```
import hashlib
import random
import time



def gen_key_pair(p,q,h,x):
#This generates the key pair for the DSA encrpytion scheme, 4 parameters are taken and p and q as large prime numbers h which is an integer between 2 and p-2 x is random such that 1 <= x <= q-1

#Generating two values g and y
    g = pow(h, (p-1)//q, p)

    y = pow(g, x, p)
#pow function will return the value of i to the power of e and since there is a third value present it will return i to the power e with modulus n
    return g, y


def sign(p,q,g,x,k, message_hash):
#The sign function will hav 6 parameters including message hash this would be the ahsh value generated for the message that needs to be signed
    r = 0
#By generating the signature for the message using the private key x and a random number k by rasing r = (g^k mod p )mod q and s = (k^-1 *(Message hash + x*r))mod q
    s = 0


    while r == 0 or s == 0:
#This while loop will ensure that r and s are not euqale to 0 if either value is 0 the signature will then be invalidated to prevent the k reuse case from occouring
        k_inverse = pow(k, q-2, q)
#If either value of r or s is equal to 0 the loop will continue with a new value of k that is generated randomly once r and s are non zeros return values
        r = pow(g, k, p) % q
        s = (k_inverse * (message_hash + x *r)) % q
        if r == 0 or s == 0:
            k = random.randint(1, q-1)

    return r, s

def verification(p, q, g, y, message_hash, r, s):
#The verification function will take in same parameters as before but this time including r and s It will proceed to verify the authenticitiy of the message using the public key and the signature
    w = pow(s, q-2, q)
#This will proceed to compute value v based on r, s, message_hash, g, and y then following it will compare if v = r if so the signature is valid and authentic if not you know that there is someones trying to access illigitemetly
    u1 = (message_hash * w) % q
    u2 = (r * w) % q
    v = ((pow(g, u1, p) * pow(y, u2, p)) %p) %q

    return v == r, w, u1, u2, v

#Example 1
print("Example 1", "\n")
p, q, h, x, k, H_M1, H_M2 = 7, 3, 3, 2, 1, 3, 4
```

```
g, y = gen_key_pair(p,q,h,x)
print("g = ", g, "\n","y = ", y)

r, s = sign( p, q, g, x, k, H_M1)
print("Value for r = ", r, "\n", "Value for s = ", s, "\n")

verification_result, w, u1, u2, v = verification(p, q, g, y, H_M1, r, s)
print("Verification Result for M_M1 = ", verification_result,"\n", "Value for w:",
w , "\n", "Value for u1 = ", u1, "\n", "Value for u2 = ", u2, "\n", "Value for v =
", v, "\n")


# Example 2
print("Example 2", "\n")
p, q, h, x, k, H_M1, H_M2 = 103, 17, 13, 9, 6, 7, 3
g, y = gen_key_pair(p,q,h,x)
print("g = ", g, "\n","y = ", y)

r, s = sign( p, q, g, x, k, H_M1)
print("Value for r = ", r, "\n", "Value for s = ", s, "\n")

verification_result, w, u1, u2, v = verification(p, q, g, y, H_M1, r, s)
print("Verification Result for M_M1 = ", verification_result,"\n", "Value for w:",
w , "\n", "Value for u1 = ", u1, "\n", "Value for u2 = ", u2, "\n", "Value for v =
", v, "\n")


#Guess and Check Example

print("Guess and Check", "\n")
time_start = time.time()

p, q, h, x, k, H_M1, H_M2 = 127, 41, 37, 33, 30, 31, 27
g, y = gen_key_pair(p,q,h,x)
print("g = ", g, "\n","y = ", y)

r, s = sign( p, q, g, x, k, H_M1)
print("Value for r = ", r, "\n", "Value for s = ", s, "\n")

verification_result, w, u1, u2, v = verification(p, q, g, y, H_M1, r, s)
print("Verification Result for M_M1 = ", verification_result,"\n", "Value for w:",
w , "\n", "Value for u1 = ", u1, "\n", "Value for u2 = ", u2, "\n", "Value for v =
", v, "\n")


time_end = time.time()

total_time = time_end - time_start
print("Total Time: ", total_time)



#Values tried in order of variables listed:
# 546, 256, 23, 19, 7, 18
# 545, 433, 45, 27, 21, 9, 22
# 667, 434, 32, 77, 555, 23, 33
# 105, 19, 15, 11, 8, 9, 5 (Only guessed values that result in a true operation)
# 127, 41, 37, 33, 30, 31 , 27
```