**NAME :** Mansi Dwivedi

**BRANCH :** IT

**UID :** 2019140016

**BATCH :** A

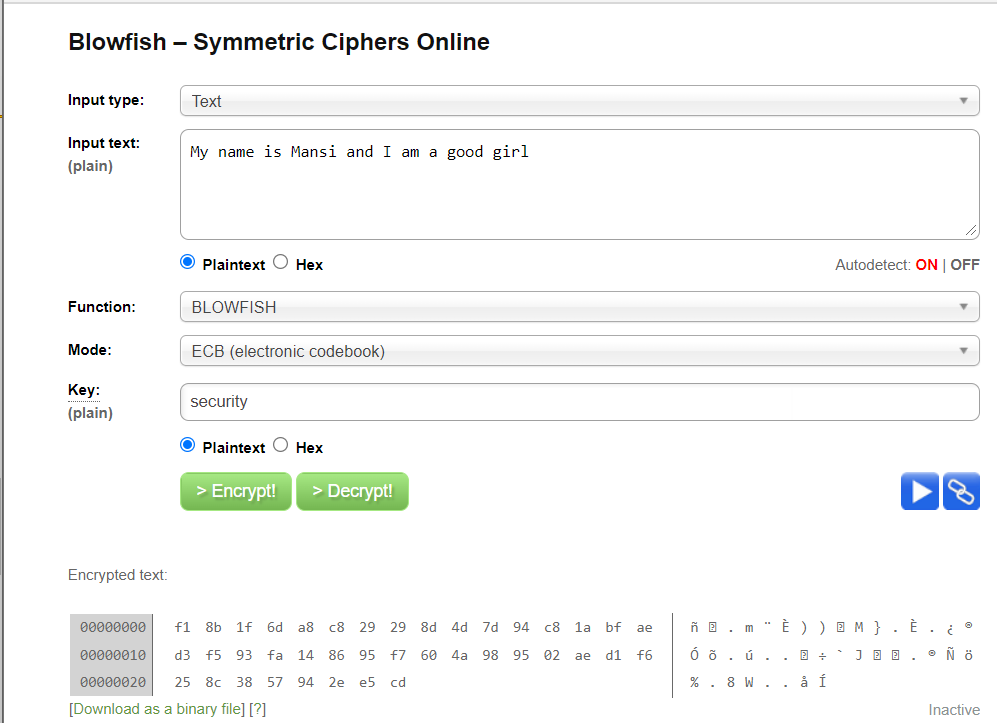
**COURSE :** CSS LAB

**EXPERIMENT :** 5

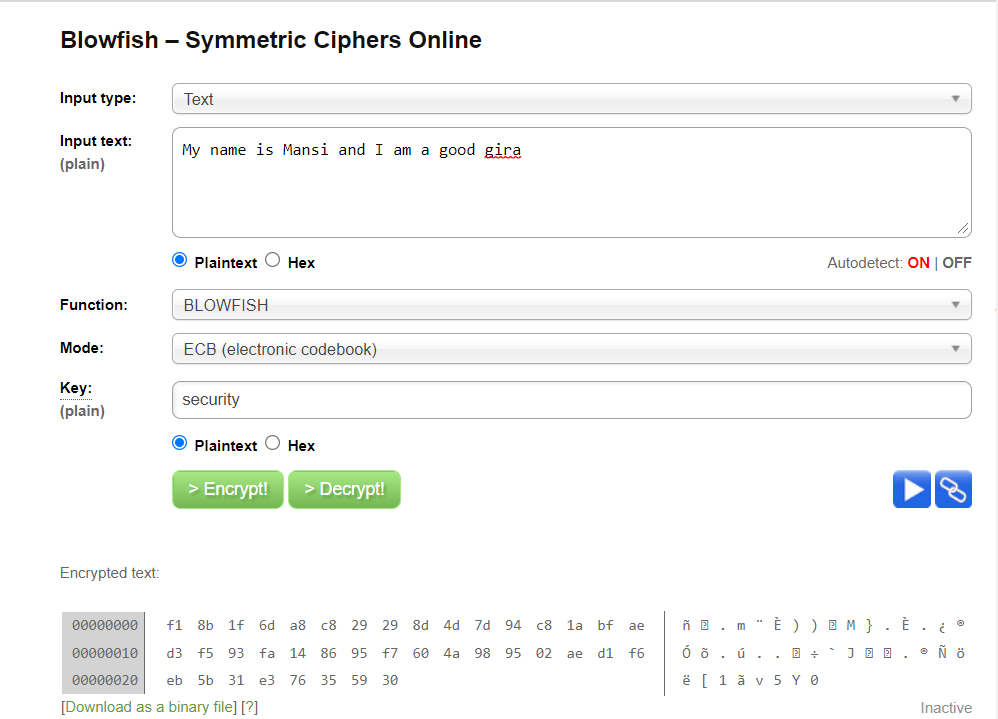
**AIM :** The aim of this lab is to experiment with an online encryption tool. We will encode a message and send it to someone else in the class, who will decode it when we supply the secret key. Note that this particular tool is of limited use in a security context, since the plaintext of the message is sent to and from the encryption website! However, it could be used to prevent people from reading your email. A similar tool downloaded and running on your computer would provide a greater level of security. Some email clients even provide support for automatic encryption and decryption of all messages.

