**OpenSSL之X509系列之1---引言和X509概述**

**【引言】**  
    X509是系列的函数在我们开发与PKI相关的应用的时候我们都会用到，但是OpenSSL中对X509的描述并不是很多，鉴于些，我将以前工作与学习过程的经验整理出来，供大家参考，不用多走弯路，可以将精力集中在自己要处理的业务逻辑上，同时也希望更多的人参与到研究与整理信息安全的理论与技术中来，提高中国的科研与应用技术水平。提高中国信息安全意识与能力从我做起。  
**【X509概述】**    X.509是国际标准化组织CCITT建议作为X.500目录检索的一部分提供安全目录检索服务。一份X.509证书是一些标准字段的集合，这些字段包含有关用户或设备及其相应公钥的信息一种非常通用的证书格式，所有的证书都符合X.509 国际标准。目前X.509有不同的版本，例如 X.509 V2和x.509 v3都是目前比较新的版本，2000年还推出V4版本，但是都在原有版本基础上进行功能的扩充，其中每一版本必须包含下列信息：  
　　(1) 用来区分X.509的不同版本号既版本号  
　　(2) 由CA给予每一个证书的分配的编号即序列号；  
　　(3) 用于产生证书所用的方法以及一切参数即签名算法  
　　(4) CA的x.500名字即发出该证书的认证机构  
　　(5) 证书有效的时间包括两个日期，在所指定的两个时间之 间有效即有效期限  
　　(6) 证书持有人的姓名、服务处所等信息即主题信息  
　　(7) 认证机构的数字签名  
    (8) 被证明的公钥值，加上使用这个公钥的方法名称即公钥信息  
**【X.509证书格式】**  
    X.509是另一种非常通用的证书格式。所有的证书都符合ITU-T X.509国际标准；因此(理论上)为一种应用创建的证书可以用于任何其他符合X.509标准的应用。但实际上，不同的公司对X.509证书进行了不同的扩展，不是所有的证书都彼此兼容。在一份证书中，必须证明公钥及其所有者的姓名是一致的。对PGP证书来说，任何人都可以扮演认证者的角色。对X.509证书来说，认证者总是 CA或由CA指定的人(其实PGP证书也完全支持使用CA来确认证书的体系结构)，一份X.509证书是一些标准字段的集合，这些字段包含有关用户或设备及其相应公钥的信息。X.509标准定义了证书中应该包含哪些信息，并描述了这些信息是如何编码的(即数据格式)，所有的X.509证书包含以下数据：  
　(1)X.509版本号：指出该证书使用了哪种版本的X.509标准，版本号会影响证书中的一些特定信息。目前的版本是3。   
　(2)证书持有人的公钥：包括证书持有人的公钥，算法(指明密钥属于哪种密码系统)的标示符和其他相关的密钥参数。  
　(3)证书的序列号：创建证书的实体(组织或个人)有责任为该证书指定一个独一无二的序列号，以区别于该实体发布的其他证书。序列号信息有许多用途；比如当一份证书被回收以后，它的序列号就被放入证书回收列表(CRL)之中。  
　(4)证书持有人唯一的标示符：(或称DN-distinguished name)这个名字在 Internet上应该是唯一的。DN由许多部分组成，看起来象这样：   
　　CN=Bob Allen, OU=Total Network Security Division,   
　　O=Network Associates, Inc., C=US   
这些信息指出该科目的通用名，组织单位,组织和国家   
  (5)证书的有效期：证书起始日期和时间以及终止日期和时间；指明证书何时失效。   
　(6)证书发布者的唯一名字：这是签发该证书的实体的唯一名字。通常是CA。.使用该证书意味着信任签发证书的实体。(注意：在某些情况下，比如根或顶级CA证书，发布者自己签发证书)   
　(7)发布者的数字签名：这是使用发布者私钥生成的签名。   
　(8)签名算法的标示符：指明CA签署证书所使用的算法。   
　X.509证书和PGP证书之间有许多不同,最明显的如下所述：   
  (1) 用户可以创建自己的PGP证书，但是必须向CA请求才能得到一份X.509证书。   
  (2)X.509证书天生只支持密钥拥有者的一个名字。  
  (3)X.509证书只支持证明密钥合法性的一个数字签名。  
    要获得一份X.509证书，必须请求CA发给你证书。用户提供自己的公钥，证明自己拥有相应的私钥，并提供有关自己的某些特定信息。然后在这些信息上数字签名，并将整个数据包(称为证书请求)发给CA。CA做一些努力来验证用户提供的信息是正确的，然后就生成证书并返回给用户。  
**【OpenSSL对X509的支持】**  
    以下是我自己对OpenSSL的理解，可以表达上不是很准确。  
    (1) 证书请求管理  
    (2) 证书生成  
    (3) 证书吊销及CRL管理  
    (4) X509名字管理  
    (5) 属性管理  
    (6) 扩展管理  
    (7) 验证及信任管理  
     在随后的一些篇幅中将对以上的这几个方面进行展开说明。

