Programming Assignment-2

Abhinav Ujjawal (2021120) & Nikhil Suri (2021268)

Data Encryption Standard (Project-0)

DES (Data Encryption Standard) operates on fixed-size blocks of data, specifically 64 bits. Each 64-bit block of plaintext is processed through a series of permutations and substitutions.

DES uses a 56-bit key, where 8 bits are used for parity checking. The effective key size is 56 bits, making it susceptible to exhaustive key search attacks.

Encryption and decryption

Encryption

Iterate over the list of plaintexts and keylists. Let's say the current iteration has plaintext and key [p,k]. DES involves key permutations, initial and final permutations, substitution boxes (S-boxes), and a Feistel network with 16 rounds.

The algorithm uses two main permutations: Initial Permutation (IP) and Final Permutation (FP) DES employs a key generation process to create 16 subkeys from the original 56-bit key. Subkeys are derived using permutations (PC1 and PC2) and left circular shifts. Each round of DES involves a Feistel network that includes expansion, XOR, substitution, and permutation operations.

Decryption

Initial Permutation (IP):

Ciphertext undergoes initial permutation using table IP.

Feistel Rounds:

Ciphertext halves (L0, R0) are processed through 16 rounds.
Each round involves expansion, XOR with subkey, substitution, permutation, and swapping.

Final Permutation (FP):

After rounds, left/right halves are swapped.
Final permutation using table FP yields decrypted plaintext.

Output:

Decrypted plaintext is obtained by converting bits back to characters.

Documentation for methods used

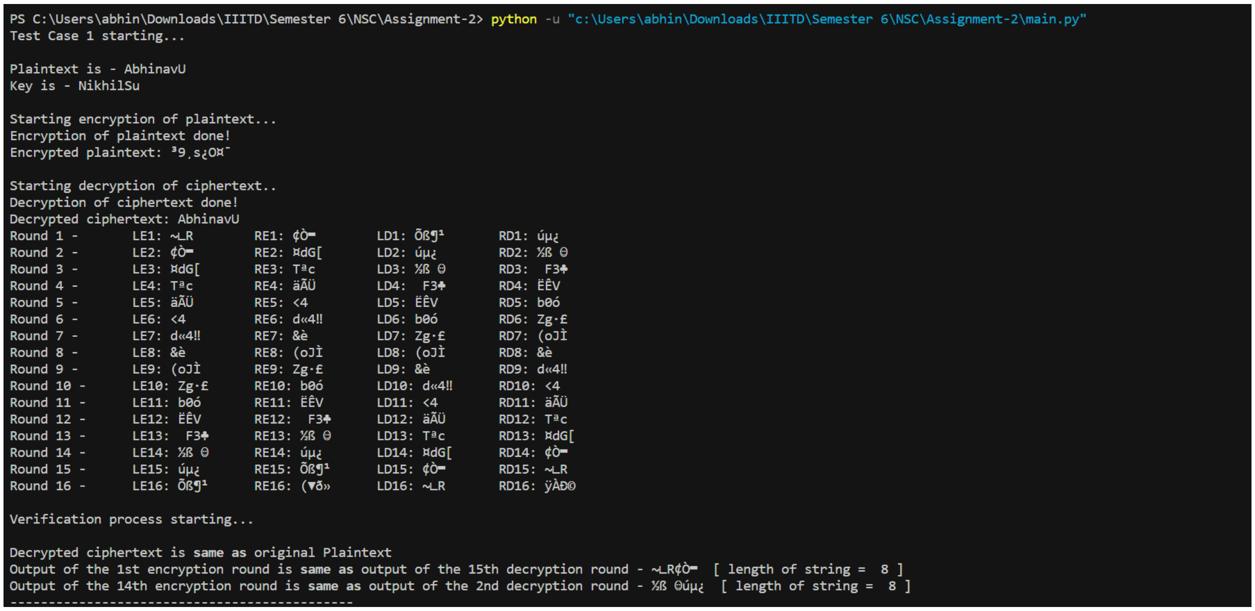
- permute(data, permutation): Performs permutation on the given data based on the specified permutation list. Returns a list of permuted elements.
- hex_to_bits(hex_number): Converts a hexadecimal number to a list of bits (0s and 1s).
- initial_permutation(block): Applies the initial permutation (IP) on the block.
- final_permutation(block): Applies the final permutation (FP) on the block.
- permutation(block): Performs permutation on the given block using the P permutation list.
- generate_subkeys(key): Generates a list of subkeys for the DES algorithm.
- feistel_network(right_half, subkey): Implements the Feistel network, a fundamental part of DES.
- des_encrypt(plaintext, key): Encrypts the plaintext using the DES algorithm with the provided key.
- des_decrypt(ciphertext, key): Decrypts the ciphertext using the DES algorithm with the provided key.
- permute(data, permutation): Utility function for permutation, similar to the first one.
- rotate_left(data, count): Rotates the elements in the data list to the left by the specified count.
- xor(a, b): Performs bitwise XOR operation on two lists a and b.
- expansion_permutation(block): Performs expansion permutation on the given block.
- substitute(data): Implements the substitution step using the S-boxes.
- input_to_8_bit_string(s): Converts a string into a list of 8-bit binary strings.
- convert_to_56_bit_key(key): Converts the key to a 56-bit binary string.
- hex_to_bits(hex_number): Converts a hexadecimal number to a list of bits.
- listOfBits(key, plaintext): Converts the key and plaintext to a list of bits.
- bits_to_char(ciphertext): Converts a list of bits to a string of characters.

