

转 源码方式向openssl中添加新算法完整详细步骤（示例：摘要算法SM3）【非engine方式】

2016年12月07日 19:00:06 阅读数：931 更多

看了本文章，可能对openssl库精简有帮助，转过来保存一下

本文转自：http://blog.csdn.net/luckymelina/article/details/38926145

openssl简介

openssl是一个功能丰富且自包含的开源安全工具箱。它提供的主要功能有：SSL协议实现(包括SSLv2、SSLv3和TLSv1)、大量软算法(对称/非对称/摘要)、大数运算、非对称算法密钥生成、ASN.1编解码库、证书请求(PKCS10)编解码、数字证书编解码、CRL编解码、OCSP协议、数字证书验证、PKCS7标准实现和PKCS12个人数字证书格式实现等功能。

openssl采用C语言作为开发语言，这使得它具有优秀的跨平台性能。openssl支持Linux、UNIX、windows、Mac等平台。openssl目前最新的版本是openssl-1.0.1i。

更多的相关知识请参阅铺天盖地的各种官方非官方文档。今天来实现怎样将一个新算法添加进openssl中。

问题描述

openssl中虽然集成了很多经典算法，对称算法、非对称算法、摘要算法等等，但有时候我们需要用到自定义的算法，比如要实现一个自定义的摘要算法，需要借助openssl带的大数库来实现，同时又希望将自己实现的算法封装到openssl里面，以便统一调用或者方便项目改造和替换。最具现实意义的案例就是国密SM2等的改造。本例将用国密中的摘要算法SM3作为添加对象，在windows下面进行编译调试，仅描述添加步骤，所以本文的前提是已经完成了基于openssl大数库的SM3的c语言实现。本文仅供依葫芦画瓢，要理解原理请自行查阅别的资料或者自己解读源码，或者等我的下一篇文章。

具体步骤

一、准备工作

- 1、下载openssl源码，openssl目前最新的版本是openssl-1.0.1i。下载地址：
- 2、sm3的c语言实现。
- 3、windows下perl、vc的安装配置。
- 4、熟悉openssl的编译过程。

二、SM3代码改造【这个步骤涉及到的文件全部新建，存放目录关系如图】

1、涉及到的文件7个：

D:\Program Files\openssl-1.0.1i\crypto\sm3\sm3.c (10 hits)
D:\Program Files\openssl-1.0.1i\crypto\sm3\sm3.h (30 hits)
D:\Program Files\openssl-1.0.1i\crypto\sm3\sm3_dgst.c (16 hits)
D:\Program Files\openssl-1.0.1i\crypto\sm3\sm3_locl.h (3 hits)
D:\Program Files\openssl-1.0.1i\crypto\sm3\sm3_one.c (10 hits)
D:\Program Files\openssl-1.0.1i\crypto\sm3\sm3test.c (9 hits)
D:\Program Files\openssl-1.0.1i\crypto\sm3\Makefile (26 hits)

1、【新建文件】sm3.h

这是最重要的一个头文件，里面定义了算法可以导出的算法，也就是里面声明的每一个函数最后都会生成对应的libey.num(将在第五章中具体讲)，能够导出到dll中供外部调用。几个主要的声明如下：

定义sm3的摘要长度：

```
#define SM3_DIGEST_LENGTH 32 /* sm3 摘要长度为256位 32字节，md5的摘要长度128位 not sure*/
```

定义SM3_CTX，这个结构其实就是对应evp封装中EVP_MD_CTX的md_data：

```
01. typedef struct SM3state_st
02. {
03.     unsigned long long total_length;
04.     unsigned char message_buffer[64];
05.     size_t message_buffer_position;
06.     size_t V_i[8];
07.     size_t V_i_1[8];
08.     size_t T_j[64];
09. } SM3_CTX;
```

声明函数，包括init、update、final，这几个函数会在封装的时候被内部调用，当然在dll中导出了也可以外部调用：

```
01. int SM3_Init(SM3_CTX *c);
02. int SM3_Update(SM3_CTX *c, const void *data, size_t len);
03. int SM3_Final(unsigned char *md, SM3_CTX *c);
04. unsignedchar *SM3(unsigned char *d, size_t n, unsigned char *md);
```

2、【新建文件】sm3_locl.h

这个头文件中包含一些要用到的宏定义。主要定义了本算法实现中需要用到内部函数，不能被导出到dll中供外部调用。

宏定义：

```
01. #ifndef ROTATE
```

```

02. #define ROTATE(a,n) (((a)<<(n))|(((a)&0xffffffff)>>(32-(n))))
03. #endif
04.
05. #define EE(b,c,d) ((b & c) | ((~ b) & d))
06. #define FF(b,c,d) (((c)^(d)) & (b)) ^ (d))
07. #define GG(b,c,d) (((b)& (c)) | ((b) & (d)) | ((c) & (d)))
08. #define HH(b,c,d) ((b)^(c) ^ (d))
09. #define HH1(a) (a^( ROTATE(a, 9)) ^ ( ROTATE(a, 17)))
10. #define HH2(a) (a^( ROTATE(a, 15)) ^ ( ROTATE(a, 23)))
11. 内部函数：
12. void init_T_j(size_t *T_j);
13. void init_V_i(size_t *V_i);
14. void CF(size_t *T_j, size_t *V_i, unsigned char *B_i, size_t *V_i_1);

