

OpenPGP-FileEncrypt-Assistant

OpenPGP-FileEncrypt-Assistant

1. 软件设计文档

1.0 写在前面

1.0.1 小组分工情况

1.0.2 任务完成情况

1.0.3 原创性说明

1.1 需求分析

1.1.1 功能需求

1.2 流程设计

1.2.1 环境声明

1.2.2 功能

1.2.3 初始状态

1.3 组件关系及其调用

1.4 概要与详细设计

1.4.1 组成模块

1.4.2 各模块功能及函数声明约定

1.5 数据字典

1.5.1 类的成员

1.5.2 常用变量

1.6 编译手册

1.7 安装手册

1.8 自测用例、用户手册、测试分析报告、测试手册

2. 源代码及注释

3. 可执行安装包

4. 测试用例和分析报告

4.1 欢迎界面与主界面

4.2 模式一、原理展示

4.3 模式二、单用户存储调阅

4.4 模式三、多用户存储调阅

4.5 模式四、多用户调阅

4.6 文件的生成展示

4.7 敏感信息的覆盖核心代码

5. 第三方资源及其说明

5.1 原创性说明

5.2 网络教程学习说明

5.3 第三方库使用说明

当前版本: released 3.0。

更新日期: 2023.04.21。

开源条款: MIT LICENSE。

勘误更正:

①欢迎界面的背景“文件夹加密助手”，更正为“文件加密助手”

②主界面的参数“模式一：输入字符串；模式二：输入待加密文件地址；模式三：输入调阅文件地址(P)”，更正为：“模式一：无需输入；模式二、三：输入待加密文件地址；模式四：输入调阅文件地址(P)”；

- ③主界面的模式一，“请先输入参数{Q,W,E,P}，更正为：“请先输入参数{Q,W,E}；
④主界面的模式四，“请先输入参数{Q,W,E,P,T,R}，更正为：“请先输入参数{Q,Y,U,I,O,P}。

优点与创新：

- ①我们完成了所有任务要求，并进行了正确性验证和UI设计；
- ②我们原创地编写了整套代码，并学习了大量的OpenPGP资料；
- ③我们的exe文件已经完成了打包封装，可免安装，解压缩直接运行；
- ④代码结构相对清晰。

缺点与不足：

- ①由于时间所限，我们没有很完整地处理非法出入问题；
- ②在由VS向Qt移植的过程中，我们破坏了VS（代码版本1.3-2.0）的代码结构，这导致class FileManager被置空，相应的功能放在mainwindow中实现，其目的是为了方便ui的调用，其缺点是mainwindow的代码有些冗长，缺少较好的函数或类的划分；
- ③部分功能还可以完善，例如，添加通过自定义新建用户id-key值对的工具，删除指定的用户id-key值对、将A-B多用户测试扩展到A-B-...-N多用户测试，等等。

帮助：

如果您有建议或疑惑，欢迎在本仓库Issues提出。

1. 软件设计文档

1.0 写在前面

1.0.1 小组分工情况

学号	Github用户名	分工	贡献度
2053182	ChestnutSilver	模块一、四、UI设计	50.00%
2052338	B0730	模块二、三、报告撰写	50.00%

模块划分，详见“3. 可执行安装包”。

1.0.2 任务完成情况

我们按照目前掌握的知识，完成了**100%的所有任务！**

我们将软件**完整封装**，解压后即可运行！

我们的代码经过**11次版本迭代**，迭代版本（1.3、1.4、1.5、1.6、1.7、1.8、2.0）和最终版代码（3.0），都可在：<https://github.com/ChestnutSilver/OpenPGP-FileEncrypt-Assistant>

1.0.3 原创性说明

我们的代码是**逐行编写的**，这保证了**100%的原创性**（包括UI的配色方案在内）。

具体第三方资源使用，详见“5. 第三方资源及其说明”。

1.1 需求分析

1.1.1 功能需求

(1) 存储者

- ①通过自己的ID和Passphrase登陆；
- ②单人模式，通过真实性认证，加密签名后文件仅自己可以调阅；
- ③多人模式，通过真实性认证，授权指定用户调阅加密签名后文件。