Constraints

- Plaintext
  - Plaintext must be 64 bits long (since DES takes 64-bit input)
  - We have taken a list of 5 plaintexts that have 8 characters. Each character is converted to 8 bits using the character’s ASCII value (adding 0s as padding in-front of the MSB if ASCII value < 256)
- Key
  - Key must be 64 bits long.
  - We have taken a list of 5 keys. Each character in a key is converted to 8 bits, using the same logic described above

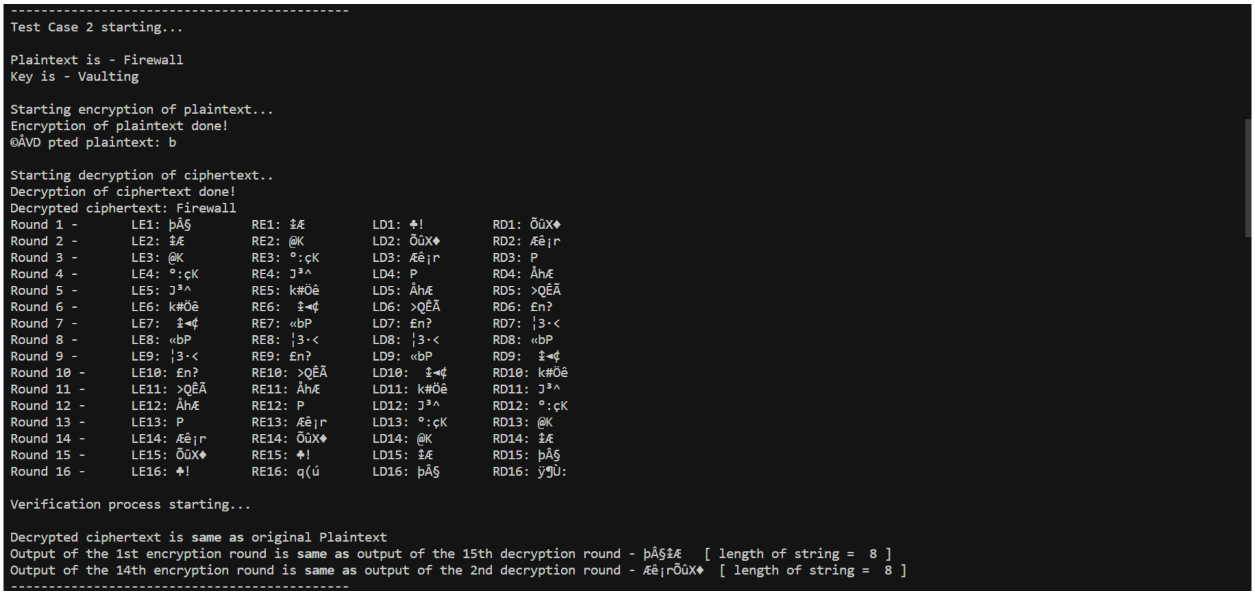
Sample Inputs & Outputs

1.