**OpenSSL之X509系列之2---证书请求管理**

**【数据结构】**  
    证书请求用到了两个重要的数据结构：证书请求信息结构X509\_REQ\_INFO与证书请求结构X509\_REQ，二者的定义如下：  
typedef struct X509\_req\_info\_st  
{  
ASN1\_ENCODING enc;  
ASN1\_INTEGER \*version;  
X509\_NAME \*subject;  
X509\_PUBKEY \*pubkey;  
/\* d=2 hl=2 l= 0 cons: cont: 00 \*/  
STACK\_OF(X509\_ATTRIBUTE) \*attributes; /\* [ 0 ] \*/  
} X509\_REQ\_INFO;  
其中version就是版本号、subject就是主题(常用的是dn)、pubkey是事先生成的公钥、attributes是一系列的属性，用于表达证书主题的额外信息，细节参见PKCS#10与PKCS#9。  
typedef struct X509\_req\_st  
{  
X509\_REQ\_INFO \*req\_info;  
X509\_ALGOR \*sig\_alg;  
ASN1\_BIT\_STRING \*signature;  
int references;  
} X509\_REQ;  
其中req\_info就是上面所说的证书请求信息、sig\_alg是签名使用的算法比如md5WithRSAEncryption、signature就是签名值了。  
**【基本操作函数概述】**  
    这些基本的操作函数主要是对证书请求项进行设置与读取操作，它的的定义如下：其中的X509\_REQ\* req对数指的是要操作的X509\_REQ对象，下面不再赘述。  
int X509\_REQ\_set\_version(X509\_REQ \*x,long version);  
int X509\_REQ\_set\_subject\_name(X509\_REQ \*req,X509\_NAME \*name);   
int X509\_REQ\_set\_pubkey(X509\_REQ \*x, EVP\_PKEY \*pkey);  
EVP\_PKEY \* X509\_REQ\_get\_pubkey(X509\_REQ \*req);  
X509\_REQ\_extract\_key(a)  
int X509\_REQ\_verify(X509\_REQ \*a, EVP\_PKEY \*r);  
int X509\_REQ\_sign(X509\_REQ \*x, EVP\_PKEY \*pkey, const EVP\_MD \*md);  
**【X509\_REQ\_set\_version】**  
    设置版本号, version就是版本号。  
**【X509\_REQ\_set\_subject\_name】**  
    该函数设置证书请求人的主题名，X509\_NAME \*name参数就是要设置的主题名。对于名字的操作到时会有一个专题来讲。  
**【X509\_REQ\_set\_pubkey】**  
    设置公钥，EVP\_PKEY \*pkey参数就是生成好的公钥，可以通过RSA\_generate\_key()来生成。  
比如：  
**EVP\_PKEY \*pNewRsaKey;  
intGenerateRSAKeyPair(char \* szKeyLength)  
{  
   if(strlen(szKeyLength)==0) return -1;  
   int keylength=atoi(szKeyLength);  
   if ((pNewRsaKey=EVP\_PKEY\_new()) == NULL) return CA\_FAIL;  
   int ret = EVP\_PKEY\_assign\_RSA(pNewRsaKey,RSA\_generate\_key(keylength,0x10001,  
          NULL, // req\_cb  
          NULL)); // cb args  
   if(ret != 1) return CA\_FAIL;  
   return CA\_OK;  
}**  
**【X509\_REQ\_get\_pubkey】**  
    读取X509\_REQ中的公钥信息，返回的是一个EVP\_PKEY对象，X509\_REQ\_extract\_key（）是它的一个宏定义，功能相同。  
**【X509\_REQ\_sign】**  
    对X509\_REQ中X509\_REQ\_INFO结构用pkey与md进行签名，并用算法标识与签名值填充X509\_REQ中的sig\_alg与signature域。  
**【X509\_REQ\_verify】**  
    与签名相对应，对签名进行验证，所以将公钥pkey传入就可以了。

