Finding hidden data with optimistic decoding.

Drexel University
Monday, November 18th, 2013 / 11:00am

Simson L. Garfinkel

http://simson.net/

The opinions expressed herein are those of the author(s), and are not necessarily representative of those of the Naval Postgraduate School, the Department of Defense (DOD); or, the United States Army, Navy, or Air Force.

Digital information is pervasive in today's society.

Many potential sources of digital information:

- Desktops; Laptops
- Tablets; Cell Phones
- Internet-Based Services
- Cars

My research makes internal, technical data usable by non-technologists

- Law Enforcement — Document a conspiracy (stock fraud; murder-for-hire; Silk Road)
- DOD — Identify members of a terrorist organization.
- Ordinary people — Recover deleted files.

These tools can also be used to audit software for privacy leaks.

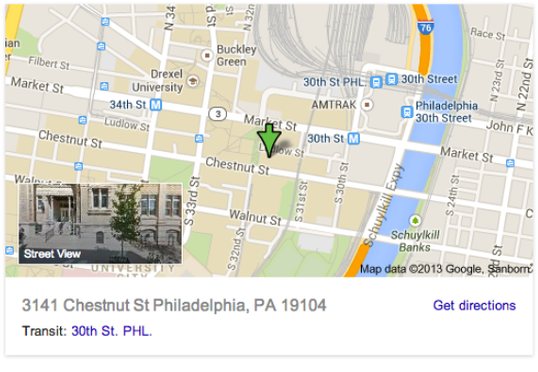
How do we *know* when information is present?

With a digital device, we look for information that we can recognize.

These devices have information:

If we find things like this:

photos



GIS information

Alumni Relations

215-895-ALUM

alumni@drexel.edu

Identify intelligence

Recognizing information can be a challenge

We commonly recognize identity information with regular expressions.

This regular expression:

```
[a-zA-Z]+@[\-a-zA-Z._]+
```

Will find this email:

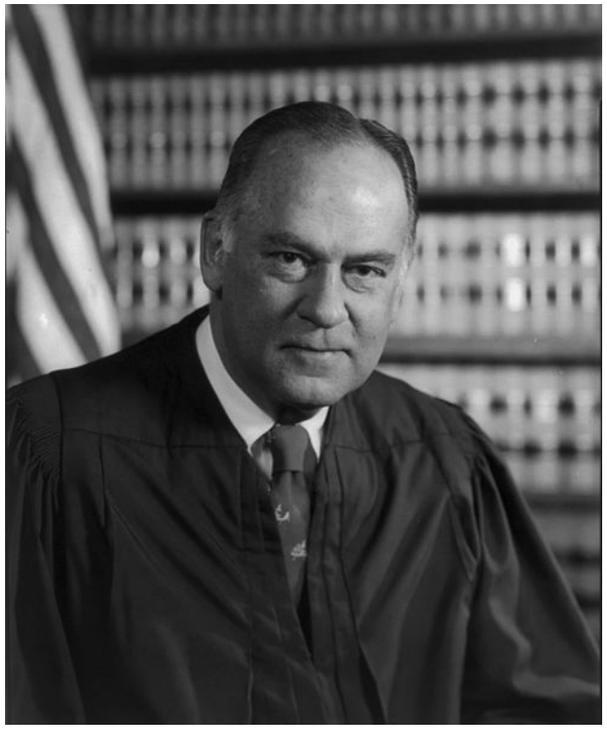
```
stewart@uscourts.gov
```

Even when the email address is surrounded by “random” data:

```
23ae c8ba 7f42 a653 3f0f 05a4 ac45 3c07  #....B.S?....E<.
7374 6577 6172 7440 7573 636f 7572 7473  stewart@uscourts
2e67 6f76 0a3a 752c e621 6398 aa14 f2c8  .gov.:u,.!c.....
4159 e6ad 0c                             AY...
```

Call this the “Stewart” test for identity intelligence.

“I know it when I see it.”



US Supreme Court Justice Potter Stewart
1976 official portrait.jpg

—*Jacobellis v. Ohio 378 US 184 (1964)*

“But I know it when I see it, and the motion picture involved in this case is not that.”

“Triage” is an important problem in digital forensics.

“Triage” means finding & prioritizing high-value items.

Data sources for triage:

- Email addresses
- Financial information
- Contacts, calendar, documents
- Temporal / time sequence
- Geolocation information
- Presence of software

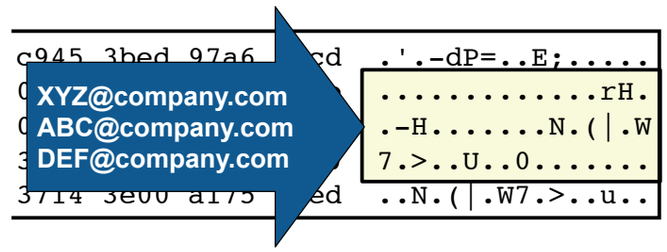
stewart@uscourts.gov

All of these techniques require identifying the information.

“Optimistic decoding” is an approach for finding and extracting identity information that is frequently missed.

Email addresses can be compressed.

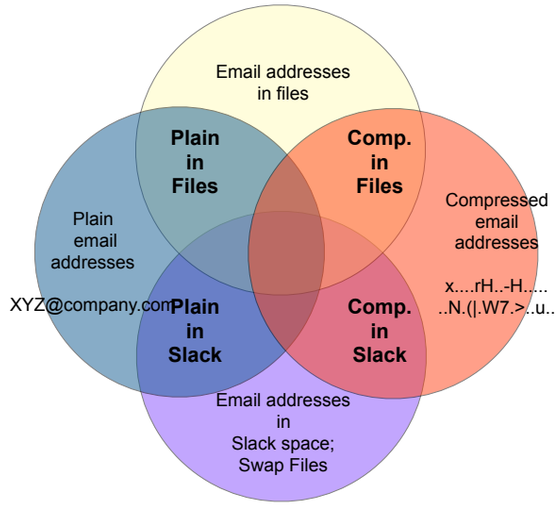
Popular forensic tools do not optimistically decompress.



bulk_extractor implements optimistic decoding.

BE

Our study of 1400 drives found thousands of email addresses that were *only in compressed data.*

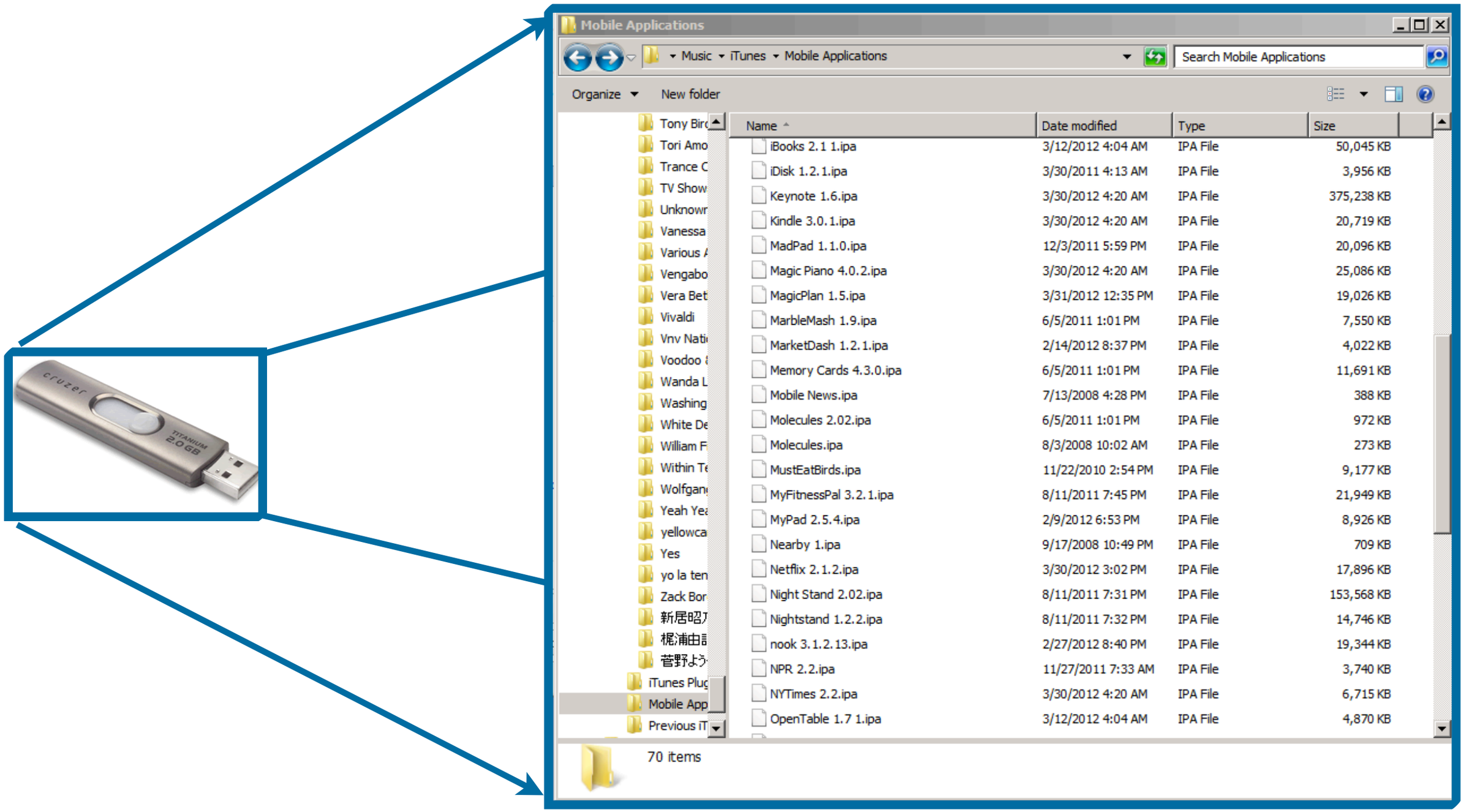


Recent successes with optimistic decoding.

```
a097 83a1 ed96 26a6 3c69 3d0f 750a 2399  ......&.<i=.u.#.
a2b5 bea7 692f 5847 a38a dd53 082c add5  ....i/XG...S.,..
5061 b64c 721d 864b 90b6 b55f bb04 735c  Pa.Lr..K..._..s\
9448 6730 5453 df64 813e b603 5795 2242  .Hg0TS.d.>..W."B
e9c8 7454 7322 7cdc b60e 97af 2f64 2728  ..tTs"|...../d'(
3cfb 84bd 2a84 2dfe 50ea 5935 c349 1513  <XYZ@COMPANY.COM
a9e9 e92c a3f8 6e46 0530 8a88 c7a2 5d2b  ...,..nF.0....]+
d89d 77cc fe1e f637 f3f3 d0af 1b47 c09b  ..w....7.....G..
```

Extracting encoded data

We think of computers as devices with *files*.



Storage devices actually store data in *blocks* or *sectors*.

“hex dump:”

hex words								ASCII representation
a2b5	bea7	692f	5847	a38a	dd53	082c	add5	....i/XG...S.,..
a097	83a1	ed96	26a6	3c69	3d0f	750a	2399	......&.<i=.u.#.
a2b5	bea7	692f	5847	a38a	dd53	082c	add5	....i/XG...S.,..
5061	b64c	721d	864b	90b6	b55f	bb04	735c	Pa.Lr..K..._..s\
9448	6730	5453	df64	813e	b603	5795	2242	.Hg0TS.d.>..W."B
e9c8	7454	7322	7cdc	b60e	97af	2f64	2728	..tTs"|...../d'(
a097	83a1	ed96	26a6	3c69	3d0f	750a	2399	......&.<i=.u.#.
a2b5	bea7	692f	5847	a38a	dd53	082c	add5	....i/XG...S.,..
5061	b64c	721d	864b	90b6	b55f	bb04	735c	Pa.Lr..K..._..s\
9448	6730	5453	df64	813e	b603	5795	2242	.Hg0TS.d.>..W."B
e9c8	7454	7322	7cdc	b60e	97af	2f64	2728	..tTs"|...../d'(
3cfb	84bd	2a84	2dfe	50ea	5935	c349	1513	<XYZ@COMPANY.COM
a9e9	e92c	a3f8	6e46	0530	8a88	c7a2	5d2b	...,..nF.0....]+
d89d	77cc	fe1e	f637	f3f3	d0af	1b47	c09b	..w....7.....G..
a097	83a1	ed96	26a6	3c69	3d0f	750a	2399	......&.<i=.u.#.
a2b5	bea7	692f	5847	a38a	dd53	082c	add5	....i/XG...S.,..
5061	b64c	721d	864b	90b6	b55f	bb04	735c	Pa.Lr..K..._..s\
a097	83a1	ed96	26a6	3c69	3d0f	750a	2399	......&.<i=.u.#.
a2b5	bea7	692f	5847	a38a	dd53	082c	add5	....i/XG...S.,..
5061	b64c	721d	864b	90b6	b55f	bb04	735c	Pa.Lr..K..._..s\
9448	6730	5453	df64	813e	b603	5795	2242	.Hg0TS.d.>..W."B
e9c8	7454	7322	7cdc	b60e	97af	2f64	2728	..tTs"|...../d'(
a097	83a1	ed96	26a6	3c69	3d0f	750a	2399	......&.<i=.u.#.
a2b5	bea7	692f	5847	a38a	dd53	082c	add5	....i/XG...S.,..
5061	b64c	721d	864b	90b6	b55f	bb04	735c	Pa.Lr..K..._..s\
9448	6730	5453	df64	813e	b603	5795	2242	.Hg0TS.d.>..W."B
e9c8	7454	7322	7cdc	b60e	97af	2f64	2728	..tTs"|...../d'(
a9e9	e92c	a3f8	6e46	0530	8a88	c7a2	5d2b	...,..nF.0....]+
d89d	77cc	fe1e	f637	f3f3	d0af	1b47	c09b	..w....7.....G..
a097	83a1	ed96	26a6	3c69	3d0f	750a	2399	......&.<i=.u.#.
a2b5	bea7	692f	5847	a38a	dd53	082c	add5	....i/XG...S.,..
5061	b64c	721d	864b	90b6	b55f	bb04	735c	Pa.Lr..K..._..s\

A 64GB stick has 125 million sectors

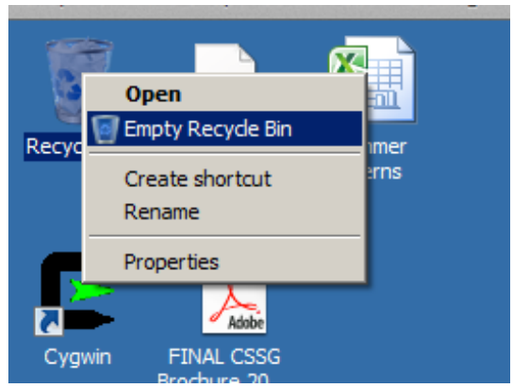
512 bytes = 1 sector

The operating system maps files to sectors.

Data in a sector can be resident:

Files can be “deleted” but the data remains:

user files
email messages
[temporary files]

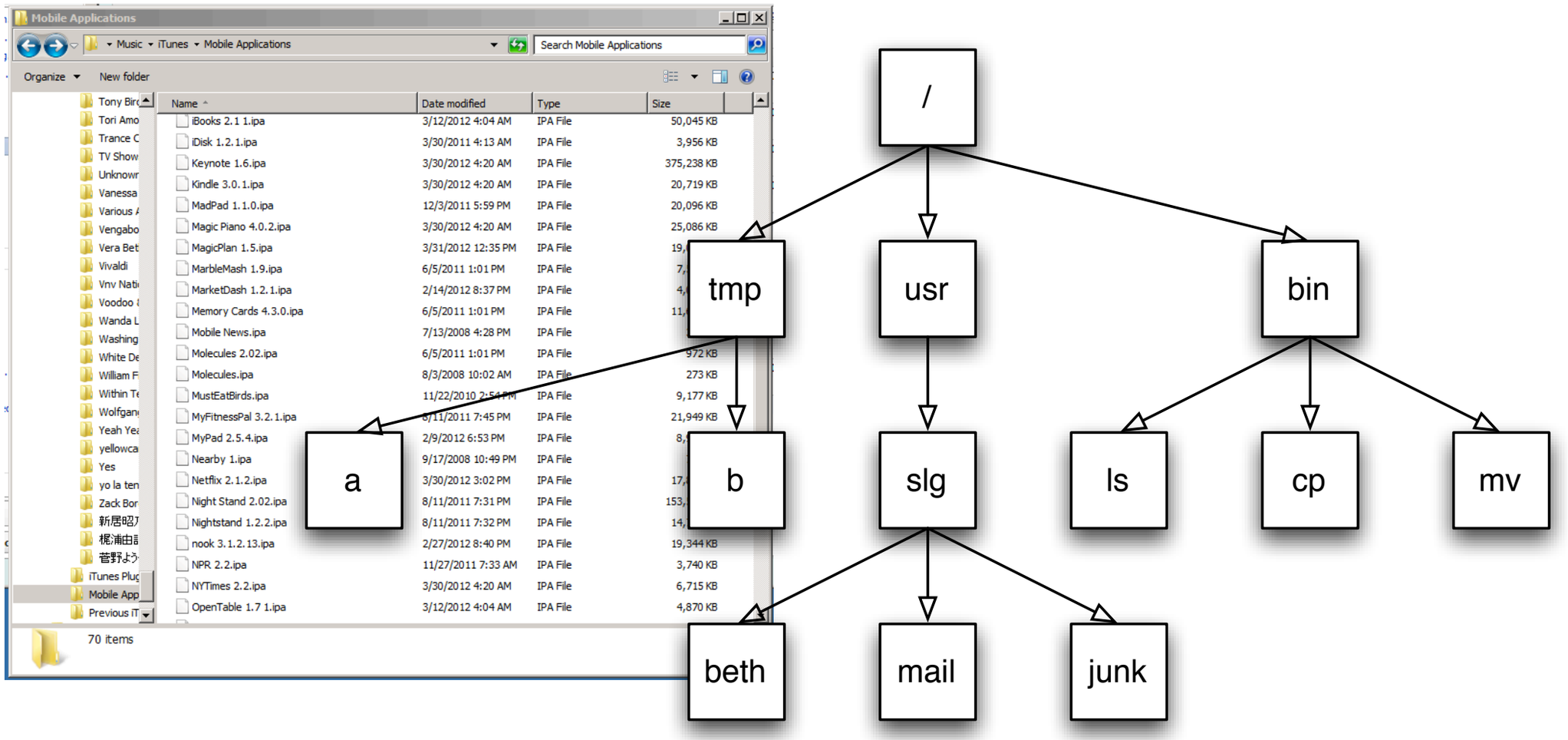


Sectors can be wiped clean:

No Data

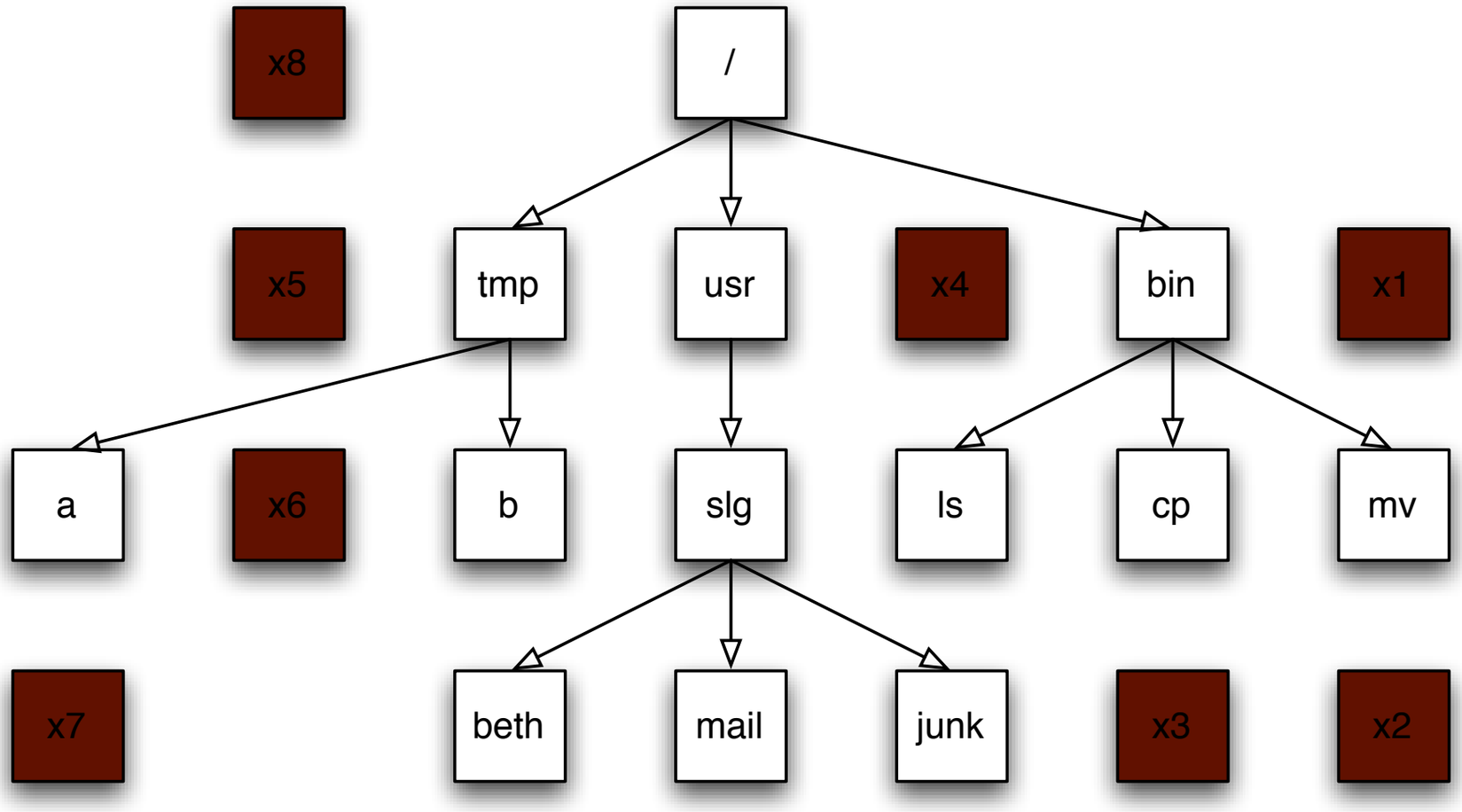
blank sectors

Allocated data are files seen from the root directory.