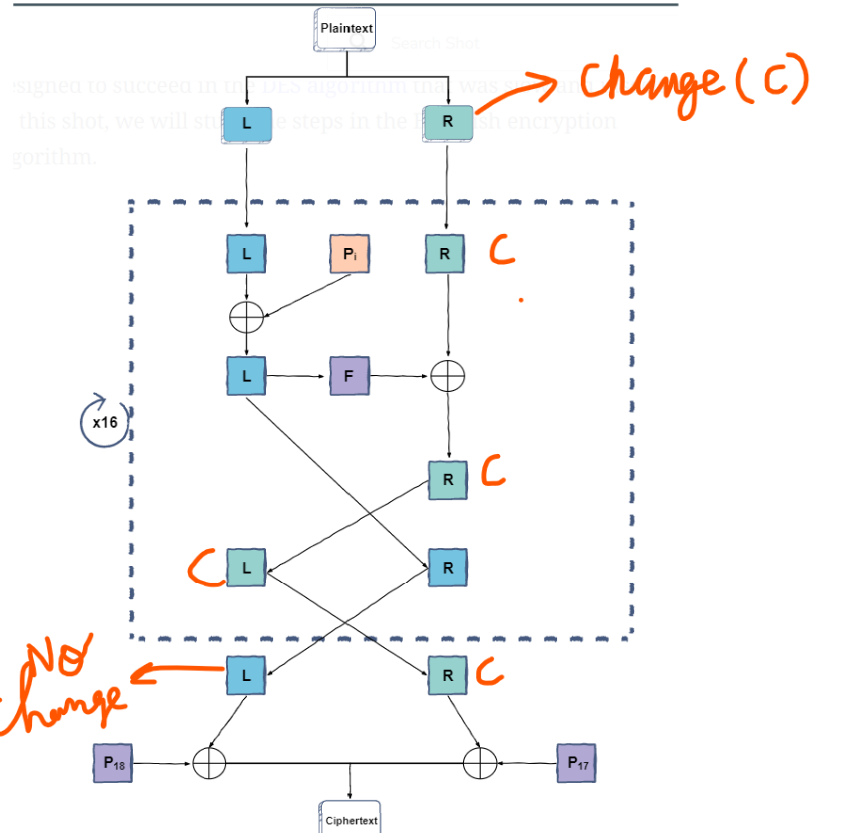
**PROBLEM STATEMENT :**

1. **Go to the encryption tool website and try it out. Enter a short key phrase and a longer piece of text to be encoded. Then submit and see what your text looks like when encrypted. Try the following experiments and note how they change the output:**

