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

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'

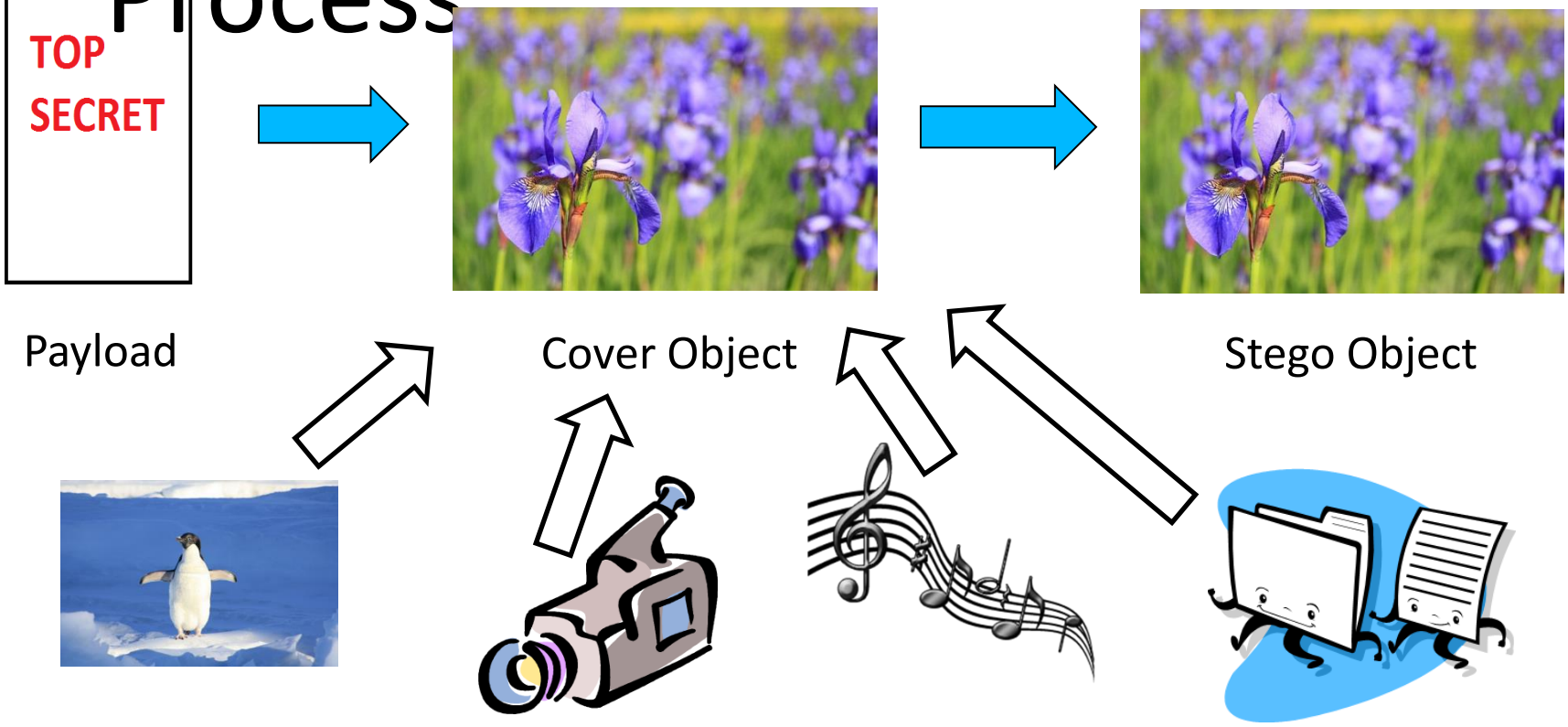
Steganography vs Cryptography

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 - 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

Security by Obscurity


ORIGINAL


12.5% HIDDEN!!


ORIGINAL


25% HIDDEN!!

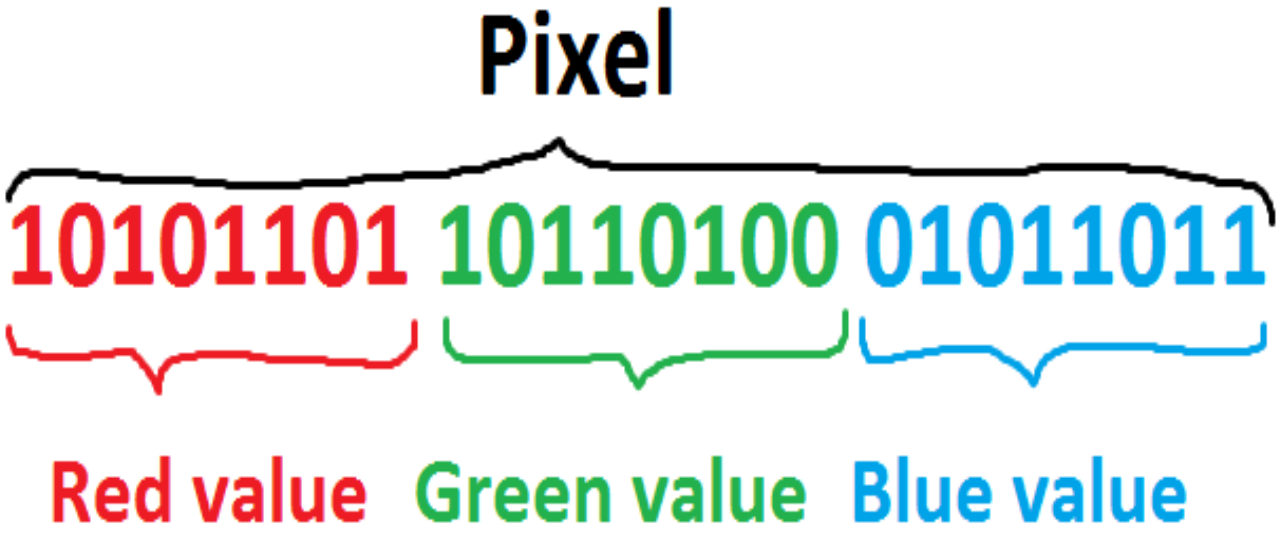

ORIGINAL


50% HIDDEN!!

