Semi-Automated Bash Script to generate TLS Certs and Keys

This script requires a self-signed Certificate Authority (CA) to be present to self-sign all certs. If your organisation has restrictions on self-signed root CA you can still adapt this script to generate all certreq and get them signed by external root CA.

For the purpose of this script, you can quickly generate a CA cert using:

```
openssl req -new -x509 -keyout ca-key.key -out ca-cert.crt -days 365 -passin pass:sameAsOthers
```

For convenience, you can keep the password same for everything. Always store passwords separately from code, and never share it with anyone outside the authorized user list.

Since hostname verification is enabled by default, the script honours X509v3 extensions so that Subject Alt Names (SAN) are added. By default, OpenSSL *will not* add SAN info in your signed cert unless ext file is also provided. This script does just that. This has both SAN and CN setup for this.

Please adjust the variable values and X500 *Distinguished Name* settings to your need. For Windows users, you can use MinGW bash shell (if you use Git, it's the best route), you can use that to run this script.

```
#!/bin/bash
# Setup params
PASSWORD=gibrishwordforyou
VALIDITY=365
PROJECT_PREFIX=my-kafka
BROKERS='funnyaddress.com unfunnyguy.com'
CLIENT_ALIAS=myclientname
CLIENT_KEYSTORE=$PROJECT_PREFIX.client.keystore.jks
CLIENT_CERT_FILE=$PROJECT_PREFIX-client-cert-file
CLIENT_CERT_SIGNED=$PROJECT_PREFIX-client-cert-signed.crt
CA_ROOT_ALIAS=ca-root
CA_CERT_NAME=ca-cert.crt
CA_KEY=ca-key.key
BROKER_TRUSTSTORE=$PROJECT_PREFIX.truststore.jks
echo -e "OpenSSL based Keys/Cert generation for Kafka"

# Generate for all brokers
echo -e "\n\n###\n###Generating Keys for listed brokers = $BROKERS\n\n###\n###"
for BROKER in $BROKERS
do
keytool -genkeypair -keysize 2048 -keyalg RSA -keystore $PROJECT_PREFIX-$BROKER.jks -alias $BROKER -dname
"CN=$BROKER,OU=SomeUnit,O=SomeOrg,L=London,S=England,C=GB" -ext SAN=DNS:$BROKER -validity $VALIDITY -keypass
$PASSWORD -storepass $PASSWORD
echo -e "subjectAltName=DNS:$BROKER" > $PROJECT_PREFIX-x509v3-$BROKER.ext
done
echo -e "\n\n###\n###Signing and importing certificates using CA file $CA_CERT_NAME and CA keys file
$CA_KEY\n\n###\n###"
for BROKER in $BROKERS
do
keytool -certreq -keystore $PROJECT_PREFIX-$BROKER.jks -alias $BROKER -ext SAN=DNS:$BROKER -file
$PROJECT_PREFIX-$BROKER-cert-file -storepass $PASSWORD -keypass $PASSWORD
openssl x509 -req -CA $CA_CERT_NAME -CAkey $CA_KEY -in $PROJECT_PREFIX-$BROKER-cert-file -out
$PROJECT_PREFIX-$BROKER-cert-signed.crt -days $VALIDITY -CAcreateserial -passin pass:$PASSWORD -extfile
$PROJECT_PREFIX-x509v3-$BROKER.ext
done
echo -e "\n\n###\n###Importing CA root $CA_CERT_NAME and signed broker certs into keystoere\n\n###\n###"
for BROKER in $BROKERS
do
keytool -import -keystore $PROJECT_PREFIX-$BROKER.jks -alias $CA_ROOT_ALIAS -file $CA_CERT_NAME -storepass
$PASSWORD -keypass $PASSWORD
keytool -import -keystore $PROJECT_PREFIX-$BROKER.jks -alias $BROKER -file $PROJECT_PREFIX-$BROKER-cert-signed.
crt -storepass $PASSWORD -keypass $PASSWORD
done
echo -e "\n\n###\n###Preparing Client Certificates and keystores###\n\n###"
keytool -genkeypair -keysize 2048 -keyalg RSA -keystore $CLIENT_KEYSTORE -alias $CLIENT_ALIAS -dname
"CN=$CLIENT_ALIAS,OU=SomeUnit,O=SomeOrg,L=London,S=England,C=GB" -validity $VALIDITY -storepass $PASSWORD -
keypass $PASSWORD
keytool -certreq -keystore $CLIENT_KEYSTORE -alias $CLIENT_ALIAS -file $CLIENT_CERT_FILE -storepass $PASSWORD -
keypass $PASSWORD
openssl x509 -req -CA $CA_CERT_NAME -CAkey $CA_KEY -in $CLIENT_CERT_FILE -out $CLIENT_CERT_SIGNED -days
$VALIDITY -CAcreateserial -passin pass:$PASSWORD
keytool -import -keystore $CLIENT_KEYSTORE -alias $CA_ROOT_ALIAS -file $CA_CERT_NAME -storepass $PASSWORD -
keypass $PASSWORD
keytool -import -keystore $CLIENT_KEYSTORE -alias $CLIENT_ALIAS -file $CLIENT_CERT_SIGNED -storepass $PASSWORD -
keypass $PASSWORD
###
# Once everything is done - import CA into broker and client trust stores correctly
```