:

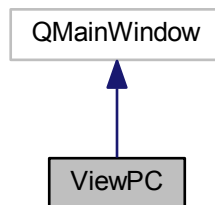
- C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.h
- C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.cpp

9.7 ViewPC Class Reference

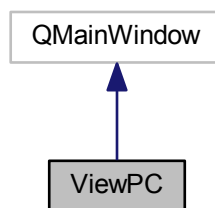
The [ViewPC](#) class View layer of the app. Controls [EncryptDialog](#) and ProgressDialog.

```
#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:

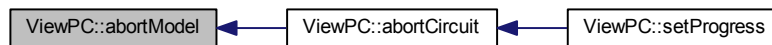


9.7.3.2 abortModel

```
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]
```

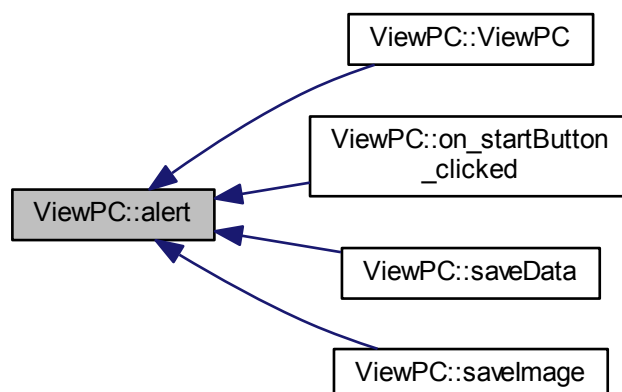
[ViewPC::alert](#) Slot to create QMessageBox with message.

Parameters

<i>message</i>	Message to be shown
<i>isWarning</i>	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

<code>_image</code>	Image for decryption
---------------------	----------------------

Here is the caller graph for this function:



9.7.3.5 encrypt

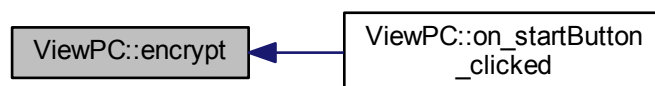
```
ViewPC::encrypt (
    QByteArray data,
    QImage * image,
    int mode ) [signal]
```

encrypt Signal calling [ModelPC::encrypt](#)

Parameters

<i>data</i>	Data to write
<i>image</i>	Image to be encrypted into.
<i>mode</i>	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

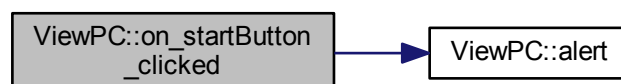
Then the [EncryptDialog](#) opens and image and key is selected. Then the [ViewPC::encrypt](#) signal is called to start [ModelPC::encrypt](#)

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

<i>Edata</i>	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

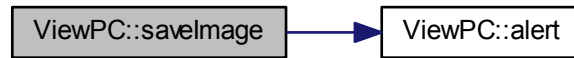
<i>image</i>	Image to be saved.
--------------	--------------------

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

<i>bitsUsed</i>	The new value
-----------------	---------------

See also

[ModelPC::bitsUsed](#)

Here is the caller graph for this function:



9.7.5.4 setEncryptMode

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

[ViewPC::setEncryptMode](#) Set the enclpt mode (`ViewPC::isEncrypt`)

Parameters

<i>encr</i>	
-------------	--

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

<i>dir</i>	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

<i>val</i>	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

<i>version</i>	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:

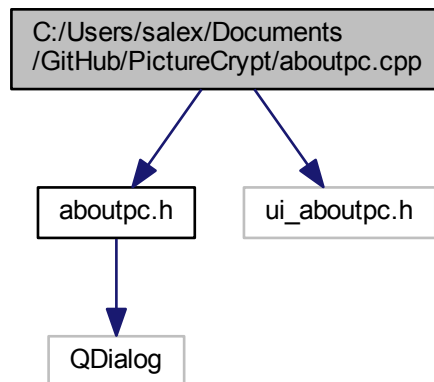
- C:/Users/salex/Documents/GitHub/PictureCrypt/[viewpc.h](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/[viewpc.cpp](#)

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:
```



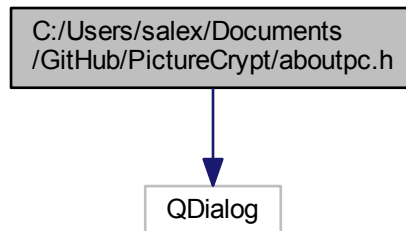
10.2 aboutpc.cpp

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

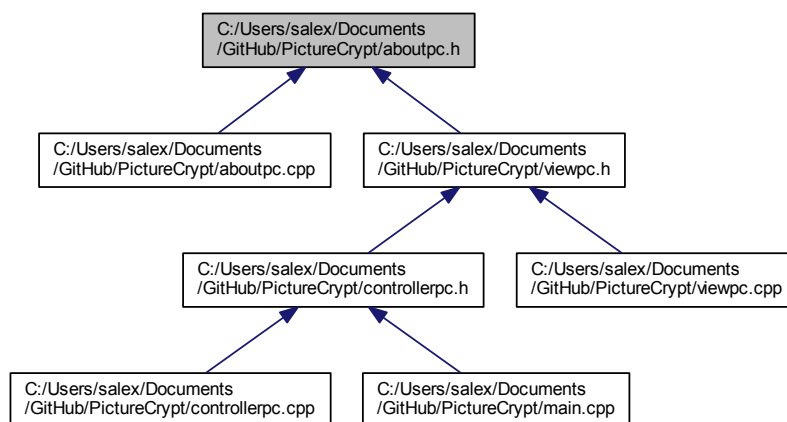
10.3 C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.h File Reference

```
#include <QDialog>
```

Include dependency graph for aboutpc.h:



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



Classes

- class [AboutPC](#)

The [AboutPC](#) class The About Page dialog.

Namespaces

- [Ui](#)

10.4 aboutpc.h

```

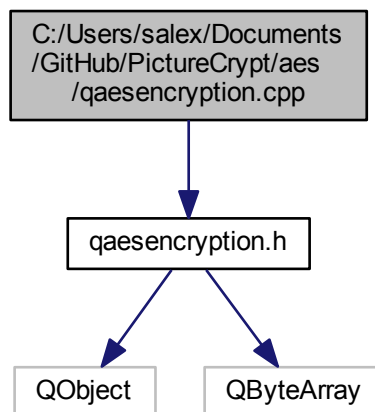
00001 #ifndef ABOUTPC_H
00002 #define ABOUTPC_H
00003
00004 #include <QDialog>
00005
00006 namespace Ui {
00007     class AboutPC;
00008 }
00012 class AboutPC : public QDialog
00013 {
00014     Q_OBJECT
00015
00016 public:
00017     explicit AboutPC(QWidget *parent = 0);
00018     ~AboutPC();
00019 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

```

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

```
#include "qaesencryption.h"
```

Include dependency graph for qaesencryption.cpp:



Functions

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

10.5.1 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

```
00001 #include "qaesencryption.h"
00002
00003 /*
00004  * Static Functions
00005  */
00006 QByteArray QAESEncryption::Crypt(QAESEncryption::Aes level,
00007     QAESEncryption::Mode mode, const QByteArray &rawText,
00008     const QByteArray &key, const QByteArray &iv,
00009     QAESEncryption::Padding padding)
00010 {
00011     return QAESEncryption(level, mode, padding).encode(rawText,
00012         key, iv);
00013 }
```

```

00011
00012 QByteArray QAESEncryption::Decrypt(QAESEncryption::Aes level,
00013                                     const QByteArray &rawText,
00014                                     const QByteArray &key, const QByteArray &iv,
00015                                     QAESEncryption::Padding padding)
00016 {
00017     return QAESEncryption(level, mode, padding).decode(rawText,
00018 key, iv);
00019 }
00020
00021 QByteArray QAESEncryption::ExpandKey(
00022     QAESEncryption::Aes level, QAESEncryption::Mode
00023     mode, const QByteArray &key)
00024 {
00025     return QAESEncryption(level, mode).expandKey(key);
00026 }
00027
00028 QByteArray QAESEncryption::RemovePadding(const QByteArray &rawText,
00029     QAESEncryption::Padding padding)
00030 {
00031     QByteArray ret(rawText);
00032     switch (padding)
00033     {
00034     case Padding::ZERO:
00035         //Works only if the last byte of the decoded array is not zero
00036         while (ret.at(ret.length()-1) == 0x00)
00037             ret.remove(ret.length()-1, 1);
00038         break;
00039     case Padding::PKCS7:
00040         ret.remove(ret.length() - ret.at(ret.length()-1), ret.at(ret.length()-1));
00041         break;
00042     case Padding::ISO:
00043         ret.truncate(ret.lastIndexOf(0x80));
00044         break;
00045     default:
00046         //do nothing
00047         break;
00048     }
00049     return ret;
00050 }
00051
00052 /*
00053 * End Static function declarations
00054 */
00055
00056 /*
00057 * Inline Functions
00058 */
00059
00060 inline quint8 xTime(quint8 x){
00061     return ((x<<1) ^ (((x>>7) & 1) * 0x1b));
00062 }
00063
00064 inline quint8 multiply(quint8 x, quint8 y){
00065     return ((y & 1) * x) ^ ((y>>1 & 1) * xTime(x)) ^ ((y>>2 & 1) * xTime(
00066 xTime(x))) ^ ((y>>3 & 1)
00067 * xTime(xTime(xTime(x)))) ^ ((y>>4 & 1) * xTime(
00068 xTime(xTime(xTime(x)))));
00069 }
00070
00071 /*
00072 * End Inline functions
00073 */
00074
00075 QAESEncryption::QAESEncryption(Aes level, Mode
00076 mode,
00077                                 Padding padding)
00078 : m_nb(4), m_blocklen(16), m_level(level), m_mode(mode), m_padding(padding)
00079 {
00080     m_state = NULL;
00081
00082     switch (level)
00083     {
00084     case AES_128: {
00085         AES128 aes;
00086         m_nk = aes.nk;
00087         m_keyLen = aes.keylen;
00088         m_nr = aes.nr;
00089         m_expandedKey = aes.expandedKey;
00090     }
00091     break;
00092     case AES_192: {
00093         AES192 aes;
00094         m_nk = aes.nk;
00095         m_keyLen = aes.keylen;
00096         m_nr = aes.nr;
00097         m_expandedKey = aes.expandedKey;
00098     }
00099     }
00100 }

```

```

00089     }
00090     break;
00091     case AES_256: {
00092         AES256 aes;
00093         m_nk = aes.nk;
00094         m_keyLen = aes.keylen;
00095         m_nr = aes.nr;
00096         m_expandedKey = aes.expandedKey;
00097     }
00098     break;
00099     default: {
00100         AES128 aes;
00101         m_nk = aes.nk;
00102         m_keyLen = aes.keylen;
00103         m_nr = aes.nr;
00104         m_expandedKey = aes.expandedKey;
00105     }
00106     break;
00107 }
00108 }
00109 }
00110 QByteArray QAESEncryption::getPadding(int currSize, int alignment)
00111 {
00112     int size = (alignment - currSize % alignment) % alignment;
00113     if (size == 0) return QByteArray();
00114     switch(m_padding)
00115     {
00116     case Padding::ZERO:
00117         return QByteArray(size, 0x00);
00118         break;
00119     case Padding::PKCS7:
00120         return QByteArray(size, size);
00121         break;
00122     case Padding::ISO:
00123         return QByteArray (size-1, 0x00).prepend(0x80);
00124         break;
00125     default:
00126         return QByteArray(size, 0x00);
00127         break;
00128     }
00129     return QByteArray(size, 0x00);
00130 }
00131
00132 QByteArray QAESEncryption::expandKey(const QByteArray &
00133                                     key)
00134 {
00135     int i, k;
00136     quint8 tempa[4]; // Used for the column/row operations
00137     QByteArray roundKey(key);
00138     // The first round key is the key itself.
00139     // ...
00140     // All other round keys are found from the previous round keys.
00141     // i == Nk
00142     for(i = m_nk; i < m_nb * (m_nr + 1); i++)
00143     {
00144         tempa[0] = (quint8) roundKey.at((i-1) * 4 + 0);
00145         tempa[1] = (quint8) roundKey.at((i-1) * 4 + 1);
00146         tempa[2] = (quint8) roundKey.at((i-1) * 4 + 2);
00147         tempa[3] = (quint8) roundKey.at((i-1) * 4 + 3);
00148
00149         if (i % m_nk == 0)
00150         {
00151             // This function shifts the 4 bytes in a word to the left once.
00152             // [a0,a1,a2,a3] becomes [a1,a2,a3,a0]
00153
00154             // Function RotWord()
00155             k = tempa[0];
00156             tempa[0] = tempa[1];
00157             tempa[1] = tempa[2];
00158             tempa[2] = tempa[3];
00159             tempa[3] = k;
00160
00161             // Function Subword()
00162             tempa[0] = getSBoxValue(tempa[0]);
00163             tempa[1] = getSBoxValue(tempa[1]);
00164             tempa[2] = getSBoxValue(tempa[2]);
00165             tempa[3] = getSBoxValue(tempa[3]);
00166
00167             tempa[0] = tempa[0] ^ Rcon[i/m_nk];
00168         }
00169         if (m_level == AES_256 && i % m_nk == 4)
00170         {
00171             // Function Subword()
00172             tempa[0] = getSBoxValue(tempa[0]);
00173             tempa[1] = getSBoxValue(tempa[1]);
00174

