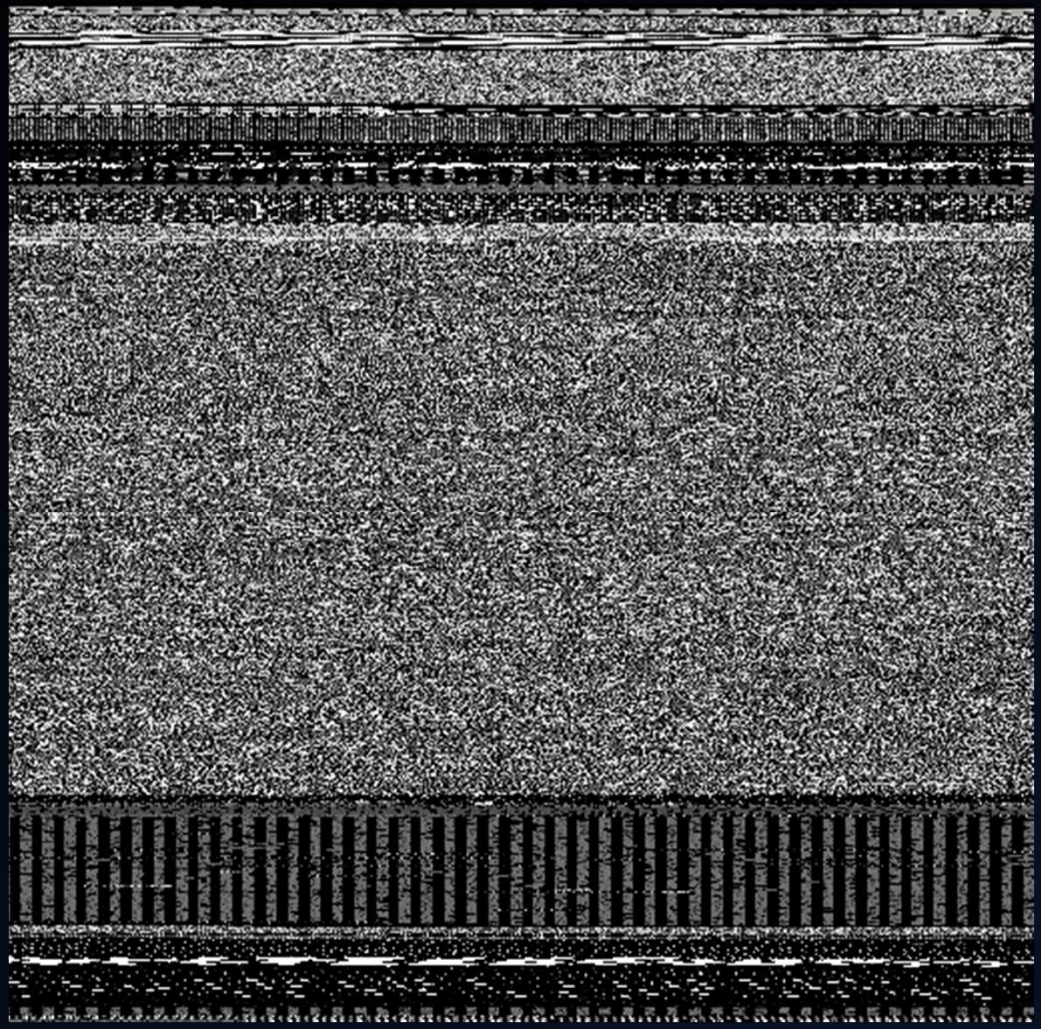
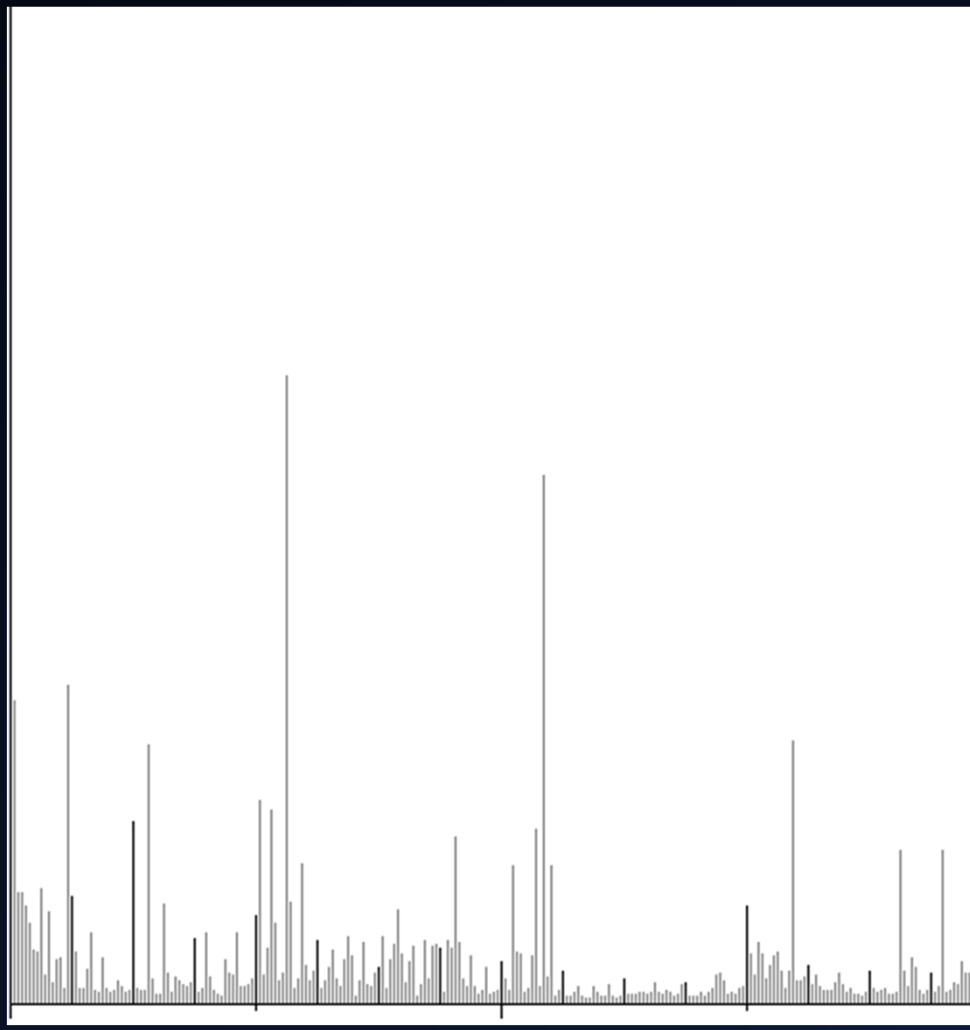
Fast Forensics Using Simple Statistics & Cool Tools

WHAT'S ALL THE **FFUSS** ABOUT?

Do You Hear What I Hear?





Overview – What Can Us Defenders Do?

- Malware Effects
  - What did the malware affect?
  - Where are all the bad files?
  - Did it modify the registry? Processes? Services?
- File Type & Content Identification
  - Is this file really a jpeg?
  - Compressed or encrypted or packed?
- Steganalysis
- Reversing XOR Encryption
- Others … ???

Overview – Attacker Tools

- Executable Packers - Ultimate Packer for eXecutables (UPX)
- Base32/64 Encoders
- Compressors – 7Zip, Winzip, gzip
- Encryptors - Axcrypt
- Wrappers*
  - Disguise a file as a bitmap or wave
- Steganography Tools
  - Steg LSB*, Steg Jpg*, many others

Overview – Defender Tools

- Hex Editors
  - XVI32 is one free one – there are many
- Strings
  - Extract sequences of characters from a file
- Footprint*
  - Snapshot of files, registry entries, processes, and services
- Write Bitmap Histogram (WBH)*
  - Image and the statistics
- Statistical Analyzer*
  - Autonomous identification

TOOL: Wrappers

- Wrappers is a small utility to put a bitmap or wave header on any arbitrary file
  - Essentially disguises a file – it has a valid header
  - You can see or hear any file
  - Wrappers.exe -f Solitaire.exe -t bmp -s g
    - Converts Solitaire.exe into the grayscale image you saw in the intro slide
  - We'll use it for demos

TOOL: Steg LSB

- Hides arbitrary data in Least Significant Bit(s) in bitmap images
- User can choose number of bits (left: 3 bits/pixel, right: 5 bits/pixel)

TOOL: Steg JPG

- Hides arbitrary data in DCT coefficients of jpeg file
- Right: original jpg, left: 22.45% randomized data embedded

MALWARE EFFECTS

- Before identifying the type of a file, you need to find it
- Malware can
  - modify/add/delete …
  - files/registry keys/services …
- After an attack, can you be SURE these modifications are fixed?
- Some malware may look legit and you install them yourself
- Did the uninstall REALLY delete everything?

TOOL: Footprint

- Footprint takes a snapshot of the existing file system, registry, running processes, and services
  - It can also sort the file listing by size and/or date
- After an attack (or install of an undesired program) take another snapshot
- Footprint compares the two and highlights changes

Footprint – File Created

- <4> - EXTRA FILE IN DIR2 --> \~Work\Forensics__Media Files\jpg

- FILE <Betrayal - Copy.jpg> SIZE:146745 bytes
- CREATED:07/07/2013 06:52:37 MODIFIED:09/13/2003 13:49:04

- NOT FOUND in Dir1 \jpg

Footprint – File Deleted

- <3> - EXTRA FILE IN DIR1 --> \~Work\Forensics\Files\IntroSlide

- FILE <hist_Solitaire.exe_z_001.bmp> SIZE:275590 bytes
- CREATED:07/06/2013 23:33:18 MODIFIED:07/06/2013 23:33:18

- NOT FOUND in Dir2 \IntroSlide

Footprint – File Modified

- <5> FILE PROPERTY MISMATCH: \~Work\Forensics\Files
- FILE <hist_TrueCrypt Setup 7.1a.exe.txt>
- <6> - FILE SIZE CHANGE OF <18> BYTES
- file1:11387
- file2:11405
- <D> - FILE MODIFY DATE DIFFERENT
- file1:07/03/2013 23:19:05
- file2:07/07/2013 06:52:06

FILE TYPE CHARACTERISTICS

- Malware often disguises itself to reduce chance of detection
  - Executable files may be named with different extensions, packed, and/or encrypted
  - Other files may contain hidden data
- I've often seen a ".dat" or ".bin" file that is actually an executable
- Double-clicking can result in execution, despite the file extension
- Can we easily determine the true data type of a file?

TOOL: Write Bitmap Histogram

- This tool was inspired by Greg Conti's presentation on visualizing network traffic
- Has been extremely useful to me over the years
- Before discussing the tool and some illustrative examples, a little *MATH*
  - Said in the same tone as "BLAH!"
- Is required

Statistical Background – Entropy & Histograms

- Entropy is a mathematical measure of the average uncertainty of a set of symbols
- Most often we consider bytes, 0 – 255 as the set of symbols we care about
  - The MAX entropy is $\log_2$(#possible symbols)
  - For 256 symbols, the max entropy is 8.0000
  - For base 32 encoded files (i.e 32 symbols), the maximum entropy is 5.0000
  - Guess what the max entropy for base 64 encoded files is???

Statistical Background – Entropy & Histograms

- Entropy is a mathematical measure of the average uncertainty of a set of symbols
- Most often we consider bytes, 0 – 255 as the set of symbols we care about
  - The MAX entropy is $\log_2$(#possible symbols)
  - For 256 symbols, the max entropy is 8.0000
  - For base 32 encoded files (i.e 32 symbols), the maximum entropy is 5.0000
  - Guess what the max entropy for base 64 encoded files is???
    - If you thought "6.0000" --- Very Good! Gold star for you!

Statistical Background – Entropy & Histograms

- P_j = probability of occurrence of a symbol
- Lg(X) = $\log_2(X)$ { 2 to what power = X }
- For byte-sized data, n = 256
- We can *estimate* the probability by counting (histogram)
  - If symbol appears 25 times in 100 byte file, p = 0.25

$$Entropy = H = -\sum_{j=0}^{n-1} P_j \lg P_j = \sum_{j=0}^{n-1} P_j \lg \frac{1}{P_j}$$

- Encrypted (random) files have the most uncertainty
- A file with a single value has the least, H = 0 (log 1 = 0)

I KNOW it looks difficul
but it is really EASY!
(Once you figure it out.)

Statistical Background – Entropy & Histograms

- Bottom Line: Higher entropy, higher uncertainty
  - Compressed: H = 7.6+
  - Encrypted: H = 7.99+
  - Text: H = 4.5 +/-
- The entropy measurement is only accurate with sufficient data
  - Can't get entropy of 7.99+ for a 1-byte encrypted file
  - For fairly accurate measurement, need around 4K
    - There is research on this, but that's for another day
  - Accuracy increases with increasing data size

Statistical Background – Entropy & Histograms

- A Histogram is a count of the number of occurrences of each symbo
  - # ZERO's in the file shown on the left edge, # 255's on the right
  - At every 16th interval, line is darker
- Extremely useful for analysis of a file's contents
- Can be used to identify the likely data content of a file
- Many file types have unique histogram characteristics
  - Some exceptions
- An image (or audio) of the file is useful too
  - Shows position of data file

Fast File Type Identification - Approach

- File Extension
  - Not super accurate, but a good start
- Magic Number, Header Validation
  - Wrappers kind of defeats this approach
- Visualization
- Audialization (Have you *heard* this word before?)
- Statistics

What's in a File?

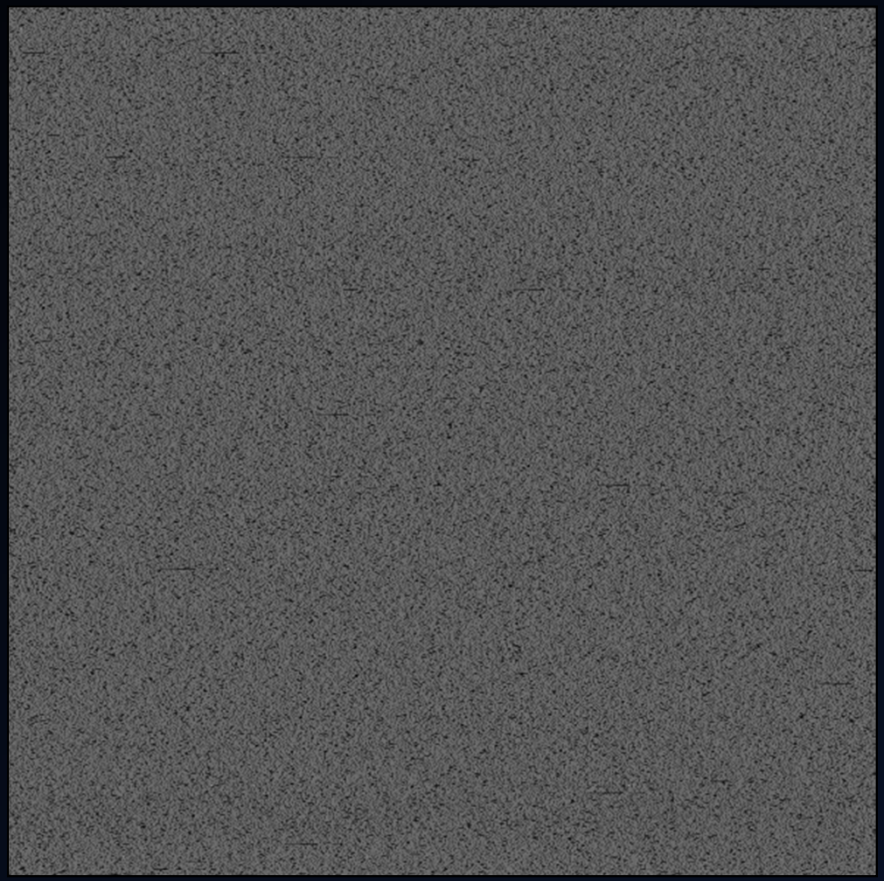
- We can use entropy, histograms, visualization, and audialization to quickly and effectively check:
  - Does the file match it's extension?
  - Does it have unusual data?
  - Does it have hidden data?
  - Is there data tacked onto the end?
  - Is it compressed/encrypted?
- Each slide will show an image of the file's contents and a histogram, as well as the estimated entropy

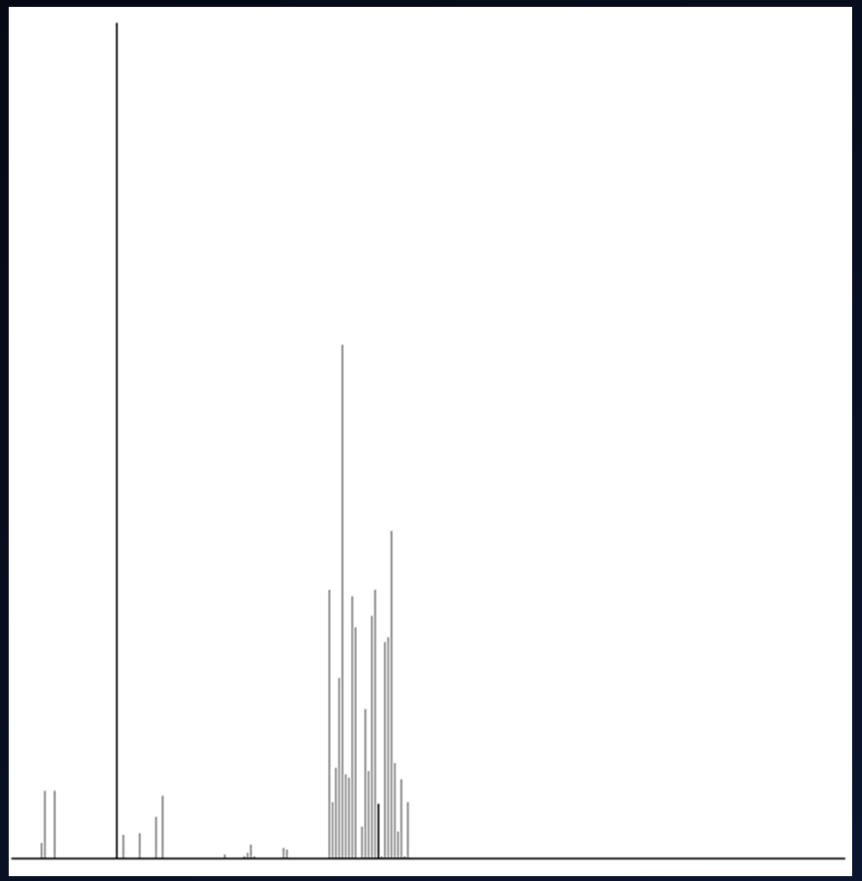
Using the Write Bitmap Histogram Tool

- Run it without any options and usage instructions are printed
- wbh_5.57.exe Novels.txt –b
- Creates a graphical and textual histogram of “Novels.txt”
- The –b option creates the image of the file
- The graphical histogram is scaled, showing relative frequency counts

