**Mod de utilizare program tema SI**

- Sidorencu Mihnea-Stefan B6 -

1. **Elementele programului:**
   * **Functia main()**
   * **Clasa NodKM**
   * **Clasa NodA**
   * **Clasa NodB**
   * **Fisierele de trimis/primit mesaje**

**2. Functiile de criptare**

**3. Functiile de decriptare**

**4. Utilizarea aplicatiei**

1. **Elementele programului:**
   1. Functia main()

Functia main() creeaza o instanta a nodului KM care are ulterior rolul de mediator intre celelalte doua clase

* 1. Clasa NodKM

Detine cele trei chei pe 128 de biti (numite K1, K2 si K3). La fiecare initializare genereaza un vector de initializare random nou.

Cand este initializat creeaza cate o instanta pentru fiecare dintre celelalte doua noduri (A si B) si le trimite in mod automat cheia K3.

Dupa ce au fost initializare cele doua noduri, acesta apeleaza functia SystemRun() prin intermediul caruia comunica cu nodurile. In primul rand primeste de la amandoua modalitatile prin care vor acestia sa comunice (1 -> ECB, 2 -> CFB). Daca modurile de criptare dorite nu coincid, se alege unul la intamplare.

In fiecare caz nodul KM transmite celorlalte doua noduri cheia necesara incriptata ori cu ECB, ori cu CFB, prin intermediul cheii K3. In cazul CFB se trimite si vectorul de initializare.

Dupa ce cele doua nodur si-au incriptat plaintext-urile, KM-ul face schimb intre cele doua. Dupa aceasta, nodul KM si-a terminat treaba.

* 1. Clasele NodA/NodB

Abele instante primesc inca de la inceput cheia K3. Ambele citesc din fisier textul pe care vor sa il trimita si hotarasc, fiecare in parte, in mod aleatoriu, ce algoritm de criptare vor sa foloseasca.

Ambele trimit la KM felul in care vor sa cripteze si, in functie de ce hotaraste KM, ii trimit acestuia plaintextul incriptat. Cand KM are ambele texte criptate, le trimite nodului destinatar.

Acestea, fiecare in parte, decripteaza textul si il pun intr-un fisier text.

* 1. Fisierele de trimis/primit mesaje

In proiect se afla doua foldere : A resources si B resources.

Fiecare dintre ele are doua fisiere .txt - unul pentru plaintext-ul mesajului ce se vrea a fi trimis si celalalt pentru a stoca textul primit dupa ce a fost decriptat.

1. **Functiile de criptare**
2. encryptECB(string plain);

Functia aceasta primeste ca argument plaintext-ul si se foloseste ce cheia K1 pentru a il incripta.

Imparte plaintextul in blocuri de aceeasi lungime precum cheia si le incripteaza adunand valoarile ASCII ale cheii si ale blocului de plaintext ccaracter cu caracter. Pentru paddingul ultimului bloc se foloseste un numar adecvat de caractere ‘-’.

Ex:

K1 = “x!A%D\*G-KaPdSgVk”;

Plaintext = “Mesaj de la A Mesaj de la A“

Mesaj de la A Me +

x!A%D\*G-KaPdSgVk

=

k,Z,TJQ8ksW\*:-Iv

saj de la A----- +

x!A%D\*G-KaPdSgVk

=

æ(QEN5g?R'7d&:)>

Criptotext final: k,Z,TJQ8ksW\*:-Ivæ(QEN5g?R'7d&:)>

Daca suma ASCII trece de 125 sau scade sub 35, apar multe caractere problematice.

Daca suma ASCII trece de 125, o recalculam la sum-125+35.

1. encryptCFB();

Aplicarea cheii are loc la fel ca la ECB.

Pentru XOR am folosit operatorul ^ pentru caractere doua cate doua.

1. Functii de decriptare
2. decryptECB()

Functioneaza la fel ca functia de incriptare, doar ca nu adunam blocurile caracter cu caracter, ci scadem caracterele din cheie pe cele din block (bloc criptat[i] - K1[i] = bloc plaintext [j])

Daca suma ASCII trece de 125 sau scade sub 35, apar multe caractere problematice.

Daca suma ASCII scade sub 35, o recalculam la sum+125-35.

1. decryptCFB()

Pentru aplicarea cheii facem la fel ca la decryptECB() si la XOR, la fel ca la incriptarea CFB.

1. Utilizarea aplicatiei:

Pasul 1:

Se scrie in fisierul “A resources/to send.txt” mesajul pe care urmeaza sa il transmita A si in fisierul “B resources/to send.txt” mesajul pe care urmeaza sa il transmita B.

Pasul 2:

Se dechide proiectul (preferabil cu Visual Studio) si se compileaza din Source.cpp

Pasul 3:

Se verifica fisierele “B resources/received.txt” si “A resources/received.txt”. Ar trebui sa contina textul decriptat primit de fiecare dintre noduri.