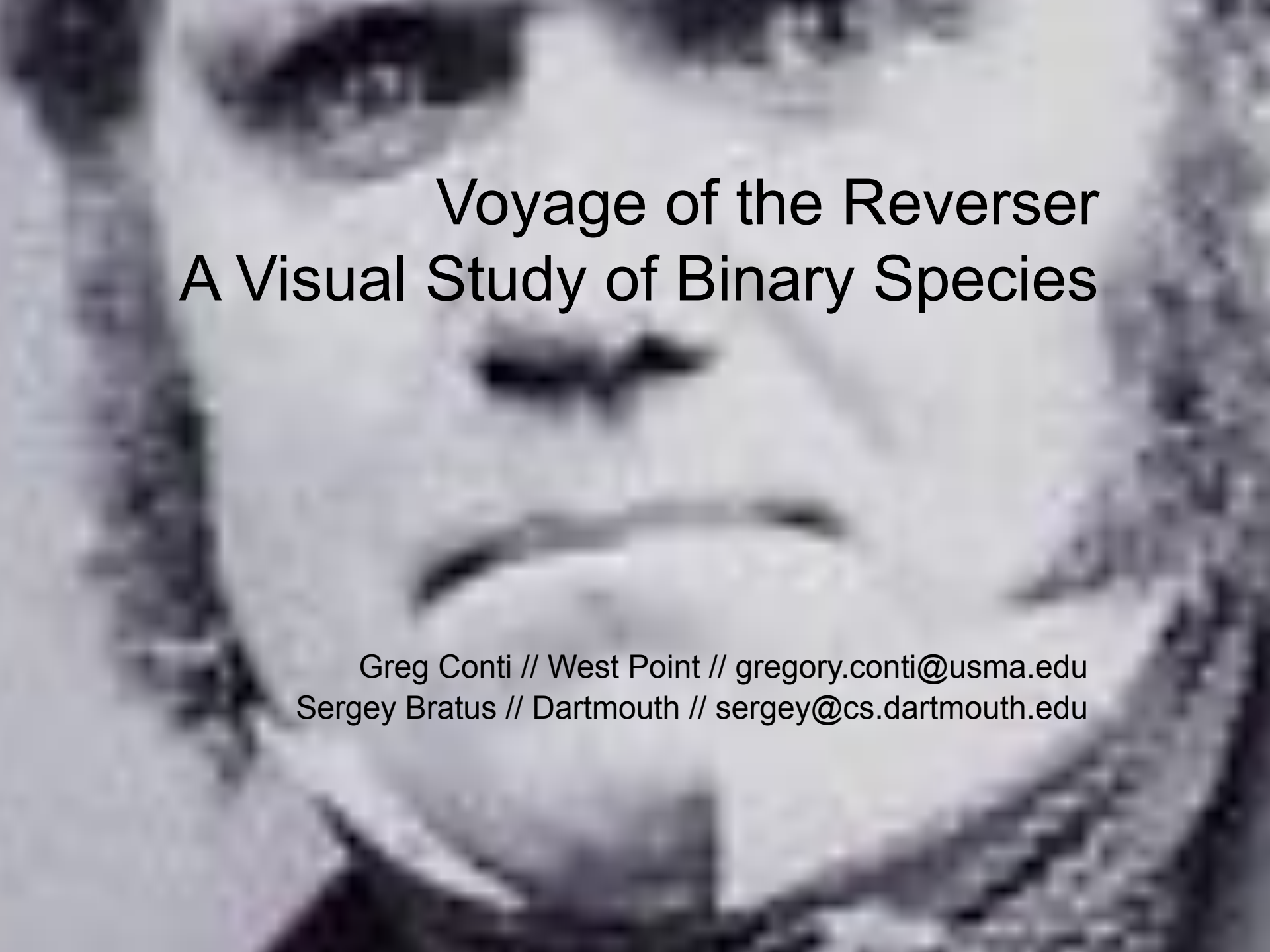


Voyage of the Reverser
A Visual Study of Binary Species

Greg Conti // West Point // gregory.conti@usma.edu
Sergey Bratus // Dartmouth // sergey@cs.dartmouth.edu

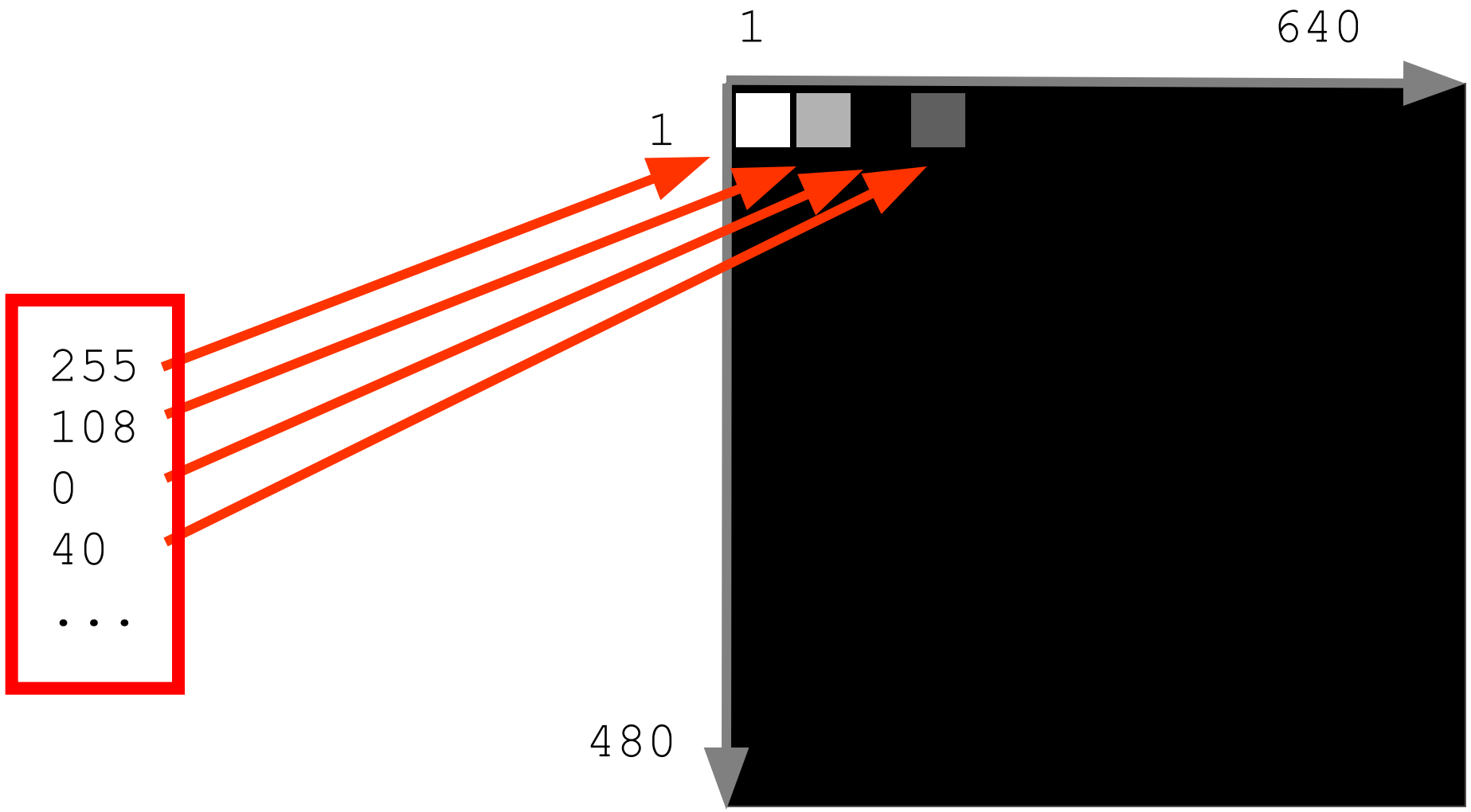
Qvfpynvzre

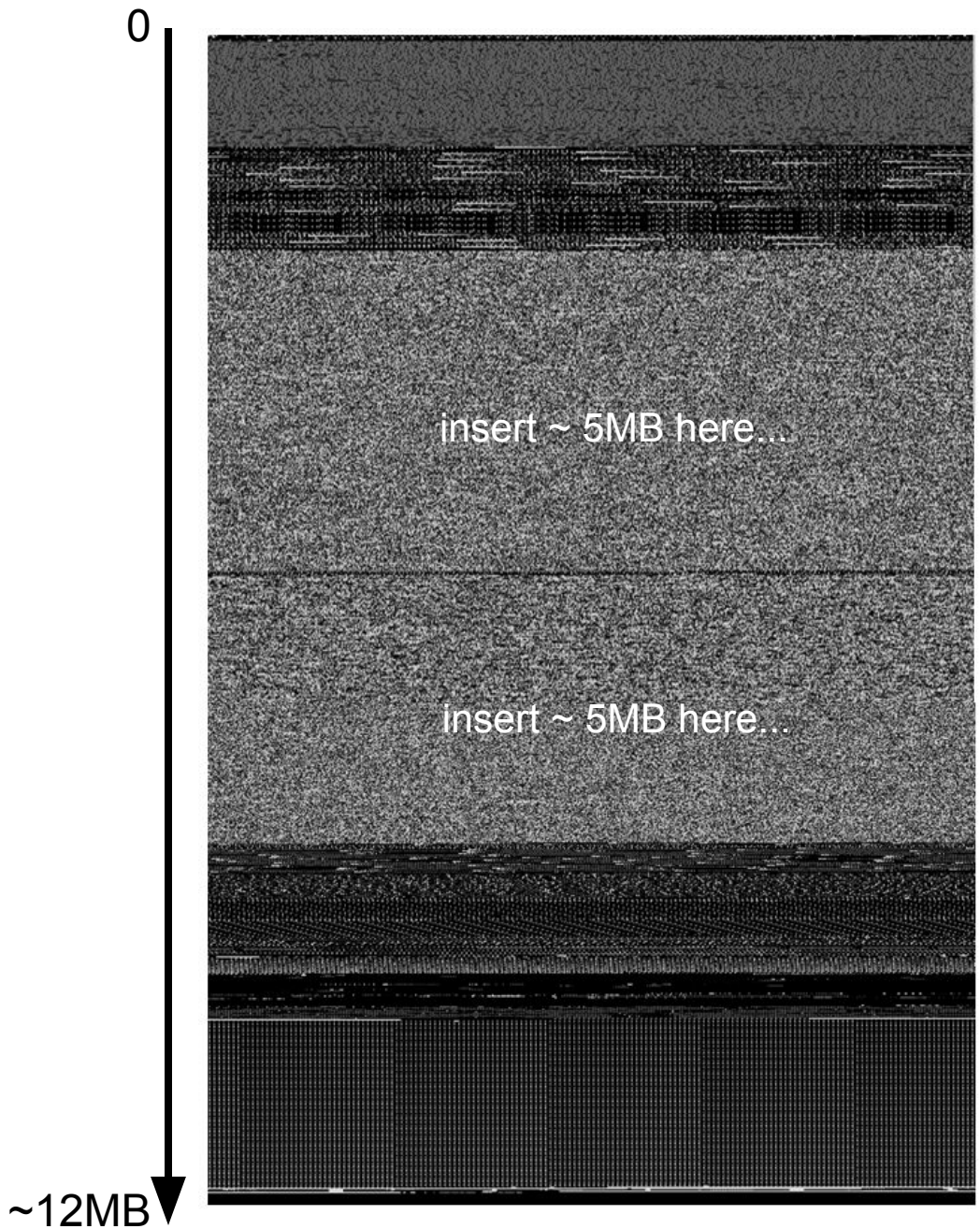
Gur ivrjf rkcerffrq va guvf cerfragngvba ner gubfr bs gur nhgube naq qb abg ersyrpg gur bssvpvny cbyvpl be cbfvgvba bs gur Havgrq Fgngrf Zvyvgnel Npnqrzl, gur Qrcnegzrag bs gur Nezl, gur Qrcnegzrag bs Qrsrafr be gur H.F. Tbireazrag.

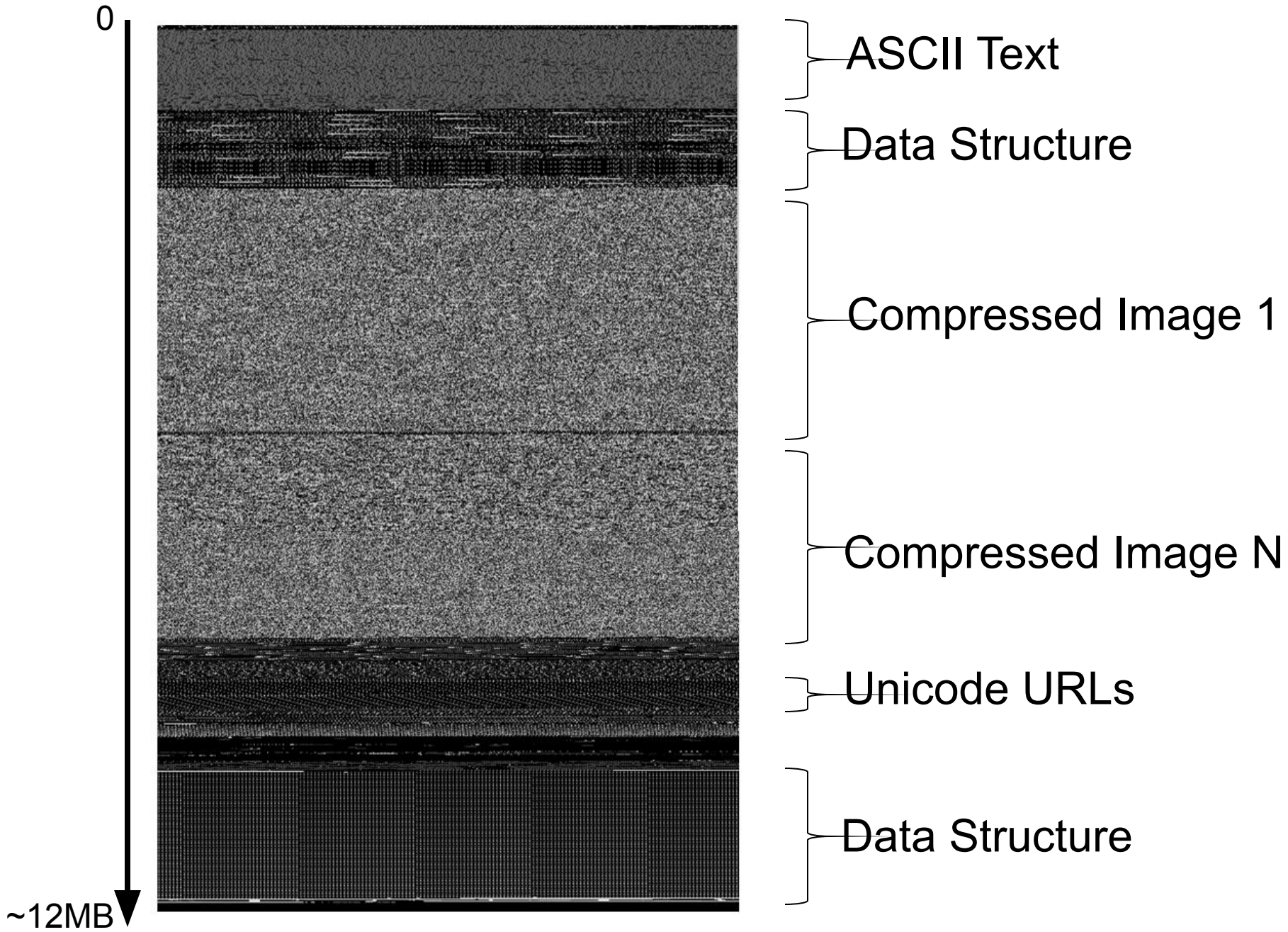
Disclaimer

The views expressed in this presentation are those of the author and do not reflect the official policy or position of the United States Military Academy, the Department of the Army, the Department of Defense or the U.S. Government.

Byte Plot




0
insert ~ 5MB here...
insert ~ 5MB here...
~12MB


0
~12MB
ASCII Text
Data Structure
Compressed Image 1
Compressed Image N
Unicode URLs
Data Structure

What is a “Primitive Type?”

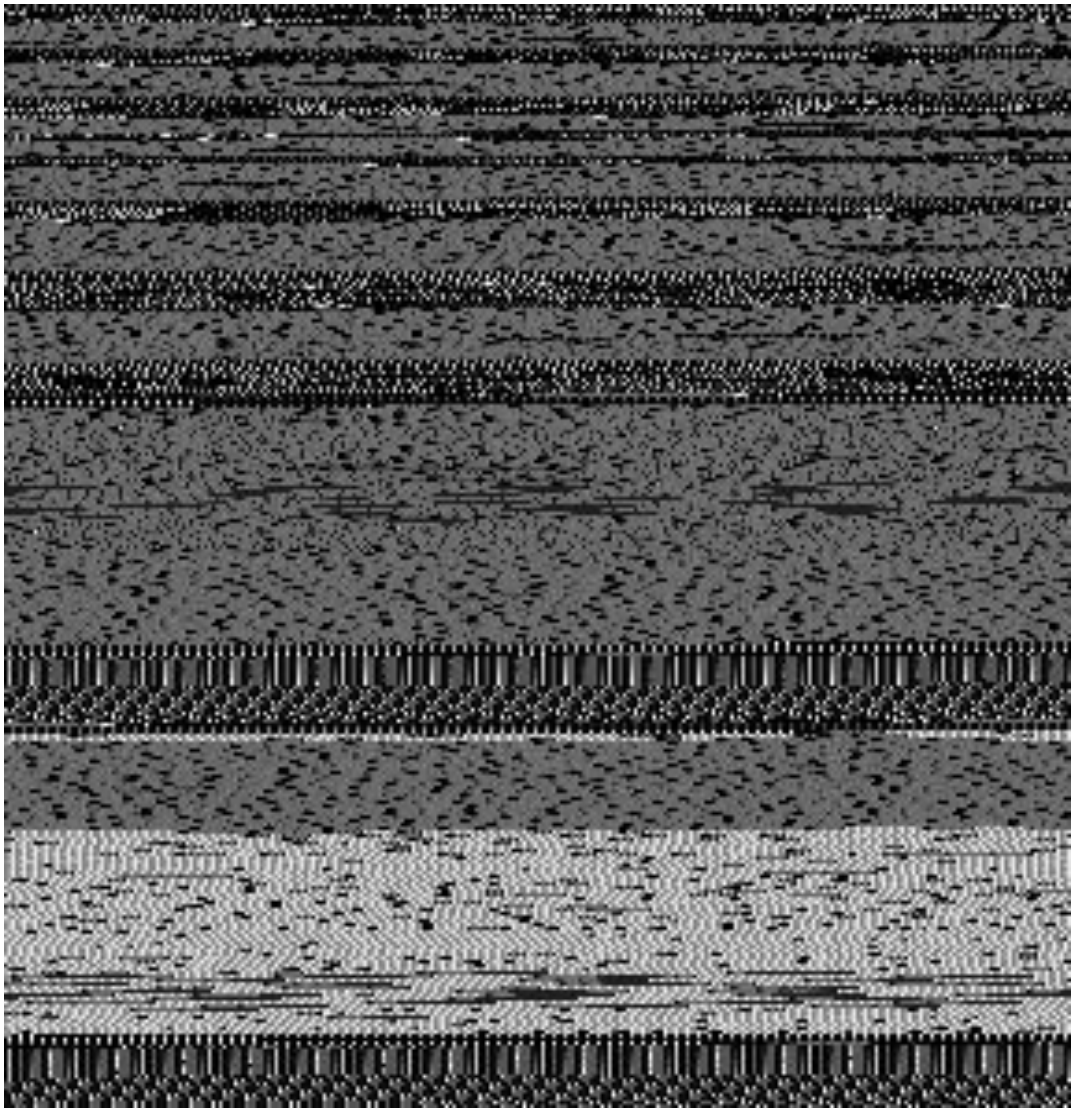
{int, long, char, string …} < **Primitive Type** < {.doc, .jar, .exe …}

What is a “Primitive Type?”

{int, long, char, string …} < **Primitive Type** < {.doc, .jar, .exe …}

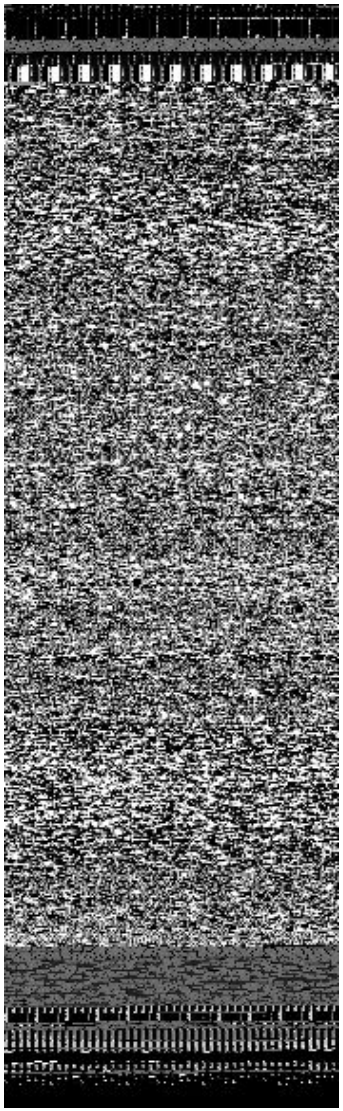
Demo shell32.dll

Archive Files



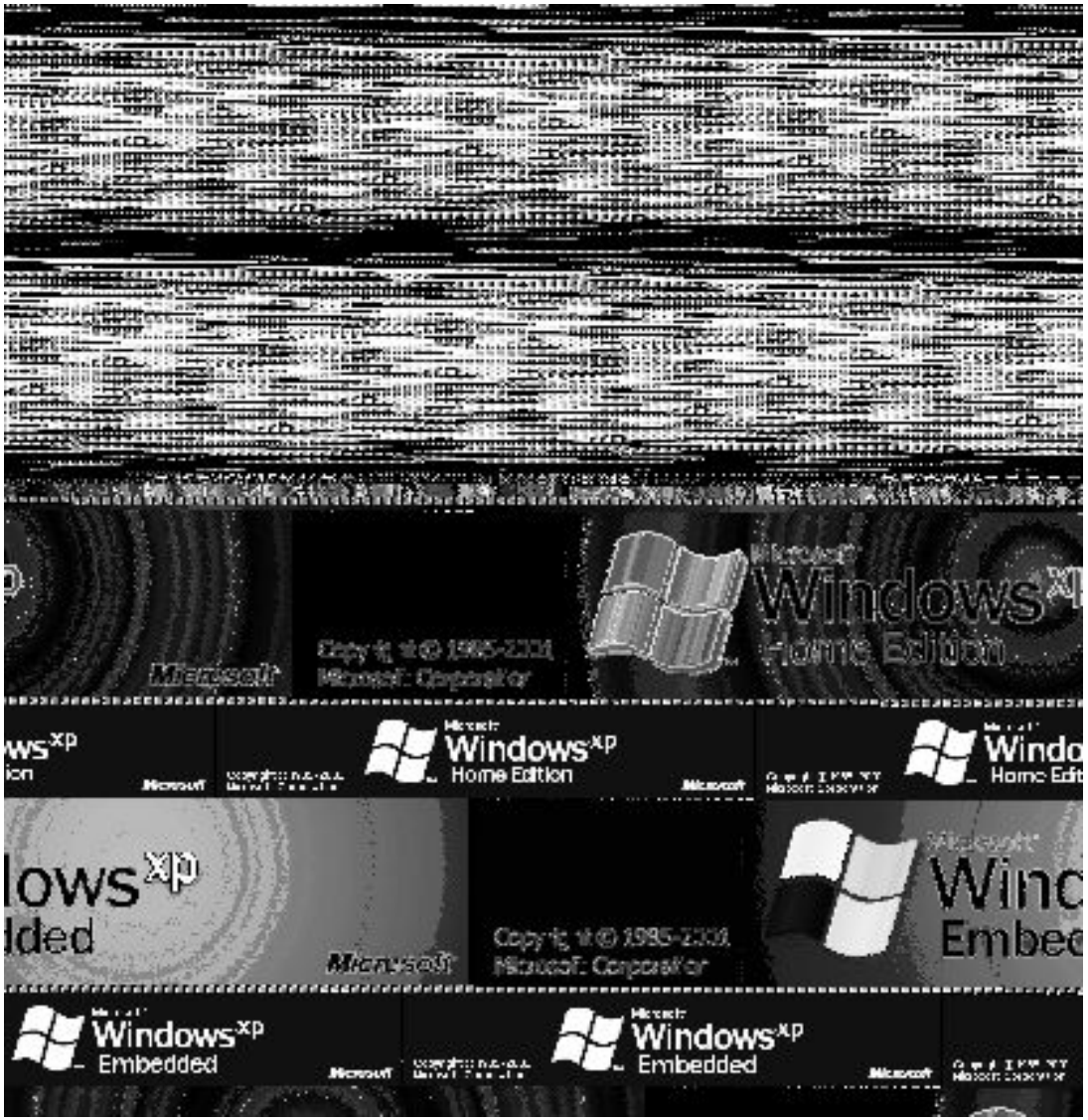
tools.jar

Executables



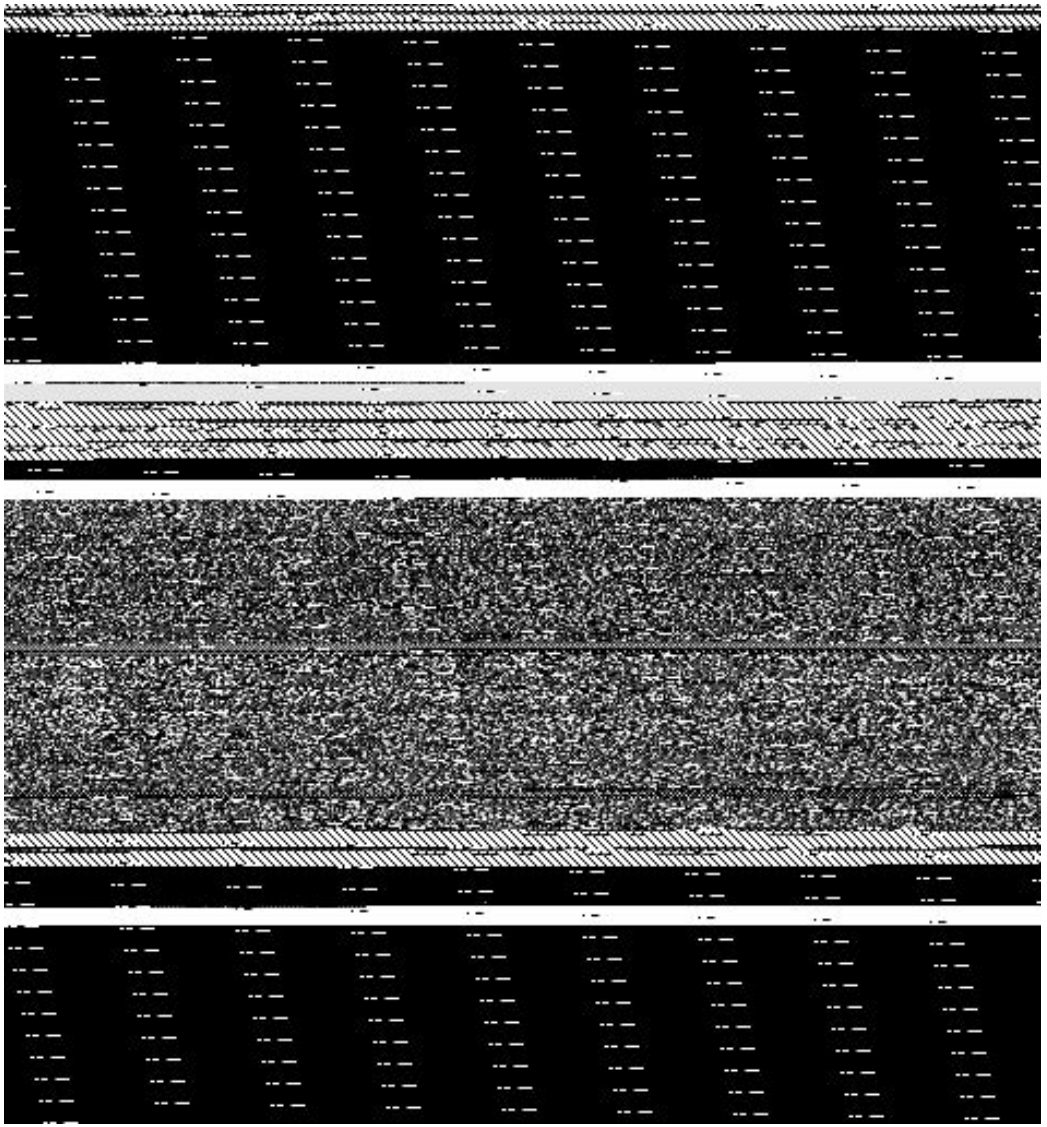
grep (elf file format)

dynamic libraries



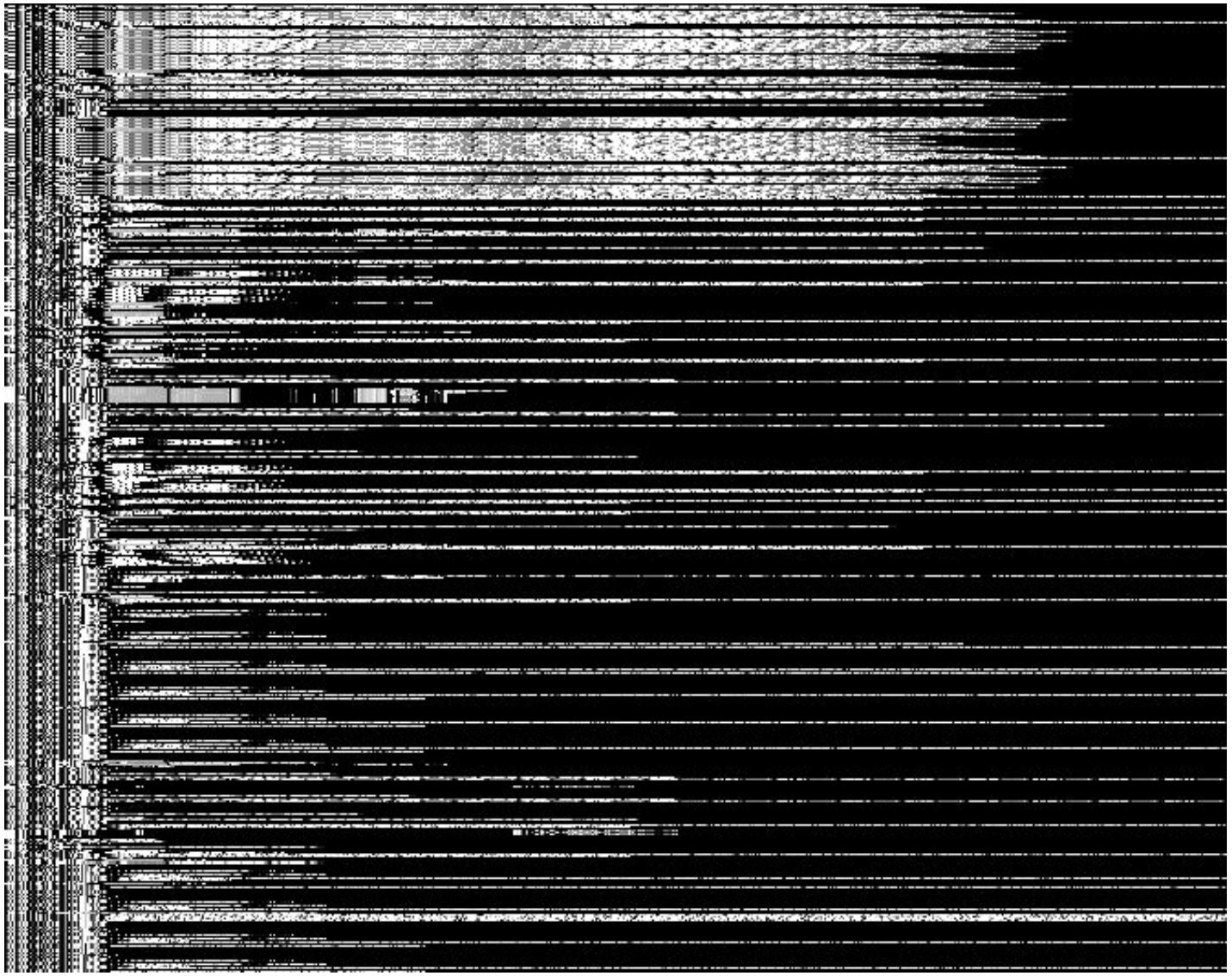
shell32.dll

System Memory



SonyEricsson K800i (DFRWS 2010)

Network Traffic



grep, strings, hex editors are insufficient

Why

- Identify unknown/unfamiliar structures
- Facilitate deep understanding
- Reversing
- Fuzzing
- Memory forensics
- General forensics
- Memory mapping
- Interactive filtering
- Dictionary

