Steganography: Hiding Data In Plain Sight

Ryan Gibson

What Is Steganography?

- “The practice of concealing messages or information within other nonsecret text or data.”
- Comes from the Greek words steganos (“covered, concealed, or protected”) and graphein (“writing”).
- Essentially, “hiding data within data.”

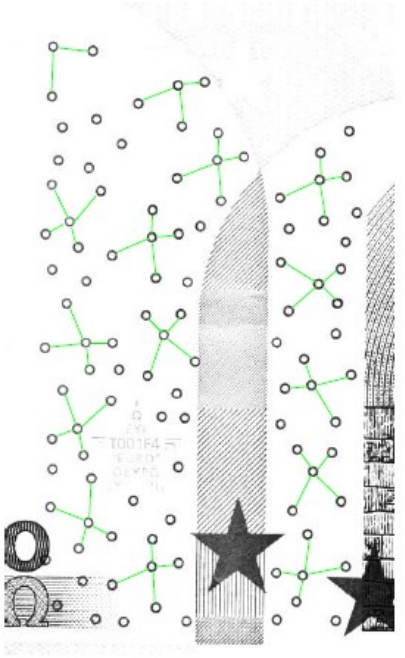
General Techniques

- Digital steganographic techniques mainly fall into two categories:
  - Alteration of numeric representations of data.
  - Overlay data or patterns on top of carrier file in the hopes that it can be recovered later.

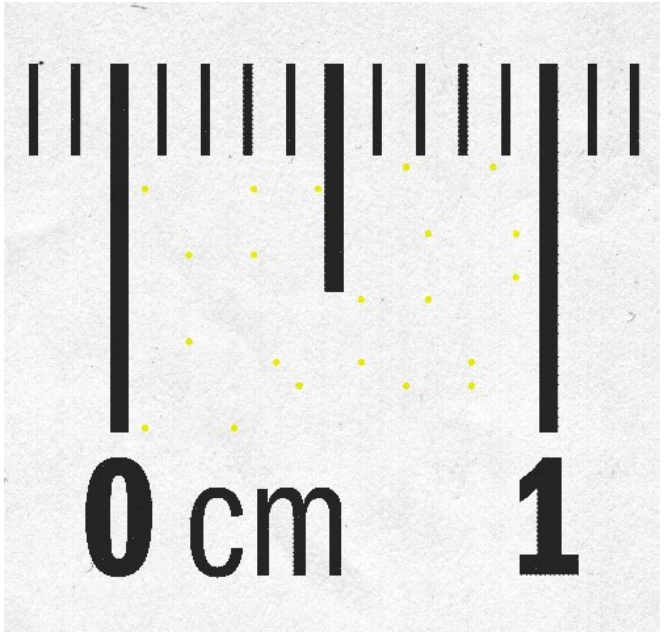
EURion Constellation

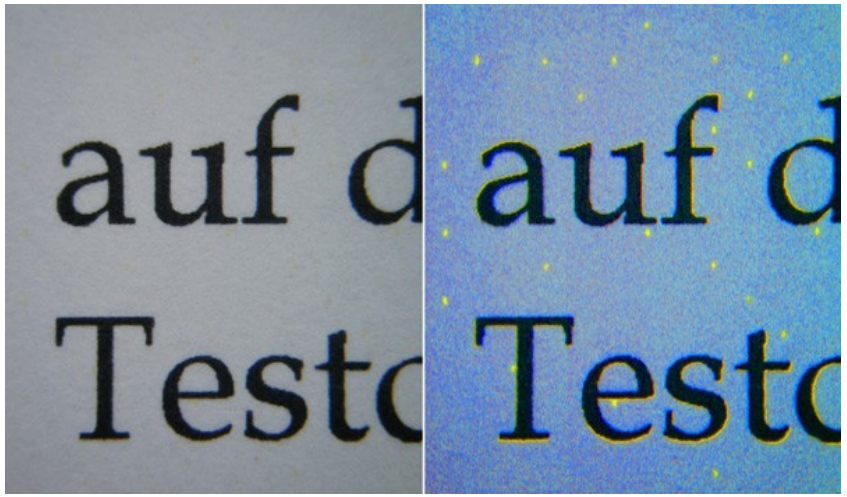
- Pattern of circles incorporated into ~50 countries' banknotes since 1996.
- Added to help imaging software detect the presence of banknotes in digital images to combat counterfeiting.
- Discovered in 2002 when Marcus Kuhn's printer refused to print banknotes.

EURion Constellation Examples



Printer Steganography



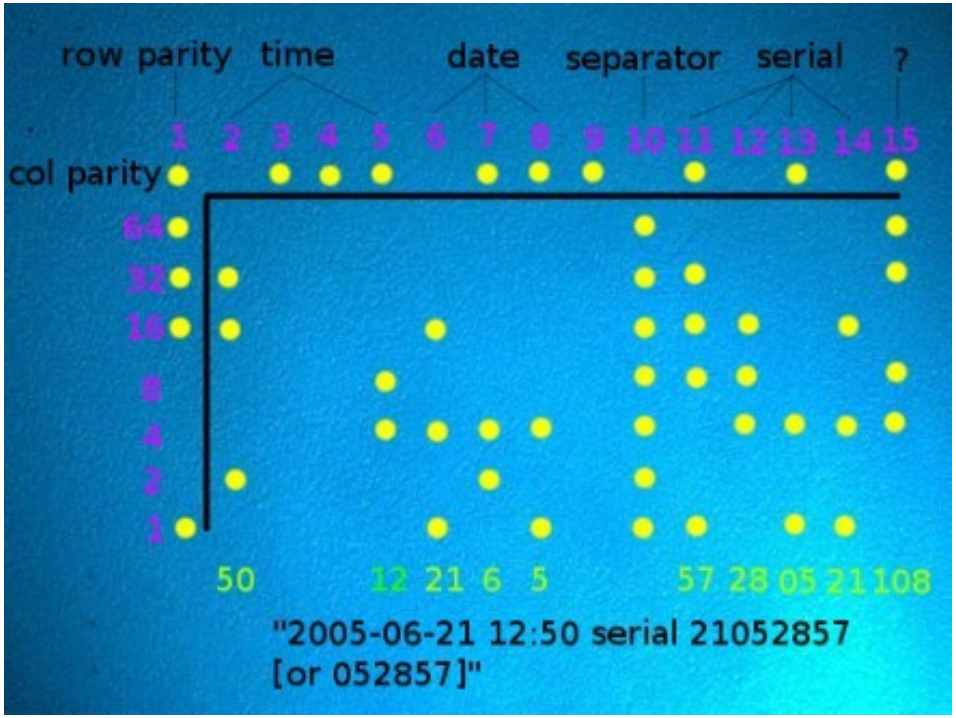


- Color laser printers (and some monochrome ones) tile a pattern of small yellow dots across printed pages.
- Intended to produce minimal visible change to the printout, ideally being imperceptible to the naked eye.

Xerox DocuColor printers

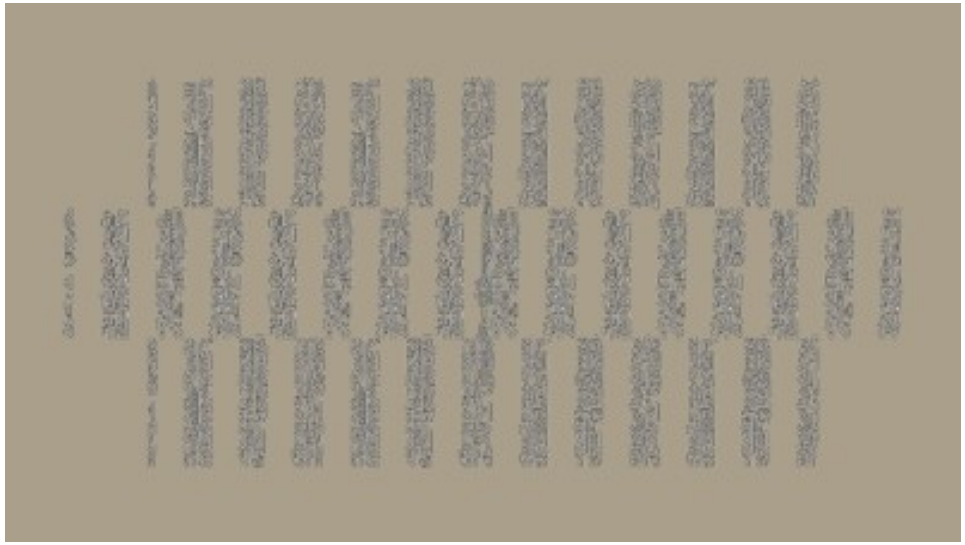
10x Magnification

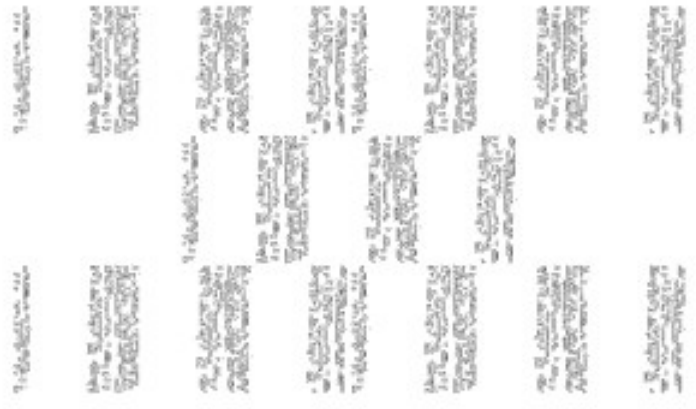
Under blue light



World Of Warcraft

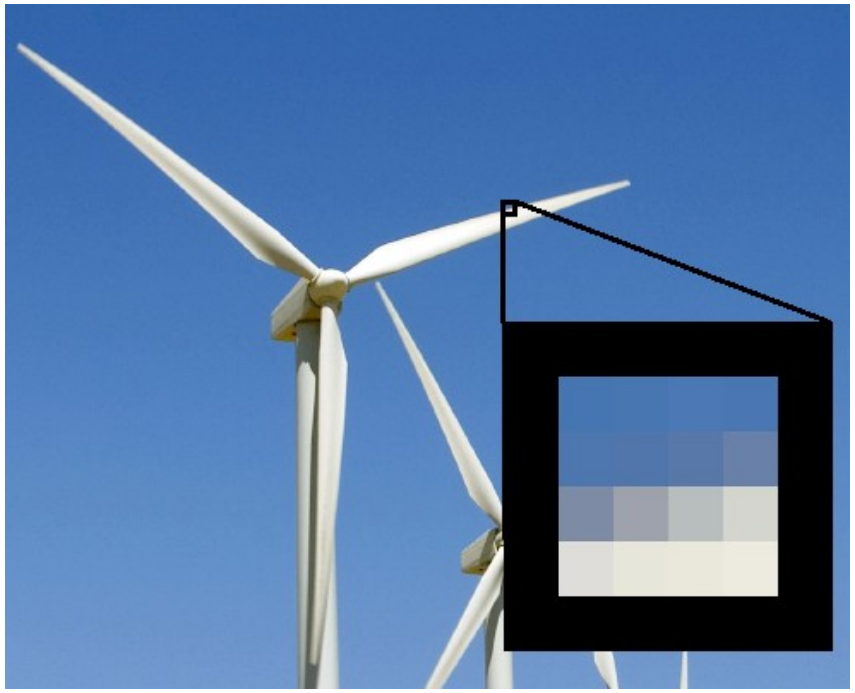
("World Of Watermarks")