```

```

00175         tempa[2] = getSBoxValue(tempa[2]);
00176         tempa[3] = getSBoxValue(tempa[3]);
00177     }
00178     roundKey.insert(i * 4 + 0, (quint8) roundKey.at((i - m_nk) * 4 + 0) ^ tempa[0]);
00179     roundKey.insert(i * 4 + 1, (quint8) roundKey.at((i - m_nk) * 4 + 1) ^ tempa[1]);
00180     roundKey.insert(i * 4 + 2, (quint8) roundKey.at((i - m_nk) * 4 + 2) ^ tempa[2]);
00181     roundKey.insert(i * 4 + 3, (quint8) roundKey.at((i - m_nk) * 4 + 3) ^ tempa[3]);
00182 }
00183 return roundKey;
00184 }
00185
00186 // This function adds the round key to state.
00187 // The round key is added to the state by an XOR function.
00188 void QAESEncryption::addRoundKey(const quint8 round, const QByteArray expKey)
00189 {
00190     QByteArray::iterator it = m_state->begin();
00191     for(int i=0; i < 16; ++i)
00192         it[i] = (quint8) it[i] ^ (quint8) expKey.at(round * m_nb * 4 + (i/4) * m_nb + (i%4));
00193 }
00194
00195 // The SubBytes Function Substitutes the values in the
00196 // state matrix with values in an S-box.
00197 void QAESEncryption::subBytes()
00198 {
00199     QByteArray::iterator it = m_state->begin();
00200     for(int i = 0; i < 16; i++)
00201         it[i] = getSBoxValue((quint8) it[i]);
00202 }
00203
00204 // The ShiftRows() function shifts the rows in the state to the left.
00205 // Each row is shifted with different offset.
00206 // Offset = Row number. So the first row is not shifted.
00207 void QAESEncryption::shiftRows()
00208 {
00209     QByteArray::iterator it = m_state->begin();
00210     quint8 temp;
00211     //Keep in mind that QByteArray is column-driven!!
00212
00213     //Shift 1 to left
00214     temp = (quint8)it[1];
00215     it[1] = (quint8)it[5];
00216     it[5] = (quint8)it[9];
00217     it[9] = (quint8)it[13];
00218     it[13] = (quint8)temp;
00219
00220     //Shift 2 to left
00221     temp = (quint8)it[2];
00222     it[2] = (quint8)it[10];
00223     it[10] = (quint8)temp;
00224     temp = (quint8)it[6];
00225     it[6] = (quint8)it[14];
00226     it[14] = (quint8)temp;
00227
00228     //Shift 3 to left
00229     temp = (quint8)it[3];
00230     it[3] = (quint8)it[15];
00231     it[15] = (quint8)it[11];
00232     it[11] = (quint8)it[7];
00233     it[7] = (quint8)temp;
00234 }
00235
00236 // MixColumns function mixes the columns of the state matrix
00237 //optimized!!
00238 void QAESEncryption::mixColumns()
00239 {
00240     QByteArray::iterator it = m_state->begin();
00241     quint8 tmp, tm, t;
00242
00243     for(int i = 0; i < 16; i += 4){
00244         t = (quint8)it[i];
00245         tmp = (quint8)it[i] ^ (quint8)it[i+1] ^ (quint8)it[i+2] ^ (quint8)it[i+3] ;
00246
00247         tm = xTime( (quint8)it[i] ^ (quint8)it[i+1] );
00248         it[i] = (quint8)it[i] ^ (quint8)tm ^ (quint8)tmp;
00249
00250         tm = xTime( (quint8)it[i+1] ^ (quint8)it[i+2] );
00251         it[i+1] = (quint8)it[i+1] ^ (quint8)tm ^ (quint8)tmp;
00252
00253         tm = xTime( (quint8)it[i+2] ^ (quint8)it[i+3] );
00254         it[i+2] = (quint8)it[i+2] ^ (quint8)tm ^ (quint8)tmp;
00255
00256         tm = xTime( (quint8)it[i+3] ^ (quint8)t );
00257         it[i+3] = (quint8)it[i+3] ^ (quint8)tm ^ (quint8)tmp;
00258     }
00259 }
00260
00261 // MixColumns function mixes the columns of the state matrix.

```

```

00262 // The method used to multiply may be difficult to understand for the inexperienced.
00263 // Please use the references to gain more information.
00264 void QAESEncryption::invMixColumns()
00265 {
00266     QByteArray::iterator it = m_state->begin();
00267     quint8 a,b,c,d;
00268     for(int i = 0; i < 16; i+=4){
00269         a = (quint8) it[i];
00270         b = (quint8) it[i+1];
00271         c = (quint8) it[i+2];
00272         d = (quint8) it[i+3];
00273
00274         it[i] = (quint8) (multiply(a, 0x0e) ^ multiply(b, 0x0b) ^
multiply(c, 0x0d) ^ multiply(d, 0x09));
00275         it[i+1] = (quint8) (multiply(a, 0x09) ^ multiply(b, 0x0e) ^
multiply(c, 0x0b) ^ multiply(d, 0x0d));
00276         it[i+2] = (quint8) (multiply(a, 0x0d) ^ multiply(b, 0x09) ^
multiply(c, 0x0e) ^ multiply(d, 0x0b));
00277         it[i+3] = (quint8) (multiply(a, 0x0b) ^ multiply(b, 0x0d) ^
multiply(c, 0x09) ^ multiply(d, 0x0e));
00278     }
00279 }
00280
00281 // The SubBytes Function Substitutes the values in the
00282 // state matrix with values in an S-box.
00283 void QAESEncryption::invSubBytes()
00284 {
00285     QByteArray::iterator it = m_state->begin();
00286     for(int i = 0; i < 16; ++i)
00287         it[i] = getSBoxInvert((quint8) it[i]);
00288 }
00289
00290 void QAESEncryption::invShiftRows()
00291 {
00292     QByteArray::iterator it = m_state->begin();
00293     uint8_t temp;
00294
00295     //Keep in mind that QByteArray is column-driven!!
00296
00297     //Shift 1 to right
00298     temp = (quint8)it[13];
00299     it[13] = (quint8)it[9];
00300     it[9] = (quint8)it[5];
00301     it[5] = (quint8)it[1];
00302     it[1] = (quint8)temp;
00303
00304     //Shift 2
00305     temp = (quint8)it[10];
00306     it[10] = (quint8)it[2];
00307     it[2] = (quint8)temp;
00308     temp = (quint8)it[14];
00309     it[14] = (quint8)it[6];
00310     it[6] = (quint8)temp;
00311
00312     //Shift 3
00313     temp = (quint8)it[15];
00314     it[15] = (quint8)it[3];
00315     it[3] = (quint8)it[7];
00316     it[7] = (quint8)it[11];
00317     it[11] = (quint8)temp;
00318 }
00319
00320 QByteArray QAESEncryption::byteXor(const QByteArray &a, const QByteArray &b)
00321 {
00322     QByteArray::const_iterator it_a = a.begin();
00323     QByteArray::const_iterator it_b = b.begin();
00324     QByteArray ret;
00325
00326     //for(int i = 0; i < m_blocklen; i++)
00327     for(int i = 0; i < std::min(a.size(), b.size()); i++)
00328         ret.insert(i, it_a[i] ^ it_b[i]);
00329
00330     return ret;
00331 }
00332
00333 // Cipher is the main function that encrypts the PlainText.
00334 QByteArray QAESEncryption::cipher(const QByteArray &expKey, const QByteArray &in)
00335 {
00336
00337     //m_state is the input buffer...
00338     QByteArray output(in);
00339     m_state = &output;
00340
00341     // Add the First round key to the state before starting the rounds.
00342     addRoundKey(0, expKey);
00343
00344     // There will be Nr rounds.

```



```

00345 // The first Nr-1 rounds are identical.
00346 // These Nr-1 rounds are executed in the loop below.
00347 for(uint8 round = 1; round < m_nr; ++round){
00348     subBytes();
00349     shiftRows();
00350     mixColumns();
00351     addRoundKey(round, expKey);
00352 }
00353
00354 // The last round is given below.
00355 // The MixColumns function is not here in the last round.
00356 subBytes();
00357 shiftRows();
00358 addRoundKey(m_nr, expKey);
00359
00360 return output;
00361 }
00362
00363 QByteArray QAESEncryption::invCipher(const QByteArray &expKey, const QByteArray &in)
00364 {
00365     //m_state is the input buffer.... handle it!
00366     QByteArray output(in);
00367     m_state = &output;
00368
00369     // Add the First round key to the state before starting the rounds.
00370     addRoundKey(m_nr, expKey);
00371
00372     // There will be Nr rounds.
00373     // The first Nr-1 rounds are identical.
00374     // These Nr-1 rounds are executed in the loop below.
00375     for(uint8 round=m_nr-1; round>0 ; round--){
00376         invShiftRows();
00377         invSubBytes();
00378         addRoundKey(round, expKey);
00379         invMixColumns();
00380     }
00381
00382     // The last round is given below.
00383     // The MixColumns function is not here in the last round.
00384     invShiftRows();
00385     invSubBytes();
00386     addRoundKey(0, expKey);
00387
00388     return output;
00389 }
00390
00391 QByteArray QAESEncryption::encode(const QByteArray &rawText, const QByteArray &
key, const QByteArray &iv)
00392 {
00393     if (m_mode >= CBC && (iv.isNull() || iv.size() != m_blocklen))
00394         return QByteArray();
00395
00396     QByteArray ret;
00397     QByteArray expandedKey = expandKey(key);
00398     QByteArray alignedText(rawText);
00399
00400     //Fill array with padding
00401     alignedText.append(getPadding(rawText.size(), m_blocklen));
00402
00403     switch(m_mode)
00404     {
00405     case ECB:
00406         for(int i=0; i < alignedText.size(); i+= m_blocklen)
00407             ret.append(cipher(expandedKey, alignedText.mid(i, m_blocklen)));
00408         break;
00409     case CBC: {
00410         QByteArray ivTemp(iv);
00411         for(int i=0; i < alignedText.size(); i+= m_blocklen) {
00412             alignedText.replace(i, m_blocklen, byteXor(alignedText.mid(i, m_blocklen), ivTemp));
00413             ret.append(cipher(expandedKey, alignedText.mid(i, m_blocklen)));
00414             ivTemp = ret.mid(i, m_blocklen);
00415         }
00416     }
00417     case CFB: {
00418         ret.append(byteXor(alignedText.left(m_blocklen), cipher(expandedKey, iv)));
00419         for(int i=0; i < alignedText.size(); i+= m_blocklen) {
00420             if (i+m_blocklen < alignedText.size())
00421                 ret.append(byteXor(alignedText.mid(i+m_blocklen, m_blocklen),
00422                                     cipher(expandedKey, ret.mid(i, m_blocklen))));
00423         }
00424     }
00425     case OFB: {
00426         QByteArray ofbTemp;
00427         ofbTemp.append(cipher(expandedKey, iv));
00428         for (int i=m_blocklen; i < alignedText.size(); i += m_blocklen){