(2) 调阅者

- ①通过自己的ID和Passphrase登陆；
- ②可通过真实性认证，调阅解密被授权文件。

(3) 其他非授权用户

- ①通过自己的ID和Passphrase登陆；
- ②不可通过身份验证，无法以可理解的方式读出文件内容。

(4) 系统管理员

- ①直接通过系统生成对应的ID和Passphrase；
- ②进行操作前仍需通过ID和Passphrase登录；
- ③调阅也需授权，无特殊权限。

(5) 敏感信息覆盖

解决方案：Windows API GetFileMapping + SecureZeroMemory。

我们通过 `createFile` 打开了指定目录下的 `a.txt` 文件，并得到了一个文件句柄 `hFile`。然后，我们使用 `CreateFileMapping` 对该文件进行映射，并得到了一个文件映射句柄 `hMapFile`。接下来，我们使用 `MapViewOfFile` 将文件映射到进程的地址空间中，并获得了指向该映射区域的指针 `lpMapView`。最后，我们可以通过访问 `lpMapView` 指向的内存区域，从而读取到该文件的内容。使用 `SecureZeroMemory` 函数被用来覆盖文件映射的地址空间。在程序结束时，我们调用 `UnmapViewOfFile`、`CloseHandle` 函数分别释放内存和关闭相关句柄，避免资源泄露。

1.2 流程设计

1.2.1 环境声明

(1) 运行环境

系统：Windows 10

软件：Visual Studio 2019 (x86 Debug模式)

库：IPWorks OpenPGP 2022 C++ Edition

(2) 调试环境

系统：Windows 10

软件：Visual Studio 2019 (x86 Debug模式)

