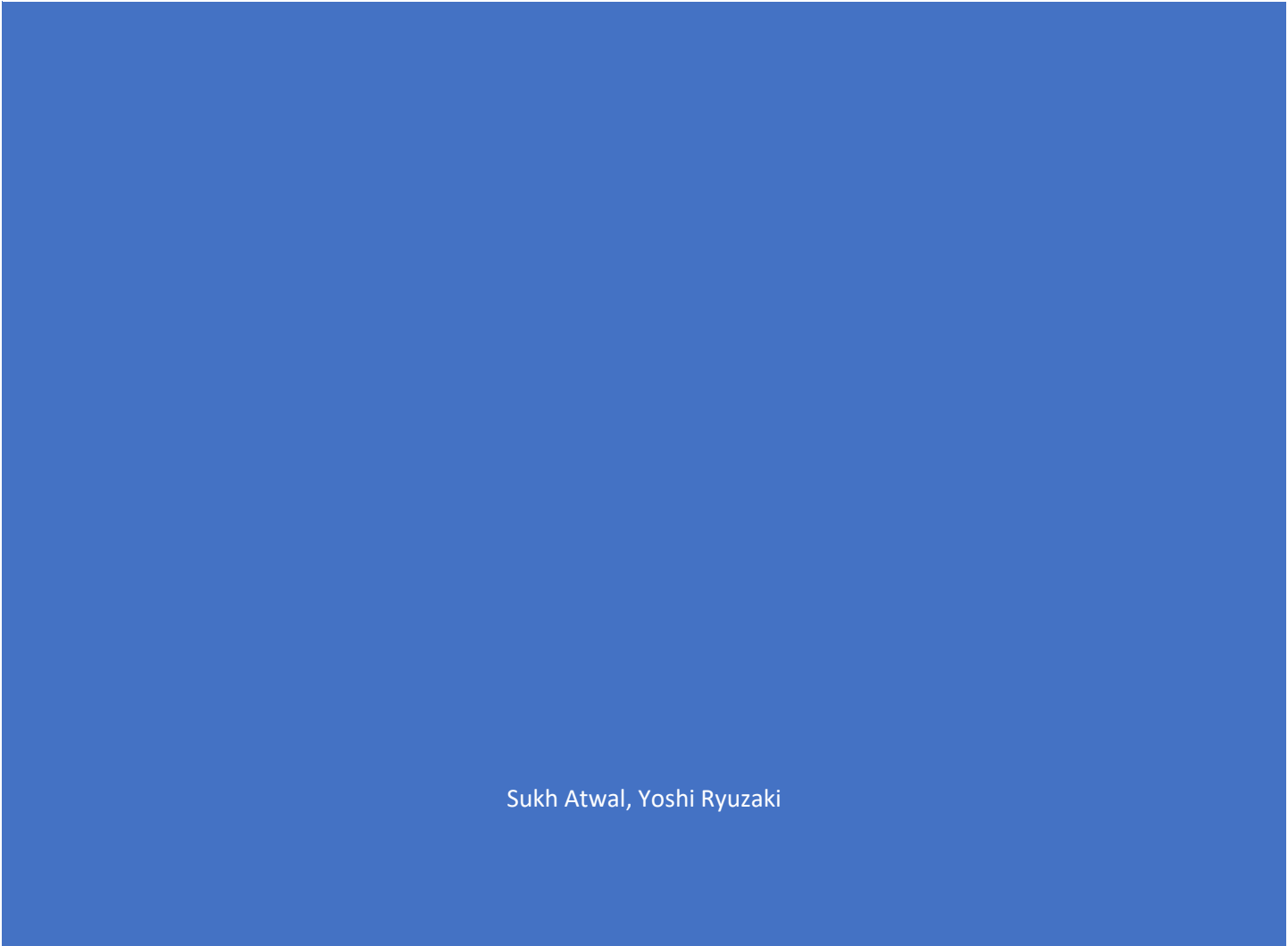




FEISTEL CIPHER



Sukh Atwal, Yoshi Ryuzaki

Table of Contents

Introduction	2
User Guide	3
Encryption	3
Decryption.....	5
Analysis	8
Diffusion.....	8
Confusion	8
Avalanche Effect	9
Design.....	11
Overall application flow	11
Text file encryption design.....	11
BMP file encryption design	12
Feistel encryption flow.....	13
Testing.....	16
Test 1.....	17
Test 2.....	18
Test 3.....	19
Test 4.....	22
Test 5.....	24
Test 6.....	26

Introduction

The Feistel cipher isn't so much a cipher in itself, it's more so of a framework for building encryption algorithm. The objective of this assignment is to design and implement an 8-round Feistel cipher with the default mode being ECB. We'll be using python as our choice of programming language and will allow users to encrypt as well as decrypt text files and images if they choose to with a key of their choice.








User Guide

Here is the user guide of the application.

Note: The application uses python3, and the functionality has only been verified using python 3.8 under Windows environment.

The application is very simple and easy to use. The README.MD file will also be included for quick reference.

The “Source” folder should contain 4 python script files and a sample text as well as 2 sample images that we used to verify the script was working.

 decryption.py	11/11/2020 3:33 PM	Python File	4 KB
 encryption.py	11/11/2020 3:10 PM	Python File	6 KB
 filehandling.py	11/11/2020 1:45 PM	Python File	1 KB
 main.py	11/11/2020 4:32 PM	Python File	3 KB
 sample.txt	10/21/2020 10:02 PM	Text Document	4 KB
 sample1.BMP	11/8/2020 6:33 PM	BMP File	770 KB
 sample2.bmp	11/9/2020 10:24 PM	BMP File	3,199 KB

Encryption

Here is a step by step guide for the encryption process. This will be for an image however you can follow the exact same steps and choose the “text” file type if you want to encrypt text instead.

1. First, we will launch the application using ‘python main.py’ or ‘python3 main.py’ command:

```
C:\Users\yryuz\Documents\BCIT\Crypto\Assingment5>python main.py
```

2. Next, after getting prompt, we will have to enter which operation mode we want to use (either encryption or decryption which you can specify with “1” or “2”):

```
Please enter the operation mode (Choose the number of the mode):
```

```
1.Encryption Mode
```

```
2.Decryption Mode
```

```
:1_
```

3. For now, we will use operation mode 1 which is the encryption mode. The application will prompt us to choose the file type.

We will select 1 (Image File type) here:

```
.1
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1_
```

4. After selecting the type of file, the application will prompt us to input the file name. We will use sample1.bmp for this case:

```
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample1.bmp_
```

5. Next, we will also have to enter the output file name of the encrypted data:

```
Please enter the output file name/path: encrypted.bmp_
```

6. The last input we must make is the key value we want to use to encrypt the file. The Key value can be any integer, we will use 225 for this example.

```
Please enter the output file name/path: encrypted.bmp
Please enter the encryption key (Any integer number) to use:225_
```

7. The application returns 'Encrypting, please wait' where we will wait until the application prints out an 'Finished Encryption' message.

```

C:\Users\yryuz\Documents\BCIT\Crypto\Assingment5>python main.py
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:1
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample1.bmp
Please enter the output file name/path: encrypted.bmp
Please enter the encryption key (Any integer number) to use:225

