21BDS0340

Abhinav Dinesh Srivatsa

Cryptography and Network Security Lab

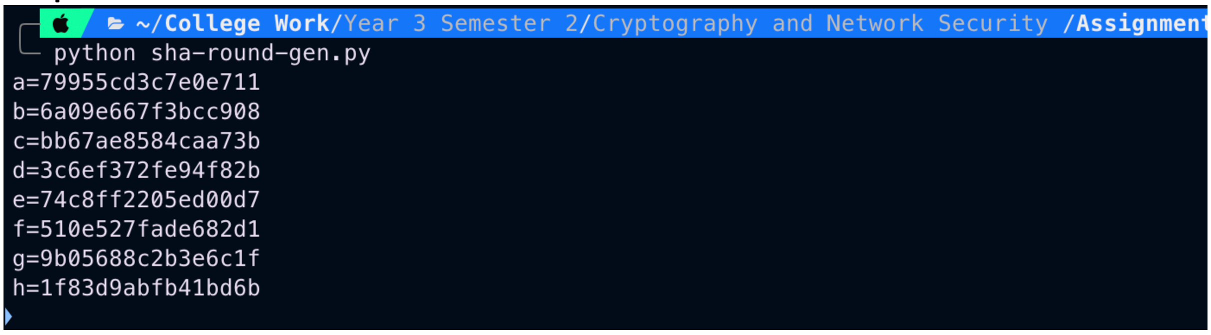
Digital Assignment – V

Question 1

Code

```python
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



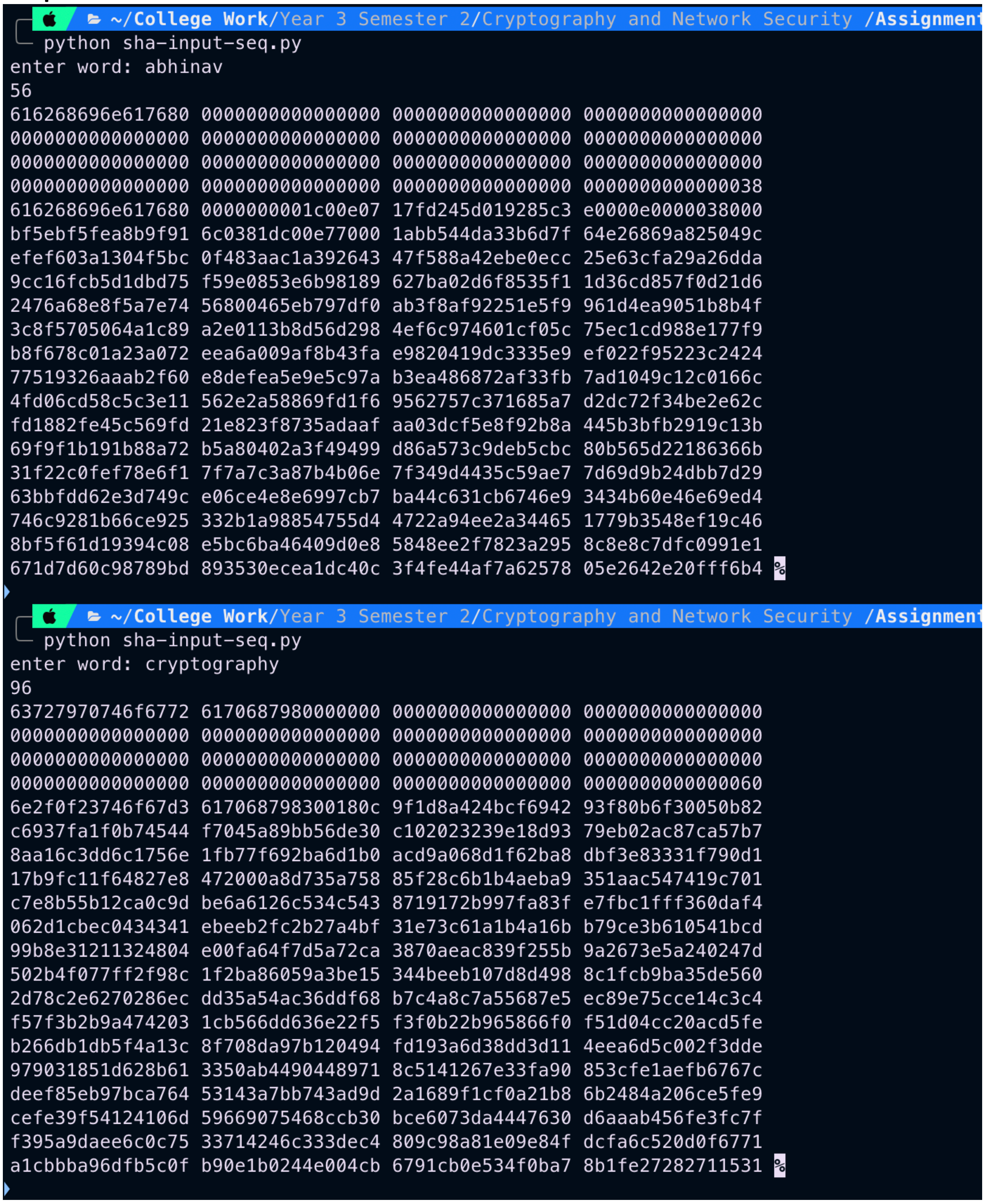
```
~/College Work/Year 3 Semester 2/Cryptography and Network Security /Assignment
 python sha-round-gen.py
a=79955cd3c7e0e711
b=6a09e667f3bcc908
c=bb67ae8584caa73b
d=3c6ef372fe94f82b
e=74c8ff2205ed00d7
f=510e527fade682d1
g=9b05688c2b3e6c1f
h=1f83d9abfb41bd6b
```

<u>Question 2</u>

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



