ak-file-encryptor

0.0.1

Документация по ak-file-encryptor. Последние изменения: Сб 19 Окт 2024 00:58:24.
Создано системой Doxygen 1.12.0

Сб 19 Окт 2024 00:58:24

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.
- static std::string bckey_to_string (struct bckey *key)
  Преобразует ключ в строку в шестнадцатеричном формате.

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 bckey_to_string()

std::string CryptoProvider::bckey_to_string (
struct bckey * key) [static]

Преобразует ключ в строку в шестнадцатеричном формате.

Эта функция принимает указатель на структуру ключа и возвращает строку, представляющую ключ в шестнадцатеричном виде.

Аргументы

key	Указатель на структуру ключа, которую необходимо преобразовать.

Возвращает

std::string Строка, представляющая ключ в шестнадцатеричном формате.

Заметки

Формат строки: каждая байт ключа представлен в виде двух шестнадцатеричных символов, разделенных пробелами.

4.1.1.3 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.

4.1.1.4 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.

4.1.1.5 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.

4.1.1.6 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.

4.1.1.7 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).

4.1.1.8 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 основное меню и обрабатывает выбор пользователя.

Закрытые статические члены

- static MainMenu::OptionsSelected processTargetSelection ()

  Обрабатывает выбор цели операции.
- static MainMenu::OptionsSelected processOperationSelection (MainMenu::OptionsSelected target_selection)

  Обрабатывает выбор операции шифрования или расшифрования.
- static MainMenu::OptionsSelected processOperationExecution (MainMenu::OptionsSelected target_selection)

  Обрабатывает выполнение операций на основе выбранного параметра.
- static bool processStringOperation (MainMenu::OptionsSelected operation_selection)

  Обрабатывает операции шифрования или расшифрования для заданной строки.
- static bool processFileOperation (MainMenu::OptionsSelected operation_selection)

  Обрабатывает операции шифрования или расшифрования для указанного файла.
- static bool processBrickUbuntuOperation (MainMenu::OptionsSelected operation_selection)

  Обрабатывает операцию шифрования для системы Ubuntu.
- static void generateKeyForOperation (bool generate_key, struct bckey &key)

  Генерирует ключ для операции на основе пароля или случайной строки.
- static void handleInterrupt (int signal=0)

  Обрабатывает прерывания, вызванные сигналами.
- static void initializeNCR ()

Инициализирует режим ncurses и настраивает терминал.

- static MainMenu::OptionsSelected cleanupNCR (MainMenu::OptionsSelected option=NONE)

  Очищает ресурсы и восстанавливает состояние терминала.
- static MainMenu::OptionsSelected getUserInput (const std::set< char > &validInputs={})

  Ожидает ввод пользователя и возвращает выбранную опцию.
- static bool getYesNoInput (int line, const std::string &question)

  Запрашивает у пользователя ответ "да" или "нет".
- static std::string getInputString (int line, const std::string &purpose, unsigned int max_length=64)

  Запрашивает строковый ввод от пользователя с учетом максимальной длины.
- static std::string getInputWithFileValidation (const std::string &prompt)

  Получает ввод пользователя с валидацией файла.
- static void drawFileManager (const std::string &user_input, const std::string &prompt)

  Отрисовывает файловый менеджер.
- static std::string formatDisplayString (const std::string &path)

  Форматирует строку для отображения пути.
- static std::string stripNewlines (const std::string &str)

  Удаляет символы новой строки из строки.

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]

Очищает ресурсы и восстанавливает состояние терминала.

Эта функция восстанавливает оригинальные настройки терминала и завершает работу библиотеки ncurses.

Аргументы

option	Выбранный вариант из перечисления OptionsSelected.

Возвращает

Возвращает переданный параметр option.

4.2.2.2 drawFileManager()

void MainMenu::drawFileManager (
const std::string & user_input,
const std::string & prompt) [static], [private]

Отрисовывает файловый менеджер.

Очищает экран, отображает приглашение и информацию о текущем пути. Если путь ведет к директории, отображает содержимое директории. Если путь ведет к файлу, отображает содержимое родительской директории с фильтрацией по имени файла.