```

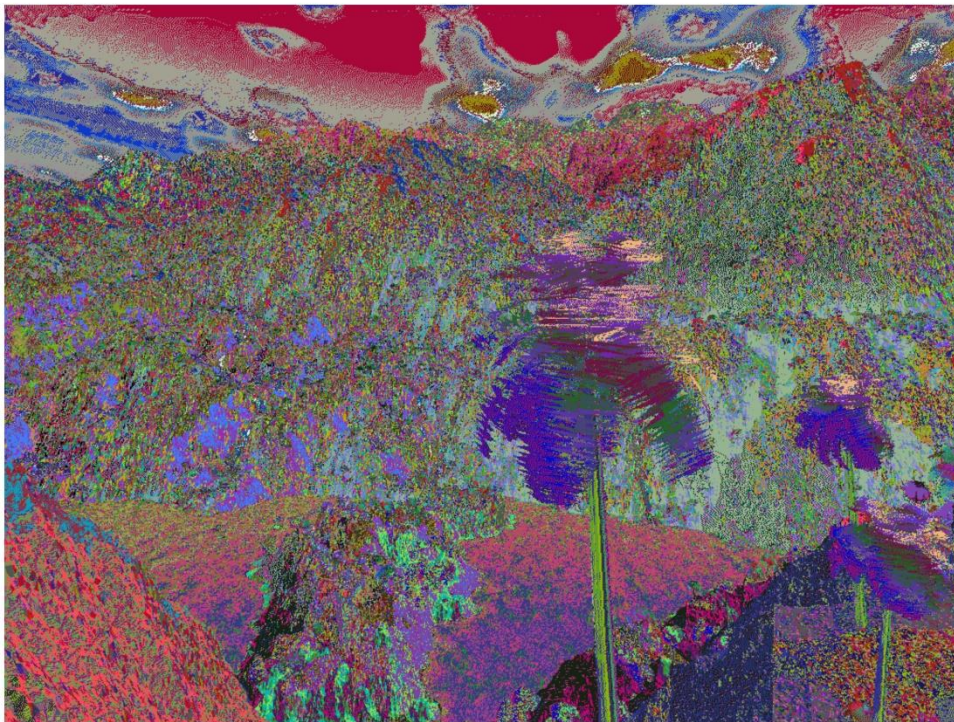
```

-----
Encrypting, please wait
-----

```

8. Finally, the encryption process is over, we can check the folder to find the encrypted.bmp file.

Finishsed encryption, please check the folder



Decryption

The decryption process is very similar to the encryption process.

Here is the step to step guide of the decryption process:

1. First, we will use 'python main.py' or 'python3 main.py' to launch the application:

```
C:\Users\yryuz\Documents\BCIT\Crypto\Assingment5>python main.py
```

2. Next, we will be choosing the operation mode. Since we want to decrypt the file this time, we will be choosing 2:

```
Please enter the operation mode (Choose the number of the mode):  
1.Encryption Mode  
2.Decryption Mode  
:2
```

3. The application then prompts us to enter the file type, we will pick 1 this time:

```
Please enter the file type (Choose the number of the file type):  
1.Image File (Only supporting BMP format)  
2.Text File  
:1
```

4. We have to pass the input and output file names/paths to the application when prompted:

```
Please enter the input file name/path: encrypted.bmp  
Please enter the output file name/path: decrypted.bmp
```

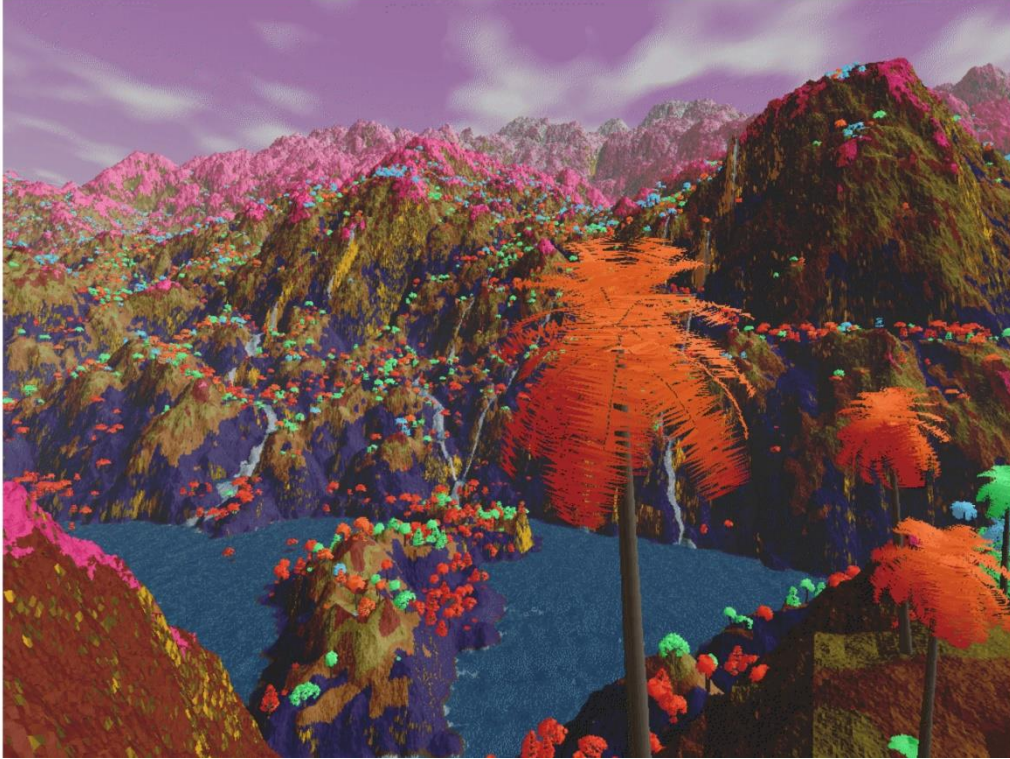
5. Last, we will enter the encryption key we used when encrypting the file and wait for the application to decrypt the file.

```
Please enter the encyrption key (Any integer number) to use:225
```

```
-----  
Decrypting, please wait  
-----
```

6. Once the application returns a 'Finished decryption' message, we can check the folder and find the decrypted.bmp file.

Finished decryption, please check the folder

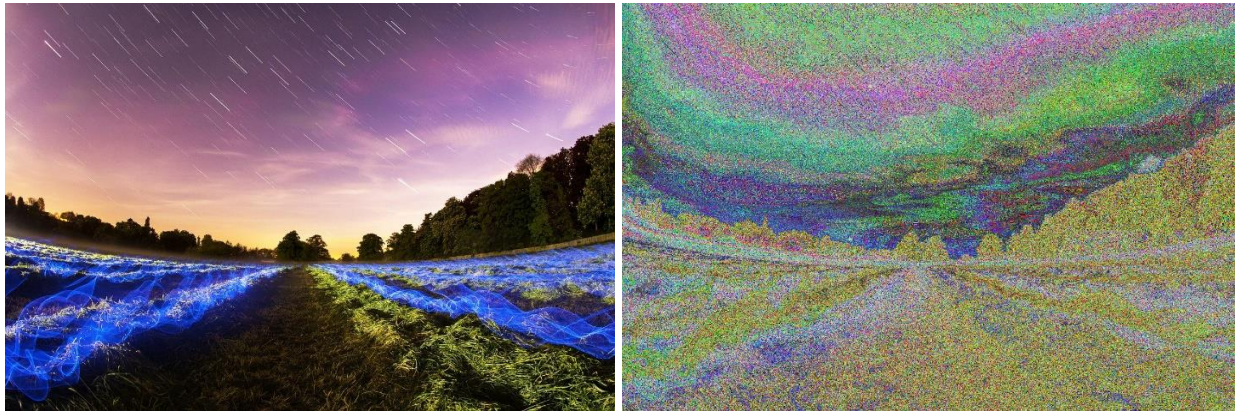


Analysis

ECB otherwise known as Electronic Codebook is one of the easier modes, where the plaintext is divided into pre-determined block sizes and then performs encryptions on them using the provided key. To decrypt, it's the same process.

