Introduction to More Advanced Steganography

John Ortiz

Crucial Security Inc.

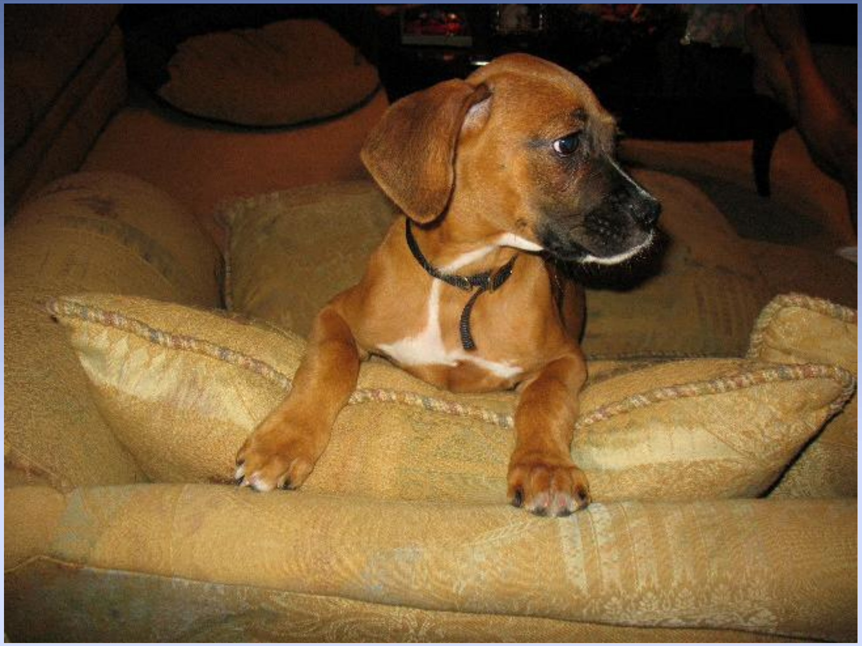
San Antonio

John.Ortiz@Harris.com

210 977-6615

Can YOU See the Difference?

- Which one of these pictures has a secret message?



- One of them contains 74108 bytes of secret data.
- The original is 574,421 bytes (new size 450,072)
- That’s 16.47% data hiding capacity

A Closer Look – Picture #1

A Closer Look – Picture #2

Who's Hiding in There?

Agenda

- Attention Getter
- A little bit about me
- Overview
- Hiding in the Least Significant Bit
- Advanced Techniques for Geeks
  - Bit Plane Complexity Segmentation (BPCS)
  - Hiding in Compressed Jpeg Images
- Questions/Comments/Complaints

About Me – the 20 min Version

- **I’m a GeeK**

Enough About Me

Overview

Overview

- Information Hiding is a branch of computer science that deals with concealing the existence of a message
  - It is related to cryptography whose intent is to render messages unreadable, except by the intended recipients
- It employs technologies from numerous science disciplines:
  - Digital Signal Processing (Images, Audio, Video)
  - Cryptography
  - Information Theory\Coding Theory
  - Data Compression
  - Discrete Math
  - Data Networks
  - Human Visual/Auditory perception

Overview

- **There are four primary sub-disciplines of Information Hiding**
  - Steganography
  - Watermarking
  - Covert Channels
  - Anonymity

Goals of Steganography

- **Steganography's primary goal is to hide data within some other data such that the hidden data cannot be detected <u>even if it is being sought</u>**

Goals of Steganography

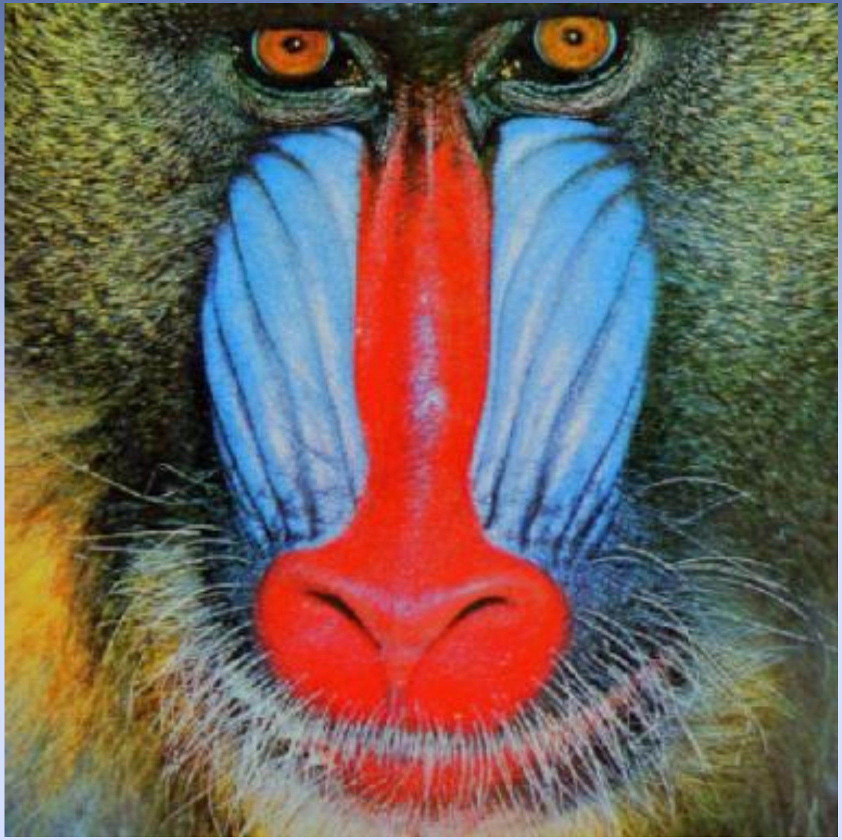
- Security
  - Perception, automated detection, levels of failure
- Capacity
  - Maximize amount of hidden data
  - Tradeoff with security/robustness
- Robustness
  - Resilience to stego file alterations

Less Advanced Steganography

Basic Hiding

We're Gonna Hide This:

In This:



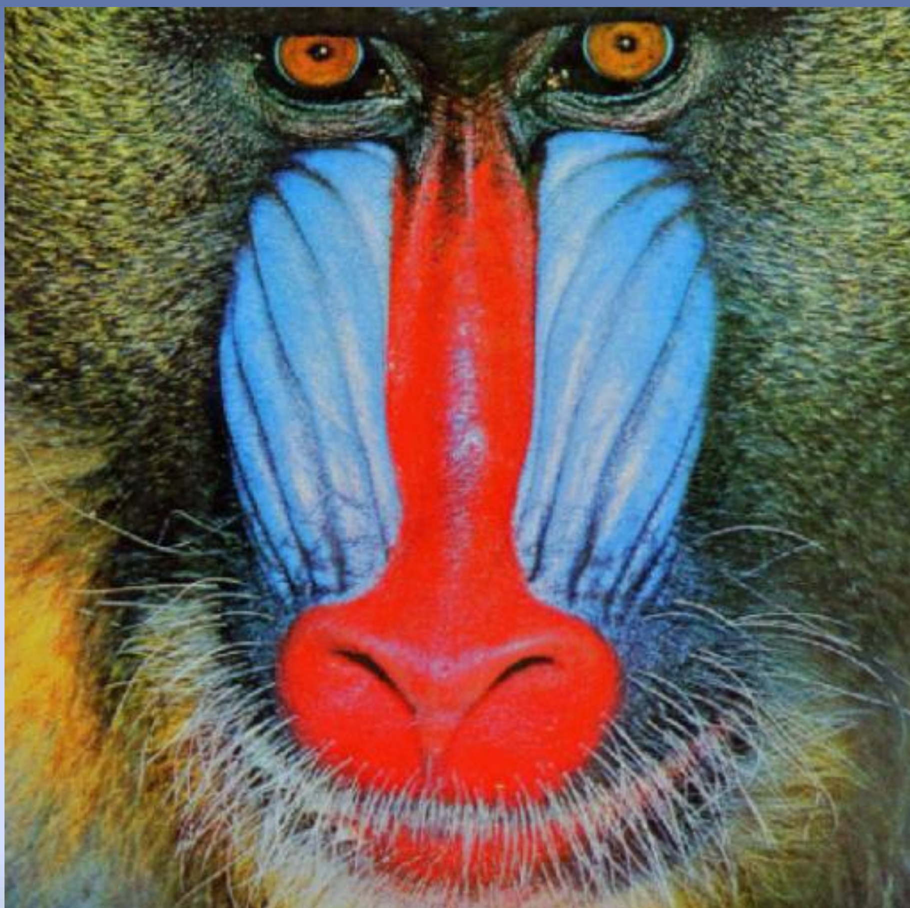
Then That In This:

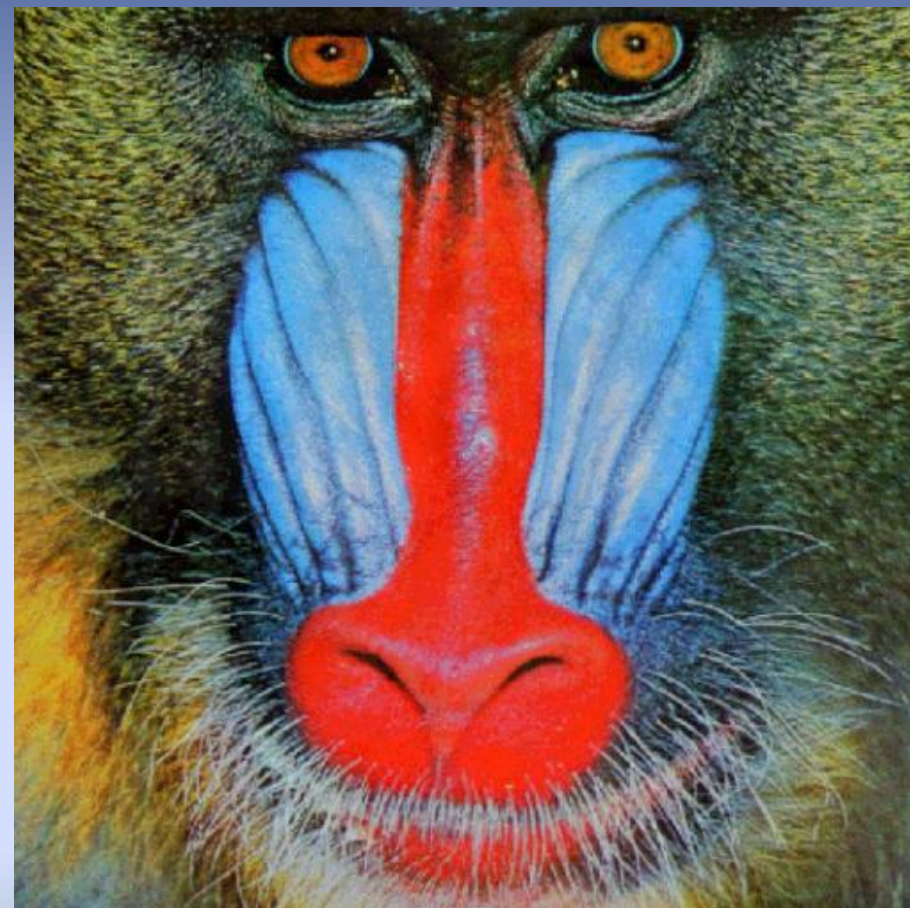
Least Significant Bits

- **Substitution: Replace information in the cover with the stego-message**
- **Most common: replace the Least Significant Bit (LSB)**
- **Each pixel in the next image is composed of 24 bits**
  - **8 bits for RED, 8 for GREEN, and 8 for BLUE (RGB)**
  - **Lower four bits of each color, hold the upper 4 bits of the hidden picture's colors in each corresponding pixel**
- **Other images with more solid backgrounds would NOT provide the same level of imperceptibility**
  - **To maximize capacity while maintaining imperceptibility, the cover image is a consideration**

Can YOU See a Difference?