```

3、【新建文件】sm3_dgst.c

实现sm3.h和sm3_locl中声明的函数，除了函数SM3（将在sm3_one.c中实现）。

4、【新建文件】sm3_one.c

实现sm3.h中声明的函数SM3。

5、【新建文件】sm3.c

实现对文件的摘要，如用在openssl作为命令行工具，文件名和摘要算法名作为选项参数实现使用算法对文件进行摘要。

6、【新建文件】sm3test.c

算法的test文件。

7、【新建文件】Makefile

编译的Makefile文件。里面描述在编译的时候以上的几个文件的角色、编译过程、依赖关系等。

三、EVP封装相关代码

1、涉及到的文件4个：

```

D:\Program Files\openssl-1.0.1i\crypto\evp\c_allid.c (2 hits)
D:\Program Files\openssl-1.0.1i\crypto\evp\evp.h (2 hits)
D:\Program Files\openssl-1.0.1i\crypto\evp\m_sm3.c (14 hits)
D:\Program Files\openssl-1.0.1i\crypto\evp\Makefile (17 hits)

```

2、【手动修改】evp.h

evp在openssl里面的主要做的是封装的事情，因此首先在evp.h头文件中添加新算法sm3：

```

[cpp]
01. #ifndef OPENSSSL_NO_SM3
02. const EVP_MD *EVP_sm3(void);
03. #endif

```

```

D:\Program Files\openssl-1.0.1i\crypto\evp\evp.h (2 hits)
Line 673: #ifndef OPENSSSL_NO_SM3
Line 674: const EVP_MD *EVP_sm3(void);

```

说明：EVP_sm3()函数将返回返回一个sm3的EVP_MD的结构，这个结构以及EVP_sm3()函数都在m_sm3.c中定义。

3、【新建文件】m_sm3.c

这是放在EVP目录中关于新算法的一个重要文件，定义sm3的EVP_MD结构，EVP_MD结构原型在evp.h中：

```

#ifndef EVP_MD
struct env_md_st
{
    int type;
    int pkey_type;
    int md_size;
    unsigned long flags;
    int (*init)(EVP_MD_CTX *ctx);
    int (*update)(EVP_MD_CTX *ctx, const void *data, size_t count);
    int (*final)(EVP_MD_CTX *ctx, unsigned char *md);
    int (*copy)(EVP_MD_CTX *to, const EVP_MD_CTX *from);
    int (*cleanup)(EVP_MD_CTX *ctx);

    /* FIXME: prototype these some day */
    int (*sign)(int type, const unsigned char *m, unsigned int m_length,
                unsigned char *sigret, unsigned int *siglen, void *key);
    int (*verify)(int type, const unsigned char *m, unsigned int m_length,
                  const unsigned char *sigbuf, unsigned int siglen,
                  void *key);
    int required_pkey_type[5]; /*EVP_PKEY_xxx */
    int block_size;
    int ctx_size; /* how big does the ctx->md_data need to be */
    /* control function */
    int (*md_ctrl)(EVP_MD_CTX *ctx, int cmd, int p1, void *p2);
} /* EVP_MD */;

```

sm3的EVP_MD结构：

```

[cpp]
01. static const EVP_MD sm3_md=
02. {
03.     NID_sm3, //这个需要定义了OID之后生成，这个步骤在第四章。
04.     NID_sm3WithRSAEncryption, //这个地方sm3WithRSA只做示例

```

```

05.     SM3_DIGEST_LENGTH, //SM3的摘要长度, 在sm3.h中定义
06.     0, //flag
07.     init, //结构中的init函数, 指向自己声明在sm3.h中的SM3_Init
08.     update, //同上
09.     final, //同上
10.     NULL,
11.     NULL,
12.     EVP_PKEY_RSA_method, //同上, 只做示例
13.     SM3_CBLOCK, //在sm3.h中声明
14.     sizeof(EVP_MD *)+sizeof(SM3_CTX),
15.     };

```

4、【手动修改】c_alld.c

openssl的算法封装之后使用时需要加载算法，实现SM3的添加：

```

[cpp]
01. #ifndef OPENSSL_NO_SM3
02.     EVP_add_digest(EVP_sm3());
03. #endif

```

```

D:\Program Files\openssl-1.0.1i\crypto\evp\c_alld.c (2 hits)
Line 70: #ifndef OPENSSL_NO_SM3
Line 71:     EVP_add_digest(EVP_sm3());

