1) Respecto de cómo guardar el tamaño del bloque.

Suponiendo que el tamaño del bloque a ocultar es 17,118.
Ese número ocupa 4 bytes.
Puede guardarse en Little Endian:

DE	42	00	00

O puede guardarse en Big Endian:

00	00	42	DE

Los archivos que les entregaremos tendrán el tamaño guardado en forma Big Endian.

2) Cambios respecto de versión Openssl 1.0 yOpenssl 1.1

El Manual de Openssl que está publicado, fue hecho para la versión 1.0.
Hay que considerar dos cambios importantes respecto de la versión 1.1.:

a. Respecto de cómo manejar contexto de cifrado (ctx)

Debe ser de tipo puntero, se inicializa **con EVP_CIPHER_CTX_new()** y se libera con **EVP_CIPHER_CTX_free(ctx)**
A continuación, se reescribe el ejemplo 2 de la página 12 del manual.

```
int
main()
{
    unsigned char *nomArch = "aes-128-cbc-evp.txt";
    unsigned char *in = "Inteligencia, dame el nombre exacto de las
cosas... Que mi palabra sea la cosa misma, creada por mi alma
nuevamente.";
    unsigned char out[MAX_ENCR_LENGTH];
    int inl, outl, templ;
    unsigned char *k = "0123456789012345";
    unsigned char iv[] = "5432109876543210";

    EVP_CIPHER_CTX *ctx;

    /*inicializar contexto*/
    ctx = EVP_CIPHER_CTX_new();
    /*establecer parámetros de encripción en el contexto*/
    EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(),NULL, k,iv);
    inl = strlen(in);
    EVP_EncryptUpdate(ctx, out, &outl, in, inl);
    /*encripta 112 bytes*/
    printf("Encripta primero %d bytes\n", outl);
    /*encripta la parte final, lo que queda del bloque + padding*/
    EVP_EncryptFinal(ctx, out+outl, &templ);
    printf("Finalmente se encriptan %d bytes\n", templ);
    saveEncryptedData(out, outl+templ, nomArch);
    /*limpiar estructura de contexto*/
    EVP_CIPHER_CTX_free(ctx);
    return EXIT_SUCCESS;
}
```

b. Respecto de cómo derivar clave e iv a partir de password.

El ejemplo del manual usa EVP_md5() porque en linea de comandos, openssl en las versiones anteriores usaba md5.
Actualmente usa sha256, así que usaremos **EVP_sha256()** para poder controlar lo hecho por código respecto de la salida que obtendríamos por linea de comando.
Pueden probar con el siguiente ejemplo (passgen.c)

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

```c
#define SUCCESS 0
#define FAILURE !SUCCESS

void mostrarKey(unsigned char key[], int len);

int
main()
{
    const EVP_CIPHER *cipher;
    const EVP_MD *dgst = NULL;

    const unsigned char *pwd = "margarita";

    unsigned char key[EVP_MAX_KEY_LENGTH];
    unsigned char iv[EVP_MAX_IV_LENGTH];
    const unsigned char *salt = NULL;
    int keylen, ivlen;

    cipher = EVP_aes_256_cbc();
    dgst = EVP_sha256();
    EVP_BytesToKey(cipher, dgst, salt, pwd, strlen(pwd),1, key, iv);
    keylen = EVP_CIPHER_key_length(cipher);
    ivlen = EVP_CIPHER_iv_length(cipher);
    printf("\nKey derivada:");
    mostrarKey(key, keylen);
    printf("\nIV derivada:");
    mostrarKey(iv, ivlen);

    return EXIT_SUCCESS;
}

void mostrarKey(unsigned char key[], int len)
{
    int i;
    for (i = 0; i < len; i++)
    {
        printf("%0x", key[i]);
    }
}
```

Se observa que las salidas en pampero son las mismas: (NO USAREMOS SALT)

```
[mroig@pampero enc]$ ./passgen

Key derivada:F1BE6354B41828A0B8AA12194A15CC0ADD2A52D5BDE5B582A1576FDFDFBD4
IV derivada:66ED36D905C1D4F5A1C4055E7B14283
[mroig@pampero enc]$ openssl enc -aes-256-cbc -k margarita -P -nosalt
key=F1BE6354B41828A0B8AA1201094A15CC00ADD2A52D5BDE5B582A1576FDFDFBD4
iv =66ED36D9005C1D4F5A1C4055E7B14283
```