UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”FAKULTETI I SHKENCAVE MATEMATIKO-NATYROREDEPARTAMENTI I MATEMATIKËSPROGRAMI: SHKENCA KOMPJUTERIKE

 **PUNIM SEMINARIK**Lënda: Siguria e të dhënave  
TEMA: Simplified Advanced Encryption Standard me CFB

**Punuar nga: Albina Grajqevci, Diellza Bajraktari, Dalina Elshani**

**Prill 2023**

Përmbatja

[1. Hyrja 2](#_Toc133411399)

[2. S-AES 3](#_Toc133411400)

[3. Kodi në JAVA 4](#_Toc133411401)

[4. Modi CFB 9](#_Toc133411402)

[5. Implementimi në JAVA i modit CFB 11](#_Toc133411403)

[6. Përfundimi 12](#_Toc133411404)

# Hyrja

AES do të thotë Advanced Encryption Standard, i cili është një algoritëm i përdorur gjerësisht i enkriptimit që mund të përdoret për të siguruar të dhëna. Është një algoritëm me çelës simetrik, që do të thotë se i njëjti çelës përdoret si për enkriptim ashtu edhe për deshifrim.

Algoritmi AES funksionon në blloqe të dhënash, me një madhësi blloku prej 128 bit. Çelësi i përdorur në AES mund të jetë 128, 192 ose 256 bit në gjatësi.

Procesi i enkriptimit AES përfshin disa raunde operacionesh zëvendësimi dhe ndërrimi që transformojnë tekstin e thjeshtë në tekst shifror duke përdorur çelësin. Procesi i deshifrimit përfshin operacionet e kundërta për të kthyer tekstin e shifruar përsëri në tekst të thjeshtë.

Në terma të thjeshtuar, algoritmi AES mund të konsiderohet si një formulë komplekse matematikore që përdor një çelës sekret për të marrur të dhënat në atë mënyrë që ato të bëhen të palexueshme pa çelës. Kjo e bën atë një mënyrë efektive për të siguruar informacione të ndjeshme, të tilla si fjalëkalimet, numrat e kartave të kreditit dhe lloje të tjera të dhënash që duhet të mbahen private.

# S-AES

Simplified AES (S-AES) është një version i thjeshtuar i algoritmit të kriptimit AES që përdoret shpesh për qëllime edukative dhe për të mësuar bazat e kriptografisë simetrike të çelësave.

S-AES përdor një madhësi më të vogël blloku prej 16 bitësh (krahasuar me madhësinë e bllokut 128-bit të AES) dhe një madhësi më të vogël çelësi prej 16 bitësh (krahasuar me madhësitë kryesore të AES prej 128, 192 ose 256 bitësh). Algoritmi S-AES ka dy pjesë kryesore: funksionet e enkriptimit dhe deshifrimit.

Përmbledhja e procesit të kriptimit S-AES:

* Zgjerimi i çelësit: Çelësi i enkriptimit 16-bit zgjerohet në një grup çelësash të rrumbullakët, të cilët përdoren në çdo raund të procesit të kriptimit.
* Raundi fillestar: Blloku i tekstit të thjeshtë 8-bitësh XORohet me tastin e parë të rrumbullakët.
* Rounds: Procesi i enkriptimit përbëhet nga dy raunde, secila prej të cilave kryen disa operacione në bllokun e të dhënave. Në çdo raund:

1. Zëvendësimi: Çdo bajt i bllokut zëvendësohet me një vlerë përkatëse nga një tabelë fikse kërkimore e quajtur S-box.
2. Permutacioni: Bajtet në bllok riorganizohen sipas një rregulli fiks permutacioni.
3. Përzierja: Bajtet në bllok kombinohen duke përdorur një operacion të shumëzimit të matricës.
4. Shtesa e çelësit: Blloku është XOR me një çelës të rrumbullakët që rrjedh nga çelësi i enkriptimit.

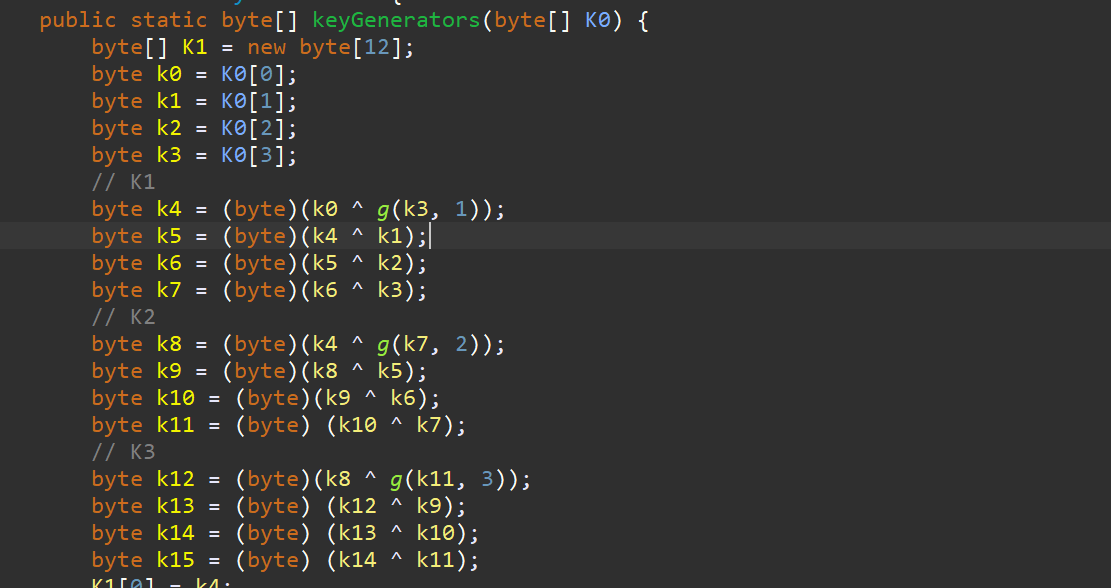
* Dalja: Blloku i fundit i koduar është teksti i koduar.

Procesi i deshifrimit S-AES është në thelb e kundërta e procesit të kriptimit, me çelësat e përdorur në rend të kundërt.