```

5、【手动修改】Makefile

将编译时候的LIBSRC、LIBOBJ中加入SM3新算法，并且添加sm3各种头文件的依赖关系。

```

D:\Program Files\openssl-1.0.1i\crypto\evp\Makefile (17 hits)
Line 25:     m_null.c m_md2.c m_md4.c m_sm3.c m_md5.c m_sha.c m_shal.c m_wp.c \
Line 38:     m_null.o m_md2.o m_md4.o m_sm3.o m_md5.o m_sha.o m_shal.o m_wp.o \
Line 504: m_sm3.o: ../../e_os.h ../../include/openssl/asn1.h ../../include/openssl/bio.h
Line 505: m_sm3.o: ../../include/openssl/buffer.h ../../include/openssl/crypto.h
Line 506: m_sm3.o: ../../include/openssl/e_os2.h ../../include/openssl/ec.h
Line 507: m_sm3.o: ../../include/openssl/ecdh.h ../../include/openssl/ecdsa.h
Line 508: m_sm3.o: ../../include/openssl/err.h ../../include/openssl/evp.h
Line 509: m_sm3.o: ../../include/openssl/lhash.h ../../include/openssl/sm3.h
Line 509: m_sm3.o: ../../include/openssl/lhash.h ../../include/openssl/sm3.h
Line 510: m_sm3.o: ../../include/openssl/obj_mac.h ../../include/openssl/objects.h
Line 511: m_sm3.o: ../../include/openssl/opensslconf.h ../../include/openssl/opensslv.h
Line 512: m_sm3.o: ../../include/openssl/ssl_typ.h ../../include/openssl/pkcs7.h
Line 513: m_sm3.o: ../../include/openssl/rsa.h ../../include/openssl/safestack.h
Line 514: m_sm3.o: ../../include/openssl/sha.h ../../include/openssl/stack.h
Line 515: m_sm3.o: ../../include/openssl/symhacks.h ../../include/openssl/x509.h
Line 516: m_sm3.o: ../../include/openssl/x509_vfy.h ../../cryptlib.h evp_locl.h m_sm3.c
Line 516: m_sm3.o: ../../include/openssl/x509_vfy.h ../../cryptlib.h evp_locl.h m_sm3.c

```

四、OID生成

1、涉及到的文件7个：

```

D:\Program Files\openssl-1.0.1i\crypto\objects\obj_mac.h (12 hits)
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_mac.num (2 hits)
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_xref.h (2 hits)
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_xref.txt (2 hits)
D:\Program Files\openssl-1.0.1i\crypto\objects\objects.h (12 hits)
D:\Program Files\openssl-1.0.1i\crypto\objects\objects.txt (4 hits)
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_dat.h (14 hits)

```

2、【手动修改】objects.txt

添加sm3的OID，特别说明：此处仅作参考，sm3的实际OID是：

```

[cpp]
01. rsadsi 2 12 : SM3 : sm3

```

添加sm3WithRSAEncryption，这个是对应PKCS1的，只是为了让sm3的EVP_MD结构有值填充，实际添加sm3的本意不用于RSA,这里仅作参考。

```

[cpp]
01. pkcs1 15 : RSA-SM3 : sm3WithRSAEncryption

```

```

D:\Program Files\openssl-1.0.1i\crypto\objects\objects.txt (4 hits)
Line 177: pkcs1 15 : RSA-SM3 : sm3WithRSAEncryption
Line 177: pkcs1 15 : RSA-SM3 : sm3WithRSAEncryption
Line 362: rsadsi 2 12 : SM3 : sm3
Line 362: rsadsi 2 12 : SM3 : sm3

```

3、【执行命令自动更新】obj_mac.h和obj_mac.num

打开cmd，切换到目录下源码的crypto\objects\所在的目录下，

执行命令：perl objects.pl objects.txt obj_mac.num obj_mac.h

```
C:\Users\xiaohei\Desktop\openssl-1.0.1i\crypto\objects>perl objects.pl objects.txt
xt obj_mac.num obj_mac.h
Added OID sm3WithRSAEncryption
Added OID sm3
http://blog.csdn.net/luckymelina
```

obj_mac.h和obj_mac.num这两个文件都会因为objects.txt的修改而更新。增加的内容如：

```
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_mac.h (12 hits)
Line 618: #define SN_sm3WithRSAEncryption "RSA-SM3"
Line 618: #define SN_sm3WithRSAEncryption "RSA-SM3"
Line 619: #define LN_sm3WithRSAEncryption "sm3WithRSAEncryption"
Line 619: #define LN_sm3WithRSAEncryption "sm3WithRSAEncryption"
Line 620: #define NID_sm3WithRSAEncryption 920
Line 621: #define OBJ_sm3WithRSAEncryption OBJ_pkcs1,15L
Line 1150: #define SN_sm3 "SM3"
Line 1150: #define SN_sm3 "SM3"
Line 1151: #define LN_sm3 "sm3"
Line 1151: #define LN_sm3 "sm3"
Line 1152: #define NID_sm3 921
Line 1153: #define OBJ_sm3 OBJ_rsads1,2L,12L
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_mac.num (2 hits)
Line 920: sm3WithRSAEncryption 920
Line 921: sm3 921
http://blog.csdn.net/luckymelina
```