Diffusion

One of the biggest issues with ECB is diffusion, more specifically the lack of it. Take a look at the following two images.

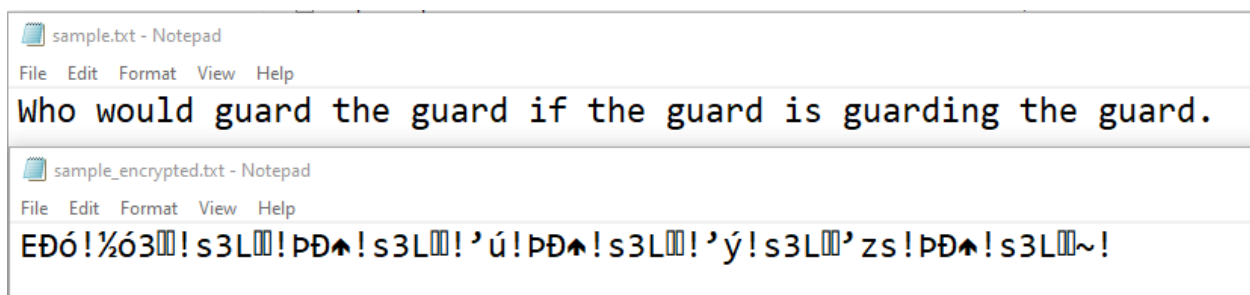


The image on the right is the original image and the one on the left is after encrypting it. We can see parts of the image match with one another, showing that ECB doesn't hide the pattern of the data very well.

Each individual pixel might be encrypted but the overall image can still be perceived since the pattern of colors in the original still remain in the encrypted. We can still see the general colors from the image on the left in the image on the right. This is more of a problem with the way the encryption mode, and there are better modes to use such as CBC, CTR or authenticated encryption modes like OCB or GCM.

Confusion

When looking at the confusion, I'm looking at the relationship between the ciphertext and key and whether it's possible to get the key from the ciphertext.



Here I used a key of 11, and it produces essentially random text, there's no way to get the key. An issue however is that if two messages and the key are the same, the ciphertext would

output the same both times. However, there's no way you can work back the key or encryption algorithm from just the ciphertext alone. Therefore, I would say that this algorithm has moderate confusion. Moderate because while implementing using a Feistel structure does improve the confusion, ECB is a bottleneck in terms of how much you can do with it. To improve the confusion, we could increase the block size or use other block mode ciphers as mentioned before such as CBC.

Avalanche Effect

The following is a table of bits changed in the plaintext as well as changes in the key for SPAC.

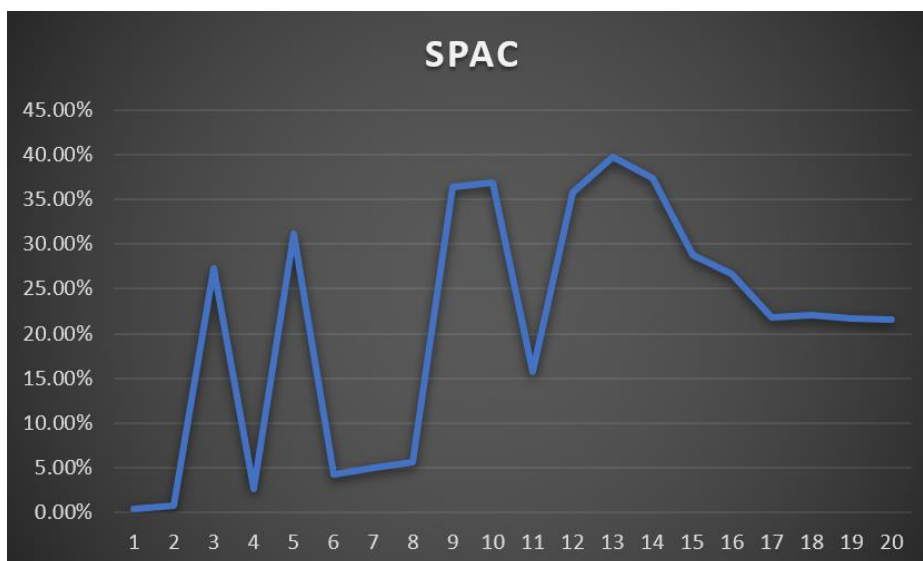
Round	Bits Changed in Cipher	Percentage Changed
1	3	0.40%
2	6	0.80%
3	204	27.35%
4	20	2.65%
5	232	31.10%
6	32	4.24%
7	37	4.96%
8	42	5.63%
9	269	36.45%
10	272	36.86%
11	117	15.66%
12	264	35.77%
13	290	39.78%
14	276	37.40%
15	215	28.78%
16	202	26.72%
17	163	21.82%
18	165	22.09%
19	162	21.69%
20	161	21.55%

As well as for SKAC.

Key	Bits Changed in Cipher	Percentage Changed
1	528	55.87%
2	471	53.40%
3	628	60.15%
4	377	46.54%
5	519	55.45%
6	346	54.15%
7	310	42.01%
8	431	51.49%

9	524	54.41%
10	487	53.58%

To get a clearer understanding, I've converted them to line graphs.



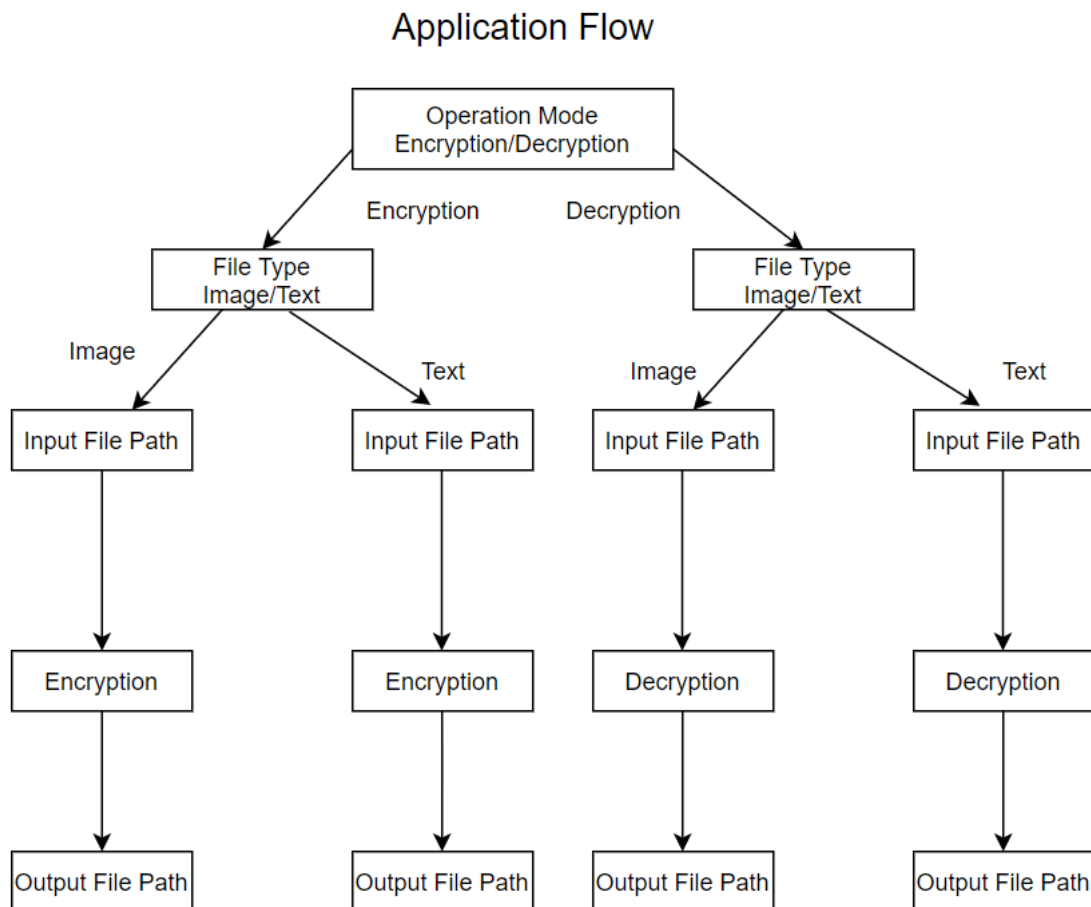
SKAC data was collected using the same plaintext but different keys (From 1-10). Meanwhile, SPAC data is from modifying plaintexts one at a time. It's hard to say whether the change percentage would remain stagnant if we went beyond round 20 for SPAC however it's still on the lower side. Compared to SKAC, which is consistently hitting 50% and only dropped to 40% in the number of bits changed once. I would say this is a somewhat desirable area to hit however there are still rooms for improvement.

