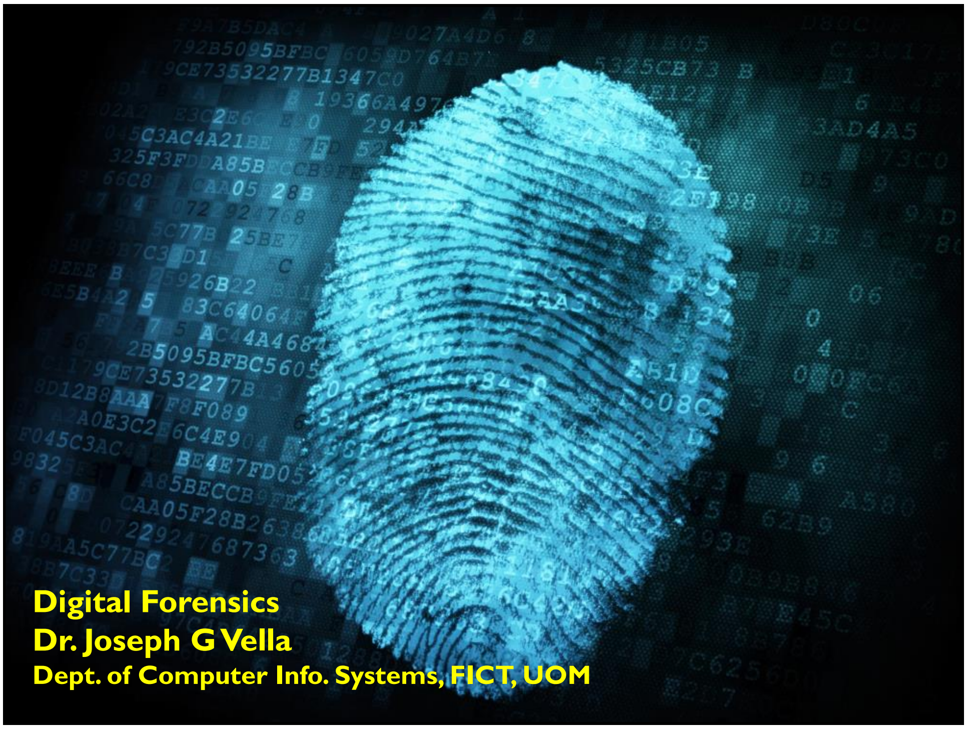
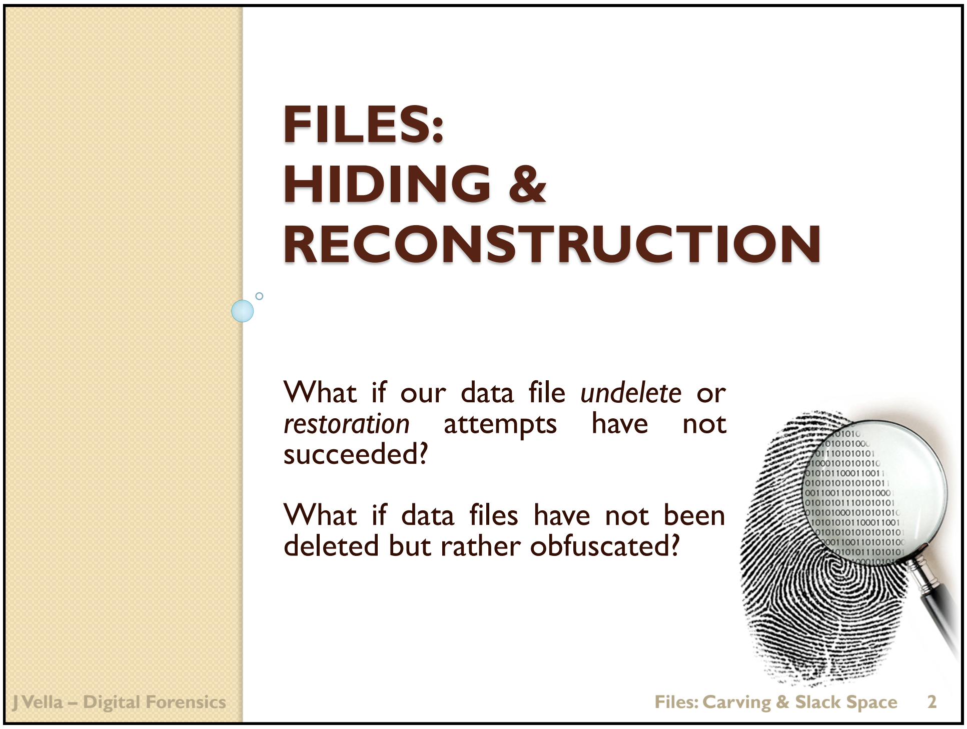


Digital Forensics

Dr. Joseph G Vella

Dept. of Computer Info. Systems, FICT, UOM



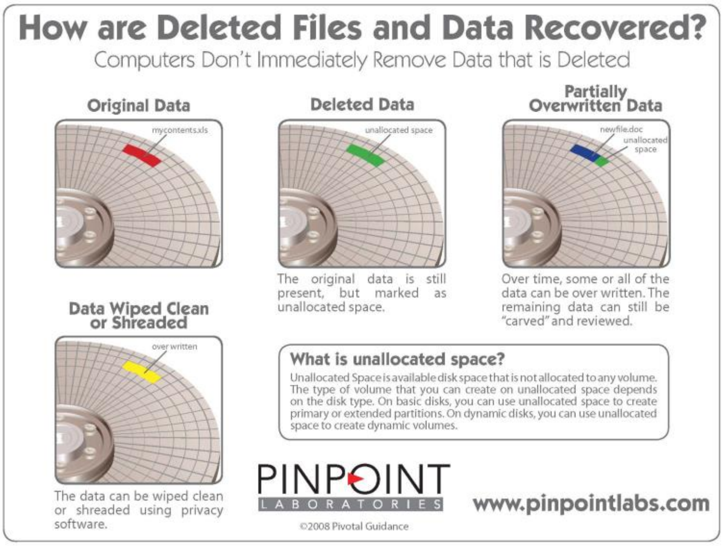
FILES: HIDING & RECONSTRUCTION

What if our data file *undelete* or *restoration* attempts have not succeeded?

What if data files have not been deleted but rather obfuscated?

Techniques:

File Carving
&
Slack Space Use



How are Deleted Files and Data Recovered?

Computers Don't Immediately Remove Data that is Deleted

Original Data

Deleted Data

The original data is still present, but marked as unallocated space.

Partially Overwritten Data

Over time, some or all of the data can be over written. The remaining data can still be "carved" and reviewed.

Data Wiped Clean or Shreaded

The data can be wiped clean or shreaded using privacy software.

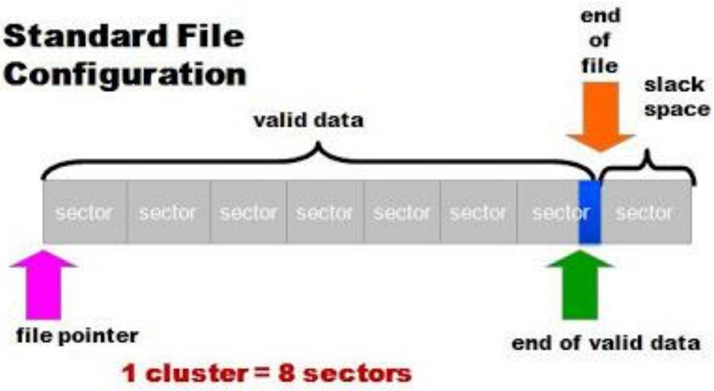
What is unallocated space?

Unallocated Space is available disk space that is not allocated to any volume. The type of volume that you can create on unallocated space depends on the disk type. On basic disks, you can use unallocated space to create primary or extended partitions. On dynamic disks, you can use unallocated space to create dynamic volumes.

PINPOINT LABORATORIES

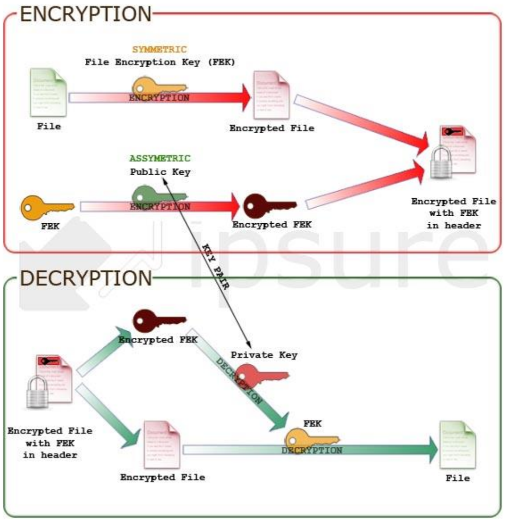
©2008 Pivotal Guidance

www.pinpointlabs.com



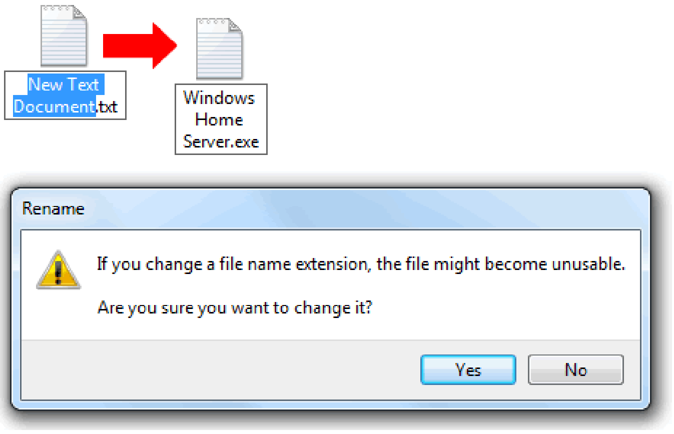
J Vella – Digital Forensics

Files: Carving & Slack Space 5



J Vella – Digital Forensics

Files: Carving & Slack Space 6

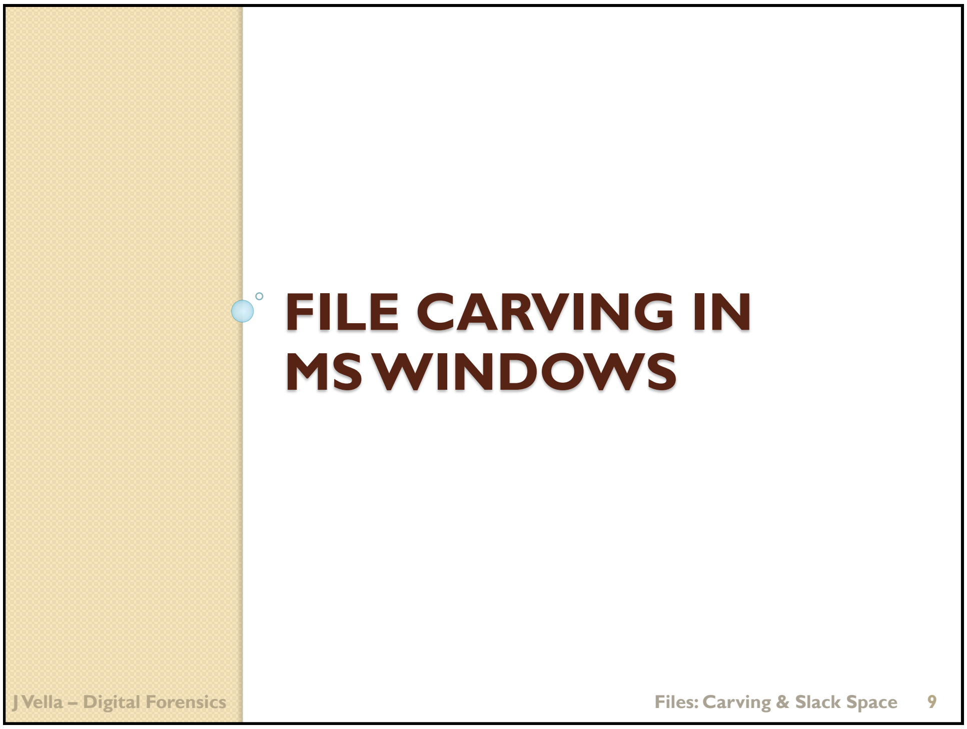


If you change a file name extension, the file might become unusable.