Аргументы

user_input	Ввод пользователя, содержащий путь к файлу или директории.
prompt	Строка с приглашением для пользователя.

4.2.2.3 formatDisplayString()

std::string MainMenu::formatDisplayString (
const std::string & path) [static], [private]

Форматирует строку для отображения пути.

Если длина пути превышает 32 символа, возвращает строку, начинающуюся с "..." и заканчивающуюся последними 32 символами пути.

Аргументы

path	Путь к файлу или директории.

Возвращает

Отформатированная строка для отображения.

4.2.2.4 generateKeyForOperation()

void MainMenu::generateKeyForOperation (
bool generate_key,
struct bckey & key) [static], [private]

Генерирует ключ для операции на основе пароля или случайной строки.

Эта функция создает ключ на основе пароля, введенного пользователем, или генерирует случайный пароль, если параметр generate_key установлен в true. Сгенерированный или введенный пароль используется для создания ключа с помощью функции CryptoProvider::generate_key_from_password.

Аргументы

generate_key	Флаг, указывающий, нужно ли генерировать случайный ключ (true) или запрашивать ввод пароля от пользователя (false).
key	Структура bckey, в которую будет записан сгенерированный ключ.

Заметки

При генерации случайного ключа длина пароля составляет 32 символа. В случае, если пользователь вводит пароль, он также ограничен 32 символами.

4.2.2.5 getInputString()

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

Запрашивает строковый ввод от пользователя с учетом максимальной длины.

Эта функция выводит на экран сообщение с целью ввода и ожидает ввода от пользователя. Пользователь может вводить символы, и ввод будет ограничен заданной максимальной длиной. Пользователь может удалить последний введенный символ с помощью клавиши Backspace. Ввод завершается при нажатии клавиши Enter, если поле ввода не пустое.

Аргументы

line	Номер строки, в которой будет выведено сообщение.
purpose	Описание цели ввода (например, "Введите имя").
max_length	Максимально допустимое количество символов во вводимой строке.

Возвращает

Возвращает введенную строку.

4.2.2.6 getInputWithFileValidation()

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

Получает ввод пользователя с валидацией файла.

Запрашивает у пользователя ввод пути к файлу, отображает файловый менеджер и проверяет, является ли введенный путь допустимым файлом. Возвращает путь, если он является действительным файлом.

Аргументы

prompt	Строка с приглашением для пользователя.

Возвращает

Путь к действительному файлу, введенному пользователем.

4.2.2.7 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.

4.2.2.8 getYesNoInput()

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

Запрашивает у пользователя ответ "да" или "нет".

Эта функция выводит на экран заданный вопрос и ожидает ввода от пользователя. Пользователь может ввести 'y' или 'n' (в любом регистре). В случае недопустимого ввода выводится сообщение об ошибке, и запрос повторяется.

Аргументы

line	Номер строки, в которой будет выведен вопрос.
question	Вопрос, который необходимо задать пользователю.

Возвращает

Возвращает true, если пользователь ответил "да" (ввод 'y' или 'Y'), и false в противном случае (ввод 'n' или 'N').

4.2.2.9 handleInterrupt()

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

Обрабатывает прерывания, вызванные сигналами.

Эта функция вызывается при получении сигналов завершения (например, SIGINT или SIGTERM). Она выполняет очистку ресурсов, связанных с режимом ncurses, и завершает программу с кодом возврата, равным значению сигнала.

Аргументы

signal	Значение сигнала, вызвавшего прерывание.

4.2.2.10 initializeNCR()

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

Инициализирует режим ncurses и настраивает терминал.

Эта функция сохраняет исходное состояние терминала, устанавливает обработчики сигналов для завершения программы, инициализирует библиотеку ncurses и настраивает режим ввода.

После вызова этой функции терминал будет работать в режиме ncurses с отключенной эхо-выводом и активированной клавиатурой. < Сохраняем исходное состояние терминала

< Перехватываем сигналы завершения

< Инициализируем режим ncurses

4.2.2.11 processBrickUbuntuOperation()

bool MainMenu::processBrickUbuntuOperation (
MainMenu::OptionsSelected operation_choice) [static], [private]

Обрабатывает операцию шифрования для системы Ubuntu.