4、【手动修改】objects.h

将obj_mac.h中新增的内容同步到objects.h中。（其实这一步不做也没有影响，因为之后不用这个文件来生成obj_dat.h，objects.h的内容没有obj_mac.h的内容全面，千万不能用objects.h来obj_dat.h，一定要用obj_mac.h来生成才是正确的。切记，血的教训！）

5、【执行命令自动更新】obj_dat.h

执行命令：perl obj_dat.pl obj_mac.h obj_dat.h

```
C:\Users\xiaohei\Desktop\openssl-1.0.1i\crypto\objects>perl obj_dat.pl objects.h
obj_dat.h
http://blog.csdn.net/luckymelina
```

将会自动更新obj_dat.h这个文件，新增的内容如下：

```
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_dat.h (14 hits)
Line 922: 0x2A,0x86,0x48,0x86,0x86,0xF7,0x0D,0x01,0x01,0x0F,/* [5973] OBJ_sm3WithRSAEncryption */
Line 923: 0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x0C,/* [5982] OBJ_sm3 */
Line 2404: {"RSA-SM3","sm3WithRSAEncryption",NID_sm3WithRSAEncryption,9,
Line 2404: {"RSA-SM3","sm3WithRSAEncryption",NID_sm3WithRSAEncryption,9,
Line 2404: {"RSA-SM3","sm3WithRSAEncryption",NID_sm3WithRSAEncryption,9,
Line 2406: {"SM3","sm3",NID_sm3,8,&(lvalues[5982]),0},
Line 2406: {"SM3","sm3",NID_sm3,8,&(lvalues[5982]),0},
Line 2406: {"SM3","sm3",NID_sm3,8,&(lvalues[5982]),0},
Line 2574: 920,/* "RSA-SM3" */
Line 2587: 921,/* "SM3" */
Line 4198: 921,/* "sm3" */
Line 4199: 920,/* "sm3WithRSAEncryption" */
Line 4672: 921,/* OBJ_sm3 1 2 840 113549 2 12 */
Line 4843: 920,/* OBJ_sm3WithRSAEncryption 1 2 840 113549 1 1 15 */
http://blog.csdn.net/luckymelina
```

注意看图上最后两排

```
[cpp]
01. Line 4672: 921,/* OBJ_sm3 1 2 840 113549 2 12 */
02. Line 4843: 920,/* OBJ_sm3WithRSAEncryption 1 2 840 113549 1 1 15 */
```

最后带的一串数字本来是算法的OID，因为上面第2步的objects.txt中是乱填的，所以这个不是真实的OID。仅作参考。

6、【手动修改】obj_xref.txt

添加：

```
[cpp]
01. sm3WithRSAEncryption sm3 rsaEncryption
```

```
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_xref.txt (2 hits)
Line 10: sm3WithRSAEncryption sm3 rsaEncryption
Line 10: sm3WithRSAEncryption sm3 rsaEncryption
http://blog.csdn.net/luckymelina
```

7、【执行命令自动更新】obj_xref.h

执行命令：perl objxref.pl obj_xref.txt obj_xref.h

obj_xref.h更新后的添加的内容如：

```
D:\Program Files\openssl-1.0.1i\crypto\objects\obj_xref.h (2 hits)
Line 24: {NID_sm3WithRSAEncryption, NID_sm3, NID_rsaEncryption},
Line 24: {NID_sm3WithRSAEncryption, NID_sm3, NID_rsaEncryption},
http://blog.csdn.net/luckymelina
```

五、make相关设置

1、涉及到的文件util\下面5个：

```
D:\Program Files\openssl-1.0.1i\util\sp-diff.pl (1 hit)
D:\Program Files\openssl-1.0.1i\util\libey.num (14 hits)
D:\Program Files\openssl-1.0.1i\util\mk1mf.pl (8 hits)
D:\Program Files\openssl-1.0.1i\util\mkdef.pl (11 hits)
D:\Program Files\openssl-1.0.1i\util\mkfiles.pl (1 hit)
```

2、【手动修改】mkfiles.pl

添加：

```
[cpp]
01. "crypto/sm3",
```

```
D:\Program Files\openssl-1.0.1i\util\mkfiles.pl (1 hit)
Line 14: "crypto/sm3",
```

添加上包含sm3的算法源码的目录，这个目录中包含sm3的Makefile文件。之后就会按照Makefile进行编译。

3、【手动修改】mkdef.pl

添加内容如图：

```
D:\Program Files\openssl-1.0.1i\util\mkdef.pl (11 hits)
Line 85: "CAST", "MD2", "MD4", "SM3", "MD5", "SHA", "SHA0", "SHA1",
Line 135: my $no_md2; my $no_md4; my $no_sm3; my $no_md5; my $no_sha; my $no_ripemd; my $no_md2;
Line 203: elsif (/^no-sm3$/) { $no_sm3=1; }
Line 203: elsif (/^no-sm3$/) { $no_sm3=1; }
Line 296: $crypto.=" crypto/sm3/sm3.h" ; # unless $no_sm3;
Line 296: $crypto.=" crypto/sm3/sm3.h" ; # unless $no_sm3;
Line 296: $crypto.=" crypto/sm3/sm3.h" ; # unless $no_sm3;
Line 958: $a .= "SM3" if($s =~ /EVP_sm3/);
Line 958: $a .= "SM3" if($s =~ /EVP_sm3/);
Line 1173: if ($keyword eq "SM3" && $no_sm3) { return 0; }
Line 1173: if ($keyword eq "SM3" && $no_sm3) { return 0; }
```