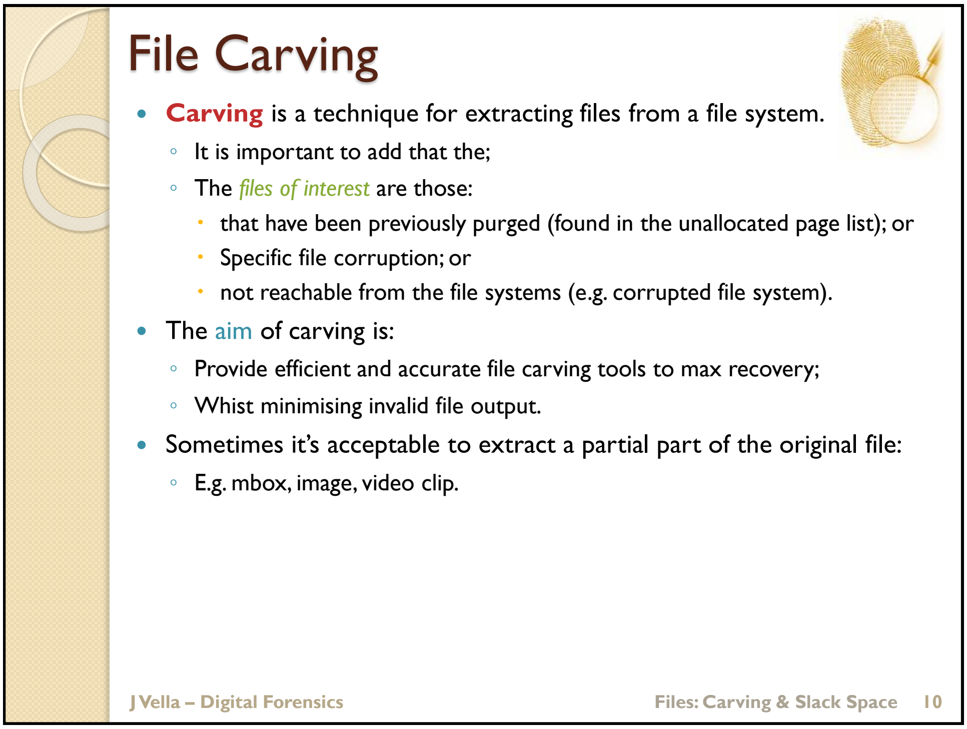
Are you sure you want to change it?

Also possible is a combination of

- Rename and compress;
- Slack space filled with compressed data;



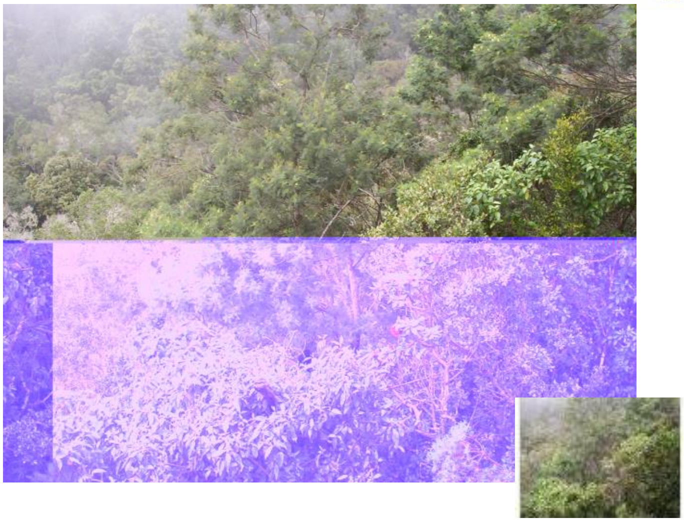
FILE CARVING IN MS WINDOWS



File Carving

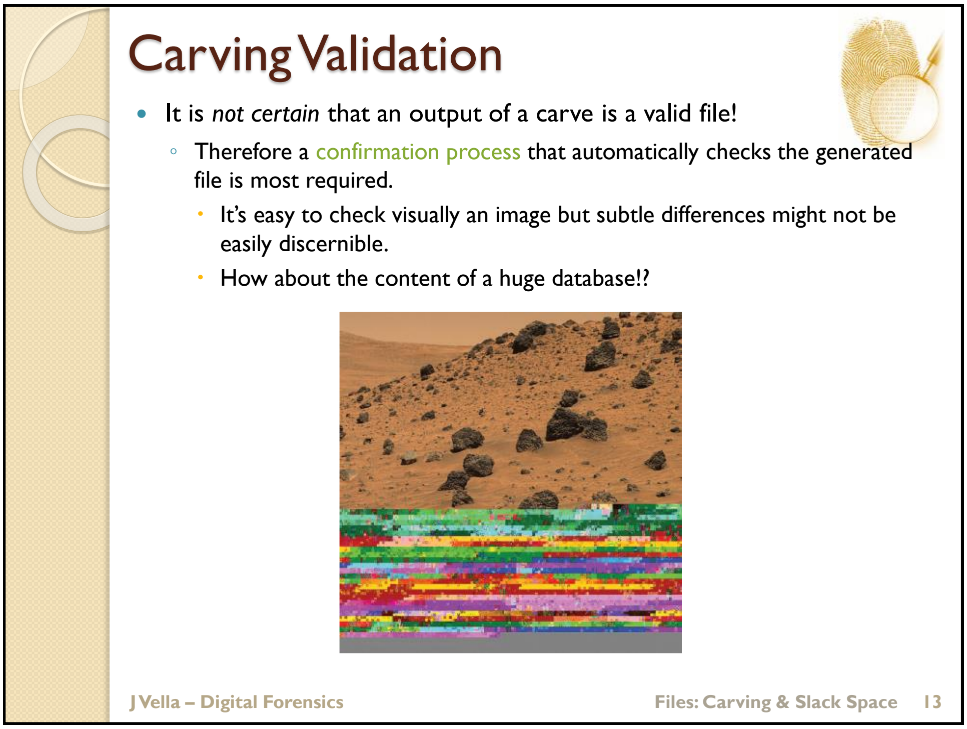
- **Carving** is a technique for extracting files from a file system.
  - It is important to add that the;
  - The *files of interest* are those:
    - that have been previously purged (found in the unallocated page list); or
    - Specific file corruption; or
    - not reachable from the file systems (e.g. corrupted file system).
- The aim of carving is:
  - Provide efficient and accurate file carving tools to max recovery;
  - Whist minimising invalid file output.
- Sometimes it's acceptable to extract a partial part of the original file:
  - E.g. mbox, image, video clip.

Example of corrupted file



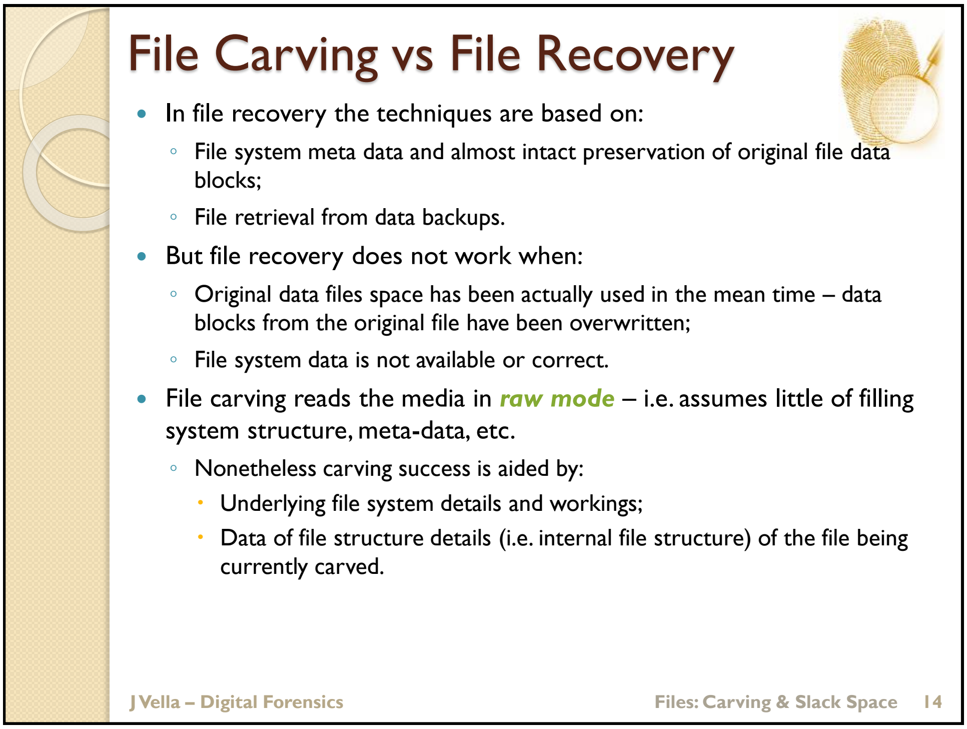
Carving Methods

- There are a number of **file carving methods**:
  - Basic ones are based on a brute force traversal of unallocated space and match and string up disk blocks to form the original files.
    - A basic technique is matching file type specific keys e.g. header and footers bit patterns.
      - Or header plus a file size (length).
  - File-structure based where intricate details of file structure and actual file are extracted and this structure leads the process.
    - Parts might neither be sequential or all present
    - These are sometime called ***deep carvers***.
  - Data content leads the selection process to string one block to a previous chain:
    - data content could be interspersed in MBOX, XML, etc
    - Statistical attributes of data;
    - Known language and subject matter.



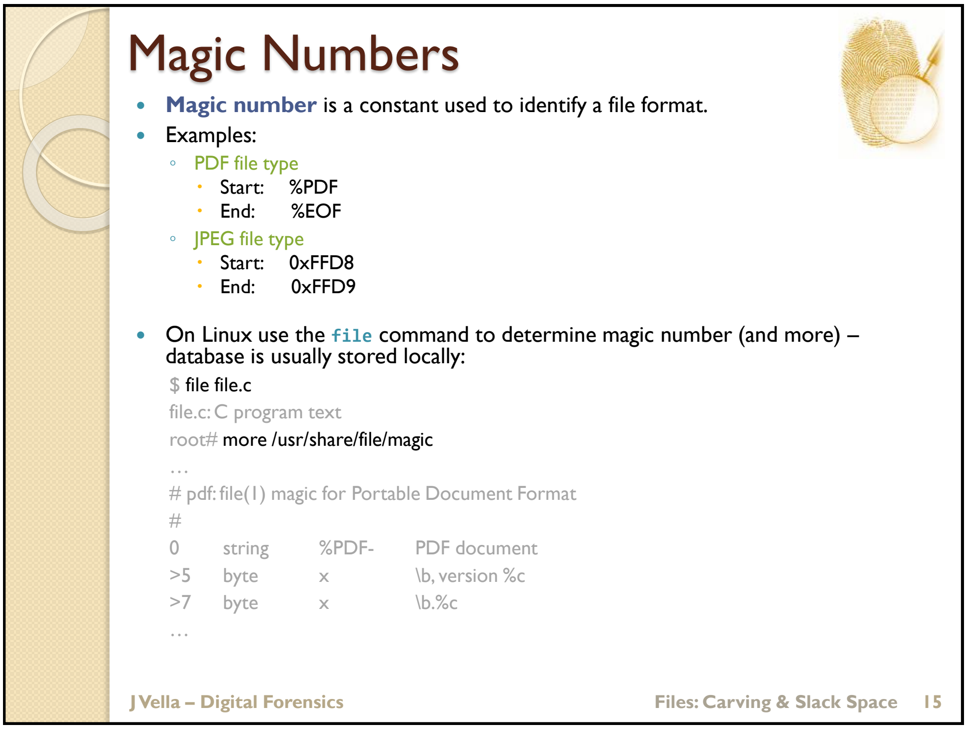
Carving Validation

- It is *not certain* that an output of a carve is a valid file!
  - Therefore a confirmation process that automatically checks the generated file is most required.
    - It's easy to check visually an image but subtle differences might not be easily discernible.
    - How about the content of a huge database!?



File Carving vs File Recovery

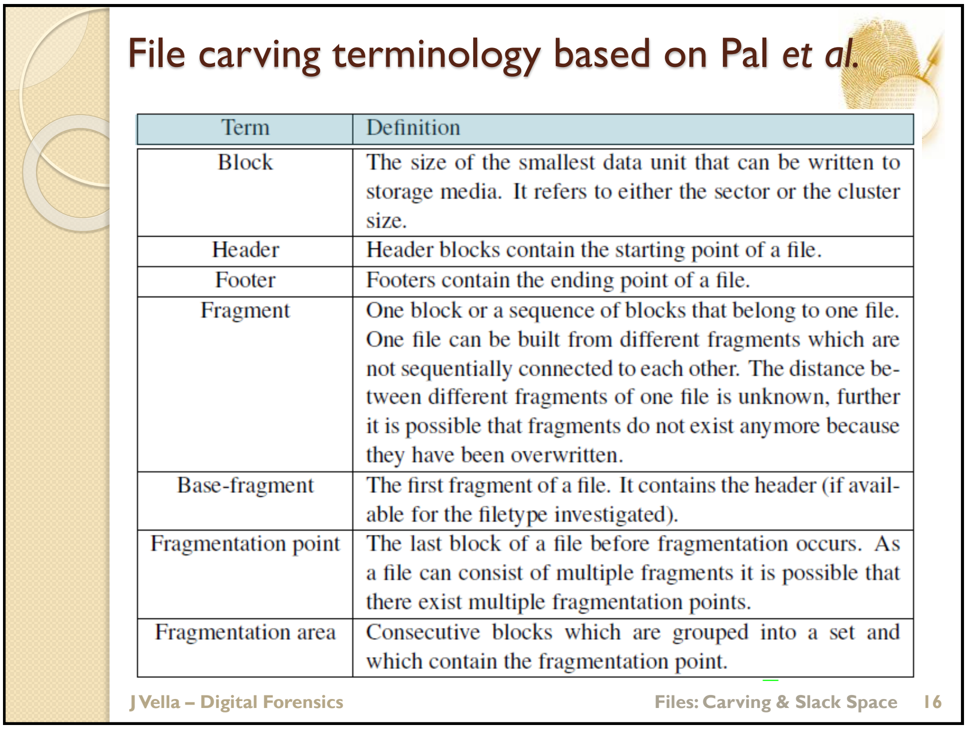
- In file recovery the techniques are based on:
  - File system meta data and almost intact preservation of original file data blocks;
  - File retrieval from data backups.
- But file recovery does not work when:
  - Original data files space has been actually used in the mean time – data blocks from the original file have been overwritten;
  - File system data is not available or correct.
- File carving reads the media in ***raw mode*** – i.e. assumes little of filling system structure, meta-data, etc.
  - Nonetheless carving success is aided by:
    - Underlying file system details and workings;
    - Data of file structure details (i.e. internal file structure) of the file being currently carved.