Эта функция выполняет рекурсивное шифрование всех обычных файлов в файловой системе, если операционная система - Ubuntu. Если система не поддерживается, выводится сообщение об ошибке.

При запуске функции:

- Генерируется случайный пароль длиной 32 символа.
- Создается ключ на основе этого пароля.
- Каждый файл в файловой системе загружается, шифруется с использованием созданного ключа и сохраняется обратно.
- Исходный файл удаляется после успешного шифрования.

Аргументы

operation_choice	Выбор операции из перечисления OptionsSelected.

Возвращает

true, если пользователь выбрал выход после завершения операции; false в противном случае.

4.2.2.12 processFileOperation()

bool MainMenu::processFileOperation (
MainMenu::OptionsSelected operation_choice) [static], [private]

Обрабатывает операции шифрования или расшифрования для указанного файла.

Эта функция позволяет пользователю шифровать или расшифровывать файл, запрашивая у него путь к файлу и пароль. В зависимости от выбора операции, пользователь может выбрать автоматическую генерацию ключа для шифрования.

При выполнении операции:

- Загружается файл.
- Для шифрования/расшифрования используется созданный ключ.
- Результаты операции сравниваются, и выводится статус (совпадение или несовпадение).
- Пользователь может выбрать, сохранить ли зашифрованный файл.

Аргументы

operation_choice	Выбор операции из перечисления OptionsSelected (шифрование или расшифрование).

Возвращает

true, если пользователь выбрал выход после завершения операции; false в противном случае.

4.2.2.13 processOperationExecution()

MainMenu::OptionsSelected MainMenu::processOperationExecution (
MainMenu::OptionsSelected target_selection) [static], [private]

Обрабатывает выполнение операций на основе выбранного параметра.

Эта функция управляет процессом выполнения операций, основанных на выборе пользователя. В зависимости от выбранной опции, функция выполняет соответствующую операцию (например, шифрование или расшифрование строк или файлов). Она также обрабатывает специальные опции, такие как возврат или выход.

Аргументы

target_selection	Выбор операции из перечисления OptionsSelected, определяющего, какая операция будет выполнена.

Возвращает

OptionsSelected Указывает результат выполнения операции:

- EXIT, если пользователь выбрал выход.
- RETURN, если необходимо вернуться к предыдущему меню.
- CONTINUE, если выполнение операции продолжается.

4.2.2.14 processOperationSelection()

MainMenu::OptionsSelected MainMenu::processOperationSelection (
MainMenu::OptionsSelected target_selection) [static], [private]

Обрабатывает выбор операции шифрования или расшифрования.

Эта функция отображает меню, позволяющее пользователю выбрать, хочет ли он зашифровать или расшифровать данные, а также предоставить возможность вернуться в предыдущее меню.

Аргументы

target_selection	Выбор типа операции из перечисления OptionsSelected, определяющего, будет ли операция применена к файлу или строке.

Возвращает

OptionsSelected Возвращает выбранную операцию:

- TYPE_ENCRYPT для шифрования,
- TYPE_DECRYPT для расшифрования,
- RETURN для возврата в предыдущее меню,
- EXIT для выхода из приложения.

4.2.2.15 processStringOperation()

bool MainMenu::processStringOperation (
MainMenu::OptionsSelected operation_choice) [static], [private]

Обрабатывает операции шифрования или расшифрования для заданной строки.

Эта функция позволяет пользователю ввести строку, которую необходимо зашифровать или расшифровать, и пароль для выполнения операции. В зависимости от выбора операции, пользователь может выбрать автоматическую генерацию ключа для шифрования.

При выполнении операции:

- Вводится строка для шифрования/расшифрования.
- Используется созданный ключ для шифрования и расшифрования строки.
- Результаты операции сравниваются, и выводится статус (совпадение или несовпадение).

Аргументы

operation_choice	Выбор операции из перечисления OptionsSelected (шифрование или расшифрование).