Design

The design of the application is relatively straightforward.

Overall application flow

First, let us look at the high-level application flow:



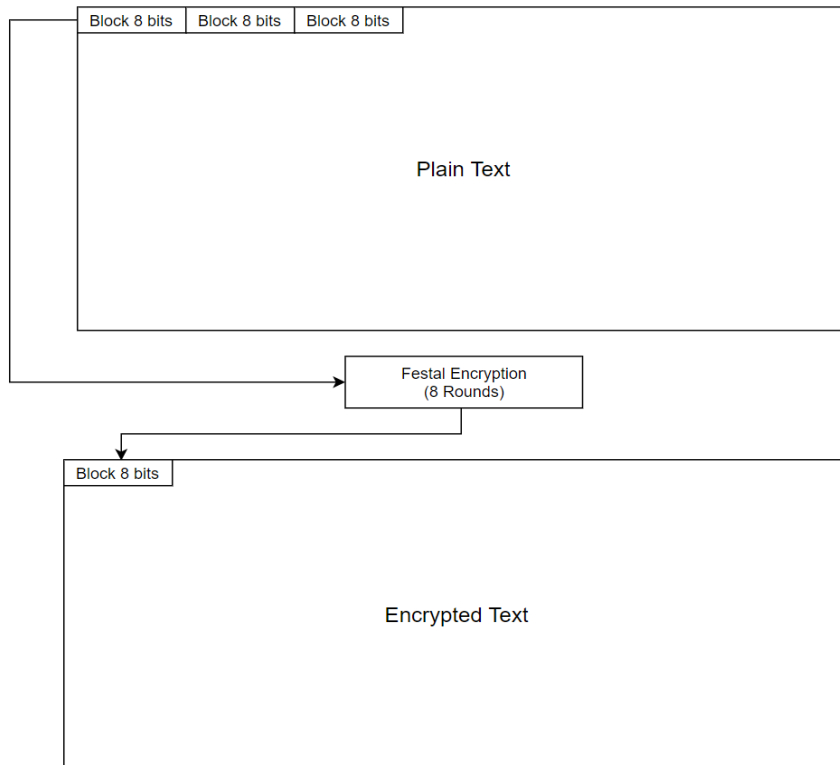
The application first prompts users for the operation mode which can be either encryption or decryption. Once the user has picked the operation mode, it asks for the file type of the input file; the reason is because the application performs different encryption and decryption processes based on the type of the file in order to avoid manipulating the original file header. After confirming the file type, the application will encrypt or decrypt the inputted file and save the encrypted or decrypted data as the output file.

Text file encryption design

Now, let us take a closer look at the encryption function designs. First, we have the text file encryption design.

When encrypting the plaintext file, the application starts by converting the input file into binary data. Next, since the encryption uses ECB mode (one of the Block cipher methods), the application splits the binary data into thousands of blocks. Each of the blocks passes into the Feistel encryption algorithm to get encrypted. Finally, the application will put all these encrypted blocks together to form the encrypted text file.

Text File Encryption (ECB Mode)



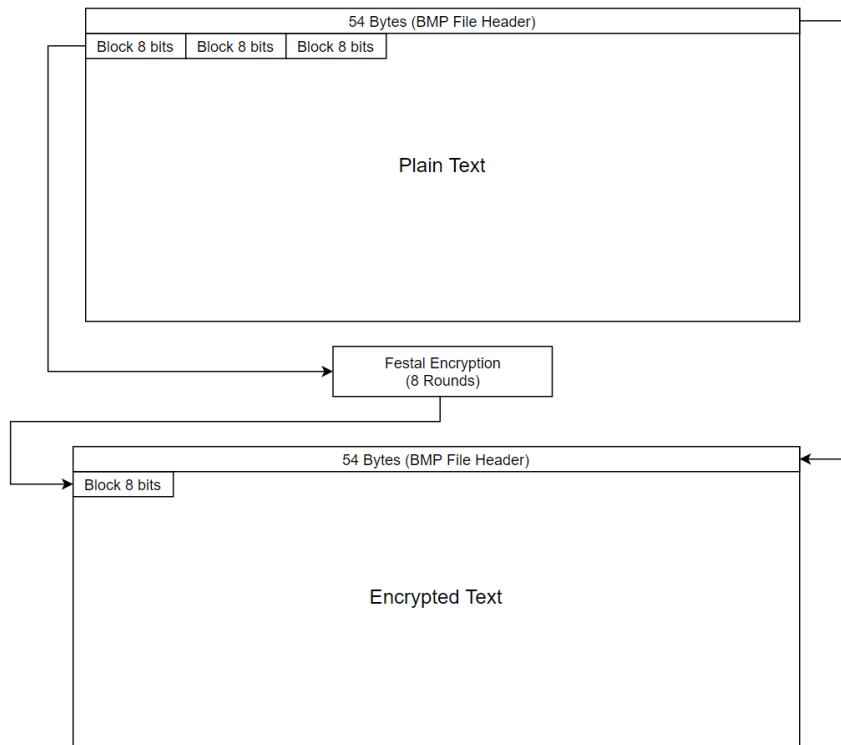
When we need to decrypt the file, the application will simply reverse the process, using the Feistel decryption algorithm to restore the original plaintext file.

BMP file encryption design

BMP is the only image file format that the application currently supports. The encryption process of a BMP file is almost identical to the text file encryption process we explained above.