**OpenSSL之X509系列之3---证书请求的IO函数**

**【输入输出函数】**    这些函数有两类：一类是将X509\_REQ信息在文件或BIO抽象层上输入输出，另一类是在控制台上将X509\_REQ信息进行显示。它们的函数定义如下：  
X509\_REQ \*d2i\_X509\_REQ\_fp(FILE \*fp,X509\_REQ \*\*req);  
int i2d\_X509\_REQ\_fp(FILE \*fp,X509\_REQ \*req);  
X509\_REQ \*d2i\_X509\_REQ\_bio(BIO \*bp,X509\_REQ \*\*req);  
int i2d\_X509\_REQ\_bio(BIO \*bp,X509\_REQ \*req);  
int X509\_REQ\_print\_ex(BIO \*bp, X509\_REQ \*x, unsigned long nmflag, unsigned long cflag);  
int X509\_REQ\_print(BIO \*bp,X509\_REQ \*req);  
int X509\_REQ\_print\_fp(FILE \*fp, X509\_REQ \*x)  
**【d2i\_X509\_REQ\_fp】**  
    将证书请求从文件中读入并转化成X509\_REQ内部结构。  
**【i2d\_X509\_REQ\_fp】**  
    将X509\_REQ对象进行DER编码输出,并写入fp指定的文件中。  
**【d2i\_X509\_REQ\_bio】**  
    功能与d2i\_X509\_REQ\_fp相同，只是读的时候从BIO抽象层上读，你可以将它与文件相关联就可以了。  
**【i2d\_X509\_REQ\_bio】**  
    功能与i2d\_X509\_REQ\_fp相同，只是写的时候从BIO抽象层上写，你可以将它与文件或者内存BIO相关联就可以输出了。  
**【X509\_REQ\_print】**  
    将X509\_REQ在BIO上输出，但输入是可以读的，比如Subject=XXX等。其实底层就是调用X509\_REQ\_print\_ex来实现的。  
**【X509\_REQ\_print\_ex】**  
    这个函数与X509\_REQ\_print的区别是可以用标志去控制输出，nmflags用于控制显示方式，cflag用于控制哪些不显示，可以按自己的需要进行定制。它们的定义在x509.h里。  
具体如下：  
#define X509\_FLAG\_COMPAT 0  
#define X509\_FLAG\_NO\_HEADER 1L  
#define X509\_FLAG\_NO\_VERSION (1L << 1)  
#define X509\_FLAG\_NO\_SERIAL (1L << 2)  
#define X509\_FLAG\_NO\_SIGNAME (1L << 3)  
#define X509\_FLAG\_NO\_ISSUER (1L << 4)  
#define X509\_FLAG\_NO\_VALIDITY (1L << 5)  
#define X509\_FLAG\_NO\_SUBJECT (1L << 6)  
#define X509\_FLAG\_NO\_PUBKEY (1L << 7)  
#define X509\_FLAG\_NO\_EXTENSIONS (1L << 8)  
#define X509\_FLAG\_NO\_SIGDUMP (1L << 9)  
#define X509\_FLAG\_NO\_AUX (1L << 10)  
#define X509\_FLAG\_NO\_ATTRIBUTES (1L << 11)  
**【X509\_REQ\_print\_fp】**  
    其实这个函数就是将可读的结果保存在文件里，内存就是生成一个BIO对象BIO\_new(BIO\_s\_file()，然后再将文件句柄传给他BIO\_set\_fp(b,fp,BIO\_NOCLOSE)，再调用X509\_REQ\_print函数进行输出。这几个print函数,具体实现在crypto/asn1/t\_req.c中。