```
~/College Work/Year 3 Semester 2/Cryptography and Network Security /Assignment
└─ python sha-input-seq.py
enter word: abhinav
56
616268696e617680 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000038
616268696e617680 0000000001c00e07 17fd245d019285c3 e0000e0000038000
bf5ebf5fea8b9f91 6c0381dc00e77000 1abb544da33b6d7f 64e26869a825049c
efef603a1304f5bc 0f483aac1a392643 47f588a42ebe0ecc 25e63cfa29a26dda
9cc16fcb5d1dbd75 f59e0853e6b98189 627ba02d6f8535f1 1d36cd857f0d21d6
2476a68e8f5a7e74 56800465eb797df0 ab3f8af92251e5f9 961d4ea9051b8b4f
3c8f5705064a1c89 a2e0113b8d56d298 4ef6c974601cf05c 75ec1cd988e177f9
b8f678c01a23a072 eea6a009af8b43fa e9820419dc3335e9 ef022f95223c2424
77519326aaab2f60 e8defea5e9e5c97a b3ea486872af33fb 7ad1049c12c0166c
4fd06cd58c5c3e11 562e2a58869fd1f6 9562757c371685a7 d2dc72f34be2e62c
fd1882fe45c569fd 21e823f8735adaaf aa03dcf5e8f92b8a 445b3bfb2919c13b
69f9f1b191b88a72 b5a80402a3f49499 d86a573c9deb5cbc 80b565d22186366b
31f22c0fef78e6f1 7f7a7c3a87b4b06e 7f349d4435c59ae7 7d69d9b24dbb7d29
63bbfdd62e3d749c e06ce4e8e6997cb7 ba44c631cb6746e9 3434b60e46e69ed4
746c9281b66ce925 332b1a98854755d4 4722a94ee2a34465 1779b3548ef19c46
8bf5f61d19394c08 e5bc6ba46409d0e8 5848ee2f7823a295 8c8e8c7dfc0991e1
671d7d60c98789bd 893530ecea1dc40c 3f4fe44af7a62578 05e2642e20fff6b4 %

~/College Work/Year 3 Semester 2/Cryptography and Network Security /Assignment
└─ python sha-input-seq.py
enter word: cryptography
96
63727970746f6772 6170687980000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000060
6e2f0f23746f67d3 617068798300180c 9f1d8a424bcf6942 93f80b6f30050b82
c6937fa1f0b74544 f7045a89bb56de30 c102023239e18d93 79eb02ac87ca57b7
8aa16c3dd6c1756e 1fb77f692ba6d1b0 acd9a068d1f62ba8 dbf3e83331f790d1
17b9fc11f64827e8 472000a8d735a758 85f28c6b1b4aeba9 351aac547419c701
c7e8b55b12ca0c9d be6a6126c534c543 8719172b997fa83f e7fbc1fff360daf4
062d1cbec0434341 ebeeb2fc2b27a4bf 31e73c61a1b4a16b b79ce3b610541bcd
99b8e31211324804 e00fa64f7d5a72ca 3870aeac839f255b 9a2673e5a240247d
502b4f077ff2f98c 1f2ba86059a3be15 344beeb107d8d498 8c1fcb9ba35de560
2d78c2e6270286ec dd35a54ac36ddf68 b7c4a8c7a55687e5 ec89e75cce14c3c4
f57f3b2b9a474203 1cb566dd636e22f5 f3f0b22b965866f0 f51d04cc20acd5fe
b266db1db5f4a13c 8f708da97b120494 fd193a6d38dd3d11 4eea6d5c002f3dde
979031851d628b61 3350ab4490448971 8c5141267e33fa90 853cfe1aefb6767c
deef85eb97bca764 53143a7bb743ad9d 2a1689f1cf0a21b8 6b2484a206ce5fe9
cefe39f54124106d 59669075468ccb30 bce6073da4447630 d6aaab456fe3fc7f
f395a9daee6c0c75 33714246c333dec4 809c98a81e09e84f dcfa6c520d0f6771
a1cbbba96dfb5c0f b90e1b0244e004cb 6791cb0e534f0ba7 8b1fe27282711531 %
```