Experiment 5

BlowFish

Name: Kashish Jain
UID: 2019130022
Class: TE Comps

Aim: To implement blowfish algorithm.

THEORY

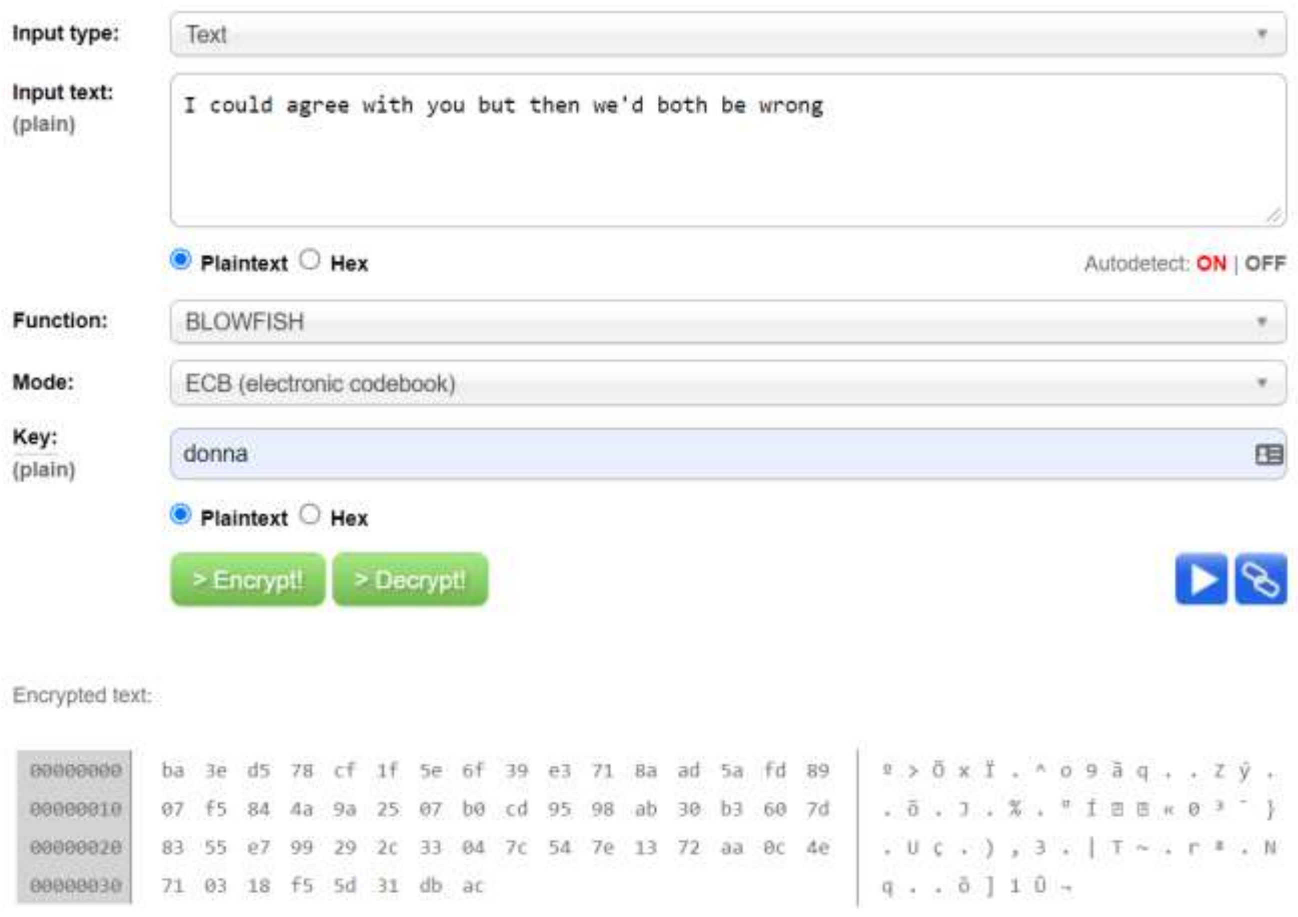
BLOWFISH ALGORITHM:

Blowfish is a symmetric-key block cipher, designed in 1993 by Bruce Schneier and included in many cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date.
Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. At the time Blowfish was released, many other designs were proprietary, encumbered by patents or were commercial or government secrets. Schneier has stated that, "Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone."
Notable features of the design include key-dependent S-boxes and a highly complex key schedule.
Blowfish is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use.