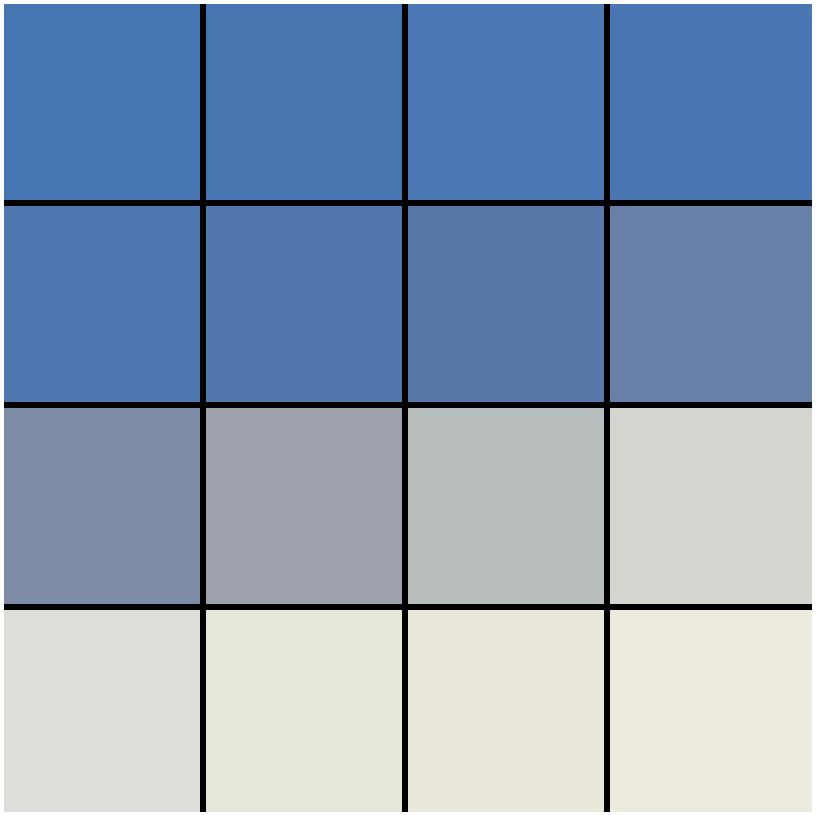
- User ID
- Time of screenshot capture
- Client version
- IP address of the server
- Implemented by Digimarc
  - Sells steganographic methods to combat piracy, copyright infringement, etc.

Least Signicant Bit Steganography: What's In A Bitmap Image?



- Grid of “pixels,” squares of color represented by integers for the red, green, and blue color components.
- These are 8-bit integers and thus range from 0 – 255.

Bitmap Example



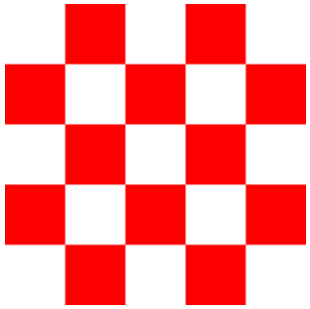
64 112 174	66 112 172	70 114 177	68 112 175
72 112 171	75 112 167	82 114 165	99 122 164
119 133 159	153 157 168	180 186 186	209 210 204
220 219 215	228 228 216	232 230 217	236 234 222

$2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0$

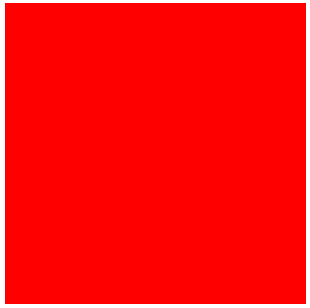
(11101000, 11100110, 11011001)

Least Significant Bit Steganography

R = 255 = 11111111



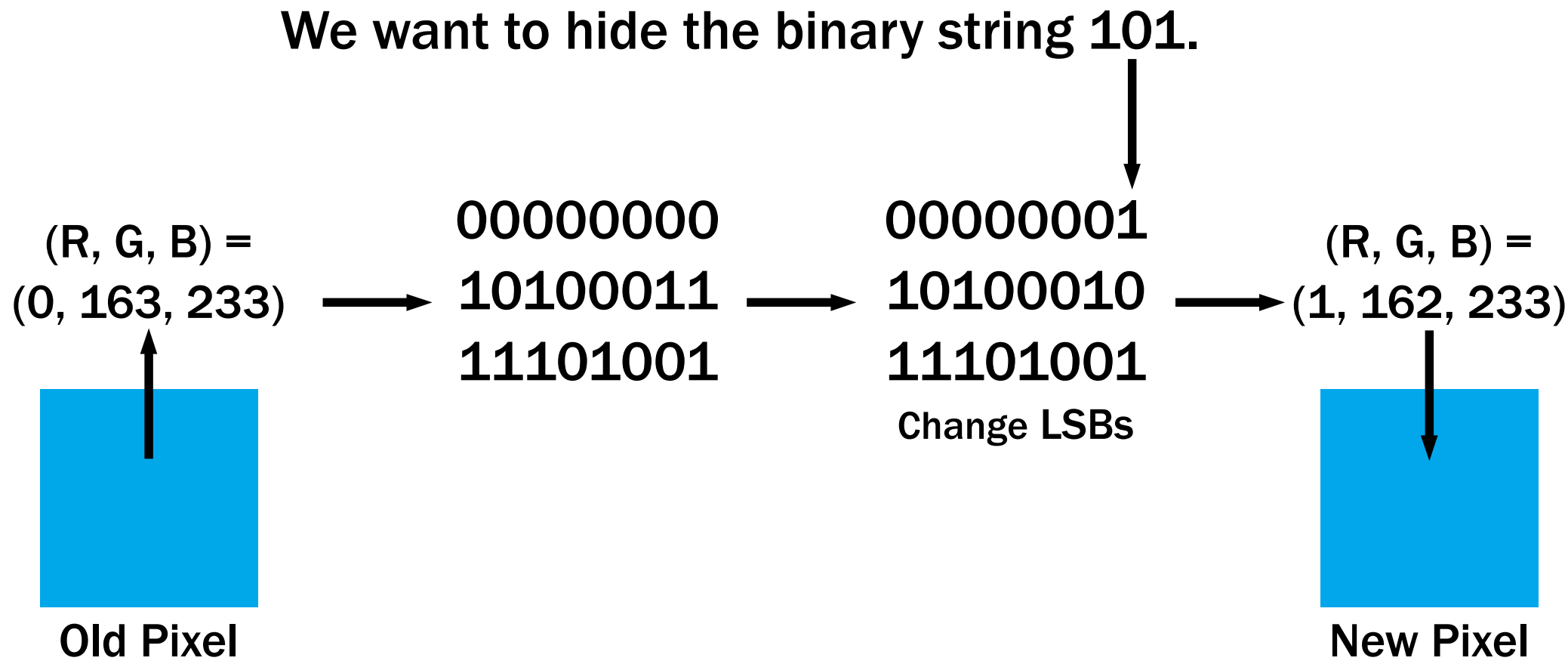
R = 254 = 11111110



(Previous Images Superimposed)

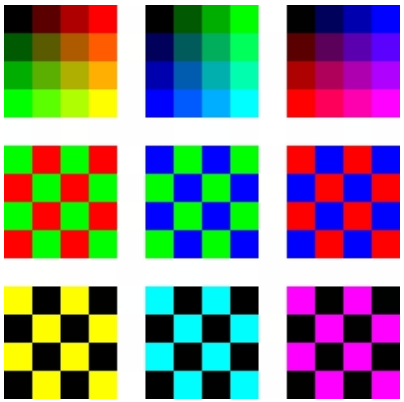
- Based on the fact that we can't differentiate between small color differences.
- Altering the least significant bits of a color channel won't make a noticeable difference.
- We can hide a binary string in the LSBs of consecutive color channels.

LSB Steganography (Hypothetical) Example

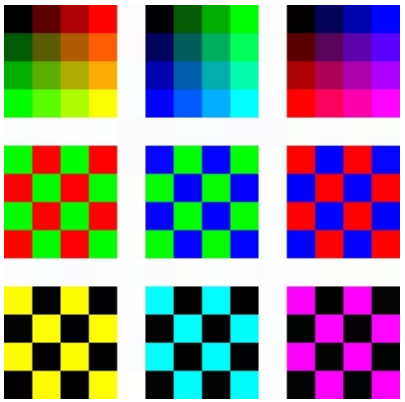


How Many Bits To Use?

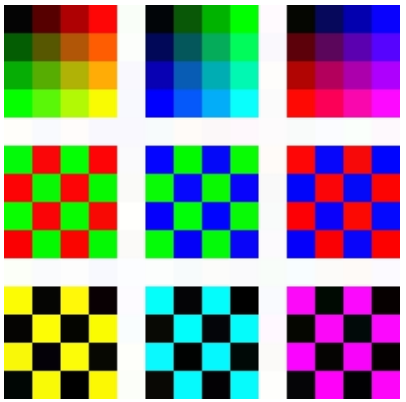
We replace a percentage of the image's data, but lose much less color information.