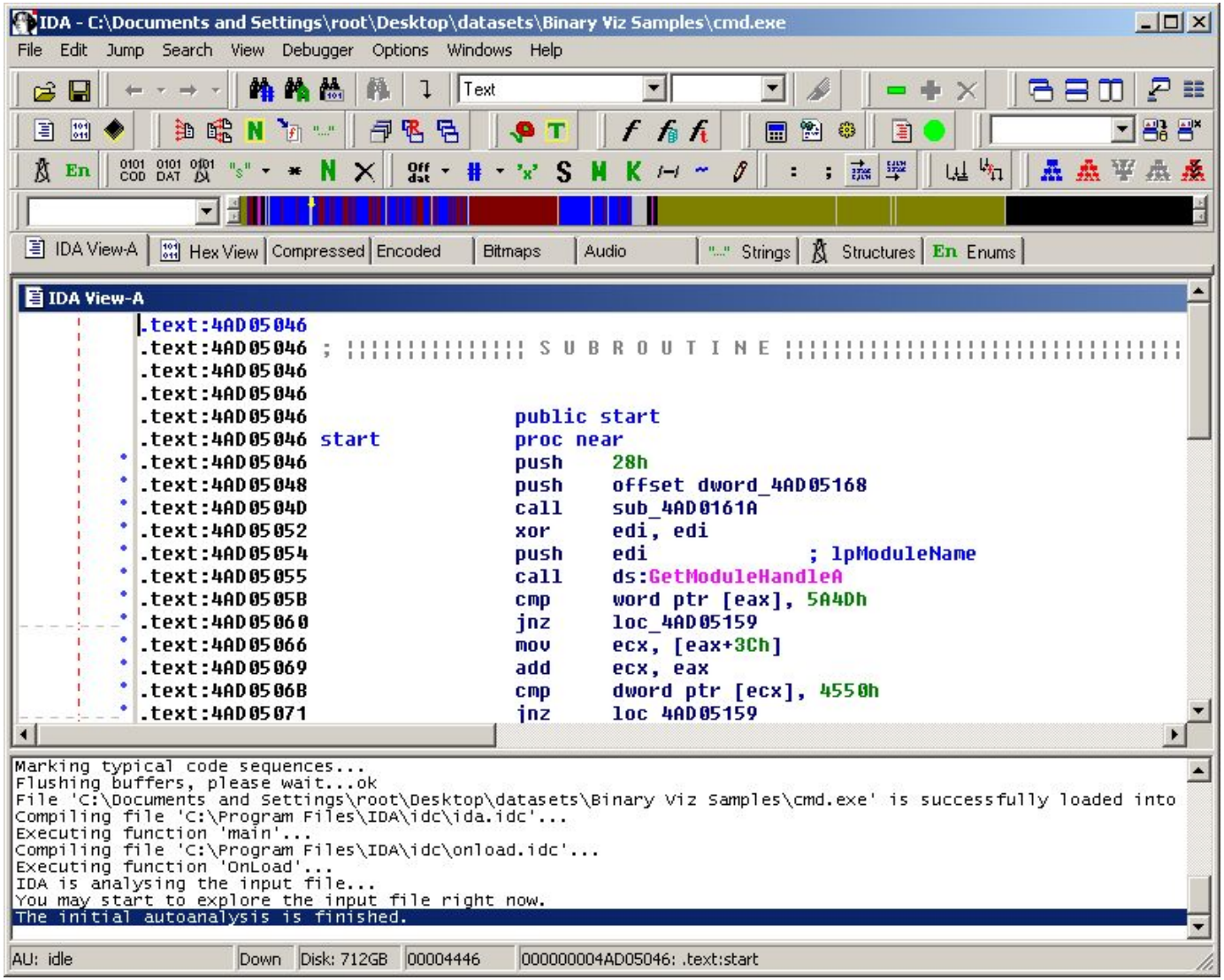
One Motivation

```
0400-07FF 1024-2047 Screen memory
0800-9FFF 2048-40959    Basic ROM memory
8000-9FFF 32758-40959   Alternate: Rom plug-in area
A000-BFFF 40960-49151   ROM : Basic
A000-BFFF 49060-59151   Alternate: RAM
C000-CFFF 49152-53247   RAM memory, including alternate
D000-D02E 53248-53294   Video Chip (6566)
D400-D41C 54272-54300   Sound Chip (6581 SID)
D800-DBFF 55296-56319   Color nybble memory
DC00-DC0F 56320-56335   Interface chip 1, IRQ (6526 CIA)
DD00-DD0F 56576-56591   Interface chip 2, NMI (6526 CIA)
D000-DFFF 53248-53294   Alternate: Character set
E000-FFFF 57344-65535   ROM: Operating System
E000-FFFF 57344-65535   Alternate : RAM
FF81-FFF5 65409-65525   Jump Table
```

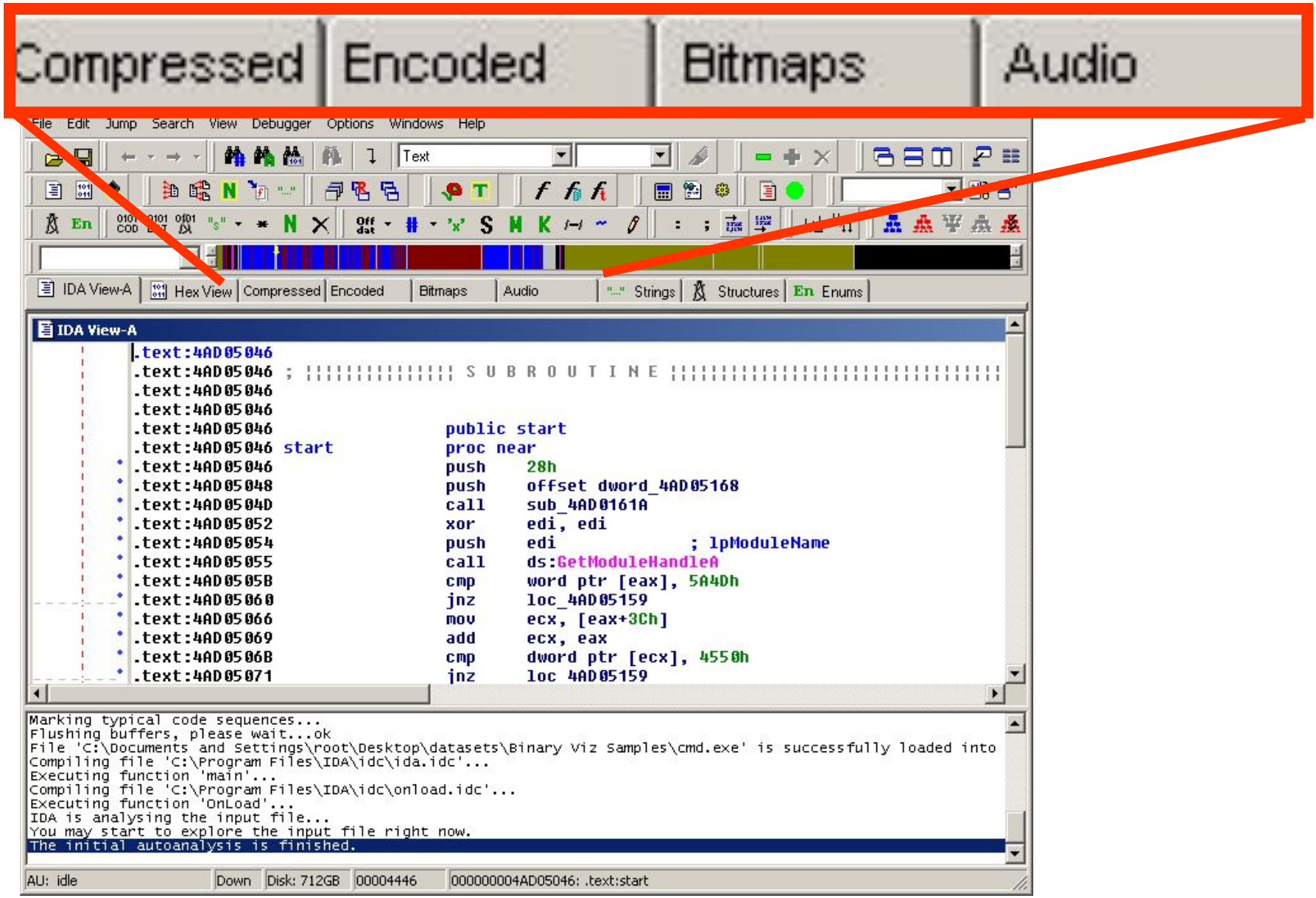
Concept

```
0400-07FF 1024-2047 ASCII Text (English)
0800-9FFF 2048-40959   Pointer Table
8000-9FFF 32758-40959  Variable Length Array
A000-BFFF 40960-49151  Compressed Data
A000-BFFF 49060-59151  Unicode (Basic Latin)
C000-CFFF 49152-53247  Unknown Region
D000-D02E 53248-53294  Repeating Value (0xFF)
D400-D41C 54272-54300  Encrypted Region (AES)
D800-DBFF 55296-56319  PNG Image
DC00-DC0F 56320-56335  JavaScript
DD00-DD0F 56576-56591  Encrypted Region (RSA Key?)
D000-DFFF 53248-53294  Unknown Region
E000-FFFF 57344-65535  BMP Image
E000-FFFF 57344-65535  Unicode (Hyperlinks?)
FF81-FFF5 65409-65525  Repeating Value (0x00)
```

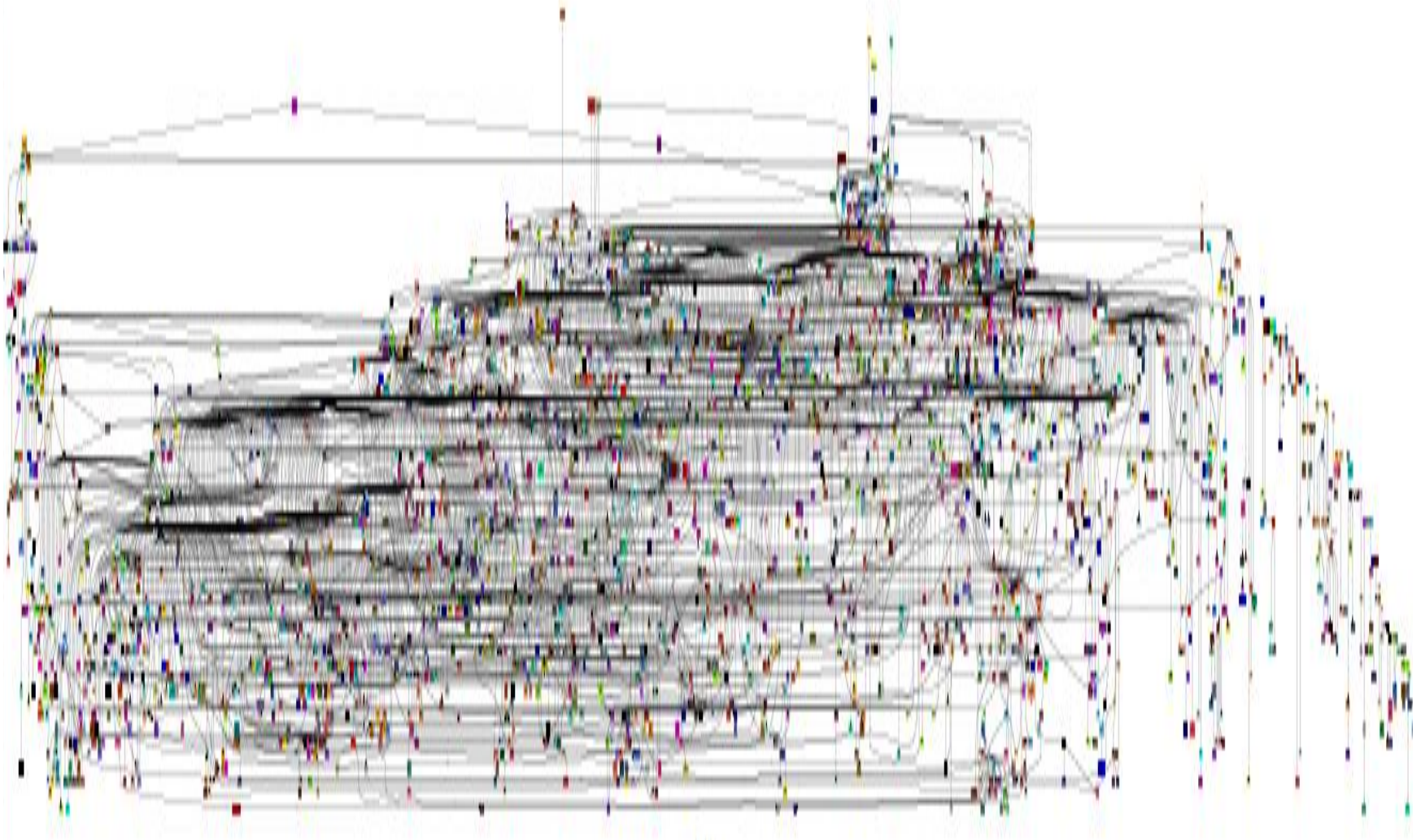
Another Concept



Another Concept

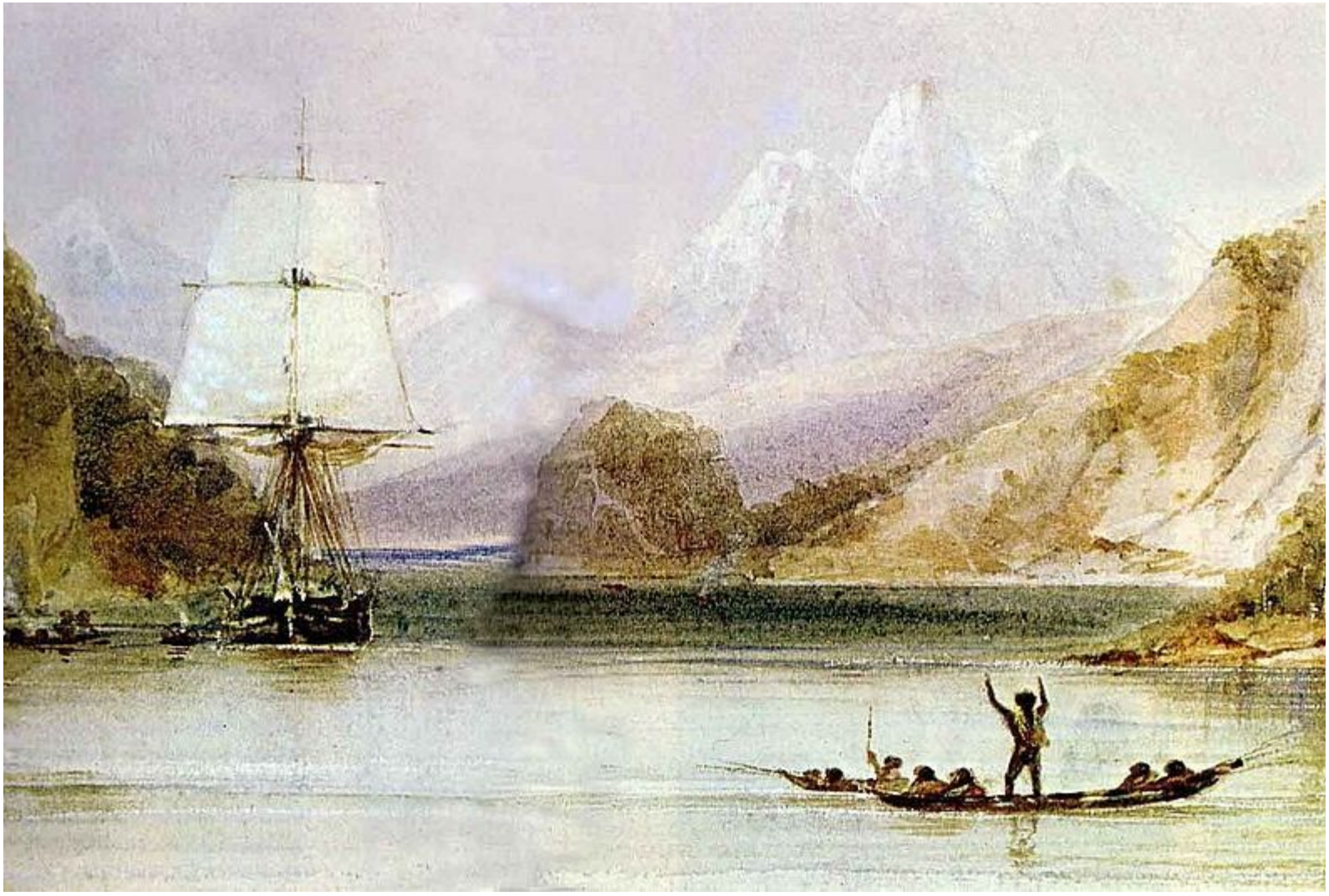


Potentially Overwhelming Complexity

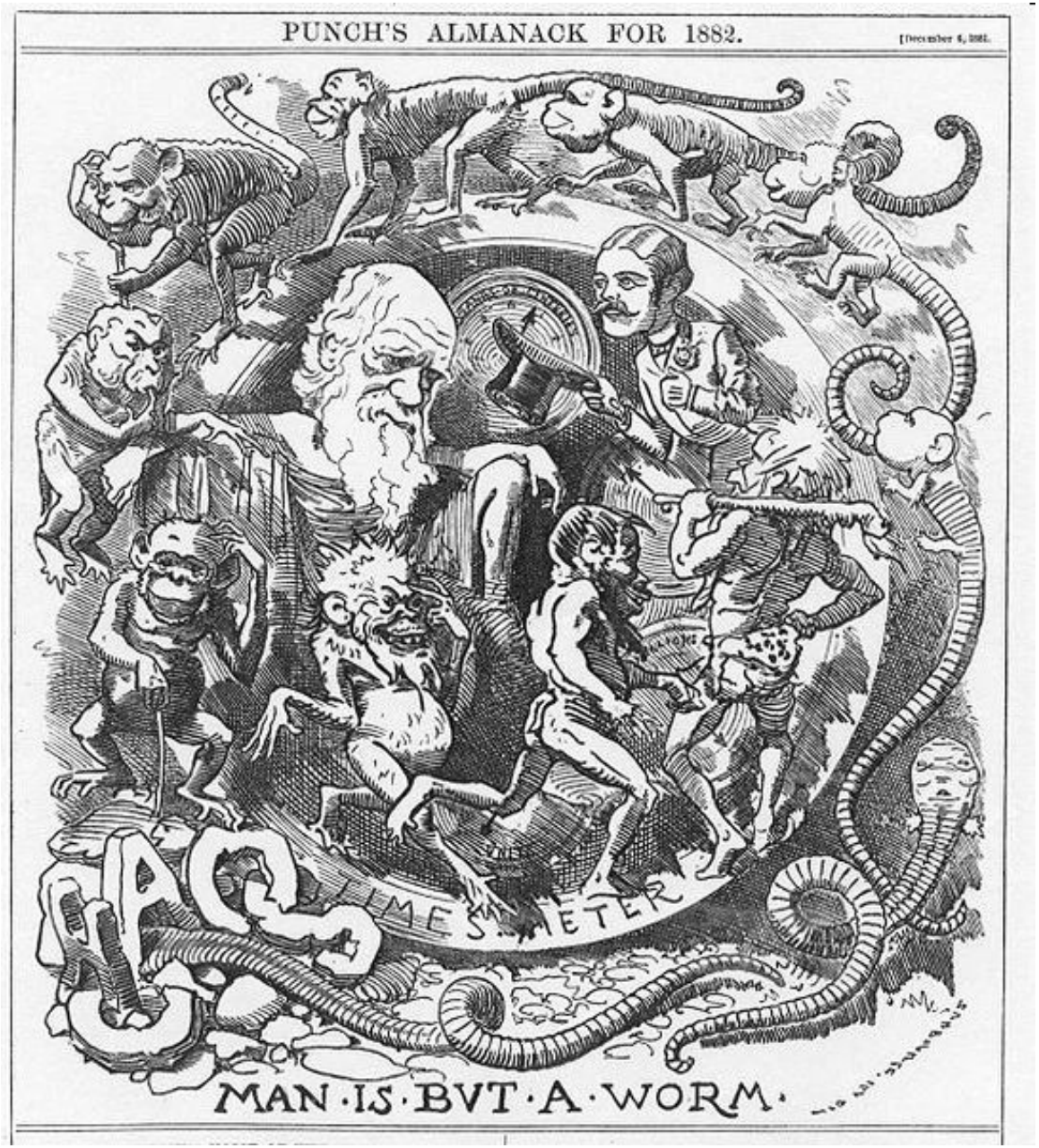


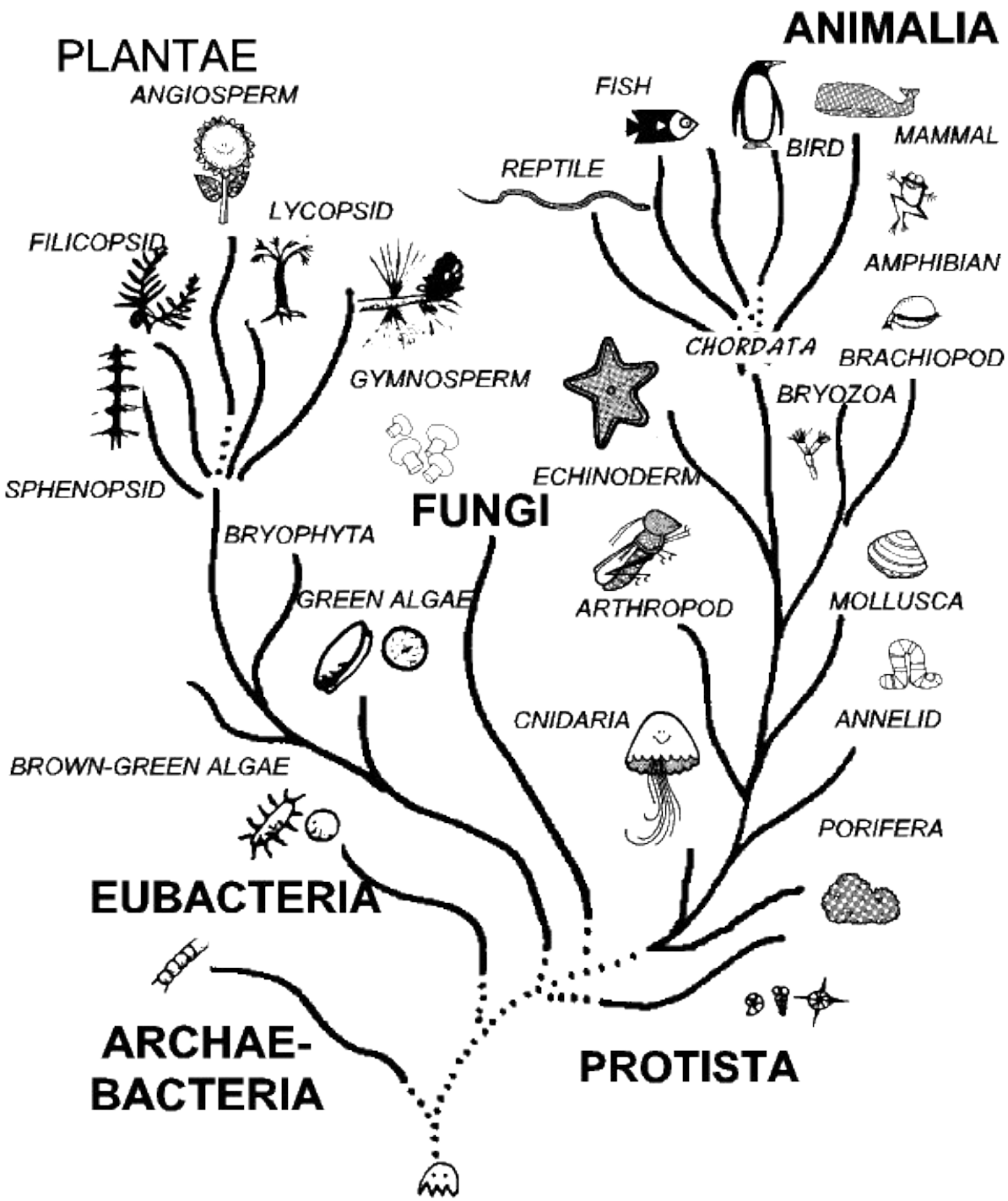
http://hopl.murdoch.edu.au/images/genealogies/tester-endo.pdf

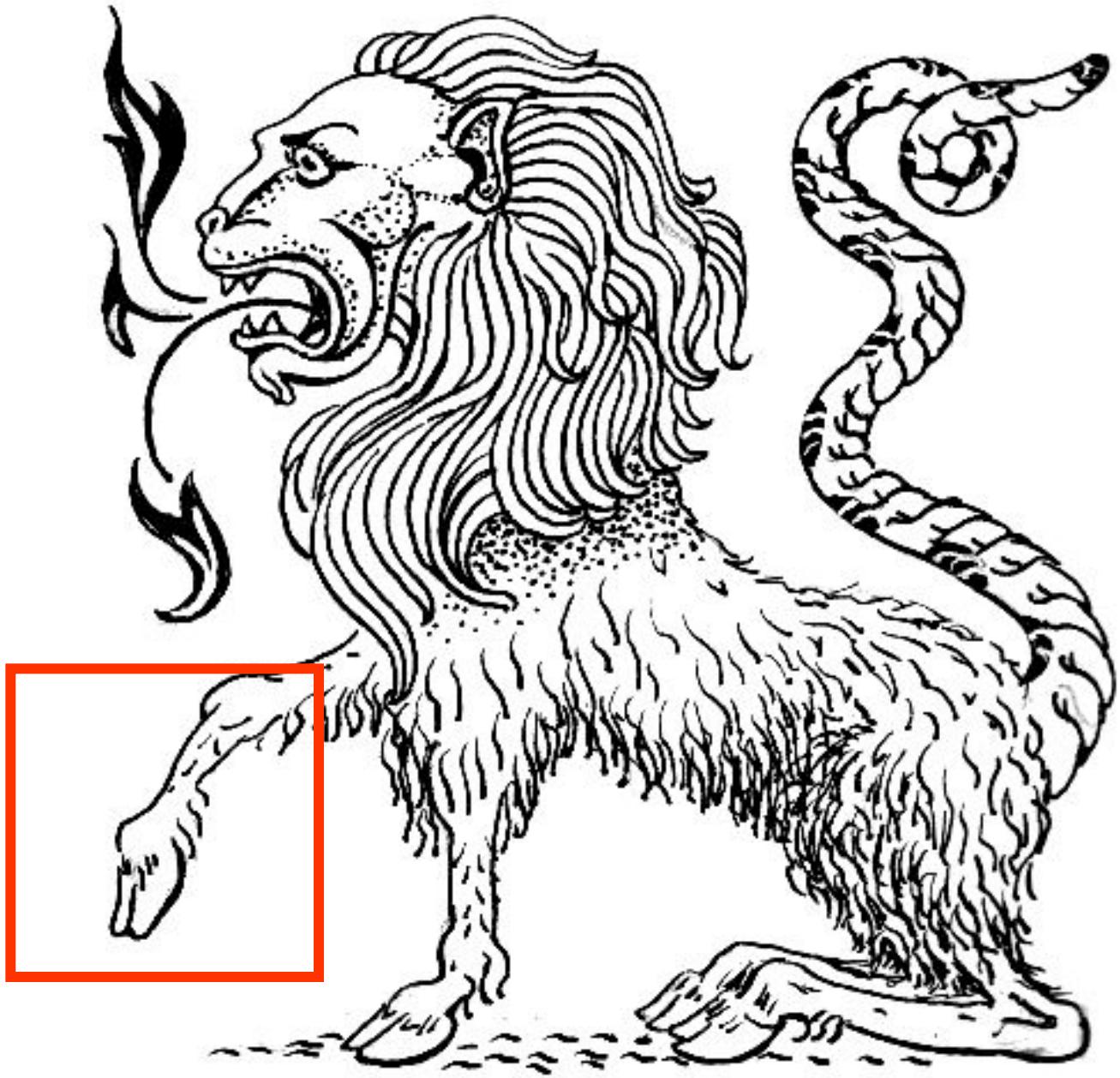
History of Categorizing Nature



http://en.wikipedia.org/wiki/File:HMS_Beagle_by_Conrad_Martens.jpg

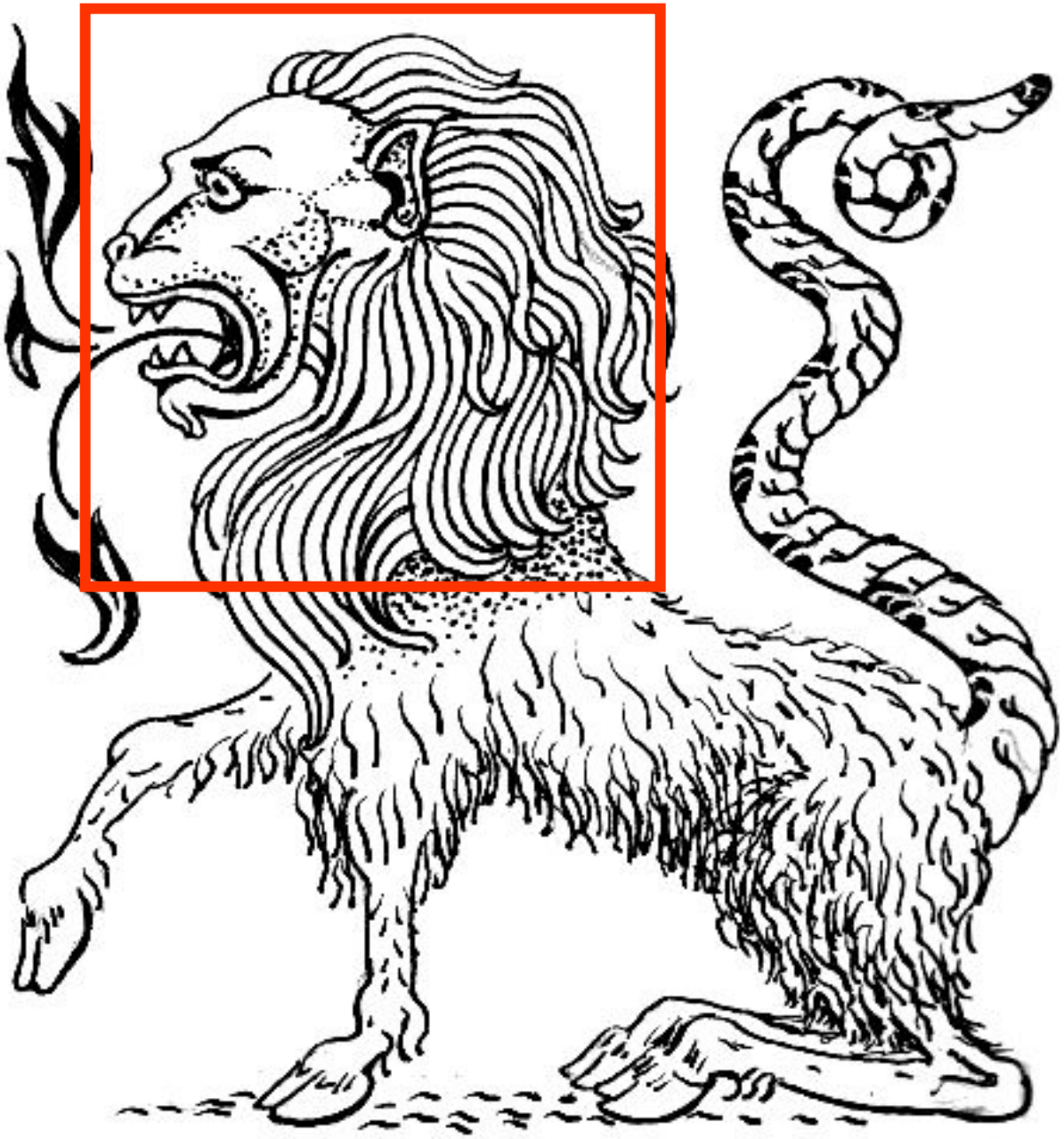






http://commons.wikimedia.org/wiki/File:Chimera_%28PSF%29.jpg

http://commons.wikimedia.org/wiki/File:Chimera_%28PSF%29.jpg



http://commons.wikimedia.org/wiki/File:Chimera_%28PSF%29.jpg

Design Choices

- When are we talking about more than a data type?
  - (e.g. int, long, char… vs. a primitive type)
- We can't identify every primitive type after the fact, but…
- Less about files and more about fragments
  - (i.e. headers and payload are distinct fragments)
- Layer transformations
  - e.g. multiple applications of encryption, compression, and/or encoding
- Coping with artifacts

Primitive Types Overview

- Text
- Image
- Audio
- Video
- Application
- Random
- Encrypted
- Repeating Values / Padding
- Other Compressed
- Other Encoded
- Other

Inspiration

- RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Media Types
  - text, image, audio, video, and application
- Internet Assigned Numbers Authority
  - registered basic media content types
- Sweetscape Software
  - 010 binary template archive
- FILExt file extension database
- File format specifications
  - especially container file formats
- Object Linking and Embedding documents

Identification

- View
  - byte plot
  - hex/ASCII
  - frequency histogram
  - digraph plot
- Compare with dictionary of similar structures
- Look for ways to automate

http://www.ehow.com/how_4836447_throw-live-murder-mystery-party.html

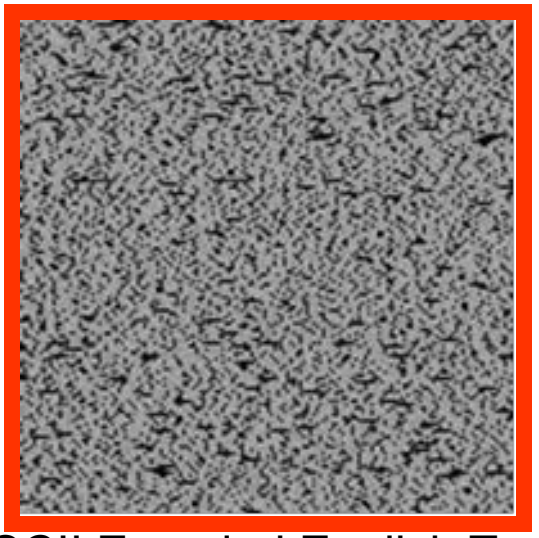
As you see these examples consider how we could algorithmically identify each type