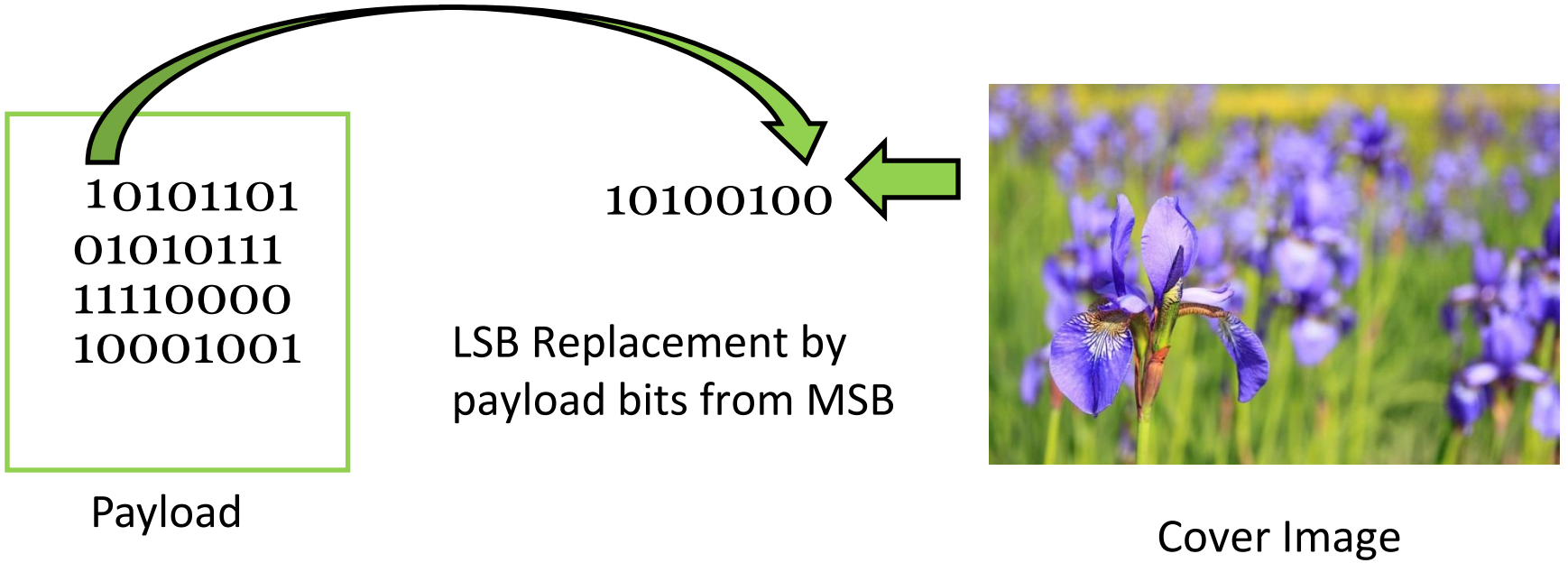
Steganography

Least Significant Bits (LSB) Algorithm

Digital Images

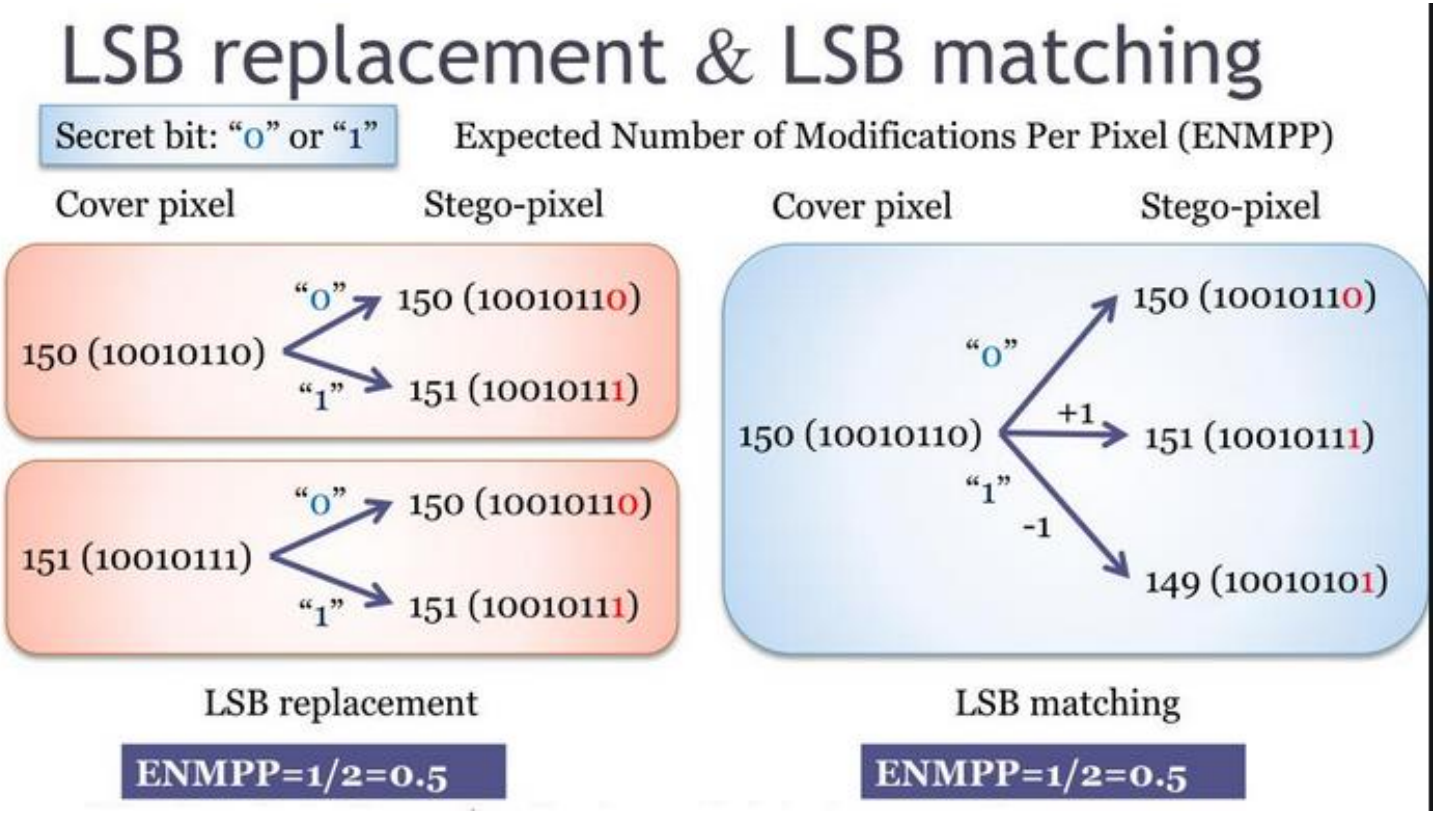


Least Significant Bits (LSB) Algorithm



- 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



- 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	10010100	00001101	11001000
	10010110	00001110	11001011
	10011111	00010001	11001011

- MSBs of payload substitute (replace) LSBs of cover image pixels.
- Example: payload letter G (01000111), start from MSB.
- 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**100**	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


```
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")
```

LSB Matching in Python – 2/2


```
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()
```

Assessed Course Work 1.0 - Requirements

Steganography

Bit Planes

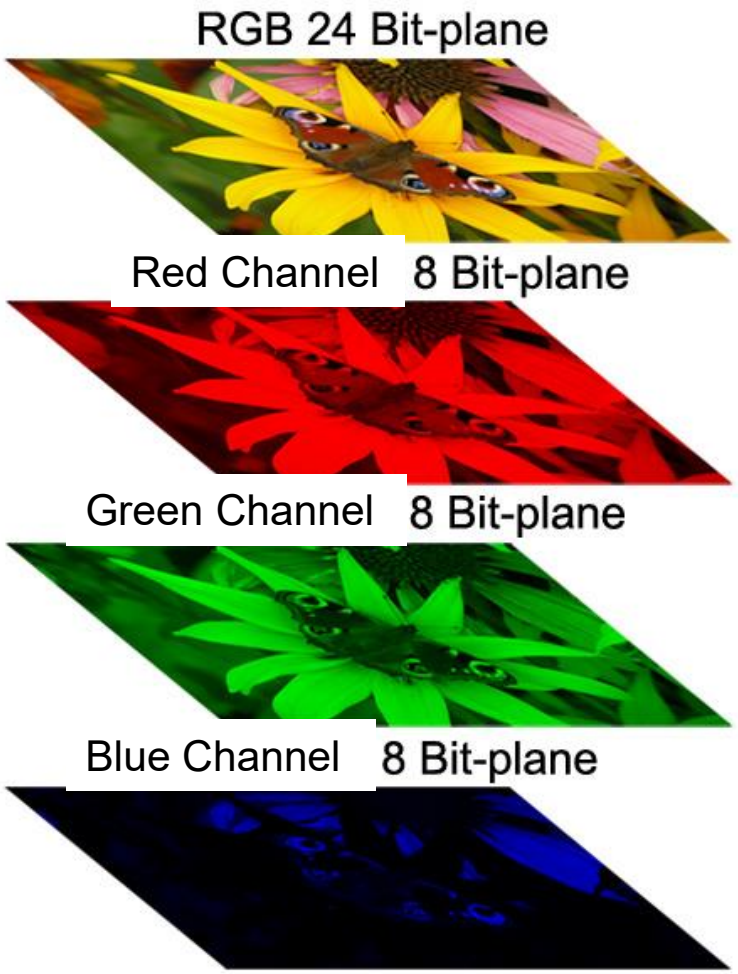
Bit Planes

01101010111010101110101

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 images result in higher accuracy classification than with RGB images

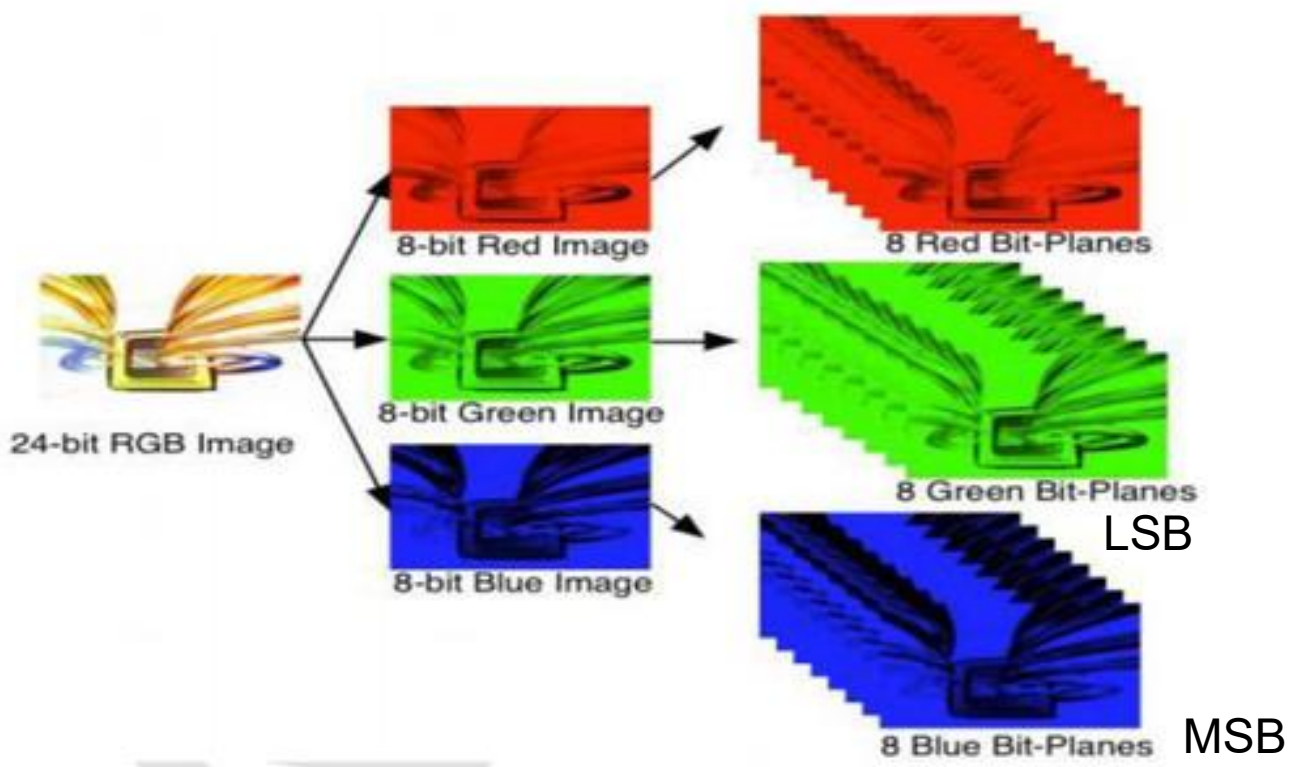
Grayscale bit plane stores intensity info



nth Bit Planes

01110101011101010111010 1

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

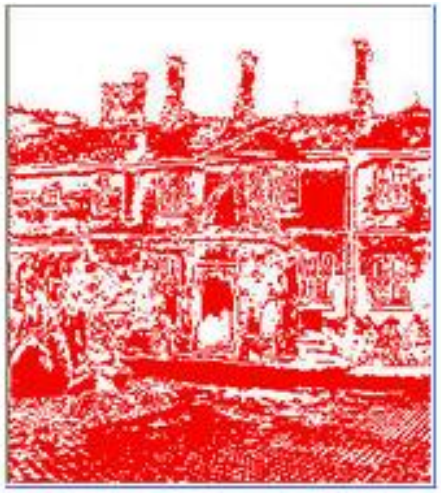


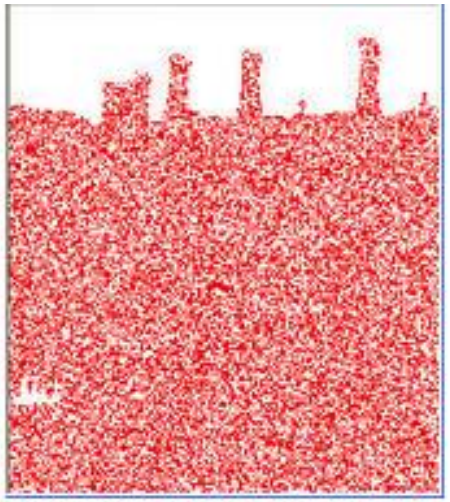
Information Bearing Bit Planes

01110101011101010111010 1

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

(A color image **P**)