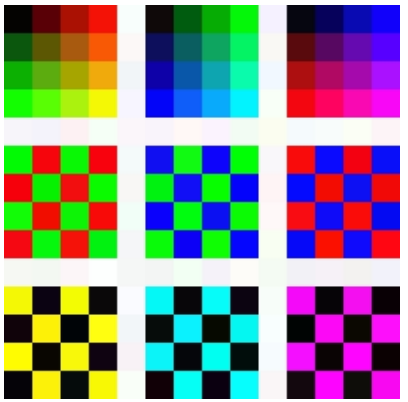
1 LSB:
Replace 12.5%
Lose ~0.4%



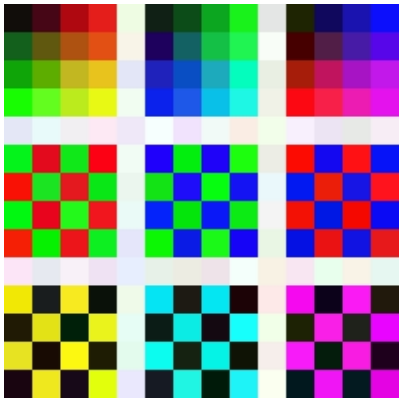
2 LSBs:
Replace 25.0%
Lose ~1.2%



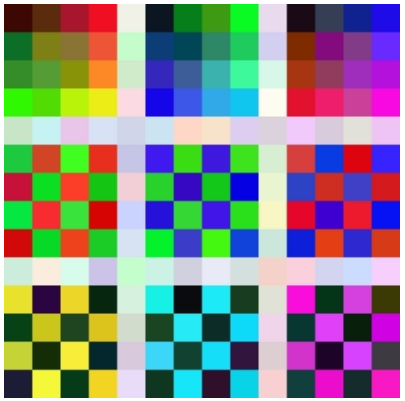
3 LSBs:
Replace 37.5%
Lose ~2.75%



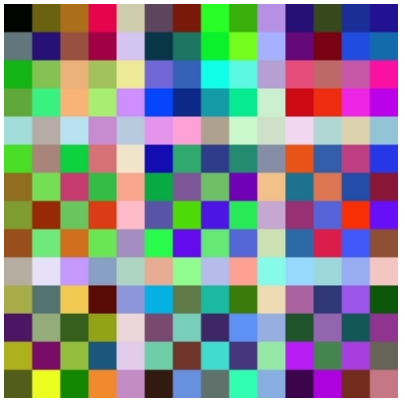
4 LSBs:
Replace 50.0%
Lose ~5.88%



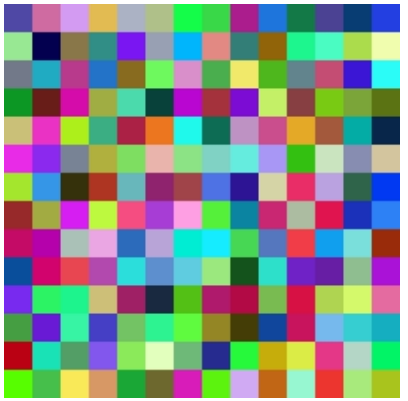
5 LSBs:
Replace 62.5%
Lose ~12.16%



6 LSBs:
Replace 75.0%
Lose ~24.71%



7 LSBs:
Replace 82.5%
Lose ~49.80%



8 LSBs:
Replace 100.0%
Lose ~100%

LSB Steganography Example (Original)

LSB Steganography Example (Steganographed)

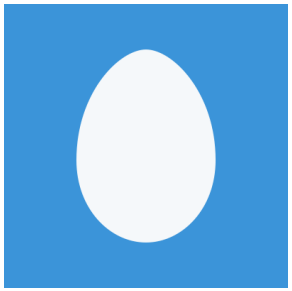
The Steganographed Image Contains:

- “The Complete Works of William Shakespeare”
  - 37 plays, 154 sonnets
  - 124,788 lines of text
  - 10,004,560 words

- Image Resolution: 1920x1080

Other Examples

Using 2 LSBs:



- Twitter Profile Picture (400x400)
- We can hide:
  - 120,000 Bytes
  - ~117.19 KiB

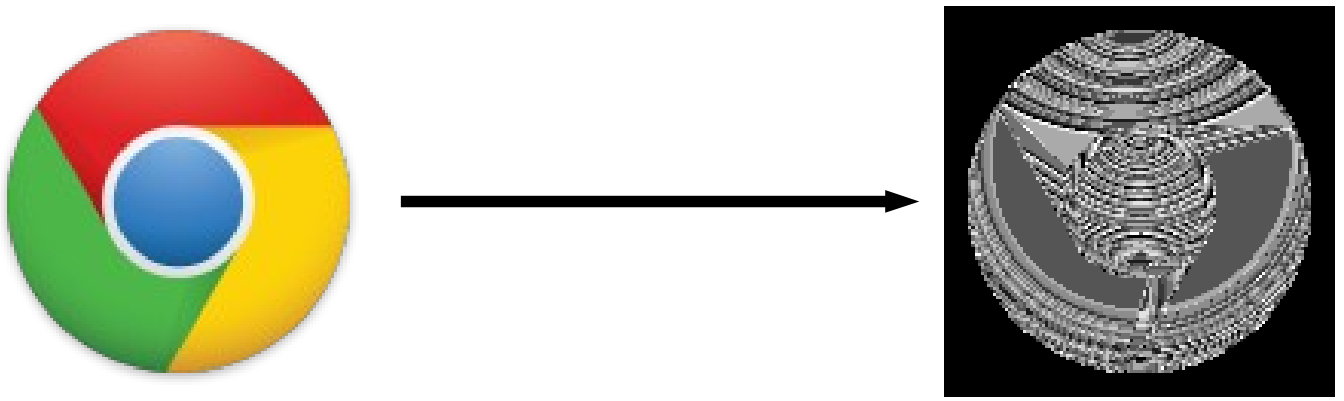
- 1080p Desktop Background (1920x1080)
- We can hide:
  - 1,555,200 Bytes
  - ~1.48 MiB

- NASA Image of Pluto (8000x8000)
- We can hide:
  - 48,000,000 Bytes
  - ~45.78 MiB

- Panorama of Milky Way (16702x6568)
- We can hide:
  - 82,274,052 Bytes
  - ~78.46 MiB

Detecting LSB Steganography

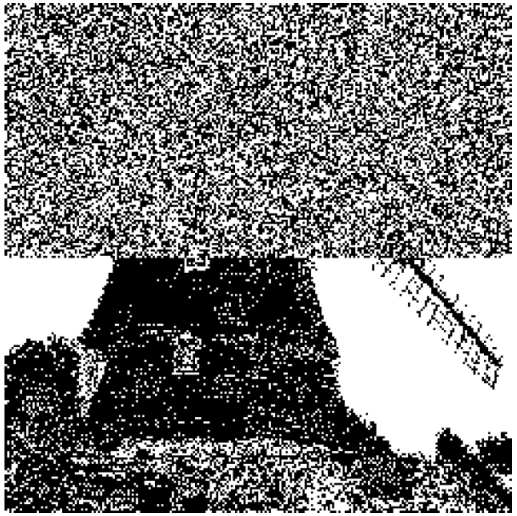
- LSB Steganography depends on altering the LSBs of each color value.
- We can isolate these LSBs and display them alone through linear normalization.
- This is called a visual attack.