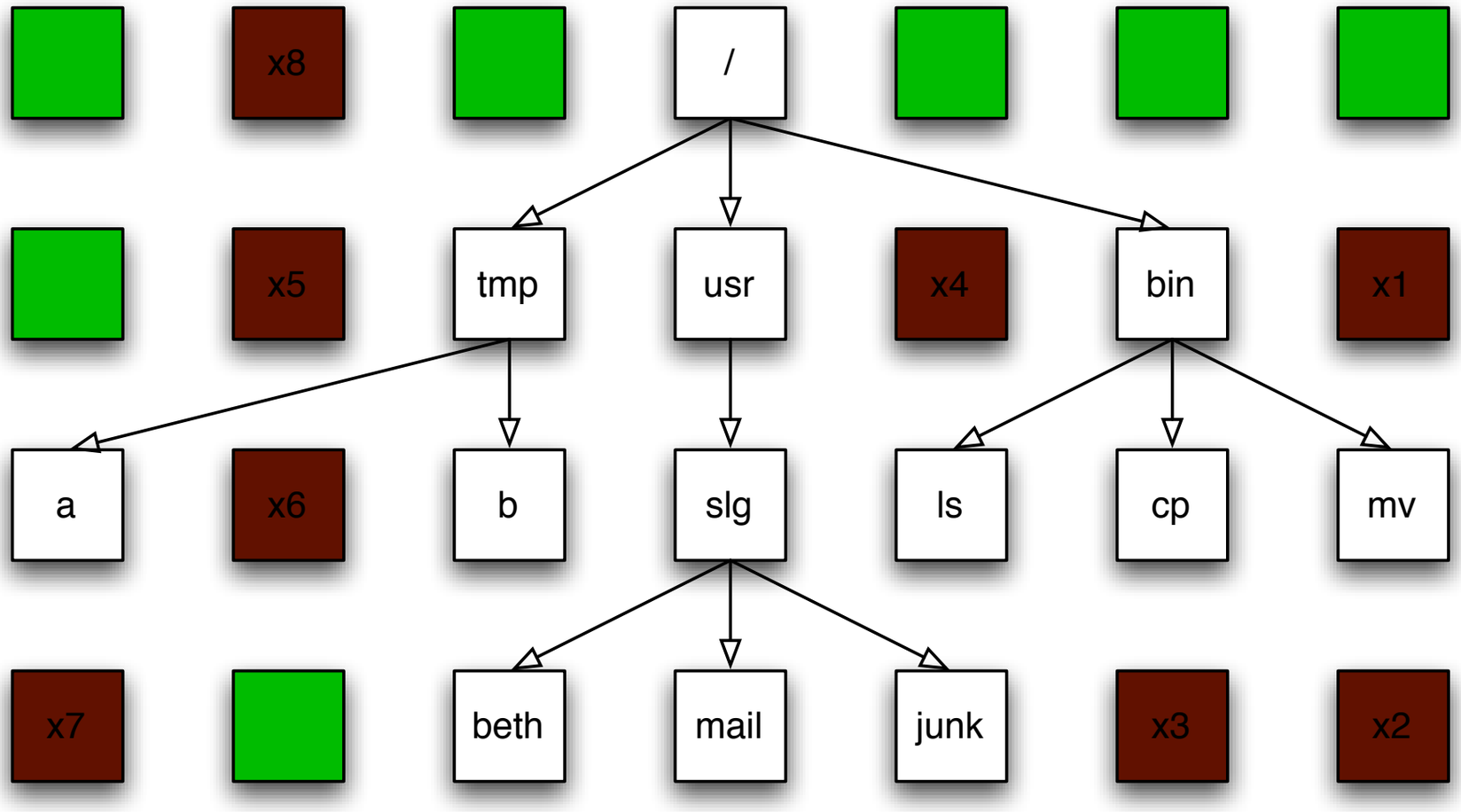
Resident Data

“Deleted data” are on the disk, but can only be recovered with forensic tools.



Deleted Data

Some sectors are blank. They have “no data.”

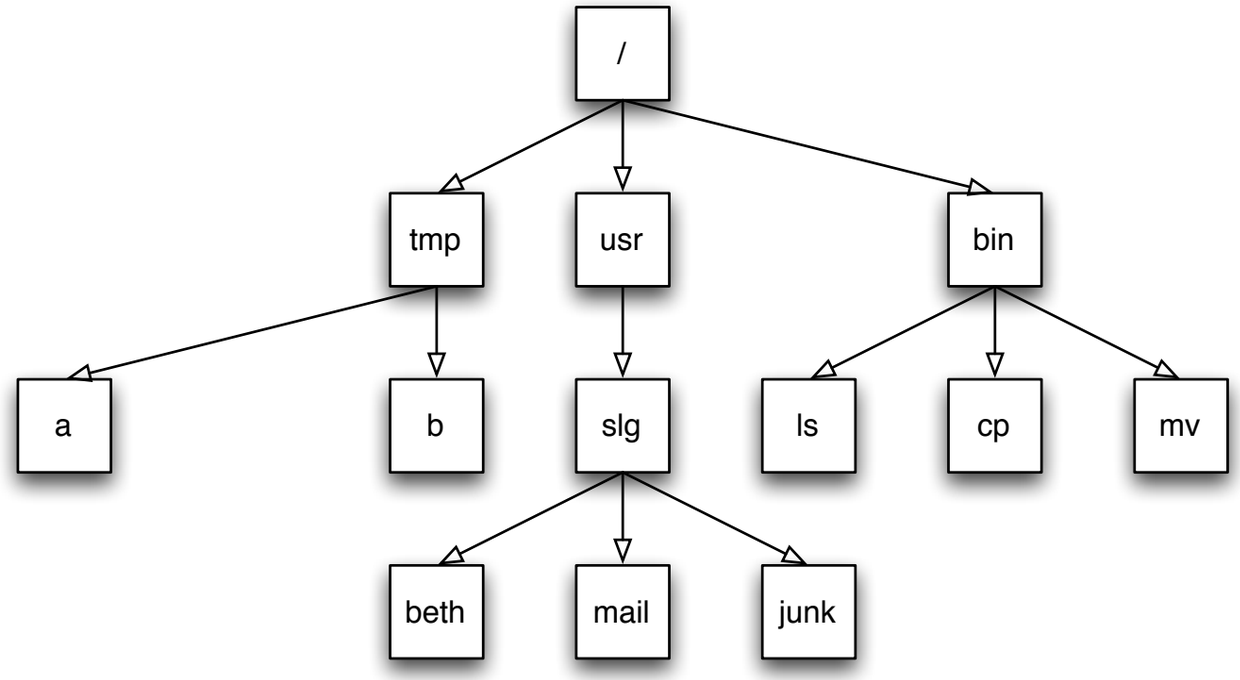


No Data

There are two ways to find email addresses on a drive.

Approach #1:

- Extract text from every file.
- Scan the files with regular expressions



Approach #2:

- Extract text from the “bulk data”
- Scan the text with regular expressions

```
$ cat /dev/disk1 | strings | grep '[a-zA-Z]+@[\-a-zA-Z._]+'
```

Email addresses are extracted from *document files* by converting to text then scanning with regular expressions.

File ➳ Text ➳ RegEx ➳ Email Addresses

Folders.pst

Mother.JPG

Presentation.pptx

Sequestration.docx

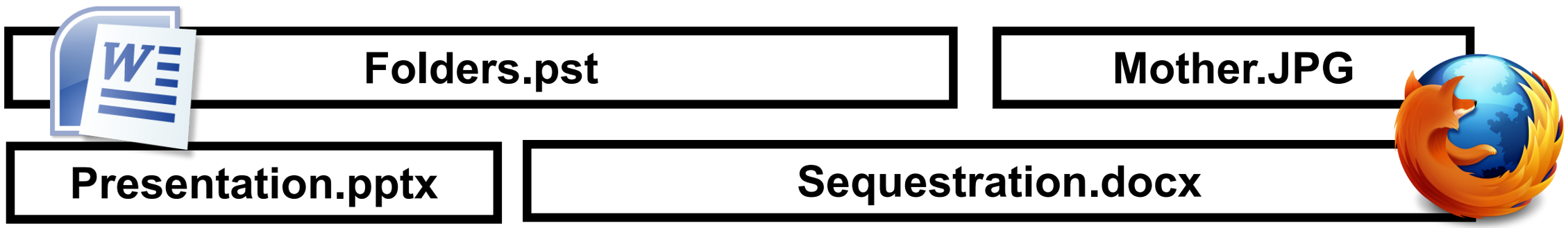
➳

TEXT TEXT TEXT
XYZ@COMPANY.
COM TEXT TEXT
TEXT TEXT

➳ **XYZ@COMPANY.COM**

Regular expressions can also extract email addresses from data not in files — “bulk data.”

[bulk data] ➳ RegEx ➳ Email Addresses




```
a097 83a1 ed96 26a6 3c69 3d0f 750a 2399   ......&.<i=.u.#.
a2b5 bea7 692f 5847 a38a dd53 082c add5   ....i/XG...S.,..
5061 b64c 721d 864b 90b6 b55f bb04 735c   Pa.Lr..K..._..s\
9448 6730 5453 df64 813e b603 5795 2242   .Hg0TS.d.>..W."B
e9c8 7454 7322 7cdc b60e 97af 2f64 2728   ..tTs"|...../d'(
3cfb 84bd 2a84 2dfe 50ea 5935 c349 1513   <XYZ@COMPANY.COM
a9e9 e92c a3f8 6e46 0530 8a88 c7a2 5d2b   ...,..nF.0....]+
d89d 77cc fe1e f637 f3f3 d0af 1b47 c09b   ..w....7.....G..
```

It’s easy to see email addresses in bulk data.

Folders.pst

Mother.JPG

Presentation.pptx

Sequestration.docx

```
a097 83a1 ed96 26a6 3c69 3d0f 750a 2399   ......&.<i=.u.#.
a2b5 bea7 692f 5847 a38a dd53 082c add5   ....i/XG...S.,..
5061 b64c 721d 864b 90b6 b55f bb04 135c   Pa.Lr..K..._..s\
9448 6730 5453 df64 813e b603 5795 1242   .Hg0TS.d.>..W."B
e9c8 7454 7322 7cdc b                     ..tTs"|...../d'(
3cfb 84bd 2a84 2dfe 5                     <XYZ@COMPANY.COM
a9e9 e92c a3f8 6e46 0                     ...,..nF.0....]+
d89d 77cc fe1e f637 f313 d0af 1b47 c90b   ..w....7.....G..
```

XYZ@company.com

Every email address is a sequence of bytes.

A simple email address:

XYZ@company.com

Stored on disk / in memory as 15 bytes:

X Y Z @ c o m p a n y . c o m

Each byte is 8-bits. Range is 0-255

88 89 90 64 99 111 109 112 97 110 121 46 99 111 109

Normally bytes are displayed in hexadecimal notation:

58 59 5a 40 63 6f 6d 70 61 6e 79 2e 63 6f 6d

This is UNICODE

Every email address is a sequence of bytes.

A simple email address:

XYZ@company.com

Stored on disk / in memory as 15 bytes:

X	Y	Z	@	c	o	m	p	a	n	y	.	c	o	m

Each byte is 8-bits. Range is 0-255

88	89	90	64	99	111	109	112	97	110	121	46	99	111	109

Normally bytes are displayed in hexadecimal notation:

58	59	5a	40	63	6f	6d	70	61	6e	79	2e	63	6f	6d

This is UNICODE

Byte sequences can be encoded in many ways.

XYZ@company.com

- Unicode: “XYZ@company.com”

```
58 59 5a 40 63 6f 6d 70 61 6e 79 2e 63 6f 6d
```

- Base 16: “58595a40636f6d70616e792e636f6d0a”

```
3538 3539 3561 3430 3633 3666 3664 3730  58595a40636f6d70
3631 3665 3739 3265 3633 3666 3664 3061  616e792e636f6d0a
```

- Base 64: “WFlaQGNvbXBhbnkuY29tCg===”

```
5746 6c61 5147 4e76 6258 4268 626e 6b75  WFlaQGNvbXBhbnku
5932 3974 4367 3d3d 3d0a                 Y29tCg===.
```

- Compression: echo “XYZ@company.com” | compress | xxd

```
1f9d 9058 b268 0132 e64d 1b38 61dc e471  ...X.h.2.M.8a..q
51b0 8d02                                Q...
```

Compression works by eliminating repeated sequences:

Computers use compression to save memory:

```
5859 5a40 636f 6d70 616e 792e 636f 6d20  XYZ@company.com
4142 4340 636f 6d70 616e 792e 636f 6d20  ABC@company.com
4445 4640 636f 6d70 616e 792e 636f 6d20  DEF@company.com
```

Compressed with “gzip:”

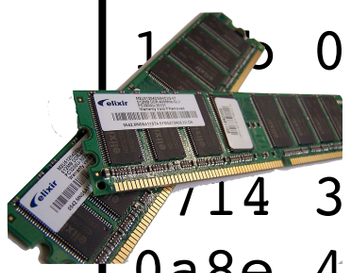
```
1f8b 0800 0000 0000 0203 8b88 8c72 48ce  .............rH.
cf2d 48cc abd4 03d2 0a8e 4ece 287c 1757  .-H.......N.(|.W
3714 3e00 b455 c1c5 3000 0000            7.>..U..0...
```

Compressed email addresses do not “look” like email addresses!

—*Forensic tools must decompress FIRST to identify compressed email addresses.*

It’s hard to see compressed email address in bulk data.

```
e327 962d 6450 3d91 c945 3bed 97a6 a4cd   .'.-dP=..E;.....
1    0800 0000 0000 0203 8b88 8c72 48ce   .............rH.
     8cc  abd4 03d2 0a8e 4ece 287c 1757   .-H.......N.(|.W
7714 3e00 b455 c1c5 3000 0000 0000 0000   7.>..U..0.......
0a8e 4ece 287c 1757 3714 3e00 a175 10ed   ..N.(|.W7.>..u..
```



Folders.pst

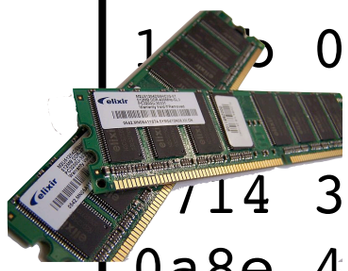
Mother.JPG

Presentation.pptx

Sequestration.docx

```
a097 83a1 ed96 26a6 3c69 3d0f 750a 2399   ......&.<i=.u.#.
a2b5 bea7 692f 5847 a38a dd53 082c add5   ....i/XG...S.,..
5061 b64c 721d 864b 90b6 b55f bb04 735c   Pa.Lr..K..._..s\
9448 6730 5453 df64 813e b603 5795 2242   .Hg0TS.d.>..W."B
e    7454 7322 7cdc b60e 97af 2f64 2728   ..tTs"|...../d'(
     94bd 2a84 2dfe 50ea 5935 c349 1513   <XYZ@COMPANY.COM
     e92c a3f8 6e46 0530 8a88 c7a2 5d2b   ...,..nF.0....]+
d89d 77cc fe1e f637 f3f3 d0af 1b47 c09b   ..w....7.....G..
```

It’s hard to see compressed email address in bulk data.



```
e327 962d 6450 3d91 c945 3bed 97a6 ....cd   .'.-dP=..E;.....
1... 0800 0000 0000 0... ........   ..............rH.
.... 8cc abd4 03d2 0... ........   .-H.......N.(|.W
7714 3e00 b455 c1c5 3... ........   7.>..U..0.......
0a8e 4ece 287c 1757 3714 3e00 a175 ....ed   ..N.(|.W7.>..u..
```

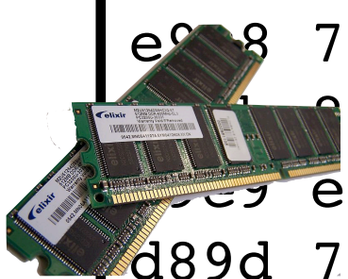
XYZ@company.com
ABC@company.com
DEF@company.com

Folders.pst

Mother.JPG

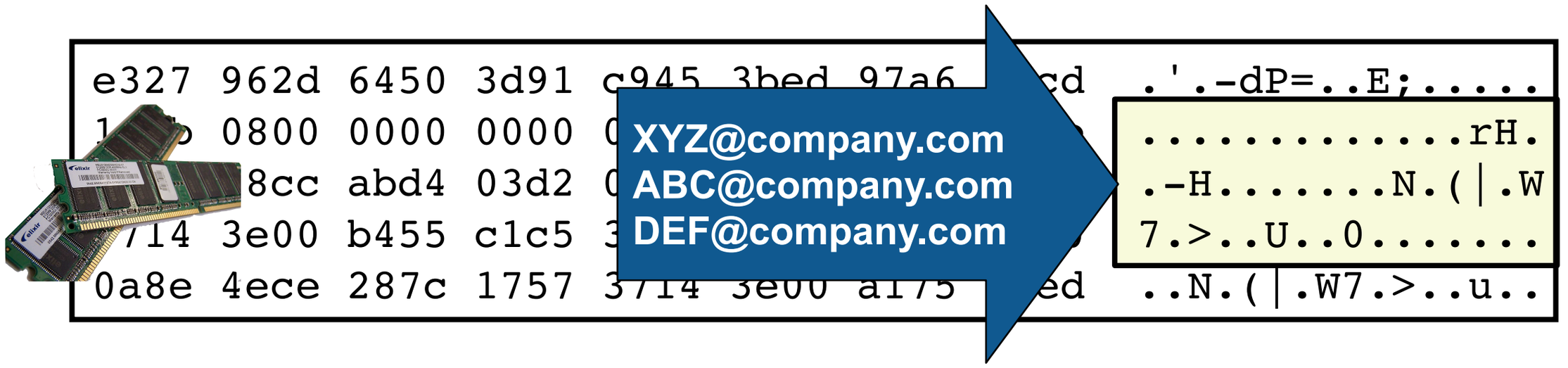
Presentation.pptx

Sequestration.docx



```
a097 83a1 ed96 26a6 3c69 3d0f 750a 2399   ......&.<i=.u.#.
a2b5 bea7 692f 5847 a38a dd53 082c add5   ....i/XG...S.,..
5061 b64c 721d 864b 90b6 b55f bb04 735c   Pa.Lr..K..._..s\
9448 6730 5453 df64 813e b603 5795 2242   .Hg0TS.d.>..W."B
e... 7454 7322 7cdc b60e 97af 2f64 2728   ..tTs"|...../d'(
.... 24bd 2a84 2dfe 50ea 5935 c349 1513   <XYZ@COMPANY.COM
.... e92c a3f8 6e46 0530 8a88 c7a2 5d2b   ...,..nF.0....]+
d89d 77cc fe1e f637 f3f3 d0af 1b47 c09b   ..w....7.....G..
```

Existing commercial digital forensic tools ignore compressed email addresses in bulk data.



Today's tools ignore most kinds of encoding.

- Compression:
  - —*zlib (gzip, ZIP)*
  - —*RAR*
  - —*Windows Hibernation (Microsoft Xpress)*
- Simple obfuscation
  - —*ROT13, XOR(255)*

Implement “optimistic decoding” by attempting to decode every byte with every algorithm.

Input sector:

```
e327 962d 6450 3d91 c945 3bed 97a6 a4cd  .'.-dP=..E;.....
1f8b 0800 0000 0000 0203 8b88 8c72 48ce  .............rH.
cf2d 48cc abd4 03d2 0a8e 4ece 287c 1757  .-H.......N.(|.W
3714 3e00 b455 c1c5 3000 0000 0000 0000  7.>..U..0.......
0a8e 4ece 287c 1757 3714 3e00 a175 10ed  ..N.(|.W7.>..u..
```

Optimistic decoding in theory:

```
Decompress(“e3 27 96 2d ...”)
Decompress(“27 96 2d 64 ...”)
Decompress(“96 2d 64 50 ...”)
```

—*e.g.:*

```
for i in range(len(buf)):
    if start_of_compressed_buffer(buf[i:]):
        try_decompress(buf[i:])
```

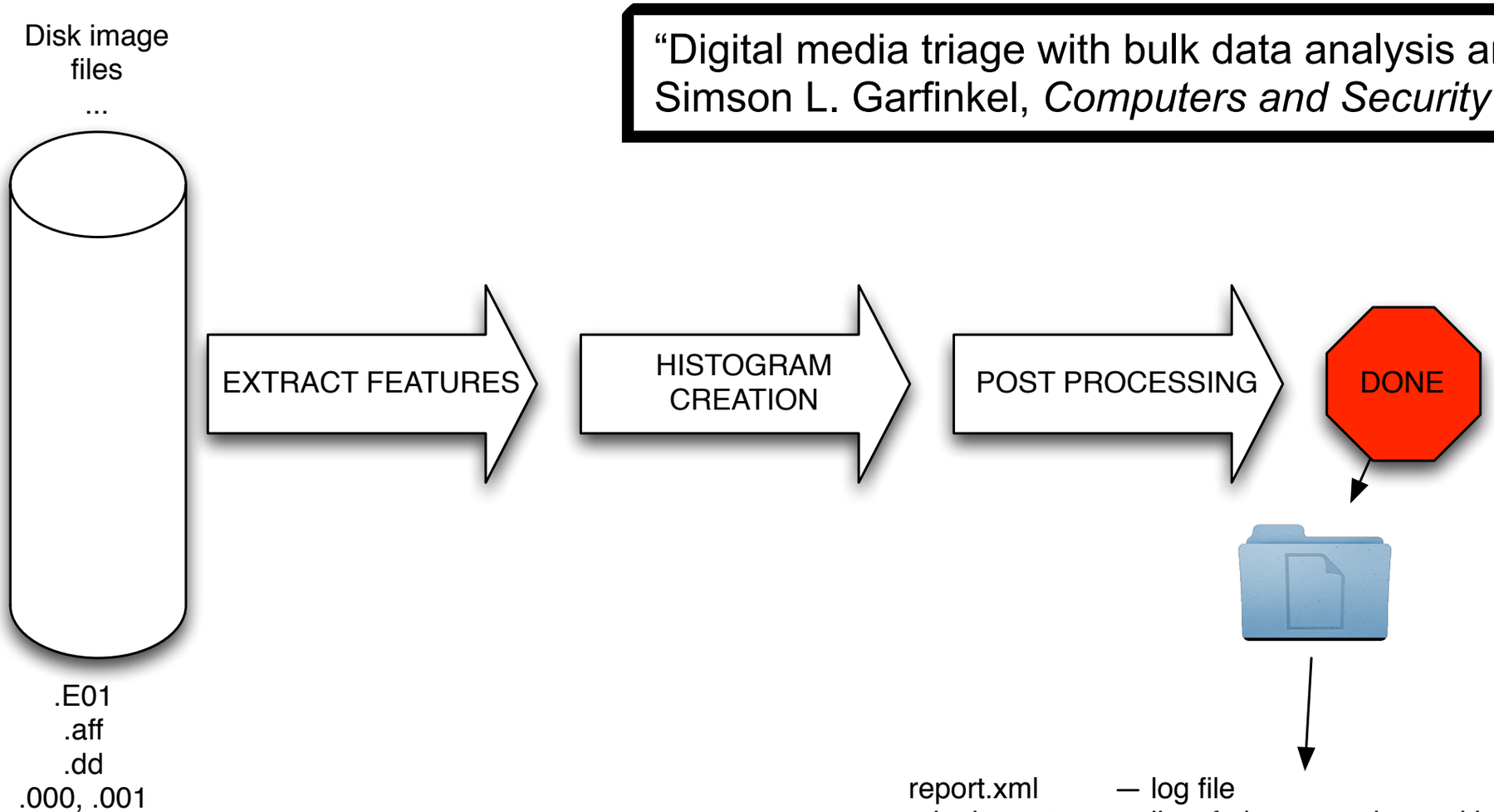
In practice, we write scanners in hand-tuned C++

BE

Extracting encoded data with bulk_extractor

bulk_extractor is a stream forensics program.
It finds and extracts "features" from bulk data.

