

OpenSSL 使用指南

0. 7p

目录

- | | |
|-------------------|-----------------------------|
| 1. 介绍 | intro |
| 2. 编译 | build |
| 3. 运行 OpenSSL.exe | openssl.exe |
| 4. 算法编程 API | Alg API |
| 4.1 对称算法 | |
| 4.1.1 DES | |
| 4.1.2 AES | |
| 4.1.3 RC4 | |
| 4.1.4 EVP_ | |
| 4.2 公钥算法 | |
| 4.3 Hash 算法 | |
| 4.4 随机数算法 | |
| 5. SSL 协议编程 API | SSL |
| 6. CA 和证书 | CA |
| 7. 参考网址 | REF |
| 8. | |
| A. 示例程序 | demo |
| B. 其他 | |
| C. 其他示例程序 | |
| D. | |

1. 介绍

OpenSSL 是使用非常广泛的 SSL 的开源实现。由于其中实现了为 SSL 所用的各种加密算法，因此 OpenSSL 也是被广泛使用的加密函数库。

1.1 SSL

SSL (Secure Socket Layer) 安全协议是由 Netscape 公司首先提出，最初用在保护 Navigator 浏览器和 Web 服务器之间的 HTTP 通信 (即 HTTPS)。后来 SSL 协议成为传输层安全通信事实上的标准，并被 IETF 吸收改进为 TLS (Transport Layer Security) 协议。

SSL/TLS 协议位于 TCP 协议和应用层协议之间，为传输双方提供认证、加密和完整性保护等安全服务。SSL 作为一个协议框架，通信双方可以选用合适的对称算法、公钥算法、MAC 算法等密码算法实现安全服务。

1.2 SSL 的工作原理

参见 <http://developer.netscape.com/tech/security/ssl/howitworks.html>。

1.3 OpenSSL

OpenSSL 是著名的 SSL 的开源实现，是用 C 语言实现的。

OpenSSL 的前身是 SSLeay，一个由 Eric Young 开发的 SSL 的开源实现，支持 SSLv2/v3 和 TLSv1。

伴随着 SSL 协议的普及应用，OpenSSL 被广泛应用在基于 TCP/Socket 的网络程序中，尤其是 OpenSSL 和 Apache 相结合，是很多电子商务网站服务器的典型配置。

2. 编译和安装 OpenSSL

OpenSSL 开放源代码，这对学习、分析 SSL 和各种密码算法提供了机会，也便于在上面进一步开发。

2.1 获得 OpenSSL

到 OpenSSL 的网站即可下载当前版本的 OpenSSL 源代码压缩包。

当前版本 openssl-0.9.8.tar.gz，只有 3M 多，比较精简。解压缩后得到一个目录 openssl-0.9.8，共有约 1800 个文件，15M。其中 crypto 子目录中是众多密码算法实现，ssl 子目录中是 SSL 协议的实现。

在 Linux 中解压缩：

```
$tar xzf openssl-0.9.8.tar.gz
```

在 Windows 中可以使用 winzip 或 winrar。

2.2 编译工具

编译 OpenSSL 需要 Perl 和 C 编译器。在 Windows 下如果要用加密算法的汇编代码实现，还需要 masm 或 nasm 汇编器。（汇编代码可以比 C 代码显著提高密码运算速度）

Perl 在 Windows 下推荐使用 Active Perl。

C 编译器可以使用 gcc。在 Windows 下可以使用 Visual C++ 编译器。

汇编器推荐使用 nasm。

这些工具所在目录必须加入到 PATH 环境变量中去。

2.3 编译和安装步骤

查看 ./readme 是个好习惯。从 readme 了解到需要进一步查看 INSTALL 和 INSTALL.W32 文件。

在 Windows 中：

```
>perl Configure VC-WIN32
```

```
>ms\do_nasm (如果不使用汇编代码实现，则可>ms\do_ms)
```

```
>nmake -f ms\ntdll.mak
```

```
>cd out32dll
```

```
>..\ms\test
```

```
or
```

```
>nmake -f ms\ntdll.mak test // after 0.9.8b
```

编译结果得到头文件、链接库、运行库和 openssl.exe 工具。头文件位于 ./inc32 或者 ./include 目录，有一个 openssl 子目录，内有几十个 .h 文件。链接库即 ./out32dll 目录中的 libeay32.lib 和 ssleay32.lib，分别是密码算法相关的和 ssl 协议相关的。运行库是 ./out32dll 目录中的 libeay32.dll 和 ssleay32.dll，和链接库相对应。在 ./out32dll 中还有一个工具 openssl.exe，可以直接用来测试性能、产生 RSA 密钥、加解密文件，甚至