Example

Original

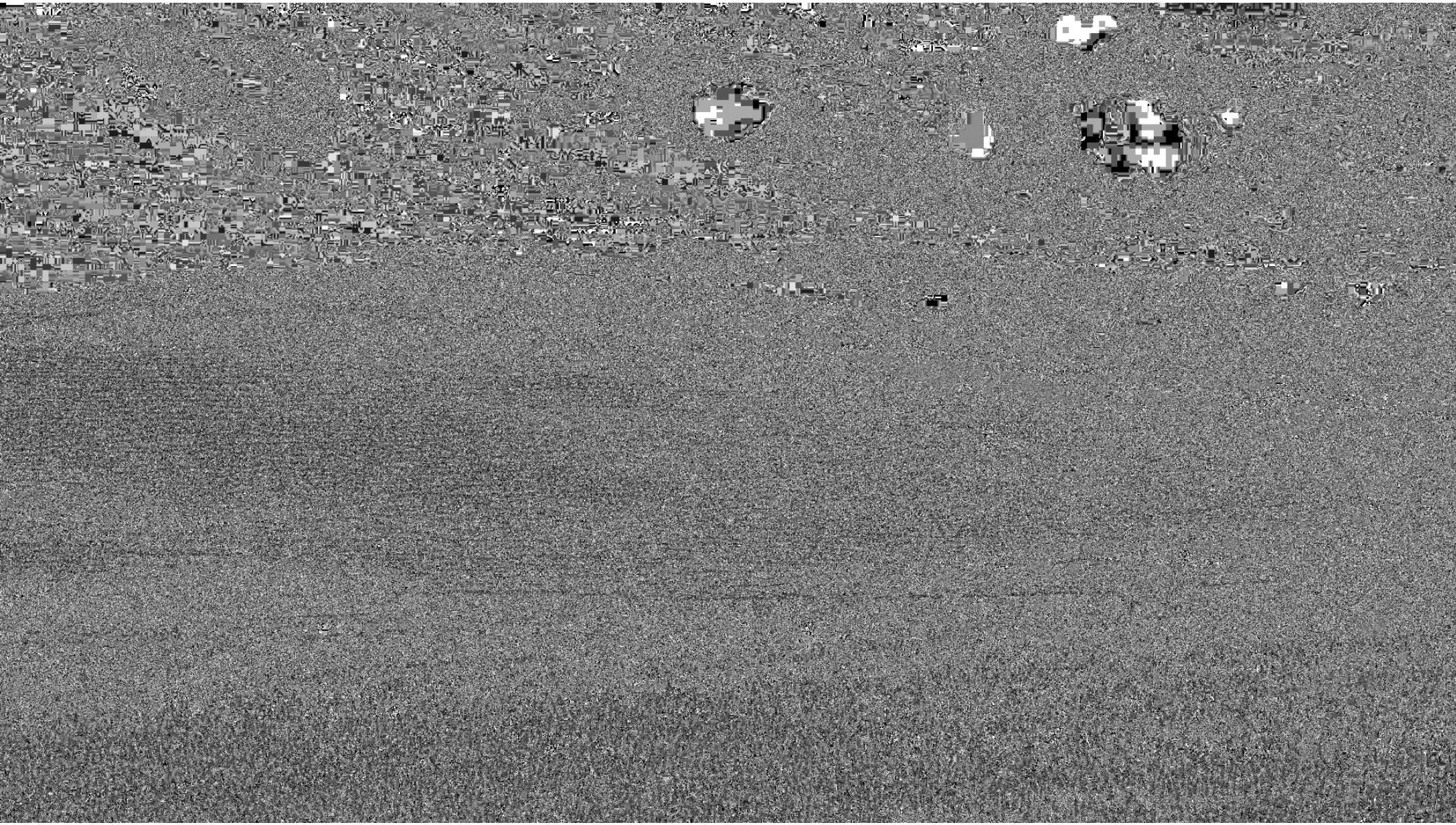
LSBs of Original



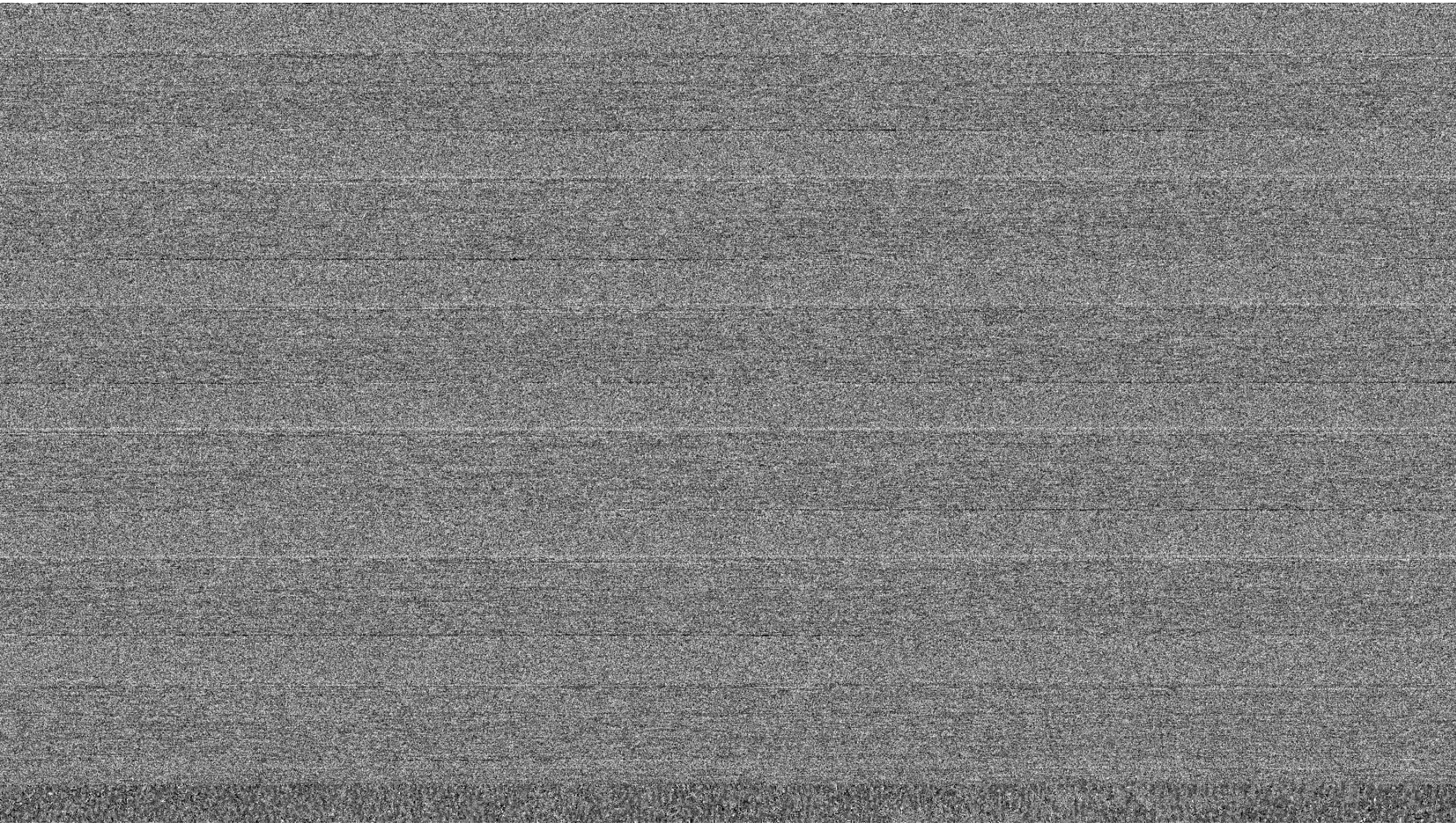
LSBs of Steganographed Version

Example (Original Image)

Example Non-Steganographed (2 LSBs)



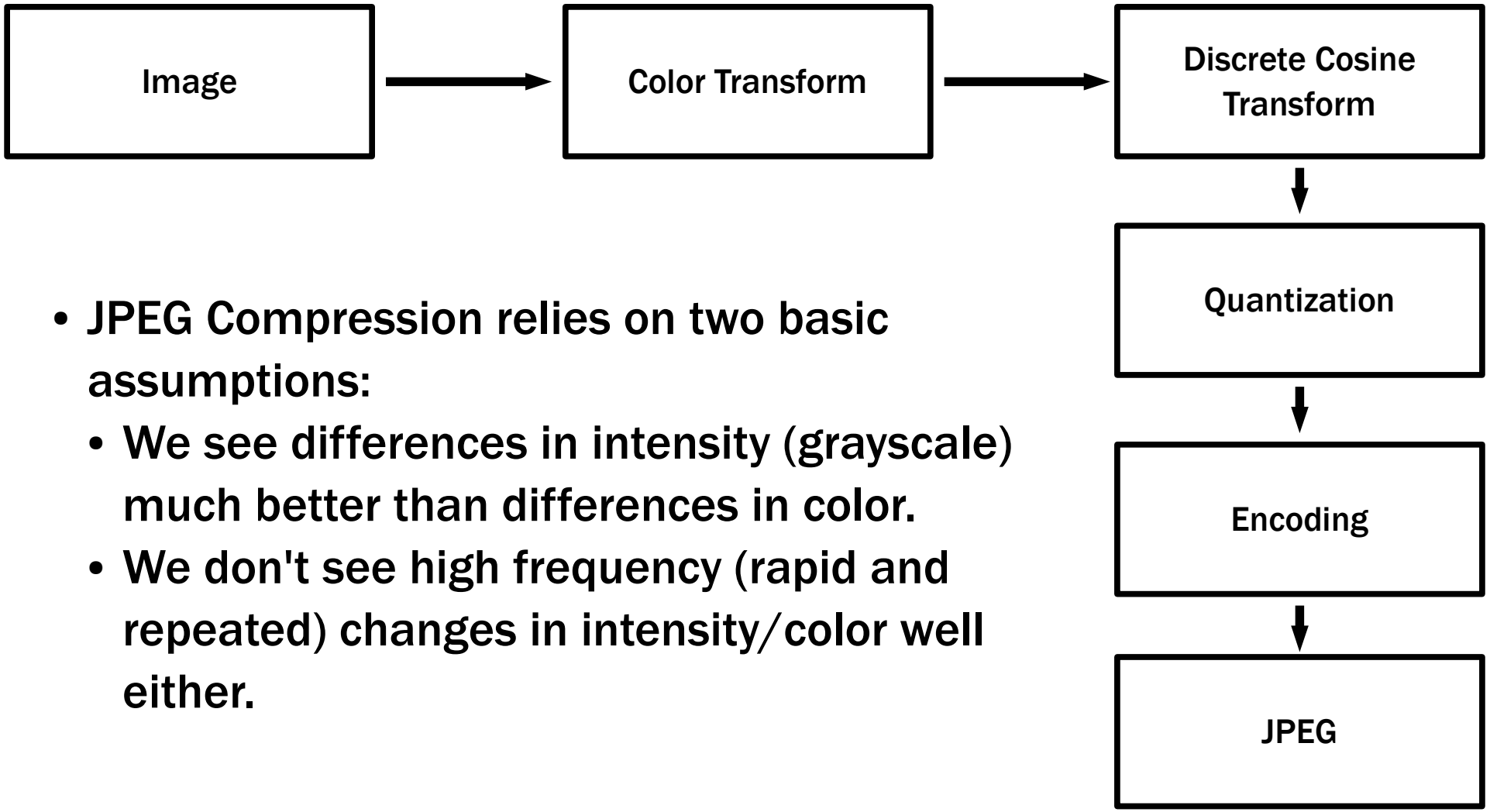
Example Steganographed (2 LSBs)



JPEG Images

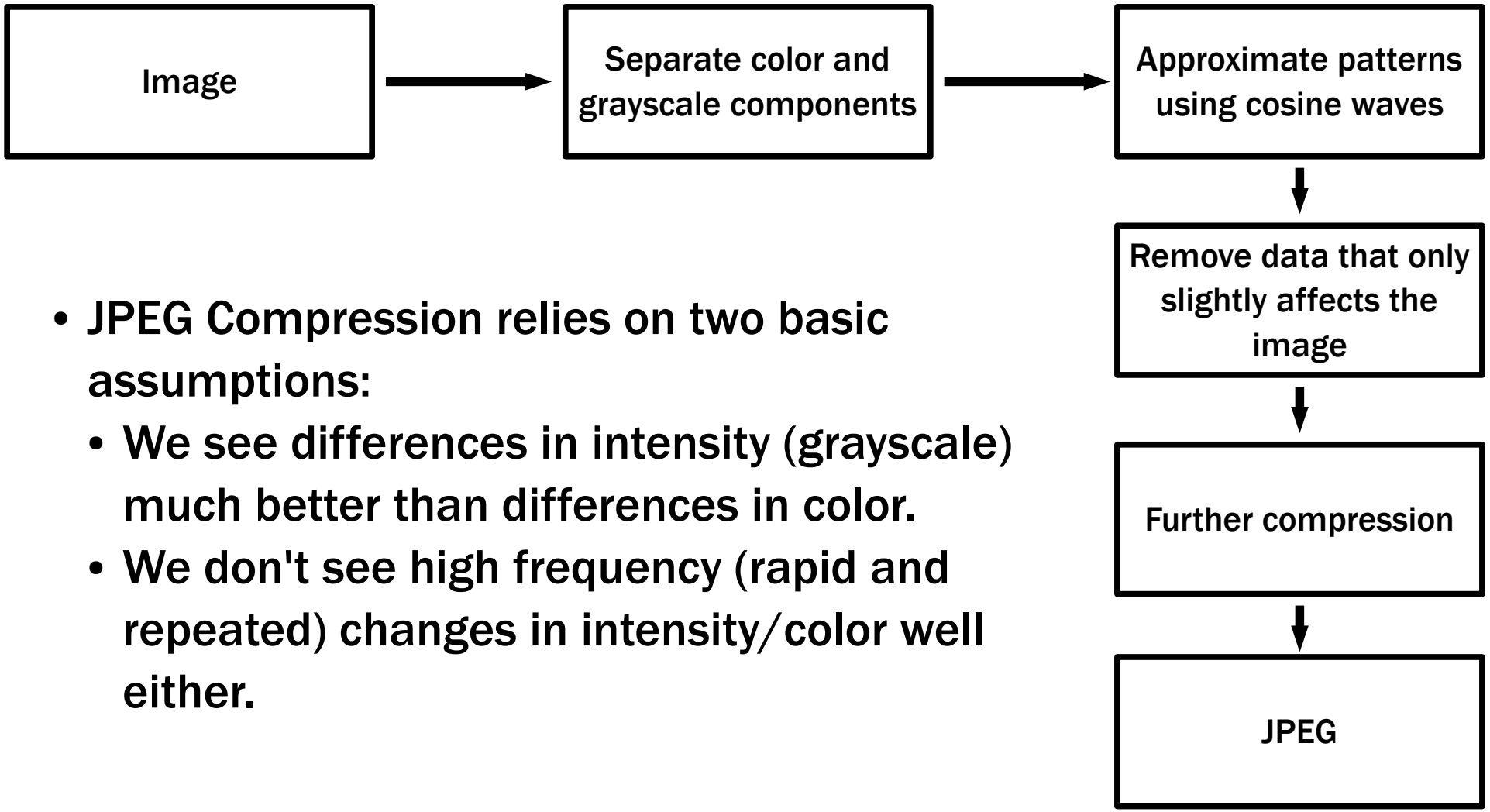
- **Much more popular on the internet than .png or .bmp images.**
- **Lossy compression – when saving an image, the output is different than the input.**
  - Least Significant Bit Steganography no longer works.