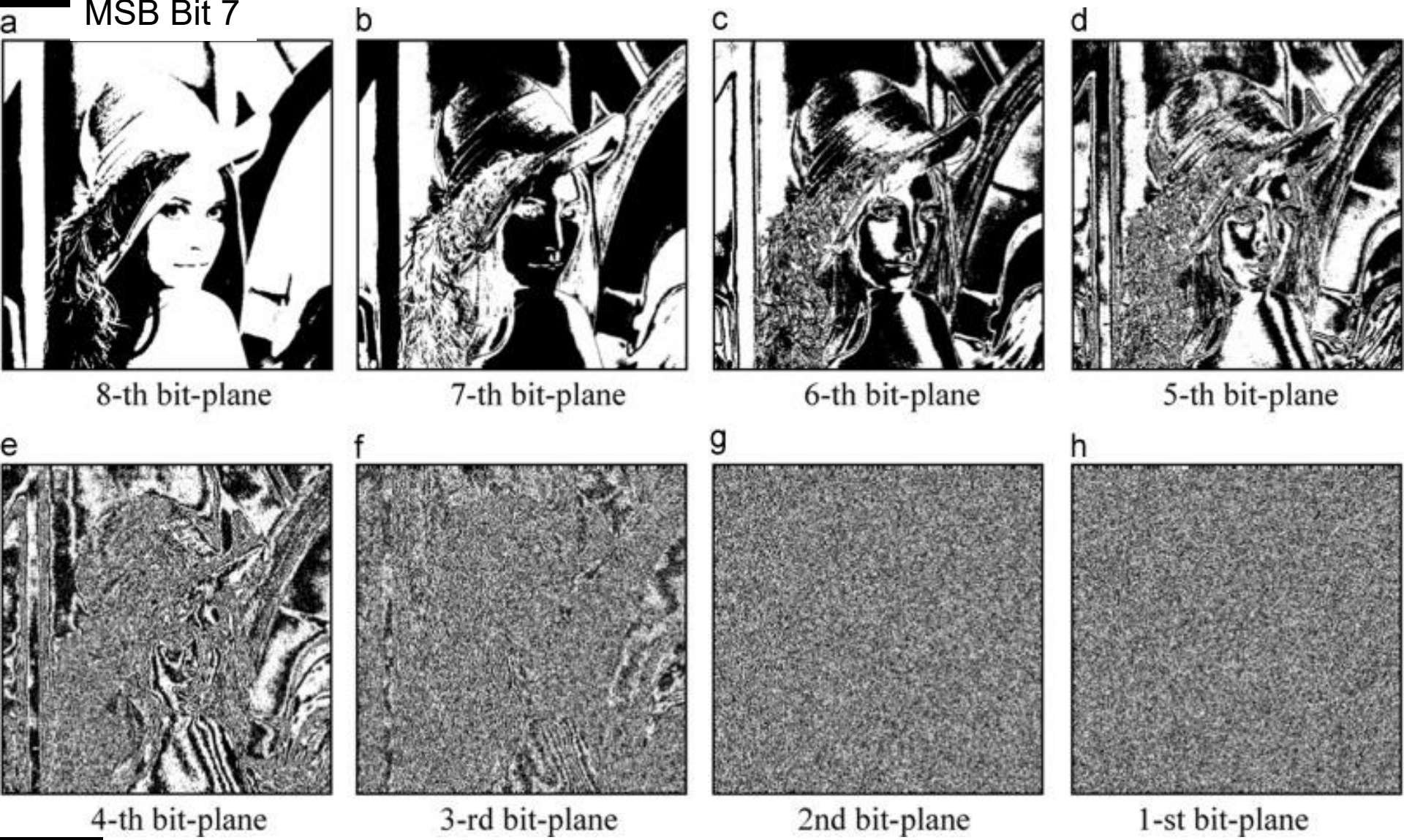


MSB ← → LSB

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

Grayscale Bit Planes

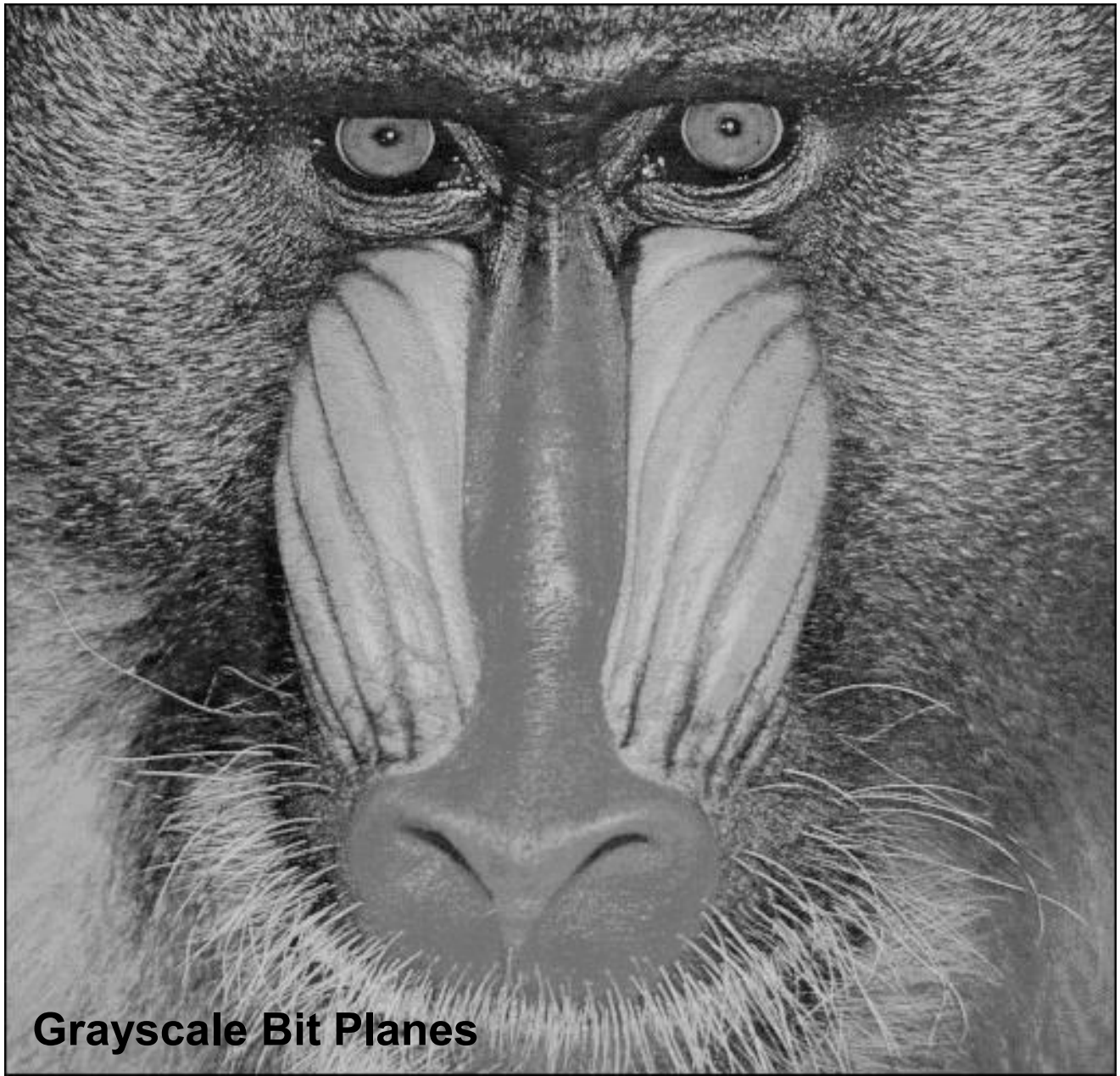
MSB Bit 7

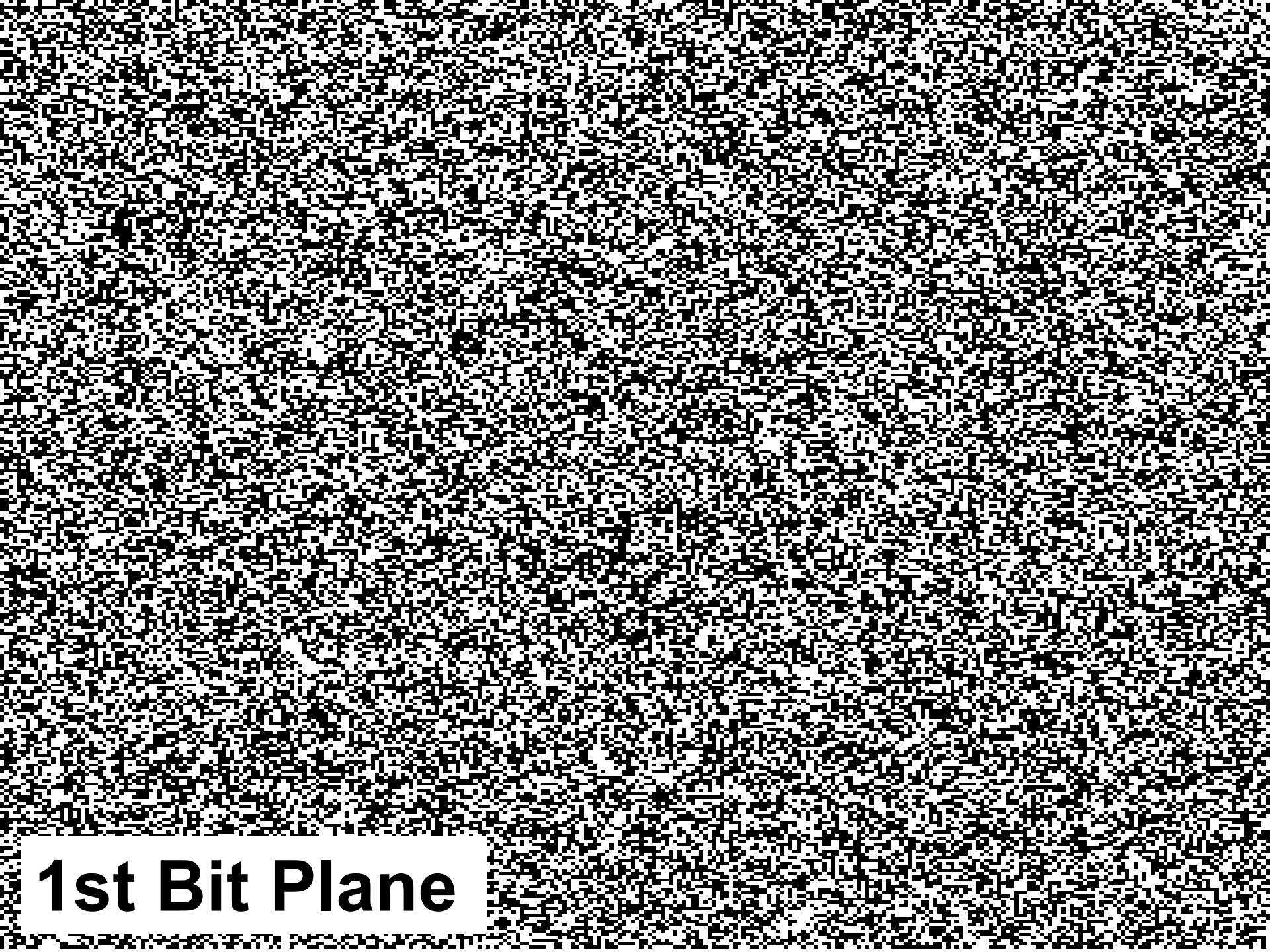


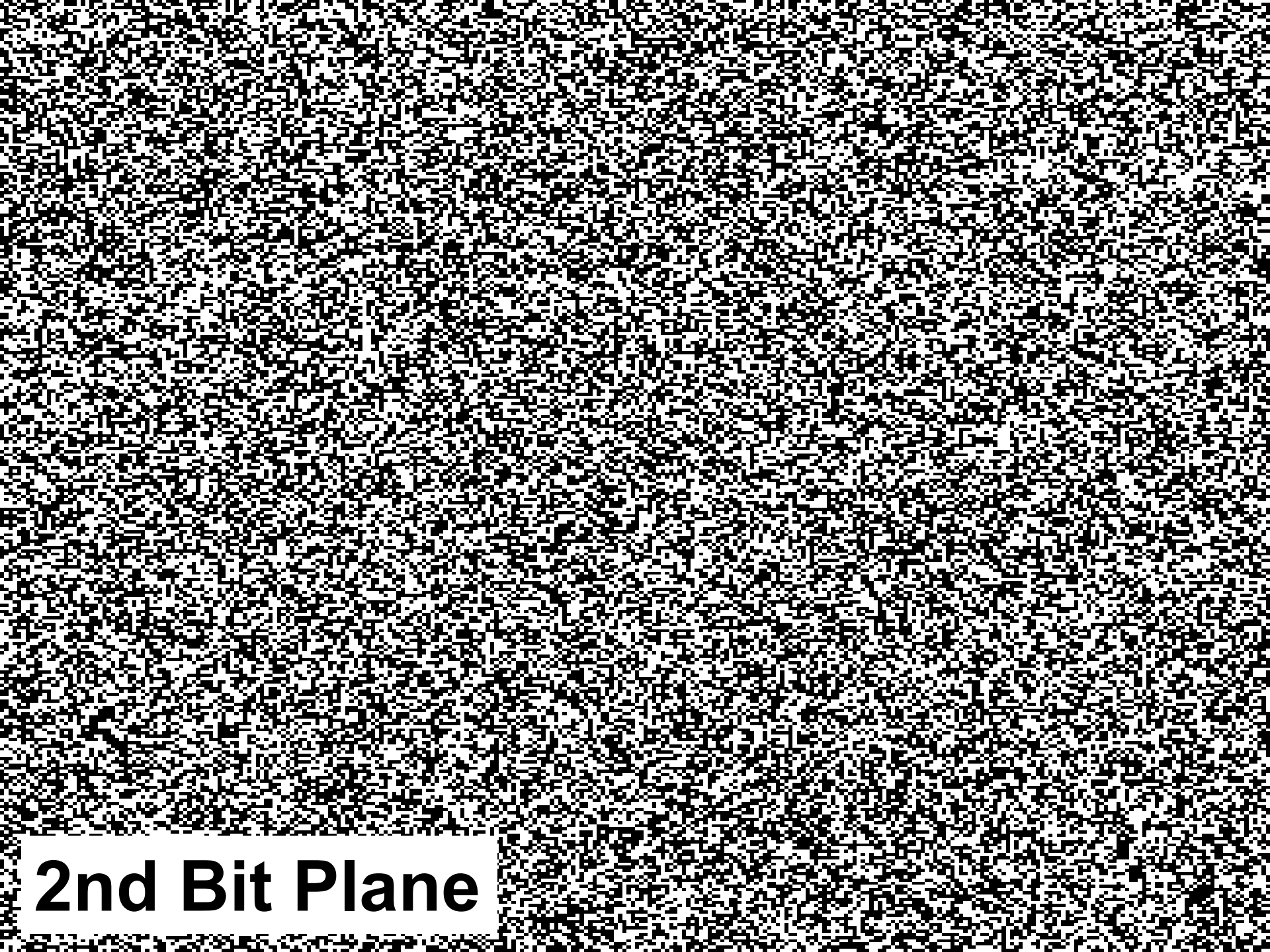
LSB Bit 0

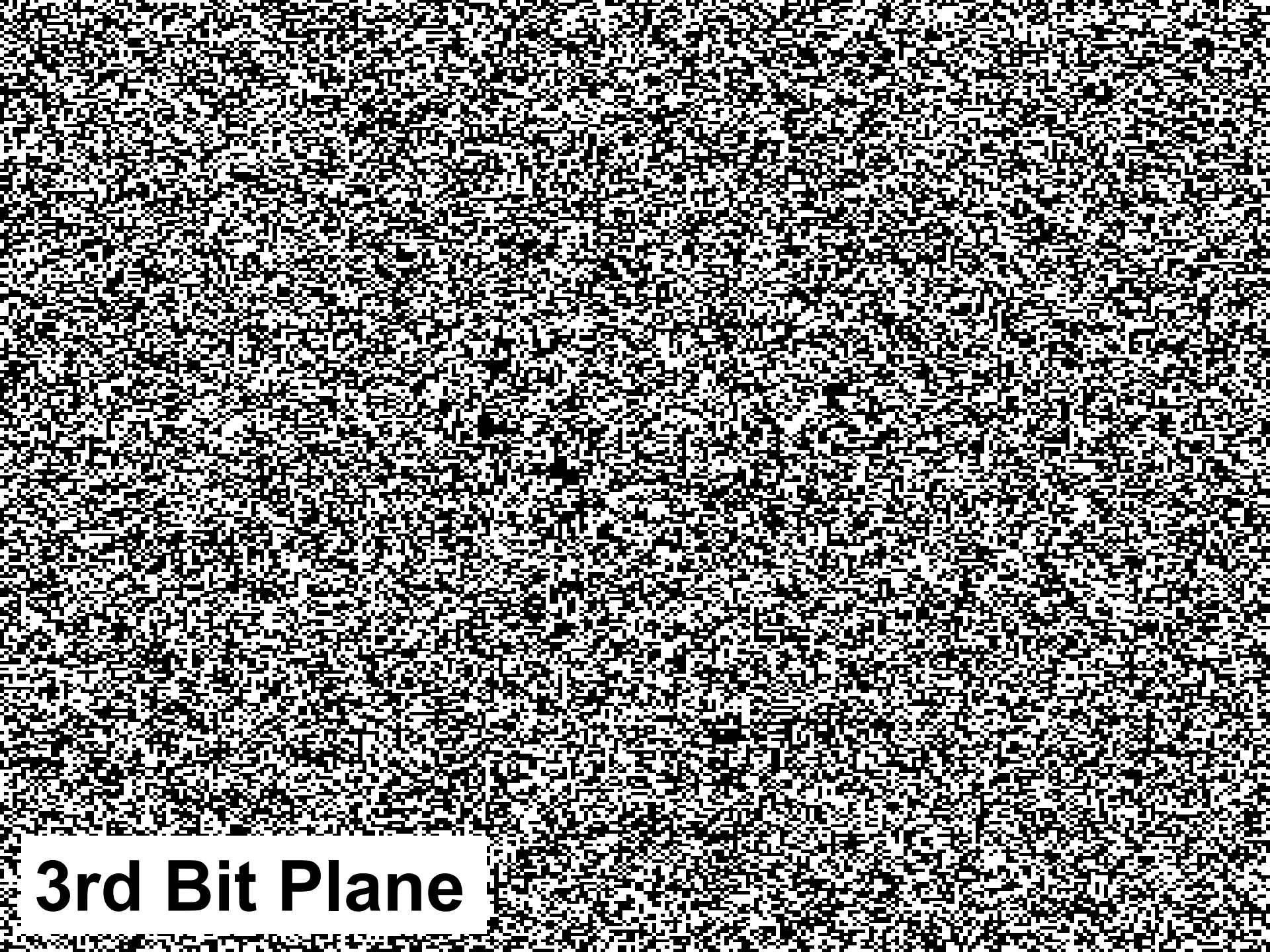
Another Example - Original Image

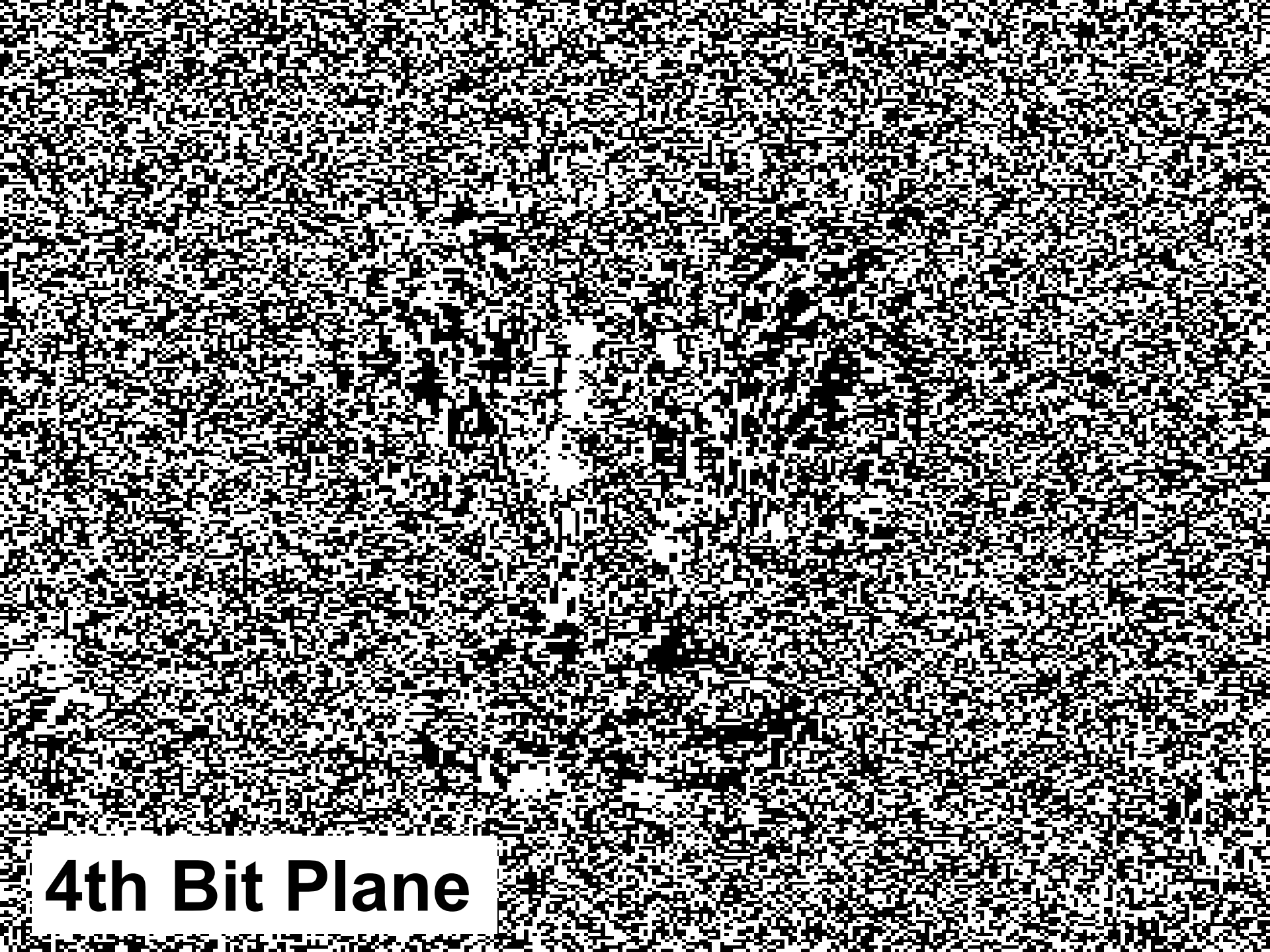
Grayscale Image

