CCA – COMPETENCE CENTRE

HTL Anichstraße

noindent

FSST-OpenSSL

Fabio Plunser

17. Februar 2021

Inhaltsverzeichnis

Abbildungsverzeichnis

Code

1 Aufgabenstellung

```
# deciphers to "Schoene Crypto Welt" with IV=BBBBBBBBBBBBBBB and key=
    BBBBBBBBBBBBBBB aes128-cbc
cyphertext = "
    AAE365272C81078AB6116B361831D0F6A5D3C8587E946B530B7957543107F15E"
bc = binascii.unhexlify(cyphertext)
data = b'D' + bytes([len(bc)]) + binascii.unhexlify(cyphertext) + b'X'
```

Listing 1: Angabge

Der cyphertext soll entschlüsselt „Schöne Crypto Welt“ bedeuten. Um dies zu überprüfen kann https://www.openssl.org/ verwendet werden.

Schreiben Sie ein Programm das unter Verwendnung von openssl obige Aussage überprüft, verbessen Sie ihr Program in dem Sinne dass sie key/iv/plaintexte/ciphertexte als Argumente/Dateien/Usereingaben verarbeiten.

Hinweise

- Sie benötigen die openssl Biblotheksheader, unter Ubuntu 20.04 können Sie diese installieren via:

```
$ sudo apt install libssl-dev
```

Listing 2: Angabge

- em Linker muss mitgeteilt werden dass sie in Ihrem Programm Funktionen verwenden die in einer externen Bibliothek bereit liegen, verwenden sie dazu das flag -l (klein-L) und den Namen der Bibliothek OHNE das führende lib. openssl besteht aus mehreren Bibliotheken, die für AES notwendingen Funktionen befinden sich in libcrypto.

```
$ gcc my_code.c -lbibliothek -o my_executable
```

Listing 3: Angabge

Sie können sich die gelinkten Bibliotheken dann via ldd Kommando ansehen

```
$ ldd my_executable
```

Listing 4: Angabge

2 Umsetzung

2.1 OpenSSL

2.2 Programm

```
// Author: FabioPlunser //
// Date: 17.2.2020 //
// GIT-Repo: https://github.com/FabioPlunser/FSST_Lezuo
// Specific Git-location: https://github.com/FabioPlunser/FSST_Lezuo/tree/
   main/Programme/openssl/openssl-Programm //
// Compiled with make, in WSL using Ubuntu 20.0.4, as you can see in my Repo
    //

// openssl //

//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

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


int do_decrypt(char *ciphertext, int ciphertext_len, char *key, char *iv,
   char* plaintext);

int main()
{

    unsigned char key[16];
    unsigned char iv[16];

    memset(key, 'B', 16);
    memset(iv, 'B', 16);

    unsigned char ciphertext[128] = {
    0xAA, 0xE3, 0x65, 0x27, 0x2C, 0x81, 0x07, 0x8A, 0xB6, 0x11, 0x6B, 0x36,
       0x18, 0x31, 0xD0, 0xF6,
    0xA5, 0xD3, 0xC8, 0x58, 0x7E, 0x94, 0x6B, 0x53, 0x0B, 0x79, 0x57, 0x54,
       0x31, 0x07,0xF1, 0x5E
    };

    unsigned char* plaintext="Schoene Crypto Welt";
    unsigned char decryptedtext[128];
    int decryptedtext_len, ciphertext_len;
```

```
    decryptedtext_len = do_decrypt(ciphertext, sizeof(ciphertext)/4, key, iv
        , decryptedtext);
    decryptedtext[decryptedtext_len] = '\0';
    printf("EVP:\nDecrypted test is: %s\n", decryptedtext);

    AES_KEY dec_key;
    AES_set_decrypt_key(key, sizeof(key)*8, &dec_key);
        AES_cbc_encrypt(ciphertext, decryptedtext, sizeof(ciphertext)/4, &
            dec_key, iv, AES_DECRYPT);
    printf("\nAES_KEY:\nDecrypted test is: %s\n", decryptedtext);

}
```

Listing 5: main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

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


void Error_handling(void)
{
    ERR_print_errors_fp(stderr);
    abort();
}

int do_decrypt(char *ciphertext, int ciphertext_len, char *key, char *iv,
    char* plaintext)
{
    EVP_CIPHER_CTX *ctx;
    int len;
    int plaintext_len;

    if(!(ctx = EVP_CIPHER_CTX_new())) Error_handling();

    EVP_CIPHER_CTX_set_padding(ctx, 0);

    if(1 != EVP_DecryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv))
        Error_handling();

    if(1 != EVP_DecryptUpdate(ctx, plaintext, &len, ciphertext,
        ciphertext_len)) Error_handling();
    plaintext_len = len;

    if(1 != EVP_DecryptFinal_ex(ctx, plaintext+len, &len)) Error_handling();
    plaintext_len += len;
```

```
    ERR_print_errors_fp(stderr);
    EVP_CIPHER_CTX_cleanup(ctx);
    return plaintext_len;
}
```

Listing 6: EVP.c

2.3 Erklärung