Text

C++ Source Code

Text

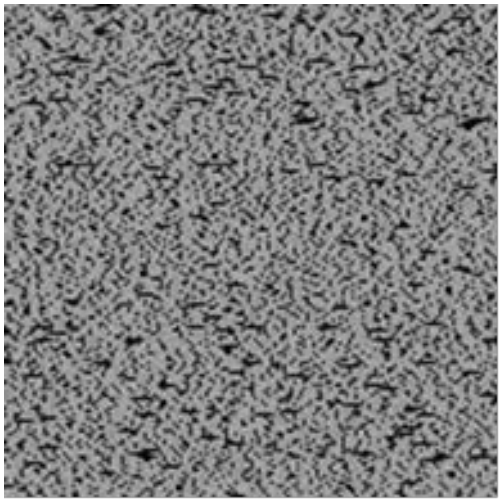
C++ Source Code



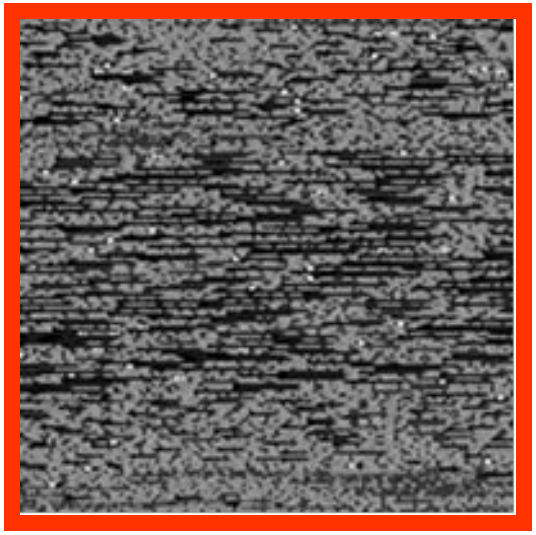
ASCII Encoded English Text

Text

C++ Source Code



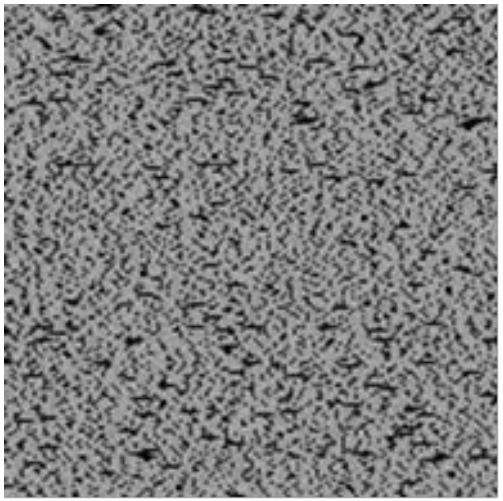
ASCII Encoded English Text



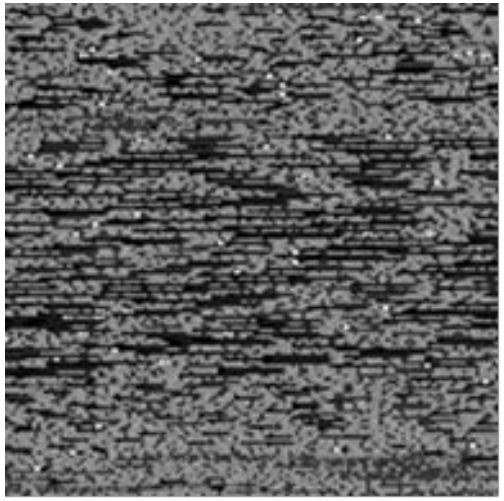
ASCII Encoded HTML

Text

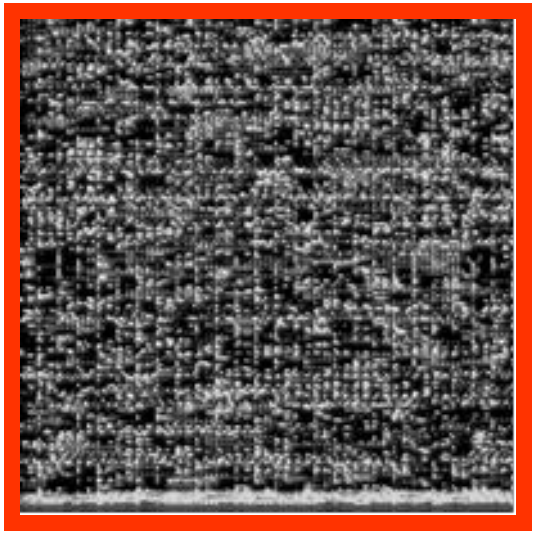
C++ Source Code



ASCII Encoded English Text



ASCII Encoded HTML



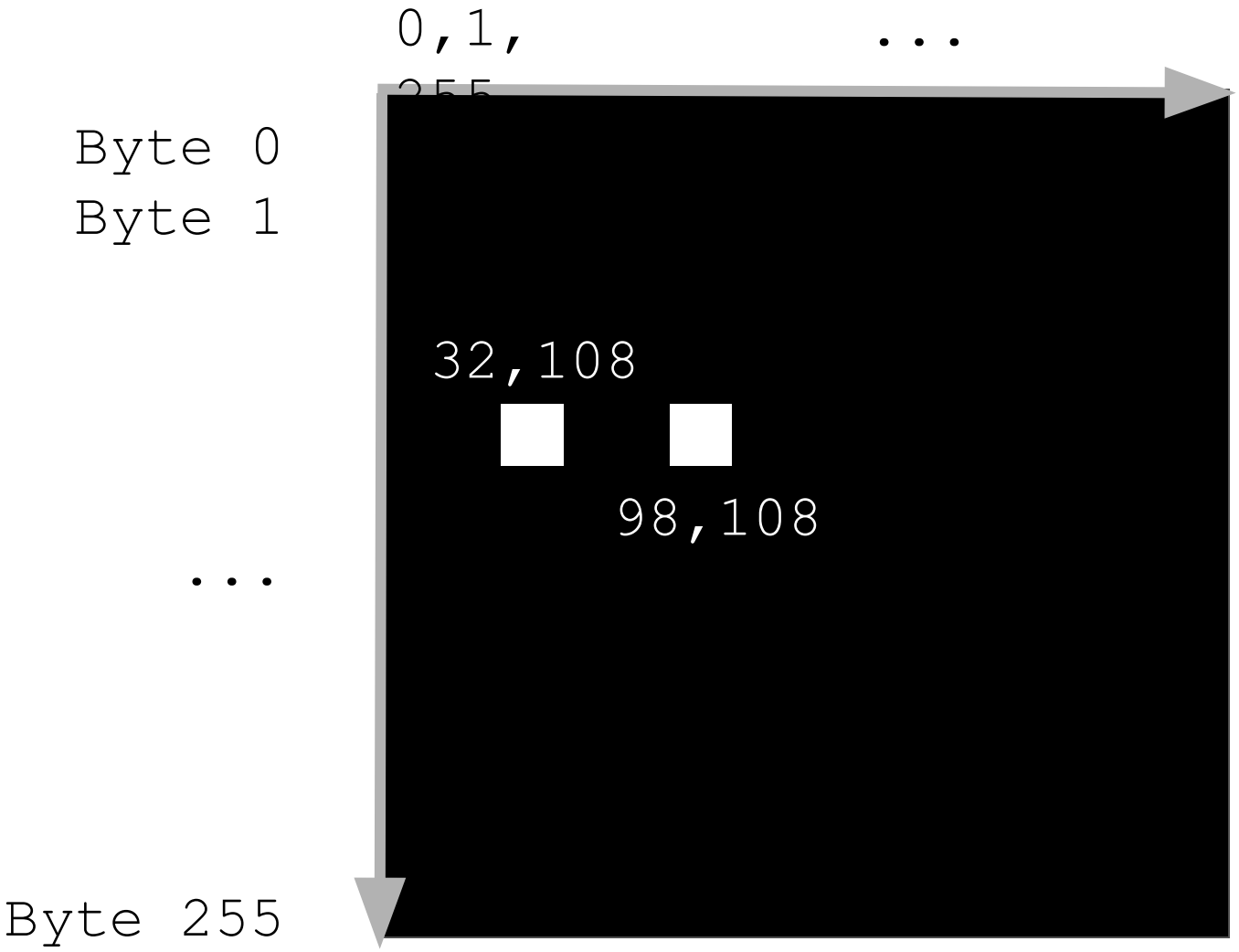
Basic Latin Unicode

Digraph View

black hat

```
bl      (98,108)
 la     (108,97)
  ac      (97,99)
   ck     (99,107)
    k_      (107,32)
     _h     (32,104)
     _ha    (104,97)
      at    (97,116)
```

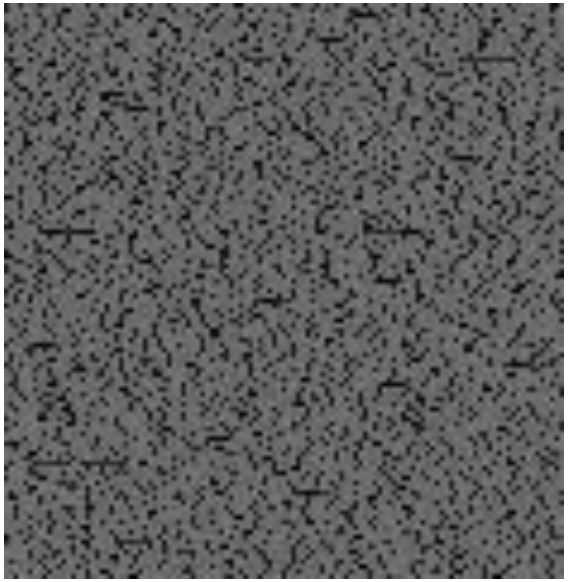
Digraph View



See also Michal Zalewski's "Strange Attractors and TCP/IP Sequence Number Analysis" work.

