# AES 加密技术文档

## 1.概述

该加密是调用了第三方 AES 加密算法，经由简单封装实现。增加了是否是加密文件的判断，以及解密时的校验。

## 2. 接口

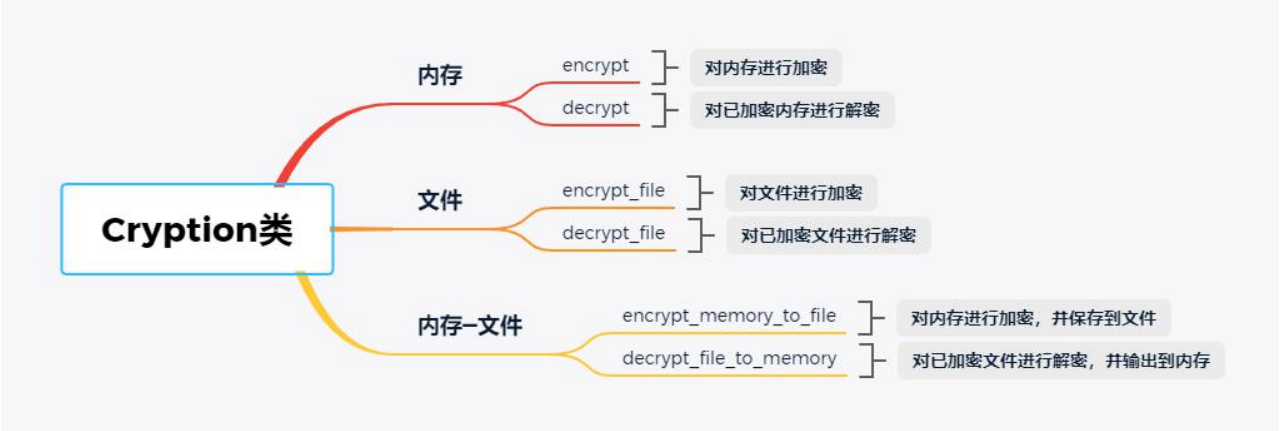


图 2-1 Cryption 类接口

```
class Cryption
{
public:
    //对内存进行加密
    int encrypt(int type, uint8_t *inData, int64_t inLen, uint8_t *key,
        uint8_t **outData, int64_t *outLen);
    //对内存进行解密
    int decrypt(int type, uint8_t *inData, int64_t inLen, uint8_t *key,
        uint8_t **outData, int64_t *outLen);

    //对文件进行加密
    int encrypt_file(int type, const char *inPath, const char *outPath,
        const char *key);
    //对文件进行解密
    int decrypt_file(int type, const char *inPath, const char *outPath,
        const char *key);
```

```
    //将内存加密后保存到文件
    int encrypt_memory_to_file(int type, uint8_t *inData, int64_t inLen,
        uint8_t *key, const char *outPath);
    //对文件进行解密，并输出到内存
    int decrypt_file_to_memory(int type, const char *inPath, uint8_t *key,
        uint8_t **outData, int64_t *outLen);
}
```

补充说明：key 可由随意字符组成，最长不超过 32 位；key 为 NULL 时，将使用内置默认密码；

# 3.实现说明

## 3.1 文件说明

第三方 AES 算法实现：rijndael.h、rijndael.c
封装类文件：encrypt.h、encrypt.cpp

## 3.2 实现流程图

### 3.1.1 加密

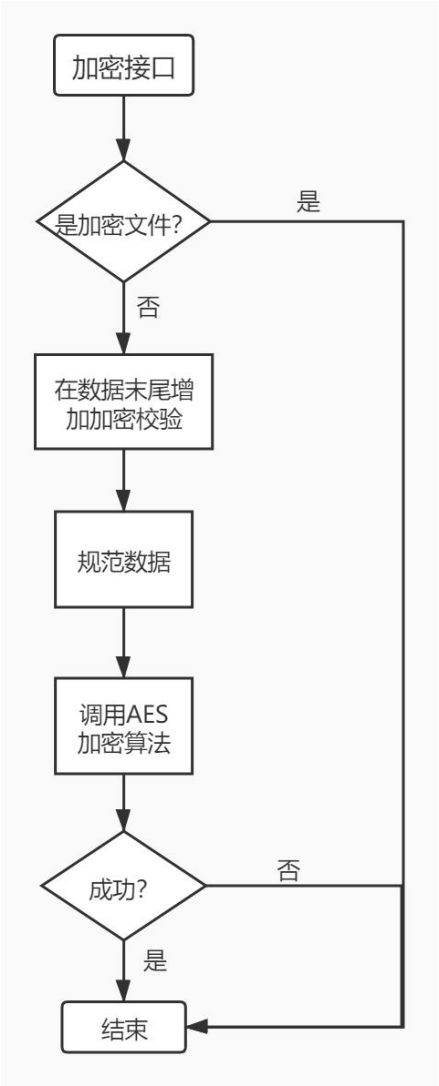


图 3-1 加密流程图

### 3.1.2 解密

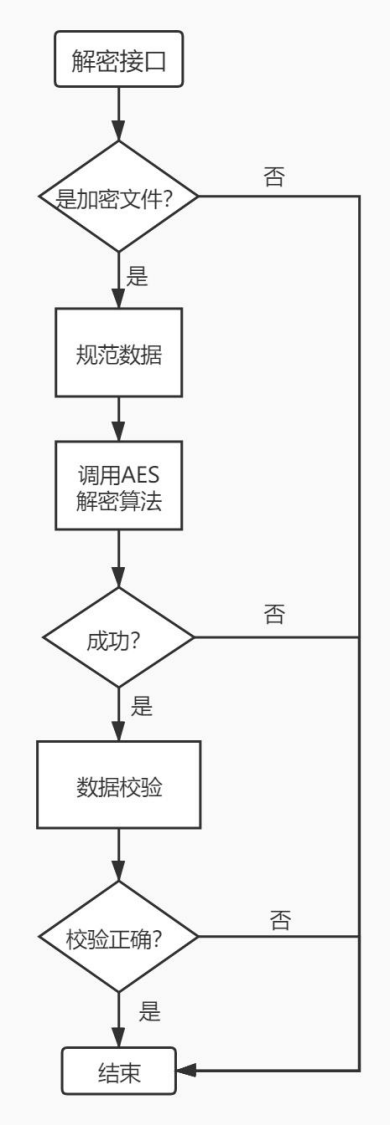


图 3-2 解密流程图

## 3.3 加密文件标识及数据校验

### 3.3.1 加密文件标识

增加加密文件标识的目的，是为了防止上层二次甚至多次调用同一数据进行加密，造成数据重复加密。实现方式是加密完成后在数据开头插入加密标识字符串 ENCRYPT_FLAG，ENCRYPT_FLAG_LEN 为标识字符串长度。

### 3.3.2 数据校验

增加数据校验的目的，是实现只有在密码正确时才返回出解析后的数据。实现方式是加密时在数据末尾插入 CHECK_CODE 字符串，然后一起进行加密，解密时对 CHECK_CODE 字符串进行校验，CHECK_CODE_LEN 为字符串长度。