What you will find?

1) OAuth2 Protocol-based Authorization token generation

2) Details of creating a self-signed Certificate chain with Java Keytool

3) Java Cryptography Eco-System

4) Creating local tomcat servers alias-ed with separate domain names

5) Use of Mongo database and MySQL database to store user-encrypted

6) Use of Bouncy Castle for Elliptic curve (EC) key

7) Use of JWKS for Digital signature verification

**Self-Signed Digital Certificate Chain using java keytool**

A certificate chain created with a self-signed top level certificate (RootCA), an intermediate CA (IntermediateCA) signed by RootCA and an End-Entity certificate (OAuth2Server.cer / OAuth2Client.cer) signed by IntermediateCA.  
  
The following steps illustrate the creation of the three certificates.

Note:- To use keytool look at the footnote below

**1) Create a Root CA keypair and store it in a keystore**

keytool -genkeypair -alias root -dname "cn=RootCA, ou=Root\_CertificateAuthority, o=CertificateAuthority, c=IN" -keyalg RSA -keystore c:/caKeyStore.keystore -storepass password -keypass password -ext KeyUsage=digitalSignature,keyCertSign -ext BasicConstraints=ca:true,PathLen:3 -validity 1000   
  
Note:- Forget not to include the ‘BasicConstraints’ = ca : true, else the RootCA would not be trusted by the browser or java as a trusted signing authority.

**2) Create an Intermediate CA keypair**

keytool -genkeypair -alias intermediate -dname "cn=IntermediateCA, ou=Intermediate\_CertificateAuthority, o=CertificateAuthority, c=IN" -keyalg RSA -keystore c:/caKeyStore.keystore -storepass password -keypass password -ext KeyUsage=digitalSignature,keyCertSign -ext BasicConstraints=ca:true,PathLen:3 -validity 1000  
  
Side Note:- The ‘PathLen’ attribute signifies the maximum number of certificates that can be obtained in the certificate chain signed by the issuing CA certificate.

**Initiate the certificate generation**

**3) Create server certificate and it’s keystore**

keytool -genkeypair -alias server -dname "cn=OAuth2Server, ou=Java, o=Oracle, c=IN" -keyalg RSA -keystore c:/oauth2server.jks -storepass password -keypass password -validity 1000 -ext KeyUsage=digitalSignature,dataEncipherment,keyEncipherment,keyAgreement -ext ExtendedKeyUsage=serverAuth,clientAuth

Note: - The oauth2server.jks is the keystore currently storing the keypair from which we will later create the certificate (i.e. the publickey in a wrapper) and store it in the same keystore

**4) Create a certificate request (CSR) and certificate for IntermediateCA**

keytool -certreq -alias intermediate -storepass password -keystore c:/caKeyStore.keystore -keyalg RSA | keytool -alias root -gencert -ext san=dns:intermediate -storepass password -keystore c:/caKeyStore.keystore -keyalg RSA -validity 1000 -ext KeyUsage=digitalSignature,dataEncipherment,keyEncipherment,keyAgreement,keyCertSign -ext ExtendedKeyUsage=serverAuth,clientAuth -ext BasicConstraints=ca:true,PathLen:3 -rfc | keytool -alias intermediate -importcert -storepass password -keyalg RSA -keystore c:/caKeyStore.keystore

Note: - The keyword san (Subject Alternate Name) is used to configure the name of the server certificate as would be visible in the browser (moden browser book for SAN to validate rather than CN).

Side Note: - The pipeline operator ‘|’ is used to chain into a single command. The command to the left of the operator produces an action that is a prerequisite for the command to the right.