Text File

- H=4.48469



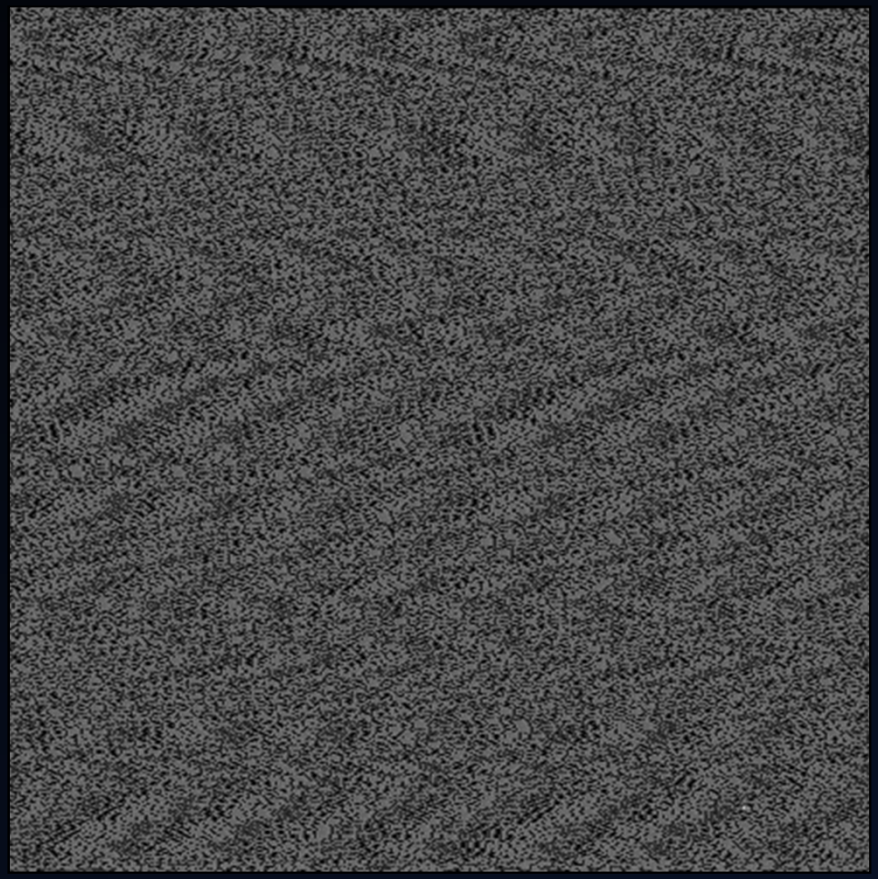


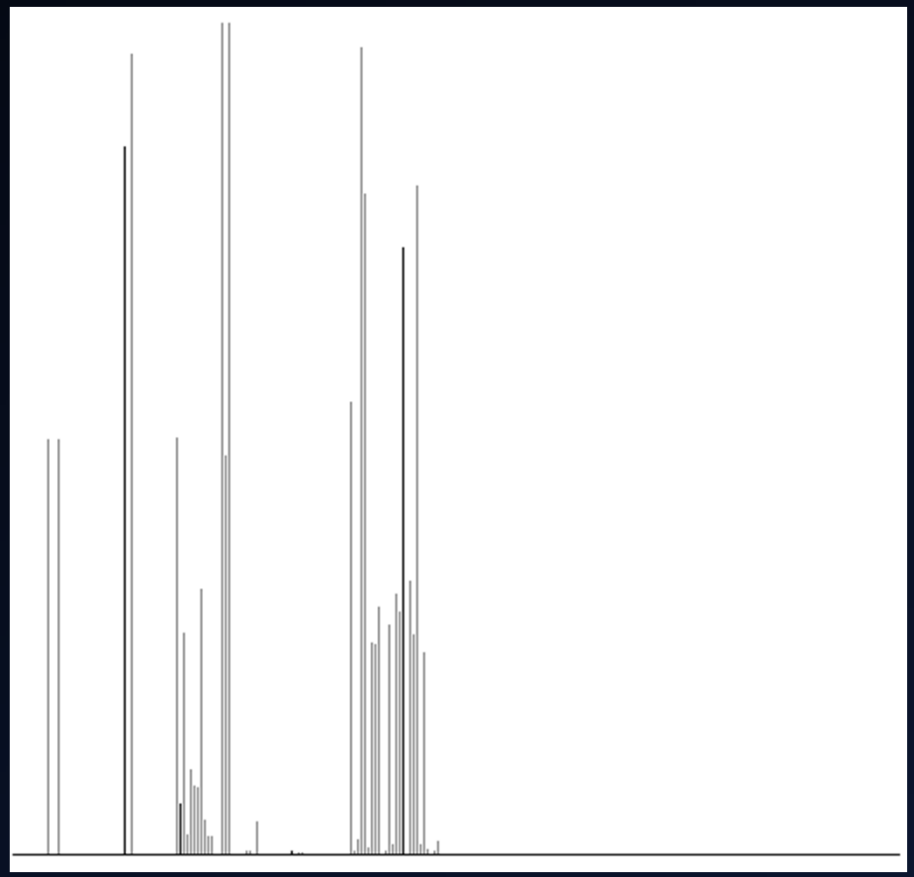
Text File – Textual Histogram

- a, 097 [61],10631 (3.755%)---------+----
- b, 098 [62],4117 (1.454%)-----
- c, 099 [63],4650 (1.642%)------
- d, 100 [64],3784 (1.336%)-----
- e, 101 [65],16391 (5.789%)---------+---------+-
- f, 102 [66],2185 (0.772%)--
- g, 103 [67],3102 (1.096%)----
- h, 104 [68],4049 (1.430%)-----
- i, 105 [69],8865 (3.131%)---------+-
- j, 106 [6A],211 (0.075%)-