加上新算法的头文件等，可以在dll中导出算法函数。

4、【执行命令自动更新】libeay.num

执行命令：perl util/mkdef.pl crypto update

```
C:\Users\xiaohei\Desktop\openssl-1.0.1i>perl util/mkdef.pl crypto update
WARNING: mkdef.pl doesn't know the following algorithms:
SM2
Updating LIBEAY info
No old symbols needed info update
Rewriting LIBEAY
Updating LIBEAY numbers
8 New symbols added
C:\Users\xiaohei\Desktop\openssl-1.0.1i>
```

Libeay.num更新后增加的内容如：

```
4313 BIO_dgram_is_sctp 4681 EXIST::FUNCTION:SCTP
4314 BIO_dgram_sctp_notification_cb 4682 EXIST::FUNCTION:SCTP
4315 SM3_Final_dword 4683 EXIST::FUNCTION:SM3
4316 SM3_Final 4684 EXIST::FUNCTION:SM3
4317 SM3 4685 EXIST::FUNCTION:SM3
4318 SM3_Init 4686 EXIST::FUNCTION:SM3
4319 EVP_sm3 4687 EXIST::FUNCTION:SM3
4320 SM3_Final_byte 4688 EXIST::FUNCTION:SM3
4321 SM3_Update 4689 EXIST::FUNCTION:SM3
4322
```

5、【手动修改】mk1mf.pl

添加内容如：

```
D:\Program Files\openssl-1.0.1i\util\mk1mf.pl (8 hits)
Line 102: no-md2 no-md4 no-sm3 no-md5 no-sha no-mdc2 - Skip this digest
Line 255: $cflags.=" -DOPENSSL_NO_SM3" if $no_sm3;
Line 255: $cflags.=" -DOPENSSL_NO_SM3" if $no_sm3;
Line 825: @a=grep(!/^sm3|(^sm3$)/,@a) if $no_sm3;
Line 825: @a=grep(!/^sm3|(^sm3$)/,@a) if $no_sm3;
Line 825: @a=grep(!/^sm3|(^sm3$)/,@a) if $no_sm3;
Line 1102: "no-sm3" => \ $no_sm3,
Line 1102: "no-sm3" => \ $no_sm3,
```

6、【手动修改】sp-diff.pl

添加内容如：

```
D:\Program Files\openssl-1.0.1i\util\sp-diff.pl (1 hit)
Line 14: foreach $a ("md2","md4","sm3","md5","sha","sha1","rc4","des_cfb","des_cbc","des_ede3",
```

7、涉及到的文件\crypto\下面2个：

```
D:\Program Files\openssl-1.0.1i\crypto\crypto-lib.com (6 hits)
```

```
D:\Program Files\openssl-1.0.1i\crypto\install-crypto.com (3 hits)
```

8、【手动修改】crypto-lib.com

添加内容如：

```
D:\Program Files\openssl-1.0.1i\crypto\crypto-lib.com (6 hits)
Line 114: "MD2,MD4,SM3,MD5,SHA,MDC2,HMAC,RIPEMD,"+ET_WHIRLPOOL+", "+ -
Line 214: $ LIB_SM3 = "sm3_dgst,sm3_one"
Line 214: $ LIB_SM3 = "sm3_dgst,sm3_one"
Line 214: $ LIB_SM3 = "sm3_dgst,sm3_one"
Line 290: $ LIB_EVP_2 = "m_null,m_md2,m_md4,m_sm3,m_md5,m_sha,m_sha1,m_wp," + -
Line 366: $ COMPILEWITH_CC5 = "md2_dgst,md4_dgst,sm3_dgst,md5_dgst,mdc2dgst," + -
```

9、【手动修改】install-crypto.com

添加内容如：

```
D:\Program Files\openssl-1.0.1i\crypto\install-crypto.com (3 hits)
Line 78: md2, md4, sm3, md5, sha, md2, hmac, ripemd, whirlpool, -
Line 91: $ exheader_sm3 := sm3.h
Line 91: $ exheader_sm3 := sm3.h
```

10、涉及到的文件根目录下面5个：

```
D:\Program Files\openssl-1.0.1i\INSTALL.VMS (1 hit)
D:\Program Files\openssl-1.0.1i\Makefile (1 hit)
D:\Program Files\openssl-1.0.1i\Makefile.bak (1 hit)
D:\Program Files\openssl-1.0.1i\Makefile.org (1 hit)
D:\Program Files\openssl-1.0.1i\makevms.com (4 hits)
```