- **The Dalmatian is hiding in 4 bits of the Mandrill**



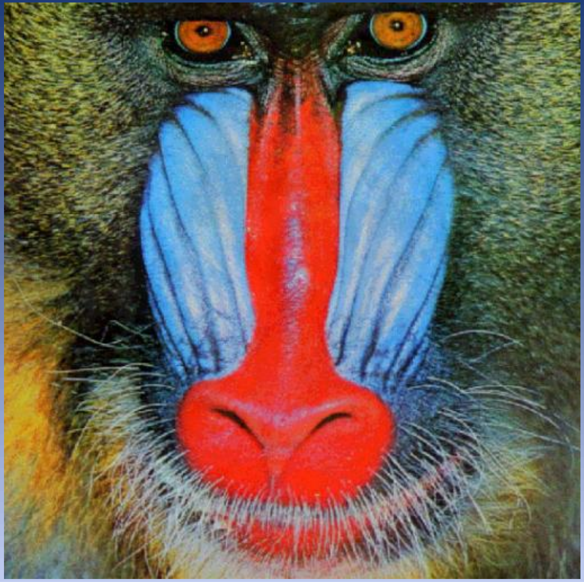


You CAN See a Difference!

- **More uniform colors in the cover is NOT effective**

Limitations

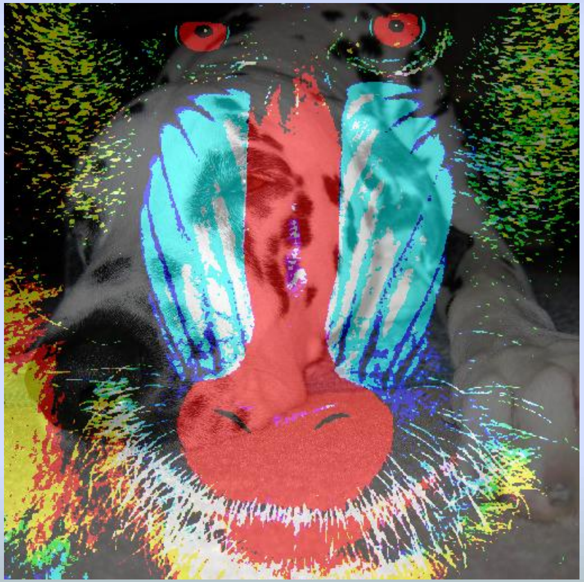
4



5

6

7



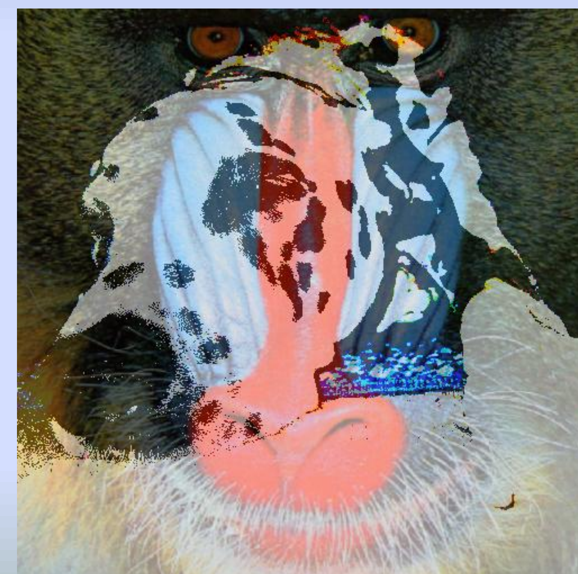
Limitations

4

5

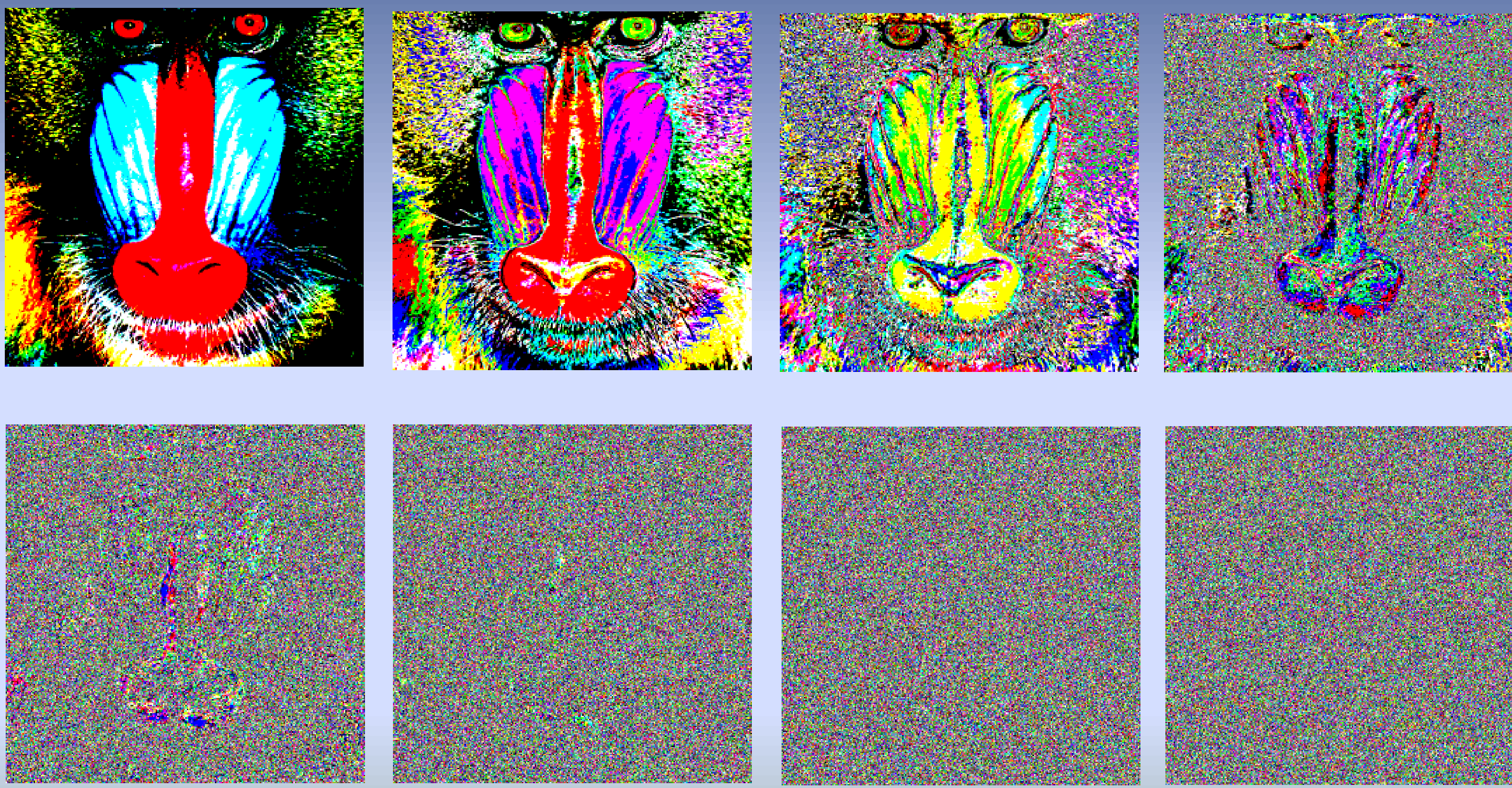
6

7

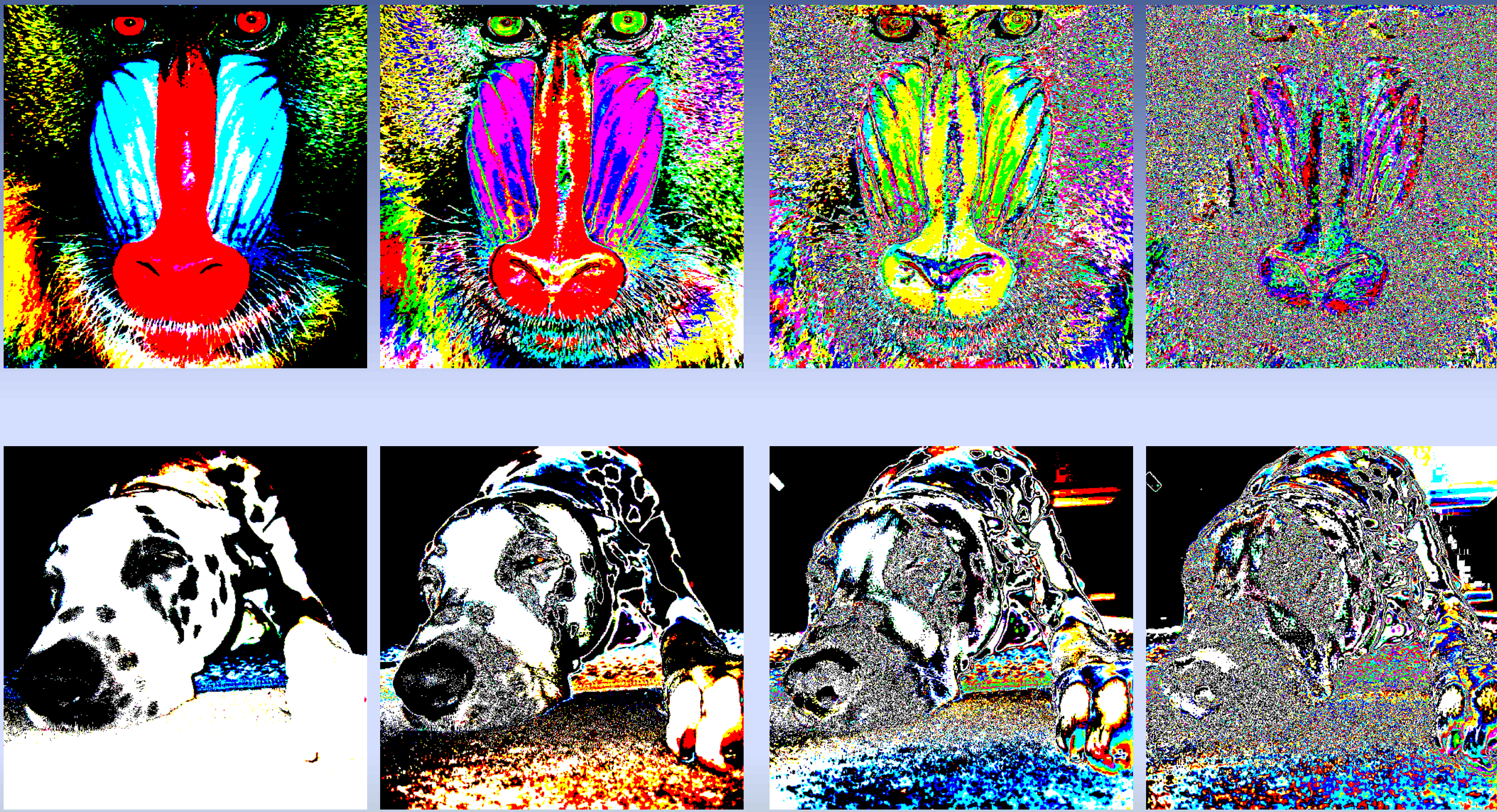


Bit Planes

- **No Hidden Image**



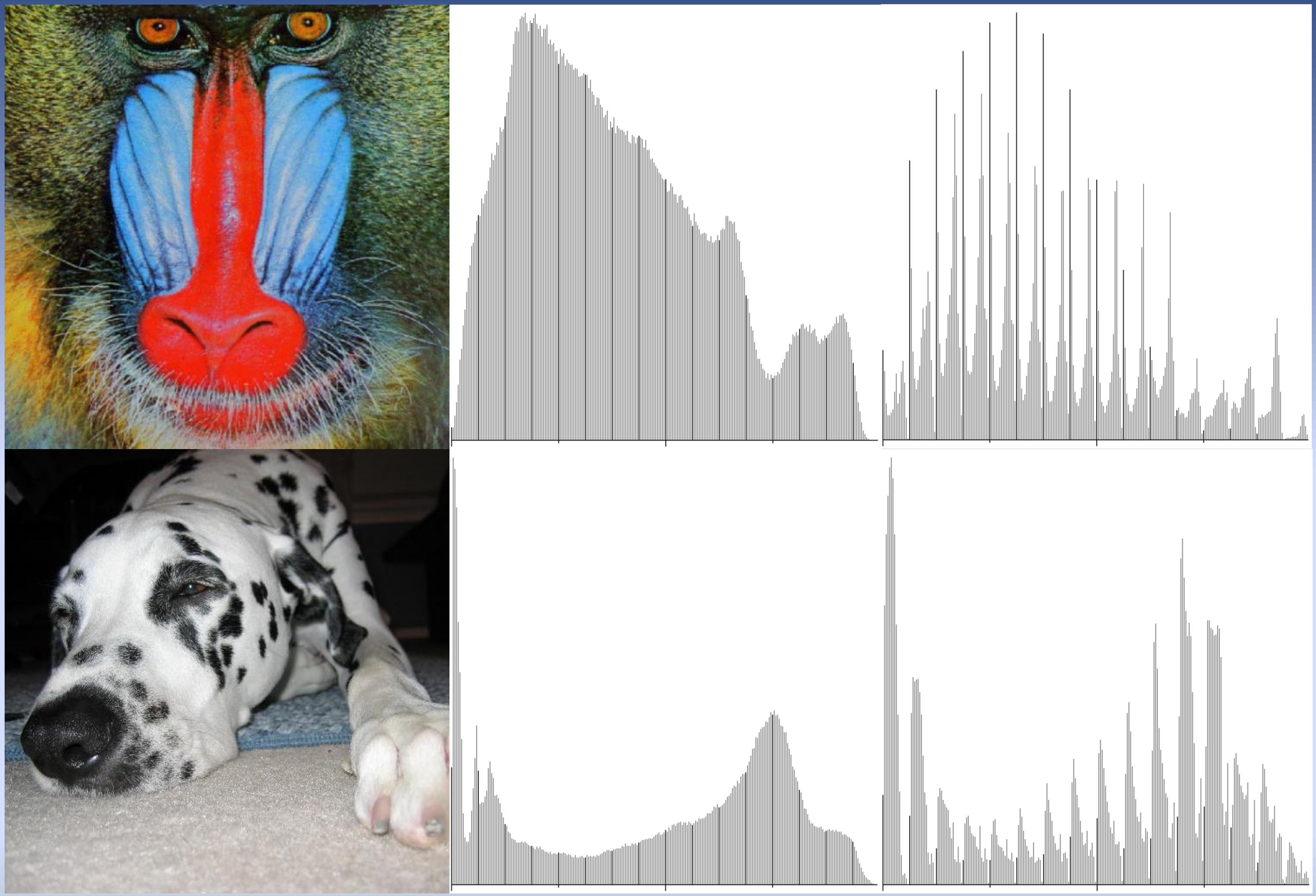
Bit Planes



Least Significant Bits

- **This particular technique substitutes image bits of one picture into another**
- **Both pictures must be the same size**
- **More typical is to substitute bits from the message one by one**
  - **Then, the message can be anything**
- **Easily detectible**
  - **Examine the histograms for anomalies**
  - **We can slice the image into bit planes**

Least Significant Bits - Histograms



More Advanced Steganography

Bit-Plane Complexity Segmentation (BPCS)

Bit Plane Complexity Segmentation

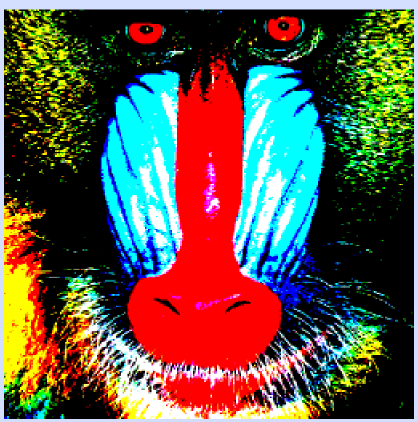
- More advanced Substitution Technique
- Little less capacity
  - Harder to detect
  - Harder to extract
- Message is spread across several bit planes
  - Possibly even the Most Significant Bit (MSB) plane

