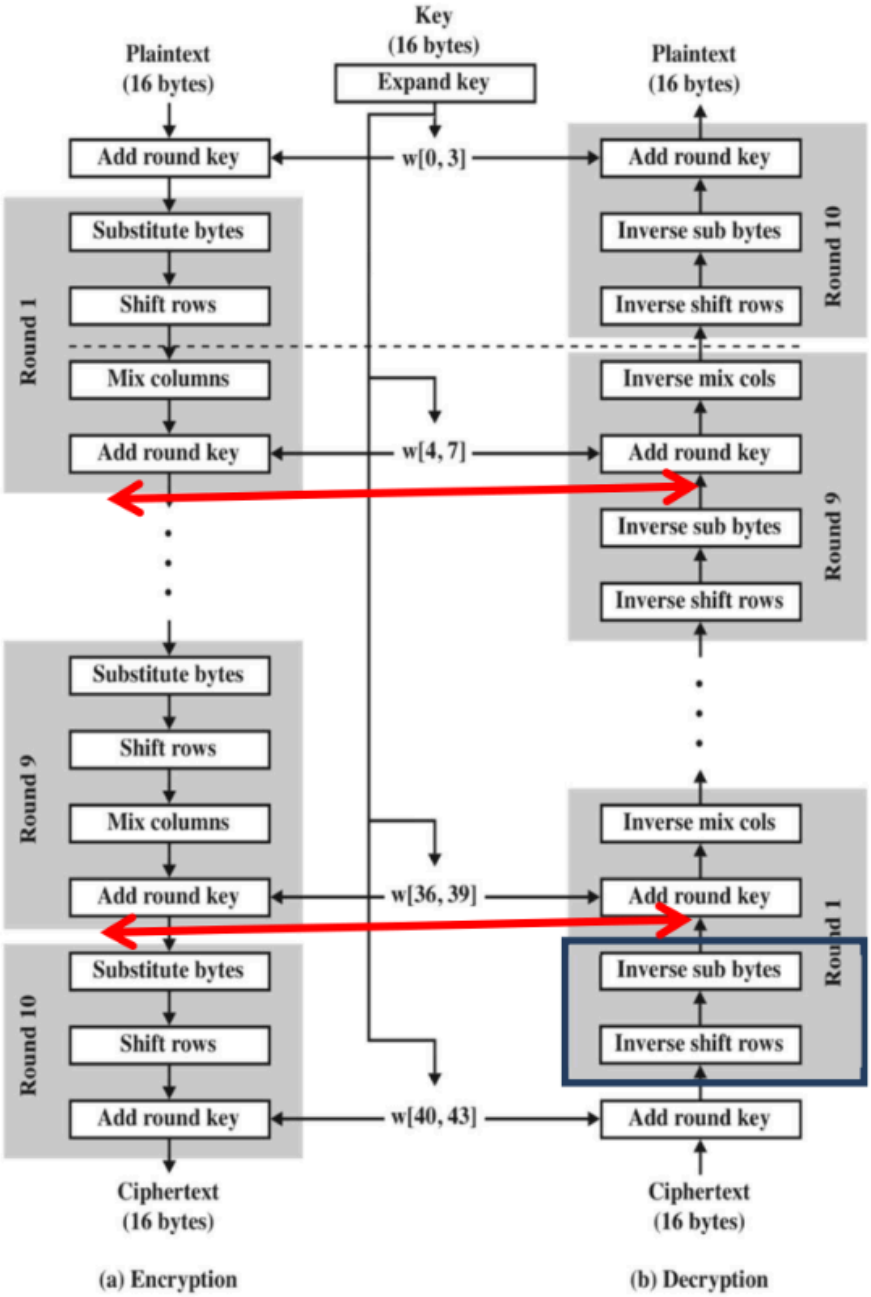
CSE350/550: Network Security : Programming assignment no. 2
Abhishek 2020014
Rishabh 2020399

Project 1:



