Image Steganography

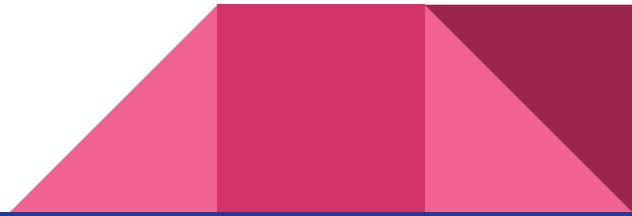
Concealing information within an image

Bit Hiding (Color Images)

Suppose you have image A and image B, both in color.

In image A, replace the least significant bits with the most significant bits of image B.

Image A will now hide Image B within it, which is imperceptible to the human eye.

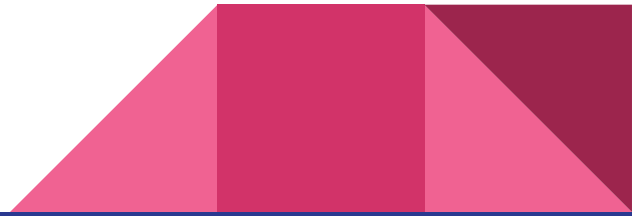


Bit Hiding (Color & Grayscale Image)

Suppose you have a color image A and grayscale image B.

For each pixel in image B, take the most significant bits and slice the binary into thirds.

Assign each binary slice into the least significant bits for the corresponding pixel's color channels in the color image.

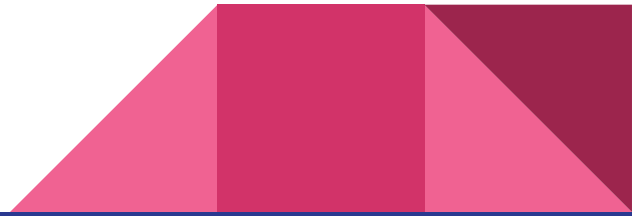


Verify Originality of an image

Hiding text using image’s bits is main idea.

Replace Least Significant bit from image pixel with a bit from the text.

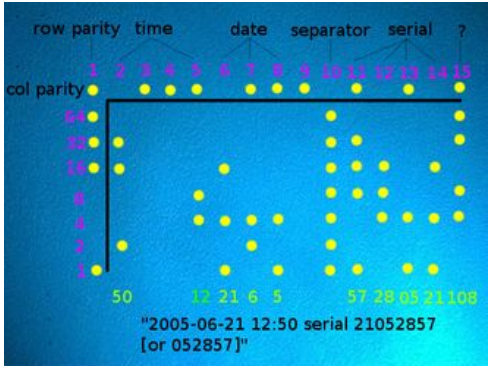
Can be used to **verify originality** or **hiding secrets**.

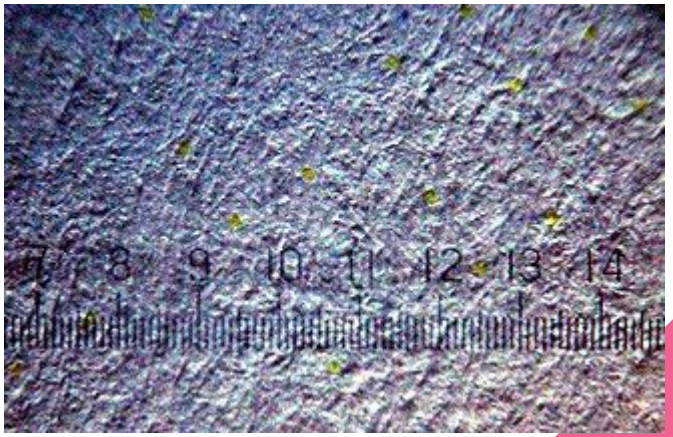


Printer Steganography

Most printers hide data within physically printed sheets of paper that include information such as printer serial number, time, date, or even ip address of the device that ordered the print.

The data is often hidden in the form of microscopic yellow dots in a repeated grid pattern on a printed paper.