**OpenSSL之X509系列之4---证书请求的扩展项操作**

**【扩展项操作函数】**  
    这些函数主要是对证书的请求的扩展项进行读取与设置操作,  
    int X509\_REQ\_extension\_nid(int nid);  
    int \* X509\_REQ\_get\_extension\_nids(void);  
    void X509\_REQ\_set\_extension\_nids(int \*nids);  
    STACK\_OF(X509\_EXTENSION) \*X509\_REQ\_get\_extensions(X509\_REQ \*req);  
    int X509\_REQ\_add\_extensions\_nid(X509\_REQ \*req, STACK\_OF(X509\_EXTENSION) \*exts,int nid);  
    int X509\_REQ\_add\_extensions(X509\_REQ \*req, STACK\_OF(X509\_EXTENSION) \*exts);  
**【X509\_REQ\_extension\_nid】**  
    判断nid是否已经在内部nid\_list列表中定义了。未定义返回0，否则返回1。  
**【X509\_REQ\_get\_extension\_nids】**  
    返回已经定义的nid列表。  
**【X509\_REQ\_set\_extension\_nids】**  
    设置定义好的nid列表。  
**【X509\_REQ\_get\_extensions】**  
    取出证书请求中的扩展项，过程是这样的，先从属性中将经过der编码的扩展项取出来，然后调用d2i\_ASN1\_SET\_OF\_X509\_EXTENSION函数，将它转化成内部结构。  
**【X509\_REQ\_add\_extensions】**  
    将定义好，且赋了值的X509\_EXTENSION扩展项加入证书请求中(其实是加到属性中，这在以后讲)。  
**【X509\_REQ\_add\_extensions\_nid】**  
   功能与X509\_REQ\_add\_extensions相同,只不过nid参数可以使用非标准的nid，其实X509\_REQ\_add\_extensions就是通过调用这个函数是实现的，只不过使用了objects.h中定义的Extension Request标准定义。  
#define NID\_ext\_req 172

**OpenSSL之X509系列之5---证书请求的其它相关操作**

**【相关操作函数】**    X509\_REQ \* X509\_to\_X509\_REQ(X509 \*x, EVP\_PKEY \*pkey, const EVP\_MD \*md);  
    X509 \* X509\_REQ\_to\_X509(X509\_REQ \*r, int days,EVP\_PKEY \*pkey);  
    int X509\_REQ\_digest(const X509\_REQ \*data,const EVP\_MD \*type,unsigned char \*md, unsigned int \*len);  
    X509\_REQ \*X509\_REQ\_dup(X509\_REQ \*req);  
**【X509\_to\_X509\_REQ】**  
    用X509证书结构直接生成一个证书请求结构,其中x就是证书结构，pkey是公钥，md是散列算法，操作的过程就是将证书里的主题名与公钥填充到X509\_REQ证书请求结构中，然后用指定的pkey与md进行签名，成功返回X509\_REQ证书请求结构。  
**【X509\_REQ\_to\_X509】**  
    从证书请求结构直接生成一个X509证书，其中的day就是证书的有效期(多少天)，pkey就是用于签名的私钥。操作过程：从证书请求结构取出主题，将它填充到X509的主题与签发者中，取出公钥填充到X509公钥域里，有MD5与私钥进行签名，所以这样生成证书应用是一张自签名的证书。  
**【X509\_REQ\_digest】**  
    将X509\_REQ用指定的散列算法type进行散列。结果在md中，len是结果的长度。  
**【X509\_REQ\_dup】**  
    复制一份X509\_REQ结构。它是宏定义，实际上是由ASN1\_dup函数来完成复制工作。