Input :

```python
if __name__ == '__main__':

    KEYS = ["Network is mythq", "Rishabh is great", "Hello NetworkSec"]
    TEXTS = ["Securityisamythy", "Abhishekishonest", "network security"]

    for i in range(3):
        print("Sample input and Output", i+1)
        main_key = string_to_hex(KEYS[i])
        main_text = string_to_hex(TEXTS[i])
        all_round_keys = key_expansion(main_key)
        ciphertext, encryption_1_output, encryption_9_output = AES_encrypt_function(main_text, all_round_keys)
        # print(ciphertext)
        ciphertext_str = hex_to_string(ciphertext)
        decrypt_text, decryption_1_output, decryption_9_output = AES_decrypt_function(ciphertext, all_round_keys)
        decrypt_text_str = hex_to_string(decrypt_text)
        print("Case (b): Verified that the output of the 1st encryption round is the same as the output of the 9th decryption round as you can see above")
        print("Case (c): Verified that the output of the 9th encryption round is the same as the output of the 1st decryption round as you can see above")
        print()
        #printing the key used
        print("Key : '{}'".format(KEYS[i]))
        # Print the plain text
        print("Plain Text : '{}'".format(TEXTS[i]))
        # Print the encrypted text
        print("Encrypted Text :", ciphertext_str)
        # Print the Deciphered text
        print("Deciphered Text : '{}'".format(decrypt_text_str))

        # Check for case (a): Verify that AES encryption and decryption is working properly
        if TEXTS[i] == decrypt_text_str:
            print("Case (a): Verified that AES encryption and decryption is working properly. Plaintext = Deciphered text")
        else:
            print("Case (a): AES Encryption/Decryption failed")

        #Check for case (b): Verify that the output of 1st encryption round is the same as output of the 9th decryption round
        #already verified above as you can see for all three cases
        #Check for case (c): Verify that the output of 9th encryption round is the same as output of the 1st decryption round
        #already verified above as you can see for all three cases
        print()
```

KEYS: This is a list containing strings representing the encryption keys. Each string serves as the key for encrypting and decrypting corresponding messages.

TEXTS: This is a list containing strings representing the plaintext messages that need to be encrypted and decrypted. Each string corresponds to a plaintext message that will be encrypted using the keys from the KEYS list.

These inputs are used to demonstrate the functionality of the AES encryption and decryption algorithms implemented in the code. Each key in the KEYS list is used to encrypt the corresponding plaintext message from the TEXTS list. Then, decryption is performed to retrieve the original plaintext messages.