Magic Numbers

- **Magic number** is a constant used to identify a file format.
- Examples:
  - PDF file type
    - Start: %PDF
    - End: %EOF
  - JPEG file type
    - Start: 0xFFD8
    - End: 0xFFD9
- On Linux use the `file` command to determine magic number (and more) – database is usually stored locally:

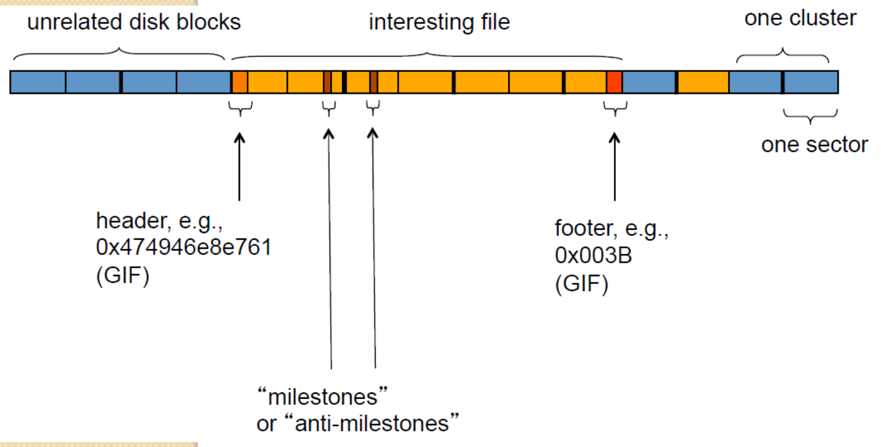
```
$ file file.c
file.c: C program text
root# more /usr/share/file/magic
…
# pdf: file(1) magic for Portable Document Format
#
0       string      %PDF-       PDF document
>5      byte        x           \b, version %c
>7      byte        x           \b.%c
…
```



File carving terminology based on Pal *et al.*

Term	Definition
Block	The size of the smallest data unit that can be written to storage media. It refers to either the sector or the cluster size.
Header	Header blocks contain the starting point of a file.
Footer	Footers contain the ending point of a file.
Fragment	One block or a sequence of blocks that belong to one file. One file can be built from different fragments which are not sequentially connected to each other. The distance between different fragments of one file is unknown, further it is possible that fragments do not exist anymore because they have been overwritten.
Base-fragment	The first fragment of a file. It contains the header (if available for the filetype investigated).
Fragmentation point	The last block of a file before fragmentation occurs. As a file can consist of multiple fragments it is possible that there exist multiple fragmentation points.
Fragmentation area	Consecutive blocks which are grouped into a set and which contain the fragmentation point.

HEADER-FOOTER CARVING



unrelated disk blocks

interesting file

one cluster

one sector

header, e.g.,
0x474946e8e761
(GIF)

footer, e.g.,
0x003B
(GIF)

“milestones”
or “anti-milestones”

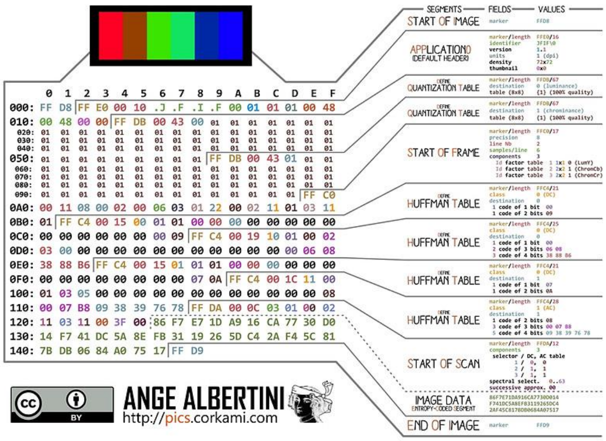
JPEG file internal structure

- Every file type has a header (i.e. hex sequence) – magic number.
  - This *header* is also used as start of image (SOI) – 0x FF D8.
  - The *footer*, called the end of image (EOI), is – 0x FF D9.
  - After the SOI is a sequence of file information blocks – field specifiers are hex 0x FF C0 to 0x FF DD.
    - Each specifier is qualified by two bytes to represent size.
    - After the size there is the data.
- The actual image starts at the signature 0x FF DA (SOS).
  - Some data files might have a *thumbnail* – again it's sandwiched between another SOI & EOI.

Short Name	Bytes	Payload	Name
SOI	0x FF D8	none	Start of Image
SOF0	0x FF C0	variable size	Start of Frame (Baseline DCT)
SOF2	0x FF C2	variable size	Start of Frame (Progressive DCT)
DHT	0x FF C4	variable size	Define Huffman Table(s)
DQT	0x FF DB	variable size	Define Quantization Table(s)
DRI	0x FF DD	2 bytes	Define Restart Interval
SOS	0x FF DA	variable size	Start of Stream
RST*n*	0x FF D0…0x FF D7	none	Restart
APP*n*	0x FF E*n*	variable size	Application-Specific
COM	0x FF FE	variable size	Comment (text)
EOI	0x FF D9	none	End of Image

JOINT PHOTOGRAPHIC EXPERT GROUP

JPEG FILE INTERCHANGE FORMAT



```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000: FF D8 FF E0 00 10 .J .F .I .F 00 01 01 01 00 48
010: 00 48 00 00 FF DB 00 43 00 01 01 01 01 01 01 01
020: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
030: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
040: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
050: 01 01 01 01 01 01 01 01 01 FF DB 00 43 01 01 01
060: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
070: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
080: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
090: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 FF C0
0A0: 00 11 08 00 02 00 06 03 01 22 00 02 11 01 03 11
0B0: 01 FF C4 00 15 00 01 01 00 00 00 00 00 00 00 00
0C0: 00 00 00 00 00 00 00 09 FF C4 00 19 10 01 00 02
0D0: 03 00 00 00 00 00 00 00 00 00 00 00 00 00 06 08
0E0: 38 88 B6 FF C4 00 15 01 01 01 00 00 00 00 00 00
0F0: 00 00 00 00 00 00 00 00 07 0A FF C4 00 1C 11 00
100: 01 03 05 00 00 00 00 00 00 00 00 00 00 00 00 08
110: 00 07 B8 09 38 39 76 78 FF DA 00 0C 03 01 00 02
120: 11 03 11 00 3F 00 86 F7 E7 1D A9 16 CA 77 30 D0
130: 14 F7 41 DC 5A 8E FB 31 19 26 5D C4 2A F4 5C 81
140: 7B DB 06 84 A0 75 17 FF D9
```

SEGMENTS — FIELDS — VALUES

START OF IMAGE
```
marker          FFD8
```

APPLICATION0 (DEFAULT HEADER)
```
marker/length   FFE0/16
identifier      JFIF\0
version         1.1
units           1 (dpi)
density         72x72
thumbnail       0x0
```

DEFINE QUANTIZATION TABLE
```
marker/length   FFDB/67
destination     0 (luminance)
table (8x8)     {1} (100% quality)
```

DEFINE QUANTIZATION TABLE
```
marker/length   FFDB/67
destination     1 (chrominance)
table (8x8)     {1} (100% quality)
```

START OF FRAME
```
marker/length   FFC0/17
precision       8
line Nb         2
samples/line    6
components      3
  Id factor table  1 1x1 0 (LumY)
  Id factor table  2 2x2 1 (ChromCb)
  Id factor table  3 2x2 1 (ChromCr)
```

DEFINE HUFFMAN TABLE
```
marker/length   FFC4/21
class           0 (DC)
destination     0
 1 code of 1 bit  00
 1 code of 2 bits 09
```

DEFINE HUFFMAN TABLE
```
marker/length   FFC4/25
class           0 (DC)
destination     0
 1 code of 1 bit  00
 2 code of 3 bits 06 08
 3 code of 4 bits 38 88 B6
```

DEFINE HUFFMAN TABLE
```
marker/length   FFC4/21
class           0 (DC)
destination     1
 1 code of 1 bit  07
 1 code of 2 bits 0A
```

DEFINE HUFFMAN TABLE
```
marker/length   FFC4/28
class           1 (AC)
destination     1
 1 code of 2 bits 08
 3 code of 3 bits 00 07 B8
 5 code of 4 bits 09 38 39 76 78
```

START OF SCAN
```
marker/length   FFDA/12
components      3
 selector / DC, AC table
        1 /  0,  0
        2 /  1,  1
        3 /  1,  1
spectral select.   0..63
successive approx. 00
```

IMAGE DATA (ENTROPY-CODED SEGMENT)
```
86F7E71DA916CA7730D014
F741DC5A8EFB3119265DC4
2AF45C817BDB0684A07517
```

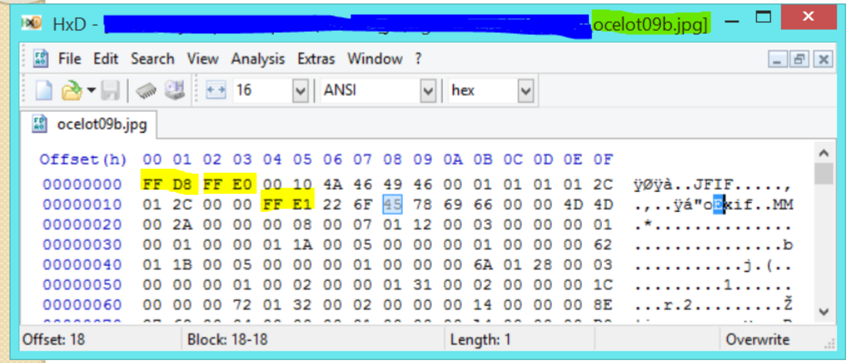
END OF IMAGE
```
marker          FFD9
```

ANGE ALBERTINI http://pics.corkami.com

JPEG IS THE ENCODING STANDARD, JFIF IS THE FILE FORMAT

Ocelot
wild cats from S America

Open any JPEG file, e.g. ocelot09b.jpg provided, with hex editor and search for hex code &FF D8 (i.e. SOI)

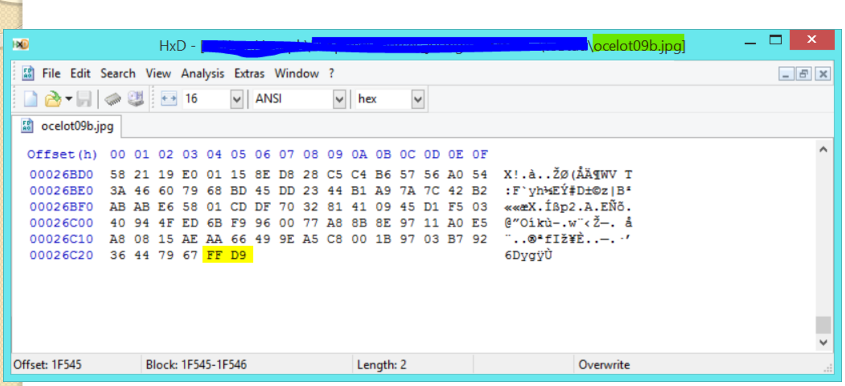


HxD - [redacted] ocelot09b.jpg]