```

```

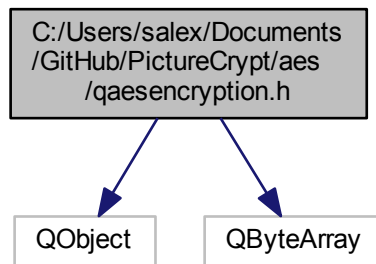
00431         ofbTemp.append(cipher(expandedKey, ofbTemp.right(m_blocklen)));
00432     }
00433     ret.append(byteXor(alignedText, ofbTemp));
00434 }
00435     break;
00436 default: break;
00437 }
00438     return ret;
00439 }
00440
00441 QByteArray QAESEncryption::decode(const QByteArray &rawText, const QByteArray &
key, const QByteArray &iv)
00442 {
00443     if (m_mode >= CBC && (iv.isNull() || iv.size() != m_blocklen))
00444         return QByteArray();
00445     QByteArray ret;
00446     QByteArray expandedKey = expandKey(key);
00447     switch(m_mode)
00448     {
00449     case ECB:
00450     {
00451         for(int i=0; i < rawText.size(); i+= m_blocklen)
00452             ret.append(invCipher(expandedKey, rawText.mid(i, m_blocklen)));
00453         break;
00454     case CBC: {
00455         QByteArray ivTemp(iv);
00456         for(int i=0; i < rawText.size(); i+= m_blocklen){
00457             ret.append(invCipher(expandedKey, rawText.mid(i, m_blocklen)));
00458             ret.replace(i, m_blocklen, byteXor(ret.mid(i, m_blocklen), ivTemp));
00459             ivTemp = rawText.mid(i, m_blocklen);
00460         }
00461     }
00462     break;
00463     case CFB: {
00464         ret.append(byteXor(rawText.mid(0, m_blocklen), cipher(expandedKey, iv)));
00465         for(int i=0; i < rawText.size(); i+= m_blocklen){
00466             if (i+m_blocklen < rawText.size()) {
00467                 ret.append(byteXor(rawText.mid(i+m_blocklen, m_blocklen),
00468                                     cipher(expandedKey, rawText.mid(i, m_blocklen))));
00469             }
00470         }
00471     }
00472     break;
00473     case OFB: {
00474         QByteArray ofbTemp;
00475         ofbTemp.append(cipher(expandedKey, iv));
00476         for (int i=m_blocklen; i < rawText.size(); i += m_blocklen){
00477             ofbTemp.append(cipher(expandedKey, ofbTemp.right(m_blocklen)));
00478         }
00479         ret.append(byteXor(rawText, ofbTemp));
00480     }
00481     break;
00482     default:
00483         //do nothing
00484         break;
00485     }
00486     return ret;
00487 }
00488 }
00489
00490 QByteArray QAESEncryption::removePadding(const QByteArray &rawText)
00491 {
00492     QByteArray ret(rawText);
00493     switch (m_padding)
00494     {
00495     case Padding::ZERO:
00496         //Works only if the last byte of the decoded array is not zero
00497         while (ret.at(ret.length()-1) == 0x00)
00498             ret.remove(ret.length()-1, 1);
00499         break;
00500     case Padding::PKCS7:
00501         ret.remove(ret.length() - ret.at(ret.length()-1), ret.at(ret.length()-1));
00502         break;
00503     case Padding::ISO:
00504         ret.truncate(ret.lastIndexOf(0x80));
00505         break;
00506     default:
00507         //do nothing
00508         break;
00509     }
00510     return ret;
00511 }

```

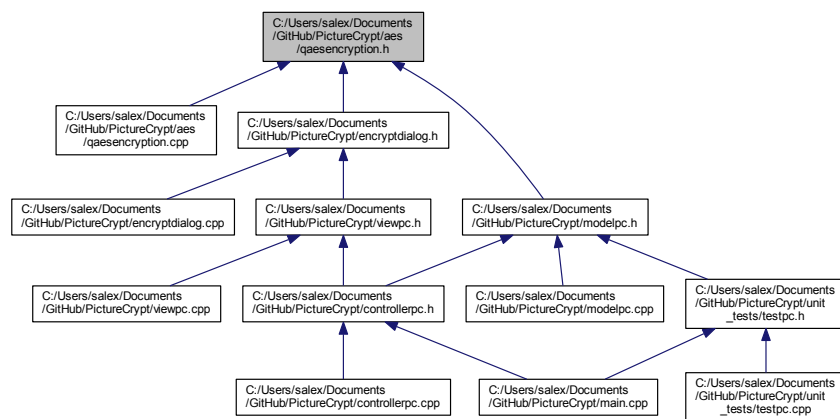
10.7 C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencryption.h File Reference

```
#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

```

00001 #ifndef QAESENCRYPTION_H
00002 #define QAESENCRYPTION_H
00003
00004 #include <QObject>
00005 #include <QByteArray>
00006
00014 class QAESEncryption : public QObject
00015 {
00016     Q_OBJECT
00017 public:
00027     enum Aes {
00028         AES_128,
00029         AES_192,
00030         AES_256
00031     };
00040     enum Mode {
00041         ECB,
00042         CBC,
00043         CFB,
00044         OFB
00045     };
00046
00055     enum Padding {
00056         ZERO,
00057         PKCS7,
00058         ISO
00059     };
00071     static QByteArray Crypt(QAESEncryption::Aes level,
QAESEncryption::Mode mode, const QByteArray &rawText, const QByteArray &
key,
00072                             const QByteArray &iv = NULL, QAESEncryption::Padding
padding = QAESEncryption::ISO);
00084     static QByteArray Decrypt(QAESEncryption::Aes level,
QAESEncryption::Mode mode, const QByteArray &rawText, const QByteArray &
key,
00085                             const QByteArray &iv = NULL,
QAESEncryption::Padding padding = QAESEncryption::ISO);
00094     static QByteArray ExpandKey(QAESEncryption::Aes level,
QAESEncryption::Mode mode, const QByteArray &key);
00102     static QByteArray RemovePadding(const QByteArray &rawText,
QAESEncryption::Padding padding);
00103
00104     QAESEncryption(QAESEncryption::Aes level,
QAESEncryption::Mode mode,
00105                   QAESEncryption::Padding padding =
QAESEncryption::ISO);
00116     QByteArray encode(const QByteArray &rawText, const QByteArray &key, const QByteArray &iv =
NULL);
00127     QByteArray decode(const QByteArray &rawText, const QByteArray &key, const QByteArray &iv =
NULL);
00136     QByteArray removePadding(const QByteArray &rawText);
00145     QByteArray expandKey(const QByteArray &key);
00146
00147 signals:
00148
00149 public slots:
00150
00151 private:
00152     int m_nb;
00153     int m_blocklen;
00154     int m_level;
00155     int m_mode;
00156     int m_nk;
00157     int m_keyLen;
00158     int m_nr;
00159     int m_expandedKey;
00160     int m_padding;
00161     QByteArray* m_state;
00162
00163     struct AES256{
00164         int nk = 8;
00165         int keylen = 32;
00166         int nr = 14;
00167         int expandedKey = 240;
00168     };
00169
00170     struct AES192{
00171         int nk = 6;
00172         int keylen = 24;
00173         int nr = 12;
00174         int expandedKey = 209;
00175     };
00176
00177     struct AES128{

```

```

00178     int nk = 4;
00179     int keylen = 16;
00180     int nr = 10;
00181     int expandedKey = 176;
00182 };
00183
00184 quint8 getSBoxValue(quint8 num){return sbox[num];}
00185 quint8 getSBoxInvert(quint8 num){return rsbox[num];}
00186
00187 void addRoundKey(const quint8 round, const QByteArray expKey);
00188 void subBytes();
00189 void shiftRows();
00190 void mixColumns();
00191 void invMixColumns();
00192 void invSubBytes();
00193 void invShiftRows();
00194 QByteArray getPadding(int currSize, int alignment);
00195 QByteArray cipher(const QByteArray &expKey, const QByteArray &plainText);
00196 QByteArray invCipher(const QByteArray &expKey, const QByteArray &plainText);
00197 QByteArray byteXor(const QByteArray &in, const QByteArray &iv);
00198
00199 const quint8 sbox[256] = {
00200     //0      1      2      3      4      5      6      7      8      9      A      B      C      D      E      F
00201     0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
00202     0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
00203     0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
00204     0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
00205     0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
00206     0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
00207     0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
00208     0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
00209     0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
00210     0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
00211     0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
00212     0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
00213     0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
00214     0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
00215     0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
00216     0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 };
00217
00218 const quint8 rsbox[256] =
00219 { 0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
00220   0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
00221   0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
00222   0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
00223   0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
00224   0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
00225   0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
00226   0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
00227   0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
00228   0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
00229   0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
00230   0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
00231   0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
00232   0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
00233   0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
00234   0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d };
00235
00236 // The round constant word array, Rcon[i], contains the values given by
00237 // x to the power (i-1) being powers of x (x is denoted as {02}) in the field GF(2^8)
00238 // Only the first 14 elements are needed
00239 const quint8 Rcon[256] = {
00240     0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab/*, 0x4d, 0x9a,
00241     0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
00242     0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,
00243     0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,
00244     0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,
00245     0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,
00246     0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
00247     0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
00248     0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
00249     0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
00250     0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
00251     0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
00252     0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
00253     0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,
00254     0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,
00255     0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d
00256 */};
00257
00258 #endif // QAESENCRIPTION_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

```

00001 # Contributor Covenant Code of Conduct
00002
00003 ## Our Pledge
00004
00005 In the interest of fostering an open and welcoming environment, we as contributors and maintainers
    pledge to making participation in our project and our community a harassment-free experience for everyone,
    regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience,
    nationality, personal appearance, race, religion, or sexual identity and orientation.
00006
00007 ## Our Standards
00008
00009 Examples of behavior that contributes to creating a positive environment include:
00010
00011 * Using welcoming and inclusive language
00012 * Being respectful of differing viewpoints and experiences
00013 * Gracefully accepting constructive criticism
00014 * Focusing on what is best for the community
00015 * Showing empathy towards other community members
00016
00017 Examples of unacceptable behavior by participants include:
00018
00019 * The use of sexualized language or imagery and unwelcome sexual attention or advances
00020 * Trolling, insulting/derogatory comments, and personal or political attacks
00021 * Public or private harassment
00022 * Publishing others' private information, such as a physical or electronic address, without explicit
    permission
00023 * Other conduct which could reasonably be considered inappropriate in a professional setting
00024
00025 ## Our Responsibilities
00026
00027 Project maintainers are responsible for clarifying the standards of acceptable behavior and are
    expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.
00028
00029 Project maintainers have the right and responsibility to remove, edit, or reject comments, commits,
    code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban
    temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening,
    offensive, or harmful.
00030
00031 ## Scope
00032
00033 This Code of Conduct applies both within project spaces and in public spaces when an individual is
    representing the project or its community. Examples of representing a project or community include using an
    official project e-mail address, posting via an official social media account, or acting as an appointed
    representative at an online or offline event. Representation of a project may be further defined and clarified by
    project maintainers.
00034
00035 ## Enforcement
00036
00037 Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the
    project team at a.kovrigin0@gmail.com. The project team will review and investigate all complaints, and will
    respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain
    confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies
    may be posted separately.
00038
00039 Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary
    or permanent repercussions as determined by other members of the project's leadership.
00040
00041 ## Attribution
00042
00043 This Code of Conduct is adapted from the [Contributor Covenant][homepage], version 1.4, available at
    [http://contributor-covenant.org/version/1/4][version]
00044
00045 [homepage]: http://contributor-covenant.org
00046 [version]: http://contributor-covenant.org/version/1/4/

```

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

10.12 ErrorsDict.json

```

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

```

10.13 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

```

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

```



```

00019 class ControllerPC : public QObject
00020 {
00021     Q_OBJECT
00022 public:
00023     ControllerPC();
00027     long int version;
00031     QString versionString;
00032 public slots:
00033     void abortCircuit();
00034     void setBitsUsed(int bitsUsed);
00035     void setJPHSDir(QString dir);
00036 private:
00037     ViewPC * view;
00038     ModelPC * model;
00039 };
00040
00041 #endif // CONTROLLERPC_H

```

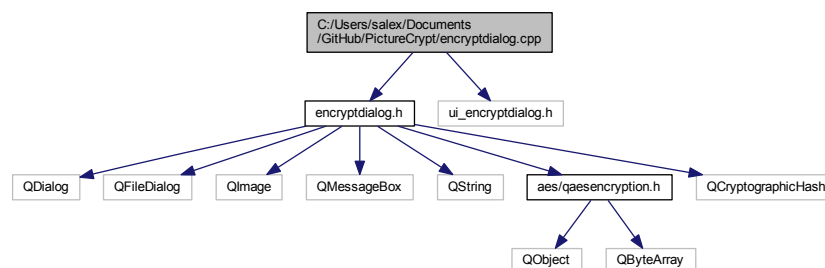
10.19 C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.cpp File Reference

```

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

