Security

OpenSSL lab session

Marco Tiloca

marco.tiloca@iet.unipi.it

Security

OpenSSL lab session

- Symmetric key encryption (2 hours)
- Hash functions (2 hours)
- Asymmetric encryption (2 hours)
- Digital signature (4 hours)
- Diffie-Hellman protocol (4 hours)

Security

OpenSSL lab session

Introduction to OpenSSL

OpenSSL library

- OpenSSL is an open source cryptographic library, providing cryptographic APIs at different layers.

- EVP APIs provide an high-level interface for cryptographic functions.
    - Symmetric key cryptography
    - Hash functions
    - …

Using OpenSSL

- GNU/Linux platforms:
  - Install **libssl0.9.8** and **libssl-dev** packages
  - Compile your code by linking the library
    - gcc –o prog prog.c **-l lcrypto**

- Microsoft Windows platforms:
  - Install **Cygwin**
    - Specify the installation of packages related to *openssl* e *gcc* in /bin
  - Compile your code by linking the library
    - gcc –o prog prog.c **–lcrypto**
  - The executable file has extension *.exe*

Managing the EVP context

- EVP **context** is a data structure which implements the symmetric key cypher.

- Context allocation

  ```
  EVP_CIPHER_CTX* ctx;
  ctx = malloc(sizeof(EVP_CIPHER_CTX));
  ```

- Context preparation

  ```
  EVP_CIPHER_CTX_init(ctx);
  EVP_CIPHER_CTX_set_key_length(ctx,key_size);
  bsize = EVP_CIPHER_CTX_block_size(ctx);
  ```

- Context deallocation

  ```
  EVP_CIPHER_CTX_cleanup(ctx);
  free(ctx);
  ```

Symmetric key encryption

- Context allocation

- Context preparation

- Encryption

  ```
  EVP_EncryptInit(ctx,EVP_des_ecb(),NULL,NULL);
  EVP_EncryptInit(ctx,NULL,key,NULL);
  EVP_Encrypt_Update(ctx,out,&loutU,in,lin);
  EVP_Encrypt_Final(ctx,&out[pos],&loutF);
  ```

- Context deallocation

Symmetric key encryption

- Encrypting operations

  ```
  EVP_Encrypt_Update(ctx,out,&loutU,in,lin);
  EVP_Encrypt_Final(ctx,&out[pos],&loutF);
  ```

  - ctx: context
  - out: encrypted text buffer
  - (loutU + loutF): encrypted text size in bytes
  - in: plain text buffer
  - lin: plain text size in bytes
  - pos == loutU

- The size of *out* must be (*lin* + *bsize*)

- The size of the encrypted text is *loutU* + *loutF*

Symmetric key decryption

- Context allocation

- Context preparation

- Decryption

```
EVP_DecryptInit(ctx,EVP_des_ecb(),NULL,NULL);
EVP_DecryptInit(ctx,NULL,key,NULL);
EVP_DecryptUpdate(ctx,out,&loutU,in,lin);
EVP_DecryptFinal(ctx,&out[pos],&loutF);
```

- Context deallocation

Symmetric key decryption

- Decryption operations

  ```
  EVP_Decrypt_Update(ctx,out,&loutU,in,lin);
  EVP_Decrypt_Final(ctx,&out[pos],&loutF);
  ```

  - ctx: context
  - out: decrypted text buffer
  - (loutU + loutF): plain text size in bytes
  - in: encrypted text buffer
  - lin: encrypted text size in bytes
  - pos == loutU

- The size of *out* must be (*lin* + *bsize*)

- The size of the decrypted text is *loutU* + *loutF*

Example

```
#include <stdio.h>
#include <string.h>
#include <openssl/evp.h>
#include <openssl/rand.h>

void printbyte(char b) {

  char c;

  c = b;
  c = c >> 4;
  c = c & 15;
  printf("%X", c);
  c = b;
  c = c & 15;
  printf("%X:", c);

}
```

Example

```
void select_random_key(char *k, int b) {

  int i;
  RAND_bytes(k, b);

  printf(“Key: ");
  for (i = 0; i < b - 1; i++)
    printbyte(k[i]);

  printbyte(k[b-1]);
  printf("\n\n");

}
```

Example

```
int main() {
  char *msg = " I can resist everything except temptation";
  char *plaintext, *ciphertext;
  char k[EVP_MAX_KEY_LENGTH]; /* encryption key */
  int nc; /* amount of bytes [de]crypted at each step */
  int nctot; /* total amount of encrypted bytes */
  int i; /* index */
  int pt_len; /* plain text size */
  int ct_len; /* encrypted text size */
  int ct_ptr; /* first available entry in the buffer */
  int msg_len; /* message length */

  /* Context allocation */
  EVP_CIPHER_CTX *ctx = (EVP_CIPHER_CTX
*)malloc(sizeof(EVP_CIPHER_CTX));

  /* Context initialization */
  EVP_CIPHER_CTX_init(ctx);

  /* Context setup for encryption */
  EVP_EncryptInit(ctx, EVP_des_ecb(), NULL, NULL);
```

Example

```
/* Output of the original message */
printf("\nOriginal message:\n%s\n\n", msg);

/* Output of the encryption key size */
printf("Key size %d\n", EVP_CIPHER_key_length(EVP_des_ecb()));
/* Output of the block size */
printf("Block size %d\n\n", EVP_CIPHER_CTX_block_size(ctx));

/* Key generation */
select_random_key(k, EVP_MAX_KEY_LENGTH);

/* Encryption key set up */
EVP_EncryptInit(ctx, NULL, k, NULL);

/* Buffer allocation for the encrypted text */
msg_len =  strlen(msg)+1;
ct_len = msg_len + EVP_CIPHER_CTX_block_size(ctx);

printf(“Message size %d\n",  msg_len);
printf(“Ciphertext size %d\n", ct_len);
ciphertext = (char *)malloc(ct_len);
```

Example

```
/* Encryption */
nc = 0;
nctot = 0;
ct_ptr = 0;

EVP_EncryptUpdate(ctx, ciphertext, &nc, msg, msg_len);
ct_ptr += nc;
nctot += nc;

EVP_EncryptFinal(ctx, &ciphertext[ct_ptr], &nc);
nctot += nc;

printf("\nCiphertext:\n");
for (i = 0; i < ct_len; i++)
   printbyte(ciphertext[i]);
printf("\n\n");
```

Example

```
/* Buffer allocation for decryption */
pt_len = ct_len + EVP_CIPHER_CTX_block_size(ctx);
plaintext = (char *)malloc(pt_len);

/* Decryption context initialization */
EVP_CIPHER_CTX_init(ctx);
EVP_DecryptInit(ctx, EVP_des_ecb(), k, NULL);

/* Decryption */

nc = 0;
nctot = 0;
ct_ptr = 0;

EVP_DecryptUpdate(ctx, &plaintext[ct_ptr], &nc, ciphertext, ct_len);
ct_ptr += nc;
nctot += nc;

EVP_DecryptFinal(ctx, &plaintext[ct_ptr], &nc);
nctot += nc;
```

Example

```
for (i = 0; i < pt_len - 1; i++)
        printf("%c:", plaintext[i]);
printf("%c\n", plaintext[pt_len-1]);

printf("\n%s\n\n", plaintext);

 EVP_CIPHER_CTX_cleanup(ctx);
 free(ctx);
 free(ciphertext);
 free(plaintext);

return 0;

}
```