可以用来维护一个测试用的 CA。

在 Linux 中的编译和安装步骤较简单：

```
$. /config
```

```
$make
```

```
$make test
```

```
$make install
```

在 Linux 下，头文件、库文件、工具都已被安装放到了合适的位置。库文件是 .a 或 .so 格式。

3. 使用 OpenSSL.exe

使用 OpenSSL.exe (Linux 中可执行文件名是 openssl) 可以做很多工作, 是一个很好的测试或调试工具。

3.1 版本和编译参数

显示版本和编译参数: `>openssl version -a`

3.2 支持的子命令、密码算法

查看支持的子命令: `>openssl ?`

SSL 密码组合列表: `>openssl ciphers`

3.3 测试密码算法速度

测试所有算法速度: `>openssl speed`

测试 RSA 速度: `>openssl speed rsa`

测试 DES 速度: `>openssl speed des`

3.4 RSA 密钥操作

产生 RSA 密钥对: `>openssl genrsa -out 1.key 1024`

取出 RSA 公钥: `>openssl rsa -in 1.key -pubout -out 1.pubkey`

3.5 加密文件

加密文件: `>openssl enc -e -rc4 -in 1.key -out 1.key.enc`

解密文件: `>openssl enc -d -rc4 -in 1.key.enc -out 1.key.dec`

3.6 计算 Hash 值

计算文件的 MD5 值: `>openssl md5 < 1.key`

计算文件的 SHA1 值: `>openssl sha1 < 1.key`

4. 算法编程 API

OpenSSL 中支持众多的密码算法，并提供了很好的封装和接口。密码算法主要分为如下几类：对称算法、公钥算法、散列算法、随机数产生算法等。

OpenSSL 的目标是实现安全协议。其中相关协议和标准包括：SSL/TLS、PKCS#1、PKCS#10、X.509、PEM、OCSP 等。

4.1 对称算法接口

OpenSSL 中实现的对称算法太多，举三个例子：DES、AES、RC4。

4.1.1 DES

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 比特密文。在实际使用中把密钥看作 64 比特可以更方便。

$$\text{DES}(\text{IN}, \text{KEY}) = \text{OUT}$$

(1) DES ECB 模式

在 OpenSSL 中 ECB 操作模式对应的函数是 `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` 指示是加密还是解密。该函数每次只加密一个分组，因此用来加密很多数据时不方便使用。

```
void    DES_ecb_encrypt(const_DES_cblock    *input, DES_cblock    *output,
DES_key_schedule *ks, int enc);

int DES_set_key(const_DES_cblock *key, DES_key_schedule *schedule);
```

(2) DES CBC 模式

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);
```

(3) DES CFB 模式

DES 算法 CFB 操作模式加解密函数是 `DES_cfb_encrypt()`。参数 `length` 指示输入字节长度。参数 `numbits` 则指示了 CFB 每次循环加密多少明文比特，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。如果在一次会话中数次调用 `DES_cfb_encrypt()`，则应该记忆 `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);
```

另有一个 `numbit` 是 64 比特的版本，既高效又没有填充的麻烦，推荐使用。`num` 中的返

返回值指示了 `ivec` 中的状态，是和下次调用衔接的。

```
void DES_cfb64_encrypt(const unsigned char *in, unsigned char *out, long length,
DES_key_schedule *schedule, DES_cblock *ivec, int *num, int enc);
```

(4) DES OFB 模式

OFB 和 CFB 类似，也有两个函数，用法一样。

```
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);
```

(5) DES 函数示例程序

见附件 A. 1。

4.1.2 AES

AES 加密算法是分组算法。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。

$AES(IN, KEY) = OUT$

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();
```

AES 示例程序见附件 A. 2。

4.1.3 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 RC4_set_key(RC4_KEY *key, int len, const unsigned char *data);
```

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

RC4 示例程序见附件 A. 3。

例子 A. 3. (1) 是利用 OpenSSL 动态库函数。例子 A. 3. (2) 是把 RC4 的实现代码从 OpenSSL 中分离出来的。例子 A. 3. (3) 是另一个演示实现。