Blowfish uses:

- **blockSize**: 64-bits
- **keySize**: 32-bits to 448-bits variable size
- **number of subkeys**: 18 [P-array]
- number of rounds: 16
- **number of substitution boxes**: 4 [each having 512 entries of 32-bits each

Original encoded message



Input type: Text

Input text: (plain)

```
I could agree with you but then we'd both be wrong
```

[✓ Plaintext] Hex   Autodetect: ON | OFF

Function: BLOWFISH

Mode: ECB (electronic codebook)

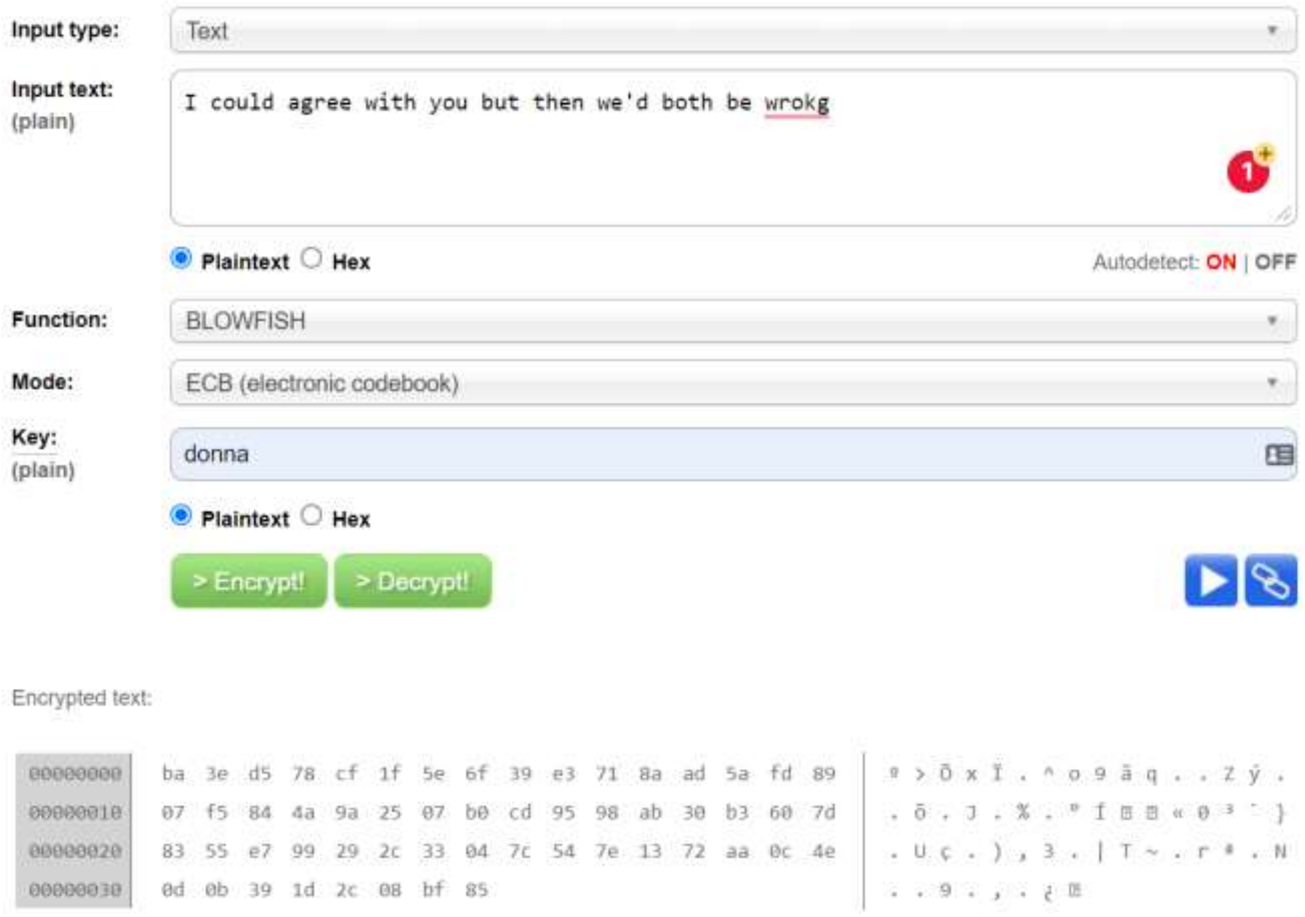
Key: (plain) donna

[✓ Plaintext] Hex

Encrypted text:

```
00000000  ba 3e d5 78 cf 1f 5e 6f 39 e3 71 8a ad 5a fd 89 | º > Õ x Ï . ^ o 9 ã q . . Z ý .
00000010  07 f5 84 4a 9a 25 07 b0 cd 95 98 ab 30 b3 60 7d | . õ . J . % . ° Í . . « 0 ³ ` }
00000020  83 55 e7 99 29 2c 33 04 7c 54 7e 13 72 aa 0c 4e | . U ç . ) , 3 . | T ~ . r ª . N
00000030  71 03 18 f5 5d 31 db ac                         | q . . õ ] 1 Û ¬
```

1)If you change one character at the end of the message, the encoded message changes in the following way:



Input type: Text

Input text: (plain)

```
I could agree with you but then we'd both be wrokg
```

[✓ Plaintext] Hex — Autodetect: ON | OFF

Function: BLOWFISH

Mode: ECB (electronic codebook)

