OUROBOROS SCHEME

NIS COURSE PROJECT 1

Prepared By:

Hamza Ahmed Khan CT - 20035

Course Title: CT- 486

Course Instructor: Dr. Muhammad Mubashir Khan

Q) Design your own encryption/decryption scheme. Implement its code in python, explain andperform some test cases with random plaintexts.

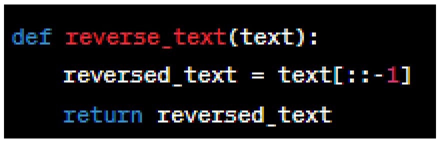
Overview:

The encryption-decryption scheme named as *"Ouroboros Scheme"* which means as a never-ending cycle and begins where it ends and ends where it begins. In technical terms this scheme utilizes a reversed ASCII transformation method to secure plaintext. The process involves reversing the input text, inserting a secret key (numerical value), adjusting ASCII codes, and creating a mapping between the modified ASCII codes and the corresponding characters.

1. Design:

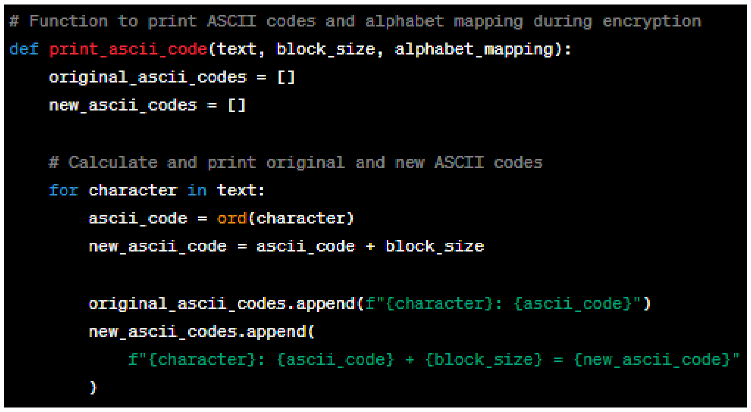
1.1 Encryption:

❖ Reverse Text: The plaintext is reversed to enhance the security of the encryption.



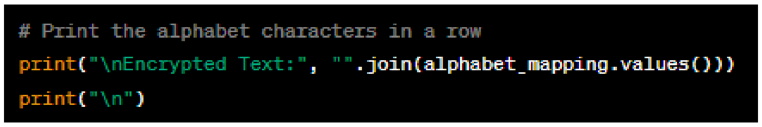
```python
def reverse_text(text):
    reversed_text = text[::-1]
    return reversed_text
```

❖ ASCII Code Transformation: The ASCII codes of the reversed characters are adjusted by adding a specified block size.



```python
# Function to print ASCII codes and alphabet mapping during encryption
def print_ascii_code(text, block_size, alphabet_mapping):
    original_ascii_codes = []
    new_ascii_codes = []

    # Calculate and print original and new ASCII codes
    for character in text:
        ascii_code = ord(character)
        new_ascii_code = ascii_code + block_size

        original_ascii_codes.append(f"{character}: {ascii_code}")
        new_ascii_codes.append(
            f"{character}: {ascii_code} + {block_size} = {new_ascii_code}"
        )
```

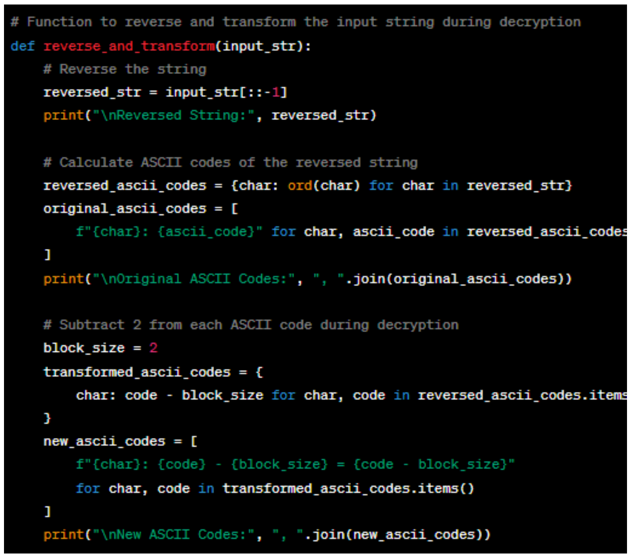
❖ Alphabet Mapping: A mapping is created between the modified ASCII codes and the corresponding characters. This mapping ensures uniqueness and handles repeating letters.



```python
# Print the alphabet characters in a row
print("\nEncrypted Text:", "".join(alphabet_mapping.values()))
print("\n")
```

