21BDS0340

Abhinav Dinesh Srivatsa

Cryptography and Network Security Lab

Digital Assignment – V

**Question 1**

**Code**

bit\_size = 8 \* 8

# K values found online

K = [

0x428a2f98d728ae22, 0x7137449123ef65cd, 0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc,

0x3956c25bf348b538, 0x59f111f1b605d019, 0x923f82a4af194f9b, 0xab1c5ed5da6d8118,

0xd807aa98a3030242, 0x12835b0145706fbe, 0x243185be4ee4b28c, 0x550c7dc3d5ffb4e2,

0x72be5d74f27b896f, 0x80deb1fe3b1696b1, 0x9bdc06a725c71235, 0xc19bf174cf692694,

0xe49b69c19ef14ad2, 0xefbe4786384f25e3, 0x0fc19dc68b8cd5b5, 0x240ca1cc77ac9c65,

0x2de92c6f592b0275, 0x4a7484aa6ea6e483, 0x5cb0a9dcbd41fbd4, 0x76f988da831153b5,

0x983e5152ee66dfab, 0xa831c66d2db43210, 0xb00327c898fb213f, 0xbf597fc7beef0ee4,

0xc6e00bf33da88fc2, 0xd5a79147930aa725, 0x06ca6351e003826f, 0x142929670a0e6e70,

0x27b70a8546d22ffc, 0x2e1b21385c26c926, 0x4d2c6dfc5ac42aed, 0x53380d139d95b3df,

0x650a73548baf63de, 0x766a0abb3c77b2a8, 0x81c2c92e47edaee6, 0x92722c851482353b,

0xa2bfe8a14cf10364, 0xa81a664bbc423001, 0xc24b8b70d0f89791, 0xc76c51a30654be30,

0xd192e819d6ef5218, 0xd69906245565a910, 0xf40e35855771202a, 0x106aa07032bbd1b8,

0x19a4c116b8d2d0c8, 0x1e376c085141ab53, 0x2748774cdf8eeb99, 0x34b0bcb5e19b48a8,

0x391c0cb3c5c95a63, 0x4ed8aa4ae3418acb, 0x5b9cca4f7763e373, 0x682e6ff3d6b2b8a3,

0x748f82ee5defb2fc, 0x78a5636f43172f60, 0x84c87814a1f0ab72, 0x8cc702081a6439ec,

0x90befffa23631e28, 0xa4506cebde82bde9, 0xbef9a3f7b2c67915, 0xc67178f2e372532b,

0xca273eceea26619c, 0xd186b8c721c0c207, 0xeada7dd6cde0eb1e, 0xf57d4f7fee6ed178,

0x06f067aa72176fba, 0x0a637dc5a2c898a6, 0x113f9804bef90dae, 0x1b710b35131c471b,

0x28db77f523047d84, 0x32caab7b40c72493, 0x3c9ebe0a15c9bebc, 0x431d67c49c100d4c,

0x4cc5d4becb3e42b6, 0x597f299cfc657e2a, 0x5fcb6fab3ad6faec, 0x6c44198c4a475817

]

def ch(e, f, g):

return (e & f) | (~e & g)

def maj(a, b, c):

return (a & b) ^ (b & c) ^ (c & a)

def circular\_left\_shift(n, a, bits):

return ((n << a) | (n >> (bits - a))) & ((1 << bits) - 1)

def sigma\_a(a):

return circular\_left\_shift(a, 28, bit\_size) ^ circular\_left\_shift(a, 34, bit\_size) ^ circular\_left\_shift(a, 39, bit\_size)

def sigma\_e(e):

return circular\_left\_shift(e, 14, bit\_size) ^ circular\_left\_shift(e, 18, bit\_size) ^ circular\_left\_shift(e, 41, bit\_size)

def round(a, b, c, d, e, f, g, h, Wt, Kt):

T1 = (h + ch(e, f, g) + sigma\_e(e) + Wt + Kt) & (2 \*\* bit\_size - 1)