S-AES është një version i thjeshtuar i AES që mund të përdoret për të mësuar bazat e kriptografisë simetrike të çelësave, por nuk rekomandohet për përdorim në aplikacionet e botës reale pasi ka një madhësi blloku shumë të vogël dhe madhësi të çelësit, duke e bërë atë të cenueshëm ndaj sulmeve të ndryshme

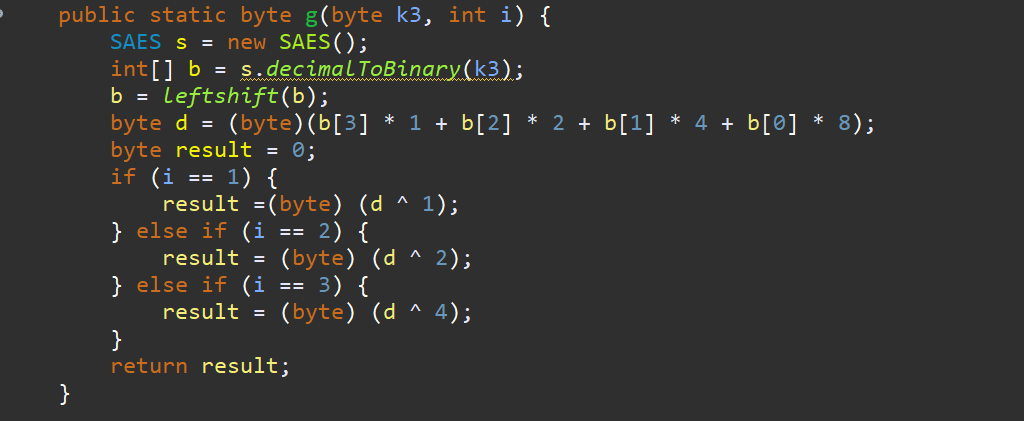
# Kodi në JAVA

Fillimisht do të implementojmë se si gjenerohen çelësat te algoritmi i thjeshtuar AES. Si hyrje kemi një çelës K0 të cilin e ruajmë si varg të gjatësisë 4, ku secili element i ka nga 4 bit dhe reprezenton një numër heksadecimal.



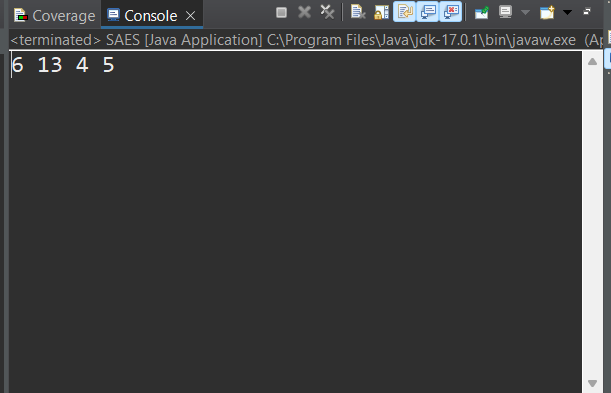
Pra kështu kemi gjeneruar çelësat K1, K2 dhe K3.

Në bazë të skemës së dhënë poashtu kemi implementuar funksionin g të cilin e kemi implementuar duke marrur njërin nga këto 4 bitët e K0 pastaj kemi bërë një leftShift dhe më pas e kemi bërë XOR me 1, 2 apo 4 varësisht në cilin raund kemi qenë.

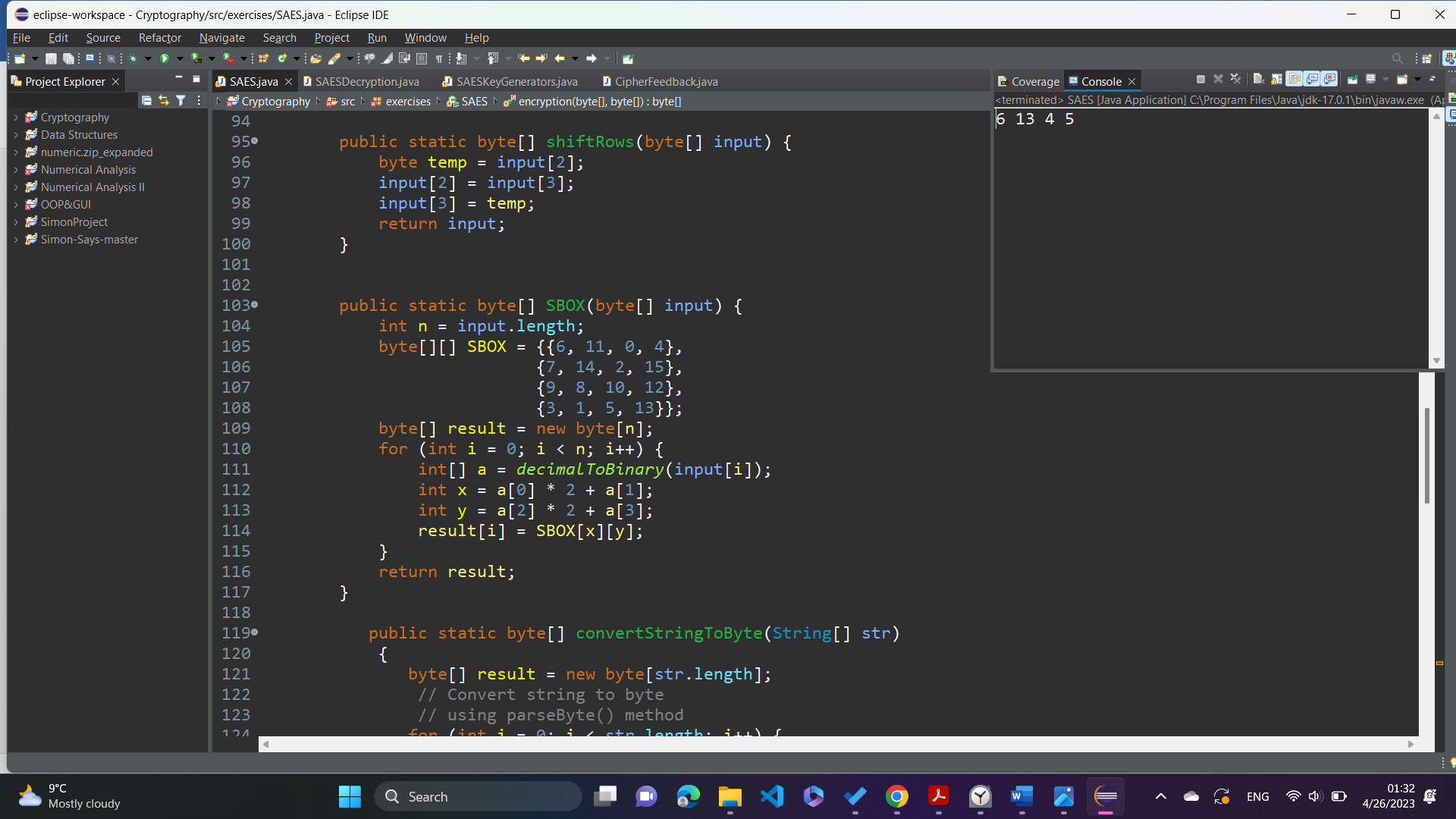


Pastaj kemi filluar t’a implementojmë se si bëhet enkriptimi i mesazhit te simplified AES, metoda e enkriptimit si hyrje do t’a ketë plaintext-in apo mesazhin që dëshirojmë t’a enkriptojmë si dhe çelësin fillestar.