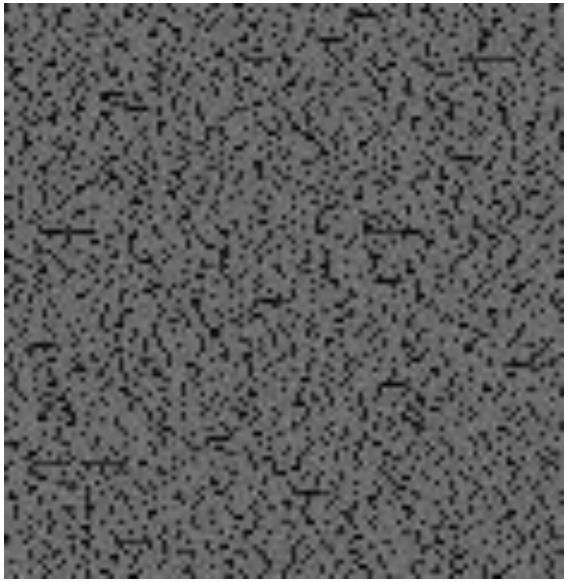
ASCII Encoded English Text

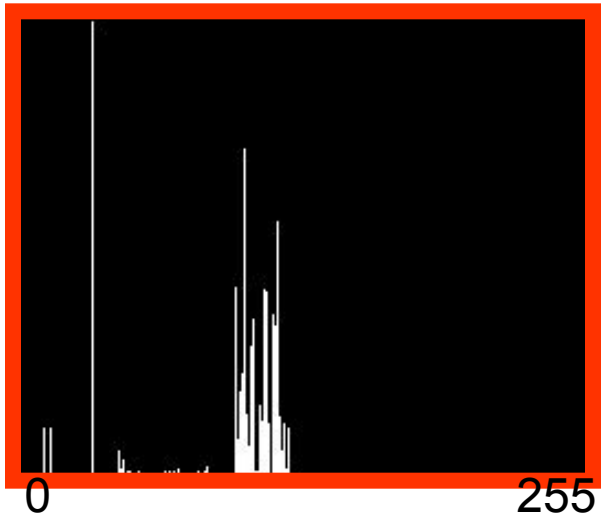
Sample



ASCII Encoded English Text

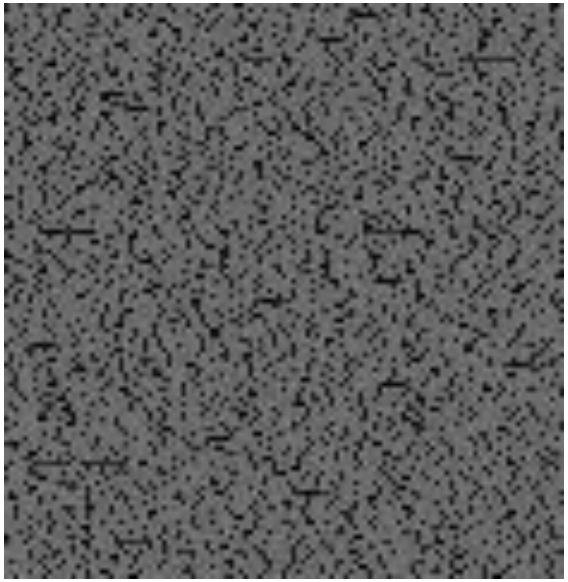
Sample

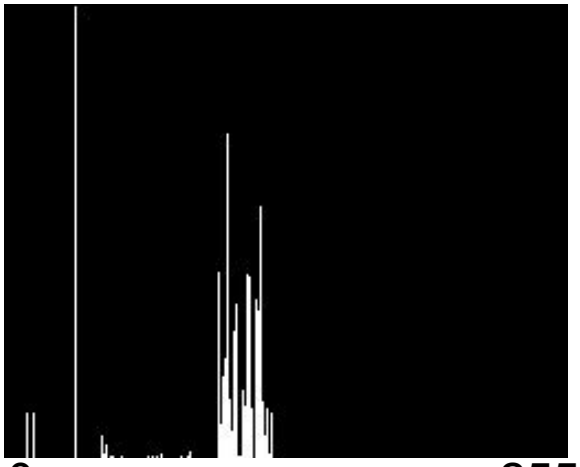




ASCII Encoded English Text

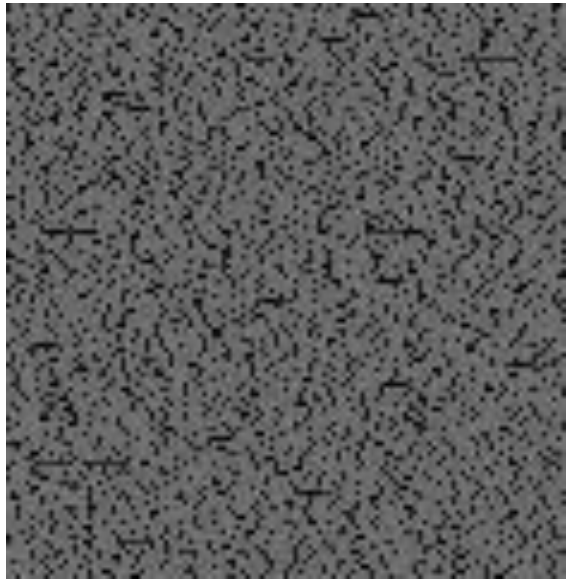
Sample

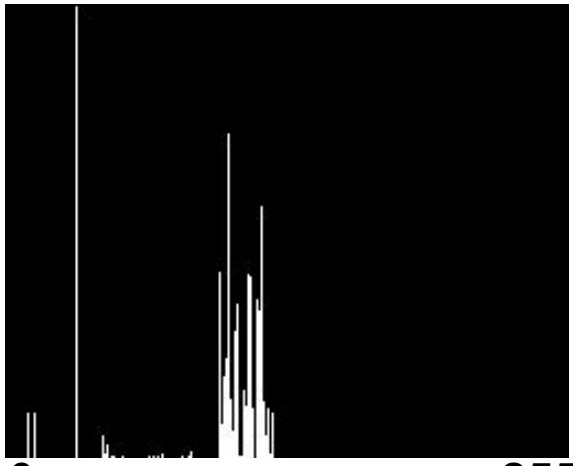




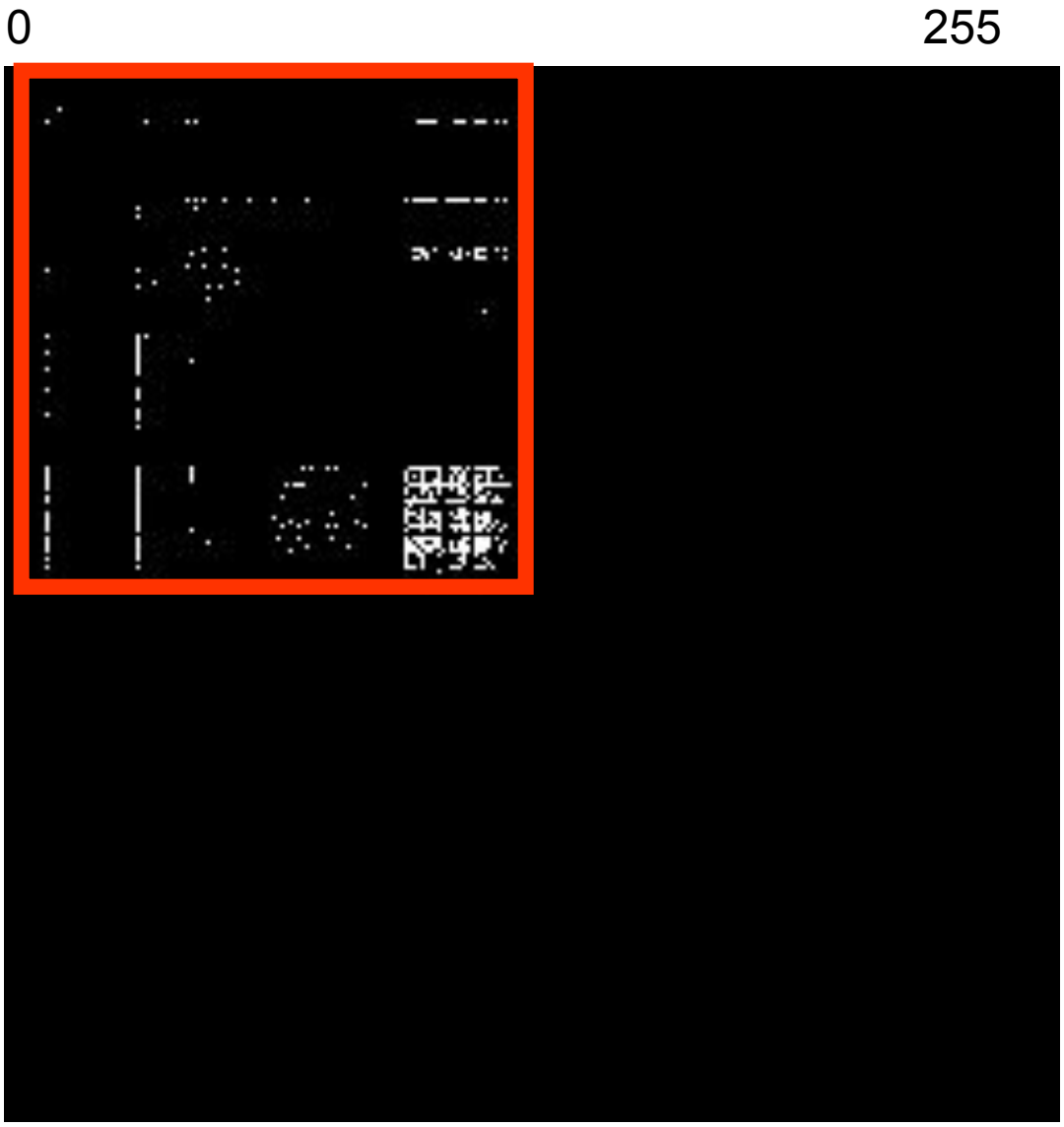
ASCII Encoded English Text

Sample

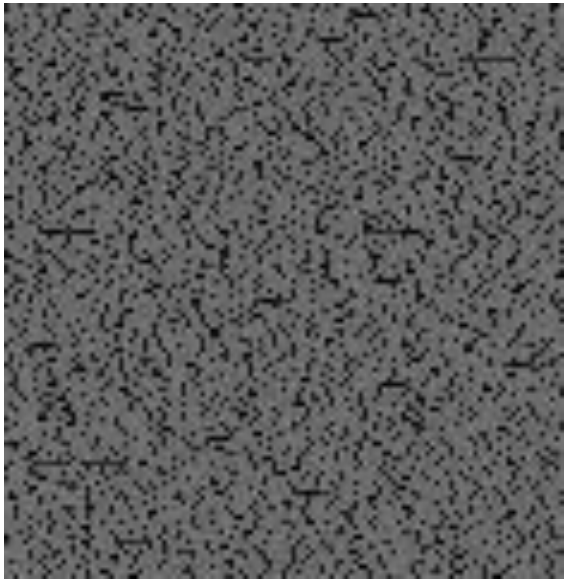


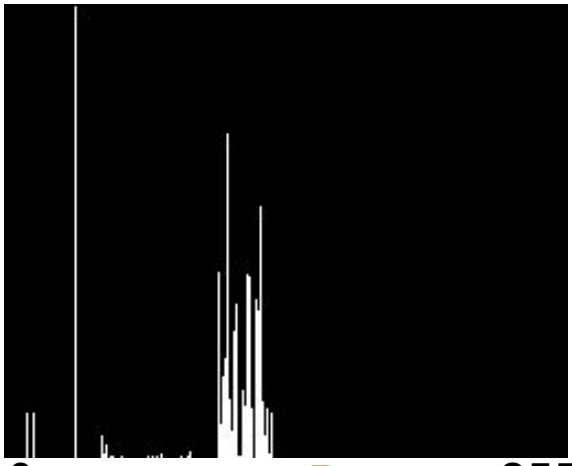


ASCII Encoded English Text



Sample





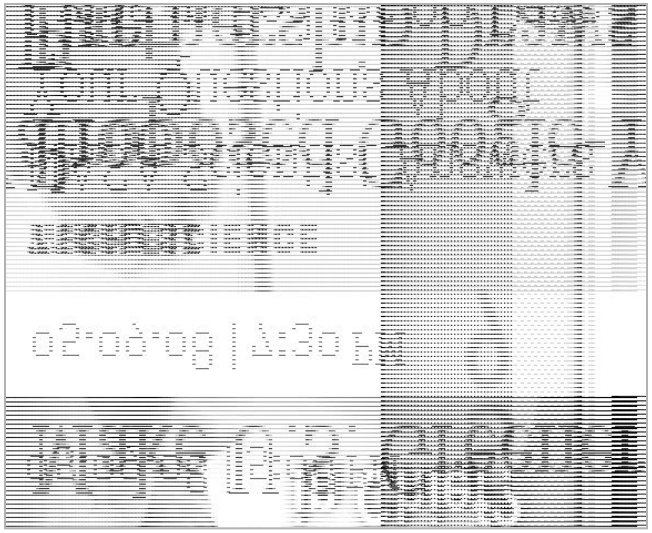
Images

Bitmap from .bmp

Bitmap from process memory

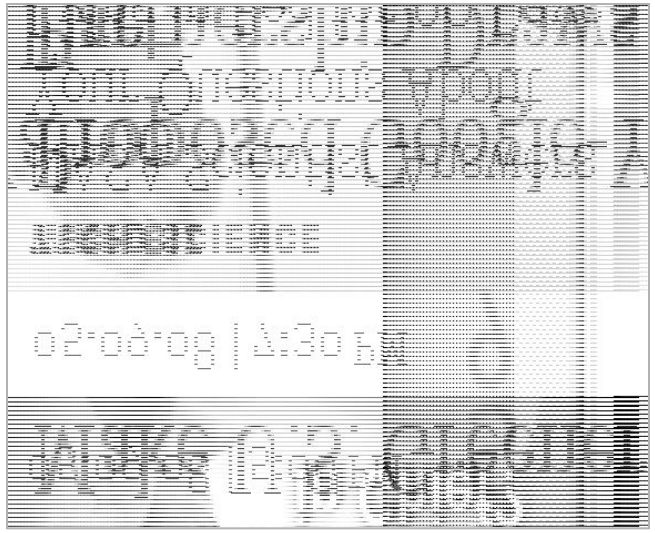
Bit Map

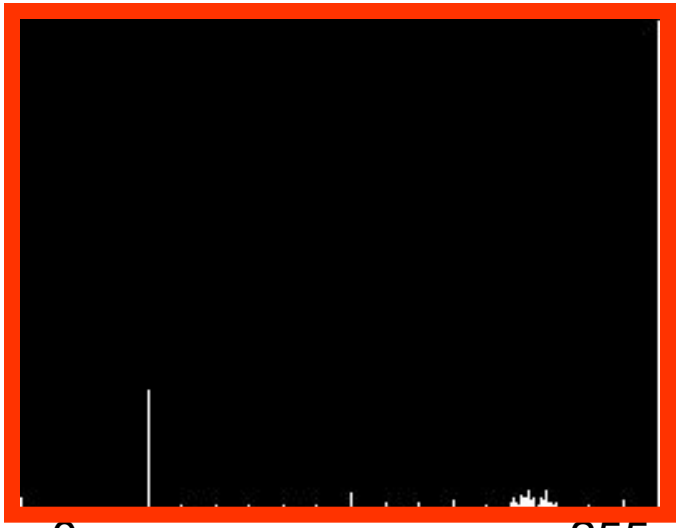
Sample



Bit Map

Sample



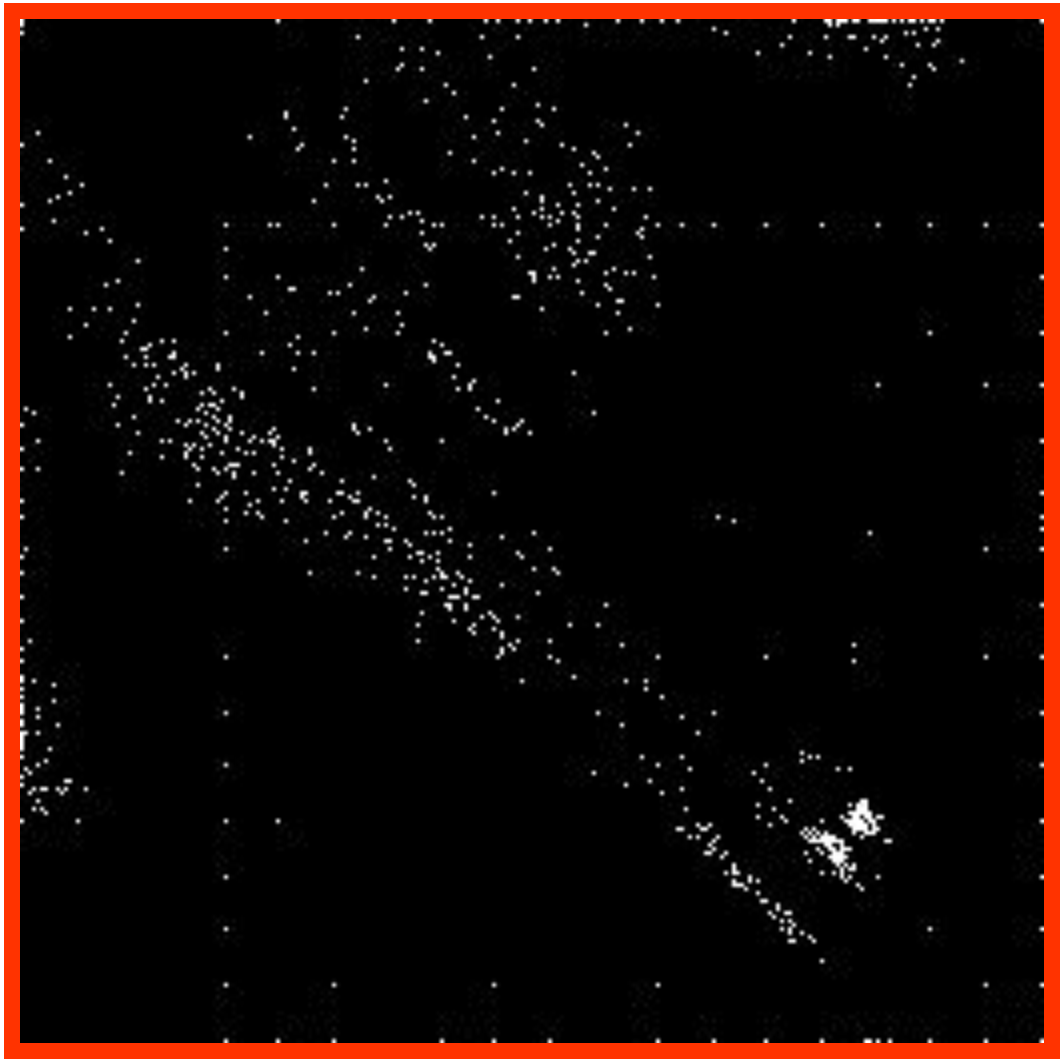


0 255

Bit Map

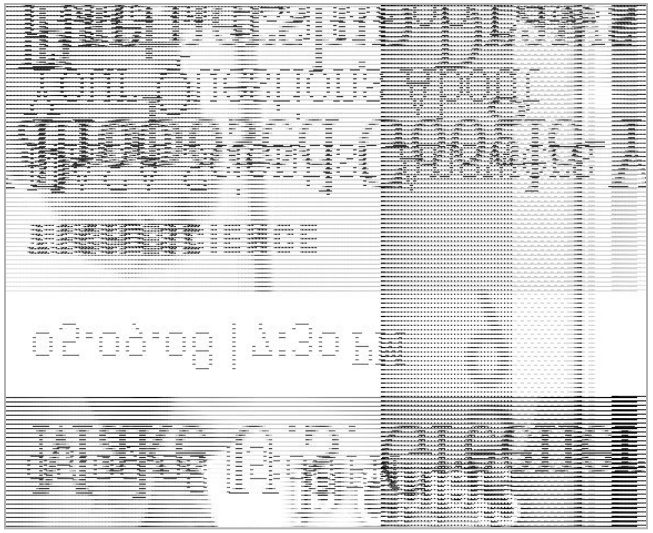
0

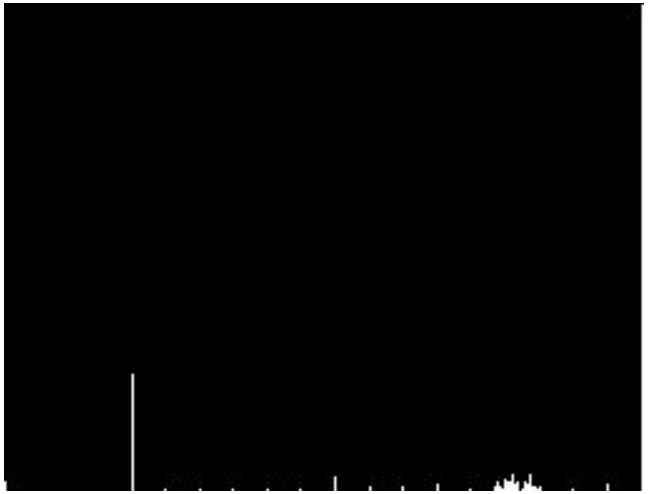
255



255

Sample





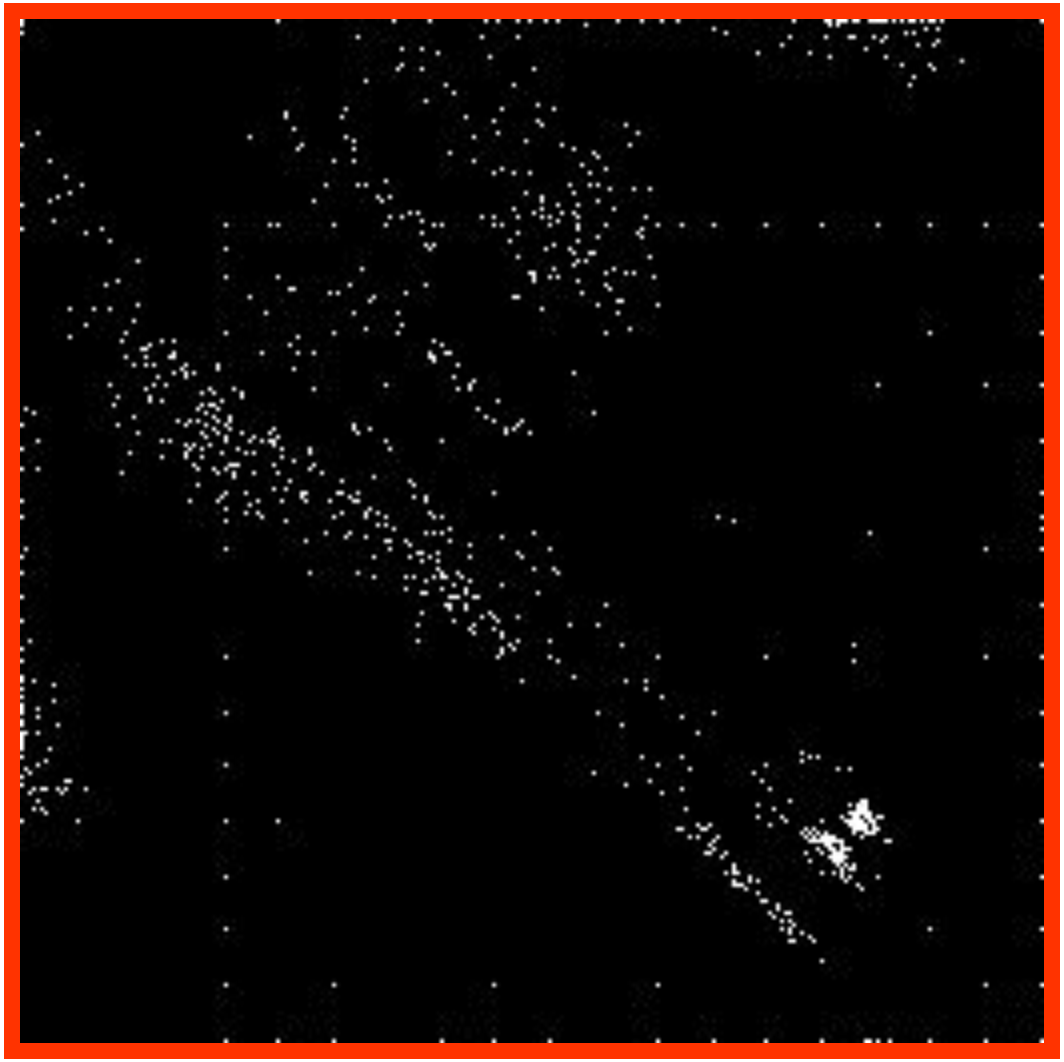
0

255

Bit Map

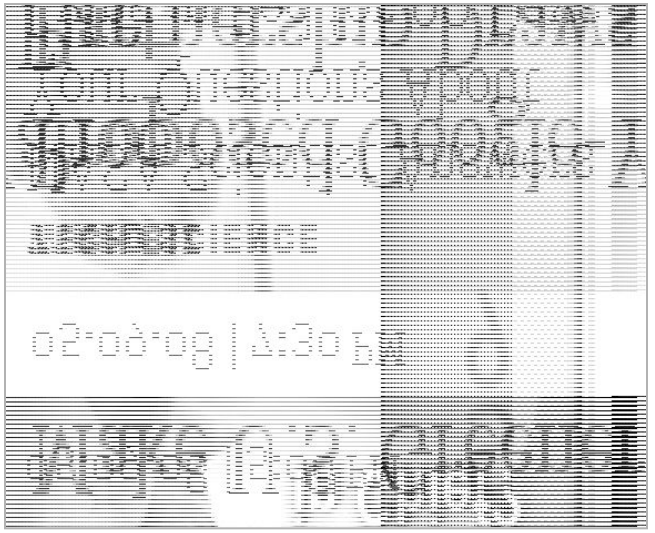
0

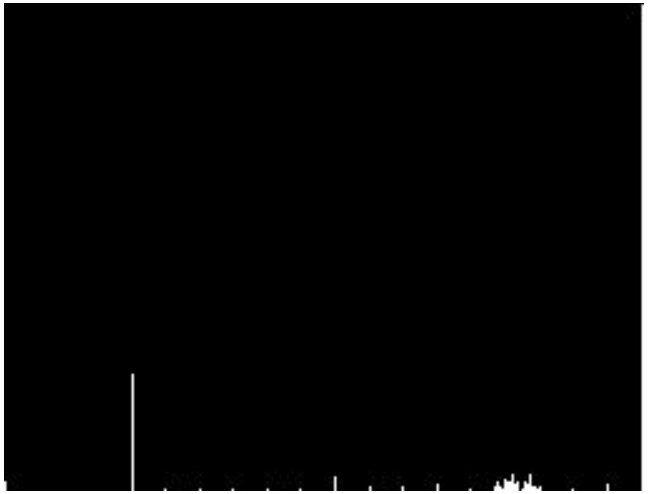
255



255

Sample





0

Demo 255

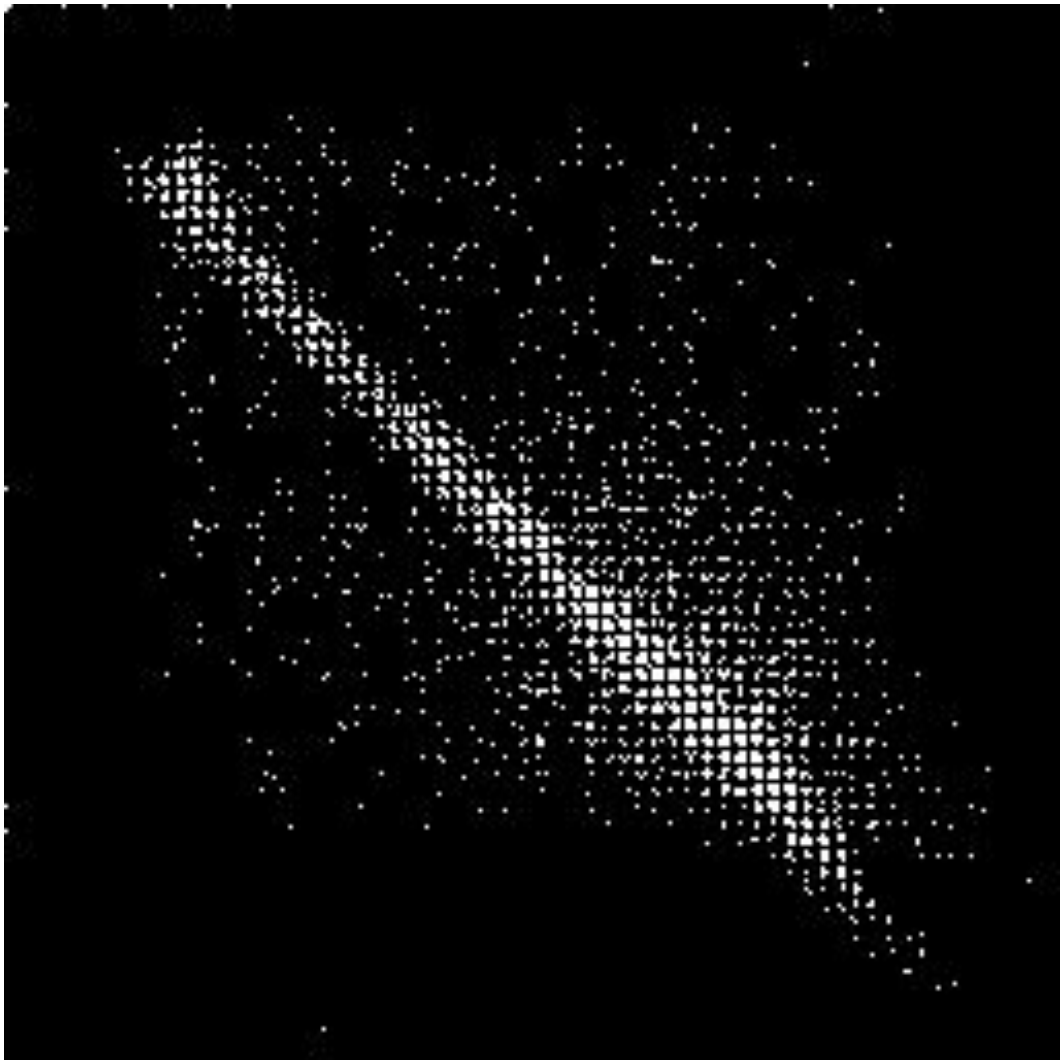
Steganography

See http://en.wikipedia.org/wiki/Steganography

Steganography

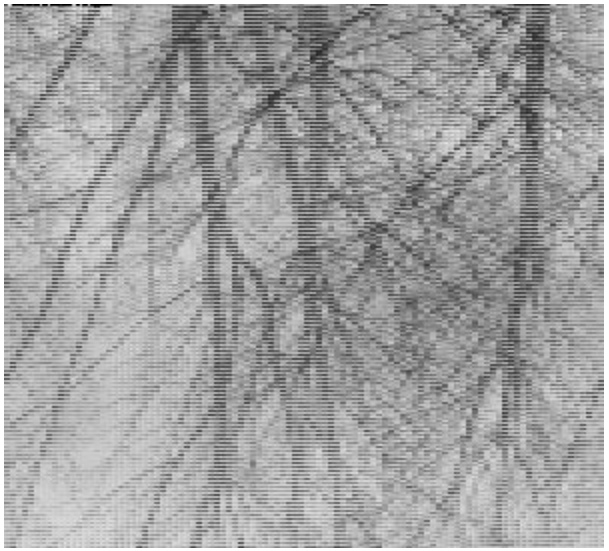
0

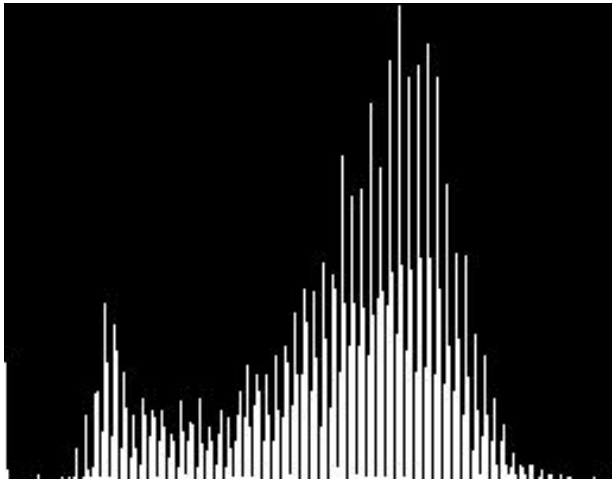
255



255

Sample

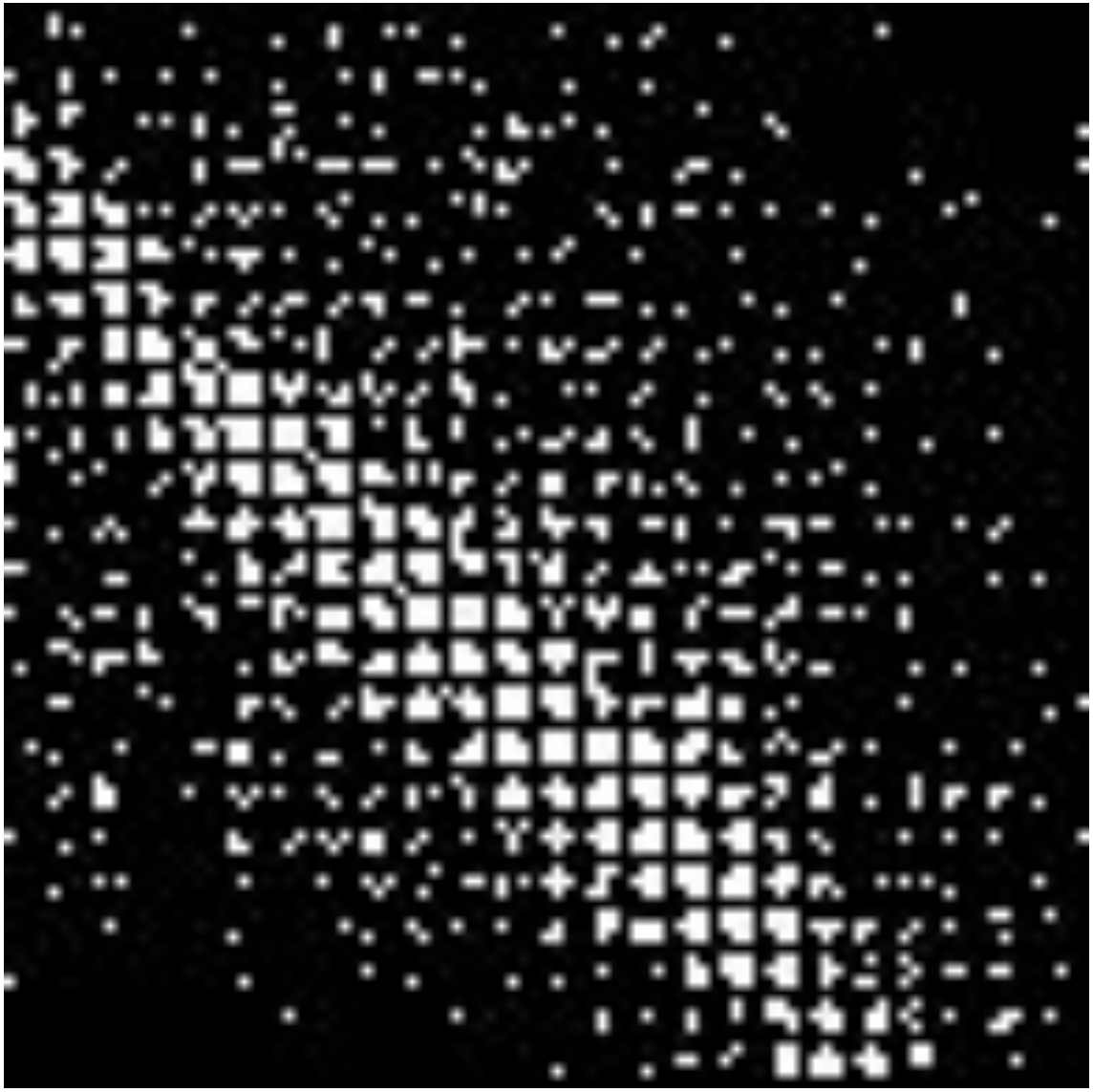




0

255

A Closer Look



Example .NET Image Formats

Format8bppIndexed Specifies that the format is 8 bits per pixel, indexed.

Format16bppGrayScale The pixel format is 16 bits per pixel. The color information specifies 65536 shades of gray.

Format16bppRgb565 Specifies that the format is 16 bits per pixel; 5 bits are used for the red component, 6 bits are used for the green component, and 5 bits are used for the blue component.

Format1bppIndexed Specifies that the pixel format is 1 bit per pixel and that it uses indexed color. The color table therefore has two colors in it.

Format24bppRgb Specifies that the format is 24 bits per pixel; 8 bits each are used for the red, green, and blue components.

Format32bppArgb Specifies that the format is 32 bits per pixel; 8 bits each are used for the alpha, red, green, and blue components.

Format48bppRgb Specifies that the format is 48 bits per pixel; 16 bits each are used for the red, green, and blue components.

Format64bppArgb Specifies that the format is 64 bits per pixel; 16 bits each are used for the alpha, red, green, and blue components.

Audio

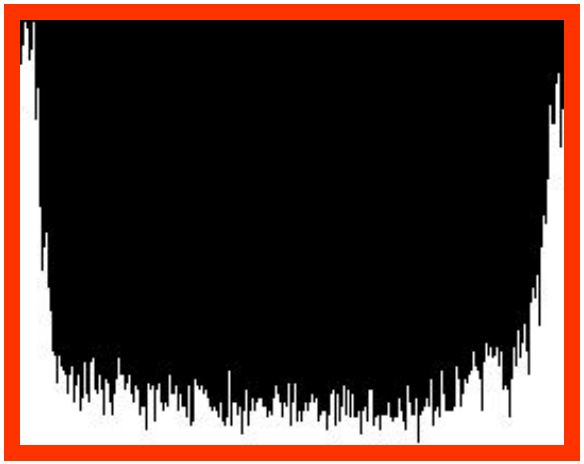
44.1 KHz, 16 bit per sample, PCM encoded audio (.wav)

Audio (.wav)

Sample

Audio (.wav)

Sample

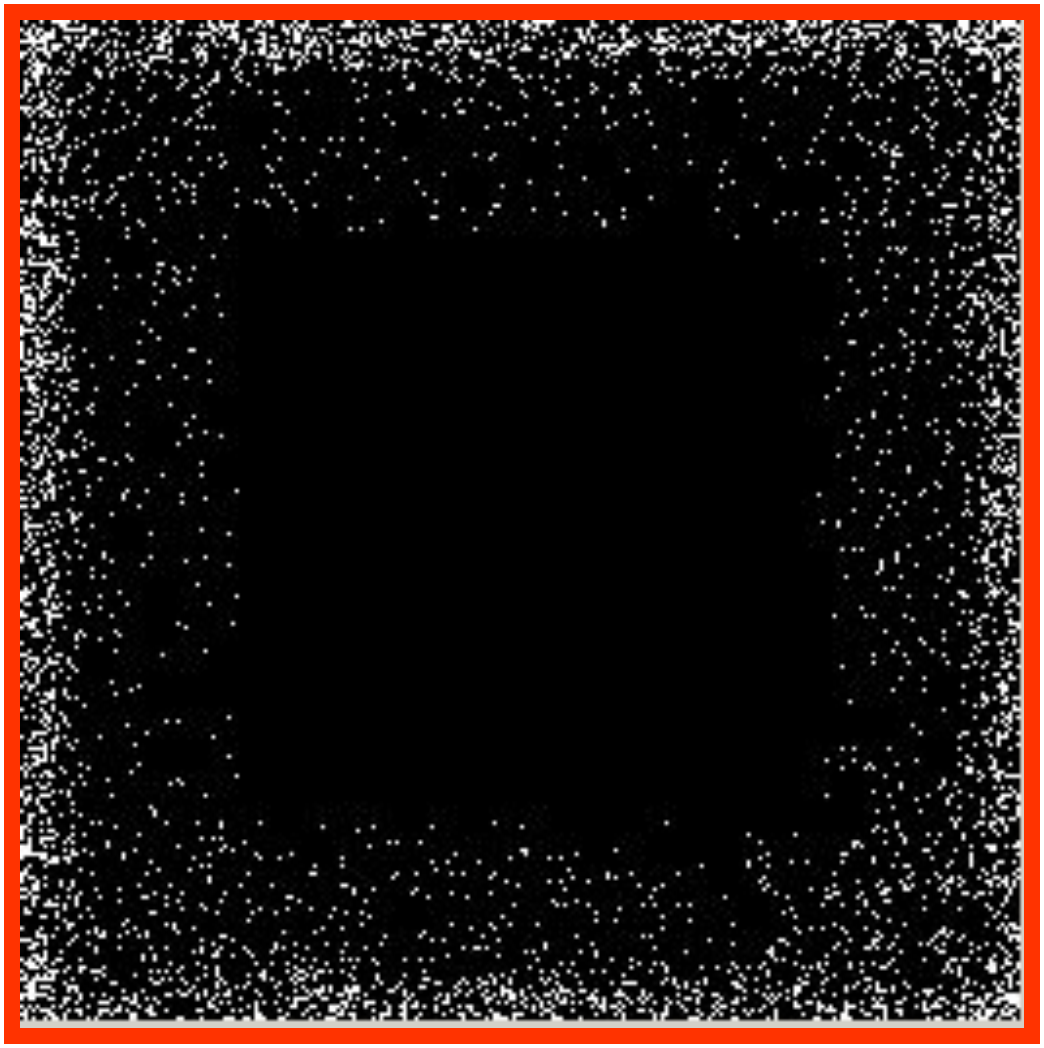


0 255

Audio (.wav)

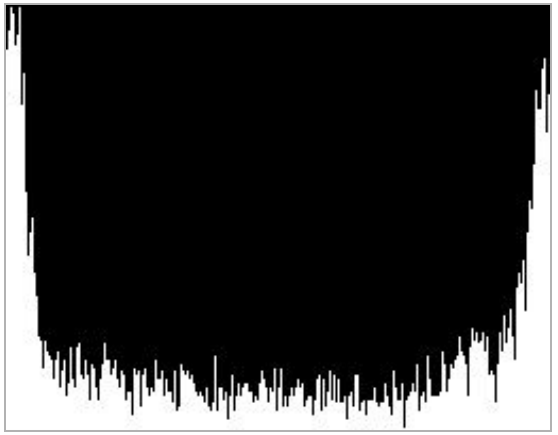
0

255



255

Sample



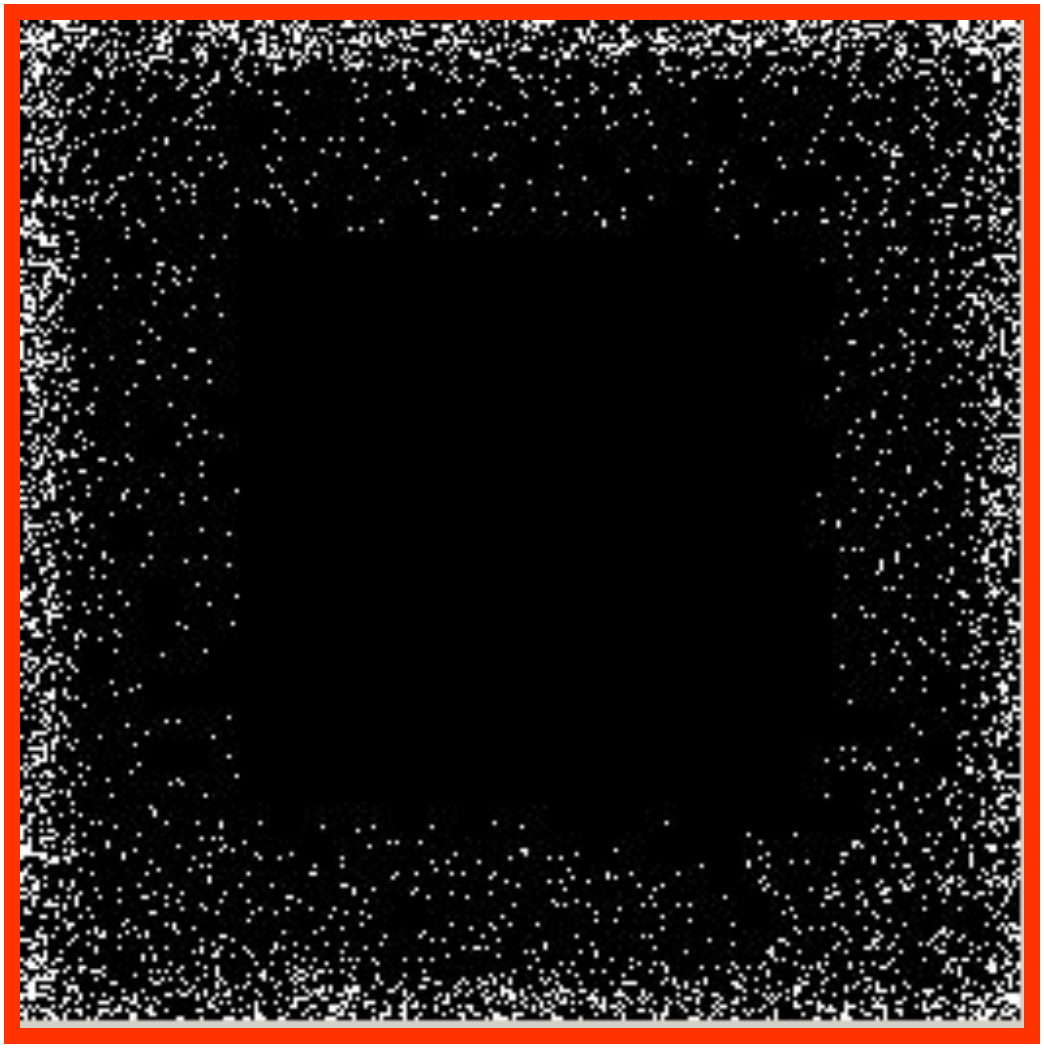
0

255

Audio (.wav)

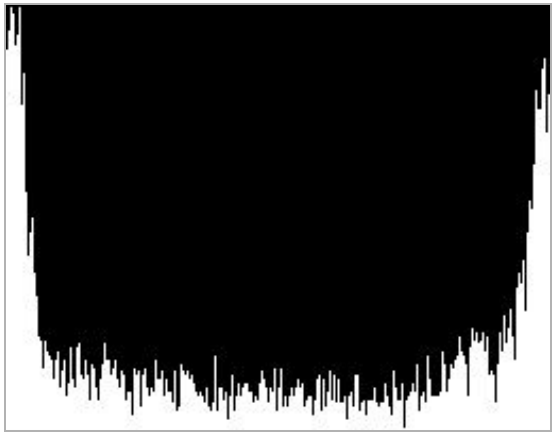
0

255



255

Sample



0

Demo

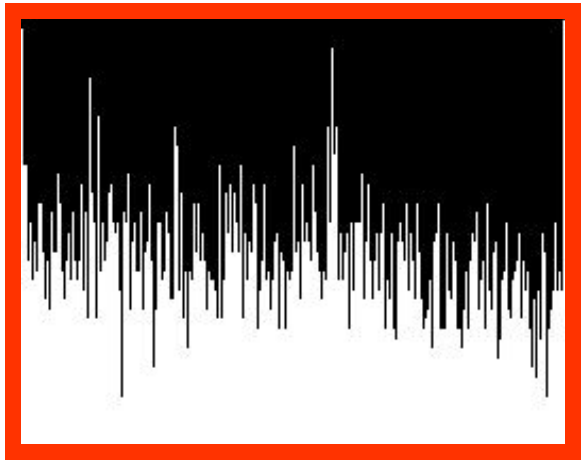
255

Compressed Audio

Sample

Compressed Audio

Sample

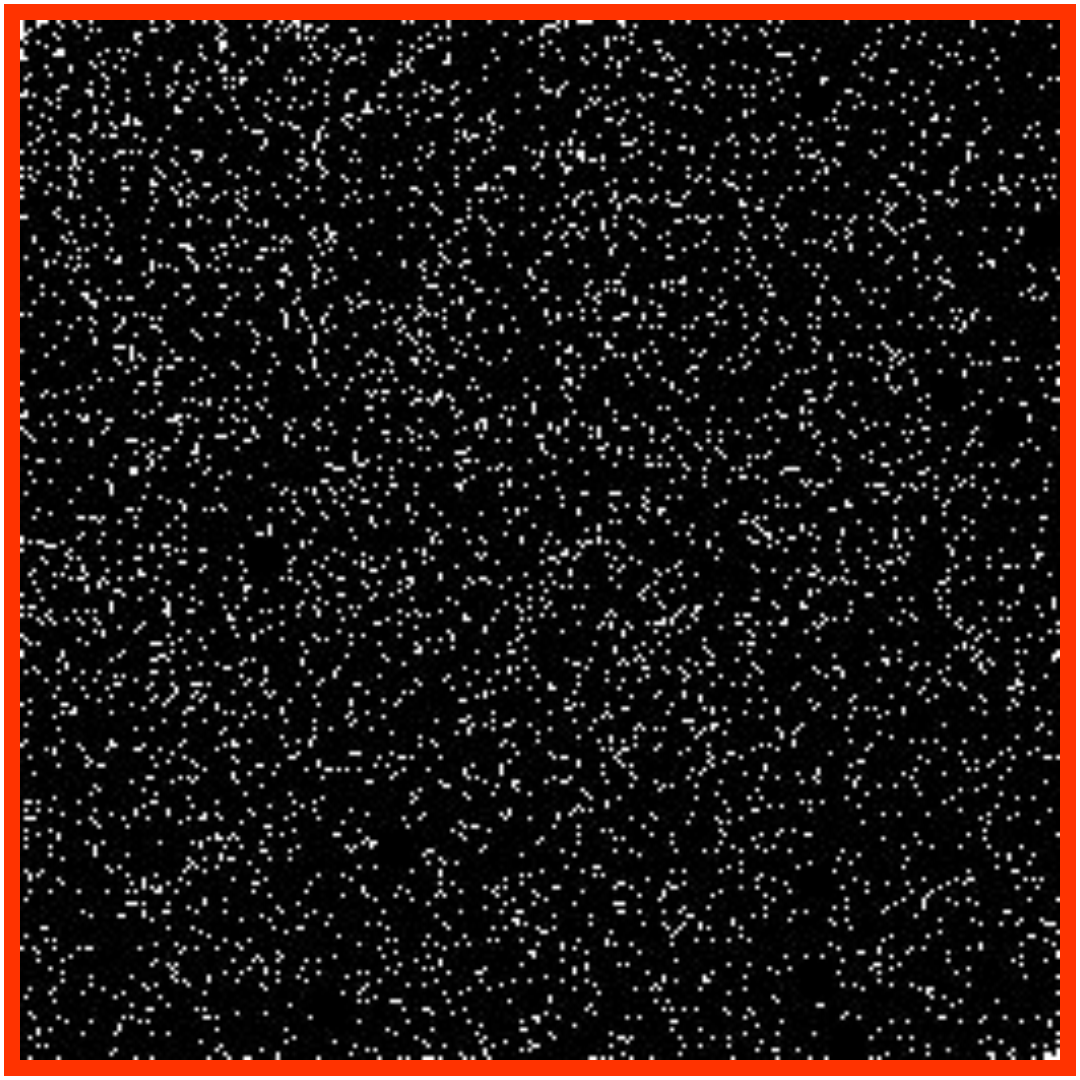


0 255

Compressed Audio

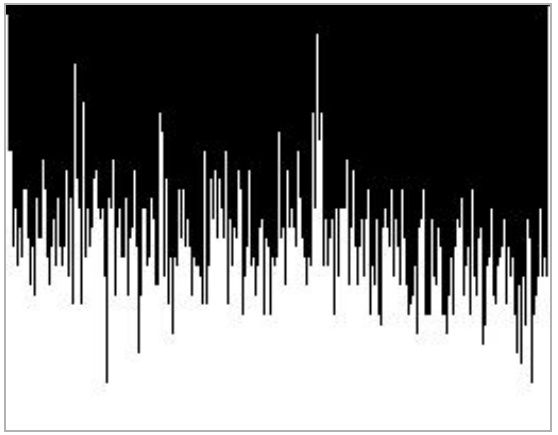
0

255



255

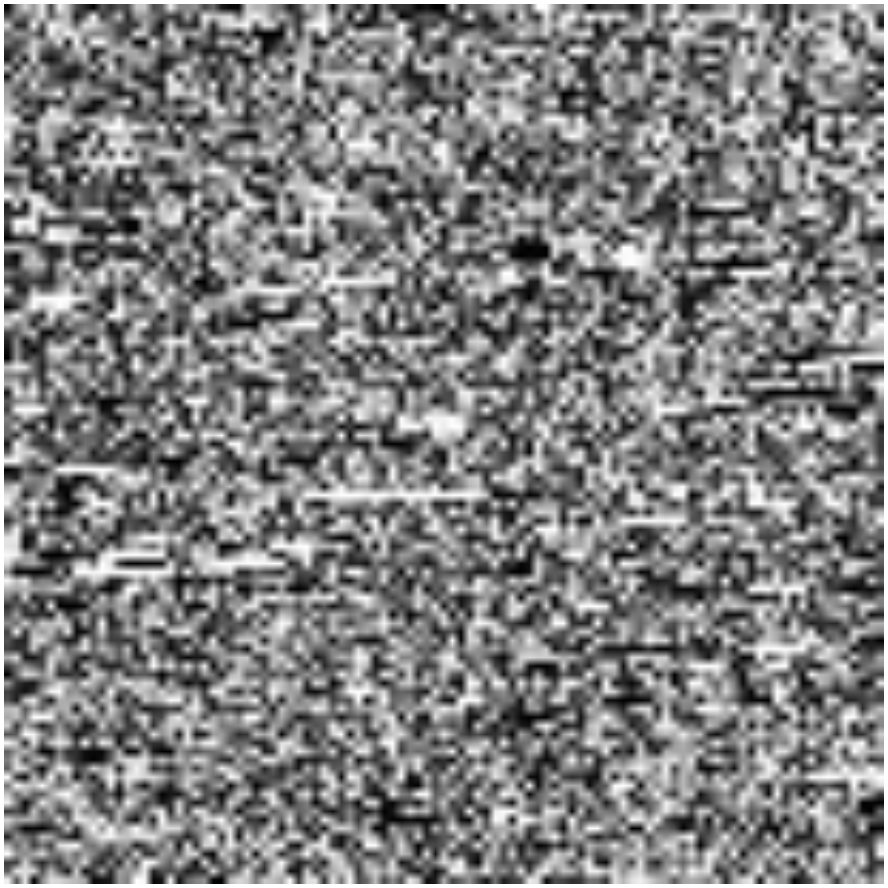
Sample



0

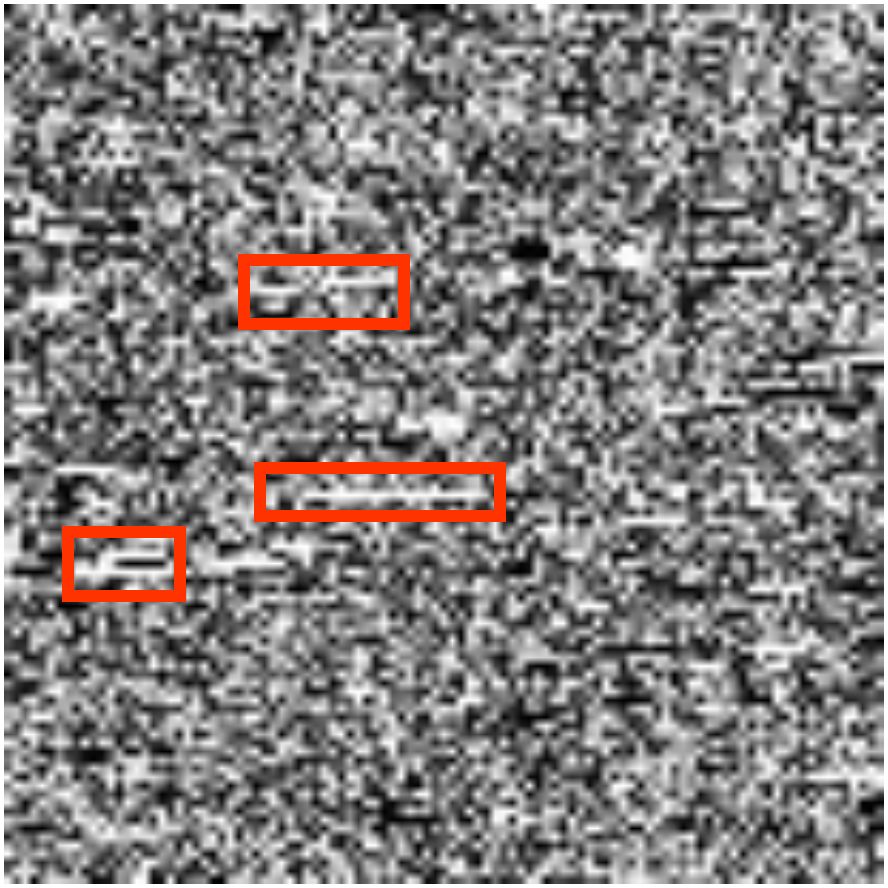
255

A Closer Look…



MPEG-1 layer 3 - 128kbit, 44100Hz (.mp3)

A Closer Look…



MPEG-1 layer 3 - 128kbit, 44100Hz (.mp3)

Dot Plots

- Jonathan Helfman's "Dotplot Patterns: A Literal Look at Pattern Languages."
- Dan Kaminsky, CCC & BH 2006

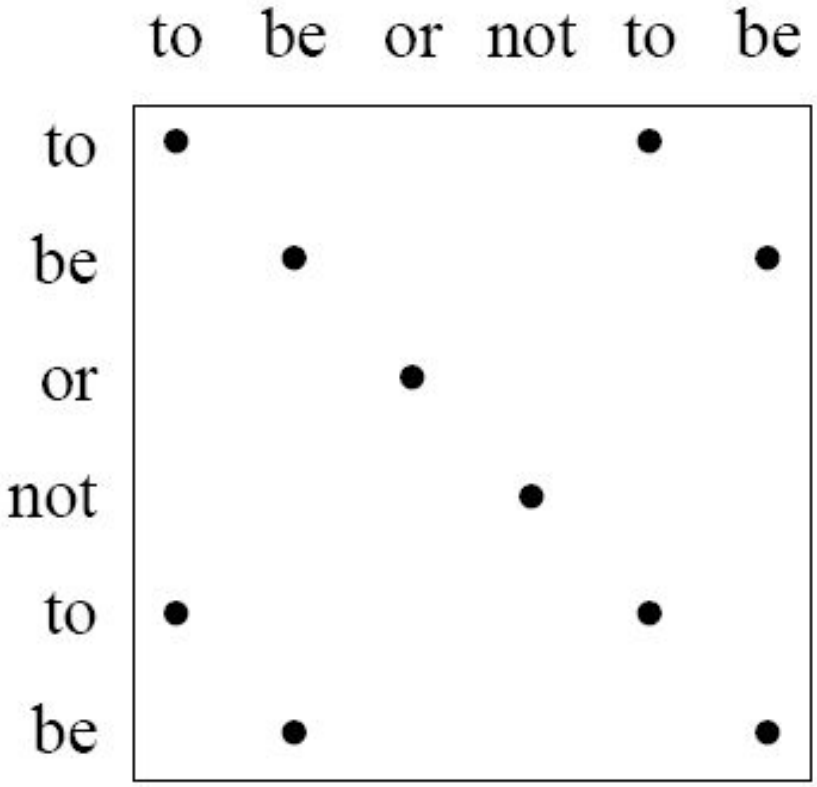
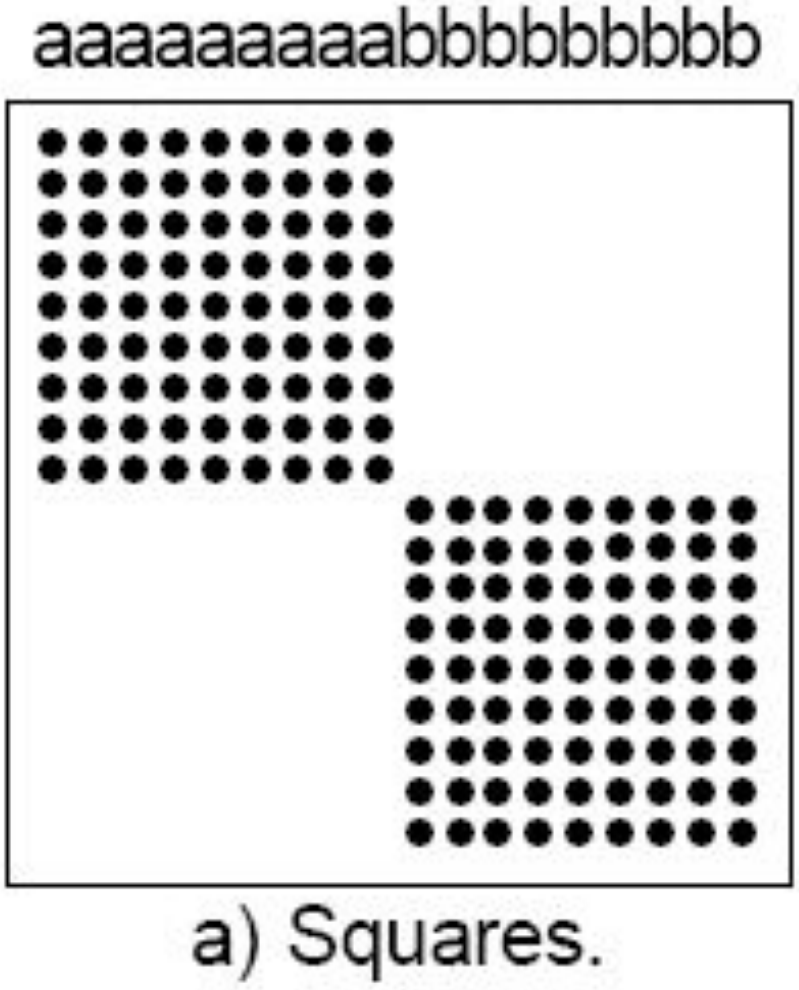
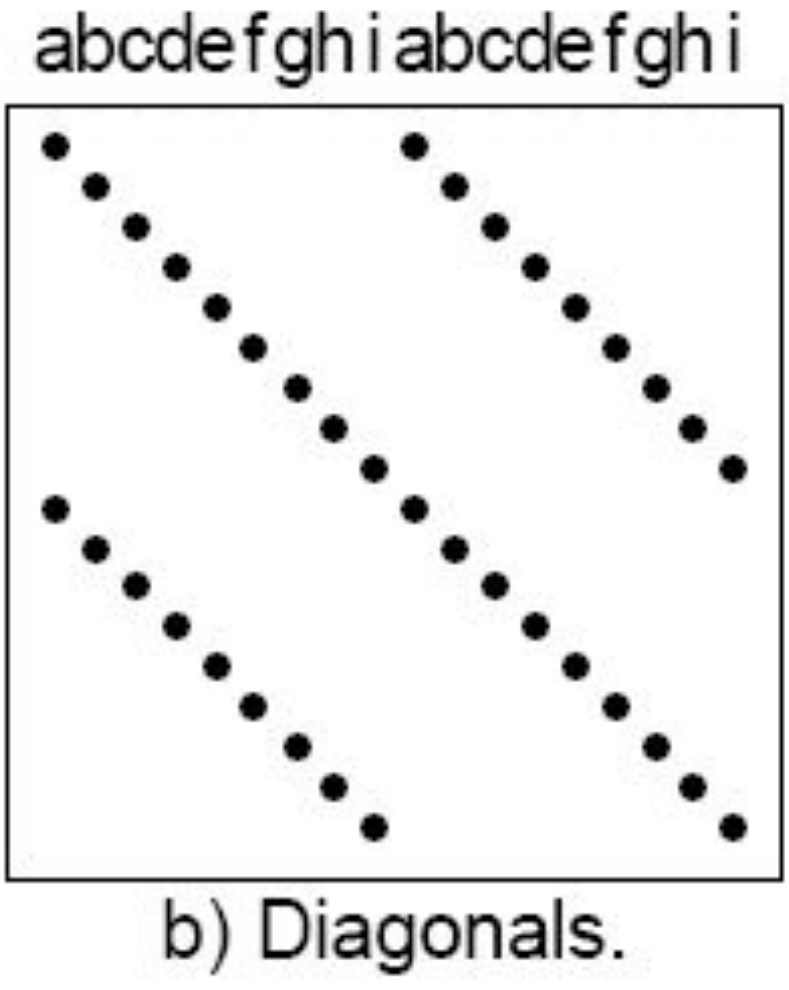


Figure 2: Six words of Shakespeare.

