**使用openssl库实现RSA、AES数据加密**

     openssl是可以很方便加密解密的库，可以使用它来对需要在网络中传输的数据加密。可以使用非对称加密：公钥加密，私钥解密。openssl提供了对RSA的支持，但RSA存在计算效率低的问题，所以一般的做法是使用对称密钥加密数据，然后再把这个只在当前有效的临时生成的对称密钥用非对称密钥的公钥加密之后传递给目标方，目标方使用约定好的非对称密钥中的私钥解开，得到数据加密的密钥，再进行数据解密，得到数据，这种使用方式很常见，可以认为是对HTTPS的裁剪。对称密钥加密可以选择AES，比DES更优秀。

     openssl库来自<http://www.openssl.org/>，下载到openssl源码之后，开始编译：

**产生动态库的做法**：

1、安装ActivePerl

2、进入OpenSSL所在文件夹，运行：perl Configure VC-WIN32 --prefix=C:\openssl-dll

3、进入VC/BIN目录，运行 VCVARS32.BAT 设置环境变量

4、返回OpenSSL目录，运行 ms\do\_ms

5、在OpenSSL目录下执行编译 nmake -f ms\ntdll.mak

6、把必要生成物拷贝到prefix定义的目录中 nmake -f ms\ntdll.mak install

注意：可以通过修改ntdll.mak文件中的CFLAG，确定编译MT、MD库

**产生静态库的做法**：

1、安装ActivePerl

2、perl configure VC-WIN32 --prefix=C:\openssl-lib

3、ms\do\_ms.bat

4、nmake -f ms\nt.mak

5、nmake -f ms\nt.mak install

注意：可以通过修改nt.mak文件中的CFLAG，确定编译MT、MD库。重编的时候把生成物删掉。

**RSA加解密需要先用openssl工具生成RSA公钥和RSA私钥**。方法：

1、产生私钥：openssl genrsa -out privkey.pem 1024；

2、根据私钥产生公钥：openssl rsa -in privkey.pem -pubout。

1024只是测试用，使用2048位才比较安全。

**RSA加密部分代码demo**：对openssl rsa生成的公钥publicKey进行RSA加密

[复制代码](javascript:void(0);)

std::string EncodeRSAKeyFile( const std::string& strPemFileName, const std::string& strData )

{

if (strPemFileName.empty() || strData.empty())

{

assert(false);

return "";

}

FILE\* hPubKeyFile = NULL;

if(fopen\_s(&hPubKeyFile, strPemFileName.c\_str(), "rb") || hPubKeyFile == NULL)

{

assert(false);

return "";

}

std::string strRet;

RSA\* pRSAPublicKey = RSA\_new();

if(PEM\_read\_RSA\_PUBKEY(hPubKeyFile, &pRSAPublicKey, 0, 0) == NULL)

{

assert(false);

return "";

}

int nLen = RSA\_size(pRSAPublicKey);

char\* pEncode = new char[nLen + 1];

int ret = RSA\_public\_encrypt(strData.length(), (const unsigned char\*)strData.c\_str(), (unsigned char\*)pEncode, pRSAPublicKey, RSA\_PKCS1\_PADDING);

if (ret >= 0)

{

strRet = std::string(pEncode, ret);

}  
 delete[] pEncode;

RSA\_free(pRSAPublicKey);

fclose(hPubKeyFile);

CRYPTO\_cleanup\_all\_ex\_data();

return strRet;

}

[复制代码](javascript:void(0);)

**RSA解密部分代码demo**：

[复制代码](javascript:void(0);)

std::string DecodeRSAKeyFile( const std::string& strPemFileName, const std::string& strData )

{

if (strPemFileName.empty() || strData.empty())

{

assert(false);

return "";

}

FILE\* hPriKeyFile = NULL;

if(fopen\_s(&hPriKeyFile, strPemFileName.c\_str(),"rb") || hPriKeyFile == NULL)

{

assert(false);

return "";

}

std::string strRet;

RSA\* pRSAPriKey = RSA\_new();

if(PEM\_read\_RSAPrivateKey(hPriKeyFile, &pRSAPriKey, 0, 0) == NULL)

{

assert(false);

return "";

}

int nLen = RSA\_size(pRSAPriKey);

char\* pDecode = new char[nLen+1];

int ret = RSA\_private\_decrypt(strData.length(), (const unsigned char\*)strData.c\_str(), (unsigned char\*)pDecode, pRSAPriKey, RSA\_PKCS1\_PADDING);

if(ret >= 0)