Key: (plain) donna

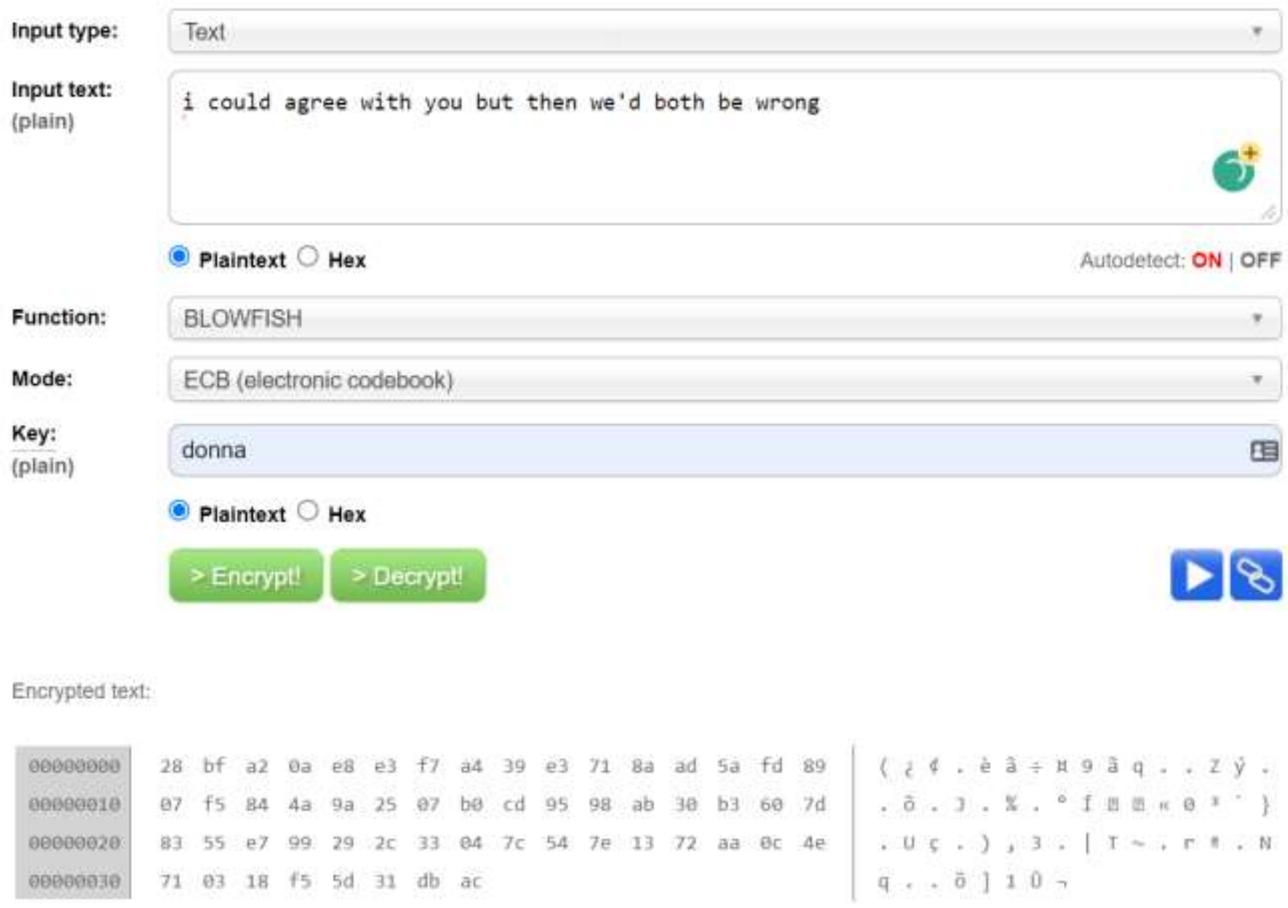
[✓ Plaintext] Hex

Encrypted text:

```
00000000  ba 3e d5 78 cf 1f 5e 6f 39 e3 71 8a ad 5a fd 89  º > Õ x Ï . ^ o 9 ã q . . Z ý .
00000010  07 f5 84 4a 9a 25 07 b0 cd 95 98 ab 30 b3 60 7d  . õ . J . % . º Í □ □ « 0 ³ ` }
00000020  83 55 e7 99 29 2c 33 04 7c 54 7e 13 72 aa 0c 4e  . U ç . ) , 3 . | T ~ . r ª . N
00000030  0d 0b 39 1d 2c 08 bf 85                          . . 9 . , . ¿ □
```

After changing the last character of plain text message, last 16 characters of the encrypted message changes, and the rest of the encrypted message remains same.

2)If you change one character at the beginning of the message, the encoded message changes as follows:



Input type: Text

Input text: (plain)

```
i could agree with you but then we'd both be wrong
```

Plaintext Hex — Autodetect: ON | OFF

Function: BLOWFISH

Mode: ECB (electronic codebook)

