华南理工大学

《PKI 原理与技术》课程实验报告

实验题目：C/C++实现证书的读取与验证数字签名

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：

掌握证书的结构，学会验证数字签名。

实验要求：

利用 OpenSSL 提供的库函数，用 C/C++编写：

1. 对于已经签发的一张个人数字证书，读出证书中每一个字段的内容，并显示出来，如果是二进制内容请用十六进制数来显示。
2. 验证证书中 CA 的数字签名是否为有效的（假设某根 CA 为可信的）。

【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. 显示个人证书的内容

得到一个 x509 的数据结构,调用 OpenSSL 库函数显示证书

代码:

```cpp
#include<iostream>
#include<openssl/ssl.h>
using namespace std;
int main(int argc,char *argv[])
{
        //得到一个x509的数据结构
        auto b=BIO_new_file(argv[1],"rb");
        auto x=PEM_read_bio_X509(b,NULL,NULL,NULL);
        BIO_free(b);
        //调用函数显示到标准输出
        b=BIO_new(BIO_s_file());
        BIO_set_fp(b,stdout,BIO_NOCLOSE);
        X509_print(b,x);
}
```

结果

```
root@cslp:/var/MyCA# g++ display.cpp -lcrypto
root@cslp:/var/MyCA# ./a.out client.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: md5WithRSAEncryption
        Issuer: CN=My Test CA, ST=HZ, C=CN/emailAddress=test@cert.com, O=Root Certification Authority
        Validity
            Not Before: May  8 01:45:15 2019 GMT
            Not After : May  7 01:45:15 2020 GMT
        Subject: CN=scut.tanglab.tang, ST=GD, C=CN/emailAddress=12345@126.com, O=SCUT, OU=TANGLAB
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:f1:b5:dc:6b:98:85:c0:17:35:d3:70:3c:03:9a:
                    19:18:1b:f5:f1:6d:f7:7c:40:d0:f5:cf:da:05:94:
                    f3:ee:74:61:fb:bc:f9:8b:3c:6b:b4:4d:7f:03:bc:
                    81:35:c3:1c:63:50:2e:93:93:92:f1:ec:49:64:b9:
                    33:d7:65:9d:bf:b9:a5:0f:2a:c9:99:e4:4a:0c:13:
                    84:30:5c:31:35:ea:f5:d7:15:c0:40:1c:89:84:7b:
                    bb:82:c9:6b:44:18:71:0f:f7:99:71:88:19:ae:92:
                    e7:97:d5:7c:bc:ec:ac:ae:1e:62:8e:ce:41:09:9f:
                    8c:b3:73:87:69:02:f4:48:15
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
```

```
    Signature Algorithm: md5WithRSAEncryption
         a5:ca:29:25:9f:b5:da:b2:a5:00:c1:8e:4c:b9:ad:fa:4f:94:
         fe:15:85:db:8a:dc:6c:22:bd:8c:31:e7:1f:a6:a2:9a:9c:6a:
         7c:06:13:48:8b:50:9b:b2:dc:c3:5a:6e:66:72:8d:e9:80:9f:
         1e:6c:48:b1:9e:ef:b4:30:07:81:fe:72:8c:5e:b1:2d:2b:ba:
         63:00:c8:30:7a:6d:31:64:ee:a2:f2:52:59:87:94:b0:84:96:
         26:c5:78:62:d2:4e:da:0f:d7:85:31:d4:cc:84:35:b0:05:35:
         7d:d1:83:7a:ae:3a:da:6b:6c:08:73:6f:5e:66:9a:f6:03:f4:
         f0:4e:e3:5d:28:6c:64:79:af:49:f6:0c:f2:09:6d:c6:df:5b:
         da:96:b1:a4:2a:6d:3a:c7:d4:df:47:8d:fb:00:9d:8a:4b:8d:
         82:b6:09:8e:eb:76:91:e7:22:dc:82:b4:98:bb:77:54:11:d8:
         90:e3:2c:29:70:18:17:c8:7a:1a:5f:b3:dc:d2:7d:fd:43:9d:
         cd:ed:3a:5b:fe:5d:f2:59:5e:c8:8b:95:7e:ec:f8:c4:f1:6a:
         cf:54:2a:05:4a:62:fc:ab:d5:32:25:91:00:f4:29:05:b7:4c:
         d4:48:20:11:6c:df:a3:90:b0:70:48:78:47:71:e4:dc:d8:a2:
         f1:2e:c8:1a
```

