بسم الله الرحمن الرحيم

Asem Jamil Ishtayah – 11316990

HW #2: Steganography

Computer Engineering Department

Information and network security

Supervisor:

Dr. Abdallah Hasan Rashed

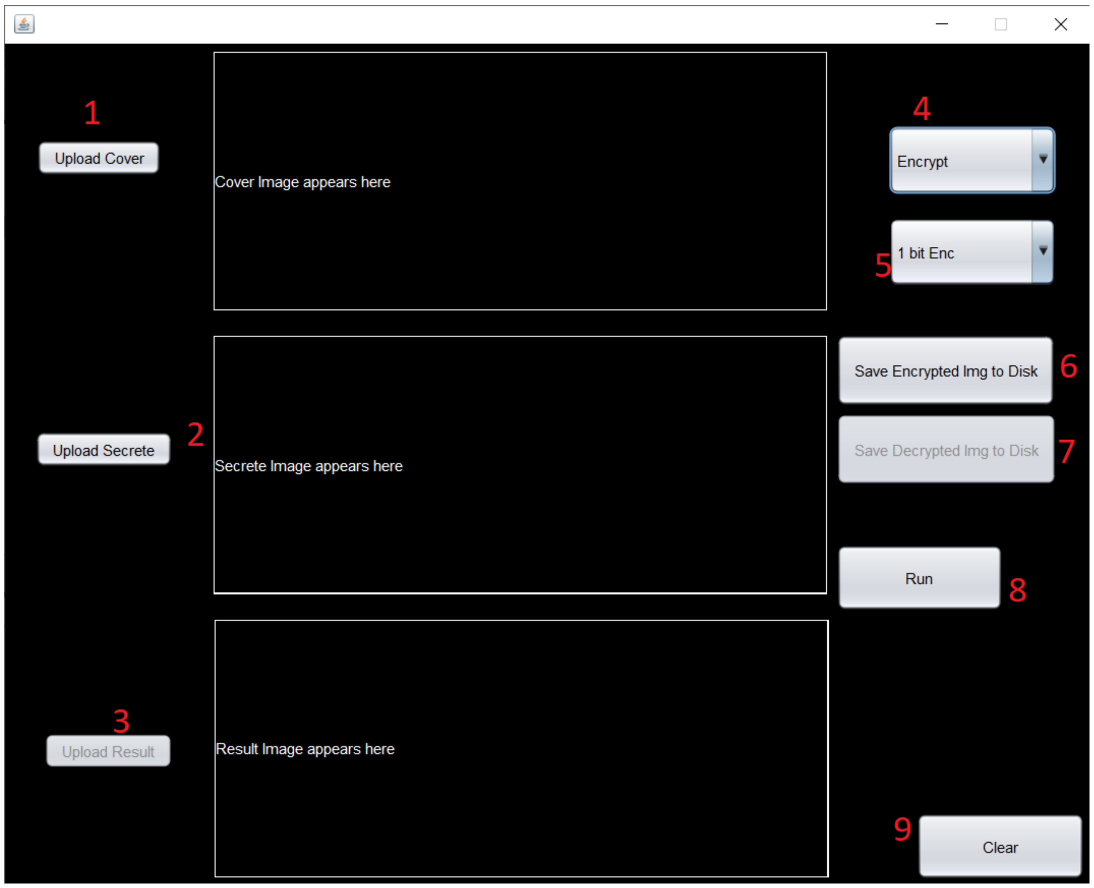
Introduction:

In this project we will implement a Steganography task, to protect some data through cover them with another image. The project was done using java programing language, with Apache netbeans 13 IDE. And I push all of source code to github, and from github repository we can see approx. all of progress.

I suggest before start reading to show the demo video.

Procedure:

1. GUI:


1
Upload Cover
Cover Image appears here
4
Encrypt
5
1 bit Enc
Save Encrypted Img to Disk
6
2
Upload Secrete
Secrete Image appears here
Save Decrypted Img to Disk
7
Run
8
3
Upload Result
Result Image appears here
9
Clear

#1: upload cover and cover image area, chose an image by pressing, and will appear on cover image area.
#2: secrete image: the same of cover image procedure.
#3: result image: the same of cover image procedure.

