

# Documentatie Tema

---

Pentru implementarea temei am utilizat limbajul C/C++ alaturi de biblioteca **OpenSSL**.

Nodul A trimite catre KM modul de operare CBC sau OFB folosind un named pipe (FIFO), nodul A are drept de scriere iar KM are drept de citire. Dupa ce KM citeste modul de operare din FIFO folosind primitiva **read(int fd, void\* buff, size\_t nbytes)** trimite catre A cheia corespunzatoare K1 sau K2. Aceasta cheie este criptata inainte de a fi trimisa cu ajutorul lui K3. Pentru criptare am utilizat AES 128 pus la dispozitie de biblioteca OpenSSL.

Nodul A primeste cheia de la KM si o trimite catre nodul B impreuna cu K3 pentru a putea fi decriptata. Comunicarea intre cele doua noduri se face tot printr-un FIFO. Atat A cat si B vor decripta cheia K1/K2 folosind metoda **void decrypt(void \*in, void \*out, unsigned char\* key)** care implementeaza decriptarea pentru AES. Dupa ce nodul A primeste confirmarea de la B precum ca detine cele doua chei acesta trimite catre B continutul fisierului test.txt criptat pe blocuri de marime **BLOCK\_SIZE**(citirea si scrierea se face pe blocuri).

In acest moment B utilizeaza cheia K1/K2 decriptata pentru a decripta continutul fisierului. Procesul de decriptare se face tot pe blocuri.

Implementarea temei este impartita in:

A.c – nodul A care initiaza comunicarea

B.c – nodul B care primeste fisierul de la A

KM.c – key managerul care se ocupa cu distributia de chei

aux.c – fisier ajutator care contine implementarea functiilor pentru criptare si decriptare

Rularea programelor se face dupa executia fisierului makefile care creeaza executabile pentru A.c, B.c si KM.c.

```
andreea@andreea-VirtualBox: ~/Documents
File Edit Tabs Help
andreea@andreea-VirtualBox:~/Documents$ make all
gcc -I/usr/local/ssl/include/ -L/usr/local/ssl/lib/ -o km KM.c -lcrypto -ldl
gcc -I/usr/local/ssl/include/ -L/usr/local/ssl/lib/ -o a A.c -lcrypto -ldl
gcc -I/usr/local/ssl/include/ -L/usr/local/ssl/lib/ -o b B.c -lcrypto -ldl
B.c: In function 'receive_file':
B.c:100:9: warning: implicit declaration of function 'memcpy' [-Wimplicit-function-declaration]
      memcpy(previousBlock, cipherTextBuffer, BLOCK_SIZE);
      ^~~~~~
B.c:100:9: warning: incompatible implicit declaration of built-in function 'memcpy'
B.c:131:10: warning: incompatible implicit declaration of built-in function 'memcpy'
      memcpy(previousBlock, plainTextBuffer, BLOCK_SIZE);
      ^~~~~~
andreea@andreea-VirtualBox:~/Documents$ ls
a A.c aux.c b B.c km KM.c makefile text.txt tmp
andreea@andreea-VirtualBox:~/Documents$
```

Dupa crearea executabilelor acestea vor fi rulate astfel: ./a [modul de operare] ./b ./km

### Exemplu:

./a C

./b

./km

```
andreea@andreea-VirtualBox:~/Documents$ ./a C
1 Sent the mode to B: C
1 Sent the mode to KM: C
Read the key from KM
Sent the key to B
1 2 3 4 5 6 7 8 9 A B C D E F 0
Sent the K3 to B
Got the response from B
Read a block of 16 bytes
Written 16 bytes to B
Read a block of 15 bytes
Padded with 1 bytes
Written 16 bytes to B
Sent the file to B
andreea@andreea-VirtualBox:~/Documents$
```

```
andreea@andreea-VirtualBox:~/Documents$ ./b
```

```
1 Read the mode C
```

```
Read the key from A
```

```
Read the K3 from A
```

```
1 2 3 4 5 6 7 8 9 A B C D E F 0
```

```
Tema Securitatea Informatiei!!!
```

```
Received the file from A
```

```
andreea@andreea-VirtualBox:~/Documents$ ./km
```

```
1 Read the mode from A: C
```

```
Sent the key to A
```

```
andreea@andreea-VirtualBox:~/Documents$
```