

实验二：密码技术

1. 实验目的

- 掌握 OpenSSL 的命令；
- 掌握在 C 程序中使用 OpenSSL 的方法；
- 掌握 PGP 的使用。

2. 实验内容

- 使用 OpenSSL 的常用命令；
- 利用 OpenSSL 编程实现 AES 加密、解密；
- 用 PGP 实现加密和解密。

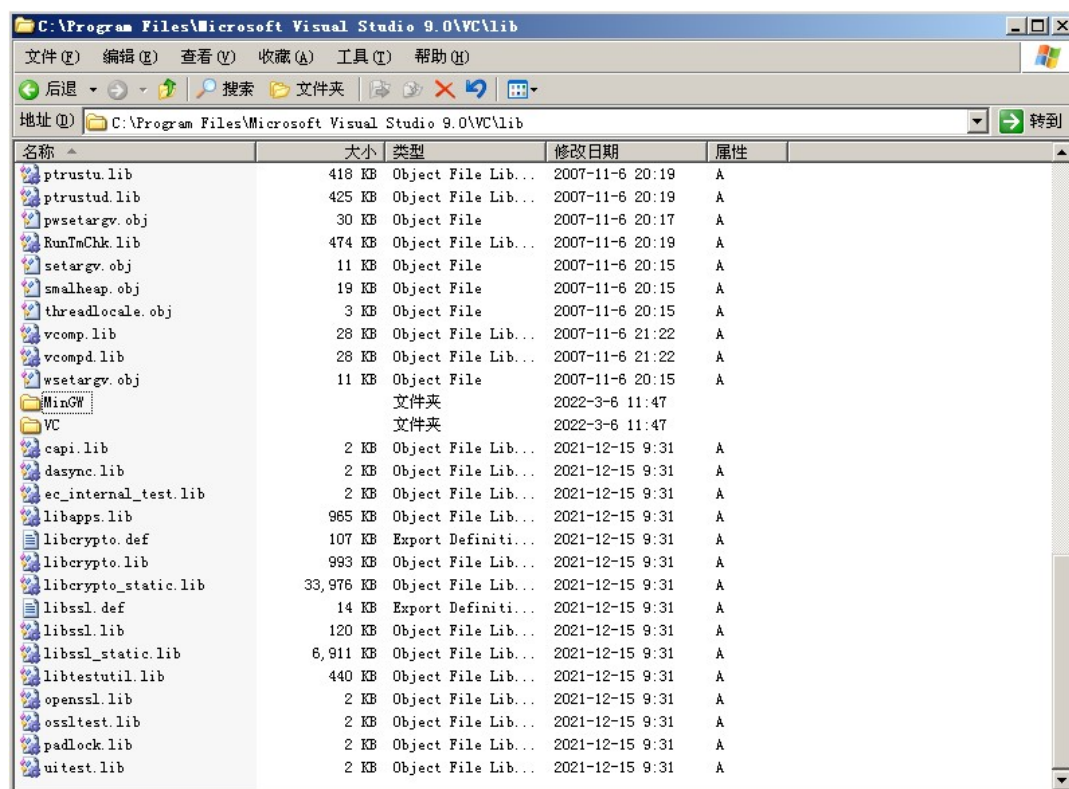
3. 实验步骤

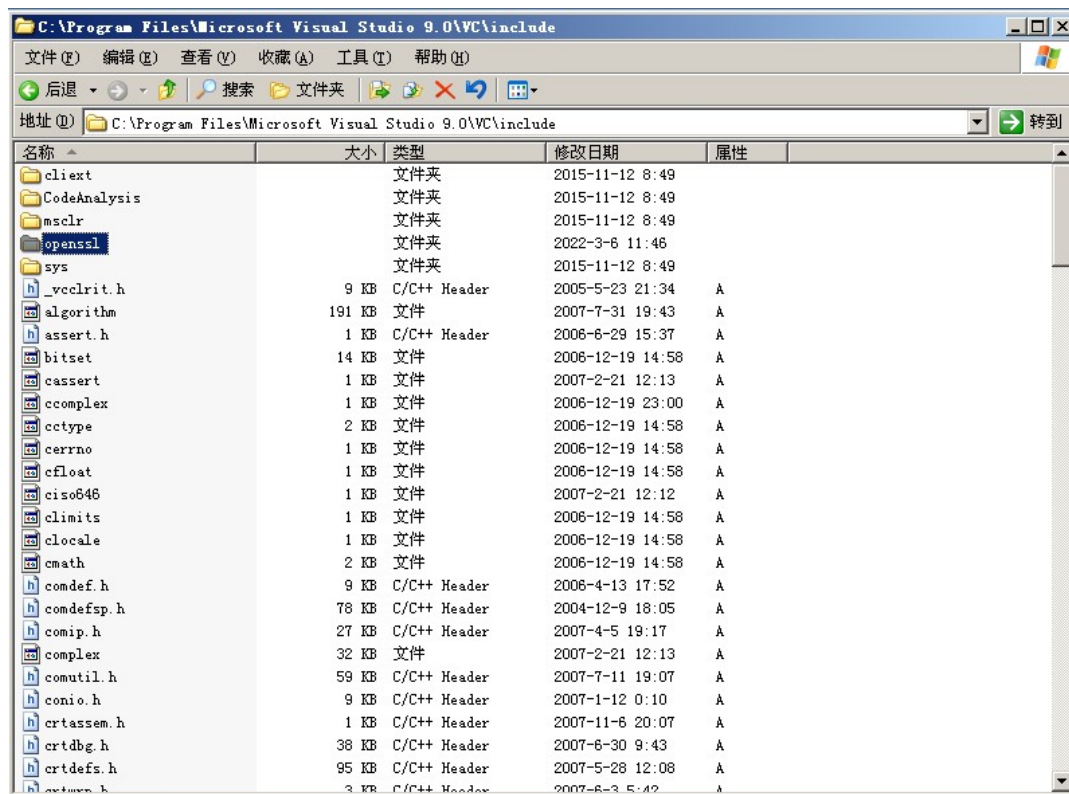
从课程网站下载 `cryptoDemo.zip`，从课程网站指定的链接下载 `openssl`，`win32openssl` 和 `pgp4win` 到目录 `C:\work\ms\chapter03`。

3.1 OpenSSL的使用

3.1.1 使用OpenSSL的常用命令

点击 `win32openssl-1_0_1i` 对应的安装文件，安装完后可以看到目录 `C:\openssl-win32`，其中包含了 OpenSSL 的各类文件。将 `C:\openssl-win32\bin` 添加到环境变量 `path` 中，将 `C:\openssl-win32` 下的 `include` 和 `lib` 目录拷贝到 `C:\Program Files\Microsoft Visual Studio 9.0\VC` 中。





参考第 2 章 PPT 的“在命令行下使用 OpenSSL的相关内容，测试其中的相关命令，如下图所示：



3.1.2 利用 OpenSSL 编程实现 AES 的加密、解密

OpenSSL 提供了对 AES 的支持，实例程序 cryptoDemo.cpp 实现了对字符串的加密和解密。启动 Visual Studio 2008 Command Prompt 编译环境。编译和运行 cryptoDemo.cpp，如下图所示：

```
Visual Studio 2008 Command Prompt
