ak-file-encryptor

0.0.1

Документация по ak-file-encryptor. Последние изменения: Пт 18 Окт 2024 03:50:23.
Создано системой Doxygen 1.12.0

Пт 18 Окт 2024 03:50:23

Глава 1

Ошибки

Файл main.cpp

Их просто много

Глава 2

Алфавитный указатель структур данных

2.1 Структуры данных

Структуры данных с их кратким описанием.

Глава 3

Список файлов

3.1 Файлы

Полный список файлов.

Глава 4

Структуры данных

4.1 Класс CryptoProvider

#include <crypto_provider.hpp>

Открытые статические члены

- static int generate_key_from_password (const std::string &password, const std::string &salt, struct bckey *key, const std::string &algorithm="magma")
  Генерирует ключ на основе пароля и соли.
- static void generate_random_string (size_t length, char *output)
  Генерирует случайную строку заданной длины.
- static std::string encrypt (const std::string &plain_text, struct bckey *key)
  Шифрует текст с использованием указанного ключа.
- static std::string decrypt (const std::string &cipher_text, struct bckey *key)
  Дешифрует текст с использованием указанного ключа.
- static ak_uint8 * encrypt (ak_uint8 *plain_text, size_t size, struct bckey *key)
  Шифрует массив байтов с использованием указанного ключа.
- static ak_uint8 * decrypt (ak_uint8 *cipher_text, size_t size, struct bckey *key)
  Дешифрует массив байтов с использованием указанного ключа.
- static bool ak_save_to_file (const ak_uint8 *data, size_t size, const std::string &original_file)
  Сохраняет данные в файл.

4.1.1 Методы

4.1.1.1 ak_save_to_file()

bool CryptoProvider::ak_save_to_file (
const ak_uint8 * data,
size_t size,
const std::string & original_file) [static]

Сохраняет данные в файл.

Эта функция сохраняет указанный массив байтов в файл. Если файл имеет расширение .akr, оно будет удалено. Если нет, будет добавлено расширение .akr.

Аргументы

data	Указатель на данные для сохранения.
size	Размер данных.
original_file	Имя исходного файла.

Возвращает

bool true, если сохранение прошло успешно, иначе false.

4.1.1.2 decrypt() [1/2]

```
ak_uint8 * CryptoProvider::decrypt (
            ak_uint8 * cipher_text,
            size_t size,
            struct bckey * key)   [static]
```

Дешифрует массив байтов с использованием указанного ключа.

Эта функция выполняет дешифрование зашифрованного массива байтов.

Аргументы

cipher_text	Указатель на зашифрованный текст для дешифрования.
size	Размер массива.
key	Указатель на структуру bckey, содержащую ключ.

Возвращает

ak_uint8* Указатель на открытый текст.

4.1.1.3 decrypt() [2/2]

```
std::string CryptoProvider::decrypt (
            const std::string & cipher_text,
            struct bckey * key)   [static]
```

Дешифрует текст с использованием указанного ключа.

Эта функция выполняет дешифрование зашифрованного текста, используя тот же ключ, что и для шифрования.

Аргументы

cipher_text	Зашифрованный текст для дешифрования.
key	Указатель на структуру bckey, содержащую ключ.

Возвращает

std::string Открытый текст.

4.1.1.4 encrypt() [1/2]

```
ak_uint8 * CryptoProvider::encrypt (
                ak_uint8 * plain_text,
                size_t size,
                struct bckey * key)    [static]
```

Шифрует массив байтов с использованием указанного ключа.

Эта функция выполняет шифрование массива байтов, возвращая зашифрованный массив.

Аргументы

plain_text	Указатель на открытый текст для шифрования.
size	Размер массива.
key	Указатель на структуру bckey, содержащую ключ.

Возвращает

ak_uint8* Указатель на зашифрованный массив байтов.

Исключения

std::runtime_error	Если шифрование не удалось.

4.1.1.5 encrypt() [2/2]

```
std::string CryptoProvider::encrypt (
                const std::string & plain_text,
                struct bckey * key)    [static]
```

Шифрует текст с использованием указанного ключа.

Эта функция выполняет шифрование открытого текста с помощью алгоритма OFB и заданного ключа, возвращая зашифрованный текст.

Аргументы

plain_text	Открытый текст для шифрования.
key	Указатель на структуру bckey, содержащую ключ.

Возвращает

std::string Зашифрованный текст.

Исключения

std::runtime_error	Если шифрование не удалось.