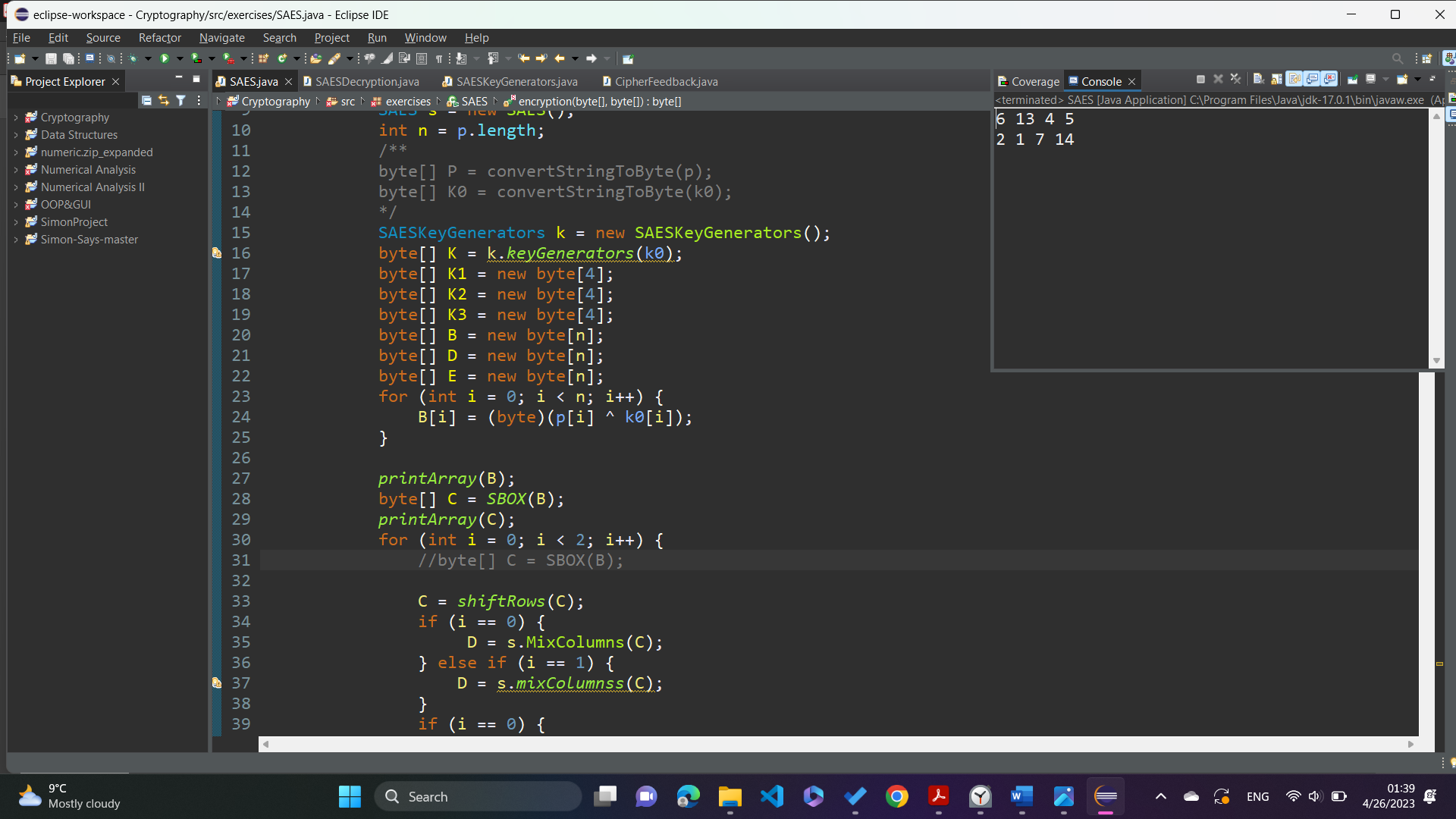
Nëse marrim mesazhin “A, 2, E, 4” si dhe çelësin K0 = “C, F, A, 1” atëherë fillimisht do t’i bëjmë XOR këto dy vargje dhe si rezultat do të kemi një varg tjetër që është (në këtë rast 13 kur e kthejmë në hexadecimal është D) :



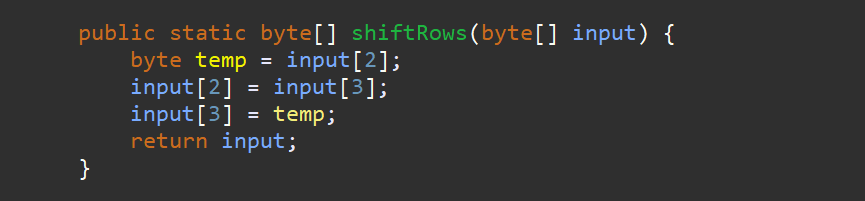
Pastaj faza e enkriptimit ka 3 raunde duke përfshirë SBOX, ShiftRows, MixColumns, AddRoundKey ku përjashtimisht në raundin e tretë nuk bëhet MixColumns.

Pastaj kemi implementuar SBOX në këtë mënyrë ku secilin numër në vargun e mësipërm e kemi kthyer në binar ku dy shifrat e para na kanë japur numrin e rreshtit dhe dy shifrat e fundit numrin e kolonës për ti zëvëndësuar më pas me matricën SBOX.

Ku do të na shfaqet rezultati :