Key: (plain) donna

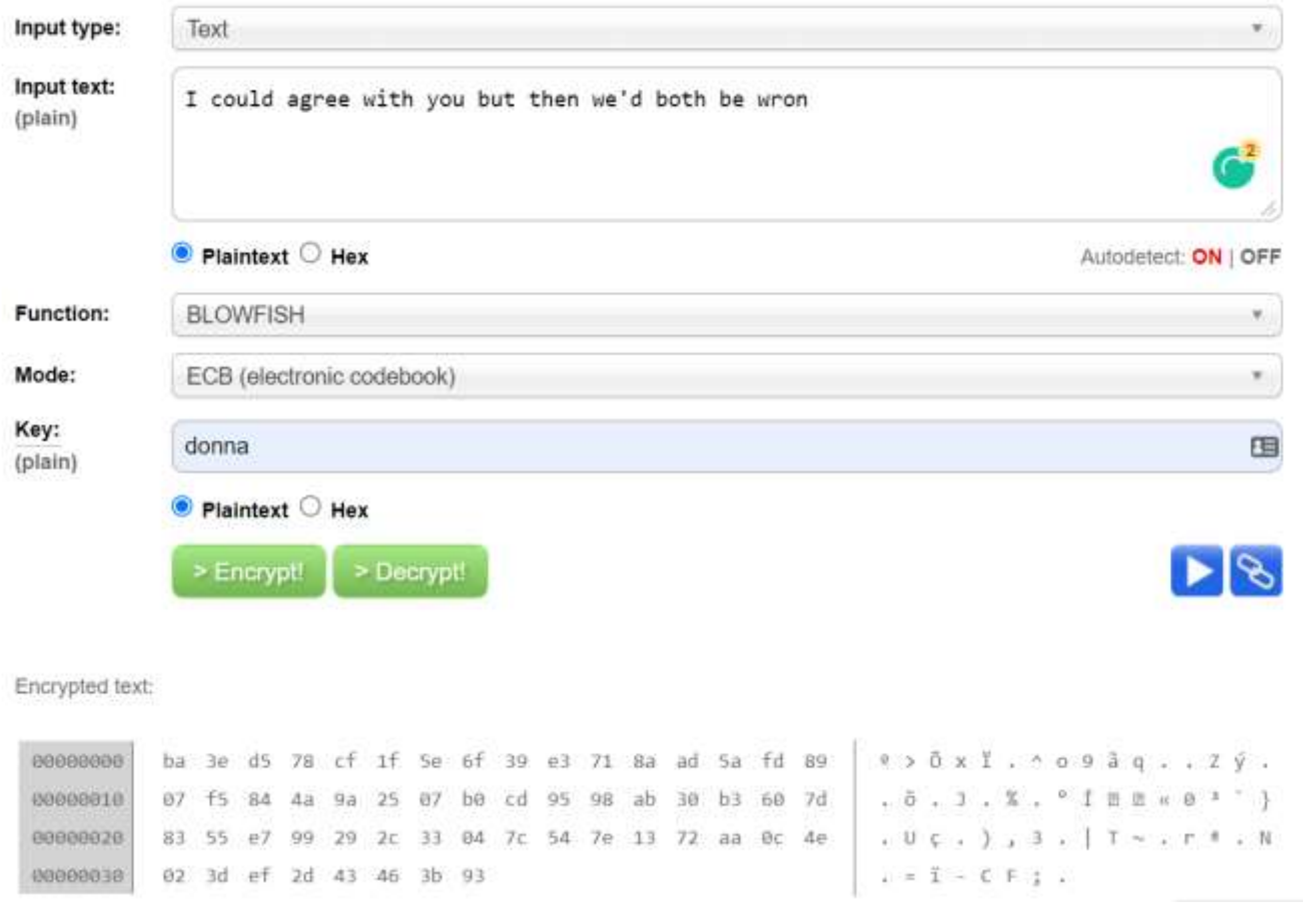
Plaintext Hex

Encrypted text:

```
00000000   28 bf a2 0a e8 e3 f7 a4 39 e3 71 8a ad 5a fd 89   ( ¿ ¢ . è ã ÷ ¤ 9 ã q . . Z ý .
00000010   07 f5 84 4a 9a 25 07 b0 cd 95 98 ab 30 b3 60 7d   . õ . J . % . ° Í . . « 0 ³ ` }
00000020   83 55 e7 99 29 2c 33 04 7c 54 7e 13 72 aa 0c 4e   . U ç . ) , 3 . | T ~ . r ª . N
00000030   71 03 18 f5 5d 31 db ac                           q . . õ ] 1 Û ¬
```

After changing the first character of plain text message, first 16 characters of the encrypted message changes, and the rest of the encrypted message remains same.

