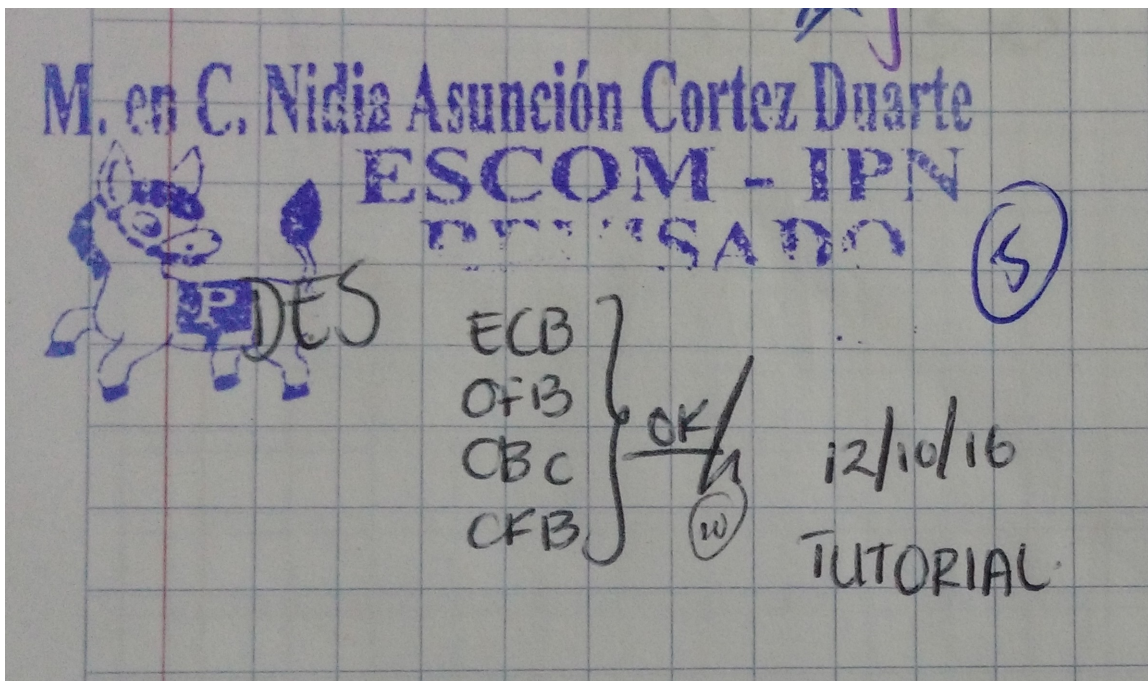


Instituto Politécnico Nacional
Escuela Superior de Cómputo
Cryptography
Tutorial Practice 5
Installation and use of the OpenSSL DES library in Linux

By:
Moreno Zárate Víctor Gibrán
Professor:
M. en C. Nidia Asunción Cortez Duarte

November 26, 2016



Contents

| | | |
|----------|-------------------------------|----------|
| 1 | Installation | 3 |
| 2 | Use in a C/C++ program | 4 |
| 2.1 | Header | 4 |
| 2.2 | Data types | 4 |
| 2.3 | Functions | 4 |
| 2.4 | Example using ECB | 4 |
| 2.5 | Example using CFB | 5 |
| 3 | Compiling | 5 |

Abstract

In this tutorial, we will review how to install and use the DES functions, included in the OpenSSL library, in a C/C++ program running in a Linux OS.

1 Installation

In order to install the OpenSSL DES library, we need to run the following commands in a terminal. (Do not copy the # symbol)