11、【手动修改】

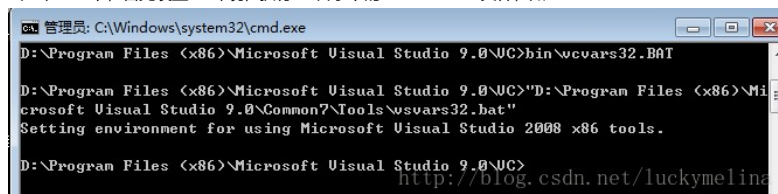
添加内容如图：Makefile、Makefile.bak、Makefile.org、makevms.com、INSTALL.VMS

```
D:\Program Files\openssl-1.0.1i\Makefile (1 hit)
Line 147: md4 sm3 md5 sha md2 hmac ripemd whirlpool \
D:\Program Files\openssl-1.0.1i\Makefile.bak (1 hit)
Line 147: md4 sm3 md5 sha md2 hmac ripemd whirlpool \
D:\Program Files\openssl-1.0.1i\Makefile.org (1 hit)
Line 145: md2 md4 sm3 md5 sha md2 hmac ripemd whirlpool \
D:\Program Files\openssl-1.0.1i\makevms.com (4 hits)
Line 284: SM3,-
Line 710: MD2, MD4, SM3, MD5, SHA, MD2, HMAC, RIPEMD, WHIRLPOOL, -
Line 723: $ EXHEADER_SM3 := sm3.h
Line 723: $ EXHEADER_SM3 := sm3.h
D:\Program Files\openssl-1.0.1i\INSTALL.VMS (1 hit)
Line 135: do this with are: RSA, DSA, DH, MD2, MD4, SM3, MD5, RIPEMD,
```

到这里，源码需要添加和修改的就结束了，下面将介绍编译和测试。

六、编译

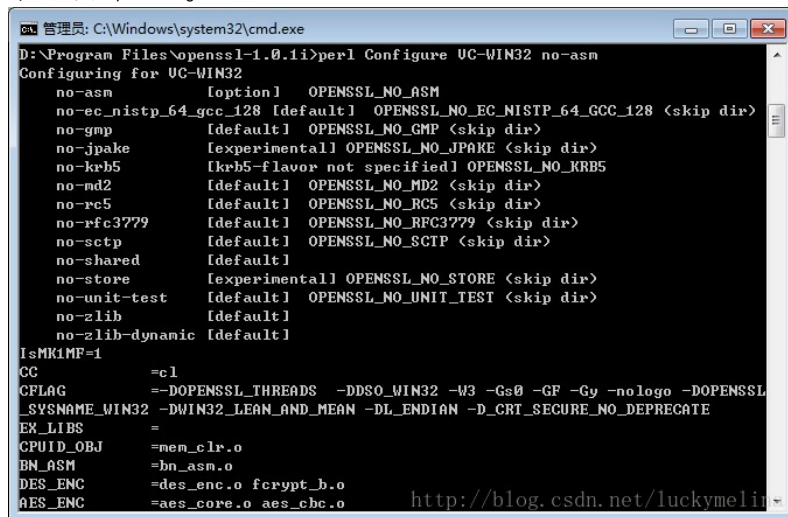
1、在cmd下，首先设置VC环境，执行VC目录下的vcvars32.BAT文件，如：



```
管理员: C:\Windows\system32\cmd.exe
D:\Program Files (x86)\Microsoft Visual Studio 9.0\VC>bin\vcvars32.BAT
D:\Program Files (x86)\Microsoft Visual Studio 9.0\VC>"D:\Program Files (x86)\Microsoft Visual Studio 9.0\Common7\Tools\vsvars32.bat"
Setting environment for using Microsoft Visual Studio 2008 x86 tools.
D:\Program Files (x86)\Microsoft Visual Studio 9.0\VC>
```

2、cmd下进入openssl源码所在的目录，依次执行命令【编32位，如果64位可能缺ml64.exe】：

1) 执行命令：perl Configure VC-WIN32 no-asm



```
管理员: C:\Windows\system32\cmd.exe
D:\Program Files\openssl-1.0.1i>perl Configure VC-WIN32 no-asm
Configuring for VC-WIN32
no-asm [option] OPENSSL_NO_ASM
no-ec_nistp_64_gcc_128 [default] OPENSSL_NO_EC_NISTP_64_GCC_128 <skip dir>
no-gmp [default] OPENSSL_NO_GMP <skip dir>
no-jpake [experimental] OPENSSL_NO_JPAKE <skip dir>
no-krb5 [krb5-flavor not specified] OPENSSL_NO_KRB5
no-md2 [default] OPENSSL_NO_MD2 <skip dir>
no-rc5 [default] OPENSSL_NO_RC5 <skip dir>
no-rc3779 [default] OPENSSL_NO_RC3779 <skip dir>
no-sctp [default] OPENSSL_NO_SCTP <skip dir>
no-shared [default]
no-store [experimental] OPENSSL_NO_STORE <skip dir>
no-unit-test [default] OPENSSL_NO_UNIT_TEST <skip dir>
no-zlib [default]
no-zlib-dynamic [default]
IsMIMP=1
CC = cl
CFLAG = -DOPENSSL_THREADS -DWIN32 -W3 -Gs0 -GF -Gy -nologo -DOPENSSL_SYSNAME_WIN32 -DWIN32_LEAN_AND_MEAN -D_ENDIAN -D_CRT_SECURE_NO_DEPRECATED
EX_LIBS =
CPUID_OBJ = mem_clr.o
BN_ASM = bn_asm.o
DES_ENC = des_enc.o fcrypt_b.o
AES_ENC = aes_core.o aes_chc.o
```