- I am going to skip some implementation details

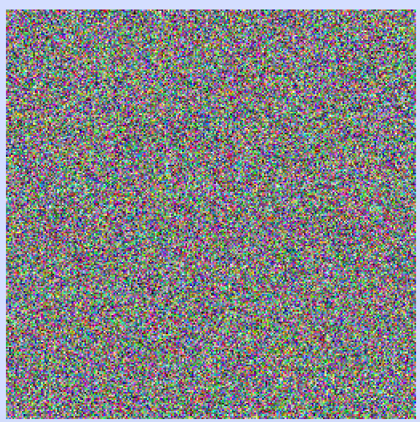
Bit Plane Complexity Segmentation

- **Hides in areas of image that are "complex"**
  - **The mandrill has a large number of complex areas**
  - **The dalmatian has much fewer complex areas**
- **The Least Significant bit plane is complex for both**

MSB

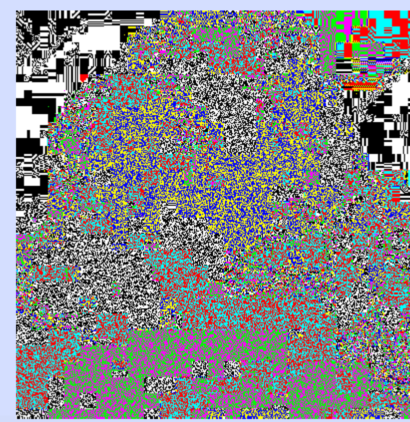


LSB



MSB

LSB



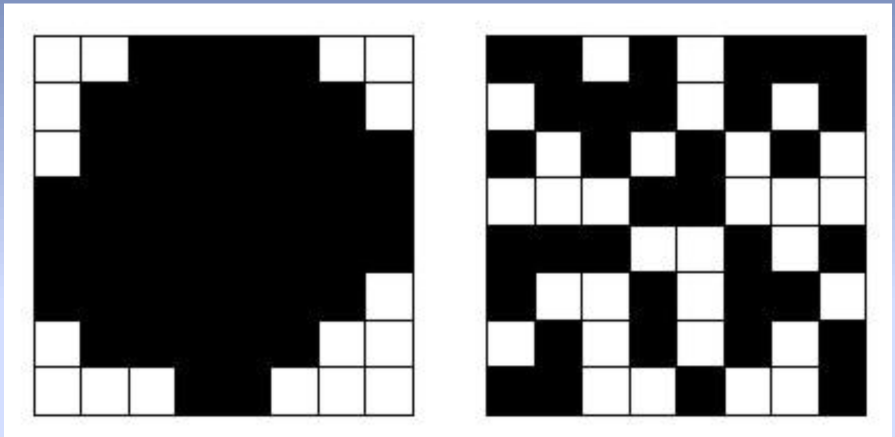
Bit Plane Complexity Segmentation

- A complexity measure is taken for each 8 x 8 matrix
- No standard complexity measure
- The one used in the initial paper is a black and white border length complexity measurement
  - If the border length is long, the image is complex
  - This technique can fail
- The total length of the border is the sum of the number of black/white changes along the rows and columns
  - Remember, we are using the measure on bit planes, so every pixel is either a one or zero
  - Ex. A black pixel, surrounded by all white, has a border length of 4

Bit Plane Complexity Segmentation

- Left: A simple block with low complexity
- Right: A complex block

$$\alpha = \frac{k}{M}$$



- α_{th} is the threshold
  - Border length over total
  - Determined to be around 0.3
  - Must be less than 0.5 (we'll see why shortly)

