



Programming with OpenSSL

Hyoungshick Kim

Department of Software

College of Software

Sungkyunkwan University

SSL and OpenSSL

- SSL is the security protocol behind secure HTTP (HTTPS). It can secure any protocol that works over TCP
- OpenSSL is a full-featured implementation of SSL, including TLS (transport layer security)

Overview of OpenSSL

- OpenSSL is an open-source implementation of cryptographic functions. It includes **executable commands with cryptographic functions**, and a **library of APIs** with which programmers can use to develop cryptographic applications.
- Its design follows the object-oriented principle. Each cryptographic algorithm is built upon a context, an object that holds the necessary parameters.

Installation of OpenSSL

- How to install:
 - Get the latest stable source from <http://www.openssl.org> and run (as root)

```
# ./config  
# make  
# make test  
# make install
```

- How to compile (as normal users):

```
% gcc -Wall test_openssl.c -o test_openssl -lcrypto
```

Symmetric key crypto

- OpenSSL supports various symmetric encryption algorithms (e.g., AES, DES, RC4)
([https://wiki.openssl.org/index.php/Manual:Enc\(1\)](https://wiki.openssl.org/index.php/Manual:Enc(1)))
- Suppose that we use AES with CBC
- We need to pad the last block if the plaintext length is not a multiple of block size (i.e., 128 bits / 16 bytes)

AES with CBC in OpenSSL (1)

- Find the actual length of the encrypted message. Make it a multiple of 128 bits.

```
#include <openssl/aes.h>

...

unsigned int message_len = strlen((char*)input_string) + 1; // including '\0'
unsigned encrypt_len = (message_len % AES_BLOCK_SIZE == 0) ? message_len :
(message_len / AES_BLOCK_SIZE + 1) * AES_BLOCK_SIZE;
```

- Define the key (assuming 128 bits)

```
unsigned char key[16];
AES_KEY aes;
int ret = AES_set_encrypt_key(key, 128, &aes); // ret < 0 → error
```

- Define the IV:

```
unsigned char iv[AES_BLOCK_SIZE];
memset(iv, 0, AES_BLOCK_SIZE);
```

AES with CBC in OpenSSL (2)

- Encrypt the plaintext (note that iv will be updated)

```
AES_cbc_encrypt(input_string, encrypt_string, encrypt_len, &aes, iv,  
AES_ENCRYPT);
```

- Decrypt the ciphertext. The decryption side must synchronize on the iv and key.

```
AES_set_decrypt_key(key, 128, &aes);  
memset(iv, 0, AES_BLOCK_SIZE);  
...  
AES_cbc_encrypt(encrypt_string, decrypt_string, encrypt_len, &aes, iv,  
AES_DECRYPT);
```

Big number

- Public key cryptography handles very large integers. Standard C data types are not enough
- BIGNUM package has virtually no limits on upper bounds of numbers
- Header file: `#include <openssl/bn.h>`
- The binary representation of BIGNUM is in **big-endian** format.
- Reference:
[https://wiki.openssl.org/index.php/Manual:Bn\(3\)](https://wiki.openssl.org/index.php/Manual:Bn(3))

Initializing and Destroying BIGNUMs

- A BIGNUM is an object (or context) that contains dynamically allocated memory

```
BIGNUM *a;  
a = BN_new();  
if(!a) /* Handle error */  
...  
BN_free(a);
```

BIGNUM to Binary

- Deep copy is required when you copy a BIGNUM object

```
BIGNUM a, b, *c;
```

```
/* wrong way */
```

```
a = b;
```

```
*c = b;
```

```
/* right away */
```

```
BN_copy(&a, &b); /* copies b to a */
```

```
c = BN_dup(&b); /* creates c and initialize it to same value as b */
```

BIGNUM to Binary

- Sometimes we need to convert a BIGNUM into binary representation for:
 - Store it in a file
 - Send it via a socket connection
- Similarly, we can convert a BIGNUM into a decimal or hexadecimal representation

Converting BIGNUM to Binary

- How to convert?

```
BIGNUM* num;

/* converting from BIGNUM to binary */
len = BN_num_bytes(num)
buf = (unsigned char*)calloc(len, sizeof(unsigned char));
len = BN_bn2bin(num, buf);

/* converting from binary to BIGNUM */
BN_bin2bn(buf, len, num);
num = BN_bin2bn(buf, len, NULL);
```

Math operations for BIGNUM

- Many operations are included in BIGNUM package, including modular arithmetic.
- Example: BN_mod_exp()

```
BIGNUM *r, *g, *x, *p;  
BN_CTX* ctx = BN_CTX_new(); /* store temporary results */  
/* .. call BN_new() on r, g, x, p */  
/* r = g^x mod p */  
BN_mod_exp(r, g, x, p, ctx);  
/* when done, free r, g, x, p, and ctx */
```

- BN_CTX stores temporary values of operations so as to improve the performance.

Generating prime BIGNUM

- BN_generate_prime() generates pseudorandom prime numbers
- Instead of factoring, check if a number is prime after a number of primality tests

```
BIGNUM* prime = BN_generate_prime(NULL, bits, safe, NULL,  
NULL, callback, NULL);
```

- **Safe** primes – p is prime and $(p-1)/2$ is also prime.

Questions?