#4: Combo Box: to choose what you will run. **Encrypt** for cover secrete image, and when run, will show result on result image area. And the input will be from secrete and cover areas. **Decrypt** to extract what hidden on result image area.
Note that: secrete and result images will appear on their fields regardless of what case we on (enc or dec).
And note that the app allow you only to upload what respect to operation mode. (enc allow to upload cover and secrete images only, dec allow to upload result image only).
#5: Combo Box to choose how much bits want to store on each rgb layers, ex. 3 bit, will store at LSB for red layer 3 bit from secrete image the same for g, and what remained on b.
#6: after you run the operation, you can save encrypted image (image contain secrete image) to disk.
Will saved at the project directory folder ("./").
#7: same of #6 but for decrypted image (secrete)
#8: Run what you choose from operation (enc/dec) and number of bits (1,2,3).
#9: Clear, and restart the app as you didn't touch it.

2. Encryption: Hide secrete image on cover image.

When we press run, the program checks what in combo lists, if encryption will check how much bits will store on each color layer.
If 1 bit:

```
        r = (r&254) + arrSecrete[n][m]%2;
        arrSecrete[n][m] = arrSecrete[n][m]/2;

        g = (g&254) + arrSecrete[n][m]%2;
        arrSecrete[n][m] = arrSecrete[n][m]/2;

        b = (b&254) + arrSecrete[n][m]%2;
        arrSecrete[n][m] = arrSecrete[n][m]/2;
```

254 equlas on binary: 1111 1110 → to reset first bit to 0.
%2 to extract first bit
/2 to shift right 1 bit (prepare the next layer and iteration).
Doing this for each secrete byte (grayscale pixel) 3 iterations (3 pixels cover per 1 pixel secrete), last iteration only r and g (grbgrbgr).

If 2 bits:

```
r = (r&252) + arrSecrete[n][m]%4;
arrSecrete[n][m] = arrSecrete[n][m]/4;

g = (g&252) + arrSecrete[n][m]%4;
arrSecrete[n][m] = arrSecrete[n][m]/4;

b = (b&252) + arrSecrete[n][m]%4;

arrSecrete[n][m] = arrSecrete[n][m]/4;
```

252 equlas on binary: 1111 1100 → to reset first 2 bits to 0.
%4 to extract first 2bits
/4 to shift right 2 bit (prepare the next layer and iteration).
Doing this for each secrete byte (grayscale pixel) 2 iterations (2 pixels cover per 1 pixel secrete), last iteration only r(rrbbggrr).

If 3 bits:

```
r = (r&248) + arrSecrete[n][m]%8;
arrSecrete[n][m] = arrSecrete[n][m]/8;

g = (g&248) + arrSecrete[n][m]%8;
arrSecrete[n][m] = arrSecrete[n][m]/8;

b = (b&248) + arrSecrete[n][m]%8;
arrSecrete[n][m] = arrSecrete[n][m]/8;
```

248 equlas on binary: 1111 1000 → to reset first 3 bits to 0.
%8 to extract first 3bits
/8 to shift right 3 bit (prepare the next layer and iteration).
Doing this for each secrete byte (grayscale pixel) only one iteration (1 pixel cover per 1 pixel secrete), blue layer only 2 bits(bbgggrrr).

3. Decryption: Extract secrete image from cover image.

If first combo box set to decrypt, meaning that we will give an image contain secrete image, and want to extract secrete.
The app will fill arrSecrete[200][300] array from the given image.
Will depends on how much it was bits encrypted.
If 1:

```
arrSecrete[n][m] = (r&1) + (g&1)*2 + (b&1)*4; // 1st pixel
arrSecrete[n][m] = arrSecrete[n][m] + (r&1)*8 + (g&1)*16 + (b&1)*32; // 2nd pixel
arrSecrete[n][m] = arrSecrete[n][m] + (r&1)*64 + (g&1)*128; // 3rd pixel
```

- color_layer&1: anding whith 0000 0001, to extract only first bit.
- Multiply for significant of this layer, depended on how encrypted.

g	r	b	g	r	b	g	r
g0000000	xr000000	xxb00000	xxxg0000	xxxxr000	xxxxxb00	xxxxxxg0	xxxxxxxb

If 2:

```
arrSecrete[n][m] = (r&3) + (g&3)*4 + (b&3)*16;
arrSecrete[n][m] = arrSecrete[n][m] + (r&3)*64;
```

- color_layer&3: anding whith 0000 0011, to extract only first 2bits.

If 3:

```
arrSecrete[n][m] = (r&7) + (g&7)*8 + (b&7)*64;
```

- color_layer&7: anding whith 0000 0111, to extract only first 3bits.

4. **Extra work:**
I built my app to accept any cover or secrete image format, size and color system the app will save cover image size and color. The app will resize secrete to 300x200, and change color to grayscale. This feature will allow the app to deal with any any any image 😍.

Conclusion:

On this app we apply Steganography theory to cover any secrete image, and to restore secrete from any image contained.