```

Include dependency graph for encryptdialog.cpp:



10.20 encryptdialog.cpp

```

00001 #include "encryptdialog.h"
00002 #include "ui_encryptdialog.h"
00009 EncryptDialog::EncryptDialog(QByteArray _data, QWidget *parent) :
00010     QDialog(parent),
00011     ui(new Ui::EncryptDialog)
00012 {
00013     ui->setupUi(this);
00014     data = _data;
00015     success = false;
00016     // UI setup
00017     ui->totalBytes->setText(QString::number(data.size()));
00018     key.clear();
00019     for(int i = 0; i < 24; i++)
00020         key.append(48 + qrand() % 75);
00021     val = 24;
00022     compr_data = zip();
00023     long long int compr_data_size = compr_data.size();
00024     ui->zippedBytes->setText(QString::number(compr_data_size));
00025     goodPercentage = false;
00026     bitsUsed = 8;
00027 }
00028
00029 EncryptDialog::~EncryptDialog()
00030 {
00031     delete ui;
00032 }
00033
00034 void EncryptDialog::alert(QString text)

```

```

00035 {
00036     QMessageBox t;
00037     t.setWindowTitle("Message");
00038     t.setIcon(QMessageBox::Warning);
00039     t.setWindowIcon(QIcon(":/mail.png"));
00040     t.setText(text);
00041     t.exec();
00042 }
00049 QByteArray EncryptDialog::zip()
00050 {
00051     // Zip
00052     QByteArray c_data = qCompress(data, 9);
00053     // Encryption
00054     QByteArray hashKey = QCryptographicHash::hash(key.toUtf8(), QCryptographicHash::Sha256);
00055     return QAESEncryption::Crypt(QAESEncryption::AES_256,
    QAESEncryption::ECB, c_data, hashKey);
00056 }
00060 void EncryptDialog::on_fileButton_clicked()
00061 {
00062     // Selet file
00063     inputFileName = QFileDialog::getOpenFileName(this, tr("Open File"), "/", tr("Images (*.png
    *.xpm *.jpg *.jpeg)"));
00064     ui->fileLabel->setText(inputFileName);
00065     // Open image
00066     QImage img(inputFileName);
00067     image = img;
00068     // Get size
00069     size = img.width() * img.height();
00070     // UI setup
00071     long long int compr_data_size = compr_data.size();
00072     ui->zippedBytes->setText(QString::number(compr_data_size));
00073     if(inputFileName.isEmpty()) {
00074         ui->percentage->setText("");
00075         return;
00076     }
00077     double perc = (compr_data_size + 14 + val) * 100 / (size * 3) *
    bitsUsed / 8;
00078     ui->percentage->setText(QString::number(perc) + "%");
00079     goodPercentage = perc < 70;
00080 }
00085 void EncryptDialog::on_buttonBox_accepted()
00086 {
00087     if(!goodPercentage) {
00088         alert("Your encoding percentage is over 70% which is a bit ambiguous :(");
00089         success = false;
00090         return;
00091     }
00092     // Final zip
00093     compr_data = zip();
00094     success = true;
00095     close();
00096 }
00100 void EncryptDialog::on_buttonBox_rejected()
00101 {
00102     success = false;
00103     close();
00104 }
00110 void EncryptDialog::on_horizontalSlider_valueChanged(int
    value)
00111 {
00112     // Key generator with value of charachters
00113     key.clear();
00114     for(int i = 0; i < value; i++)
00115         key.append(48 + qrand() % 75);
00116     val = value;
00117     ui->keyLabel->setText(QString::number(value));
00118 }
00123 void EncryptDialog::on_bitsSlider_valueChanged(int value)
00124 {
00125     bitsUsed = value;
00126     ui->bitsUsedLbl->setText(QString::number(value));
00127     if(ui->percentage->text().isEmpty())
00128         return;
00129     double perc = (compr_data.size() + 14 + val) * 100 / (size * 3) * 8 /
    bitsUsed;
00130     ui->percentage->setText(QString::number(perc) + "%");
00131 }

```

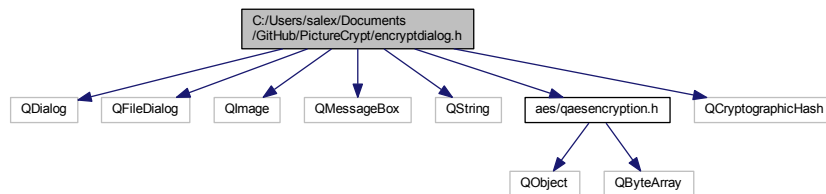
10.21 C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.h File Reference

```

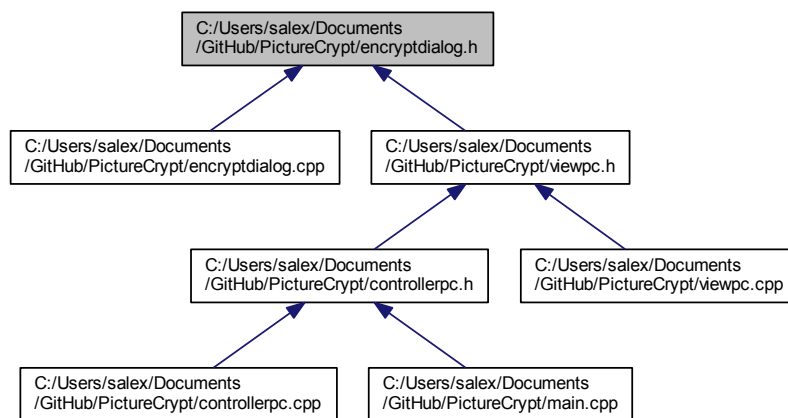
#include <QDialog>
#include <QFileDialog>

```

```
#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:



Classes

- class [EncryptDialog](#)

The [EncryptDialog](#) class Class to get the image and key to store secret info.

Namespaces

- [Ui](#)

10.22 encryptdialog.h

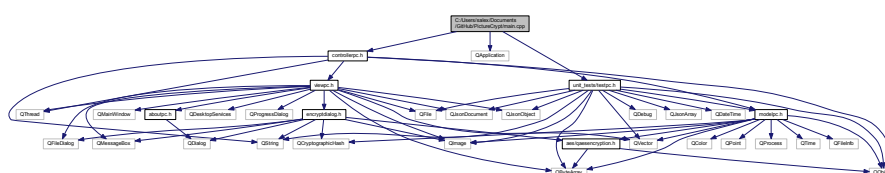
```

00001 #ifndef ENCRYPTDIALOG_H
00002 #define ENCRYPTDIALOG_H
00003
00004 #include <QDialog>
00005 #include <QFileDialog>
00006 #include <QImage>
00007 #include <QMessageBox>
00008 #include <QString>
00009
00010 #include <aes/qaesencryption.h>
00011 #include <QCryptographicHash>
00012
00013 namespace Ui {
00014 class EncryptDialog;
00015 }
00021 class EncryptDialog : public QDialog
00022 {
00023     Q_OBJECT
00024
00025 public:
00026     explicit EncryptDialog(QByteArray _data, QWidget *parent = 0);
00027     ~EncryptDialog();
00028
00029 public slots:
00030     void on_fileButton_clicked();
00031
00032     void on_buttonBox_accepted();
00033
00034     void on_buttonBox_rejected();
00035
00036     void on_horizontalSlider_valueChanged(int
value);
00037
00038 public:
00042     QByteArray data;
00046     bool success;
00050     QByteArray compr_data;
00054     QString inputFileNames;
00058     long long int size;
00062     QString key;
00066     bool goodPercentage;
00070     int val;
00075     int bitsUsed;
00079     QImage image;
00080     QByteArray zip();
00081 private slots:
00082     void on_bitsSlider_valueChanged(int value);
00083
00084 private:
00085     Ui::EncryptDialog *ui;
00086     void alert(QString text);
00087 };
00088
00089 #endif // ENCRYPTDIALOG_H

```

10.23 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:



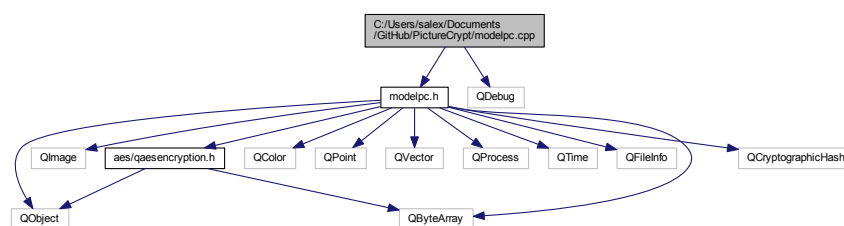
10.24 main.cpp

```
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

```

00001 #include "modelpc.h"
00002 #include <QDebug>
00008 ModelPC::ModelPC()
00009 {
00010     // Version control
00011     versionString = "1.3.0";
00012
00013     auto ver = versionString.split(".");
00014     version = ver[0].toInt() * pow(2, 16) + ver[1].toInt() * pow(2, 8) + ver[2].toInt();
00015
00016     ver_byte = bytes(ver[0].toInt()) +
00017               bytes(ver[1].toInt()) +
00018               bytes(ver[2].toInt());
00019     // Random seed
00020     qsrand(randSeed());
00021 }
00034 QImage * ModelPC::start(QByteArray data, QImage * image, int
mode, QString key, int _bitsUsed, QString *_error)
00035 {
00036     // Error management
00037     *_error = "ok";
00038     error = _error;
00039
00040     if(data.isEmpty()) {
00041         fail("nodata");
00042         return nullptr;
00043     }
00044     if(image == nullptr || image->isNull()) {
00045         fail("nullimage");
00046         return nullptr;
00047     }
00048     if(_bitsUsed < 1 || _bitsUsed > 8) {
00049         fail("bitsWrong");
00050         return nullptr;
00051     }
00052     if(key.isEmpty()) {
00053         qsrand(randSeed());
00054         for(int i = 0; i < 32; i++)
00055             key.append(48 + qrand() % 75);
00056     }
00057     else if(key.size() > 255) {
00058         fail("bigkey");
00059         return nullptr;
00060     }
00061     long long usedBytes = data.size() + 14 + key.size();
00062     long long size = image->width() * image->height();
00063     if(usedBytes * 100 / (size * 3) * 8 / _bitsUsed > 70) {
00064         fail("muchdata");
00065         return nullptr;
00066     }
00067
00068     curMode = mode;
00069     bitsUsed = _bitsUsed;
00070
00071     QByteArray key_data = key.toUtf8();
00072     QByteArray zipped_data = zip(data, key_data);
00073     QByteArray encr_data = bytes(key_data.size()) + key_data + zipped_data;
00074
00075     if(*error == "ok")
00076         return encrypt(encr_data, image, curMode, error);
00077     else
00078         return nullptr;
00079 }
00080
00090 QImage * ModelPC::encrypt(QByteArray encr_data, QImage * image, int
mode, QString *_error)
00091 {
00092     // Error management
00093     *_error = "ok";
00094     error = _error;
00095
00096     // TODO Remove debug mode = 0
00097     mode = 0;
00098
00099     if(encr_data.isEmpty()) {
00100         fail("nodata");
00101         return nullptr;
00102     }
00103     if(image == nullptr || image->isNull()) {
00104         fail("nullimage");
00105         return nullptr;
00106     }
00107
00108     encr_data = ver_byte + encr_data;

