OpenSSL

在linux 中编译和安装

1. 获得 OpenSSL

   从OpenSSL的 官方网站 即可下载当前版本的 OpenSSL 源代码压缩包。当前版本为 openssl-1.0.2l。

   在Linux中解压缩：

   $tar zxf openssl-1.0.2l.tar.gz

2. 编译和安装

   $sudo ./config

   $sudo make

   $sudo make test

   $sudo make install

   安装OpenSSL 开发包，这在所有的现代Linux发行版的标准软件仓库中都有。

   $sudo apt-get install libssl-dev

实验三 对称加密算法实验

1. DES 算法接口

   DES 加密算法是分组算法。 DES 的基本操作是把64比特明文在 56 比特密钥指引下加密成64位密文

   DES(IN,KEY) = OUT

   在openSSL中，DES算法的基本函数就是ECB 操作模式对应的函数 DES_ecb_encrypt() 。该函数把一个8字节明文分组input加密成为一个 8 字节密文分组 output。

   参数中密钥结构 ks 是用函数 DES_set_key()准备好的， 而密钥 key 是用随机数算法产生的 64 个随机比特。

   参数enc 指示数 enc 指示是加密还是解密。

   ECB操作模式

   电子密码本模式，是最古老最简单的模式，将加密的数据分成若干组，每组的大小跟加密密钥长度相同，每组都使用相同的密钥加密，如果最后一个分组长度不够64位，要补齐64位。

   该函数每次只加密一个分组，因此用来加密很多数据时并不方便。

```
1  void DES_ecb_encrypt(const_DES_cblock *input,DES_cblock *output,
   DES_key_schedule *ks,int enc);
2
3  int DES_set_key(const_DES_cblock *key,DES_key_schedule *schedule);
```

特点：每次Key、明文、密文的长度都必须是64位 ； 一个错误仅仅会对一个密文块产生影响；

CBC操作模式

加密块链模式，与ECB模式最大的不同是加入了初始向量

DES 算法 CBC 操作模式加解密函数是 DES_ncbc_encrypt()。参数length指示输入字节长度。如果长度不是8字节的倍数，则会被用0填充到8字节的倍数。因此，输出可能比length长，而且必然是8字节的倍数。

```
void DES_ncbc_encrypt(const unsigned char *input,unsigned char
*output, long length, DES_key_schedule *schedule, DES_cblock *ivec,
int enc)
```

特点：

- 每次加密的密文长度为64位(8字节)；
- 当相同的明文使用相同的密钥和初始向量的时候CBC模式总是产生相同的密文；
- 密文块要依赖以前的操作结果，所以密文块不能进行重新排列
- 可以使用不同的初始化向量来避免产生相同的密文，一定程度抵御字典共计
- 一个错误发生后，当前和以后的密文都将被影响

CFB操作模式

加密反馈模式，克服了需要等待8个字节才能加密的缺点，采用了分组密码作为流密码的密钥流生成器。

DES算法 CFB 操作模式加密函数是 DES_cfb_encrypt() 。参数length 指示输入字节长度。参数numbits 指示CFB 每次循环加密多少比特明文，也即密文反馈的比特数目。 ivec是初始向量，被看做第0个密文分组，是不用保密但应随机取值的8个字节。如果在一次会话中多次调用DES_cfb_encrypt()，则应记忆ivec。
由于 CFB 模式中每次 DES 基本操作
只加密 numbits 比特明文，因此如果 numbits 太小则效率太低。

```
void DES_cfb_encrypt(const unsigned char *in, unsigned char *out,
int numbits, long length, DES_key_schedule *schedule, DES_cblock
*ivec, int enc);
```

OFB操作模式

与CFB模式不同之处在于，加密移位寄存器与密文无关了，仅与加密key和加密算法有关。

输出反馈模式，允许对不同字区大小编密码，但是它与密码反馈模式主要的区别在于编字区函数密码的输出是反馈（而不是密码）

```
void DES_ofb_encrypt(const unsigned char *in,unsigned char *out,int
numbits,long length,DES_key_schedule *schedule,DES_cblock *ivec);
```

```
void DES_ofb64_encrypt(const unsigned char *in,unsigned char
*out,long length,DES_key_schedule *schedule,DES_cblock *ivec,int*
num);
```

2. AES 算法接口

   典型参数的 AES 的基本操作是把 128 比特明文在 128 比特密钥指引下加密成 128 比特密文。OpenSSL 中关于 AES 的函数名和参数接口和 DES 的雷同。相关函数名如下(参数略)。

```
    int AES_set_encrypt_key();
    int AES_set_decrypt_key();
    void AES_ecb_encrypt();
    void AES_cbc_encrypt();
    void AES_cfb128_encrypt();
    void AES_ofb128_encrypt();
```

