# Feladat

Java JSSE-t használva végezzük el a következő feladatokat:

1. Írjunk egy kliens alkalmazást, ami SSL-t használva kapcsolódik a https://bnr.ro/Home.aspx címre és egy HTTP GET kérést küld a Román Nemzeti Bank szerverének. A szerver által válaszként küldött HTML tartalmat a kliens mentse le egy szöveges állományba. A kapcsolat létrejötte után a képernyőre írja a tanúsítvány főbb adatait: verziószám, szériaszám, a tanúsító hatóság neve, kibocsátás dátuma, érvényességi ideje, a tanúsítvány alanyának adatai (név, internetes cím(ei)), a nyilvános kulcs adatai (a titkosító eljárás típusa, az alany nyilvános kulcsa). Ezeket az adatokat a böngészőben is meg lehet nézni (például: https://support.mozilla.org/en-US/kb/secure-website-certificate), de a kliens legyen képes kinyerni és kiírni a főbb adatokat.
2. Írjunk egy szerver alkalmazást, ami az 1. pontnál lementett szöveges állomány (HTML) tartalmát téríti vissza egy HTTP GET kérésre válaszolva, SSL kapcsolaton keresztül. A szervert lokálisan futtatjuk (localhost) és egy általunk készített tanúsítványt használ. A tanúsítvány azt állítja, hogy ez a helyileg futtatott szerver a Román Nemzeti Bank szervere (tehát az internetes címe bnr.ro és minden egyéb adatot átvesz az igazi BNR tanúsítványtól). Valódi tanúsító hatóság ezt természetesen nem fogja igazolni, ezért a tanúsítványt saját magunk írjuk alá (self-signed certificate). Próbáljuk "becsapni" a klienst! Az operációs rendszer hosts állományába ideiglenesen írjuk be, hogy a bnr.ro webhely IP címe 127.0.0.1 (localhost). Győződjünk meg arról, hogy az átállítás sikerült (futtassuk például a "ping bnr.ro" parancsot, ami az átállítás után a 127.0.0.1 IP címről válaszol). Ezzel azt a helyzetet szimuláljuk, amikor egy támadó "man-in-the-middle" típusú támadást próbál végrehajtani és átállítja a DNS válaszát. A kliens alkalmazás (aminek rá kell jönnie a "csalásra") egy hibaüzenettel tájékoztatja a felhasználót arról, hogy baj van, ez nem az igazi Román Nemzeti Bank szervere!
3. Hozzunk létre egy saját, elsődleges tanúsító hatóságot (RootCA), és még két másikat: az egyik a kliens tanúsítványát fogja aláírni (ClientCA), a másik pedig a szerverét (ServerCA). A ClientCA és a ServerCA tanúsítványát az elsődleges tanúsító hatóság (RootCA) fogja aláírni. A hatóságok tanúsítványait a következő adatokkal hozzuk létre:
    - birtokló neve (Common Name): "[scs.ubbcluj azonosító]-RootCA", "[scs.ubbcluj azonosító]-ClientCA", illetve "[scs.ubbcluj azonosító]-ServerCA"
    - servezet (Organization): "BBTE"
    - cím: Country - "RO", State/Province - "Kolozs", Locality - "Kolozsvár"
    - titkosító eljárás típusa: elliptikus görbe (Elliptic Curve), kulcsméret: 256 bit
    - érvényesség: 2021. február 28.
4. Készítsünk egy tanúsítványt a kliens számára (amit a ClientCA hitelesít) a következő adatokkal:
    - birtokló neve (Common Name): "[scs.ubbcluj azonosító]-client"
    - servezet (Organization): "BBTE"
    - cím: Country - "RO", State/Province - "Kolozs", Locality - "Kolozsvár"
    - titkosító eljárás típusa: elliptikus görbe (Elliptic Curve), kulcsméret: 256 bit
    - érvényesség: 2021. február 28.
5. Készítsünk egy tanúsítványt a szerver számára (amit a ServerCA hitelesít):
    - birtokló neve (Common Name): a számítógépünk neve (hostname)
    - servezet (Organization): "BBTE"
    - cím: Country - "RO", State/Province - "Kolozs", Locality - "Kolozsvár"
    - titkosító eljárás típusa: RSA, kulcsméret: 2048 bit
    - érvényesség: 2021. február 28.

6. Készítsük el az 1. pontnál létrehozott kliens, illetve a 2. pontnál létrehozott szerver egy-egy új változatát úgy, hogy a 4. illetve az 5. pontnál létrehozott tanúsítványokat használják. Az új szerver tehát hitelesíti a klienst, és csak a ClientCA által aláírt tanúsítvánnyal rendelkező klienstől fogad el kapcsolatot. Az adatcsere a kliens és a szerver között lehet ugyanaz, mint amit a 2. pontnál kértünk. Próbáljunk kapcsolódni a szerverhez a régi klienssel is (ami nem rendelkezik a 4. pontnál létrehozott tanúsítvánnyal). A szervernek ezt a kapcsolatot vissza kell utasítania!

A feladat megoldásához ajánlott a Java 8 vagy 11 használata, továbbá telepíteni kell az OpenSSL-t (https://www.openssl.org). A két kliens illetve a két szerver alkalmazásnak nem szükséges grafikus kezelőfelületet létrehozni, elég a parancssoros felület. A 2-5 pontoknál a felhasznált OpenSSL parancsokat (openssl, keytool) mentsétek el egy szöveges állományba (magyarázattal együtt). Ezt az állományt érdemes megtartani, jól jöhet majd valamikor...

## Beadandó anyagok:

A két kliens és a két szerver forráskódja, a 2-5 pontoknál létrehozott tanúsítványok (nem titkosított, *.pem formátumban) és az ezeket létrehozó parancsokat tartalmazó szöveges állomány.

## Segédanyagok:

- Java, JSSE:
  - https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html (JavaSE 8)
  - https://docs.oracle.com/en/java/javase/11/security/java-secure-socket-extension-jsse-reference-guide.html#GUID-93DEEE16-0B70-40E5-BBE7-55C3FD432345 (JavaSE 11)
  - https://www.javaworld.com/article/2075291/build-secure-network-applications-with-ssl-and-the-jsse-api.html
  - https://www.naschenweng.info/2018/02/01/java-mutual-ssl-authentication-2-way-ssl-authentication/
  - https://www.opencodez.com/java/implement-2-way-authentication-using-ssl.htm
- OpenSSL:
  - https://www.feistyduck.com/books/openssl-cookbook/
  - https://blog.dbi-services.com/pki-ssl-certificates-management-with-java-keytool-and-openssl/
  - https://gist.github.com/fntlnz/cf14feb5a46b2eda428e000157447309
  - https://smartnets.wordpress.com/2017/04/27/create-certificate-chain-and-sign-certificates-using-openssl/

## Pontozás

1. 2 pont
2. 3 pont
3. 2 pont
4. 0.5 pont
5. 0.5 pont
6. 2 pont