Bit Plane Complexity Segmentation

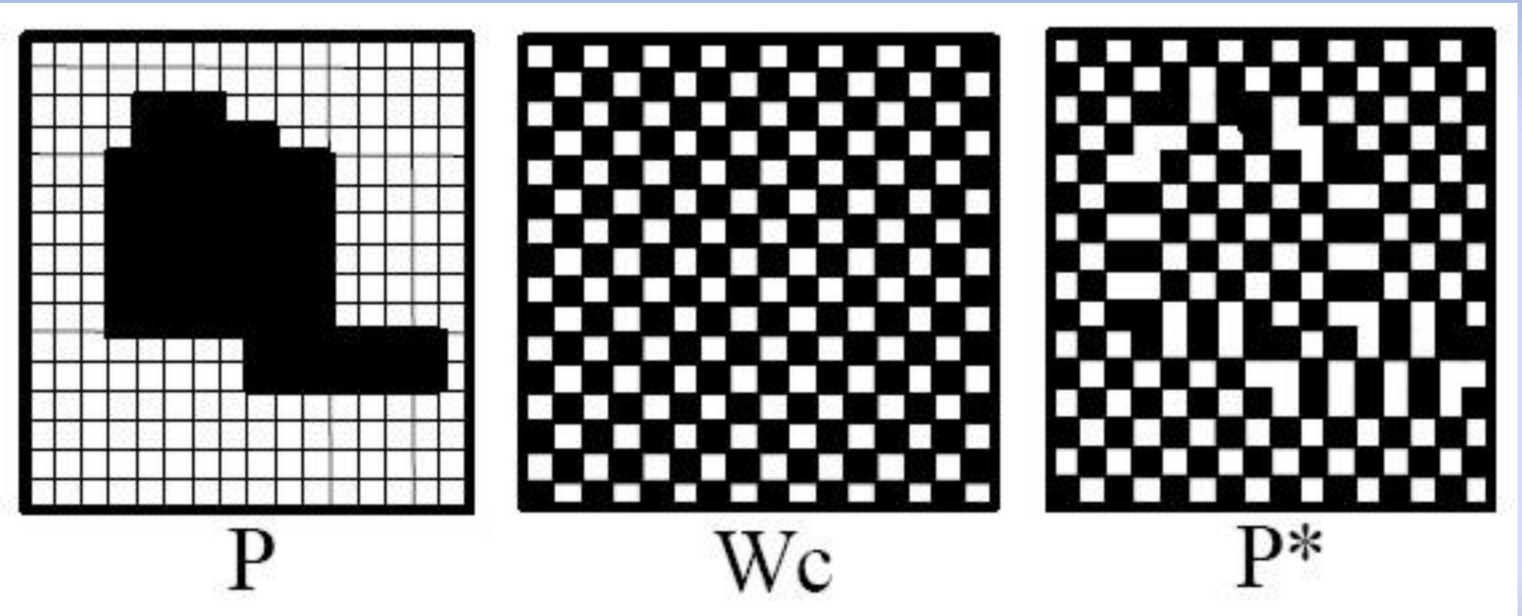
- If a region is complex enough, the image data is replaced by the message data
- The message data is first transformed into an 8x8 bit array, and that array is stored in place of the original data
- Does anyone see a problem during extraction?

Bit Plane Complexity Segmentation

- What if the message data itself is not complex?
  - During extraction, the region will no longer exceed the complexity threshold
  - No data will be extracted
- Must conjugate the resource data
- The 8x8 matrix is exclusive-or’d with a checkerboard pattern

Bit Plane Complexity Segmentation

- **Conjugation shown graphically**



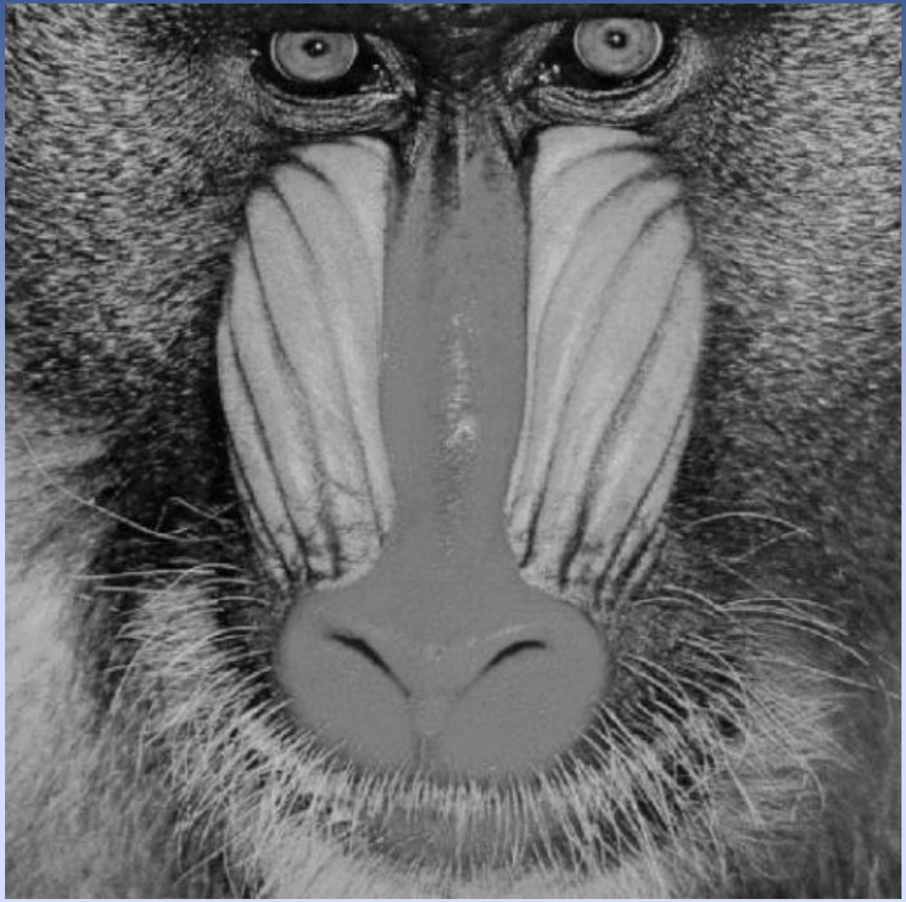
Bit Plane Complexity Segmentation

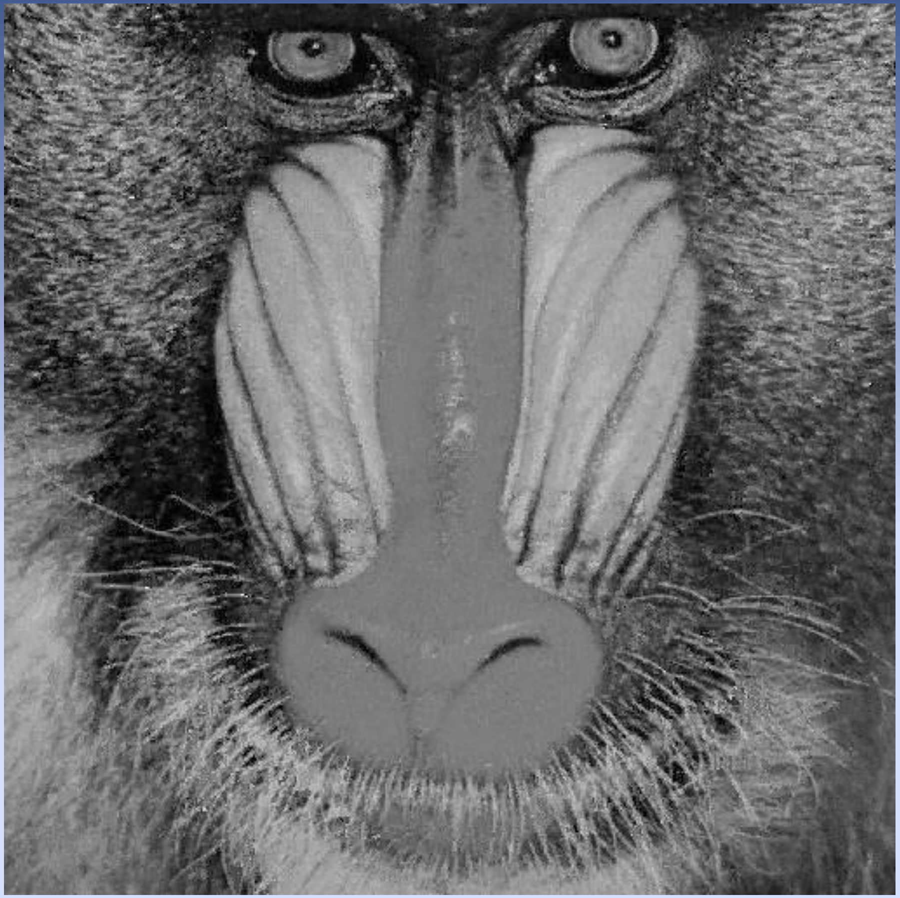
- The complexity of P* is ($1 - \alpha_P$)
- As long as α_{th} is less than 0.5, if P is not complex enough, P* will be
- Note: (P*)* = P → (a xor b) xor a == b
- This ensures that whenever information is embedded, the complexity will be greater than the threshold
- Now the problem is determining which regions are original data and which ones are conjugate data

Bit Plane Complexity Segmentation

- **Solution!**
- **Reserve one bit of each region to indicate conjugation**
  - Make the lower left bit of the 8x8 matrix a zero
  - If conjugation occurs, it will become a one
- **This does use 1/64 of your embedding capacity**
- **Other solutions proposed, this is the simplest**