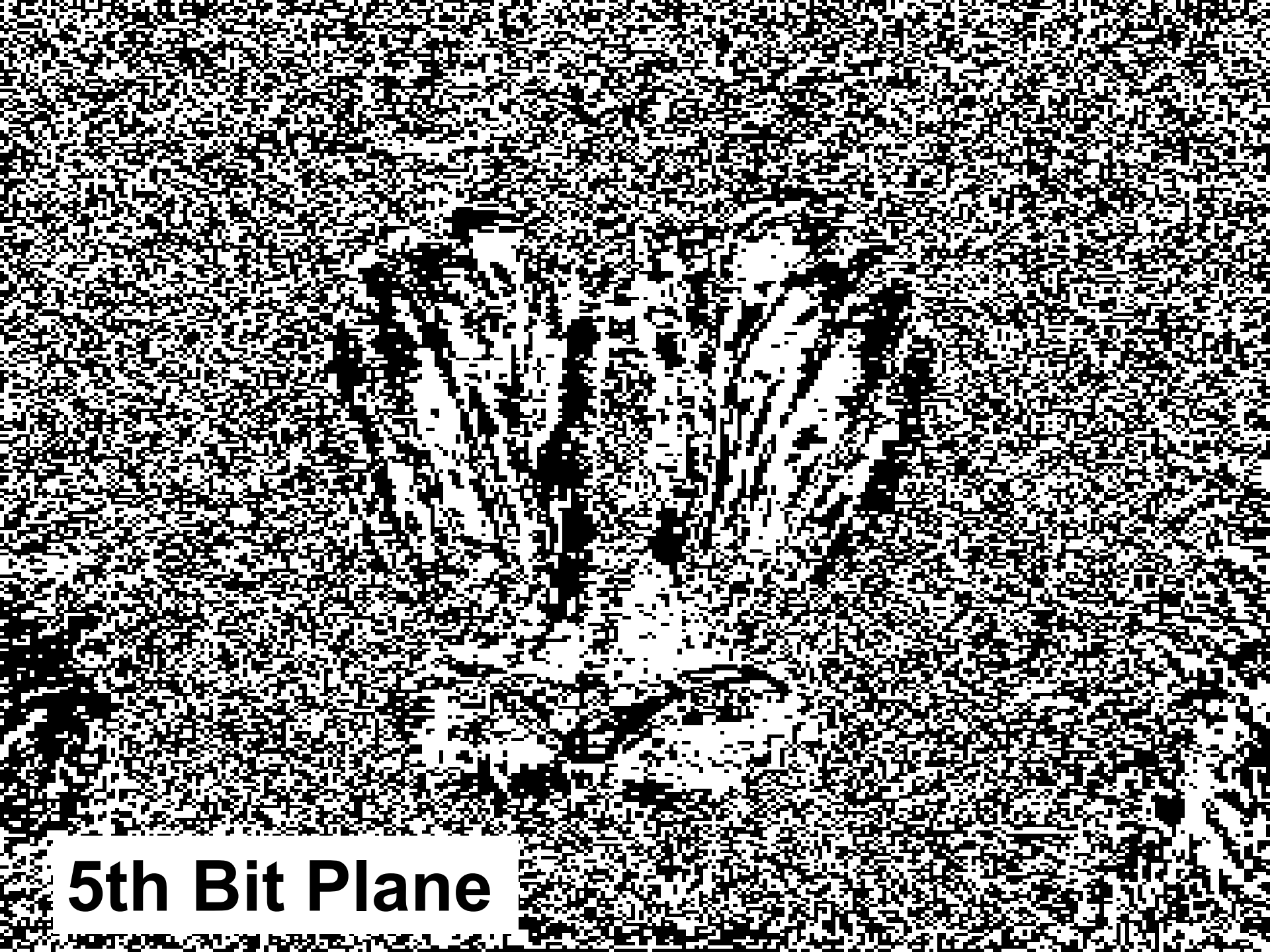


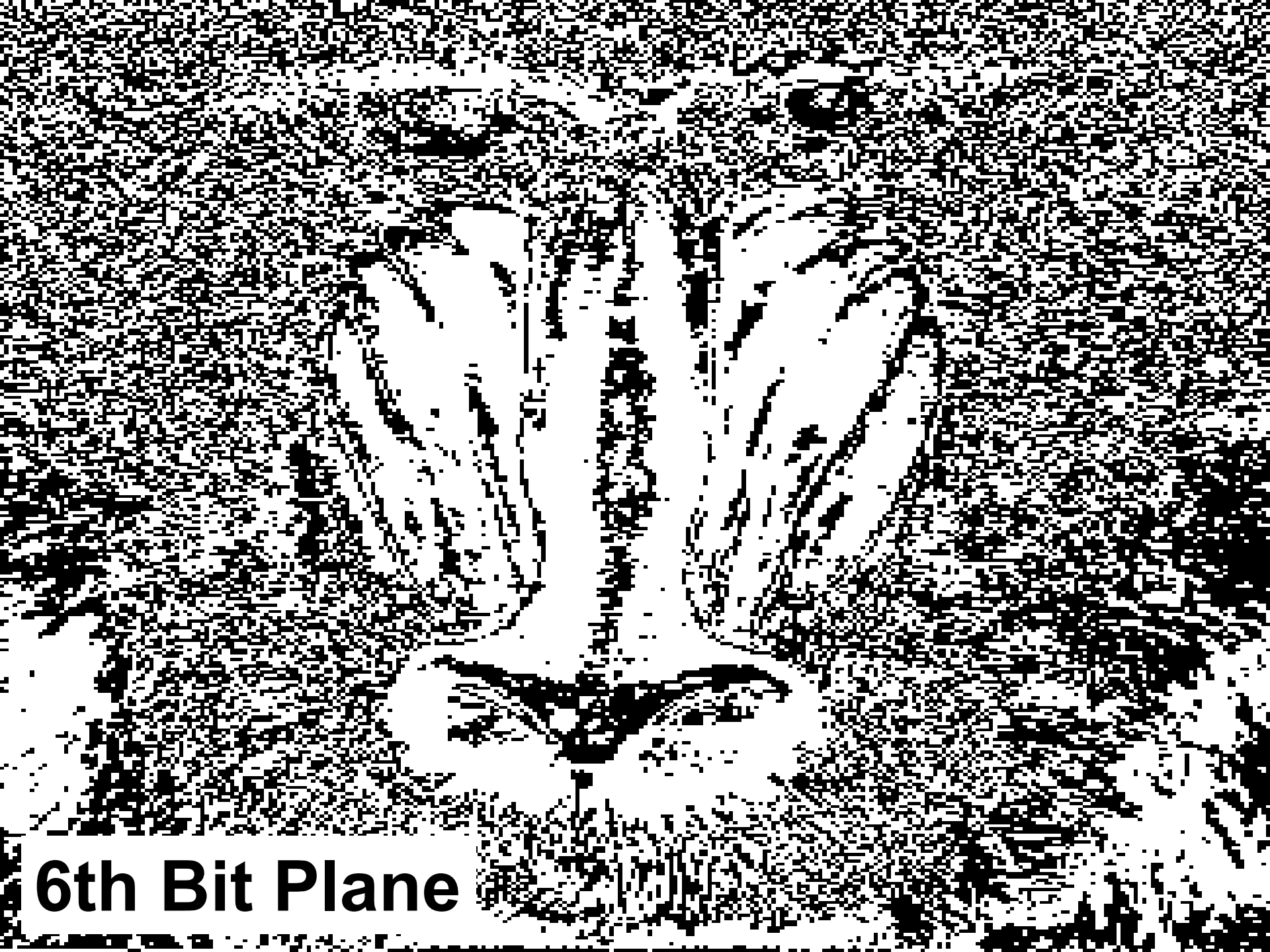

1st Bit Plane

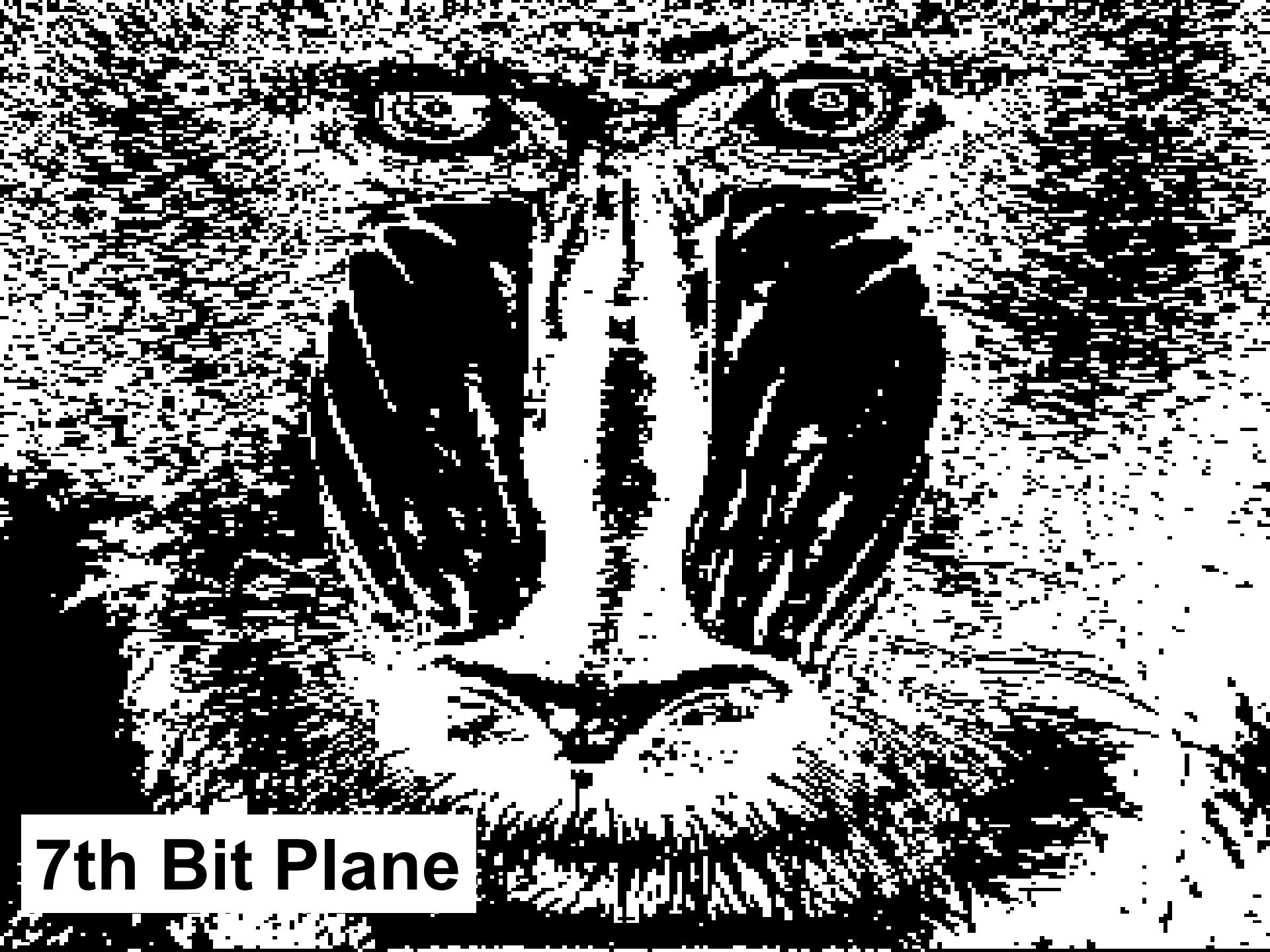

2nd Bit Plane


3rd Bit Plane


4th Bit Plane


5th Bit Plane


6th Bit Plane

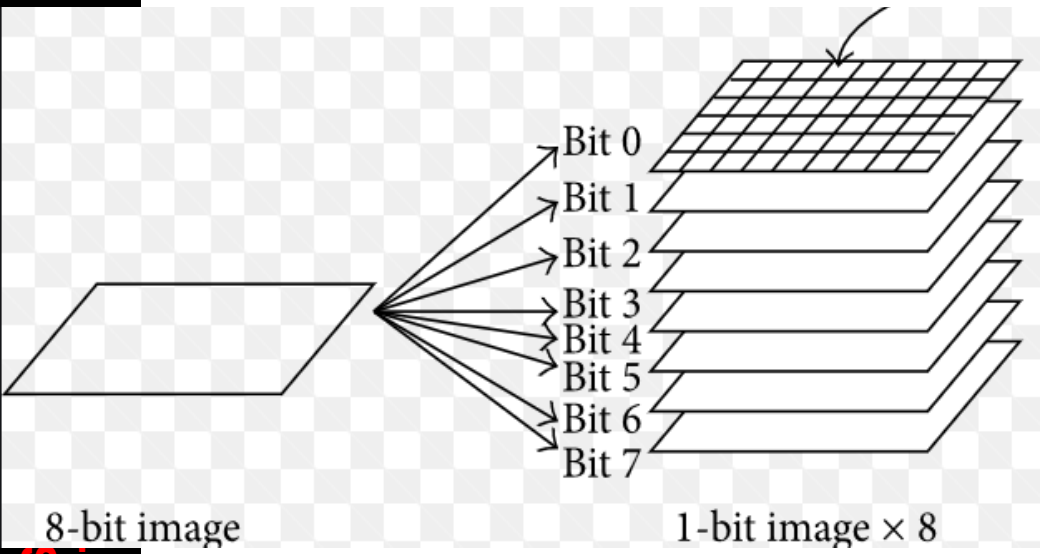

7th Bit Plane


8th Bit Plane

Steganography

Bit Plane Complexity Segmentation Algorithm

Bit Plane Complexity Segmentation (BPCS)Algorithm