C:\work\chapter03>cl cryptoDemo.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation. All rights reserved.

cryptoDemo.cpp
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation. All rights reserved.

/out:cryptoDemo.exe
cryptoDemo.obj

C:\work\chapter03>cryptoDemo.exe
test success
The original string is:
This is a sample. I am a programmer.
The encrypted string is:
?/桊 ?8析?96C+m磕
+!=!◎}?曾? The decrypted string is:
This is a sample. I am a programmer.

C:\work\chapter03>
```

修改例程 `cryptoDemo.cpp` 为 `encfile.cpp`：从命令行接受 3 个字符串类型的参数：参数 1，参数 2，参数 3。参数 1=enc 表示加密，参数 1=dec 表示解密；参数 2 为待加密、解密的文件名；参数 3 为密码。以文件 `cryptoDemo.cpp` 为测试文件，以你的学号为密码，验证你的程序 `encfile.cpp` 的正确性。

- 设计思路

该程序建立在 `cryptoDemo.cpp` 的基础上，对 `cryptoDemo.cpp` 已经实现的功能进行分割，根据要求，经过分析主要涉及以下要点

- 命令行参数

根据C语言语法和每一个参数的功能要求，就可以实现命令行参数的输入。

- 文件的读写

利用C语言对文件的读写操作，就可以轻松实现对源文件的读取，对加密文件的写入，对待解密文件的读取和对解密文件的写入。需要注意的是，在对源文件的读取和对解密文件的写入时的读写模式分别为 `r` 和 `w`。而对加密文件的写入和对待解密文件的读取则需要设置为二进制模式，分别为 `rb` 和 `wb`。

