Moduri operare Criptosisteme Bloc

Descriere proiect

Proiectul este facut in Java 8.

Clasa KeyGenerator care genereaza chei pentru criptare sau decriptare de catre clasa AES. Cheia este de tipul „SecretKeySpec”, o librarile din Java.

Clasa AES este clasa ce contine metodele de criptare, respectiv decriptare. Se genereaza o cheie pentru criptare sau decriptare si apoi se foloseste un obiect de tipul Cipher (librarie Java) pentru a se realiza actiunile necesare. Pentru criptare/decriptare se returneaza mereu un string ce contine textul criptat.

Key Manager-ul este clasa Server care gestioneaza comunicarea intre cei doi clienti: Clasa NodeA, Clasa NodeB. Serverul asteapta sa se conecteze cele doua noduri ale comunicarii, apoi executa metoda runServer care face posibila comunicarea intre cei doi clienti. Metoda serverului „runServer” este documentata in cod.

Nodul A. Citeste de la tastatura modul de operare (ECB/OFB). Il trimite la server, iar acesta din urma il va retine pentru sine, dar il va trimite si nodului B. Asteapta de la server cheia (criptata) pentru modul de operare ales. O decripteaza cu cheia comul K3. Asteapta raspuns de la B ca totul este in regula si ca poate incepe comunicarea (i.e. trimiterea textului in blocuri criptate). **Criptarea ECB** se realizeaza pe blocuri de 16 caractere care sunt criptate prin intermediul metodei clasei AES: encrypt. Se iau blocuri de cate 16 caractere in mod repetat. In cazul in care s-a ajuns la ultimul bloc, iar acesta are mai putin de 16 caractere, se ia cat a ramas, se cripteaza si se trimite catre server. **Criptarea OFB** se realizeaza tot pe blocuri de 16 caractere. Vectorul de initializare (iv) este global si se foloseste pentru criptarea/decriptarea in modul de operare OFB astfel: se cripteaza (sau recripteaza pentru iteratiile urmatoare) vectorul de initializare cu cheia aleasa (K1/K2). Se realizeaza aceeasi metoda de extragere a blocului de 16 caractere din textul original. Apoi se face XOR pe caracterele blocului si al vectorului criptat. Se trimite blocul rezultat catre server, care mai departe nodului B.

Nodul B. Primeste de la server modul de operare, apoi cheia pentru a decripta blocurile pe care urmeaza sa le citeasca. Trimite mesaj ca totul este in regula („OK!”), apoi incepe primirea blocurilor criptate. **Decriptarea ECB** se realizeaza

destul de simplu: pentru fiecare bloc primit, se apeleaza functia de decriptare din clasa AES si se face „append” unui StringBuilder care va fi afisat la final. **Decriptarea OFB** se efectueaza intr-un bloc infinit while in care se citeste blocul criptat. Se realizeaza din nou aceeasi criptare ca mai sus (de la criptarea OFB) intre vectorul de initializare si cheie si apoi se face XOR intre caractere. Se face append la un StringBuilder si se afiseasza rezultatul.

Observatie: comunicarea in modul de operare ECB functioneaza perfect. Cea in OFB are o problema care nu cred ca depinde implementare, ci de cum se transmit informatiile in socket (cu toate ca fac un **flush()** dupa fiecare scriere in socket). Uneori functioneaza perfect, dar se intampla sa nu primeasca nodul B in regula blocurile criptate. **DAR** exista clasa main unde puteti observa ca criptarea si decriptarea in modul de operare OFB functioneaza perfect. Nu este problema la criptare/decriptare, ci la cum se transmit datele prin socket. Este destul de anormal ce se intampla deoarece modalitatea de transmitere a blocurilor prin socketuri este identica in cele doua moduri de operare.