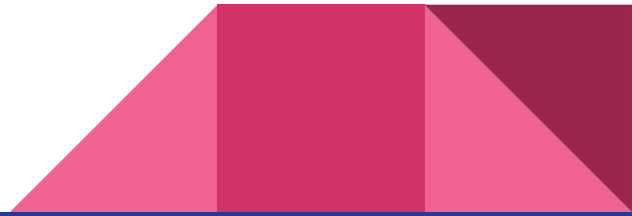




Visual Cryptography (Noise Images)

The idea is you can hide an image behind N unique shares. Each share is essentially just a noise image.

Only two of the N shares are needed to retrieve the secret image back.

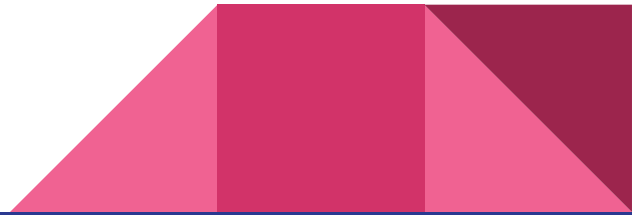


Hiding Message in Discrete Cosine Transform

There are countless algorithms for hiding messages in DCT. Some algorithms use a shortened alphabet.

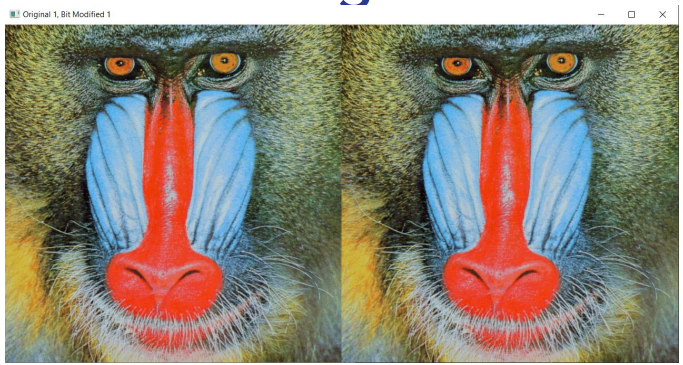
Often the message is converted to characters in binary and split up such that the bits could be hidden in various parts of the DCT, often in a grid-like formation.

For lossy hiding, the bits could just be hidden anywhere in the 8x8 blocks that are formed. For lossless hiding, the bits could be hidden towards the upper left after applying the quantization matrix.

