CCA – COMPETENCE CENTRE

HTL Anichstraße

FSST-OpenSSL

Fabio Plunser

9. März 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 Theorie

2.1 OpenSSL

OpenSSL umfasst Implementierungen der Netzwerkprotokolle und verschiedener Verschlüsselungen sowie das Programm openssl für die Kommandozeile zum Beantragen, Erzeugen und Verwalten von Zertifikaten. Die in C geschriebene Basisbibliothek stellt allgemeine kryptographische Funktionen zum Ver- und Entschlüsseln sowie diverse weitere Werkzeuge bereit. [1]

2.2 AES

Beim **Advanced Encryption Standard**handelt sich um ein symmetrisches Verschlüsselungsverfahren, d. h. der Schlüssel zum Ver- und Entschlüsseln ist identisch. Der Rijndael-Algorithmus besitzt variable, voneinander unabhängige Block- und Schlüssellängen von 128, 160, 192, 224 oder 256 Bit. Rijndael bietet ein sehr hohes Maß an Sicherheit; erst mehr als zehn Jahre nach seiner Standardisierung wurde der erste theoretisch interessante, praktisch aber nicht relevante Angriff gefunden.

AES schränkt die Blocklänge auf 128 Bit und die Wahl der Schlüssellänge auf 128, 192 oder 256 Bit ein. Die Bezeichnungen der drei AES-Varianten AES-128, AES-192 und AES-256 beziehen sich jeweils auf die gewählte Schlüssellänge. AES ist frei verfügbar und darf ohne Lizenzgebühren eingesetzt sowie in Soft- und Hardware implementiert werden.[2]

[1] Quelle: `https://de.wikipedia.org/wiki/OpenSSL`
[2] Quelle: `https://de.wikipedia.org/wiki/Advanced_Encryption_Standard`

3 Erstes - Programm

```
// Author: FabioPlunser //
// Date: 17.2.2021 - 8.03.2021 //
// 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 //

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

//Basierend auf http://www.firmcodes.com/how-do-aes-128-bit-cbc-mode-
   encryption-c-programming-code-openssl/
//und https://wiki.openssl.org/index.php/
   EVP_Symmetric_Encryption_and_Decryption

#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 :\n 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 ("\n AES_KEY :\n nDecrypted test is: %s\n", decryptedtext);
}
```

Listing 5: alt-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: alt-EVP.c

3.0.1 Programm Output:

```
peppi@Peppi:/mnt/c/Users/fplun/GoogleDrive/Schule/2020_21/FSST/FSST_Lezuo/Programme/openssl/openssl-Programm$ ./main
EVP:
Decrypted test is: Schoene Crypto Welt

AES_KEY:
Decrypted test is: Schoene Crypto Welt
```

