# # Table of Content



## Note to Reader

This analysis assumes following:

- The selected DCT positions being 60,61,62,63 in the Y channel
- The chance of corruption of bits is position independent

## Structure of Embeded Data

The following data structure will be encoded in the four highest frequency of DCT tables of the input jpeg image.

```
struct data_header {
    uint data_len;
    uint complete_data_checksum;
}
struct data_block {
    char data[16];
    uint checksum;
}
```

## Efficiency of the Algorithm

### Capacity of an Image

Considering `n` pixels in an image, we will have `n / 64` dct blocks. For each block, we will have space to encode 4 bits. So per `p` DCT entry we can encode `p * 4` bits, that is `p * 2` bytes.
For each 16 bytes of data, we have 4 bytes of overhead of checksum, plus a 8 bytes overhead for the global header. Therefor to encode `m` bytes of data, we will need to encode `m + (m / 16) * 4 + 8`

bytes in total.
In order to find out the data capacity of an image, we will need to

```
byte_capacity = (n_pixels / 64) * 2
real_byte_capacity = (byte_capacity - 8) - (((byte_capacity - 8) / 16) * 4)
```

For instance, encoding capacity for a `1920 * 1080` image will be 48594 kilobytes.

### Space Needed to Encode Data

Encoding 128 bytes requires:
8 bytes of global header
8 * 20 bytes of data_block

Thus 168 bytes of encoding data, That is 1344 encoded bits.
If we use 4 low freqs of each dct table, we need to manipulate 336.0 DCT tables.

## Probability of Corruption of data

### Corruption of One Bit

The corruption of data depends on whether values of the DCT entries in which we embed our data are corrupted or not. As we encode 1 bit in each DCT table entry, if any bit of the 8 bits of the entry is corrupted, the encoded bit is lost and should be flagged during checksum verification. Therefore, the probability of corruption of a bit of each encoded bit is `1 - (1 - BER)**8` since we the whole DCT table entry to be intact.

### Chance of Recovering Data Blocks

The chance of at least one bit to be corrupted is then `1 - ( 1 - BER ) ** 448`, if we consider BER to be 1%, this probability becomes 98%.
When we consider the structure of our encoding schema (8 bytes of global header + data blocks of 20 bytes ), we realize that if the global header's bits are intact, we will still be able to recover uncorrupted data.
The chance of at least one bit of error and having the header uncorrupted is `(1 - BER) ** 64 * (1 - (1 - BER) ** (j - 64))`. in the context of our example, this probability will be 51%, therefore, the recoverability of the data is almost random, considering at least 1 bit of error.