ΕΠΛ326: Εργαστήριο 3
Κρυπτογραφία AES σε C

Εισαγωγή

Στο σημερινό εργαστήριο θα κατασκευάσουμε προγράμματα σε C, τα οποία πραγματοποιούν συμμετρική κρυπτογράφηση με AES.

Εκτέλεση Εργαστηρίου

Βήμα 1

Η υλοποίηση του AES θα χρησιμοποιηθεί μέσω της βιβλιοθήκης OpenSSL. Το πρώτο πρόγραμμα που θα κατασκευάσουμε είναι ένα κενό πρόγραμμα (π.χ., “Hello World”), το οποίο όμως χρησιμοποιεί την εν λόγω βιβλιοθήκη.

```
#include <openssl/evp.h>
#include <openssl/aes.h>
#include <openssl/err.h>

int main(int argc, char *argv[]) {
        return 1;
}
```

Χρησιμοποιήστε τον μεταγλωττιστή.

```
-bash-4.2$ gcc -Wall -lcrypto first-crypto.c -o first-crypto
-bash-4.2$ ./first-crypto
```

Βήμα 2

Στο επόμενο βήμα θα κατασκευάσουμε ένα βήμα το οποίο κρυπτογραφεί ένα μήνυμα με AES σε ECB mode. Αρχικά θα κατασκευάσουμε μια συνάρτηση, η οποία κρυπτογραφεί ένα μήνυμα που βρίσκεται στη μνήμη (σε ένα buffer), με ένα κλειδί και ένα συγκεκριμένο Initialization Vector (IV). Το αποτέλεσμα θα αποθηκεύεται επίσης στη μνήμη.

```c
#include <openssl/evp.h>
#include <openssl/aes.h>
#include <openssl/err.h>
#include <string.h>

void handleErrors(void)
     {
     unsigned long errCode;

    printf("An error occurred\n");
   while((errCode = ERR_get_error()))
        {
       char *err = ERR_error_string(errCode, NULL);
       printf("%s\n", err);
        }
   abort();
    }
.
.
int encrypt(unsigned char *plaintext, int plaintext_len,
            const unsigned char *key,
            unsigned char *ciphertext)
{
    EVP_CIPHER_CTX *ctx = NULL;
    int len = 0, ciphertext_len = 0;

    /* Create and initialise the context */
    if(!(ctx = EVP_CIPHER_CTX_new()))
        handleErrors();

    /* Initialise the encryption operation. */
    if(1 != EVP_EncryptInit_ex(ctx, EVP_aes_128_ecb(), NULL, NULL, NULL))
        handleErrors();

    /* Initialise key */
    if(1 != EVP_EncryptInit_ex(ctx, NULL, NULL, key, NULL)) handleErrors();

    /* Provide the message to be encrypted, and obtain the encrypted output.
     * EVP_EncryptUpdate can be called multiple times if necessary
     */
    if(plaintext)
    {
        if(1 != EVP_EncryptUpdate(ctx, ciphertext, &len, plaintext, plaintext_len))
            handleErrors();

        ciphertext_len = len;
    }

    if(1 != EVP_EncryptFinal_ex(ctx, ciphertext + len, &len)) handleErrors();
    ciphertext_len += len;

    /* Clean up */
    EVP_CIPHER_CTX_free(ctx);

    return ciphertext_len;
}
```

Βήμα 3

Στη συνέχεια θα κατασκευάσουμε τη συνάρτηση, η οποία αποκρυπτογραφεί ένα κρυπτογραφημένο μήνυμα.

```
64 int decrypt(unsigned char *ciphertext, int ciphertext_len,
65              const unsigned char *key,
66              unsigned char *plaintext)
67 {
68     EVP_CIPHER_CTX *ctx = NULL;
69     int len = 0, plaintext_len = 0, ret;
70
71     /* Create and initialise the context */
72     if(!(ctx = EVP_CIPHER_CTX_new())) handleErrors();
73
74     /* Initialise the decryption operation. */
75     if(!EVP_DecryptInit_ex(ctx, EVP_aes_128_ecb(), NULL, NULL, NULL))
76         handleErrors();
77
78     /* Initialise key */
79     if (!EVP_DecryptInit_ex(ctx, NULL, NULL, key, NULL)) handleErrors();
80
81     /* Provide the message to be decrypted, and obtain the plaintext output.
82      * EVP_DecryptUpdate can be called multiple times if necessary
83      */
84     if(ciphertext)
85     {
86         if(!EVP_DecryptUpdate(ctx, plaintext, &len, ciphertext, ciphertext_len))
87             handleErrors();
88
89         plaintext_len = len;
90     }
91
92     /* Finalise the decryption. A positive return value indicates success,
93      * anything else is a failure - the plaintext is not trustworthy.
94      */
95     ret = EVP_DecryptFinal_ex(ctx, plaintext + len, &len);
96
97     /* Clean up */
98     EVP_CIPHER_CTX_free(ctx);
99
100    if(ret > 0)
101    {
102        /* Success */
103        plaintext_len += len;
104        return plaintext_len;
105    }
106    else
107    {
108        /* Verify failed */
109        return -1;
110    }
111 }
```

Βήμα 4

Τέλος, θα φτιάξουμε ένα πρόγραμμα που χρησιμοποιεί αυτές τις συναρτήσεις.

```
113 int main(int arc, char *argv[])
114 {
115     OpenSSL_add_all_algorithms();
116     ERR_load_crypto_strings();
117 
118     /* A 256 bit key */
119     static const unsigned char key[] = "01234567890123456789012345678901";
120 
121     /* The message for encryption (Plaintext).  */
122     unsigned char plaintext[] = "CS475 is an awesome course about computer
security in the University of Cyprus.";
123 
124     /* Buffer to store the ciphertext. The size may be different due to padding.
*/
125     unsigned char ciphertext[128];
126 
127     /* Buffer for the decrypted text for verifying decryption.  */
128     unsigned char decryptedtext[128];
129 
130     int decryptedtext_len = 0, ciphertext_len = 0;
131 
132     /* Encryption. */
133     ciphertext_len = encrypt(plaintext, strlen((char *)plaintext), key,
ciphertext);
134     printf("Ciphertext is:\n");
135     BIO_dump_fp(stdout, (const char *)ciphertext, ciphertext_len);
136 
137     /* Decryption. */
138     decryptedtext_len = decrypt(ciphertext, ciphertext_len, key, decryptedtext);
139 
140     if(decryptedtext_len < 0)
141     {
142         /* Verify error */
143         printf("Decrypted text failed to verify\n");
144     }
145     else
146     {
147         /* Add a NULL terminator. We are expecting printable text */
148         decryptedtext[decryptedtext_len] = '\0';
149 
150         /* Show the decrypted text */
151         printf("Decrypted text is:\n");
152         printf("%s\n", decryptedtext);
153     }
154 
155     /* Remove error strings */
```

```
156     ERR_free_strings();
157
158     return 0;
159 }
```

Βήμα 5

Εφόσον έχετε πειραματιστεί με το παραπάνω πρόγραμμα, κατασκευάστε ένα αντίστοιχο που κρυπτογραφεί και αποκρυπτογραφεί μηνύματα με τη βοήθεια του AES σε CBC mode.