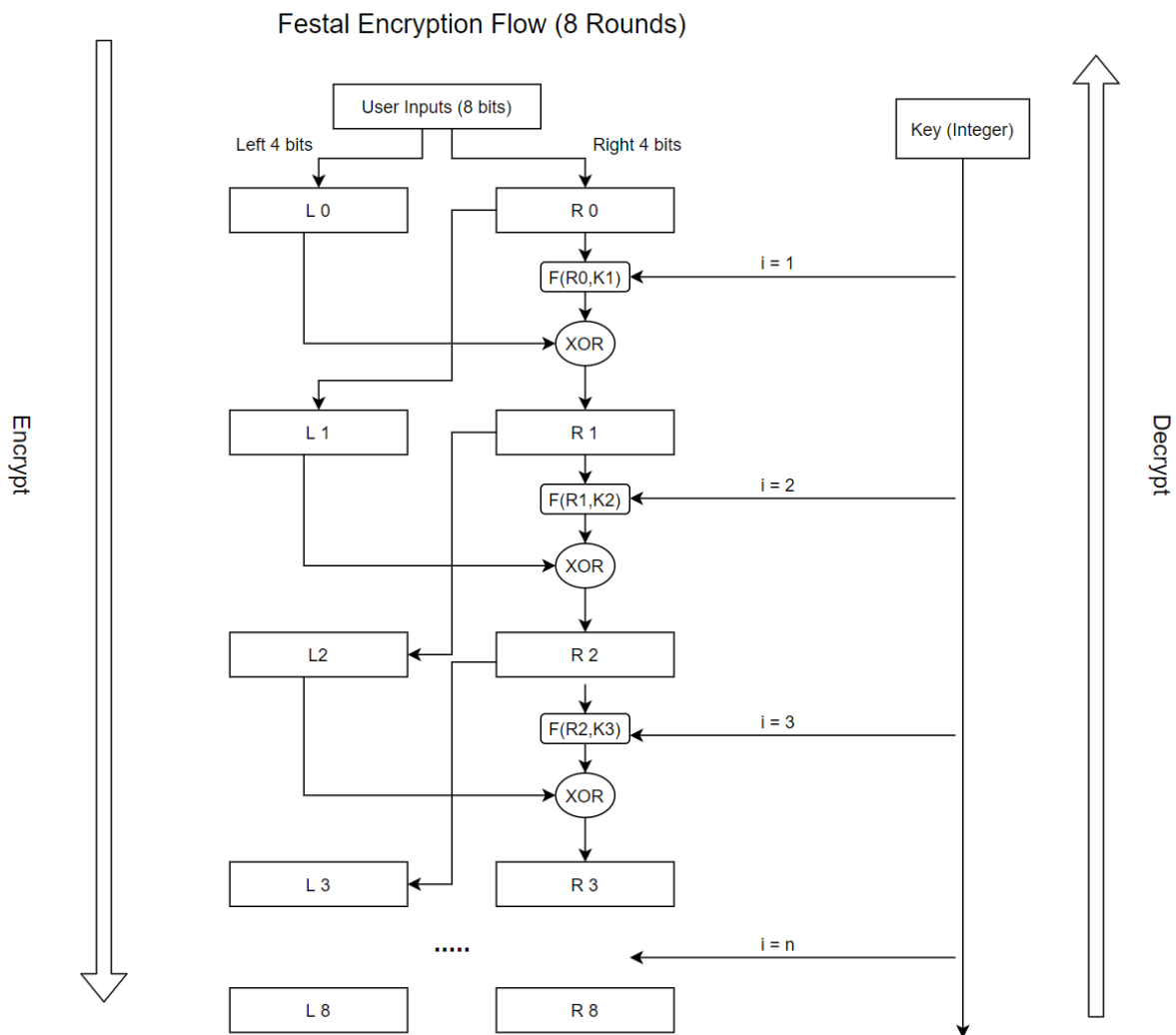
The only major difference is when the application encrypts or decrypts the input BMP file, it will skip over the first 54 bytes of the binary data. The reasoning behind it is because the first 54 bytes is the file header of BMP files; so it is unnecessary for the application to encrypt or decrypt the part, also it could cause file corruption when the application does encrypt or decrypt that part of the BMP files.

BMP File Encryption (ECB Mode)



Feistel encryption flow

As we mentioned before, each block of the plaintext binary data is encrypted through the Festal encryption algorithm. So now let us look at the Fest encryption algorithm the application uses.



Scrambling Function:

$$f_i(x, k) = [(2i * k)^x] \bmod 15$$

Where i is the number of the round; and k is the encryption key (Any integer)

Like any other Festal encryption algorithms, the application first splits the plaintext binary block into two parts the left part - L and the right part – R. Next, the application will add the value in the current round of R to the next round of L. At the same time, it will also pass the value in the current round of R to a scrambling function, using the result from the scrambling function to perform a XOR comparison against the value in the current round of L. The output of the XOR comparison will then be added to the next round of R.

To summarize the Feistel encryption process of each round, we can use the following two equations:

$$L_i = R_{i-1}$$

$$R_i = L_i \oplus F(R_{i-1}, K_i)$$

Currently, the application is configured to perform 8 rounds of the Feistel encryption process, but it can be changed anytime.

The decryption process is also very straightforward, the application will simply reverse the above Feistel encryption processes to decrypt the encrypted files.

Testing

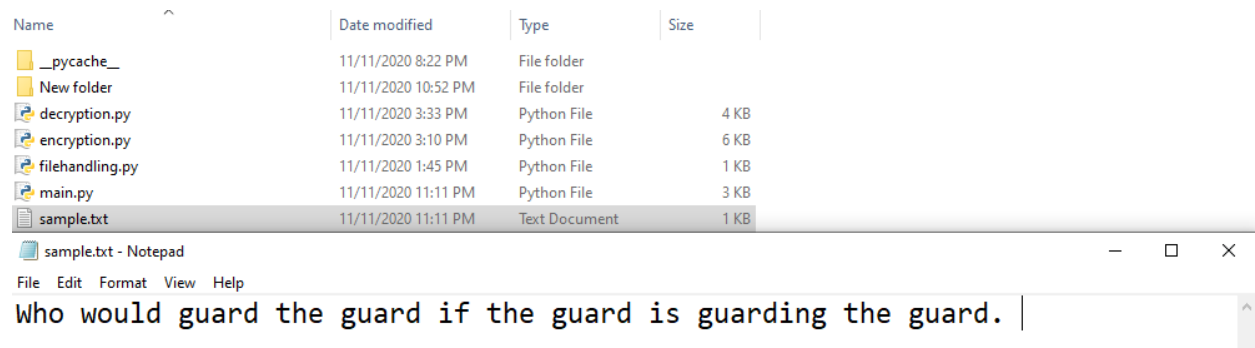
My setup for testing is a Windows 10 computer, where I'll be using the terminal to run the tests from. You can view all the tests in the 'Videos' folder as well.

Test #	Tools Used	Expectation	Actuality	Result
1	Command Prompt	I should be able to encrypt the text file using a key that I can choose	I was able to encrypt the text file using a key I chose	PASS
2	Command Prompt	I should be able to decrypt the encrypted text from Test #1 using the same key	I was able to decrypt the text file from Test #1 using the same key	PASS
3	Command Prompt	I shouldn't be able to decrypt the encrypted text from Test #1 using a different key	I wasn't able to decrypt the text file from Test #1 using the key that wasn't used for the encryption originally	PASS
4	Command Prompt	I should be able to encrypt the image file using a key that I can choose	I was able to encrypt the images using a key I chose	PASS
5	Command Prompt	I should be able to decrypt the image file from Test #4 using the same key	I was able to decrypt the images from Test #4 using the same key	PASS
6	Command Prompt	I shouldn't be able to decrypt the encrypted image from Test #4 using a different key	I wasn't able to decrypt the images from Test #4 using the key that wasn't used for the encryption originally	PASS

Test 1

For this test, I should be able to encrypt the text file using a key that I can choose.

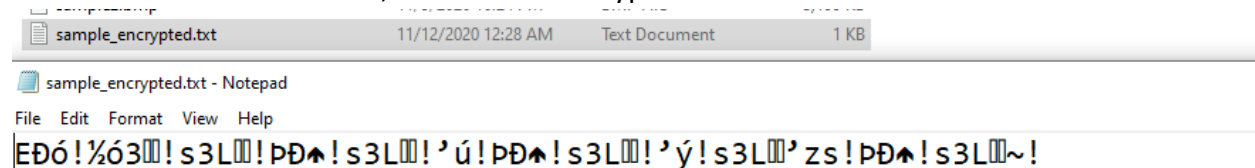
I'll be using a text file called sample.txt with the following line in it, "Who would guard the guard if the guard is guarding the guard."