```

```

00109     long long int countBytes = encr_data.size();
00110     curMode = mode;
00111     switch(curMode)
00112     {
00113     case 0:
00114         circuit(image, &encr_data, countBytes);
00115         break;
00116     case 1:
00117         jphs(image, &encr_data);
00118         break;
00119     default:
00120         fail("wrongmode");
00121         return nullptr;
00122     }
00123
00124
00125     // Saving
00126     if(success) {
00127         emit saveImage(image);
00128         return image;
00129     }
00130     else
00131         return nullptr;
00132 }
00140 QByteArray ModelPC::decrypt(QImage * image, QString *_error)
00141 {
00142     // Error management
00143     *_error = "ok";
00144     error = _error;
00145     if(image == nullptr || image->isNull()) {
00146         fail("nullimage");
00147         return nullptr;
00148     }
00149     // Image opening
00150     int w = image->width();
00151     int h = image->height();
00152
00153     // Getting corner pixels
00154     QColor colUL = image->pixelColor(0, 0).toRgb();
00155     QColor colUR = image->pixelColor(w - 1, 0).toRgb();
00156     QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();
00157
00158     // Getting verification code
00159     int verifCode = (((colUR.green() % 2) << 5) + colUR.blue() % 32) << 2;
00160     verifCode += colDR.blue() % 4;
00161     if(verifCode != 166){
00162         fail("veriffail");
00163         return nullptr;
00164     }
00165     // Getting number of bytes
00166     long long int countBytes = (colUL.blue() % 32 + ((colUL.green() % 32) << 5) + ((colUL.red() % 32) << 10
00167 )) << 9;
00168     countBytes += ((colUR.red() % 32) << 4) + (colUR.green() >> 1) % 16;
00169
00170     bitsUsed = (colDR.blue() >> 2) % 8 + 1;
00171     curMode = colDR.green() % 32;
00172
00173     // Start of the circuit
00174     QByteArray data;
00175     circuit(image, &data, countBytes);
00176
00177     // Check if circuit was successful
00178     if(!success)
00179         return nullptr;
00180     if(data.isEmpty())
00181     {
00182         fail("noreaddata");
00183         return nullptr;
00184     }
00185     // Version check
00186     long long int _ver = mod(data.at(0) * pow(2, 16));
00187     _ver += mod(data.at(1) * pow(2, 8));
00188     _ver += mod(data.at(2));
00189     data.remove(0, 3);
00190     if(_ver > version) {
00191         fail("Picture's app version is newer than yours. Image version is "
00192             + generateVersionString(_ver) + ", yours is "
00193             + generateVersionString(version) + ".");
00194         return nullptr;
00195     }
00196     else if(_ver < version) {
00197         fail("Picture's app version is older than yours. Image version is "
00198             + generateVersionString(_ver) + ", yours is "
00199             + generateVersionString(version) + ".");
00200         return nullptr;
00201     }

```



```

00202
00203 // Obtain the key
00204 int key_size = mod(data.at(0));
00205 QByteArray key = data.mid(1, key_size);
00206 data.remove(0, key_size + 1);
00207
00208 // Unzip
00209 QByteArray unzipped_data = unzip(data, key);
00210 emit saveData(unzipped_data);
00211 return unzipped_data;
00212 }
00217 void ModelPC::fail(QString message)
00218 {
00219     *error = message;
00220     alert(message, true);
00221     success = false;
00222     emit setProgress(101);
00223 }
00229 void ModelPC::jphs(QImage *image, QByteArray *data)
00230 {
00231     // Under Development
00232     return;
00233
00234     // Dead code
00235
00236     success = true;
00237     bool isEncrypt = !data->isEmpty();
00238     QString targetEXE = defaultJPHSDir + (isEncrypt ? "/jphide.exe" : "/jpseek.exe");
00239     if(!fileExists(targetEXE))
00240     {
00241         fail("nojphs");
00242         return;
00243     }
00244
00245     QString randomFileName = defaultJPHSDir + "/";
00246     qsrand(randSeed());
00247     for(int i = 0; i < 10; i++)
00248         randomFileName.append(97 + qrand() % 25);
00249     image->save(randomFileName + ".jpg");
00250     if(isEncrypt) {
00251         QFile file(randomFileName + ".pc");
00252         if(!file.open(QFile::WriteOnly)) {
00253             fail("savefilefail");
00254             return;
00255         }
00256         file.write(*data);
00257         file.close();
00258
00259         QStringList args;
00260         args << (randomFileName + ".jpg") << (randomFileName + "_out.jpg") << (randomFileName + ".pc");
00261         QProcess prog(this);
00262         prog.start(targetEXE, args);
00263         prog.waitForStarted();
00264         prog.write("test\n");
00265         prog.waitForBytesWritten();
00266         prog.write("test\n");
00267         prog.waitForBytesWritten();
00268         prog.waitForReadyRead();
00269         QByteArray bytes = prog.readAll();
00270         prog.waitForFinished();
00271         //QByteArray readData = prog.readAll();
00272         prog.close();
00273         // Cleaning - Deleting temp files
00274
00275     }
00276     else {
00277
00278     }
00279
00280 }
00281
00290 void ModelPC::circuit(QImage *image, QByteArray *data, long long countBytes)
00291 {
00292
00293     // Some flags and creation of the ProgressDialog
00294     success = true;
00295     emit setProgress(-1);
00296     bool isEncrypt = !data->isEmpty();
00297
00298     // Image setup
00299     int w = image->width();
00300     int h = image->height();
00301
00302     // Visited pixels array
00303     QVector<QPoint> were;
00304     were.push_back(QPoint(0, 0));
00305     were.push_back(QPoint(0, h - 1));

```

```

00306     were.push_back(QPoint(w - 1, 0));
00307     were.push_back(QPoint(w - 1, h - 1));
00308
00309     long long int offset = 0;
00310
00311     // Pre-start Cleaning
00312     circuitData = data;
00313     circuitImage = image;
00314     circuitCountBytes = countBytes;
00315     cur = 0;
00316     bitsBuffer.clear();
00317
00318     // Writing Top-Left to Bottom-Left
00319     for(int i = 1; i < h - 1 && mustGoOn(isEncrypt); i++) {
00320         QPoint pos(0, i);
00321         processPixel(pos, &were, isEncrypt);
00322     }
00323     // Writing Bottom-Right to Top-Right
00324     if(mustGoOn(isEncrypt))
00325     {
00326         for(int i = h - 2; i >= 1 && mustGoOn(isEncrypt); i--){
00327             QPoint pos(w - 1, i);
00328             processPixel(pos, &were, isEncrypt);
00329         }
00330     }
00331     // Main cycle
00332     // Strong is considered as actual corner pixel and weak as pixel near it like (1, 0) or (0, 1)
00333     while(mustGoOn(isEncrypt))
00334     {
00335         // Strong Top-Right to Strong Bottom-Right
00336         for(int i = offset; i < h - offset && mustGoOn(isEncrypt); i++){
00337             QPoint pos(w - offset - 2, i);
00338             processPixel(pos, &were, isEncrypt);
00339         }
00340         // Strong Top-Left to Weak Top-Right
00341         for(int i = offset + 1; i < w - offset - 2 && mustGoOn(isEncrypt); i++){
00342             QPoint pos(i, offset);
00343             processPixel(pos, &were, isEncrypt);
00344         }
00345         // Weak Bottom-Right to Weak Bottom-Left
00346         for(int i = w - 3 - offset; i >= offset + 2 && mustGoOn(isEncrypt); i--){
00347             QPoint pos(i, h - offset - 1);
00348             processPixel(pos, &were, isEncrypt);
00349         }
00350         // Weak Top-Left to Strong Bottom-Left
00351         for(int i = offset + 1; i < h - offset && mustGoOn(isEncrypt); i++){
00352             QPoint pos(offset + 1, i);
00353             processPixel(pos, &were, isEncrypt);
00354         }
00355         offset++;
00356     }
00357     // Extra writing
00358     if(!success)
00359         return;
00360     if(isEncrypt)
00361     {
00362         // Getting past colors
00363         QColor colUL = image->pixelColor(0, 0).toRgb();
00364         QColor colUR = image->pixelColor(w - 1, 0).toRgb();
00365         QColor colDL = image->pixelColor(0, h - 1).toRgb();
00366         QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();
00367         int red = 0;
00368         int green = 0;
00369         int blue = 0;
00370
00371         // Writing Upper Left
00372         red = (colUL.red() & 224) + (countBytes >> 19);
00373         green = (colUL.green() & 224) + (countBytes >> 14) % 32;
00374         blue = (colUL.blue() & 224) + (countBytes >> 9) % 32;
00375         image->setPixelColor(0, 0, QColor(red, green, blue));
00376
00377         // Writing Upper Right
00378         red = (colUR.red() & 224) + (countBytes >> 4) % 32;
00379         green = (colUR.green() & 224) + ((countBytes % 16) << 1) + 1;
00380         blue = (colUR.blue() & 224) + 9;
00381         image->setPixelColor(w - 1, 0, QColor(red, green, blue));
00382
00383         // Getting extra bytes if left
00384         while(cur < countBytes)
00385             push(mod(circuitData->at(cur++), 8));
00386         if(bitsBuffer.size() > 20) {
00387             fail("bitsBufferFail");
00388             return;
00389         }
00390         // Getting extra data as long.
00391         long extraData = pop(-2);
00392

```

```

00393         // Writing Down Left
00394         red = (colDL.red() & 224) + (extraData >> 15);
00395         green = (colDL.green() & 224) + (extraData >> 10) % 32;
00396         blue = (colDL.blue() & 224) + (extraData >> 5) % 32;
00397         image->setPixelColor(0, h - 1, QColor(red, green, blue));
00398
00399         // Writing Down Right
00400         red = (colDR.red() & 224) + extraData % 32;
00401         green = (colDR.green() & 224);
00402         blue = (colDR.blue() & 224) + ((bitsUsed - 1) << 2) + 2;
00403         image->setPixelColor(w - 1, h - 1, QColor(red, green, blue));
00404     }
00405     else
00406     {
00407         // Read the past pixels
00408         QColor colDL = image->pixelColor(0, h - 1).toRgb();
00409         QColor colDR = image->pixelColor(w - 1, h - 1).toRgb();
00410
00411         // Read extra data
00412         long extraData = ((colDL.red() % 32) << 15) + ((colDL.green() % 32) << 10);
00413         extraData += ((colDL.blue() % 32) << 5) + colDR.red() % 32;
00414
00415         // Add extra data to the bitsBuffer
00416         push(extraData, (countBytes - cur) * 8 - bitsBuffer.size());
00417
00418         // Move bits from bitsBuffer to the QByteArray
00419         while(!bitsBuffer.isEmpty())
00420             data->append(pop(8));
00421     }
00422     emit setProgress(101);
00423 }
00424
00432 void ModelPC::processPixel(QPoint pos, QVector<QPoint> *were, bool isEncrypt)
00433 {
00434     if(!success)
00435         return;
00436     // Check if point was already visited
00437     if(were->contains(pos)){
00438         fail("Point (" + QString::number(pos.x()) + "," + QString::number(pos.y()) + ") was visited
twice! Error code 2");
00439         return;
00440     }
00441     else
00442         were->push_back(pos);
00443     if(isEncrypt)
00444     {
00445         // Make sure that there are enough bits in bitsBuffer to write
00446         while(bitsBuffer.size() < 3 * bitsUsed)
00447             push(mod(circuitData->at(cur++), 8));
00448         // Read past contains
00449         QColor pixelColor = circuitImage->pixelColor(pos);
00450         int red = pixelColor.red();
00451         int green = pixelColor.green();
00452         int blue = pixelColor.blue();
00453
00454         // Write new data in last bitsUsed pixels
00455         red += pop() - red % (int) pow(2, bitsUsed);
00456         green += pop() - green % (int) pow(2, bitsUsed);
00457         blue += pop() - blue % (int) pow(2, bitsUsed);
00458
00459         circuitImage->setPixelColor(pos, QColor(red, green, blue));
00460     }
00461     else
00462     {
00463         QColor read_color = circuitImage->pixelColor(pos).toRgb();
00464         // Reading the pixel
00465         int red = read_color.red();
00466         int green = read_color.green();
00467         int blue = read_color.blue();
00468
00469         // Reading the last bitsUsed pixels
00470         red %= (int) pow(2, bitsUsed);
00471         green %= (int) pow(2, bitsUsed);
00472         blue %= (int) pow(2, bitsUsed);
00473
00474         // Getting the data in the bitsBuffer.
00475         push(red);
00476         push(green);
00477         push(blue);
00478
00479         // Getting data to QByteArray
00480         while(bitsBuffer.size() >= 8) {
00481             circuitData->append(pop(8));
00482             cur++;
00483         }
00484     }
00485     emit setProgress(100 * cur / circuitCountBytes);