****

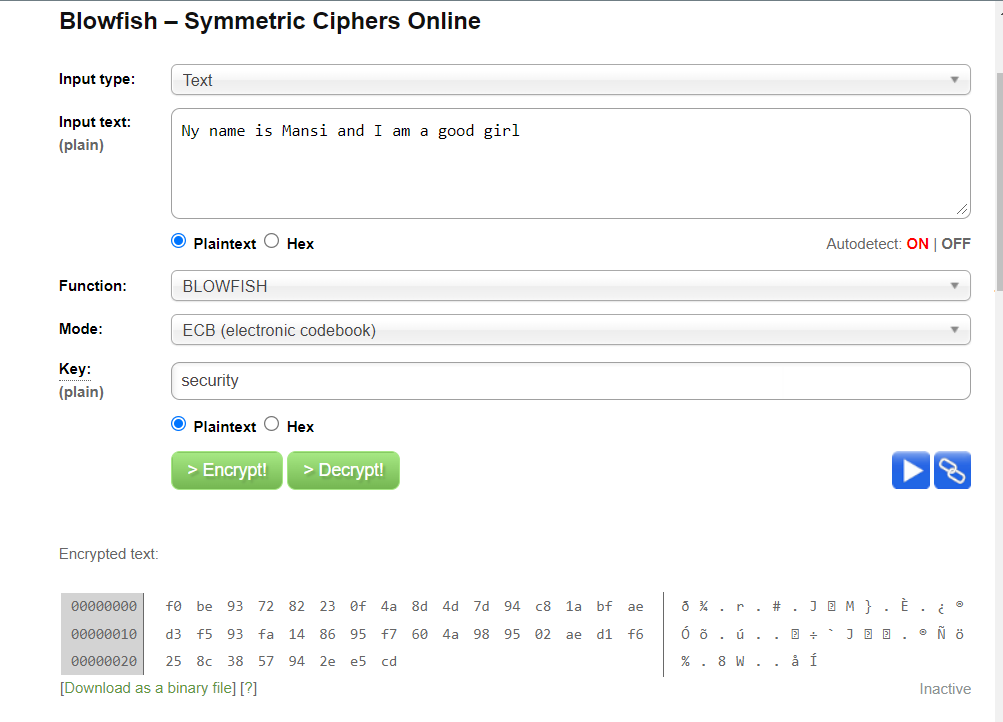
* **Change one character at the end of the message. How much of the encoded message changes?**

****

****

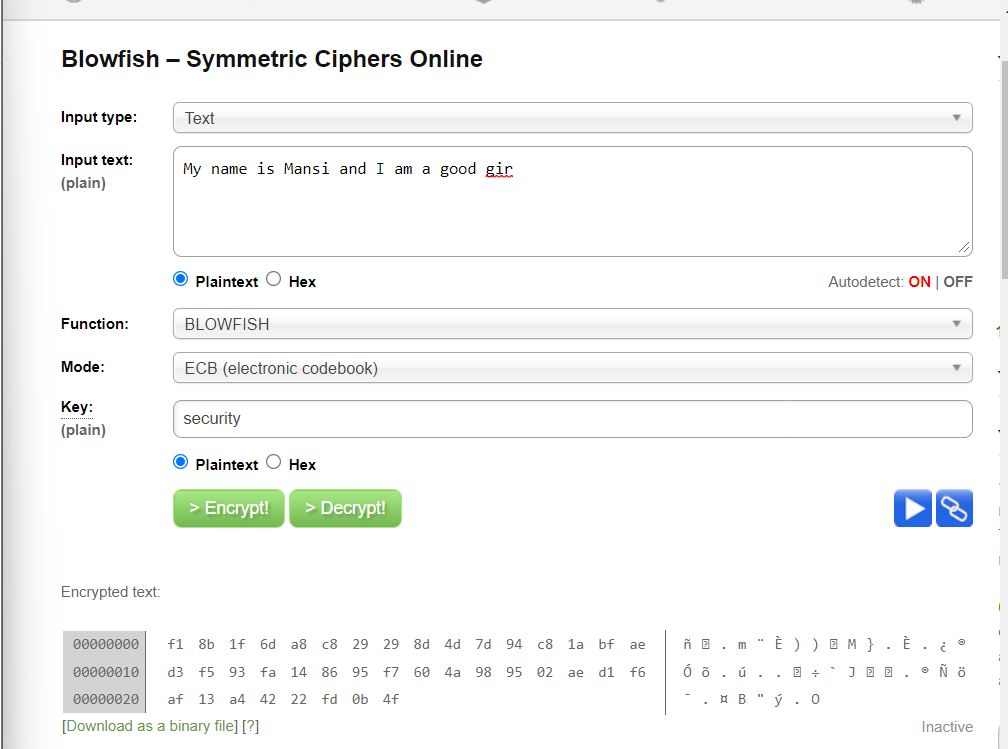
From the above diagram, we can understand that the leftmost 32 bits of the plain text after each iteration keep on exchanging with the rightmost 32 bits. This happens in every step until the loop breaks that is after 16 turns. So when inside the loop, the block where the change of character occurred always ends up being in the leftmost 32 bits but finally due to out of the loop swapping the changed bit ends up in the rightmost part only. So the change that I observed was that the entire last block ended up changing in the encrypted text.