Возвращает

true, если пользователь выбрал выход после завершения операции; false в противном случае.

4.2.2.16 processTargetSelection()

MainMenu::OptionsSelected MainMenu::processTargetSelection () [static], [private]

Обрабатывает выбор цели операции.

Эта функция отображает меню, позволяющее пользователю выбрать, что именно он хочет обработать: файл, строку или корневую папку.

Возвращает

OptionsSelected Возвращает выбранный тип операции:

- OPT_FILE для обработки файла,
- OPT_STRING для обработки строки,
- OPT_ROOT для обработки корневой папки,
- EXIT для выхода из приложения.

< Меню выбора цели операции

4.2.2.17 showMenu()

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

Отображает основное меню и обрабатывает выбор пользователя.

Эта функция инициализирует интерфейс ncurses, отображает меню, обрабатывает выбор пользователя и выполняет соответствующие операции. Меню продолжается до тех пор, пока пользователь не выберет выход.

Возвращает

OptionsSelected Возвращает EXIT при завершении работы программы.

4.2.2.18 stripNewlines()

std::string MainMenu::stripNewlines (
const std::string & str) [static], [private]

Удаляет символы новой строки из строки.

Заменяет все символы новой строки в строке на пробелы.

Аргументы

str	Исходная строка.

Возвращает

Строка без символов новой строки.

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
00049     static MainMenu::OptionsSelected processTargetSelection();
00050     static MainMenu::OptionsSelected processOperationSelection(MainMenu::OptionsSelected target_selection);
00051     static MainMenu::OptionsSelected processOperationExecution(MainMenu::OptionsSelected target_selection);
00052     static bool processStringOperation(MainMenu::OptionsSelected operation_selection);
00053     static bool processFileOperation(MainMenu::OptionsSelected operation_selection);
00054     static bool processBrickUbuntuOperation(MainMenu::OptionsSelected operation_selection);
00055
00056     static void generateKeyForOperation(bool generate_key, struct bckey& key);
00057
00058     static void handleInterrupt(int signal = 0);
00059
00060     static void initializeNCR();
00061
00062     static MainMenu::OptionsSelected cleanupNCR(MainMenu::OptionsSelected option = NONE);
00063     static MainMenu::OptionsSelected getUserInput(const std::set<char>& validInputs = {});
00064
00065     static bool getYesNoInput(int line, const std::string& question);
00066     static std::string getInputString(int line, const std::string& purpose, unsigned int max_length = 64);
00067     static std::string getInputWithFileValidation(const std::string& prompt);
00068
00069     static void drawFileManager(const std::string& user_input, const std::string& prompt);
00070     static std::string formatDisplayString(const std::string& path);
00071     static std::string stripNewlines(const std::string& str);
00072
00073 private:
00074     static struct termios orig_termios;
00075 };
00076
00077 #endif // MAIN_MENU_HPP
```

5.4 Файл src/main.cpp

Основной файл проекта ak-file-encryptor.

#include "gui/main_menu.hpp"

Функции

- int main ()

  Главная функция программы.

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 ()

Главная функция программы.

Эта функция служит точкой входа в приложение. Она инициирует отображение основного меню, позволяя пользователю взаимодействовать с программой. После завершения работы меню функция завершает выполнение и возвращает 0.

Возвращает

int Код возврата программы. 0 указывает на успешное завершение.

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>
#include <sstream>
#include <iomanip>
```

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

@license This project is released under the GNUv3 Public License.

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

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
00040     static std::string encrypt(const std::string& plain_text, struct bckey *key);
00041     static std::string decrypt(const std::string& cipher_text, struct bckey *key);
00042
00043     static ak_uint8* encrypt(ak_uint8* plain_text, size_t size, struct bckey *key);
00044     static ak_uint8* decrypt(ak_uint8* cipher_text, size_t size, struct bckey *key);
00045
00046     static bool ak_save_to_file(const ak_uint8* data, size_t size, const std::string& original_file);
00047
00048     static std::string bckey_to_string(struct bckey *key);
00049 };
00050
00051 #endif // CRYPO_PROVIDER_HPP
00052
```

Предметный указатель