- 程序设计

```
#include <memory.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define N 2500

#include "openssl\aes.h"

#pragma comment(lib,"libcrypto.lib")

void AES_encrypt(char filename[], char password[])
{
    int i, j, len, nLoop, nRes, pwdlen, inLen, enlen;
    char inString[N];
    char enString[N];
    FILE* fp;
```

```

unsigned char buf[16];
unsigned char buf2[16];
unsigned char aes_keybuf[32];
AES_KEY aeskey;

// 准备32字节(256位)的AES密码字节
pwdlen = strlen(password);
memset(aes_keybuf, 0x90, 32);
if (pwdlen < 32)
    len = pwdlen;
else
    len = 32;
for (i = 0; i < len; i++)
    aes_keybuf[i] = password[i];
AES_set_encrypt_key(aes_keybuf, 256, &aeskey);
//读入需要加密的文件
if ((fp = fopen(filename, "r")) == NULL)
{
    printf("Source File Open Error\n");
    return;
}
else
    printf("Source File Open Success\n");
memset(inString, 0, N);
memset(enString, 0, N);
fread(inString, sizeof(char), N, fp);
fclose(fp);

inLen = strlen(inString);
nLoop = inLen / 16;
nRes = inLen % 16;
// 加密输入的字节串
for (i = 0; i < nLoop; i++)
{
    memset(buf, 0, 16);
    for (j = 0; j < 16; j++)
        buf[j] = inString[i * 16 + j];
    AES_encrypt(buf, buf2, &aeskey);
    for (j = 0; j < 16; j++)
        enString[i * 16 + j] = buf2[j];
}
if (nRes > 0)
{
    memset(buf, 0, 16);
    for (j = 0; j < nRes; j++)
        buf[j] = inString[i * 16 + j];
    AES_encrypt(buf, buf2, &aeskey);
    for (j = 0; j < 16; j++)
        enString[i * 16 + j] = buf2[j];
}
enString[i * 16 + j] = 0;

//将加密结果输出
if ((fp = fopen("ciphertext.out", "wb")) == NULL)
{
    printf("Ciphertext File Open Error\n");
    return;
}

```

```

else
    printf("Ciphertext File Open Success\n");
fwrite(enString, sizeof(char), strlen(inString), fp);
fclose(fp);
printf("Encrypt Success!");
}

void AES_decrypt(char filename[], char password[])
{
    int i, j, len, nLoop, nRes, pwdlen, inLen;
    char inString[N];
    char deString[N];
    FILE* fp;
    unsigned char buf[16];
    unsigned char buf2[16];
    unsigned char aes_keybuf[32];
    AES_KEY aeskey;

    // 准备32字节(256位)的AES密码字节
    pwdlen = strlen(password);
    memset(aes_keybuf, 0x90, 32);
    if (pwdlen < 32)
        len = pwdlen;
    else
        len = 32;
    for (i = 0; i < len; i++)
        aes_keybuf[i] = password[i];
    AES_set_decrypt_key(aes_keybuf, 256, &aeskey);
    // 读入需要解密的文件
    if ((fp = fopen(filename, "rb")) == NULL)
    {
        printf("Source File Open Error\n");
        return;
    }
    else
        printf("Source File Open Success\n");
    memset(inString, 0, N);
    fread(inString, sizeof(char), N, fp);

    inLen = 0;
    for (i = 0; i < N; i++)
    {
        if (inString[i] == 0 && inString[i + 1] == 0)
            break;
        else
            inLen++;
    }

    nLoop = inLen / 16;
    nRes = inLen % 16;
    // 解密输入的字节串
    for (i = 0; i < nLoop; i++)
    {
        memset(buf, 0, 16);
        for (j = 0; j < 16; j++)
            buf[j] = inString[i * 16 + j];
        AES_decrypt(buf, buf2, &aeskey);
        for (j = 0; j < 16; j++)

