Cyber Security Fundamentals

Introduction to Steganography

This is Wally

Now, where is Wally?

Hiding in Plain Sight

- To be unnoticeable or inconspicuous.
- By staying visible in a setting that masks presence.
- If you're in a natural setting, wearing camouflage gear will allow you to blend in with your environment.
- **Steganography** – practice of concealing messages or information within other non-secret text or multi-media data

History of Steganography

The Evolution of Secret Writing

Herodotus tells us about: Demaratus, the Persian Invasion of Greece in 480 BC, and the Wax Tablets

Past use :

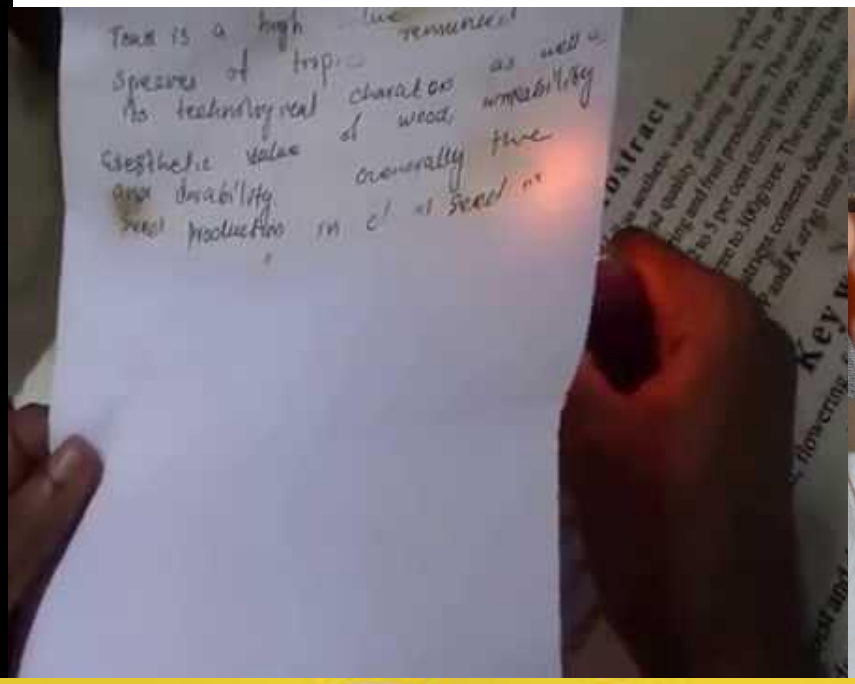
- Hidden messages within **wax tablets** — in ancient Greece, people wrote messages on the wood, then covered it with **wax** .

Figure 1: Wax tablet with stylus from Demaratus' time.

Steganographic history starts from the Histories of Herodotus:

Herodotus tells how Demeratus, a Greek at the Persian court, warned Sparta of an imminent invasion by Xerxes, king of Persia: he removed the wax from a writing tablet, wrote his message on the wood underneath and then covered the message with wax.

Non-Digital Steganography Examples



Lemon juice can be used for invisible writing; another example of steganography

Acrostics

In “Through the Looking Glass”, the final chapter "A Boat, Beneath A Sunny Sky” is an acrostic of the real Alice's name: Alice Pleasance Liddell.

A boat, beneath a sunny sky
Lingering onward dreamily
In an evening of July –
Children three that nestle near,
Eager eye and willing ear,

An acrostic is a poem that has a 'hidden word'

Steganography in Malware Delivery

Malware authors hide executable code inside an image file, and this code is extracted and run only after a number of existing security checks are passed.

Examples:
Stegoloader, LokiBot

SHA256 hashes of LokiBot:
ed5550d3047903d3e09363f90b6d49f519d1484af4e528fd95f1e5f3e5a008b2
691c65e4fb1d19f82465df1d34ad51aaeceba14a78167262dc7b2840a6a6aa87

- Steganography - the hiding of data in other content types such as images, audio-visual files, network traffic etc.
- Stegomalware – concealing malware, concealing C&C requests

Steganography in Cyber Espionage

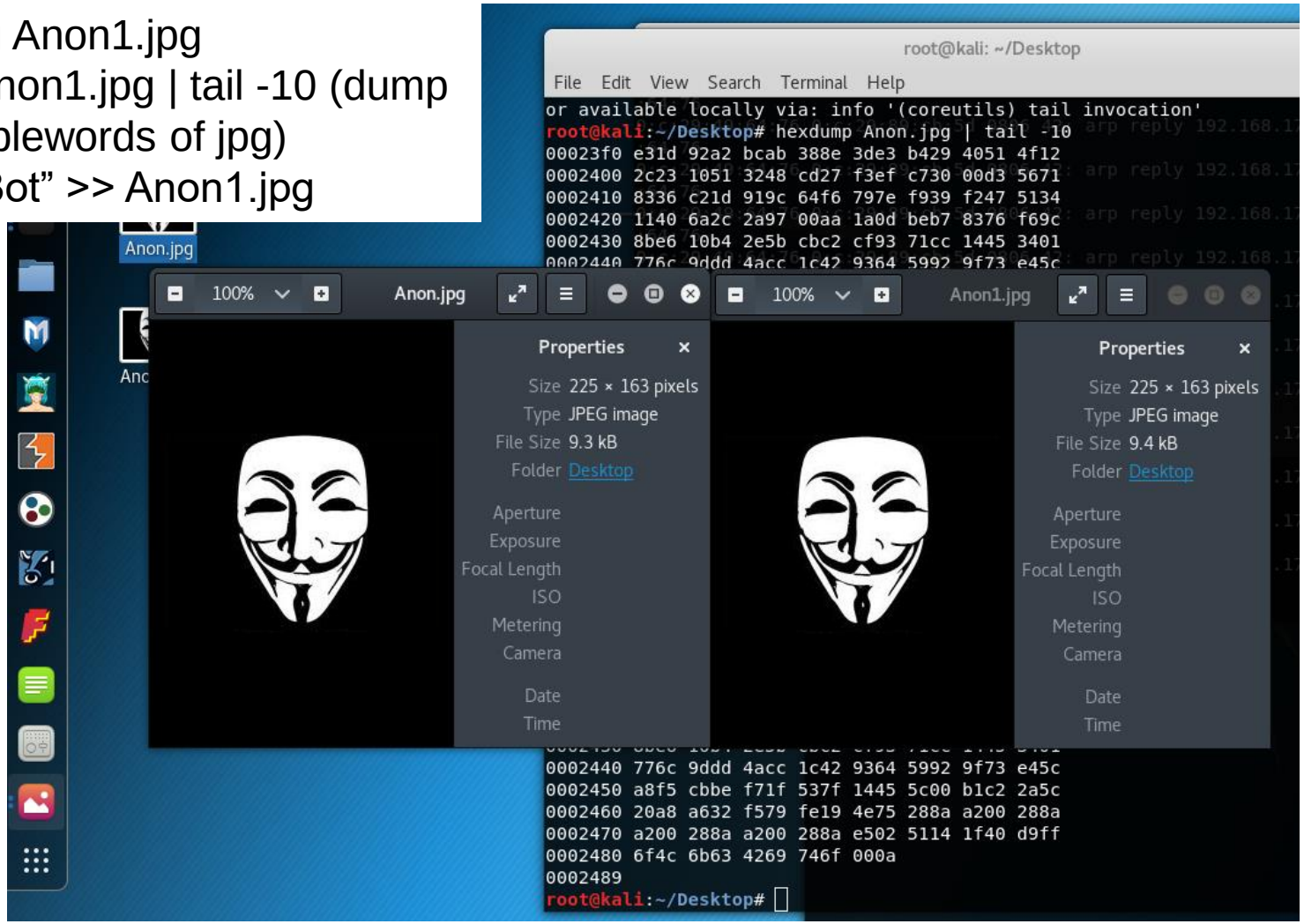
WARNING: PLATINUM hacking group is back

Published June 7, 2019 by Mark Ko

Kaspersky researchers have uncovered a highly sophisticated cyber espionage campaign aimed at stealing information from South Asian diplomatic, government and military entities. The campaign lasted almost six years and had ties to other recent attacks detected in the region. Further investigation into the tools and methods used in the campaign led researchers to the conclusion that the attacker behind it is the PLATINUM group – a cyber espionage actor that they thought had gone. For the activity to remain unseen for such a long time, the group encoded its information using a technique called steganography, which conceals the fact that there is any information there at all.

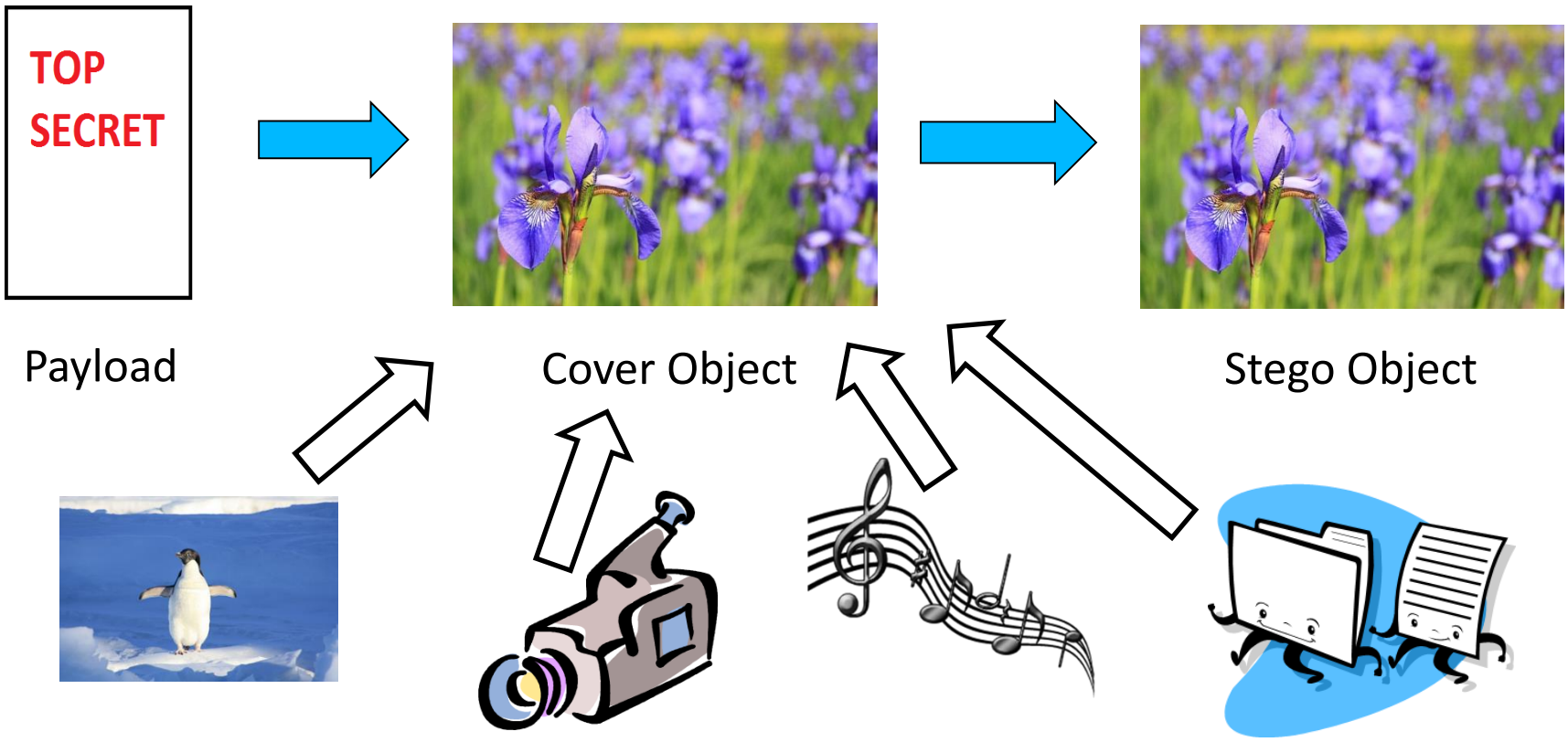
Steganography Application Demo

cp Anon.jpg Anon1.jpg
hexdump Anon1.jpg | tail -10 (dump last 10 doublewords of jpg)
echo “LokiBot” >> Anon1.jpg



- Steganography – can be used to protect privacy of data.
- Do you see any difference in the 2 pics? **Why?**

Steganography - The Process



- Payload – what you want to hide
- Cover Object – what you use to hide the payload eg. image, audio-visual, text files (but suitability varies)
- Stego Object – the resultant object with the hidden payload

Steganography vs Cryptography

DIFFERENCE?

QKRQW UQTZK FXZOM JFOYR HYZWV
BXYSI WMMVW BLEBD MWUWB
TVHMR FLKSD CCEXI YPAHR MPZIO
VBBRV LNHZU POSYE IPWJT UGYOS
LAOXR HKVCH QOSVD TRBPD JEUKS
BBXHT TGVHG FICAC VGUVO QFAQW
BKXZJ SQJFZ PEVJR OJTOE SLBQH QTRAA
HXVYA UHTNB GIBVC LBLXC YBDMQ
RTVPY KFFZX NDDPC CJBHQ FDKXE
EYWPB

Stego attempts to hide the presence of a message but crypto doesn't

Steganography

Security by Obscurity


ORIGINAL


12.5% HIDDEN!!


ORIGINAL


25% HIDDEN!!


ORIGINAL


50% HIDDEN!!

