应用密码学实践

实验报告

课　　程 应用密码学实践

实验题目 RSA 实验

学生姓名 魏来　武玉豪

学　　号 2017141451031　2017141411140

实验时间 4 月 8 日

信息安全专业实验室

一、 实验内容描述

本实验利用 c 语言编写程序实现 RSA 算法的密钥生成、数据加密和数据解密。

实验要求生成不小于 10^10 的大素数 p、q，并且所有与 RSA 算法相关的数据都要以十六进制字符串的形式存储在相应的文件中。

实验程序编写完成后，首先对给出的测试数据进行加密解密，以验证程序的正确性，然后将实验小组成员的姓名学号以学号 1-姓名 1；学号 2-姓名 2； ……的形式写入明文文件进行加密。

二、 实验环境描述

根据实验要求，本小组采用 C 语言编写程序。

由于 RSA 算法的对密钥长度有一定的要求，因此需要能够对一定位数的大数进行运算，包括基本的加、减、乘、除、模，以及求最大公因数，求乘法逆元等。为此,本小组在 windows 下搭建好了 gmp 环境，在 powershell 中利用 gcc 编译并调试程序，来完成各项实验要求。

gmp 的全称是 GNU Multiple Precision Arithmetic Library，即 GNU 高精度算术运算库，它的功能非常强大，接口很简单，文档详尽，有 C 风格的接口也有 C++的精心封装后的接口，其中不但有普通的整数、实数、浮点数的高精度运算，还有随机数生成，尤其是提供了非常完备的数论中的运算接口，比如 Miller-Rabin 素数测试算法，大素数生成，欧几里德算法，求域中元素的逆，Jacobi 符号，legendre 符号等。它本身提供了很多例子程序，学习过程非常快，很容易将它们集成到自己的代码中去。

三、 实验过程简介

本程序分为 file.h、crypt.h、initial.h、rsa.c,其中 file.h 封装文件的读写、命令行等功能；crypt.h 实现加密和解密；initial.h 实现 rsa 算法的参数初始化；rsa.c 是程序运行的源文件

1 密钥的生成

1.1 生成大素数 p、q:

利 gmp 库，我们可以轻易的声明一个 mpz_t 型的大数。gmp 中有用于产生随机数的函数，直接调用即可。实验说明中要求产生的数大于 10^10，因此我们程序中生成的素数 p、q 位数为 64 位

```
41      mpz_urandomb(keyp, rand, 64);
42      mpz_urandomb(keyq, rand, 64);
43      mpz_t temp;
44      mpz_init_set_str(temp,"10000000000",10);
45      mpz_add(keyp,keyp,temp);
46      mpz_add(keyq,keyp,temp);
47
```

图 1 部分代码

代码中的 mpz_urandomb()函数随机产生一个 0-2^n-1 位的整数，然后将 p、q 加上一个 10 位的十进制整数“1000000000”，确保产生的密钥长度满足实验要求。

Function: *void* **mpz_urandomb** *(mpz_t rop, gmp_randstate_t state, mp_bitcnt_t n)*

Generate a uniformly distributed random integer in the range 0 to *2^n-1*, inclusive.

The variable *state* must be initialized by calling one of the `gmp_randinit` functions (Random State Initialization) before invoking this function.

图 2 mpz_urandomb()函数的定义

但此时 p、q 还不是素数，这时利用 mpz_nextprime（）函数，将 p、q 的值更新为素数。

Function: *void* **mpz_nextprime** *(mpz_t rop, const mpz_t op)*

Set *rop* to the next prime greater than *op*.

This function uses a probabilistic algorithm to identify primes. For practical purposes it's adequate, the chance of a composite passing will be extremely small.

图 3 mpz_nextprime（）函数的定义，

产生一个距离 op 最近的素数然后赋值给 rop

此时完成了 p、q 的初始化，接下来进行公钥和私钥的计算。

1.2. 生成公钥及私钥

公钥 n = q * p；利用 gmp 库中的加法函数 mpz_mul()，可以直接计算得到 n。私钥 Φ（n）=(p-1)*(q-1)，也可以轻易的计算得到。另一公钥 e 的生成，利用 create_e（）函数产生。

create_e（）函数内部，先调用随机数生成函数，将 mpz_t 类型的 e0 初始化为 20 位整数，然后判断是否 e0 与 Φ（n）互素，如果互素，则将 e0 返回，作为公钥使用；不互素则将 e0+1，直至 e0 与 Φ（n）互素。

求得 e 以后直接调用 gmp 库中的函数 mpz_invert（），即可求得另一私钥 d。

到此，完成了 RSA 算法全部密钥的初始化。

根据实验要求，调用函数 mpz_out_str()，将产生的这些数据以十六进制字符串的形式分别存入相应的文件。

Function: *size_t* **mpz_out_str** *(FILE *stream, int base, const mpz_t op)*

Output *op* on stdio stream *stream*, as a string of digits in base *base*. The base argument may vary from 2 to 62 or from -2 to -36.