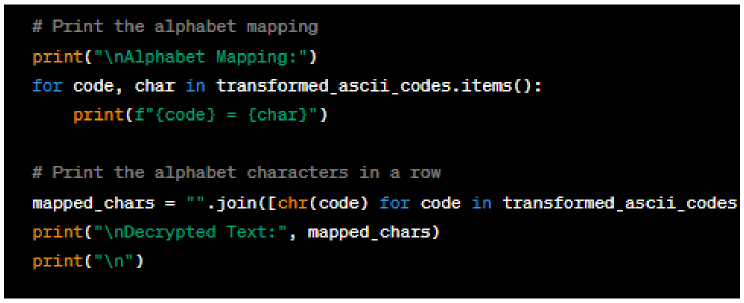
1.2 Decryption:

❖ Reverse and Transform: The reversed text is obtained, and the original ASCII codes are restored by subtracting the block size.



```python
# Function to reverse and transform the input string during decryption
def reverse_and_transform(input_str):
    # Reverse the string
    reversed_str = input_str[::-1]
    print("\nReversed String:", reversed_str)

    # Calculate ASCII codes of the reversed string
    reversed_ascii_codes = {char: ord(char) for char in reversed_str}
    original_ascii_codes = [
        f"{char}: {ascii_code}" for char, ascii_code in reversed_ascii_codes
    ]
    print("\nOriginal ASCII Codes:", ", ".join(original_ascii_codes))

    # Subtract 2 from each ASCII code during decryption
    block_size = 2
    transformed_ascii_codes = {
        char: code - block_size for char, code in reversed_ascii_codes.items
    }
    new_ascii_codes = [
        f"{char}: {code} - {block_size} = {code - block_size}"
        for char, code in transformed_ascii_codes.items()
    ]
    print("\nNew ASCII Codes:", ", ".join(new_ascii_codes))
```

❖ Alphabet Mapping: The decrypted ASCII codes are mapped back to their corresponding characters.



```
# Print the alphabet mapping
print("\nAlphabet Mapping:")
for code, char in transformed_ascii_codes.items():
    print(f"{code} = {char}")

# Print the alphabet characters in a row
mapped_chars = "".join([chr(code) for code in transformed_ascii_codes
print("\nDecrypted Text:", mapped_chars)
print("\n")
```

2. Explanation:

2.1 Encryption Process:

The reversal of the plaintext adds an additional layer of complexity. The ASCII code transformation introduces variability, and the mapping ensures uniqueness, preventing information loss during encryption. This scheme is designed to handle repeating letters gracefully.

Program asks the user if he wants to perform Encryption or Decryption.

User Press 1 for Encryption:

- The user provides an input Plaintext.
- User enters the secret key (numerical value)
- The input Plaintext is then reversed using the reverse_text function.
- For each character in the reversed text:
    - The original ASCII code of the character is determined (e.g., k: 107).
    - The block size (secret key) is added to the original ASCII code to obtain the new ASCII code.
- If there are repeating letters, an index is appended to the new ASCII code to ensure uniqueness.
- The new ASCII codes are mapped to corresponding alphabet characters

- The encrypted text is formed by concatenating the mapped characters.

Example:

Plaintext: hulk

Secret Key: 2

Reversed String**:** kluh

Original ASCII Codes: k: 107, l: 108, u: 117, h: 104

New ASCII Codes: k: 107 + 2 = 109, l: 108 + 2 = 110, u: 117 + 2 = 119, h: 104 + 2 = 106

Alphabet Mapping:

109 = m

110 = n

119 = w

106 = j

Encrypted Text: mnwj

2.2 Decryption Process:

The decryption process involves reversing the text and restoring the original ASCII codes. The alphabet mapping facilitates the reconstruction of the original message.

Program asks the user if he wants to perform Encryption or Decryption.