4.1.1.6 generate_key_from_password()

```
int CryptoProvider::generate_key_from_password (
                const std::string & password,
                const std::string & salt,
                struct bckey * key,
                const std::string & algorithm = "magma")    [static]
```

Генерирует ключ на основе пароля и соли.

Эта функция инициализирует библиотеку libakrypt и создает ключ, используя указанный алгоритм (kuznechik или magma). Затем она устанавливает ключ на основе предоставленного пароля и соли.

Аргументы

password	Пароль, используемый для генерации ключа.
salt	Соль, используемая для генерации ключа.
key	Указатель на структуру bckey, где будет храниться ключ.
algorithm	Алгоритм для генерации ключа (kuznechik или magma).

Возвращает

int Код ошибки (EXIT_SUCCESS при успехе, EXIT_FAILURE при ошибке).

4.1.1.7 generate_random_string()

```
void CryptoProvider::generate_random_string (
                size_t length,
                char * output)    [static]
```

Генерирует случайную строку заданной длины.

Эта функция создает строку длиной length, заполняя ее случайными символами из заданного набора символов.

Аргументы

length	Длина генерируемой строки.
output	Указатель на буфер, куда будет записана строка.

Объявления и описания членов классов находятся в файлах:

- src/processor/crypto_provider.hpp
- src/processor/crypto_provider.cpp

4.2 Класс MainMenu

#include <main_menu.hpp>

Открытые типы

- enum OptionsSelected {
  NONE = 0 , RETURN , EXIT , CONTINUE ,
  PICK_ANOTHER , OPT_FILE , OPT_STRING , OPT_ROOT ,
  TYPE_ENCRYPT , TYPE_DECRYPT }

Открытые статические члены

- static MainMenu::OptionsSelected showMenu ()

  Отображает главное меню и обрабатывает выбор пользователя.

Закрытые статические члены

- static void handleInterrupt (int signal=0)

  Обработчик прерываний SIGINT и SIGTERM.
- static void initializeNCR ()

  Инициализирует интерфейс ncurses и настраивает терминал.
- static MainMenu::OptionsSelected cleanupNCR (MainMenu::OptionsSelected option=NONE)

  Восстанавливает терминал в исходное состояние и завершает режим ncurses.
- static MainMenu::OptionsSelected getUserInput (const std::set< char > &validInputs={})

  Ожидает ввод пользователя и возвращает выбранную опцию.
- static std::string getInputString (int line, unsigned int max_length=64)

  Интерфейс получения строки от пользователя.
- static bool getYesNoInput (int line, const std::string &question)

  Запрашивает у пользователя ввод "Да" или "Нет".
- static std::string getInputWithFileValidation (const std::string &prompt)

  Получает строковый ввод от пользователя с проверкой на существующий файл.

Закрытые статические данные

- static struct termios orig_termios

4.2.1 Перечисления

4.2.1.1 OptionsSelected

enum MainMenu::OptionsSelected

Элементы перечислений

NONE	
RETURN	
EXIT	
CONTINUE	
PICK_ANOTHER	
OPT_FILE	
OPT_STRING	
OPT_ROOT	
TYPE_ENCRYPT	
TYPE_DECRYPT	

4.2.2 Методы

4.2.2.1 cleanupNCR()

MainMenu::OptionsSelected MainMenu::cleanupNCR (
MainMenu::OptionsSelected option = NONE) [static], [private]

Восстанавливает терминал в исходное состояние и завершает режим ncurses.

Аргументы

option	Опция для возврата.

Возвращает

MainMenu::OptionsSelected Возвращаемая опция.

4.2.2.2 getInputString()

std::string MainMenu::getInputString (
int line,
unsigned int max_length = 64) [static], [private]

Интерфейс получения строки от пользователя.

Аргументы

line	Строка в интерфейсе, где выводится ввод.
max_length	Максимальная длина строки.

Возвращает

std::string Введенная строка.

4.2.2.3 getInputWithFileValidation()

std::string MainMenu::getInputWithFileValidation (
const std::string & prompt) [static], [private]

Получает строковый ввод от пользователя с проверкой на существующий файл.

Функция позволяет пользователю вводить строку, при нажатии на Tab проверяется текущий введенный путь. Если это каталог, отображается его содержимое. Пользователь не может нажать Enter до тех пор, пока не введет путь до существующего файла.

Аргументы

prompt	Сообщение, отображаемое пользователю перед вводом.