"Digital media triage with bulk data analysis and bulk_extractor," Simson L. Garfinkel, *Computers and Security 32 (2013) 56-72*



Output is a *directory* containing:

- feature files; histograms; carved objects
- Mostly in UTF-8; some XML
- Can be bundled into a ZIP file and process with bulk_extractor_reader.py

Stream-based disk forensics:
Scan the disk from beginning to end; do your best.

0 1TB

3 hours, 20 min to *read* the data

1. Read all of the blocks in order.
2. Look for information that might be useful.
3. Identify & extract what's possible in a single pass.

Primary advantage of stream-based forensics: Speed

No disk seeking.

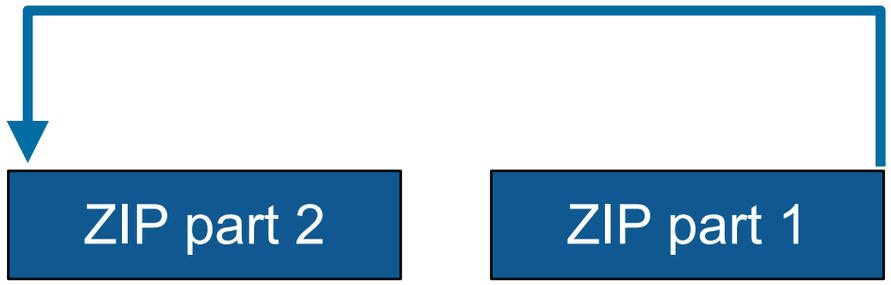
Easy to parallelize:

- Potential to read and process at disk's maximum transfer rate.

Reads all the data — allocated files, deleted files, file fragments.

- Separate metadata extraction required to get the file names.

Primary disadvantage: completeness



Fragmented files won't be recovered:

- Compressed files with part2-part1 ordering (possibly .docx)
- Files with internal fragmentation (.doc but not .docx)

Fortunately, most files are *not* fragmented.

- Individual components of a ZIP file can be fragmented.

Most files that *are* fragmented have carvable internal structure:

- Log files, Outlook PST files, etc.

bulk_extractor: architectural overview

Written in C, C++ and GNU flex

- Command-line tool.
- Linux, MacOS, Windows (compiled with mingw)

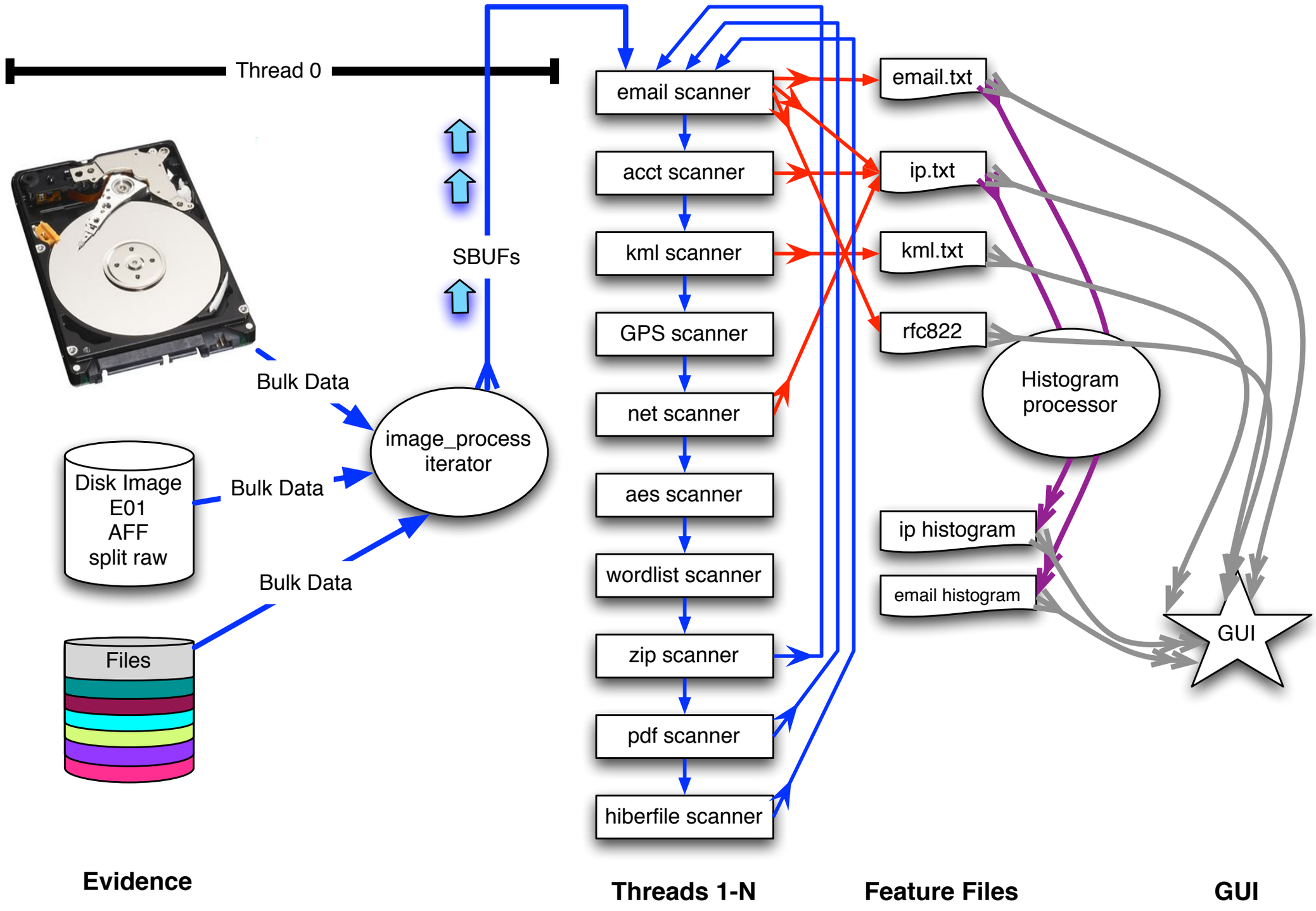
Key features:

- “Scanners” look for information of interest in typical investigations.
- Recursively re-analyzes compressed data.
- Results stored in “feature files”
- Multi-threaded

Java GUI

- Runs command-line tool and views results

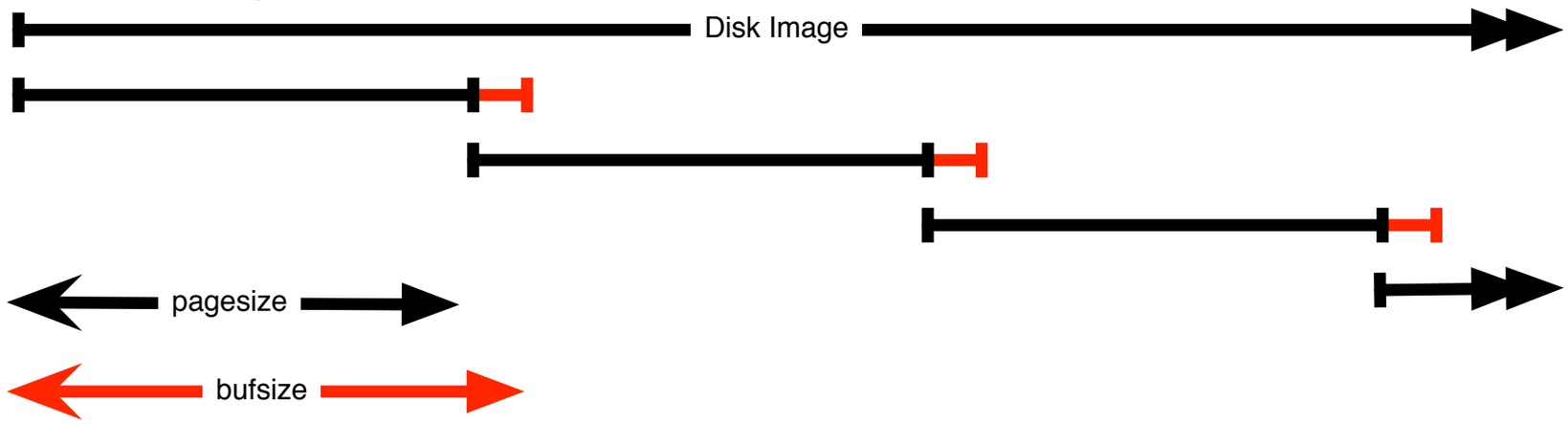
bulk_extractor: system diagram



The “pages” overlap to avoid dropping features that cross buffer boundaries.

The overlap area is called the *margin*.

- Each sbuf can be processed in parallel — they don’t depend on each other.
- Features start in the page but end in the margin are *reported*.
- Features that start in the margin are *ignored* (we get them later)
  - *Assumes that the feature size is smaller than the margin size.*
  - *Typical margin: 1MB*



Entire system is automatic:

- Image_process iterator makes **sbuf_t** buffers.
- Each buffer is processed by every scanner
- Features are automatically combined.

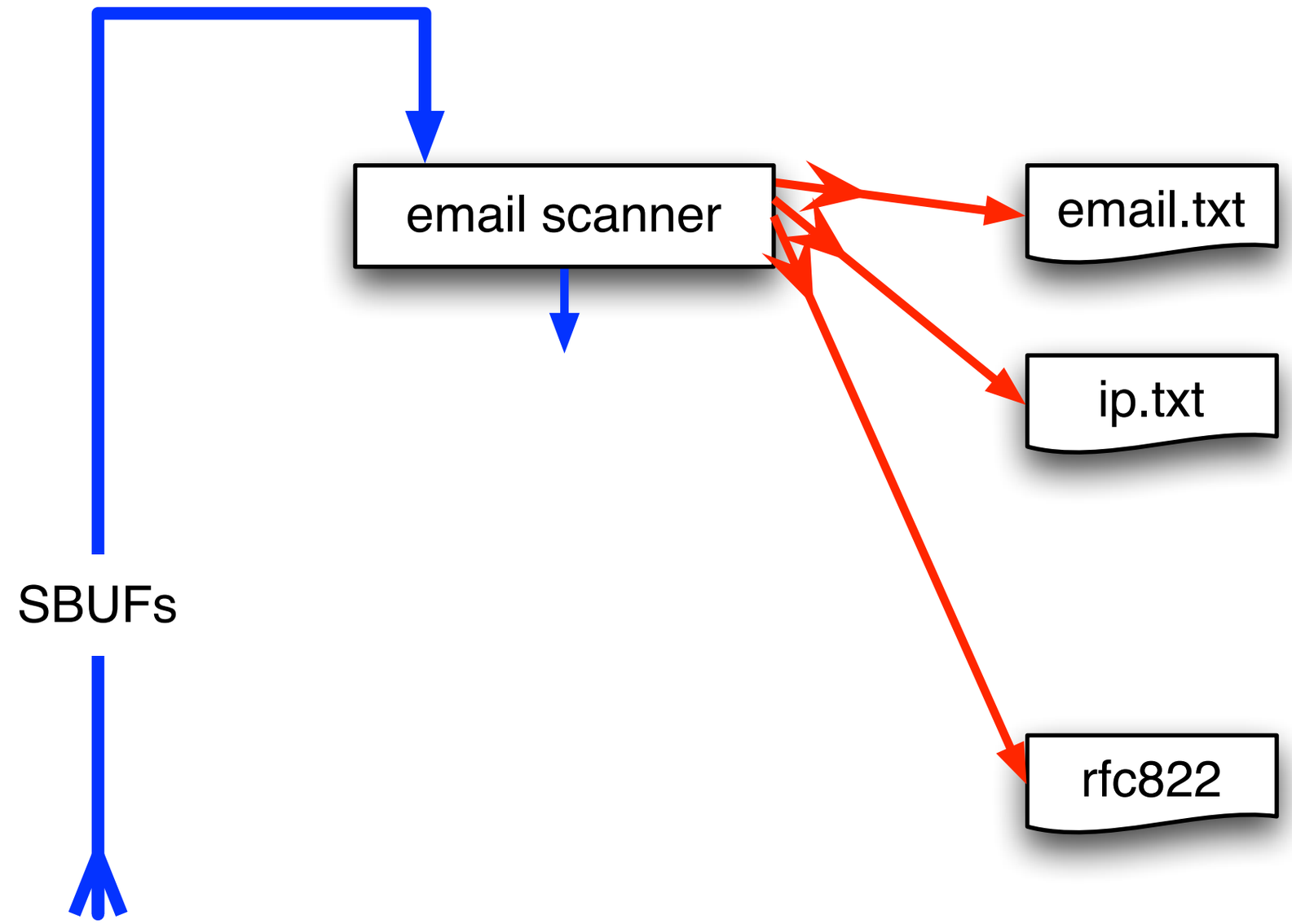
Scanners process an sbuf and extract features

scan_email is the email scanner.

- inputs: **sbuf** objects

outputs:

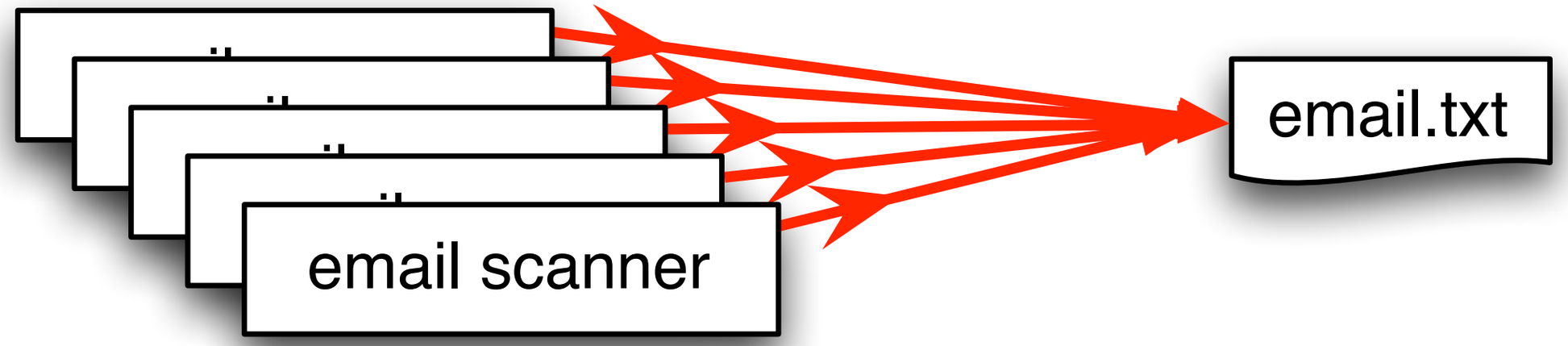
- **email.txt**
  - —*Email addresses*
- **rfc822.txt**
  - —*Message-ID*
  - —*Date:*
  - —*Subject:*
  - —*Cookie:*
  - —*Host:*
- **domain.txt**
  - —*IP addresses*
  - —*host names*



The *feature recording system* saves features to disk.

Feature Recorder objects store the features.

- Scanners are given a (feature_recorder *) pointer
- Feature recorders are *thread safe*.



Features are stored in a *feature file:*

```
48198832  domexuser2@gmail.com    tocol>____<name>domexuser2@gmail.com/Home</name>____
48200361  domexuser2@live.com     tocol>____<name>domexuser2@live.com</name>____<pass
48413829  siege@preoccupied.net   siege) O'Brien <siege@preoccupied.net>_hp://meanwhi
48481542  danilo@gnome.org        Danilo __egan <danilo@gnome.org>_Language-Team:
48481589  gnom@prevod.org         : Serbian (sr) <gnom@prevod.org>_MIME-Version:
49421069  domexuser1@gmail.com    server2.name", "domexuser1@gmail.com");__user_pref("
49421279  domexuser1@gmail.com    er2.userName", "domexuser1@gmail.com");__user_pref("
49421608  domexuser1@gmail.com    tp1.username", "domexuser1@gmail.com");__user_pref("
```

offset **feature** **feature in evidence context**

bulk_extractor has *multiple* feature extractors. Each scanner runs in order. (Order doesn't matter.)

Scanners can be turned on or off

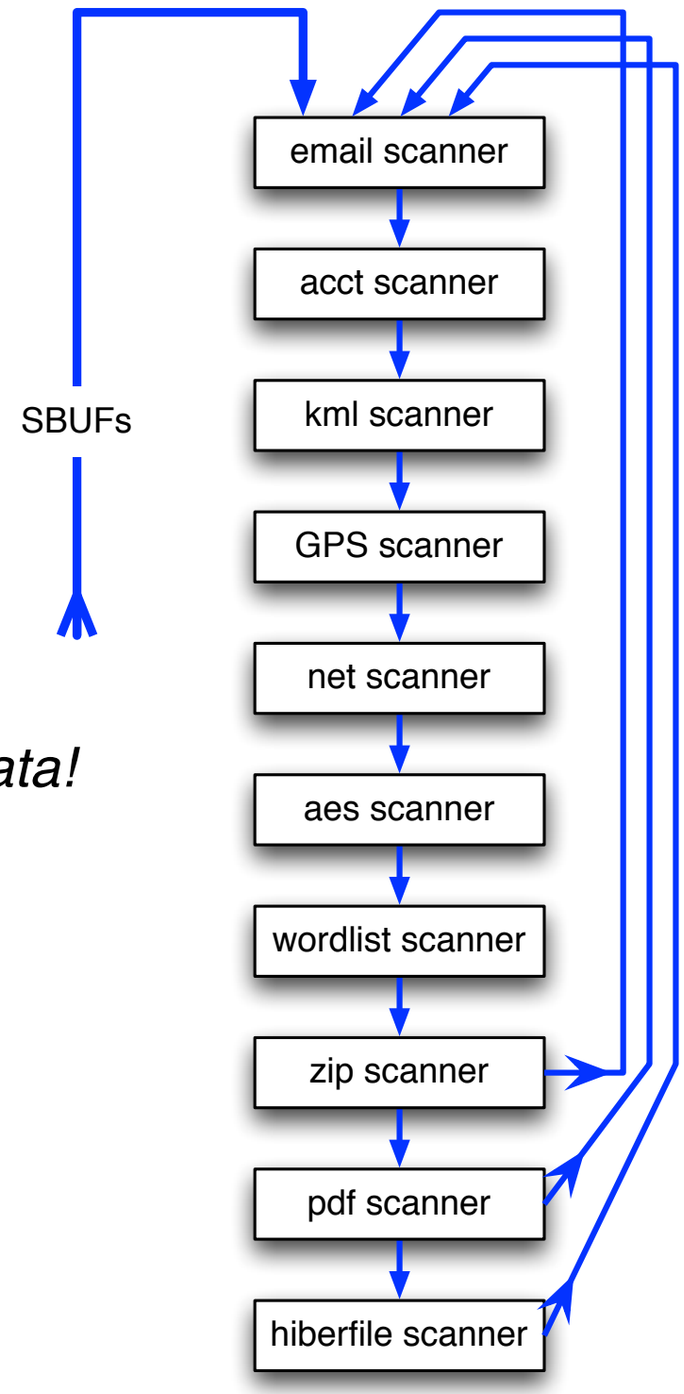
- Useful for debugging.
- AES key scanner is *very slow* (off by default)

Some scanners are *recursive.*

- *e.g.* scan_zip will find zlib-compressed regions
- An **sbuf** is made for the decompressed data
- The data is re-analyzed by the other scanners
  - *This finds email addresses in compressed data!*