库：IPWorks OpenPGP 2022 C++ Edition

1.2.2 功能

(1) 原理展示

目标：展现用户通过密钥签名加密和解密验证字符串的过程。

步骤：

- ①输入待加密字符串；
- ②通过用户私钥签名，公钥加密后输出处理后字符串；
- ③通过用户私钥解密，公钥验证签名后输出解密后字符串。

(2) 存储功能

(2.1) 单用户授权

目标：A 用户想要存储自己的文件，并仅自己拥有调阅权限。

步骤：

- ①输入待加密文件路径；
- ②通过A用户私钥签名，A用户公钥加密后保存为.gpg文件。

(2.2) 多用户授权

目标：A 用户想要存储自己的文件，并使 B 用户都拥有调阅权限。

步骤：

- ①设置授权用户B；
- ②输入待加密文件路径；
- ③通过A用户私钥签名，B用户公钥加密后保存为.gpg文件。

(3) 调阅功能

目标：A 用户调阅以.gpg 后缀结尾的文件。

步骤：

- ①输入待调阅文件；
- ②A用户用自己的私钥解密文件，用公钥库中的公钥对文件中的签名进行验证，如果验证成功，则输出文件及创建者（签名者）的身份。

1.2.3 初始状态

(1) 存储路径默认

输出文件：D:\OpenPGP_File_Manage_show\username\File

密钥相关：D:\OpenPGP_File_Manage_show\username\Key

keyring相关：D:\OpenPGP_File_Manage_show\username\Key\key-store

ps:其中username会据情况变化。

(2) 用户初始状态

均无相关权限，系统用户默认登陆。

1.3 组件关系及其调用

主要由user、gloabl、FileManage、KeyManage、ModeManage五个类组成，还包括库中含有的KeyMgr、OpenPGP等。

类名	调用类	函数/成员及功能
KeyManage	user	getUserID 获取 UserID 便于加密
	global	.pathStringKey 用于设置存储路径
	KeyMgr	CreateKey、SaveKeyring、ExportPublicKey 等函数用于生成保存密钥对
FileManage	OpenPGP	SetASCIIArmor、SetKeyCount、 SetKeyKeyring、.SetKeyUserId、SetKeyPassphrase、 SetRecipientKeyCount、SetRecipientKeyKeyring、 SetRecipientKeyId、SetSignerKeyCount、 SetSignerKeyKeyring、SetSignerKeyId、 SetInputMessage、GetOutputMessage、SetInputFile、 SetOutputFile、SignAndEncrypt、 DecryptAndVerifySignature 等函数，用于设置参数及完成签名加密、解密验证等功能
ModeManage	global	.pathString、.pathStringFile、.pathStringKey、.pathStringUser 用于设置各个路径，便于后续直接设置
	FileManage	SignAndEncryptString、SignAndEncryptSingle、 VerifySingle、SignAndEncryptMultiple、Verify 等函数实现字符串及文件签名加密、解密、验证的主要功能
	user	getUserID 获取 UserID 作为函数参数传递

1.4 概要与详细设计

1.4.1 组成模块

该程序由用户模块、全局文件存储模块、密钥管理模块、模式选择响应模块、文件/字符串处理模块组成。

1.4.2 各模块功能及函数声明约定

(1) 用户模块

功能：获取当前用户信息，储存用户名、用户安全标识符、授权用户ID，并提供获取相应值的函数。

对应函数声明如下：

```
void currentUserInfo();  
string getUsername();  
PSID getSid();  
LPWSTR getStringId();  
string getUserId();
```

(2) 全局文件存储模块

其中包含路径合并、创建文件夹、获取当前用户子功能。

对应函数声明如下：

```
void set_baseName();
string combineStrings(string left, string right); //返回合并后路径
int createDirectoryByString(string path);
string globalGetUserName(); //返回username
```

(3) 密钥管理模块

功能：生成密钥对，导出公钥，导出私钥，列出密钥情况（含userID、KeyID，但不含密钥具体信息）；其中还涉及密钥相关信息的初始化。

对应函数声明如下：

```
void GenerateKeyPairRSA(string userID, string pwd);
void ExportPublicKey(string userID, string pwd);
void ExportPrivateKey(string userID, string pwd);
void KeyStoreListKeys(string pwd);
void init(Global global, User user);
```

(4) 模式选择响应模块

功能：选择模式，处理对应模式。

对应函数声明如下：

```
int modeInput(Global global, User user); //返回选择模式对应数值
void modeControl(int mode, User user, string pwd);
```

(5) 文件/字符串处理模块

功能：签名加密字符串，解密验证字符串，单用户模式下对文件的签名加密，多用户模式下对文件的签名加密，单用户模式下身份真实性验证，多用户（普通）情况下真实性验证（含揭秘过程）

对应函数声明如下：

```
void SignAndEncryptString(string pwd, string userID, string keyringDir);
void DecryptAndVerifyString(string signedAndEncryptedMessage, string pwd,
string userID);
string SignAndEncryptSingle(string pwd, string filePath, string userID,
string pathStringFile, string keyringDir); //返回加密后文件的路径
string SignAndEncryptMultiple(string pwd, string filePath, string userID,
string pathStringFile, string keyringDir); //返回加密后文件的路径
bool VerifySingle(string pwd, string filePath, string userID, string
pathStringFile, string keyringDir); //返回身份认证是否通过
bool Verify(string pwd, string filePath, string userID, string
pathStringFile, string keyringDir); //返回身份认证是否通过
```



1.5 数据字典

1.5.1 类的成员

(1) user

```

class User
{
private:
    string username;      //用户名
    PSID sid;           //用户安全标识符
    LPWSTR stringSid;   //用户安全标识符, LPWSTR格式
    string userID; //当前用户ID
    vector<string> allUsers; //存授权ID, 含当前用户
    公共成员函数省略
}

```

(2) global

```

class Global
{
public:
    string baseName = "D:\\\\";
    string folderName = "OpenPGP_File_Manage_show";
    string folderName1 = "Key";
    string folderName2 = "File";
    string pathString; //总文件夹路径
    string pathStringUser; //用户文件夹路径
    string pathStringKey; //密钥存储文件夹路径
    string pathStringFile; //输出文件存储文件夹路径
//公共成员函数省略
}