Возвращает

std::string Строка, введенная пользователем.

4.2.2.4 getUserInput()

MainMenu::OptionsSelected MainMenu::getUserInput (
const std::set< char > & valid_inputs = {}) [static], [private]

Ожидает ввод пользователя и возвращает выбранную опцию.

Эта функция отображает время до окончания ожидания и отслеживает ввод пользователя для выбора опции из допустимых значений. Если время истекает или пользователь вводит недопустимое значение, функция завершает работу и возвращает OptionsSelected::EXIT.

Аргументы

valid_inputs	Набор допустимых символов для выбора опций.

Возвращает

MainMenu::OptionsSelected Выбранная пользователем опция.

4.2.2.5 getYesNoInput()

bool MainMenu::getYesNoInput (
int line,
const std::string & question) [static], [private]

Запрашивает у пользователя ввод "Да" или "Нет".

Аргументы

line	Строка для вывода вопроса.
question	Вопрос для пользователя.

Возвращает

true Если пользователь выбрал "Да".

false Если пользователь выбрал "Нет".

4.2.2.6 handleInterrupt()

void MainMenu::handleInterrupt (
int signal = 0) [static], [private]

Обработчик прерываний SIGINT и SIGTERM.

Аргументы

signal	Номер сигнала.

4.2.2.7 initializeNCR()

void MainMenu::initializeNCR () [static], [private]

Инициализирует интерфейс ncurses и настраивает терминал.

< Сохраняем исходное состояние терминала

< Перехватываем сигналы завершения

< Инициализируем режим ncurses

4.2.2.8 showMenu()

MainMenu::OptionsSelected MainMenu::showMenu () [static]

Отображает главное меню и обрабатывает выбор пользователя.

Возвращает

MainMenu::OptionsSelected Опция, выбранная пользователем.

< Меню выбора цели операции

< Обработка выбора цели операции

< Меню выбора типа операции

< Меню выбора типа операции

4.2.3 Поля

4.2.3.1 orig_termios

struct termios MainMenu::orig_termios [static], [private]

Объявления и описания членов классов находятся в файлах:

- src/gui/main_menu.hpp
- src/gui/main_menu.cpp

Глава 5

Файлы

5.1 Файл src/gui/main_menu.cpp

```
#include "main_menu.hpp"
#include "crypto_provider.hpp"
#include <cstring>
#include <ncurses.h>
#include <unistd.h>
#include <termios.h>
#include <signal.h>
#include <stdlib.h>
#include <string>
#include <chrono>
#include <thread>
#include <filesystem>
#include <libakrypt.h>
```

5.2 Файл src/gui/main_menu.hpp

Основной файл графической части ak-file-encryptor.

```
#include <set>
#include <string>
```

Структуры данных

- class MainMenu

5.2.1 Подробное описание

Основной файл графической части ak-file-encryptor.

Хэдер графической части ak-file-encryptor.

>

Содержит в себе кучу мусора, а так же пару интересных наработак связвнных с ncurses.

Автор

THE_CHOODICK

Дата

18-10-2024

Версия

0.0.1

Предупреждения

Этот проект предназначен только для ознокомительных целей, сам проект содежит огромное количество говнокода и багов.

Авторство

5.3 main_menu.hpp

См. документацию.

```
00001
00019 #ifndef MAIN_MENU_HPP
00020 #define MAIN_MENU_HPP
00021
00022 #include <set>
00023 #include <string>
00024
00025 class MainMenu
00026 {
00027 public:
00028     enum OptionsSelected
00029     {
00030         NONE = 0,
00031         RETURN,
00032         EXIT,
00033         CONTINUE,
00034         PICK_ANOTHER,
00035
00036         OPT_FILE,
```

```
00037         OPT_STRING,
00038         OPT_ROOT,
00039
00040         TYPE_ENCRYPT,
00041         TYPE_DECRYPT
00042     };
00043
00044 public:
00045     static MainMenu::OptionsSelected showMenu();
00046
00047 private:
00048     static void handleInterrupt(int signal = 0);
00049
00050     static void initializeNCR();
00051     static MainMenu::OptionsSelected cleanupNCR(MainMenu::OptionsSelected option = NONE);
00052     static MainMenu::OptionsSelected getUserInput(const std::set<char>& validInputs = {});
00053
00054     static std::string getInputString(int line, unsigned int max_length = 64);
00055     static bool getYesNoInput(int line, const std::string& question);
00056     static std::string getInputWithFileValidation(const std::string& prompt);
00057
00058 private:
00059     static struct termios orig_termios;
00060 };
00061
00062 #endif // MAIN_MENU_HPP
```

5.4 Файл src/main.cpp

Основной файл проекта ak-file-encryptor.