3)If you delete one character at the end of the message, the encoded message changes as follows:



Input type: Text

Input text: (plain)
I could agree with you but then we'd both be wron

[✓ Plaintext] Hex   Autodetect: ON | OFF

Function: BLOWFISH

Mode: ECB (electronic codebook)

Key: (plain)
donna

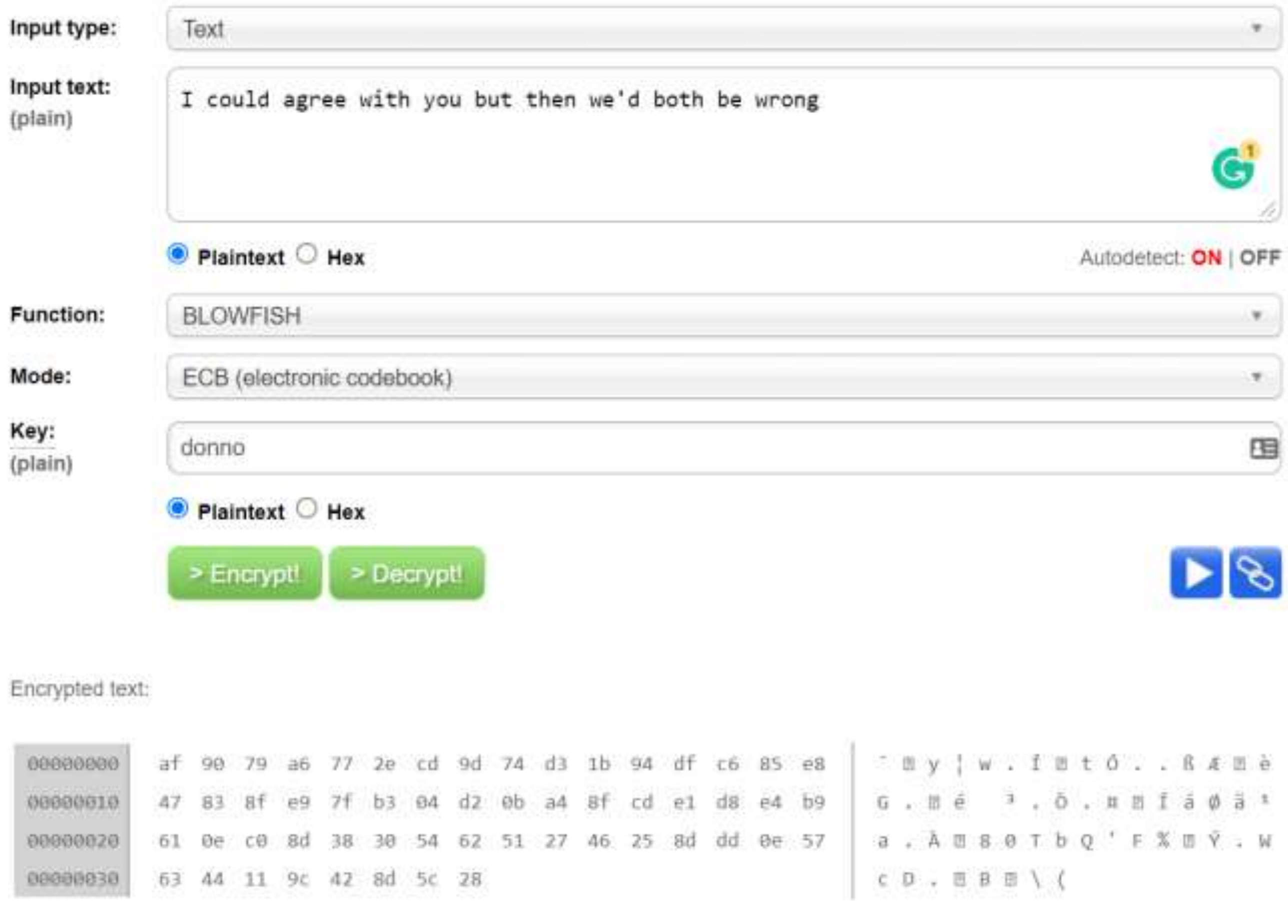
[✓ Plaintext] Hex

Encrypted text:

```
00000000  ba 3e d5 78 cf 1f 5e 6f 39 e3 71 8a ad 5a fd 89 | º > Õ x Ï . ^ o 9 ã q . . Z ý .
00000010  07 f5 84 4a 9a 25 07 b0 cd 95 98 ab 30 b3 60 7d | . õ . J . % . ° Í . . « 0 ³ ` }
00000020  83 55 e7 99 29 2c 33 04 7c 54 7e 13 72 aa 0c 4e | . U ç . ) , 3 . | T ~ . r ª . N
00000030  02 3d ef 2d 43 46 3b 93                         | . = ï - C F ; .
```

After deleting last character of plain text message, last 16 characters of the encrypted message changes, and the rest of the encrypted message remains same.
Size still remains the same since ECB is used which is a block cipher.

4)If you change one character in a key, the encoded message changes as follows:



Input type: Text

Input text: (plain)

I could agree with you but then we'd both be wrong

[✓ Plaintext] Hex

Autodetect: ON | OFF

Function: BLOWFISH

Mode: ECB (electronic codebook)

Key: (plain)

donno

[✓ Plaintext] Hex

Encrypted text:

```
00000000   af 90 79 a6 77 2e cd 9d 74 d3 1b 94 df c6 85 e8   ¯ ▯ y ¦ w . Í ▯ t Ó . . ß Æ ▯ è
00000010   47 83 8f e9 7f b3 04 d2 0b a4 8f cd e1 d8 e4 b9   G . ▯ é   ³ . Ò . ¤ ▯ Í á Ø ä ¹
00000020   61 0e c0 8d 38 30 54 62 51 27 46 25 8d dd 0e 57   a . À ▯ 8 0 T b Q ' F % ▯ Ý . W
00000030   63 44 11 9c 42 8d 5c 28                           c D . ▯ B ▯ \ (
```