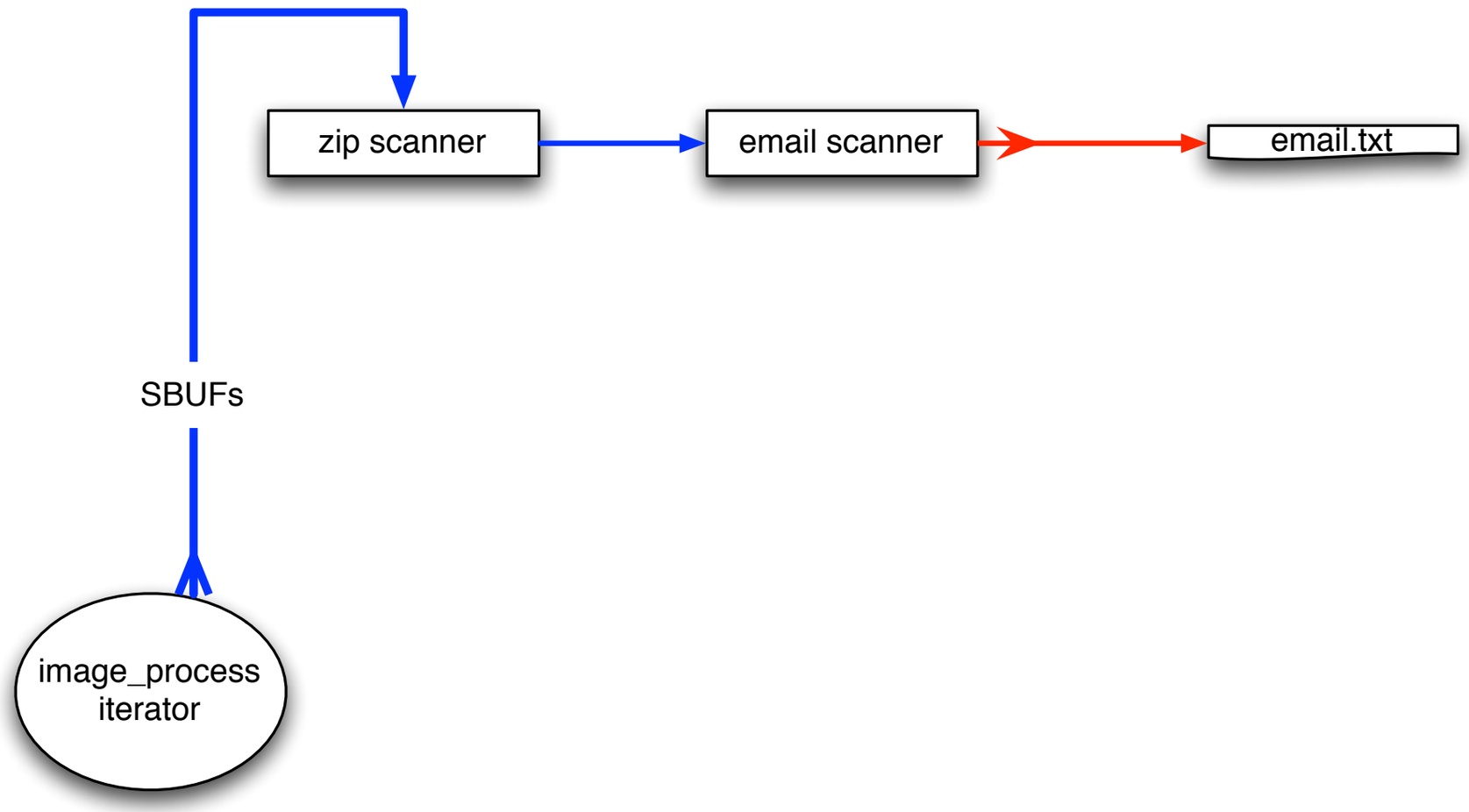
Recursion used for:

- Decompressing ZLIB, Windows HIBERFILE,
- Extracting text from PDFs
- Handling compressed browser cache data



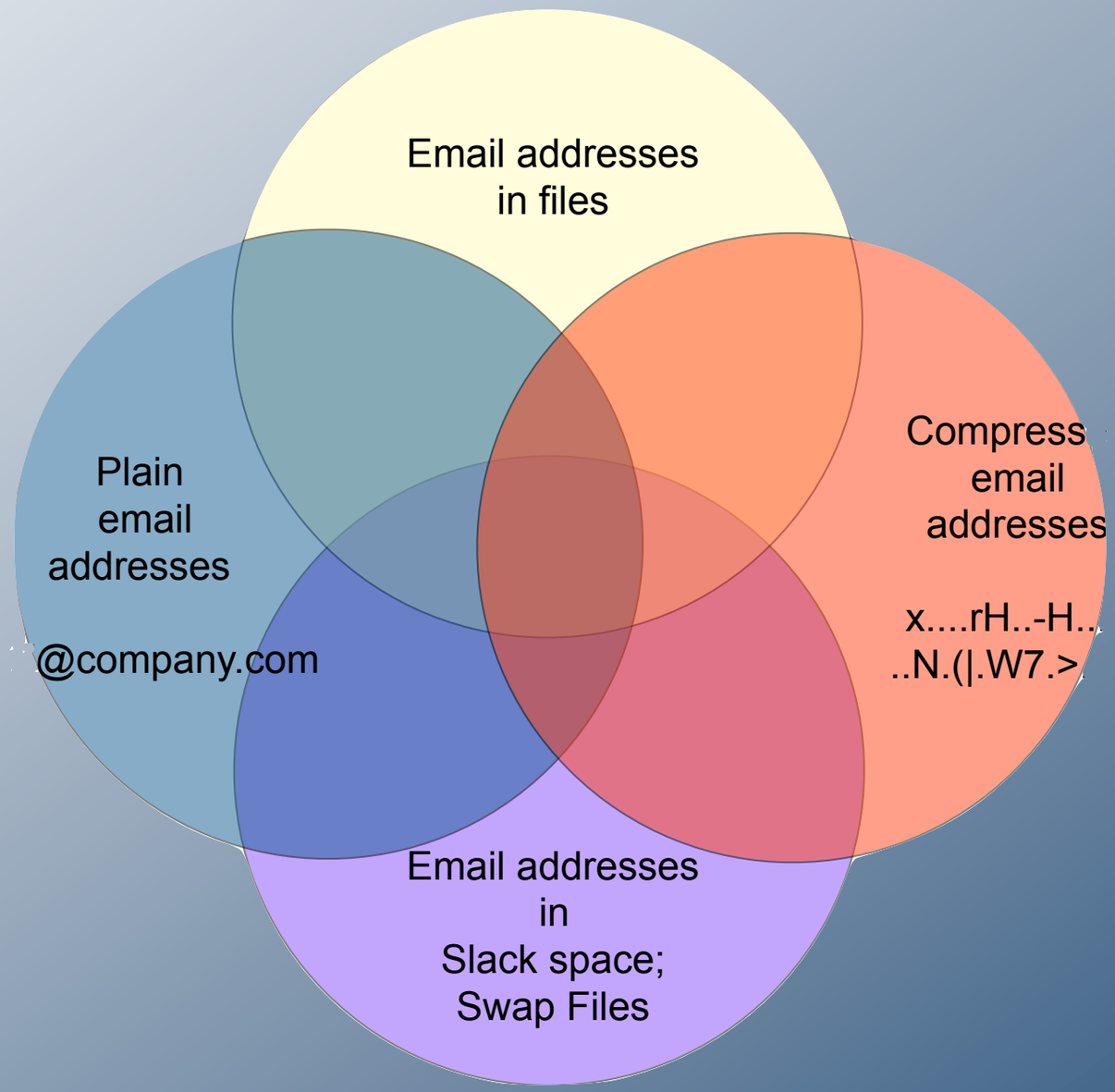
Recursion requires a *new way* to describe offsets. bulk_extractor introduces the "forensic path."

Consider an HTTP stream that contains a GZIP-compressed email:



We can represent this as:

```
11052168704-GZIP-3437   live.com    eMn='domexuser1@live.com';var srf_sDispM
11052168704-GZIP-3475   live.com    pMn='domexuser1@live.com';var srf_sPreCk
11052168704-GZIP-3512   live.com    eCk='domexuser1@live.com';var srf_sFT='<
```



What is the prevalence of encoded identity information?

Email addresses can be in files

Files

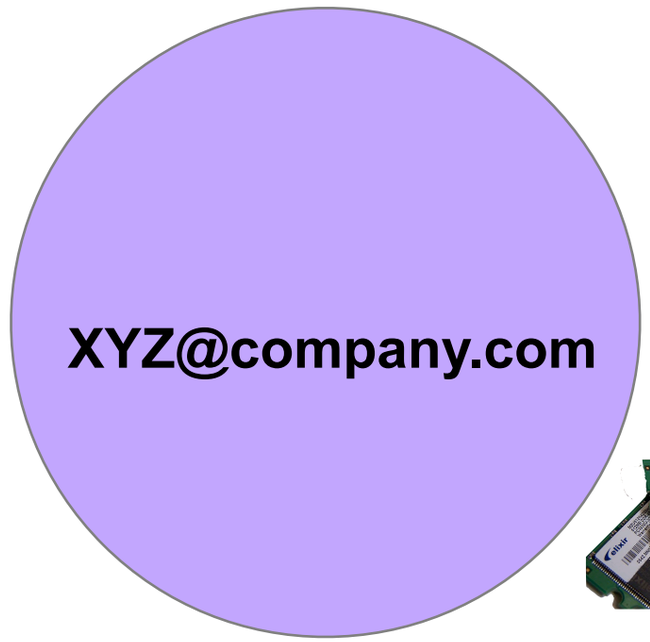
- Documents
- Address book
- Email messages

ABC@company.com
DEF@company.com

Browser Cache:

- Web mail
- Facebook Data

Email addresses can be in non-file disk sectors



Swap Files
Hibernation Files
File fragments

Some may be in *both* files and in non-files. (A file that's read into RAM before the system hibernates.)

ABC@company.com
DEF@company.com

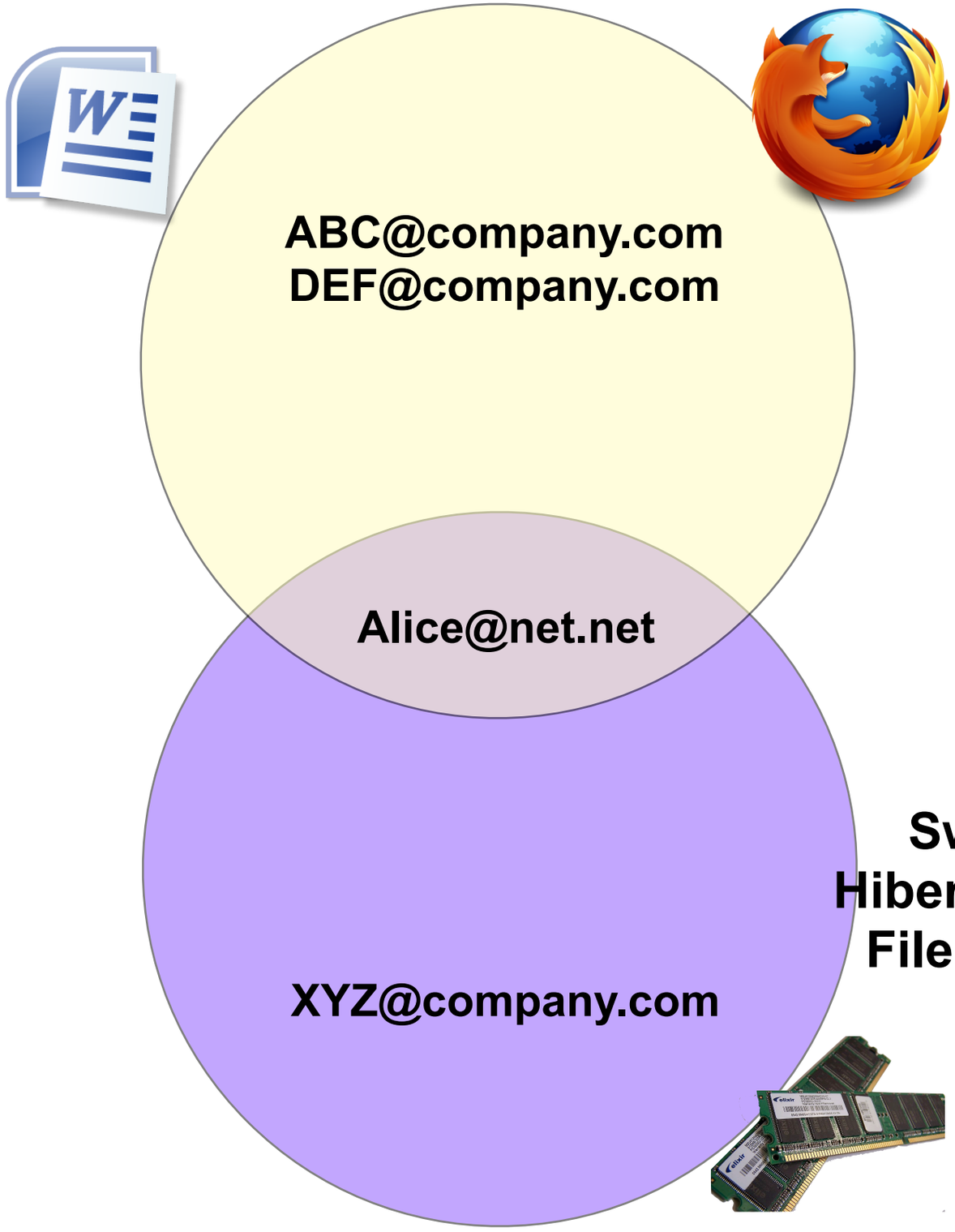
Alice@net.net

Alice@net.net

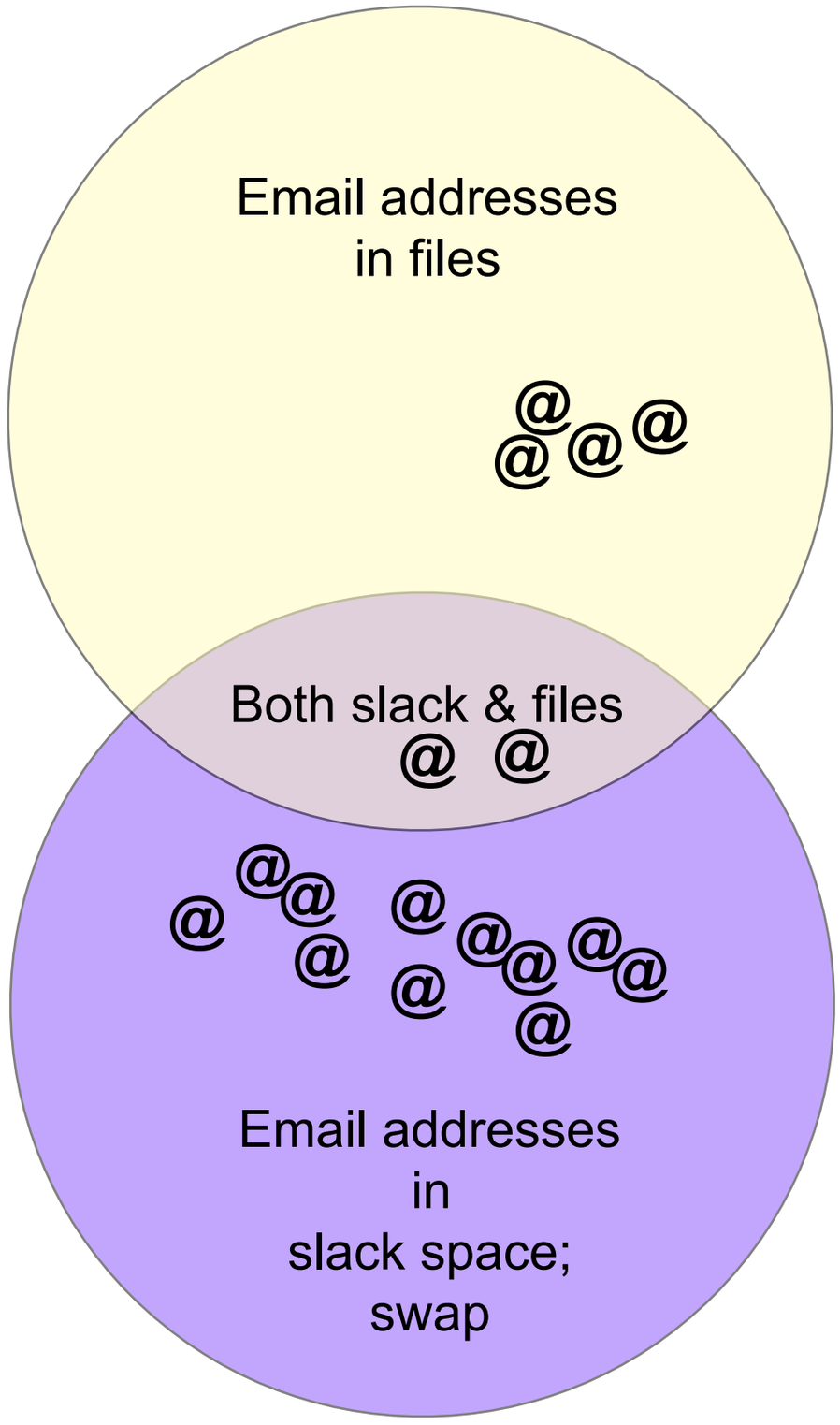
XYZ@company.com

Swap Files
Hibernation Files
File fragments

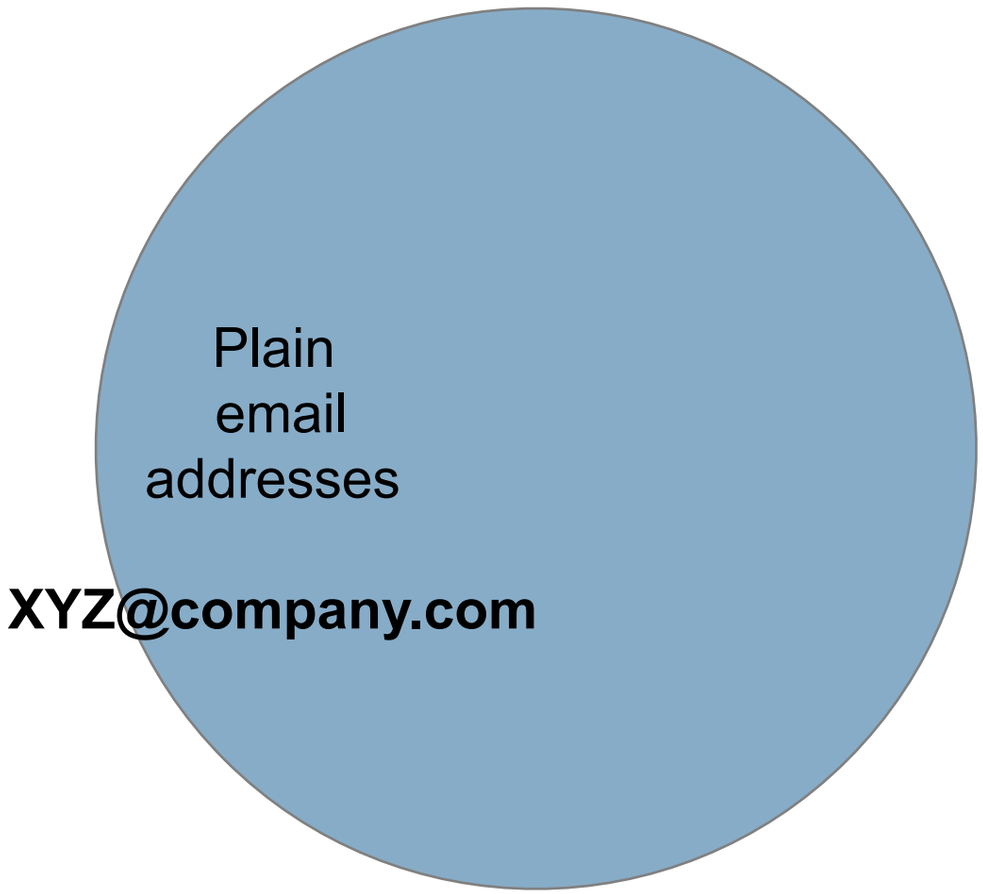
This diagram represents email addresses on media.



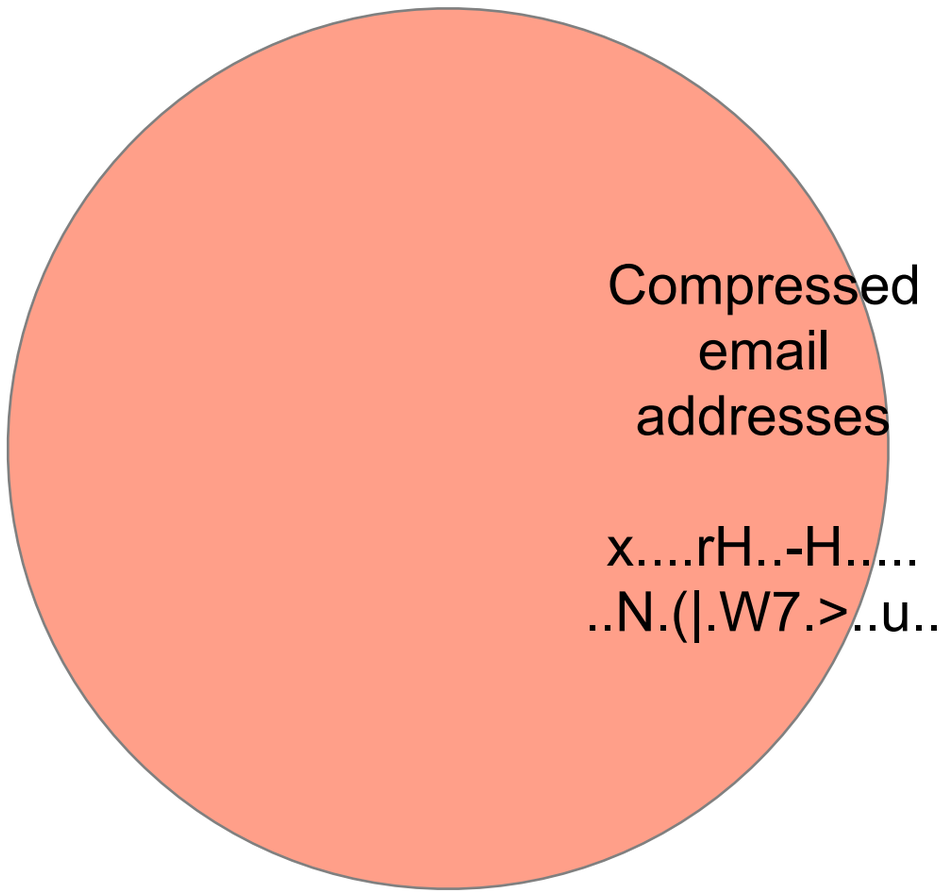
The number in each region depends on the media.