Abbildung 1: Programm-Output

4 Finales Programm als kleines Command line tool

```
// Author: FabioPlunser //
// Date: 17.2.2021 - 8.03.2021 //
// 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 //

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

//Basierend auf http://www.firmcodes.com/how-do-aes-128-bit-cbc-mode-
   encryption-c-programming-code-openssl/
//und https://wiki.openssl.org/index.php/
   EVP_Symmetric_Encryption_and_Decryption
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>

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


#if defined(__linux__)

int input_plaintext_from_file(char *source, unsigned char *key, unsigned
   char *iv, unsigned char *plaintext, unsigned char *ciphertext);
int input_ciphertext_from_file(char *source, unsigned char *key, unsigned
   char *iv, unsigned char *plaintext, unsigned char *ciphertext);



int main()
{
    unsigned char *key = malloc(16);
    unsigned char *iv = malloc(16);
    unsigned char ciphertext[128];
    unsigned char plaintext[128];
    unsigned char path_plaintext[1000];
    unsigned char path_ciphertext[1000];
    int choose;
    printf("\033[1m\033[32m");
    printf("\n \e[4mWillkommen zu AES-128, encrypt und decrypt\e\n\n");
    printf("\033[0m");
```

```
    printf("1 fuer encrypt\n2 fuer decrypt\n3 Encrypt eines Plaintextes aus
        einer txt datei\n4 Decrypt eines Cipthertextes aus einer txt Datei\n5
         schliessen\n\nEingabe:");
    scanf("%i", &choose);

    if (choose == 1)
    {
        printf("Key: ");
        //scanf("%[^\n]", key);
        scanf("%s", key);
        printf("IV: ");
        scanf("%s", iv);
        printf("Text der zu verschluesseln ist: ");
        scanf(" %[^\n]", plaintext); //%%[^\n]
        printf("\n");
        int plaintext_len = strlen(plaintext);
        int ciphertext_len = diyencryption(key, iv, plaintext, plaintext_len
            , ciphertext);
        printf("\033[1m\033[32m");
        printf("Verschluesselung: \n");
        printf("\033[0m");
        printf("\033[0;31m");
        for (int i = 0; i < ciphertext_len; i++)
        {
            printf("%02X", ciphertext[i]);
        }
        printf("\033[0m");
        printf("\n");
        main();
    }
    else if (choose == 2)
    {
        printf("Key: ");
        scanf("%s", key);

        printf("IV: ");
        scanf("%s", iv);

        printf("Zu entschluesselnder Text in \033[0;31m HEX \033[0m");
        scanf("%s", ciphertext);

        diydecryption(key, iv, plaintext, ciphertext);

        printf("Entschuesellung: ");
        printf("\033[0;31m");
        printf("%s", plaintext);
        printf("\033[0m");
        printf("\n\n");

        main();
    }
    else if (choose == 3)
    {
```

```
        printf("Key: ");
        scanf("%s", key);

        printf("IV: ");
        scanf("%s", iv);

        printf("Path zur \033[0;31m txt \033[0m Datei wo der Plaintext
            drinnen steht: ");
        scanf("%s", path_plaintext);
        printf("\n\n");

        int ciphertext_len = input_plaintext_from_file(path_plaintext, key,
            iv, plaintext, ciphertext);
        printf("Verschluesselung: \n");
        printf("\033[0;31m");
        for (int i = 0; i < ciphertext_len; i++)
        {
            printf("%02X", ciphertext[i]);
        }
        printf("\033[0m");
        printf("\n");
        main();
    }
    else if (choose == 4)
    {
        printf("Key: ");
        scanf("%s", key);

        printf("IV: ");
        scanf("%s", iv);

        printf("Path zur \033[0;31m txt \033[0m Datei wo der Ciphertext
            drinnen steht: ");
        scanf("%s", path_ciphertext);
        printf("\n\n");

        printf("Path Ciphertext: %s\n", path_ciphertext);

        input_ciphertext_from_file(path_ciphertext, key, iv, plaintext,
            ciphertext);
        printf("Entschuesellung: ");
        printf("\033[0;31m");
        printf("%s", plaintext);
        printf("\033[0m");
        printf("\n\n");
        main();
    }
    else if (choose == 5)
    {
        printf("\033[0;31m");
        printf("Wird geschlossen\n\n");
        printf("\033[0m");
```

```
        main();
    }
    else
    {
        printf("\033[0;31m");
        printf("Es wurde keine moegliche Aktion ausgewaehlt\n\n");
        printf("\033[0m");

        main();
    }
}
#endif

#if defined(_WIN64) || defined(_WIN32)

int main()
{
    printf("Only for Linux Use")
}
#endif
```

Listing 7: main.c

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

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


/*
Code From https://wiki.openssl.org/index.php/
   EVP_Symmetric_Encryption_and_Decryption
*/
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;
}

int do_encrypt(unsigned char *plaintext, int plaintext_len, unsigned char *
   key, unsigned char *iv, unsigned char *ciphertext)
{
    EVP_CIPHER_CTX *ctx;

    int len;

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

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

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

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

    EVP_CIPHER_CTX_free(ctx);

    return ciphertext_len;
}

/*
Encryption init
```

```
*/
int diyencryption(unsigned char *key, unsigned char *iv, unsigned char *
    plaintext, int plaintext_len, unsigned char *ciphertext)
{
    int ciphertext_len;
    ciphertext_len = do_encrypt(plaintext, strlen(plaintext), key, iv,
        ciphertext);
    return ciphertext_len;
}