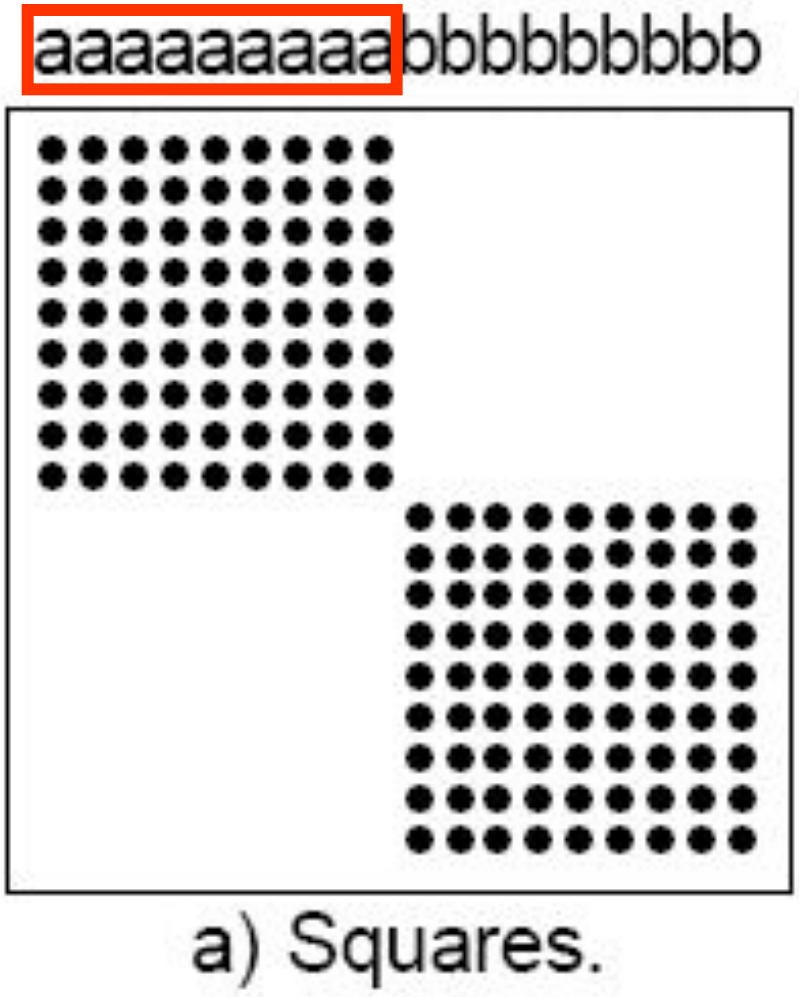
DotPlot Examples

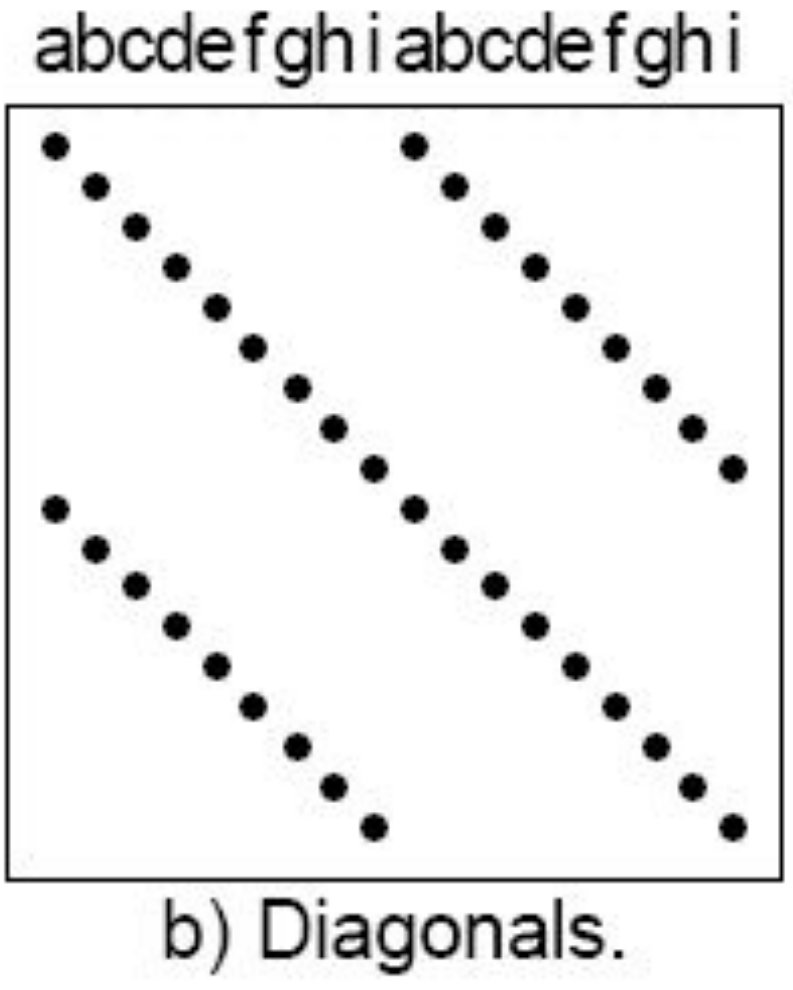




Images: Jonathan Helfman, “Dotplot Patterns: A Literal Look at Pattern Languages.”

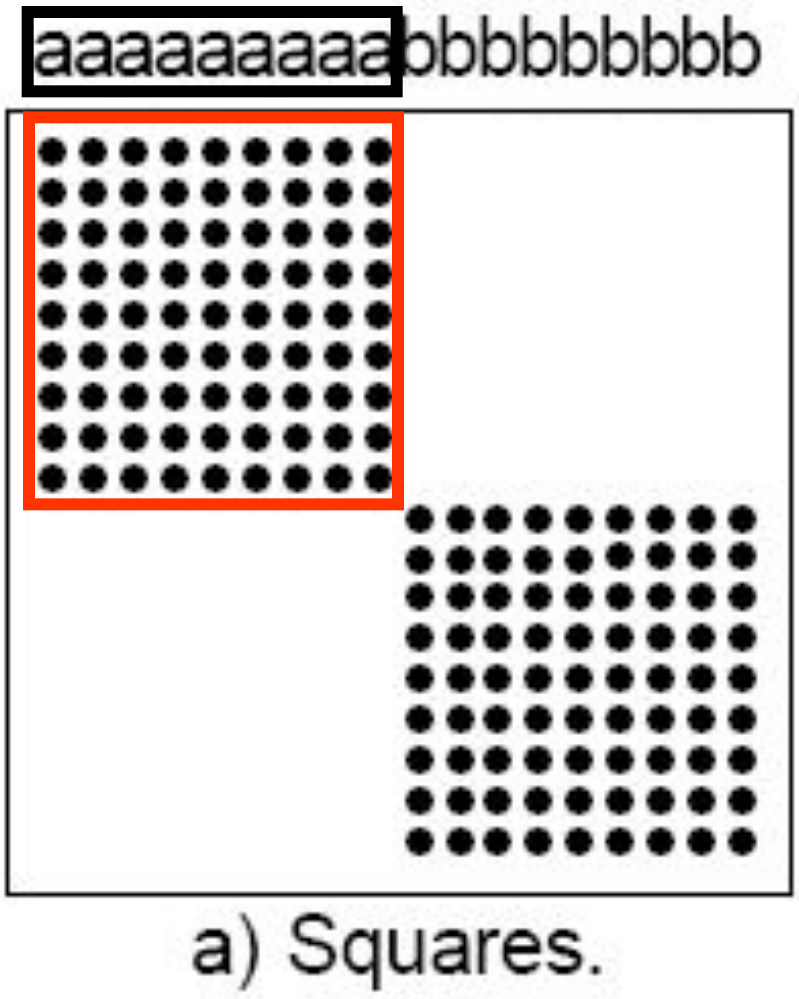
DotPlot Examples



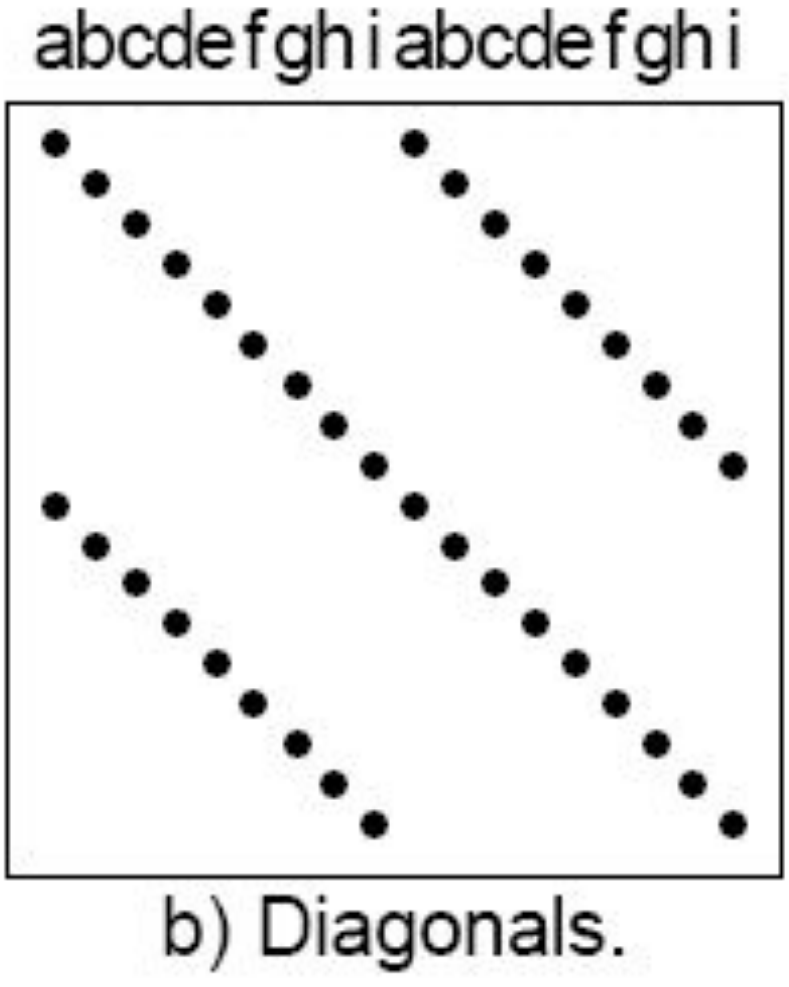


Images: Jonathan Helfman, “Dotplot Patterns: A Literal Look at Pattern Languages.”

DotPlot Examples



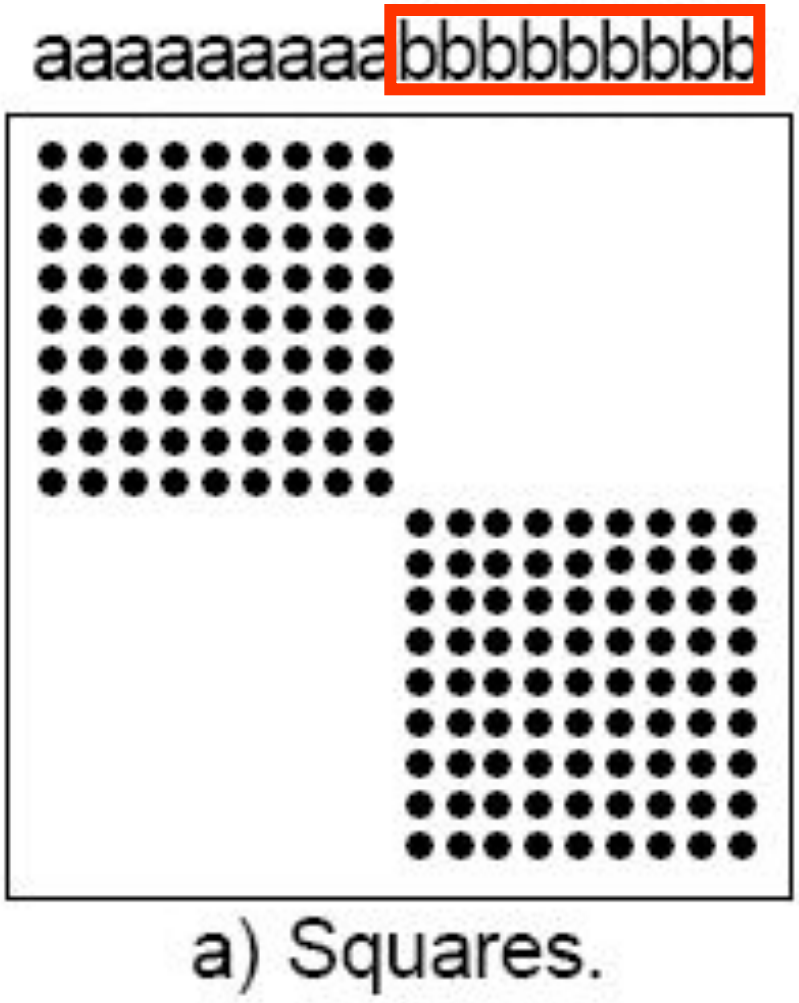
a) Squares.



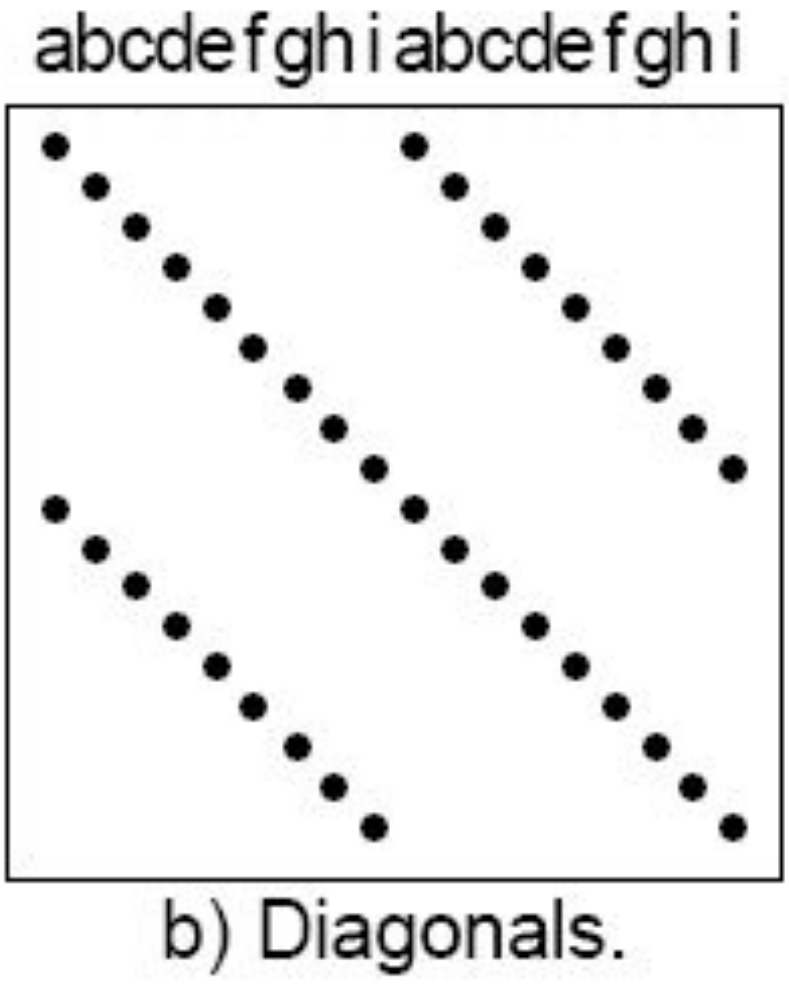
b) Diagonals.

Images: Jonathan Helfman, “Dotplot Patterns: A Literal Look at Pattern Languages.”

DotPlot Examples



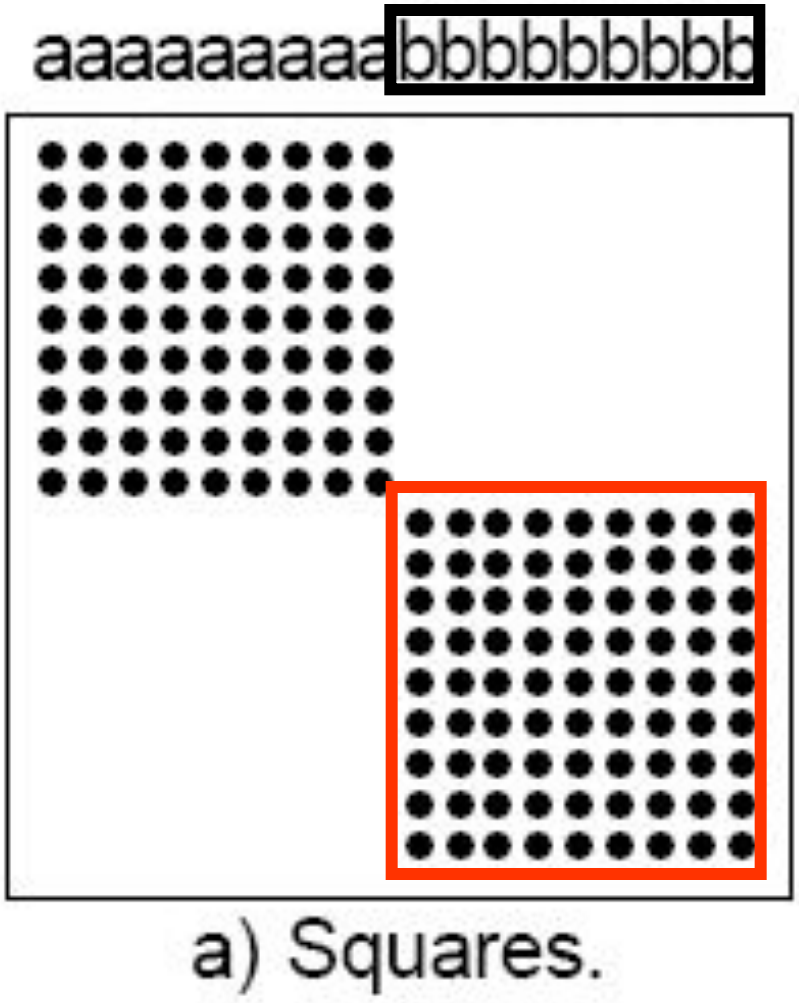
a) Squares.



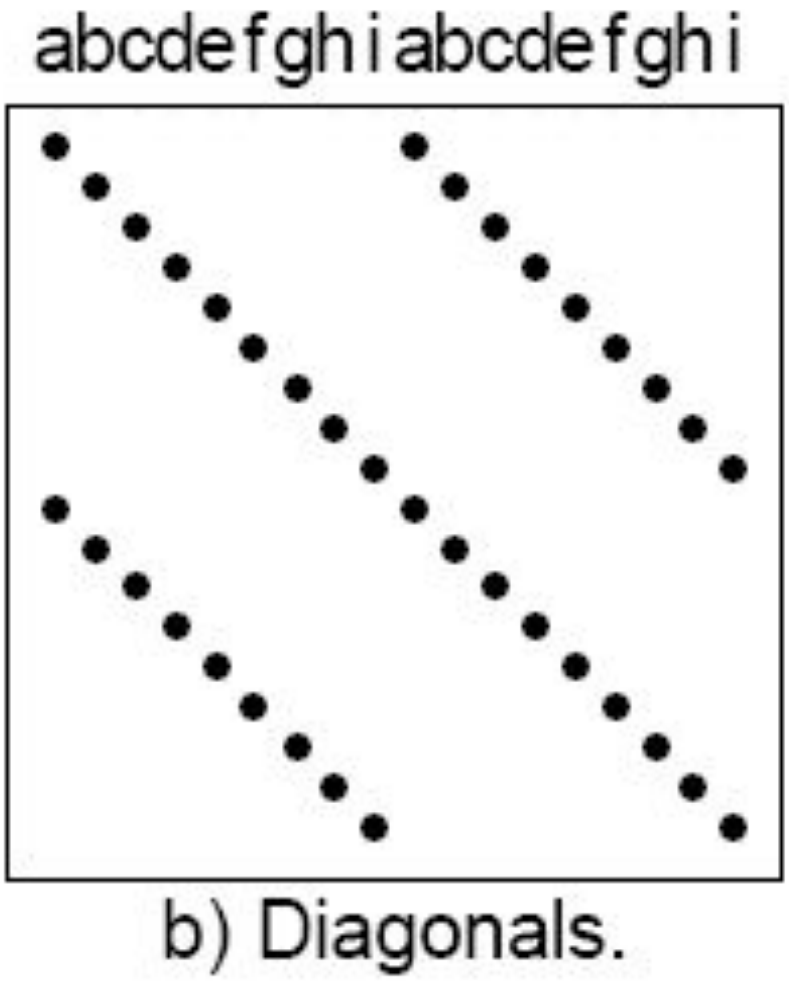
b) Diagonals.

Images: Jonathan Helfman, “Dotplot Patterns: A Literal Look at Pattern Languages.”

DotPlot Examples



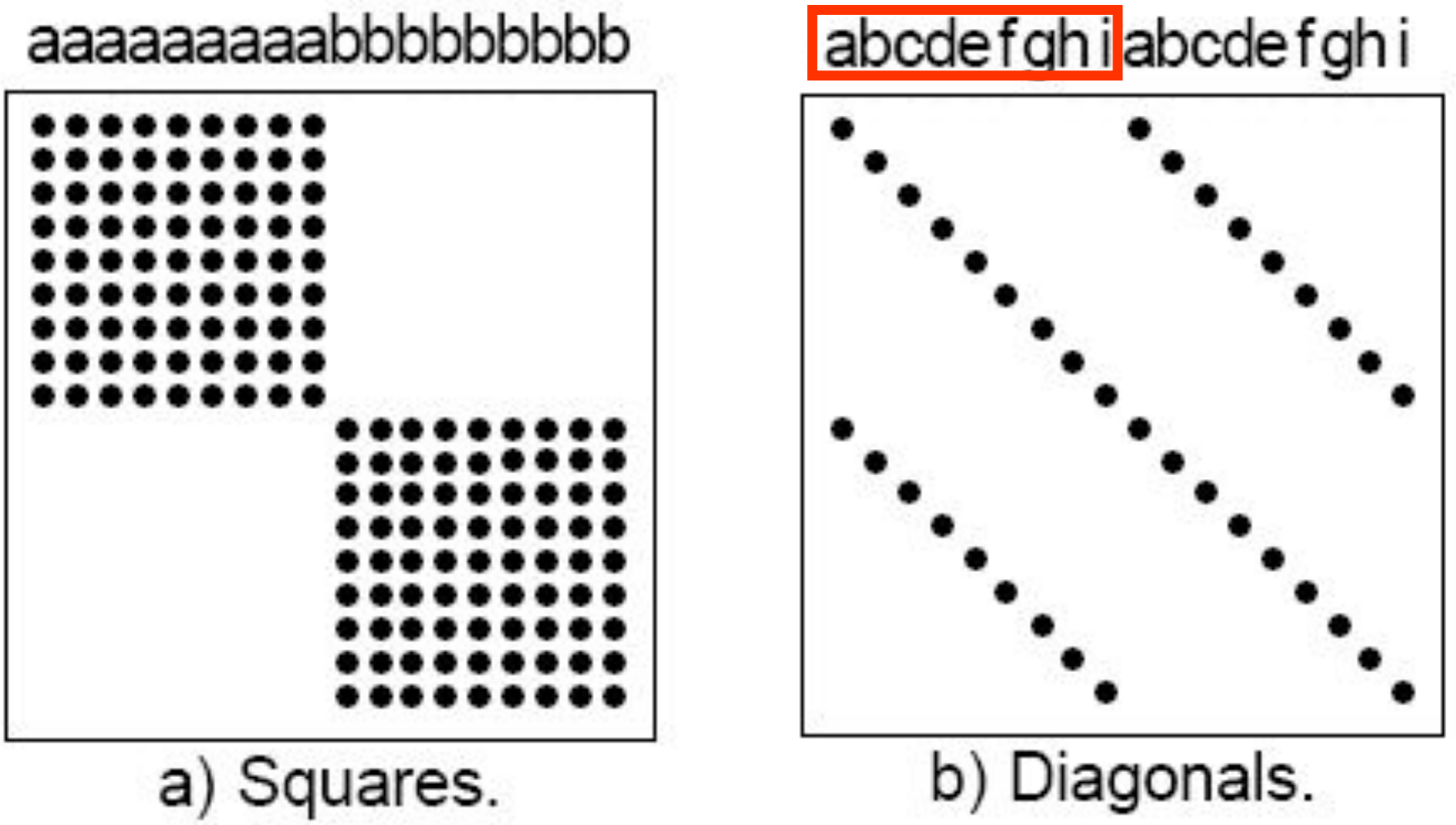
a) Squares.



b) Diagonals.

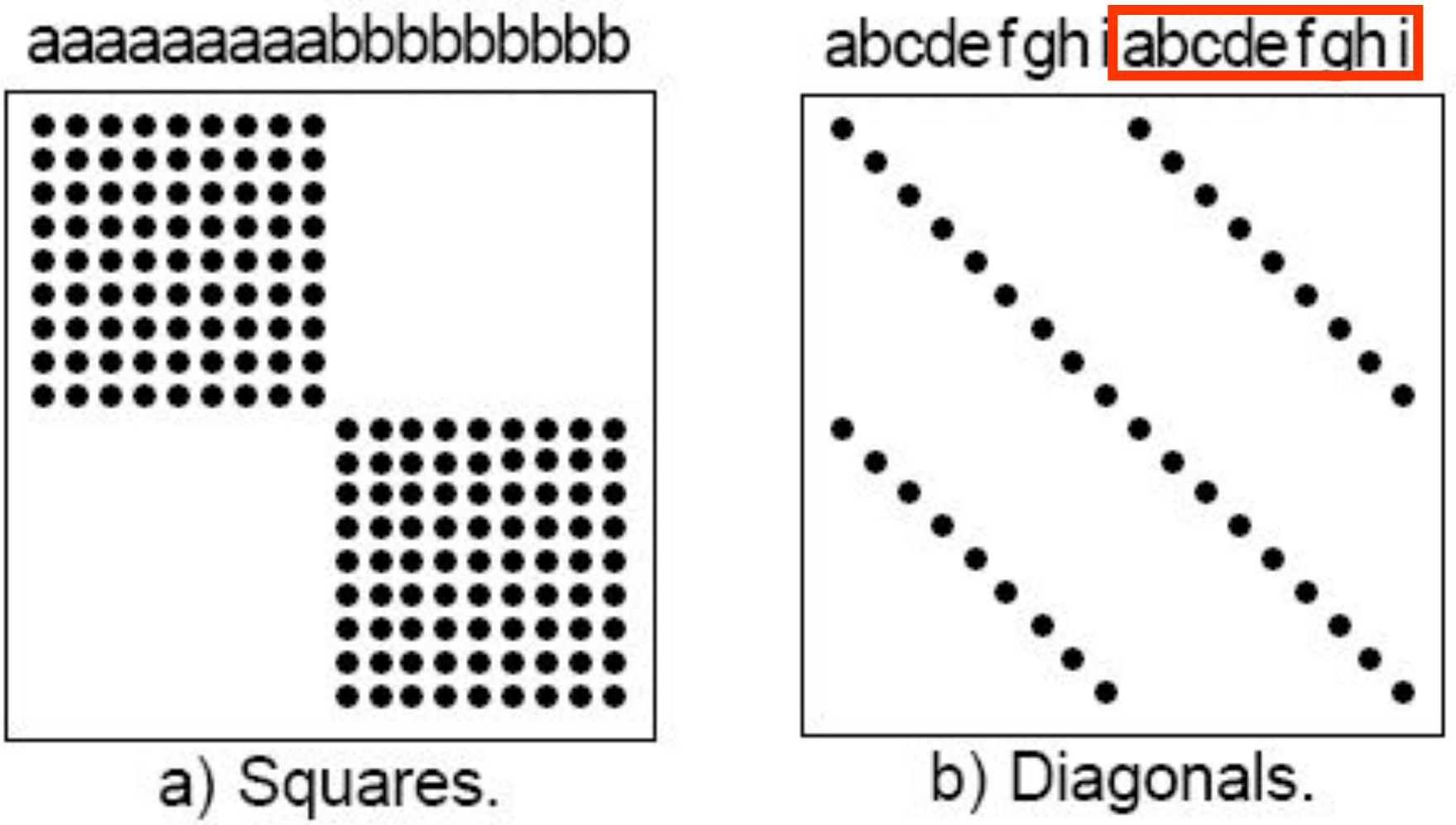
Images: Jonathan Helfman, “Dotplot Patterns: A Literal Look at Pattern Languages.”

DotPlot Examples



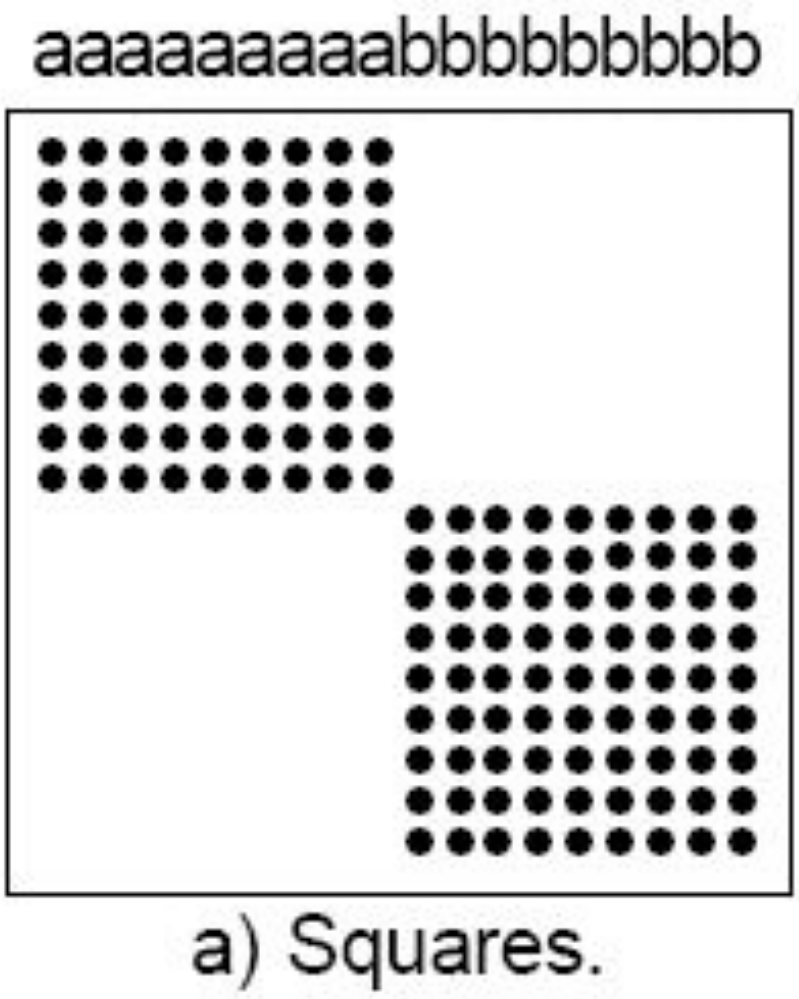
Images: Jonathan Helfman, “Dotplot Patterns: A Literal Look at Pattern Languages.”