For *base* in the range 2..36, digits and lower-case letters are used; for -2..-36, digits and upper-case letters are used; for 37..62, digits, upper-case letters, and lower-case letters (in that significance order) are used.

Return the number of bytes written, or if an error occurred, return 0.

图 4 函数 mpz_out_str()的定义

2 数据加密

根据初始化的情况，依次打开公钥和明文的文件，将其中的内容读出，然后调用函数 mpz_powz()加密，然后将加密的结果写入密文文件，同样，格式为 16 进制字符串。

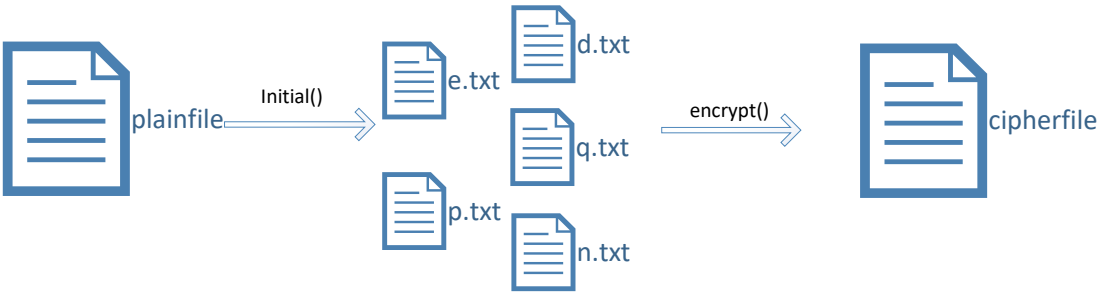


图 5 加密流程图

3 数据解密

依次打开私钥和密文的文件，将其中的内容读出，然后调用函数 mpz_powz()解密，然后将加密的结果写入明文文件，同样，格式为 16 进制字符串。

图 6 解密流程图

3.1 命令行

实验要求以命令行的形式，指定明文文件、密钥文件的位置和名称以及加密完成后密文文件的位置和名称。加密时先分别从指定的明文文件、密钥文件中读取有关信息，然后进行加密，最后将密文写入指定的密文文件。

因此我们采用 unsitd.h 中的 getopt()函数来对控制台输入的命令行做出相应的反应。给定了命令参数的数量（argc)、指向这些参数的数组(argv）和选项字符串（optstring）后，getopt() 将返回第一个选项，并设置一些全局变量。使用相同的参数再次调用该函数时，它将返回下一个选项，并设置相应的全局变量。如果不再有识别到的选项，将返回 -1，此任务就完成了。

getopt() 所设置的全局变量包括：

optarg——指向当前选项参数（如果有）的指针。

optind——再次调用 getopt() 时的下一个 argv 指针的索引。

optopt——最后一个已知选项。

在本实验中参数-n、-p、-e 是必选参数，程序运行时必须指定明文文件，存放整数 n 的文件，密文所在文件；然后根据加密或者解密的需求指定存放公钥或者私钥的文件。

当参数数量不为 4 个或者同时键入了-d -e 命令时，程序会返回一个报错信息。

四、 实验与测试结果

测试结果

明文：(16 进制表示，明文文件为：rsa_plain.txt)

63727970746F677261706879

公钥：(16 进制表示，公钥文件为：n.txt 和 e.txt)

n=73299B42DBD959CDB3FB176BD1

e=10001

私钥：(16 进制表示，私钥文件为：n.txt 和 d.txt)