4.2 公钥算法

OpenSSL 中实现了 RSA、DSA、ECDSA 等公钥算法。

4.2.1 RSA

RSA 是分组算法，典型的密钥模长度 1024 比特时，分组即是 1024 比特，即 128 字节。

(1) RSA 密钥

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);
```

如果从文件中读取密钥，可使用函数 `PEM_read_bio_PrivateKey()`/`PEM_read_bio_PUBKEY()`；`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);
```

(2) RSA 加密解密

RSA 加密函数 `RSA_public_encrypt()` 使用公钥部分，解密函数 `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 个字节（在 `RSA_PKCS1_PADDING` 时？）。输出长度等于 RSA 钥的模长。

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

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

(3) 签名和验证

签名使用私钥，验证使用公钥。RSA 签名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` 或 `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);
```

(4) RSA 函数示例程序

RSA 示例程序见附件 A.4。

例子 A.4. (1) 是加密解密例子。例子 A.4. (2) 是签名验证例子。

4.2.2 DSA

(TOBE)

4.2.2 ECDSA

(TODO or NOT TODO)

4.3 Hash 算法

Hash 算法举 MD5 和 SHA1 两个例子。Hash 算法重复接收用户输入，直到最后一次结束时输出散列结果。

4.3.1 MD5

MD5 算法输出的散列值是 16 字节。

```
int MD5_Init(MD5_CTX *c);
int MD5_Update(MD5_CTX *c, const void *data, size_t len);
int MD5_Final(unsigned char *md, MD5_CTX *c);
```

4.3.2 SHA1

SHA1 算法输出的散列值是 20 字节。

```
int SHA1_Init(SHA_CTX *c);
int SHA1_Update(SHA_CTX *c, const void *data, size_t len);
int SHA1_Final(unsigned char *md, SHA_CTX *c);
```

4.3.3 MD5 例子

MD5 示例程序见附件 A.5。

md5sum 这是一个实用小工具，可以计算一个文件的 MD5 值。

4.4 随机数算法

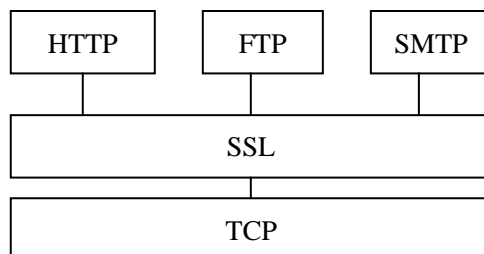
随机性是密码安全的基石。为了产生安全的伪随机数，必须有好的随机因素作为种子。OpenSSL 在内部做了努力，但是仍建议在实用随机数产生函数之前添加随机因素。

函数 RAND_add() 可以添加随机因素到内部状态中去。然后，即可以使用 RAND_bytes() 获得随机数。

```
void RAND_add(const void *buf, int num, double entropy);
int RAND_bytes(unsigned char *buf, int num);
```

5. SSL 协议编程 API

SSL 协议的主要功能即把 TCP 的字节流变成了一个安全的流，所有基于 TCP 的程序可以很容易地采用 SSL 协议。主要的变化在于 `accept/connect()` 变化为使用 OpenSSL 提供的 `SSL_accept/SSL_connect()`，`read/write()` 变化为 `SSL_read/SSL_write()`。



5.0 初始化函数库

```
SSL_load_error_strings();
OpenSSL_add_ssl_algorithms(); // OpenSSL_add_all_algorithms();
```

5.1 客户端

```
SSLv3_client_method();
SSL_CTX_new();
SSL_CTX_set_accepted_Cas(); // 认可的 CAs
SSL_CTX_use_certificate_file(); // 自己的证书
SSL_CTX_use_PrivateKey_file(); // 自己的私钥
SSL_CTX_check_private_key(); // 检查证书-私钥一致性
SSL_CTX_set_cipher(); // 自己喜欢的算法组合
```

5.2 服务器端