HTML File

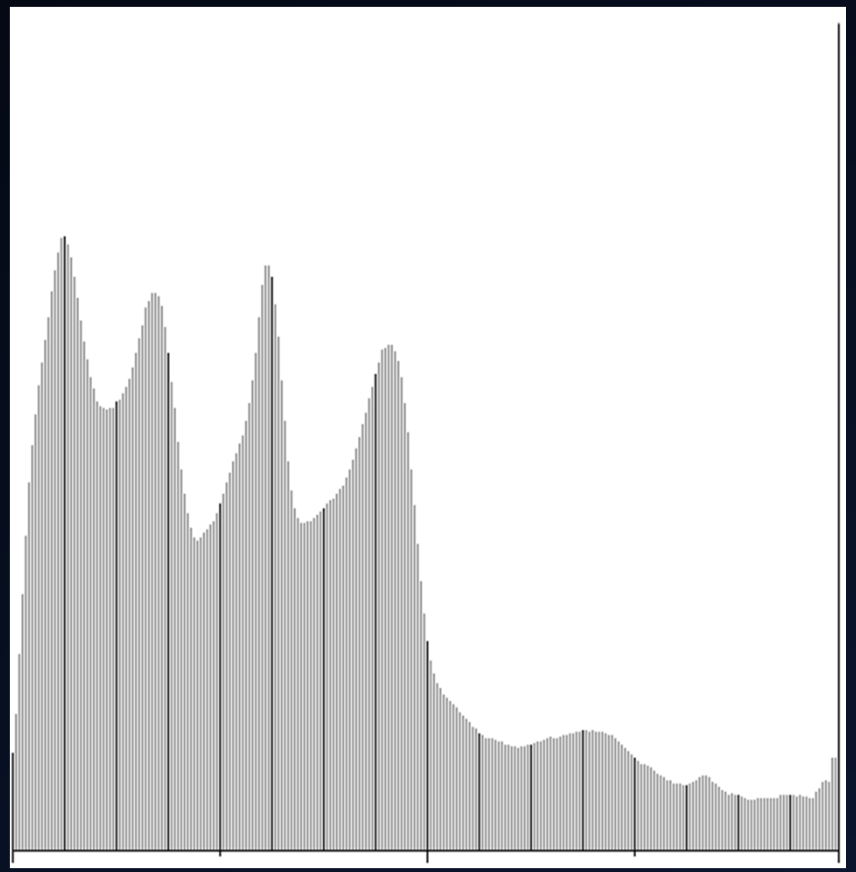
- H=4.70042





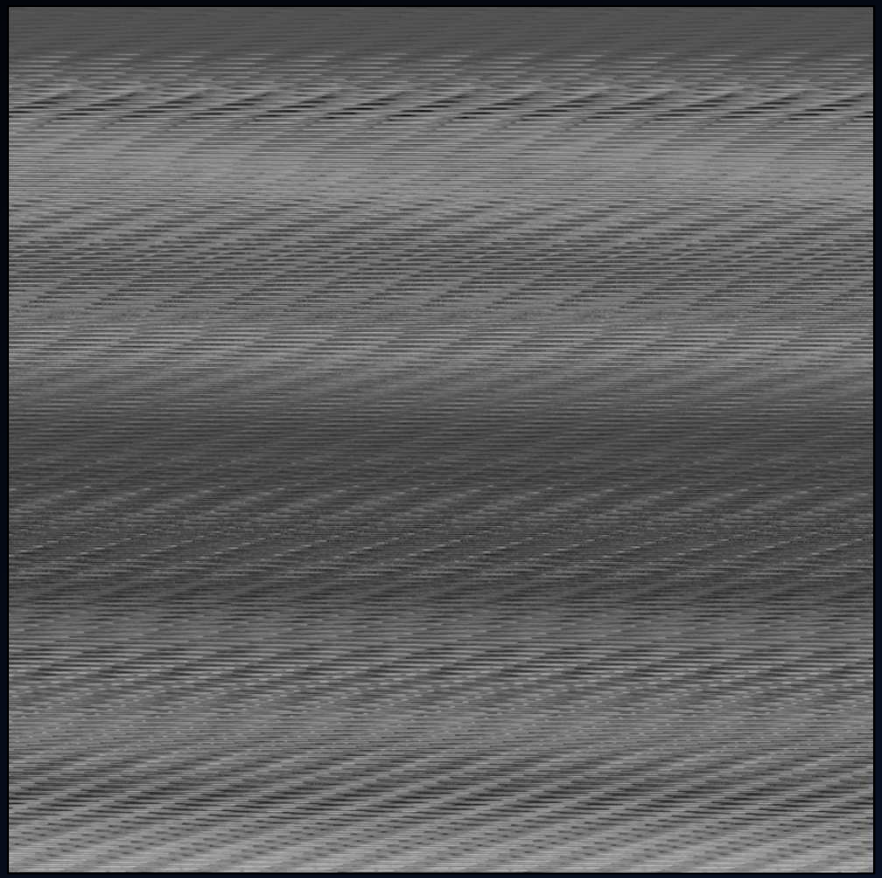
24-Bit Full Color Bitmap

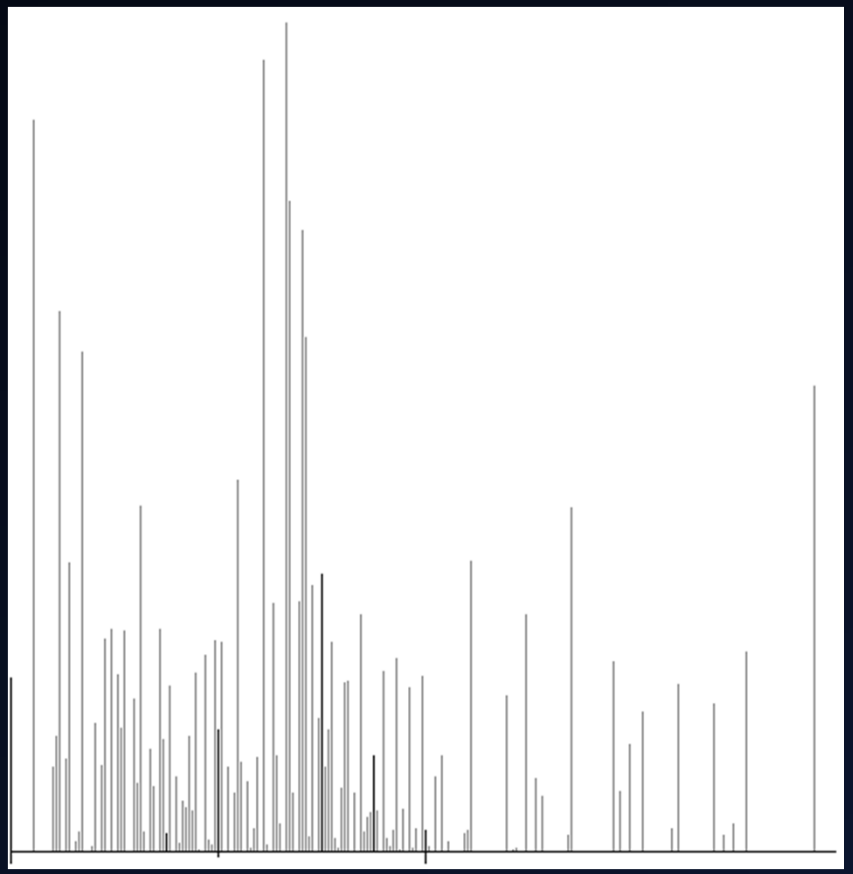
- H=7.63054



8-Bit Grayscale Bitmap

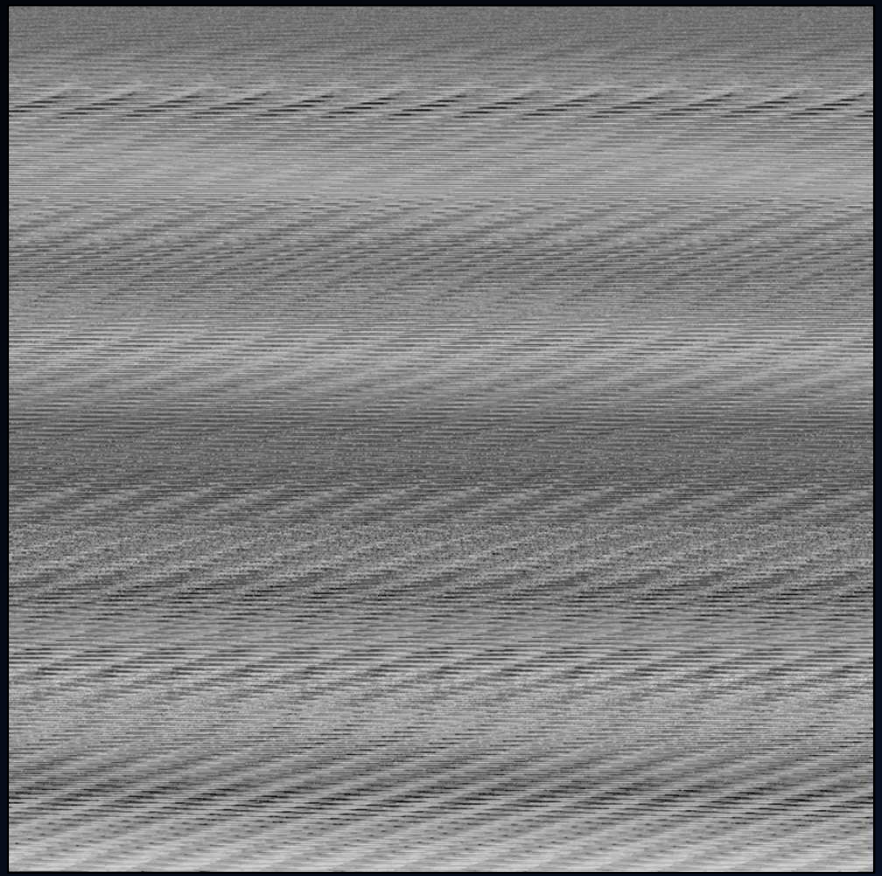
- H=6.14182

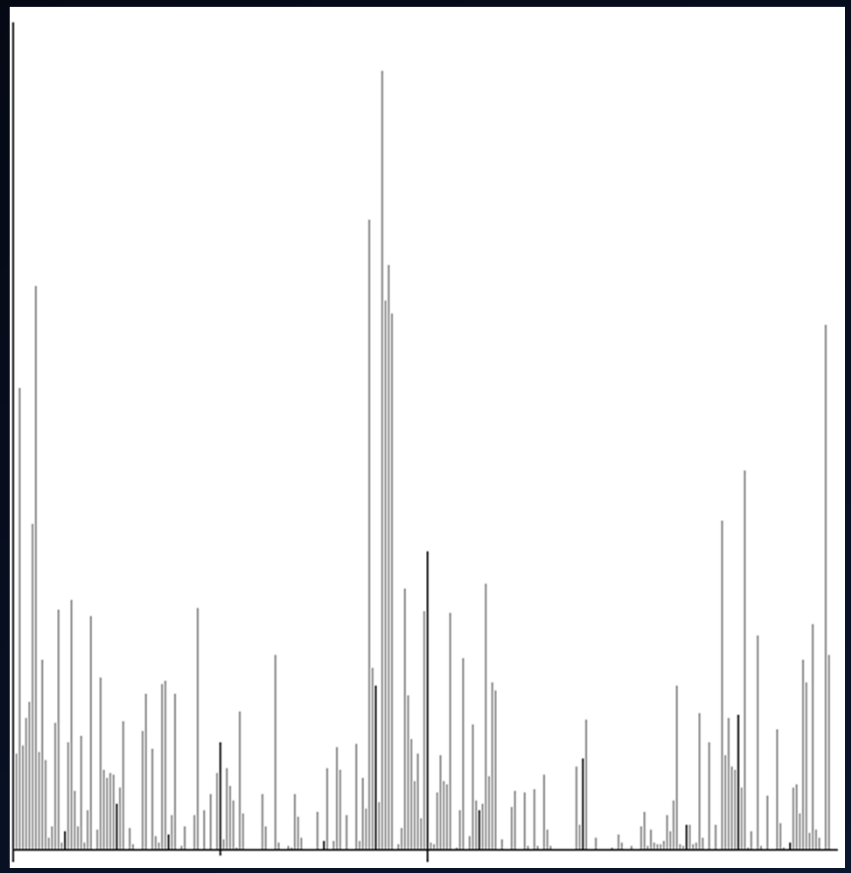




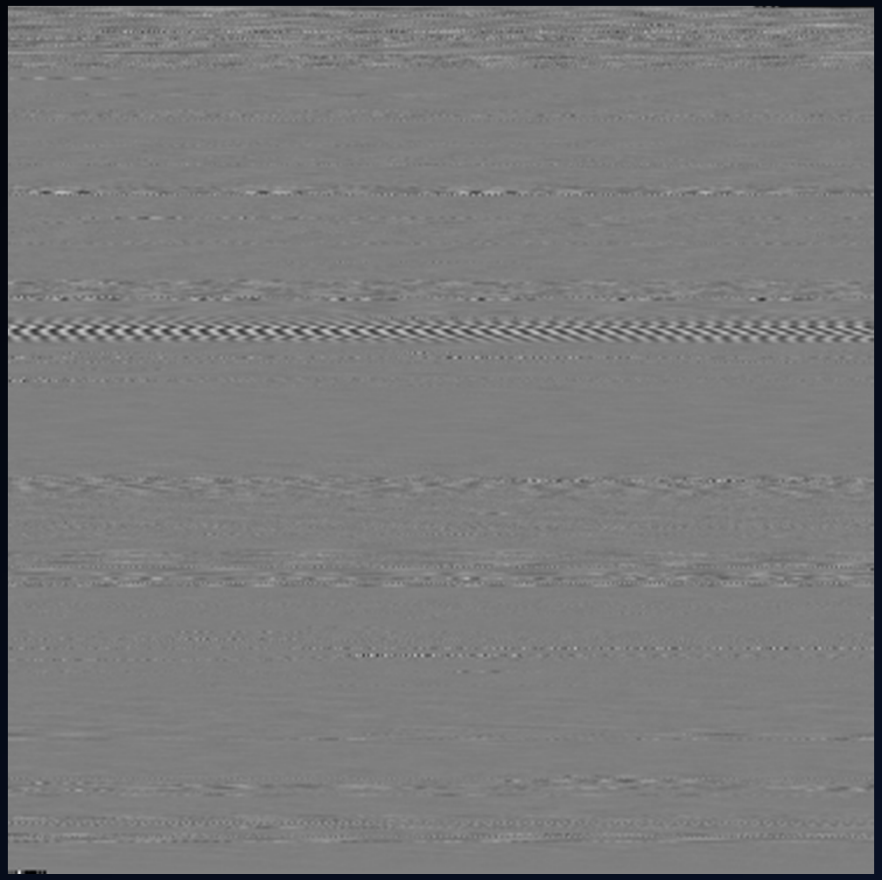
8-Bit Color Bitmap

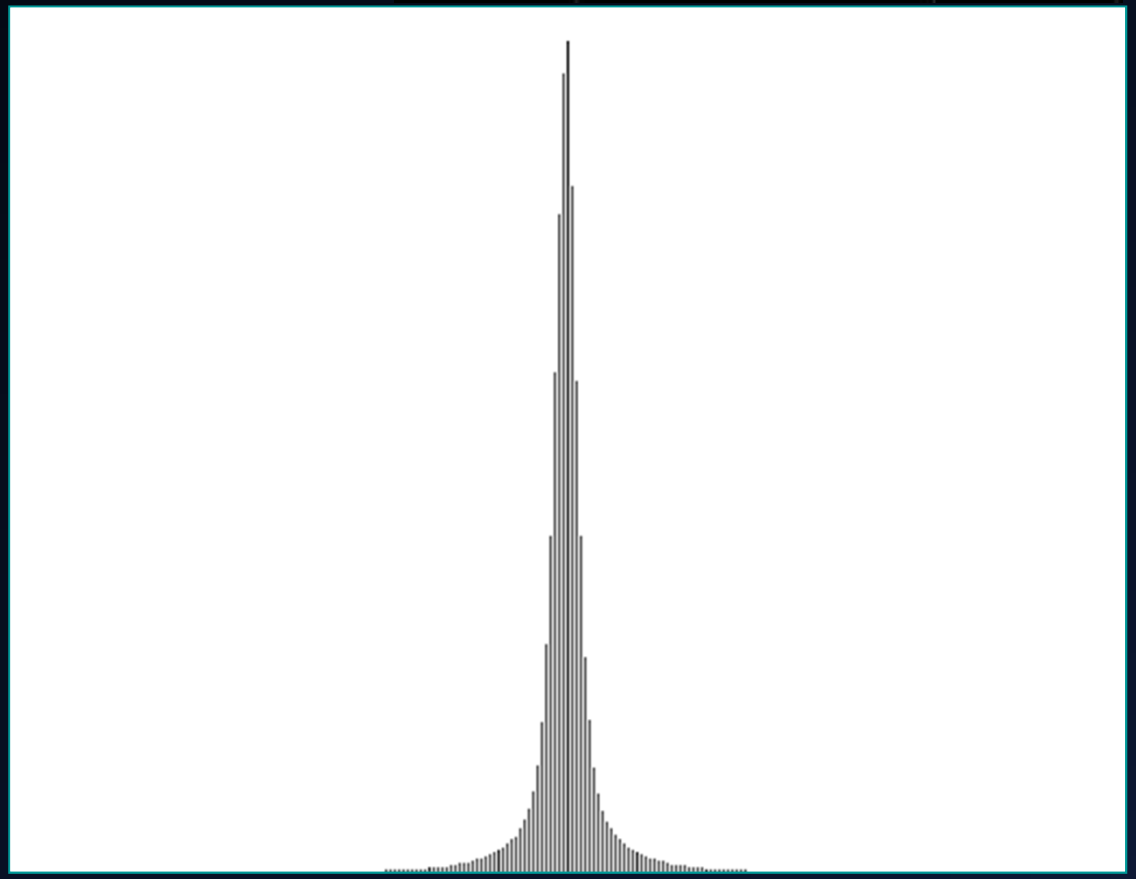
- H=6.68248



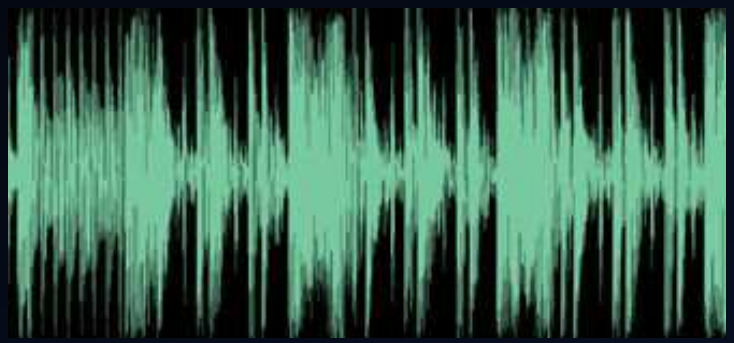


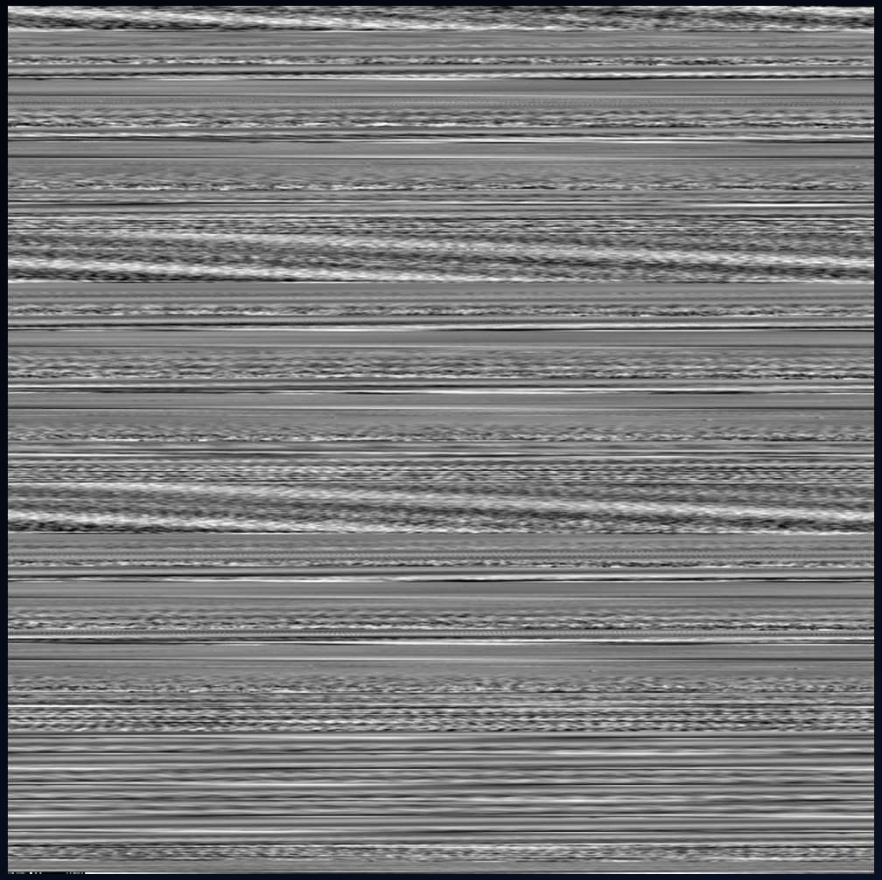
8-Bit Wave (Speech)

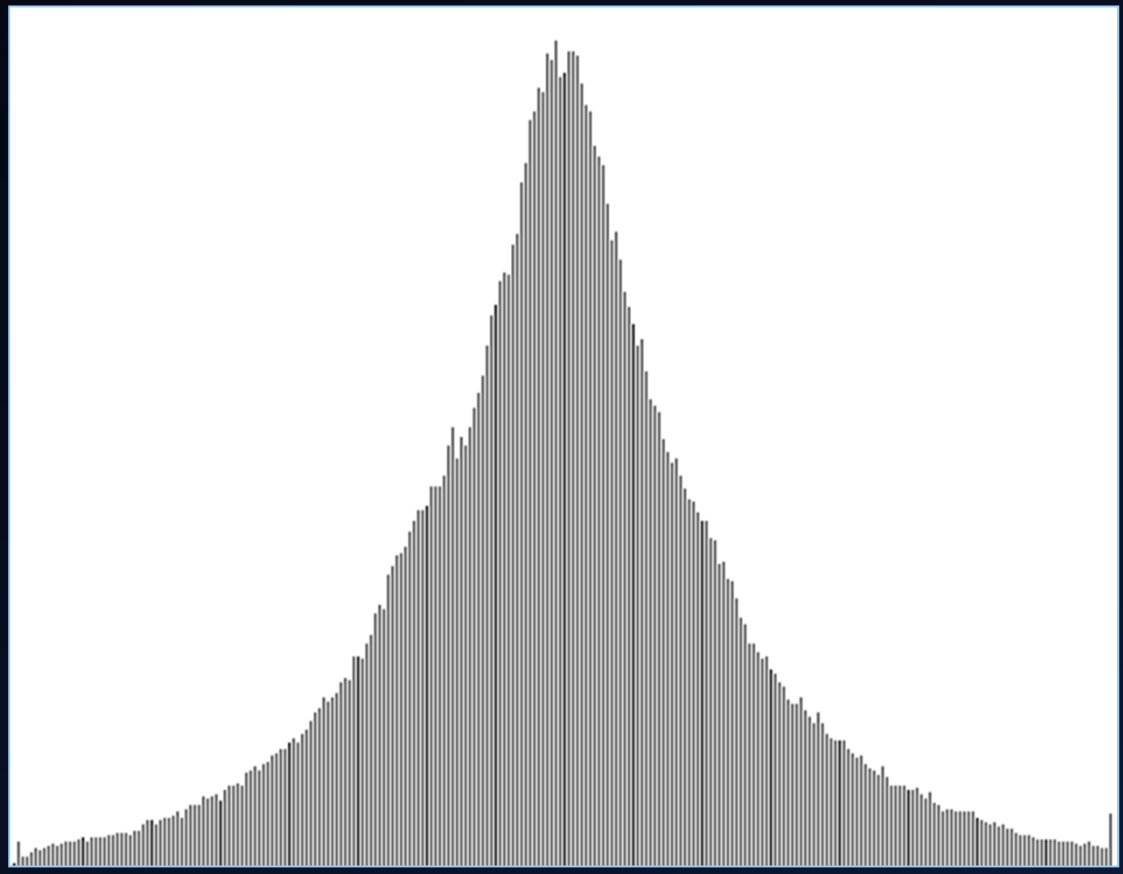




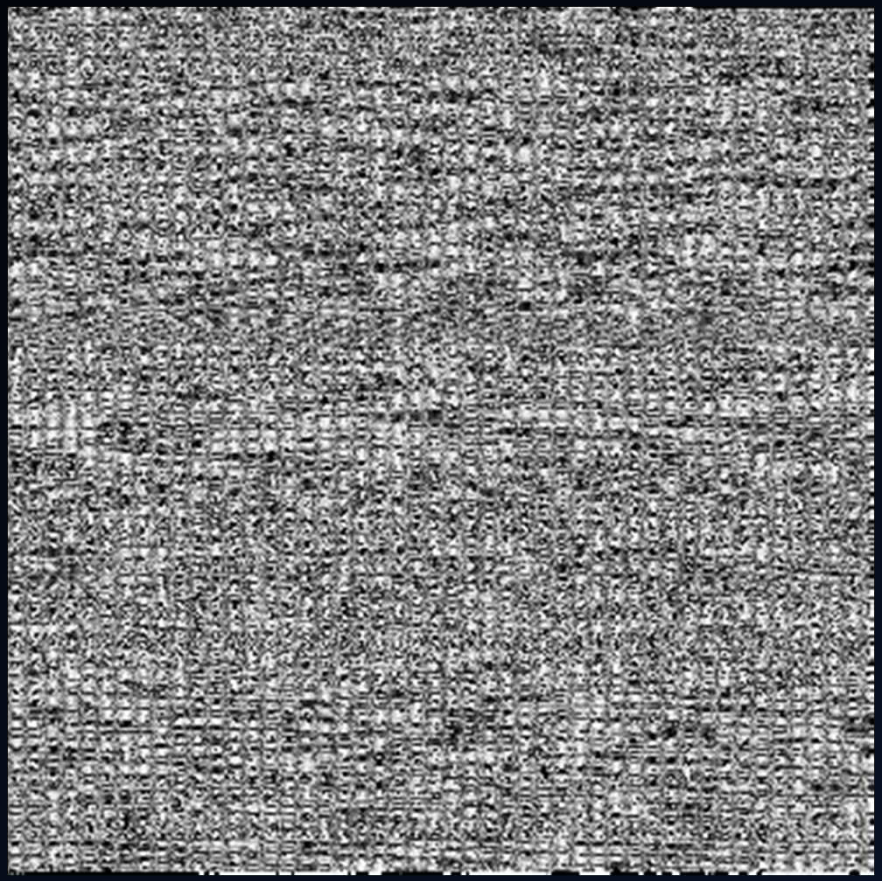
8-Bit Wave (Music)

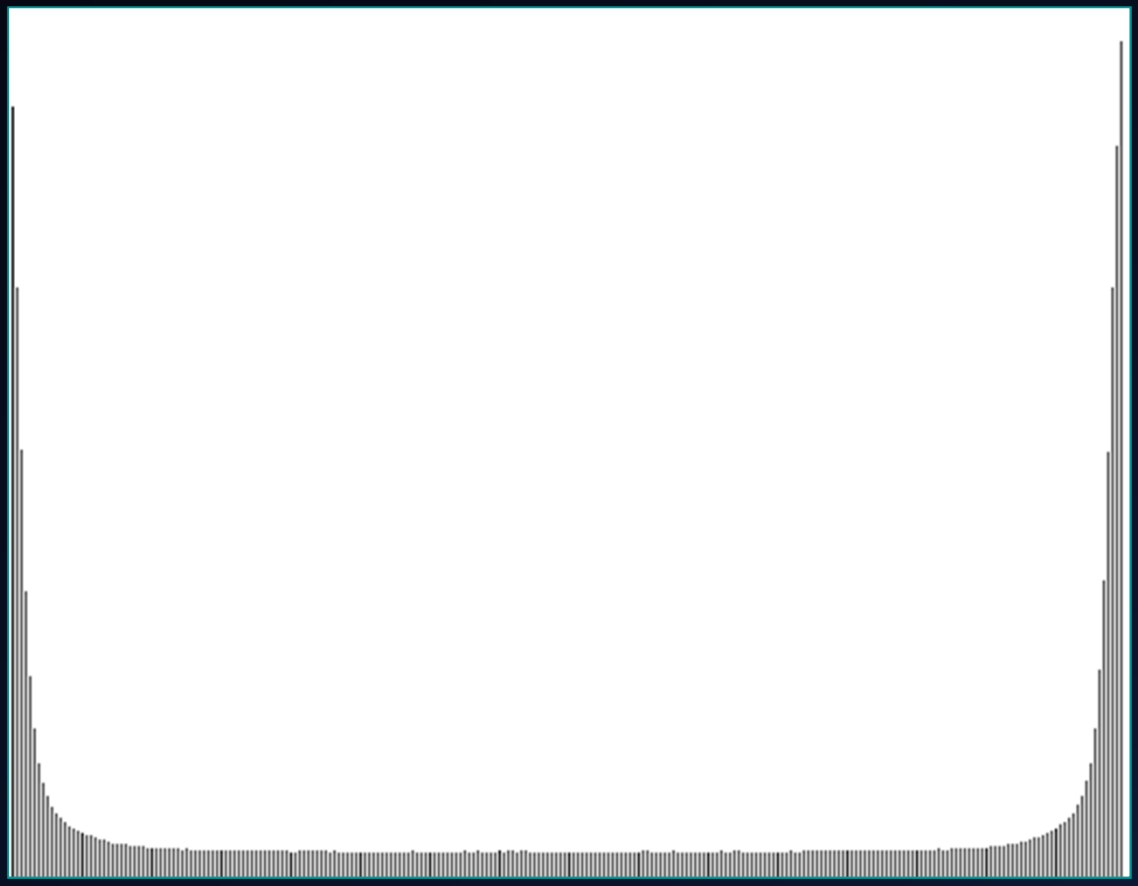




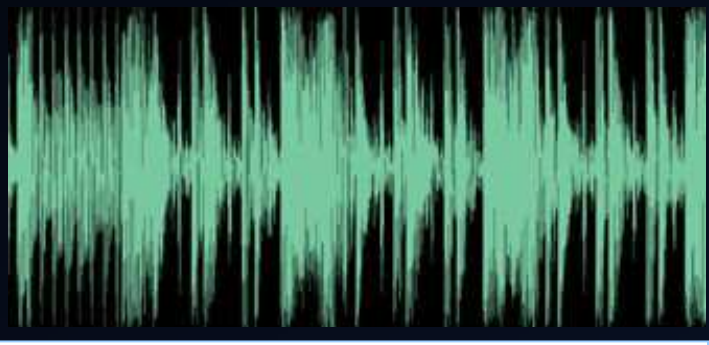


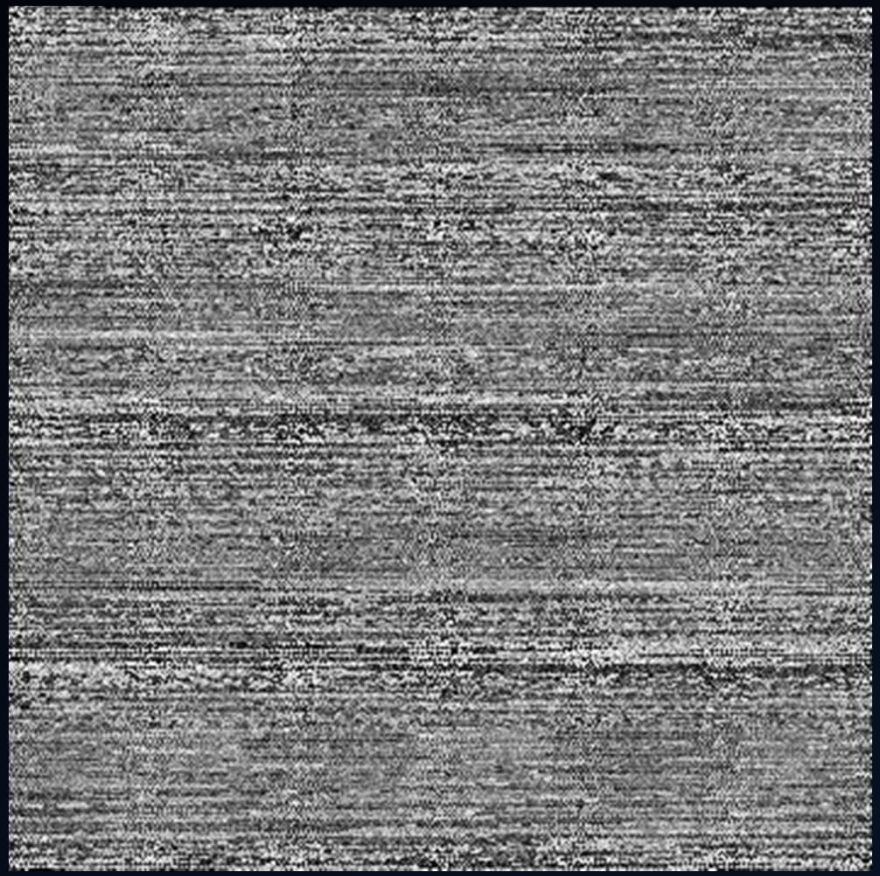
16-Bit Wave (Speech)

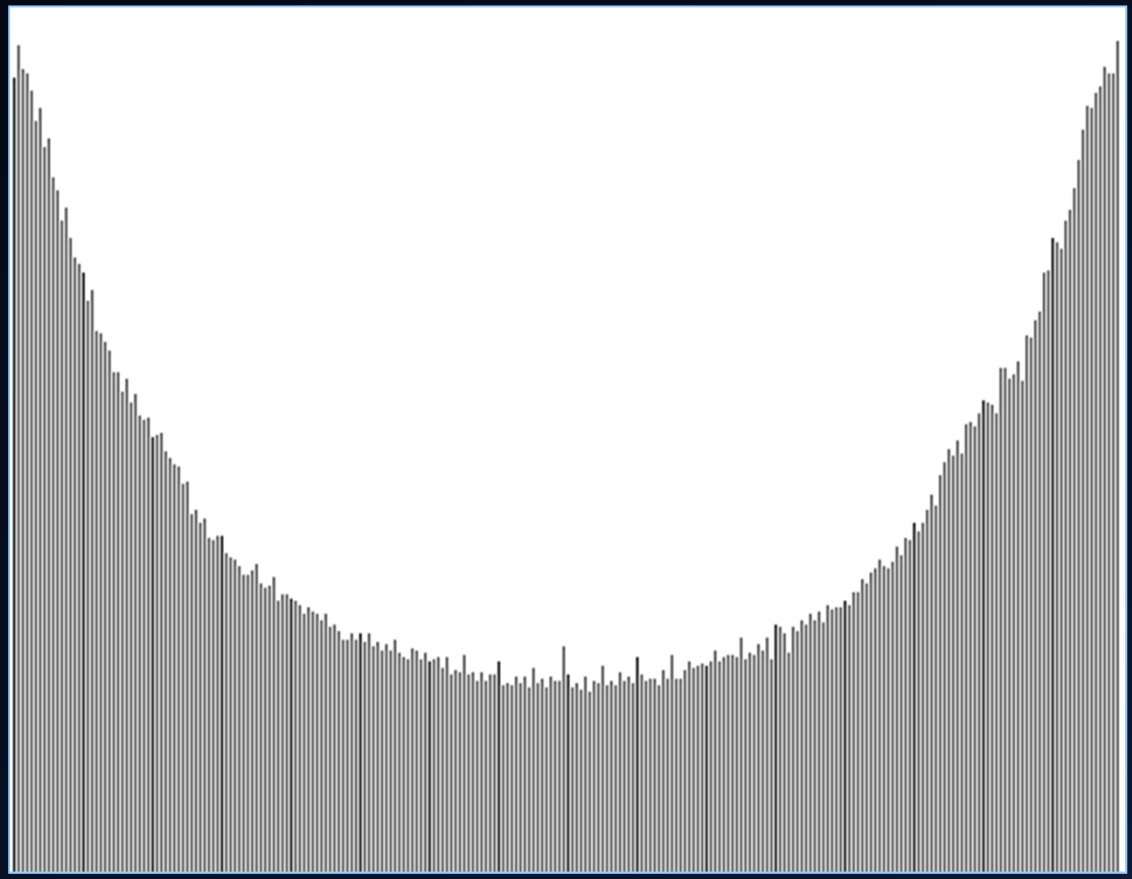




16-Bit Wave (Music)

