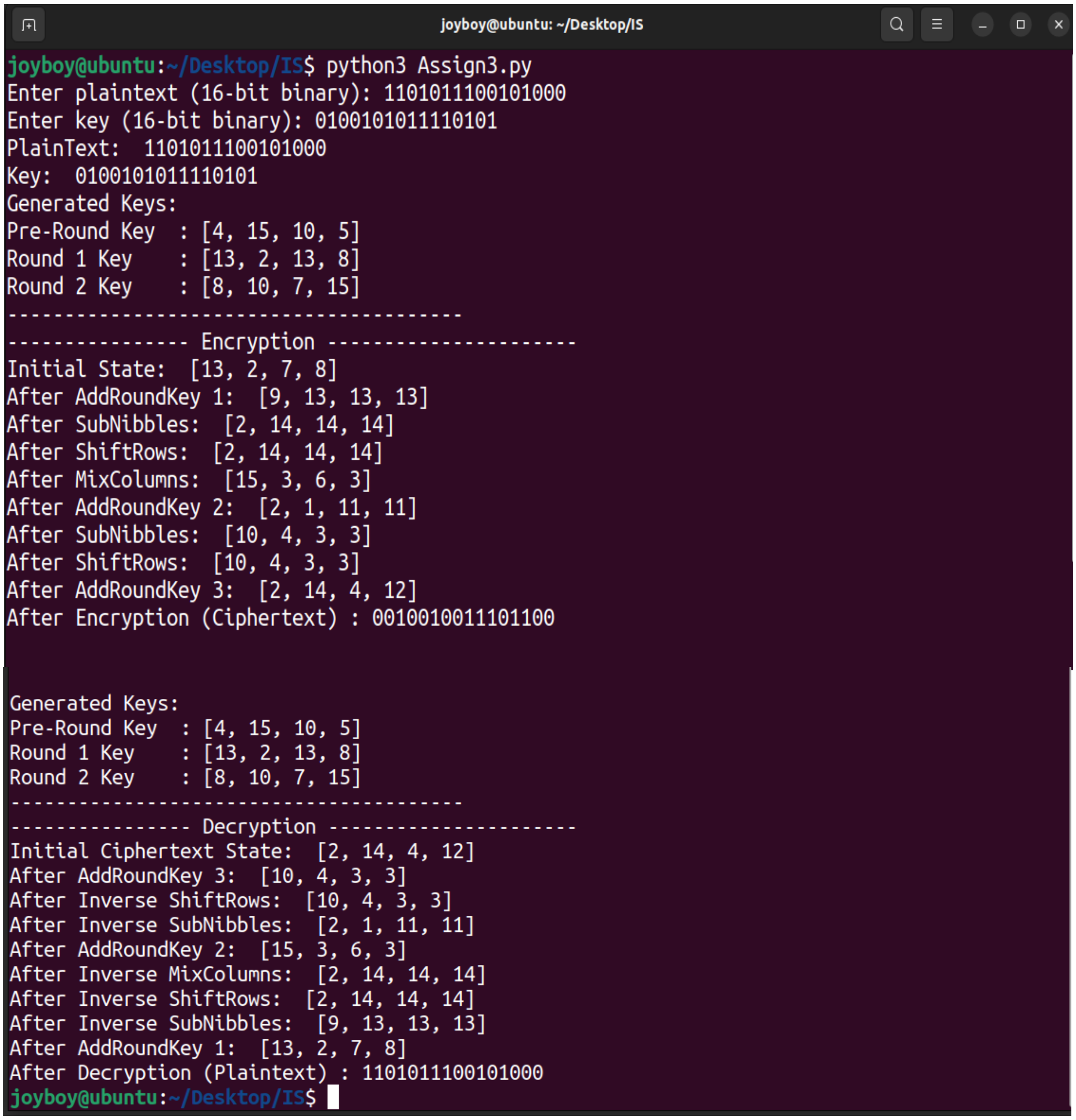
1. **input.py**

```
class SimplifiedAES:
    """
    Simplified AES (S-AES) Implementation
    """
    # S-Box
    sBox = [0x9, 0x4, 0xA, 0xB,
            0xD, 0x1, 0x8, 0x5,
            0x6, 0x2, 0x0, 0x3,
            0xC, 0xE, 0xF, 0x7]

    # Inverse S-Box
    sBoxI = [0xA, 0x5, 0x9, 0xB,
            0x1, 0x7, 0x8, 0xF,
            0x6, 0x0, 0x2, 0x3,
            0xC, 0x4, 0xD, 0xE]

    def __init__(self, key):
        self.pre_round_key, self.round1_key, self.round2_key =
self.key_expansion(key)
        print("Generated Keys:")
        print("Pre-Round Key  :", self.pre_round_key)
        print("Round 1 Key    :", self.round1_key)
        print("Round 2 Key    :",self.round2_key)
        print("-" * 40)

    def sub_word(self, word):
        return (self.sBox[(word >> 4)] << 4) + self.sBox[word &
0x0F]

    def rot_word(self, word):
        return ((word & 0x0F) << 4) + ((word & 0xF0) >> 4)

    def key_expansion(self, key):
        Rcon1, Rcon2 = 0x80, 0x30
        w = [None] * 6
        w[0], w[1] = (key & 0xFF00) >> 8, key & 0x00FF
        w[2] = w[0] ^ (self.sub_word(self.rot_word(w[1])) ^ Rcon1)
        w[3] = w[2] ^ w[1]
        w[4] = w[2] ^ (self.sub_word(self.rot_word(w[3])) ^ Rcon2)
        w[5] = w[4] ^ w[3]
        return [self.int_to_state((w[i] << 8) + w[i+1]) for i in
range(0, 6, 2)]

    def gf_mult(self, a, b):
        product = 0
        a, b = a & 0x0F, b & 0x0F
        while a and b:
            if b & 1:
                product ^= a
            a <<= 1
            if a & (1 << 4):
                a ^= 0b10011
            b >>= 1
        return product

    def int_to_state(self, n):
        return [n >> 12 & 0xF, (n >> 4) & 0xF, (n >> 8) & 0xF, n &
0xF]

    def state_to_int(self, m):
        return (m[0] << 12) + (m[2] << 8) + (m[1] << 4) + m[3]

    def add_round_key(self, s1, s2):
        return [i ^ j for i, j in zip(s1, s2)]

    def sub_nibbles(self, sbox, state):
        return [sbox[nibble] for nibble in state]

    def shift_rows(self, state):
        return [state[0], state[1], state[3], state[2]]

    def mix_columns(self, state):
        return [
            state[0] ^ self.gf_mult(4, state[2]),
