山东大学 软件 学院

信息安全导论 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201600301079 | 姓名： 崔玉峰 | | 班级： 2016级软件4班 |
| 实验题目： OpenSSL的使用 | | | |
| 实验学时：4 | | 实验日期： 2018/10/24 | |
| 实验目的：  **OpenSSL的使用**  （1）了解对称算法的基本工作流程。  （2）掌握对称算法的使用方法。  **内容：**  （1）认识OpenSSL工具包。  （2）用简短的程序代码演示：分组加密算法（DES、AES）和流密码算法（RC4）的使用，其中包括分组算法的四种应用模式ECB、CBC，CFB、OFB。  （3）编写一个简单但是安全的文件加密程序。 | | | |
| 硬件环境：  PC  CPU : intel-i5  RAM: 8G | | | |
| 软件环境：  虚拟机：VMWARE  操作系统：ubuntu 18.04.5 64位  软件库： OpenSSL | | | |
| 实验步骤与内容：   1. **Linux环境下安装openssl** 2. **直接安装** 3. 从openssl官网下载最新的稳定版本 Opensssl-1.0.2.tar.gz 将下载好的文件解压 . 4. 查看文件目录中INSTALL文件，帮助安装。      1. 根据所给出的指令进行安装分别执行   Sudo ./config  Make  Make test  Make install  等待数秒后安装成功 运行openssl version -a 命令查看是否安装功       1. **编译自己的openssl库**   因为上述安装方式会替换Linux系统自带Openssl库所以会非常危险，并且不能向openssl自行修改添加新的算法,所以应该编译安装自己的opnessl库：  ① ./config  ② make -j4  ③ make test  ④ export LD\_LIBRARY\_PATH=.  执行过后通过strace ./aes 2>&1 | grep libcry命令查看引用的库情况，可以看到引用自己的openssl库。     1. **DES算法模拟与实现：**     Openssl对于DES对称加密算法的封装非常完整，主要的步骤是设置密钥(最好随机) ，然后通过调用DES\_string\_to\_key，DES\_set\_key\_checked设置密钥。接着调用加密算法可采用分组算法的四种应用模式。Ecb ncbc cfb ofb  解密过程调用相同算法解密。具体实现代码如下：  #include <stdio.h>  #include <openssl/des.h>  /\*\*gcc DES.c -lcrypto  \* create by cyf in 2018/10/31  \* implement DES use openssl  \*/  int main(int argc, char \*\*argv)  {  //key string  unsigned char \*keystring = "a123456789";  DES\_cblock key;  DES\_key\_schedule schedule;  //create key  DES\_string\_to\_key(keystring, &key);  DES\_set\_key\_checked(&key, &schedule);  //cleartext  const\_DES\_cblock cleartext = "CYFDES12";  //print cleartext  printf("cleartext: %s\n", cleartext);  //ciphertext  DES\_cblock ciphertext;  //Encrypted ECB  DES\_ecb\_encrypt(&cleartext, &ciphertext, &schedule, DES\_ENCRYPT);  printf("Encrypted!\n");  //print ciphertext  printf("ciphertext: ");  for (int i = 0; i < sizeof(ciphertext); i++)  printf("%02x", ciphertext[i]);  printf("\n");  //Deciphertext  const\_DES\_cblock deciphertext;  //Decrypted  DES\_ecb\_encrypt(&ciphertext, &deciphertext, &schedule, DES\_DECRYPT);  printf("Decrypted!\n");  printf("deciphertext:%s\n", deciphertext);  return 0;  }  **运行结果：**       1. **AES算法的模拟与实现**     AES对称加密算法的实现与DES大体相同，同样有四种应用模式，具体实现代码如下：  #include <stdio.h>  #include <openssl/aes.h>  /\*\*gcc AES.c -lcrypto  \* create by cyf in 2018/10/31  \* implement AES use openssl  \*/  int main(int argc, char \*\*argv)  {  //key string  unsigned char keystring[16 + 1] = "3456789cuiyufeng";  AES\_KEY key;  //create key  AES\_set\_encrypt\_key(keystring, 16 \* 8, &key);  //cleartext  char cleartext[16 + 1] = "cuiyufengAES4567";  unsigned char iv1[16 + 1] = { "1023456789abcdef" }; // 16+1，加密用  unsigned char iv2[16 + 1] = { "1023456789abcdef" }; // 16+1，解密用  //print cleartext  printf("cleartext: %s\n", cleartext);  //ciphertext  char ciphertext[16 + 1];  //Encrypted ASE  int num = 0;  AES\_cfb128\_encrypt((unsigned char\*)cleartext, (unsigned char\*)ciphertext,  16 + 5, &key, (unsigned char\*)iv1, &num, AES\_ENCRYPT);  //AES\_encrypt((unsigned char\*)cleartext, (unsigned char\*)ciphertext,&key);  printf("Encrypted!