```

```

00486 }
00487
00488 long ModelPC::pop(int bits)
00489 {
00490     // Hard to say
00491     long res = 0;
00492     int poppedBits = bits == -1 ? bitsUsed : bits;
00493     if(bits == -2)
00494         poppedBits = bitsBuffer.size();
00495     for(int i = 0; i < poppedBits; i++)
00496         res += bitsBuffer[i] * pow(2, poppedBits - i - 1);
00497     bitsBuffer.remove(0, poppedBits);
00498     return res;
00499 }
00500
00501 void ModelPC::push(int data, int bits)
00502 {
00503     // That's easier, but also hard
00504     int buf_size = bitsBuffer.size();
00505     int extraSize = bits == -1 ? bitsUsed : bits;
00506     bitsBuffer.resize(buf_size + extraSize);
00507     for(int i = bitsBuffer.size() - 1; i >= buf_size; i--, data >>= 1)
00508         bitsBuffer[i] = data % 2;
00509 }
00510
00511 bool ModelPC::mustGoOn(bool isEncrypt)
00512 {
00513     return success && (isEncrypt ? (circuitCountBytes - cur) * 8 + bitsBuffer.size() >=
00514         bitsUsed * 3 :
00515         circuitData->size() * 8 + bitsBuffer.size() <
00516         circuitCountBytes * 8 - (circuitCountBytes * 8) % (
00517             bitsUsed * 3));
00518 }
00519
00520 QByteArray ModelPC::unzip(QByteArray data, QByteArray key)
00521 {
00522     // Decryption
00523     QByteArray hashKey = QCryptographicHash::hash(key, QCryptographicHash::Sha256);
00524     QAESEncryption encryption(QAESEncryption::AES_256,
00525         QAESEncryption::ECB);
00526     QByteArray new_data = encryption.decode(data, hashKey);
00527     // Decompressing
00528     return qUncompress(new_data);
00529 }
00530
00531 QByteArray ModelPC::zip(QByteArray data, QByteArray key)
00532 {
00533     // Zip
00534     QByteArray c_data = qCompress(data, 9);
00535     // Encryption
00536     QByteArray hashKey = QCryptographicHash::hash(key, QCryptographicHash::Sha256);
00537     return QAESEncryption::Crypt(QAESEncryption::AES_256,
00538         QAESEncryption::ECB, c_data, hashKey);
00539 }
00540
00541 bool ModelPC::fileExists(QString path)
00542 {
00543     QFileInfo check_file(path);
00544     return check_file.exists() && check_file.isFile();
00545 }
00546
00547 QByteArray ModelPC::bytes(long long n)
00548 {
00549     return QByteArray::fromHex(QByteArray::number(n, 16));
00550 }
00551
00552 unsigned int ModelPC::mod(int input)
00553 {
00554     if(input < 0)
00555         return (unsigned int) (256 + input);
00556     else
00557         return (unsigned int) input;
00558 }
00559
00560 void ModelPC::alert(QString message, bool isWarning)
00561 {
00562     emit alertView(message, isWarning);
00563 }
00564
00565 QColor ModelPC::RGBbytes(long long byte)
00566 {
00567     int blue = byte % 256;
00568     int green = (byte / 256) % 256;
00569     int red = byte / pow(2, 16);
00570     return QColor(red, green, blue);
00571 }
00572
00573 QString ModelPC::generateVersionString(long ver)
00574 {
00575     return QString::number((int) (ver / pow(2, 16))) + "." + QString::number(((int) (ver / 256)) % 256) + "
00576         ." + QString::number(ver % 256);
00577 }

```

```

00607
00608 uint ModelPC::randSeed()
00609 {
00610     QTime time = QTime::currentTime();
00611     uint randSeed = time.msecsSinceStartOfDay() % 65536 + time.minute() * 21 + time.second() * 2;
00612     return randSeed;
00613 }

```

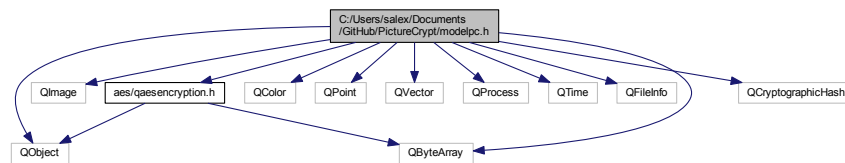
10.27 C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.h File Reference

```

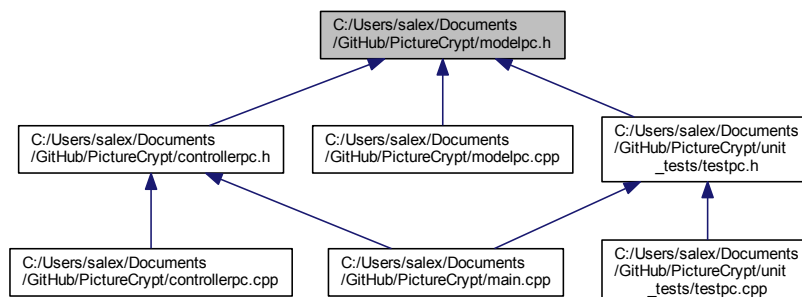
#include <QObject>
#include <QImage>
#include <QByteArray>
#include <QColor>
#include <QPoint>
#include <QVector>
#include <QProcess>
#include <QTime>
#include <QFileInfo>
#include <aes/qaesencryption.h>
#include <QCryptographicHash>

```

Include dependency graph for modelpc.h:



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



Classes

- class [ModelPC](#)

The *ModelPC* class Model Layer of the app. Controlled 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

```

00001 #ifndef MODELPC_H
00002 #define MODELPC_H
00003
00004 #include <QObject>
00005 #include <QImage>
00006 #include <QByteArray>
00007 #include <QColor>
00008 #include <QPoint>
00009 #include <QVector>
00010 #include <QProcess>
00011 #include <QTime>
00012 #include <QFileInfo>
00013
00014 #include <aes/qaesencryption.h>
00015 #include <QCryptographicHash>
00016
00027 class ModelPC : public QObject
00028 {
00029     Q_OBJECT
00030 public:
00031     ModelPC();
00032
00033 signals:
00040     alertView(QString messageCode, bool isWarning);
00045     saveData(QByteArray data);
00050     saveImage(QImage *image);
00056     setProgress(int val);
00057
00058 public slots:
00059     QImage *start(QByteArray data, QImage *image, int mode = 0, QString
key = "", int _bitsUsed = 8, QString *_error = nullptr);
00060     QImage *encrypt(QByteArray encr_data, QImage * image, int mode = 0, QString *_error =
nullptr);
00061     QByteArray decrypt(QImage * image, QString *_error = nullptr);
00062     void fail(QString message);
00063
00064 public:
00069     bool success;
00073     long version;
00077     QString versionString;
00081     int curMode;
00085     int bitsUsed;
00089     QString defaultJPHSDir;
00093     QString * error;
00094     QByteArray unzip(QByteArray data, QByteArray key);
00095     void alert(QString message, bool isWarning = false);
00096 protected:
00097     void circuit(QImage * image, QByteArray * data, long long int countBytes);
00098     void jphs(QImage * image, QByteArray * data);
00099     void processPixel(QPoint pos, QVector<QPoint> *were, bool isEncrypt);
00100     QByteArray zip(QByteArray data, QByteArray key);
00101 private:
00102     bool fileExists(QString path);
00103     QByteArray bytes(long long n);
00104     unsigned int mod(int input);
00105     QByteArray ver_byte;
00106     QColor RGBbytes(long long byte);
00107     QString generateVersionString(long ver);
00108     uint randSeed();
00109
00110     QByteArray * circuitData;
00111     QImage * circuitImage;
00112     long long circuitCountBytes;
00113     long cur;

```

```

00114     bool mustGoOn(bool isEncrypt);
00115
00116     QVector <bool> bitsBuffer;
00117     long pop(int bits = -1);
00118     void push(int data, int bits = -1);
00119
00120     void setError(QString word);
00121 };
00122
00123 #endif // MODELPC_H

```

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

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

```

00001 # PictureCrypt
00002 Make your pictures crypted.
00003
00004
00005 ## About
00006 Project is made only using QT.
00007 [QAESEncryption](http://github.com/bricke/Qt-AES) by bricke was also used.
00008 MVC pattern used.
00009 PictureCrypt project is UI based, the model contains all buisness logic and can work as standalone
      class.
00010
00011 ## External use
00012 ModelPC class can be used externally (without UI)
00013 ```
00014 #include <modelpc.h>
00015 #include <QByteArray>
00016 #include <QImage>
00017
00018 ...
00019
00020 ModelPC * model = new ModelPC(ver);
00021 // ver is version of the app, used to check the data structure version
00022 // ver is type long and is calculated as if version is "x.y.z" => ver = x * 65536 + y * 256 + z
00023 // Default parameter is 2^17 (2.0.0)
00024
00025 // Connecting signals
00026
00027 // Essential ones
00028
00029 model->saveData(QByteArray data)
00030 // Used to return the retrieved data
00031
00032 model->saveImage(QImage * image)
00033 // Used to return the modified image
00034
00035 // Extra ones
00036
00037 model->alertView(QString message, bool isWarning)
00038 // Used for messages to be shown to users
00039
00040 model->setProgress(int val)
00041 // Used to show user the progress of embedding
00042 // -1 indicates the creation of some kind of progress dialog
00043 // from 0 to 100 shows the progress
00044 // 101 indicates that progress dialog should be closed
00045
00046 ```
00047
00048 ## Avaible methods
00049 ### Essential ones
00050 ##### start
00051 Used for embedding
00052
00053 Parameters:
00054 data    Data to be encrypted
00055 _image  Image to be encrypted into.
00056 _bitsUsed Bits per byte, see also ModelPC::bitsUsed
00057 key     Key, if default (empty), random key of 64 charachters will be generated.
00058 mode    Mode of encryption
00059 ```
00060 model->start(QByteArray data, QImage image, int mode = 0, QString key = "", int _bitsUsed = 8);
00061 ```
00062
00063 ##### decrypt

```



```

00006 { }
00018 bool TestPC::test(QByteArray data, QImage rImage, QString expectedOutput, int
mode, QString key, int bitsUsed)
00019 {
00020     // Error outputs
00021     QString error1, error2;
00022     // Embedding
00023     QImage * retImage = model->start(data, &rImage, mode, key,
bitsUsed, &error1);
00024     // De-embedding
00025     QByteArray output = model->decrypt(retImage, &error2);
00026
00027     // Success of error outputs
00028     bool er1 = error1 == expectedOutput;
00029     bool er2 = error2 == expectedOutput;
00030     if(expectedOutput == "ok")
00031         return er1 && er2 && data == output;
00032     else
00033         return er1 || er2;
00034 }
00042 int TestPC::startTest()
00043 {
00044     qDebug() << "Testing started...\n";
00045     model = new ModelPC();
00046
00047     // Long text open
00048     QFile file(":/unit_tests/longtext.txt");
00049     if(!file.open(QFile::ReadOnly))
00050         return false;
00051     text = file.readAll();
00052     file.close();
00053
00054     // Big picture open
00055     image = QImage(":/unit_tests/bigpicture.jpg");
00056     if(image.isNull())
00057         return false;
00058
00059     // JSON tests list open
00060     QFile json_file(":/unit_tests/tests.json");
00061     QJsonDocument doc;
00062     if(!json_file.open(QFile::ReadOnly | QFile::Text))
00063         return false;
00064     QByteArray readData = json_file.readAll();
00065     json_file.close();
00066     doc = QJsonDocument::fromJson(readData);
00067     // Testing
00068     return autoTest(doc);
00069 }
00077 bool TestPC::autoTest(QJsonDocument doc)
00078 {
00079     // Opening the tests array
00080     QJsonObject o = doc.object();
00081     QJsonArray arr = o["tests"].toArray();
00082     int sum = 0;
00083
00084     // Info about tests
00085     QString extraText;
00086     for(int i = 0; i < arr.size(); i++) {
00087         // Reading the data
00088         QJsonObject obj = arr[i].toObject();
00089
00090         QString t = obj["data"].toString();
00091         if(t == "/text/")
00092             t = text;
00093         QByteArray data = t.toUtf8();
00094
00095         QString im = obj["image"].toString();
00096         QImage img(":/unit_tests/" + im);
00097
00098         QString expect = obj["expectation"].toString();
00099
00100         int mode = obj["mode"].toInt();
00101
00102         QString key = obj["key"].toString();
00103
00104         int bitsUsed = obj["bitsUsed"].toInt();
00105
00106         // Testing
00107         bool s = test(data, img, expect, mode, key,
bitsUsed);
00108
00109         sum += s;
00110         extraText += "\n * Test #" + QString::number(i + 1) + " " + (s ? "completed." : "failed.");
00111     }
00112     // Writing log
00113     QFile file("tests.log");
00114     bool testsSuc = sum == arr.size();

```

```

00115     if(!file.open(QFile::WriteOnly | QFile::Text))
00116         return testsSuc;
00117     QDateTime curTime = QDateTime::currentDateTime();
00118     QString date = curTime.toString("dd.MM.yyyy HH:mm");
00119     QString logtext = "#####\n"
00120         "####Log file created at " + date + "####\n"
00121         "#####\n\n"
00122         "Status: " + (testsSuc ? "All tests completed" : "Tests failed") + "\n\n"
00123         "Tests list:\n";
00124     logtext += extraText;
00125     file.write(logtext.toUtf8());
00126     file.close();
00127     // Cleaning up
00128     qDebug() << "Testing completed\n";
00129     delete model;
00130     return testsSuc;
00131 }