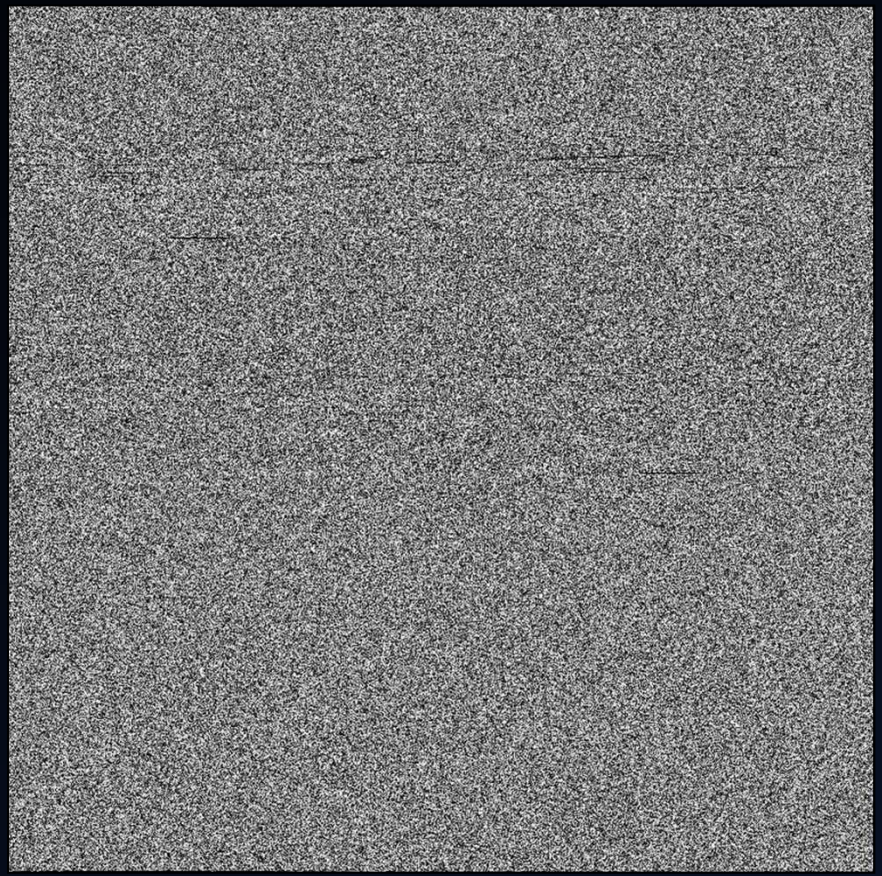


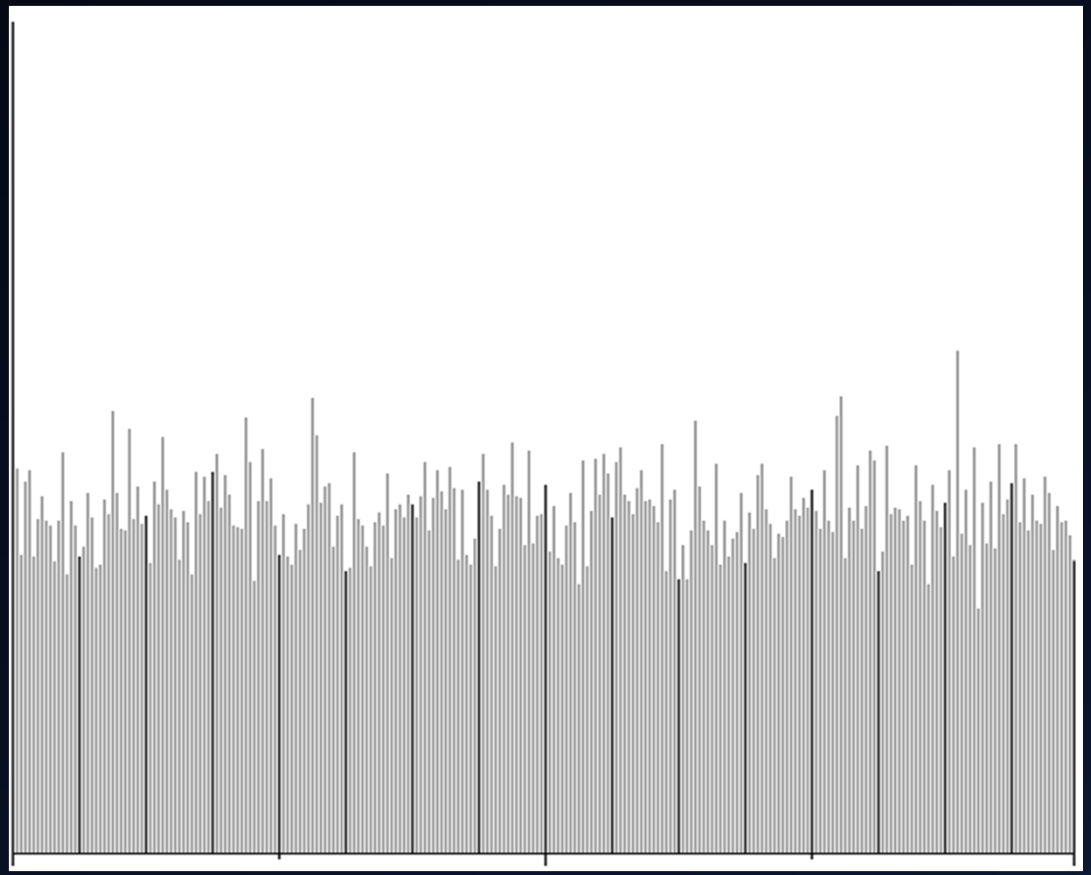




Jpeg

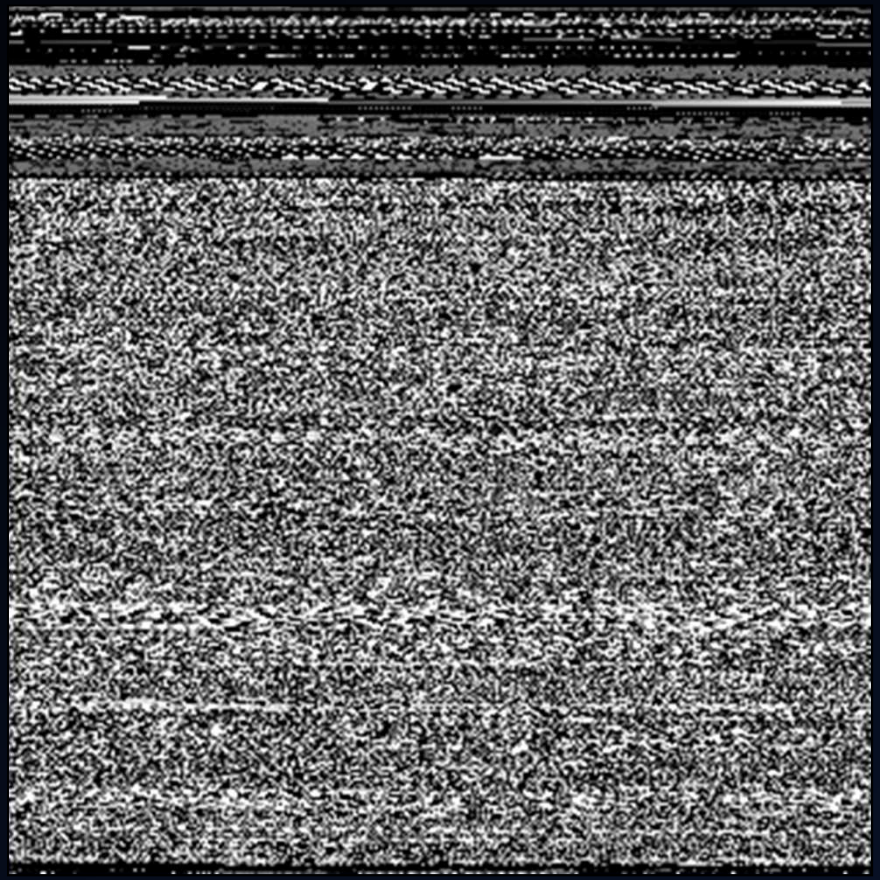
- H=7.98698

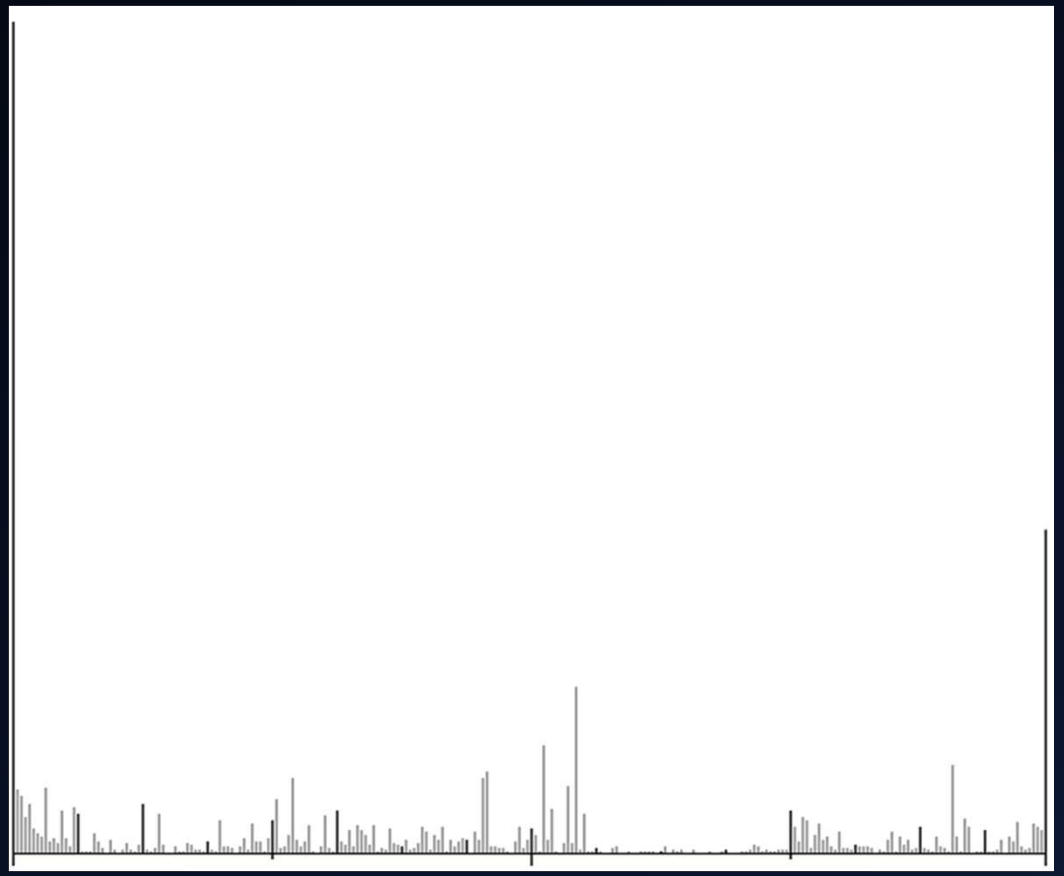




Portable Executable (PE)

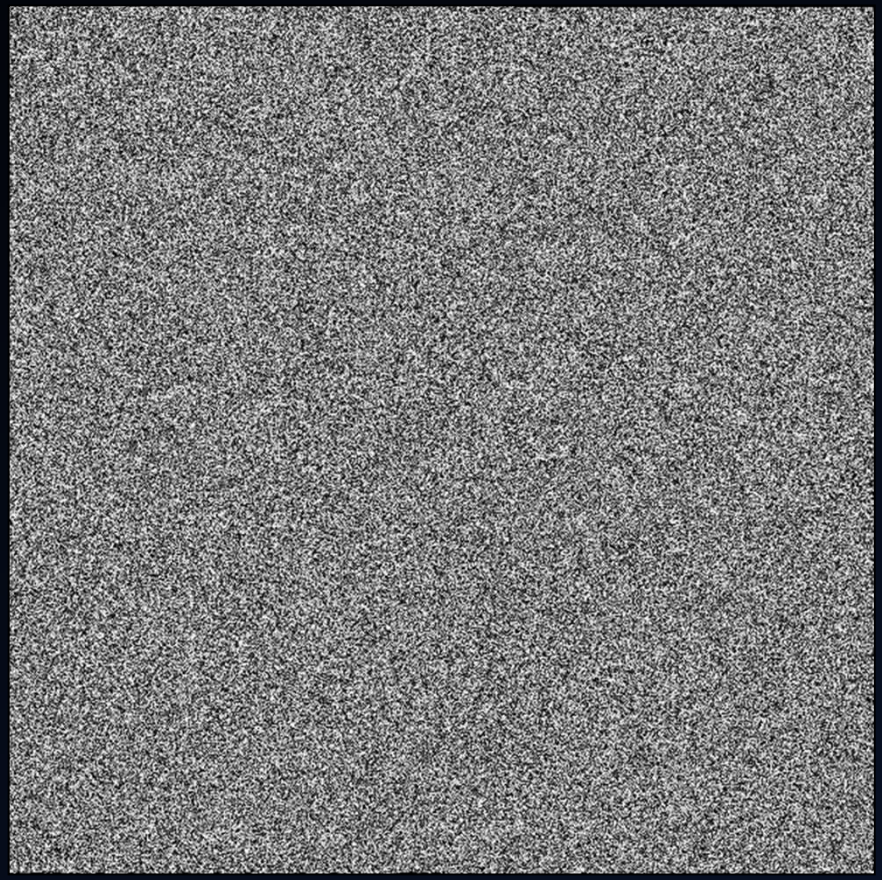
- H=6.58289

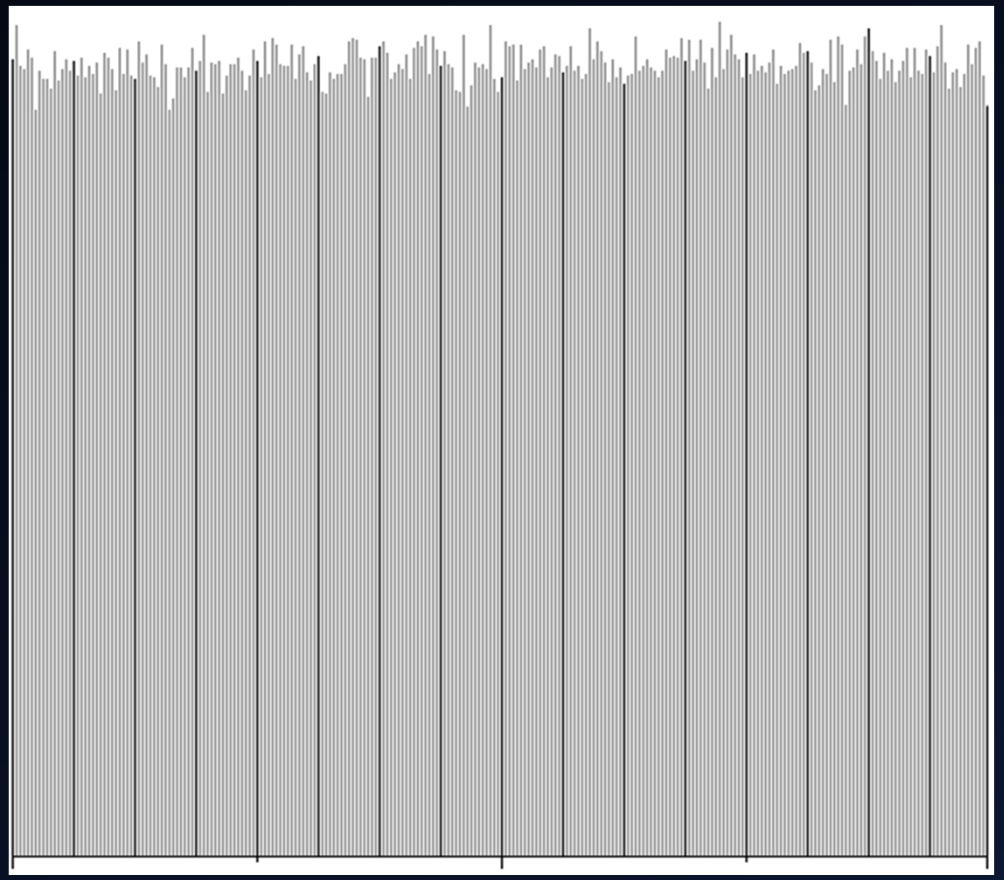




Encrypted with AES using AxCrypt

- H=7.99968



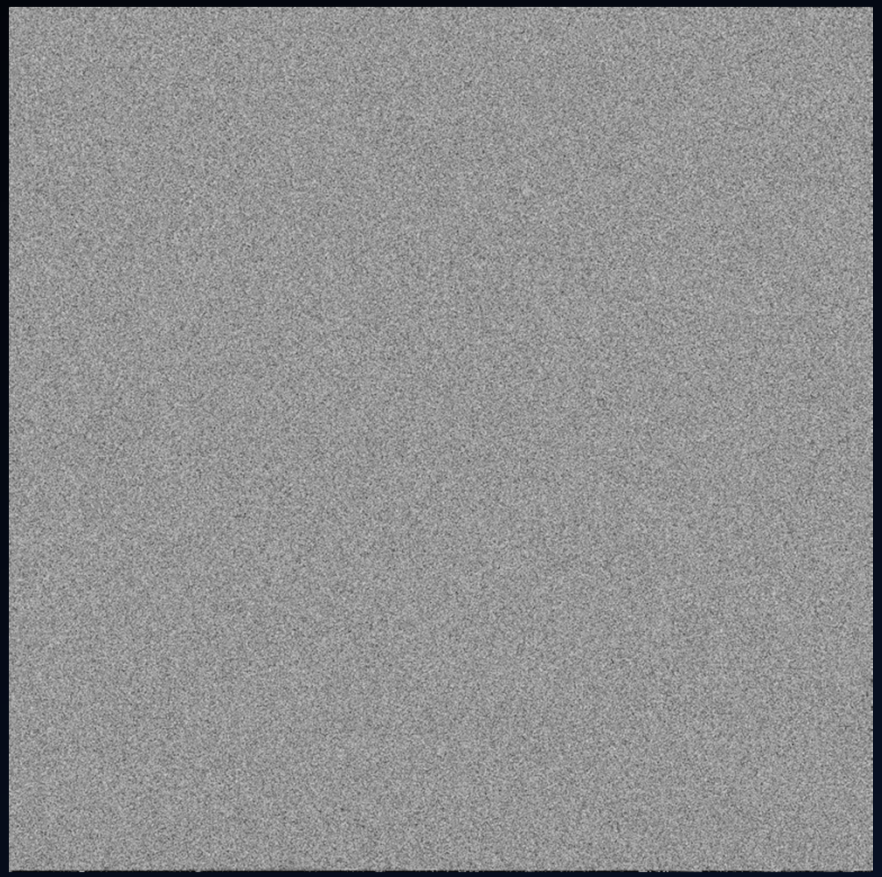


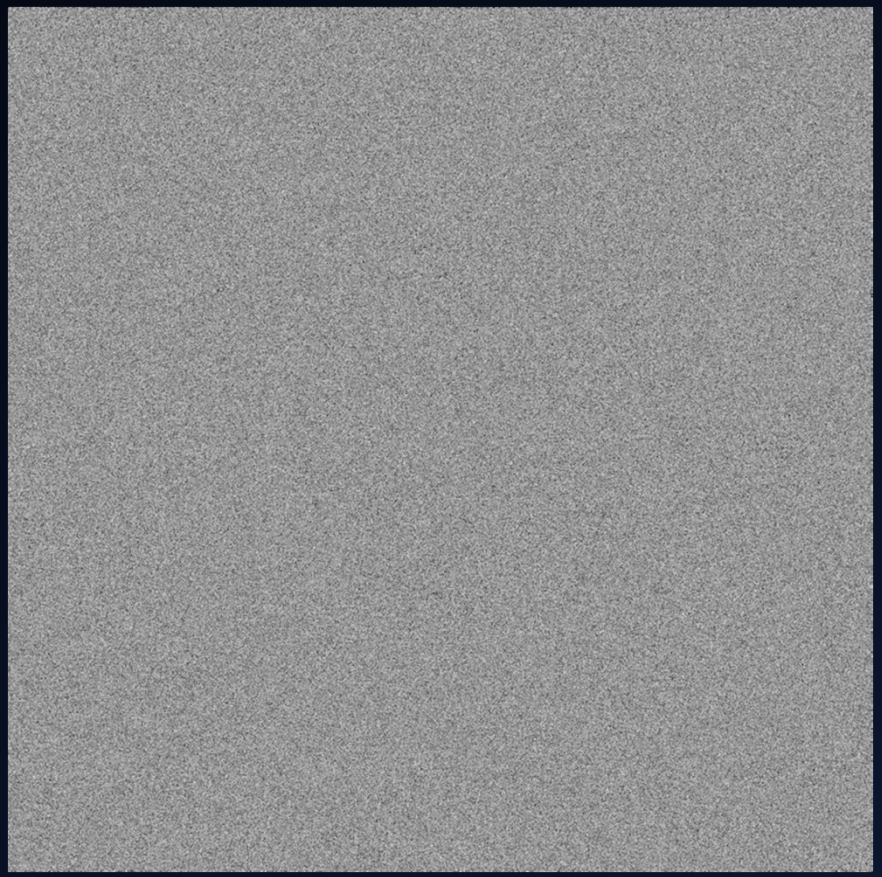
FILE TYPE IDENTIFICATION

- Knowing the characteristics of various file types is critical to identifying them
- Now we'll use the tools to

Compressed or Encrypted?