```

```

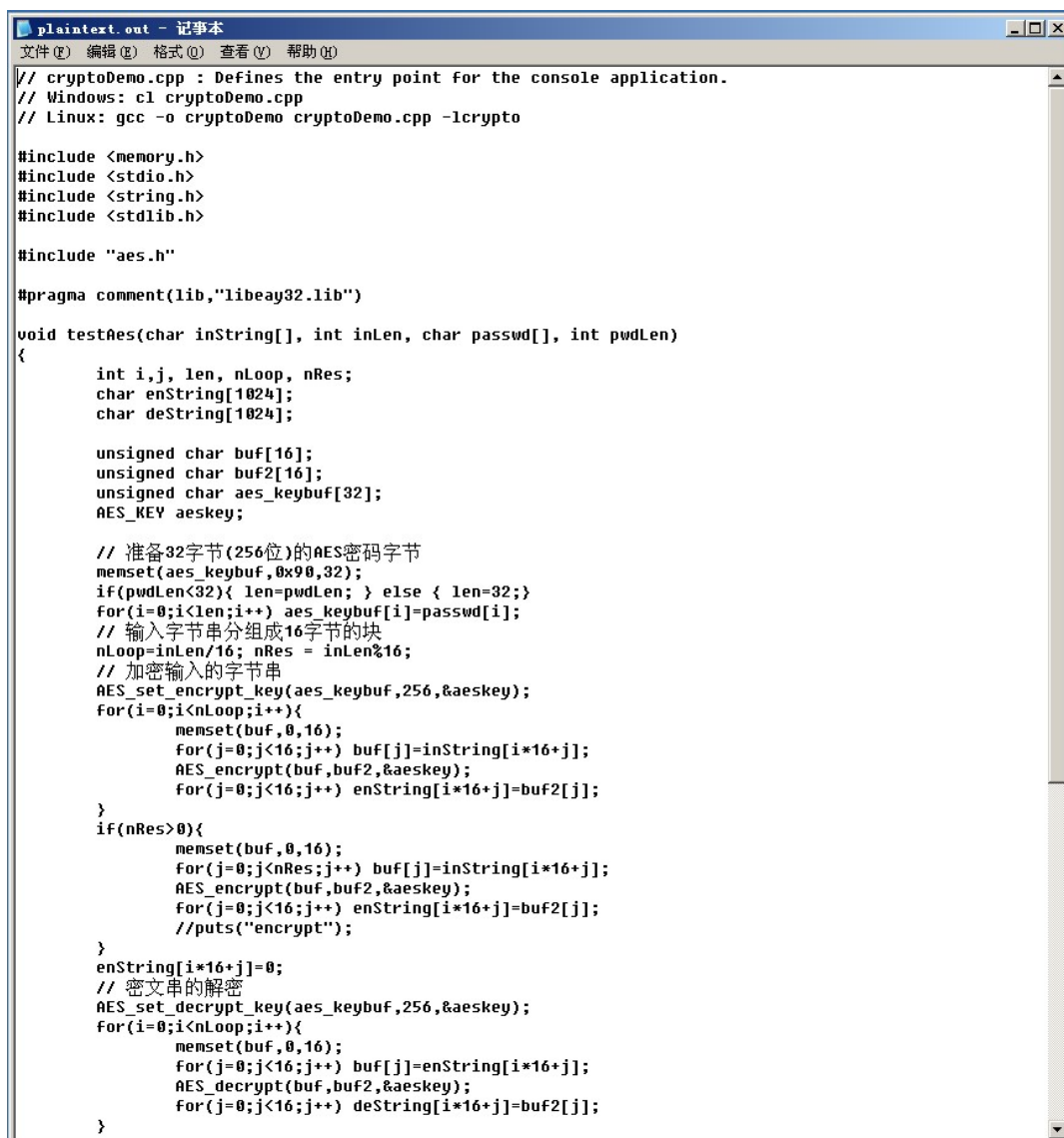
        deString[i * 16 + j] = buf2[j];
    }
    if (nRes > 0)
    {
        memset(buf, 0, 16);
        for (j = 0; j < 16; j++)
            buf[j] = inString[i * 16 + j];
        AES_decrypt(buf, buf2, &aeskey);
        for (j = 0; j < 16; j++)
            deString[i * 16 + j] = buf2[j];
    }
    deString[i * 16 + nRes] = 0;
    fclose(fp);
    //将解密结果输出
    if ((fp = fopen("plaintext.out", "w")) == NULL)
    {
        printf("Plaintext File Open Error\n");
        return;
    }
    else
        printf("Plaintext File Open Success\n");
    fwrite(deString, sizeof(char), inLen, fp);
    fclose(fp);
    printf("Decrypt Success!\n");
}

int main(int argc, char* argv[])
{
    if (memcmp(argv[1], "enc", 3) == 0)
        AES_encrypt(argv[2], argv[3]);
    else
    {
        if (memcmp(argv[1], "dec", 3) == 0)
            AES_decrypt(argv[2], argv[3]);
        else
            printf("Mode Error!\n");
    }

    return 0;
}