#include "gui/main_menu.hpp"

Функции

- int main ()

5.4.1 Подробное описание

Основной файл проекта ak-file-encryptor.

>

По своей сути просто запускает графический интерфейс.

Автор

THE_CHOODICK

Дата

18-10-2024

Версия

0.0.1

Предупреждения

Этот проект предназначен только для ознокомительных целей, сам проект содежит огромное количество говнокода и багов.

Ошибка Их просто много

Авторство

5.4.2 Функции

5.4.2.1 main()

int main ()

5.5 Файл src/processor/crypto_provider.cpp

Основной файл криптографической части ak-file-encryptor.

```
#include "crypto_provider.hpp"
#include <iostream>
#include <filesystem>
#include <fstream>
#include <libakrypt.h>
```

5.5.1 Подробное описание

Основной файл криптографической части ak-file-encryptor.

>

Содержит в себе несколько оберточных функций для более простой работы с libakrypt.

Автор

THE_CHOODICK

Дата

18-10-2024

Версия

0.0.1

Предупреждения

Этот проект предназначен только для ознокомительных целей, сам проект содежит огромное количество говнокода и багов.

Авторство

5.6 Файл src/processor/crypto_provider.hpp

Хэдер криптографической части ak-file-encryptor.

#include <string>
#include <stddef.h>

Структуры данных

- class CryptoProvider

Макросы

- #define SALT_SIZE 16
- #define BLOCK_SIZE 16
- #define ITERATIONS 10000
- #define IV { 0x01, 0x02, 0x03, 0x04, 0x11, 0xaa, 0x4e, 0x12 }
- #define IV_SIZE 8

Определения типов

- typedef unsigned char ak_uint8

5.6.1 Подробное описание

Хэдер криптографической части ak-file-encryptor.

>

Содержит в себе обявления несколько оберточных функций,
предназначенных для более простой работы с libakrypt.

Автор

THE_CHOODICK

Дата

18-10-2024

Версия

0.0.1

Предупреждения

Этот проект предназначен только для ознокомительных целей, сам проект содежит огромное количество говнокода и багов.

Авторство

(Not really)

5.6.2 Макросы

5.6.2.1 BLOCK_SIZE

#define BLOCK_SIZE 16

5.6.2.2 ITERATIONS

#define ITERATIONS 10000

5.6.2.3 IV

#define IV { 0x01, 0x02, 0x03, 0x04, 0x11, 0xaa, 0x4e, 0x12 }

5.6.2.4 IV_SIZE

#define IV_SIZE 8

5.6.2.5 SALT_SIZE

#define SALT_SIZE 16

5.6.3 Типы

5.6.3.1 ak_uint8

typedef unsigned char ak_uint8

5.7 crypto_provider.hpp

См. документацию.

```
00001
00020 #ifndef CRYPO_PROVIDER_HPP
00021 #define CRYPO_PROVIDER_HPP
00022
00023 #include <string>
00024 #include <stddef.h>
00025
00026 #define SALT_SIZE 16
00027 #define BLOCK_SIZE 16
00028 #define ITERATIONS 10000
00029 #define IV { 0x01, 0x02, 0x03, 0x04, 0x11, 0xaa, 0x4e, 0x12 }
00030 #define IV_SIZE 8
00031
00032 typedef unsigned char ak_uint8;
00033
00034 class CryptoProvider
00035 {
00036 public:
00037     static int generate_key_from_password(const std::string &password, const std::string &salt, struct bckey *key, const
      std::string &algorithm = "magma");
00038     static void generate_random_string(size_t length, char *output);
00039     static std::string encrypt(const std::string& plain_text, struct bckey *key);
00040     static std::string decrypt(const std::string& cipher_text, struct bckey *key);
00041     static ak_uint8* encrypt(ak_uint8* plain_text, size_t size, struct bckey *key);
00042     static ak_uint8* decrypt(ak_uint8* cipher_text, size_t size, struct bckey *key);
00043     static bool ak_save_to_file(const ak_uint8* data, size_t size, const std::string& original_file);
00044 };
00045
00046 #endif // CRYPO_PROVIDER_HPP
00047
```

Предметный указатель