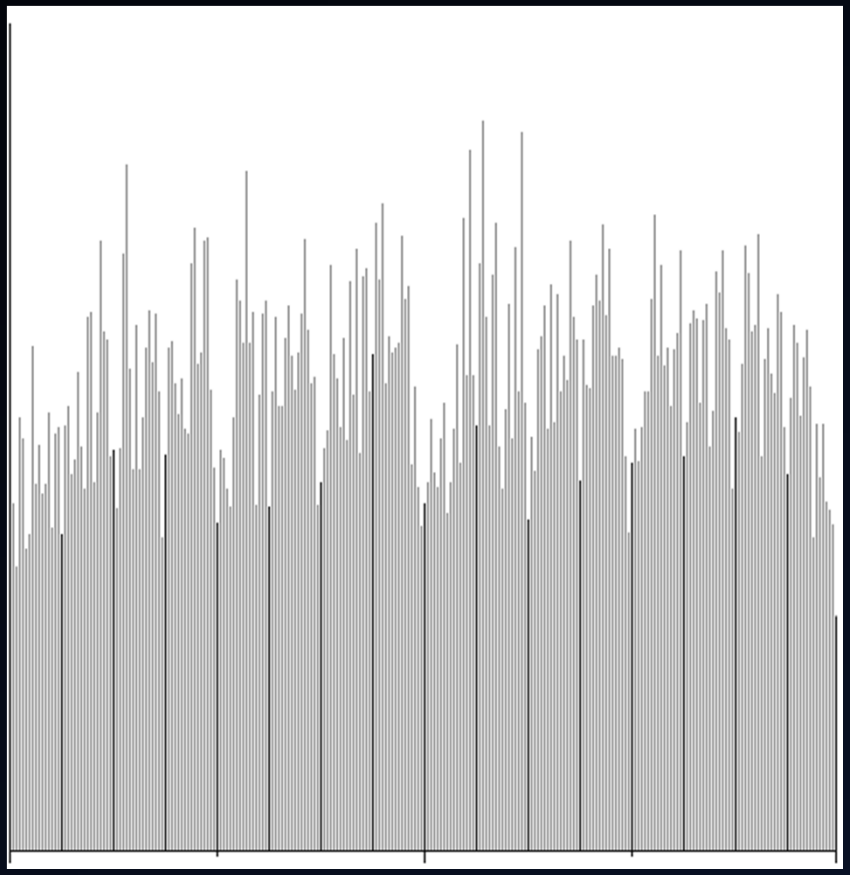
- Looking at images of the file, it's impossible to tell

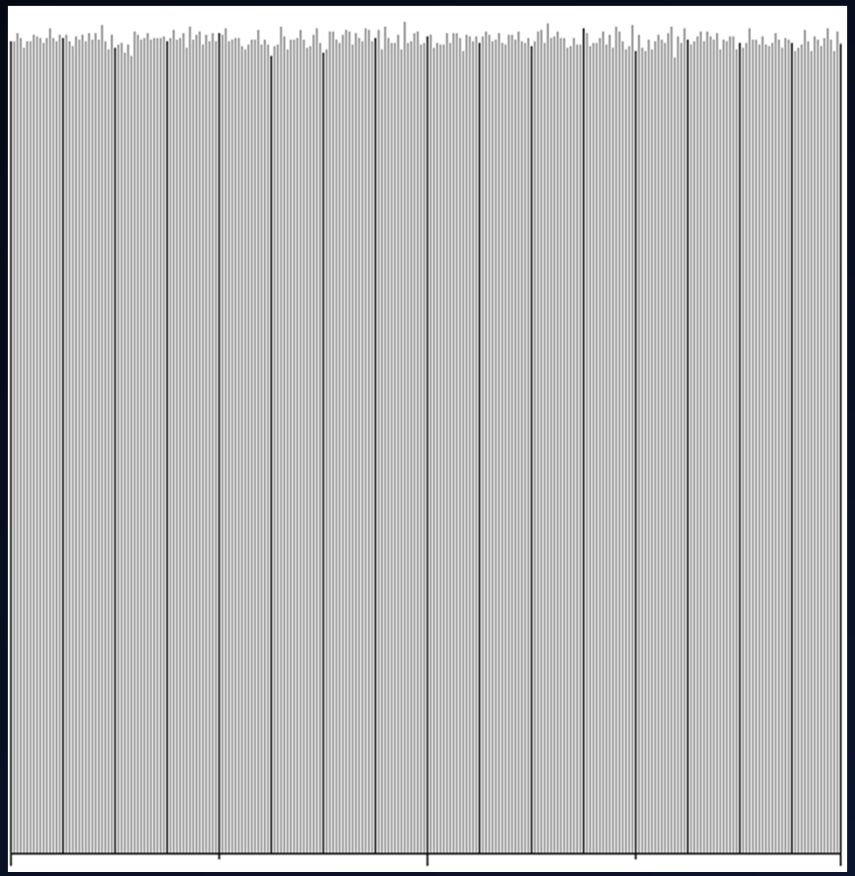




Compressed or Encrypted?

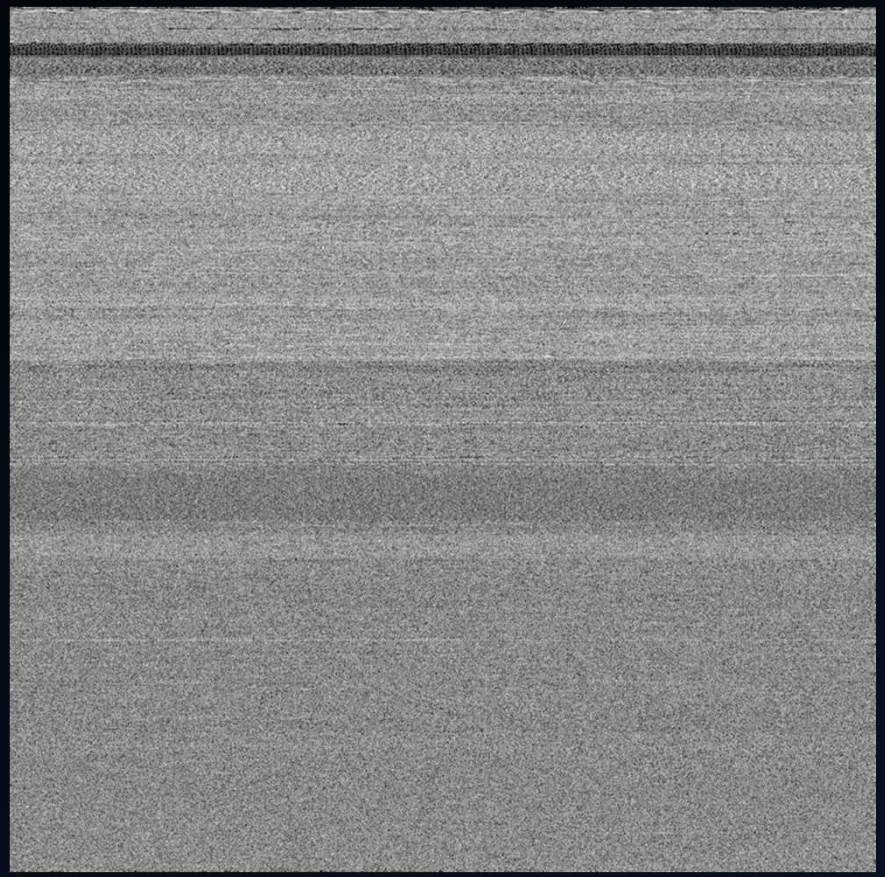
- A histogram makes it easy!

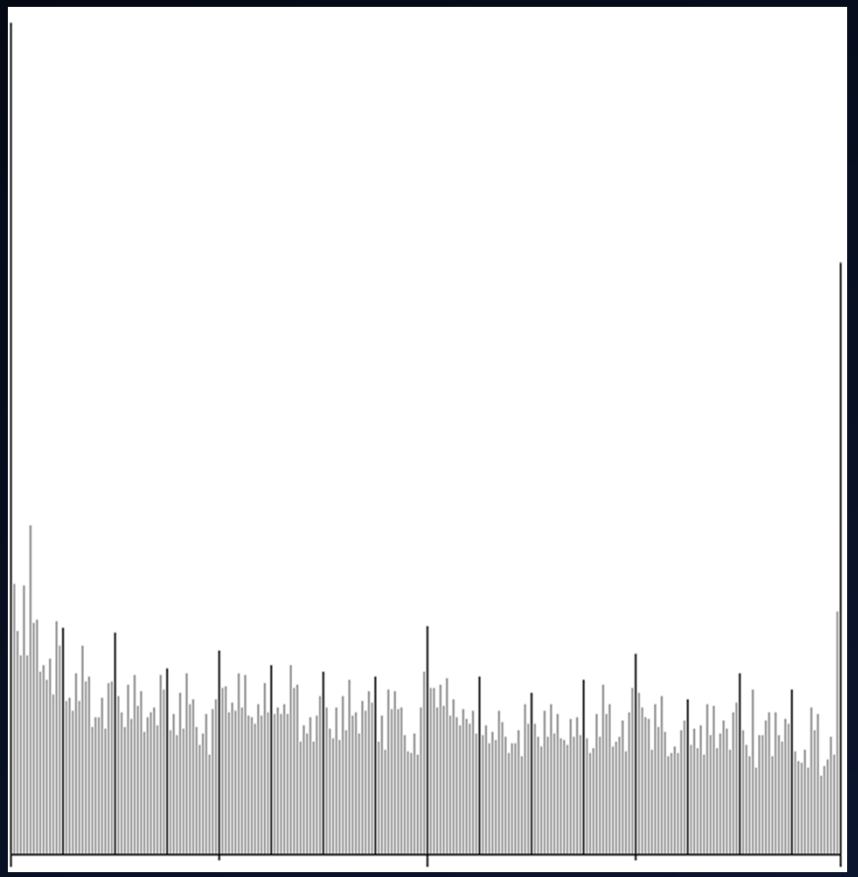




Packed or Not Packed?

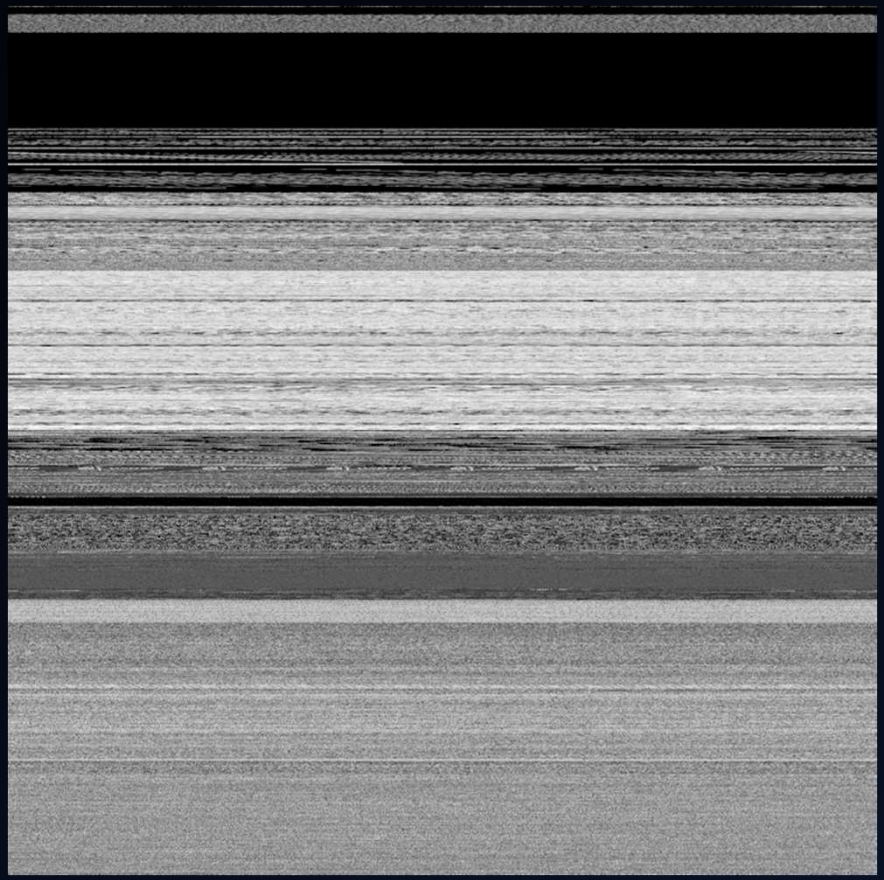
- WinZip32.exe

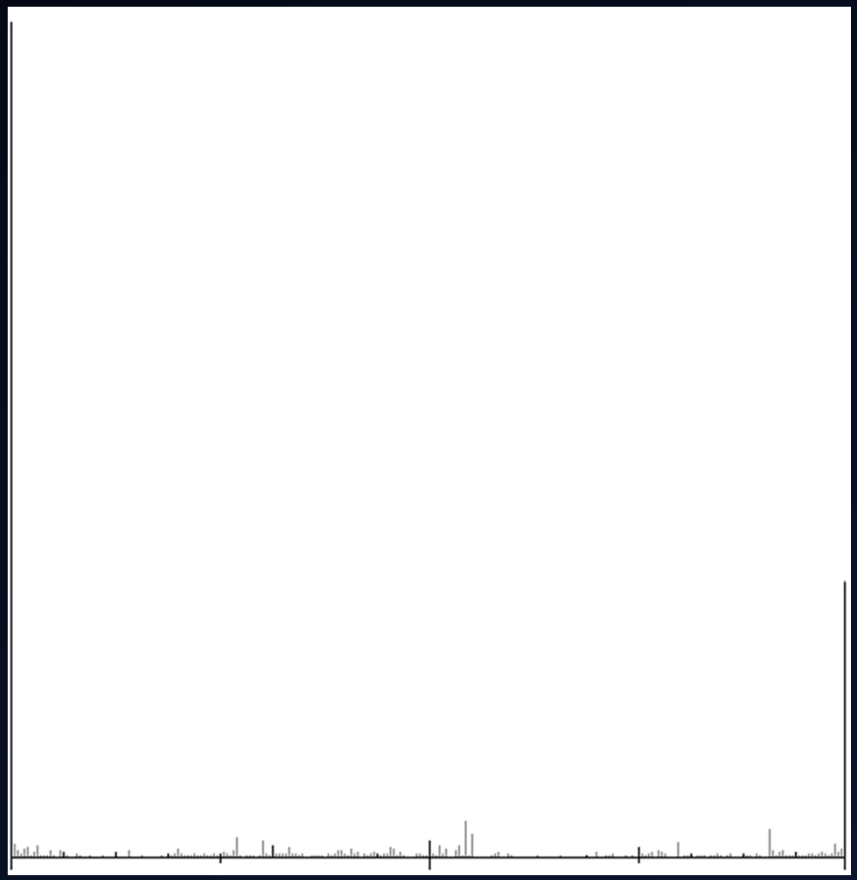




Packed or Not Packed?

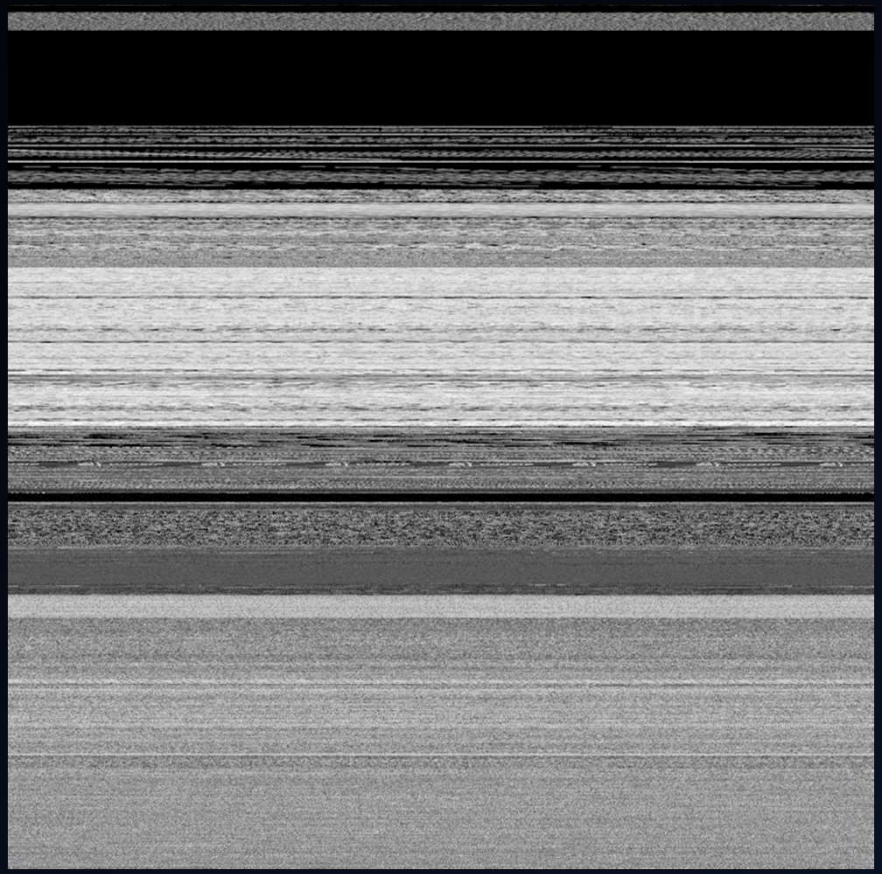
- WinZip32.exe – Histogram shows LARGE number of Zeros

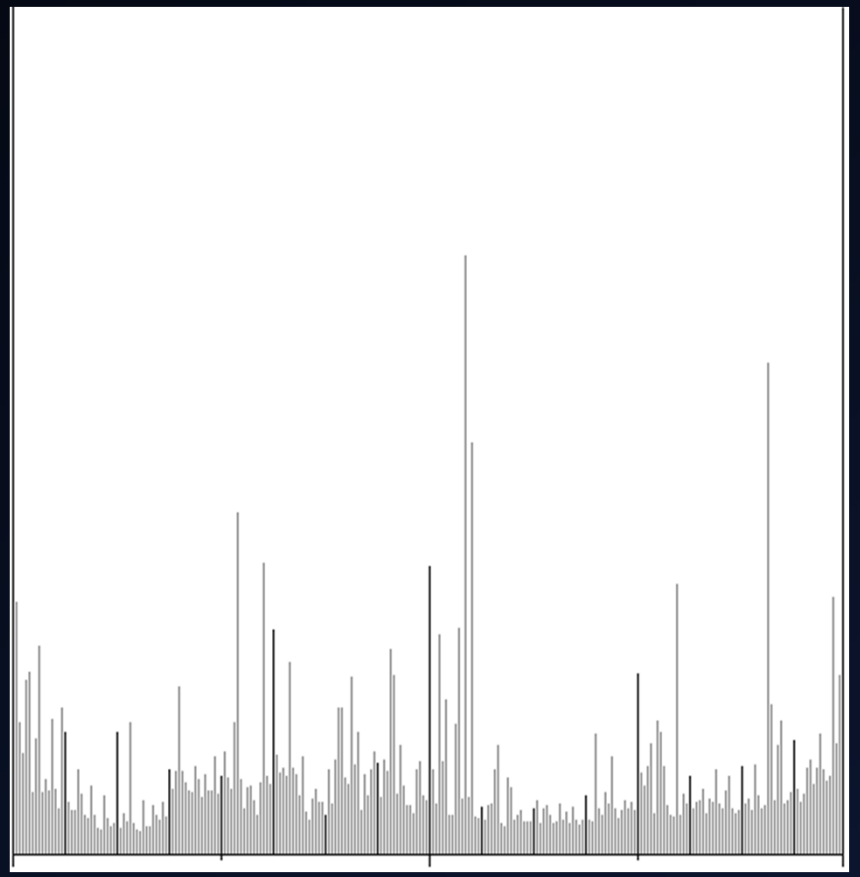




Packed or Not Packed?