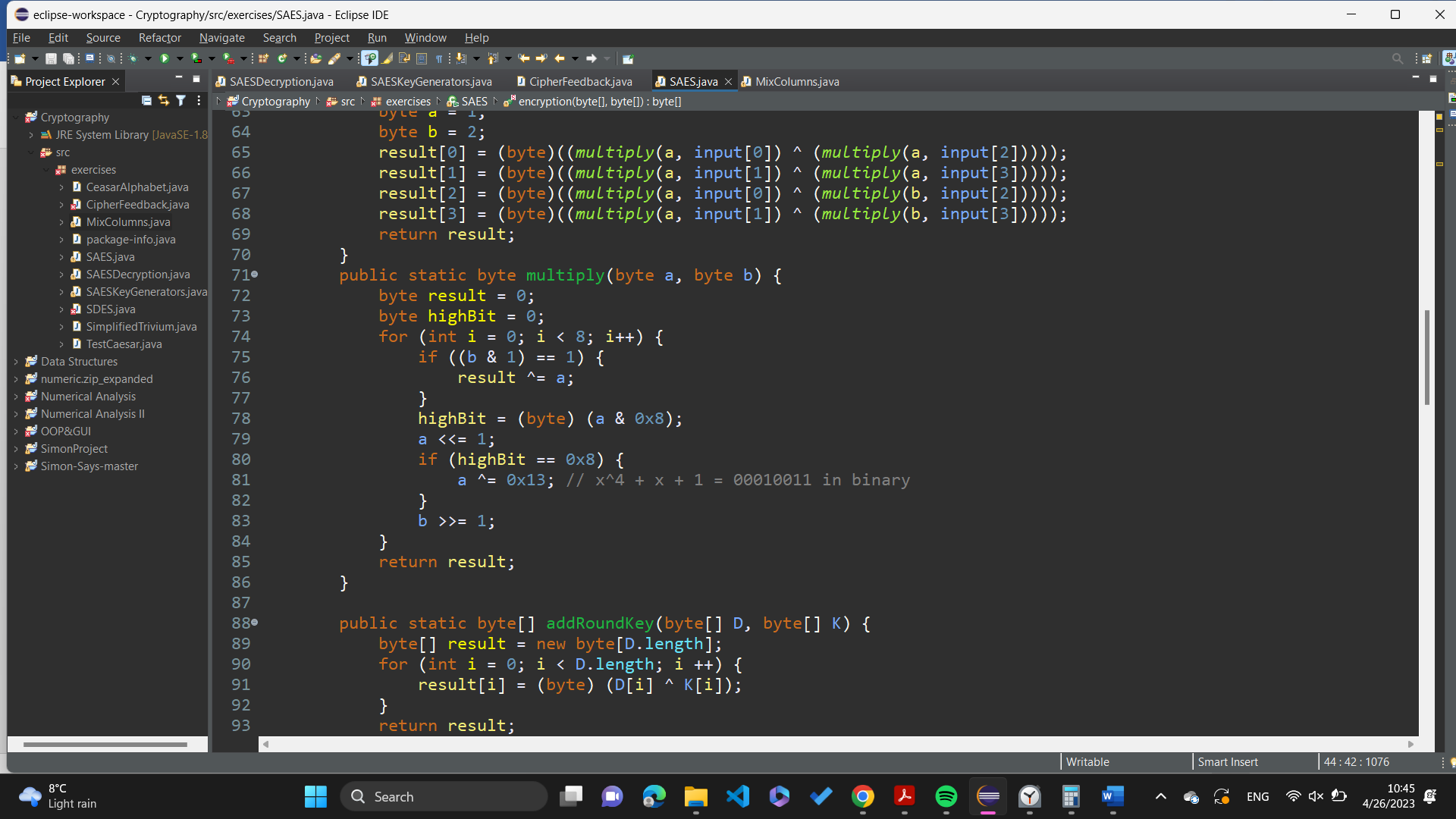


Pastaj do të aplikojmë metodën ShiftRows ku thjesht do të zhvendosim rreshtin e dytë për 4 bit majtas:

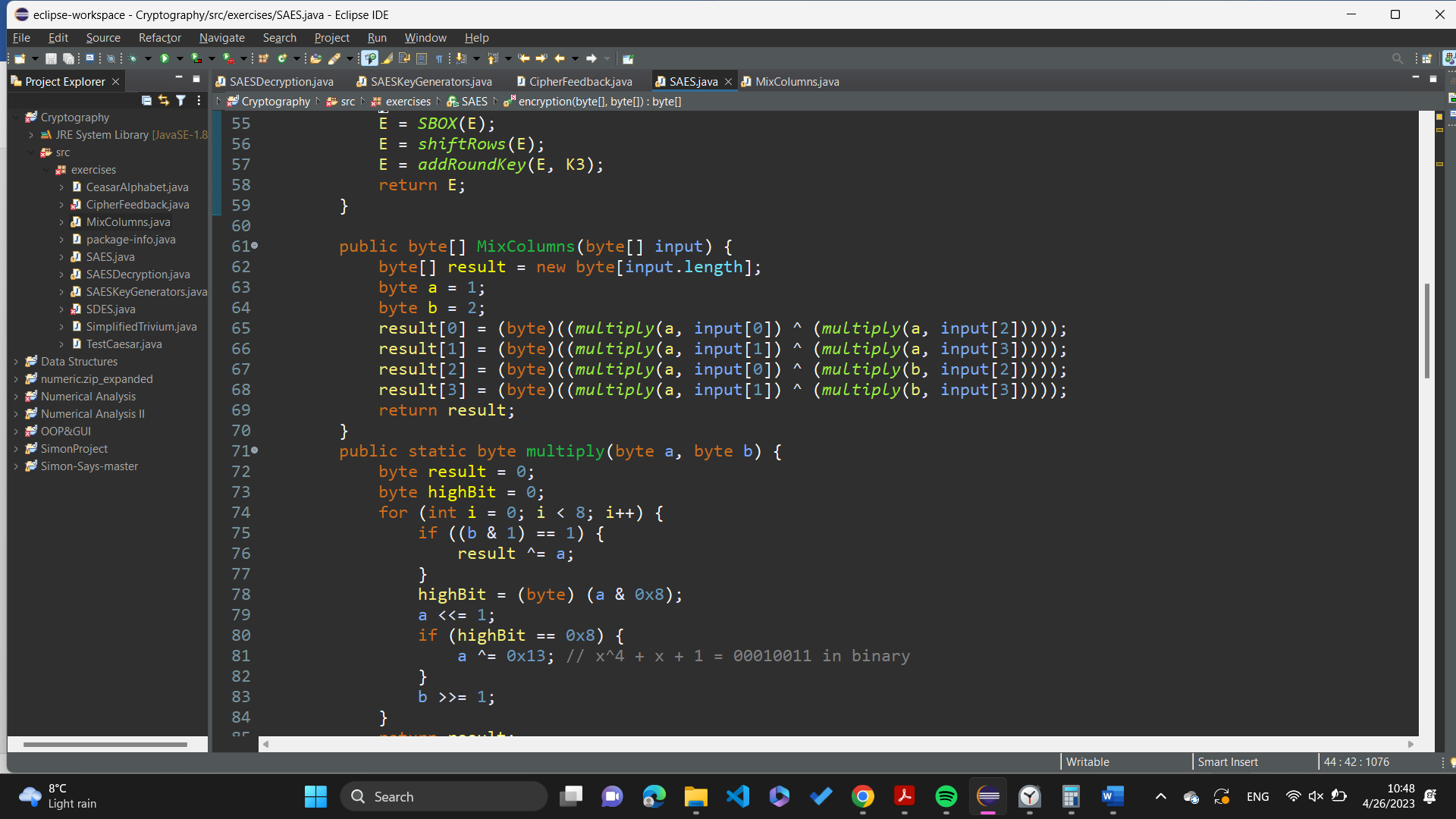


Pas shiftRows do të implementojmë MixColumns ku për mbledhjen në GF(2^4) përdorim XOR ndërsa për shumëzimin përdorim polinomin ireducibil x^4+x+1.

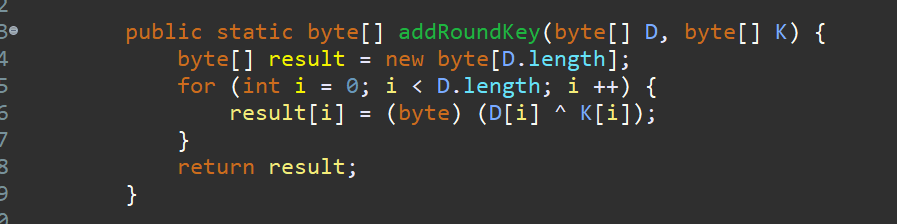
Ku shumëzimin në fushën Galua e kemi implementuar në këtë mënyrë:



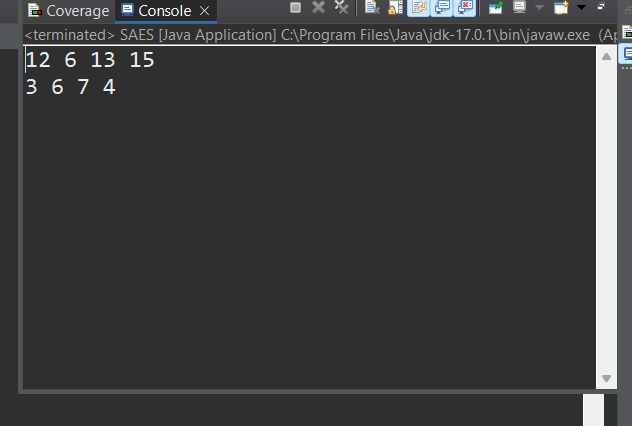
Pastaj kemi vazhduar me implementimin e MixColumns ku kemi shumëzuar matricen e kthyer nga ShiftRows me matricen {1, 1, 1, 2} ne këtë mënyrë:



Pastaj kemi implementuar metodën addRoundKey ku rezultatin e kthyer nga MixColumns me një for loop do t’a bëjmë XOR me çelësin K1 të gjeneruar nga K0:



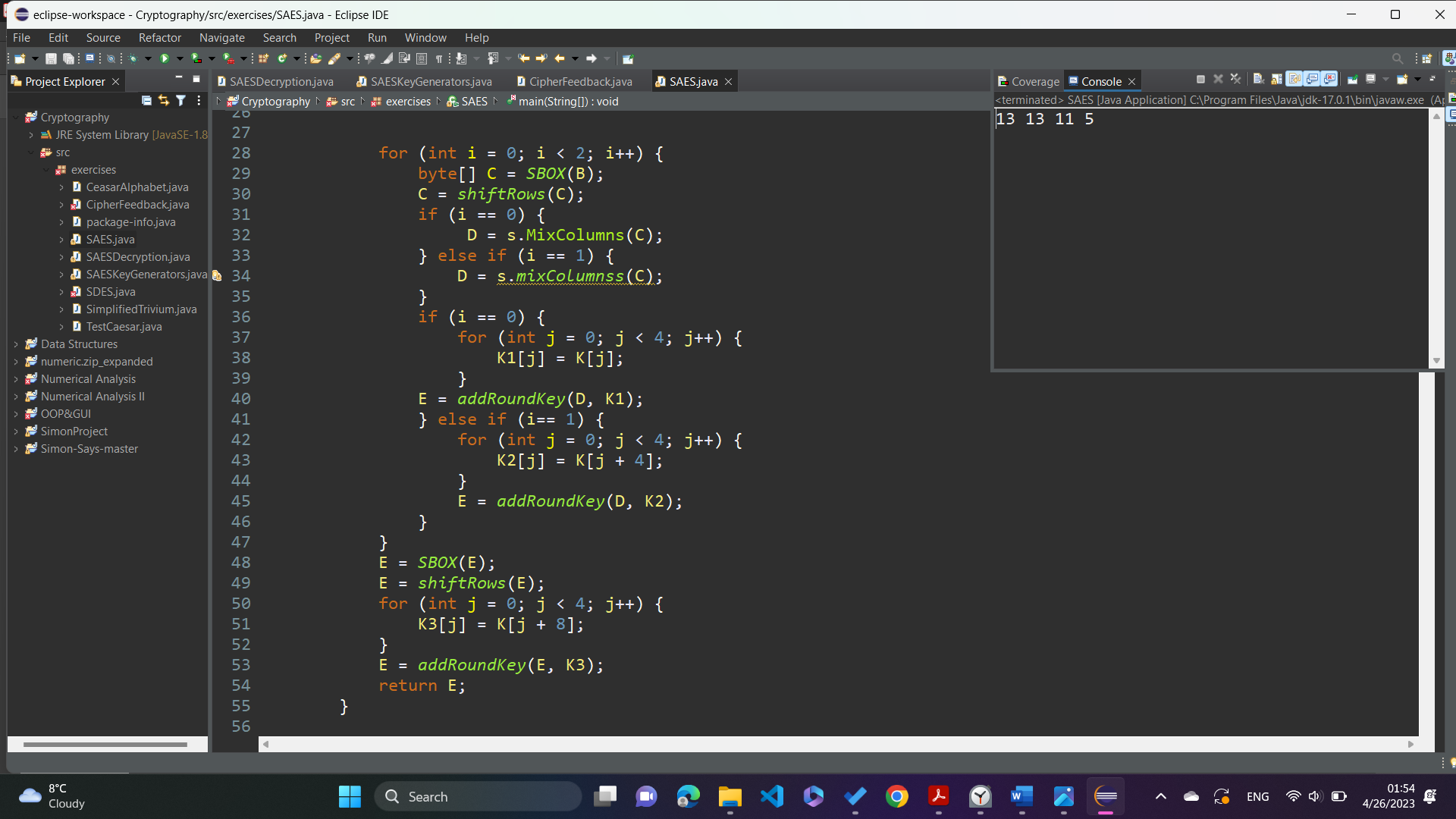
Ku si rezultat do të kemi:



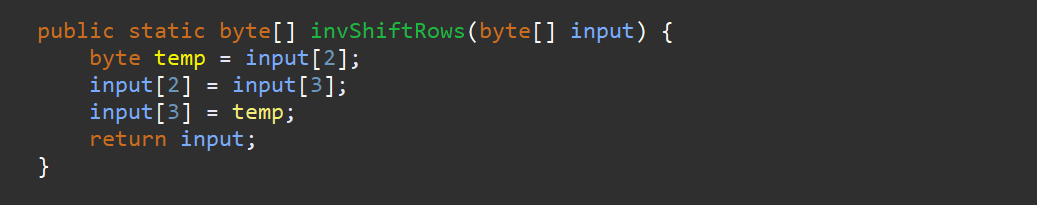


Pastaj do t’a përsërisim të njëjat metoda edhe ne raundin e dytë përveç se këtu te addRoundKey do të kemi çelësin K2.