Suggested Threshold of 0.3

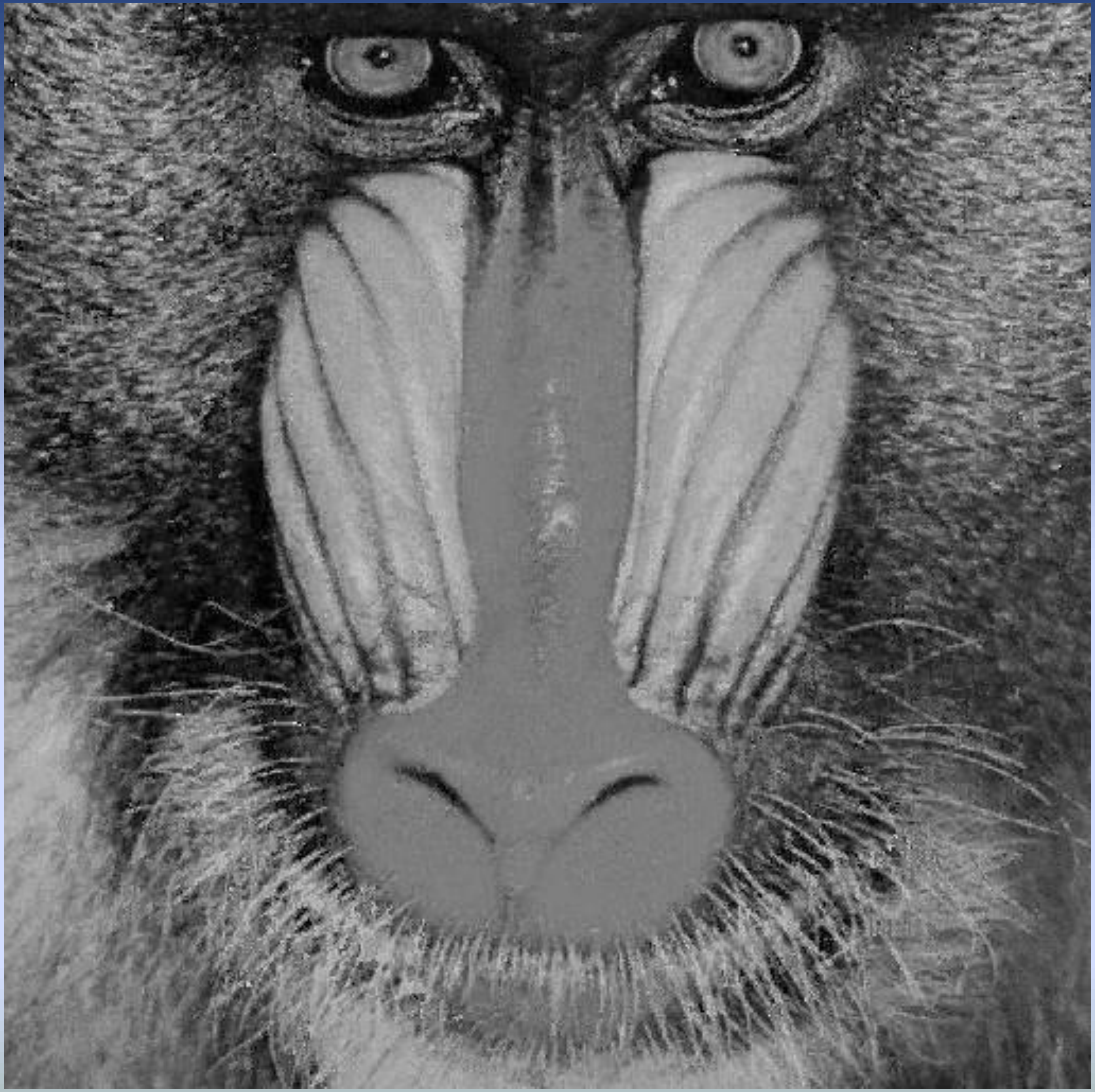




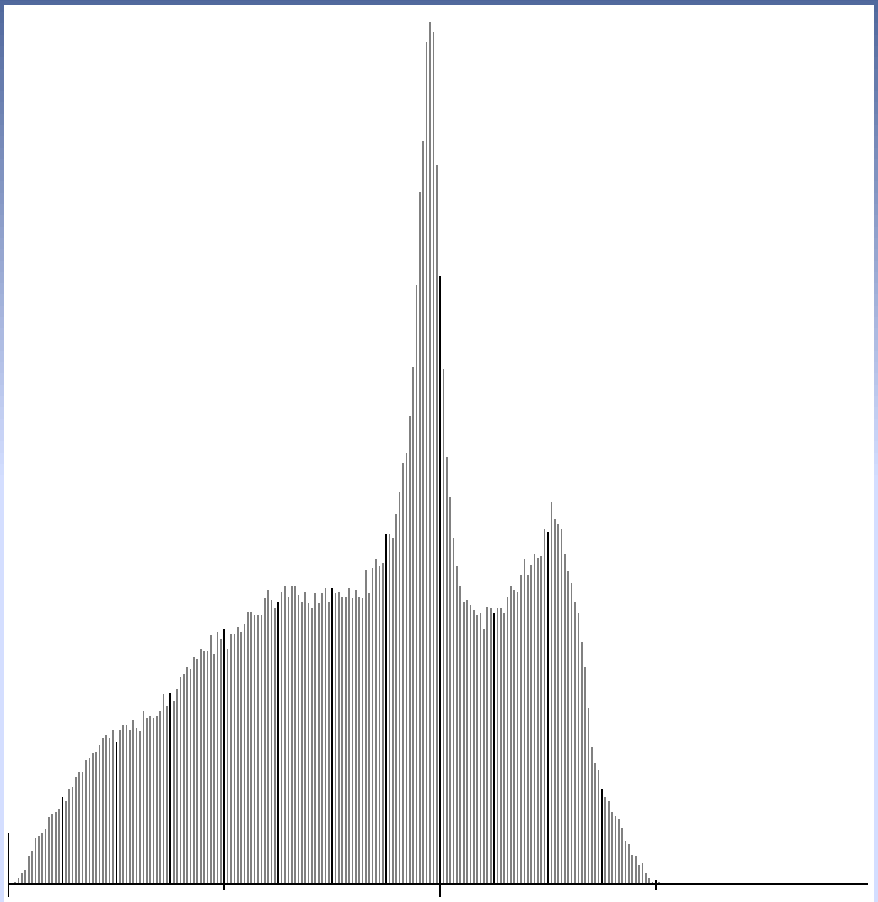
Thresh=0.3, cap = 134KB/258KB

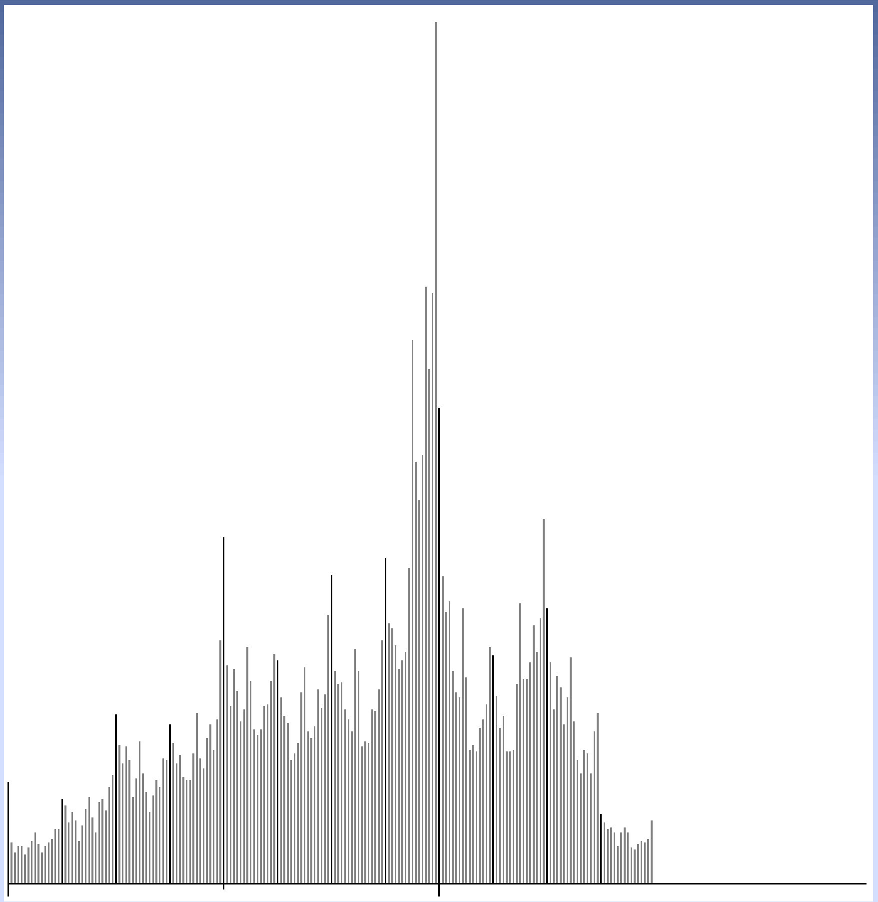
Unmodified

Thresh=0.3, cap = 134KB/258KB

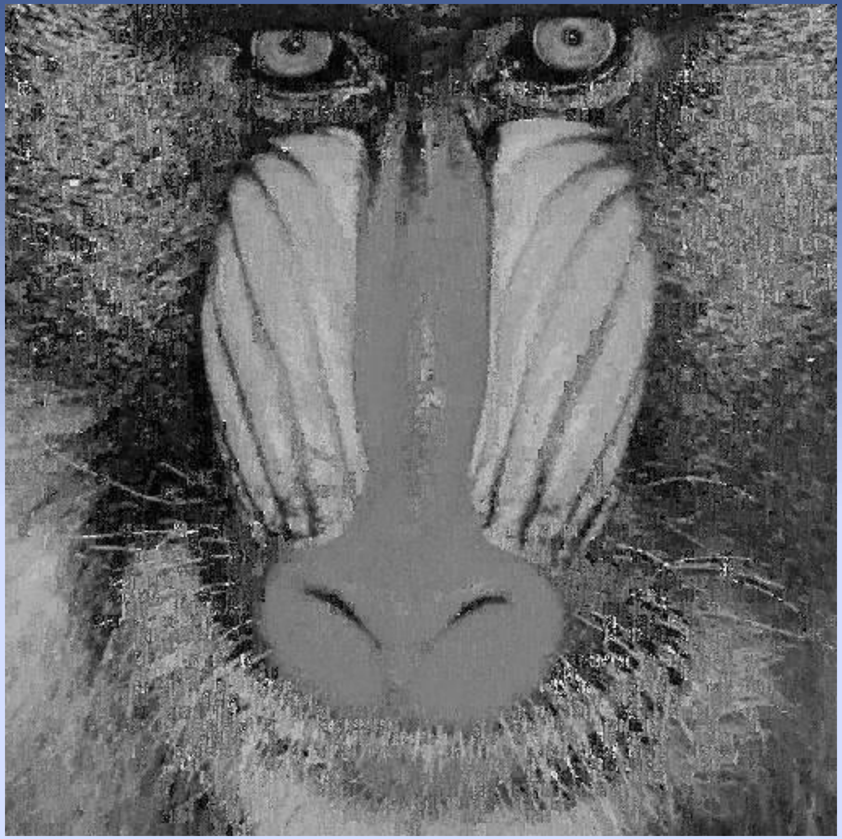


Bit Plane Complexity Segmentation



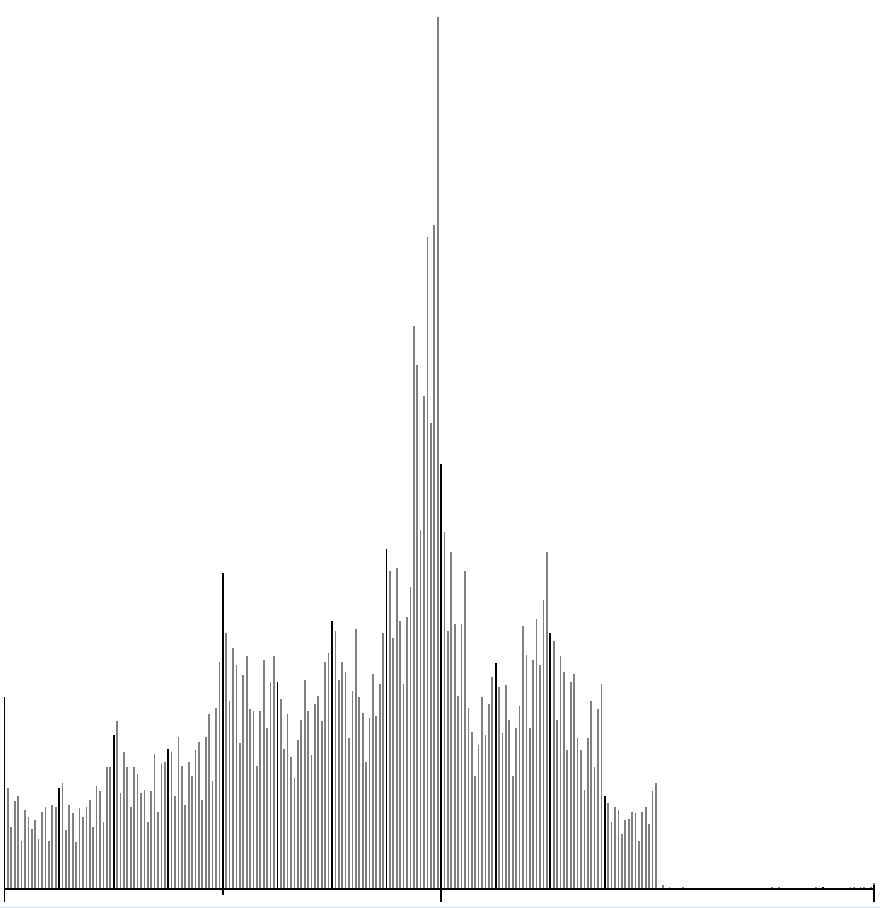


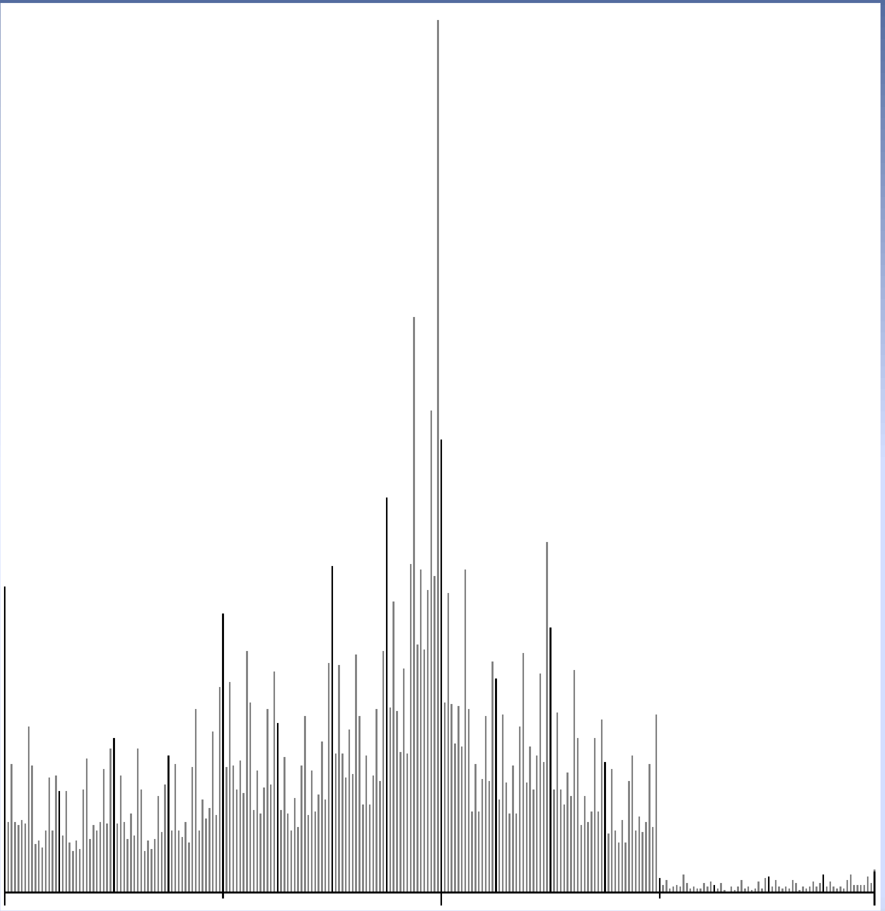
Lower Complexity Threshold



Thresh=0.2, cap = 163KB/258KB **Thresh=0.1, cap = 193KB/258KB**

Lower Complexity Threshold

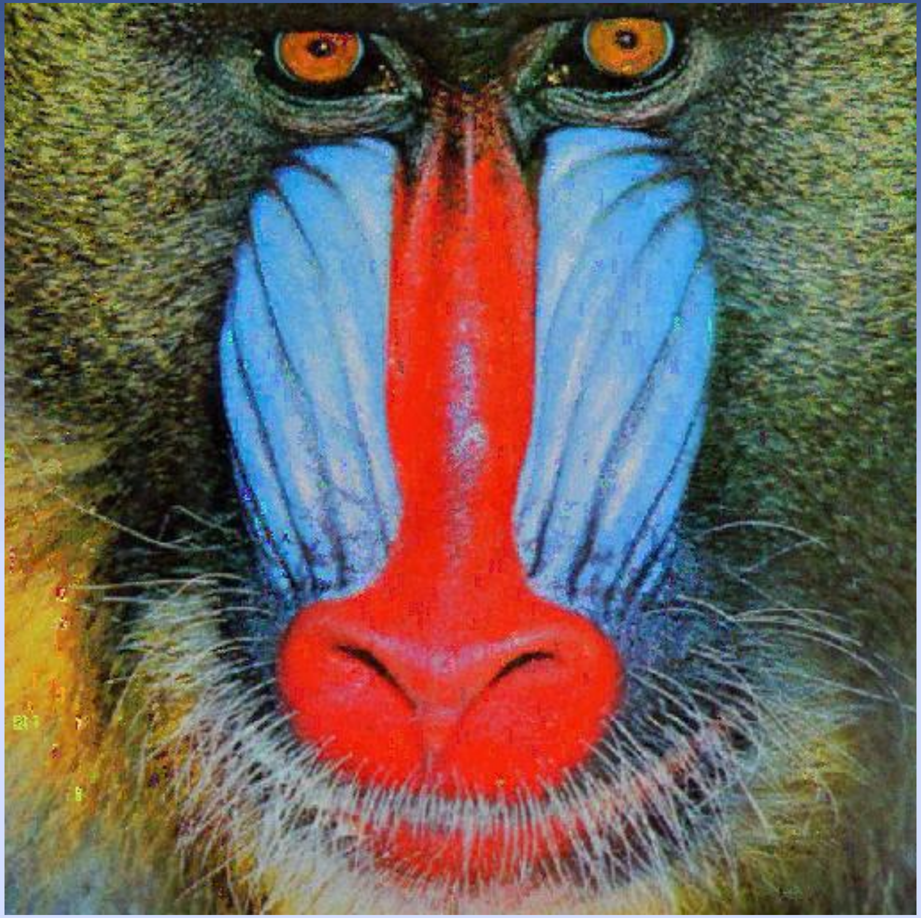




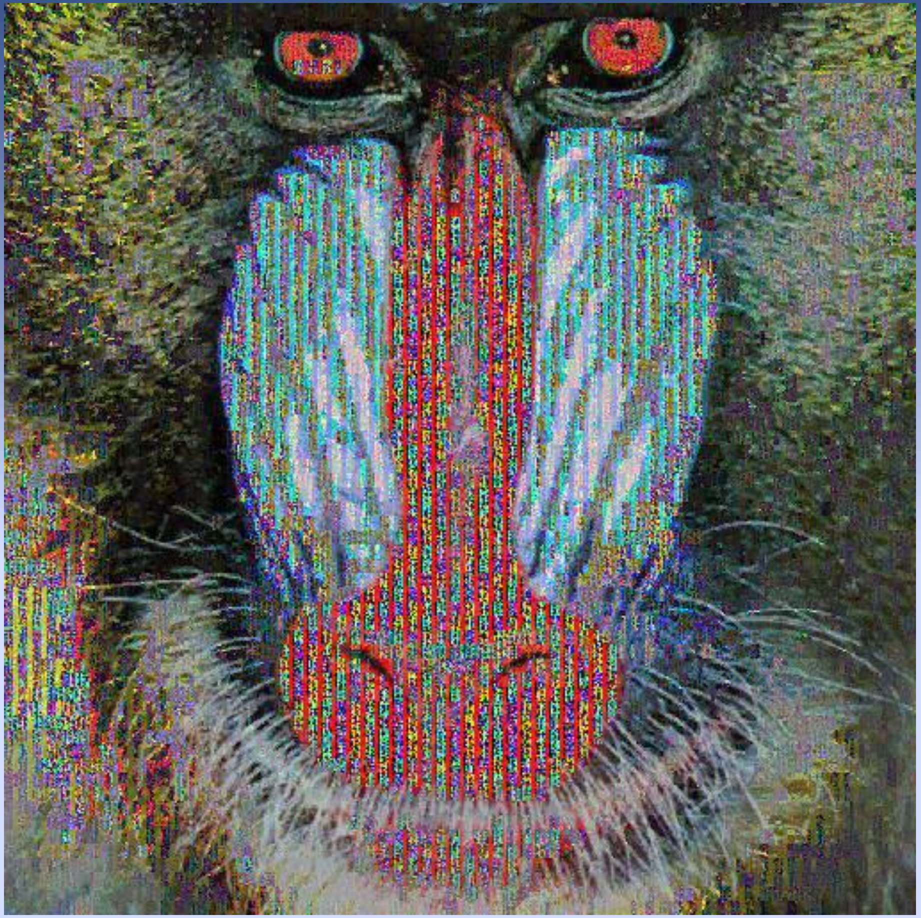
Less Complex Image

Thresh=0.3, cap = 106KB/258KB

Color BPCS



Thresh=0.4, cap = 405KB/769KB



Thresh=0.3, cap = 557KB/769KB

Bit Plane Complexity Segmentation

- **The cover image matters!!!**
- **Other authors proposed better complexity measures**
  - **Less perceptible**
  - **BUT, less capacity**

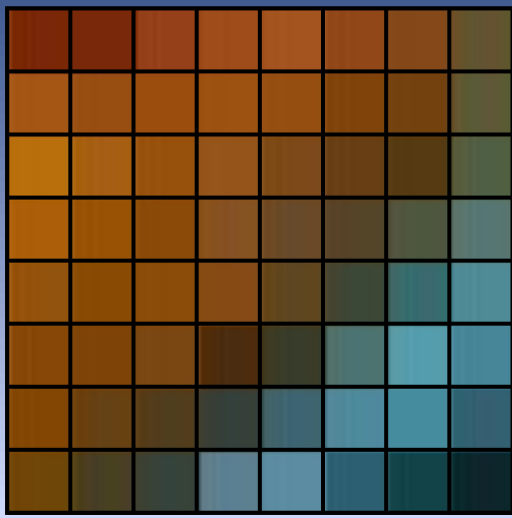
More Advanced Steganography

Transform Domain

Transform Domain

- **Transform domain** methods hide data in <u>significant</u> portions of the cover
  - as opposed to least significant
- Generally more robust to manipulation
  - affine transforms
    - scaling, rotating, shearing, translating, flipping
  - lossy compression
  - analog to digital and digital to analog conversions