Discrete Cosine Transform Steganography: JFIF (JPEG) Compression



- JPEG Compression relies on two basic assumptions:
  - We see differences in intensity (grayscale) much better than differences in color.
  - We don't see high frequency (rapid and repeated) changes in intensity/color well either.

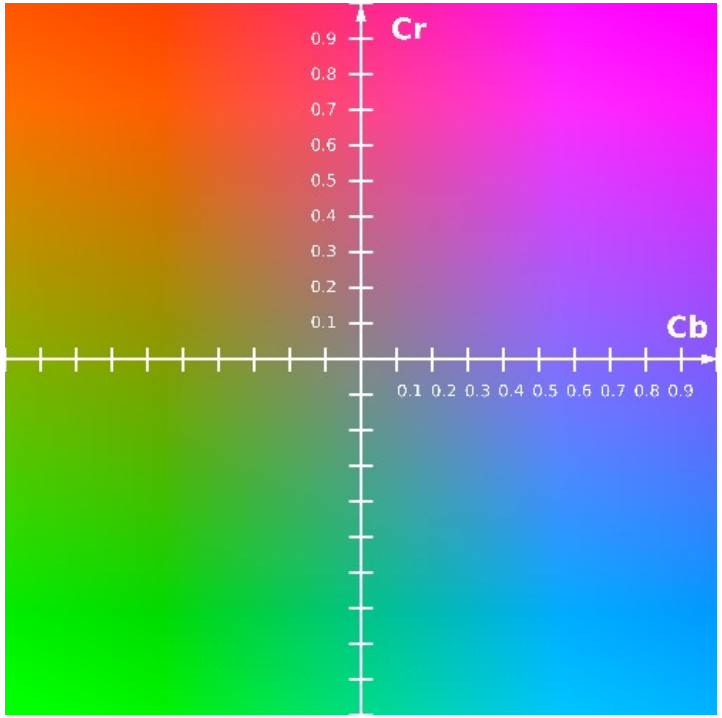
Discrete Cosine Transform Steganography: JFIF (JPEG) Compression

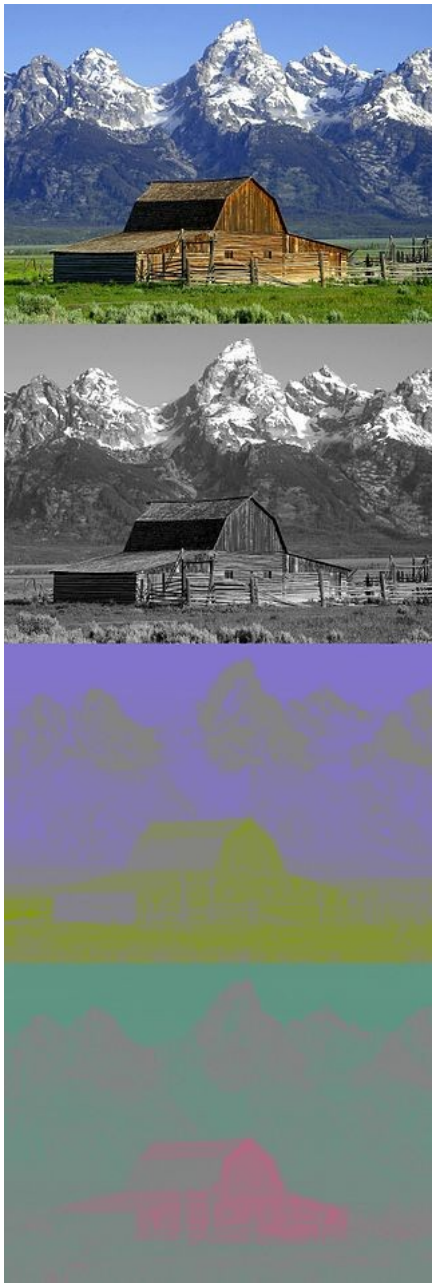


- JPEG Compression relies on two basic assumptions:
  - We see differences in intensity (grayscale) much better than differences in color.
  - We don't see high frequency (rapid and repeated) changes in intensity/color well either.

Color Transform

- Convert image from RGB to YCbCr.
  - Y = Luminance, “Intensity”
  - Cb = Chroma Blue
  - Cr = Chroma Red





Color Transform (Downsampling)

- To save space, we can downsample the chrominance components of our image.
  - Normally downsample by a factor of 4.
- Most people won't be able to see too much of a difference.

Original

25x Less Color

Color Transform (Downsampling)

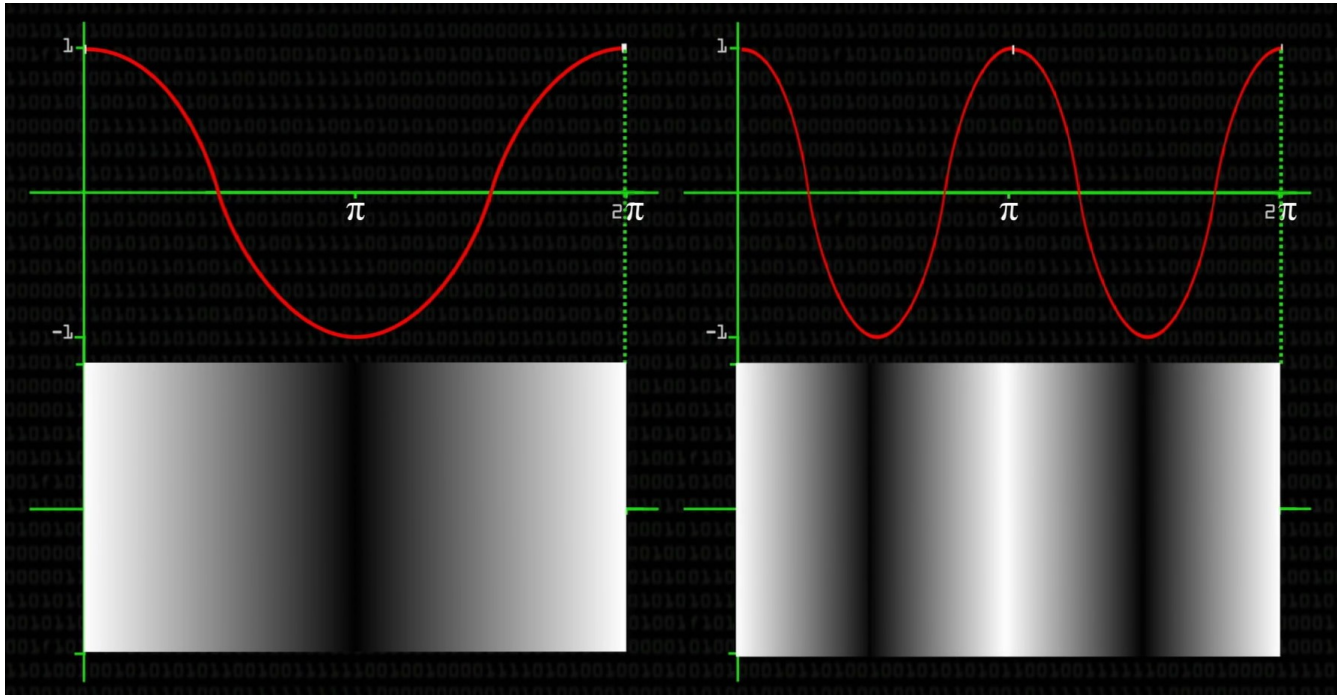
Y unchanged

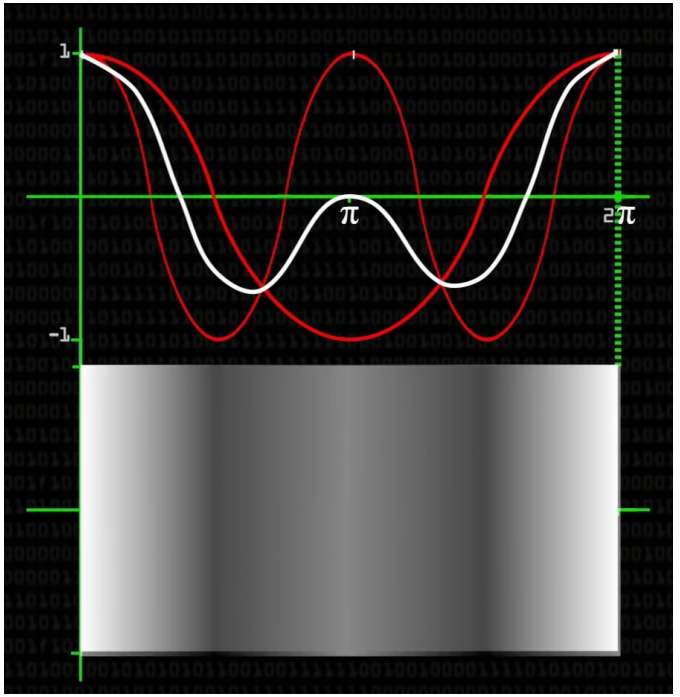
Cb downsampled

Cr downsampled

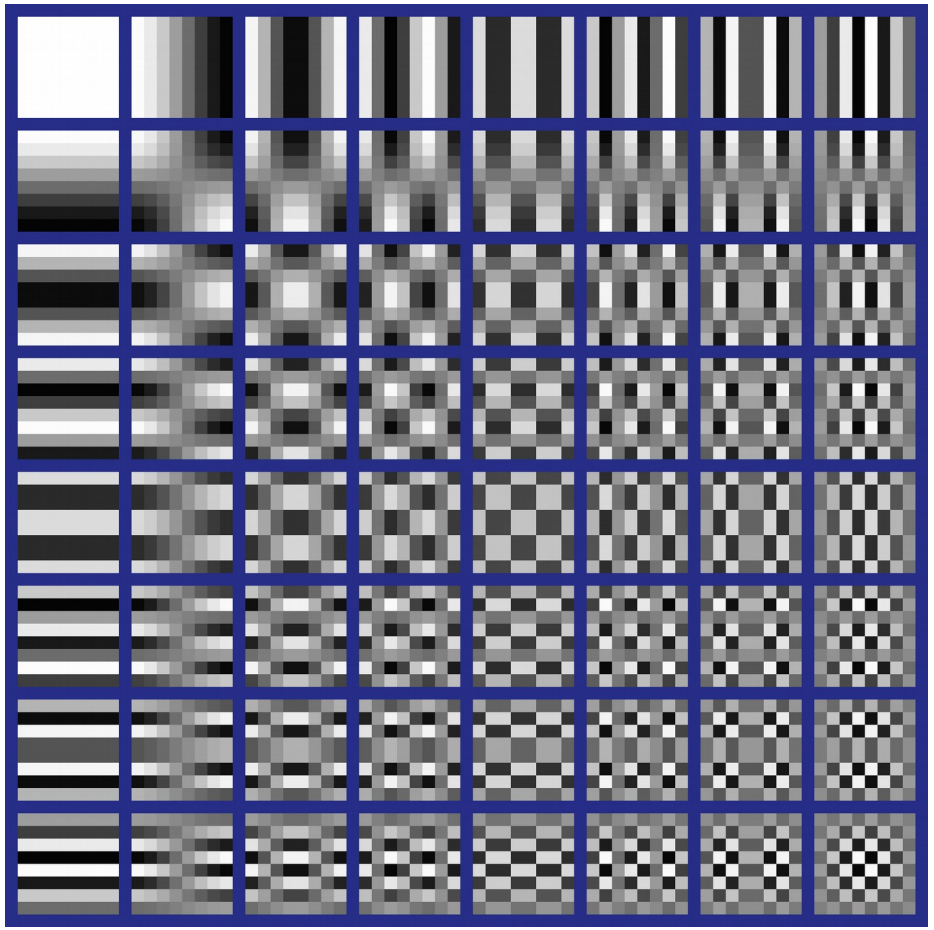
Discrete Cosine Transform

- In general, we represent a sequence of data points as the sum of weighted cosine waves.
- n data points can be represented by n cosine waves of different frequencies.