* **Change one character at the beginning of the message. How much of the encoded message changes?**

****

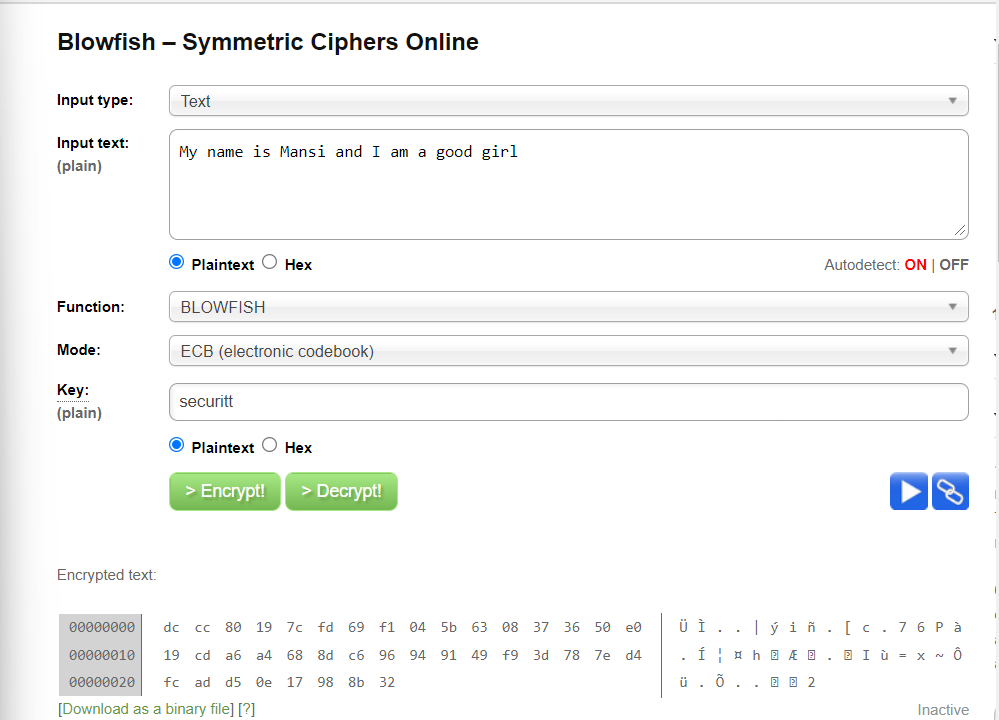
Similar to the above case the entire left block ended up changing on account of relacing an M by N and this behavior can be mapped to the block cipher property of Blowfish.

* **Delete one character at the end of the message. How much of the encoded message changes?**