```

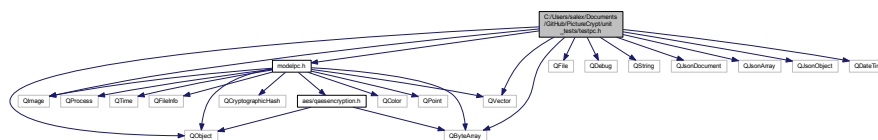
10.33 C:/Users/salex/Documents/GitHub/PictureCrypt/unit_tests/testpc.h File Reference

```

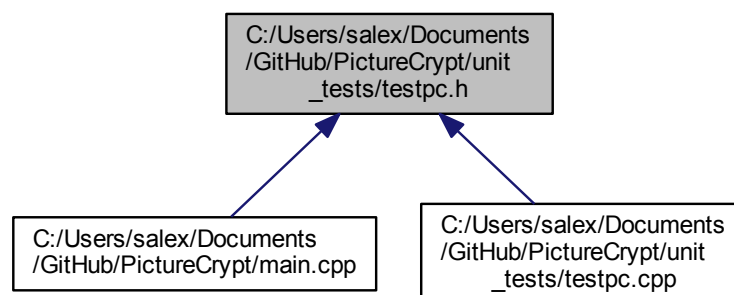
#include <QObject>
#include <modelpc.h>
#include <QFile>
#include <QDebug>
#include <QString>
#include <QImage>
#include <QByteArray>
#include <QVector>
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonObject>
#include <QDateTime>

```

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

```

00001 #ifndef TESTPC_H
00002 #define TESTPC_H
00003
00004 #include <QObject>
00005 #include <modelpc.h>
00006
00007 #include <QFile>
00008 #include <QDebug>
00009 #include <QString>
00010 #include <QImage>
00011 #include <QByteArray>
00012 #include <QVector>
00013
00014 #include <QJsonDocument>
00015 #include <QJsonArray>
00016 #include <QJsonObject>
00017
00018 #include <QDateTime>
00023 class TestPC : public QObject
00024 {
00025     Q_OBJECT
00026 public:
00027     TestPC();
00028 public slots:
00029     int startTest();
00030 protected slots:
00031     bool test(QByteArray data, QImage rImage,
00032              QString expectedOutput = "ok", int mode = 0,
00033              QString key = "", int bitsUsed = 8);
00034 private:
00038     ModelPC * model;
00042     QByteArray text;
00046     QImage image;
00047
00048     bool autoTest(QJsonDocument doc);
00049 };
00050
00051 #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

```

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

```

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

10.38 tests.json

```

00001 {
00002     "tests": [
00003         {
00004             "data": "/text/",
00005             "image": "bigpicture.jpg",
00006             "expectation": "ok",
00007             "mode": 0,
00008             "key": "",
00009             "bitsUsed": 8
00010         },
00011         {
00012             "data": "/text/",
00013             "image": "bigpicture.jpg",
00014             "expectation": "ok",
00015             "mode": 0,
00016             "key": "",
00017             "bitsUsed": 7
00018         },
00019         {
00020             "data": "/text/",
00021             "image": "bigpicture.jpg",
00022             "expectation": "ok",
00023             "mode": 0,
00024             "key": "",

```

```

00025         "bitsUsed": 1
00026     },
00027     {
00028         "data": "/text/",
00029         "image": "tinypicture.png",
00030         "expectation": "muchdata",
00031         "mode": 0,
00032         "key": "",
00033         "bitsUsed": 8
00034     },
00035     {
00036         "data": "",
00037         "image": "bigpicture.jpg",
00038         "expectation": "nodata",
00039         "mode": 0,
00040         "key": "",
00041         "bitsUsed": 8
00042     },
00043     {
00044         "data": "/text/",
00045         "image": "invalid.jpg",
00046         "expectation": "nullimage",
00047         "mode": 0,
00048         "key": "",
00049         "bitsUsed": 8
00050     },
00051     {
00052         "data": "/text/",
00053         "image": "bigpicture.jpg",
00054         "expectation": "bitsWrong",
00055         "mode": 0,
00056         "key": "",
00057         "bitsUsed": 12
00058     }
00059 ]
00060 }

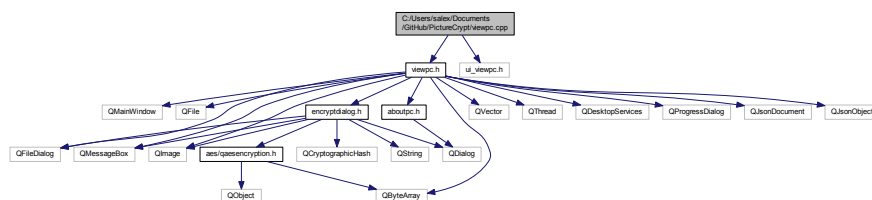
```

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

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

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

Include dependency graph for viewpc.cpp:



10.40 viewpc.cpp

```

00001 #include "viewpc.h"
00002 #include "ui_viewpc.h"
00003
00004 ViewPC::ViewPC(QWidget *parent) :
00005     QMainWindow(parent),
00006     ui(new Ui::ViewPC)
00007 {
00008     ui->setupUi(this);
00009
00010     progressDialogClosed = true;
00011
00012     // Alerts dictionary setup
00013     // TODO Add relative path
00014     QFile file(":/config/ErrorsDict.json");
00015     if(!file.open(QFile::ReadOnly | QFile::Text)) {

```

```

00016         alert("Cannot open config file!");
00017         return;
00018     }
00019     QByteArray readData = file.readAll();
00020     file.close();
00021
00022     QJsonParseError error;
00023     QJsonDocument doc = QJsonDocument::fromJson(readData, &error);
00024     errorsDict = doc.object();
00025 }
00026
00027 ViewPC::~ViewPC()
00028 {
00029     delete ui;
00030 }
00031
00032 void ViewPC::on_encryptMode_clicked()
00033 {
00034     // Encrypt radio button clicked
00035     setEncryptMode(true);
00036 }
00037
00038 void ViewPC::on_decryptMode_clicked()
00039 {
00040     // Decrypt radio button clicked
00041     setEncryptMode(false);
00042 }
00043
00044 void ViewPC::on_fileButton_clicked()
00045 {
00046     // Opening QFileDialog depending on isEncrypt
00047     if(isEncrypt)
00048         inputFileName = QFileDialog::getOpenFileName(this, tr("Select file"), "/untitled.txt", tr("Text
00049 files (*.txt);;All Files (*)"));
00050     else
00051         inputFileName = QFileDialog::getOpenFileName(this, tr("Select file"), "/untitled.png", tr("PNG
00052 files (*.png);;All Files (*)"));
00053     // Display the file name
00054     ui->fileLabel->setText(inputFileName.isEmpty() ? "File not chosen" : inputFileName);
00055 }
00056
00057 void ViewPC::on_startButton_clicked()
00058 {
00059     if(isEncrypt)
00060     {
00061         // Getting the data
00062         QString text = ui->text->toPlainText();
00063         QByteArray data;
00064         if(text.isEmpty()) {
00065             if(inputFileName.isEmpty()) {
00066                 alert("No input file or text was not given. Cannot continue!", true);
00067                 return;
00068             }
00069             // Opening the file
00070             QFile file(inputFileName);
00071             if (!file.open(QIODevice::ReadOnly))
00072             {
00073                 alert("Cannot open file. Cannot continue!", true);
00074                 return;
00075             }
00076             // Check the data size
00077             auto size = file.size();
00078             if(size > pow(2, 24)) {
00079                 alert("Your file is too big, our systems can handle it, but it requires a lot of time.
00080 We decline.", true);
00081                 file.close();
00082                 return;
00083             }
00084             data = file.readAll();
00085             file.close();
00086         }
00087         else
00088             data = text.toUtf8();
00089         // Select image via EncryptDialog
00090         EncryptDialog * dialog = new EncryptDialog(
00091 data);
00092         dialog->exec();
00093         if(!dialog->success)
00094             return;
00095
00096         // Get the data
00097         QByteArray encr_data = dialog->compr_data;
00098
00099         // Save the key
00100         QByteArray key_data = dialog->key.toUtf8();
00101
00102         encr_data = bytes(key_data.size()) + key_data + encr_data;
00103         // TODO do the mode thing
00104         emit setBitsUsed(dialog->bitsUsed);
00105     }
00106 }

```

```

00114         emit encrypt(encr_data, &dialog->image, 0);
00115     }
00116     else
00117     {
00118         // Get the filename of the image
00119         if(!ui->text->toPlainText().isEmpty())
00120             alert("Obviously, the text browser isn't supported for decryption, use File Dialog
instead.");
00121         if(inputFileName.isEmpty()) {
00122             alert("File not selected. Cannot continue!", true);
00123             return;
00124         }
00125         QImage * res_image = new QImage(inputFileName);
00126         emit decrypt(res_image);
00127     }
00128 }
00134 void ViewPC::alert(QString message, bool isWarning)
00135 {
00136     // Get message
00137     if(errorsDict.contains(message))
00138         message = errorsDict[message].toString();
00139     // Create message box
00140     QMessageBox box;
00141     if(isWarning)
00142         box.setIcon(QMessageBox::Warning);
00143     else
00144         box.setIcon(QMessageBox::Information);
00145     box.setText(message);
00146     box.setWindowIcon(QIcon(":/icons/mail.png"));
00147     box.setWindowTitle("Message");
00148     box.exec();
00149 }
00155 void ViewPC::saveData(QByteArray Edata)
00156 {
00157     // Save data using QFileDialog
00158     QString outputFileName = QFileDialog::getSaveFileName(this, tr("Save File"),
00159                                                         "/untitled.txt",
00160                                                         tr("Text(*.txt);;All files (*)"));
00161     QFile writeFile(outputFileName);
00162     if (!writeFile.open(QIODevice::WriteOnly))
00163     {
00164         alert("Cannot access file path. Cannot continue!", true);
00165         return;
00166     }
00167     writeFile.write(Edata);
00168     writeFile.close();
00169     alert("Decryption completed!");
00170 }
00176 void ViewPC::saveImage(QImage * image)
00177 {
00178     // Save image using QFileDialog
00179     QString outputFileName = QFileDialog::getSaveFileName(this, tr("Save Image"),
00180                                                         "/untitled.png",
00181                                                         tr("Images(*.png)"));
00182     if(!image->save(outputFileName)) {
00183         alert("Cannot save file. Unable to continue!", true);
00184         return;
00185     }
00186     alert("Encryption completed!");
00187 }
00194 void ViewPC::setProgress(int val)
00195 {
00196     if(val < 0) {
00197         // Create dialog
00198         dialog = new QProgressDialog("Cryption in progress.", "Cancel", 0, 100);
00199         connect(dialog, SIGNAL(canceled()), this, SLOT(abortCircuit()));
00200         progressDialogClosed = false;
00201         dialog->setWindowTitle("Processing");
00202         dialog->setWindowIcon(QIcon(":/icons/loading.png"));
00203         dialog->show();
00204     }
00205     else if(val >= 100 && !progressDialogClosed) {
00206         // Close dialog
00207         dialog->setValue(100);
00208         QThread::msleep(25);
00209         dialog->close();
00210         dialog->reset();
00211         progressDialogClosed = true;
00212     }
00213     // Update the progress
00214     else if(!progressDialogClosed)
00215         dialog->setValue(val);
00216 }
00220 void ViewPC::abortCircuit()
00221 {
00222     // Set the flag
00223     progressDialogClosed = true;

```