2) 执行命令：msldo_ms

```

C:\Windows\system32\cmd.exe
cl /Fotmp32dll\ts.obj -DMONOLITH -Iinc32 -Itmp32dll /MD /Ox /O2 /Ob2 -DOPENSSL_THREADS -DDSO_WIN32 -W3 -Gs0 -GF -Gy -nologo -DOPENSSL_SYSNAME_WIN32 -DWIN32_LEAN_AND_MEAN -DL_ENDIAN -D_CRT_SECURE_NO_DEPRECATED -DOPENSSL_USE_APPLINK -I. -DOPENSSL_NO_RC5 -DOPENSSL_NO_MD2 -DOPENSSL_NO_KRB5 -DOPENSSL_NO_JPAKE -DOPENSSL_NO_STATIC_ENGINE /Zi /Fdtmp32dll/app -c .\apps\ts.c
ts.c

cl /Fotmp32dll\srp.obj -DMONOLITH -Iinc32 -Itmp32dll /MD /Ox /O2 /Ob2 -DOPENSSL_THREADS -DDSO_WIN32 -W3 -Gs0 -GF -Gy -nologo -DOPENSSL_SYSNAME_WIN32 -DWIN32_LEAN_AND_MEAN -DL_ENDIAN -D_CRT_SECURE_NO_DEPRECATED -DOPENSSL_USE_APPLINK -I. -DOPENSSL_NO_RC5 -DOPENSSL_NO_MD2 -DOPENSSL_NO_KRB5 -DOPENSSL_NO_JPAKE -DOPENSSL_NO_STATIC_ENGINE /Zi /Fdtmp32dll/app -c .\apps\srp.c
srp.c

cl /Fotmp32dll\openssl.obj -DMONOLITH -Iinc32 -Itmp32dll /MD /Ox /O2 /Ob2 -DOPENSSL_THREADS -DDSO_WIN32 -W3 -Gs0 -GF -Gy -nologo -DOPENSSL_SYSNAME_WIN32 -DWIN32_LEAN_AND_MEAN -DL_ENDIAN -D_CRT_SECURE_NO_DEPRECATED -DOPENSSL_USE_APPLINK -I. -DOPENSSL_NO_RC5 -DOPENSSL_NO_MD2 -DOPENSSL_NO_KRB5 -DOPENSSL_NO_JPAKE -DOPENSSL_NO_STATIC_ENGINE /Zi /Fdtmp32dll/app -c .\apps\openssl.c
openssl.c

link /nologo /subsystem:console /opt:ref /debug /out:out32dll\openssl.exe
E:\Users\xiaohai\AppData\Local\Temp\vn6D9F.tmp
正在创建库 tmp32dll\junk.lib 和对 tmp32dll\junk.exp
IF EXIST out32dll\openssl.exe.manifest mt -nologo -manifest out32dll\openssl.exe.manifest -outputresource:out32dll\openssl.exe;1

D:\Program Files\openssl-1.0.1.i> http://blog.csdn.net/luckymeli

```

```
5) 执行命令: nmake -f ms\ntdll.mak install
```

```
管理员: C:\Windows\system32\cmd.exe
D:\Program Files\openssl-1.0.1i>nmake -f ms\ntdll.mak install

Microsoft (R) 程序维护实用工具 9.00.21022.08 版
版权所有 (C) Microsoft Corporation。保留所有权利。

Building OpenSSL
perl util/mkdir-p.pl "usr\local\ssl"
created directory `usr'
created directory `usr/local'
created directory `usr/local/ssl'
perl util/mkdir-p.pl "usr\local\ssl\bin"
created directory `usr/local/ssl/bin'
perl util/mkdir-p.pl "usr\local\ssl\include"
created directory `usr/local/ssl/include'
perl util/mkdir-p.pl "usr\local\ssl\include\openssl"
created directory `usr/local/ssl/include/openssl'
perl util/mkdir-p.pl "usr\local\ssl\lib"
created directory `usr/local/ssl/lib'
perl util/copy.pl "inc32\openssl\*.ch" "usr\local\ssl\include\openssl"
"
Copying: inc32/openssl/aes.h to /usr/local/ssl/include/openssl/aes.h
Copying: inc32/openssl/applink.c to /usr/local/ssl/include/openssl/applink.c
Copying: inc32/openssl/asn1.h to /usr/local/ssl/include/openssl/asn1.h
Copying: inc32/openssl/asn1_mac.h to /usr/local/ssl/include/openssl/asn1_mac.h
Copying: inc32/openssl/asn1t.h to /usr/local/ssl/include/openssl/asn1t.h
```