- WinZip32.exe – Zoomed in on Histogram

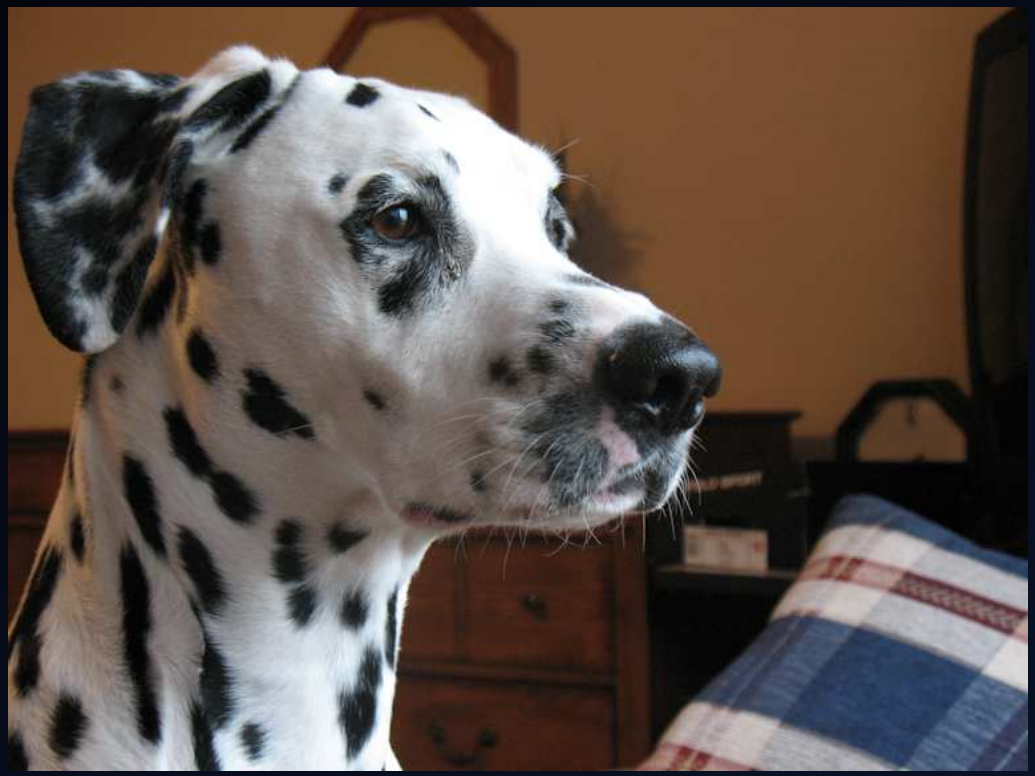


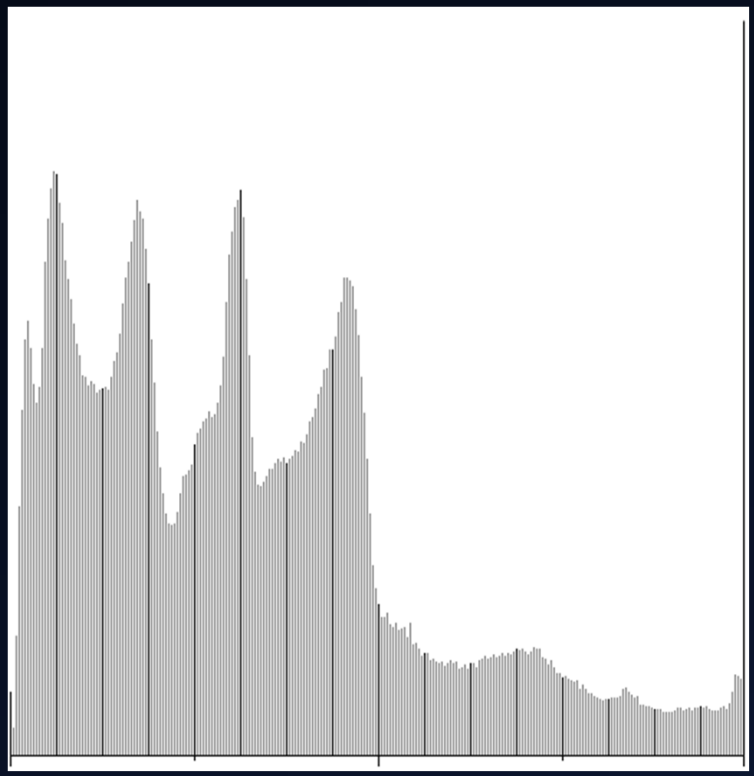


Are You Hiding Something?

- Sometimes histograms and entropy are less effective
- Original Image

H= 7.61037

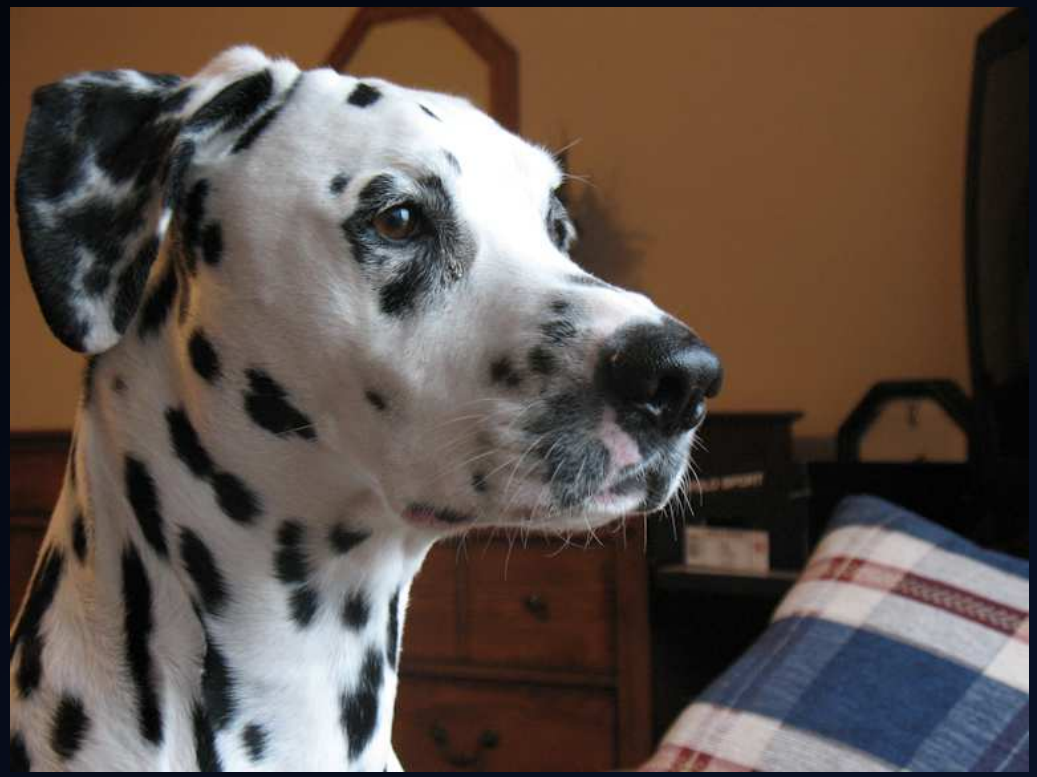


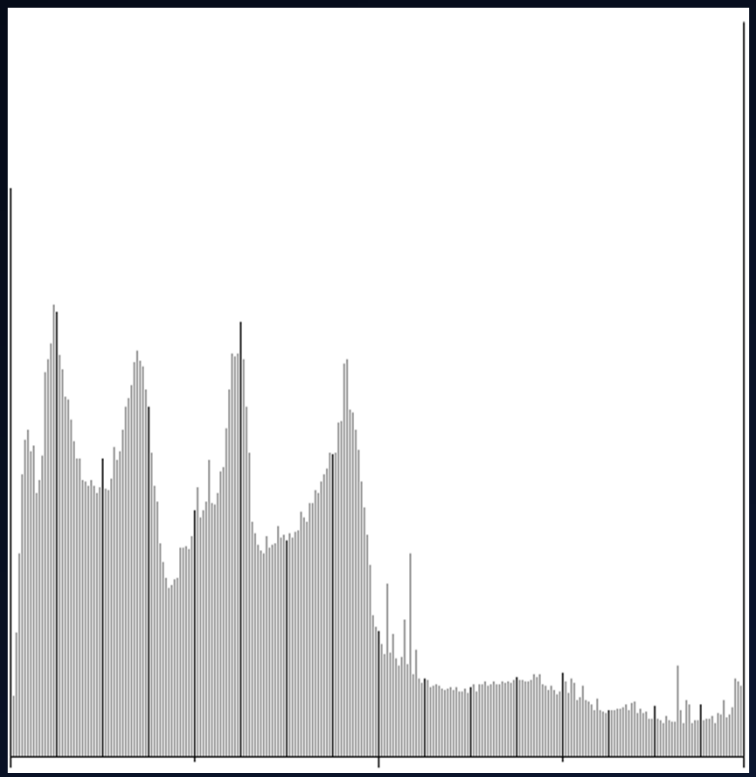


Are You Hiding Something?

- Data appended to end of file - not easily noticed in statistics
- Small aberration in histogram, no entropy indication

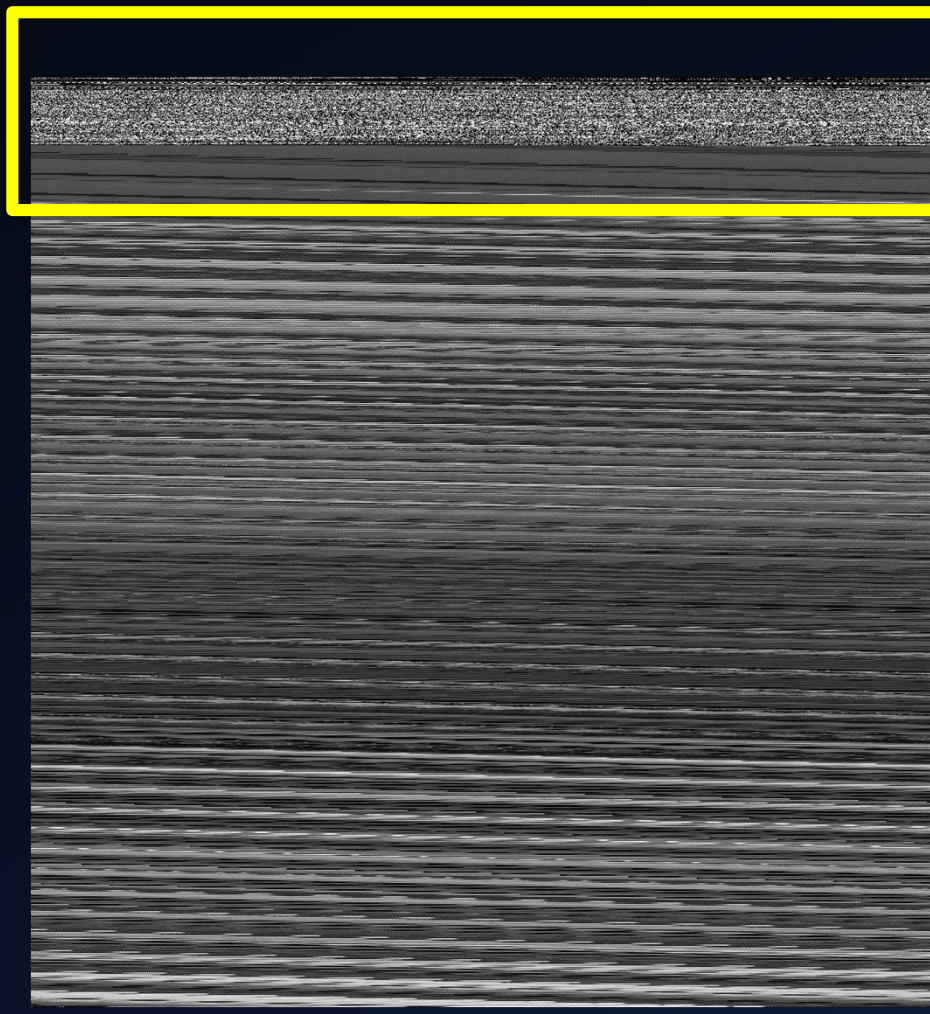
H= 7.63532





Are You Hiding Something?

- Image of the file reveals appended data at end
  - Remember, bitmaps start from bottom up
- Entropy of original image already fairly high
  - The larger the appended data, the more its entropy characteristics show



Using Steganography?

- LSB Steganography hides data in the Least Significant Bit(s) of an image
- Very difficult to see if number of bits < 4
- Often times difficult using 4 bits
- At 5 bits, the hidden data begins to be very noticeable
- Can we detect the alteration of the lower bits???

Using Steganography?

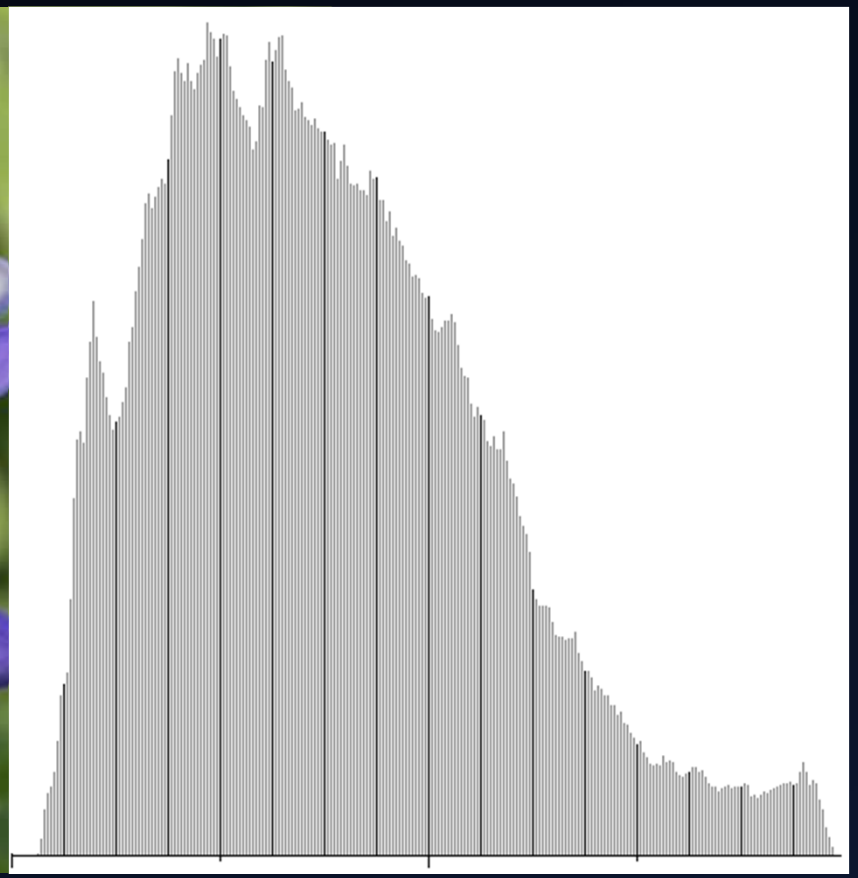
- LSB Steganography hides data in the Least Significant Bit(s) of an image
- Very difficult to see if number of bits < 4
- Often times difficult using 4 bits
- At 5 bits, the hidden data begins to be very noticeable
- Can we detect the alteration of the lower bits???

- Duh. Why ELSE would I bring it up?

Using Steganography?

- Original, zero bits altered

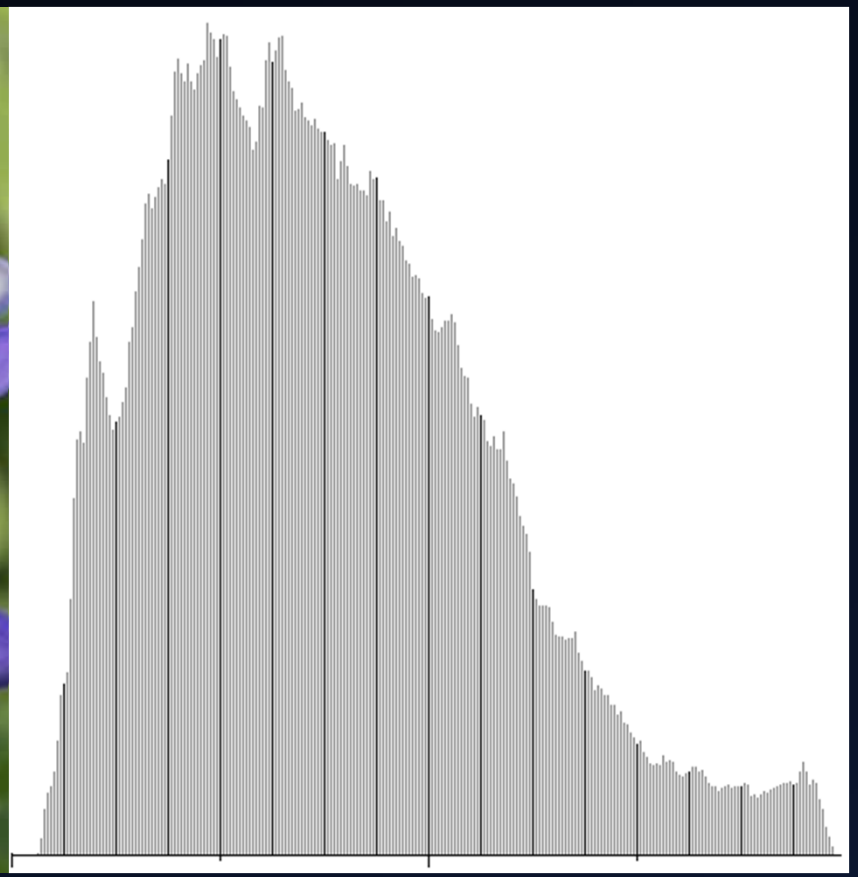
H= 7.55730



Using Steganography?

- 1 bit of randomized data

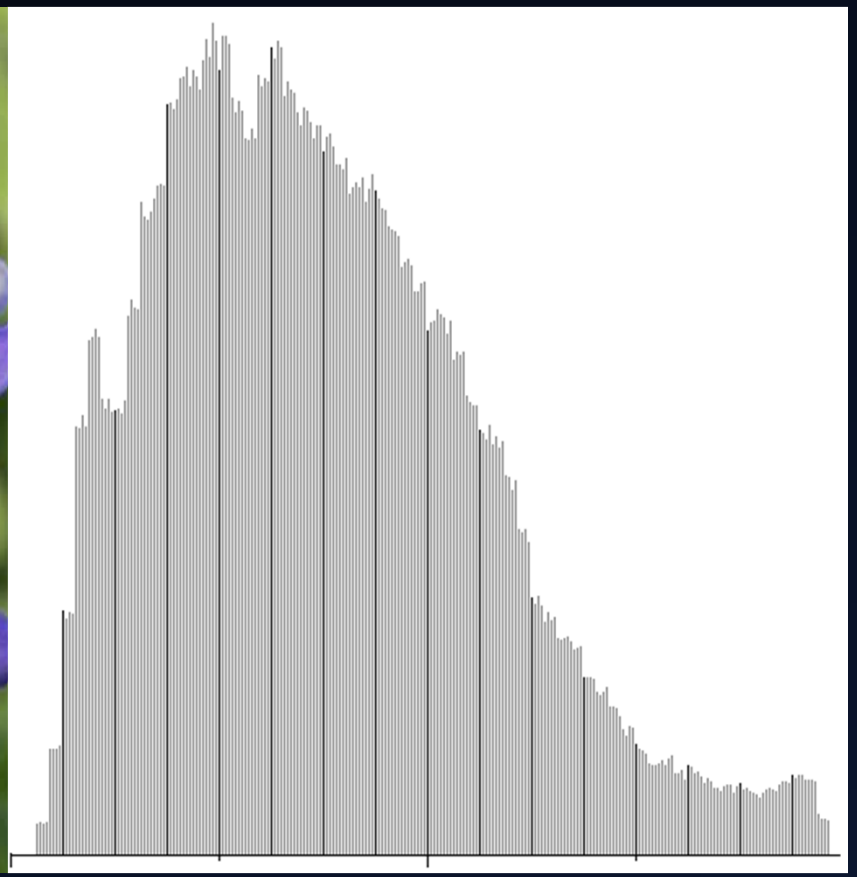
H= 7.55782



Using Steganography?

- 2 bits of randomized data

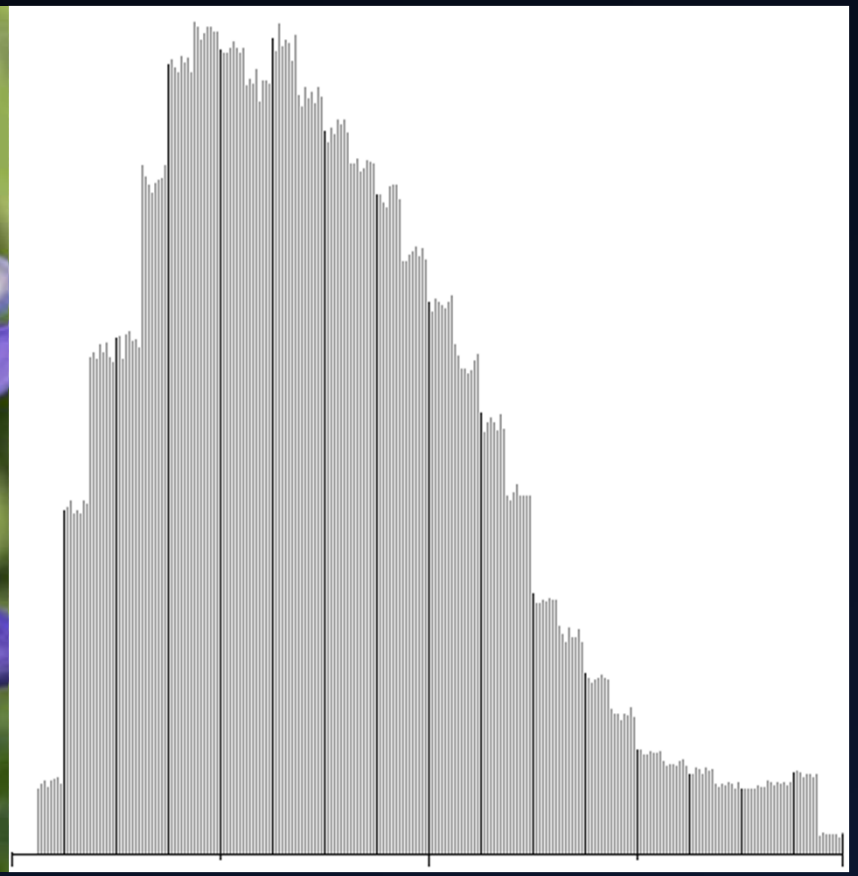
H= 7.55962



Using Steganography?