```

(3) keyManage

```

class MyKeyMgr : public KeyMgr
class KeyManage
{
private:
    struct KeyGlobal {
        string pathString; //总文件夹路径
        string pathStringUser; //用户文件夹路径
        string pathStringKey; //密钥存储文件夹路径
        string pathStringFile; //输出文件存储文件夹路径
    }keyGlobal; //在KeyManage类中存储global相关信息
    struct KeyUser {
        string userID; //用户名
    }keyUser; //在KeyManage类中存储user相关信息
    MyKeyMgr keymgr; //用于调用库中与密钥相关函数
//公共成员函数省略
}

```

(4) ModeManage

```

class ModeManage
{
private:
    struct ModeGlobal {
        string pathString; //总文件夹路径
        string pathStringUser; //用户文件夹路径
        string pathStringKey; //密钥存储文件夹路径
        string pathStringFile; //输出文件存储文件夹路径
    }modeGlobal; //在ModeManage类中存储global相关信息
//公共成员函数省略
}

```

(5) FileManager

```
class FileManager
{
private:
    char* Output;//存储加密或解密后的字符串
public:
    struct ModeUserIDPwd {
        string modeUserID;//用户ID
        string modePwd;//用户密码
    }modeUserIDPwd;
//公共成员函数省略
}
```

1.5.2 常用变量

```
//下面三个存储均为用户ID
char privateKey[LINE_LEN];//操作用户
char recipientKey[LINE_LEN];//接收者
char signerKey[LINE_LEN];//签名者/创建者

//下面两个存用户密码
char passphrase[LINE_LEN];
string pwd

//下面两个存的是keyring存放路径
char mykeyringDir[LINE_LEN];
string keyringDir

//存部分库函数返回值便于出错处
int ret_code = 0;
```

1.6 编译手册

我们的源代码可在**Qt 5.12.0版本下编译**，这需要配置IPWorks OpenPGP库。

当然，**可以直接运行exe文件**进行测试。

1.7 安装手册

解压缩即可运行，详见“3.可执行安装包”。

1.8 自测用例、用户手册、测试分析报告、测试手册

详见“4.测试用例和分析报告”。

2. 源代码及注释

我们对重要的函数添加了相应的注释。

我们保持了较好的命名习惯。

源代码文件结构如下：

- ISP-03.pro
- Headers

- filemanage.h
- global.h
- keymanage.h
- logindialog.h
- main.h
- mainwindow.h
- modemanage.h
- user.h
- Sources
 - filemanage.cpp
 - global.cpp
 - keymanage.cpp
 - logindialog.cpp
 - main.cpp
 - mainwindow.cpp
 - modemanage.cpp
 - user.cpp
- Forms
 - logindialog.ui
 - mainwindow.ui
- Resources
 - covers.qrc
 - cover.png
 - mainbkg2.png