1、API编程测试，在vs建工程，将openssl目录下生成的out32dll文件夹下的libeay32.dll和libeay32.lib到工程。



2、设置工程的附加库和附加包含目录：

右键工程->属性-> C/C++ -> 常规->附加包含目录



右键工程->属性-> 链接器 -> 常规->附加库目录



3、 sm3test.c源码如：

sm3test.c

```
[cpp]
01. #include <stdio.h>
02. #include <openssl/evp.h>
03. #include <openssl/sm3.h>
04. #pragma comment(lib, "libeay32.lib")
05. static size_t hash[8] = {0};
06.
07. void out_hex(size_t *list1)
08. {
09.     size_t i = 0;
10.     for (i = 0; i < 8; i++)
11.     {
12.         printf("%08x ", list1[i]);
13.     }
14.     printf("\r\n");
15. }
16.
17. main(int argc, char *argv[])
18. {
19.     EVP_MD_CTX mdctx;
20.     const EVP_MD *md;
21.     char mess1[] = "abc";
22.     char mess2[] = "abc";
23.     unsigned char md_value[EVP_MAX_MD_SIZE];
24.     int md_len, i;
25.     //使EVP_Digest系列函数支持所有有效的信息摘要算法
26.     OpenSSL_add_all_digests();
27.
28.     argv[1] = "sm3";
29.
30.     if(!argv[1]) {
31.         printf("Usage: mdtest digestname\n");
32.         exit(1);
33.     }
34.     //根据输入的信息摘要函数的名字得到相应的EVP_MD算法结构
35.     md = EVP_get_digestbyname(argv[1]);
36.     //md = EVP_sm3();
37.
38.     if(!md) {
39.         printf("Unknown message digest %s\n", argv[1]);
40.         exit(1);
41.     }
42.     //初始化信息摘要结构mdctx，这在调用EVP_DigestInit_ex函数的时候是必须的。
43.     EVP_MD_CTX_init(&mdctx);
44.     //使用md的算法结构设置mdctx结构，impl为NULL，即使用缺省实现的算法（openssl本身提供的信息摘要算法）
45.     EVP_DigestInit_ex(&mdctx, md, NULL);
46.     //开始真正进行信息摘要运算，可以多次调用该函数，处理更多的数据，这里只调用了两次
47.     EVP_DigestUpdate(&mdctx, mess1, strlen(mess1));
48.     //EVP_DigestUpdate(&mdctx, mess2, strlen(mess2));
```



```

49. //完成信息摘要计算过程，将完成的摘要信息存储在md_value里面，长度信息存储在md_len里面
50. EVP_DigestFinal_ex(&mdctx, md_value, &md_len);
51. //使用该函数释放mdctx占用的资源，如果使用_ex系列函数，这是必须调用的。
52. EVP_MD_CTX_cleanup(&mdctx);
53.
54. printf("Digest is: ");
55. for(i = 0; i < md_len; i++) printf("%02x", md_value[i]);
56. printf("\n");
57.
58. //SM3("abc",3,hash);
59. //out_hex(hash);
60.
61. system("pause");
62. }
63.
64.
65.
66. int main1(int argc, char* argv[])
67. {
68.     SM3_CTX *c = (SM3_CTX*)malloc(sizeof(SM3_CTX));
69.     SM3_Init(c);
70.     //SM3_Final_dword(hash, c);
71.     SM3_Update(c, "abc", 3);
72.     SM3_Final(hash, c);
73.     out_hex(hash);
74.     //66c7f0f4 62eedd9 d1f2d46b dc10e4e2 4167c487 5cf2f7a2 297da02b 8f4ba8e0
75.
76.     SM3_Init(c);
77.     SM3_Update(c, "abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcd", 64);
78.     /*for (i = 0; i<16; i++){
79.         SM3_Update(c, "abcd", 4);
80.     }*/
81.     SM3_Final(hash, c);
82.     out_hex(hash);
83.     //debe9ff9 2275b8a1 38604889 c18e5a4d 6fdb70e5 387e5765 293dcb3 9c0c5732
84.
85.     //SM3("abc",3,hash);
86.     //out_hex(hash);
87.     system("pause");
88.     return 0;
89. }

```

4、 sm3test.c运行使用sm3算法对“abc”进行摘要后的结果如图：



和国密标准的附录示例一致：

杂凑值
66c7f0f4 62eedd9 d1f2d46b dc10e4e2 4167c487 5cf2f7a2 297da02b 8f4ba8e0

八、总结

本例以摘要算法为例，具体介绍了一下通过直接添加源码的方式怎样在openssl里面添加新算法，而非Engine方式。当然如果认为engine方式更方便，则请忽略本文。本文只是在描述怎么做，但是没有讲为什么。因为例如openssl本身的封装机制是需要解读源码才能更好地理解上述文中为什么需要做那些步骤，因为这些步骤已经够复杂了，所以解读源码将另起一篇文章。本文仅供依葫芦画瓢，要理解原理请自行查阅别的资料或者自己解读源码，或者等我的下一篇文章。

任何疑问或者指正请联系我，谢谢！

小伙伴们加油！GOOD LUCK！

想对作者说点什么？

我来说两句