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 01 2C  ÿØÿà..JFIF.....,
00000010  01 2C 00 00 FF E1 22 6F 45 78 69 66 00 00 4D 4D  .,..ÿá"oExif..MM
00000020  00 2A 00 00 00 08 00 07 01 12 00 03 00 00 00 01  .*..............
00000030  00 01 00 00 01 1A 00 05 00 00 00 01 00 00 00 62  ...............b
00000040  01 1B 00 05 00 00 00 01 00 00 00 6A 01 28 00 03  ...........j.(..
00000050  00 00 00 01 00 02 00 00 01 31 00 02 00 00 00 1C  .........1......
00000060  00 00 00 72 01 32 00 02 00 00 00 14 00 00 00 8E  ...r.2.........Ž
```

Offset: 18 | Block: 18-18 | Length: 1 | Overwrite

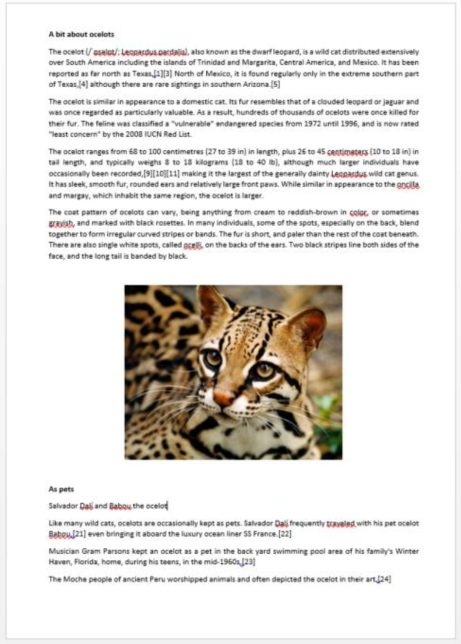
Ensure that end of file is marked with a JPEG EOI marker (i.e. hex & FF D9)


HxD - [redacted]\ocelot09b.jpg]

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00026BD0  58 21 19 E0 01 15 8E D8 28 C5 C4 B6 57 56 A0 54  X!.à..ŽØ(ÅÄ¶WV T
00026BE0  3A 46 60 79 68 BD 45 DD 23 44 B1 A9 7A 7C 42 B2  :F`yh½EÝ#D±©z|B²
00026BF0  AB AB E6 58 01 CD DF 70 32 81 41 09 45 D1 F5 03  ««æX.Íßp2.A.EÑõ.
00026C00  40 94 4F ED 6B F9 96 00 77 A8 8B 8E 97 11 A0 E5  @”Oíkù–.w¨‹Ž—. å
00026C10  A8 08 15 AE AA 66 49 9E A5 C8 00 1B 97 03 B7 92  ¨..®ªfIž¥È..—.·’
00026C20  36 44 79 67 FF D9                                6DygÿÙ
```

Offset: 1F545 Block: 1F545-1F546 Length: 2 Overwrite

Dealing with Embedded Files


A bit about ocelots

The ocelot (/ˈosəlot/; Leopardus pardalis), also known as the dwarf leopard, is a wild cat distributed extensively over South America including the islands of Trinidad and Margarita, Central America, and Mexico. It has been reported as far north as Texas.[1][3] North of Mexico, it is found regularly only in the extreme southern part of Texas,[4] although there are rare sightings in southern Arizona.[5]

The ocelot is similar in appearance to a domestic cat. Its fur resembles that of a clouded leopard or jaguar and was once regarded as particularly valuable. As a result, hundreds of thousands of ocelots were once killed for their fur. The feline was classified a "vulnerable" endangered species from 1972 until 1996, and is now rated "least concern" by the 2008 IUCN Red List.

The ocelot ranges from 68 to 100 centimetres (27 to 39 in) in length, plus 26 to 45 centimeters (10 to 18 in) in tail length, and typically weighs 8 to 18 kilograms (18 to 40 lb), although much larger individuals have occasionally been recorded,[9][10][11] making it the largest of the generally dainty Leopardus wild cat genus. It has sleek, smooth fur, rounded ears and relatively large front paws. While similar in appearance to the oncilla and margay, which inhabit the same region, the ocelot is larger.

The coat pattern of ocelots can vary, being anything from cream to reddish-brown in color, or sometimes grayish, and marked with black rosettes. In many individuals, some of the spots, especially on the back, blend together to form irregular curved stripes or bands. The fur is short, and paler than the rest of the coat beneath. There are also single white spots, called ocelli, on the backs of the ears. Two black stripes line both sides of the face, and the long tail is banded by black.

As pets

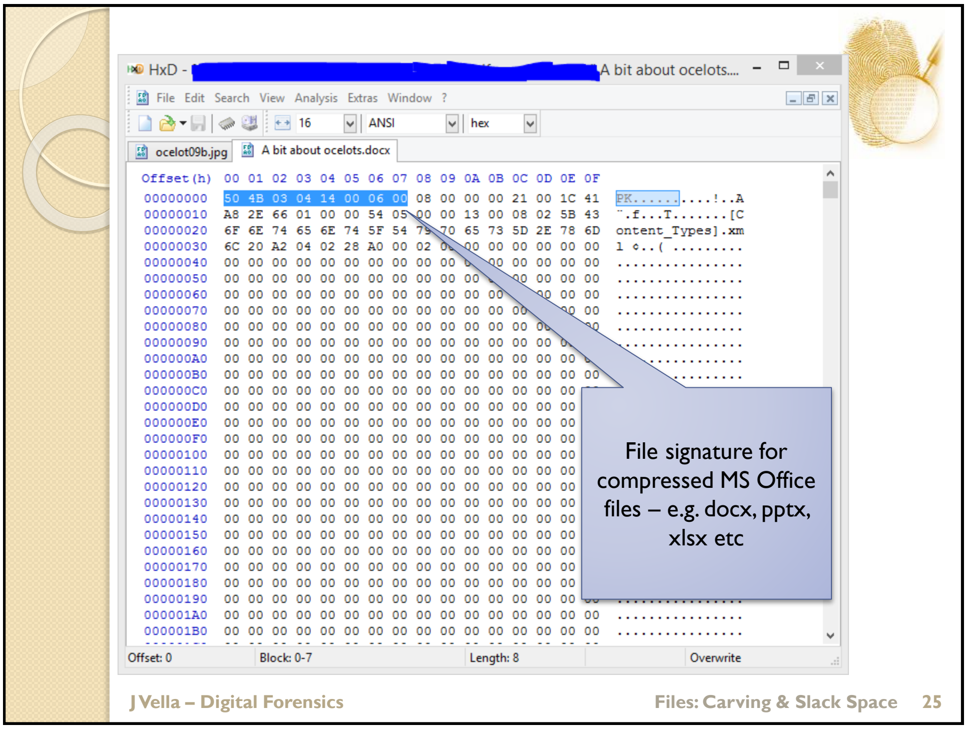
Salvador Dalí and Babou the ocelot

Like many wild cats, ocelots are occasionally kept as pets. Salvador Dalí frequently traveled with his pet ocelot Babou,[21] even bringing it aboard the luxury ocean liner SS France.[22]

Musician Gram Parsons kept an ocelot as a pet in the back yard swimming pool area of his family's Winter Haven, Florida, home, during his teens, in the mid-1960s.[23]

The Moche people of ancient Peru worshipped animals and often depicted the ocelot in their art.[24]

How about extracting a JPEG from a MS word doc (e.g. provided "A bit about ocelots.docx"?

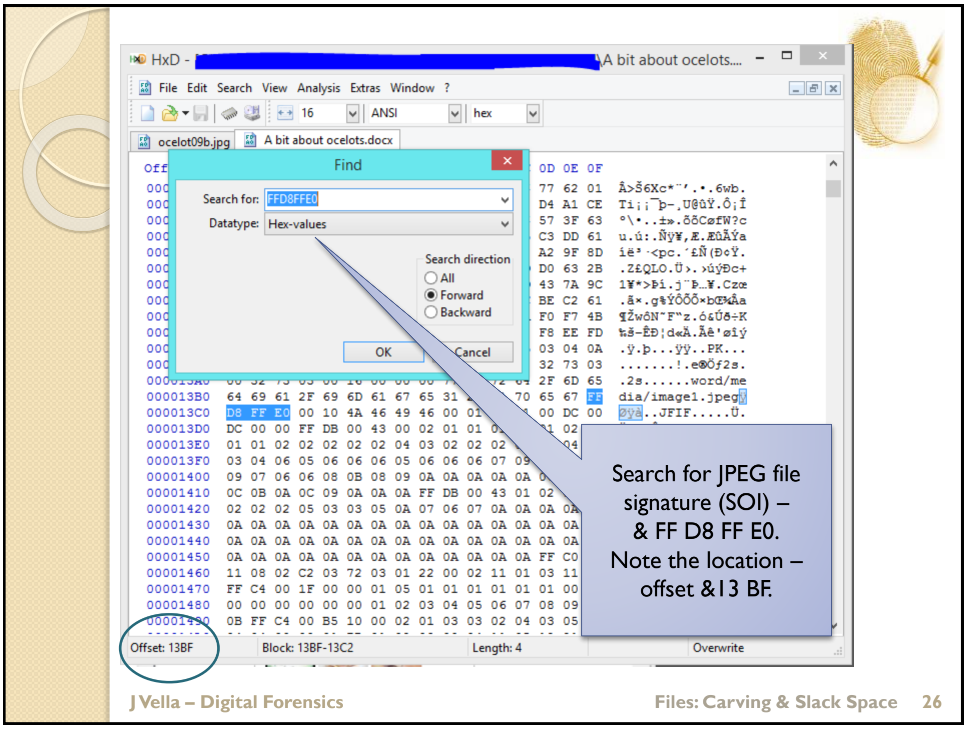


```
HxD - [redacted]A bit about ocelots....
File  Edit  Search  View  Analysis  Extras  Window  ?
16  ANSI  hex
ocelot09b.jpg | A bit about ocelots.docx

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  50 4B 03 04 14 00 06 00 08 00 00 00 21 00 1C 41  PK..........!..A
00000010  A8 2E 66 01 00 00 54 05 00 00 13 00 08 02 5B 43  ¨.f...T.......[C
00000020  6F 6E 74 65 6E 74 5F 54 79 70 65 73 5D 2E 78 6D  ontent_Types].xm
00000030  6C 20 A2 04 02 28 A0 00 02 00 00 00 00 00 00 00  l ¢..( .........
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Offset: 0        Block: 0-7        Length: 8        Overwrite
```

File signature for compressed MS Office files – e.g. docx, pptx, xlsx etc

J Vella – Digital Forensics

Files: Carving & Slack Space 25

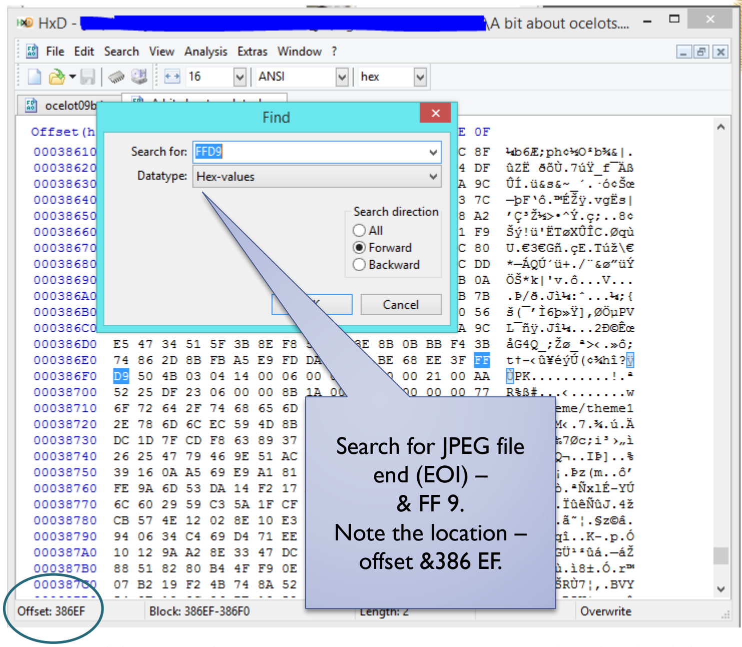


```
HxD - [redacted]A bit about ocelots....
File  Edit  Search  View  Analysis  Extras  Window  ?
16  ANSI  hex
ocelot09b.jpg | A bit about ocelots.docx

Find
Search for: FFD8FFE0
Datatype: Hex-values
Search direction
○ All
● Forward
○ Backward
OK  Cancel

Off                                           0D 0E 0F

000                                           77 62 01  Â>Š6Xc*¨'.•.6wb.
000                                           D4 A1 CE  Ti¡¡¯þ-¸U@ûŸ.Ô¡Î
000                                           57 3F 63  °\•..±».õõCøfW?c
000                                           C3 DD 61  u.ú:.Ñý¥,Æ.ÆûÃÝa
000                                           A2 9F 8D  ìë³·<pc.´£Ñ(Ð¢Ÿ.
000                                           D0 63 2B  .Z£QLO.Ü›.›úýÐc+
000                                           43 7A 9C  1¥*>Þí.j¨Þ…¥.Czœ
000                                           BE C2 61  .ã×.g‰ÝÔÕÕ×bŒ¾Âa
000                                           F0 F7 4B  ¶ŽwôN˜F"z.ó&Úð÷K
000                                           F8 EE FD  ‰š–ÊÐ¦d«Ä.Ãê'øîý
000                                           03 04 0A  .ÿ.þ...ÿÿ..PK...
000                                           32 73 03  .......!.e®Öf2s.
000013A0  00 32 73 03 00 10 00 00 00 77 6F 72 64 2F 6D 65  .2s......word/me
000013B0  64 69 61 2F 69 6D 61 67 65 31 2E 6A 70 65 67 FF  dia/image1.jpegÿ
000013C0  D8 FF E0 00 10 4A 46 49 46 00 01       00 DC 00  øÿà..JFIF.....Ü.
000013D0  DC 00 00 FF DB 00 43 00 02 01 01 01    01 02
000013E0  01 01 02 02 02 02 02 04 03 02 02 02       04
000013F0  03 04 06 05 06 06 06 05 06 06 06 07 09
00001400  09 07 06 06 08 0B 08 09 0A 0A 0A 0A 0A 0
00001410  0C 0B 0A 0C 09 0A 0A 0A FF DB 00 43 01 02
00001420  02 02 02 05 03 03 05 0A 07 06 07 0A 0A 0A 0A
00001430  0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
00001440  0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
00001450  0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A FF C0
00001460  11 08 02 C2 03 72 03 01 22 00 02 11 01 03 11
00001470  FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00
00001480  00 00 00 00 00 00 01 02 03 04 05 06 07 08 09
00001490  0B FF C4 00 B5 10 00 02 01 03 03 02 04 03 05

Offset: 13BF        Block: 13BF-13C2        Length: 4        Overwrite
```

Search for JPEG file signature (SOI) – & FF D8 FF E0. Note the location – offset &13 BF.