****

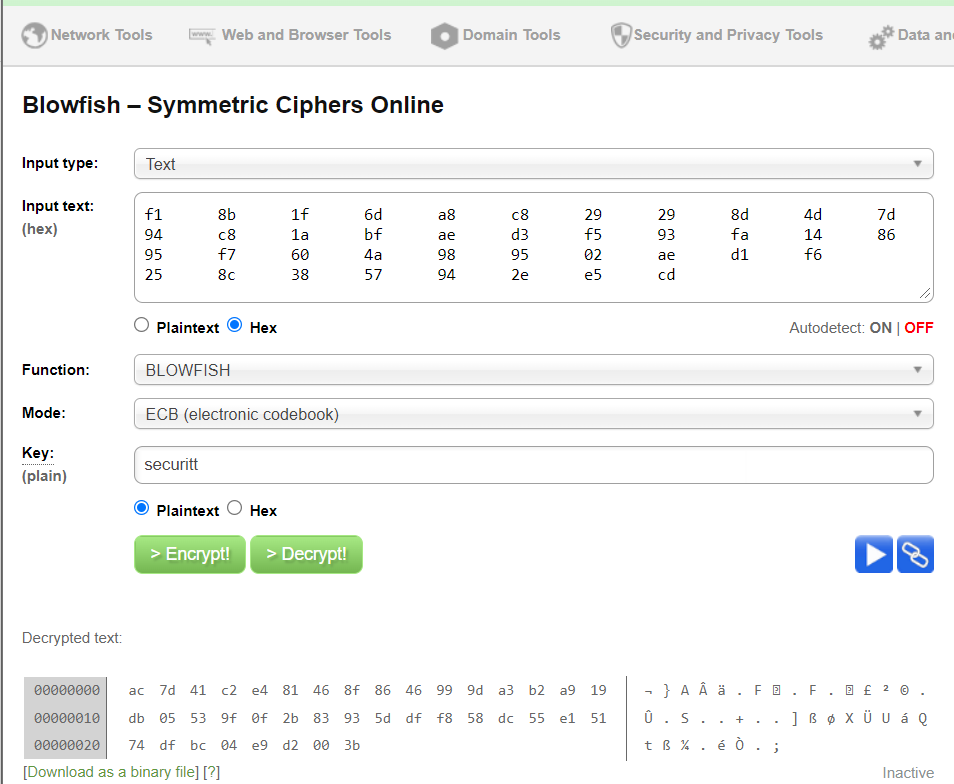
Again after removing the last character, the entire last block ended up changing.

* **Change one character in the key. How much of the encoded message changes?**

****

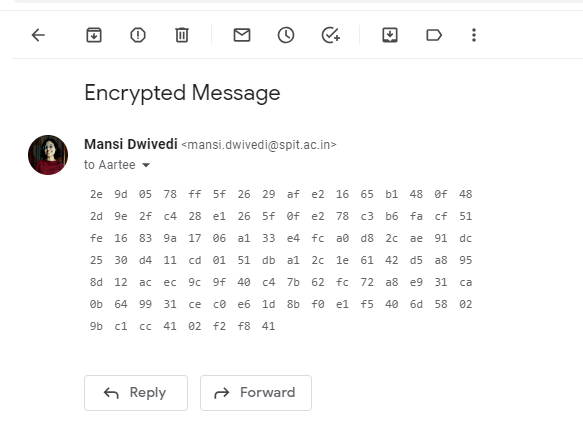
This behavior was expected because blowfish being a symmetric encryption algorithm the value of Pi(Permutation Box values) is dependent solely on the key which keeps on changing in every loop hence even a minor change in the key will lead to an entirely different encrypted text.

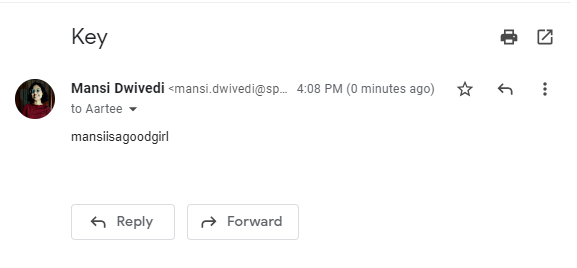
* **Decrypt a message using a key with one character changed. Does it look anything like the original?**

****

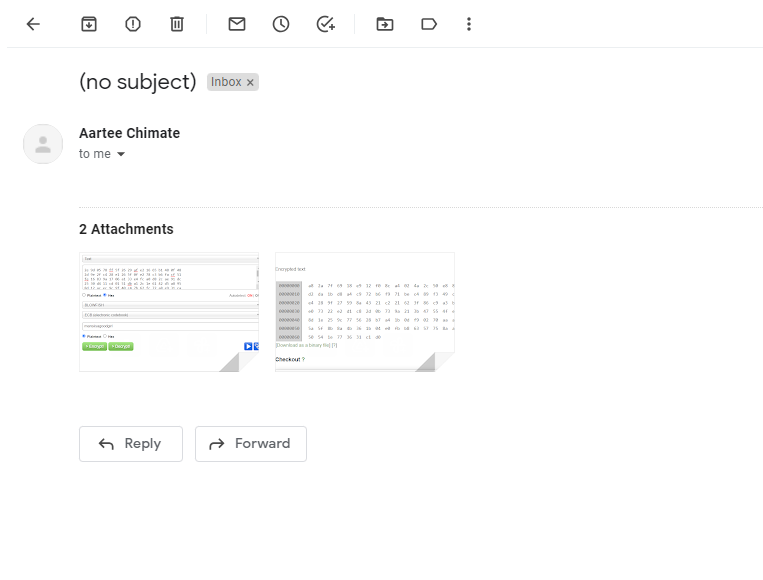
Decrypting a message with a changed key leads to a different message altogether. There is absolutely no similarity between the actual plain text and the message obtained in the latter case.