3. 可执行安装包

我们使用 Enigma Virtual Box，将我们的软件封装完毕！

直接解压，运行 ISP-03_boxed.exe 即可！

我们在**本仓库 release 文件夹下**也提供了**ISP-03_boxed.exe**，请下载整个文件夹运行，以免缺少 dll 文件。

输入参数请见“4. 自测用例及测试分析报告”文档。

4. 测试用例和分析报告

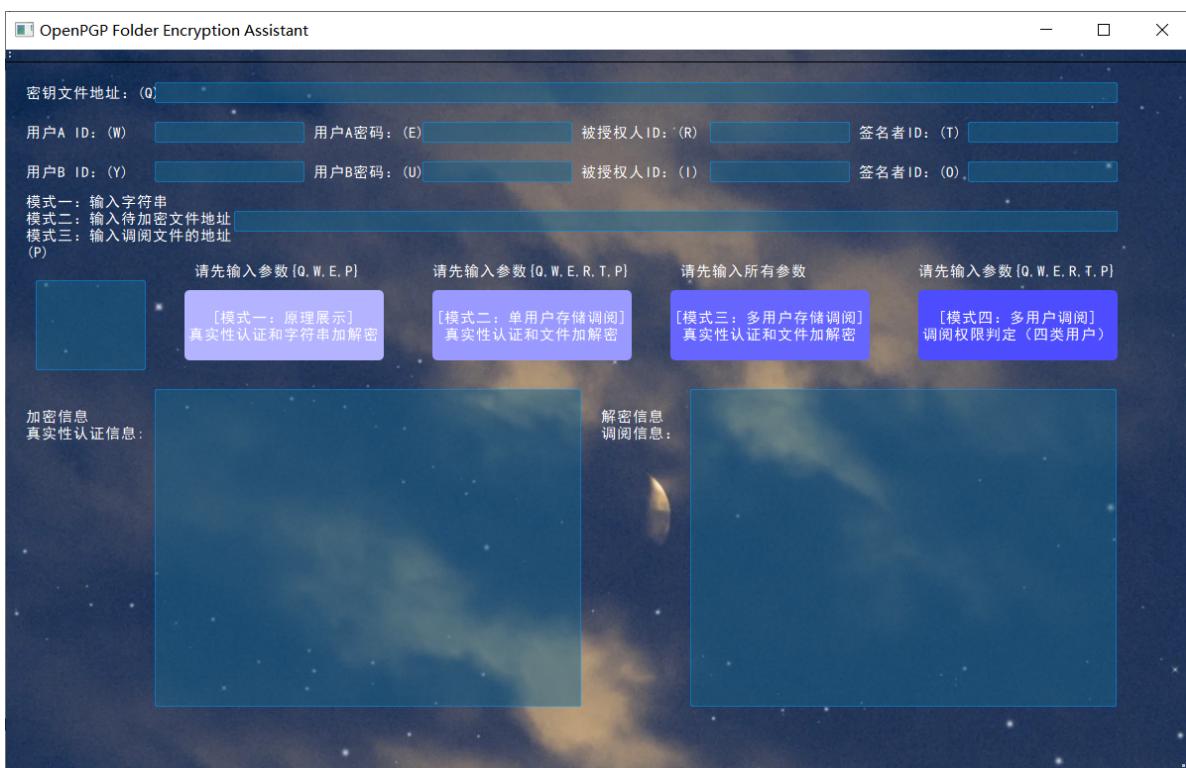
在此，一并展现参数的输入方法。

4.1 欢迎界面与主界面

欢迎界面设计：



主界面设计：



4.2 模式一、原理展示

进行真实性认证和字符串的加解密。

在输入了密钥文件地址、用户A输入了ID和密码以后，成功进行对字符串的加密和真实性认证，并进行解密。

分别输出在左右两侧。

(注：中间模式选择左边的小方块，展示了每一次运行，都会生成一个ID和密码对，可以使用此ID和密码进行检验。)