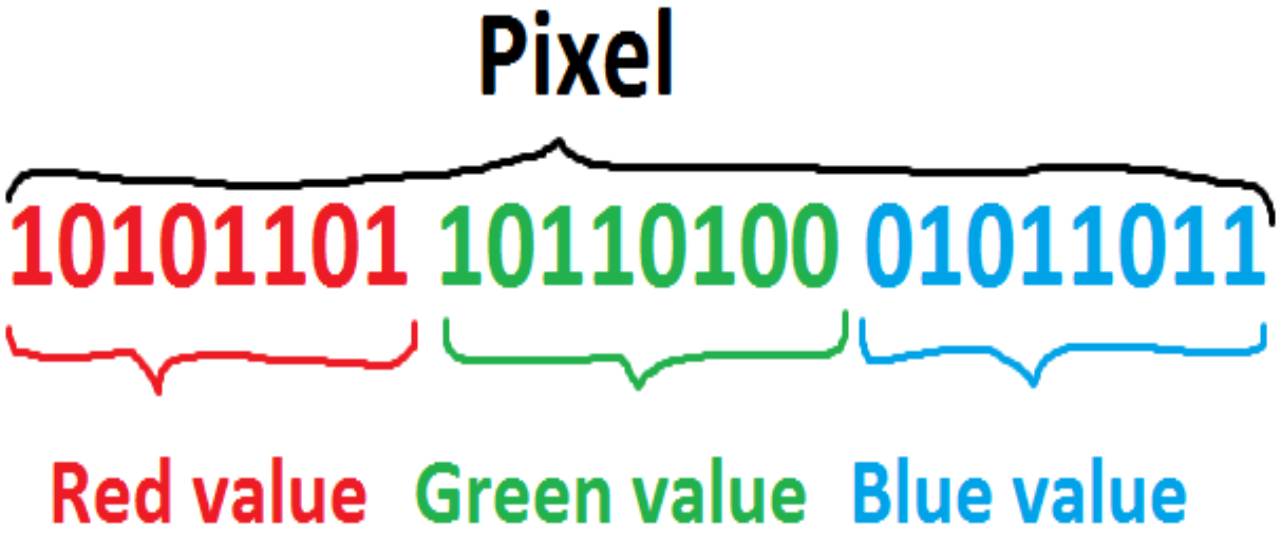
Redundancy in Language

Aoccdrnig to a rscheearch at an Elingsh uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer is at the rghit pclae.

The rset can be a toatl mses and you can sitll raed it wouthit a porbelm.

Tihs is bcuseae we do not raed ervey lteter by itslef but the wrod as a wlohe.

Digital Images



Significant Bits

10100100

Most Significant Bit

Least Significant Bit

165

164

Significant Bits

00100101

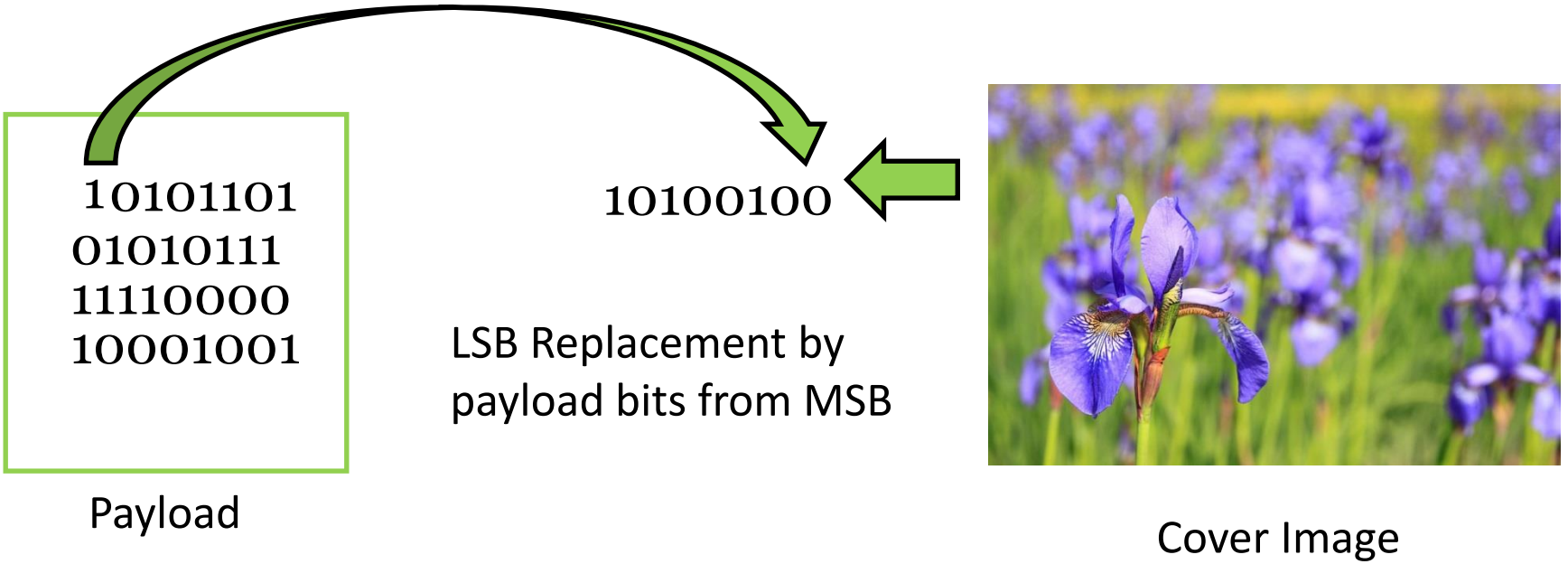
37

If however we change the left most bit, the MSB, from a 1 to a 0 it goes from 165 to 37 – a big difference

Steganography

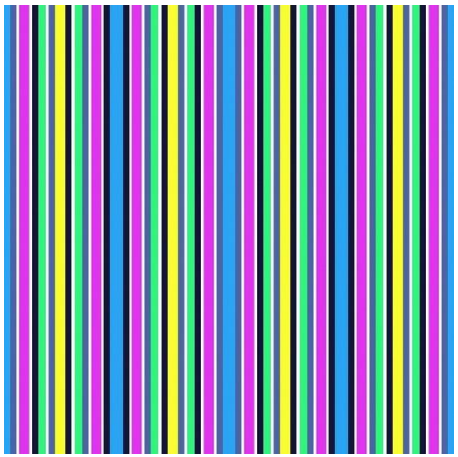
Least Significant Bits (LSB) Algorithm

Least Significant Bits (LSB) Algorithm

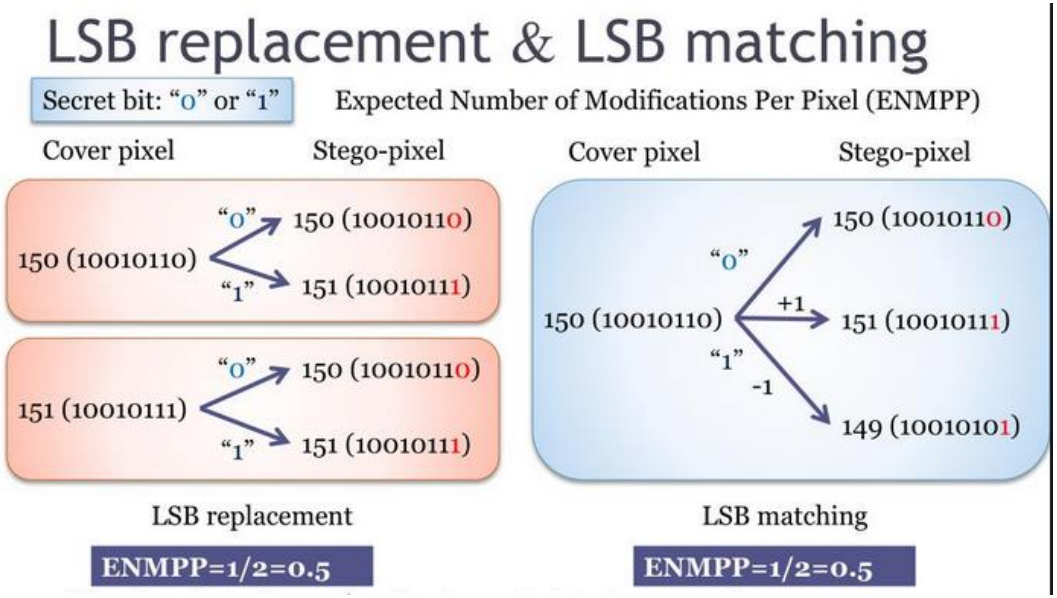


- LSB algorithm hides a payload (eg. messages) inside a cover object (eg. image) by replacing the Least Significant Bits of image pixel bytes with payload bits.
- By modifying only the first right most bit of an image pixel byte we can insert our secret payload and it will also render the changes unnoticeable.
- But, if our payload is too large, the algo will start modifying the next right most bit and so on and eventually, an attacker may notice these changes.

Constraints with LSB Algorithm



If LSB algo is implemented in a linear fashion, may be easier to recover the payload but difficult to detect



- LSB only works on *lossless-compression* images, which means that the files are stored in a compressed format, but the compression does not result in the data being lost or modified, PNG, TIFF, and BMP are examples of lossless-compression image file formats.
- Two main versions – LSB replacement and LSB matching

LSB Replacement Example

Original data	10010101	00001101	11001001
	10010110	00001111	11001011
	10011111	00010000	11001011
Stego data	1001010**0**	0000110**1**	1100100**0**
	1001011**0**	0000111**0**	1100101**1**
	1001111**1**	0001000**1**	11001011

- MSBs of payload substitute (replace) LSBs of cover image pixels.
- Example: payload letter G (01000111).
- On average, only 50% of the cover image pixel LSB bits are changed.

LSB Matching Algorithm - Example

Cover Image Pixel Values:

160	60	53	128	111	43	84	125
10100000	00111100	00110101	10000000	01101111	00101011	01010100	01111101

Payload Character Value:

65
01000001

Stego Image Pixel Values:

160	61	52	128	110	44	84	125
1010000**0**	0011110**1**	0011010**0**	1000000**0**	0110111**0**	00101**10**0	0101010**0**	0111110**1**

- Here, the pixel values of the cover image is given in decimal and binary in the topmost table.
- The next table shows the ASCII values of the payload character A.
- Add +1 or -1 randomly if the LSB of the pixel value does not match the payload bits (from MSB) – the 3rd table is a possible representation of this.

LSB Replacement in Python – 1/3

```
import cv2, numpy as np

def to_bin(data):
    """Convert `data` to binary format as string"""
    if isinstance(data, str):
        return ''.join([ format(ord(i), "08b") for i in data ])
    elif isinstance(data, bytes) or isinstance(data, np.ndarray):
        return [ format(i, "08b") for i in data ]
    elif isinstance(data, int) or isinstance(data, np.uint8):
        return format(data, "08b")
    else:
        raise TypeError("Type not supported.")
```

- The above function converts any type of data into binary.
- We will use this to convert the payload and pixel values to binary in the payload hiding and extracting phase.
- Note that to run, required computer vision (cv2) and array-based math (numpy) libraries must be imported.

LSB Replacement in Python – 2/3

```python
def encode(image_name, secret_data):
    image = cv2.imread(image_name) # read the image
    n_bytes = image.shape[0] * image.shape[1] * 3 // 8 # maximum bytes to encode
    print("[*] Maximum bytes to encode:", n_bytes)
    secret_data += "=====" # add stopping criteria
    if len(secret_data) > n_bytes:
        raise ValueError("[!] Insufficient bytes, need bigger image or less data.")
    print("[*] Encoding data...")

    data_index = 0
    binary_secret_data = to_bin(secret_data) # convert data to binary
    data_len = len(binary_secret_data) # size of data to hide
    for row in image:
        for pixel in row:
            r, g, b = to_bin(pixel) # convert RGB values to binary format
            if data_index < data_len: # modify the least significant bit only if there is still data to store
                pixel[0] = int(r[:-1] + binary_secret_data[data_index], 2) # least significant red pixel bit
                data_index += 1
            if data_index < data_len:
                pixel[1] = int(g[:-1] + binary_secret_data[data_index], 2) # least significant green pixel bit
                data_index += 1
            if data_index < data_len:
                pixel[2] = int(b[:-1] + binary_secret_data[data_index], 2) # least significant blue pixel bit
                data_index += 1
            if data_index >= data_len: # if data is encoded, just break out of the loop
                break
    return image
```

The above function is responsible for hiding the *secret_data* into the image.

LSB Replacement in Python – 3/3

```
def decode(image_name):
    print("[+] Decoding...")
    # read the image
    image = cv2.imread(image_name)
    binary_data = ""
    for row in image:
        for pixel in row:
            r, g, b = to_bin(pixel)
            binary_data += r[-1]
            binary_data += g[-1]
            binary_data += b[-1]
    # split by 8-bits
    all_bytes = [ binary_data[i: i+8] for i in range(0, len(binary_data), 8) ]
    # convert from bits to characters
    decoded_data = ""
    for byte in all_bytes:
        decoded_data += chr(int(byte, 2))
        if decoded_data[-5:] == "=====":
            break
    return decoded_data[:-5]
```

The above function is responsible for extracting *secret_data* from the image.