```

00224     // Close the dialog
00225     dialog->close();
00226     dialog->reset();
00227     emit abortModel();
00228 }
00233 void ViewPC::setEncryptMode(bool encr)
00234 {
00235     ui->text->setEnabled(encr);
00236     isEncrypt = encr;
00237 }
00242 void ViewPC::setVersion(QString version)
00243 {
00244     // Version setup
00245     versionString = version;
00246 }
00247
00248 QByteArray ViewPC::bytes(long long n)
00249 {
00250     return QByteArray::fromHex(QByteArray::number(n, 16));
00251 }
00255 void ViewPC::on_actionAbout_triggered()
00256 {
00257     AboutPC about;
00258     about.setVersion(versionString);
00259     about.exec();
00260 }
00261
00265 void ViewPC::on_actionHelp_triggered()
00266 {
00267     QUrl docLink("http://doc.alex.unaux.com/picturecrypt");
00268     QDesktopServices::openUrl(docLink);
00269 }
00270
00271 void ViewPC::on_actionJPHS_path_triggered()
00272 {
00273     QString dir = QFileDialog::getExistingDirectory(this, tr("Open JPHS folder"),
00274                                                    "/home",
00275                                                    QFileDialog::ShowDirsOnly
00276                                                    | QFileDialog::DontResolveSymlinks);
00277     emit setJPHSDir(dir);
00278 }

```

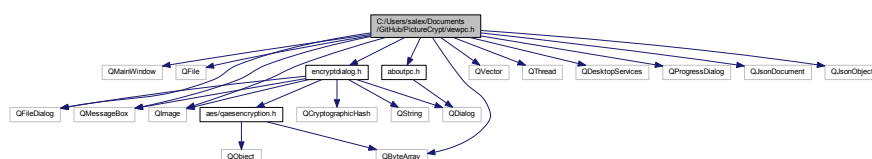
10.41 C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.h File Reference

```

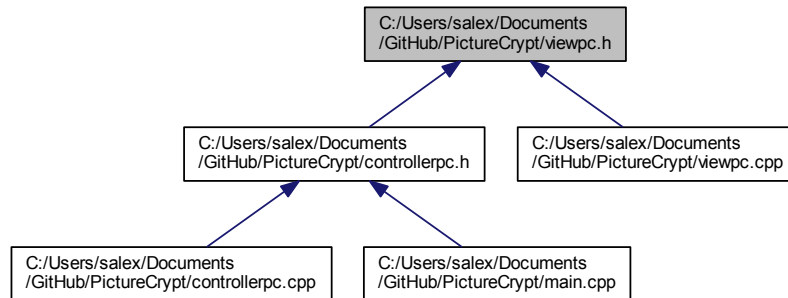
#include <QMainWindow>
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include <QImage>
#include <QByteArray>
#include <QVector>
#include <QThread>
#include <QDesktopServices>
#include <encryptdialog.h>
#include <QProgressDialog>
#include <aboutpc.h>
#include <QJsonDocument>
#include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

```

Include dependency graph for viewpc.h:



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



Classes

- class [ViewPC](#)

The [ViewPC](#) class View layer of the app. Controls [EncryptDialog](#) and [ProgressDialog](#).

Namespaces

- [Ui](#)

10.41.1 Detailed Description

Header of [ViewPC](#) class

See also

[ControllerPC](#), [ModelPC](#), [ViewPC](#)

Definition in file [viewpc.h](#).

10.42 viewpc.h

```

00001 #ifndef VIEWPC_H
00002 #define VIEWPC_H
00003
00004 #include <QMainWindow>
00005 #include <QFile>
00006 #include <QFileDialog>
00007 #include <QMessageBox>
00008 #include <QImage>
00009 #include <QByteArray>
00010 #include <QVector>
00011 #include <QThread>
00012 #include <QDesktopServices>
00013
00014 #include <encryptdialog.h>
00015 #include <QProgressDialog>
00016 #include <aboutpc.h>
00017

```

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

Index

- ~AboutPC
 - AboutPC, [26](#)
- ~EncryptDialog
 - EncryptDialog, [33](#)
- ~ViewPC
 - ViewPC, [68](#)
- abortCircuit
 - ControllerPC, [29](#)
 - ViewPC, [68](#)
- abortModel
 - ViewPC, [68](#)
- AboutPC, [25](#)
 - ~AboutPC, [26](#)
 - AboutPC, [26](#)
 - setVersion, [26](#)
- Aes
 - QAESEncryption, [54](#)
- alert
 - ModelPC, [40](#)
 - ViewPC, [69](#)
- alertView
 - ModelPC, [41](#)
- arr
 - tests-setup, [21](#)
- bitsUsed
 - EncryptDialog, [35](#)
 - ModelPC, [51](#)
 - tests-setup, [21](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/COD↔
E_OF_CONDUCT.md, [92](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/REA↔
DME.md, [109](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.↔
cpp, [79](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/aboutpc.↔
h, [80](#), [81](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencrypt.↔
cpp, [81](#), [82](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/aes/qaesencrypt.↔
h, [89](#), [90](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/config/↔
ErrorsDict.json, [92](#), [93](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/config/↔
ErrorsDictSetup.py, [93](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.↔
cpp, [94](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/controllerpc.↔
h, [94](#), [95](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.↔
cpp, [96](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/encryptdialog.↔
h, [97](#), [99](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/main.↔
cpp, [99](#), [100](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.↔
cpp, [100](#), [101](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/modelpc.↔
h, [107](#), [108](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/unit_↔
tests/testpc.cpp, [110](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/unit_↔
tests/testpc.h, [112](#), [113](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/unit_↔
tests/tests-setup.py, [113](#), [114](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/unit_↔
tests/tests.json, [114](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.↔
cpp, [115](#)
- C:/Users/salex/Documents/GitHub/PictureCrypt/viewpc.↔
h, [118](#), [119](#)
- circuit
 - ModelPC, [41](#)
- compr_data
 - EncryptDialog, [35](#)
- ControllerPC, [27](#)
 - abortCircuit, [29](#)
 - ControllerPC, [28](#)
 - setBitsUsed, [29](#)
 - setJPHSDir, [30](#)
 - version, [30](#)
 - versionString, [31](#)
- Crypt
 - QAESEncryption, [56](#)
- curMode
 - ModelPC, [51](#)
- data↔
EncryptDialog, [36](#)
- ErrorsDictSetup, [19](#)
 - tests-setup, [21](#)
- decode
 - QAESEncryption, [57](#)
- Decrypt
 - QAESEncryption, [58](#)
- decrypt
 - ModelPC, [42](#)

- ViewPC, 70
- defaultJPHSDir
 - ModelPC, 51
- dialog
 - ViewPC, 76
- encode
 - QAESEncryption, 59
- encrypt
 - ModelPC, 43
 - ViewPC, 70
- EncryptDialog, 31
 - ~EncryptDialog, 33
 - bitsUsed, 35
 - compr_data, 35
 - data, 36
 - EncryptDialog, 33
 - goodPercentage, 36
 - image, 36
 - inputFileName, 36
 - key, 36
 - on_buttonBox_accepted, 33
 - on_buttonBox_rejected, 33
 - on_fileButton_clicked, 34
 - on_horizontalSlider_valueChanged, 34
 - size, 37
 - success, 37
 - val, 37
 - zip, 34
- error
 - ModelPC, 52
- errorsDict
 - ViewPC, 76
- ErrorsDictSetup, 19
 - data, 19
 - f, 19
 - filename, 19
 - indent, 20
 - input_data, 20
 - key, 20
 - raw, 20
 - value, 20
- ExpandKey
 - QAESEncryption, 60
- expandKey
 - QAESEncryption, 60
- expect
 - tests-setup, 21
- f
 - ErrorsDictSetup, 19
 - tests-setup, 22
- fail
 - ModelPC, 44
- filename
 - ErrorsDictSetup, 19
 - tests-setup, 22
- goodPercentage
- EncryptDialog, 36
- image
 - EncryptDialog, 36
 - tests-setup, 22
- indent
 - ErrorsDictSetup, 20
 - tests-setup, 22
- input_data
 - ErrorsDictSetup, 20
 - tests-setup, 22
- inputFileName
 - EncryptDialog, 36
- jphs
 - ModelPC, 45
- js
 - tests-setup, 22
- key
 - EncryptDialog, 36
 - ErrorsDictSetup, 20
 - tests-setup, 23
- main
 - main.cpp, 100
- main.cpp
 - main, 100
- Mode
 - QAESEncryption, 55
- mode
 - tests-setup, 23
- ModelPC, 38
 - alert, 40
 - alertView, 41
 - bitsUsed, 51
 - circuit, 41
 - curMode, 51
 - decrypt, 42
 - defaultJPHSDir, 51
 - encrypt, 43
 - error, 52
 - fail, 44
 - jphs, 45
 - ModelPC, 40
 - processPixel, 46
 - saveData, 46
 - saveImage, 47
 - setProgress, 47
 - start, 48
 - success, 52
 - unzip, 49
 - version, 52
 - versionString, 52
 - zip, 50
- multiply
 - qaesencryption.cpp, 81
- obj

- tests-setup, [23](#)
- on_actionAbout_triggered
 - ViewPC, [71](#)
- on_actionHelp_triggered
 - ViewPC, [71](#)
- on_buttonBox_accepted
 - EncryptDialog, [33](#)
- on_buttonBox_rejected
 - EncryptDialog, [33](#)
- on_fileButton_clicked
 - EncryptDialog, [34](#)
 - ViewPC, [72](#)
- on_horizontalSlider_valueChanged
 - EncryptDialog, [34](#)
- on_startButton_clicked
 - ViewPC, [72](#)
- Padding
 - QAESEncryption, [55](#)
- processPixel
 - ModelIPC, [46](#)
- progressDialogClosed
 - ViewPC, [76](#)
- QAESEncryption, [53](#)
 - Aes, [54](#)
 - Crypt, [56](#)
 - decode, [57](#)
 - Decrypt, [58](#)
 - encode, [59](#)
 - ExpandKey, [60](#)
 - expandKey, [60](#)
 - Mode, [55](#)
 - Padding, [55](#)
 - QAESEncryption, [55](#)
 - RemovePadding, [61](#)
 - removePadding, [62](#)
- qaesencryption.cpp
 - multiply, [81](#)
 - xTime, [82](#)
- raw
 - ErrorsDictSetup, [20](#)
 - tests-setup, [23](#)
- RemovePadding
 - QAESEncryption, [61](#)
- removePadding
 - QAESEncryption, [62](#)
- saveData
 - ModelIPC, [46](#)
 - ViewPC, [72](#)
- savelImage
 - ModelIPC, [47](#)
 - ViewPC, [73](#)
- sep
 - tests-setup, [23](#)
- setBitsUsed
 - ControllerPC, [29](#)
- ViewPC, [74](#)
- setEncryptMode
 - ViewPC, [74](#)
- setJPHSDir
 - ControllerPC, [30](#)
 - ViewPC, [75](#)
- setProgress
 - ModelIPC, [47](#)
 - ViewPC, [75](#)
- setVersion
 - AboutPC, [26](#)
 - ViewPC, [75](#)
- size
 - EncryptDialog, [37](#)
- start
 - ModelIPC, [48](#)
- startTest
 - TestPC, [64](#)
- success
 - EncryptDialog, [37](#)
 - ModelIPC, [52](#)
- test
 - TestPC, [64](#)
- TestPC, [62](#)
 - startTest, [64](#)
 - test, [64](#)
 - TestPC, [64](#)
- tests-setup, [21](#)
 - arr, [21](#)
 - bitsUsed, [21](#)
 - data, [21](#)
 - expect, [21](#)
 - f, [22](#)
 - filename, [22](#)
 - image, [22](#)
 - indent, [22](#)
 - input_data, [22](#)
 - js, [22](#)
 - key, [23](#)
 - mode, [23](#)
 - obj, [23](#)
 - raw, [23](#)
 - sep, [23](#)
- Ui, [23](#)
- unzip
 - ModelIPC, [49](#)
- val
 - EncryptDialog, [37](#)
- value
 - ErrorsDictSetup, [20](#)
- version
 - ControllerPC, [30](#)
 - ModelIPC, [52](#)
- versionString
 - ControllerPC, [31](#)
 - ModelIPC, [52](#)

ViewPC, [66](#)
 ~ViewPC, [68](#)
 abortCircuit, [68](#)
 abortModel, [68](#)
 alert, [69](#)
 decrypt, [70](#)
 dialog, [76](#)
 encrypt, [70](#)
 errorsDict, [76](#)
 on_actionAbout_triggered, [71](#)
 on_actionHelp_triggered, [71](#)
 on_fileButton_clicked, [72](#)
 on_startButton_clicked, [72](#)
 progressDialogClosed, [76](#)
 saveData, [72](#)
 saveImage, [73](#)
 setBitsUsed, [74](#)
 setEncryptMode, [74](#)
 setJPHSDir, [75](#)
 setProgress, [75](#)
 setVersion, [75](#)
 ViewPC, [68](#)

xTime
 qaesencryption.cpp, [82](#)

zip
 EncryptDialog, [34](#)
 ModelPC, [50](#)