2. 验证证书中 CA 的数字签名是否为有效的

代码:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/pem.h>

void app_abort (char *msg)
{
        fprintf(stderr, msg);
        exit(-1);
}

int main(int argc, char *argv[]){
        int i,ret;
        X509_LOOKUP *lookup;
        BIO *in = NULL;
        X509 *x = NULL;
        X509_STORE *cert_ctx = NULL;
        X509_STORE_CTX *csc;

        if (argc!=3)
                app_abort("./a.out <CAfile> <certfile>");
        if ((cert_ctx = X509_STORE_new()) == NULL)
                app_abort("Can't create cert_ctx");

        if ((lookup = X509_STORE_add_lookup(cert_ctx, X509_LOOKUP_file())) == NULL)
                app_abort("Lookup CA file error");
        if (!(i = X509_LOOKUP_ctrl(lookup, X509_L_FILE_LOAD,
        argv[1],(long)X509_FILETYPE_PEM, NULL)))
                app_abort("Can't open CAfile");
        ERR_clear_error();
```

```
        if ((in = BIO_new(BIO_s_file())) == NULL)
                app_abort("certfile BIO error");
        if (BIO_read_filename(in, argv[2])<=0)
                app_abort("open certfile error");
        if ((x = PEM_read_bio_X509(in, NULL, NULL, NULL)) == NULL)
                app_abort("load certfile error");

        if ((csc = X509_STORE_CTX_new()) == NULL)
                app_abort("ctx init error");
        X509_STORE_CTX_init(csc, cert_ctx, x, NULL);
        if ((i = X509_verify_cert(csc)) == 1)
                printf("Status=OK(%d)\n", i);
        else
                printf("Status=Error(%d)\n", i);
}
```

结果:

```
root@cslp:/var/MyCA# g++ verify.cpp -lcrypto
verify.cpp: In function ‘void app_abort(char*)’:
verify.cpp:14:21: warning: format not a string literal and no format arguments [-Wformat-security]
  fprintf(stderr, msg);
                  ^
verify.cpp: In function ‘int main(int, char**)’:
verify.cpp:27:45: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
      app_abort("./a.out <CAfile> <certfile>");
                                             ^
verify.cpp:29:39: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
      app_abort("Can't create cert_ctx");
                                       ^
verify.cpp:32:38: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
      app_abort("Lookup CA file error");
                                      ^
verify.cpp:35:36: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
       app_abort("Can't open CAfile");
                                     ^
verify.cpp:39:37: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
       app_abort("certfile BIO error");
                                      ^
verify.cpp:41:37: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
      app_abort("open certfile error");
                                      ^
verify.cpp:43:37: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
      app_abort("load certfile error");
                                      ^
verify.cpp:46:29: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   app_abort("ctx init error");
                             ^
root@cslp:/var/MyCA# ./a.out ca.crt client.crt
Status=OK(1)
```

小结

学过 x509 证书,也基本掌握了它的内容和结构,这次用 openssl 编程实现显示其内容以及验证其正确性的功能.加深了对证书的认识.主要的困难就是对 openssl 库不熟悉,不了解其各种库函数的使用以及要包含的头文件,不过最后这些问题都通过上网查找解决了.

指导教师评语及成绩

评语：

成绩：　　指导教师签名：

批阅日期：