LSB Matching in Python – 1/2

```
import sys
import cv2
import numpy as np
import random

def extract():
    J=cv2.imread('plane_stego.png')
    f = open('output_payload.txt', 'w+', errors="ignore")

    idx=0
    bitidx=0
    bitval=0
    for i in range(J.shape[0]):
        if (I[i, 0, 0] == '-'):
            break
        for j in range(J.shape[1]):
            for k in range(3):
                if (I[i, j, k] == '-'):
                    break
                if bitidx==8:
                    f.write(chr(bitval))
                    bitidx=0
                    bitval=0
                bitval |= (I[i, j, k]%2)<<bitidx
                bitidx+=1

    f.close()
```

LSB Matching in Python – 2/2

```
bits=[]
f=open('payload.txt', 'r')
blist = [ord(b) for b in f.read()]
for b in blist:
    for i in range(8):
        bits.append((b >> i) & 1)

I = np.asarray(cv2.imread('C:\\Users\\A88252\\Saved Games\\plane.png'))

sign=[1,-1]
idx=0
for i in range(I.shape[0]):
    for j in range(I.shape[1]):
        for k in range(3):
            if idx<len(bits):
                if I[i][j][k]%2 != bits[idx]:
                    s=sign[random.randint(0, 1)]
                    if I[i][j][k]==0: s=1
                    if I[i][j][k]==255: s=-1
                    I[i][j][k]+=s
                idx+=1

cv2.imwrite('plane_stego.png', I)

print("Extracting ... ")
extract()
print("Completed")
```

Assessed Course Work 1.0 - Requirements

- Design and develop an LSB Replacement steganography program
- This program can be written in a language of your choice (eg. Python, C, Java)
- Program Requirements:
  - Support steganography of as many of the following payloads as possible:
    - ✓ Image files (eg. JPG, BMP, PNG)
    - ✓ Document files (eg. Word, .txt, .xls)
    - ✓ Audio-visual files (eg. mp3, mp4, wav)
  - Number of LSBs to use be selectable from bits 0 – 5 (eg. bits 0,1, and 2); selection to be integrated as part of GUI
  - Drag and drop or explorer-type functionality as part of GUI to select both cover object and payload
  - Program to implement limit check and display error message should selected payload be too large for selected cover object
  - GUI to display BOTH cover object and stego objects
- Work in your own team and be responsible for each member's role/task assignments

Assessed Course Work 1.0 - Rubrics

- Support for range of payload types:
  - Supports different types of image payloads?
  - Supports different types of document payloads?
  - Supports different types of audio-visual payloads?
- Support the use of up to 6 LSBs of the cover object for payload hiding and retrieval:
  - Able to use 1 – 3 LSBs of the cover object?
  - Able to use 1 – 6 LSBs of the cover object?
- GUI design:
  - Facilitates selection of payload and cover object?
  - Facilitates display of both cover and stego objects?
  - Facilitates selection of LSBs of the cover object to use?
- Understanding and contribution: - individual
- Bonus: innovations eg. can use text, audio files as cover objects

Assessed Course Work 1.0 – Evaluation

- Develop GUI-Based LSB Replacement steganography program (you may modify from open source):
  - Attempt to meet as many of the requirements on previous slide
  - Test program thoroughly before the demo
- You may use any of the programming languages you have learnt previously – eg. C, Python, Java.
- You may not use online steganography portals.
- Successful demo with clear explanation and no errors.
- Demo dates: Week 6 (Tues 12/10 and Wed 13/10)
- All team members must be present. Members who are absent will be separately assessed.
- No report is needed.

Steganography

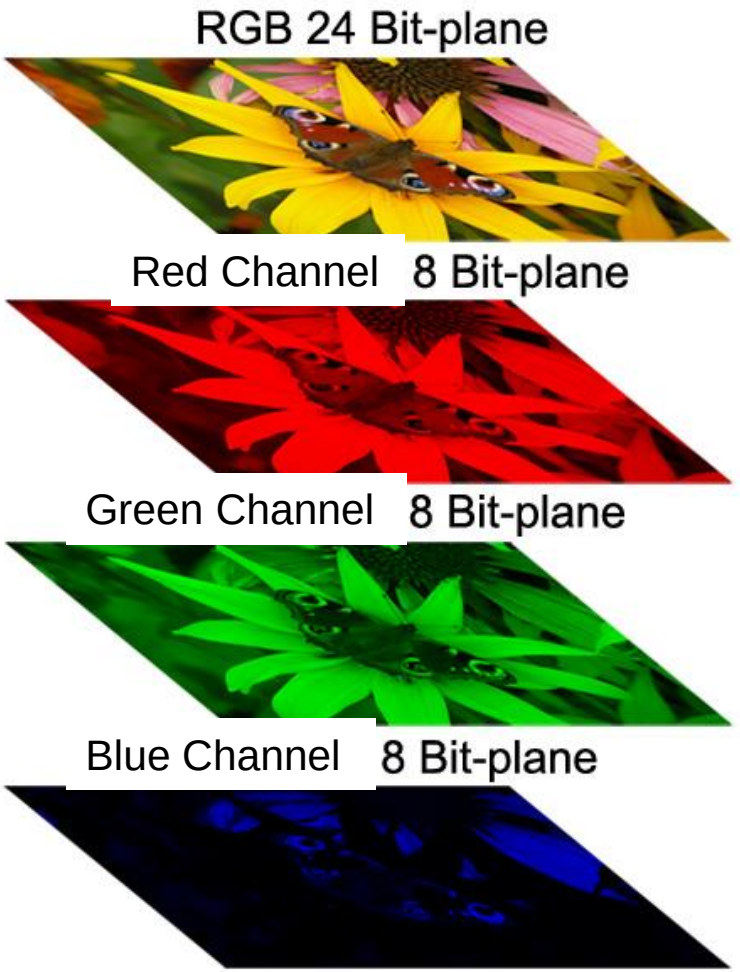
Bit Planes

Bit Planes

01110101011101010111010101

red	green	blue	color
50	50	50	dark gray
120	120	120	medium gray
200	200	200	light gray
250	200	200	not gray, reddish
0	0	0	black (a sort of gray)
255	255	255	white (ditto)

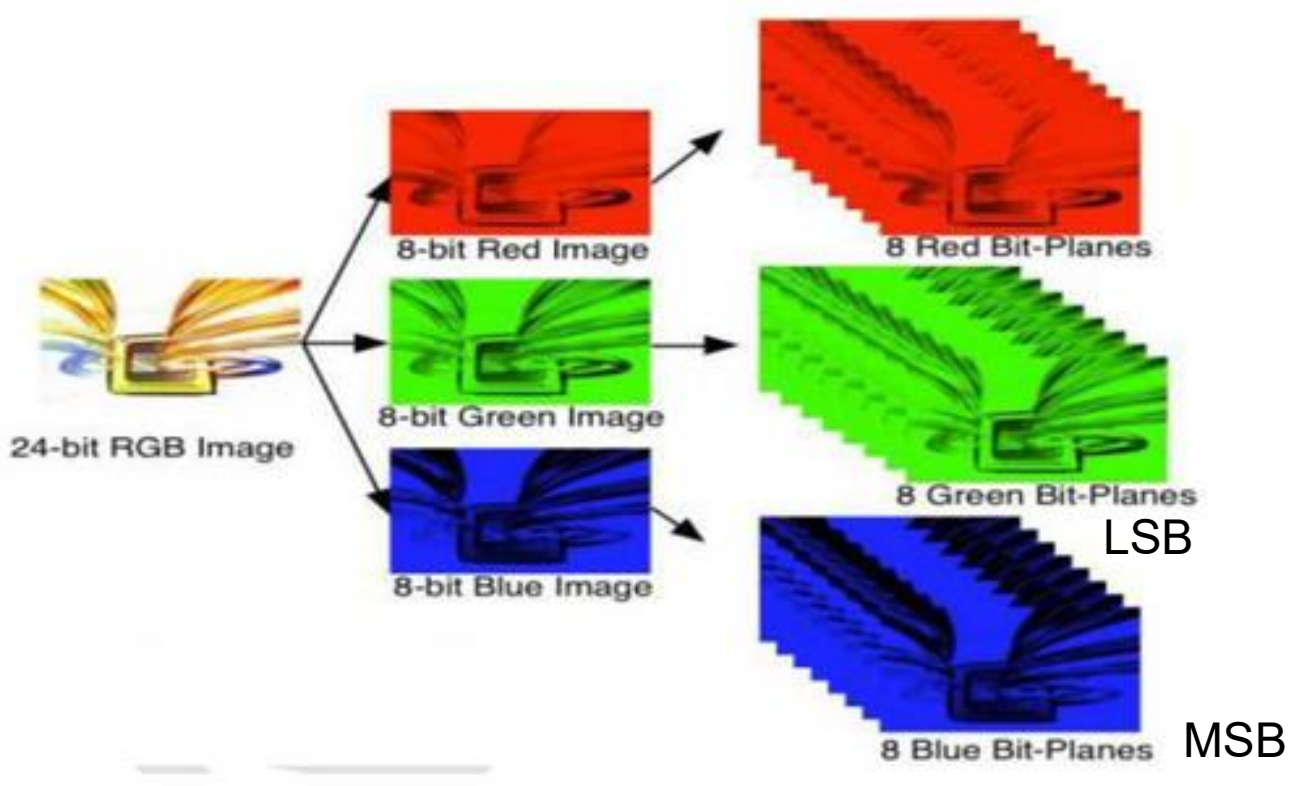
Grayscale bit plane
stores intensity info



nth Bit Planes

01110101011101010111010 1

e.g. the 0th blue bit plane is all the LSBs of the blue bytes

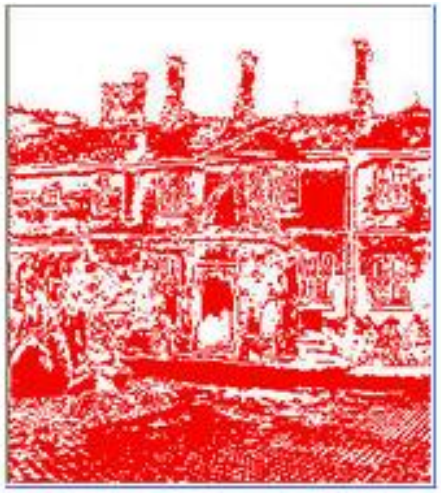


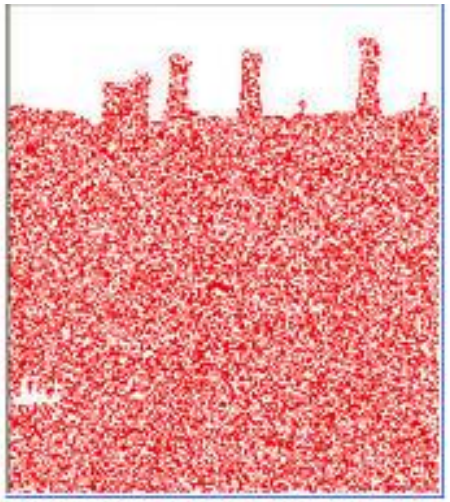
Information Bearing Bit Planes

01110101011101010111010 1

Take all red bytes in every pixel: the red channel 8-bit plane

(A color image **P**)