After running the python script, I choose encryption(1), Text File (2) and named the output sample_encrypted.txt using an encryption key of 11.

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:1
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:2
Please enter the input file name/path: sample.txt
Please enter the output file name/path: sample_encrypted.txt
Please enter the encryption key (Any integer number) to use: 11
-----
Encrypting, please wait
-----
Encryption has finished, please check the folder
```

And if we view the file made, it should be encrypted which it is.



The file has been successfully encrypted.

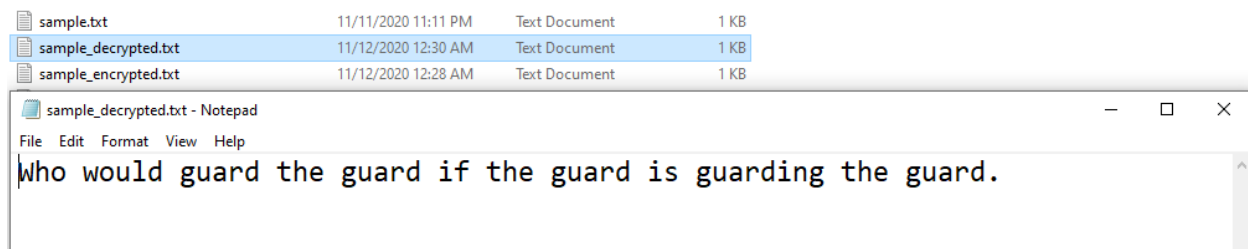
Test 2

For this test, I should be able to decrypt the encrypted text from Test #1 using the same key that I used which was 11. I'll be decrypting the file created before, sample_encrypted.txt.

After running the python script, I choose the decrypt option (2) and text (2). I then entered the encrypted file name and named the output file sample_decrypted.txt along with key 11.

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:2
Please enter the input file name/path: sample_encrypted.txt
Please enter the output file name/path: sample_decrypted.txt
Please enter the encryption key (Any integer number) to use: 11
-----
Dec decrypting, please wait
-----
Decryption has finished, please check the folder
```

If we open up sample_decrypted.txt, we get the following.



Showing that we've successfully been able to decrypt the cipher using the same key.

Test 3

For Test 3, I shouldn't be able to decrypt the encrypted text from Test #1 if I use a key that is different from the one used to originally encrypt it. I'll be using 5, 13, and 18 as the random keys for this scenario which I'll be applying to the sample_encrypted.txt file. I'll be showing the image of the decryption process and then the contents of the text file afterwards for all 3 keys.

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:2
Please enter the input file name/path: sample_encrypted.txt
Please enter the output file name/path: sample_decrypted5.txt
Please enter the encryption key (Any integer number) to use: 5
```

Decrypting, please wait

Decryption has finished, please check the folder

File Name	Date/Time	File Type	Size
sample.txt	11/11/2020 11:11 PM	Text Document	1 KB
sample_decrypted.txt	11/12/2020 12:30 AM	Text Document	1 KB
sample_decrypted5.txt	11/12/2020 12:47 AM	Text Document	1 KB
sample_decrypted13.txt	11/12/2020 12:47 AM	Text Document	1 KB
sample_decrypted18.txt	11/12/2020 12:49 AM	Text Document	1 KB

sample_decrypted5.txt - Notepad

File Edit Format View Help






ûRo½<ocV^½gcdp^½²RÐ½gcdp^½v½²RÐ½gcdp^½v½gcdp^v½g½²RÐ½gcdp^½


Ln 1, Col 1

```

Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:2
Please enter the input file name/path: sample_encrypted.txt
Please enter the output file name/path: sample_decrypted13.txt
Please enter the encryption key (Any integer number) to use: 13
-----
Dec decrypting, please wait
-----
Decryption has finished, please check the folder

```

	sample.txt	11/11/2020 11:11 PM	Text Document	1 KB
	sample_decrypted.txt	11/12/2020 12:30 AM	Text Document	1 KB
	sample_decrypted5.txt	11/12/2020 12:47 AM	Text Document	1 KB
	sample_decrypted13.txt	11/12/2020 12:47 AM	Text Document	1 KB
	sample_decrypted18.txt	11/12/2020 12:49 AM	Text Document	1 KB

 sample_decrypted13.txt - Notepad

File Edit Format View Help