Output:

```
[Running] python -u "e:\8th_sem\NSc\Assignments\NSc_Assignment_2_2020014_2020399\AES.py"
Sample input and Output 1
1 Round of Encryption:  [['d7', '50', '83', '62'], ['6f', 'e2', '44', 'c8'], ['91', '24', '21', 'ca'], ['7a', 'f8', '37', '3d']]
9 Round of Encryption:  [['97', '05', '99', '52'], ['4b', '70', '33', 'f8'], ['1a', '36', '6a', 'de'], ['49', '72', '6b', '1f']]
1 Round of Decryption:  [['97', '05', '99', '52'], ['4b', '70', '33', 'f8'], ['1a', '36', '6a', 'de'], ['49', '72', '6b', '1f']]
9 Round of Decryption:  [['d7', '50', '83', '62'], ['6f', 'e2', '44', 'c8'], ['91', '24', '21', 'ca'], ['7a', 'f8', '37', '3d']]
Case (b): Verified that the output of the 1st encryption round is the same as the output of the 9th decryption round as you can see above
Case (c): Verified that the output of the 9th encryption round is the same as the output of the 1st decryption round as you can see above

Key : 'Network is mythq'
Plain Text : 'Securityisamythy'
Encrypted Text : '□Ü¢ú¯
z<Åÿ□9!AÚ
Deciphered Text : 'Securityisamythy'
Case (a): Verified that AES encryption and decryption is working properly. Plaintext = Deciphered text

Sample input and Output 2
1 Round of Encryption:  [['7c', 'e6', '69', 'dd'], ['a0', 'd7', '6b', '6a'], ['fa', '2e', '22', '9a'], ['5c', '1a', '7b', '38']]
9 Round of Encryption:  [['8b', '43', '16', '93'], ['df', '57', '33', 'ac'], ['49', '14', 'd1', '0c'], ['e0', '43', '2f', 'e8']]
1 Round of Decryption:  [['8b', '43', '16', '93'], ['df', '57', '33', 'ac'], ['49', '14', 'd1', '0c'], ['e0', '43', '2f', 'e8']]
9 Round of Decryption:  [['7c', 'e6', '69', 'dd'], ['a0', 'd7', '6b', '6a'], ['fa', '2e', '22', '9a'], ['5c', '1a', '7b', '38']]
Case (b): Verified that the output of the 1st encryption round is the same as the output of the 9th decryption round as you can see above
Case (c): Verified that the output of the 9th encryption round is the same as the output of the 1st decryption round as you can see above

Key : 'Rishabh is great'
Plain Text : 'Abhishekishonest'
Encrypted Text : »÷B□¡h3X(0□_x<\
Deciphered Text : 'Abhishekishonest'
Case (a): Verified that AES encryption and decryption is working properly. Plaintext = Deciphered text

Sample input and Output 3
1 Round of Encryption:  [['0d', '60', '76', '38'], ['64', 'd0', '23', 'ba'], ['7a', '86', '00', '4e'], ['b0', 'dd', 'a1', 'cd']]
9 Round of Encryption:  [['bf', 'ab', 'fd', '32'], ['75', '1e', '00', '70'], ['ef', '08', '1c', '3c'], ['cf', 'c8', 'ea', '6d']]
1 Round of Decryption:  [['bf', 'ab', 'fd', '32'], ['75', '1e', '00', '70'], ['ef', '08', '1c', '3c'], ['cf', 'c8', 'ea', '6d']]
9 Round of Decryption:  [['0d', '60', '76', '38'], ['64', 'd0', '23', 'ba'], ['7a', '86', '00', '4e'], ['b0', 'dd', 'a1', 'cd']]
Case (b): Verified that the output of the 1st encryption round is the same as the output of the 9th decryption round as you can see above
Case (c): Verified that the output of the 9th encryption round is the same as the output of the 1st decryption round as you can see above

Key : 'Hello NetworkSec'
Plain Text : 'network security'
Encrypted Text : Uá<□¡□CÇ
□7ú□$□□
Deciphered Text : 'network security'
Case (a): Verified that AES encryption and decryption is working properly. Plaintext = Deciphered text
```

a. Verify that the ciphertext when decrypted will yield the original plaintext.
-> As is visible from the screenshot above, the deciphered text matches the pliantext for all the three inputs.

b. Verify that output of the 1st encryption round is same as output of the 15th decryption round.
c. Verify that the output of 9th encryption round is same as output of the 1st decryption round
-> As is visible from the screenshot above, both the conditions have been satisfied as per the output seen.

Funtions:

1. generate_2D_matrix(hex_string)
2. string_to_hex(input_string)
3. hex_to_string(hex_array)
4. galois_multiplication(a, b)
5. mix_column_both(column, bool)
6. mix_columns(matrix, bool)
7. left_and_right_shift(arr, bool)
8. left_shift(arr) and right_shift(arr)
9. shift_row(matrix) and inverse_shift_row(matrix)
10. key_expansion(main_key)
11. add_round_key(text_x, key_x)
12. substitute_bytes(matrix, bool)
13. AES_encrypt_function(text_hex, all_round_keys)
14. AES_decrypt_function(cipher_text, all_round_keys).

Explanations:

1. generate_2D_matrix(hex_string)

- A hex string is the input for this function.
- The hex string is transformed into a list of lists, with each inner list standing in for an individual row of the matrix, to create the 4x4 matrix.
- Finally, zip(*matrix_4x4) is used to return the transposed matrix, effectively converting row-major ordering to column major ordering.

2. string_to_hex(input_string)

- A string is the input for this function.
- Using the ord(char) and hex() functions, it transforms each character in the input string into the corresponding representation of a HEXAdecimal number.
- Hexadecimal representations are returned after being stored in a list.

3. hex_to_string(hex_array)

- A list of hexadecimal strings is entered into this function.
- Using int(hex_char, 16) to obtain the ASCII value and chr(), it iterates over each hemidecimal string before returning it to its corresponding character.
- The original string is created by adding the characters together, and it is then returned.