```

- 实验结果
 - 编译并运行



```
// cryptoDemo.cpp : Defines the entry point for the console application.
// Windows: cl cryptoDemo.cpp
// Linux: gcc -o cryptoDemo cryptoDemo.cpp -lcrypto

#include <memory.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "aes.h"

#pragma comment(lib, "libeay32.lib")

void testAes(char inString[], int inLen, char passwd[], int pwdLen)
{
    int i, j, len, nLoop, nRes;
    char enString[1024];
    char deString[1024];

    unsigned char buf[16];
    unsigned char buf2[16];
    unsigned char aes_keybuf[32];
    AES_KEY aeskey;

    // 准备32字节(256位)的AES密码字节
    memset(aes_keybuf, 0x90, 32);
    if(pwdLen < 32){ len=pwdLen; } else { len=32; }
    for(i=0; i<len; i++) aes_keybuf[i]=passwd[i];
    // 输入字节串分组成16字节的块
    nLoop=inLen/16; nRes = inLen%16;
    // 加密输入的字节串
    AES_set_encrypt_key(aes_keybuf, 256, &aeskey);
    for(i=0; i<nLoop; i++){
        memset(buf, 0, 16);
        for(j=0; j<16; j++) buf[j]=inString[i*16+j];
        AES_encrypt(buf, buf2, &aeskey);
        for(j=0; j<16; j++) enString[i*16+j]=buf2[j];
    }
    if(nRes > 0){
        memset(buf, 0, 16);
        for(j=0; j<nRes; j++) buf[j]=inString[i*16+j];
        AES_encrypt(buf, buf2, &aeskey);
        for(j=0; j<16; j++) enString[i*16+j]=buf2[j];
        //puts("encrypt");
    }
    enString[i*16+j]=0;
    // 密文串的解密
    AES_set_decrypt_key(aes_keybuf, 256, &aeskey);
    for(i=0; i<nLoop; i++){
        memset(buf, 0, 16);
        for(j=0; j<16; j++) buf[j]=enString[i*16+j];
        AES_decrypt(buf, buf2, &aeskey);
        for(j=0; j<16; j++) deString[i*16+j]=buf2[j];
    }
}
```