- 3 bits of randomized data

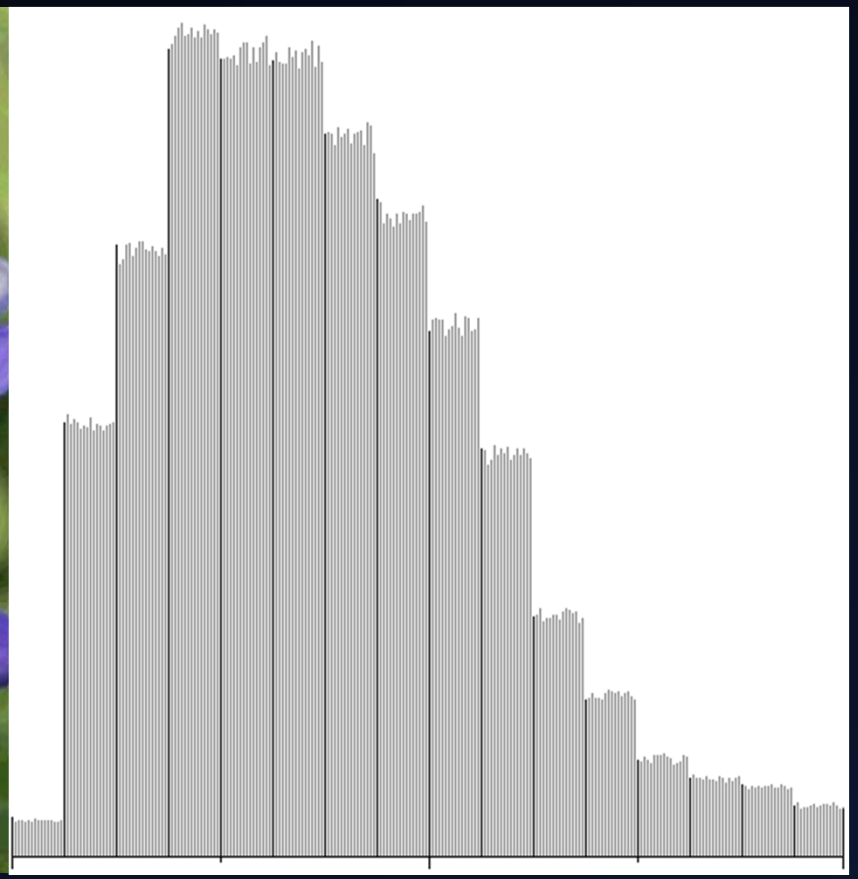
H= 7.56456



Using Steganography?

- 4 bits of randomized data

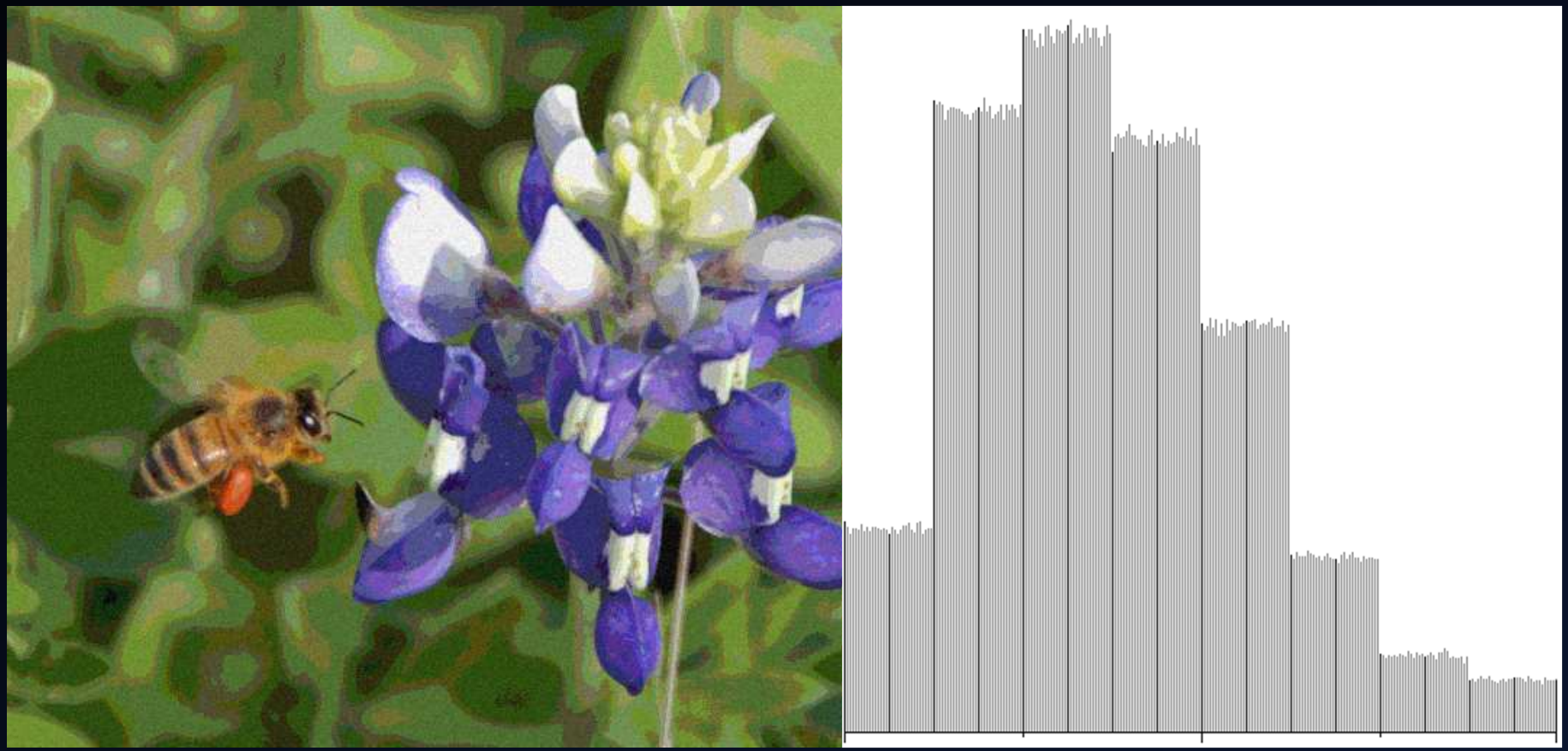
H= 7.57645



Using Steganography?

- 5 bits of randomized data

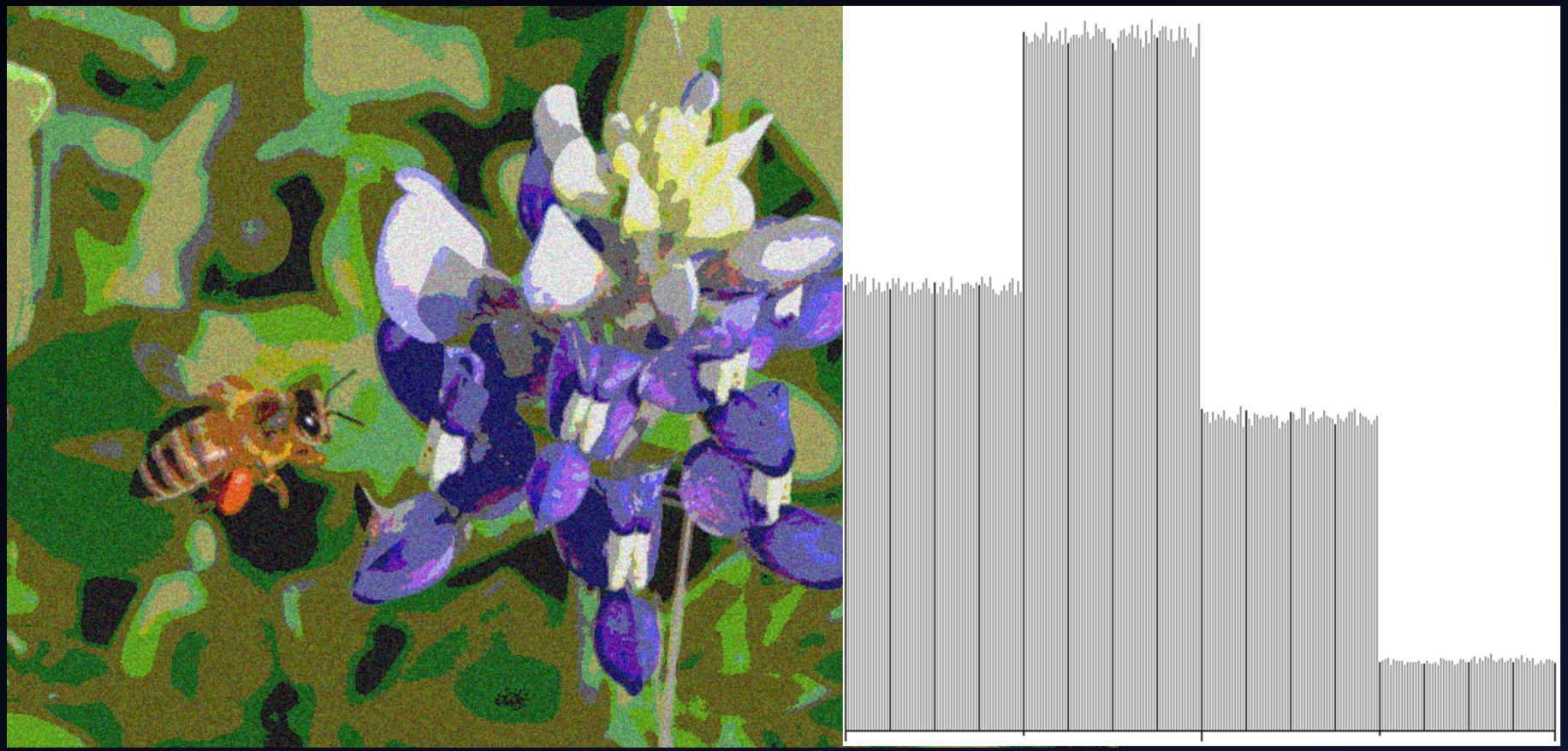
H= 7.62805



Using Steganography?

- 6 bits of randomized data

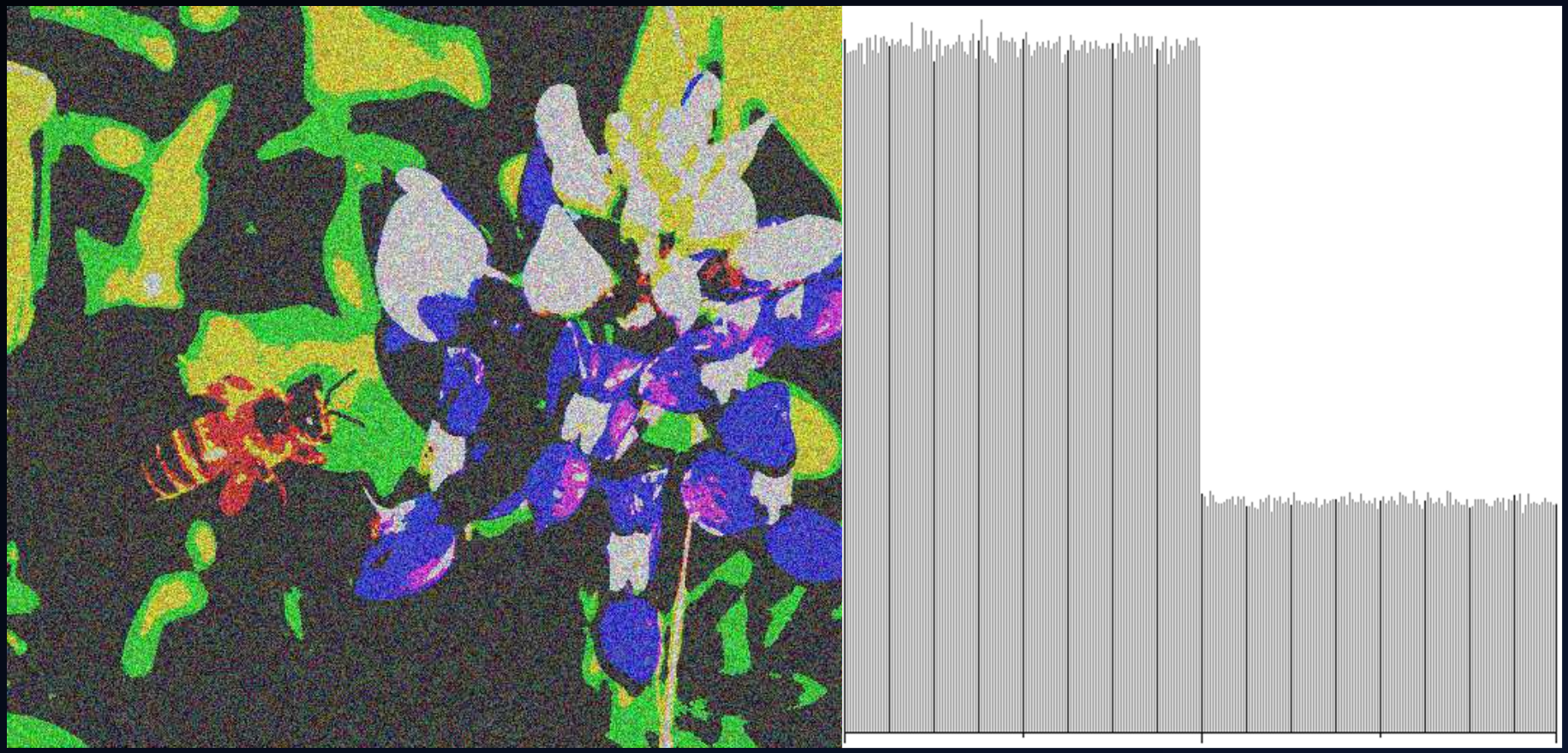
H= 7.71131



Using Steganography?

- 7 bits of randomized data

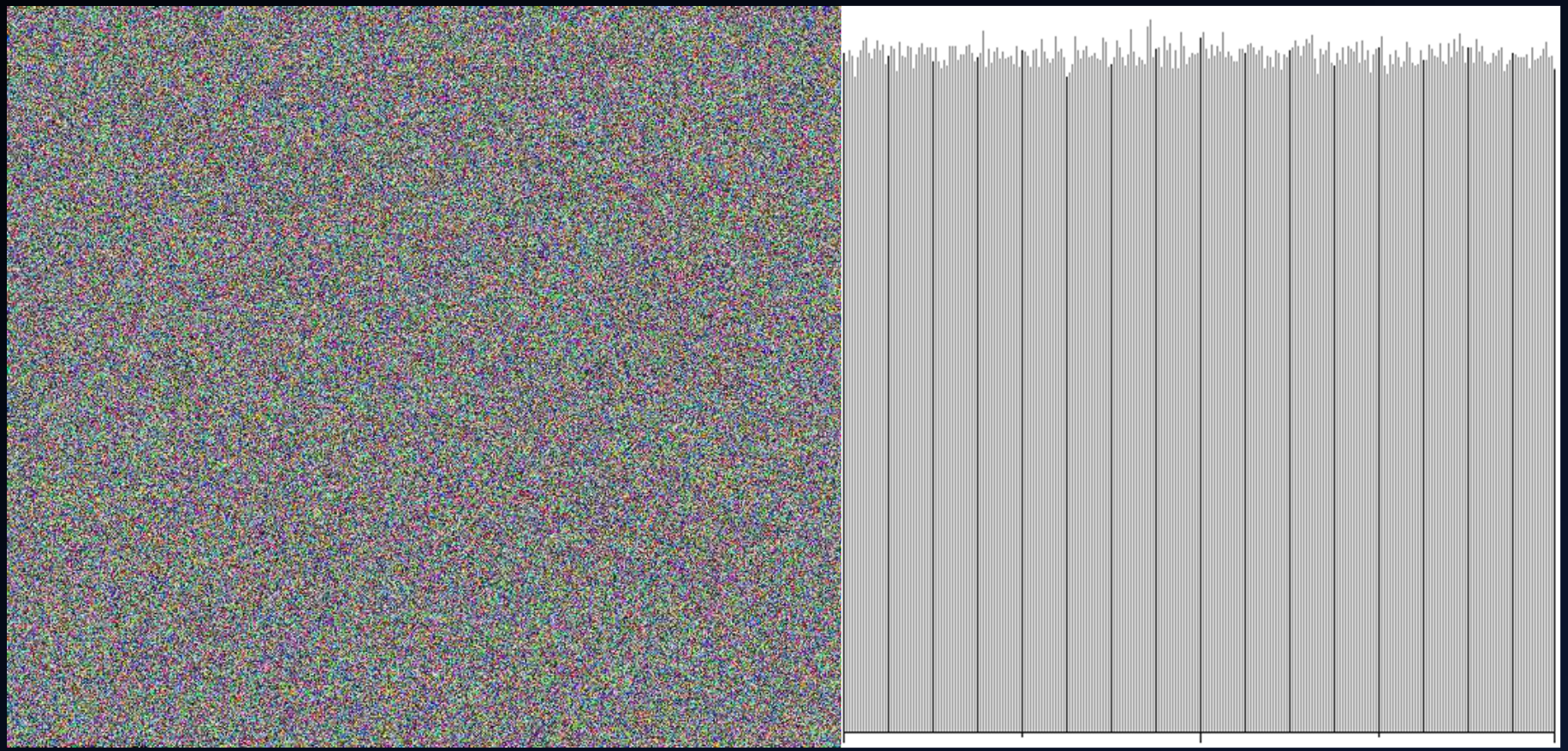
H= 7.81565



Using Steganography?

- 8 bits of randomized data

H= 7.99986



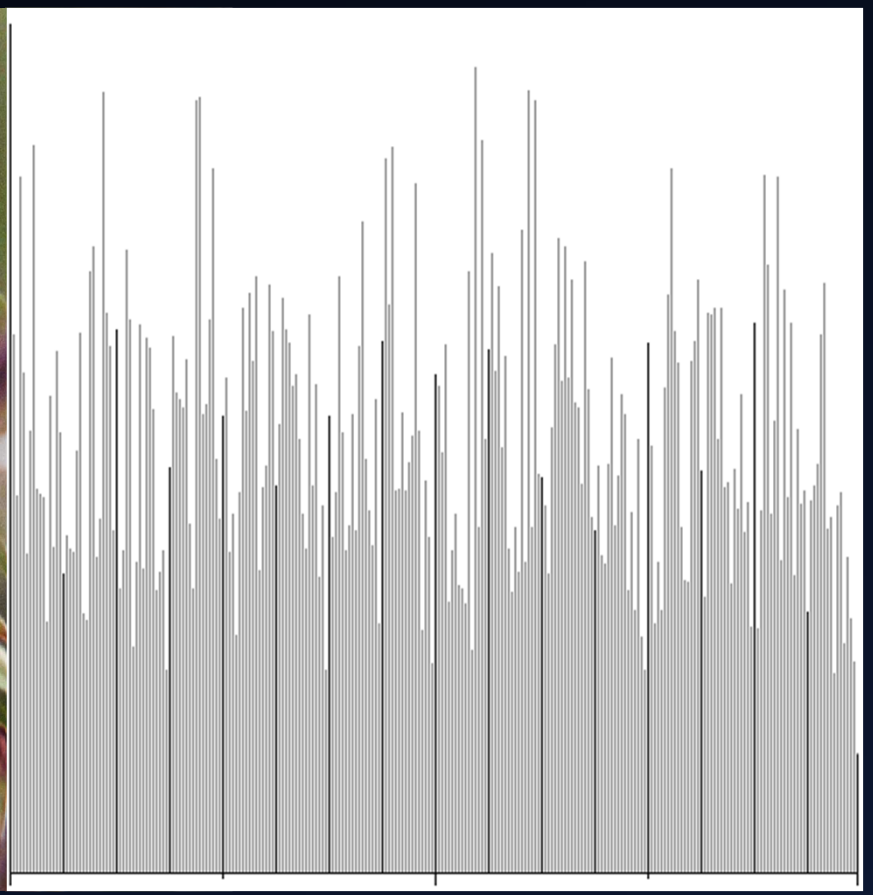
Does This Work for Jpeg?

- A jpeg is a compressed file, so any images of the file, histograms, or entropy will show the characteristics of compression
- The technique works on the decompressed components

Does This Work for Jpeg?

