华南理工大学

《PKI 原理与技术》课程实验报告

实验题目：C/C++实现 RSA 密钥对的生成及数字签名

姓名：_____ 学号：_____

班级：_____ 组别：_____

合作者：_____

指导教师：徐玲玲

实验概述

【实验目的及要求】

实验目的：

熟练掌握密钥对的生成以及利用密钥对做数字签名等操作。

实验要求：

利用 OpenSSL 提供的库函数，用 C/C++编写：

1. 编写程序生成 RSA 密钥对，并保存公钥到文件。
2. 对某个文件 Hash 后计算其数字签名，并把得到的签名信息保存到文件。
3. 利用公钥对所生成的数字签名进行验证。

【实验环境】

Linux 内核 2.6 及以上，安装有 OpenSSL。

实验内容

【实验过程】

一、实验步骤：

1. 编写程序生成 RSA 密钥对。(详情见代码注释,下同)

代码:

```cpp
#include<iostream>
#include<openssl/ssl.h>
#include<openssl/pem.h>
using namespace std;
int main()
{
    //分配空间生成密钥对
    RSA *r = RSA_new();
    int bits = 1024;
    BIGNUM *e = BN_new();
    BN_set_word(e, 65537);
    RSA_generate_key_ex(r, bits, e, NULL);

   // RSA_print_fp(stdout, r, 0);打印密钥对信息

    BIO *out;
    out = BIO_new_file("/var/MyCA/pri.pem","w");//保存私钥,默认pem格式
    //这里生成的私钥没有加密，可选加密
    int ret = PEM_write_bio_RSAPrivateKey(out, r, NULL, NULL, 0, NULL, NULL);
    printf("writepri:%d\n",ret);
    BIO_flush(out);
    BIO_free(out);

    out = BIO_new_file("/var/MyCA/pub.pem","w");//保存公钥
    ret = PEM_write_bio_RSAPublicKey(out, r);
    printf("writepub:%d\n",ret);
    BIO_flush(out);
    BIO_free(out);

    BN_free(e);
    RSA_free(r);
}
```

结果

原来无密钥

```
root@cslp:/var/MyCA# ls
2.cpp  ca.crt  client.crt  display.cpp  hello.txt       index.txt.old  serial
3.cpp  ca.key  client.csr  genRSA.cpp   index.txt       openssl.cnf    serial.old
a.out  certs   client.key  hello.sign   index.txt.attr  private        verify.cpp
```

执行完之后生成密钥并保存到相应文件

```
root@cslp:/var/MyCA# g++ genRSA.cpp -lcrypto
root@cslp:/var/MyCA# ./a.out
writepri:1
writepub:1
root@cslp:/var/MyCA# ls
2.cpp  ca.crt  client.crt  display.cpp  hello.txt       index.txt.old  private  serial.old
3.cpp  ca.key  client.csr  genRSA.cpp   index.txt       openssl.cnf    pub.pem  verify.cpp
a.out  certs   client.key  hello.sign   index.txt.attr  pri.pem        serial
```

2. 对某个文件 Hash 后计算其数字签名,并把得到的签名信息保存到文件

代码:

```cpp
#include<iostream>
#include<openssl/ssl.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<openssl/md5.h>
#include<openssl/bio.h>
#include<openssl/pem.h>
#include<openssl/rsa.h>
#include<fstream>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
using namespace std;
//读取密钥
int main()
{
        //读取私钥信息,用作签名
        OpenSSL_add_all_algorithms();
        RSA*pri_key=RSA_new();
        BIO* in;
        in=BIO_new_file("/var/MyCA/pri.pem","rb");
        pri_key=PEM_read_bio_RSAPrivateKey(in,&pri_key,NULL,NULL);
        //读取代签名文件并存储到字符数组
        int fd;
        fd = open("hello.txt", O_RDWR);
        char data[100];
        int n = read(fd,data,100);//read() Linux C 函数
        data[n] = 0;  //等效于data[n]='\0';
        //char data[100]="0";
        //read(fd,data,100);
        //data[strlen(data)]  也行,注意要初始化,不然极易出问题
        close(fd);
//      cout<<data;
//      调用MD5函数对字符串作哈希,同理存储hash后的字符数组也要初始化
        char hashdata[1000];
```

```cpp
        char hashdata[1000];
        memset(hashdata,0,sizeof(hashdata));
        //char hashdata[1000]={'0'};
        MD5((unsigned char*)data,strlen(data),(unsigned char*)hashdata);
        //调用私钥对hash值签名
        char signdata[1000]="0";
        unsigned int signlen=0;
        if(RSA_sign(NID_md5,(unsigned char *)hashdata,strlen(hashdata),(unsigned char*)signdata,&signlen,p
ri_key)!=1)
                printf("RSA_sign err\n");
        //签名后的文件写入hello.sign
        fd=open("hello.sign",O_CREAT|O_RDWR,S_IRUSR|S_IWUSR);
        if(fd)
        write(fd,signdata,signlen);
        else printf("open hello.sign err!\n");
        close(fd);
}
```