J Vella – Digital Forensics

Files: Carving & Slack Space 26



HxD - [redacted] A bit about ocelots....
File Edit Search View Analysis Extras Window ?
16 ANSI hex
ocelot09b
Find
OK
Cancel

Search for: FFD9

Datatype: Hex-values

Search direction

○ All

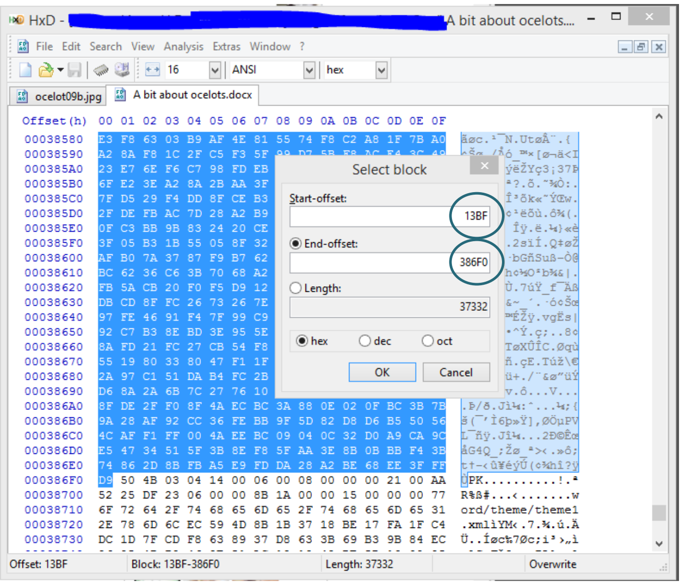
● Forward

○ Backward

```
000386D0  E5 47 34 51 5F 3B 8E F8
000386E0  74 86 2D 8B FB A5 E9 FD DA
000386F0  D9 50 4B 03 04 14 00 06 00
```

Offset: 386EF | Block: 386EF-386F0 | Length: 2 | Overwrite

Search for JPEG file end (EOI) – & FF 9.
Note the location – offset &386 EF.



HxD - [redacted] A bit about ocelots....
File Edit Search View Analysis Extras Window ?
16 ANSI hex
ocelot09b.jpg
A bit about ocelots.docx
Select block
OK
Cancel

Start-offset: 13BF

● End-offset: 386F0

○ Length: 37332

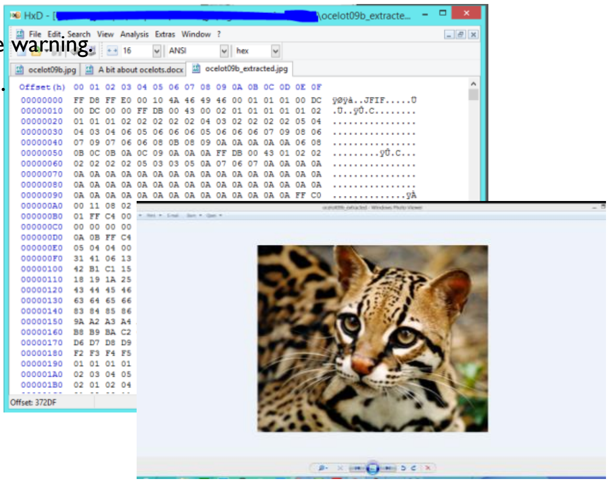
● hex ○ dec ○ oct

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00038580  E3 F8 63 03 B9 AF 4E 81 55 74 F8 C2 A8 1F 7B A0
000386A0  8F DE 2F F0 8F 4A EC BC 3A 88 0E 02 0F BC 3B 7B
000386B0  9A 28 AF 92 CC 36 FE BB 9F 5D 82 D8 D6 B5 50 56
000386C0  4C AF F1 FF 00 4A EE BC 09 04 0C 32 D0 A9 CA 9C
000386D0  E5 47 34 51 5F 3B 8E F8 5F AA 3E 8B 0B BB F4 3B
000386E0  74 86 2D 8B FB A5 E9 FD DA 28 A2 BE 68 EE 3F FF
000386F0  D9 50 4B 03 04 14 00 06 00 08 00 00 00 21 00 AA
00038700  52 25 DF 23 06 00 00 8B 1A 00 00 15 00 00 00 77
00038710  6F 72 64 2F 74 68 65 6D 65 2F 74 68 65 6D 65 31
00038720  2E 78 6D 6C EC 59 4D 8B 1B 37 18 BE 17 FA 1F C4
00038730  DC 1D 7F CD F8 63 89 37 D8 63 3B 69 B3 9B 84 EC
```

Offset: 13BF | Block: 13BF-386F0 | Length: 37332 | Overwrite

Create *new* JPEG

- Copy Selection.
- Start a new file (in hex).
- Copy Insert
  - Accept file size warning.
- Save As new file.
- Preview file!



HxD - [[redacted]\ocelot09b_extracte...

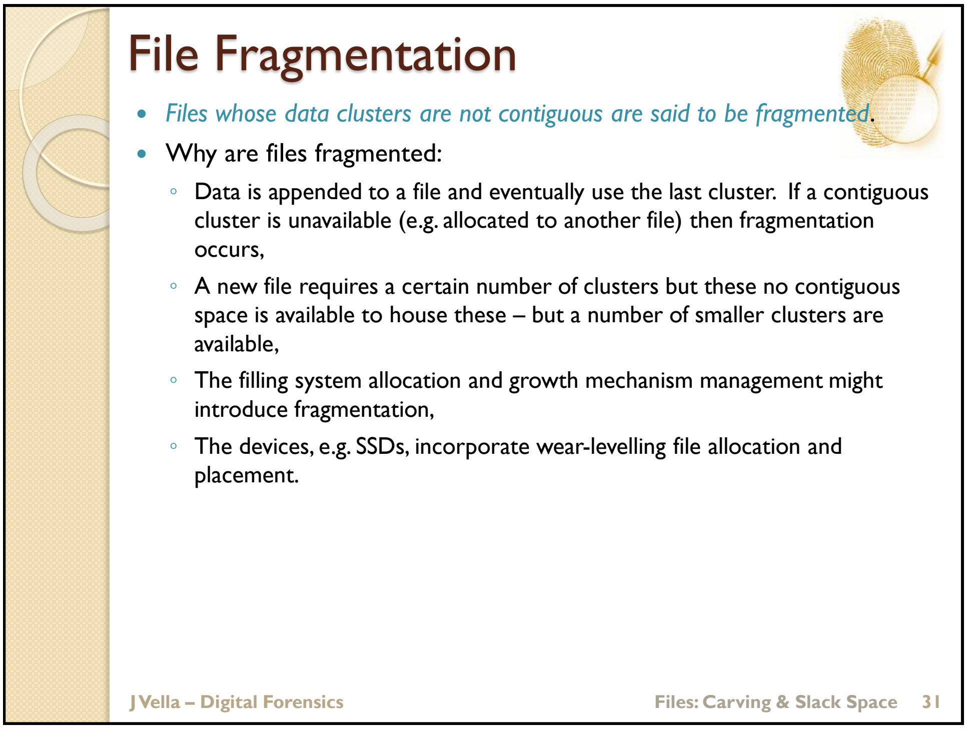
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 DC  ÿØÿà..JFIF.....Ü
00000010  00 DC 00 00 FF DB 00 43 00 02 01 01 01 01 01 02  .Ü..ÿÛ.C........
00000020  01 01 01 02 02 02 02 02 04 03 02 02 02 02 05 04  ................
00000030  04 03 04 06 05 06 06 06 05 06 06 06 07 09 08 06  ................
00000040  07 09 07 06 06 08 0B 08 09 0A 0A 0A 0A 0A 06 08  ................
00000050  0B 0C 0B 0A 0C 09 0A 0A 0A FF DB 00 43 01 02 02  .........ÿÛ.C...
00000060  02 02 02 02 05 03 03 05 0A 07 06 07 0A 0A 0A 0A  ................
00000070  0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A  ................
00000080  0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A  ................
00000090  0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A FF C0  ..............ÿÀ
000000A0  00 11 08 02
000000B0  01 FF C4 00
000000C0  00 00 00 00
000000D0  0A 0B FF C4
000000E0  05 04 04 00
000000F0  31 41 06 13
00000100  42 B1 C1 15
00000110  18 19 1A 25
00000120  43 44 45 46
00000130  63 64 65 66
00000140  83 84 85 86
00000150  9A A2 A3 A4
00000160  B8 B9 BA C2
00000170  D6 D7 D8 D9
00000180  F2 F3 F4 F5
00000190  01 01 01 01
000001A0  02 03 04 05
000001B0  02 01 02 04
```

Offset: 372DF

Header-Footer Carving: Summary

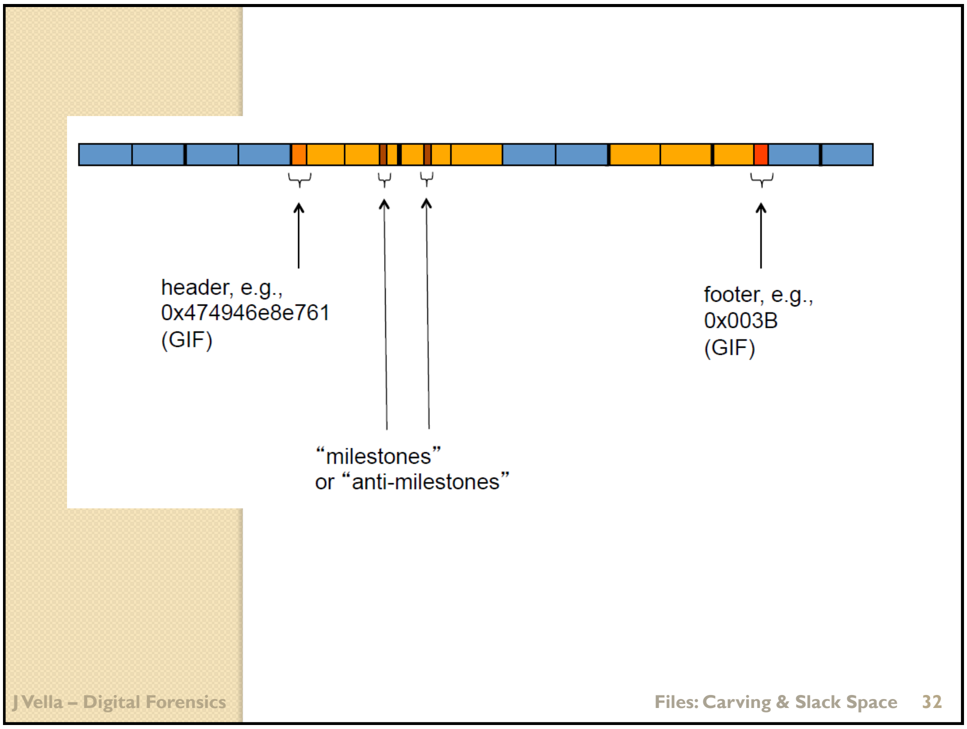
- As we have seen this type of carving is really simple and straightforward.
- But:
  - The files to carve *are not fragmented* into a sequence of clusters;
  - The first block (and beginning of file) *is available*; and
  - The signature being searched *does not match* with data (i.e. false positives).