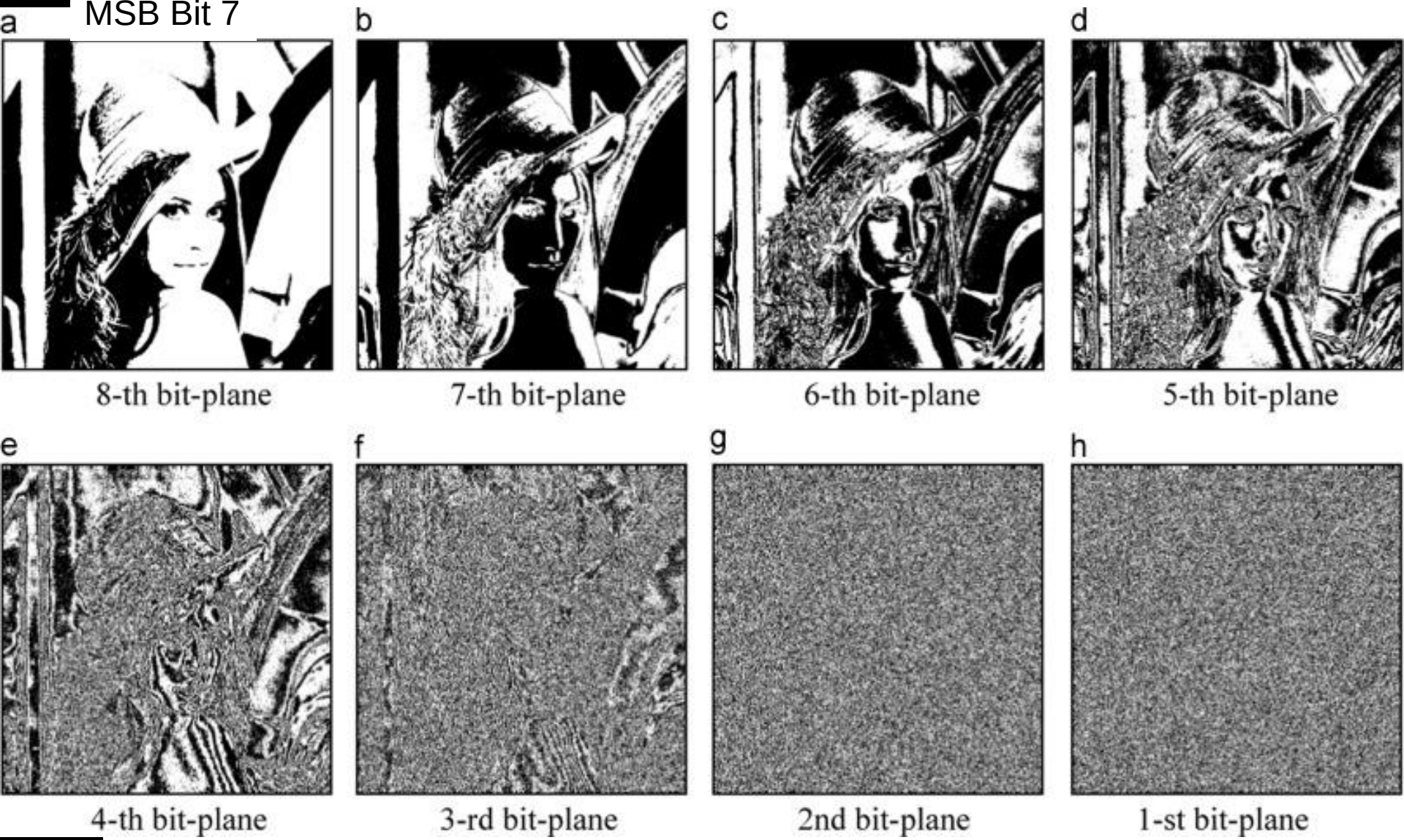


MSB ← → LSB

Red Channel 8-bit plane divided into 8 1-bit planes

Grayscale Bit Planes

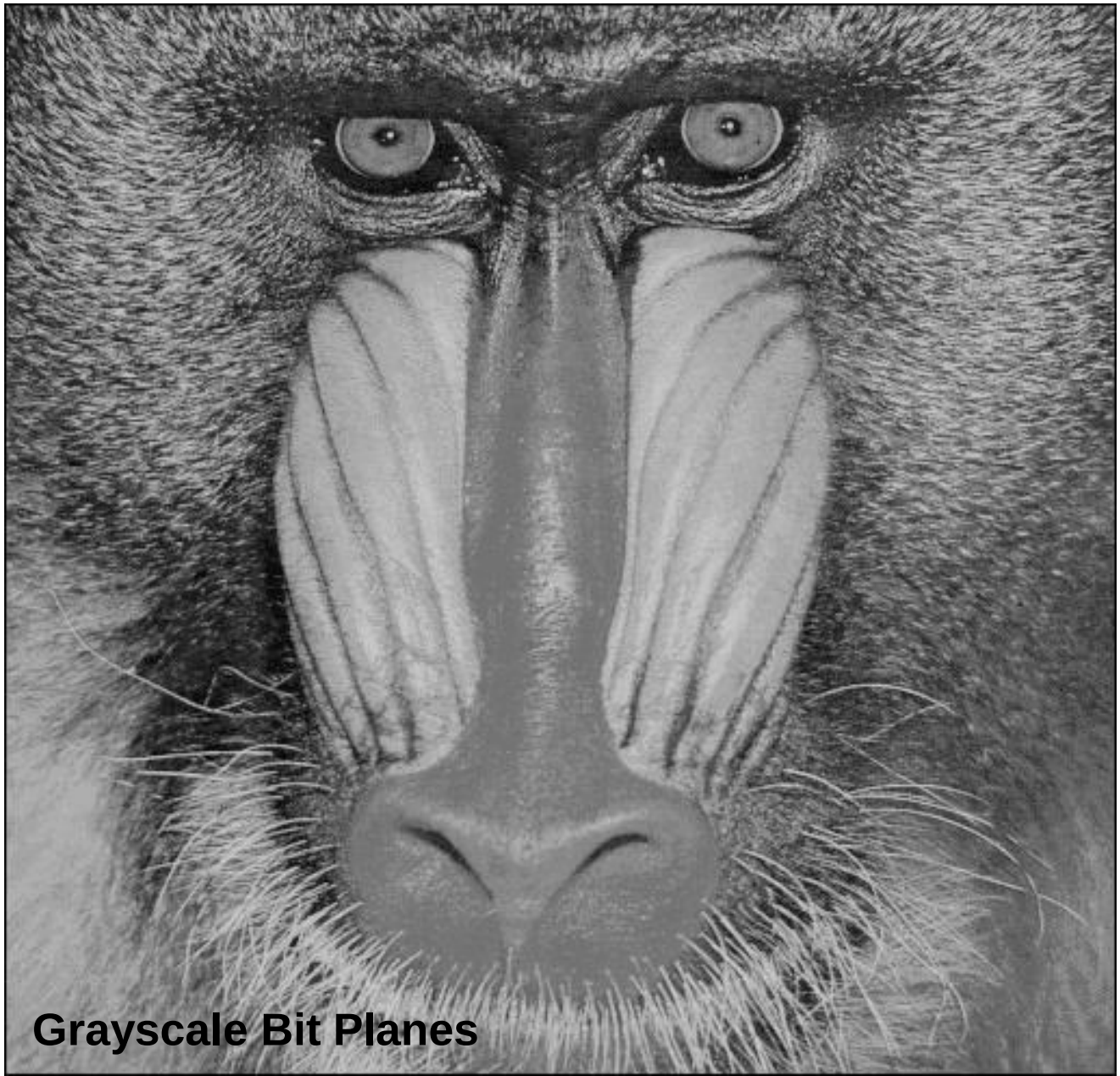
MSB Bit 7

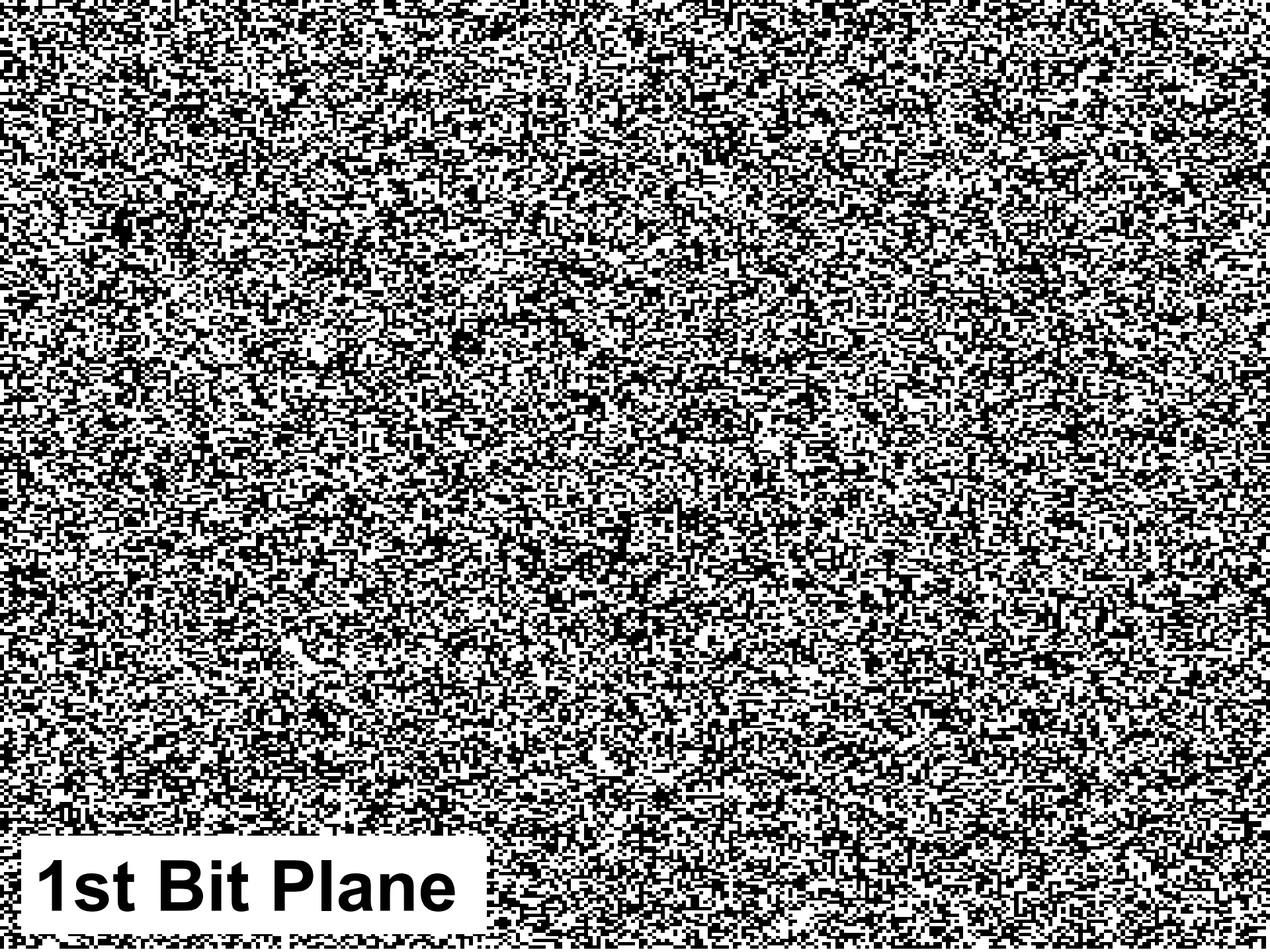


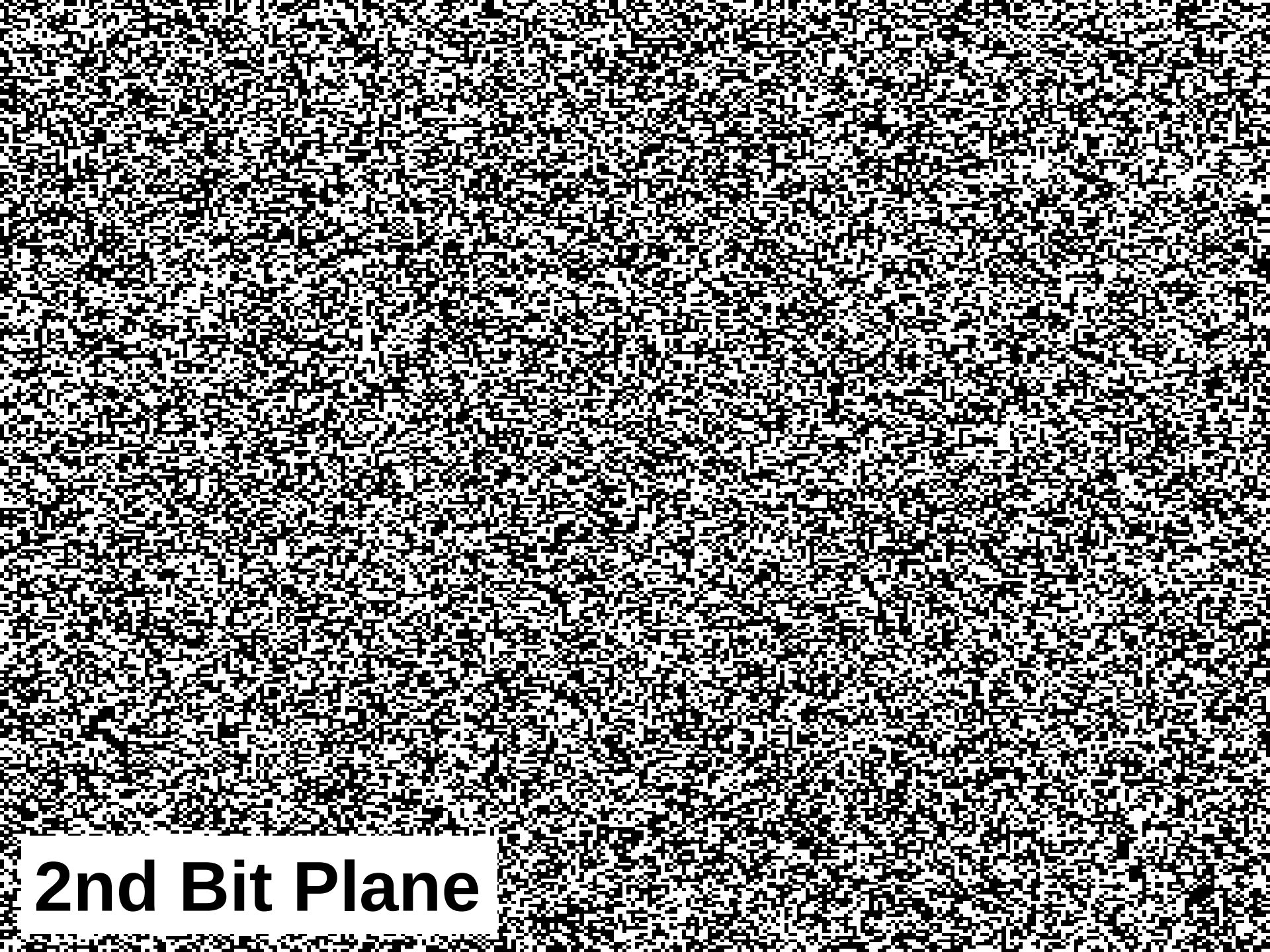
LSB Bit 0

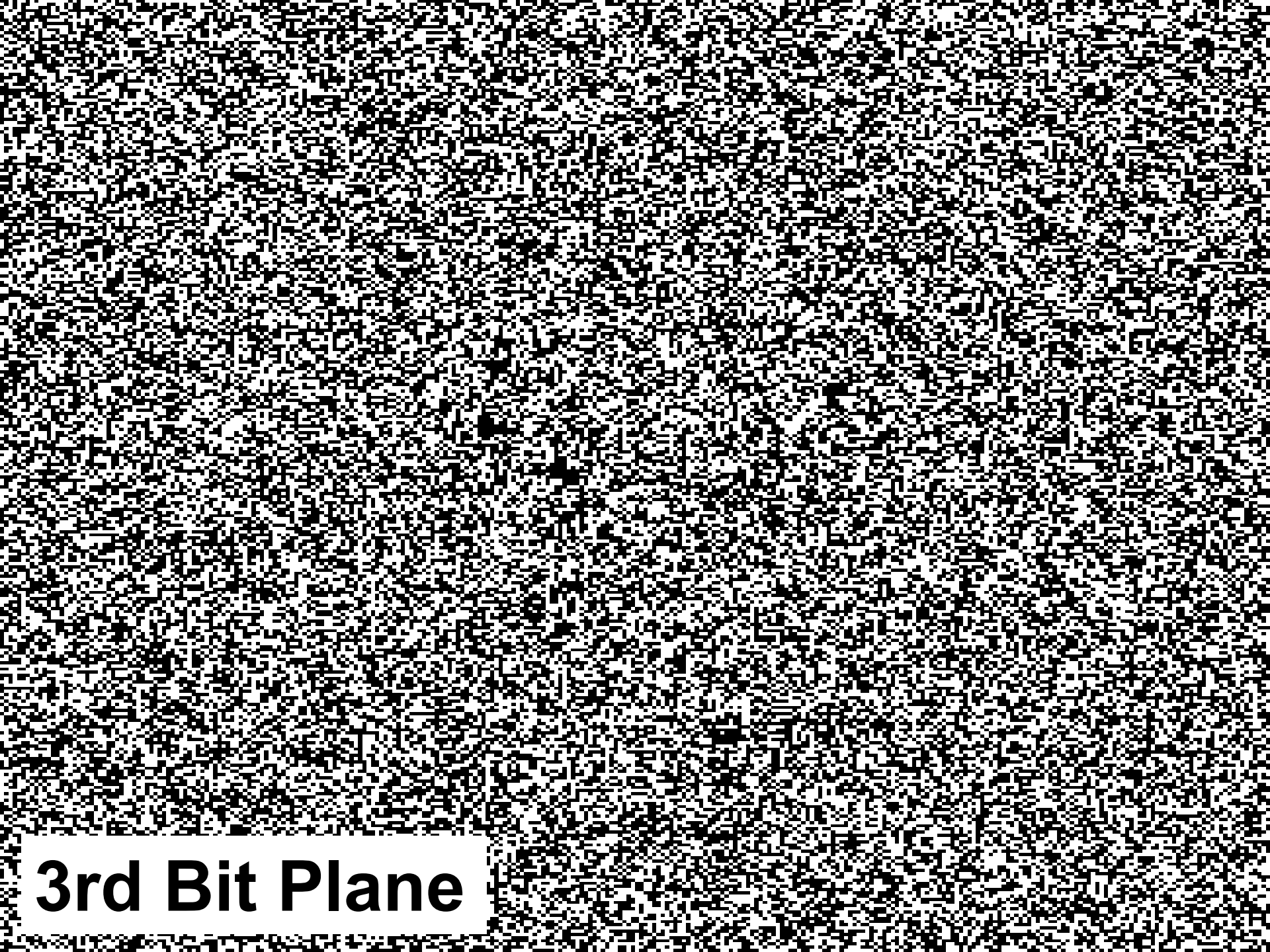
Another Example - Original Image

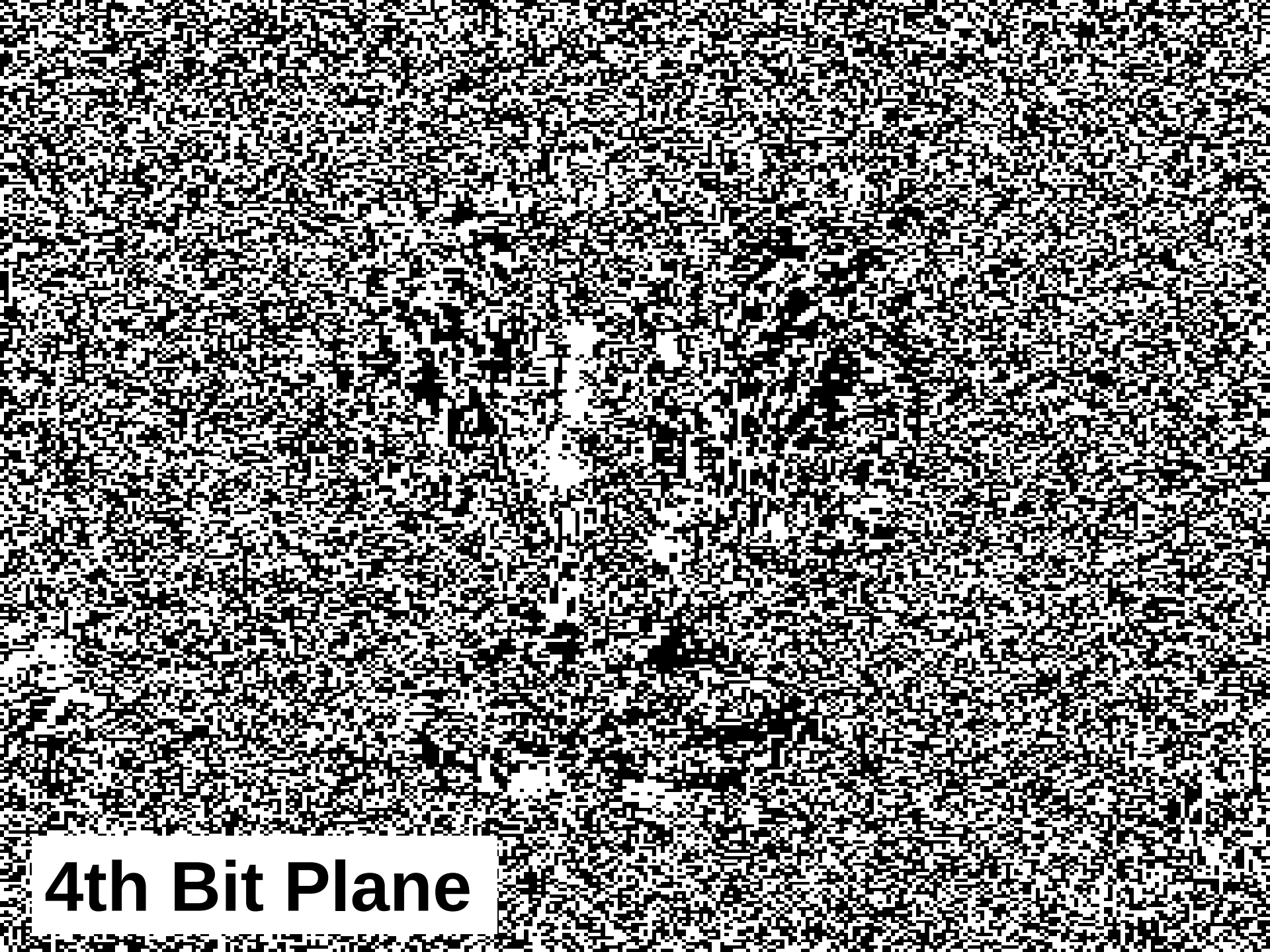
Grayscale Image

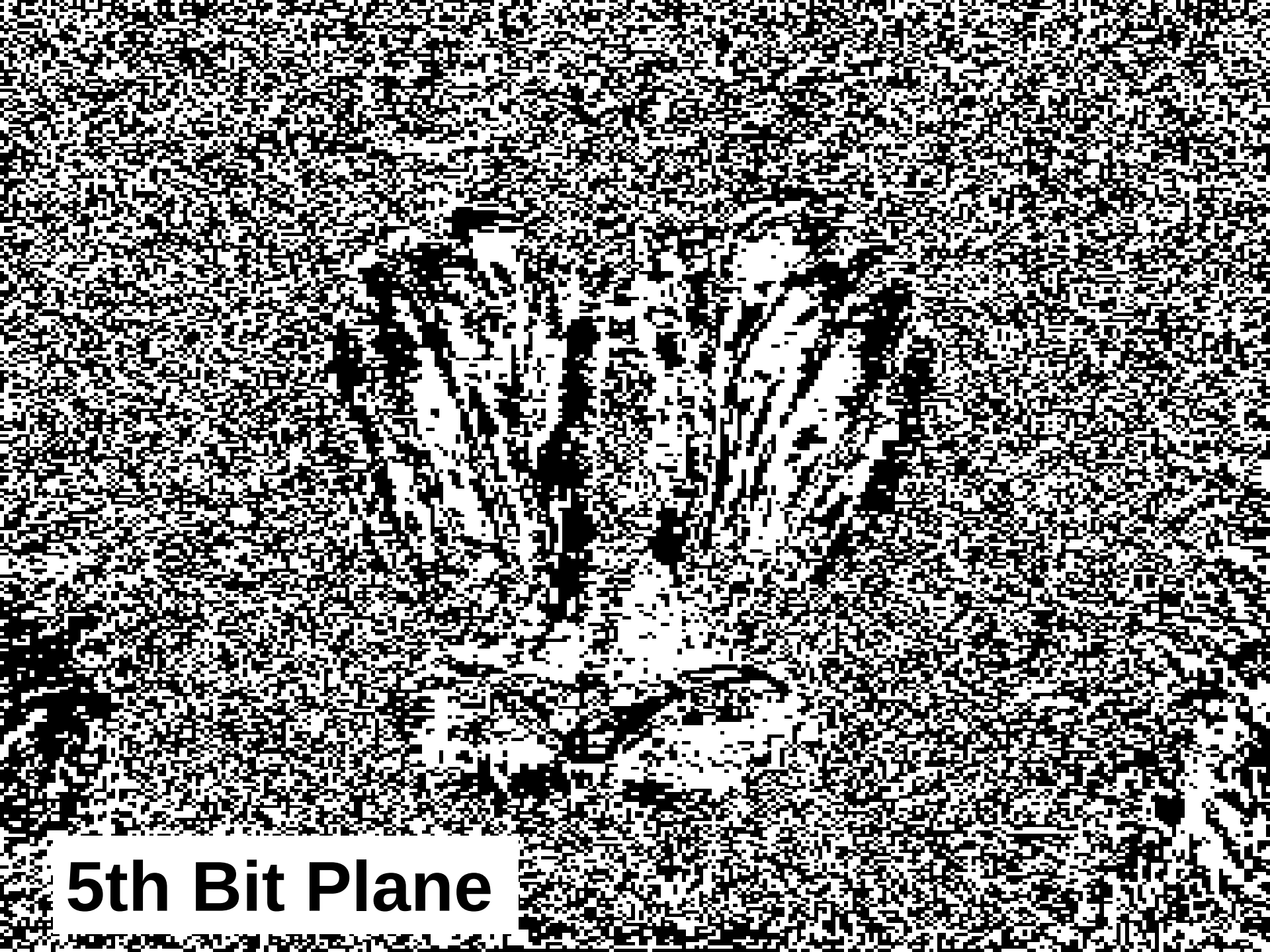


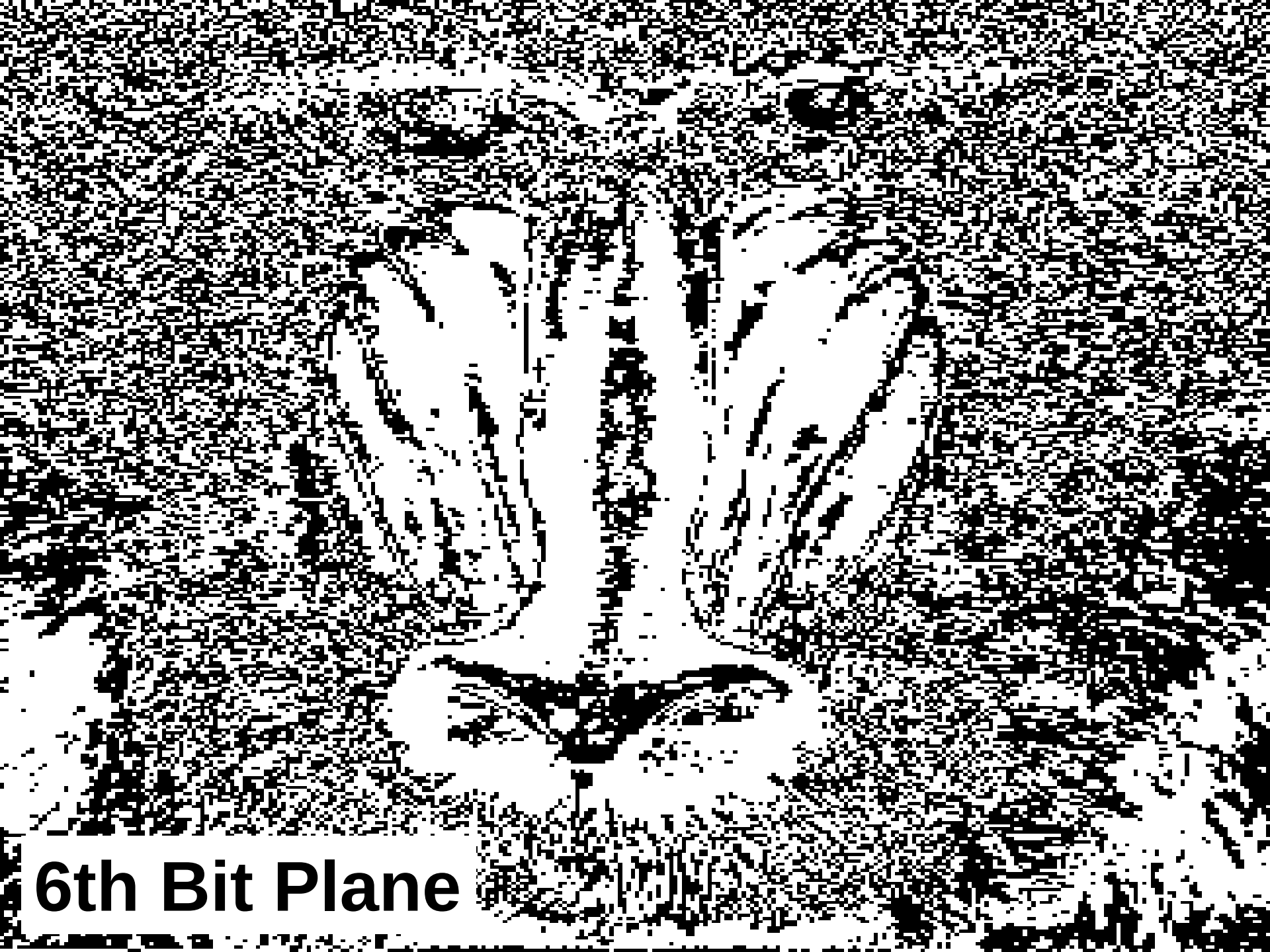

1st Bit Plane

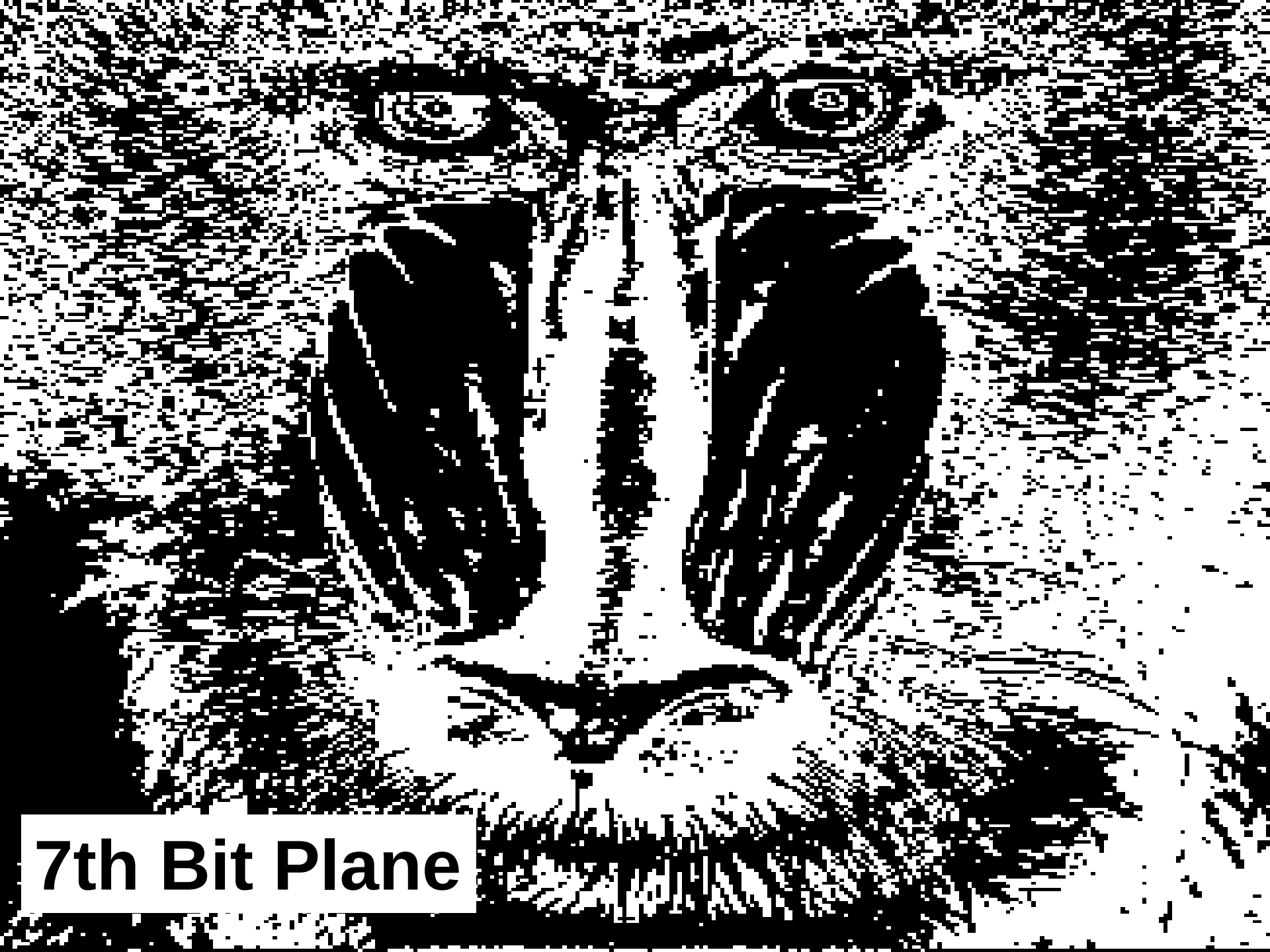

2nd Bit Plane


3rd Bit Plane


4th Bit Plane


5th Bit Plane


6th Bit Plane

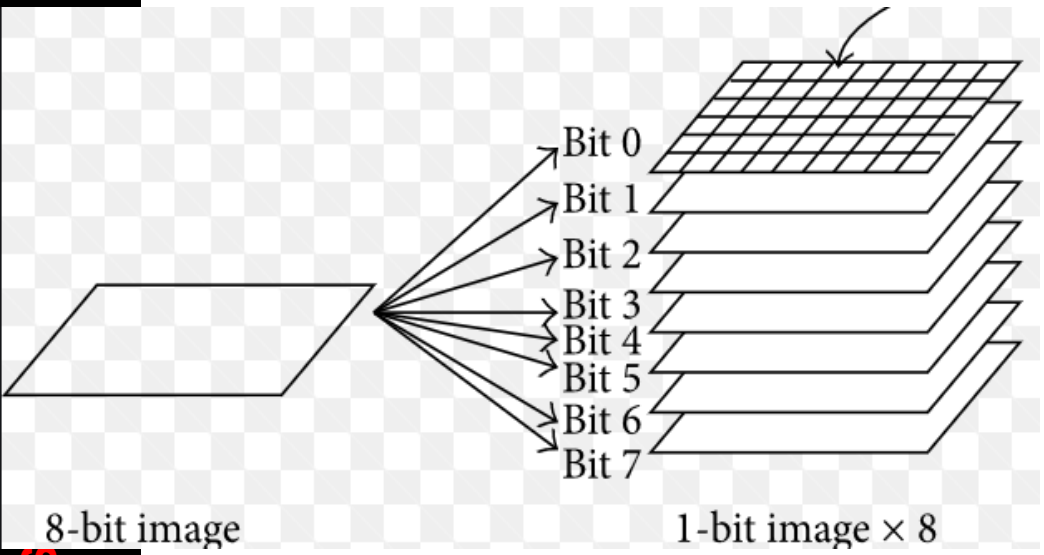

7th Bit Plane


8th Bit Plane

Steganography

Bit Plane Complexity Segmentation Algorithm

Bit Plane Complexity Segmentation (BPCS)Algorithm



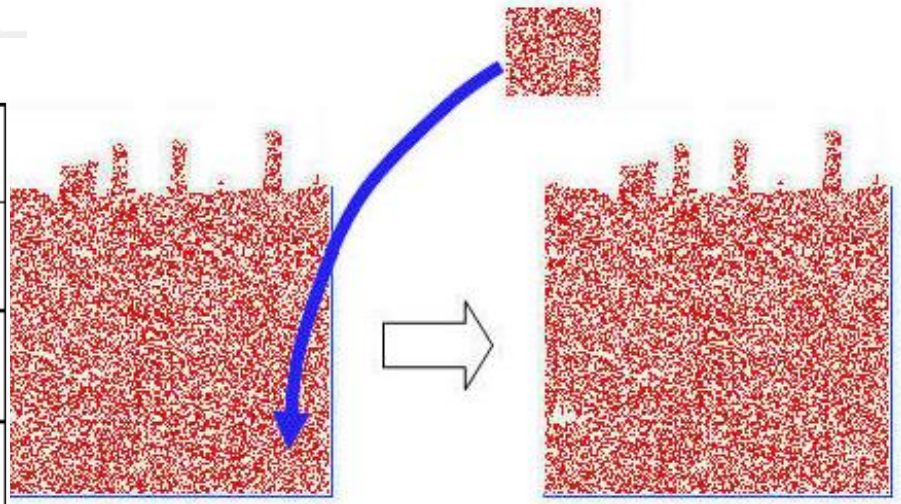