{

strRet = std::string((char\*)pDecode, ret);

}  
 delete [] pDecode;

RSA\_free(pRSAPriKey);

fclose(hPriKeyFile);

CRYPTO\_cleanup\_all\_ex\_data();

return strRet;

}

[复制代码](javascript:void(0);)

     RSA的API中当使用参数RSA\_PKCS1\_PADDING时，明文长度不能大于密文长度-11；当使用参数RSA\_NO\_PADDING时，明文长度需要正好是128。

**AES加密部分代码**：

[复制代码](javascript:void(0);)

std::string EncodeAES( const std::string& password, const std::string& data )

{

AES\_KEY aes\_key;

if(AES\_set\_encrypt\_key((const unsigned char\*)password.c\_str(), password.length() \* 8, &aes\_key) < 0)

{

assert(false);

return "";

}

std::string strRet;

std::string data\_bak = data;

unsigned int data\_length = data\_bak.length();

int padding = 0;

if (data\_bak.length() % AES\_BLOCK\_SIZE > 0)

{

padding = AES\_BLOCK\_SIZE - data\_bak.length() % AES\_BLOCK\_SIZE;

}

data\_length += padding;

while (padding > 0)

{

data\_bak += '\0';

padding--;

}

for(unsigned int i = 0; i < data\_length/AES\_BLOCK\_SIZE; i++)

{

std::string str16 = data\_bak.substr(i\*AES\_BLOCK\_SIZE, AES\_BLOCK\_SIZE);

unsigned char out[AES\_BLOCK\_SIZE];

::memset(out, 0, AES\_BLOCK\_SIZE);

AES\_encrypt((const unsigned char\*)str16.c\_str(), out, &aes\_key);

strRet += std::string((const char\*)out, AES\_BLOCK\_SIZE);

}

return strRet;

}

[复制代码](javascript:void(0);)

**AES解密部分代码**：

[复制代码](javascript:void(0);)

std::string DecodeAES( const std::string& strPassword, const std::string& strData )

{

AES\_KEY aes\_key;

if(AES\_set\_decrypt\_key((const unsigned char\*)strPassword.c\_str(), strPassword.length() \* 8, &aes\_key) < 0)

{

assert(false);

return "";

}

std::string strRet;

for(unsigned int i = 0; i < strData.length()/AES\_BLOCK\_SIZE; i++)

{

std::string str16 = strData.substr(i\*AES\_BLOCK\_SIZE, AES\_BLOCK\_SIZE);

unsigned char out[AES\_BLOCK\_SIZE];

::memset(out, 0, AES\_BLOCK\_SIZE);

AES\_decrypt((const unsigned char\*)str16.c\_str(), out, &aes\_key);

strRet += std::string((const char\*)out, AES\_BLOCK\_SIZE);

}

return strRet;

}

[复制代码](javascript:void(0);)

     AES加密，块大小必须为128位（16字节），如果不是，则要补齐，密钥长度可以选择128位、192位、256位。

**不同语言解密补充**：

　　使用python解密的时候，public key可能要求是PKCS#1格式，而openssl是不支持的，openssl默认是x509格式的public key，为此，如果要把上边生成的public key提供给python使用，需要先从x509格式转换为PKCS#1格式。网络上的资料显示，php有一个api支持这种转换，但我没试过。由于我的私钥是2048位的，所以可以很方便的实现x509转PKCS#1，转换是可逆的，说下PKCS#1转x509的方法：首先删除head和foot的“RSA”，然后在第二行开头增加文本“MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A”，最后，对齐文本。如果私钥不是2048的怎么办呢？可以使用php的api转一下了，或者到网上查找转换的资料解决。[RSA key Formats](http://www.cryptosys.net/pki/rsakeyformats.html)。 2013.11.27.

一些资料：

<http://www.ibm.com/developerworks/cn/linux/l-openssl.html>

<http://www.cnblogs.com/aLittleBitCool/archive/2011/09/22/2185418.html>

明文长度限制：

<http://bbs.chinaunix.net/thread-1860492-1-1.html>

AES加密例子：

<http://blog.csdn.net/wklnewlife/article/details/8244893>

<http://www.lovelucy.info/openssl-aes-encryption.html>

<http://www.thinkemb.com/wordpress/?p=18>

编译资料补充：

<http://developer.covenanteyes.com/building-openssl-for-visual-studio/>