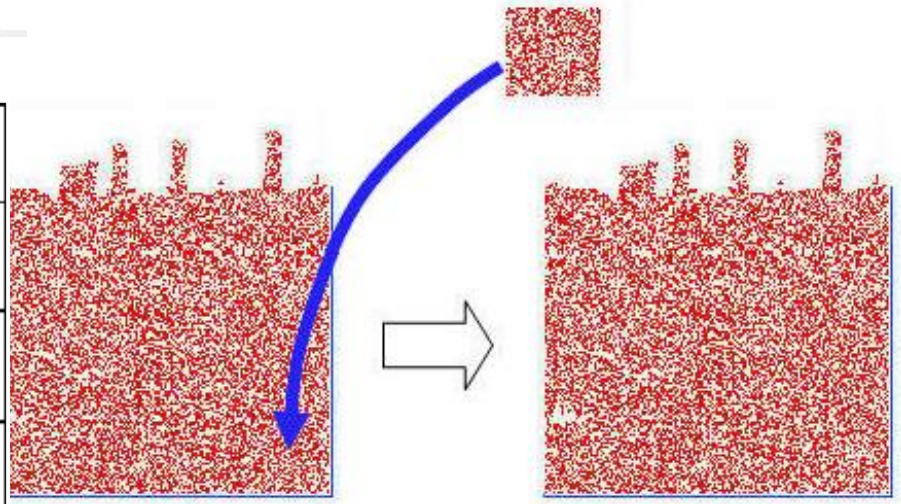
- Cover image is split into 8 bit planes
- Bit plane is divided into complex and not complex regions
- Payload is hidden in complex regions without image deterioration

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

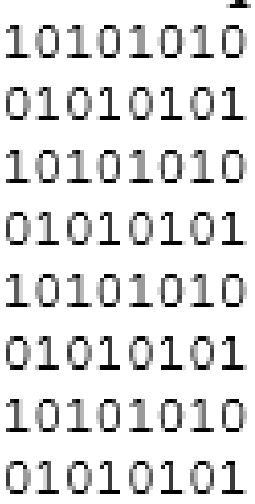
Figure : (a) all white pixels in ima

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

(b) black-white checker board