Jpeg Process

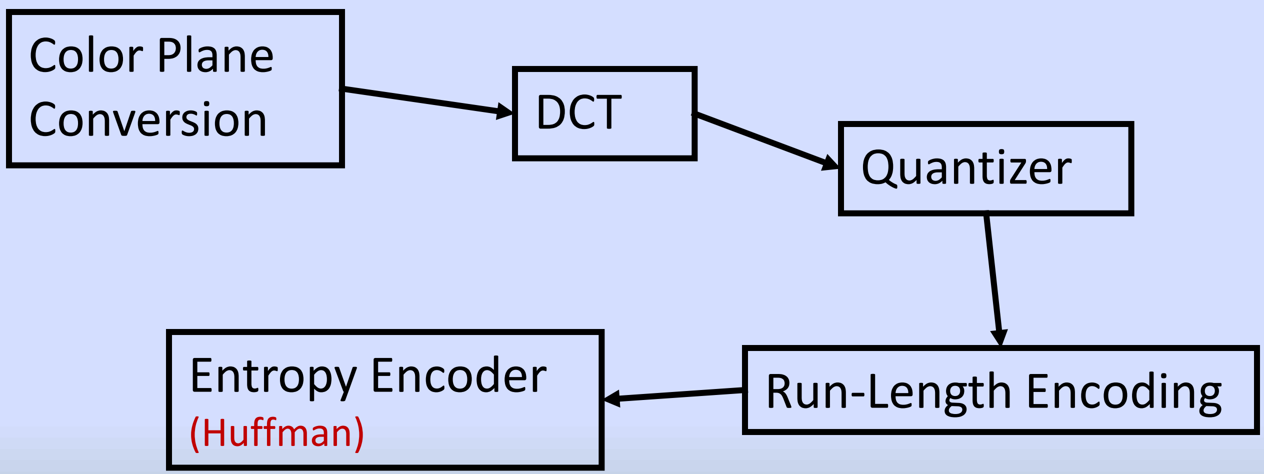


8 X 8
Image
Block

Quantization Table

(0,0)

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



JPEG Process

- **Converts color RGB to YC_rC_b**
  - Y is the luminance component
  - C_r & C_b are the chrominance components
  - Grayscale images only have the Y component
- **The image is divided into 8 x 8 blocks**
- **A 2-dimensional Discrete Cosine Transform (DCT) is performed on each**

JPEG Process

- Results is quantized according to desired quality
  - The quantization is the primary “lossy” part
- A combination of Run-Length Encoding (RLE) and Huffman coding is applied to finish the compression
  - This process is lossless
- To get the image back, the process is reversed
- The restored image is similar in appearance, but mathematically different from the original
- If high quality is used, there is little, if any, perceptible difference

DCT Hiding Technique

- "High Capacity Data Hiding in JPEG Compressed Images"
  - Chang, C.C. and Tseng, Hsien-Wen
- An adaptive Discrete Cosine Transform, Least Significant Bit technique
  - Hides in lower and middle frequency components
  - Adapts to different characteristics of each block
  - Performs capacity estimation
- >> Greater than 1 bit per 8x8 block

DCT Hiding Technique

- **Capacity Estimation**
  - Determine max number of bits that can be modified while remaining imperceptible
- **Uses a capacity table based upon the quantization table**
  - user sets an α (alpha) factor
    - higher α, higher bit rate, but increased distortion
  - lower frequency components hold fewer bits
  - higher frequency can hold more bits, but there are fewer

DCT Hiding Technique

- Each table is 8x8, we'll use x,y to denote a specific element
- $C_Q (x,y) = \lg(\alpha * Q(x,y))$
  - Capacity based on Quantization table
- $M (x,y) = \lg (| D(x,y) |)$
  - Capacity based on DCT coefficients
- Use the lower of these two

DCT Hiding Technique

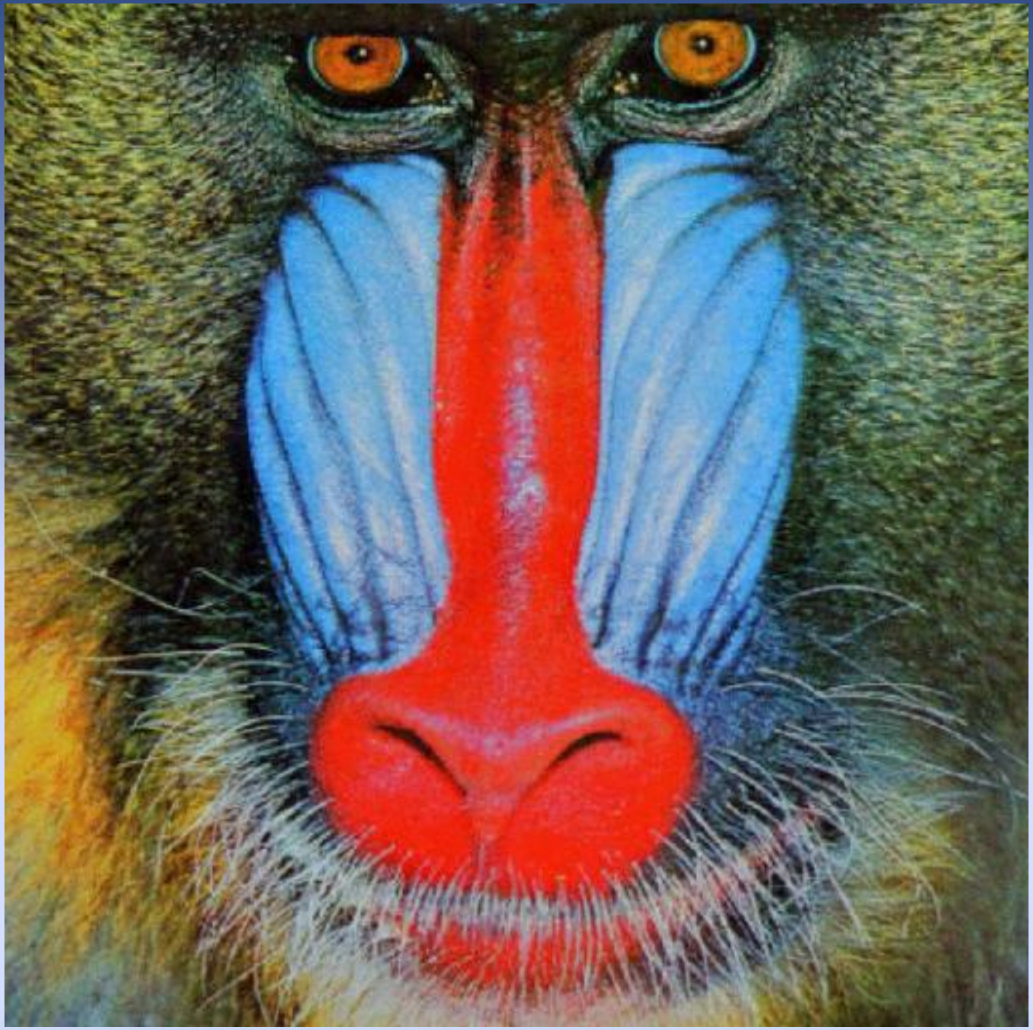
- Block Classification – determine which blocks are better candidates for hiding
- If a background has a strong texture, the Human Visual System (HVS) is less sensitive to distortions
- Blocks divided into two classes:
  - uniform blocks
  - non-uniform blocks
- Non-uniform blocks use a larger α value (1.2 * α)
- D_x is the x^{th} AC coefficient
- If G is below a threshold, the block is uniform

$$G = \sqrt{\sum_{x=1}^{63} (D_x)^2}$$

Embedding Algorithm:

- **Set the α value**
- **Choose the block to be embedded**
- **Determine classification of block:**
  - **uniform, non-uniform**
- **Determine number of bits to hide in each *quantized* DCT coefficient**
- **Embed the data**
- **Apply the normal JPEG entropy coding**

High Capacity Example #1 (quality=95%)



Mandrill512.bmp_q95_a8_u8.jpg ---> 71745/ 71745 22.24% of stego

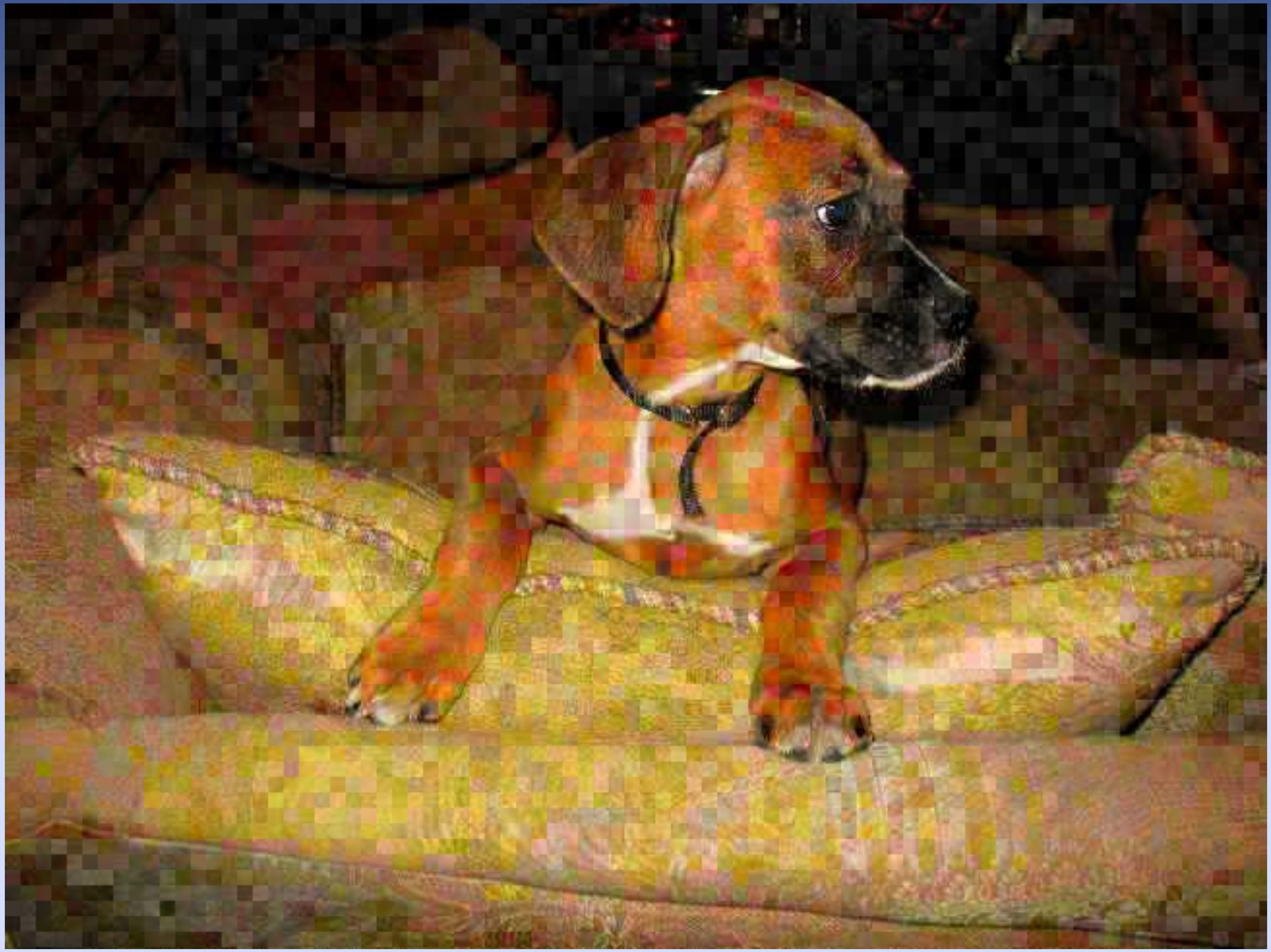
High Capacity Example #2 (quality=95%)