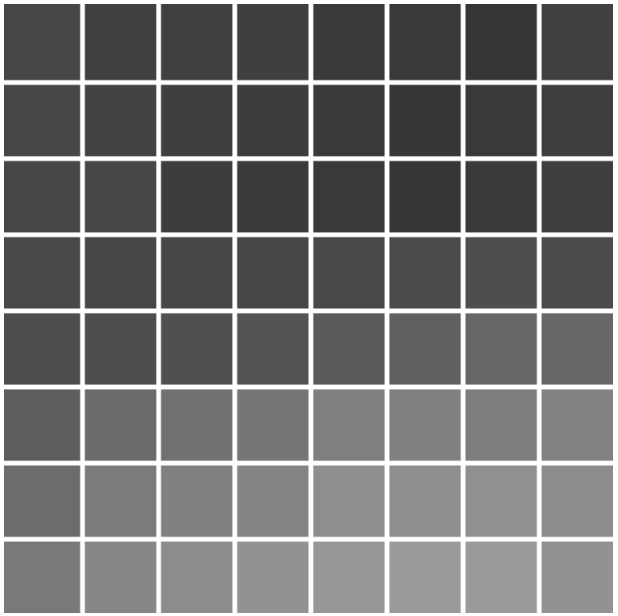
Discrete Cosine Transform



- We split a JFIF image into 8x8 squares and apply DCT separately for each one.
- We have 64 data points (8x8 pixels), so we can represent them with 64 weighted cosine waves.

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1.$$

Discrete Cosine Transform (Example)



Get values

62	55	55	54	49	48	47	55
62	57	54	52	48	47	48	53
61	60	52	49	48	47	49	54
63	61	60	60	63	65	68	65
67	67	70	74	79	85	91	92
82	95	101	106	114	115	112	117
96	111	115	119	128	128	130	127
109	121	127	133	139	141	140	133

Input block

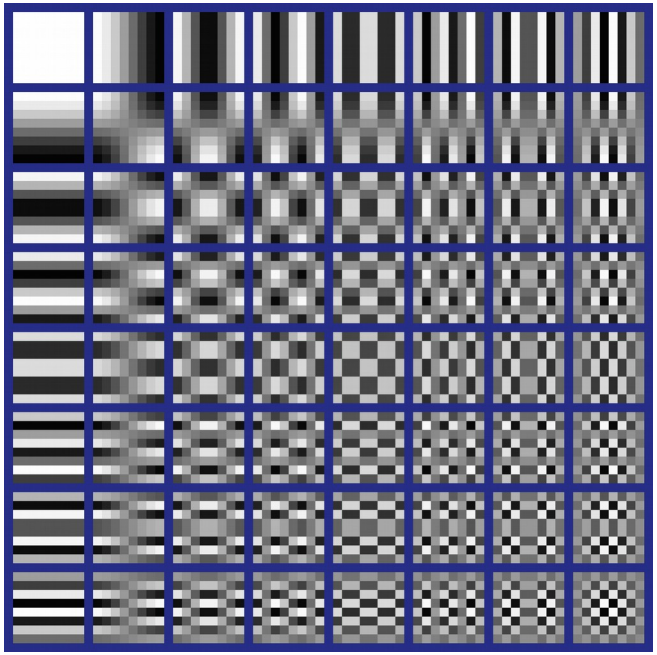
Shift values down by 127 to center them around 0 (like a cosine wave)

-66	-73	-73	-74	-79	-80	-81	-73
-66	-71	-74	-76	-80	-81	-80	-75
-67	-68	-76	-79	-80	-81	-79	-74
-65	-67	-68	-68	-65	-63	-60	-63
-61	-61	-58	-54	-49	-43	-37	-36
-46	-33	-27	-22	-14	-13	-16	-11
-32	-17	-13	-9	0	0	2	-1
-19	-7	-1	5	11	13	12	5

Shifted Block

Discrete Cosine Transform (Example)

-370	-29.7	-2.6	-2.5	-1.1	-3.7	-1.5	-0.08
-231	44.9	24.5	-0.3	9.3	3.9	4.3	-1.4
62.8	8.5	-7.6	-2.7	0.3	-0.4	0.5	-0.8
12.5	-14.6	-3.5	-3.4	2.4	-1.3	2.7	-0.4
-4.9	-3.9	0.9	3.6	0.1	5.1	1.1	0.5
-0.5	3.1	-1.4	0.2	-1.1	-1.5	-1.1	0.9
4.4	2.3	-1.7	-1.6	1.1	-2.7	1.1	1.4
-10.2	-1.8	5.9	-0.4	0.3	0.4	-1	0



DCTII Coefficients (range from -1024 to 1024)

Apply DCTII

-66	-73	-73	-74	-79	-80	-81	-73
-66	-71	-74	-76	-80	-81	-80	-75
-67	-68	-76	-79	-80	-81	-79	-74
-65	-67	-68	-68	-65	-63	-60	-63
-61	-61	-58	-54	-49	-43	-37	-36
-46	-33	-27	-22	-14	-13	-16	-11
-32	-17	-13	-9	0	0	2	-1
-19	-7	-1	5	11	13	12	5

Shifted Block

Quantization

- We can remove some of the smaller DCT coefficients to save more space.
- High frequency cosine waves (near the bottom right of the table) are not seen by the eye well and can be removed as well.
- We divide each coefficient by the corresponding quantization value and round to the nearest integer.

16	12	14	14	18	24	49	72
11	12	13	17	22	35	64	92
10	14	16	22	37	55	78	95
16	19	24	29	56	64	87	98
24	26	40	51	68	81	103	112
40	58	57	87	109	104	121	100
51	60	69	80	103	113	120	103
61	55	56	62	77	92	101	99

Standard Y JPEG quantization table for 50% quality

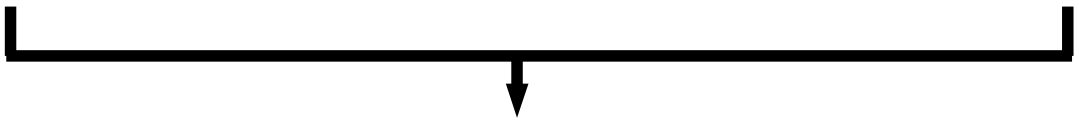
Discrete Cosine Transform (Example)

-370	-29.7	-2.6	-2.5	-1.1	-3.7	-1.5	-0.08
-231	44.9	24.5	-0.3	9.3	3.9	4.3	-1.4
62.8	8.5	-7.6	-2.7	0.3	-0.4	0.5	-0.8
12.5	-14.6	-3.5	-3.4	2.4	-1.3	2.7	-0.4
-4.9	-3.9	0.9	3.6	0.1	5.1	1.1	0.5
-0.5	3.1	-1.4	0.2	-1.1	-1.5	-1.1	0.9
4.4	2.3	-1.7	-1.6	1.1	-2.7	1.1	1.4
-10.2	-1.8	5.9	-0.4	0.3	0.4	-1	0

DCTII Coefficients (range from -1024 to 1024)

16	12	14	14	18	24	49	72
11	12	13	17	22	35	64	92
10	14	16	22	37	55	78	95
16	19	24	29	56	64	87	98
24	26	40	51	68	81	103	112
40	58	57	87	109	104	121	100
51	60	69	80	103	113	120	103
61	55	56	62	77	92	101	99