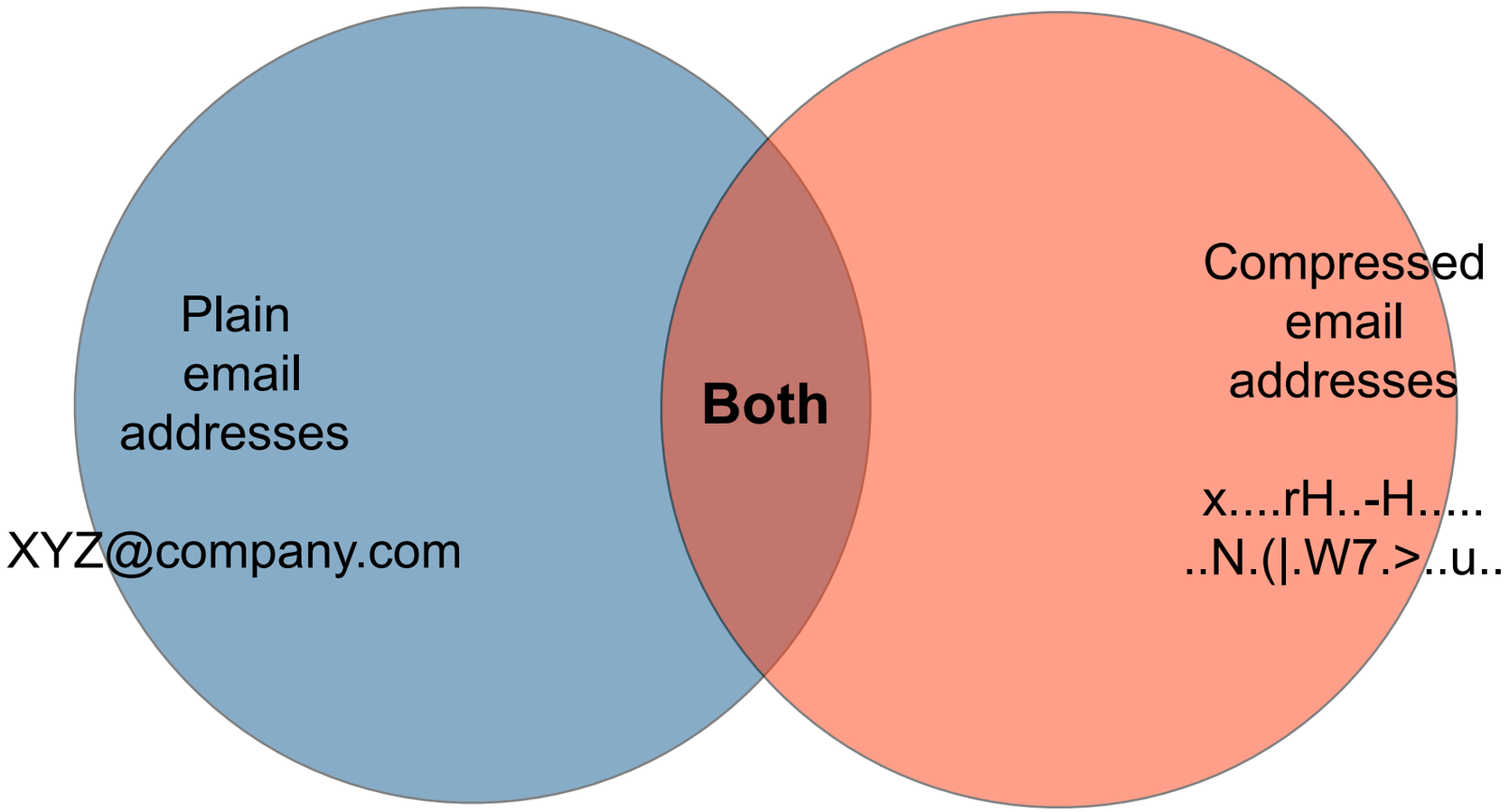
Email addresses can be plain text.
“XYZ@company.com”



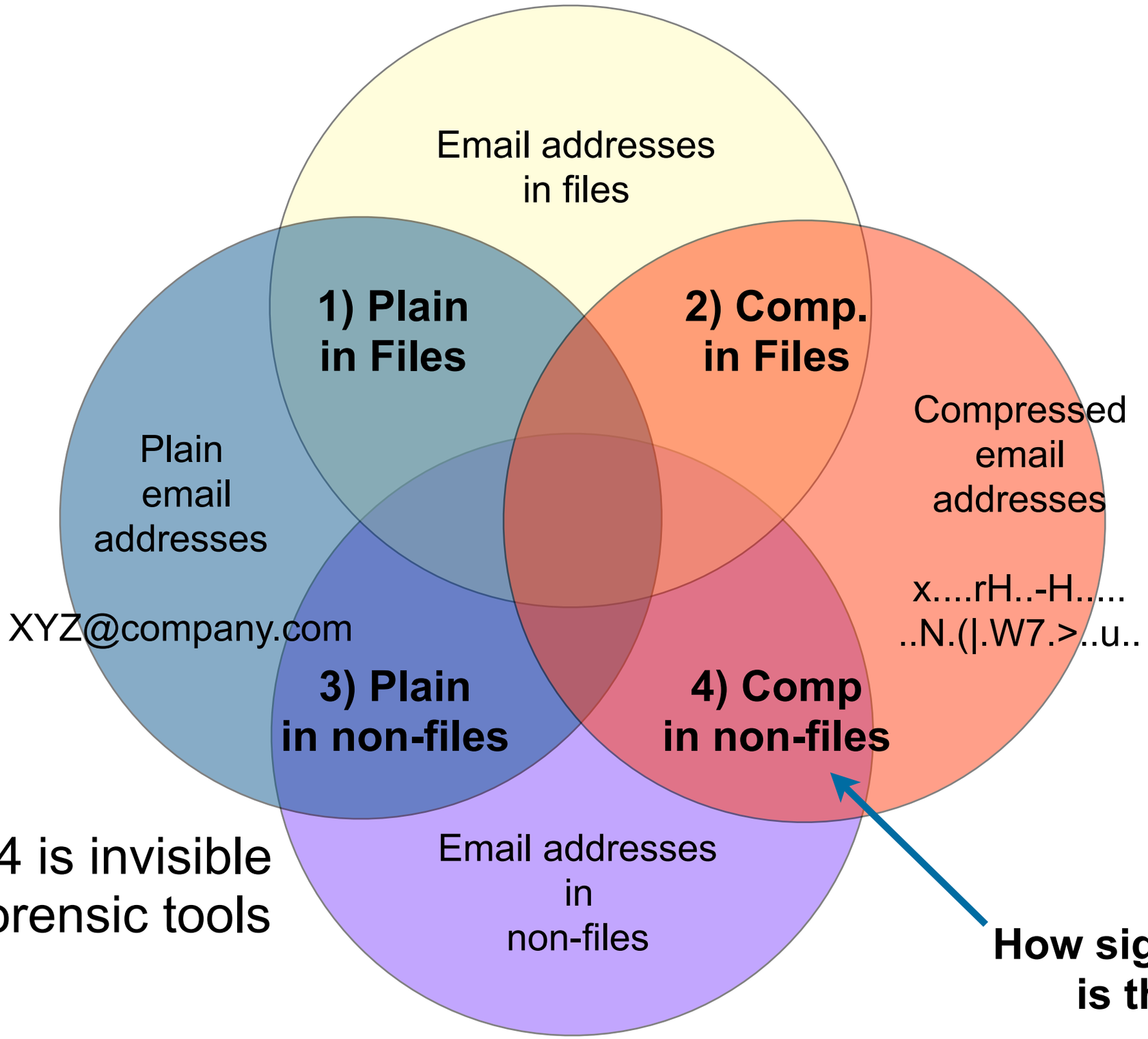
Email addresses can be compressed or encoded. “x....rH..-H.....N.(|.W7.>..u..”



Each address can be present *plain*, *compressed*, or both.



There are four different conditions for an email address on the media.

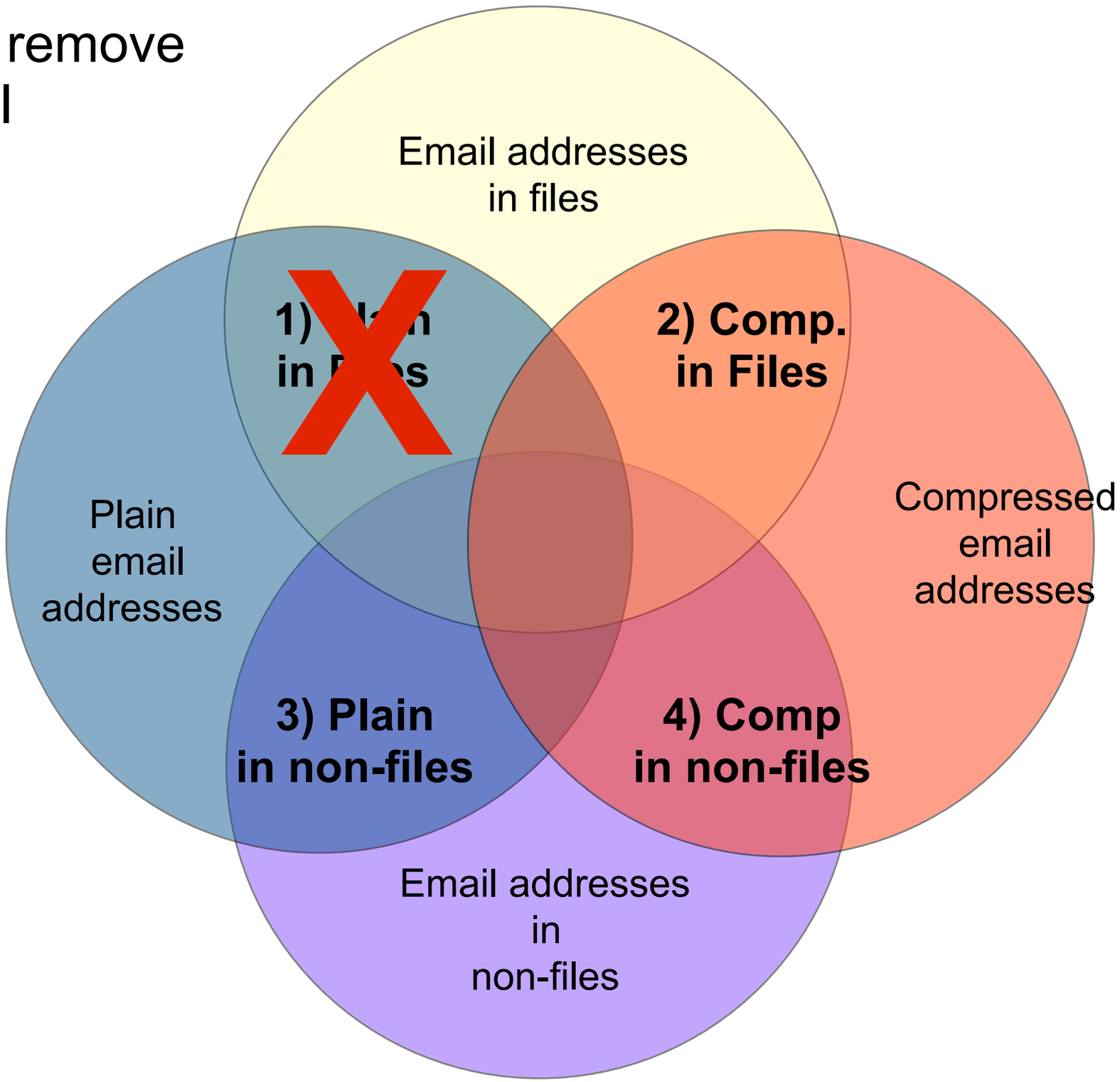


Condition #4 is invisible to today's forensic tools

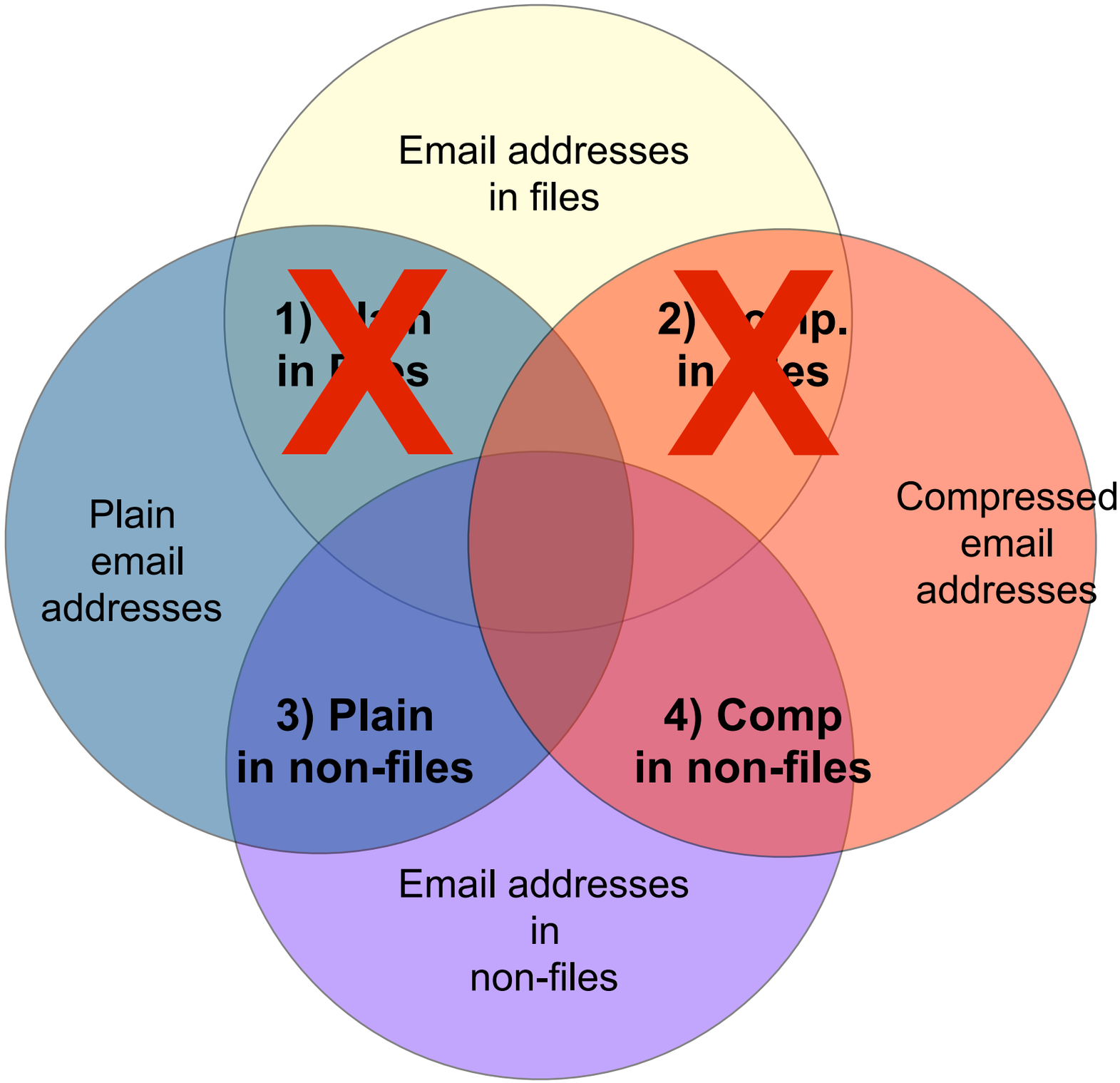
How significant is this?

We devised an experiment to determine the size of condition #4 for a specific drive.

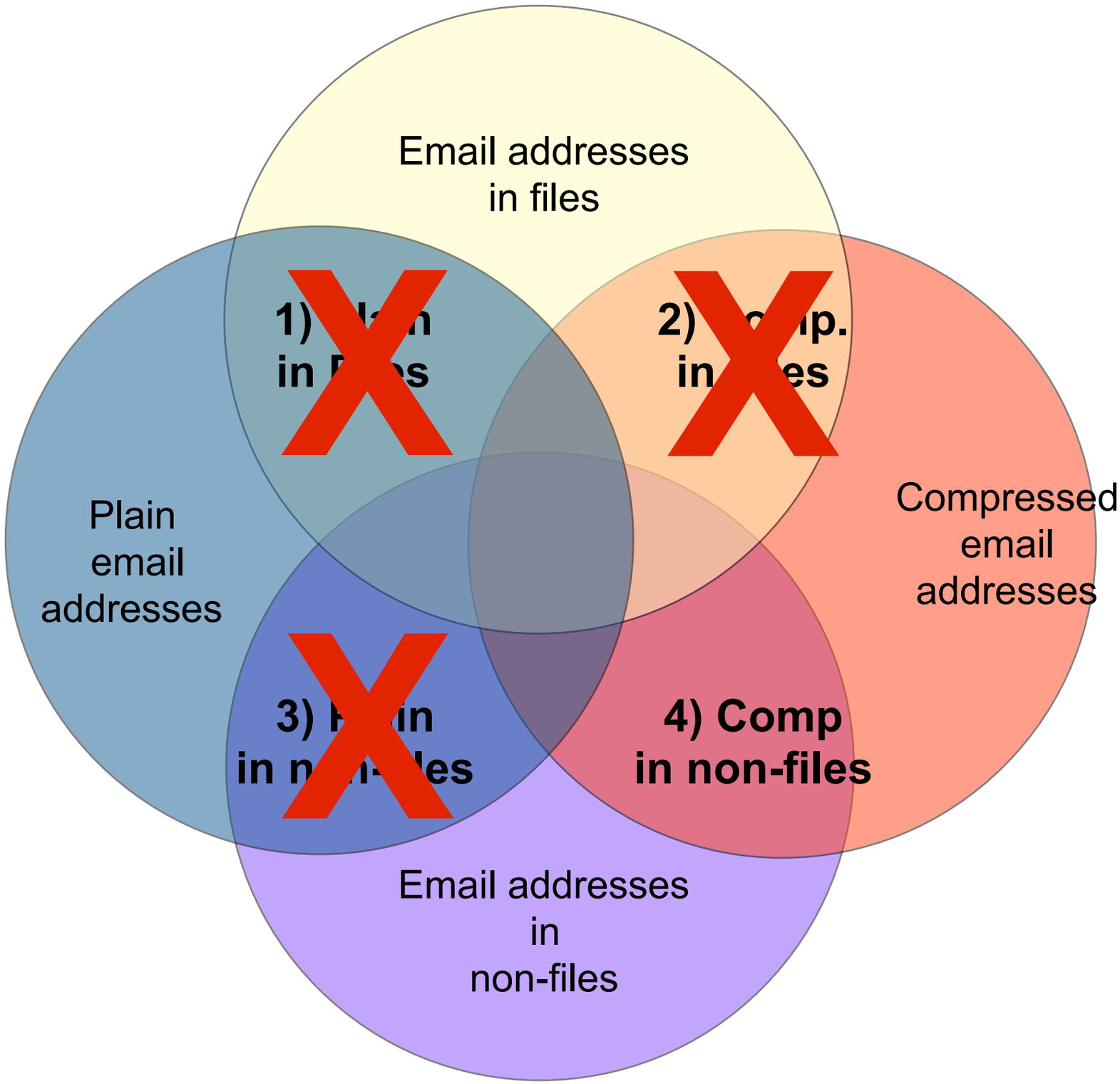
First, find and remove the plain email addresses in files.



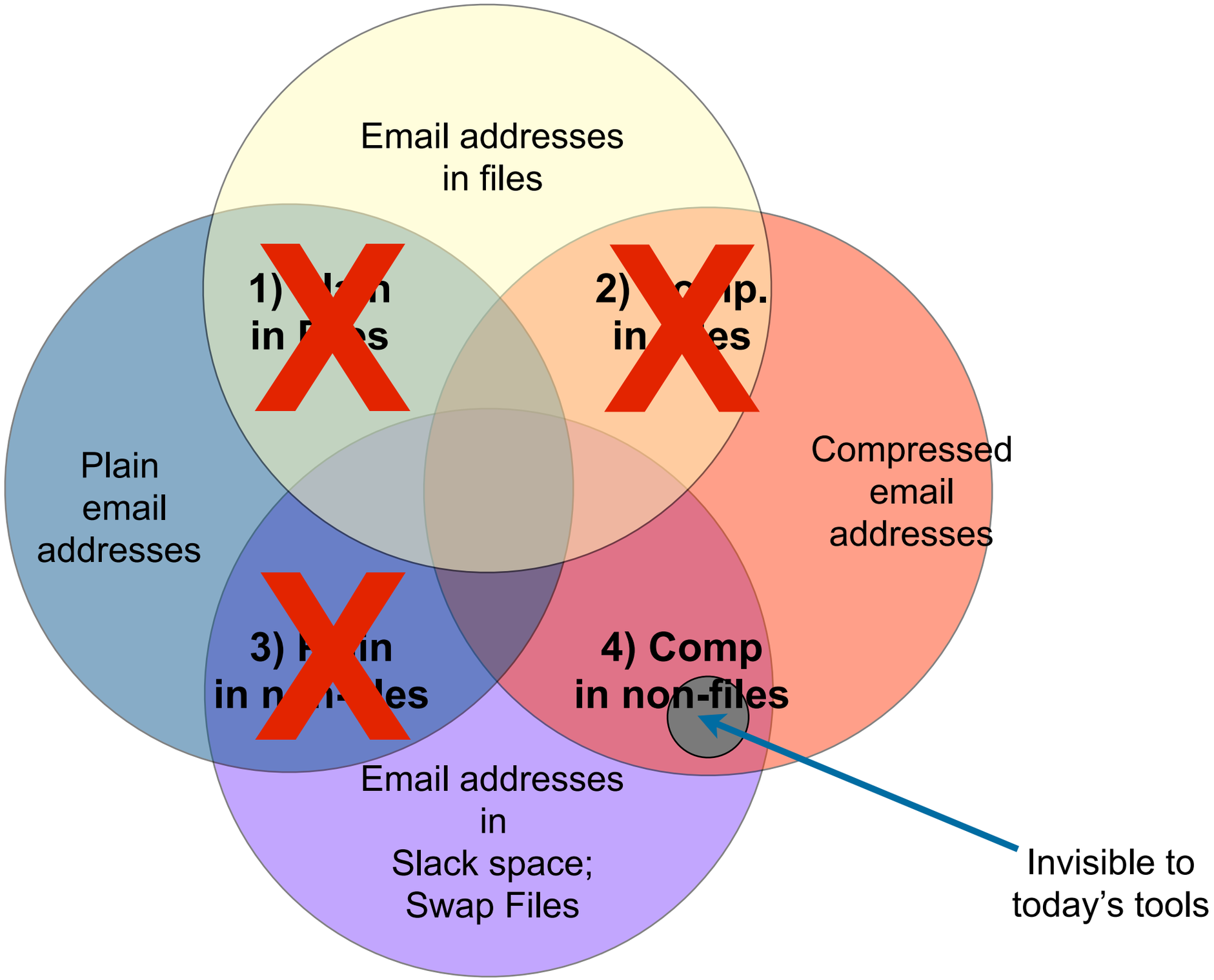
...Remove the addresses compressed and in files....



