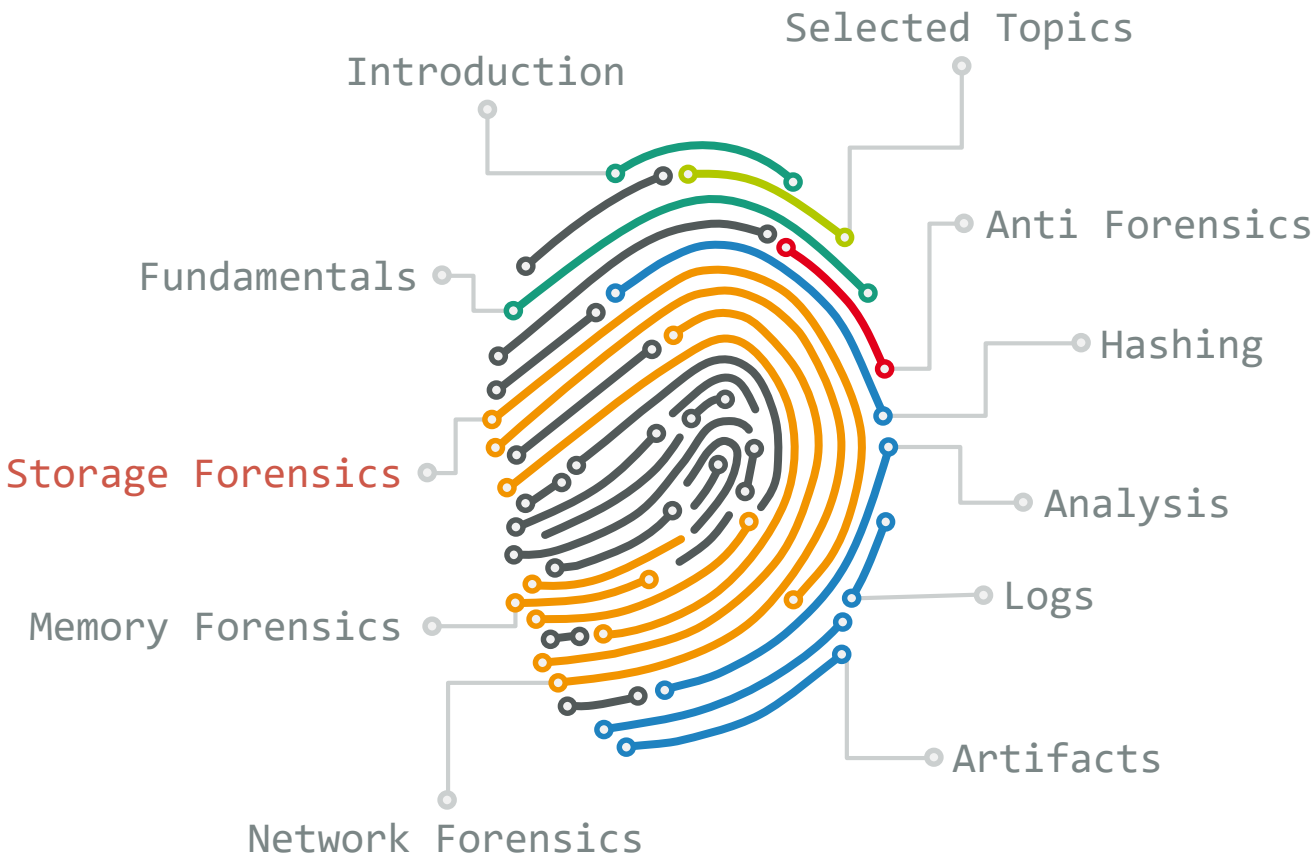
Prof. Dr. Elmar Padilla et al.