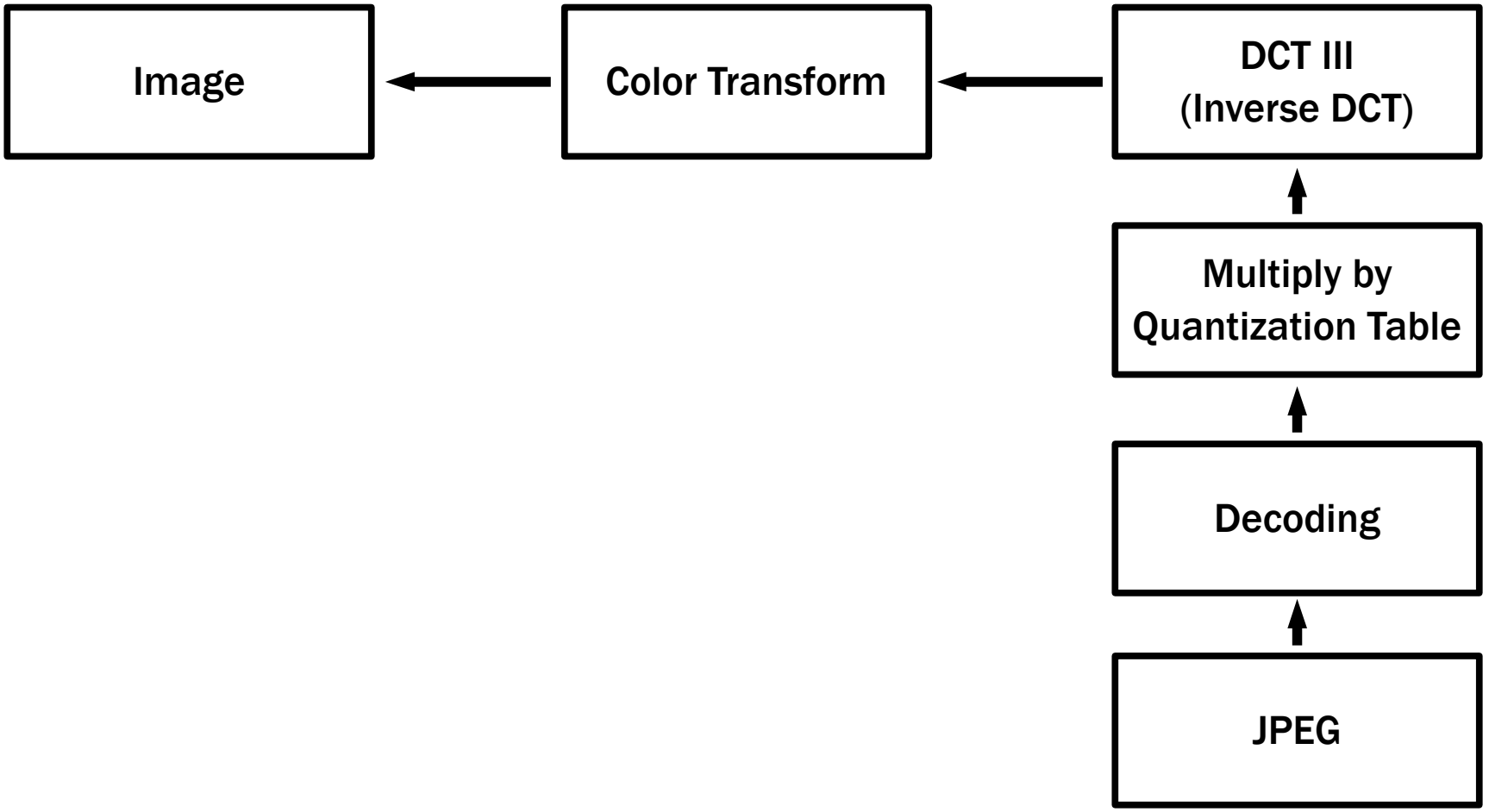
Quantization Table



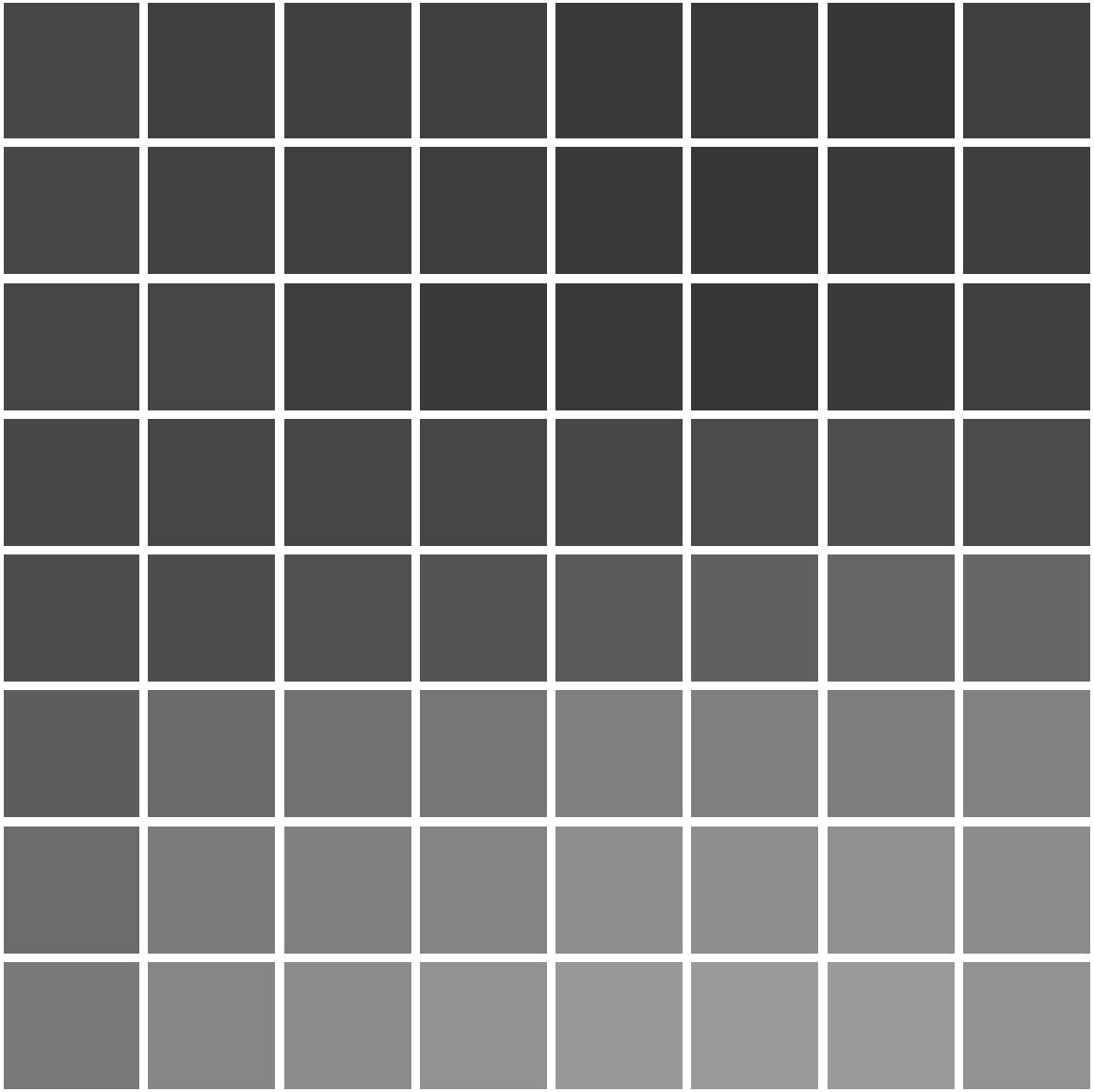
-23	-2	0	0	0	0	0	0
-21	4	2	0	0	0	0	0
6	1	0	0	0	0	0	0
1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized Table

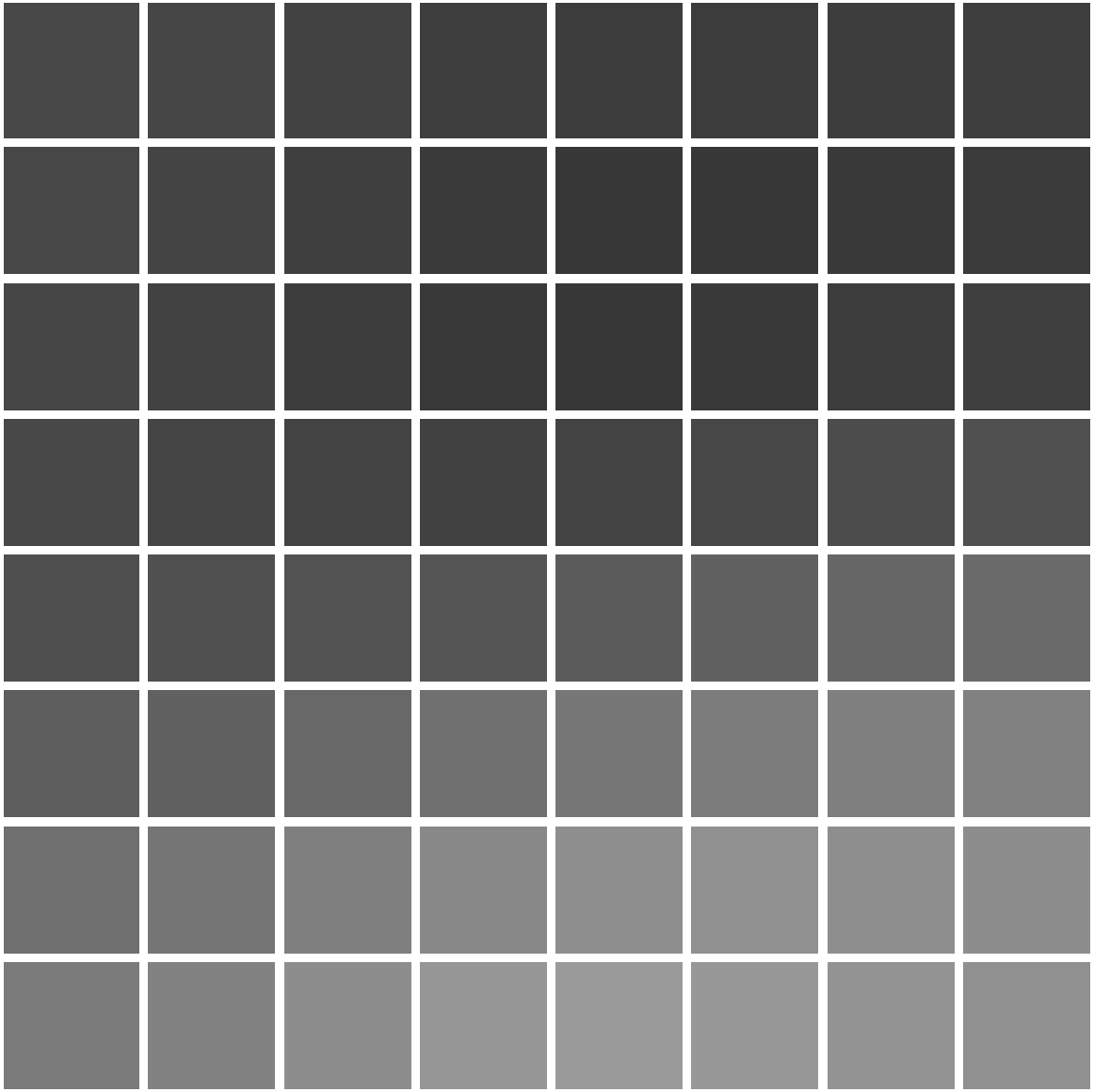
Retrieving the Image



Input Image (Example)



Output Image (Example)



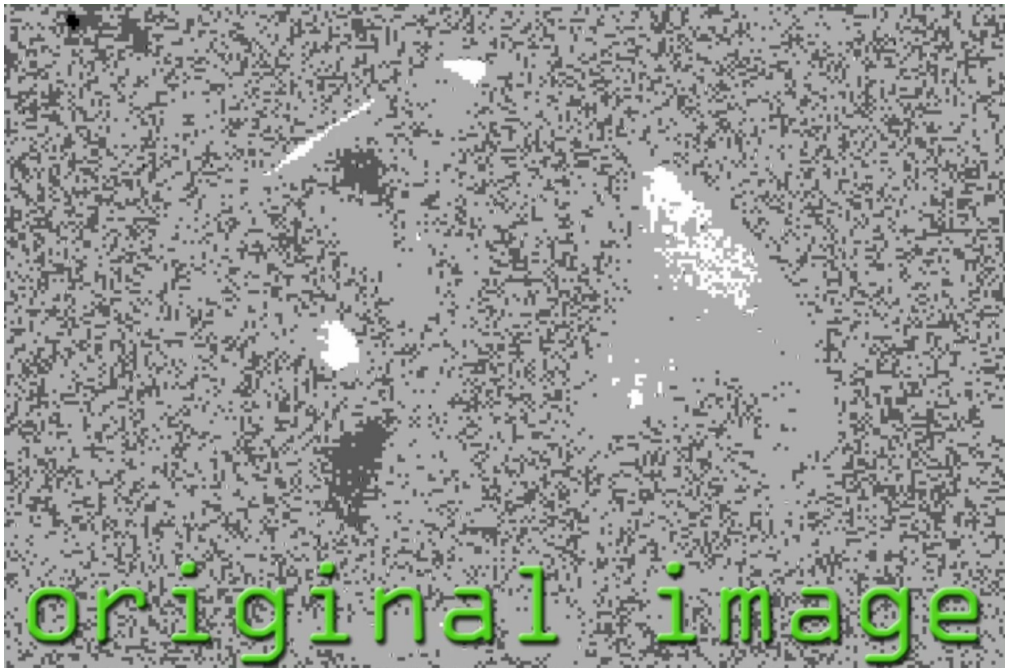
DCT Steganography

- We can hide data in the least significant bits of the values in the quantized tables.
- In practice, we don't alter the value in the upper-left corner (the DC coefficient) or any values that are 0 or 1.