4. galois_multiplication(a, b)(http://blog.simulacrum.me/2019/01/aes-galois/)

- The AES encryption algorithm's Galois Field(GF) multiplication of the numbers a and b is carried out by this function.
- It performs conditional XOR operations on a based on the bits of b while it iterates over each bit.
- The GF(2^ 8) field property is maintained by performing an XOR with 0x1b if the most important bit of a is set after left-shifting.
- The multiplication's outcome is given back.

5. mix_column_both(column, bool)

- This function applies the MixColumns operation to a single column of the AES state matrix.
- Depending on whether the matrix is being used for encryption or decryption, it multiplies a column as input.
- The input column has the updated result in place.

6. mix_columns(matrix, bool)

- The entire state matrix is covered by this function's MixColumns operation.
- Each column of the matrix is iterated over, and each column is modified with the mix_column_both() function.
- The input matrix is updated with the outcome.

7. left_and_right_shift(arr, bool)

- Depending on the boolean value, this function shifts an array's elements left or right circularly.
- Function performs left shift if bool is True; otherwise, it shifts to the right.
- For the right shift the last element of the array is moved to the first position and similarly right shift

8. left_shift(arr) and right_shift(arr)

- These functions are utility functions for performing left and right circular shifts.
- With the appropriate boolean value, they call left_and_right_shift().

9. shift_row(matrix) and inverse_shift_row(matrix)

- The ShiftRows operation for encryption and decryption is carried out by these functions.
- The state matrix's rows are moved in accordance with the requirements of the AES algorithm.
- Rows are moved to the left for encryption and the right for decryption.

10. key_expansion(main_key)

- Using the KeyExpansion algorithm in AES, this function generates all round keys from the primary encryption key.
- The main key is first transformed into a 4x4 integer matrix.
- Each word in the expanded key is then iterated over, being derived from the one before it.
- For some iterations, round constants are used, and each word performs operations similar to rotation, substitution, or XOR on the one before it.
- Each inner list represents a 4x4 round key matrix, and the expanded key is returned as an array of lists.

11. add_round_key(text_x, key_x)

- Input: text_x - a 4x4 matrix representing the current state of the text, key_x - a 4x4 matrix representing the current round key.
- Output: Returns a new 4x4 matrix obtained by performing an XOR operation between each element of text_x and key_x.
- Description: The AddRoundKey operation is carried out by this function, which XORs each state matrix byte to the round key matrix. During encryption and decryption, this operation adds an additional layer of encryption.

12. substitute_bytes(matrix, bool)

- Input: matrix - a 4x4 matrix representing the state matrix, bool - a boolean indicating whether to use the S-Box for encryption (True) or the inverse S-Box for decryption (False).
- Output: Modifies the matrix in-place.
- Description: During the encryption process, the S-Box lookup table's corresponding byte is used in place of each member of the state matrix. The inverse S-Box is substituted during decryption. The data becomes more confusing as a result of this operation, which also increases the security of the encryption process.

13. AES_encrypt_function(text_hex, all_round_keys)

- Input: text_hex - the plaintext represented as a 1D array of hexadecimal strings, all_round_keys - a list containing all round keys.
- Output: Returns the ciphertext represented as a 1D array of hexadecimal strings.
- Description: Using the provided round keys, this function performs AES encryption on the input plaintext. As it moves through the encryption rounds, it sequentially performs AddRoundKey, SubBytes, ShiftRows, and MixColumns. After all rounds have been completed, the final ciphertext is obtained.

14. AES_decrypt_function(cipher_text, all_round_keys)

- Input: cipher_text - the ciphertext represented as a 1D array of hexadecimal strings, all_round_keys - a list containing all round keys.
- Output: Returns the plaintext represented as a 1D array of hexadecimal strings.
- Description: Using the provided round keys, this function decrypts the input ciphertext. It performs inverse operations on SubBytes, ShiftRows, MixColumn, and AddRoundKey in reverse order as it goes through the decryption rounds. After all rounds have been completed, the final plaintext is obtained.

References:

1. GFG
2. Class slides
3. https://medium.com/wearesinch/building-aes-128-from-the-ground-up-with-python-8122af44ebf9
4. http://blog.simulacrum.me/2019/01/aes-galois/