DotPlot Examples

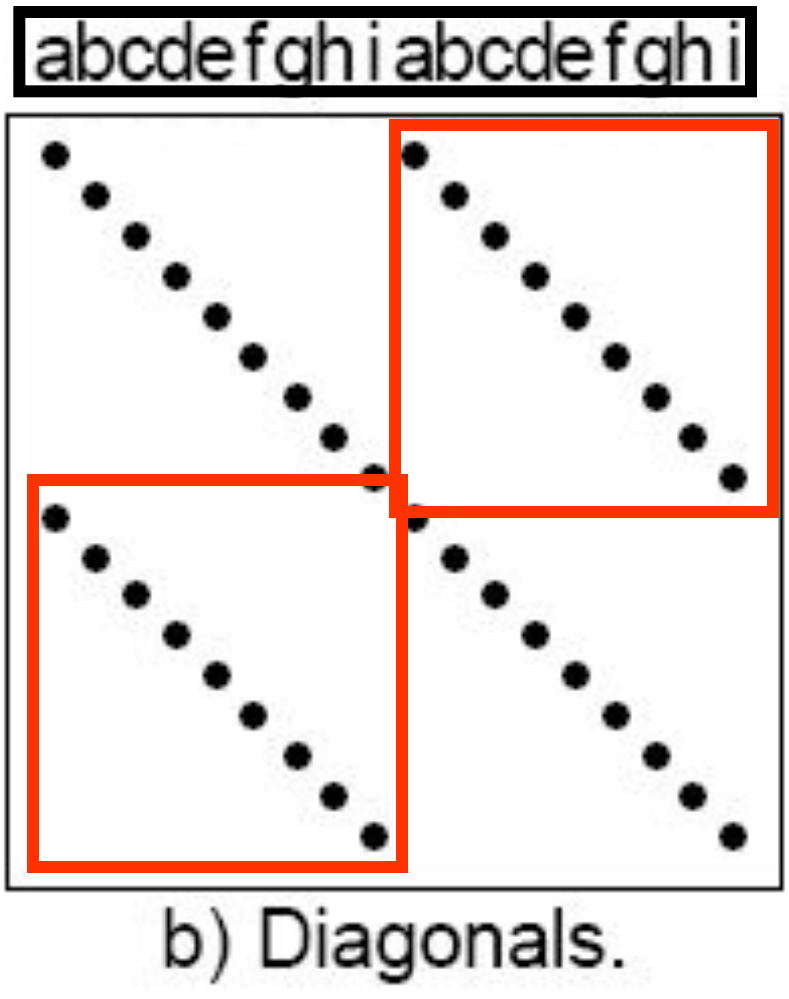


Images: Jonathan Helfman, “Dotplot Patterns: A Literal Look at Pattern Languages.”

DotPlot Examples



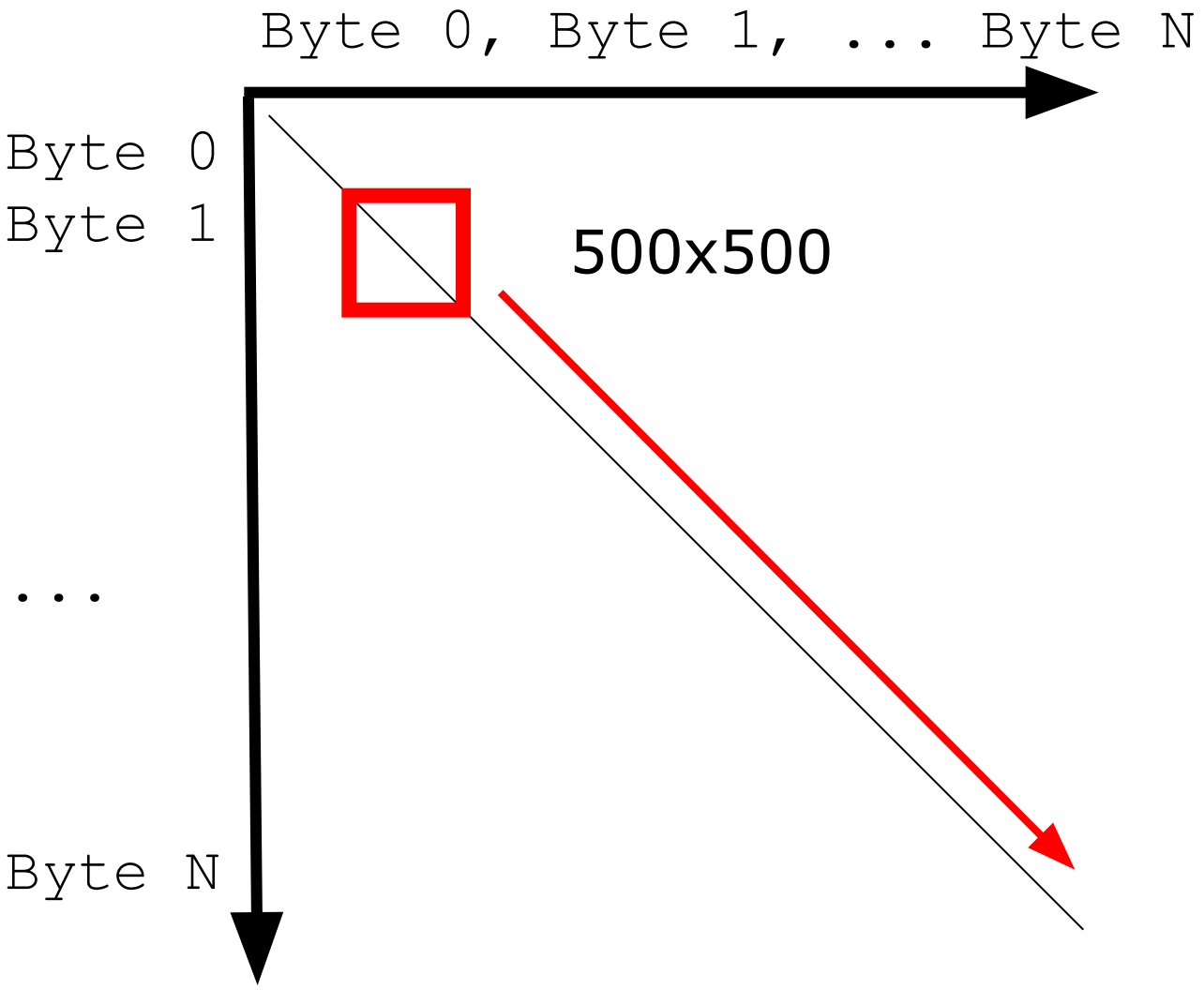
a) Squares.



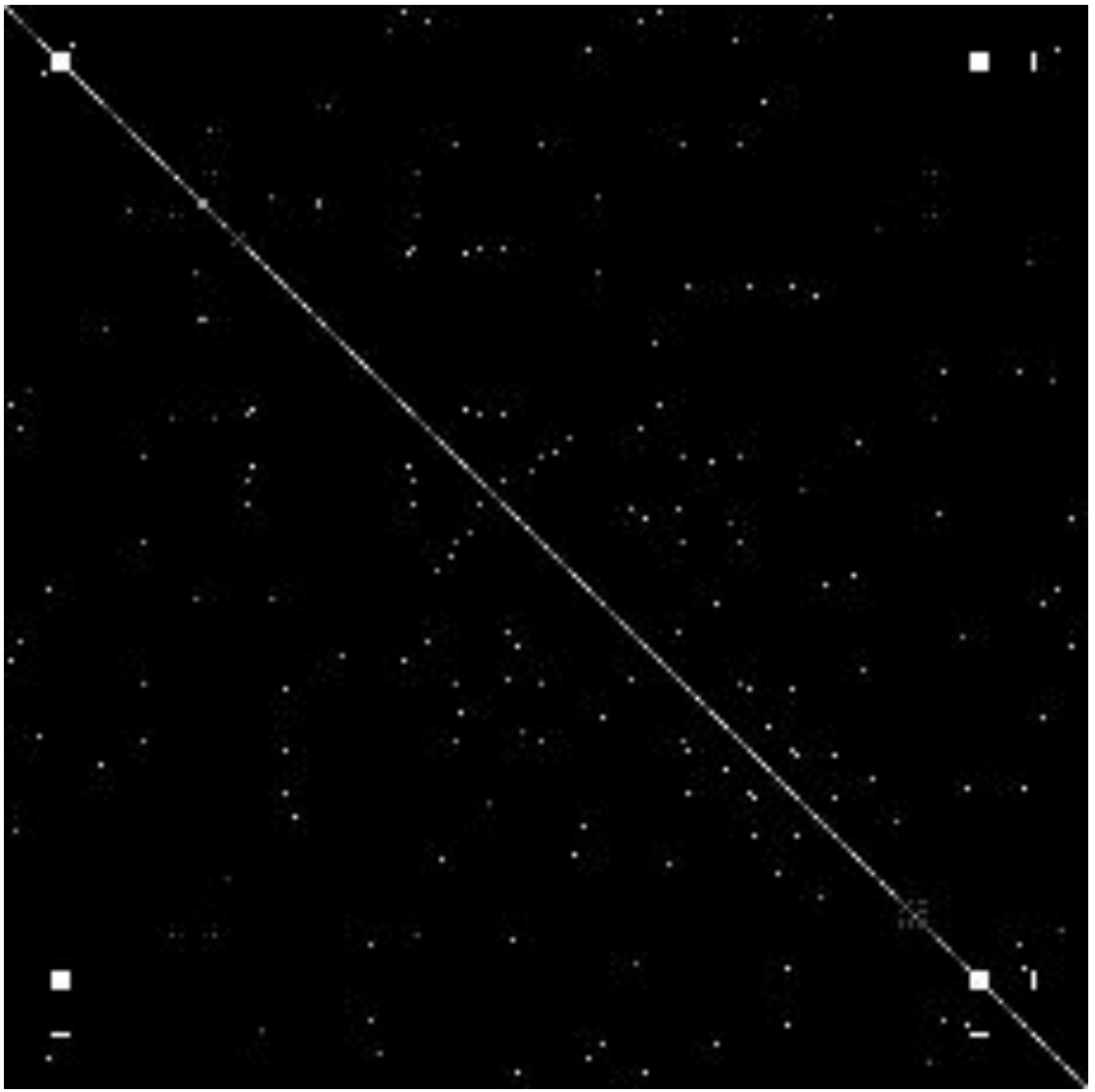
b) Diagonals.

Images: Jonathan Helfman, “Dotplot Patterns: A Literal Look at Pattern Languages.”

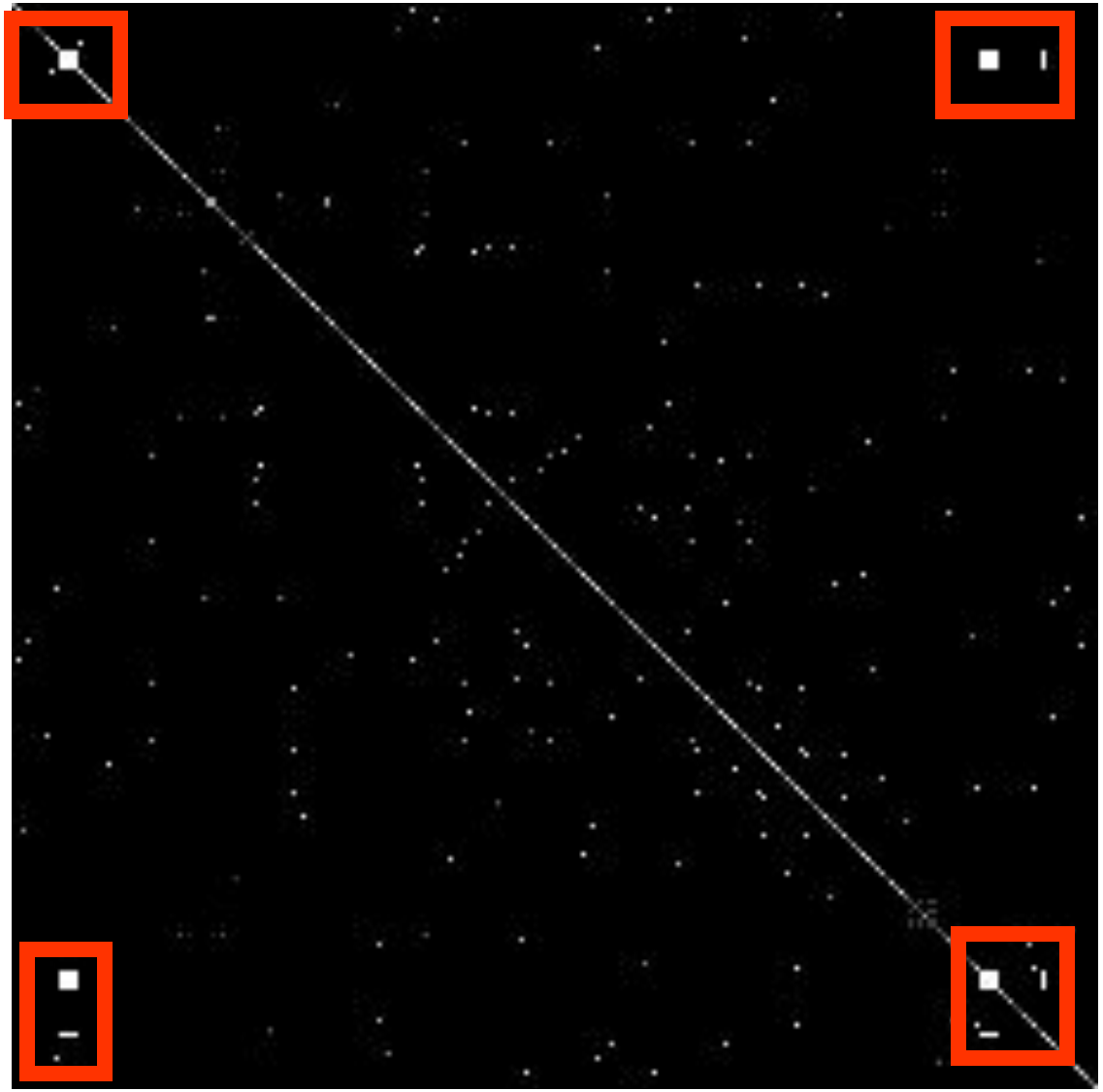
Sliding Window DotPlot



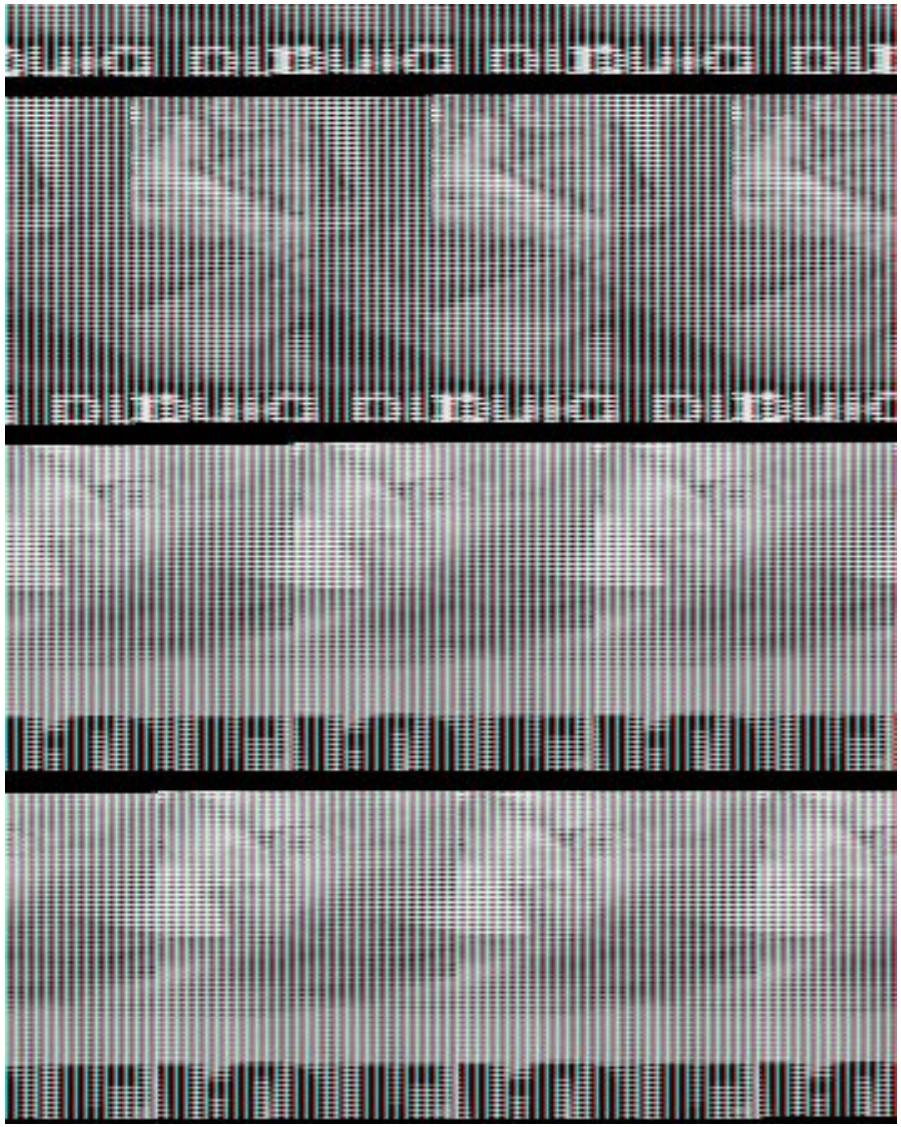
Dot Plot



Dot Plot



Video



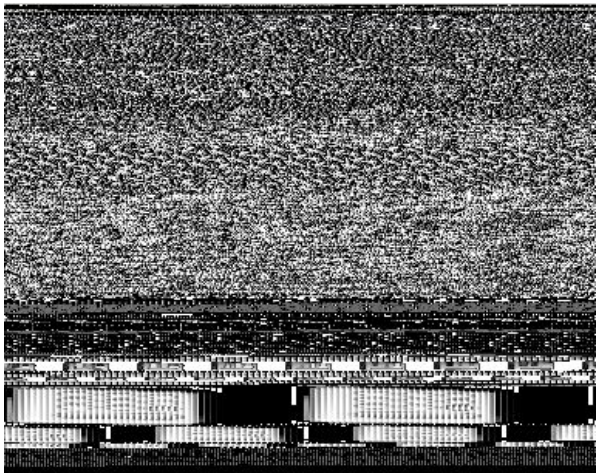
Full Frame .avi

Compressed AVI

Key Frame

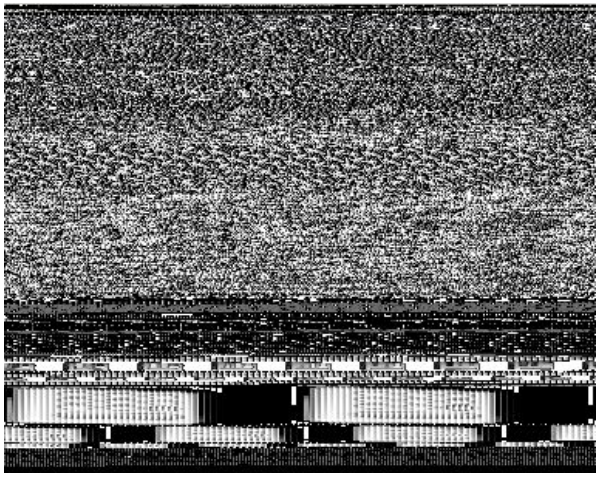
Key Frame

Windows PE

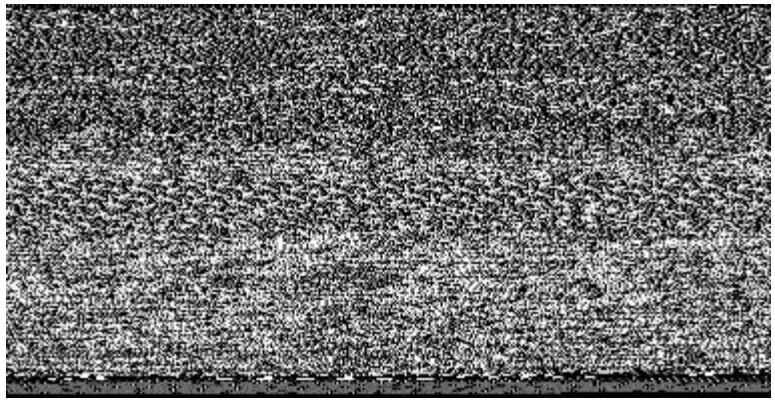


calc.exe

Windows PE



calc.exe

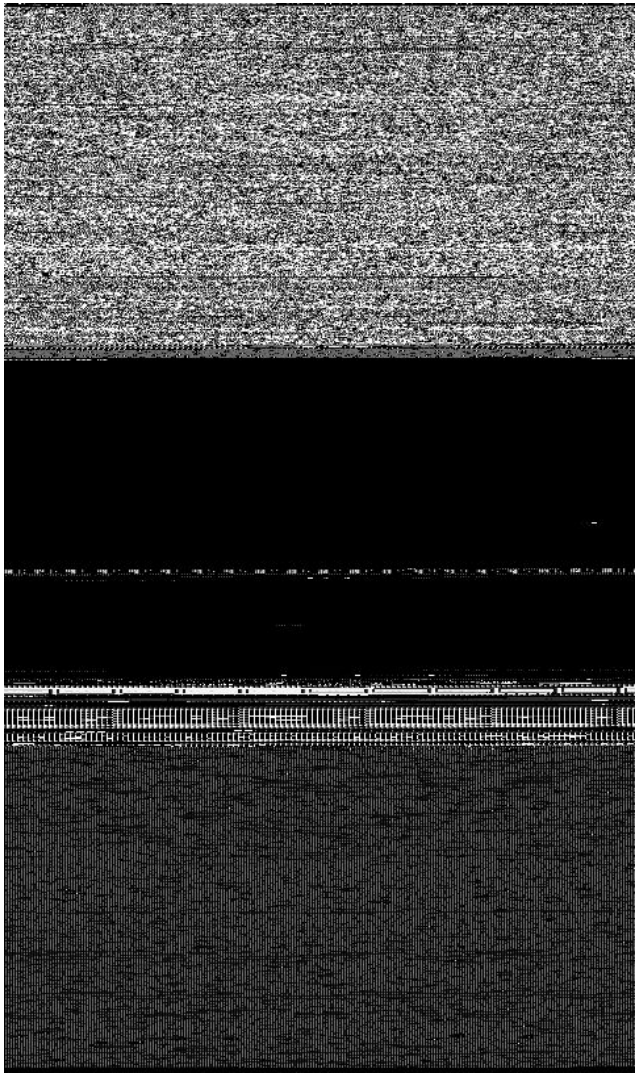


.text

.data

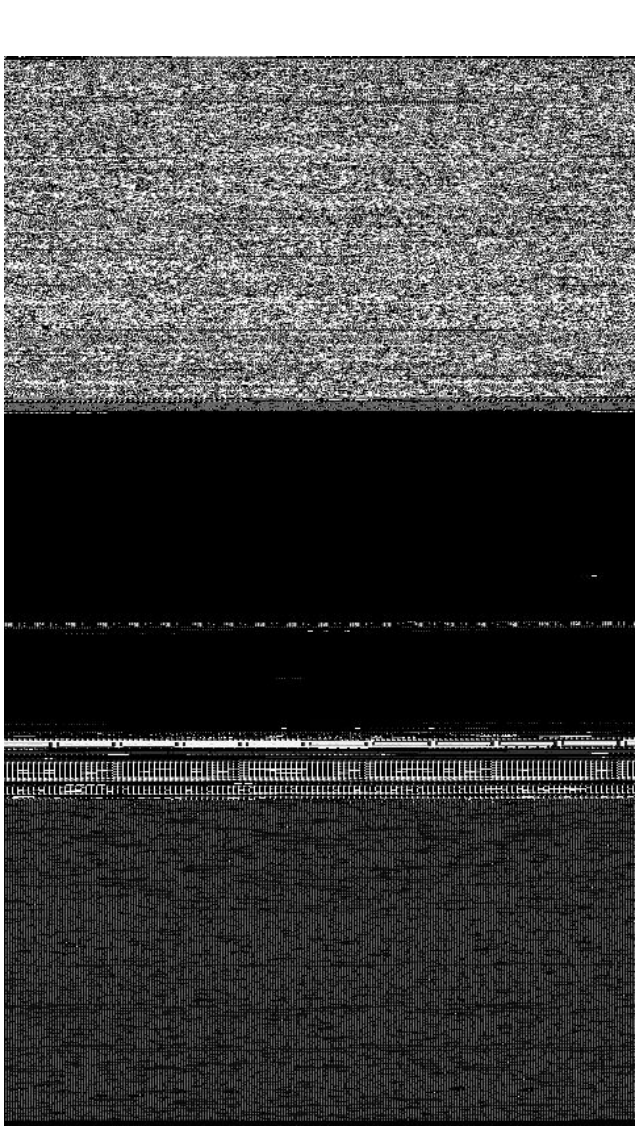
.rsrc

Windows PE

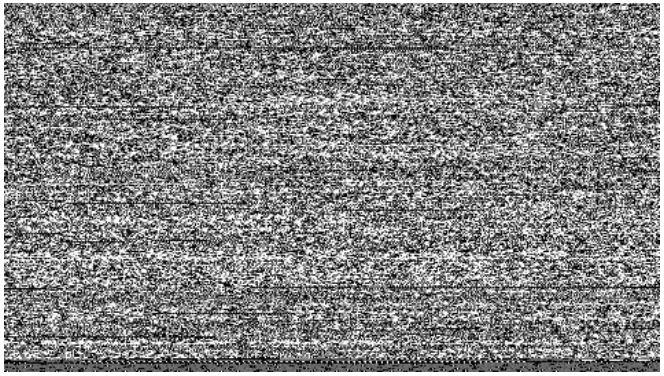


cmd.exe

Windows PE

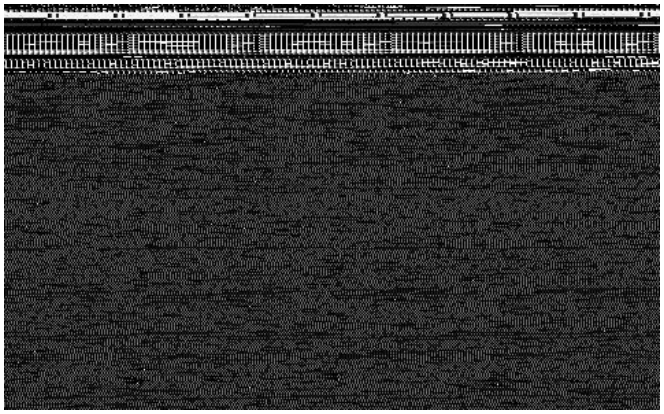


cmd.exe



.text

.data



.rsrc

Machine Code

(Windows PE cmd.exe)

Sample

Machine Code

(Windows PE cmd.exe)

Sample

0

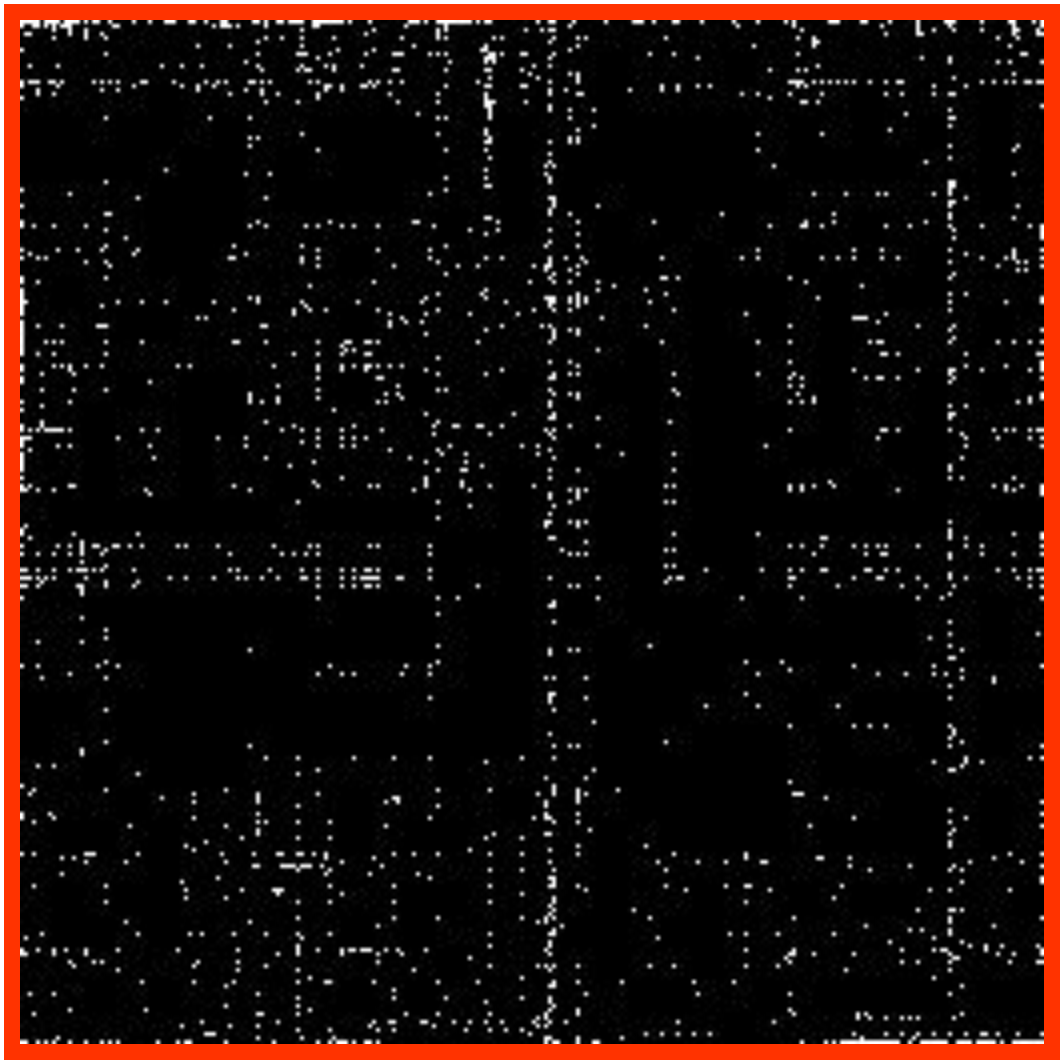
255

Machine Code

(Windows PE cmd.exe)

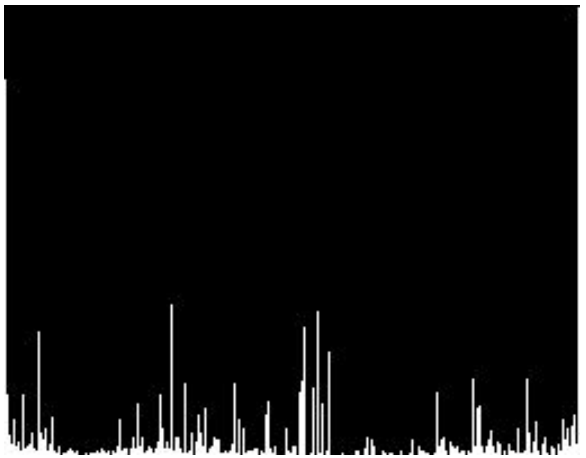
0

255



255

Sample



0

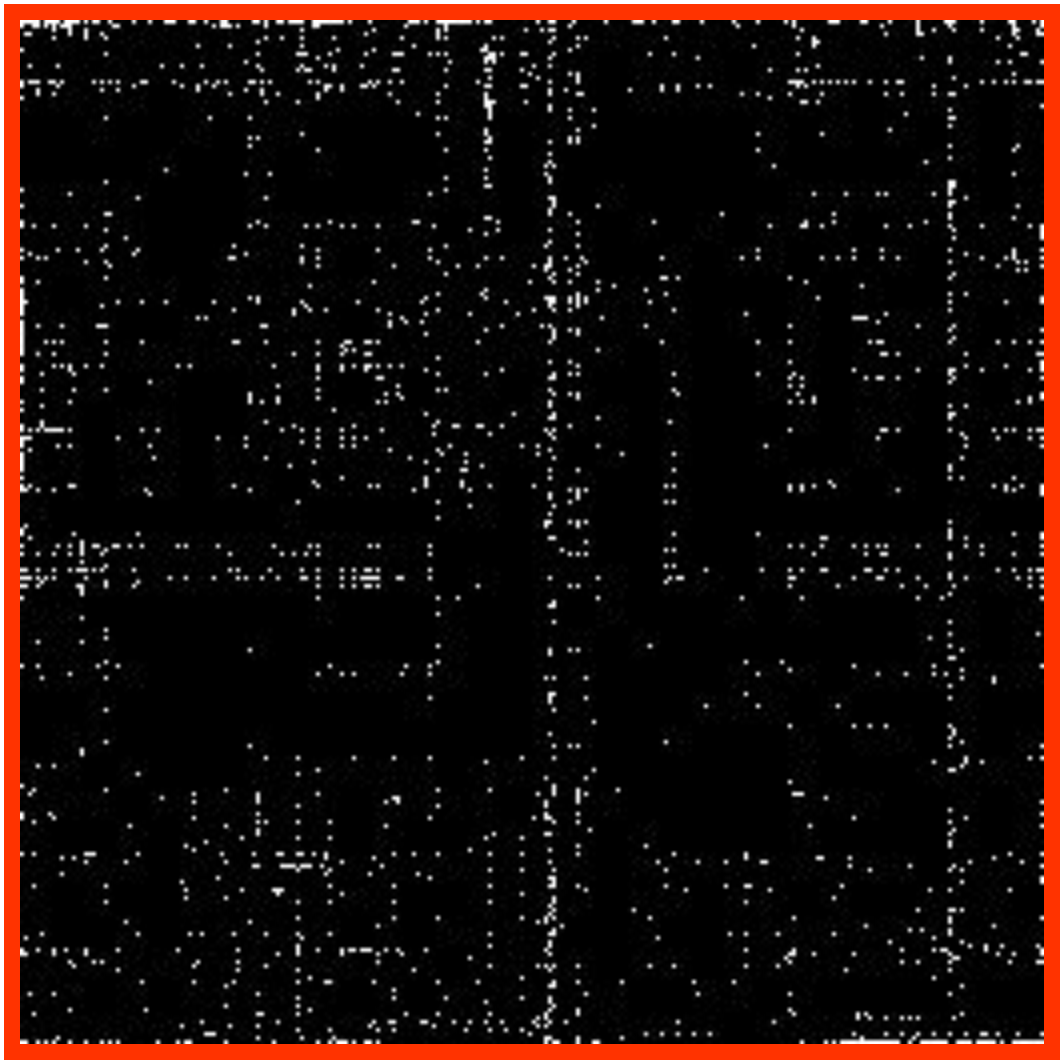
255

Machine Code

(Windows PE cmd.exe)