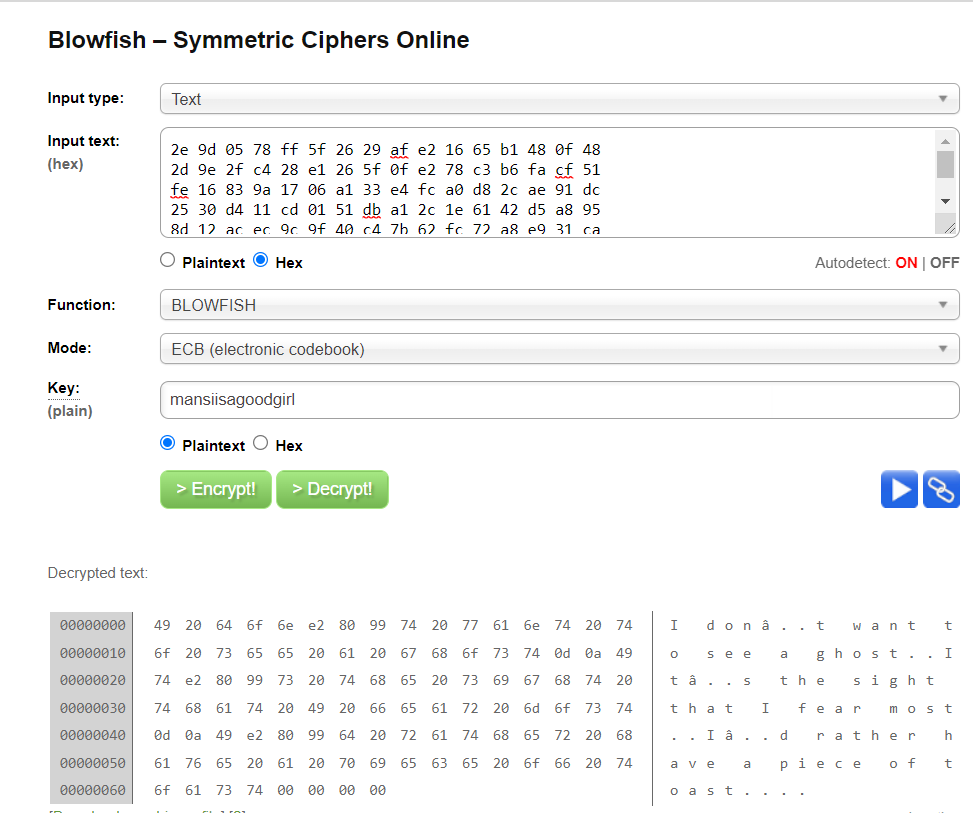
1. **A Secret Message When you have finished the above, see if you can decode the following message. ED85E0929D1248116C52FA6AFFB1DAC1** E2D472B6E8EA93AECD0D518D04DF3188 715D3AF7877684AC34EEB0FF3768B8DD 9E227C12E7340390987FDD12F9B9C156 F05A0748FBACFBC48D4B70C99780413F 652E6676330AC76F1DE7380E81B12E11 (Blowfish: By PV-J)
2. **Now it is time to send a secret message to someone else in the class. Use the tool to encode your message (without your partner seeing it) and copy the encoded text into an email. Send the key in a separate email, or tell it to the recipient. She/He should be able to decode the message using the same tool.**





**Output Obtained by the friend :**

****



1. **Public Key Cryptography Experiment with this page designed to demo cryptography with public/private key pairs. Note how a message encrypted with one key can be decrypted using the other.**