4.3 模式二、单用户存储调阅

进行单用户的真实性认证和对文件的加解密。

用户A输入自己的ID、密码和签名、授权信息以后，自己能够加密存储文件，并且进行真实性认证、调阅并解密自己的文件。



4.4 模式三、多用户存储调阅

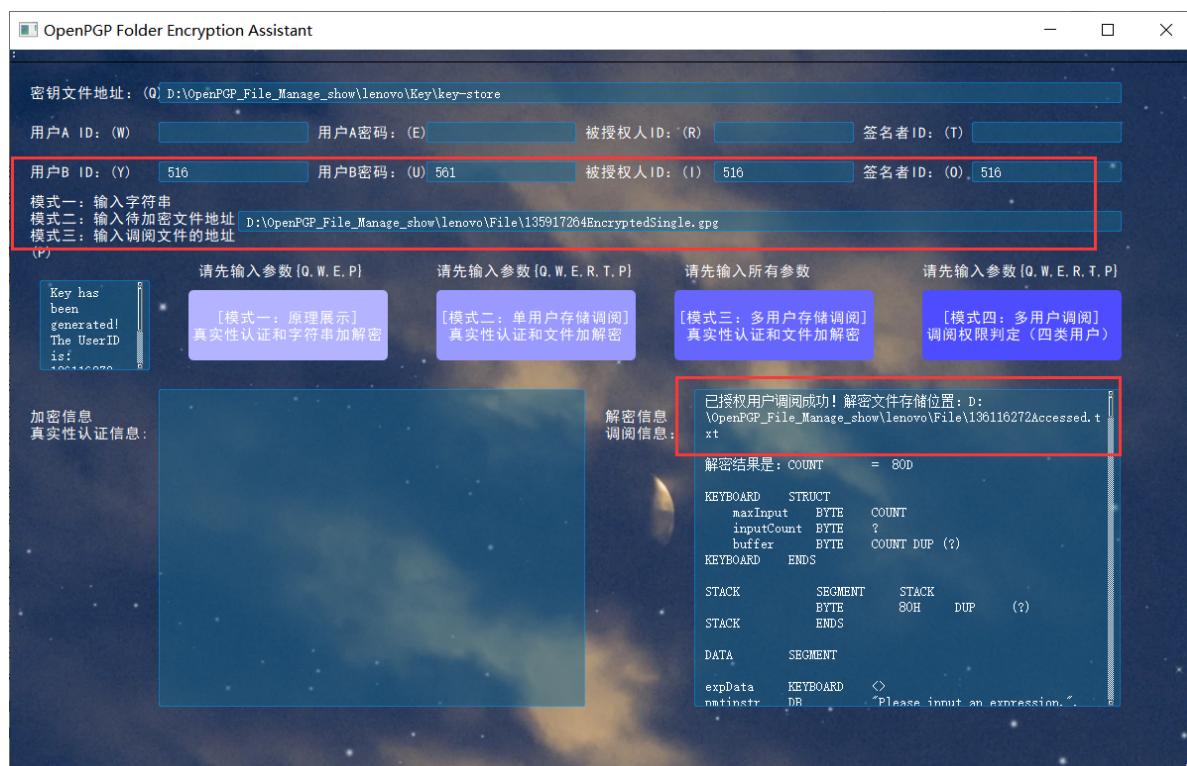
真实性认证和多用户文件加解密。

用户A能够授权给用户B，使用户B获取调阅权限，能够进行真实性认证和解密操作。

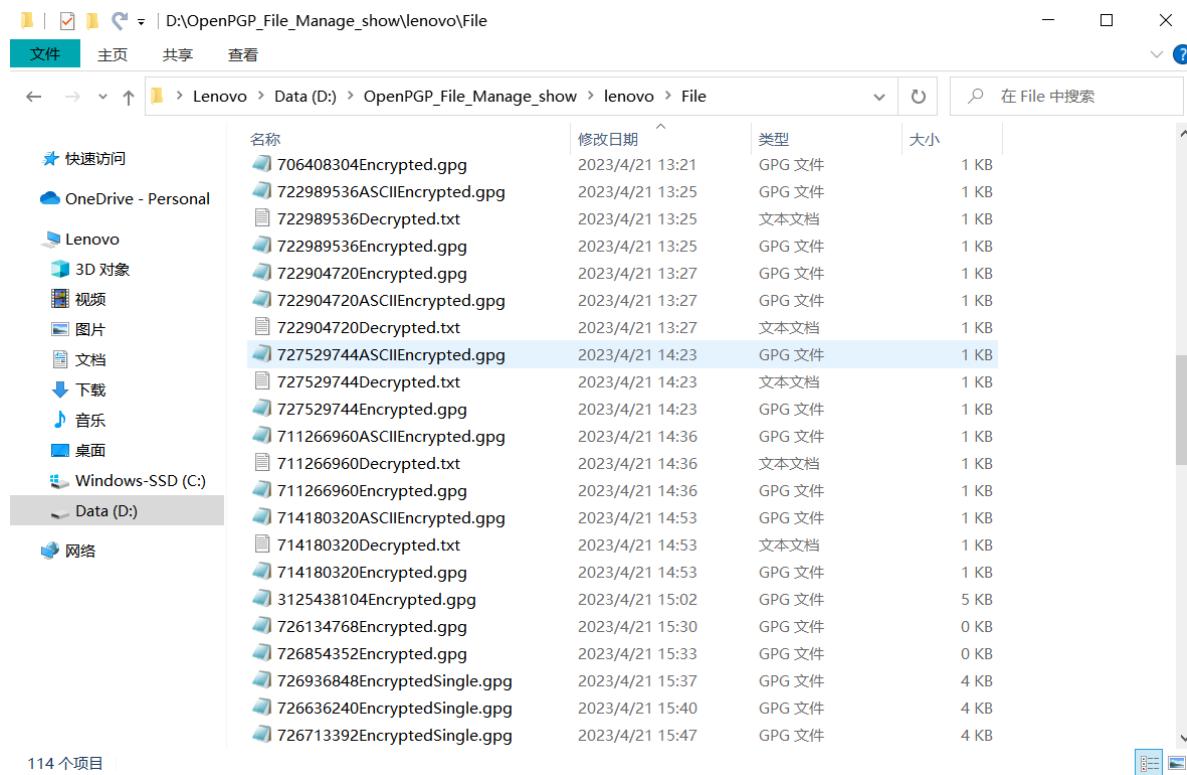


4.5 模式四、多用户调阅

有调阅权限用户能够调阅相应的文件。



4.6 文件的生成展示



4.7 敏感信息的覆盖核心代码

```
#include <iostream>
#include <windows.h>
#include <string>
#include <cstring>
#include <vector>
using namespace std;

int main()
{
    size_t size = inputfilepathEncryptSingle.length();
    std::vector<wchar_t> buffer(inputfilepathEncryptSingle.size() + 1);
    MultiByteToWideChar(CP_ACP, 0, inputfilepathEncryptSingle.c_str(), -1,
    buffer.data(), static_cast<int>(buffer.size()));

    // 打开文件句柄
    HANDLE hFile = CreateFile(buffer.data(), GENERIC_READ, FILE_SHARE_READ,
    NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if (hFile == INVALID_HANDLE_VALUE) {
        std::cerr << "Failed to open file." << std::endl;
    }

    // 创建文件映射对象
    HANDLE hMapFile = CreateFileMapping(hFile, NULL, PAGE_READONLY, 0, 0, NULL);
    if (hMapFile == NULL) {
        std::cerr << "Failed to create file mapping object." << std::endl;