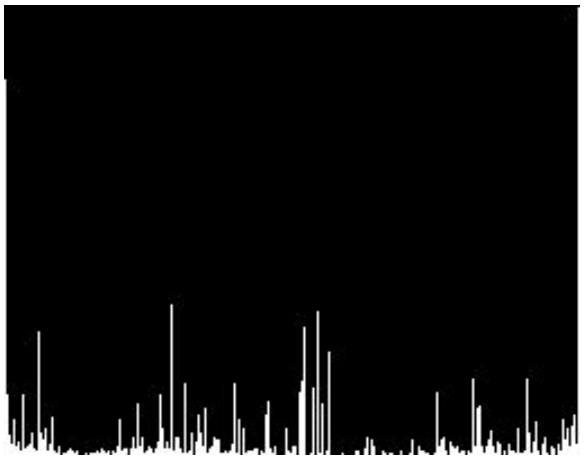
0

255



255

Sample



0

Demo

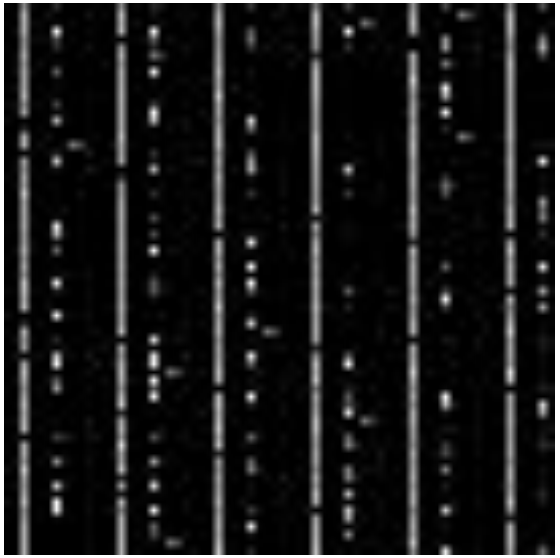
255

Data Structures

Firefox Process Memory

Microsoft Word 2003 .doc

Windows .dll



Neverwinter Nights Database

Random

Sequence of random bytes

Repeating Values

Blocks of repeating 0xFF values

Transformations

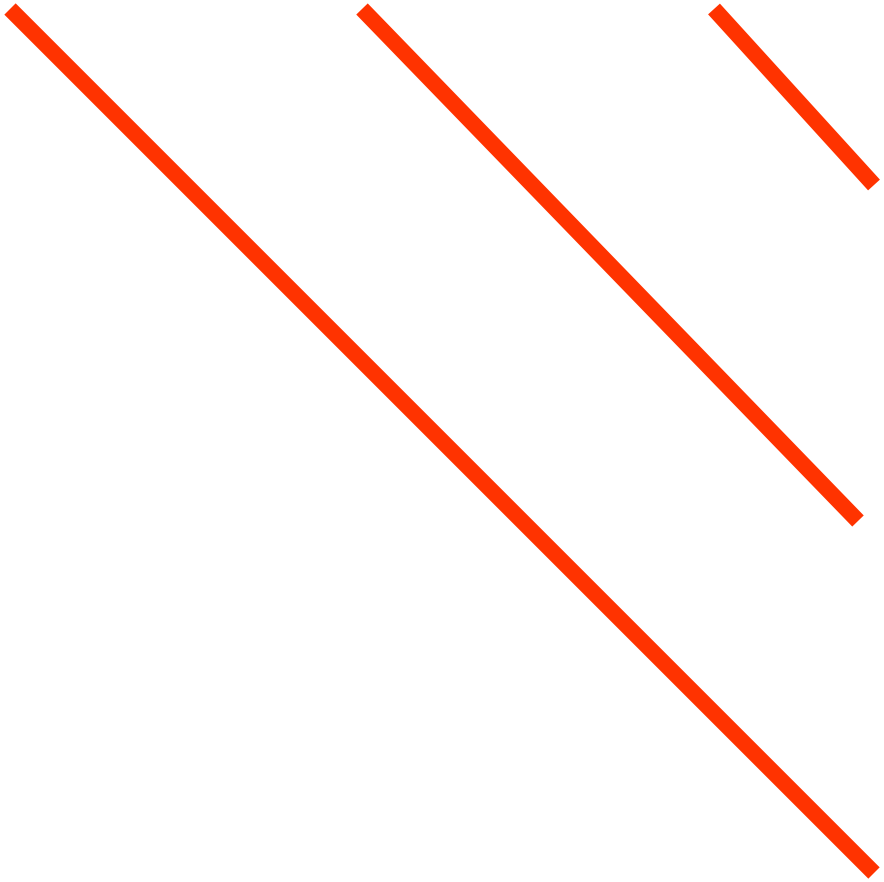
{encryption, compression, encoding}

Consider an image...

Encoding
(Base64 Windows PE)

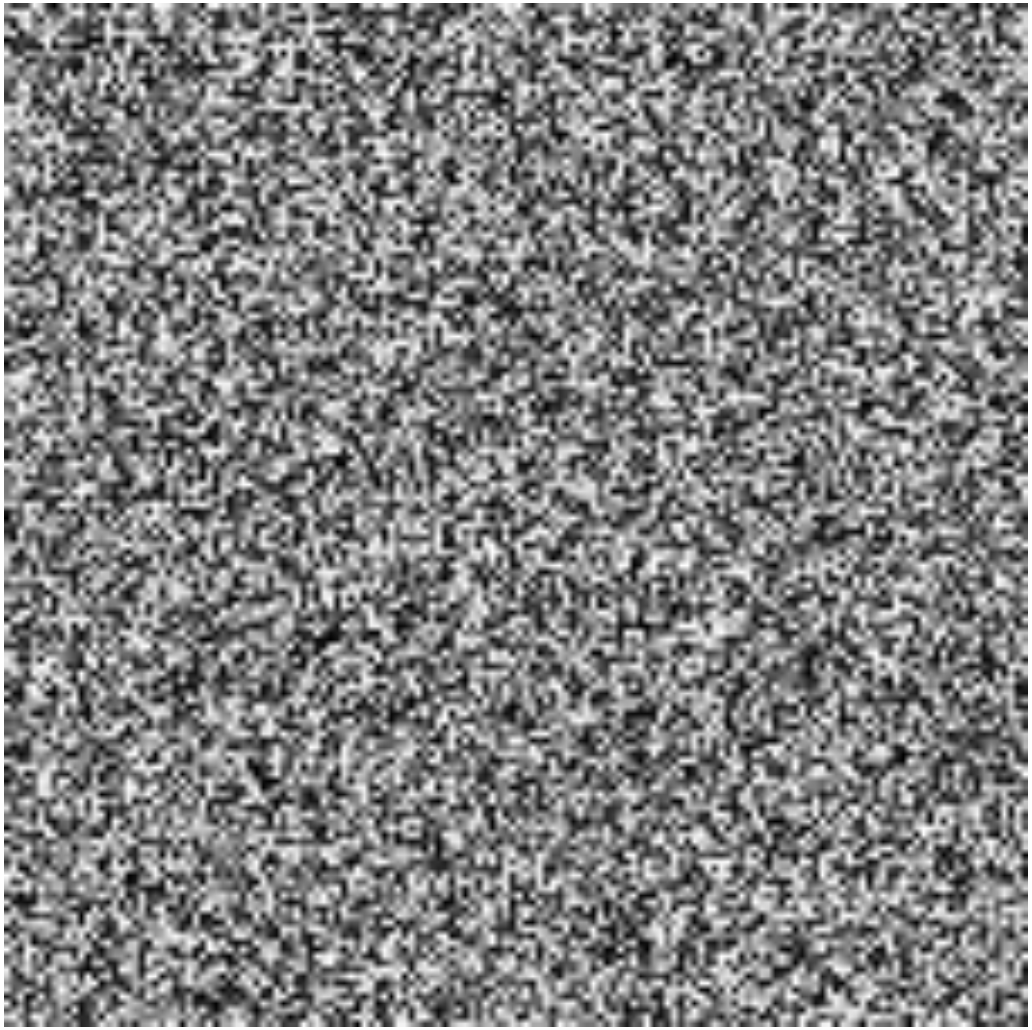
Compression

Compression



Packing (UPX)

Encrypted



AES Encrypted Word Document

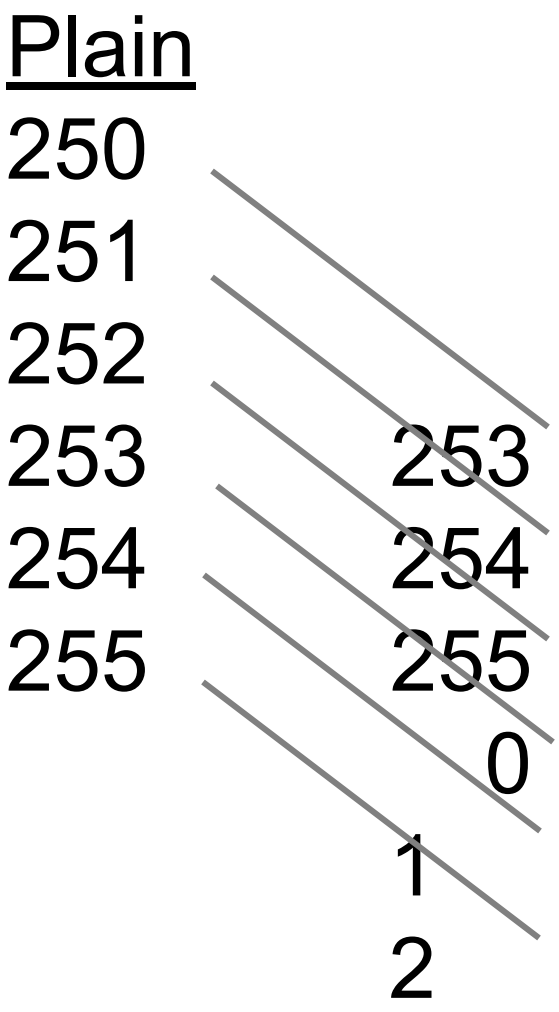
Adding a Constant

Plain				Cipher
b	98	+ 150	=	248
l	108	+ 150	=	2
a	97	+ 150	=	247
c	99	+ 150	=	249
k	107	+ 150	=	1
	32	+ 150	=	182
h	104	+ 150	=	254
a	97	+ 150	=	247
t	116	+ 150	=	10

Adding a Constant

Plain	Cipher
250	
251	
252	
253	253
254	254
255	255
	0
	1
	2

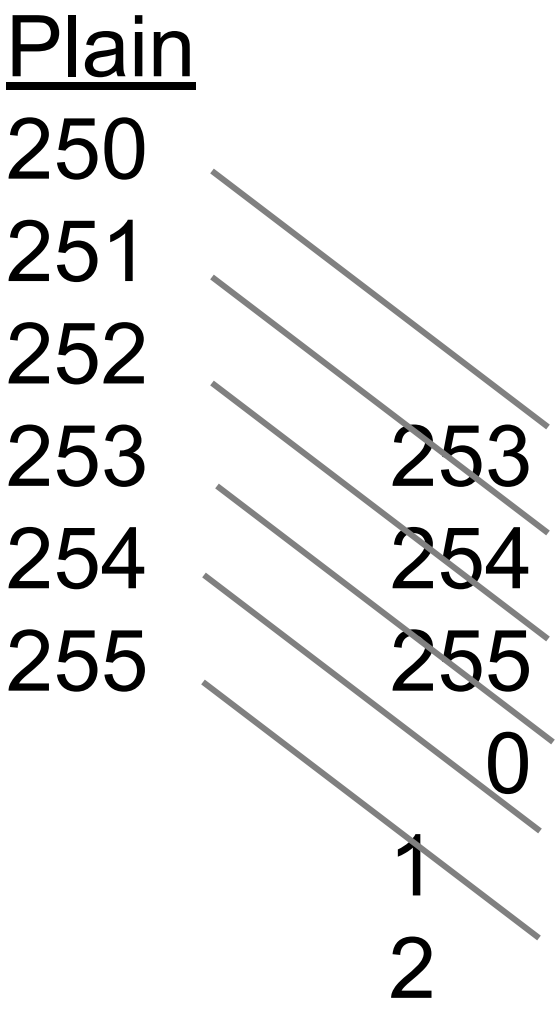
Adding a Constant



Adding a constant is the equivalent of a shift or Caesar cipher.

The byte frequency distribution is merely shifted

Adding a Constant



Adding a constant is the equivalent of a shift or Caesar cipher.

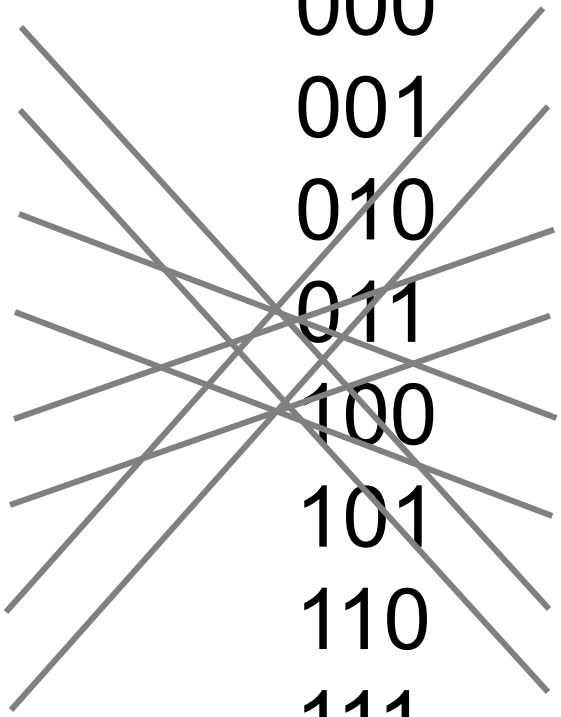
The byte frequency distribution is merely shifted

8 Bit XOR

Plain			Cipher
b	98	XOR 150	= 244
l	108	XOR 150	= 250
a	97	XOR 150	= 247
c	99	XOR 150	= 245
k	107	XOR 150	= 253
	32	XOR 150	= 182
h	104	XOR 150	= 254
a	97	XOR 150	= 247
t	116	XOR 150	= 226

XOR

Plain	Cipher
000	000
001	001
010	010
011	011
100	100
101	101
110	110
111	111



8 bit XOR is equivalent to a monoalphabetic substitution cipher

16 Bit XOR

Plain		Cipher		
byte 1	□	KEY1	□	BYTE 1
byte 2	□	KEY2	□	BYTE 2
byte 3	□	KEY1	□	BYTE 3
byte 4	□	KEY2	□	BYTE 4

...

32 Bit XOR

Plain		Cipher
byte 1	KEY1	BYTE 1
byte 2	KEY2	BYTE 2
byte 3	KEY3	BYTE 3
byte 4	KEY4	BYTE 4
byte 5	*KEY1*	BYTE 5
byte 6	*KEY2*	BYTE 6

8 bit XOR is equivalent to a monoalphabetic substitution cipher

16 bit and 32 bit XOR are polyalphabetic (2 and 4 alphabets)

N Bit XOR

Plain		Cipher
byte 1	KEY1	BYTE 1
byte 2	KEY2	BYTE 2
byte 3	KEY3	BYTE 3
byte 4	KEY4	BYTE 4
...		
byte N	KEYN	BYTE N

N Bit XOR

Plain		Cipher
byte 1	KEY1	BYTE 1
byte 2	KEY2	BYTE 2
byte 3	KEY3	BYTE 3
byte 4	KEY4	BYTE 4
...		
byte N	KEYN	BYTE N

8 bit XOR is equivalent to a monoalphabetic substitution cipher

16 bit and 32 bit XOR are polyalphabetic (2 and 4 alphabets)

N bit XOR, where N equals message length is a one time pad

N Bit XOR

Plain		Cipher
byte 1	KEY1	BYTE 1
byte 2	KEY2	BYTE 2
byte 3	KEY3	BYTE 3
byte 4	KEY4	BYTE 4
...		
byte N	KEYN	BYTE N

8 bit XOR is equivalent to a monoalphabetic substitution cipher

16 bit and 32 bit XOR are polyalphabetic (2 and 4 alphabets)

N bit XOR, where N equals message length is a one time pad

Demos

	Average Byte Value		Shannon Entropy	
		σ		σ
random	127.40	2.34	9.98	0.01
encrypt (AES256/text)	127.47	2.31	9.98	0.01
compress (bzip2/text)	126.68	4.23	9.98	0.01
compress (compress/text)	113.72	8.87	9.96	0.05
compress (deflate (png)	121.78	12.94	9.71	0.70
compress (LZW (gif) / image)	113.75	8.23	9.94	0.05
compress (mpeg/music)	126.26	7.22	9.87	0.44
compress (jpeg/image)	130.76	12.77	9.73	0.88
encoded (base64/zip)	84.46	0.74	9.76	0.02
encoded (uuencoded/zip)	63.71	0.69	9.70	0.02
machine code (linux elf)	116.42	14.97	7.61	0.44
machine code (windows PE)	107.39	18.46	8.06	0.73
bitmap	156.47	69.12	6.22	3.62
text (mixed)	88.52	7.48	7.43	0.24

	Average Byte Value		Shannon Entropy	
		σ		σ
random	127.40	2.34	9.98	0.01
encrypt (AES256/text)	127.47	2.31	9.98	0.01
compress (bzip2/text)	126.68	4.23	9.98	0.01
compress (compress/text)	113.72	8.87	9.96	0.05
compress (deflate (png)	121.78	12.94	9.71	0.70
compress (LZW (gif) / image)	113.75	8.23	9.94	0.05
compress (mpeg/music)	126.26	7.22	9.87	0.44
compress (jpeg/image)	130.76	12.77	9.73	0.88
encoded (base64/zip)	84.46	0.74	9.76	0.02
encoded (uuencoded/zip)	63.71	0.69	9.70	0.02
machine code (linux elf)	116.42	14.97	7.61	0.44
machine code (windows PE)	107.39	18.46	8.06	0.73
bitmap	156.47	69.12	6.22	3.62
text (mixed)	88.52	7.48	7.43	0.24

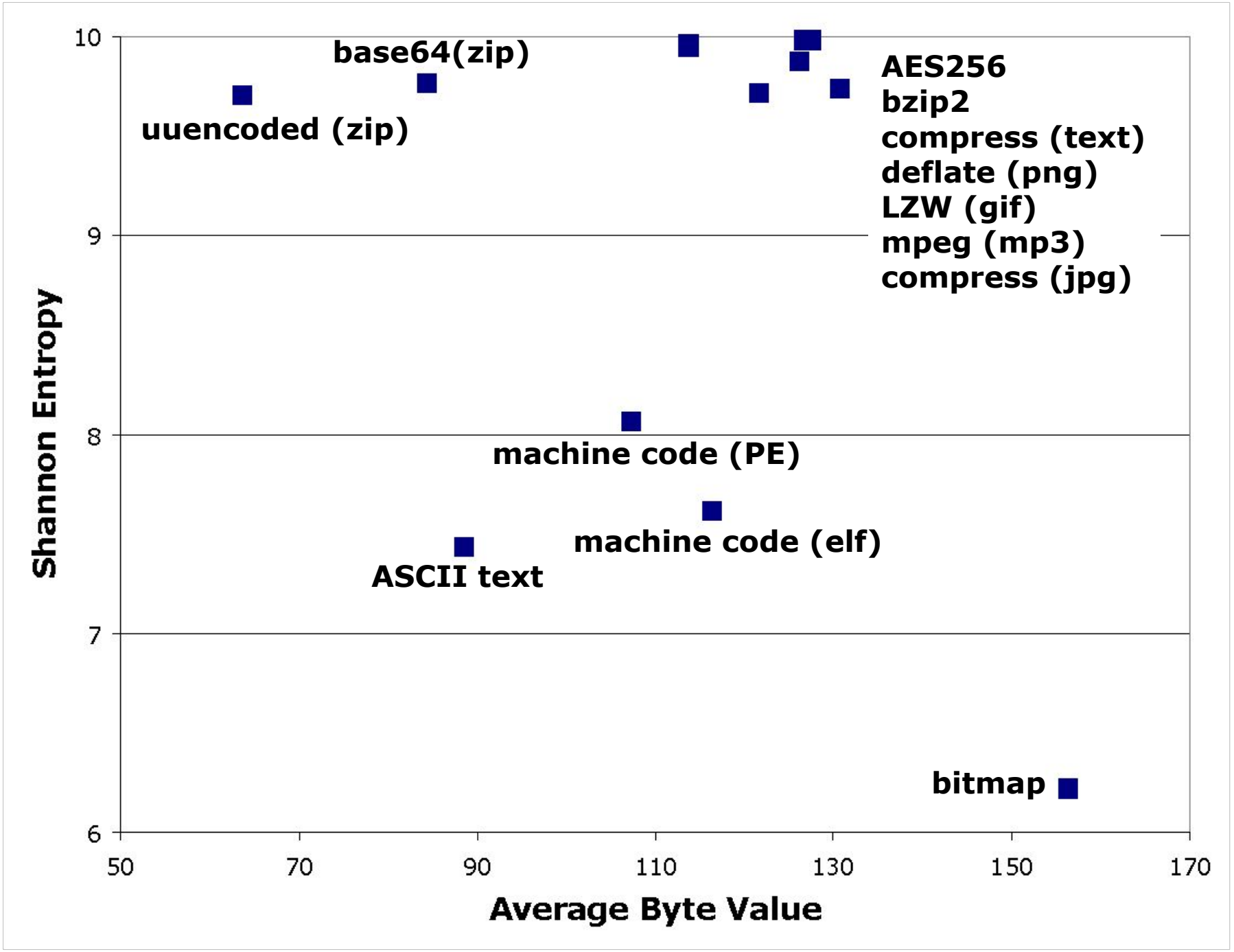
	Average Byte Value		Shannon Entropy	
		σ		σ
random	127.40	2.34	9.98	0.01
encrypt (AES256/text)	127.47	2.31	9.98	0.01
compress (bzip2/text)	126.68	4.23	9.98	0.01
compress (compress/text)	113.72	8.87	9.96	0.05
compress (deflate (png)	121.78	12.94	9.71	0.70
compress (LZW (gif) / image)	113.75	8.23	9.94	0.05
compress (mpeg/music)	126.26	7.22	9.87	0.44
compress (jpeg/image)	130.76	12.77	9.73	0.88
encoded (base64/zip)	84.46	0.74	9.76	0.02
encoded (uuencoded/zip)	63.71	0.69	9.70	0.02
machine code (linux elf)	116.42	14.97	7.61	0.44
machine code (windows PE)	107.39	18.46	8.06	0.73
bitmap	156.47	69.12	6.22	3.62
text (mixed)	88.52	7.48	7.43	0.24

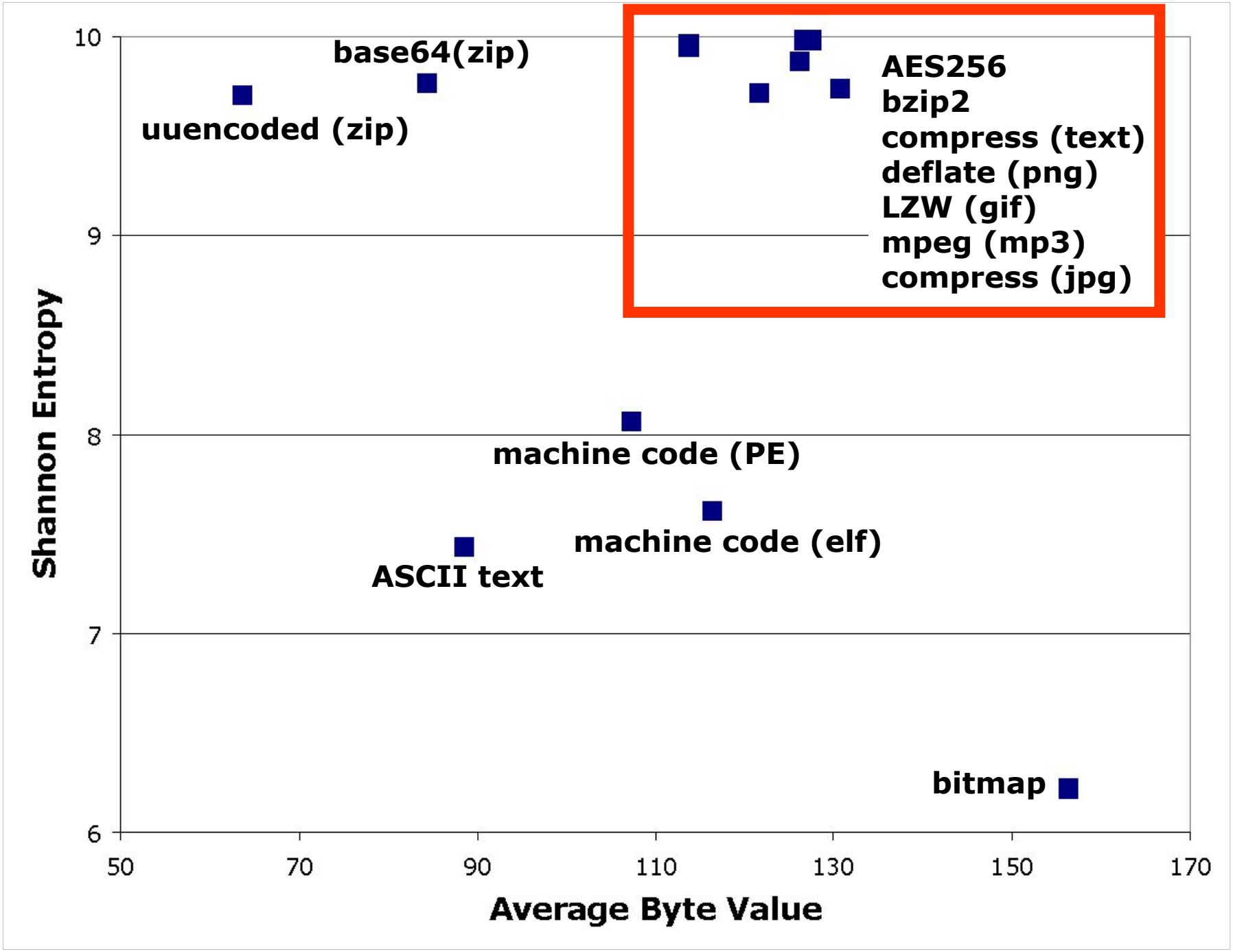
	Average Byte Value		Shannon Entropy	
		σ		σ
random	127.40	2.34	9.98	0.01
encrypt (AES256/text)	127.47	2.31	9.98	0.01
compress (bzip2/text)	126.68	4.23	9.98	0.01
compress (compress/text)	113.72	8.87	9.96	0.05
compress (deflate (png)	121.78	12.94	9.71	0.70
compress (LZW (gif) / image)	113.75	8.23	9.94	0.05
compress (mpeg/music)	126.26	7.22	9.87	0.44
compress (jpeg/image)	130.76	12.77	9.73	0.88
encoded (base64/zip)	84.46	0.74	9.76	0.02
encoded (uuencoded/zip)	63.71	0.69	9.70	0.02
machine code (linux elf)	116.42	14.97	7.61	0.44
machine code (windows PE)	107.39	18.46	8.06	0.73
bitmap	156.47	69.12	6.22	3.62
text (mixed)	88.52	7.48	7.43	0.24

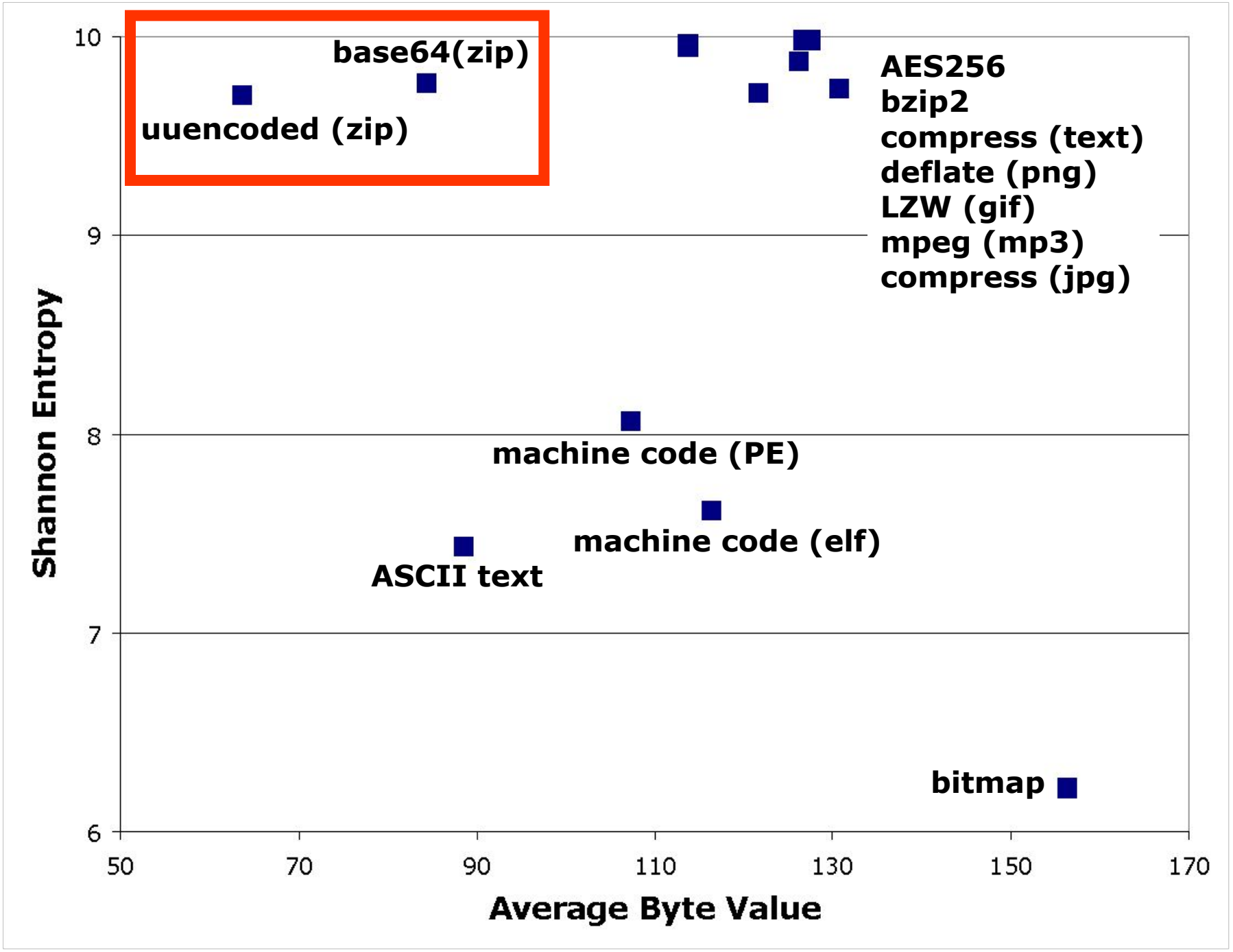
	Average Byte Value		Shannon Entropy	
		σ		σ
random	127.40	2.34	9.98	0.01
encrypt (AES256/text)	127.47	2.31	9.98	0.01
compress (bzip2/text)	126.68	4.23	9.98	0.01
compress (compress/text)	113.72	8.87	9.96	0.05
compress (deflate (png)	121.78	12.94	9.71	0.70
compress (LZW (gif) / image)	113.75	8.23	9.94	0.05
compress (mpeg/music)	126.26	7.22	9.87	0.44
compress (jpeg/image)	130.76	12.77	9.73	0.88
encoded (base64/zip)	84.46	0.74	9.76	0.02
encoded (uuencoded/zip)	63.71	0.69	9.70	0.02
machine code (linux elf)	116.42	14.97	7.61	0.44
machine code (windows PE)	107.39	18.46	8.06	0.73
bitmap	156.47	69.12	6.22	3.62
text (mixed)	88.52	7.48	7.43	0.24