...Remove email addresses that are not compressed.

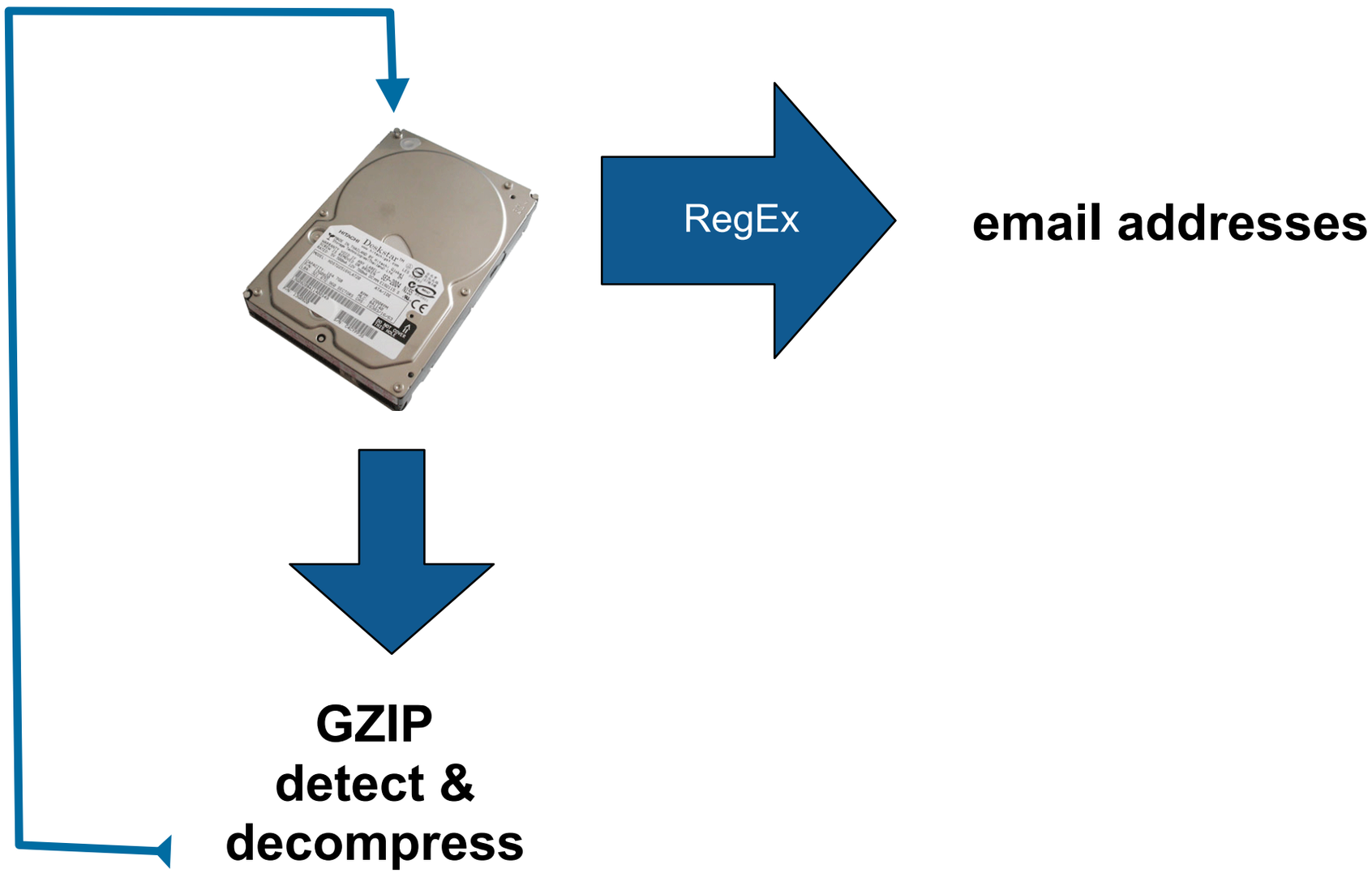


...those that remain are the “invisible” email addresses.



bulk_extractor is an experimental email extraction tool.

"Digital media triage with bulk data analysis and bulk_extractor,"
Simson L. Garfinkel, *Computers and Security 32 (2013) 56-72*



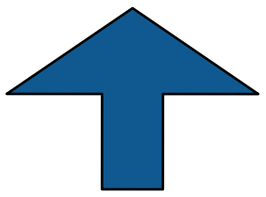
bulk_extractor can find both plain and compressed text.

“Feature files” contain the extracted email addresses.

```
# UTF-8 Byte Order Marker; see http://unicode.org/faq/utf_bom.html
#
@
...
392175418    WindowsXP@gn.microsoft.com    Name=WindowsXP@gn.microsoft.com\015\012
...
3772517888-GZIP-28322  user@company.com  onterey-<nobr>user@company.com</nobr>
...
```

Offset

Feature



Context

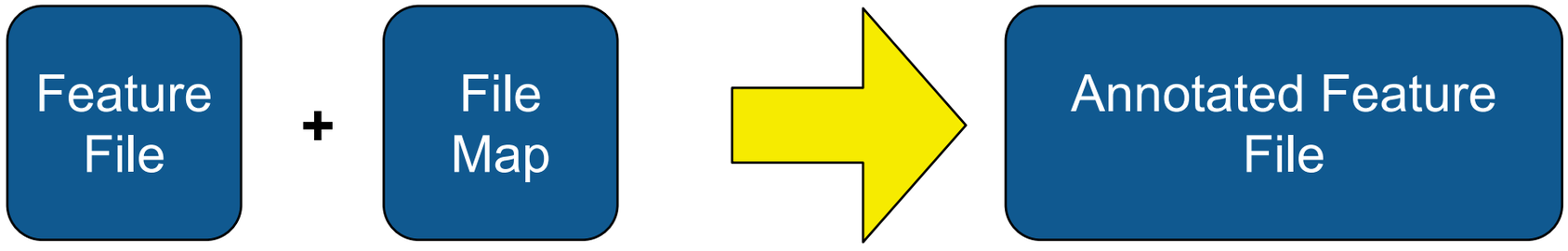
Plain text features have numeric offsets:

392175418

Compressed features will indicate the algorithm:

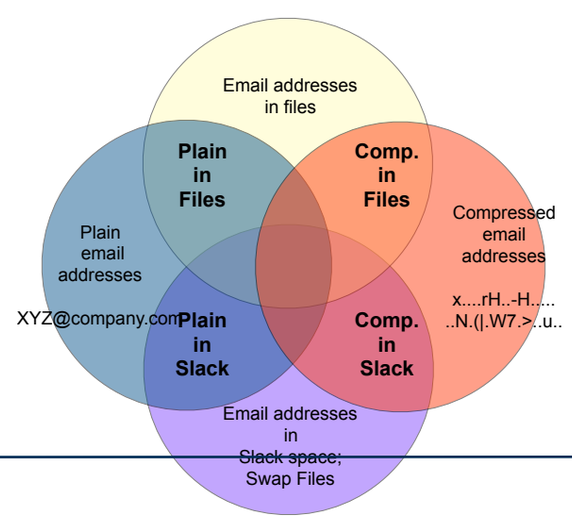
3772517888-GZIP-28322

Post-processing with identify_files.py reveals file names




```
Offset:  392175418
Feature: WindowsXP@gn.microsoft.com
Context: \012[User]\015\012Name=WindowsXP@gn.microsoft.com
         \015\012Password=B@ji0
Filename:WINDOWS/system32/oobe/migx25a.dun
MD5:     2b00042f7481c7b056c4b410d28f33cf
```

For each feature, we can determine if category #1, #2, #3 and #4!



bulk_extractor 1.4 recognizes a wide variety of features and encoding types:

Feature types:

- Domain Names; Email addresses; URLs, CCNs
- Search terms; Facebook IDs; JSON data
- KML files; EXIF data
- VCARDs
- word search output
- PCAP files; Ethernet Addresses; TCP/IP Connections; etc.
- ELF & PE headers; Windows Prefetch files

```
-rw-r--r--@ 1 simsong  staff       476 Jul  7 23:50 aes_keys.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  7 23:48 alerts.txt
-rw-r--r--@ 1 simsong  staff      2743 Jul  7 23:59 ccn.txt
-rw-r--r--@ 1 simsong  staff       454 Jul  8 00:03 ccn_histogram.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  7 23:48 ccn_track2.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  8 00:03 ccn_track2_histogram.txt
-rw-r--r--@ 1 simsong  staff  23369167 Jul  8 00:03 domain.txt
-rw-r--r--@ 1 simsong  staff    185266 Jul  8 00:03 domain_histogram.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  7 23:48 elf.txt
-rw-r--r--@ 1 simsong  staff   1719842 Jul  8 00:03 email.txt
-rw-r--r--@ 1 simsong  staff     35073 Jul  8 00:03 email_histogram.txt
-rw-r--r--@ 1 simsong  staff     23961 Jul  8 00:00 ether.txt
-rw-r--r--@ 1 simsong  staff       337 Jul  8 00:03 ether_histogram.txt
-rw-r--r--@ 1 simsong  staff  11188830 Jul  8 00:03 exif.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  7 23:48 find.txt
-rw-r--r--@ 1 simsong  staff      1112 Jul  8 00:01 gps.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  7 23:48 hex.txt
-rw-r--r--@ 1 simsong  staff     95835 Jul  8 00:03 ip.txt
-rw-r--r--@ 1 simsong  staff     11603 Jul  8 00:03 ip_histogram.txt
-rw-r--r--@ 1 simsong  staff   2025702 Jul  8 00:03 json.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  7 23:48 kml.txt
-rw-r--r--@ 1 simsong  staff    194991 Jul  8 00:03 packets.pcap
-rw-r--r--@ 1 simsong  staff     21343 Jul  8 00:03 report.xml
-rw-r--r--@ 1 simsong  staff   3782598 Jul  8 00:03 rfc822.txt
-rw-r--r--@ 1 simsong  staff    213746 Jul  8 00:03 tcp.txt
-rw-r--r--@ 1 simsong  staff     61255 Jul  8 00:03 tcp_histogram.txt
-rw-r--r--@ 1 simsong  staff     59469 Jul  8 00:03 telephone.txt
-rw-r--r--@ 1 simsong  staff      6612 Jul  8 00:03 telephone_histogram.txt
-rw-r--r--@ 1 simsong  staff  67205326 Jul  8 00:03 url.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  8 00:03 url_facebook-id.txt
-rw-r--r--@ 1 simsong  staff   5706665 Jul  8 00:03 url_histogram.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  8 00:03 url_microsoft-live.txt
-rw-r--r--@ 1 simsong  staff      8504 Jul  8 00:03 url_searches.txt
-rw-r--r--@ 1 simsong  staff    151673 Jul  8 00:03 url_services.txt
-rw-r--r--@ 1 simsong  staff         0 Jul  7 23:48 vcard.txt
-rw-r--r--@ 1 simsong  staff  18549729 Jul  8 00:03 windirs.txt
-rw-r--r--@ 1 simsong  staff  29051041 Jul  8 00:03 winpe.txt
-rw-r--r--@ 1 simsong  staff   1984759 Jul  8 00:03 winprefetch.txt
-rw-r--r--@ 1 simsong  staff  34128889 Jul  8 00:03 zip.txt
```

Encoding types:

- ZIP; GZIP; RAR; Windows Hibernation
- BASE16, BASE64

Some drives have a lot of compressed data

This drive contains a GZIP stream in a Windows Hibernation File.

```
...
…6464-HIBER-49691-GZIP-1526   groups-noreply@linkedin.com    3d\134"groups-noreply@linkedin.com
…6464-HIBER-49691-GZIP-2018   m*****************@gmail.com   3d\134"m*****************@gmail.co
…6464-HIBER-49691-GZIP-2128   sur*******1@gmail.com          3d\134"sur*******1@gmail.com\134"\
…6464-HIBER-49691-GZIP-2625   *******.consultancy@gmail.com 3d\134"*******.consultancy@gmail.c
…6464-HIBER-49691-GZIP-2736   sur*******1@gmail.com          3d\134"sur*******1@gmail.com\134"\
…6464-HIBER-49691-GZIP-3186   san****@***********.com        \134" "san****@***********.com\134"\134u
…6464-HIBER-49691-GZIP-3685   Careers@******bank.com         3d\134"Careers@******bank.com\134"
…6464-HIBER-49691-GZIP-4124   par****@team******.com         3d\134"par****@team******.com\134"
…6464-HIBER-49691-GZIP-4149   u003epar****@team******.com    \134u003epar****@team******.com\13
…6464-HIBER-49691-GZIP-4607   d****.*****@gmail.com          3d\134"d****.*****@gmail.com\134"\
…6464-HIBER-49691-GZIP-4631   u003ed****.*****@gmail.com     \134u003ed****.*****@gmail.com\134
…6464-HIBER-49691-GZIP-5114   raj******@bsnl.in              3d\134"raj******@bsnl.in\134"\134u
…6464-HIBER-49691-GZIP-5558   kiran.***@****technology.com   3d\134"kiran.***@****technology.co
…6464-HIBER-49691-GZIP-5671   sur*******1@gmail.com          3d\134"sur*******1@gmail.com\134"\
...
```

- JSON object downloaded from Facebook by compressed HTTP
- In RAM, written to HIBER on disk when the system went into sleep.

We ran bulk_extractor and identify_filenames.py on drive IN10-0138 and examined the email encodings:

Emails seen	count	1) Plain in Files	2) Comp. in Files	3) Plain in non-files	4) Comp in non-files
Cleartext		358	--	5341	--
All Comp		--	9	--	135
GZIP	50		14		36
HIBER	39		7		32
HIBER-GZIP	23				23
PDF	88		1		87
ZIP	28		7		21
ZIP-PDF	18				18

135 out of 5700 email addresses are invisible to existing tools.

Many of these email addresses are significant

Example email addresses (sanitized)

Encoding	Email Address (*Sanitized)	Note
GZIP	****@*****.dk	PII
ZIP	******@desktopsidebar.com	PII
HIBER	ntIV@std.do	false positive
ZIP	****************@digital.com	source code?
ZIP	pcg@goof.com	ECGS Compiler
ZIP	andrew@northwindtraders.com	MS Office Sample
ZIP	ActiveSh@eet.Na	false positive
GZIP	linux-ntfs-dev@lists.sourceforge.net	mailing list

Questions:

- How common are compressed email addresses in unallocated space?
- Is this technique worth the effort?

We do science with “real data.”

The Real Data Corpus (60TB)

- Disks, camera cards, & cell phones purchased on the secondary market.
- Most contain data from previous users.
- Mostly acquire outside the US:
  —*Canada, China, England, Germany, France, India, Israel, Japan, Pakistan, Palestine, etc.*
- Thousands of devices (HDs, CDs, DVDs, flash, etc.)

Mobile Phone Application Corpus

- Android Applications; Mobile Malware; etc.

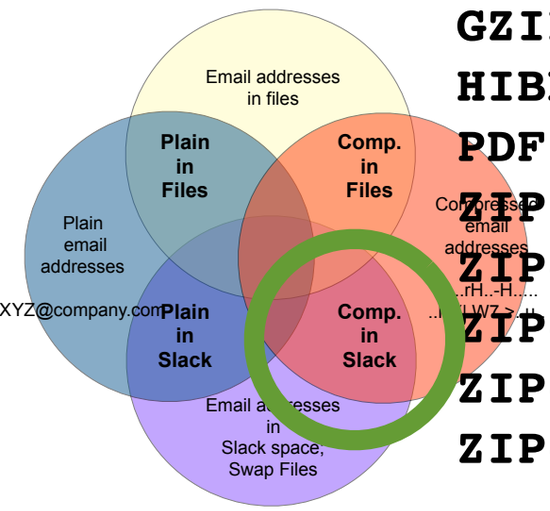
The problems we encounter obtaining, curating and exploiting this data mirror those of national organizations

—*http://digitalcorpora.org/*

Garfinkel, Farrell, Roussev and Dinolt, “Bringing Science to Digital Forensics with Standardized Forensic Corpora”, DFRWS 2009. BEST PAPER AWARD.

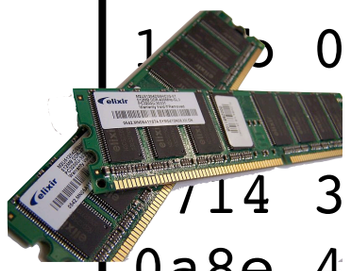
We analysis 1,646 disk images that had intact file systems. Many email addresses existed only encoded, in non-files.

Coding	Drives	Emails	avg	max	σ
1) Plain in files	739	81,920	110	4,206	253
2) Comp in files	355	19,711	55	5,454	388
3) Plain in non-files	860	1,956,059	2,274	178,073	9,248
4) Comp in non-files	474	165,481	349	59,376	2,889
BASE64 Comp	54	219	4	50	7
BASE64-GZIP Comp	2	64	32	37	5
GZIP Comp	234	66,195	282	9,103	981
GZIP-BASE64 Comp	7	44	6	11	3
GZIP-GZIP Comp	15	12,663	844	11,845	2,944
GZIP-GZIP-BASE64 Comp	2	38	19	30	11
GZIP-GZIP-GZIP Comp	4	58	14	38	14
GZIP-GZIP-ZIP Comp	1	12	12	12	0
GZIP-PDF Comp	5	38	7	30	11
GZIP-ZIP Comp	6	49	8	30	9
HIBER Comp	79	1,433	18	217	44
PDF Comp	162	2,352	14	238	31
ZIP Comp	388	85,252	219	59,369	3,025
ZIP-BASE64 Comp	5	30	6	13	5
ZIP-BASE64-GZIP Comp	2	65	32	38	5
ZIP-GZIP Comp	14	261	18	132	34
ZIP-PDF Comp	26	115	4	18	4



Some drives had more than 10,000 compressed email addrs.

Remember — compressed email addresses in non-files are ignored by today's forensic tools.



```
e327 962d 6450 3d91 c945 3bed 97a6 ..cd   .'.-dP=..E;.....
1... 0800 0000 0000 0... ........ ....   ............rH.
.... 8cc abd4 03d2 0... ........ ....   .-H.......N.(|.W
7714 3e00 b455 c1c5 3... ........ ....   7.>..U..0.......
0a8e 4ece 287c 1757 3714 3e00 a175 ..ed   ..N.(|.W7.>..u..
```

XYZ@company.com
ABC@company.com
DEF@company.com

Folders.pst

Mother.JPG

Presentation.pptx

Sequestration.docx

```
a097 83a1 ed96 26a6 3c69 3d0f 750a 2399   ......&.<i=.u.#.
a2b5 bea7 692f 5847 a38a dd53 082c add5   ....i/XG...S.,..
5061 b64c 721d 864b 90b6 b55f bb04 735c   Pa.Lr..K..._..s\
9448 6730 5453 df64 813e b603 5795 2242   .Hg0TS.d.>..W."B
e... 7454 7322 7cdc b60e 97af 2f64 2728   ..tTs"|...../d'(
.... 94bd 2a84 2dfe 50ea 5935 c349 1513   <XYZ@COMPANY.COM
.... e92c a3f8 6e46 0530 8a88 c7a2 5d2b   ...,..nF.0....]+
d89d 77cc fe1e f637 f3f3 d0af 1b47 c09b   ..w....7.....G..
```

(Compressed email in files are also ignored...)

"Digital media triage with bulk data analysis and bulk_extractor,"
Simson L. Garfinkel, *Computers and Security 32 (2013) 56-72*

email address	Application (encoding)	*strings* & *grep*	EnCase	BE
plain_text@textedit.com	Apple TextEdit (UTF-8)	✓	✓	✓
plain_text_pdf@textedit.com	Apple TextEdit print-to-PDF (/FlateDecode)			✓
rtf_text@textedit.com	Apple TextEdit (RTF)	✓	✓	✓
rtf_text_pdf@textedit.com	Apple TextEdit print-to-PDF (/FlateDecode)			✓
plain_utf16@textedit.com	Apple TextEdit (UTF-16)		✓	✓
plain_utf16_pdf@textedit.com	Apple TextEdit print-to-PDF (/FlateDecode)			✓
pages@iwork09.com	Apple Pages '09	✓	✓	✓
pages_comment@iwork09.com	Apple Pages (comment) '09			✓
keynote@iwork09.com	Apple Keynote '09			✓
keynote_comment@iwork09.com	Apple Keynote '09 (comment)			✓
numbers@iwork09.com	Apple Numbers '09			✓
numbers_comment@iwork09.com	Apple Numbers '09 (comment)			✓
user_doc@microsoftword.com	Microsoft Word 2008 (Mac) (.doc file)	✓	✓	✓
user_doc_pdf@microsoftword.com	Microsoft Word 2008 (Mac) print-to-PDF			
user_docx@microsoftword.com	Microsoft Word 2008 (Mac) (.docx file)			✓
user_docx_pdf@microsoftword.com	Microsoft Word 2008 (Mac) print-to-PDF (.docx file)			
xls_cell@microsoft_excel.com	Microsoft Word 2008 (Mac)	✓	✓	✓
xls_comment@microsoft_excel.com	Microsoft Word 2008 (Mac)			✓
xlsx_cell@microsoft_excel.com	Microsoft Word 2008 (Mac)			✓
xlsx_cell_comment@microsoft_excel.com	Microsoft Word 2008 (Mac) (Comment)			✓
doc_within_doc@document.com	Microsoft Word 2007 (OLE .doc file within .doc)	✓	✓	✓
docx_within_docx@document.com	Microsoft Word 2007 (OLE .doc file within .doc)	✓	✓	✓
ppt_within_doc@document.com	Microsoft PowerPoint and Word 2007 (OLE .ppt file within .doc)	✓	✓	✓
pptx_within_docx@document.com	Microsoft PowerPoint and Word 2007 (OLE .pptx file within .docx)			✓
xls_within_doc@document.com	Microsoft Excel and Word 2007 (OLE .xls file within .doc)	✓	✓	✓
xlsx_within_docx@document.com	Microsoft Excel and Word 2007 (OLE .xlsx file within .docx)			✓
email_in_zip@zipfile1.com	text file within ZIP			✓
email_in_zip_zip@zipfile2.com	ZIP'ed text file, ZIP'ed			✓
email_in_gzip@gzipfile.com	text file within gzip			✓
email_in_gzip_gzip@gzipfile.com	gzip'ed text file, gzip'ed			✓

21 out of 30 compressed email addresses in test files were ignored.

