Sicurezza dei Sistemi Informatici

Esercitazioni OpenSSL

Esercitazione 1

Cifratura chiave simmetrica

Obiettivo

- Scambio file su applicazione client-server.

- Client e server condividono una chiave simmetrica.

- Il client vuole inviare al server un file, garantendo la confidenzialità.

Funzioni comuni a client e server

- 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);
```

Operazioni lato client

- Allocazione contesto

- Preparazione contesto

```
EVP_CIPHER_CTX_init(ctx);
EVP_EncryptInit(ctx,EVP_des_ecb(),NULL,NULL);
EVP_EncryptInit(ctx,NULL,key,NULL);
EVP_CIPHER_CTX_set_key_length(ctx,key_size);
bsize = EVP_CIPHER_CTX_block_size(ctx);
```

- Cifratura

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

- Deallocazione contesto

Cifratura lato client

- 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

Operazioni lato server

- Allocazione contesto

- Preparazione contesto

```
EVP_CIPHER_CTX_init(ctx);
EVP_DecryptInit(ctx,EVP_des_ecb(),NULL,NULL);
EVP_DecryptInit(ctx,NULL,key,NULL);
EVP_CIPHER_CTX_set_key_length(ctx,key_size);
bsize = EVP_CIPHER_CTX_block_size(ctx);
```

- Decifratura

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

- Deallocazione contesto

Decifratura lato server

- 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

Generazione della chiave

```
void select_random_key
(char *k, int b) {
  int i;
  RAND_bytes(k, b);
  for (i = 0; i < b - 1; i++)
    printbyte(k[i]);
  printbyte(k[b-1]);
  printf("\n");
}
```

```
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);
}
```

- La dimensione della chiave DES ECB viene ritornata dalla funzione di libreria EVP_CIPHER_key_length(EVP_des_ecb())

- Includere gli header *openssl/evp.h* e *openssl/rand.h*

Esercizio

- Si consideri il cifrario DES in modalità ECB.

- Generare una chiave DES_ECB. Trasferire la chiave sulle cartelle di client e server.

- Definire una funzione *retrieve_key()* che, data la dimensione, recuperi la chiave da file e la ritorni al chiamante.

- Il client legge un file, lo cifra, e lo invia al server.

- Il server riceve il contenuto cifrato, lo decifra, e lo salva su file locale.

Estensioni

- Cifrare e decifrare *n* byte alla volta.

- Utilizzare il cifrario DES in modalità CBC.
  - Serve anche un vettore di inizializzazione *iv*
  - Generare *iv* con RAND_pseudo_bytes()
  - Il client deve inviare anche *iv* al server
  - Cambia l'inizializzazione del contesto EVP

- L'utente specifica al client anche il cifrario da usare, come argomento da riga di comando.
  - Il client deve specificare al server il cifrario che intende usare.
  - Il client attende conferma da parte del server prima di procedere.

Qualche consiglio…

- Dichiarare variabili e costanti **in cima** al blocco.
- Chiudere i file aperti e deallocare la memoria.
- Non sottovalutare le warning del compilatore.
- Durante il debugging:
  - Stampare usando la funzione *sprintf(stderr, …)*, se occorre in esadecimale.
  - Controllare i valori ritornati dalle funzioni
- Durante il testing:
  - Usare vari tipi di file (.pdf, .txt, …)
  - Utilizzare il comando *diff.*
  - Utilizzare *valgrind* con l'opzione *--leak-check=full* e verificare l'assenza di errori.