Short Name	Bytes	Payload	Name
SOI	0x FF D8	none	Start of Image
SOF0	0x FF C0	variable size	Start of Frame (Baseline DCT)
SOF2	0x FF C2	variable size	Start of Frame (Progressive DCT)
DHT	0x FF C4	variable size	Define Huffman Table(s)
DQT	0x FF DB	variable size	Define Quantization Table(s)
DRI	0x FF DD	2 bytes	Define Restart Interval
SOS	0x FF DA	variable size	Start of Stream
RST*n*	0x FF D0…0x FF D7	none	Restart
APP*n*	0x FF E*n*	variable size	Application-Specific
COM	0x FF FE	variable size	Comment (text)
EOI	0x FF D9	none	End of Image



File Fragmentation

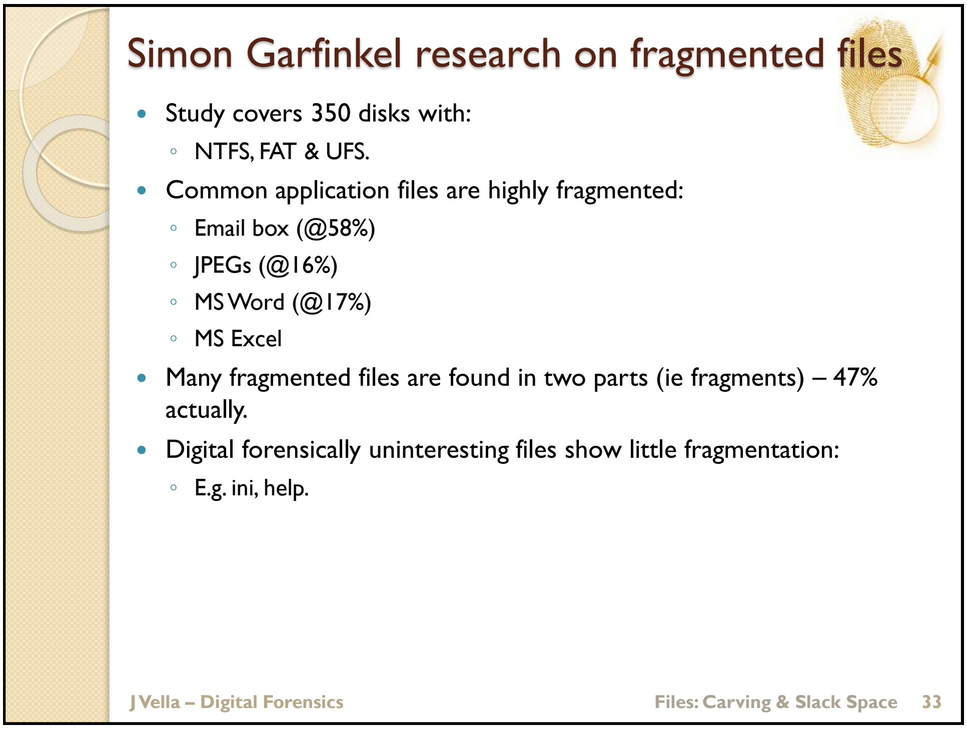
- *Files whose data clusters are not contiguous are said to be fragmented*.
- Why are files fragmented:
  - Data is appended to a file and eventually use the last cluster. If a contiguous cluster is unavailable (e.g. allocated to another file) then fragmentation occurs,
  - A new file requires a certain number of clusters but these no contiguous space is available to house these – but a number of smaller clusters are available,
  - The filling system allocation and growth mechanism management might introduce fragmentation,
  - The devices, e.g. SSDs, incorporate wear-levelling file allocation and placement.



header, e.g.,
0x474946e8e761
(GIF)

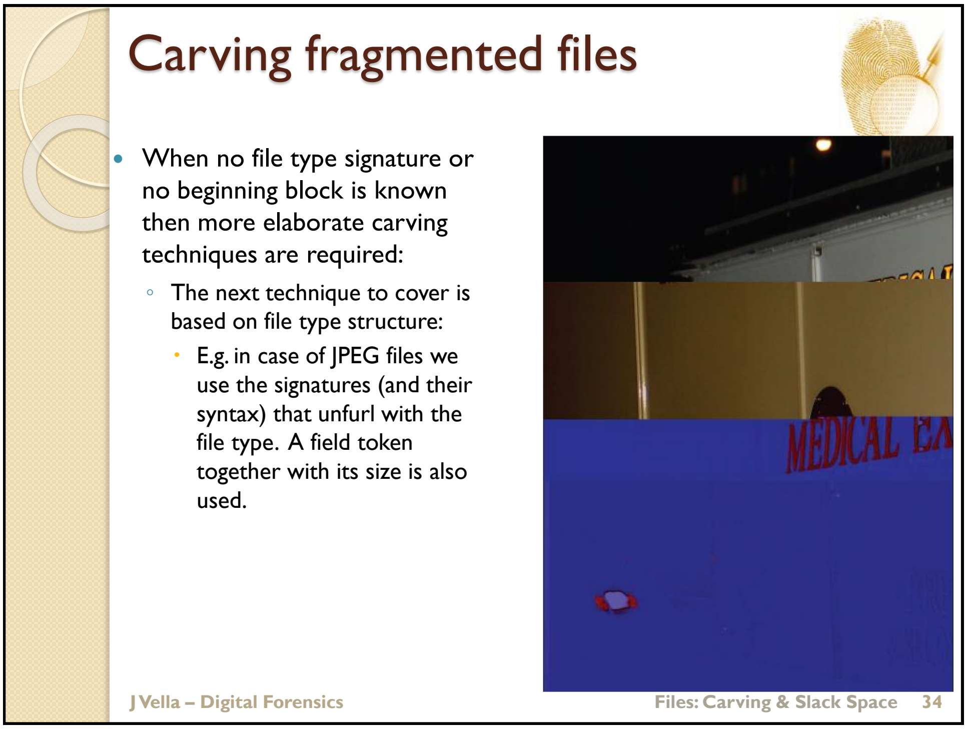
footer, e.g.,
0x003B
(GIF)

“milestones”
or “anti-milestones”



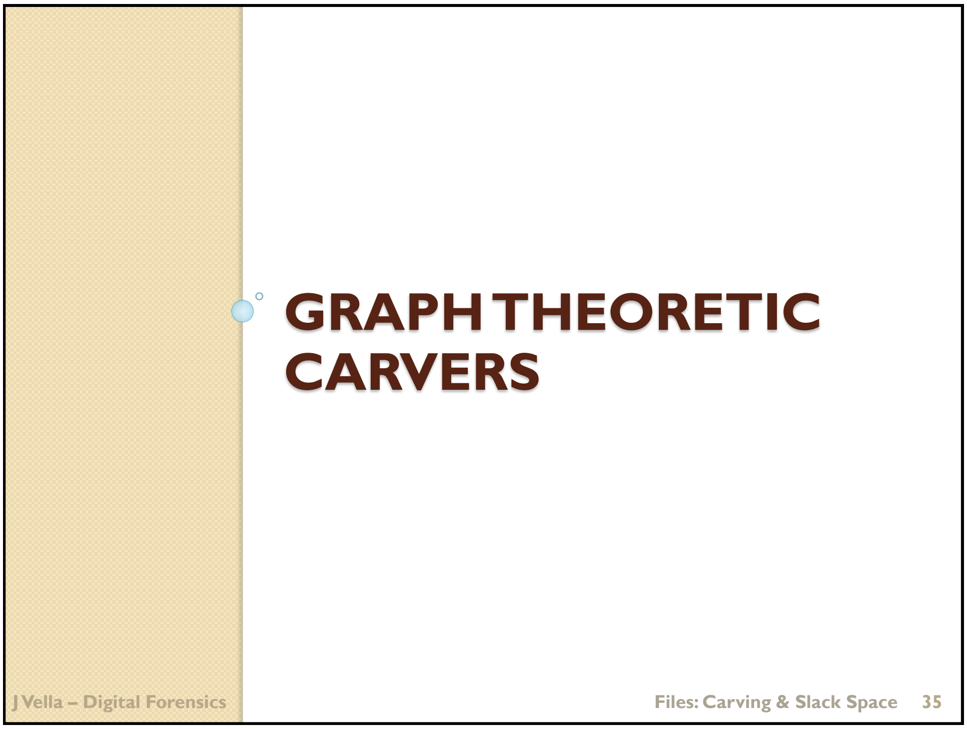
Simon Garfinkel research on fragmented files

- Study covers 350 disks with:
  - NTFS, FAT & UFS.
- Common application files are highly fragmented:
  - Email box (@58%)
  - JPEGs (@16%)
  - MS Word (@17%)
  - MS Excel
- Many fragmented files are found in two parts (ie fragments) – 47% actually.
- Digital forensically uninteresting files show little fragmentation:
  - E.g. ini, help.

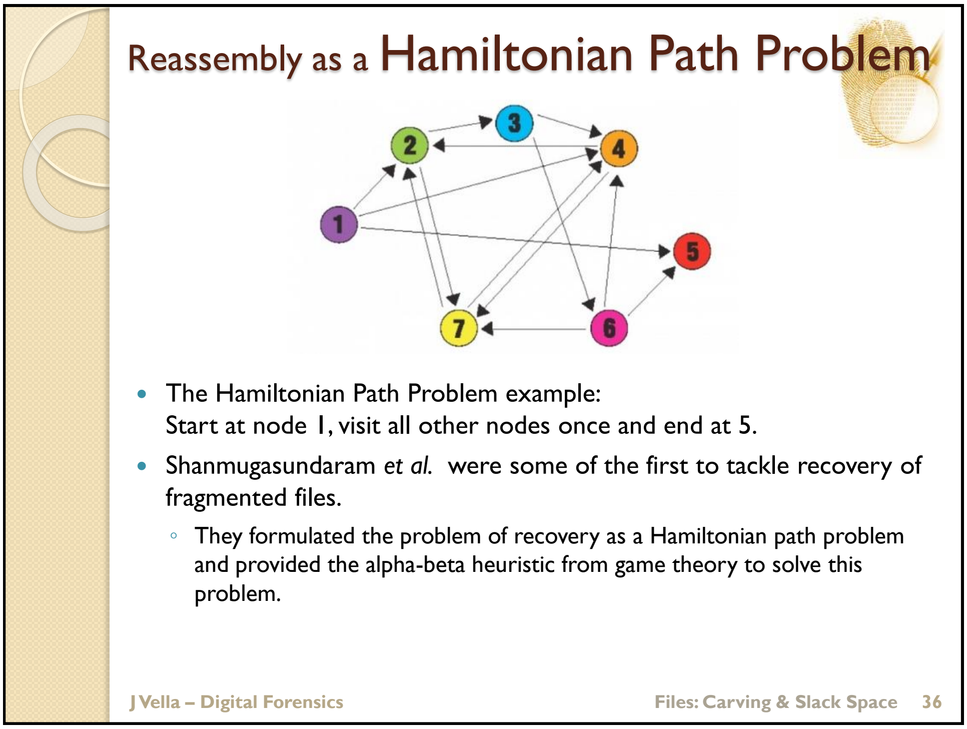


Carving fragmented files

- When no file type signature or no beginning block is known then more elaborate carving techniques are required:
  - The next technique to cover is based on file type structure:
    - E.g. in case of JPEG files we use the signatures (and their syntax) that unfurl with the file type. A field token together with its size is also used.



GRAPH THEORETIC CARVERS



Reassembly as a Hamiltonian Path Problem

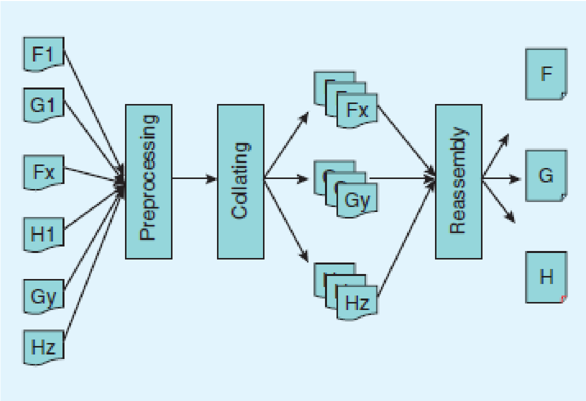
- The Hamiltonian Path Problem example:
  Start at node 1, visit all other nodes once and end at 5.
- Shanmugasundaram *et al.* were some of the first to tackle recovery of fragmented files.
  - They formulated the problem of recovery as a Hamiltonian path problem and provided the alpha-beta heuristic from game theory to solve this problem.