            state[1] ^ self.gf_mult(4, state[3]),
            state[2] ^ self.gf_mult(4, state[0]),
            state[3] ^ self.gf_mult(4, state[1]),
        ]

    def inverse_mix_columns(self, state):
        return [
            self.gf_mult(9, state[0]) ^ self.gf_mult(2, state[2]),
            self.gf_mult(9, state[1]) ^ self.gf_mult(2, state[3]),
            self.gf_mult(9, state[2]) ^ self.gf_mult(2, state[0]),
            self.gf_mult(9, state[3]) ^ self.gf_mult(2, state[1]),
        ]

    def encrypt(self, plaintext):
        print("---------------- Encryption ----------------------")
        state = self.int_to_state(plaintext)
        print("Initial State: ", state)

        state = self.add_round_key(self.pre_round_key, state)
        print("After AddRoundKey 1: ", state)

        state = self.sub_nibbles(self.sBox, state)
        print("After SubNibbles: ", state)

        state = self.shift_rows(state)
        print("After ShiftRows: ", state)

        state = self.mix_columns(state)
        print("After MixColumns: ", state)

        state = self.add_round_key(self.round1_key, state)
        print("After AddRoundKey 2: ", state)

        state = self.sub_nibbles(self.sBox, state)
        print("After SubNibbles: ", state)

        state = self.shift_rows(state)
        print("After ShiftRows: ", state)

        state = self.add_round_key(self.round2_key, state)
        print("After AddRoundKey 3: ", state)

        return self.state_to_int(state)

    def decrypt(self, ciphertext):
        print("---------------- Decryption ----------------------")
        state = self.int_to_state(ciphertext)
        print("Initial Ciphertext State: ", state)

        state = self.add_round_key(self.round2_key, state)
        print("After AddRoundKey 3: ", state)

        state = self.shift_rows(state)
        print("After Inverse ShiftRows: ", state)

        state = self.sub_nibbles(self.sBoxI, state)
        print("After Inverse SubNibbles: ", state)

        state = self.add_round_key(self.round1_key, state)
        print("After AddRoundKey 2: ", state)

        state = self.inverse_mix_columns(state)
        print("After Inverse MixColumns: ", state)