```

|--À~.ÀÌ©;~ÈÌ«
;~G-~ÈÌ«
;~Žü~G-~ÈÌ«
;~Ž*~ÈÌ«
;Ž'È~G-~ÈÌ«
ig~

```

```

Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:2
Please enter the input file name/path: sample_encrypted.txt
Please enter the output file name/path: sample_decrypted18.txt
Please enter the encryption key (Any integer number) to use: 18
-----
Dec decrypting, please wait
-----
Decryption has finished, please check the folder

```

sample.txt	11/11/2020 11:11 PM	Text Document	1 KB
sample_decrypted.txt	11/12/2020 12:30 AM	Text Document	1 KB
sample_decrypted5.txt	11/12/2020 12:47 AM	Text Document	1 KB
sample_decrypted13.txt	11/12/2020 12:47 AM	Text Document	1 KB
sample_decrypted18.txt	11/12/2020 12:49 AM	Text Document	1 KB

sample_decrypted18.txt - Notepad

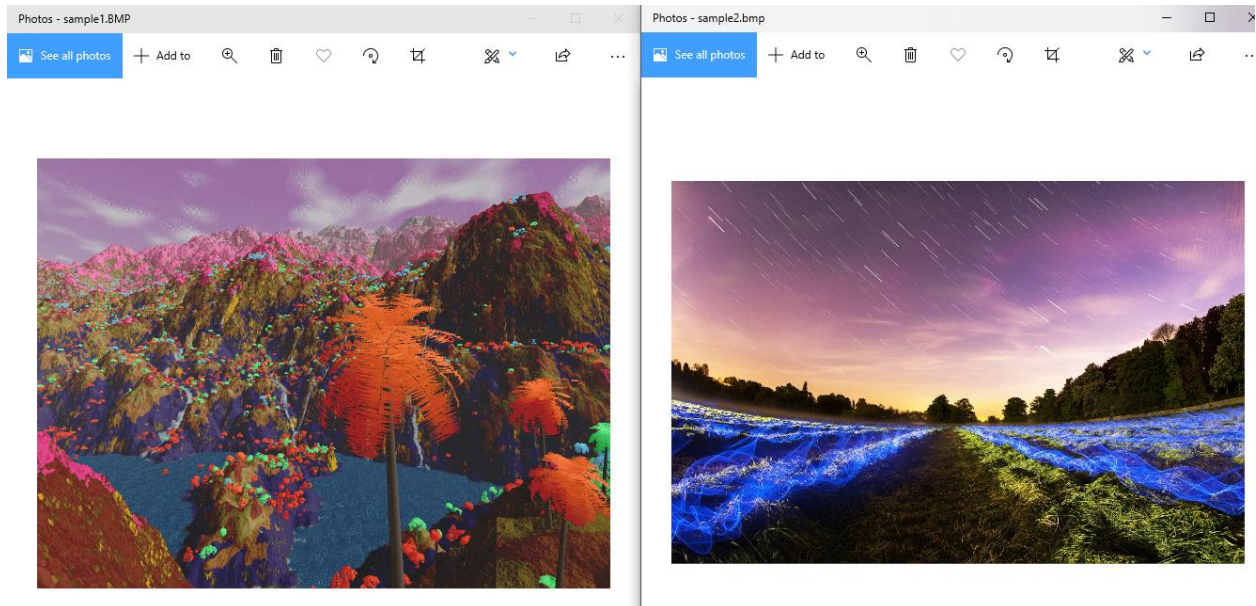
File Edit Format View Help

~Ž@~B@Ÿš'~iŸ¢|'~qžs~iŸ¢|'~µæ~qžs~iŸ¢|'~µF~iŸ¢|'µ>i~qžs~iŸ¢|'Ű~

We can see for each scenario that the decrypted text was no where close to what the original pre-encrypted text was therefore showing that you can only decrypt the text if you have the correct key and the success of this test.

Test 4

For this test, I should be able to encrypt an image file using a key that I can choose. I'll be using the following two images labeled sample1 and sample2 in bmp format. Here is how they originally look.



Using an encryption key of 9 for sample1 and key of 23 for sample2, I did the following for the two images.

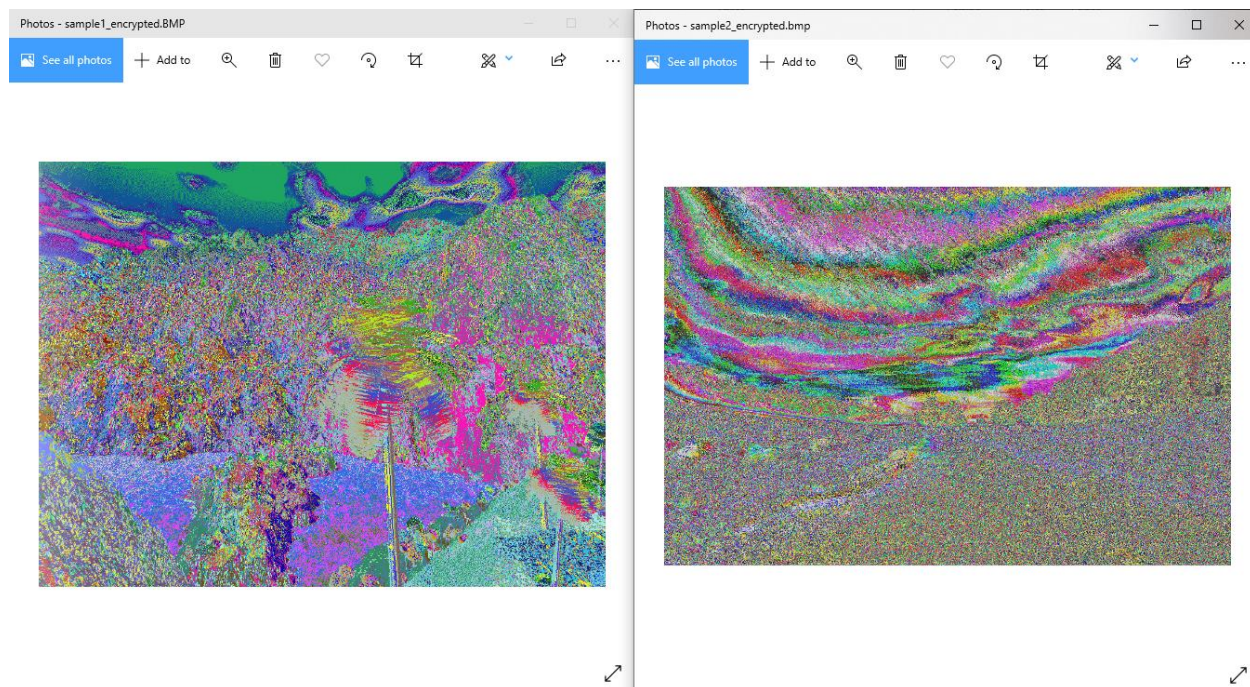
```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:1
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample1.BMP
Please enter the output file name/path: sample1_encrypted.BMP
Please enter the encryption key (Any integer number) to use: 9
-----
Encrypting, please wait
-----
54
Encryption has finished, please check the folder
```

```

Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:1
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample2.bmp
Please enter the output file name/path: sample2_encrypted.bmp
Please enter the encryption key (Any integer number) to use: 23
-----
Encrypting, please wait
-----
54
Encryption has finished, please check the folder

```

If we view the two images, they should be different from the original.



Due to the poor nature of ECB, we can still partially see the image however it's been encrypted to the best of the cipher ability.

Test 5

Using the encrypted images from Test #4, I should be able to decrypt them using the same key I used to encrypt them.

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample1_encrypted.BMP
Please enter the output file name/path: sample1_decrypted.BMP
Please enter the encryption key (Any integer number) to use: 9
```

```
-----
Decrypted, please wait
-----
```

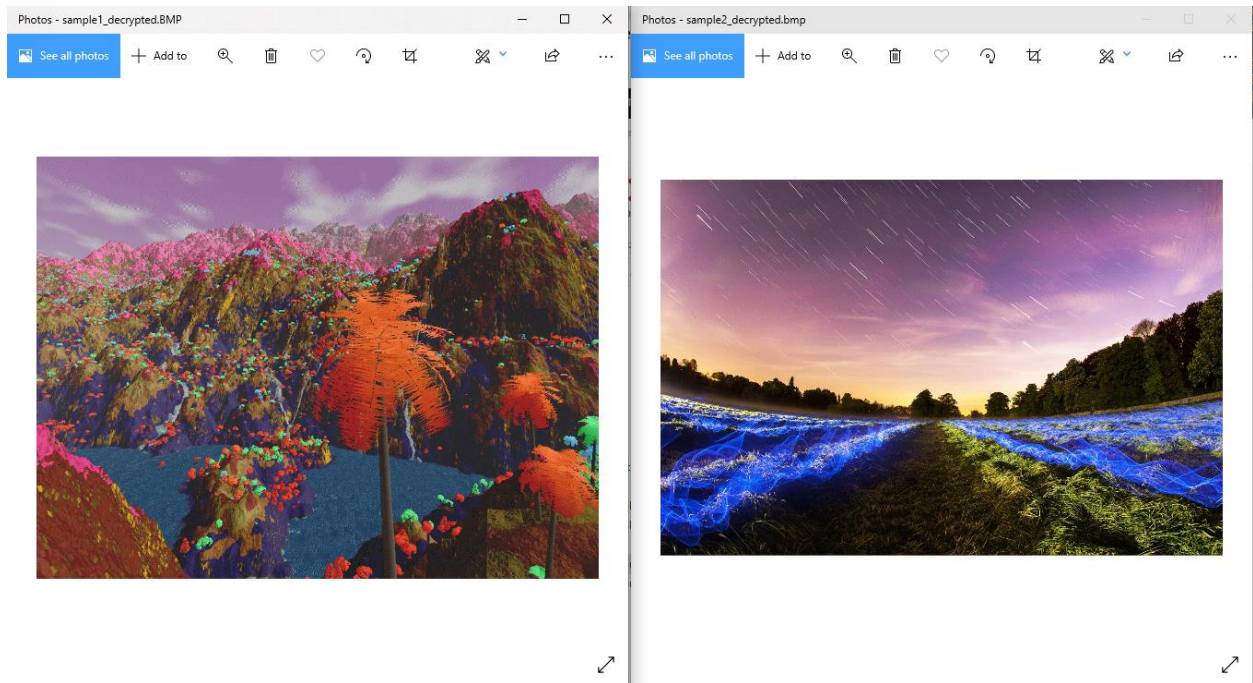
```
54
Decryption has finished, please check the folder
```

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample2_encrypted.bmp
Please enter the output file name/path: sample2_decrypted.bmp
Please enter the encryption key (Any integer number) to use: 23
```

