File: server.py

```
import socket
import hashlib
import time
import math
from Crypto.Util import number
import random

s = socket.socket()

s.bind(("localhost", 8888))

s.listen(5)

c, addr = s.accept()
print ("Connected to client ", addr)

message = raw_input("Enter message:")

#finding HASH of message
hashObject = hashlib.sha1(message)
hexDig = hashObject.hexdigest()
hashInt = int(hexDig, 16)

#KEY GENERATION
p=0
q=0
g=0
A=0
a=0

q = number.getPrime(8) #8 is bit length for 'q'

while True:
     p = number.getPrime(64) #64 bit length for 'p'
     if( (p-1) % q == 0):
          break

while True:
     x = random.randint(1, p) #take {1, ...., p-1}
     g = pow(x, ((p-1)/q), p)   #find value of 'g'
     if ( g != 1 ):
          break

a = random.randint(0, q) #take {0, ...., q-1} private key is 'a'
A = pow(g, a, p)#find public key 'A'

#public key is (p, q, g, A)
print "Public Key is: "
print "p =", p
print "q =", q
print "g =", g
print "A =", A

#SIGNING
r = 0
s = 0
while True:
     k = random.randint(1, q) #take {1, ...., q-1}
     r = pow(g, k, p) % q
     if( r != 0 ):
          break
```

```
#find modular multiplicative inverse of k
kInverse = 0
while(k*kInverse % q != 1):
     kInverse = random.randint(1, q)

s = kInverse * (hashInt + a * r) % q

#signature of message is (r, s)
print "\nDigital Signature of message is: \n( r = ", r, ",\n s = ", s, " )"

#send the message, public key and digital signature to Client
c.send(message+"\t"+str(p)+"\t"+str(q)+"\t"+str(g)+"\t"+str(A)+"\t"+str(r)
+"\t"+str(s))
c.close()

"""To install pycrypto (from Crypto.Util import number) for number.getPrime
use command $sudo pip install pycrypto"""
#Reference Link:
https://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1003_DSA.pdf
```

File: client.py

```
import socket
import hashlib
import random

s = socket.socket()
s.connect(("localhost", 8888))

receivedString = s.recv(4096)
message =  receivedString.split("\t")[:-6]
message = "\t".join([word for word in message])

print "\nReceived Message: ", message
p = int(receivedString.split("\t")[-6])
q = int(receivedString.split("\t")[-5])
g = int(receivedString.split("\t")[-4])
A = int(receivedString.split("\t")[-3])
r = int(receivedString.split("\t")[-2])
s = int(receivedString.split("\t")[-1])

print "\nPublic Key received is: "
print "p = ", p
print "q = ", q
print "g = ", g
print "A = ", A

print "\nDigital Signature received is: \n( r = ", r, ",\n s = ", s, " )"

#finding HASH of message
hashObject = hashlib.sha1(message)
hexDig = hashObject.hexdigest()
hashInt = int(hexDig, 16)

#finding sInverse
#find modular multiplicative inverse of s
sInverse = 0
while(s*sInverse % q != 1):
     sInverse = random.randint(1, q)
```

```
#Verification of Digital Signature
v = 0
if((r>=1 and r<=q-1) and (s>=1 and s<=q-1)):
      u1 = (sInverse * hashInt) % q
      u2 = (r * sInverse) % q
      v = ( ( pow(g, u1, p) * pow(A, u2, p) ) % p ) % q
      print "\nv = ", v

      if(v == r):
            print "Here, v = r"
            print "\nHence, Digital Signature Verified and Accepted!!\n"
      else:
            print "\nDigital Signature can't be Verified, hence, Rejected!!\n"
else:
      print "\nDigital Signature can't be Verified, hence, Rejected!!\n"
```

#OUTPUT: