MANUAL INTRODUCTORIO PARA PROGRAMAR CON OPENSSL

ACTUALIZACIÓN 2024

LIC. ANA MARÍA ARIAS ROIG

1. Passwords

- **Funciones para derivar key e iv.**

Para generar clave e iv a partir de una password, se puede utilizar la función **`EVP_BytesToKey()`**.

```
int EVP_BytesToKey (const EVP_CIPHER *type, EVP_MD *md, const unsigned char *salt, const
unsigned char *data, int datal, int count, unsigned char *key, unsigned char *iv)
```

`EVP_BytesToKey()` devuelve la longitud de la clave correspondiente al algoritmo de cifrado especificado en el argumento **`type`**.

Para obtener la clave y el vector de inicialización, a partir del password indicado en el parámetro **`data`**, de longitud **`datal`**, la función realiza un algoritmo que involucra una o más iteraciones (según el valor del argumento **`count`**) de transformación en las que un algoritmo de hash especificado en el argumento **`md`** va obteniendo una clave del tamaño deseado. El parámetro **`salt`** es opcional (puede ser NULL).

Este algoritmo, como ya se vio en el Manual, no es estándar, y se debería usar con preferencia el algoritmo PBKDF2 (descripto en RFC 2898)

Para esto, lo más simple que ofrece OpenSSL es usar la función **`PKCS5_PBKDF2_HMAC()`**.

```
int PKCS5_PBKDF2_HMAC(const char *pass, int passlen,
                      const unsigned char *salt, int saltlen, int iter,
                      const EVP_MD *digest,
                      int keylen, unsigned char *out);
```

`PKCS5_PBKDF2_HMAC()` devuelve en out una cadena con la clave y el iv concatenados.

Los parámetros **`pass`** y **`passlen`** permiten pasar los datos de la password.

Los parámetros **`salt`** y **`saltlen`** permiten pasar los datos de la salt.

En el parámetro **`digest`** se indica el algoritmo de hash a usar.

En **`keylen`** hay que pasar el tamaño de la clave más el iv . Es requisito que el usuario previamente calcule las longitudes de key y de iv para el algoritmo requerido, usando **`EVP_CIPHER_key_length()`** y **`EVP_CIPHER_iv_length()`**.

- **Ejemplos:**

Ejemplo 1: Encripción del texto: "*Hoy encripto.*", en AES 128 modo CBC con password "margarita", sin **`salt`**, mostrando el cifrado en codificación base 64.

Equivale a:[1]

```
openssl enc -aes-128-cbc -e -in hoy.txt -out hoyE.txt -k margarita -a -p -nosalt -pbkdf2
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/evp.h>
#include "base64.h"

void mostrarKey(unsigned char key[],unsigned char keylen);
int
main()
{
    unsigned char *pwd = "margarita";
    unsigned char *key;
    unsigned char *iv;
```

[1] El default de iter es 10000, y el default de hash sha256.

```
    unsigned char keylen = EVP_CIPHER_key_length(EVP_des_ecb());
    unsigned char ivlen = EVP_CIPHER_iv_length(EVP_des_ecb());
    printf("Clave AES: %d bytes.\n", keylen);
    printf("IV AES: %d bytes.\n", ivlen);
    unsigned char out[keylen+ivlen];

    PKCS5_PBKDF2_HMAC(pwd,strlen(pwd),NULL,0,10000, EVP_sha256(), keylen+ivlen, out);

    key = malloc(keylen);
    strncpy(key,out,keylen);
    printf("\nKey derivada:");
    mostrarKey(key,keylen);

    if(ivlen!=0){
        iv = malloc(ivlen);
        strncpy(iv,out+keylen,ivlen);
        printf("\nIV derivada:");
        mostrarKey(iv,ivlen);
    }
    return EXIT_SUCCESS;
}
void mostrarKey(unsigned char key[],unsigned char keylen)
{
    int i;
    for (i = 0; i < keylen; i++)
    {
        printf("%0x", key[i]);
    }
}
```

2. Algoritmo DES obsoleto.

A partir de openssl 3.0, no se proveen los algoritmos para DES simple. Si bien las funciones están, sólo generan un cifrado NULO.

Lo que sí está disponible es triple des, con 3 claves, en los modos ecb, cbc, cfb y ofb:

- **EVP_des_ede3()**
- **EVP_des_ede3_cbc()**
- **EVP_des_ede3_cfb()**
- **EVP_des_ede3_cfb1()**
- **EVP_des_ede3_cfb8()**
- **EVP_des_ede3_cfb64()**
- **EVP_des_ede3_ecb()**
- **EVP_des_ede3_ofb()**