User Press 2 for Decryption:

- The user provides an input Ciphertext.
- User enters the secret key (numerical value)

  - Note: the same key used for encryption should be used

- The input Ciphertext is reversed using the reverse_and_transform function.
- For each character in the reversed text:
  - The original ASCII code of the character is determined (e.g., j: 106).
  - The original ASCII code of each character is subtracted by the block size to obtain the new ASCII code.
- The new ASCII codes are mapped back to the corresponding alphabet characters
- The decrypted text is formed by concatenating the mapped characters.

Example:

Ciphertext: mnwj

Secret Key: 2

Reversed String**:** jwnm

Original ASCII Codes: j: 106, w: 119, n: 110, m: 109

New ASCII Codes: j: 104 - 2 = 102, w: 117 - 2 = 115, n: 108 - 2 = 106, m: 107 - 2 = 105

Alphabet Mapping:

j = 104

w = 117

n = 108

m = 107

Decrypted Text: hulk

3. Implementation (Code) :

3.1 Colab Link:

https://colab.research.google.com/drive/1I45JewGcfkV7VV08O2wdFI0PJ
Xl0bt3s?usp=sharing

4. Analysis:

4.1 Strengths:

Reversal Technique: Enhances security by making patterns less predictable.

Variable Block Size: The scheme allows adjusting the block size for different security levels.

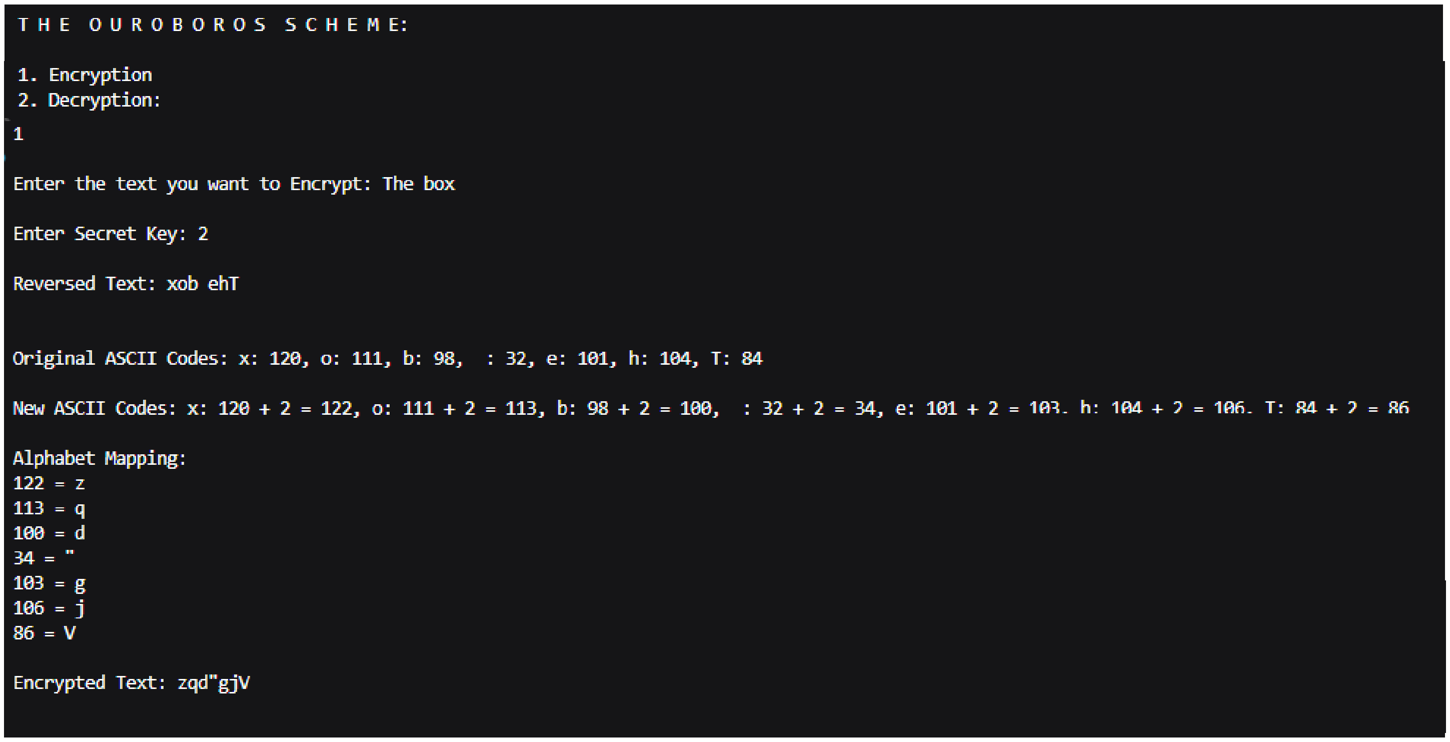
4.2 Weaknesses:

Block Size Dependency: Security relies on the secrecy of the block size; if known, it could compromise the encryption.

Mapping Complexity: For long texts, the alphabet mapping could become computationally expensive.

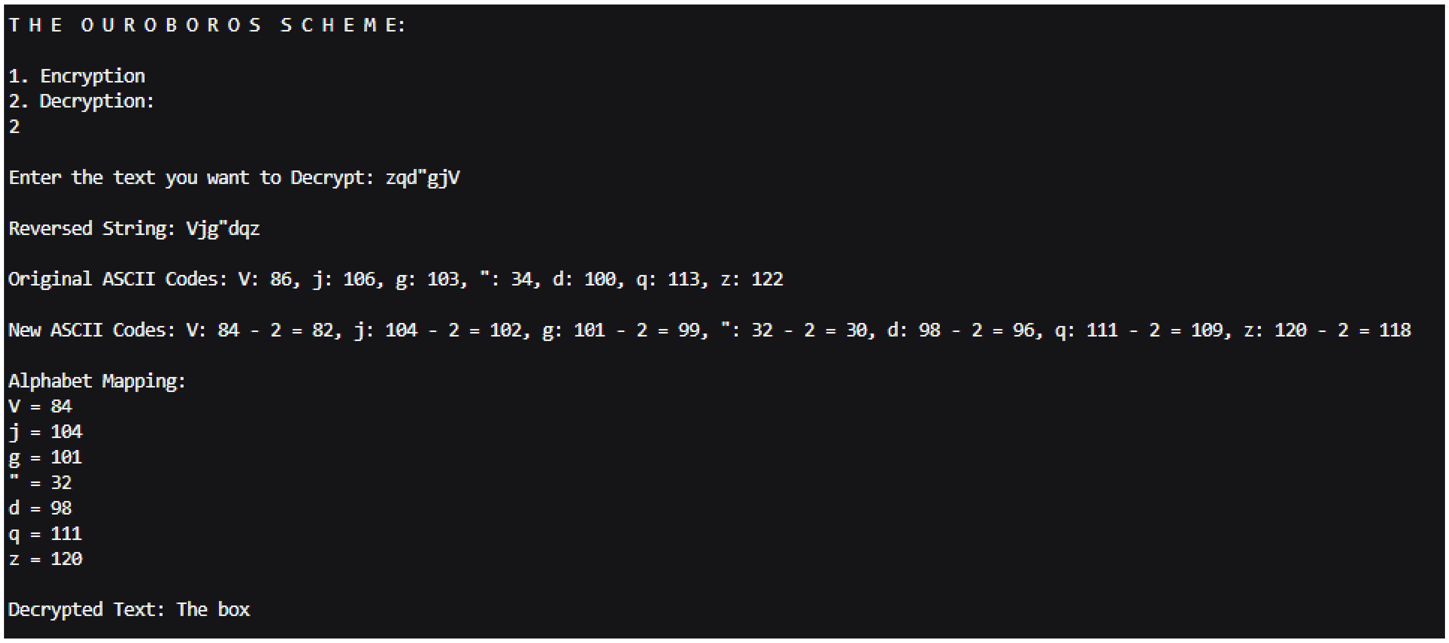
5. Samples of Test Cases:

5.1 Test case for Encryption: (String)



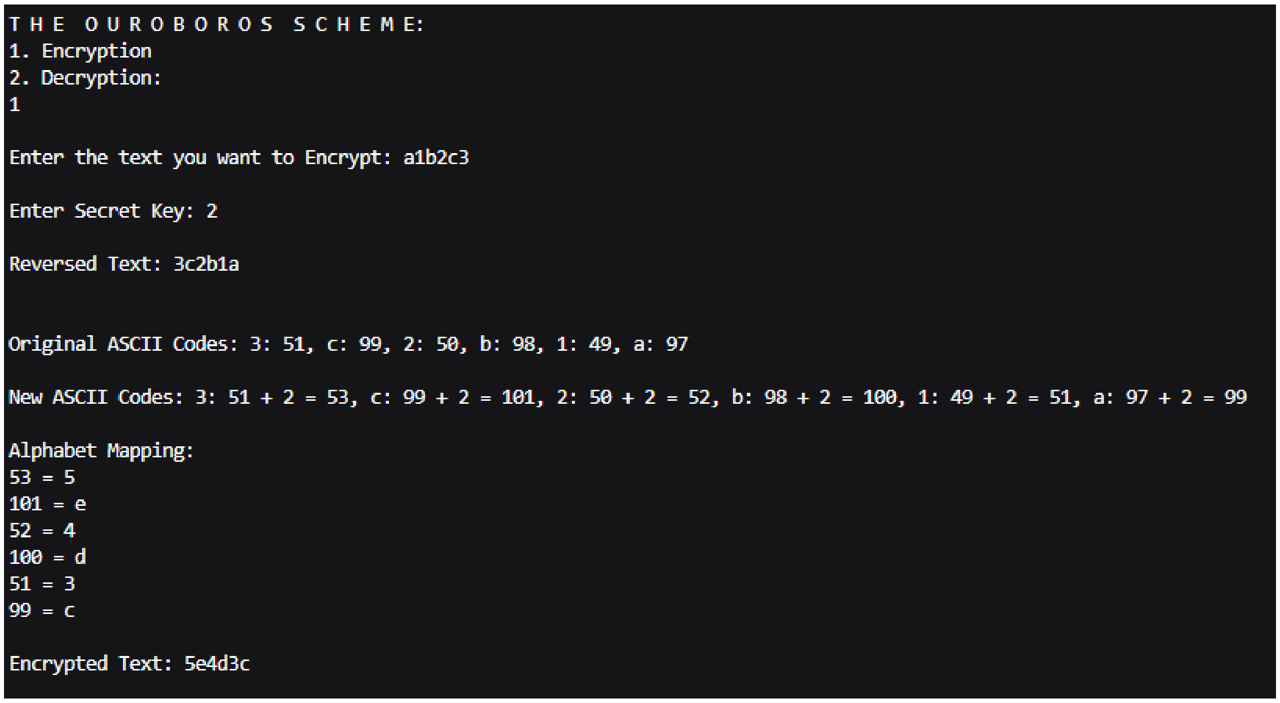
```
T H E  O U R O B O R O S  S C H E M E:

1. Encryption
2. Decryption:
1

Enter the text you want to Encrypt: The box

Enter Secret Key: 2

Reversed Text: xob ehT


Original ASCII Codes: x: 120, o: 111, b: 98,  : 32, e: 101, h: 104, T: 84

New ASCII Codes: x: 120 + 2 = 122, o: 111 + 2 = 113, b: 98 + 2 = 100,  : 32 + 2 = 34, e: 101 + 2 = 103, h: 104 + 2 = 106, T: 84 + 2 = 86

Alphabet Mapping:
122 = z
113 = q
100 = d
34 = "
103 = g
106 = j
86 = V

Encrypted Text: zqd"gjV
```