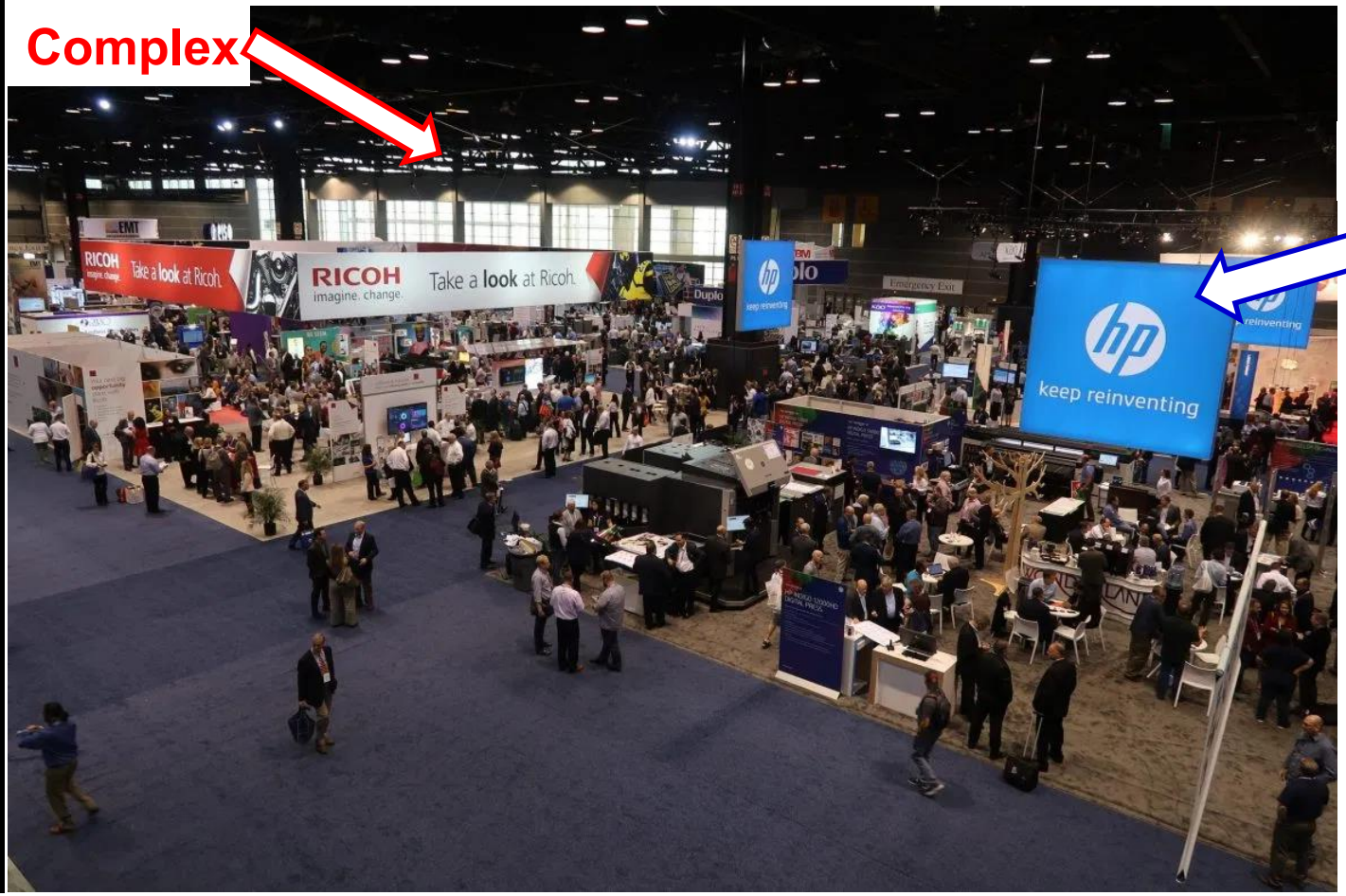
BPCS Algorithm



Conjugate with this:

- 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

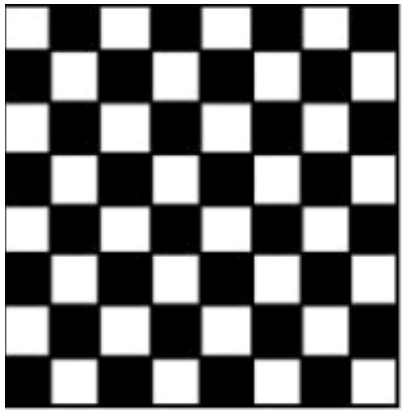


- Use segments of bit planes which are ‘noisy’ (complex) to hide payload
- If payload not noisy enough, perform ‘conjugation’ operation to make it complex
- Works because human eye cannot notice difference in ‘rapidly changing bit patterns’