3. RC4

   RC4 密码算法是流算法，也叫序列算法。流算法是从密钥作为种子产生密钥流，明文比特流和密钥流异或即加密。RC4 算法由于算法简洁，速度极快，密钥长度可变，而且也没有填充的麻烦，因此在很多场合值得大力推荐。

   OpenSSL 中 RC4 算法有两个函数: RC4_set_key()设置密钥，RC4()加解密。可以把 RC4 看作异或，因此加密两次即解密。

```
void RC4set_key(RC4KEY key, int len, const unsigned chardata);

void RC4(RC4_KEY key, unsigned long len, const unsigned charindata,
unsigned char *outdata);
```

实验四 公钥算法实验

使用公钥密码算法可以克服协商对称密钥的困难，也可以用来认证和签名。为了避免公钥算法的速度缺陷，当前普遍使用混合密码体制，即使用公钥算法做鉴别和协商会话密钥，使用对称算法加密批量数据。

公钥密码算法当前仍是 RSA 算法占据统治地位。OpenSSL加密函数库中提供了对RSA等算法的支持。

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 很难计算和推导出 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。

加密使用公钥，解密使用私钥

ENC (P , K1) = C

DEC (C , K2) = P

RSA 算法大致步骤

1. 寻找两个随机大素数 p 和 q ，保证p，q不相同
2. 计算模 n = pq 和 Euler 函数 f(n) = (p-1)(q-1)
3. 寻找一个与 f(n) 互质的数 e ，满足 1 < e < f(n)
4. 选取 e 后用扩展 Euclid算法求数 d 满足 e * d ≡1 mod f(n)
5. 保密私钥 KR = (d , n) ，发布公钥 KU = (e, n)
6. 加密，先将明文变换成 0 至 n-1 的一个整数 M。若明文较长，可先分割成适当的组，然后再进行交换， 加密过程为 $C \equiv M^e \pmod{n}$
7. 解密，$M \equiv C^d \pmod{n}$

RSA 密钥产生

RSA 密钥产生函数 RSA_generate_key()，需要指定模长比特数 bits 和公钥指数 e。另外两个参数为 NULL 即可。

```
RSA * RSA_generate_key(int bits, unsigned long e, void (*callback)
(int,int,void *),void *cb_arg);
```

目前对于长达663比特的 RSA 模数已经有成功分解的先例，因此当前典型的应用场合使用1024比特模长的 RSA 算法，此时一个分组是 128 字节。

如果从文件中读取密钥，可使用函数PEM_read_bio_PrivateKey()/ PEM_read_bio_PUBKEY()，其中 EVP_PKEY 中包含一个 RSA 结构，可以引用。

```
EVP_PKEY *PEM_read_bio_PrivateKey(BIO *bp, EVP_PKEY **x,
pem_password_cb *cb, void *u);
```

RSA 加密和解密

RSA算法生成密钥

1. 生成一个密钥

```
openssl genrsa -out test.key 1024
```

选项 -out 指定生成文件。该文件包含了公钥和私钥两部分，既可以用来加密也可以用来解密。 1024 表示生成密钥长度

2.将公钥提取出来

```
openssl rsa -in test.key -pubout -out test_pub.key
```

选项 -in 指定输入文件，-out 指定提取生成公钥的文件名。现在我们有一个公钥，一个私钥（包含公钥）

在以后的 RSA 代码中都会用到的公钥和私钥文件

RSA 加密函数 RSA_public_encrypt()使用公钥部分，解密函 RSA_private_decrypt() 使用私钥。填 充 方 式 常 用 的 有 两 种 RSA_PKCS1_PADDING 和 RSA_PKCS1_OAEP_PADDING。出错时返回-1>输入必须比 RSA 钥模长短至少 11 字节(在 RSA_PKCS1_PADDING 时?)。输出长度等于 RSA 钥的模长。

```
int RSA_public_encrypt(int flen, const *from,unsigned char *to, RSA
*rsa,int padding);
int RSA_private_decrypt(int flen, const *from,unsigned char *to, RSA
*rsa,int padding);
```

RSA 签名和验证

签名使用私钥，验证使用公钥。RSA签名操作是把被签署消息的散列值编码后用私钥加密，因此函数中参数 type 用来指示散列函数的类型，一般是NID_MD5,NID_sha1 正确情况下返回 0 。

```
int RSA_sign(int type, const unsigned char *m, unsigned int m_length,
unsigned char *sigret, unsigned int *siglen, RSA *rsa);
int RSA_verify(int type, const unsigned char *m, unsigned int m_length,
unsigned char *sigbuf, unsigned int siglen, RSA *rsa);
```

复习

抗否认

抵赖性

完整性

认证问题

实验报告提交要求

danny0128@126.com

主题： 2014IS 一个doc 文档

doc 文档命名： 学号+姓名.doc

deadline :

周五那一天 半夜12点

实验结束后实验报告邮件

平时成绩： 网络登录 + 小测验 +