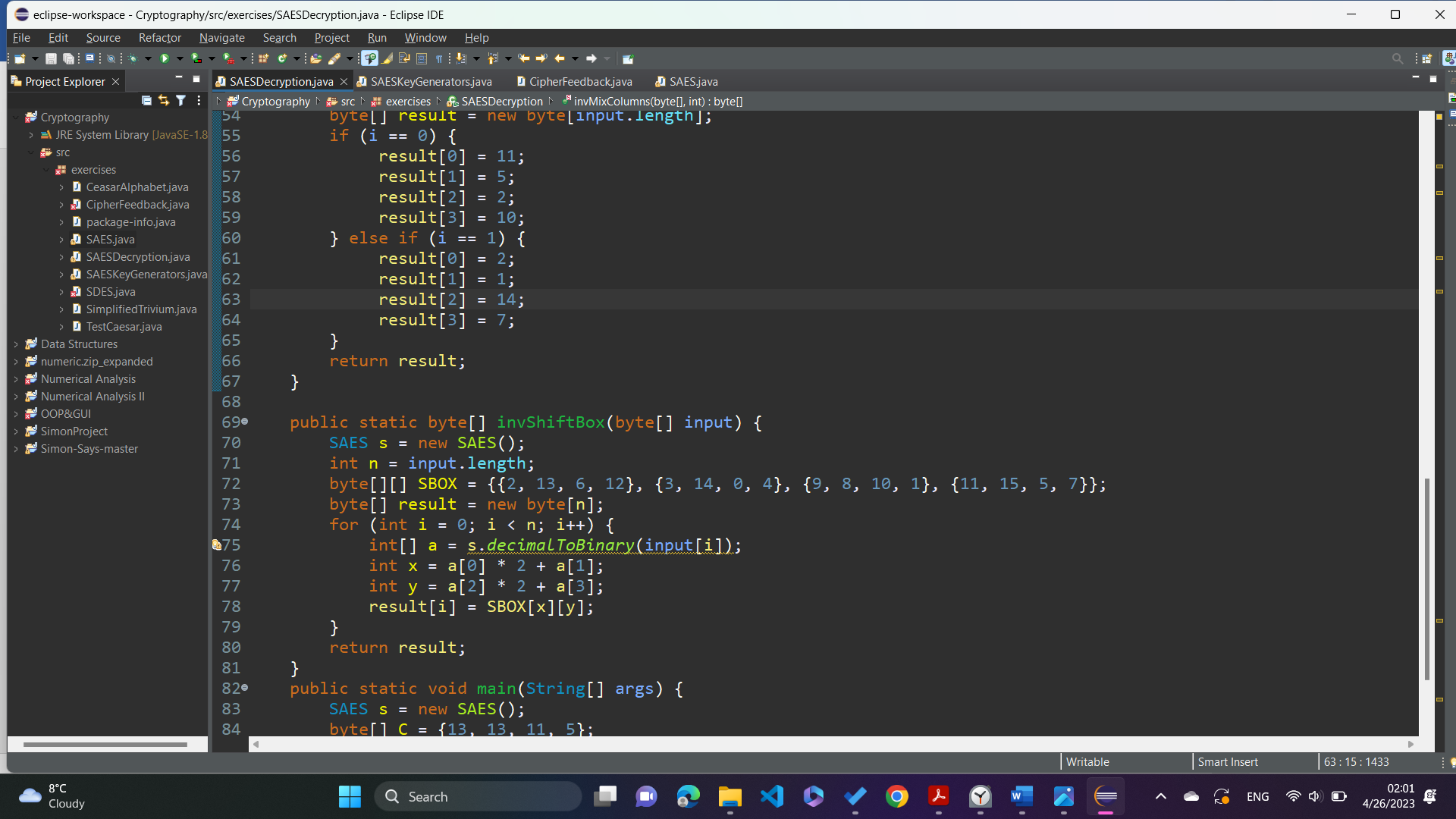
Pastaj dalim nga raundi i dytë dhe thirrim edhe njëherë metodat SBOX dhe ShiftRows e cila bëhët më pas XOR me K3 dhe na kthehet mesazhi i enkriptuar.

Kështu kemi implementuar metoden e enkriptimit.

Pastaj kemi vazhduar me dekriptimin ku kemi marrë mesazhin e enkriptuar: 3, 7, 2, 2 dhe çelësin K3 = {3, 11, 15, 6}. Ku kemi përdorur metodën e njëjtë si te enkriptimi addRoundKey duke përdorur mesazhin e enkriptuar dhe çelësin K3. Pastaj kemi implementuar një metodë invShiftRows në këtë mënyrë:



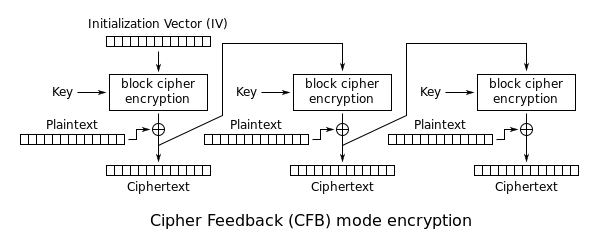
Pastaj kemi implementuar invShiftBox:



Pastaj kemi implementuar dy raunde të tjera duke përdorur metodat e njëjta: addRoundKey (ku mirret në raundin e parë çelësi K2, dhe në raundin e dytë çelësi K1), pastaj thirrim InvMixColum, InvShiftRows dhe InvSBOX. Dhe në fund rezultatin që na kthehet e bëjmë XOR me çelësin K0 ku do të na kthehet mesazhi fillestar apo plain text.

# Modi CFB

Në modalitetin CFB, procesi i enkriptimit kryhet në bazë bllok pas blloku, ku çdo bllok i tekstit të thjeshtë kodohet me bllokun e mëparshëm të tekstit të shifruar në vend që të përdoret drejtpërdrejt teksti i thjeshtë. Blloku i mëparshëm i tekstit të shifruar kthehet në procesin e enkriptimit, dhe dalja më pas XORizohet me bllokun e tekstit të thjeshtë për të prodhuar tekstin e shifruar.



Madhësia e regjistrit të reagimit është zakonisht e njëjtë me madhësinë e bllokut të algoritmit të enkriptimit. Për shembull, me AES-128, madhësia e bllokut është 128 bit (16 bit), kështu që regjistri i reagimeve do të ishte gjithashtu 128 bit.

Një avantazh i modalitetit CFB është se ai mbështet enkriptimin e transmetimit, që do të thotë se teksti i thjeshtë mund të kodohet një bajt në një kohë dhe jo në blloqe. Kjo mund të jetë e dobishme në situatat kur teksti i thjeshtë gjenerohet ose merret në kohë reale, si p.sh. me komunikimin në rrjet.

Sidoqoftë, modaliteti CFB mund të jetë i prekshëm ndaj llojeve të caktuara të sulmeve, të tilla si sulmet me bit-flipping, të cilat mund të modifikojnë tekstin e koduar në një mënyrë të parashikueshme. Si rezultat, mënyra të tjera të kriptimit, si GCM (Galois/Counter Mode), mund të preferohen në disa situata.

Për të përdorur modalitetin CFB, do t'ju duhet një algoritëm kriptimi që e mbështet atë, siç është AES.

Këtu janë hapat e përgjithshëm për enkriptimin e të dhënave duke përdorur modalitetin CFB:

* Zgjedhim një çelës dhe një vektor inicializimi (IV). Çelësi duhet të jetë një vlerë e sigurt e rastësishme dhe IV duhet të jetë një vlerë e rastësishme që është e njëjtë me madhësinë e bllokut të algoritmit të kriptimit.
* Ndajmë tekstin e thjeshtë në blloqe me të njëjtën madhësi si madhësia e bllokut të algoritmit të enkriptimit.
* Inicializojmë regjistrin e komenteve me IV.
* Enkriptojmë regjistrin e reagimeve duke përdorur algoritmin e enkriptimit me çelësin e zgjedhur.
* XOR daljen e algoritmit të enkriptimit me bllokun e parë të tekstit të thjeshtë për të prodhuar bllokun e parë të tekstit të shifruar.
* Fusim bllokun e tekstit të shifruar përsëri në regjistrin e reagimeve.
* Përsëriten hapat 4-6 për çdo bllok pasues të tekstit të thjeshtë, duke përdorur daljen e algoritmit të enkriptimit si hyrje për bllokun tjetër.
* Transmetojmë tekstin e shifruar, së bashku me IV.

Për të deshifruar tekstin e shifruar, do të ndjekim një proces pothuajse të ngjajshëm.