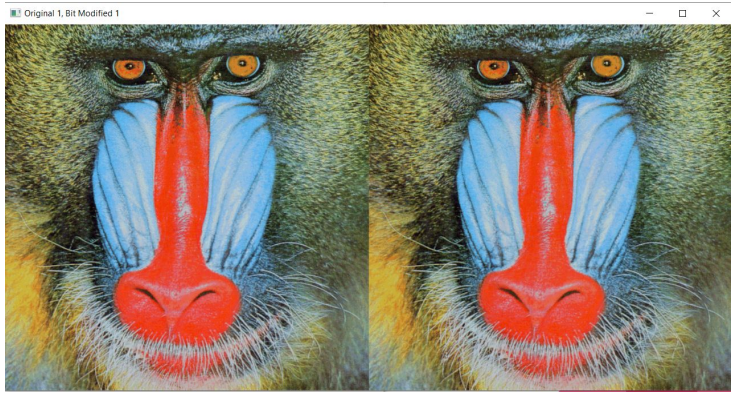


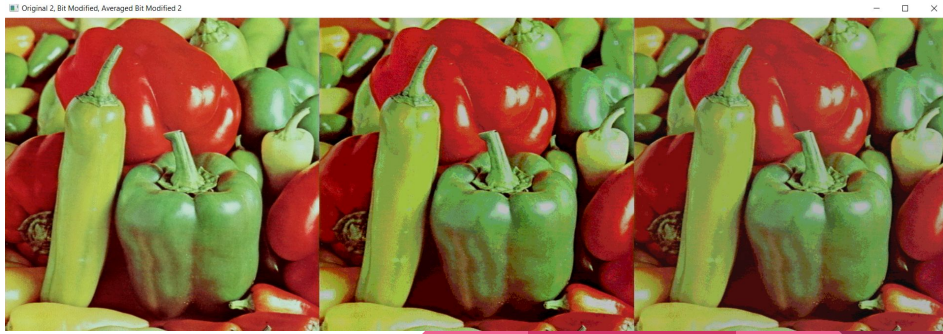
Color Bit Hiding

1 bit



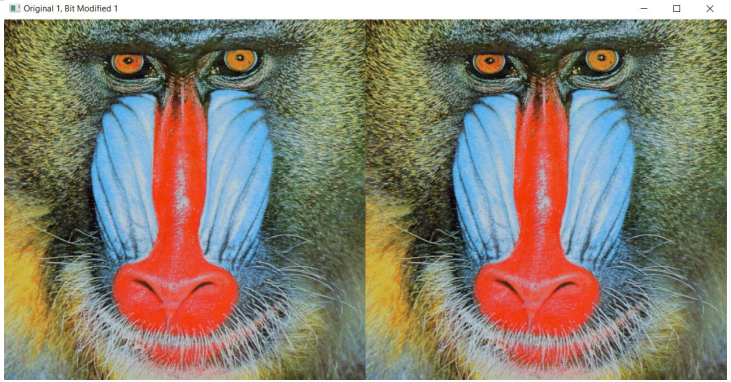
3 bits



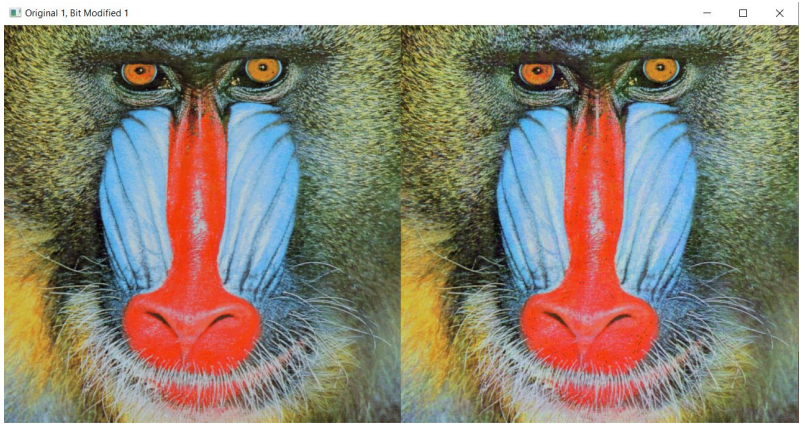


