Tema1 Securitatea Informatiei

Constantinescu George-Gabriel 3E3

19 Octombrie 2021

1 Detalii asupra mediului de lucru

Programul este realizat in doua parti: prima parte este facuta intr-un proiect Maven - Java(partea de ECB) iar partea a2a este facuta in Python(partea de CBC + ECB).

Motivul pentru care l-am facut in doua limbaje diferite(pot spune diametral opuse) este ca atunci cand am inceput implementarea in Java a algoritmului de criptare CBC m-am blocat si nu am stiut cum sa rezolv, asa ca am trecut la python).

Clasele din proiectul Maven se vor gasi in temaSIJava/src/main/java iar cele din Python direct in folderul parinte(temaSIPython).

Ca medii de lucru, am folosit Intellij Ultimate(pentru Java) si PyCharm(pentru Python). Pentru criptarea AES, am folosit in cazul python libraria Crypto.Cipher din care am importat AES + altele: Crypto.Random, Crypto.Util.Padding, Socket, Hashlib si base64 iar in Java am folosit java.security.MessageDigest si javax.crypto.

Pe partea de comunicare, in Python am facut o comunicare intre 2 clienti diferiti(UDP) astfel incat sa isi poata transmita actiuni diferite(unul citeste, altul cripteaza, altul decripteaza s.a.m.d) . In Java este realizat in mod clasic, prin apeluri de metode specifice.(decrypt, encrypt, etc.)

2 Pre requirements

In cazul Java trebuie adaugate/instalate si indexate bibliotecile noi(in cazul asta Intellij se ocupa de acest lucru) iar in Python trebuie importate bibliotecile amintite mai sus(in acest caz se ocupa PyCharm de acest lucru).

3 Modul de rezolvare

3.1 Java

Cum am explicat mai sus, in Java am realizat o arhitectura clasica a problemei(orientata obiect). Pentru proiect am realizat 5 clase: MainClass, KeyManager, AESComponent si cele doua noduri NodeA si NodeB.

- In MainClass se regasesc toate functiile necesare pentru a rula o instanta de la linia de comanda.
- Cele doua clase specifice nodurilor si KeyManager sunt responsabile pentru criptarea/decriptarea textului pe care il dam ca input de la tastatura. Pentru criptarea mesajului este responsabila clasa AESComponent prin sistemul AES folosind Key3.
- Functia initializeVector din MainClass are scopul de a genera un vector de litere in mod randomizat.
- Alte elemente importante in cod ar fi metoda care permite comunicarea intre noduri (associateWithNodeA/NodeB), functiile care permit criptarea/decriptarea mesajului(decryptKeyWithAES, decryptAlgorithmECB).

3.2 Python

In Python am realizat o arhitectura concurenta, cu doi clienti(noduri) care comunica intre ele. Pentru proiect am realizat 3 fisiere .py: keymanager.py, nodeA.py si nodeB.py.

- In toate cele 3 fisiere se regaseste clasa AESController, care lafel ca in Java realizeaza criptarea/decriptarea textului pe care il primim ca input de la tastatura.
- Metodele de transmitere a mesajului de la un nod la altul, acest fapt realizat prin intermediul comunicarii intre socket-uri. Metodele receive asteapta textul de la celalalt nod, utilizand recvfrom din biblioteca socket.
- encryptionCBC, pentru algoritmul de criptare CBC care preia cheie, o decripteaza, ulterior cripteaza mesajul si il trimite la celalalt nod.
- encryptionECB, pentru algoritmul de criptare ECB care primeste parole de la KeyManager si ulterior o decripteaza. Acest algoritm se realizeaza prin intermediul metodei encryptionblockbyblockECB. Aceasta imparte mesajul in mai multe blocuri de text pe care le cripteaza si ulterior le trimite catre celalalt nod.

4 Testare si rezultate

4.1 Java/Python

In ambele cazuri, in urma algoritmului de criptare (CBC sau ECB) textul este cu succes criptat si ulterior decriptat, astfel incat sa se ajunge la forma initiala. In cazul Java, in urma executarii programului se va afisa textul criptat, cheia cu care s-a criptat textul iar ulterior se va realiza decriptarea astfel incat sa se observe ca ambele secvente de text coincid.

In cazul Python, se va afisa textul criptat prin intermediul comunicarii celor doua noduri. Cele doua noduri vor crea o conexiune prin intermediul careia mesajul va fi transmis la ambele noduri dar si la KeyManager, simuland un server de tip UDP care trimite cheia de criptare inapoi la ambele noduri. Ambele noduri pot decripta cheia, mesajul se transmite inapoi intre noduri si se reia acest proces.

Numarul de caractere nu a reprezentat o problema in acest caz, criptarea si decriptarea realizandu-se pe string-uri cu o dimensiuni mai mare.

Observatie: Problema de implementare cea mai dificila pentru mine a fost la comunicarea intre noduri si transmiterea mesajului criptat in Python + decriptarea aceastuia. Fiind elemente noi de programare pentru mine(partea de crypto) se pot observa mici ocolisuri prin partea de cod dar consider ca in viitor, prin asimilarea cunostintelor, va fi un rezultat mult mai bun.

5 Bibliografie

1) https://howtodoinjava.com/java/java-security/java-aes-encryption-example/
2) https://howtodoinjava.com/java/java-security/java-aes-encryption-example/
3) https://stackoverflow.com/questions/19623267/importerror-no-module-named-crypto-cipher

4) https://www.baeldung.com/java-aes-encryption-decryption

5) https://stackoverflow.com/questions/2626835/is-there-functionality-to-generate-a-random-character-in-java

6) https://profs.info.uaic.ro/ nica.anca/is.html

7) https://newbedev.com/how-to-fix-invalid-aes-key-length

8) https://docs.python.org/3.5/library/hashlib.html

9) https://stackoverflow.com/questions/2778840/socket-error-errno-10013-an-attempt-was-made-to-access-a-socket-in-a-way-forb