/*
Dencryption init. Was very hard to figure out!!
*/
int diydecryption(unsigned char *key, unsigned char *iv, unsigned char *
    plaintext, unsigned char *ciphertext)
{
    char temp[2];                                   //temp char array
    unsigned char *hex = malloc(sizeof(unsigned char) * 128);   //array to
        write ciphertext to
    int x = 0;
    for (int i = 0; i < strlen(ciphertext); i++)    //go through string
        ciphertext and alwas set two charakters than EOF
    {
        temp[0] = ciphertext[i];
        temp[1] = ciphertext[++i];
        temp[2] = '\0';
        hex[x] = (unsigned int)strtol(temp, NULL, 16);  //add to array of
            hex values the hex value of the two values put into temp
        x++;
    }

    int ciphertext_len = strlen(hex);
    int decryptedtext_len;
    decryptedtext_len = do_decrypt(hex, ciphertext_len, key, iv, plaintext);
    plaintext[decryptedtext_len] = '\0';
}
```

Listing 8: Encryption.c

Listing 9: FileInput.c captionpos

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

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

```
void Error_handling(void);
int do_decrypt(char *ciphertext, int ciphertext_len, char *key, char *iv,
    char *plaintext);
int do_encrypt(unsigned char *plaintext, int plaintext_len, unsigned char *
    key, unsigned char *iv, unsigned char *ciphertext);
int diyencryption(unsigned char *key, unsigned char *iv, unsigned char *
    plaintext, int plaintext_len, unsigned char *ciphertext);
int diydecryption(unsigned char *key, unsigned char *iv, unsigned char *
    plaintext, unsigned char *ciphertext);
int main();


/*
Import plaintext from file and encrypt it.
*/
int input_plaintext_from_file(char *source, unsigned char *key, unsigned
    char *iv, unsigned char *plaintext, unsigned char *ciphertext)
{
    int size, file, file2, read_length, write_length, plaintext_len;
    struct stat st;

    int istxt = strcmp(source + strlen(source) - 4, ".txt");
    if (istxt == 0)
    {
        printf("Angebene Datei valide\n");
    }
    else
    {
        printf("Angegebne Datei ist keine .txt Datei\n Sie kann nicht
            verwendet werden\n");
        main();
    }

    stat(source, &st);
    size = st.st_size;  //check File size for correct buffer allocation

    char *charBuffer = malloc(size);
    file = open(source, O_RDONLY);  //open File
    read_length = read(file, charBuffer, size); //read File
    printf("Eingelesene Datei: %s\n", charBuffer);  //show Contents of file

    plaintext_len = strlen(charBuffer); //get length of content
    int ciphertext_len = diyencryption(key, iv, charBuffer, plaintext_len,
        ciphertext); //encrypt content
    close(file);
    free(charBuffer);
    printf("Home directory: %s\n", getenv("HOME"));
    return ciphertext_len;
}

/*
Import ciphertext from file and decrypt it.
```

```
*/
int input_ciphertext_from_file(char *source, unsigned char *key, unsigned
    char *iv, unsigned char *plaintext, unsigned char *ciphertext)
{
    int size, file, file2, read_length, write_length, plaintext_length;
    struct stat st;

    int istxt = strcmp(source + strlen(source)-4, ".txt");
    if (istxt == 0)
    {
        printf("\033[0;31m");
        printf("Angebene Datei valide\n");
        printf("\033[0m");
    }
    else
    {
        printf("\033[0;31m");
        printf("Angegebne Datei ist keine .txt Datei\n Sie kann nicht
            verwendet werden\n");
        printf("\033[0m");
        main();
    }
    stat(source, &st);
    size = st.st_size;

    char *charBuffer = malloc(size);
    file = open(source, O_RDONLY);
    read_length = read(file, charBuffer, size);
    printf("Eingelesene Datei: %s\n", charBuffer);
    plaintext_length = diydecryption(key, iv, plaintext, charBuffer);
    close(file);
    free(charBuffer);
}
```

4.1 Programm Output:

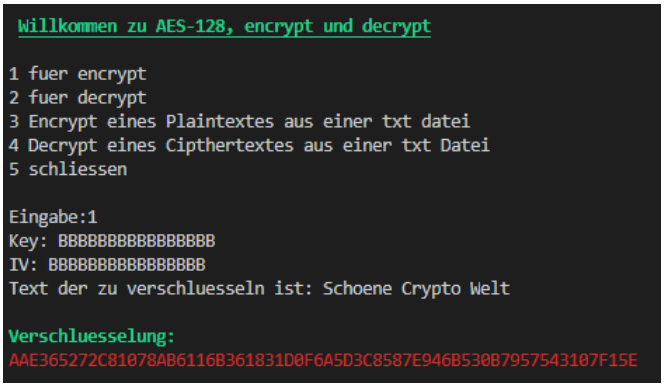

Abbildung 2: Programm Output