**Approach**

The basic motive behind this project was to see if the concepts of visual cryptography can be extended to encryption and decryption of confidential data by means of embedding it in the image.

The scheme is as follows:

We first declare six fundamental blocks that are used as a basis for expanding the white and black pixels of the original image.



Each pixel in the original image is mapped to a block of 2x2 dimensions.

If the original pixel is white, two out of the four corresponding pixels are white and two are black while if the original pixel is black, all four parts are black.

The rationale behind doing this is that when the two shares are superimposed and the original image revealed, the overall appearance of the image should be discernable. For that the number of black elements in each black block should be greater than that of each white block.

This means that the two shares should be so created that on superimposition they give an appropriate block on the basis of whether the original pixel was white or black.



The above pairs are combinations whose stacking gives a white block (two black and two white parts).



These pairs represent the combinations whose stacking gives a black box (all four black parts).

The added advantage of this scheme is that it encodes confidential data into the shares.

Let the confidential data be a string of digits.

D=d1,d2,d3,d4,d5,d6,d7,d8,d9……

Here each di represents a digit in the range of 0 to 5.

This is done so that each of these can map to one of the six fundamental blocks discussed above.

To embed this data into the transparencies, we split each di into xi and yi such that:

**(xi + yi) = di mod 6**

To do this we randomly select and xi in the range 0 to 5 and then choose the corresponding yi in the range 0 to 5 that satisfies the above relation.

Let us assume that the data is 50342514…..

Some of the possible values of xi and yi can be given by:

5 = 2 + 3 mod 6

0 = 4 + 2 mod 6

3 = 5 + 4 mod 6

4 = 5 + 5 mod 6

2 = 5 + 3 mod 6

5 = 0 + 5 mod 6

1 = 3 + 4 mod 6

4 = 2 + 2 mod 6

Now that we have the values of xi and yi we can start generating the transparencies.

We take each xi and yi and interleave them in the transparencies as follows:

Transparency 1: 2 \_ 4 \_ 5 \_ 5 \_ 5 \_ 0 \_ 3 \_ 2 \_

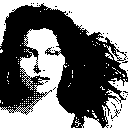
Transparency 2: \_ 3 \_ 2 \_ 4 \_ 5 \_ 3 \_ 5 \_ 4 \_ 2

The \_ marks are computed on the basis of the original image pixels.

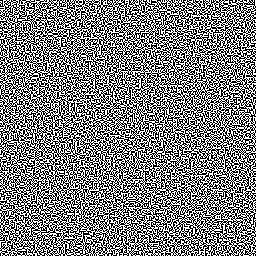
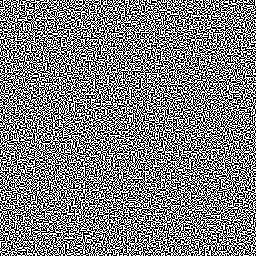
For instance, if the first pixel is white we need a block in share 2 that when superimposed with ‘2’ in share 1 gives a white block. We see from the tables above that we need to use another 2 for the purpose. If the second pixel is black, we need a block in share 1 that when superimposed with ‘3’ in share 2 gives a black block. This also turns out to be a 2 (refer the table for the computation).

While decoding, directly stacking the two shares will reveal the image and the confidential data can be extracted by the formula xi + yi = di. Here, xi is the block type of the (2i−1)th block of transparency 1, and yi is the block type of the (2i)th block of transparency 2.

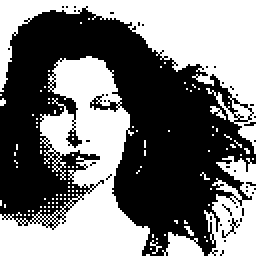
In case the attacker gains access to both share 1 and share 2 it is easy for him to decipher the image and obtain the confidential data. To prevent this we can permute the initial data by using a **Random Permutation Function.** The receiver at the other end uses the inverse of this function to obtain the data.



Sample Image



Share 1 Share 2



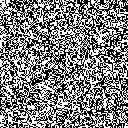
Stacking Result

We see that as expected the result is 4 times the size of the actual image.

A further extension of the scheme is the verification part that tells the receiver whether the image has been tampered with or not.

To facilitate this we take the Share 1 and shift the Share 2 by ‘h’ horizontal bits and ‘v’ vertical bits before superimposing.

This stacking results in a gibberish image.



Result of Shift Stacking

The common area between the two that is generated as a result of the stacking is written into a file and again a **Random Permutation Function** is applied to this data.

The result is transmitted as the signature along with the two shares.

At the receiver’s end we receive this signature along with the shares 1 and 2.

The receiver knows the shift values of ‘h’ and ‘v’ and also the random permutation function used.

He calculates the signature by stacking the shifted shares and applies the function on the result obtained. He then compares it to the received signature file. If these two match, he can be sure that the data has not been tampered with.

This process is analogous to the technique of using Hash Values where the plaintext (in this case the shares are not exactly the plaintext but for the purposes of considering only image authentication can be taken to be so) and the hash value (the signature file generated) are transmitted and the receiver computes the hash value of the plaintext using a key (the key in this case is the values of ‘h’ and ‘v’ along with the Random Permutation Function).