Domino512.bmp_q95_a8_u8.jpg ---> 38827 / 38827 22.07% of stego

High Capacity Example #3 (quality=99%)

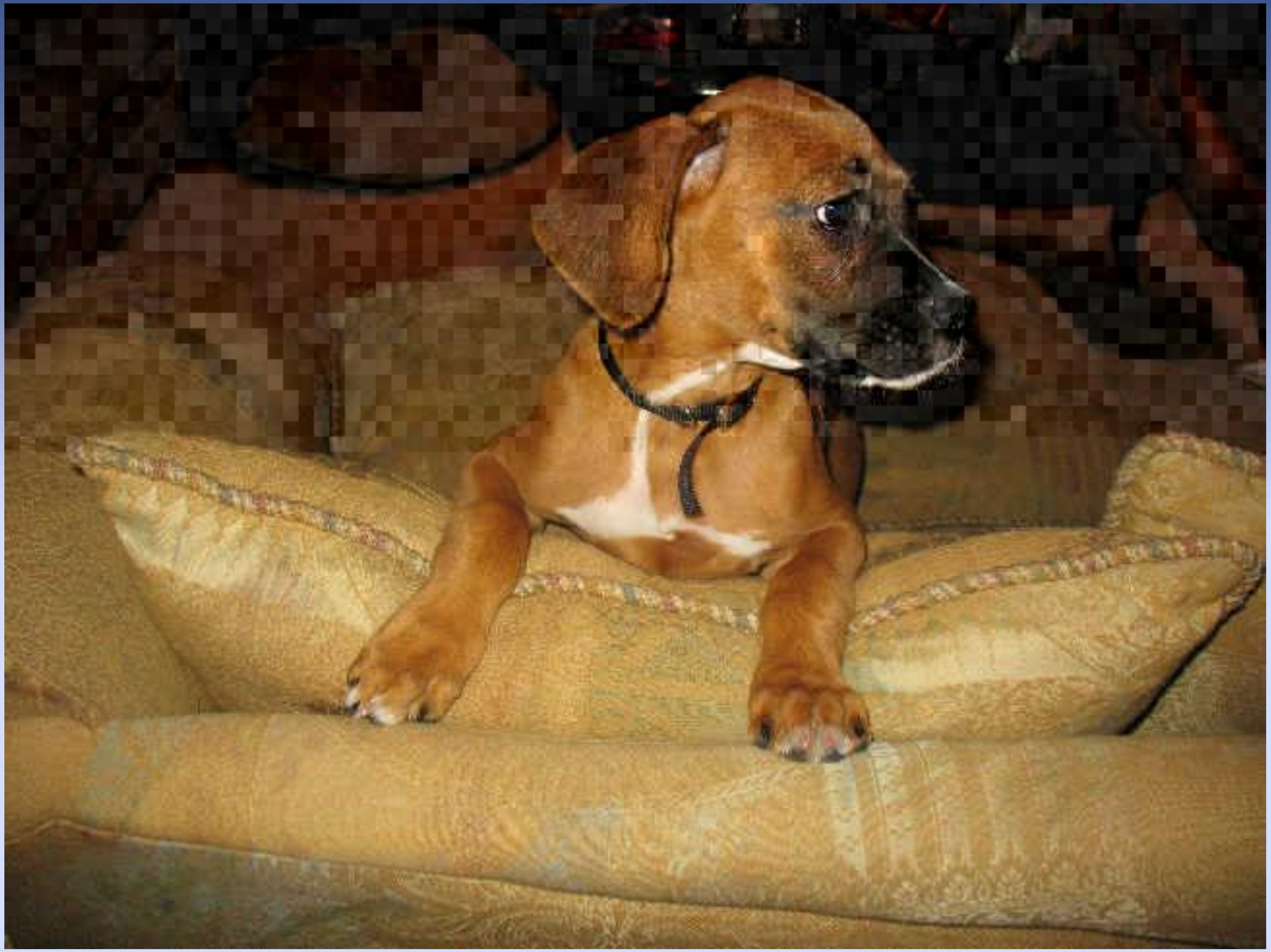
S2_Rocky.jpg_q99_a8_u8.jpg ---> 107643 / 107643   21.06% of stego

High Capacity Example #4 (quality=50%)



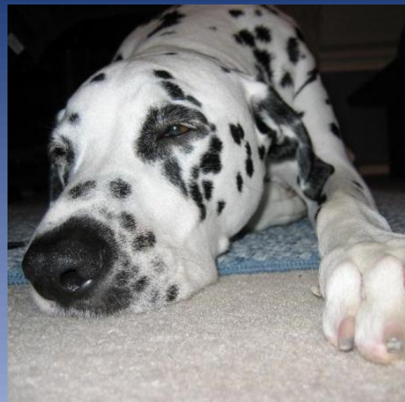
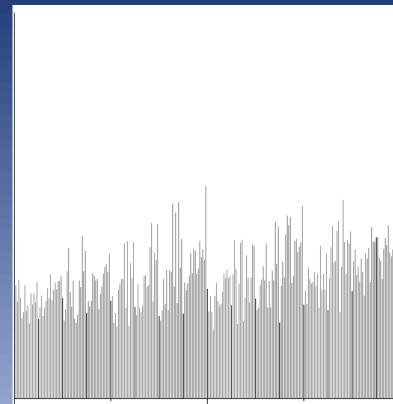
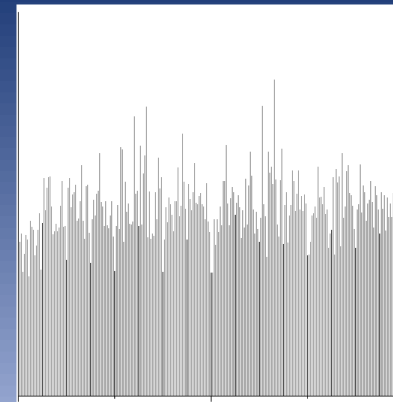
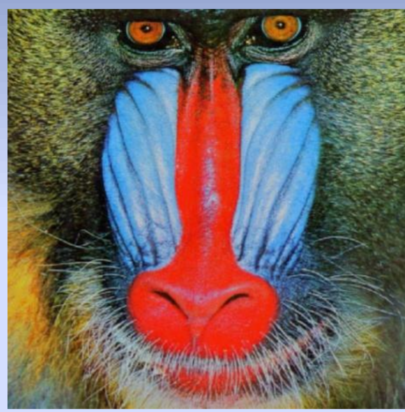
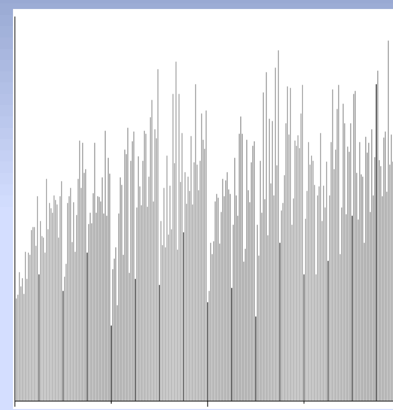
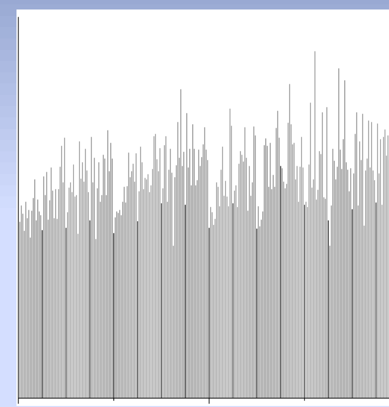
S2_Rocky.jpg_q50_a8_u8.jpg ---> 6612/ 6612   17.66% of stego

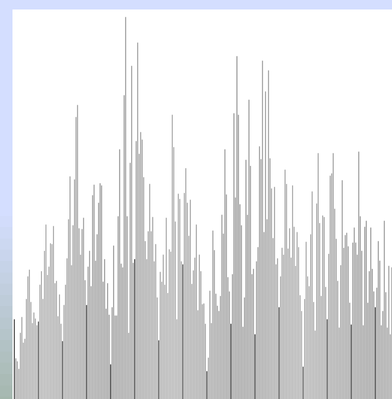
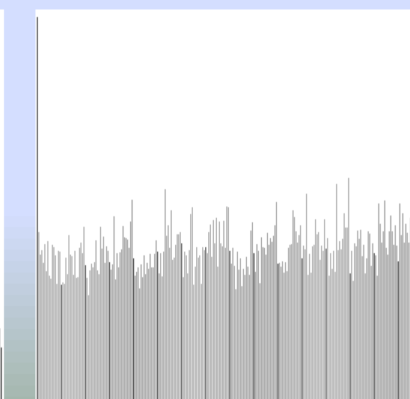
High Capacity Example #5 (quality=50%)



S2_Rocky_A.jpg_q50_a8_u8.jpg ---> **1575/ 6612**   18.81% of stego

Histograms Are Not Effective for Jpeg

Fix Your Hair and Take a Breath Now

What Questions Do You Have?

- **For more information or the actual software contact me @ John.Ortiz@Harris.com**

- **PLEASE COMPLETE PRESENTATION EVALUATIONS**