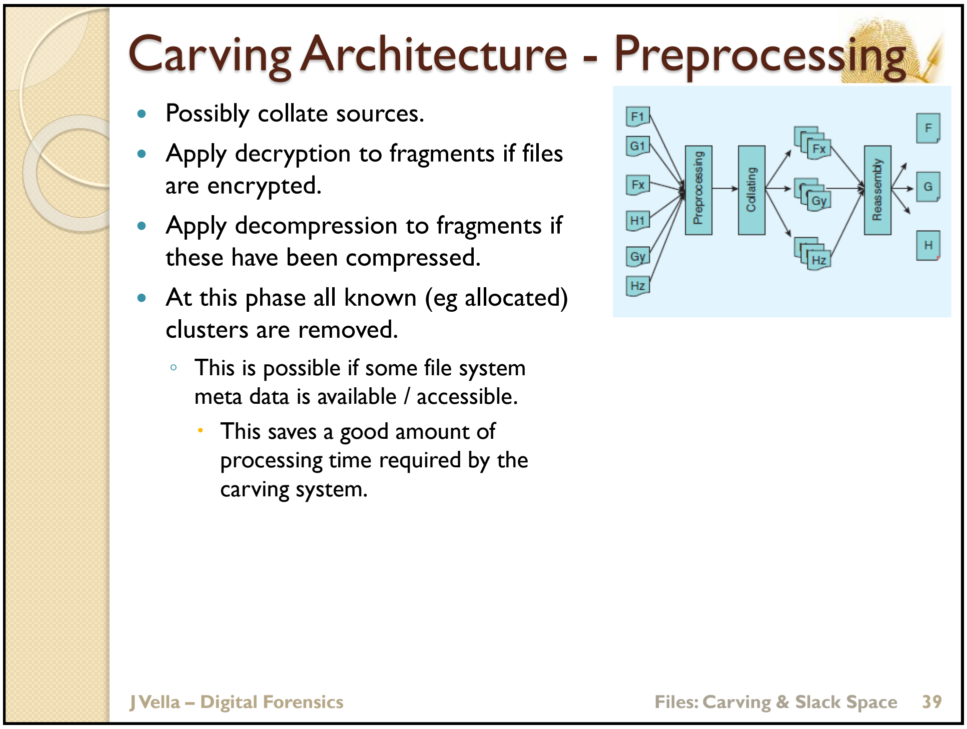
Problem Definition

- Given a set of unallocated clusters b0, b1 . . . bn belonging to a document A, one would like to compute a permutation P of the set that represents the original structure of the document.
    - To determine the correct cluster ordering one needs to identify fragment pairs that are adjacent in the original document. This is achieved by assigning candidate weights ($W_{x,y}$) between two clusters b_x and b_y that represents the likelihood that cluster b_y follows b_x.
    - The proper sequence P is a path in this graph that traverses all the vertices and maximizes the sum of candidate weights along that path. Finding this path is equivalent to finding a maximum weight Hamiltonian path in a complete graph.

Carving Architecture

- Design of an application for carving randomly allocated fragment.
    - The application has a set of default computed weights that mimic common fragmentation tactics in filing systems.
    - There are three distinct phases:
        - Pre-processing;
        - Collating;
        - Reassembly.

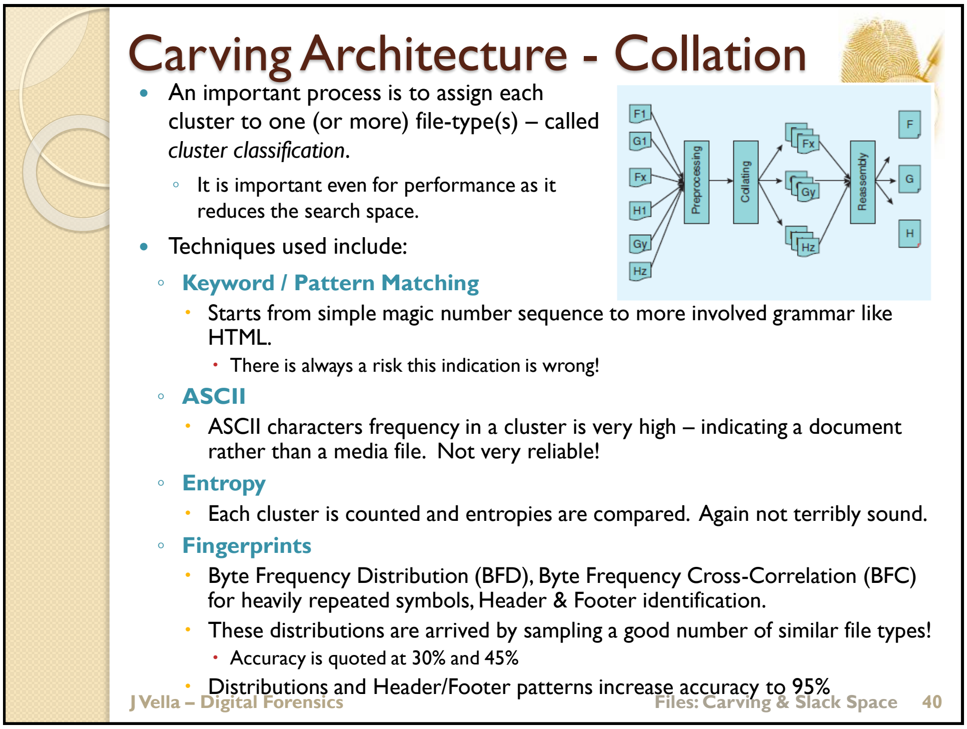




Carving Architecture - Preprocessing

F1
G1
Fx
H1
Gy
Hz
Preprocessing
Collating
Fx
Gy
Hz
Reassembly
F
G
H

- Possibly collate sources.
- Apply decryption to fragments if files are encrypted.
- Apply decompression to fragments if these have been compressed.
- At this phase all known (eg allocated) clusters are removed.
  - This is possible if some file system meta data is available / accessible.
    - This saves a good amount of processing time required by the carving system.



Carving Architecture - Collation

F1
G1
Fx
H1
Gy
Hz
Preprocessing
Collating
Fx
Gy
Hz
Reassembly
F
G
H

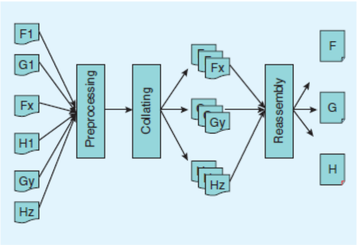
- An important process is to assign each cluster to one (or more) file-type(s) – called *cluster classification*.
  - It is important even for performance as it reduces the search space.
- Techniques used include:
  - **Keyword / Pattern Matching**
    - Starts from simple magic number sequence to more involved grammar like HTML.
      - There is always a risk this indication is wrong!
  - **ASCII**
    - ASCII characters frequency in a cluster is very high – indicating a document rather than a media file. Not very reliable!
  - **Entropy**
    - Each cluster is counted and entropies are compared. Again not terribly sound.
  - **Fingerprints**
    - Byte Frequency Distribution (BFD), Byte Frequency Cross-Correlation (BFC) for heavily repeated symbols, Header & Footer identification.
    - These distributions are arrived by sampling a good number of similar file types!
      - Accuracy is quoted at 30% and 45%
    - Distributions and Header/Footer patterns increase accuracy to 95%

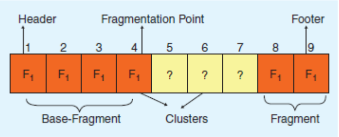
Aside: progress in fingerprinting

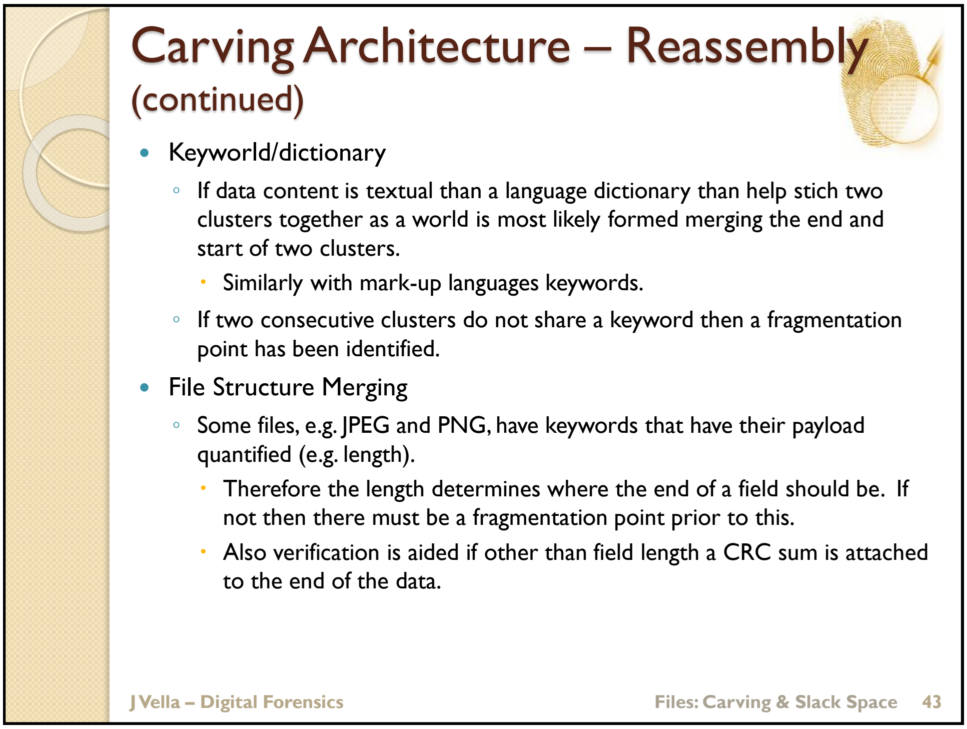
- Fingerprinting is also used to monitor anomalous data in network transmissions (Wang & Stalfo, and Wang et al):
  - Specifically using the BFDs and standard deviation of standard traffic to and then comparing both.
    - The same technique was improved by using BFA on set of models, rather than one average to capture all possible representations in a file type.
    - Research quotes very high success rates –e.g. 100%. For JPEGs it was above 75%.
      - But success rate very much linked to size of file; bigger the higher success.

Carving Architecture - Reassembly

- Basically it involves finding the fragmentation point of each unrecovered file and then finding the next fragment's starting point.
  - Earlier we stated that Garfinkel's work indicated that most files, if fragmented, are in two pieces.
- Therefore this process needs to find the starting cluster and then uses a technique to identify the ending point of the fragment.
  - If a file has three fragments then there are two fragmentation points.
- An early method studied the structure and sequence of bytes that straddled the boundary of the two clusters.

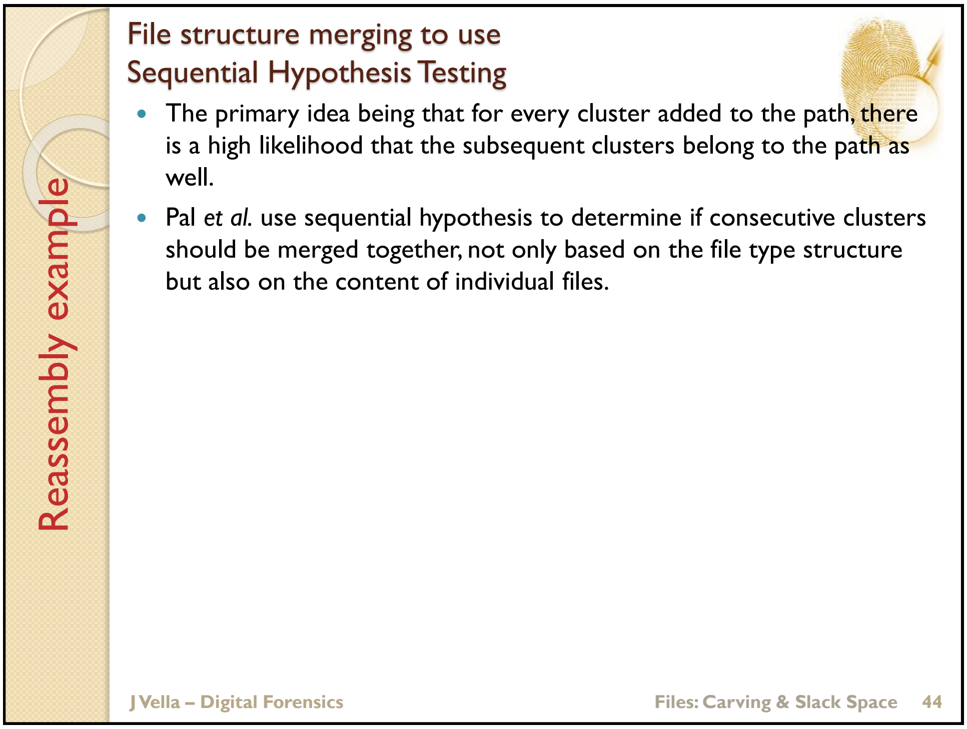






Carving Architecture – Reassembly (continued)