There are many sources of compressed and encoded data. Today's tools ignore these data when not in files.

Documents:

- Microsoft Office (.docx, .xlsx, .pptx); PDF files (text is compressed)
- Browser Cache (downloads are compressed)

Archives:

- ZIP files; GZIP (GZ) files

System Resources:

- Hibernation files & file fragments

If forensic examiners miss an email address:

- A perpetrator or an accomplice may not be identified
- Media may not be associated with a crime

XOR

Summer 2013 Research Project

Each summer for the past three years, NPS NCR has hosted interns to research digital forensics.

Previous projects:

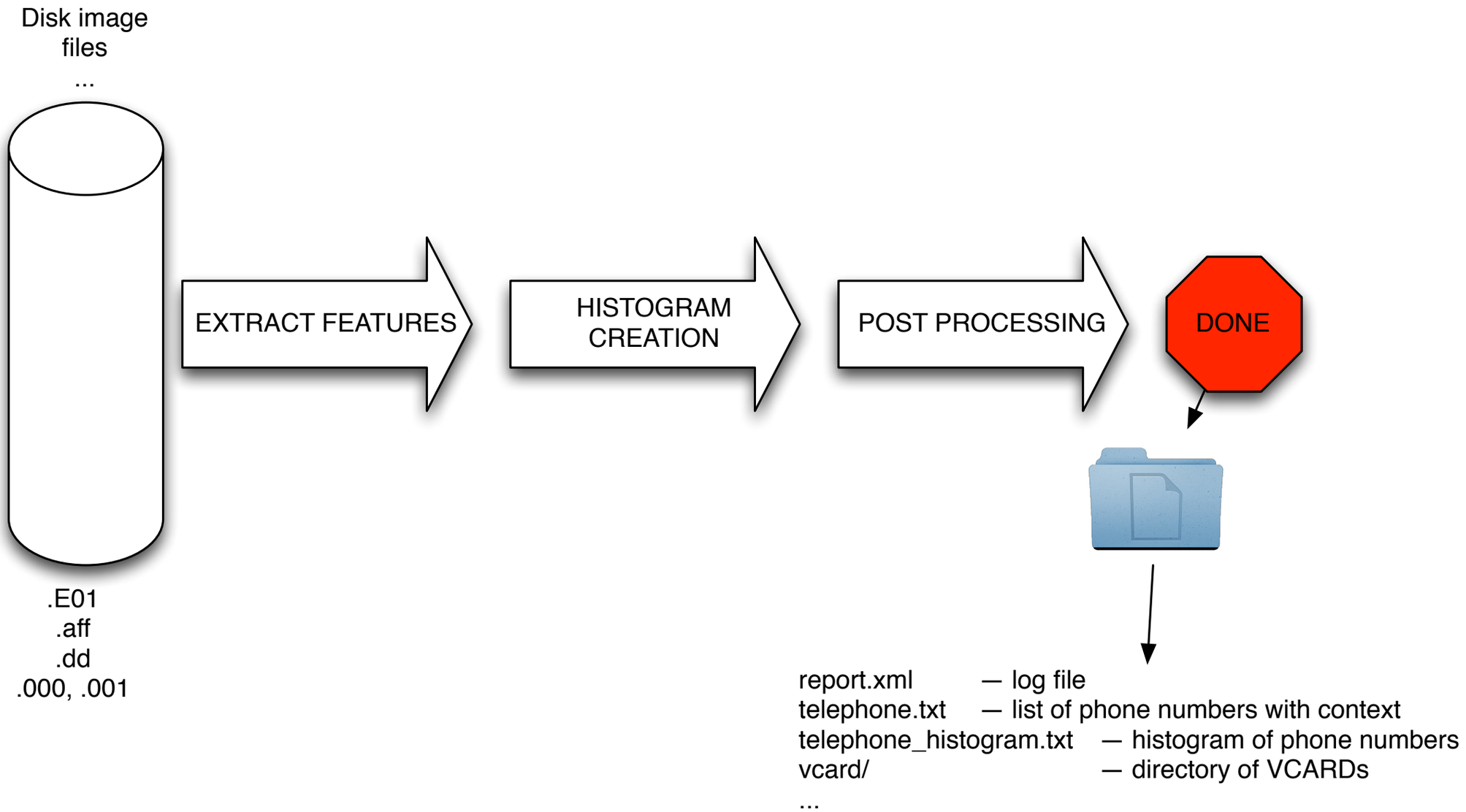
- Summer 2011 — bulk_extractor enhancements
- Summer 2012 — National Gallery DC Scenario

Summer 2013 — XOR usage in the Real Data Corpus

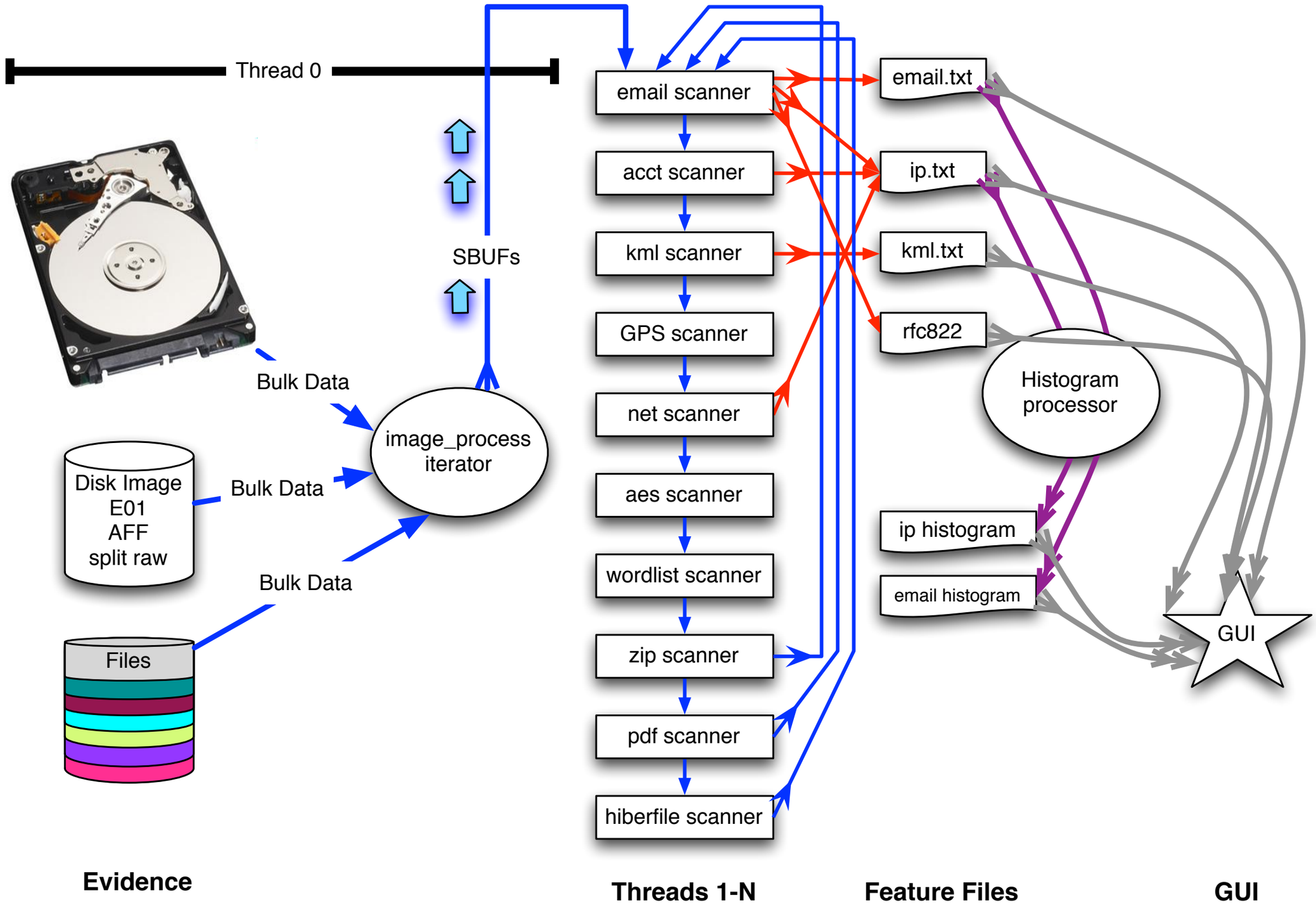
- CDT Aubin Heffernan, USMA
- CDT Scott Horras, USMA
- CDT Kyle Gorak, USMA
- Ms. Carolina Zarate, Poolesville High School

The students analyzed bulk_extractor output.

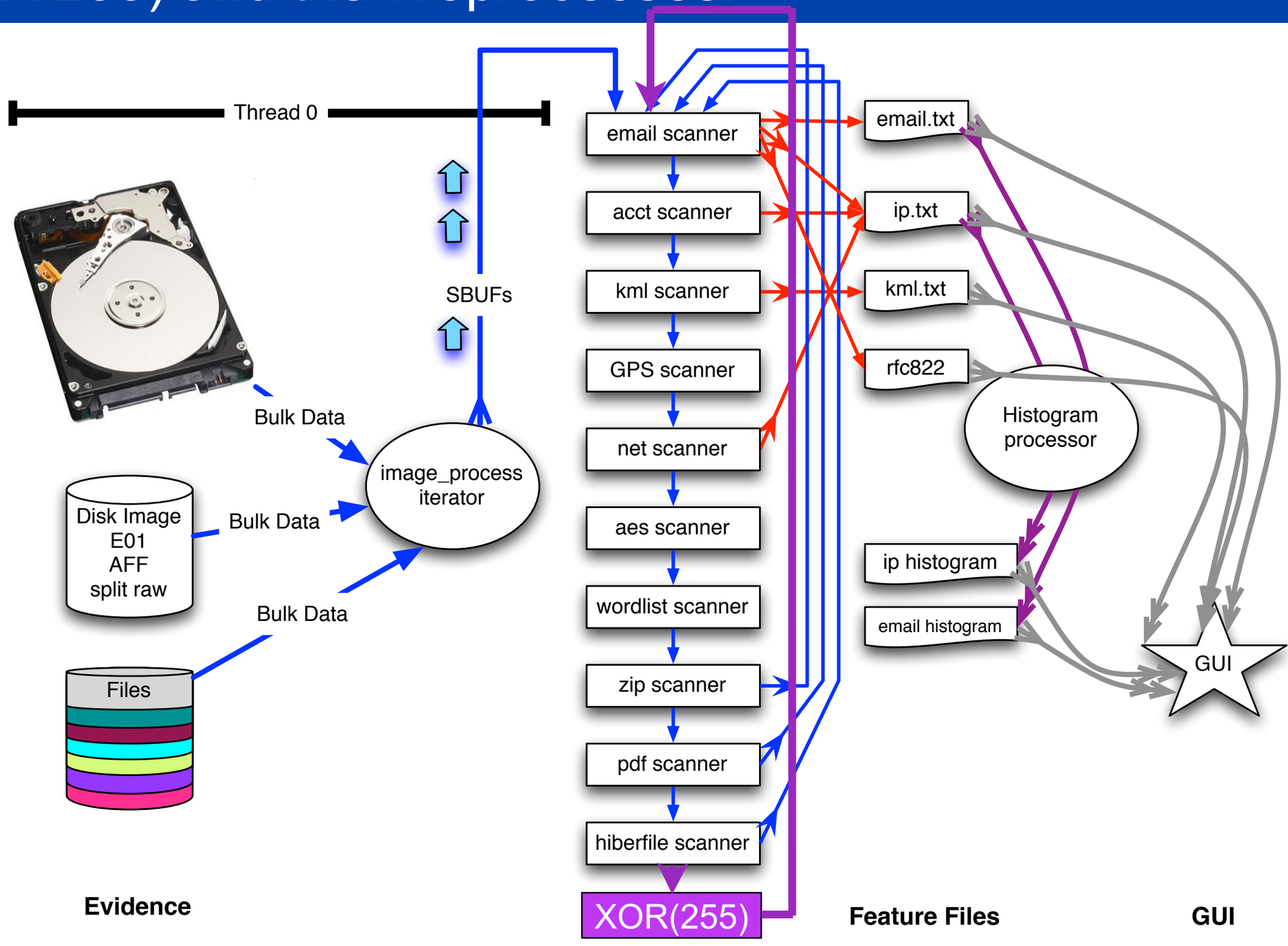
Recall that bulk_extractor processes data and outputs feature files:



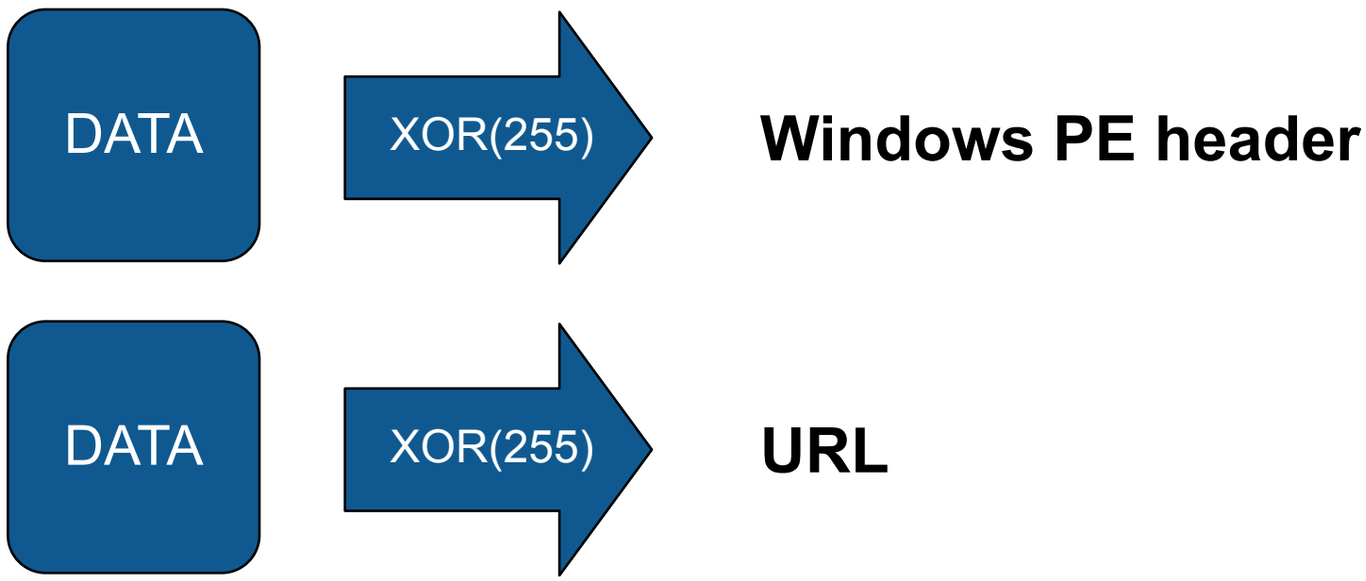
bulk_extractor's internal design:



We created another “scanner” that inverts the SBuf (XOR 255) and then reprocesses.

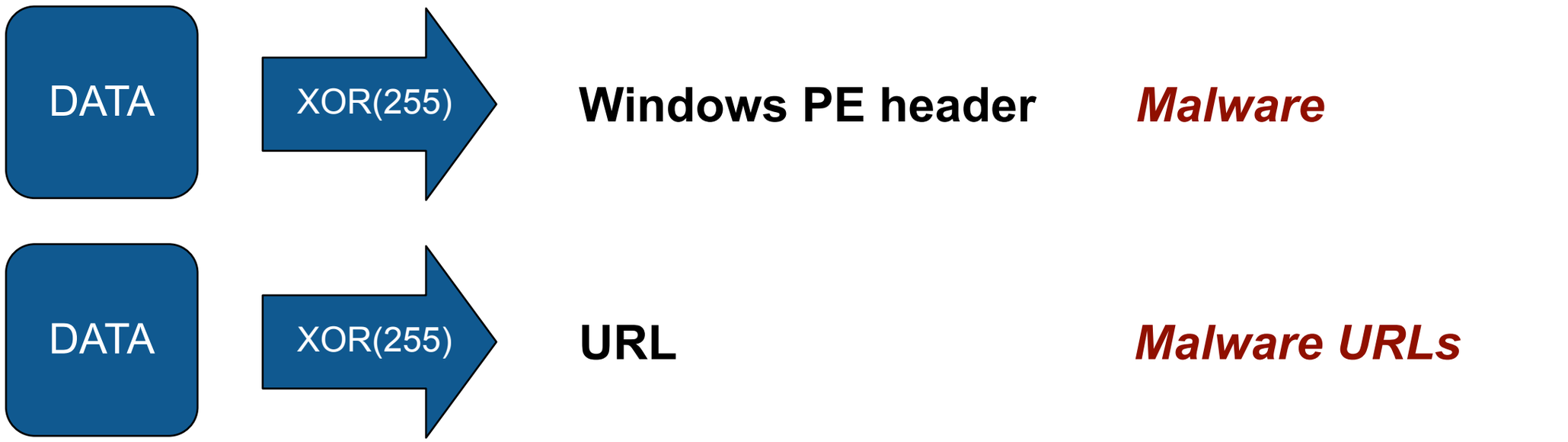


We searched for valid data that had been XOR'ed.



These kinds of data can be recognized with high reliability.

We found substantial XOR'ed data.



These kinds of features can be recognized with high reliability.

—*We XOR encoded features by running "grep XOR" over the feature files.*

We found legitimate and illicit use of XOR(255) to hide data.

We examined anti-virus systems and found:

- Malware used XOR(255) to hide download URLs
- AV XOR'ed Malware that was put into QUARANTINE
- VirusTotal did not recognize uploaded malware that had been XORed.

XOR(255) in commercial software:

- Real Audio — to obscure Dr. Yuriy Reznik's email address.
- Nero 7 — to hide a watermark (http://www.nero.com)

XOR(255) in Exfil'ed data:

- Fragments of a ZIP file that had been XOR'ed.
- Contents were Excel spreadsheets with names & salary data.

XOR(255) is throughout our corpus.

Year	# URL	# WinPE	# ZIP	#Exif
1980	4	7	0	0
1981	6	0	0	0
1985	15	0	0	0
1990	0	20	0	0
1996	2	11	0	0
1997	185	15	0	0
1998	443	126	3	0
1999	261	526	44	0
2000	252	526	12	0
2001	593	238	1	0
2002	734	234	1	0
2003	224	87	0	0
2004	1,359	427	34	0
2005	2,640	184	0	0
2006	1,934	3,840	6	0
2007	315	16,782	0	0
2008	1,376	1,973	0	0
2009	1,722	489	0	0
2010	802	468	0	0
2011	14,594	8	74	0
2013	11	1	0	0
2014	49	1	0	0
2016	10	1	0	0
2018	818	0	0	0
2019	3	0	0	0
2023	2	0	0	0
2027	346	0	0	0
2029	4	1	0	0
2030	4	2	0	0
2033	218	0	0	0
2037	14	0	0	0
2080	20	14	0	0
2081	10	2	0	0
no file	252,742	11,550	4,594	130
Total	281,712	37,533	4,769	130

Table 1: Validated XOR features by year for the analyzed drives, where the "year" is corresponds to the modification time of the file within which each XOR-encoded feature was found. "no file" indicates that the XOR-encoded features could not be located to a specific file. Timestamps prior to 1996 and after 2011 are likely the result of an improperly set system clock or on-disk corruption and are reported here for completeness.