- Cover image is split into 8 bit planes
- Bit plane is divided into complex and not complex regions
- Payload is hidden in complex regions without image deterioration

W	W	W	W
W	W	W	W
W	W	W	W
W	W	W	W

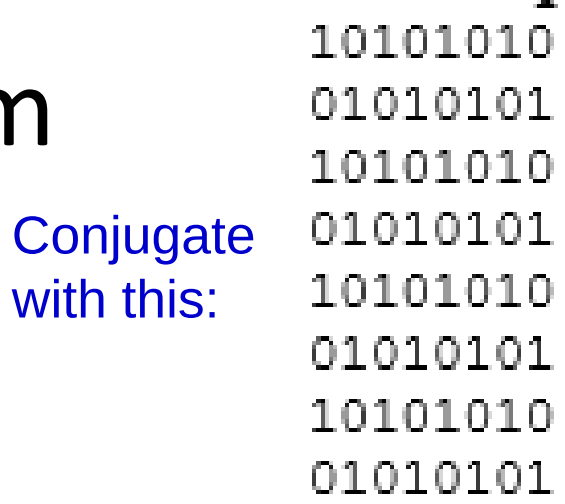
Figure : (a) all white pixels in ima

W	B	W	B
B	W	B	W
W	B	W	B
B	W	B	W

(b) black-white checker board

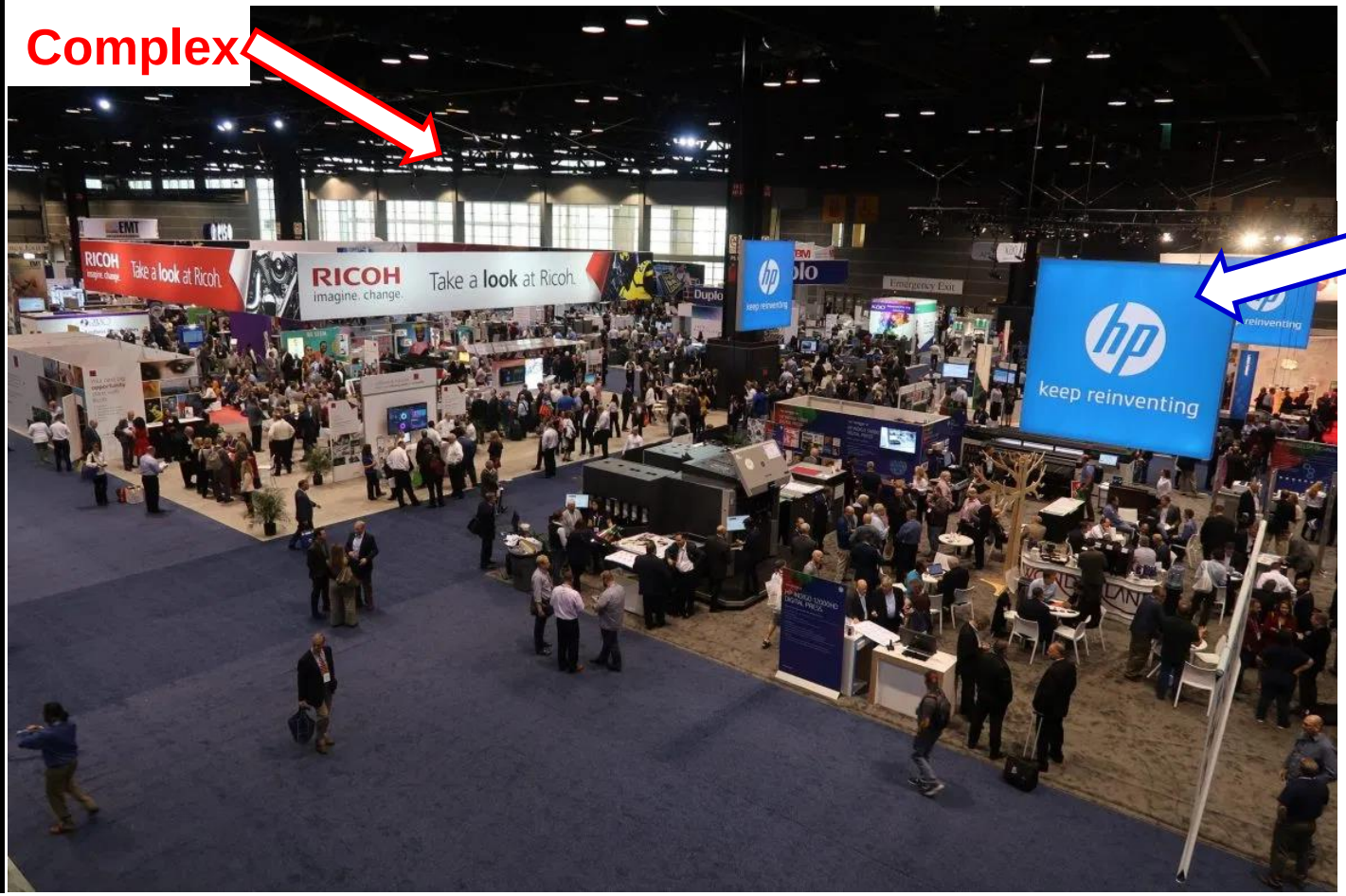


BPCS Algorithm



- While there is more to hide
  - Get the next bit plane
    - While there is more space in the current bit plane
      - Get the next complex segment
      - Get the next block of payload
      - If the payload information is complex, then hide it in the current segment
      - Else conjugate by performing an exclusive or operation then hide it in the current segment.

Complex Segment Example



Conjugate with this:

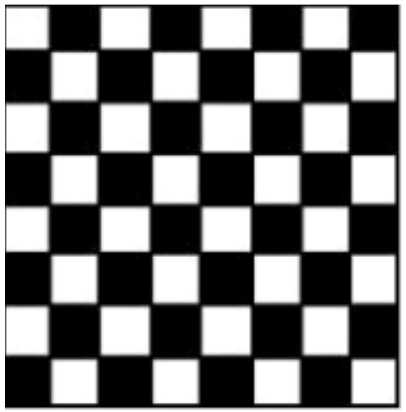
```
10101010
01010101
10101010
01010101
10101010
01010101
10101010
01010101
```

- Use segments of bit planes which are ‘noisy’ (complex) to hide payload
- If payload not noisy enough, perform ‘conjugation’ operation to make it complex
- Works because human eye cannot notice difference in ‘rapidly changing bit patterns’

What is considered Complex?

- Complexity essentially measures how often neighbouring bits in a segment change value.
- Assume there are 8 bytes in a segment (block):
  - Count the number of times the bits change value.
  - Maximum value is 7, e.g. 10101010
- Do the same for each of the 8 bytes
- The maximum number of changes is 2 x 8 x 7 = 112 (horizontal and vertical).
- The complexity of a segment c = actual changes / 112 .
- Define a threshold value T, which is a parameter of the algorithm.
- If c > T then the segment is complex.