Grayscale Bit Hiding

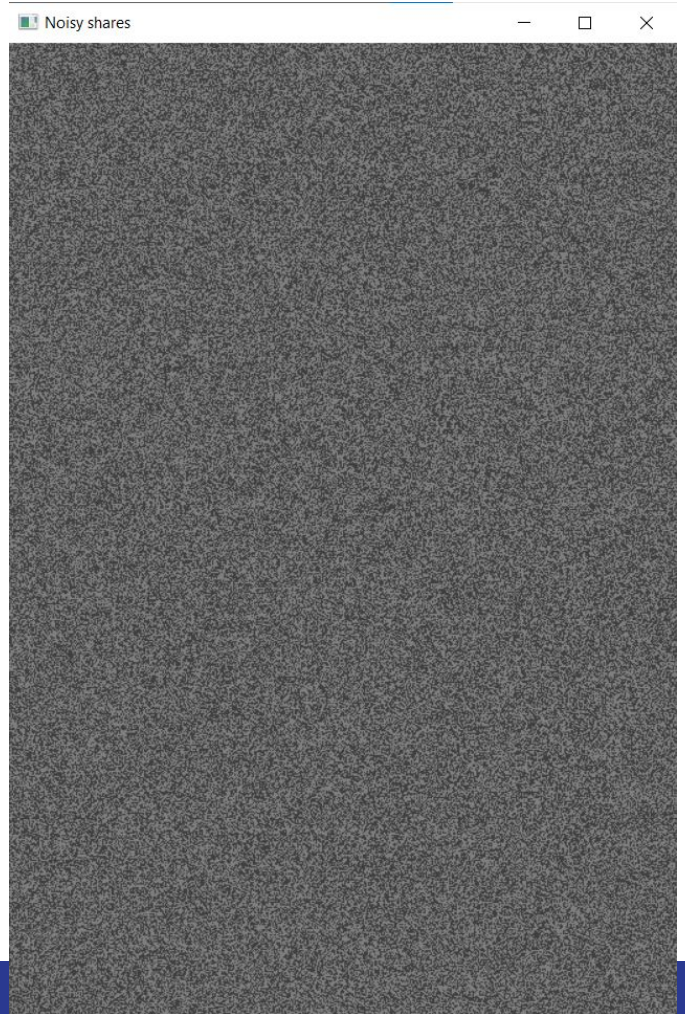
1 bit



2 bits



Printer Hiding


Noisy shares

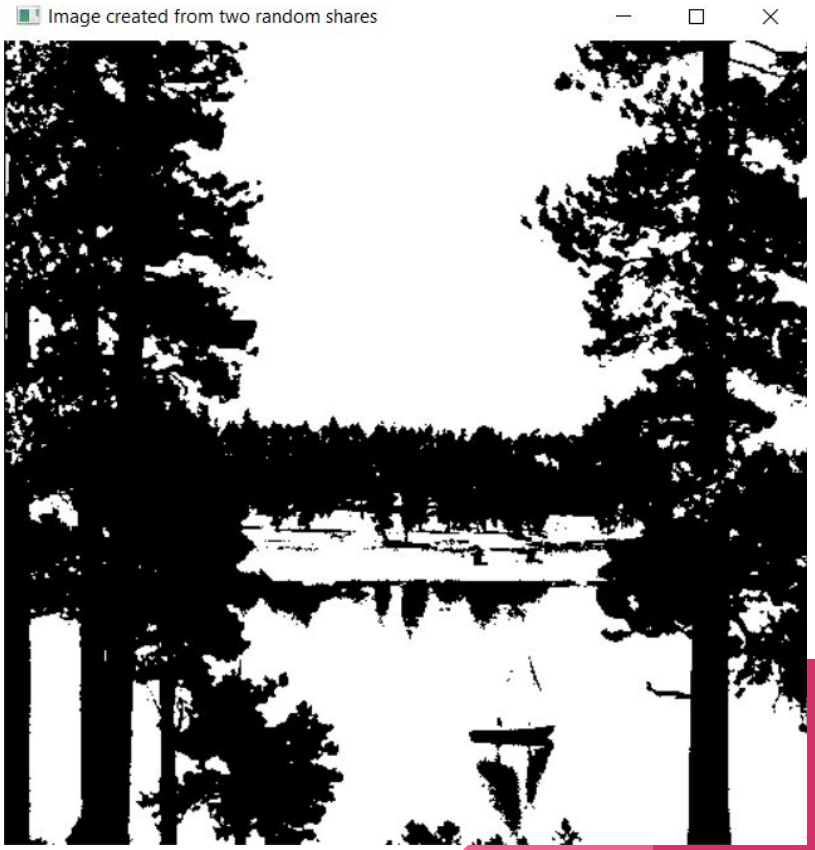
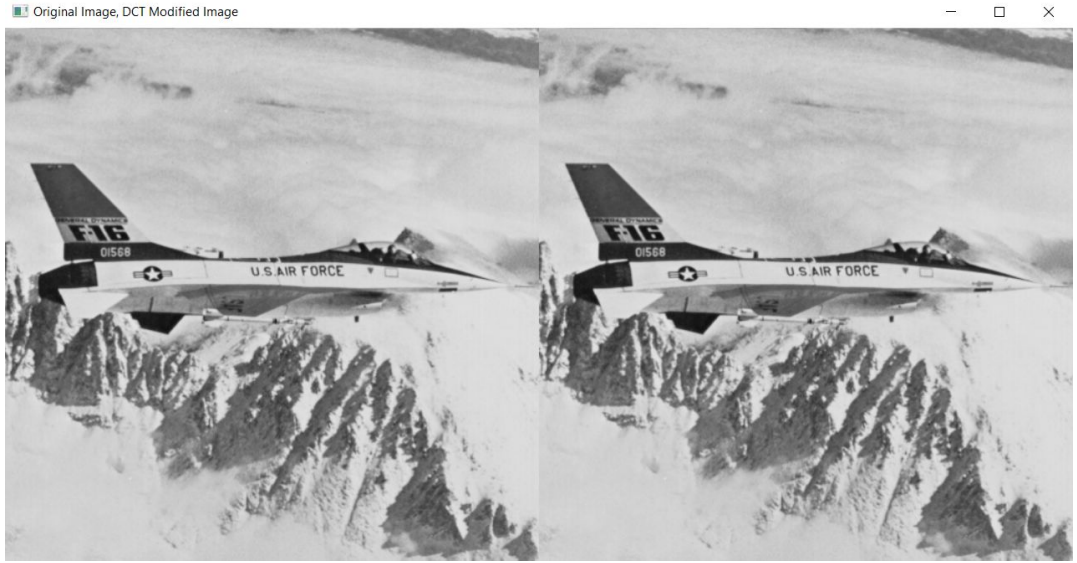