\n");  //print ciphertext  printf("ciphertext: ");  for (int i = 0; i < sizeof(ciphertext); i++)  printf("%02x", ciphertext[i]);  printf("\n");  //Deciphertext  AES\_set\_encrypt\_key(keystring, 16 \* 8, &key);  //AES\_set\_decrypt\_key(keystring, 16\*8, &key);  char deciphertext[16 + 1];  //Decrypted  num = 0;  AES\_cfb128\_encrypt((unsigned char\*)ciphertext, (unsigned char\*)deciphertext,  16 + 5, &key, (unsigned char\*)iv2, &num, AES\_DECRYPT);  //AES\_decrypt((unsigned char\*)ciphertext,(unsigned char\*)deciphertext, &key);  printf("Decrypted!\n");  printf("deciphertext:%s\n", deciphertext);  return 0;  }  执行结果：     1. **RC4算法的模拟和实现**   RC4作为流加密算法，没有加密大小的限制，openssl的实现方式与之前大致相同，先设置密钥然后调用RC4（）进行加密和解密，具体实现的代码如下：  #include <stdio.h>  #include <openssl/rc4.h>  #include <string.h>  #include <stdlib.h>  /\*\*gcc RC4.c -lcrypto  \* create by cyf in 2018/10/31  \* implement AES use openssl  \*/  int main(int argc, char \*\*argv)  {  //key string  char keystring[16] = "3456789cuiyufeng";  //create key  RC4\_KEY key;  RC4\_set\_key(&key, 16, keystring);  //cleartext  char \* cleartext = "cuiyufengRC41234";  //print cleartext  printf("cleartext: %s\n", cleartext);  //ciphertext  char \* ciphertext = malloc(sizeof(unsigned char)\*(strlen(cleartext) + 1));  //Encrypted RC4  RC4(&key, strlen(cleartext), cleartext, ciphertext);  printf("Encrypted!\n");  //print ciphertext  printf("ciphertext: ");  for (int i = 0; i < sizeof(ciphertext); i++)  printf("%02x", ciphertext[i]);  printf("\n");  //Deciphertext  char deciphertext[16 + 1];  RC4\_set\_key(&key, 16, keystring);  //Decrypted  RC4(&key, strlen(ciphertext), ciphertext, deciphertext);  printf("Decrypted!\n");  printf("deciphertext:%s\n", deciphertext);  return 0;  }  **运行结果：**       1. **RSA加密解密演示**   RSA是最著名的非对称加密算法，不同于对称加密算法中加密和解密使用同样的密钥，公钥算法分为加密密钥K1和解密密钥K2两部分，而且从K1很难计算推导出K2。这样就可以保密K2而公布K1，从而大大简化了密钥管理。习惯上K1称为公钥，K2称为私钥。  #include <stdio.h>  #include <openssl/rsa.h>  int main(int argc, char \*\*argv)  {  //RSA generate key  RSA\* rsa = RSA\_generate\_key(1024, 71, NULL, NULL);  RSA\_print\_fp(stdout, rsa, 0);  printf("\n");  //RSA Encrypt  unsigned char \*ClearString = "cuiyufeng201600301079";  unsigned char EncryptString[1024];  printf("ClearString: ");  printf(ClearString);  printf("\n");  RSA\_public\_encrypt(22, ClearString, EncryptString, rsa, RSA\_PKCS1\_PADDING);  printf("EncryptString: ");  printf(EncryptString);  printf("\n");  //RSA Decrypt  unsigned char DecryptString[1024];  RSA\_private\_decrypt(128, EncryptString, DecryptString, rsa, RSA\_PKCS1\_PADDING);  printf("DecryptString: ");  printf(DecryptString);  printf("\n");  }  **运行结果：**  公钥密钥信息：    加密结果： | | | |
| 结论分析与体会：    通过编程熟悉了openssl库的安装和使用，并且对对称加密算法des,aes,流算法rc4,都进行了编写和模拟，了解了算法的执行步骤，如何对数据进行加密或者验证。  如何评价市面上有的“高强度文件夹加密”软件声称“上百G的数据仅需1秒钟完成”，根据最近对加密算法的学习，可以完全认为这里有很大的虚构成分，即使用速度最快的流加密算法，想要加密上百G的数据也是不可能在1秒内完成的；如果真的达到了这个速度，那么它加密的安全性肯定无法保证。 | | | |