        closeHandle(hFile);
    }

    // 映射文件到进程地址空间
    LPVOID lpMapView = MapViewOfFile(hMapFile, FILE_MAP_READ, 0, 0, 0);
    if (lpMapView == NULL) {
```

```
    std::cerr << "Failed to map view of file." << std::endl;
    CloseHandle(hMapView);
    CloseHandle(hFile);
}

// 从内存中读取文件
char* pBuffer = static_cast<char*>(lpMapView);
std::cout << "Content of the file:" << std::endl;
std::cout << pBuffer << std::endl;

// 覆盖文件映射的地址空间
SecureZeroMemory(lpMapView, GetFileSize(hFile, NULL));

// 关闭文件映射
UnmapViewOfFile(lpMapView);
CloseHandle(hMapView);

// 关闭文件句柄
CloseHandle(hFile);

std::cout << "Succeeded in overwriting the mapped view of the file!" <<
std::endl;
return 0;
}
```

5. 第三方资源及其说明

5.1 原创性说明

我们的代码是逐行编写的，这保证了**100%的原创性**（包括UI的配色方案在内）。

我们的代码开源在：<https://github.com/ChestnutSilver/OpenPGP-FileEncrypt-Assistant>

此仓库一并体现了我们**十一次版本迭代过程**（其中，版本1.3、1.4、1.5、1.6、1.7、1.8、2.0、3.0）均可查看。

5.2 网络教程学习说明

在代码的实现过程中，阅读并学习了网络资源：

<http://yuanxinhang.fun/2021/12/11/%E5%9F%BA%E4%BA%8EopenPGP%E5%AE%9E%E7%8E%B0%E6%9C%AC%E5%9C%B0%E5%8A%A0%E5%AF%86%E6%96%87%E4%BB%B6/>

这个网络教程帮助我们理清了实现思路，同时，我们学习了此资源的代码结构。

5.3 第三方库使用说明

我们使用了 IPWorks OpenPGP 库。

我们使用了 Qt 相关库。