5.2 Test case for Decryption: (String)



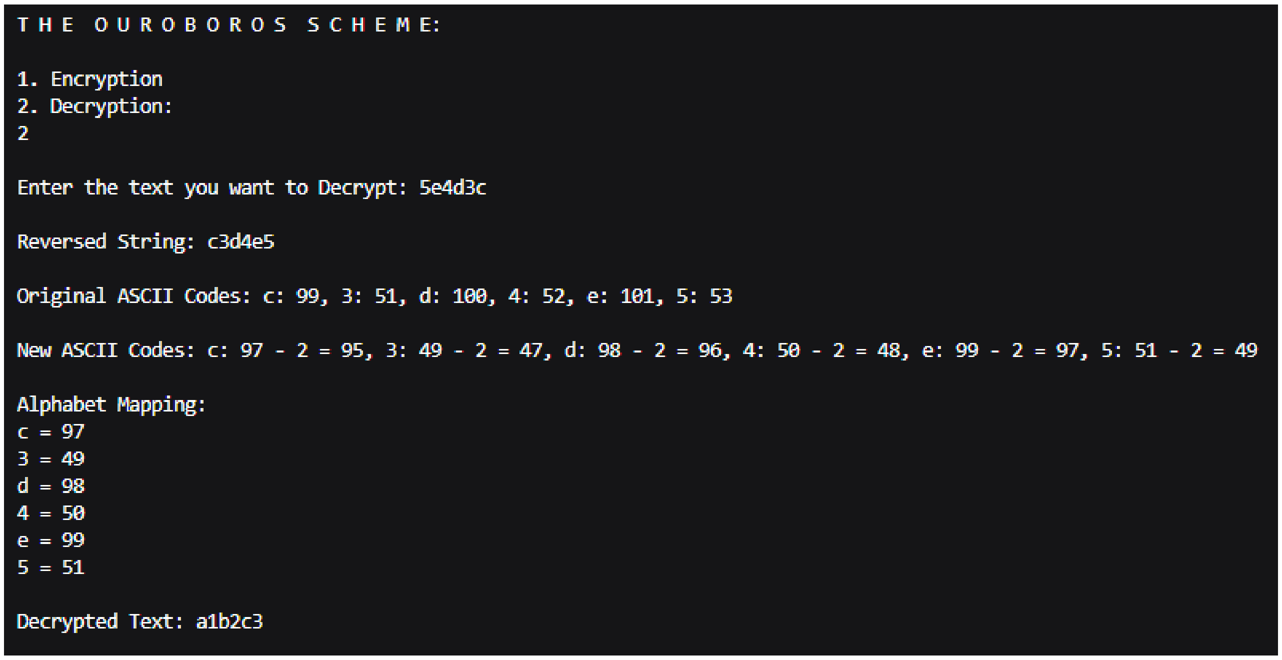
```
T H E  O U R O B O R O S  S C H E M E:

1. Encryption
2. Decryption:
2

Enter the text you want to Decrypt: zqd"gjV

Reversed String: Vjg"dqz

Original ASCII Codes: V: 86, j: 106, g: 103, ": 34, d: 100, q: 113, z: 122

New ASCII Codes: V: 84 - 2 = 82, j: 104 - 2 = 102, g: 101 - 2 = 99, ": 32 - 2 = 30, d: 98 - 2 = 96, q: 111 - 2 = 109, z: 120 - 2 = 118

Alphabet Mapping:
V = 84
j = 104
g = 101
" = 32
d = 98
q = 111
z = 120

Decrypted Text: The box
```

5.3 Test case for Encryption: (Alphanumeric String)



```
T H E  O U R O B O R O S  S C H E M E:
1. Encryption
2. Decryption:
1

Enter the text you want to Encrypt: a1b2c3

Enter Secret Key: 2

Reversed Text: 3c2b1a


Original ASCII Codes: 3: 51, c: 99, 2: 50, b: 98, 1: 49, a: 97

New ASCII Codes: 3: 51 + 2 = 53, c: 99 + 2 = 101, 2: 50 + 2 = 52, b: 98 + 2 = 100, 1: 49 + 2 = 51, a: 97 + 2 = 99

Alphabet Mapping:
53 = 5
101 = e
52 = 4
100 = d
51 = 3
99 = c

Encrypted Text: 5e4d3c
```

5.4 Test case for Decryption: (Alphanumeric String)



```
T H E  O U R O B O R O S  S C H E M E:

1. Encryption
2. Decryption:
2

Enter the text you want to Decrypt: 5e4d3c

Reversed String: c3d4e5

Original ASCII Codes: c: 99, 3: 51, d: 100, 4: 52, e: 101, 5: 53

New ASCII Codes: c: 97 - 2 = 95, 3: 49 - 2 = 47, d: 98 - 2 = 96, 4: 50 - 2 = 48, e: 99 - 2 = 97, 5: 51 - 2 = 49

Alphabet Mapping:
c = 97
3 = 49
d = 98
4 = 50
e = 99
5 = 51

Decrypted Text: a1b2c3
```