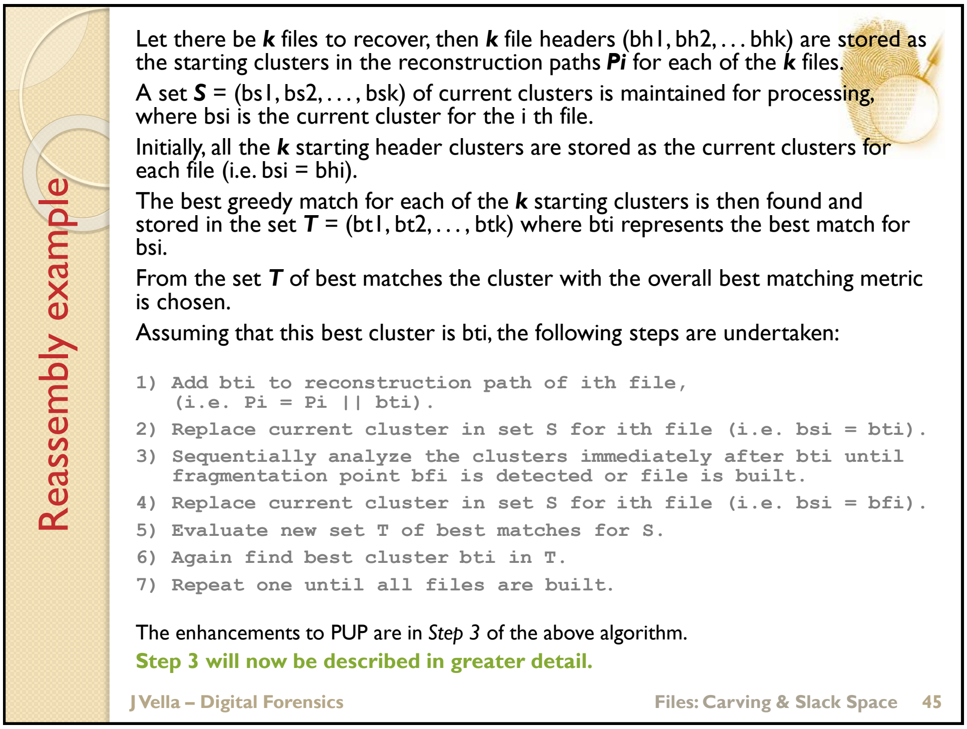
- Keyworld/dictionary
  - If data content is textual than a language dictionary than help stich two clusters together as a world is most likely formed merging the end and start of two clusters.
    - Similarly with mark-up languages keywords.
  - If two consecutive clusters do not share a keyword then a fragmentation point has been identified.
- File Structure Merging
  - Some files, e.g. JPEG and PNG, have keywords that have their payload quantified (e.g. length).
    - Therefore the length determines where the end of a field should be. If not then there must be a fragmentation point prior to this.
    - Also verification is aided if other than field length a CRC sum is attached to the end of the data.



Reassembly example

File structure merging to use Sequential Hypothesis Testing

- The primary idea being that for every cluster added to the path, there is a high likelihood that the subsequent clusters belong to the path as well.
- Pal *et al.* use sequential hypothesis to determine if consecutive clusters should be merged together, not only based on the file type structure but also on the content of individual files.



Reassembly example

Let there be ***k*** files to recover, then ***k*** file headers (bh1, bh2, . . . bhk) are stored as the starting clusters in the reconstruction paths ***Pi*** for each of the ***k*** files.

A set ***S*** = (bs1, bs2, . . . , bsk) of current clusters is maintained for processing, where bsi is the current cluster for the i th file.

Initially, all the ***k*** starting header clusters are stored as the current clusters for each file (i.e. bsi = bhi).

The best greedy match for each of the ***k*** starting clusters is then found and stored in the set ***T*** = (bt1, bt2, . . . , btk) where bti represents the best match for bsi.

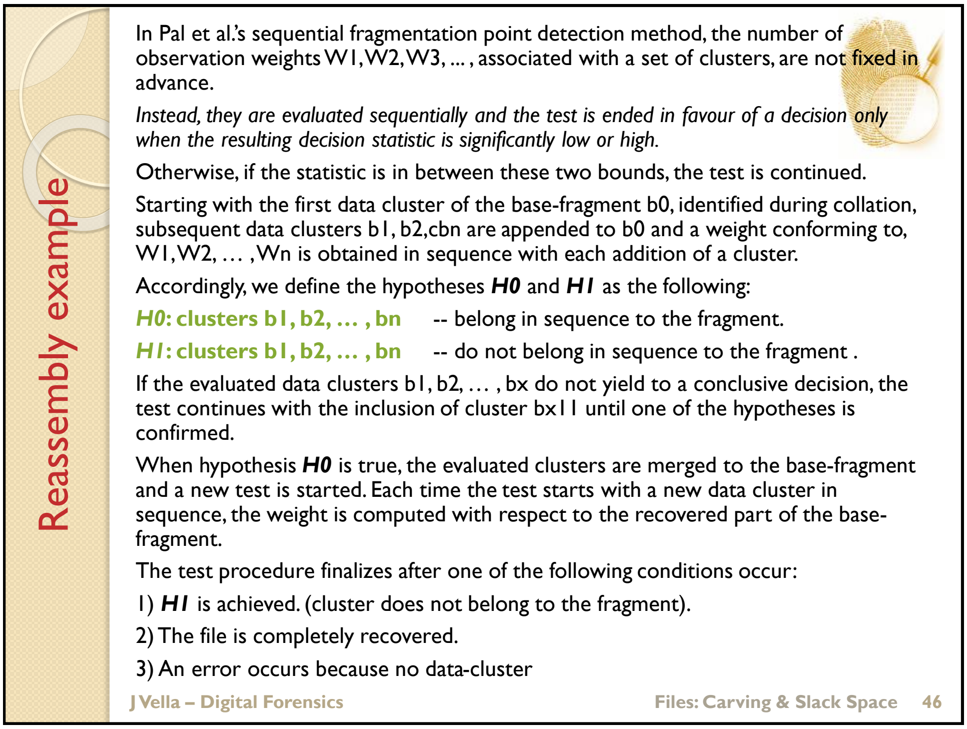
From the set ***T*** of best matches the cluster with the overall best matching metric is chosen.

Assuming that this best cluster is bti, the following steps are undertaken:

```
1) Add bti to reconstruction path of ith file,
   (i.e. Pi = Pi || bti).
2) Replace current cluster in set S for ith file (i.e. bsi = bti).
3) Sequentially analyze the clusters immediately after bti until
   fragmentation point bfi is detected or file is built.
4) Replace current cluster in set S for ith file (i.e. bsi = bfi).
5) Evaluate new set T of best matches for S.
6) Again find best cluster bti in T.
7) Repeat one until all files are built.
```

The enhancements to PUP are in *Step 3* of the above algorithm.

Step 3 will now be described in greater detail.



Reassembly example

In Pal et al.'s sequential fragmentation point detection method, the number of observation weights W1, W2, W3, … , associated with a set of clusters, are not fixed in advance.

Instead, they are evaluated sequentially and the test is ended in favour of a decision only when the resulting decision statistic is significantly low or high.

Otherwise, if the statistic is in between these two bounds, the test is continued.

Starting with the first data cluster of the base-fragment b0, identified during collation, subsequent data clusters b1, b2,cbn are appended to b0 and a weight conforming to, W1, W2, … , Wn is obtained in sequence with each addition of a cluster.

Accordingly, we define the hypotheses ***H0*** and ***H1*** as the following:

***H0*: clusters b1, b2, … , bn** -- belong in sequence to the fragment.

***H1*: clusters b1, b2, … , bn** -- do not belong in sequence to the fragment .

If the evaluated data clusters b1, b2, … , bx do not yield to a conclusive decision, the test continues with the inclusion of cluster bx11 until one of the hypotheses is confirmed.

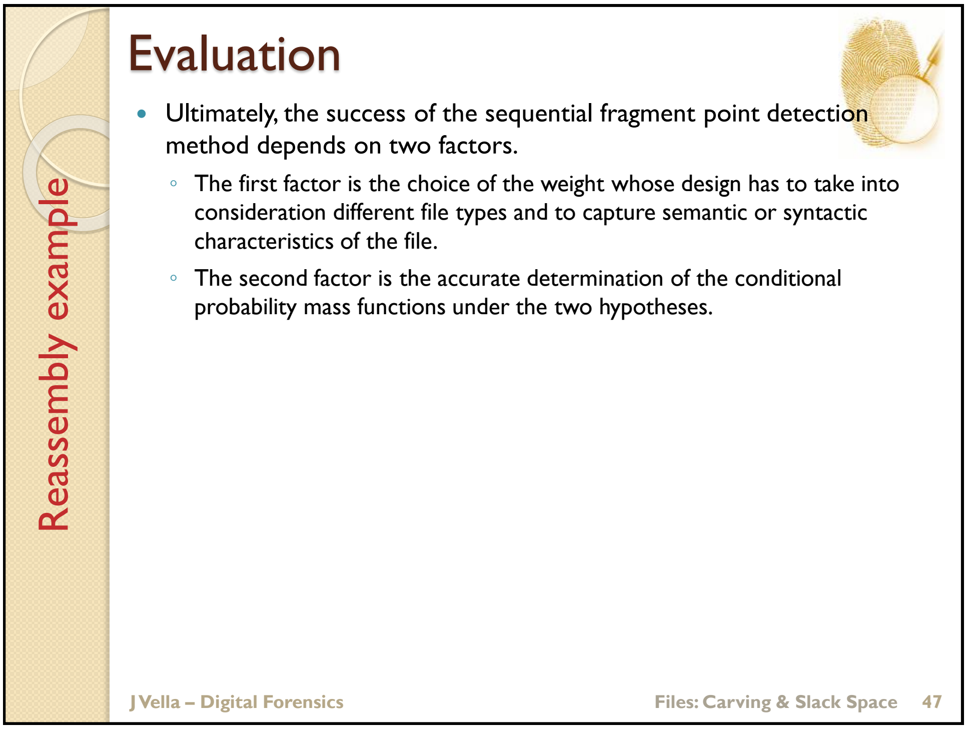
When hypothesis ***H0*** is true, the evaluated clusters are merged to the base-fragment and a new test is started. Each time the test starts with a new data cluster in sequence, the weight is computed with respect to the recovered part of the base-fragment.

The test procedure finalizes after one of the following conditions occur:

1) ***H1*** is achieved. (cluster does not belong to the fragment).

2) The file is completely recovered.

3) An error occurs because no data-cluster



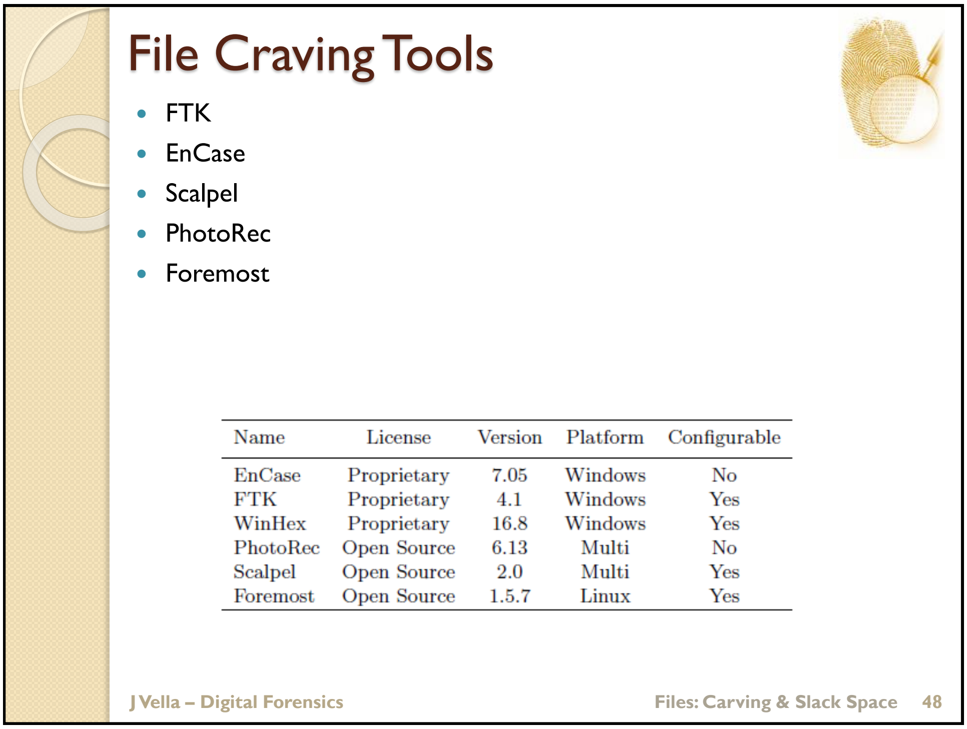
Reassembly example

Evaluation

- Ultimately, the success of the sequential fragment point detection method depends on two factors.
  - The first factor is the choice of the weight whose design has to take into consideration different file types and to capture semantic or syntactic characteristics of the file.
  - The second factor is the accurate determination of the conditional probability mass functions under the two hypotheses.

J Vella – Digital Forensics

Files: Carving & Slack Space 47



File Craving Tools

- FTK
- EnCase
- Scalpel
- PhotoRec
- Foremost

Name	License	Version	Platform	Configurable
EnCase	Proprietary	7.05	Windows	No
FTK	Proprietary	4.1	Windows	Yes
WinHex	Proprietary	16.8	Windows	Yes
PhotoRec	Open Source	6.13	Multi	No
Scalpel	Open Source	2.0	Multi	Yes
Foremost	Open Source	1.5.7	Linux	Yes

J Vella – Digital Forensics

Files: Carving & Slack Space 48