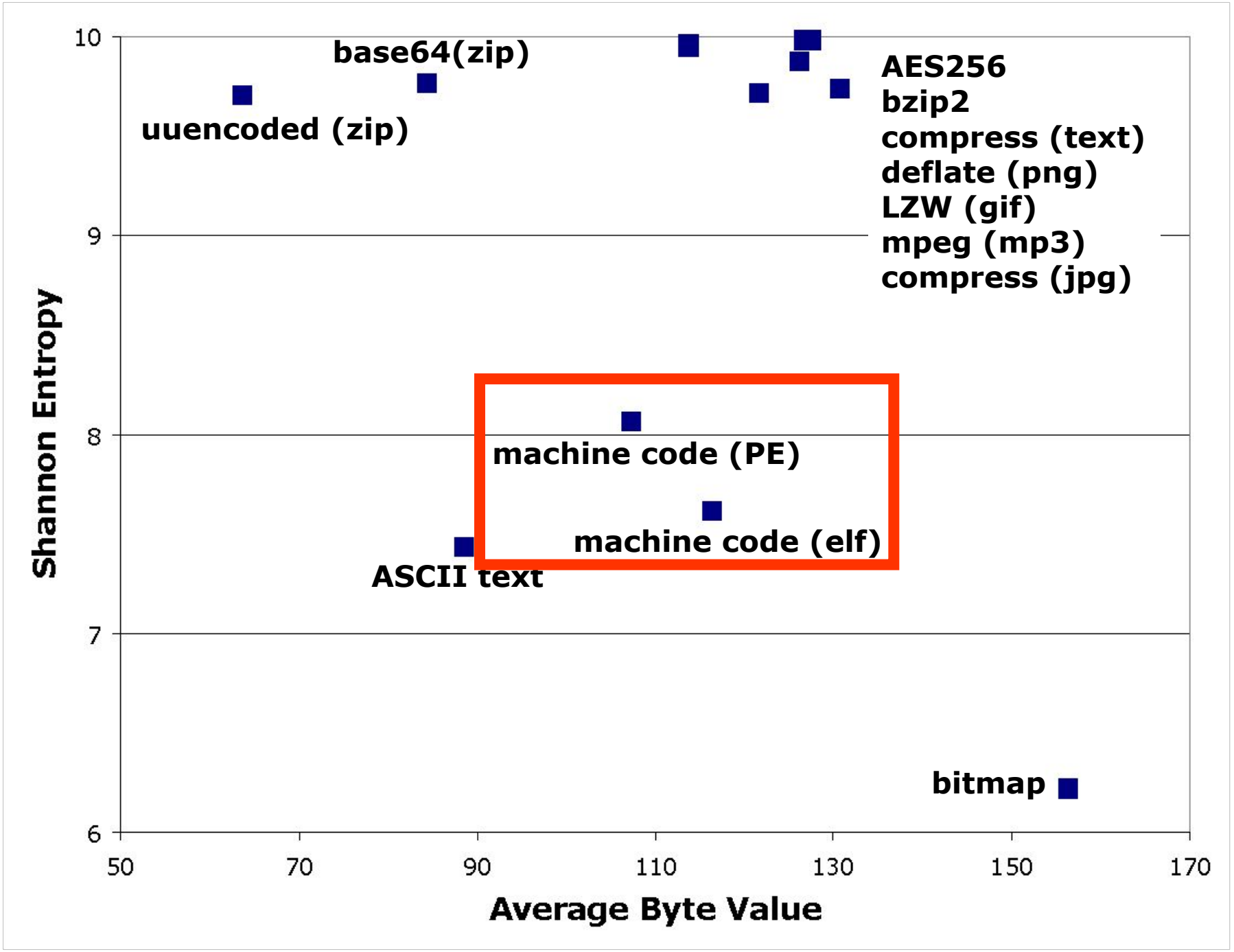
	Average Byte Value		Shannon Entropy	
		σ		σ
random	127.40	2.34	9.98	0.01
encrypt (AES256/text)	127.47	2.31	9.98	0.01
compress (bzip2/text)	126.68	4.23	9.98	0.01
compress (compress/text)	113.72	8.87	9.96	0.05
compress (deflate (png)	121.78	12.94	9.71	0.70
compress (LZW (gif) / image)	113.75	8.23	9.94	0.05
compress (mpeg/music)	126.26	7.22	9.87	0.44
compress (jpeg/image)	130.76	12.77	9.73	0.88
encoded (base64/zip)	84.46	0.74	9.76	0.02
encoded (uuencoded/zip)	63.71	0.69	9.70	0.02
machine code (linux elf)	116.42	14.97	7.61	0.44
machine code (windows PE)	107.39	18.46	8.06	0.73
bitmap	156.47	69.12	6.22	3.62
text (mixed)	88.52	7.48	7.43	0.24

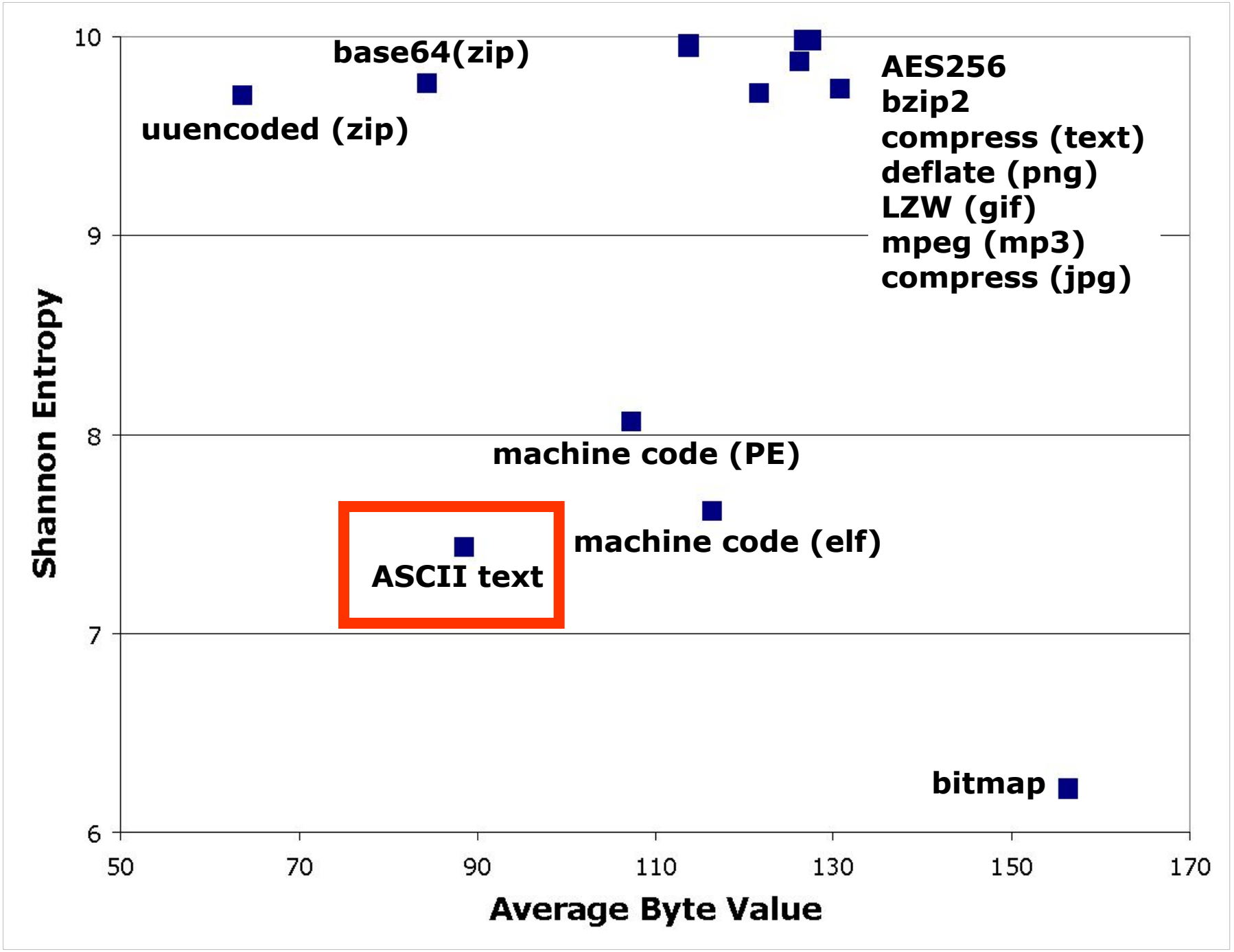
	Average Byte Value		Shannon Entropy	
		σ		σ
random	127.40	2.34	9.98	0.01
encrypt (AES256/text)	127.47	2.31	9.98	0.01
compress (bzip2/text)	126.68	4.23	9.98	0.01
compress (compress/text)	113.72	8.87	9.96	0.05
compress (deflate (png)	121.78	12.94	9.71	0.70
compress (LZW (gif) / image)	113.75	8.23	9.94	0.05
compress (mpeg/music)	126.26	7.22	9.87	0.44
compress (jpeg/image)	130.76	12.77	9.73	0.88
encoded (base64/zip)	84.46	0.74	9.76	0.02
encoded (uuencoded/zip)	63.71	0.69	9.70	0.02
machine code (linux elf)	116.42	14.97	7.61	0.44
machine code (windows PE)	107.39	18.46	8.06	0.73
bitmap	156.47	69.12	6.22	3.62
text (mixed)	88.52	7.48	7.43	0.24

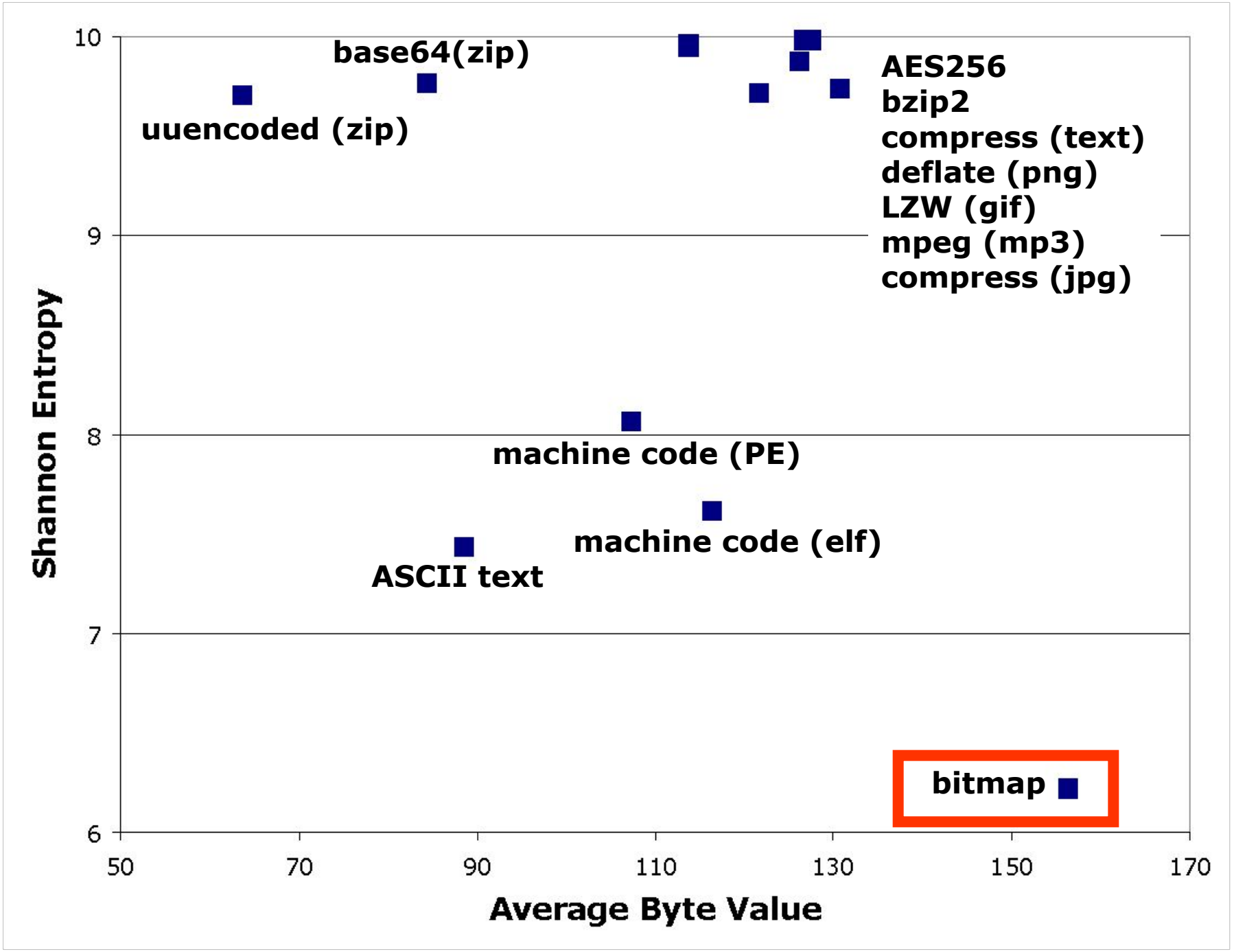

10
9
8
7
6
Shannon Entropy
50
70
90
110
130
150
170
Average Byte Value
base64(zip)
uuencoded (zip)
AES256
bzip2
compress (text)
deflate (png)
LZW (gif)
mpeg (mp3)
compress (jpg)
machine code (PE)
machine code (elf)
ASCII text
bitmap


10
9
8
7
6
Shannon Entropy
50
70
90
110
130
150
170
Average Byte Value
base64(zip)
uuencoded (zip)
AES256
bzip2
compress (text)
deflate (png)
LZW (gif)
mpeg (mp3)
compress (jpg)
machine code (PE)
machine code (elf)
ASCII text
bitmap


base64(zip)
uuencoded (zip)
AES256
bzip2
compress (text)
deflate (png)
LZW (gif)
mpeg (mp3)
compress (jpg)
machine code (PE)
machine code (elf)
ASCII text
bitmap
Shannon Entropy
Average Byte Value
10
9
8
7
6
50
70
90
110
130
150
170

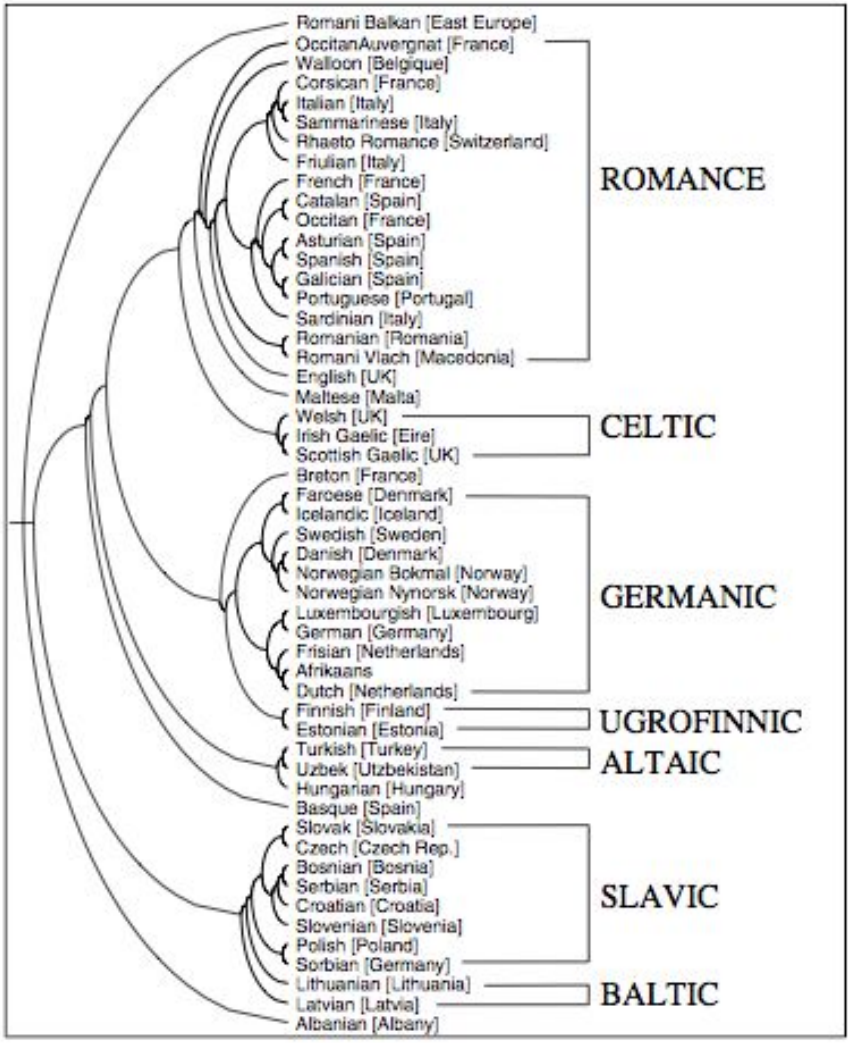

10
9
8
7
6
Shannon Entropy
50
70
90
110
130
150
170
Average Byte Value
base64(zip)
uuencoded (zip)
AES256
bzip2
compress (text)
deflate (png)
LZW (gif)
mpeg (mp3)
compress (jpg)
machine code (PE)
machine code (elf)
ASCII text
bitmap


10
9
8
7
6
Shannon Entropy
50
70
90
110
130
150
170
Average Byte Value
base64(zip)
uuencoded (zip)
AES256
bzip2
compress (text)
deflate (png)
LZW (gif)
mpeg (mp3)
compress (jpg)
machine code (PE)
machine code (elf)
ASCII text
bitmap


10
9
8
7
6
Shannon Entropy
50
70
90
110
130
150
170
Average Byte Value
base64(zip)
uuencoded (zip)
AES256
bzip2
compress (text)
deflate (png)
LZW (gif)
mpeg (mp3)
compress (jpg)
machine code (PE)
machine code (elf)
ASCII text
bitmap

Compression FTW!

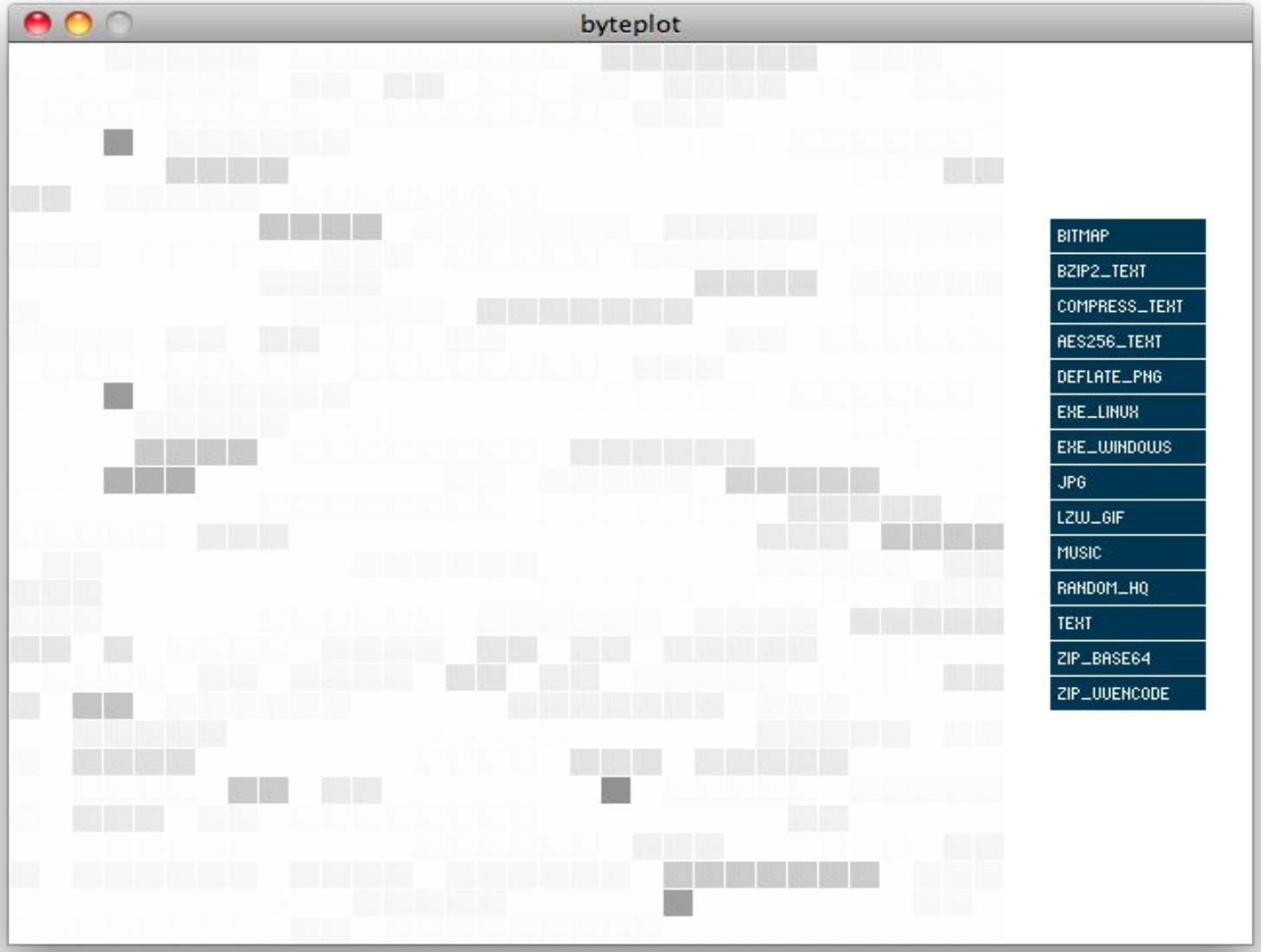
- D. Benedetto, E. Caglioti, and V. Loreto. **Language trees and zipping**. Physical Review Letters, 88, 2002
- Similar files compress together better



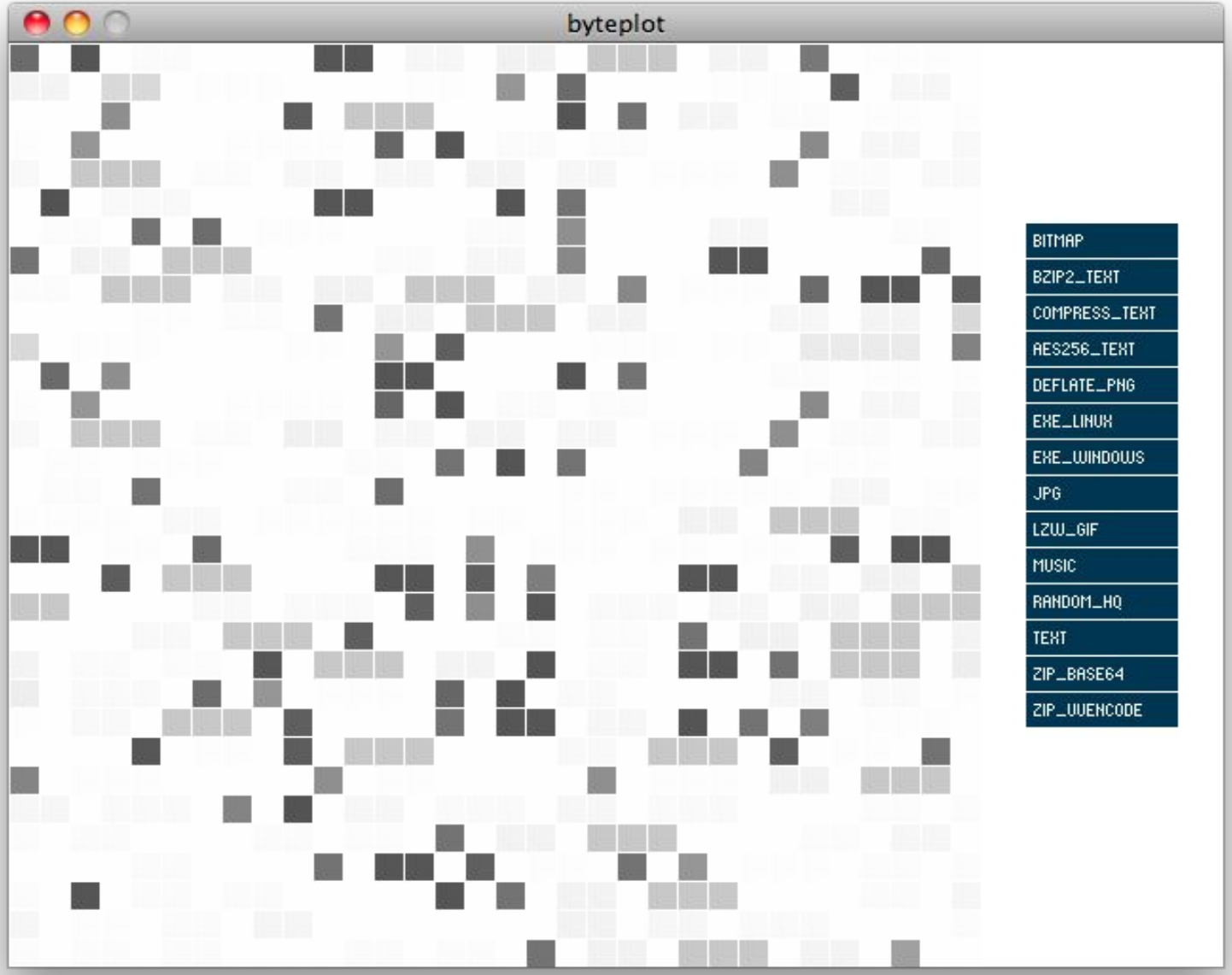
Visualize compression & “bathroom tiles”

- Get **many** file fragments of different types, group by type
- Compress an unknown file fragment together with each group (using their Lempel-Ziv string tables)
- Show where substring matches went
- See if the “tiling” is good

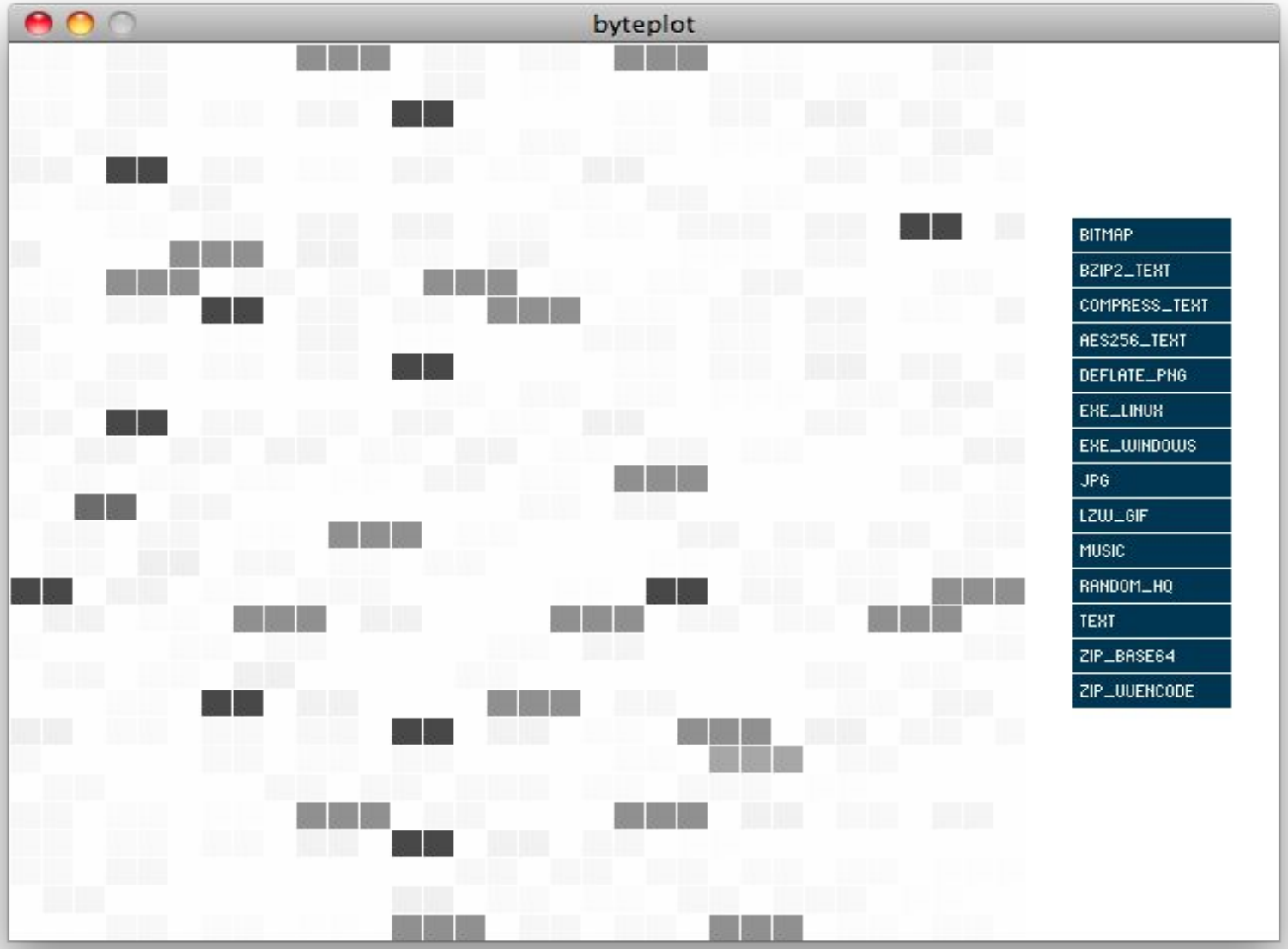
Executable, with executables



Executable, with bitmaps



Executable, with music



Analysis

- Bitmap diversity
- Data structure diversity
- High entropy primitive types
- Transformations
- Minimum size
- Obfuscation
  - J. Erikson's "Dissembler" (ASCII-only Shellcode Generator)
  - J. Mason, S. Small, F. Monrose, G. MacManus. English Shellcode. In the proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), Chicago, IL. November 2009.

    http://www.cs.jhu.edu/~sam/ccs243-mason.pdf

A Visual Study of Primitive Binary Fragment Types

Gregory Conti*, Sergey Bratus, Anna Shubina†
Andrew Lichtenberg‡, Roy Ragsdale, Robert Perez-Alemany,
Benjamin Sangster, and Matthew Supan*

July 4, 2010

Abstract

We argue that visual analysis of binary data objects such as data files, process memory, and file systems presented as grayscale graphical depictions helps distinguish structurally different regions of data and thus facilitates a wide range of analytic tasks such as fragment classification, file type identification, location of regions of interest, and other tasks that require an understanding of the "primitive" data types the objects contain. We believe that, due to the high visual value of this data presentation, such visual analysis is an invaluable help in low-level study of binary data objects and in understanding their structure, and that tools for such visual analysis belong in the toolkit of every researcher studying binary data.

In an effort to facilitate development of such tools, this paper presents a visual study of binary fragments created by common kinds of software, and offers a descriptive taxonomy of primitive binary fragments and their graphical depictions. Although significant research has gone into the study of binary fragments, the depth and breadth of this study to date has been limited. Thus the primary contribution of this paper is an extensible and visual taxonomy to assist and inform researchers conducting low-level analysis of binary objects.

1 Introduction

Binary data is central to the modern computing paradigm and exists in large, complex objects such as process memory, file systems, network flows, as well as in data and executable files. In many cases, however, these objects are

*West Point
†Dartmouth College
‡Skidmore College

Table 1. Taxonomy of Primitive Binary Fragment Types

	Major Category	Minor Category	Subcategory	Notes
Binary Fragment	Text	programming language source code	C	Typically encoded in US-ASCII. May alternatively be grouped by compiled and interpreted languages.
			Python	
			JavaScript	
			...	
		written language	English	Typically encoded in US-ASCII, and more recently, Unicode. Further variants include all upper and all lower case characters.
			Russian	
			...	
		markup language	LaTeX	Some markup languages may encapsulate other binary fragment types, such as JavaScript within HTML and binary images in XML. We recommend treating such embedded objects as distinct types.
			HTML	
			XML	
			...	
	Image	uncompressed	bitmap	Images are commonly encoded using 2^n bits per pixel, where n=1, 2, 4, 8, 16, 24, or 32. The process of generating an image may generate associated data structures, such as image headers and palettes.
		compressed	LZ77 / Huffman	
			JPEG	
			...	
	Audio	uncompressed	PCM	Commonly stored in the .wav file format.
		compressed	MPEG-1 Audio Layer 3	MPEG-1 Audio Layer 3 is commonly referred to as mp3. Vorbis is the lossy audio compression commonly paired with the Ogg container format.
			Vorbis	
			...	
	Video	uncompressed	Full Frame	Infrequently used. Stored in the .avi file format.
		compressed	MPEG-4	The video class may include interleaved image and audio information.
			WMV	
			...	
	Application	packed executable	UPX	UPX and ASPack do not list their specific compression algorithm(s), which might be preferable to include in the taxonomy.
			ASPack	
			...	
		machine code	native	16 bit real mode x86, 32 bit protected mode x86... This category could be extended to include processor type and specific modes.
			virtual	An example is Java bytecode.
		data structure	fixed length	Examples include some arrays, databases, log files.
			variable length	Examples include some stacks, heaps, pointer tables, and database files. pcap format packet captures.
	Random	high quality random	atmospheric noise	Atmospheric noise and patterns in lava lamps are just two of many techniques that generate random numbers, with varying degrees of success.
			lava lamps	
			...	
		pseudo random	Linear Feedback Shift Register	Pseudo random sequences appear random under light scrutiny, but are actually deterministic.
			Mersenne Twister	
			...	
	Encrypted	symmetric	AES	Encryption algorithms may be further categorized based on their input parameters, such as key length, and possibly the entropy of the plaintext data. Encryption is commonly applied to virtually every category in the taxonomy.
			Blowfish	
			...	
		asymmetric	RSA	
			ElGamal	
			...	
	Other Compressed		RLE	Compression is commonly used on virtually every category in the taxonomy, except those that are already compressed or encrypted.
			LZW	
			...	
	Other Encoded		Base64	Encoding is commonly used to convert binary objects to US-ASCII byte values for use on text based network protocols, but is also used in many other ways.
			uuencode	
			...	
	Repeating Values			A single value of one or more bits that is repeated regularly.
	Other			An unforeseen or newly created fragment type.

Future

- Automated identification
- Classification / Clustering / Data Mining
- Dictionary
- Incorporating semantic information
  - (i.e. file format)
- Extending set of primitive types
- Toward memory mapping

- Feedback welcome...

For More Information…

G. Conti, S. Bratus, A. Shubinay, A. Lichtenberg, R. Ragsdale, R. Perez-Alemany, B. Sangster, and M. Supan; “A Visual Study of Primitive Binary Fragment Types;” Black Hat USA White Paper; August 2010. (on CD)

G. Conti, S. Bratus, B. Sangster, R. Ragsdale, M. Supan, A. Lichtenberg, R. Perez and A. Shubina; "Automated Mapping of Large Binary Objects Using Primitive Fragment Type Classification; *Digital Forensics Research Conference (DFRWS)*; August 2010.

B. Sangster, R. Ragsdale, G. Conti; “Automated Mapping of Large Binary Objects;” *Shmoocon*; Work in Progress Talk; February 2009.

G. Conti, E. Dean, M. Sinda, and B. Sangster; “Visual Reverse Engineering of Binary and Data Files;” *Workshop on Visualization for Computer Security (VizSEC)*; September 2008.

G. Conti and E. Dean; “Visual Forensic Analysis and Reverse Engineering of Binary Data;” *Black Hat USA*; August 2008.

binviz (on CD)
Marius Ciepluch (wishi) extending binvis - http://code.google.com/p/binvis/

We would like to thank our white paper co-authors: Anna Shubina, Andrew Lichtenberg, Roy Ragsdale, Robert Perez-Alemany, Benjamin Sangster, and Matthew Supan.

Voyage of the Reverser: A Visual Study of Binary Species

Greg Conti // West Point // gregory.conti@usma.edu
Sergey Bratus // Dartmouth // sergey@cs.dartmouth.edu