**EFFICIENT STEGANOGRAPHIC TECHNIQUE USING SECURED DATA HIDING WITH BIT STREAM DATA TRANSFER**

**Description**

In the efficient steganographic techniques using secured data hiding ,we propose dual image with data hiding , but in the current sceneries there is a single image with data hiding which does not provided satisfactory security.

Planning to build the web based application using VB.Net

**Technology details**

VB.NET, C#, Micro soft SQL

**Excepted output of project**

The user can send the data by hiding the data with the dual image steganography with the key and the receiver can extract the data which hidden by the sender if only known the key image and key value.

**Screen details**

1. Authentication
2. Pre processing in cover image
3. Adding text with key to second image
4. Encryption and compression
5. Decompression and decryption
6. Exacting image and text

**In preprocessing**

1. **Image verification**

This is the intro module that contains the input methodology, which gets the image as input and text for hiding. The image should be in bitmap format, this is because bitmap naturally have the capacity of handling the pixel flexibility. So we are using bitmap format here. Here we want to initialize the original file to the embedded and the key file which use to embed the original file with the secret document. The original file is no more needed after the process; this is because a new file will be generated after the process.

**Adding text with key to second image**

A key image will be given as input, this key image act as a symmetric key. With the help of the symmetric key the document will be hided inside the image and the key will be converted into frames. With the converted frames a new image will be generated, the generated new image will can be stored in the user defiled area. With the new generated image the doc will be scarce into pixels, so the other people can’t able to see the document embedded in to the image. We can use the same key file to the extraction process also.

**Encryption and compression**

While hiding the text, the text will be converted into pixels and scarce inside the image. This process will be done according to pixels and the color of the pixels mentioned in the images. Usually high resolution images will take longer time to do this process. This is because pixel ratio will be differing from high resolution image to low resolution image. After that the key file will be taken from the image (i.e.) pixels from the image . And the next process will be triggered. Here we using the table is Encryption\_table and Compression\_table

**Decompression and decryption**

In this module the scarce pixels will be retrieved with the help of the key image and again roll back as the image format. Here user wants to specify the correct location where the stegano image wants to be stored. Here we using the table is Decryption\_table and Decompression\_table.

**Text and image extraction**

This Module will finalize the process. Here the text and the image will be extracted separately. This process will also do according to the key image. So user can finally view the hidden.

**Table Description**

PK Primary key

Img Image

**1.Encryption \_table:**

Img id – int [PK]

Img name – nchar(50)

Img path – nchar(50)

User id – int

**2. Compression\_table:**

Compression name – nchar(50)

Img name – nchar(50)

User id – int [PK]

Compression size – int

Compression path – nchar(50)

**3.Decompression \_table:**

Decompression name – nchar(50)

Img name– nchar(50)

User id – int [PK]

Decompression size – int

Decompression path – nchar(50)

**4.Decryption \_table:**

Img id – int [PK]

Img name – nchar(50)

Img path – nchar(50)

User id – int

**Testing**

Following this step a variety of tests are conducted.

* + Unit testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case no** | **Description** | **Actual result** | **Expected Result** | **Result** |
| 1. | Test for all cache responses. | All cache responses should be in the approximate value around 28.9 ms | All cache responses should be in the approximate value around 28.9ms | Pass |
| 2. | Test for various responses | The result after execution should give the accurate result | The result after execution should give the accurate result | Pass |