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 or 01010101
- Do the same for each of the 8 bytes
- The maximum number of changes is 2 x 8 x 7 = 112 (horizontal and vertical variations).
- 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

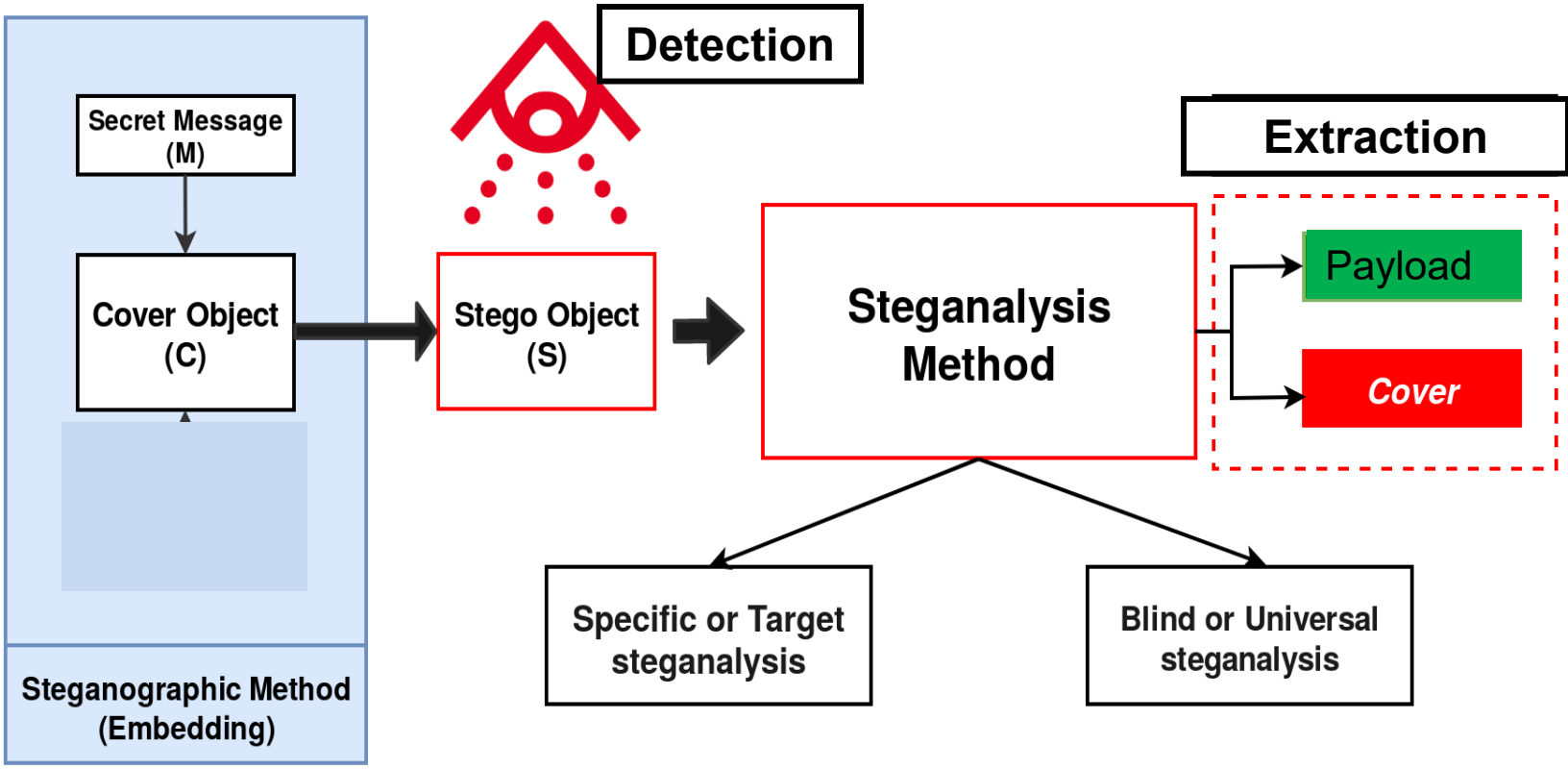


Steganography

Steganalysis

Introduction to Steganalysis

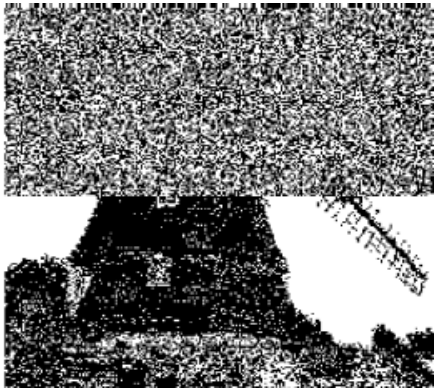
Alice



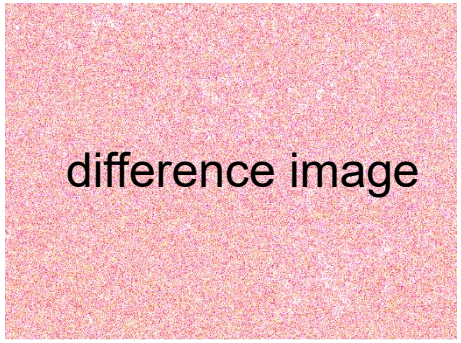
- Steganalysis is an attack on steganography.
- Primary objective is hidden payload detection.
- Secondary objective is to extract the payload.
- Hiding info in digital media changes some media content.
- May introduce visual degradation or unusual characteristic(s).

Visual Steganalysis

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



ORIGINAL STEGO-IMAGE (suspected)

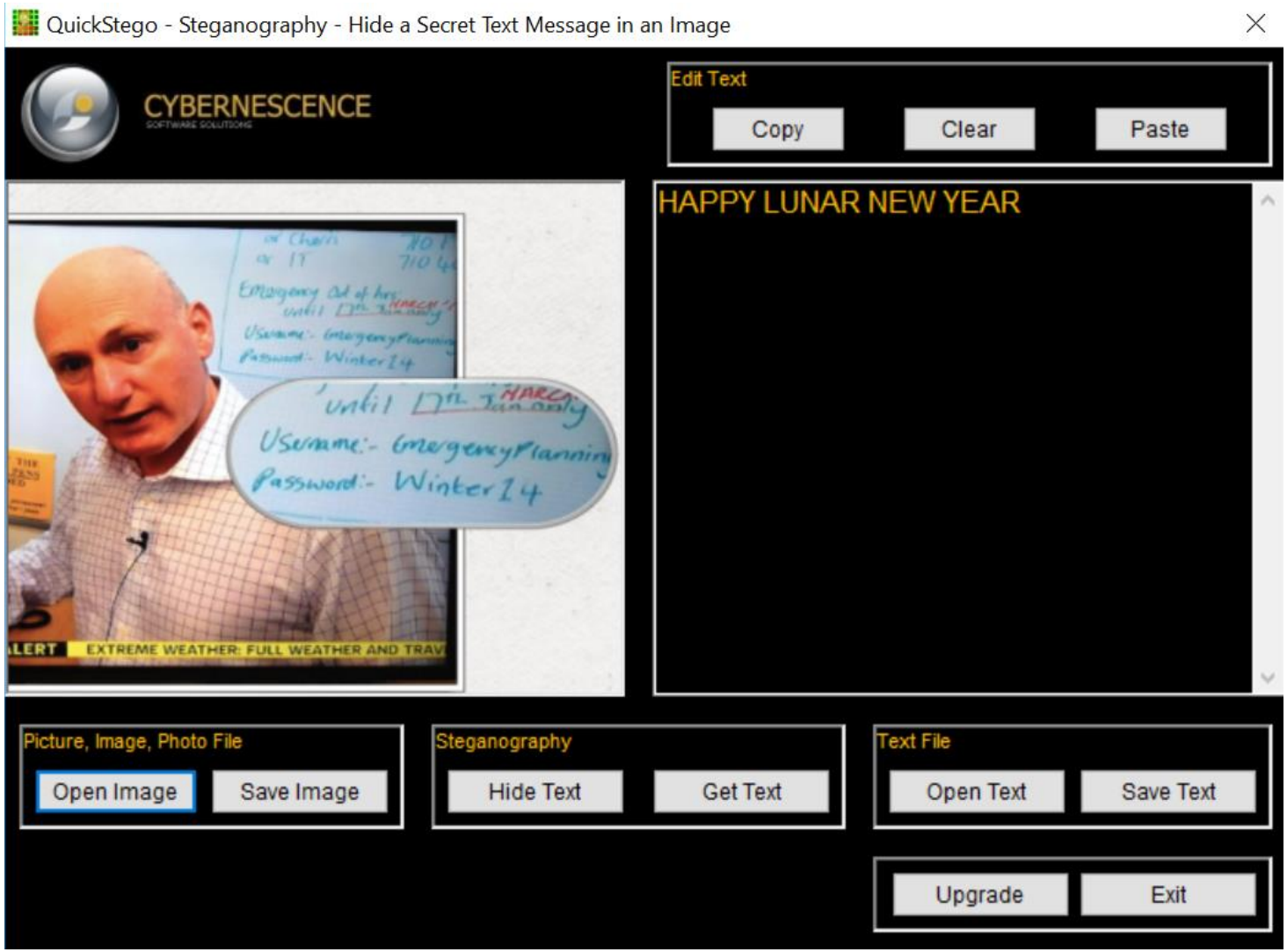


https://www.rapidtables.com/web/color/RGB_Color.html

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

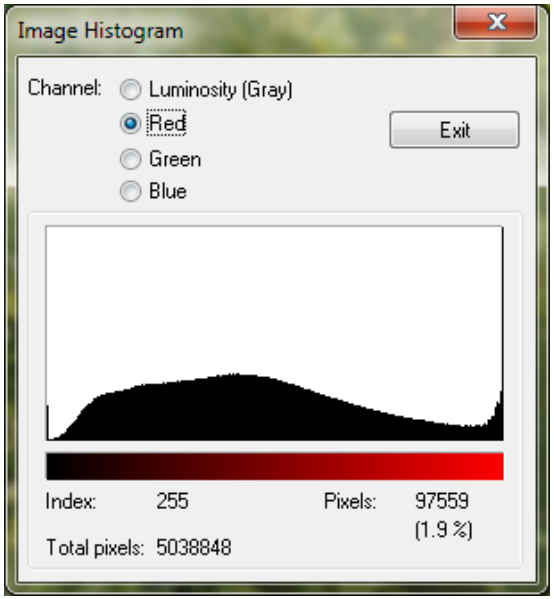
https://www.diffchecker.com/image-compare/