Modaliteti CFB është një mënyrë enkriptimi i përdorur gjerësisht dhe është zbatuar në shumë biblioteka dhe aplikacione kriptografike. Këtu janë disa shembuj se ku mund të përdoret modaliteti CFB:

* Komunikimi në rrjet: Modaliteti CFB mund të përdoret për të enkriptuar të dhënat që transmetohen përmes një rrjeti, të tilla si mesazhet me email ose transferimet e skedarëve. Për shkak se modaliteti CFB mbështet enkriptimin e transmetimit, ai mund të përdoret për të kriptuar të dhënat pasi ato transmetohen në kohë reale.
* Kriptimi i diskut: Modaliteti CFB mund të përdoret për të enkriptuar të dhënat e ruajtura në një hard disk ose pajisje tjetër ruajtëse. Për shembull, nënsistemi dm-crypt i kernel Linux përdor modalitetin CFB për të enkriptuar të dhënat në disk.
* Ruajtja në cloud kompjuterike: Modaliteti CFB mund të përdoret për të enkriptuar të dhënat e ruajtura në re, të tilla si skedarët e ruajtur në Dropbox ose Google Drive. Kjo mund të ndihmojë në mbrojtjen e të dhënave nga aksesi i paautorizuar në rast të një shkeljeje të të dhënave.
* Aplikacionet e mesazheve: Modaliteti CFB mund të përdoret për të enkriptuar mesazhet e dërguara përmes aplikacioneve të mesazheve, të tilla si WhatsApp ose Signal. Kjo mund të ndihmojë që mesazhet të mbahen private dhe të sigurta.

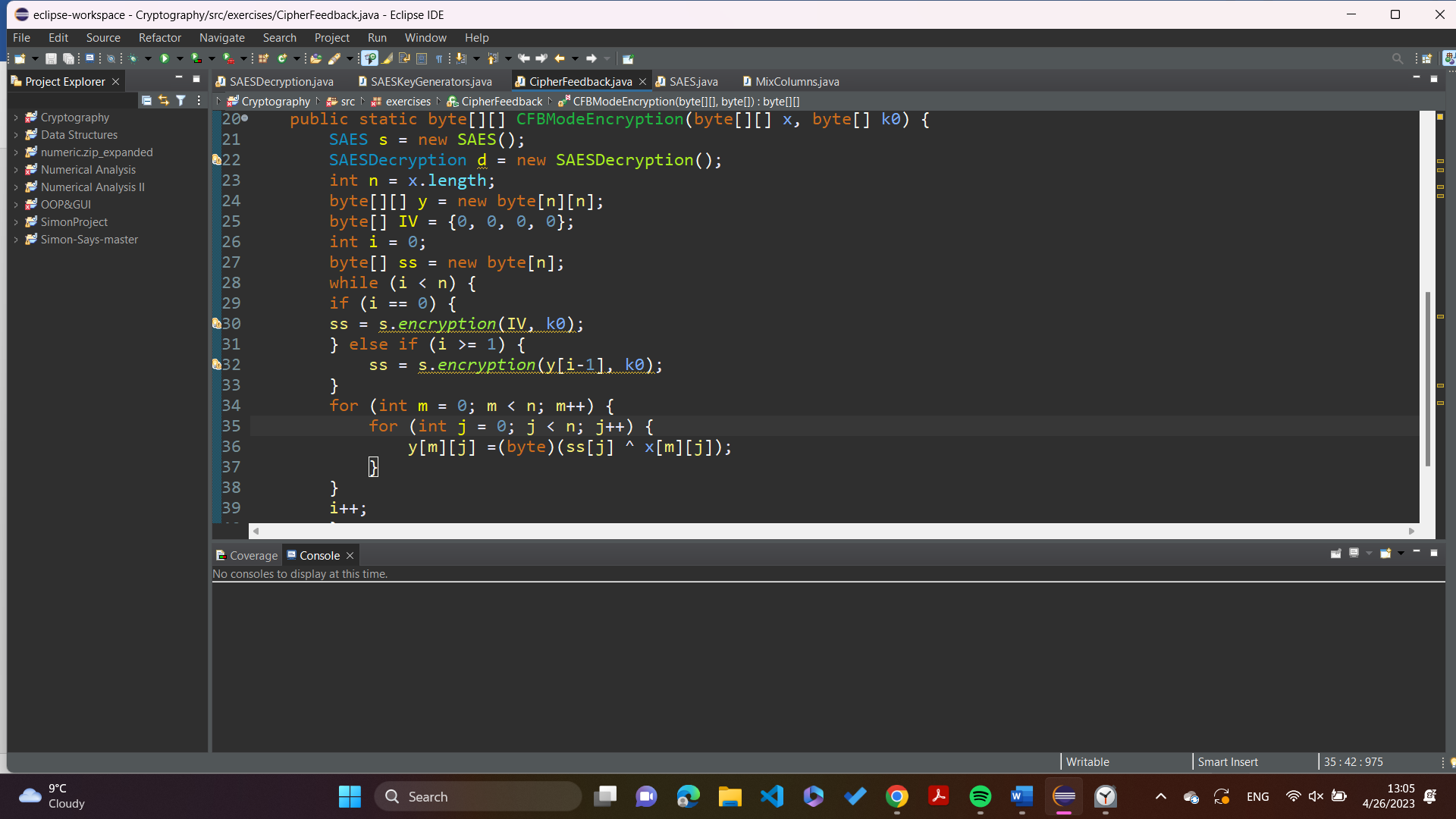
Vlen të përmendet se modaliteti CFB është vetëm një nga shumë mënyra të kriptimit që mund të përdoret në këto dhe aplikacione të tjera. Zgjedhja e mënyrës së kriptimit varet nga faktorë të ndryshëm, siç janë kërkesat e sigurisë së aplikacionit, karakteristikat e performancës së algoritmit të kriptimit dhe burimet e disponueshme.