No standard complexity threshold value,
Typically, T is around 0.3

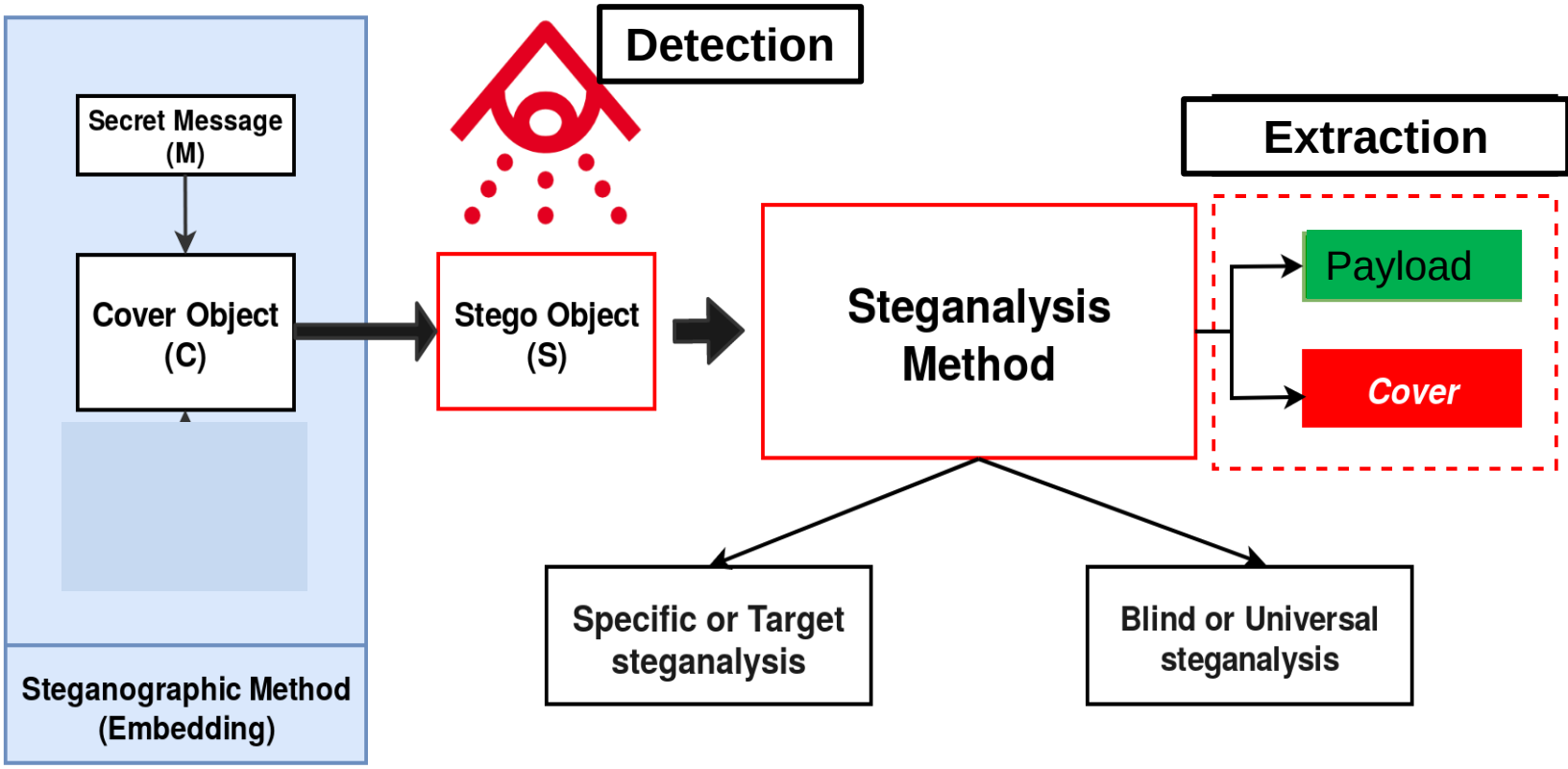


Steganography

Steganalysis

Introduction to Steganalysis

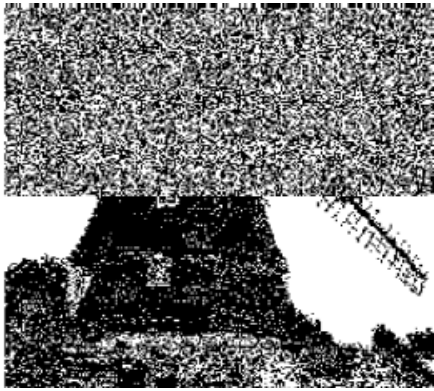
Alice



- Steganalysis is an attack on steganography.
- Primary objective is hidden payload detection.
- Secondary objective is to extract the payload.
- Hiding info in digital media changes some media content.
- May introduce visual degradation or unusual characteristic(s).

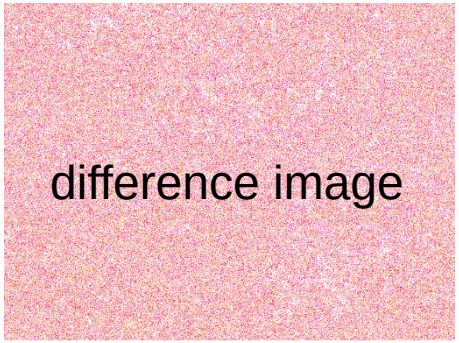
Visual Steganalysis

Black-white filtering: even values are black and odd values white. Without filter, LSB value changes are indistinguishable.



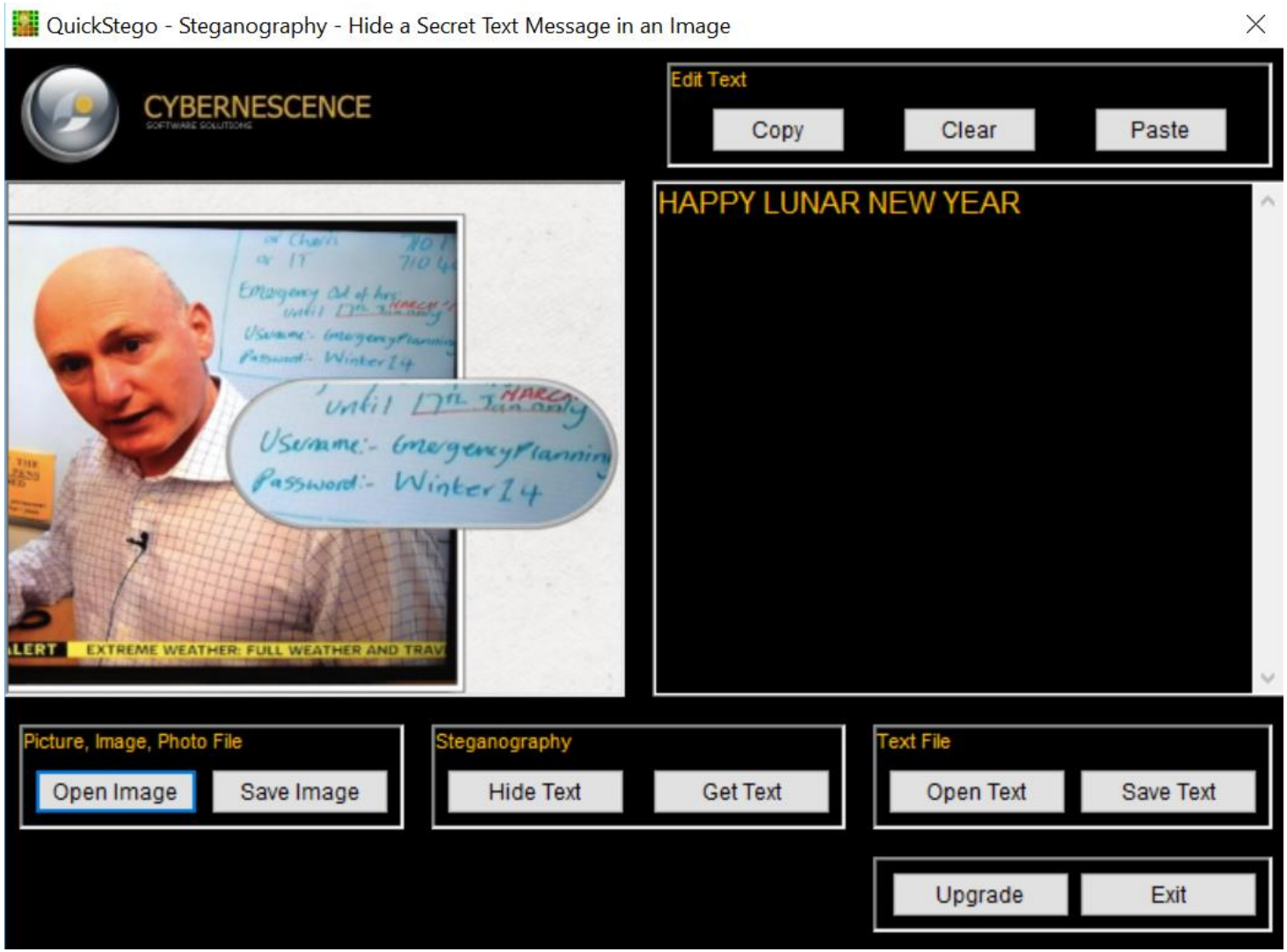
ORIGINAL STEGO-IMAGE (suspected)