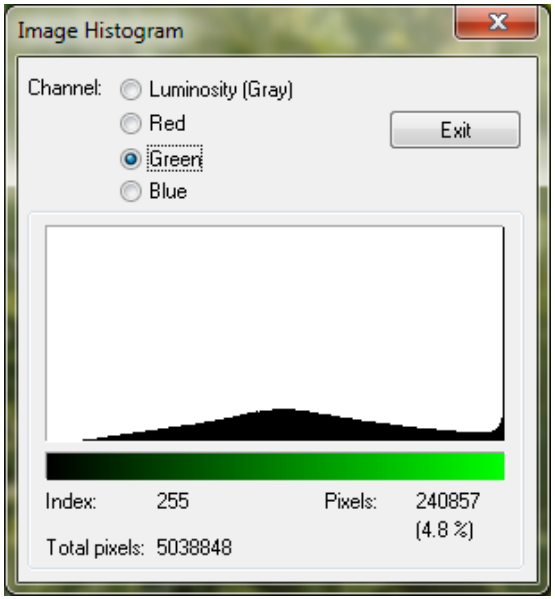
Tool-based Steganalysis

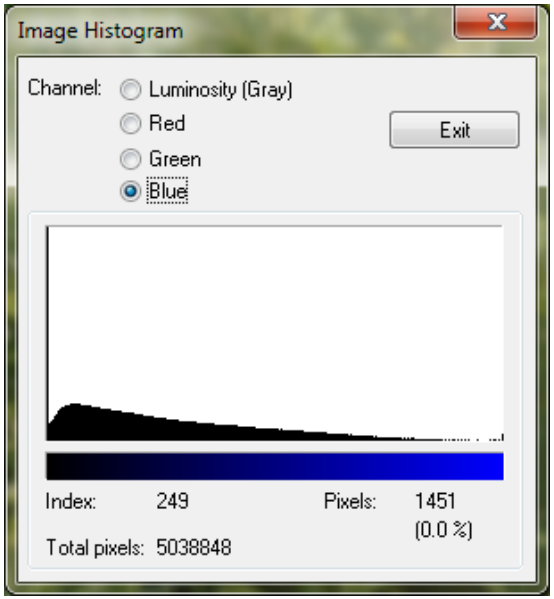


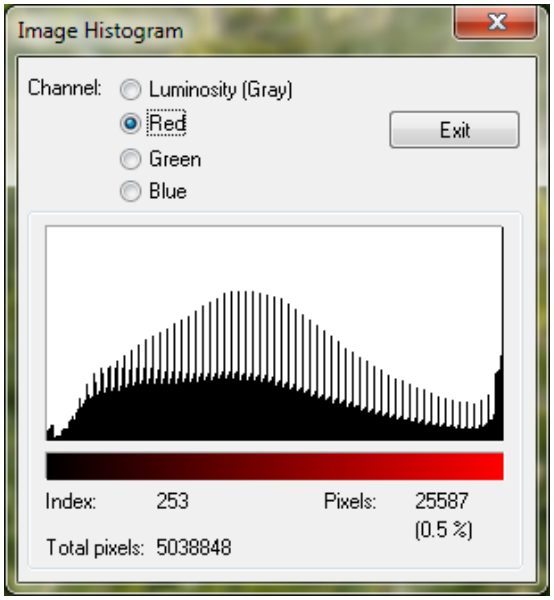
EXAMPLE TOOL: QUICK STEGO

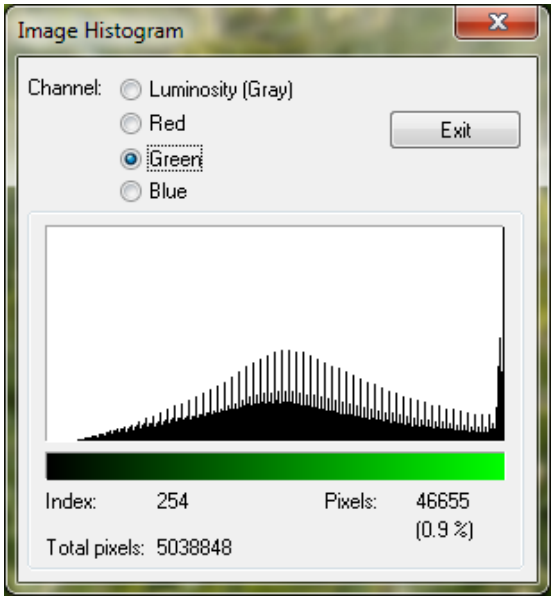
Statistical Steganalysis – 1/2

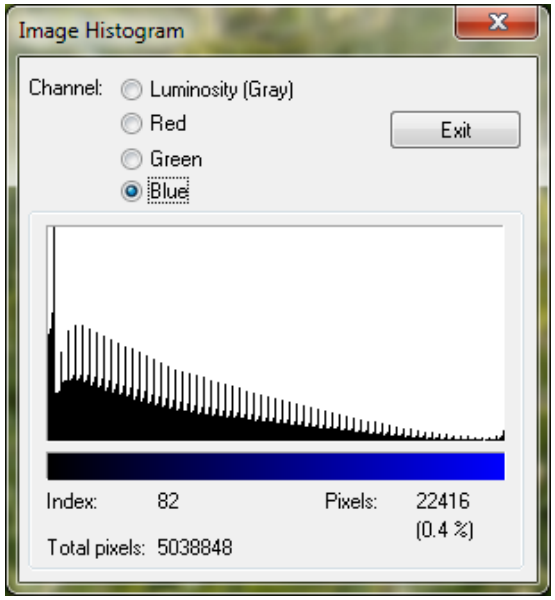




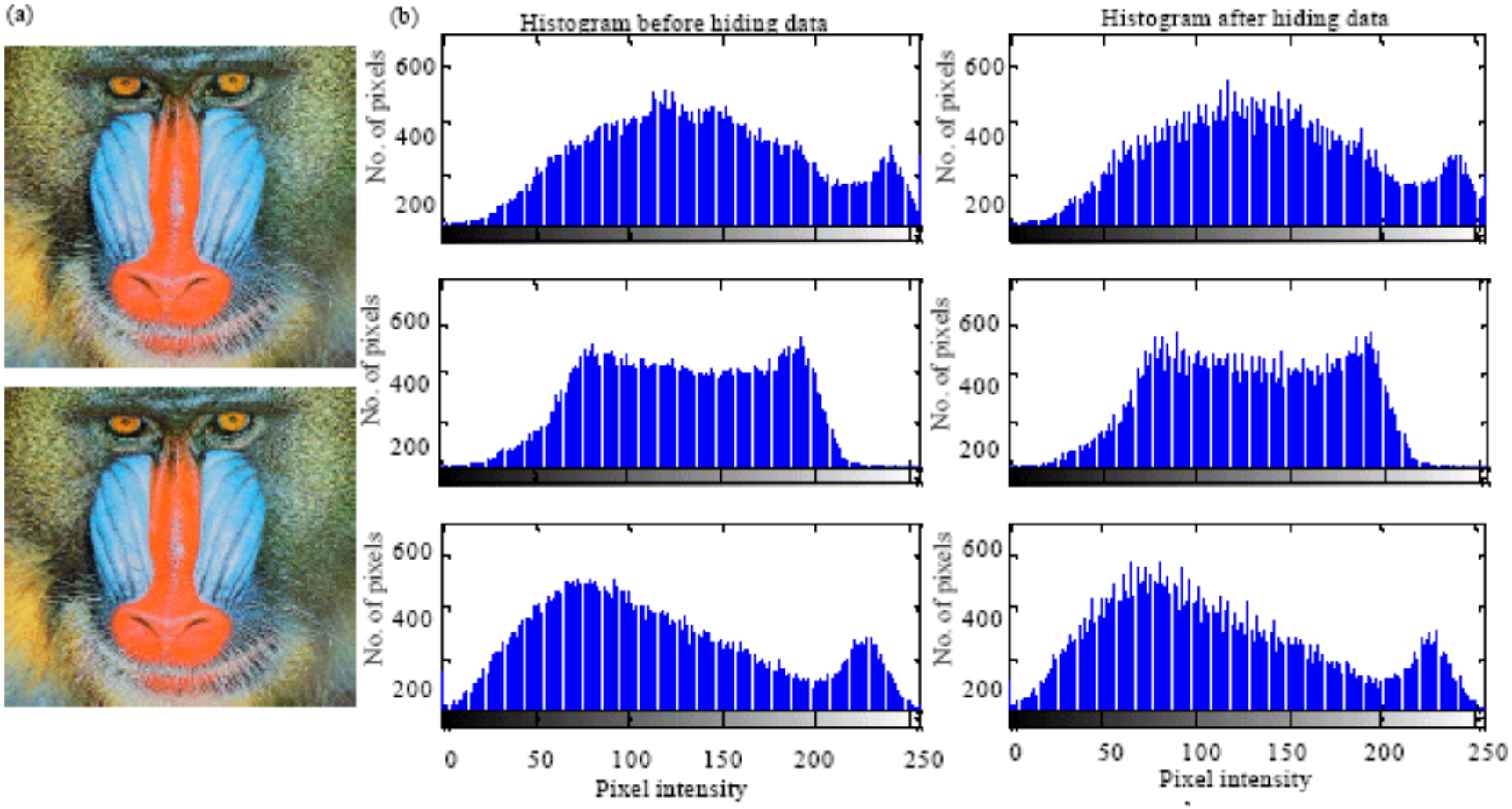








Statistical Steganalysis – 2/2



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

Applied Steganography

Real-World Steganography

From an article about Zheng Xiaoqing, an American convicted of spying for China:

> According to a Department of Justice (DOJ) indictment, the US citizen hid confidential files stolen from his employers in the binary code of a digital photograph of a sunset, which Mr Zheng then mailed to himself.

EDITED TO ADD (2/14): The 2018 criminal complaint has a "Steganography Egress Summary" that spends about 2 pages describing Zheng's steps (p 6-7). That document has some really good detail.

Posted on January 20, 2023 at 7:25 AM • View Comments

https://www.bbc.com/news/world-asia-china-64206950

https://www.courtlistener.com/docket/14983061/1/united-states-v-zheng/ (page 6 – how he did it)

Reference : https://www.schneier.com/tag/steganography/

Summary

- Overview of “Hiding in Plain Sight”.
- Steganography vs Cryptography.
- The Steganography Process – payload, cover object (document or multimedia), stego object.
- 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 – BPCS algorithm.
- Steganalysis (attacking stego) – Visual, Tool-Based, Statistical
- **Next Lecture – Access Control**