XOR(255) was found in drives from (all) 21 countries .

country	total drives	drives with XOR WinPE	drives with XOR URL	drives with XOR ZIP	drives with XOR exif
BANGLADESH	57	15	5	0	0
BOSNIA AND HERZEGOVINA	7	0	0	0	0
CANADA	48	8	1	0	0
CHINA	807	25	1	0	0
EGYPT	7	2	2	0	0
GERMANY	37	22	6	1	0
GHANA	19	8	1	0	0
GREECE	10	2	0	0	0
INDIA	603	185	77	13	4
ISRAEL	260	84	39	9	0
MEXICO	173	73	16	3	1
MONACO	11	6	2	0	1
NEW ZEALAND	1	0	0	0	0
PAKISTAN	81	31	2	0	0
PALESTINE, STATE OF	140	39	8	3	0
SINGAPORE	34	4	1	0	0
SWITZERLAND	2	0	0	0	0
THAILAND	17	9	1	2	1
TURKEY	10	6	2	0	0
UNITED ARAB EMIRATES	87	62	7	19	0
Total	2,411	581	171	50	7

Table 2: Incidence of drives with Validated XOR features, by country

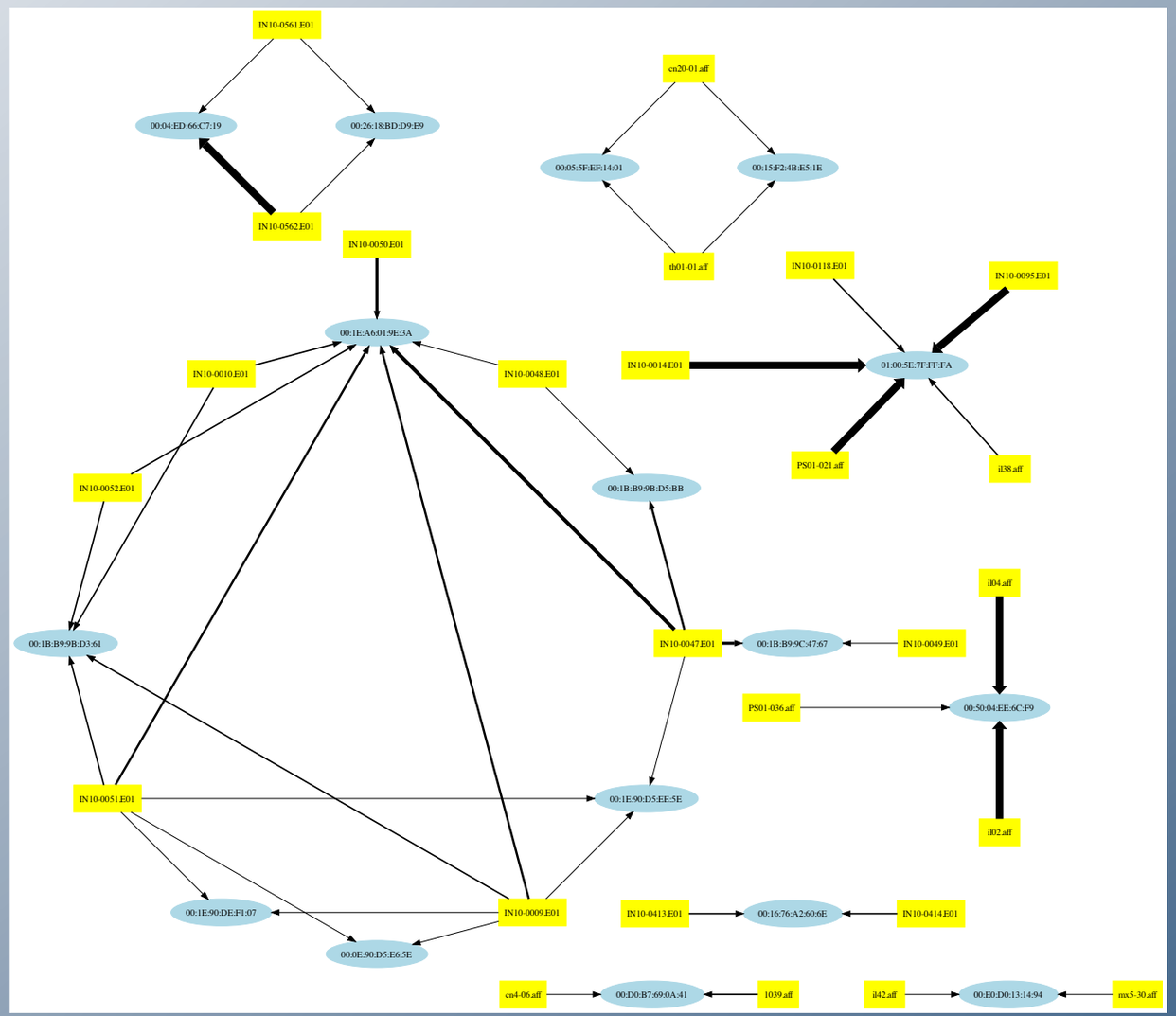
Unfortunately, our current XOR implementation significantly increases processing time.

test image	Size	without XOR	with XOR	Δ
nps-2009-domexusers	40GB	522 sec	799 sec	+53%
nps-2011-2tb	2TB	34,140 sec	58,147 sec	+70%

Table 4: Observed processing times for *bulk_extractor* with and without XOR scanner.

Solution 1 — Only use when "necessary."

Solution 2 — Examine data *before* XORing



Beverly, Robert, Simson Garfinkel and Greg Cardwell, "Forensic Carving of Network Packets and Associated Data Structures", DFRWS 2011, Aug. 1-3, 2011, New Orleans, LA. BEST PAPER AWARD

Packet carving with optimistic decompression

Are binary network data structures present on non-volatile media?

"binary network data structures" includes:

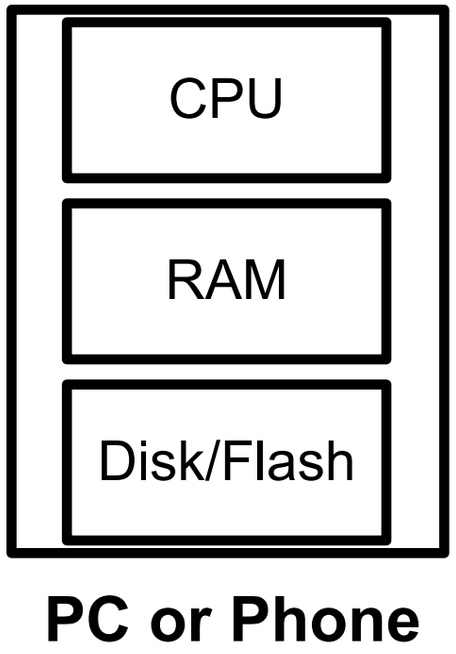
- Network packets
- Memory structures associated with network connections

We know that binary data are present in RAM.

Are they present on storage?

Mechanisms:

- Swap files
- Hibernation files
- Binary data structures stored in files.



Approach: use bulk_extractor to find the packets.

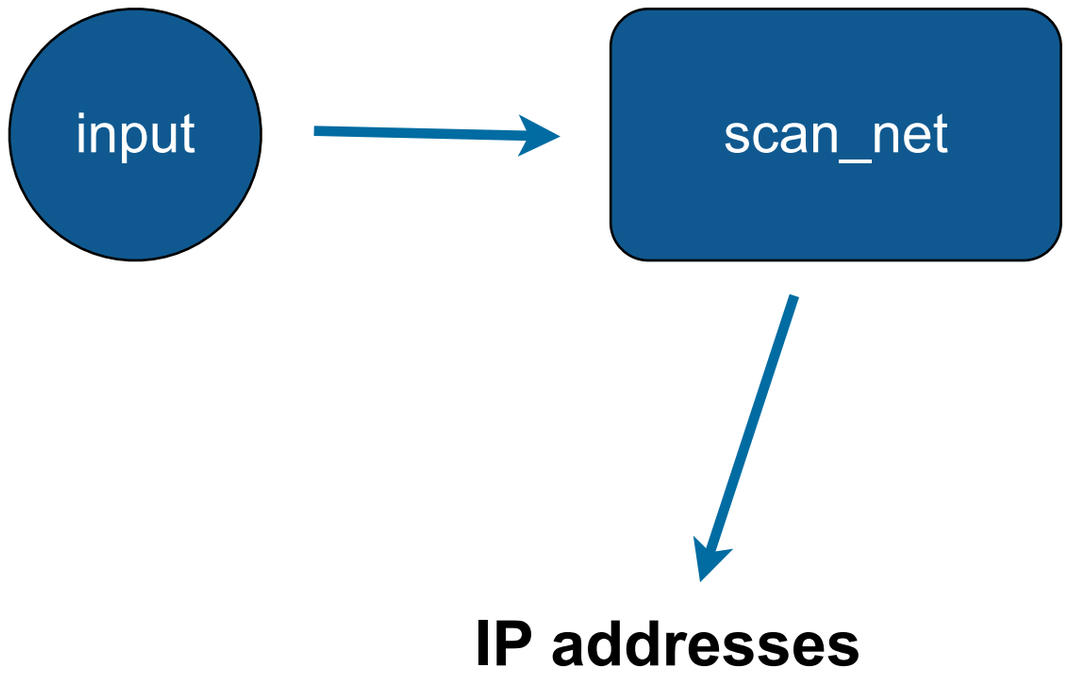
We created **scan_net**.

- Searches for
  - —*BPF "pcap" packet headers*
  - —*IPv4 & IPv6 packets*
  - —*Ethernet headers*
  - —*Windows Socket Structures*

```
struct ip {
  u_int   ip_v:4,                  /* version */
          ip_hl:4;                 /* header length */
  u_char  ip_tos;                  /* type of service */
  u_short ip_len;                  /* total length */
  u_short ip_id;                   /* identification */
  u_short ip_off;                  /* fragment offset field */
  u_char  ip_ttl;                  /* time to live */
  u_char  ip_p;                    /* protocol */
  u_short ip_sum;                  /* checksum */
  struct  in_addr ip_src,ip_dst;   /* source and dest address */
}
```

Ran with bulk_extractor

- Processed 1817 disk images
- Recorded found IP addresses in feature files.



Binary carving found information not in ASCII

Of the 1817 images:

- binary IP addresses found on 723 drives (≈40%)
- Average text IP addresses per drive: 2258
- Average binary IP addresses per drive: 21
- Some IP addresses not present in text (on 66 drives)

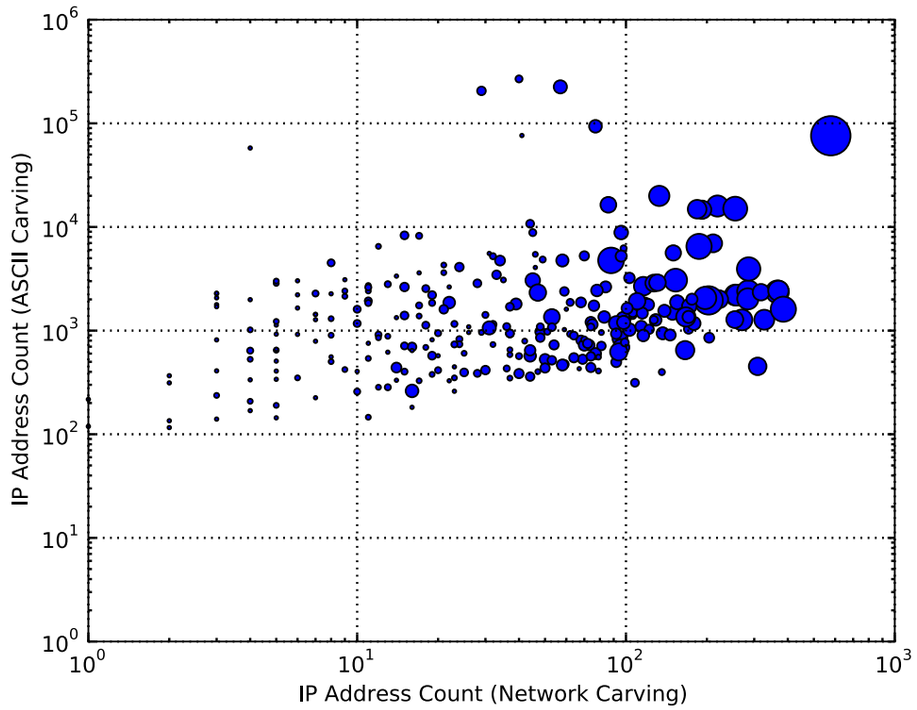


Fig. 4 — Correlation between scanning modalities across 723 images in the corpus. Each circle corresponds to a single hard drive, where the X axis indicates the number of addresses found through binary carving, the Y axis indicates the number of addresses found by ASCII carving, and the size of the circle indicates the number of addresses that are the same.

Hibernation decompression proved essential.

Why we focused on hibernation:

- Network data structures are in system memory
- Memory is stored in hibernation files
- Windows overwrites the beginning of hibernation files when resuming

  —*But not the whole file!*
- Fragments of hibernation files left in unallocated space when windows defragments
- We find an 8-byte XPress compression signature within the compressed memory page header and decompress the entire page.

Opportunistically decompress XPress pages

Address	Count	Decompressed Count
172.20.105.74	25	600
172.20.104.199	41	434
18.26.0.230	43	162
172.20.20.11	0	4
...	...	...

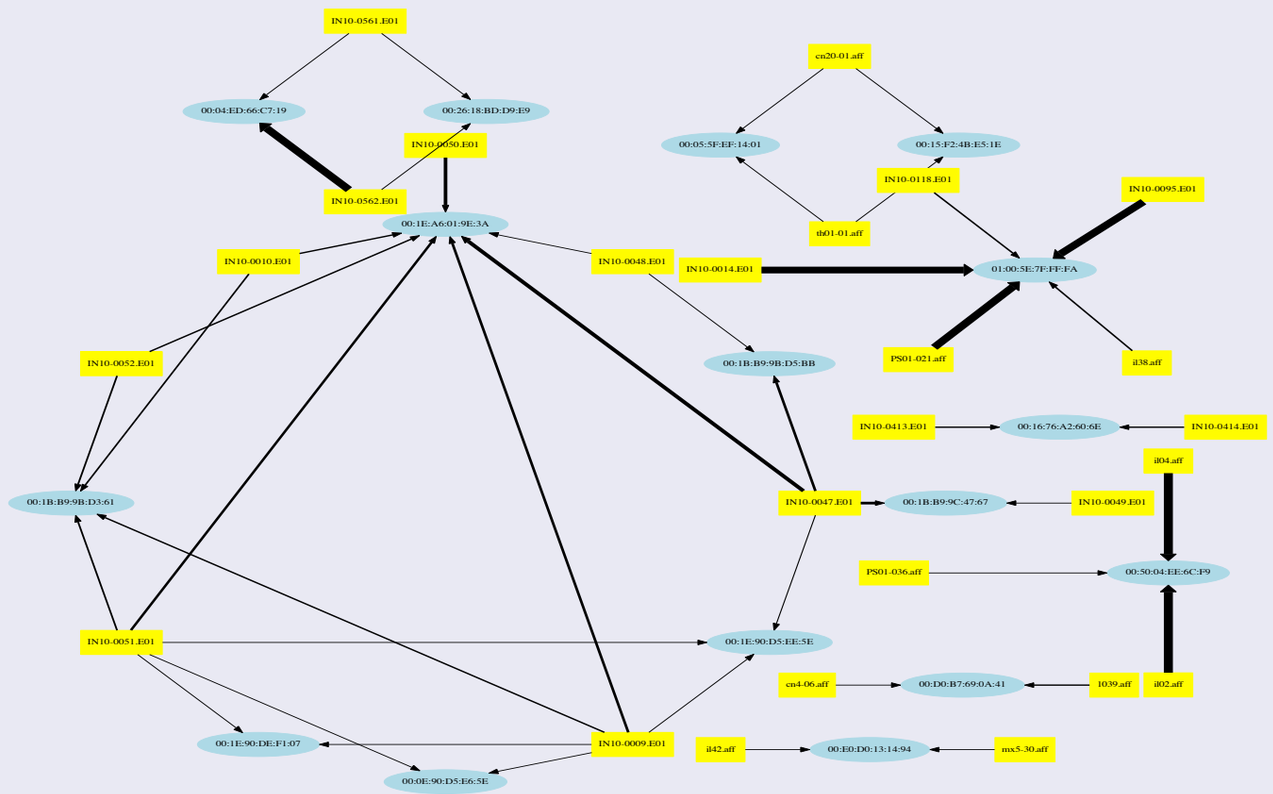
- Improves recall by an order of magnitude on our test image!

With cross-drive analysis, we could find drives from the same network!

Cross-Drive MAC Analysis

Cross-Drive MAC Analysis

- Many RDC images bought in batches
- We find 16 Ethernet common between images!
- Graph shows 8 distinct clusters:



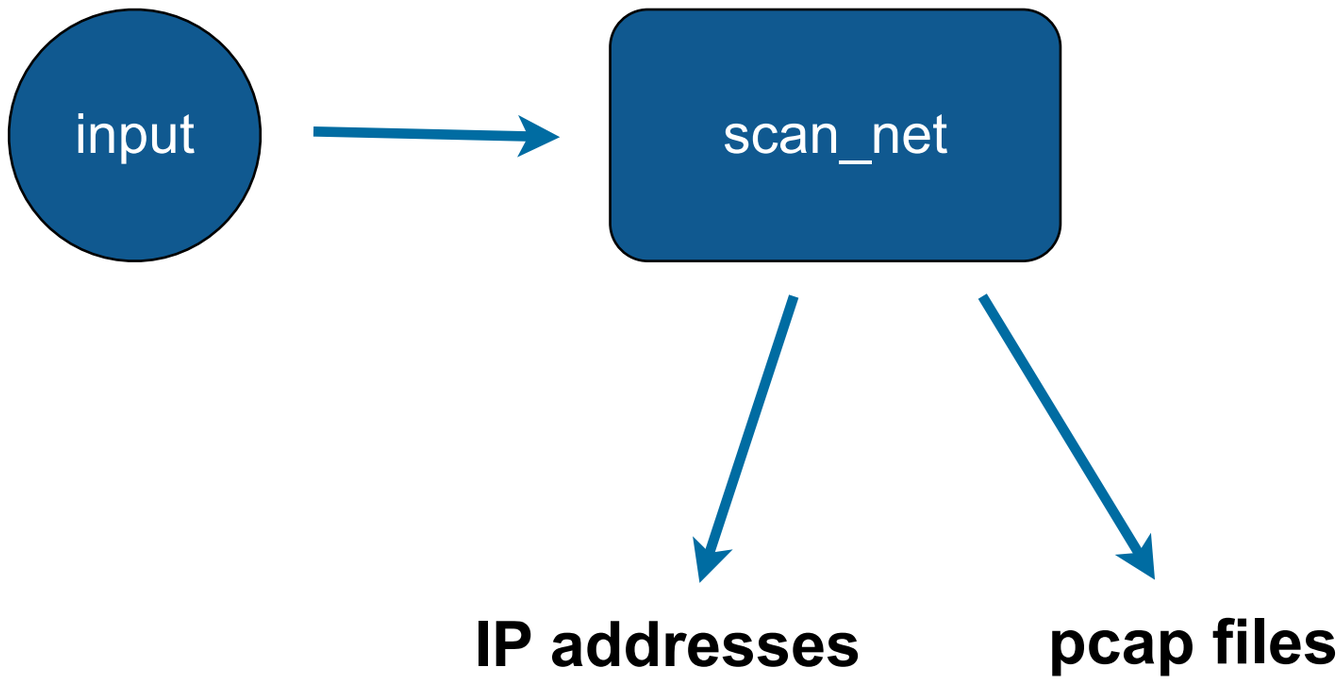
In 2012, scan_net was expanded to create pcap files

"pcap" files are much easier to process.

- Existing tools: Wireshark, tcpflow, tcpdump,...
- Include timestamp metadata for each packet.

Two sources of packets:

- Ethernet or IP packet in memory.
- Sniffer left packets on drive.



We find a lot of packets on disks!

August 2013 study of bulk_extractor on corpus.

- Total disks processed: 2418
- Disks with extractable packets: 710

Top 5 drives:

Drive	PCAP size
IN4001-1026	31MB
IL2-0086	10MB
BD1-1071	6MB
IL3-0212	5MB
TH0001_0010	3MB

Example of packets from IN4001-1026:

Packets in file	204,202
UDP packets	93,130
TCP packets	111,039

Sample UDP

```
-5:00:00.000000 IP 10.48.231.2.hsrp >
                all-routers.mcast.net.hsrp:
        HSRPv0-hello 20: state=active group=5 addr=10.48.231.1
```

Sample TCP:

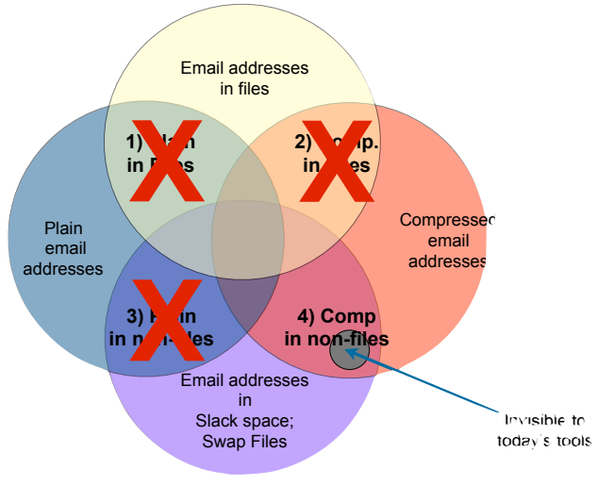
```
-5:00:00.000000 IP 10.48.133.228.http > 10.48.231.44.chip-lm:
Flags [.], seq 6301:7561, ack 763, win 65535, length 1260
```

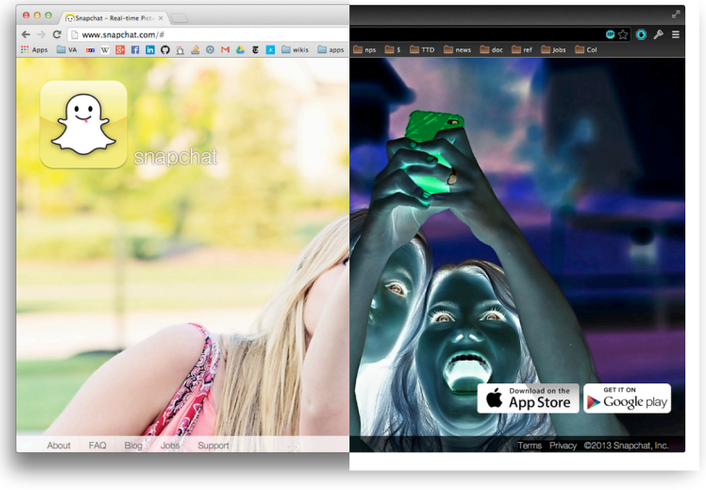
Note:

- No time set, so these came from memory, not a carved PCAP file.

In conclusion:
Optimistic decompression finds important data.

Important, relevant data is ignored by today's tools.





We demonstrated the extent of the problem with:

- bulk_extractor, a high-performance stream-based feature extractor
  - —*https://github.com/simsong/bulk_extractor (dev tree)*
  - —*http://digitalcorpora.org/downloads/bulk_extractor (downloads)*
  - —*http://www.sciencedirect.com/science/article/pii/S1742287612001472 (paper)*
  - —*http://simson.net/clips/academic/2013.COSE.bulk_extractor.pdf*
- Real Data Corpus:
  - —*http://digitalcorpora.org/*

We found:

- email addresses, malware, packets, and more.

Contact Information:
Simson L. Garfinkel
simsong@acm.org
http://simson.net/