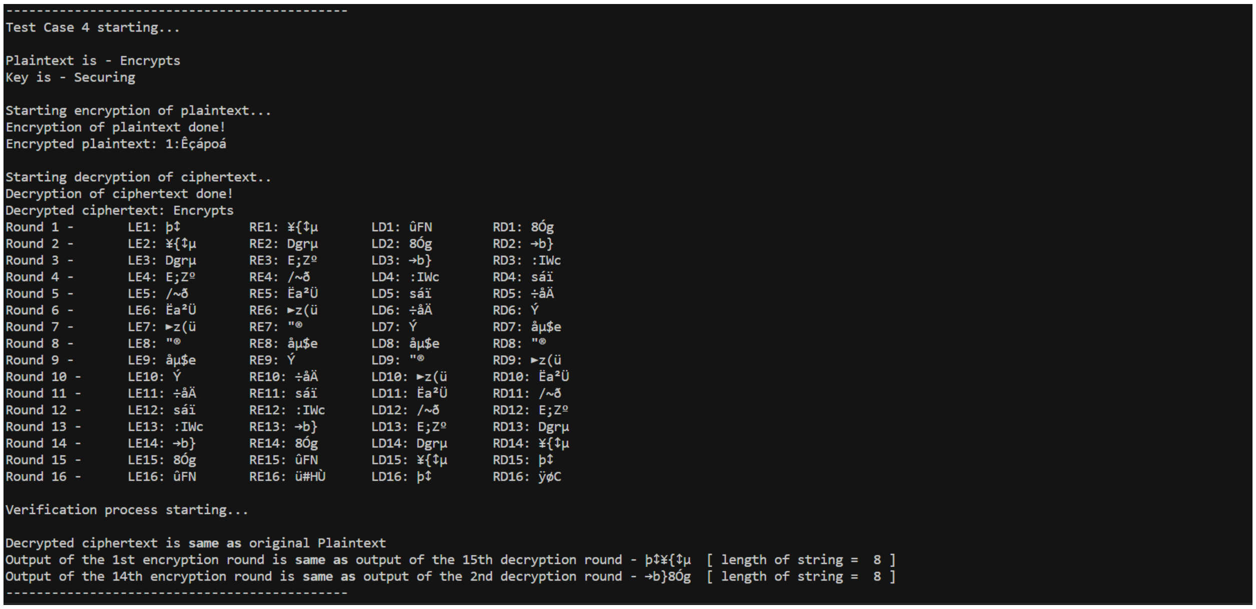
```
PS C:\Users\abhin\Downloads\IIITD\Semester 6\NSC\Assignment-2> python -u "c:\Users\abhin\Downloads\IIITD\Semester 6\NSC\Assignment-2\main.py"
Test Case 1 starting...

Plaintext is - AbhinavU
Key is - NikhilSu

Starting encryption of plaintext...
Encryption of plaintext done!
Encrypted plaintext: ³9¸s¿O¤¯

Starting decryption of ciphertext..
Decryption of ciphertext done!
Decrypted ciphertext: AbhinavU
Round 1 -       LE1: ~∟R        RE1: ¢Ò▬        LD1: Õß¶¹       RD1: úµ¿
Round 2 -       LE2: ¢Ò▬        RE2: ¤dG[       LD2: úµ¿        RD2: ½ß ☺
Round 3 -       LE3: ¤dG[       RE3: Tªc        LD3: ½ß ☺       RD3:  F3♣
Round 4 -       LE4: Tªc        RE4: äÃÜ        LD4:  F3♣       RD4: ËÊV
Round 5 -       LE5: äÃÜ        RE5: <4         LD5: ËÊV        RD5: b0ó
Round 6 -       LE6: <4         RE6: d«4‼       LD6: b0ó        RD6: Zg·£
Round 7 -       LE7: d«4‼       RE7: &è         LD7: Zg·£       RD7: (oJÌ
Round 8 -       LE8: &è         RE8: (oJÌ       LD8: (oJÌ       RD8: &è
Round 9 -       LE9: (oJÌ       RE9: Zg·£       LD9: &è         RD9: d«4‼
Round 10 -      LE10: Zg·£      RE10: b0ó       LD10: d«4‼      RD10: <4
Round 11 -      LE11: b0ó       RE11: ËÊV       LD11: <4        RD11: äÃÜ
Round 12 -      LE12: ËÊV       RE12:  F3♣      LD12: äÃÜ       RD12: Tªc
Round 13 -      LE13:  F3♣      RE13: ½ß ☺      LD13: Tªc       RD13: ¤dG[
Round 14 -      LE14: ½ß ☺      RE14: úµ¿       LD14: ¤dG[      RD14: ¢Ò▬
Round 15 -      LE15: úµ¿       RE15: Õß¶¹      LD15: ¢Ò▬       RD15: ~∟R
Round 16 -      LE16: Õß¶¹      RE16: (▼ð»      LD16: ~∟R       RD16: ÿÀÐ©

Verification process starting...

Decrypted ciphertext is same as original Plaintext
Output of the 1st encryption round is same as output of the 15th decryption round - ~∟R¢Ò▬  [ length of string =  8 ]
Output of the 14th encryption round is same as output of the 2nd decryption round - ½ß ☺úµ¿  [ length of string =  8 ]
--------------------------------------------
```