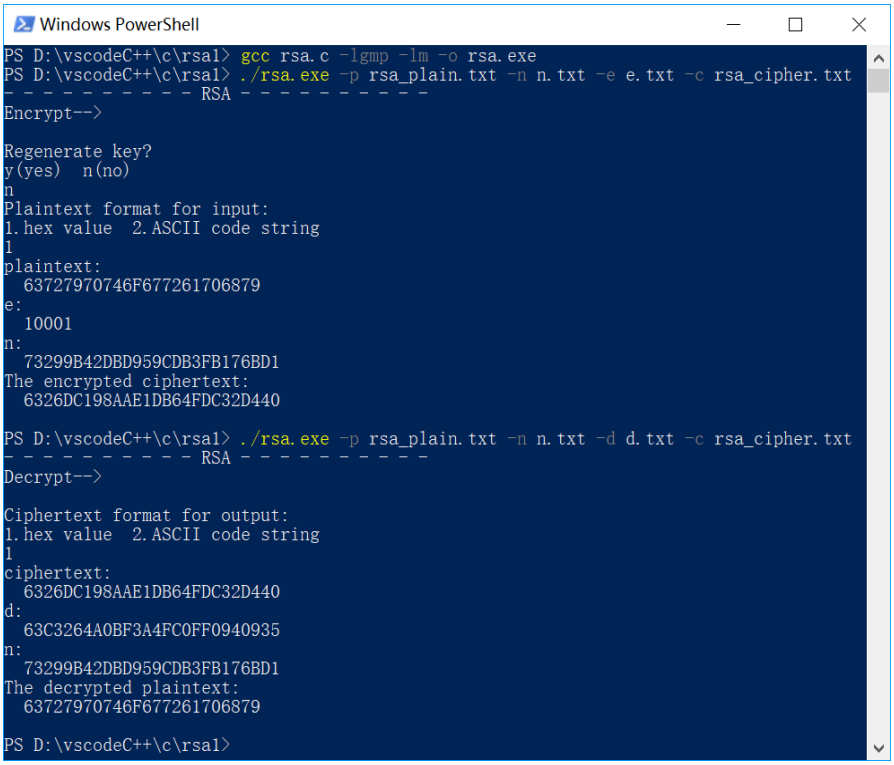
n=73299B42DBD959CDB3FB176BD1

d=63C3264A0BF3A4FC0FF0940935

密文：(16 进制表示，密文文件为：rsa_ cipher.txt)

6326DC198AAE1DB64FDC32D440

加密与解密过程：



实验结果

明文：(16 进制表示，明文文件为：rsa_plain.txt)

2017141411140-wuyuhao;2017141451031-weilai

公钥：(16 进制表示，公钥文件为：n.txt 和 e.txt)

n=307ABECB5D12F1295A2EC9FFD1C73BE189DD88809D07B

D741FDE7AEC410C8FA536736E62A81661E1E358C27FB57ED355

F032A313809CC767187761B84C008749

e = 6413F

私钥：(16 进制表示，私钥文件为：n.txt 和 d.txt)

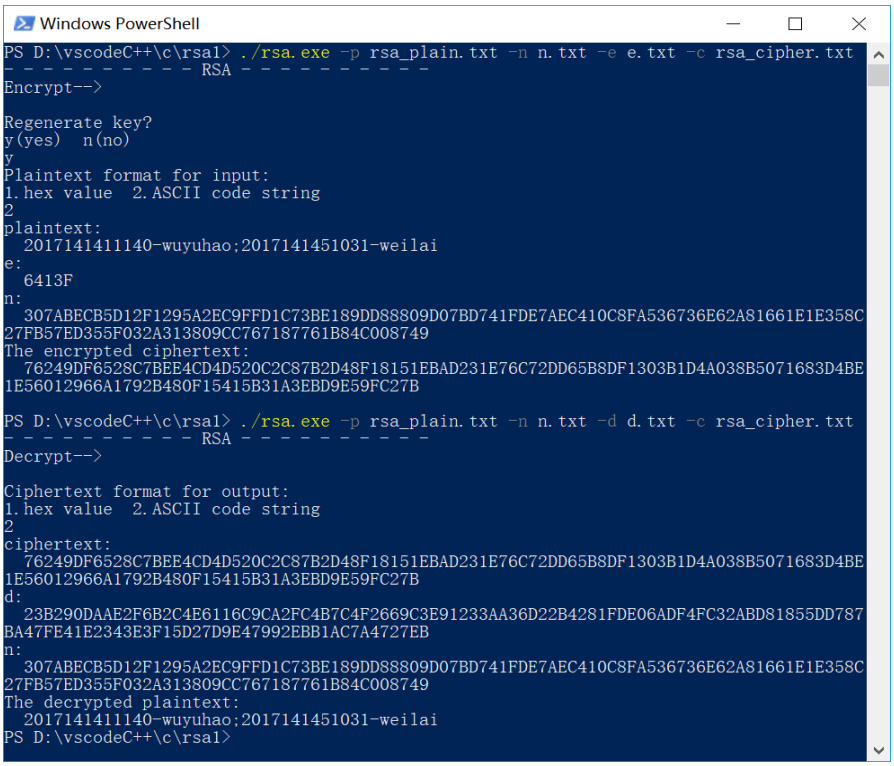
n=307ABECB5D12F1295A2EC9FFD1C73BE189DD88809D07B
D741FDE7AEC410C8FA536736E62A81661E1E358C27FB57ED355
F032A313809CC767187761B84C008749

d=23B290DAAE2F6B2C4E6116C9CA2FC4B7C4F2669C3E9123
3AA36D22B4281FDE06ADF4FC32ABD81855DD787BA47FE41E23
43E3F15D27D9E47992EBB1AC7A4727EB

密文：(16 进制表示，密文文件为：rsa_cipher.txt)

76249DF6528C7BEE4CD4D520C2C87B2D48F18151EBAD231E
76C72DD65B8DF1303B1D4A038B5071683D4BE1E56012966A17
92B480F15415B31A3EBD9E59FC27B

加密与解密过程：



五、 实验的收获和体会

通过本次实验，本小组成员加深了对密码学在实践中应用的理解，并对 RSA 密码体制有了更进一步的掌握。

在实验过程中，我们遇到了很多困难，比如 C 语言中大数运算的解决方法、对明文消息的处理、素数的检测方法、gmp 大数库的使用、对文本的处理等；我们通过对 RSA 密码体制的深入学习和研究，以及对代码的不断调试和优化，将这些困难一一克服。

最终，我们完成了实验的各项要求。在这个过程中，我们不仅仅学会利用书本中的理论知识，还加深了理解，进一步提升了自己的编程能力与数据处理的能力。