Image created from two random shares

Hidden in DCT

Top secret message is hidden here Do not share

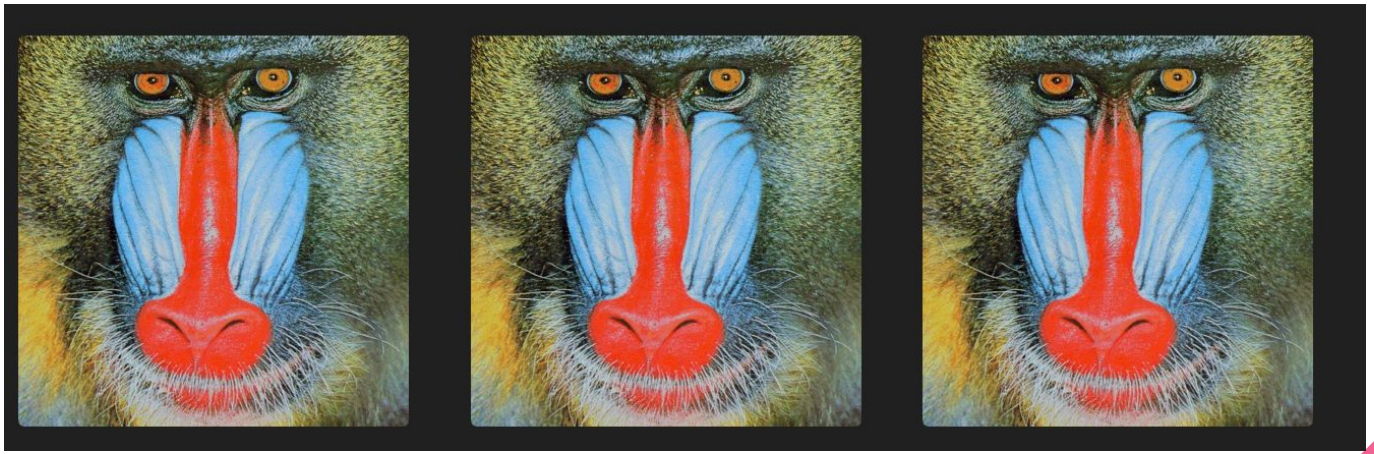
toosecretmessaoeishiddenheredonotshare



Hidden In LSB

The secret message/token is hidden in Least Significant Bit.

Extracting it allows to compare original and fake images



Thank you!

Thanks for your attention!

Any questions?