结果:

执行前无 hello.sign

```
root@cslp:/var/MyCA# ls
2.cpp  ca.crt  client.crt  display.cpp  index.txt       openssl.cnf  pub.pem     verify.cpp
3.cpp  ca.key  client.csr  genRSA.cpp   index.txt.attr  pri.pem      serial
a.out  certs   client.key  hello.txt    index.txt.old   private      serial.old
```

待签名文件 hello.txt

```
root@cslp:/var/MyCA# cat hello.txt
Hello World,jjjj
Hello PKI
NIUBI TK!
NIUBI LP!
NIUBI LP
NIUBI LPLPLPLPLP
```

执行后生成 hello.sign

```
root@cslp:/var/MyCA# g++ 2.cpp -lcrypto
root@cslp:/var/MyCA# ./a.out
root@cslp:/var/MyCA# ls
2.cpp  ca.crt   client.crt  display.cpp  hello.txt       index.txt.old  private  serial.old
3.cpp  ca.key   client.csr  genRSA.cpp   index.txt       openssl.cnf    pub.pem  verify.cpp
a.out  certs    client.key  hello.sign   index.txt.attr  pri.pem        serial
```

3．利用公钥对所生成的数字签名进行验证

代码:

```
#include<iostream>
#include<openssl/ssl.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<openssl/md5.h>
#include<openssl/bio.h>
#include<openssl/pem.h>
#include<openssl/rsa.h>
#include<fstream>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
using namespace std;
//读取密钥
int main()
{
        //同上面,打开hello.txt文件并做hash保存在hashdata字符数组中用来比对
        int fd;
        fd = open("hello.txt", O_RDWR);
        char data[100]="0";
        read(fd,data,100);
        close(fd);
        char hashdata[1000]={'0'};
        MD5((unsigned char*)data,strlen(data),(unsigned char*)hashdata);
        //读取公钥
        BIO* in;
        in=BIO_new_file("/var/MyCA/pub.pem","rb");
        auto pub_key=PEM_read_bio_RSAPublicKey(in,NULL,NULL,NULL);
        if(pub_key==NULL)
        {
                printf("read public key err!\n");
                return -1;
        }
        //读取hello.sign文件中的签名
        fd=open("hello.sign",O_RDWR);
        char signdata[300]="0";
        read(fd,signdata,300);
        close(fd);
        //比对
        if(RSA_verify(NID_md5,(unsigned char*)hashdata,strlen(hashdata),(unsigned char*)signdata,128,pub_k
ey)!=1)
        {
                printf("RSA_verify err!\n");
                return -1;
        }
        printf("\nRSA_verify OK!\n");
}
```

结果:

```
root@cslp:/var/MyCA# g++ 3.cpp -lcrypto
root@cslp:/var/MyCA# ./a.out

RSA_verify OK!
```

小结

以前对于 RSA 公钥体系与数字签名,求摘要都是理论上的了解.这次借助 openssl 库编程实现了生成公私钥,数字签名,加密与验证的程序,加深了对整个安全流程的理解.主要的困难一是对 openssl 不熟悉,另外就是对 c 语言不熟悉,对于 linux 下编程的环境,库配置等也不熟悉,不过通过这次的实验,已经较好的掌握了上述的内容.

指导教师评语及成绩

评语： 成绩：　　指导教师签名： 批阅日期：