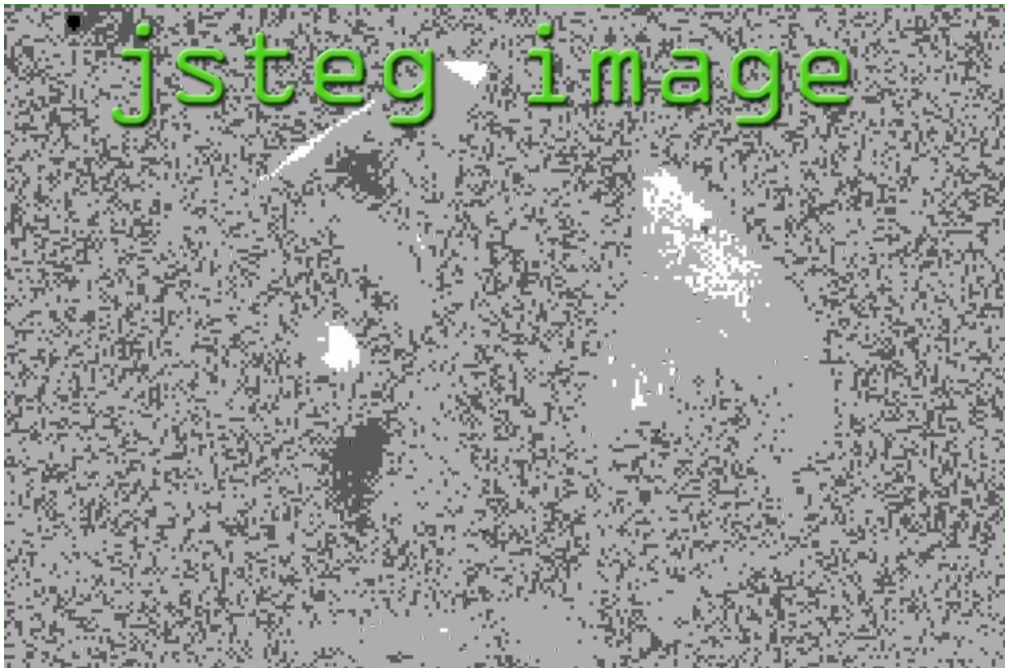
-23	-2	0	0	0	0	0	0
-21	4	2	0	0	0	0	0
6	1	0	0	0	0	0	0
1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized Table

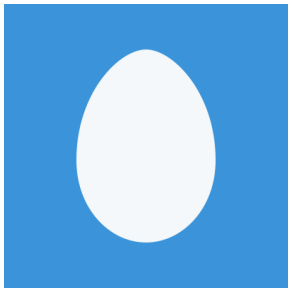
DCT Steganography Example



DCT Steganography Example



Other Examples (Images Converted to JPEG, 90% Quality)



- Twitter Profile Picture (400x400)
- We can hide:
  - 309 Bytes
  - ~0.30 KiB

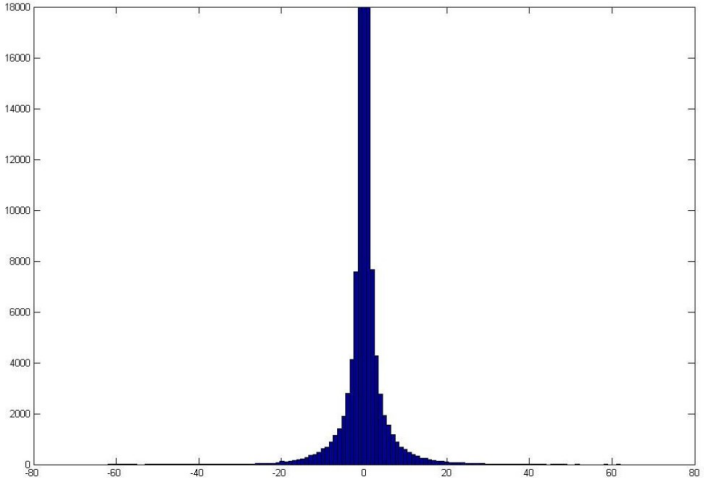
- 1080p Desktop Background (1920x1080)
- We can hide:
  - 28,955 Bytes
  - ~28.28 KiB

- NASA Image of Pluto (8000x8000)
- We can hide:
  - 778,313 Bytes
  - ~760.07 KiB

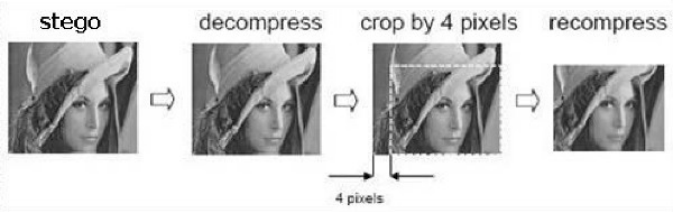
- Panorama of Milky Way (16702x6568)
- We can hide:
  - 2,791,778 Bytes
  - ~2.66 MiB

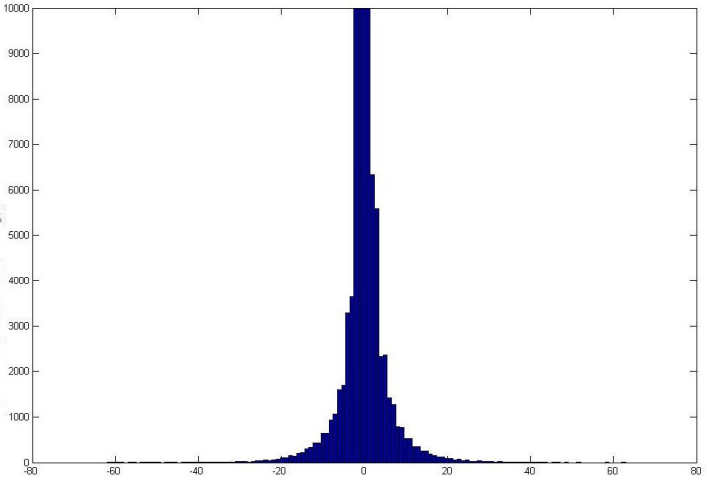
Detecting DCT Steganography

- The method is robust to visual attack, but…
- A histogram of the DCT coefficients in an image will normally be fairly symmetric about 0.
- Simple steganographic techniques will cause an image to deviate from this pattern.
- Analysis of this symmetry can usually estimate the percentage of the image used for data embedding within ~1%.



DCT Coefficients of unaltered JPEG

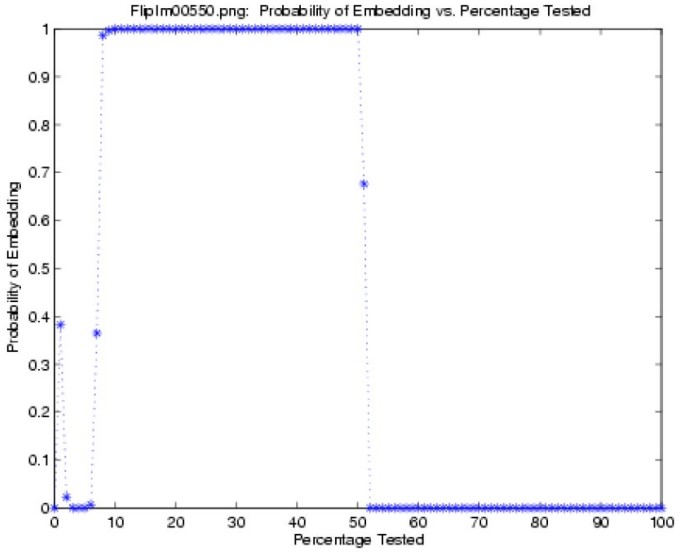




DCT Coefficients of steganographed JPEG

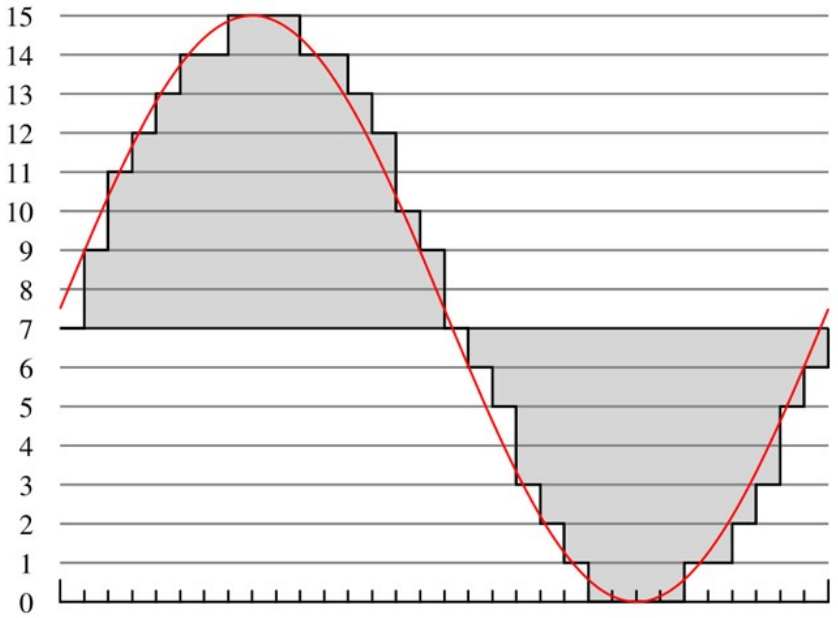
Detecting DCT Steganography

- In unaltered JPEG compression, rapid changes in color/intensity distort nearby pixels in a predictable manner.
- A chi-square test will tell us the probability of an image deviating from this expectation and the probability of the presence of embedded data.
- Accuracy depends on the significance level used.
  - Lower significance levels result in more false negatives.
  - Higher significance levels result in more false positives.



Extension to Audio

- Similar to images, audio files are composed of a series of integers representing amplitude at any given time.
- These integers can be represented as binary strings for Least Significant Bit embedding.



Questions?