```
SSLv3_server_method();
SSL_CTX_new();
SSL_CTX_set_accepted_Cas();
SSL_CTX_use_certificate_file();
SSL_CTX_use_PrivateKey_file();
SSL_CTX_check_private_key();
SSL_CTX_set_cipher();
```

5.3 SSL 示例程序

参见 A.6。

6. CA 和证书

公钥是不需要保密的。事实上，公钥需要可靠地发布、彻底地公开。很不幸，公钥的公开是应用密码学的最后一个难题：当你面对一个公钥（就是两个数 n 和 e ）时，你怎么知道这是谁的公钥？

为了让人相信这个公钥就是某个人的，一般要有一个权威机构（即 CA）的保证。CA 用自己的私钥给公钥和持有人信息签名。任何人可以用 CA 的公钥验证签名，以此相信这个公钥是这个人的。

CA 的公钥怎么来呢？一般是事先内置在程序中的，或者通过其他可靠的方法取得。

6.0 CA 的作用

OpenSSL 中有一个小巧但可以用作测试的 CA。如果想使用一个功能全面强大的 CA，建议使用 EJBCA 或 OpenCA。Windows Server 中有个可选的 CA 组件，对于支持 IIS、IE 和 Outlook 中的使用是方便的。使用商业 CA（如山东省 CA）颁发的证书应该也是可以的。

6.1 OpenSSL 中 CA 的配置示例

参见 A.7. (1)。

6.3 证书解析和示例程序

参见 A.7. (2)。

7. 参考网址

SSL 3.0 Specification <http://www.netscape.com/eng/ssl3/>

Transport Layer Security (tls) Charter

<http://www.ietf.org/html.charters/tls-charter.html>

OpenSSL: The Open Source toolkit for SSL/TLS

<http://www.openssl.org/>

SSLeay

<http://www2.psy.uq.edu.au/~ftp/Crypto/>

OpenSSL 中文论坛

<http://openssl.cn/>

Perl

<http://www.cpan.org/src/README.html>

ActivePerl

<http://www.activestate.com/Products/ActivePerl/>

NASM

<http://www.perl.com/>

山东省 CA

<http://www.sdca.com.cn/>

A. 示例程序

注: 此嵌入的文件对象可以被“拖放”到磁盘目录中去。

1. DES 示例程序



2. AES 示例程序



3. RC4 示例程序



(1).

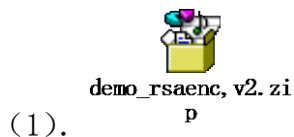


(2).

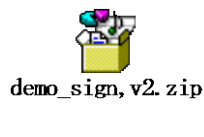


(3)

4. RSA 示例程序



(1).



(2).

5. Hash 算法示例程序



6. SSL 示例程序



7. CA 配置示例和证书解析示例程序



(1).



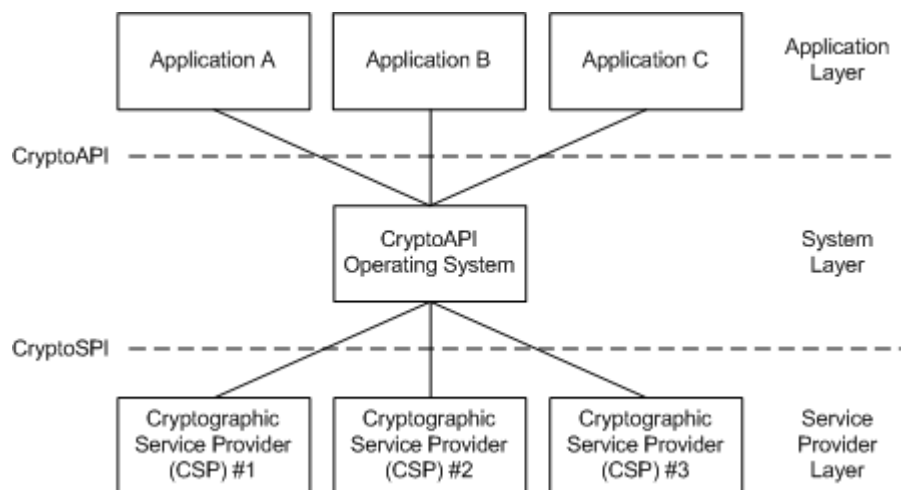
(2).

8.

B. 其他

[1] CSP - Cryptographic Service Provider (in Windows)

美国不准出口高强度的密码技术和产品。从安全考虑, 很多国家也禁止使用进口的密码系统。因此, 像 Windows 等产品提供一种折中的方法, 即顶层使用统一的接口, 而底层允许用户使用自己的密码实现。



[2]

C. 其他示例程序

[1] DES/RC2 加解密 in C#

使用 MS CSP, C# in Windows



ConsoleApplication1.zip

[2] MD5/DES in Java



encrywithkey.java