使用SSL前，先有 基本的TCP套接字连接。见demo代码

SSL\_library\_init();//在使用OpenSSL 之前，必须进行相应的协议初始化工作

OpenSSL\_add\_all\_algorithms();///\* 载入所有 SSL 算法 \*/

SSL\_load\_error\_strings();/\* 载入所有 SSL 错误消息 \*/

const SSL\_METHOD \*meth = SSLv23\_client\_method();创建本次会话连接所使用的协议，如果是客户端可以使用

pSockInfo->pSslCtx = SSL\_CTX\_new(meth);//申请SSL 会话环境

/\* 以 SSL V2 和 V3 标准兼容方式产生一个 SSL\_CTX ，即 SSL Content Text \*/

 pSockInfo->pSSL = SSL\_new(pSockInfo->pSslCtx);申请一个SSL 套节字；

SSL\_set\_fd(pSockInfo->pSSL, pSockInfo->sockFd);把SSL attach 到已经连接的套接字上了

ret = SSL\_connect(pSockInfo->pSSL);//成功创建SSL 套接字后，客户端应使用函数SSL\_connect( )替代传统的函数connect( )

//来完成握手过程

DebugPrintf("Connected with %s encryption\n",SSL\_get\_cipher(pSockInfo->pSSL));//;///\*打印所有加密算法的信息(可选)\*/

/\* SSL 库初始化 \*/

    SSL\_library\_init();

    /\* 载入所有 SSL 算法 \*/

    OpenSSL\_add\_all\_algorithms();

    /\* 载入所有 SSL 错误消息 \*/

    SSL\_load\_error\_strings();

    /\* 以 SSL V2 和 V3 标准兼容方式产生一个 SSL\_CTX ，即 SSL Content Text \*/

    ctx = SSL\_CTX\_new(SSLv23\_server\_method());

    /\* 也可以用 SSLv2\_server\_method() 或 SSLv3\_server\_method() 单独表示 V2 或 V3标准 \*/

详细介绍：

现行网上银行和电子商务等大型的网上交易系统普遍采用HTTP和SSL相结合的方式。服务器端采用支持SSL的Web服务器，用户端采用支持SSL的浏览器实现安全通信。

    SSL是Secure Socket Layer（安全套接层协议）的缩写，可以在Internet上提供秘密性传输。Netscape公司在推出第一个Web浏览器的同时，提出了SSL协议标准,目前已有3.0版本。SSL采用公开密钥技术。其目标是保证两个应用间通信的保密性和可靠性,可在服务器端和用户端同时实现支持。目前，利用公开密钥技术的SSL协议，已成为Internet上保密通讯的工业标准。本文着重在SSL协议和SSL程序设计两方面谈谈作者对SSL的理解。

SSL协议初步介绍

    安全套接层协议能使用户/服务器应用之间的通信不被攻击者窃听，并且始终对服务器进行认证，还可选择对用户进行认证。SSL协议要求建立在可靠的传输层协议(TCP)之上。SSL协议的优势在于它是与应用层协议独立无关的，高层的应用层协议(例如：HTTP，FTP，TELNET等)能透明地建立于SSL协议之上。SSL协议在应用层协议通信之前就已经完成加密算法、通信密钥的协商及服务器认证工作。在此之后应用层协议所传送的数据都会被加密，从而保证通信的私密性。

通过以上叙述，SSL协议提供的安全信道有以下三个特性：

1.数据的保密性

信息加密就是把明码的输入文件用加密算法转换成加密的文件以实现数据的保密。加密的过程需要用到密匙来加密数据然后再解密。没有了密钥，就无法解开加密的数据。数据加密之后，只有密匙要用一个安全的方法传送。加密过的数据可以公开地传送。

2.数据的一致性

加密也能保证数据的一致性。例如:消息验证码（MAC），能够校验用户提供的加密信息，接收者可以用MAC来校验加密数据，保证数据在传输过程中没有被篡改过。

3.安全验证

加密的另外一个用途是用来作为个人的标识，用户的密匙可以作为他的安全验证的标识。

    SSL是利用公开密钥的加密技术（RSA）来作为用户端与服务器端在传送机密资料时的加密通讯协定。目前，大部分的Web 服务器及浏览器都广泛支持SSL 技术。当浏览器试图连接一个具有SSL认证加密的服务器时,就会唤醒一个SSL会话,浏览器检查认证,必须具备下面三个条件:

1）有一个权威机构发放证书，当然可以创建自我签订的证书（x509 结构）。

2）证书不能过期。

3）证书是属于它所连接的服务器的。

    只有全部具备了这三个条件,浏览器才能成功完成认证。通过这三个条件,用户能确认其浏览器连接到正确的服务器,而不是连接到一些想盗取用户密码等重要信息的虚假的服务器上。

    在当今的电子商务中还有一项被广泛使用的安全协议是SET协议。SET(Secure Electronic Transaction，安全电子交易)协议是由VISA和MasterCard两大信用卡公司于1997年5月联合推出的规范。SET能在电子交易环节上提供更大的信任度、更完整的交易信息、更高的安全性和更少受欺诈的可能性。SET交易分三个阶段进行:用户向商家购物并确定支付;商家与银行核实;银行向商家支付货款。每个阶段都涉及到RSA对数据加密，以及RSA数字签名。使用SET协议，在一次交易中，要完成多次加密与解密操作，故有很高的安全性，但SET协议比SSL协议复杂，商家和银行都需要改造系统以实现互操作。

在Linux 下，比较流行支持SSL认证的是OpenSSL服务器。OpenSSL项目是一个合作的项目，开发一个健壮的、商业等级的、完整的开放源代码的工具包，用强大的加密算法来实现安全的Socket层（Secure Sockets Layer，SSL v2/v3）和传输层的安全性（Transport Layer Security，TLS v1）。这个项目是由全世界的志愿者管理和开发OpenSSL工具包和相关文档。