**5) Export the root certificate RootCA and intermediateCA and import them to oauth2server keystore (required for trust chain) before the server certificate is imported in server keystore**

keytool -exportcert -alias root -storepass password -keystore c:/caKeyStore.keystore -validity 1000| keytool -importcert -alias root -keystore c:/oauth2server.jks -storepass password -noprompt –trustcacerts

keytool -exportcert -alias intermediate -storepass password -keystore c:/caKeyStore.keystore -validity 1000 | keytool -importcert -alias intermediate -keystore c:/oauth2server.jks -storepass password -noprompt -trustcacerts

Note: - The order of certificate insertion in the keystore is important and should be followed in the order of **root CA -> intermediate CA -> server**

**6) Create certificate CSR for the server and then import it into server keystore**

keytool -certreq -alias server -storepass password -keystore c:/oauth2server.jks -keyalg RSA | keytool -alias intermediate -gencert -ext san=dns:OAuth2Server -storepass password -keystore c:/caKeyStore.keystore -keyalg RSA -ext KeyUsage=digitalSignature,dataEncipherment,keyEncipherment,keyAgreement -ext ExtendedKeyUsage=serverAuth,clientAuth -rfc | keytool -alias server -importcert -storepass password -keyalg RSA -keystore c:/oauth2server.jks -validity 1000

**7) Delete the imported CA certificates from the server keystore**

keytool -delete -alias root -keystore c:/oauth2server.jks -storepass password

keytool -delete -alias intermediate -keystore c:/oauth2server.jks -storepass password

Slide Note: - The CA certificates were imported in the server keystore in specified order only for trust generation(chain validation) and are not needed later.

**8) Create a trust keystore with root and the intermediate certificates: (first root then intermediate)**

keytool -exportcert -alias root -storepass password -keystore c:/caKeyStore.keystore | keytool -importcert -alias root -keystore c:/trust.jks -storepass password -trustcacerts -noprompt

keytool -exportcert -alias intermediate -storepass password -keystore c:/caKeyStore.keystore | keytool -importcert -alias intermediate -keystore c:/trust.jks -storepass password -trustcacerts -noprompt

Side Note: - The trust keystore will be required for the SSL (‘https’) configuration of the tomcat server.

**9) To extract certificates**

keytool -exportcert -alias root -keystore C:/trust.jks -file c:/rootCA.cer -storepass password

keytool -exportcert -alias intermediate -keystore C:/trust.jks -file c:/intermediateCA.cer -storepass password

keytool -exportcert -alias server -keystore C:/oauth2server.jks -file c:/oauth2server.cer -storepass password

Side Note: - For configuring SSL on a Tomcat server, you'll need a keystore containing your server certificate and potentially a truststore containing the CA certificates.

**10) Import rootCA into jvm’s truststore (cacerts)**

keytool -import -trustcacerts -keystore "C:\Program Files\Java\jdk-17.0.4\lib\security\cacerts " -storepass changeit -alias rootCA -file “C:\rootCA.cer”

Side Note: - To obtain the path to ‘cacerts’ in your windows system start from your ***Java*** home directory ***Program Files*** and navigate to a***security*** folder in ***lib***. Observe the version number of your jdk may vary. This is important for your java application to trust your serve certificate (signed by intermediate CA). Observe only the root certificate is required to be imported in jvm’s cacerts for certificate chain validation.

**11) Import the root certificate in your browser’s trusted certificate directory**

Open **RUN** dialog box (Win + R), then type **certmgr.msc**. Click **Trusted Root Certificate Authorities** in the left pane. Right click the subfolder Certificates. Click **All Task -> Import**. This will open the **Certificate Import Wizard**. Click Next and browse to your folder containing your certificate file (rootCA.cer). Click **Next -> Next -> Finish**. This is quintessential for your browser to trust your root certificate of your certificate chain, else your server certificate will **NOT** be validated for a self signed certificate chain.  
  
This ends your certificate chain generation and their imports into the trusted certificate stores. Incase you want to verify your import in cacert use the following command.

To verify use the following command

(keytool -list -v -keystore " C:\caKeyStore.keystore " -storepass password -alias rootCA) Incase you want to verify your import in cacert

keytool -list -v -keystore "C:\Program Files\Java\jdk-20\lib\security\cacerts" -alias rootCA

Note :- To open keytool command open a command terminal (type cmd in windows search box) with administrative privilege. Then change to bin directory of your jdk (type ***C:\Program Files\Java\jdk-version no\bin*** in your command prompt).