2.



```
--------------------------------------------
Test Case 2 starting...

Plaintext is - Firewall
Key is - Vaulting

Starting encryption of plaintext...
Encryption of plaintext done!
©ÅVD pted plaintext: b

Starting decryption of ciphertext..
Decryption of ciphertext done!
Decrypted ciphertext: Firewall
Round 1 -       LE1: þÂ§        RE1: ↕Æ         LD1: ♣!         RD1: ÕûX♦
Round 2 -       LE2: ↕Æ         RE2: @K         LD2: ÕûX♦       RD2: Æê¡r
Round 3 -       LE3: @K         RE3: °:çK       LD3: Æê¡r       RD3: P
Round 4 -       LE4: °:çK       RE4: J³^        LD4: P          RD4: ÅhÆ
Round 5 -       LE5: J³^        RE5: k#Öê       LD5: ÅhÆ        RD5: >QÊÃ
Round 6 -       LE6: k#Öê       RE6:  ↕◄¢       LD6: >QÊÃ       RD6: £n?
Round 7 -       LE7:  ↕◄¢       RE7: «bP        LD7: £n?        RD7: ¦3·<
Round 8 -       LE8: «bP        RE8: ¦3·<       LD8: ¦3·<       RD8: «bP
Round 9 -       LE9: ¦3·<       RE9: £n?        LD9: «bP        RD9:  ↕◄¢
Round 10 -      LE10: £n?       RE10: >QÊÃ      LD10:  ↕◄¢      RD10: k#Öê
Round 11 -      LE11: >QÊÃ      RE11: ÅhÆ       LD11: k#Öê      RD11: J³^
Round 12 -      LE12: ÅhÆ       RE12: P         LD12: J³^       RD12: °:çK
Round 13 -      LE13: P         RE13: Æê¡r      LD13: °:çK      RD13: @K
Round 14 -      LE14: Æê¡r      RE14: ÕûX♦      LD14: @K        RD14: ↕Æ
Round 15 -      LE15: ÕûX♦      RE15: ♣!        LD15: ↕Æ        RD15: þÂ§
Round 16 -      LE16: ♣!        RE16: q(ú       LD16: þÂ§       RD16: ÿ¶Ù:

Verification process starting...

Decrypted ciphertext is same as original Plaintext
Output of the 1st encryption round is same as output of the 15th decryption round - þÂ§↕Æ   [ length of string =  8 ]
Output of the 14th encryption round is same as output of the 2nd decryption round - Æê¡rÕûX♦  [ length of string =  8 ]
--------------------------------------------
```

3.



```
---------------------------------------------
Test Case 4 starting...

Plaintext is - Encrypts
Key is - Securing

Starting encryption of plaintext...
Encryption of plaintext done!
Encrypted plaintext: 1:Êçápoá

Starting decryption of ciphertext..
Decryption of ciphertext done!
Decrypted ciphertext: Encrypts
Round 1 -       LE1: þ↕         RE1: ¥{↕µ       LD1: ûFN        RD1: 8Óg
Round 2 -       LE2: ¥{↕µ       RE2: Dgrµ       LD2: 8Óg        RD2: →b}
Round 3 -       LE3: Dgrµ       RE3: E;Zº       LD3: →b}        RD3: :IWc
Round 4 -       LE4: E;Zº       RE4: /~ð        LD4: :IWc       RD4: sáï
Round 5 -       LE5: /~ð        RE5: Ëa²Ü       LD5: sáï        RD5: ÷åÄ
Round 6 -       LE6: Ëa²Ü       RE6: ►z(ü       LD6: ÷åÄ        RD6: Ý
Round 7 -       LE7: ►z(ü       RE7: "®         LD7: Ý          RD7: åµ$e
Round 8 -       LE8: "®         RE8: åµ$e       LD8: åµ$e       RD8: "®
Round 9 -       LE9: åµ$e       RE9: Ý          LD9: "®         RD9: ►z(ü
Round 10 -      LE10: Ý         RE10: ÷åÄ       LD10: ►z(ü      RD10: Ëa²Ü
Round 11 -      LE11: ÷åÄ       RE11: sáï       LD11: Ëa²Ü      RD11: /~ð
Round 12 -      LE12: sáï       RE12: :IWc      LD12: /~ð       RD12: E;Zº
Round 13 -      LE13: :IWc      RE13: →b}       LD13: E;Zº      RD13: Dgrµ
Round 14 -      LE14: →b}       RE14: 8Óg       LD14: Dgrµ      RD14: ¥{↕µ
Round 15 -      LE15: 8Óg       RE15: ûFN       LD15: ¥{↕µ      RD15: þ↕
Round 16 -      LE16: ûFN       RE16: ü#HÙ      LD16: þ↕        RD16: ÿøC

Verification process starting...

Decrypted ciphertext is same as original Plaintext
Output of the 1st encryption round is same as output of the 15th decryption round - þ↕¥{↕µ  [ length of string =  8 ]
Output of the 14th encryption round is same as output of the 2nd decryption round - →b}8Óg  [ length of string =  8 ]
---------------------------------------------
```