T2 = (sigma\_a(a) + maj(a, b, c)) & (2 \*\* bit\_size - 1)

return (T1 + T2) & (2 \*\* bit\_size - 1), a, b, c, (d + T1) & (2 \*\* bit\_size - 1), e, f, g

a = 0x6a09e667f3bcc908

b = 0xbb67ae8584caa73b

c = 0x3c6ef372fe94f82b

d = 0xa54ff53a5f1d36f1

e = 0x510e527fade682d1

f = 0x9b05688c2b3e6c1f

g = 0x1f83d9abfb41bd6b

h = 0x5be0cd19137e2179

word = 0x0123456789abcdef

# performing rounds

for i in range(1):

a, b, c, d, e, f, g, h = round(a, b, c, d, e, f, g, h, word, K[i])

print(f"a={format(a, '016x')}")

print(f"b={format(b, '016x')}")

print(f"c={format(c, '016x')}")

print(f"d={format(d, '016x')}")

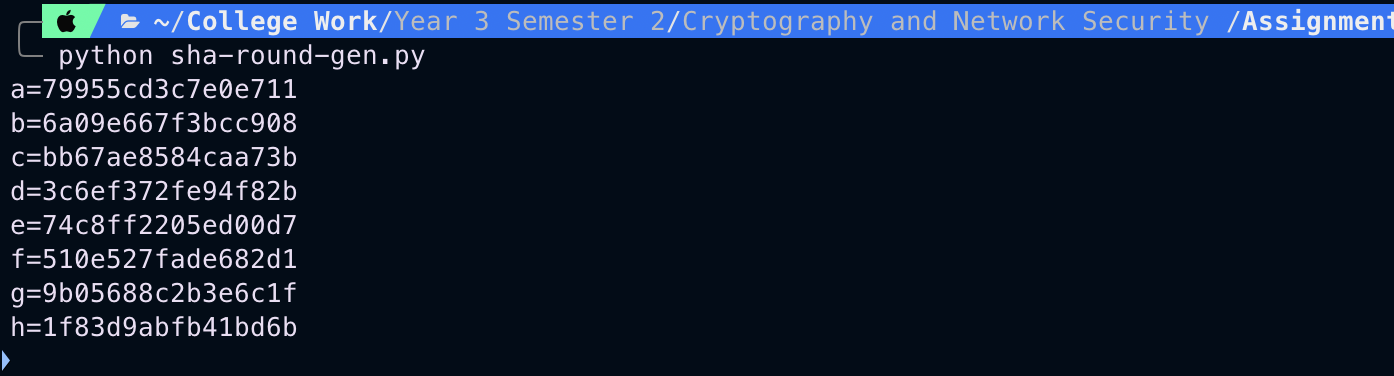
print(f"e={format(e, '016x')}")

print(f"f={format(f, '016x')}")

print(f"g={format(g, '016x')}")

print(f"h={format(h, '016x')}")

**Output**



**Question 2**

**Code**

import struct

def pad(message):

message\_bits = 8 \* len(message)

padded\_message = message.encode('utf-8') + b'\x80'

padding\_length = (896 - (message\_bits + 1)) % 1024

padded\_message += b'\x00' \* (padding\_length // 8)

print(message\_bits)

padded\_message += message\_bits.to\_bytes(16, byteorder='big')

return [struct.unpack('>Q', padded\_message[i:i+8])[0] for i in range(0, len(padded\_message), 8)]

def circular\_left\_shift(n, a, bits):

return ((n << a) | (n >> (bits - a))) & ((1 << bits) - 1)

def sigma\_0(x):

return circular\_left\_shift(x, 1, 64) ^ circular\_left\_shift(x, 8, 64) ^ (x << 7)

def sigma\_1(x):

return circular\_left\_shift(x, 19, 64) ^ circular\_left\_shift(x, 61, 64) ^ (x << 6)

def sequence(padded):

seq = [p for p in padded]

for i in range(16, 80):

seq.append((

sigma\_1(seq[i - 2]) + seq[i - 7] +

sigma\_0(seq[i - 15]) + seq[i - 16]

) & (2 \*\* 64 - 1))

return seq

word = input("enter word: ")

padded = pad(word)

input\_sequence = sequence(padded)

j = 0

for i in input\_sequence:

if j % 4 == 0 and j != 0:

print("")

print(f"{format(i, '016x')} ", end="")

j += 1

**Output**