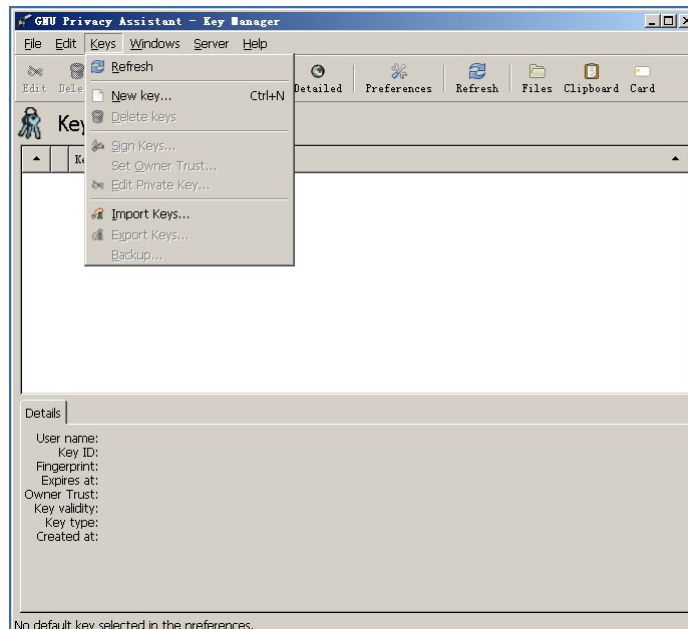
说明程序无误。

3.2 PGP实现加密和解密

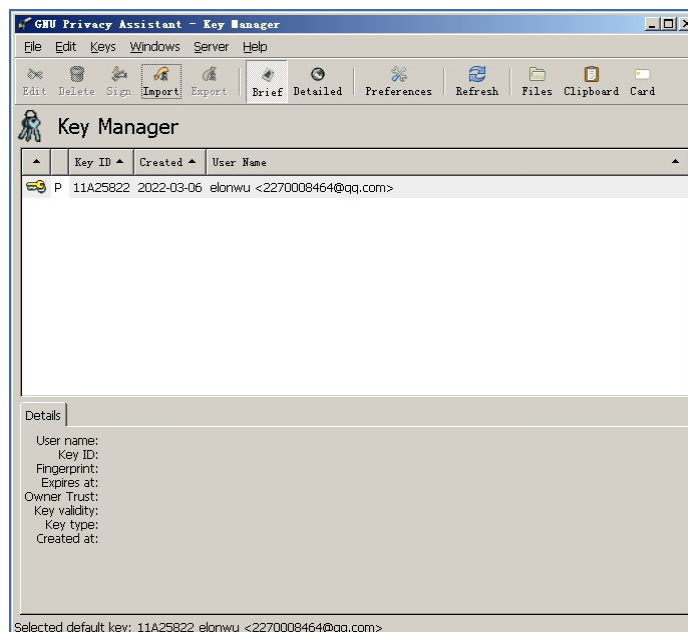
安装 `Gpg4win2.2.3`，选择安装所有的组件，安装结束后认真阅读 `README.en` 文件。按以下步骤使用加密和解密功能。

3.2.1 产生一对 RSA 密钥

启动 GPA，产生一对密钥，如下图所示。



输入用户名以及电子邮箱，接下来输入 passphrase，然后就会生成 2048 位的公钥/私钥对（注意：可能要等待一段时间）。如果成功生成了密钥，则会显示在列表中。如下图所示：

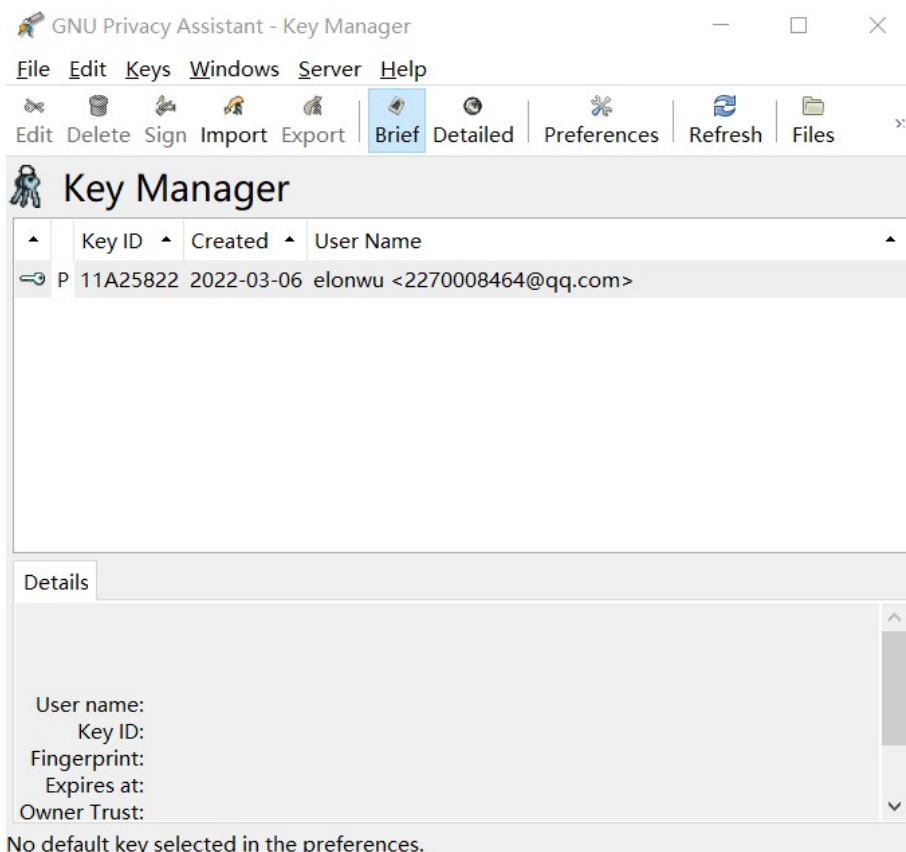


3.2.2 互换公钥

将公钥导出(Export Keys)到一个文件中(文件名为 `Pkeys.key`)，传递给需要给自己发送加密文件的电脑。

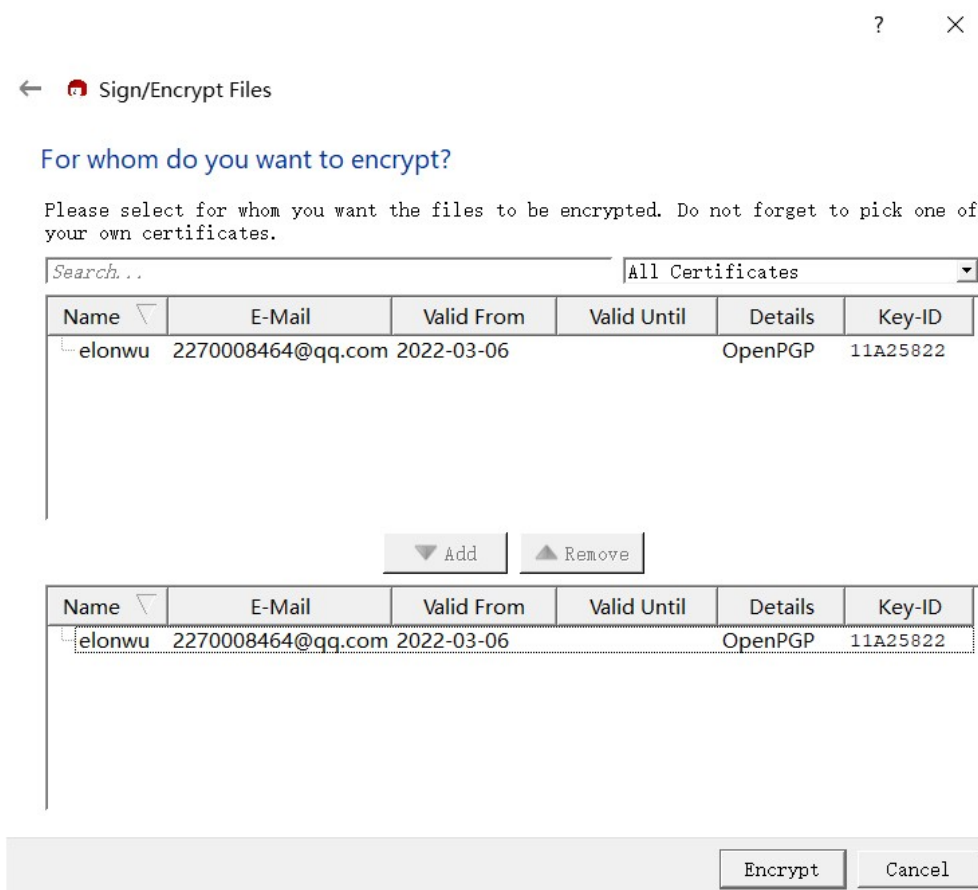


对方收到公钥文件(`Pkeys.key`)后，将公钥导入到本机。如果导入成功，将在本机的 GPA 中列出该公钥。如图所示导入了 ID 号为 `11A25822`、邮件地址为 `2270008464@qq.com` 的公钥。



3.2.3 向对方发送加密文件

启动资源管理器，选择要加密的文件，选择 **Sign and encrypt**。选择要接收该加密文件的用户 (与公钥对应的私钥持有者)：



点击 **Encrypt** 按钮将加密指定的文件，得到扩展名为 **gpg** 的加密文件，将该文件发送给私钥持有者。

Results

Status and progress of the crypto operations is shown here.

OpenPGP: All operations completed.

encfile.cpp → encfile.cpp.gpg: **Encryption succeeded.**

[Show Details](#)

☒ Keep open after operation completed

Finish

Cancel

encfile.cpp.gpg

2022/3/6 15:31

GPG 文件

2 KB

私钥持有者对其解密(需要输入 passphrase)后可以恢复出原文件。