        state = self.shift_rows(state)
        print("After Inverse ShiftRows: ", state)

        state = self.sub_nibbles(self.sBoxI, state)
        print("After Inverse SubNibbles: ", state)

        state = self.add_round_key(self.pre_round_key, state)
        print("After AddRoundKey 1: ", state)

        return self.state_to_int(state)


plaintext = int(input("Enter plaintext (16-bit binary): "), 2)
key = int(input("Enter key (16-bit binary): "), 2)


print("PlainText: ", format(plaintext,'016b'))
print("Key: ", format(key,'016b'))

ciphertext = SimplifiedAES(key).encrypt(plaintext)
print("After Encryption (Ciphertext) :", format(ciphertext,
'016b'))
print("\n\n\n")
decrypted_text = SimplifiedAES(key).decrypt(ciphertext)
print("After Decryption (Plaintext) :", format(decrypted_text,
'016b'))
```

2. Output



```
joyboy@ubuntu:~/Desktop/IS$ python3 Assign3.py
Enter plaintext (16-bit binary): 1101011100101000
Enter key (16-bit binary): 0100101011110101
PlainText:  1101011100101000
Key:  0100101011110101
Generated Keys:
Pre-Round Key  : [4, 15, 10, 5]
Round 1 Key    : [13, 2, 13, 8]
Round 2 Key    : [8, 10, 7, 15]
----------------------------------------
---------------- Encryption ----------------------
Initial State:  [13, 2, 7, 8]
After AddRoundKey 1:  [9, 13, 13, 13]
After SubNibbles:  [2, 14, 14, 14]
After ShiftRows:  [2, 14, 14, 14]
After MixColumns:  [15, 3, 6, 3]
After AddRoundKey 2:  [2, 1, 11, 11]
After SubNibbles:  [10, 4, 3, 3]
After ShiftRows:  [10, 4, 3, 3]
After AddRoundKey 3:  [2, 14, 4, 12]
After Encryption (Ciphertext) : 0010010011101100


Generated Keys:
Pre-Round Key  : [4, 15, 10, 5]
Round 1 Key    : [13, 2, 13, 8]
Round 2 Key    : [8, 10, 7, 15]
----------------------------------------
---------------- Decryption ----------------------
Initial Ciphertext State:  [2, 14, 4, 12]
After AddRoundKey 3:  [10, 4, 3, 3]
After Inverse ShiftRows:  [10, 4, 3, 3]
After Inverse SubNibbles:  [2, 1, 11, 11]
After AddRoundKey 2:  [15, 3, 6, 3]
After Inverse MixColumns:  [2, 14, 14, 14]
After Inverse ShiftRows:  [2, 14, 14, 14]
After Inverse SubNibbles:  [9, 13, 13, 13]
After AddRoundKey 1:  [13, 2, 7, 8]
After Decryption (Plaintext) : 1101011100101000
joyboy@ubuntu:~/Desktop/IS$
```