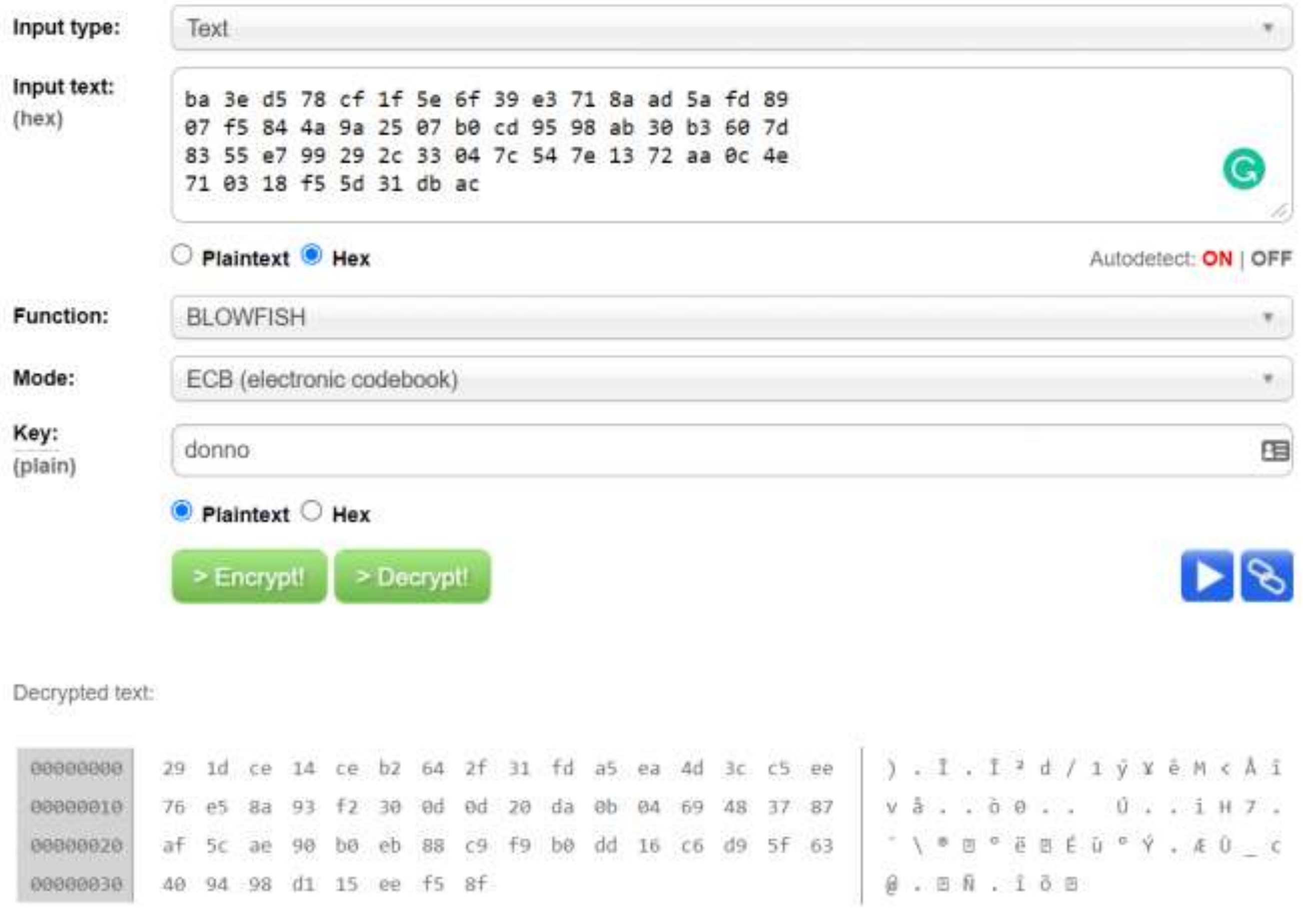
After changing one character in a key, entire encrypted message changes. Still the size of the encrypted message remains the same since the key length is the same.

Github Links

Repository

https://github.com/kashishvjain/CSS-Lab

5)Decrypt a message using a key with one character changed. Does it look anything like the original?



Input type: Text

Input text: (hex)

```
ba 3e d5 78 cf 1f 5e 6f 39 e3 71 8a ad 5a fd 89
07 f5 84 4a 9a 25 07 b0 cd 95 98 ab 30 b3 60 7d
83 55 e7 99 29 2c 33 04 7c 54 7e 13 72 aa 0c 4e
71 03 18 f5 5d 31 db ac
```

[Plaintext ✓ Hex] Autodetect: ON | OFF

Function: BLOWFISH

Mode: ECB (electronic codebook)

Key: (plain) donno

[✓ Plaintext] Hex

Decrypted text:

```
00000000  29 1d ce 14 ce b2 64 2f 31 fd a5 ea 4d 3c c5 ee  | ) . Ì . Î ² d / 1 ý ¥ ê M < Å î
00000010  76 e5 8a 93 f2 30 0d 0d 20 da 0b 04 69 48 37 87  | v å . . ò 0 . .   Ú . . i H 7 .
00000020  af 5c ae 90 b0 eb 88 c9 f9 b0 dd 16 c6 d9 5f 63  | ¯ \ ® ▯ ° ë ▯ É ù ° Ý . Æ Ù _ c
00000030  40 94 98 d1 15 ee f5 8f                          | @ . ▯ Ñ . î õ ▯
```

1. Here key used is donno and message is original encoded message.
2. It does not look like original message and decrypted message consists of lots of special characters.

CONCLUSION

1. Studied and understood the Blowfish algorithm.
2. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use.
3. Blowfish is considered to be a block Cipher since changing one text alters that section of the block encryption.
4. It is also a symmetric cypher because it encrypts and decrypts with the same key. Any change in key causes the ciphered text to be incorrectly deciphered.