# Implementimi në JAVA i modit CFB

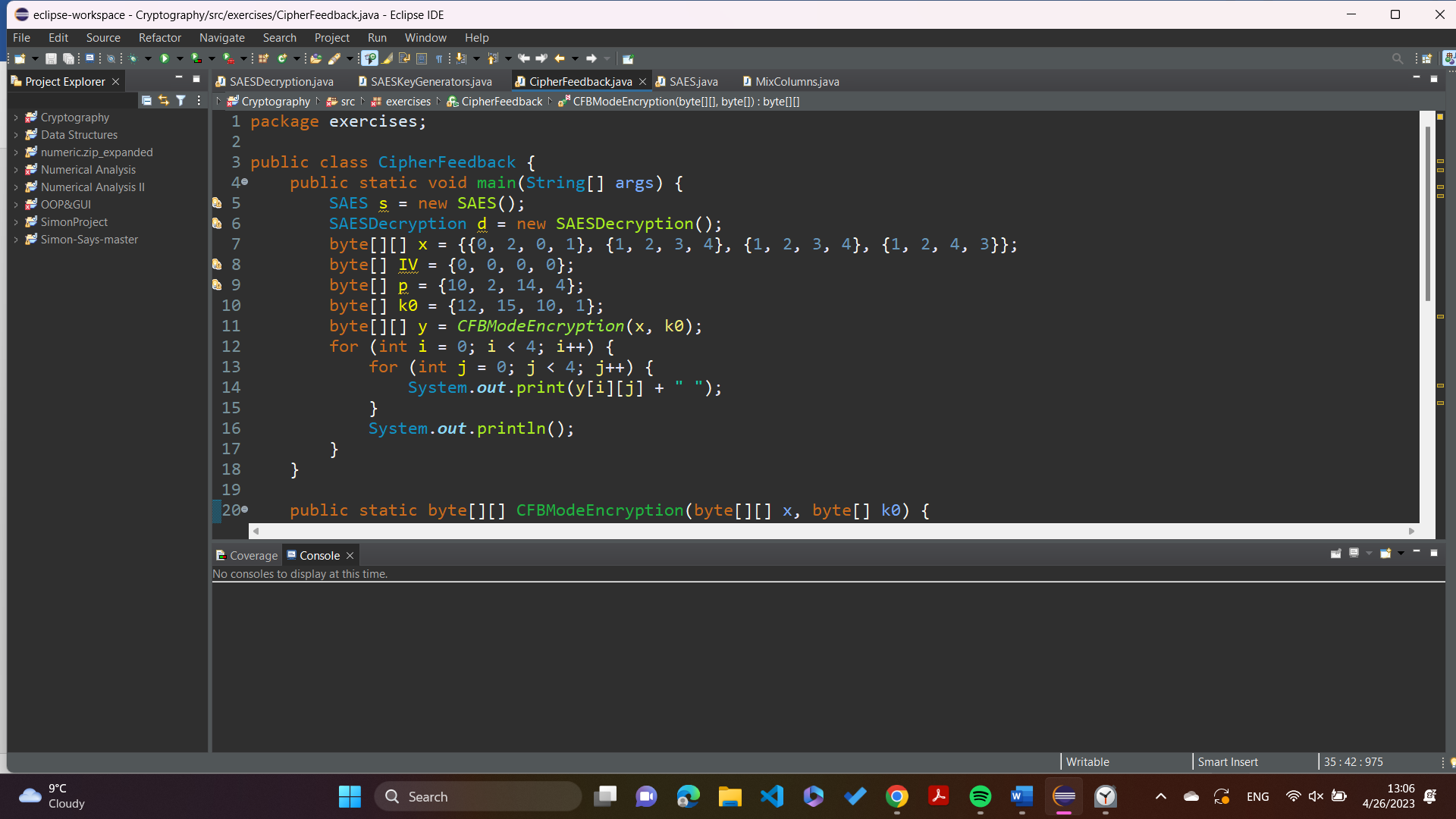
Kemi përdorur formulën për të gjetur bllokun i të enkriptuar:

**Ci = EK(Ci-1) ⊕ Bi**

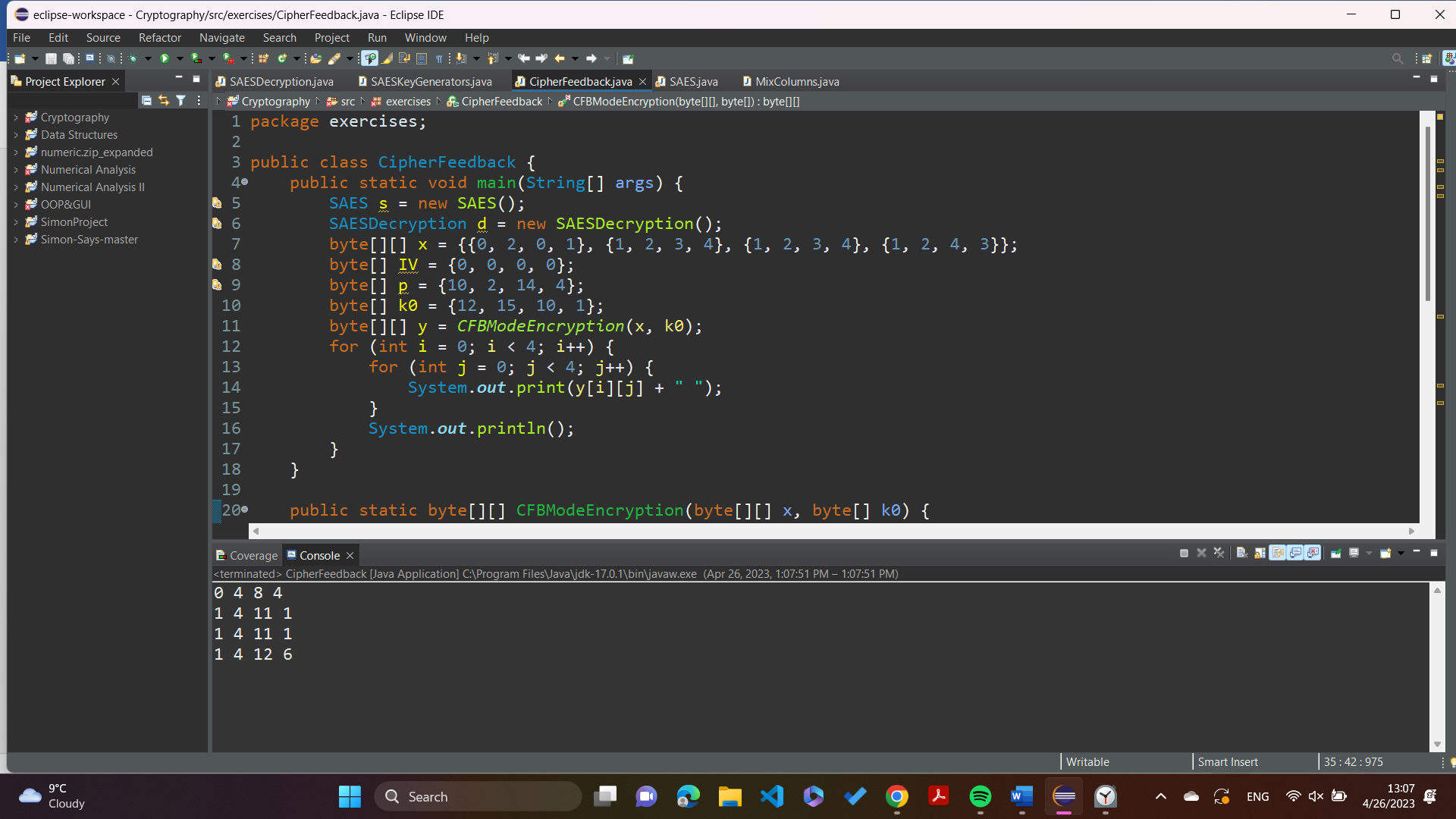
Ku vargun C e kemi ruajtuar si një matricë dy dimensionale ku secili rresht paraqet një varg y të mesazhit të enkriptuar dhe vargu B është poashtu një varg dy dimensional ku secili rresht paraqet mesazhin që dëshirojmë t’a enkriptojmë:



Ku si fillim nëse marrim këto mesazhe që dëshirojmë të enkriptojme:



Si rezultat do të kemi:



Njëjtë veprojmë edhe te dekriptimi me dallimin e vetëm që te dekriptimi si hyrje do te kemi mesazhin e enkriptuar për të fituar më pas mesazhin fillestar.

# Përfundimi

AES i thjeshtuar me modalitetin CFB është një algoritëm i lehtë kriptimi që mund të sigurojë një nivel të mirë sigurie për aplikacionet me burime të kufizuara. Duke përdorur një version të thjeshtuar të algoritmit të kriptimit AES dhe mënyrës së funksionimit CFB, ky zbatim është në gjendje të arrijë kriptim efikas dhe efektiv të të dhënave.