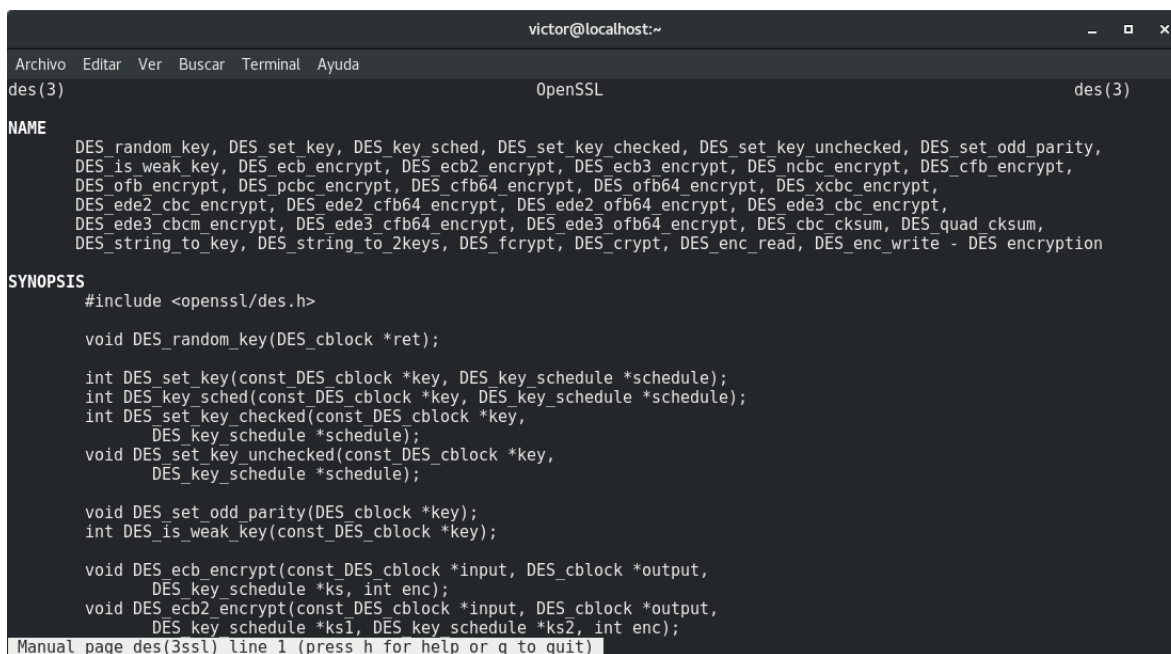
- In a Debian based distro:
`# sudo apt-get update`

`# sudo apt-get install libssl-dev`
- In a Fedora based distro:
`# sudo dnf install openssl-devel`

After the installation ends, we can use this library in a C/C++ program, in the next section we will review how to do this.

We can read the documentation of all the functions and how they work by typing in a terminal the following command:

```
# man des
```



```
victor@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
des(3) OpenSSL des(3)  
NAME  
DES_random_key, DES_set_key, DES_key_sched, DES_set_key_checked, DES_set_key_unchecked, DES_set_odd_parity,  
DES_is_weak_key, DES_ecb_encrypt, DES_ecb2_encrypt, DES_ecb3_encrypt, DES_ncbc_encrypt, DES_cfb_encrypt,  
DES_ofb_encrypt, DES_pcbc_encrypt, DES_cfb64_encrypt, DES_ofb64_encrypt, DES_xcbc_encrypt,  
DES_edec2_cbc_encrypt, DES_edec2_cfb64_encrypt, DES_edec2_ofb64_encrypt, DES_edec3_cbc_encrypt,  
DES_edec3_cbc_encrypt, DES_edec3_cfb64_encrypt, DES_edec3_ofb64_encrypt, DES_cbc_cksum, DES_quad_cksum,  
DES_string_to_key, DES_string_to_2keys, DES_fcrypt, DES_crypt, DES_enc_read, DES_enc_write - DES encryption  
SYNOPSIS  
#include <openssl/des.h>  
  
void DES_random_key(DES_cblock *ret);  
  
int DES_set_key(const DES_cblock *key, DES_key_schedule *schedule);  
int DES_key_sched(const DES_cblock *key, DES_key_schedule *schedule);  
int DES_set_key_checked(const DES_cblock *key,  
    DES_key_schedule *schedule);  
void DES_set_key_unchecked(const DES_cblock *key,  
    DES_key_schedule *schedule);  
  
void DES_set_odd_parity(DES_cblock *key);  
int DES_is_weak_key(const DES_cblock *key);  
  
void DES_ecb_encrypt(const DES_cblock *input, DES_cblock *output,  
    DES_key_schedule *ks, int enc);  
void DES_ecb2_encrypt(const DES_cblock *input, DES_cblock *output,  
    DES_key_schedule *ks1, DES_key_schedule *ks2, int enc);  
Manual page des(3ssl) line 1 (press h for help or q to quit)
```

Figure 1: Example of the command man des.