```
-----
Decrypted, please wait
-----
```

```
54
Decryption has finished, please check the folder
```

If I go to the folder, both images should return to their original form.



Which they are, showing that decryption works on images.

Test 6

For this test, I shouldn't be able to decrypt the encrypted image from Test #4 using a key that's different from the one I used when encrypting them. The keys I used were 9 for sample1 and 23 for sample2. To prove that the decryption won't work, I'll be using key 5 and 21 for both images. I'll show the process of decryption followed by an image of the picture.

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample1_encrypted.BMP
Please enter the output file name/path: sample1_decrypted5.BMP
Please enter the encryption key (Any integer number) to use: 5
```

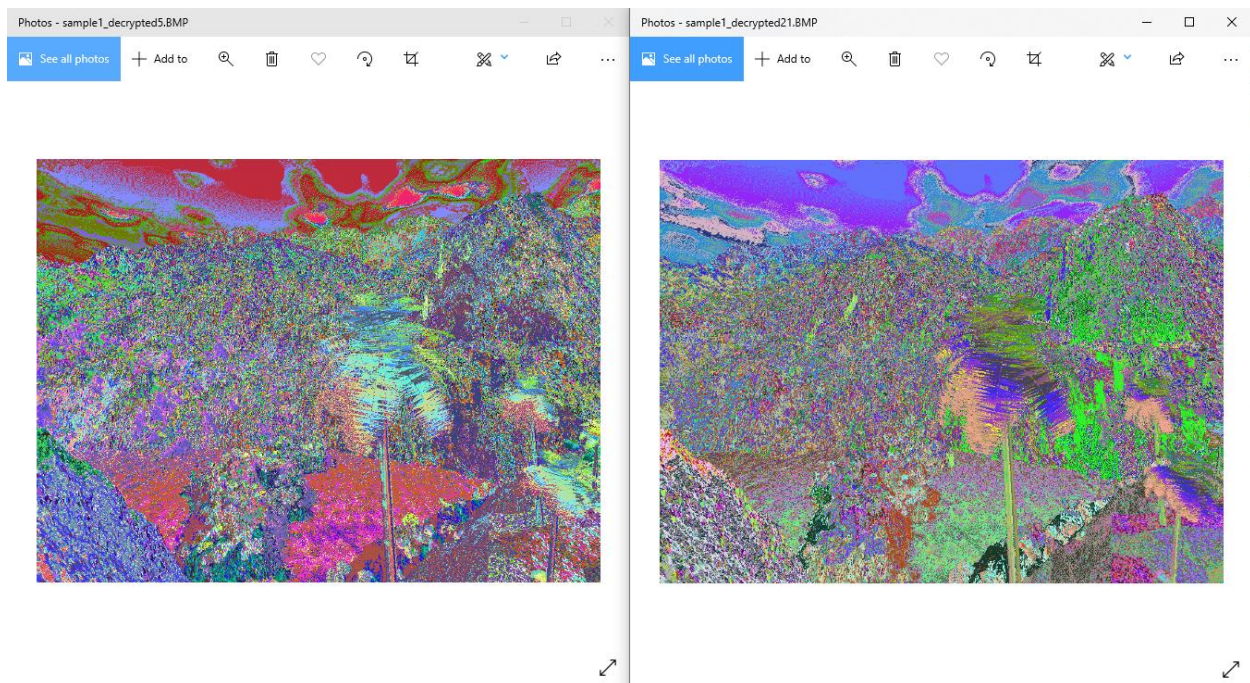
```
-----
Decrypted, please wait
-----
```

```
54
Decryption has finished, please check the folder
```

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample1_encrypted.BMP
Please enter the output file name/path: sample1_decrypted21.BMP
Please enter the encryption key (Any integer number) to use: 21
```

```
-----
Decrypted, please wait
-----
```

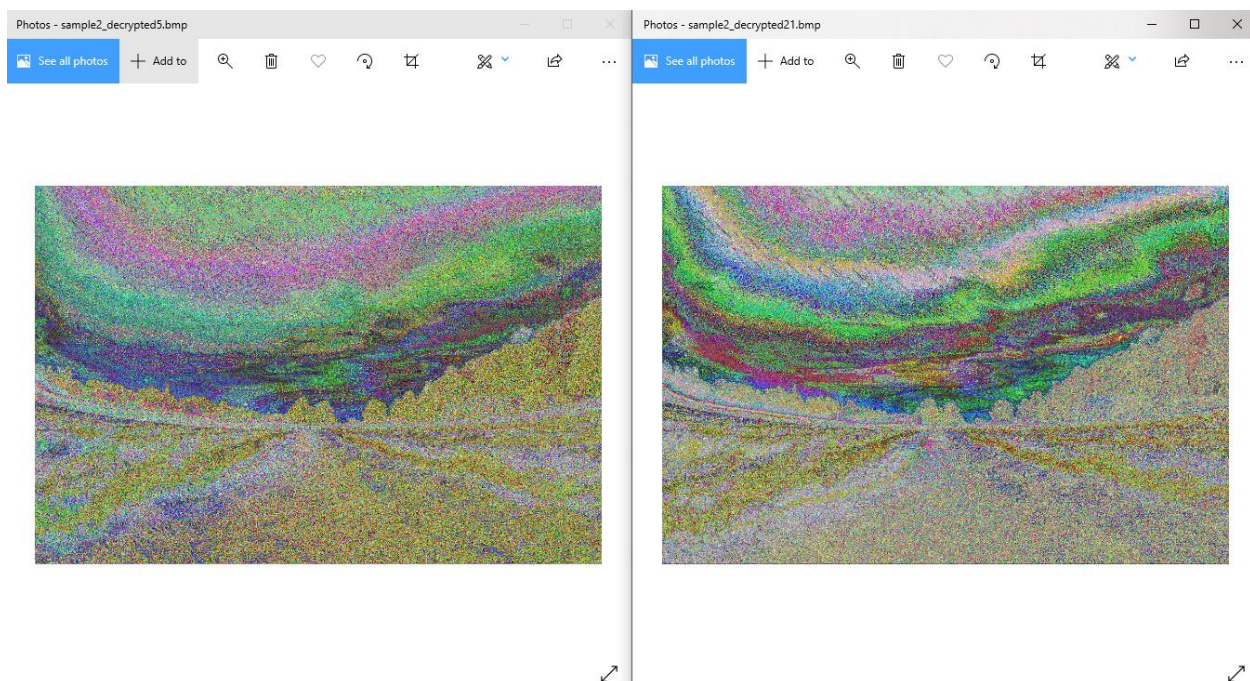
```
54
Decryption has finished, please check the folder
```



We can see that while the images has changed, it hasn't been decrypted (Returned to the original image).

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample2_encrypted.bmp
Please enter the output file name/path: sample2_decrypted5.bmp
Please enter the encryption key (Any integer number) to use: 5
-----
Decrypting, please wait
-----
54
Decryption has finished, please check the folder
```

```
Please enter the operation mode (Choose the number of the mode):
1.Encryption Mode
2.Decryption Mode
:2
Please enter the file type (Choose the number of the file type):
1.Image File (Only supporting BMP format)
2.Text File
:1
Please enter the input file name/path: sample2_encrypted.bmp
Please enter the output file name/path: sample2_decrypted21.bmp
Please enter the encryption key (Any integer number) to use: 21
-----
Dec decrypting, please wait
-----
54
Decryption has finished, please check the folder
```



Similar to the first image, this image is also still encrypted. This shows that we cannot get the original image back without using the same key used for the original encryption.