Sicurezza dei Sistemi Informatici

Esercitazioni OpenSSL

Marco Tiloca

marco.tiloca@iet.unipi.it

Sicurezza dei Sistemi Informatici

Esercitazioni OpenSSL

- Cifratura chiave simmetrica (2 ore)
- Funzioni hash (2 ore)
- Cifratura asimmetrica (2 ore)
- Firma digitale (4 ore)
- Il protocollo Diffie-Hellman (4 ore)

Sicurezza dei Sistemi Informatici

Esercitazioni OpenSSL

Introduzione OpenSSL

Libreria OpenSSL

- OpenSSL è una libreria crittografica open source, e fornisce API crittografiche di basso e alto livello.

- Le API EVP forniscono un'interfaccia di alto livello per funzioni crittografiche.
  - Crittografia a chiave simmetrica
  - Funzioni hash
  - …

Uso libreria OpenSSL

- Sistemi GNU/Linux:
  - Installare i pacchetti **libssl0.9.8** e **libssl-dev**
  - Compilare linkando manualmente la libreria
    - gcc –o prog prog.c **-l ssl**

- Sistemi Microsoft Windows:
  - Installare l'applicativo **Cygwin**
    - Specificare di installare in /bin i pacchetti relativi a *openssl* e *gcc*
  - Compilare linkando manualmente la libreria
    - gcc –o prog prog.c **–lcrypto**
  - Il file eseguibile prodotto è un *.exe*

Gestione del contesto EVP

- Il **contesto** EVP è una struttura dati che implementa, il cifrario a chiave simmetrica.

- Allocazione contesto

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

- Preparazione contesto

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

- Deallocazione contesto

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

Cifratura chiave simmetrica

- Allocazione contesto
- Preparazione contesto
- Cifratura

```
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);
```

- Deallocazione contesto

Cifratura chiave simmetrica

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

  - ctx: contesto
  - out: buffer per contenere il testo cifrato
  - (loutU + loutF): numero byte cifrati prodotti
  - in: buffer contenente il testo in chiaro
  - lin: dimensione del testo in chiaro in byte
  - pos == loutU

- La dimensione di out deve essere (lin + bsize)

- La dimensione del testo cifrato è loutU + loutF

Decifratura chiave simmetrica

- Allocazione contesto
- Preparazione contesto
- Decifratura

```
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);
```

- Deallocazione contesto

Decifratura chiave simmetrica

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

  - ctx: contesto
  - out: buffer per contenere il testo in chiaro
  - (loutU + loutF): numero byte decifrati prodotti
  - in: buffer contenente il testo cifrato
  - lin: dimensione del testo cifrato in byte
  - pos == loutU

- La dimensione di out deve essere (lin + bsize)

- La dimensione del testo in chiaro è loutU + loutF

Esempio

```
#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);

}
```

Esempio

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

  int i;
  RAND_bytes(k, b);

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

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

}
```

Esempio

```
int main() {
  char *msg = "Resisto a tutto fuorche' alle tentazioni";
  char *plaintext, *ciphertext;
  char k[EVP_MAX_KEY_LENGTH]; /* chiave di cifratura */
  int nc; /* numero di byte [de]crittati ad ogni passo*/
  int nctot; /* numero totale di byte crittati */
  int i; /* indice */
  int pt_len; /* lunghezza del testo in chiaro */
  int ct_len; /* lunghezza del testo cifrato */
  int ct_ptr; /* puntatore alla prima posizione libera del buffer */
  int msg_len; /* lunghezza del messaggio */

  /* Allocazione del contesto */
  EVP_CIPHER_CTX *ctx = (EVP_CIPHER_CTX
*)malloc(sizeof(EVP_CIPHER_CTX));

  /* Inizializzazione del contesto */
  EVP_CIPHER_CTX_init(ctx);

  /* Setup del contesto per la cifratura */
  EVP_EncryptInit(ctx, EVP_des_ecb(), NULL, NULL);
```

Esempio

```
/* Output su std output del messaggio originale */
printf("\nMessaggio originale:\n%s\n\n", msg);

/* Output su std output della dimensione della chiave */
printf("Key size %d\n", EVP_CIPHER_key_length(EVP_des_ecb()));
/* Output su std output della dimensione del blocco */
printf("Block size %d\n\n", EVP_CIPHER_CTX_block_size(ctx));

/* Selezione random della chiave */
select_random_key(k, EVP_MAX_KEY_LENGTH);

/* Impostazione della chiave */
EVP_EncryptInit(ctx, NULL, k, NULL);

/* Allocazione del buffer per ciphertext */
msg_len =  strlen(msg)+1;
ct_len = msg_len + EVP_CIPHER_CTX_block_size(ctx);

printf("Dimensione del msg %d\n",  msg_len);
printf("Dimensione del ciphertext %d\n", ct_len);
ciphertext = (char *)malloc(ct_len);
```

Esempio

```
/* Cifratura */
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");
```

Esempio

```
/* Allocazione del buffer per la decifratura */
pt_len = ct_len + EVP_CIPHER_CTX_block_size(ctx);
plaintext = (char *)malloc(pt_len);

/* Inizializzazione contesto decifratura */
EVP_CIPHER_CTX_init(ctx);
EVP_DecryptInit(ctx, EVP_des_ecb(), k, NULL);

/* Decifratura */

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;
```

Esempio

```
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;

}
```