2 Use in a C/C++ program

2.1 Header

The functions of this library are defined in “openssl/des.h”, we need to include the following line in our programs to tell the compiler where they are defined:

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

2.2 Data types

This header defines two data types, the first one is called `DES_cblock`. We are going to use this data type to store the key, plain data and cipher data. The `DES_cblock` is internally defined as an array of unsigned char with fixed length of 8 bytes.

The other defined data type is called `DES_key_schedule`, we need to use this data type to process our key before passing it to the functions, in the next section we will review how.

Also, the constants `DES_ENCRYPT` and `DES_DECRYPT` are defined. We use them to tell the functions what we are going to do, encrypt or decrypt.

2.3 Functions

There are several functions defined in the SSL library, the ones of our interest are:

- `void DES_ecb_encrypt(const_DES_cblock *input, DES_cblock *output, DES_key_schedule *ks, int enc);`
- `void DES_cbc_encrypt(const unsigned char *input, unsigned char *output, long length, DES_key_schedule *schedule, DES_cblock *ivec, int enc);`
- `void DES_cfb_encrypt(const unsigned char *in, unsigned char *out, int numbits, long length, DES_key_schedule *schedule, DES_cblock *ivec, int enc);`
- `void DES_ofb_encrypt(const unsigned char *in, unsigned char *out, int numbits, long length, DES_key_schedule *schedule, DES_cblock *ivec);`

2.4 Example using ECB

In the next example, we encrypt a block (`plain_data`) using ECB mode, with a key read from the keyboard. We need to call the function `setkey` to schedule our key and make it usable to the `ecb` function. The output data is stored in `cipher_data`.

```
char plaintext[8];
int mode = DES_ENCRYPT; // If we need to decrypt, use DES_DECRYPT
// SSL Data types
DES_cblock key;
DES_cblock plain_data; // Input Data Block
DES_cblock cipher_data; // Resultant Data Block
DES_key_schedule schedule; // Scheduled Key
// Cleaning Blocks
```

```

bzero(key , sizeof(DES_cblock));
bzero(plain_data , sizeof(DES_cblock));
bzero(cipher_data , sizeof(DES_cblock));
bzero(&schedule , sizeof(DES_key_schedule));
//Asking key
printf("ECB. _Insert_key:_\n");
scanf("%s" ,key);
fflush(stdin);
//Generate Key
DES_set_key(&key,&schedule);
//Encrypt Data
DES_ecb_encrypt((C_Block *)plain_data ,(C_Block *)cipher_data,&schedule ,mode);
//Work done

```

2.5 Example using CFB

In this mode, we pass to the function a block of data whose length is stored in the variable `length` (it does not matter the block size, the function encrypts/decrypts the whole block). The initial vector is hardcoded in the variable `ivec`, and the key is read from the stdin. The other modes are analogue to this one.

```

int numbits=8;
int mode = DES_ENCRYPT; //If we need to decrypt, use DES_DECRYPT
unsigned char llave [9];
long length;
//SSL Data Types
unsigned char *input;
unsigned char *output;
DES_key_schedule schedule;
DES_cblock ivec={0xE1, 0xE2, 0xE3, 0xD4, 0xD5, 0xC6, 0xC7, 0xA8};
DES_cblock key;
//Cleaning Blocks
bzero(key , sizeof(DES_cblock));
bzero(&schedule , sizeof(DES_key_schedule));
//Allocating Buffers
input=calloc(length , sizeof(unsigned char));
output=calloc(length , sizeof(unsigned char));
//Asking Key
printf("%s. _Insert_key:_\n",type);
scanf("%s" ,llave);
DES_string_to_key((const char*)llave , &key); //
fflush(stdin);
//Generate Key
DES_set_key(&key,&schedule);
//Encrypt
DES_cfb_encrypt(input , output ,numbits,length,&schedule ,(C_Block *)ivec ,mode);
//Work done

```

3 Compiling

To compile, we use the following command (assuming that our program is called `des.c`) :

```
# gcc des.c -lcrypto -o DES
```