     如何在Linux下配置OpenSSL服务器，首先从OpenSSL的主页（）上下载openssl-version.tar.gz软件包来编译安装，与Apache服务器配合可以建立支持SSL的Web服务器，并可以使用自我签订的证书做认证，关于如何编译、安装OpenSSL服务器，可以参考一下OpenSSL HOWTO文档。

SSL 程序设计初步介绍

    SSL 通讯模型为标准的C/S 结构，除了在 TCP 层之上进行传输之外，与一般的通讯没有什么明显的区别。在这里，我们主要介绍如何使用OpenSSL进行安全通讯的程序设计。关于OpenSSL 的一些详细的信息请参考OpenSSL的官方主页 。

在使用OpenSSL前，必须先对OpenSSL 进行初始化，以下的三个函数任选其一:

SSL\_library\_init(void);

OpenSSL\_add\_ssl\_algorithms();

SSLeay\_add\_ssl\_algorithms();

事实上 后面的两个函数只是第一个函数的宏。

如果要使用OpenSSL的出错信息，使用SSL\_load\_error\_strings (void)进行错误信息的初始化。以后可以使用void ERR\_print\_errors\_fp(FILE \*fp) 打印SSL的错误信息。

一次SSL连接会话一般要先申请一个SSL 环境，基本的过程是：

1. SSL\_METHOD\* meth = TLSv1\_client\_method(); 创建本次会话连接所使用的协议，如果是客户端可以使用

SSL\_METHOD\* TLSv1\_client\_method(void); TLSv1.0 协议

SSL\_METHOD\* SSLv2\_client\_method(void); SSLv2 协议

SSL\_METHOD\* SSLv3\_client\_method(void); SSLv3 协议

SSL\_METHOD\* SSLv23\_client\_method(void); SSLv2/v3 协议

服务器同样需要创建本次会话所使用的协议:

SSL\_METHOD \*TLSv1\_server\_method(void);

SSL\_METHOD \*SSLv2\_server\_method(void);

SSL\_METHOD \*SSLv3\_server\_method(void);

SSL\_METHOD \*SSLv23\_server\_method(void);

需要注意的是客户端和服务器需要使用相同的协议。

2．申请SSL会话的环境 CTX，使用不同的协议进行会话，其环境也是不同的。申请SSL会话环境的OpenSSL函数是

SSLK\_CTX\* SSL\_CTX\_new (SSL\_METHOD\*); 参数就是前面我们申请的 SSL通讯方式。返回当前的SSL 连接环境的指针。

然后根据自己的需要设置CTX的属性，典型的是设置SSL 握手阶段证书的验证方式和加载自己的证书。

void SSL\_CTX\_set\_verify (SSL\_CTX\* , int , int\* (int, X509\_STORE\_CTX\*) )

设置证书验证的方式。

第一个参数是当前的CTX 指针，第二个是验证方式，如果是要验证对方的话，就使用 SSL\_VERIFY\_PEER。不需要的话，使用SSL\_VERIFY\_NONE.一般情况下，客户端需要验证对方，而服务器不需要。第三个参数是处理验证的回调函数，如果没有特殊的需要，使用空指针就可以了。

void SSL\_CTX\_load\_verify\_locations(SSL\_CTX\*, const char\* , const char\*);

加载证书;

第一个参数同上，参数二是证书文件的名称，参数三是证书文件的路径;

int SSL\_CTX\_use\_certificate\_file(SSL\_CTX \*ctx, const char \*file, int type);

加载本地的证书;type 指明证书文件的结构类型;失败返回-1

int SSL\_CTX\_use\_PrivateKey\_file(SSL\_CTX \*ctx, const char \*file, int type);

加载自己的私钥；type 参数指明私钥文件的结构类型；失败返回-1

加载了证书和文件之后，就可以验证私钥和证书是否相符：

BOOl SSL\_CTX\_check\_private\_key (SSL\_CTX\*);

3．既然SSL 使用TCP 协议，当然需要把SSL attach 到已经连接的套接字上了：

SSL\* SSL\_new (SSL\_CTX\*); 申请一个SSL 套节字；

int SSL\_set\_rfd (SSL\*); 绑定只读套接字

int SSL\_set\_wfd (SSL\*); 绑定只写套接字

int SSL\_set\_fd ( SSL\*); 绑定读写套接字

绑定成功返回 1， 失败返回0;

4． 接下来就是SSL 握手的动作了

int SSL\_connect (SSL\*); 失败返回 -1

5. 握手成功之后，就可以进行通讯了，使用SSL\_read 和SS\_write 读写SSL 套接字代替传统的read 、write

int SSL\_read (SSL \*ssl, char \*buf, int num );

int SSL\_write (SSL \*ssl, char \*buf, int num);

如果是服务器，则使用 SSL\_accept 代替传统的 accept 调用

int SSL\_accept(SSL \*ssl);

6. 通讯结束，需要释放前面申请的 SSL资源

int SSL\_shutdown(SSL \*ssl); 关闭SSL套接字;

void SSL\_free (ssl); 释放SSL套接字;

void SSL\_CTX\_free (ctx); 释放SSL环境;

    OpenSSL 虽然已经发展到了0.9.96版本，但是它的文档还很少，甚至连最基本的man 函数手册都没有完成。所以，本文紧紧是讲述了使用OpenSSL 进行程序设计的框架。更加详细的资料可以参考OpenSSL 的文档或者 Apache mod\_ssl 的文档。

    通过以上的介绍，我想读者对SSL协议已经有了一定的了解，作者有机会将会继续给大家介绍SSL协议的其他方面的内容。

框架图：