https://www.rapidtables.com/web/color/RGB_Color.html



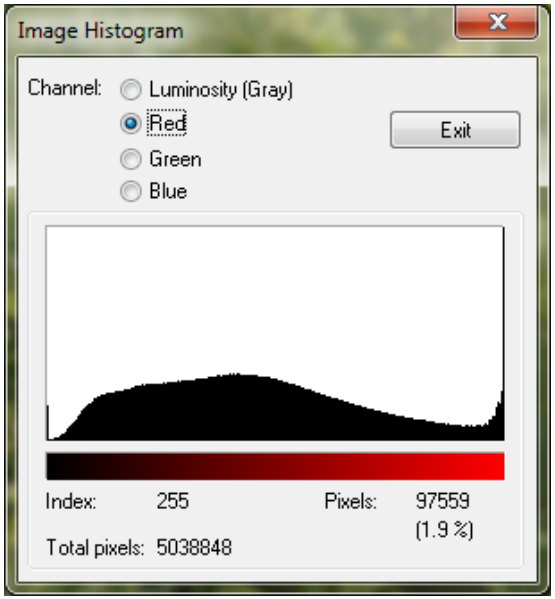
https://online-image-comparison.com/

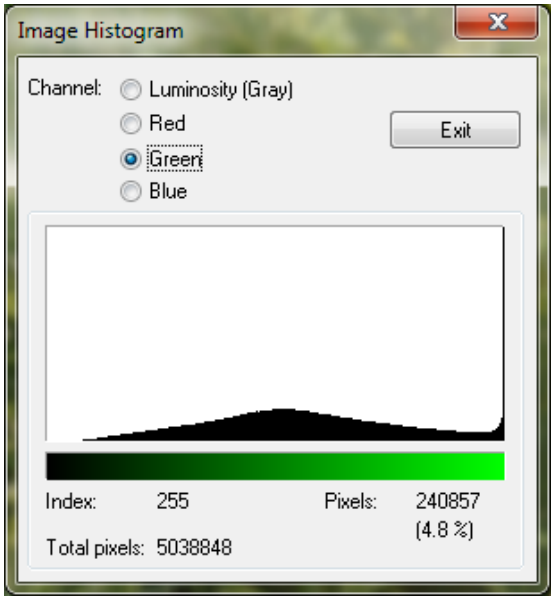
Tool-based Steganalysis

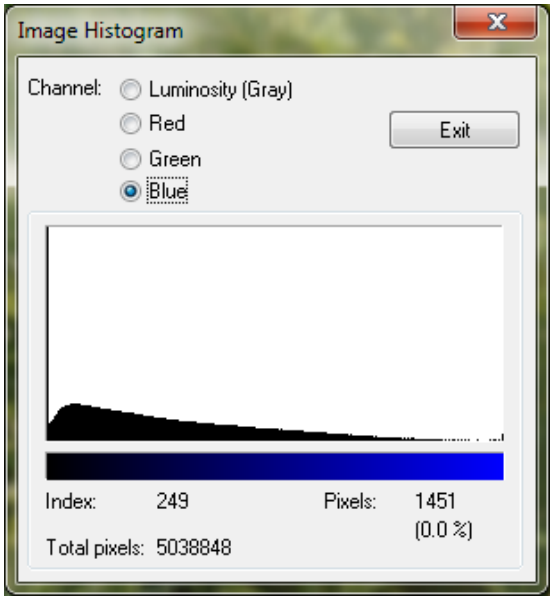


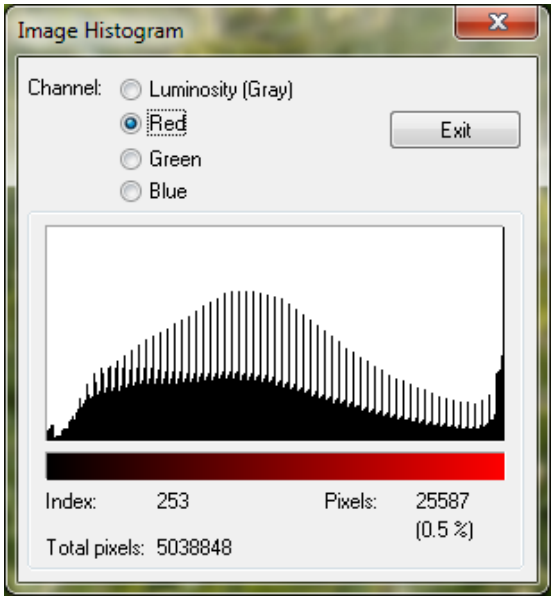
EXAMPLE TOOL: QUICK STEGO

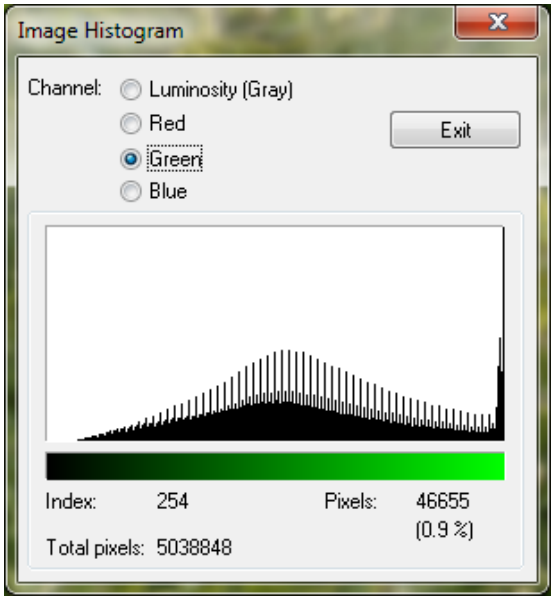
Statistical Steganalysis – 1/4

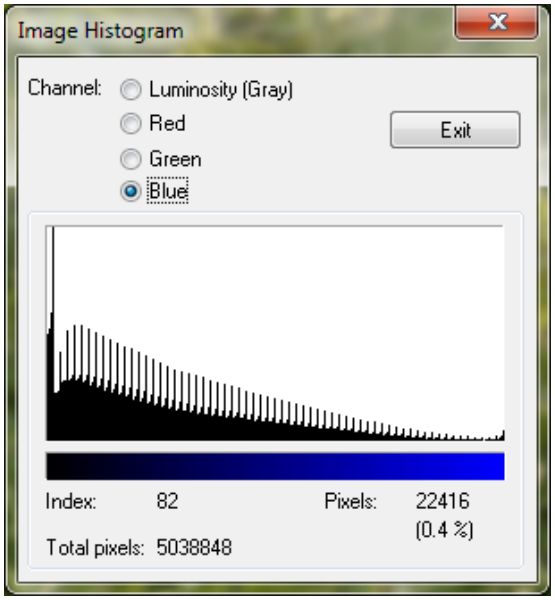




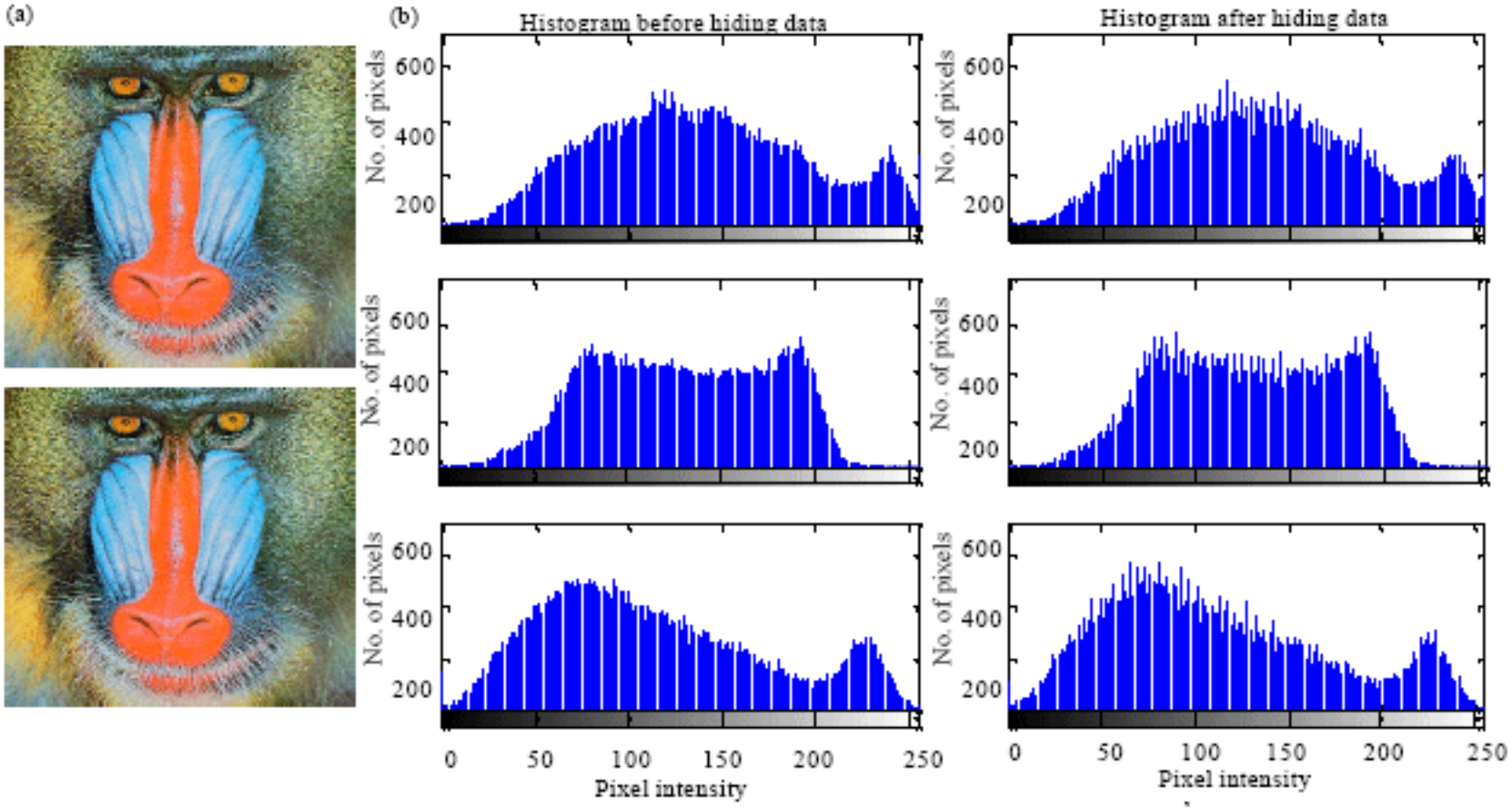






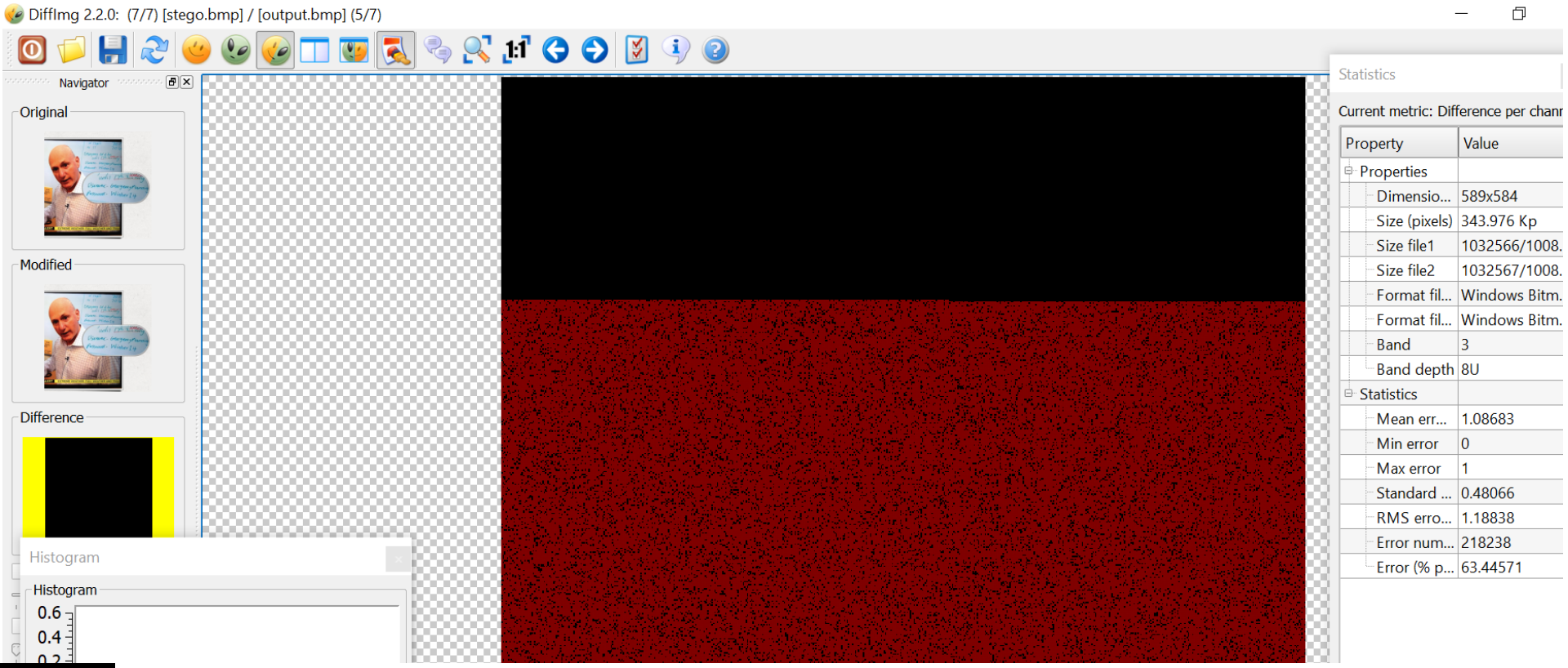


Statistical Steganalysis – 2/4



(a) Baboon cover image and its corresponding stego image and (b) Subsequent histograms for cover and stego images

Statistical Steganalysis – 3/4

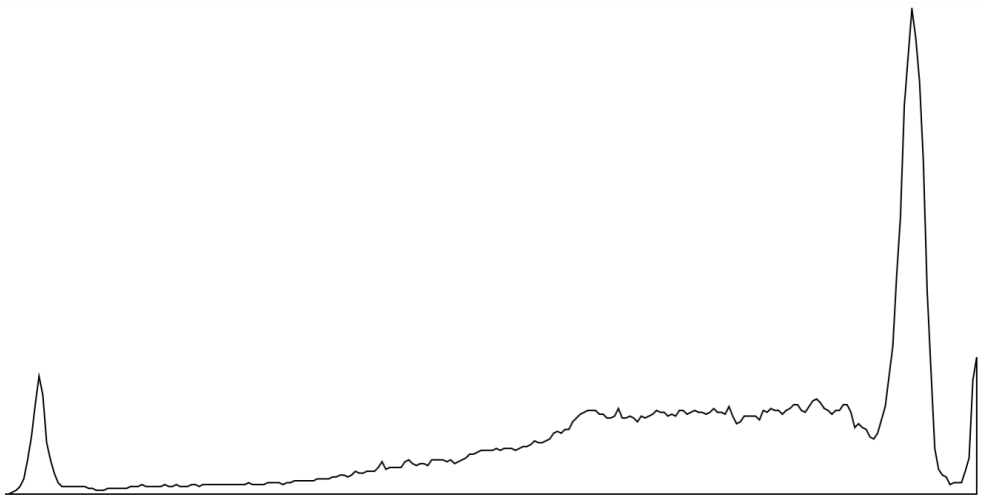


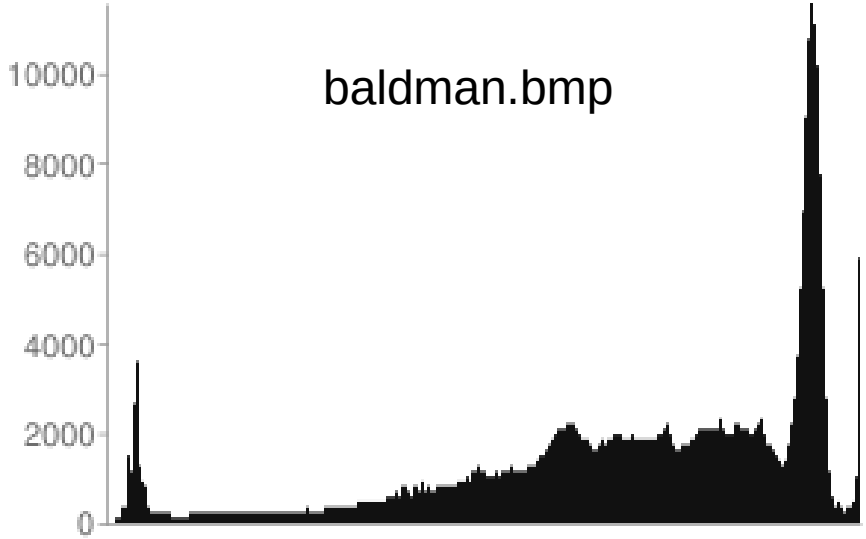
- DiffImg – shows difference image; stats values are displayed numerically
- Download from the following link:

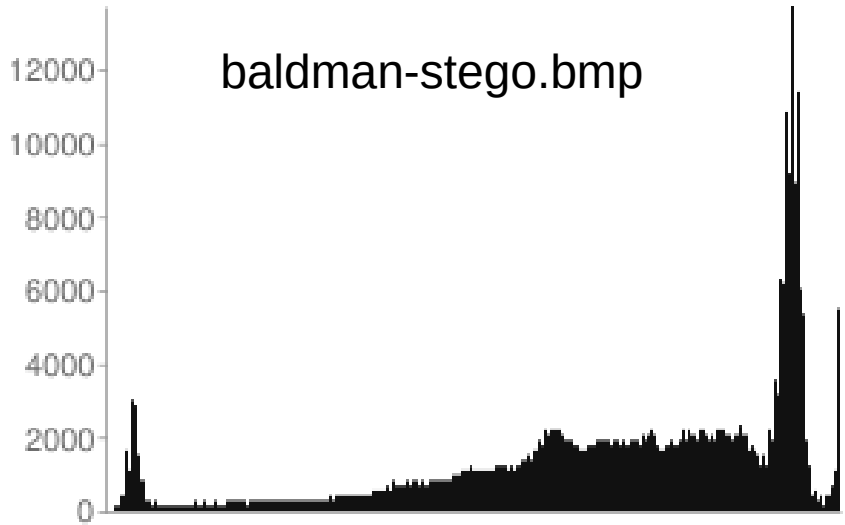
https://www.softpedia.com/get/Multimedia/Graphic/Graphic-Viewers/DiffImg.shtml

Statistical Steganalysis – 4/4

https://www.sisik.eu/histo







https://www.dcode.fr/image-histogram

Summary

- Overview of "Hiding in Plain Sight" (non-digital examples, steganography in malware, espionage)
- The Steganography Process – payload, cover object (multimedia), stego object.
- Steganography vs Cryptography.
- Principle of Security by Obscurity – make use of redundancies.
- Exploiting image bit redundancies – LSB Replacement, LSB Matching algorithms.
- Using complex segments in image bit planes and conjugating non-complex payloads – Bit Plane Complexity algorithm
- Steganalysis (attacking stego) – Visual, Tool-Based, Statistical
- **Next Lecture – Ethical Hacking**