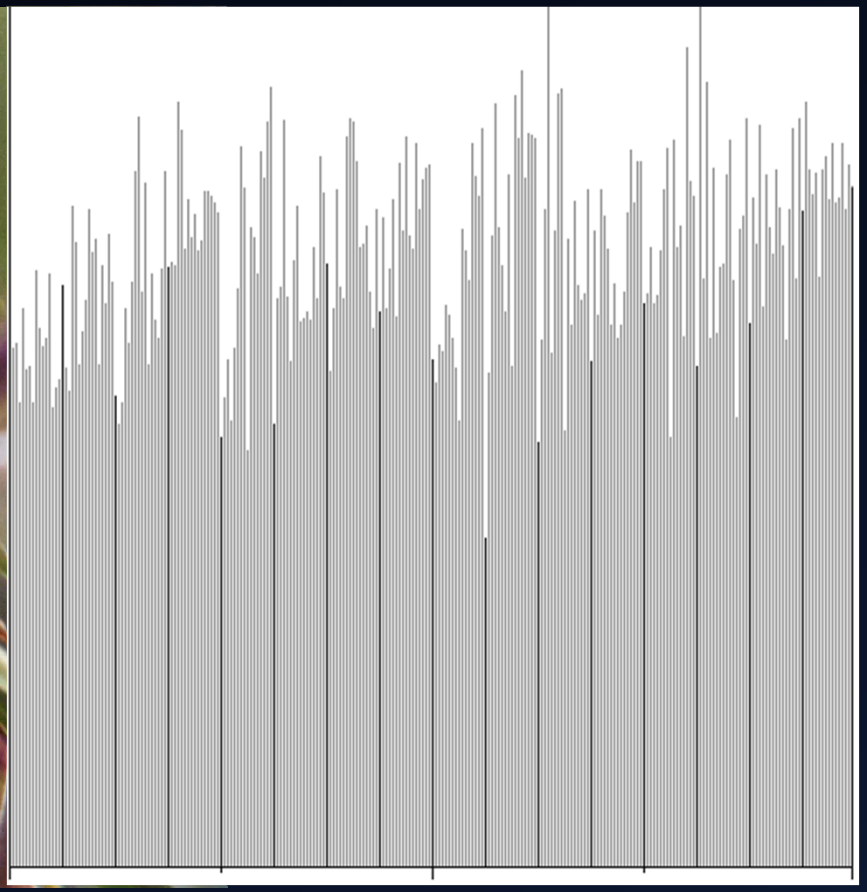
- Original image and its histogram

H= 7.93



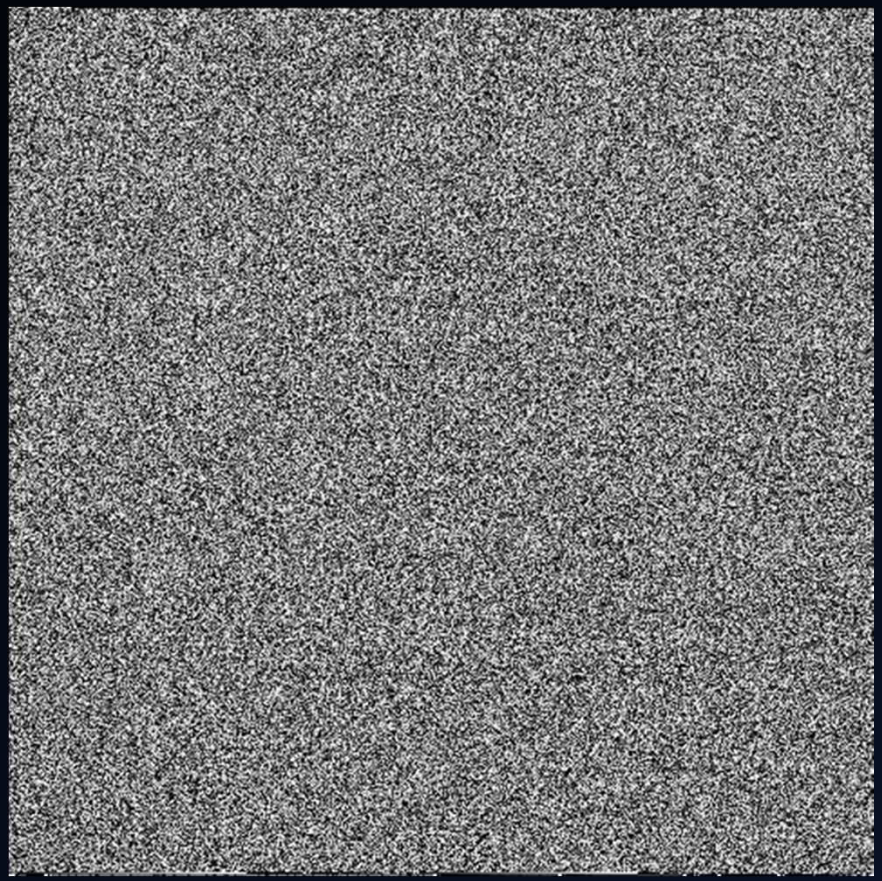
Does This Work for Jpeg?

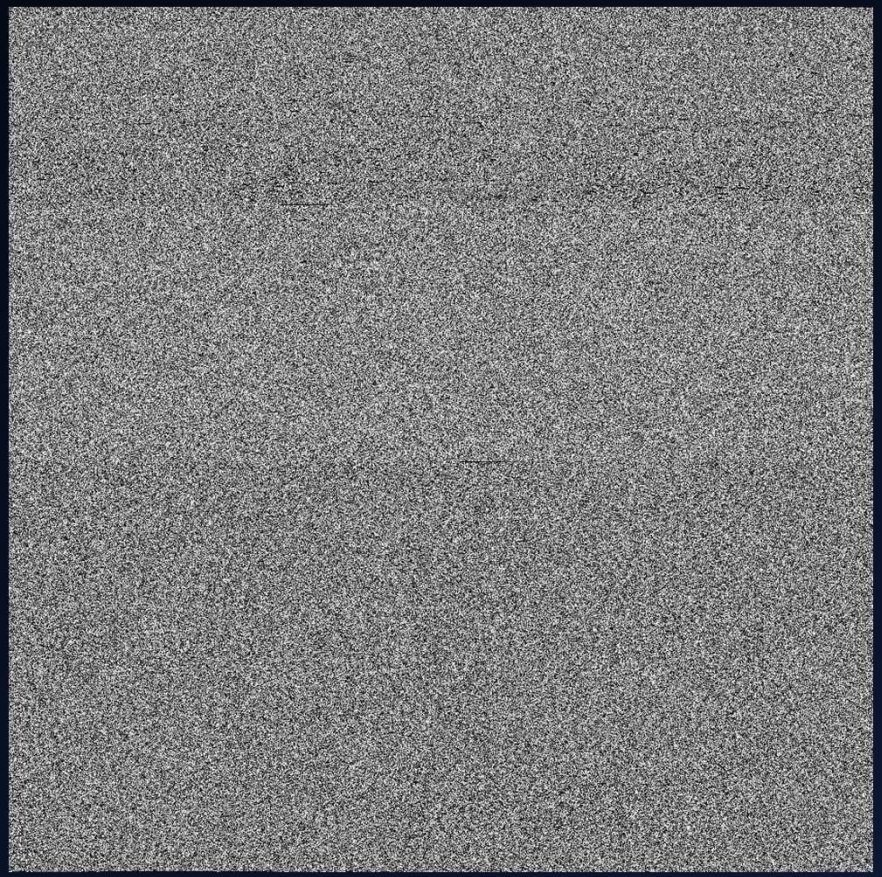
- Stego Image: 146,256 bytes of hidden data out of 967,442 H= 7.97



How About Using an Image of the Jpeg?

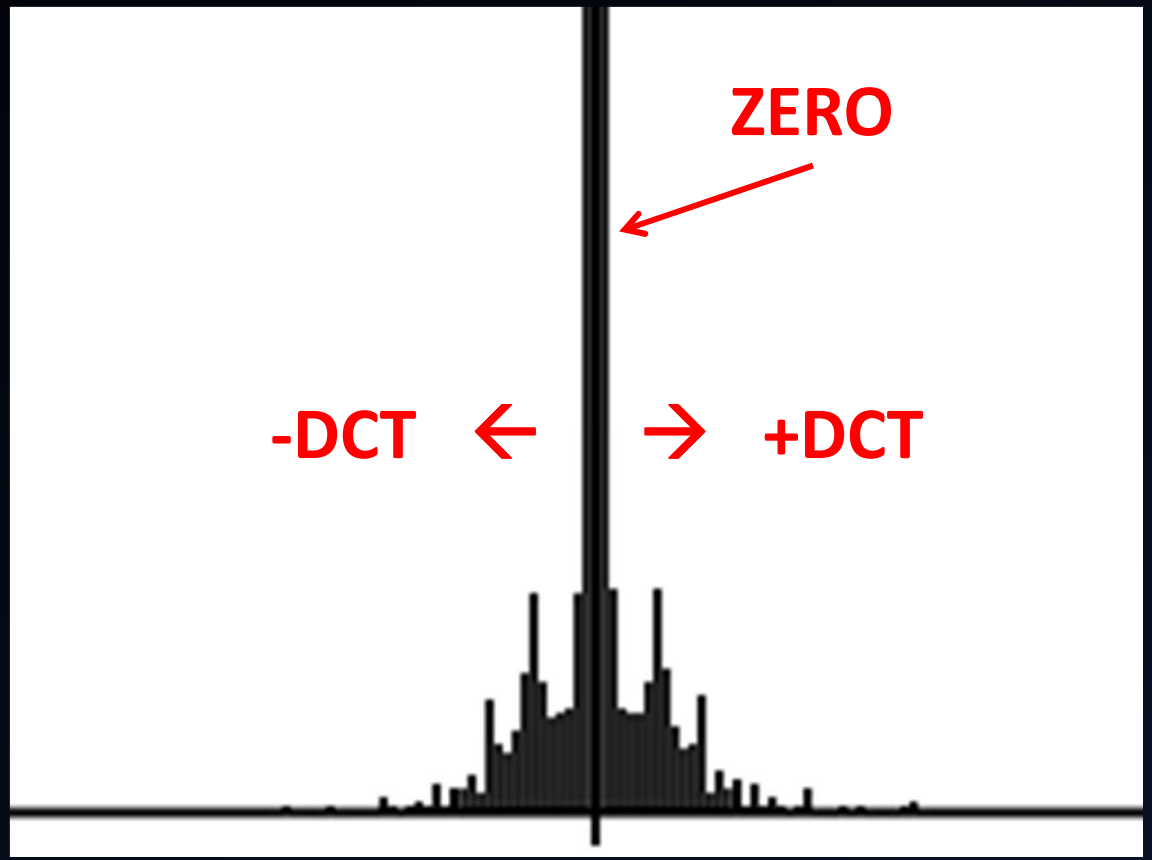
- None of these techniques work!

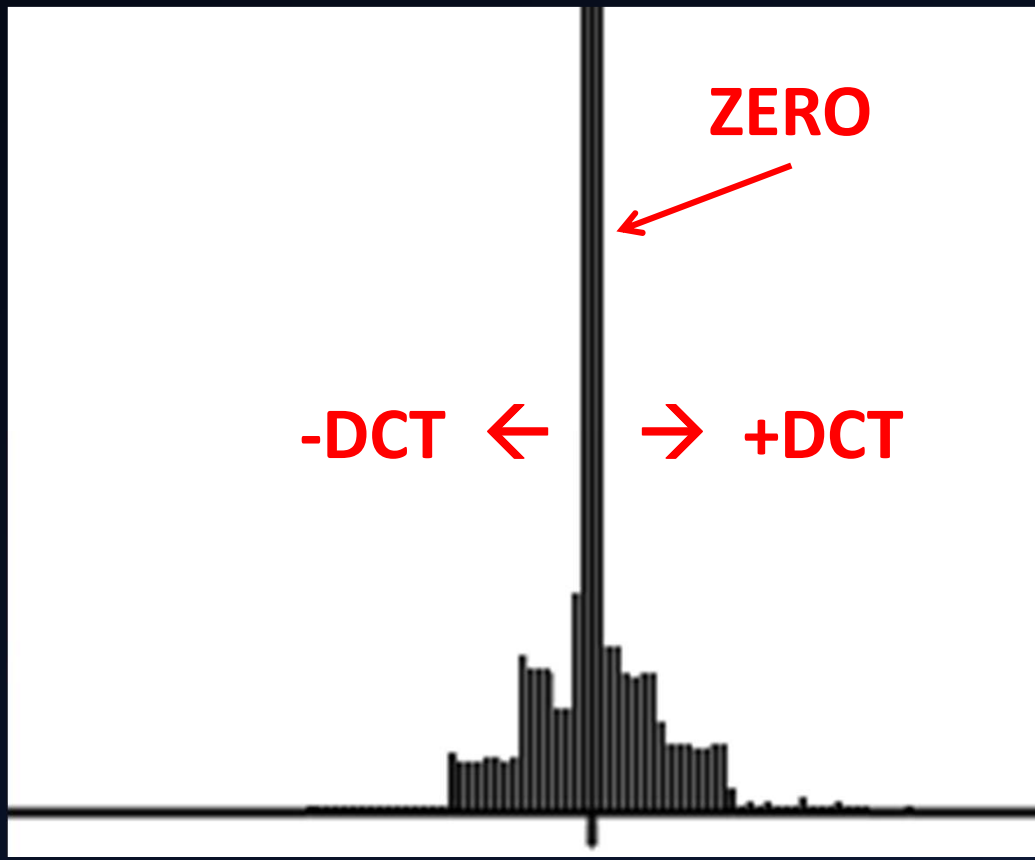




Histogram of DCT Coefficients

- The non-symmetrical histogram has the hidden data





Reversing XOR

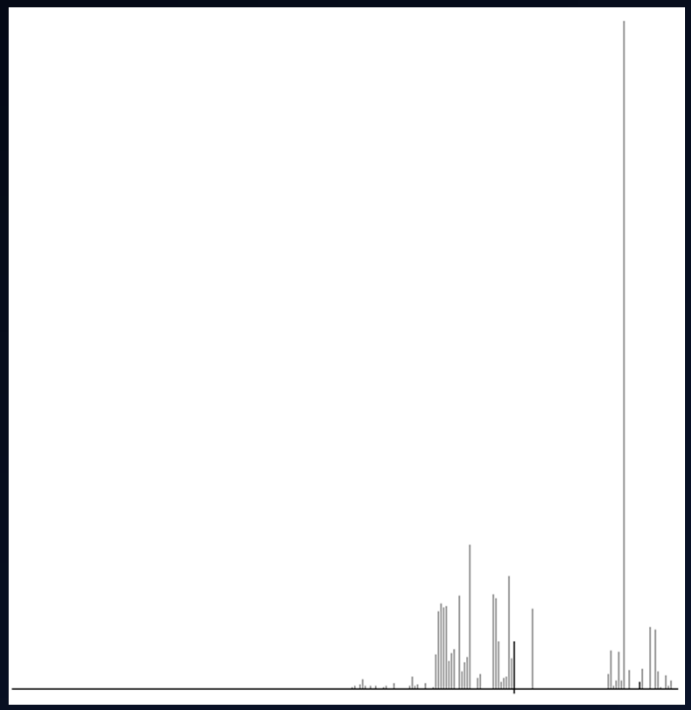
- XOR is used for encryption because it is fast simple

Reversing XOR - Observations

- Something XOR'd with itself is zero.
  - Whenever you find a zero in the target file, the original character is equal to the XOR key used.
- Something XOR'd with zero will be itself.
  - Knowing that a file type has a large number of zeros, particularly if the location is known, can yield the key.
- A letter XOR'd with the space character (0x20) will change the case
  - In an English text file, the space is typically the most common character
- XORing with a single character will not affect the entropy

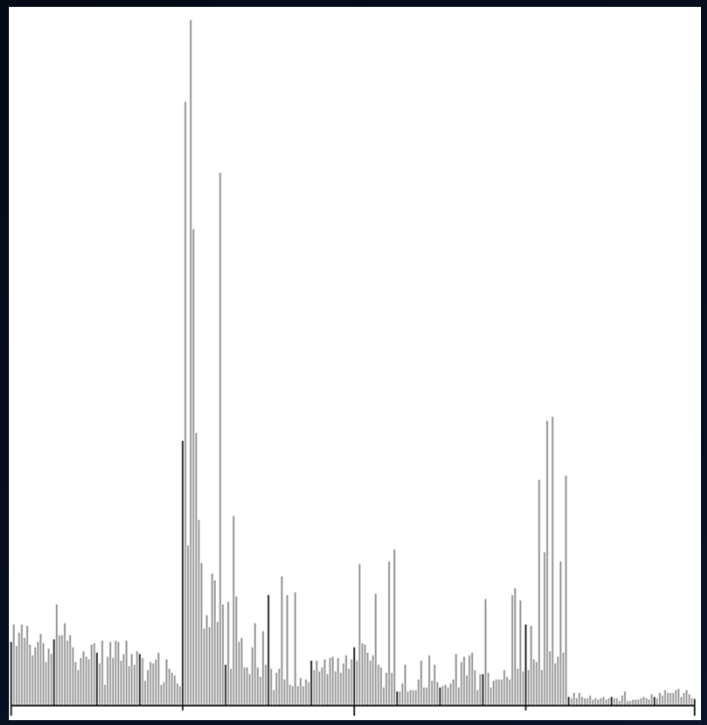
Reversing XOR

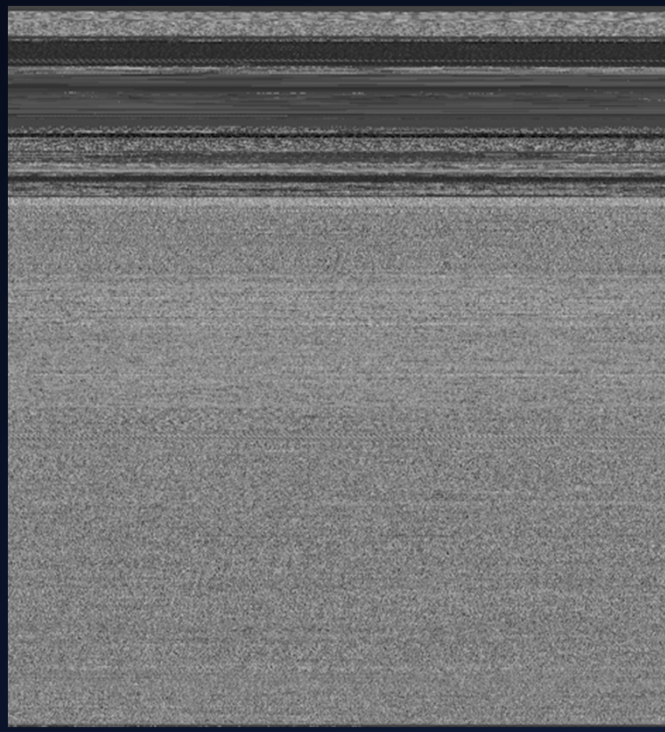
- Looks like text, but shifted …
- Image shows uniform file characteristics
- Space is most common text character
- Textual histogram reveals actual counts



Reversing XOR - 6.67321,

- Histogram does not match any previous file types (H = 7.28069)
- Image of file looks like an executable
- Entropy suggests compression, or … weak encryption
  - First 2 bytes in exe file are “MZ”
  - Zero is prevalent





Reversing XOR

- In target file, first two bytes are 0x09, 0x14
  - 0x09 XOR 0x4d ----> 0x44 “D”
  - 0x14 XOR 0x5a ----> 0x4e “N”
- Looking at textual histogram, “C”, “A”, “N”, “D” are much more prevalent than others
  - Something XOR’d with zero is itself
- With some sleuthing, assumptions, analysis tools, and a bit of luck, you’ve got it!

TOOL: Statistical Analyzer

- This combines the file searching of Footprint and the file type identification of Write Bitmap Histogram
- It searches an entire directory structure and attempts to identify a file's type
  - Uses histograms and a multitude of statistics
  - In its current prototype state, it does not use magic numbers as a clue
- It highlights any abnormalities
- The details are, in and of themselves, an entire 50+ min presentation

Wrap-Up

- Hope you have learned something useful
- Enjoy experimenting and using the tools
- Feel free to contact me by email if you have any other questions

- stego@satx.rr.com

Links to Relevant Harris Blogs

- http://crucialsecurityblog.harris.com/2011/07/06/decoding-data-exfiltration-%E2%80%93-reversing-xor-encryption/

- http://crucialsecurityblog.harris.com/2012/04/16/file-type-identification-and-its-application-for-reversing-xor-encryption/

Link to Irrelevant Harris Blogs

- I wrote this one too and it has very little to do with this presentation, but I'll lay odds most of you will like it!
- http://crucialsecurityblog.harris.com/2012/04/09/on-the-difficulty-of-autonomous-pornography-detection/

References

- Conti, Greg; Grizzard, Julian; Ahamad, Mustaque; Owen, Henry; Visual Exploration of Malicious Network Objects Using Semantic Zoom, Interactive Encoding and Dynamic Queries. Georgia Institute of Technology

QUESTIONS ???

COMMENTS? COMPLAINTS?