Digitale Forensik

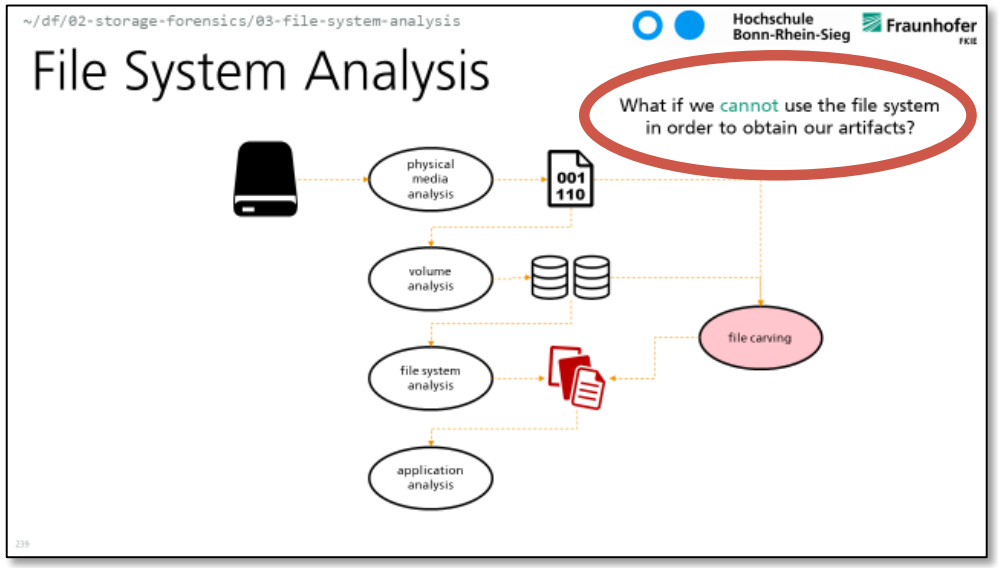
02 - Storage Forensics



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving



What if we cannot use the file system to obtain artifacts?

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving

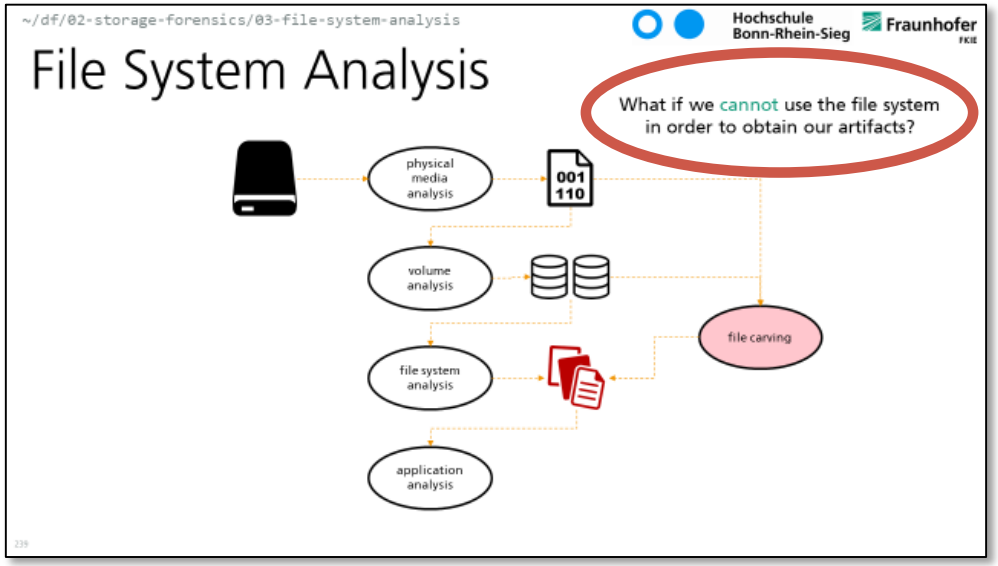
→ overwritten file system metadata

→ damaged file system

→ untrustworthy file system

→ wiped file system

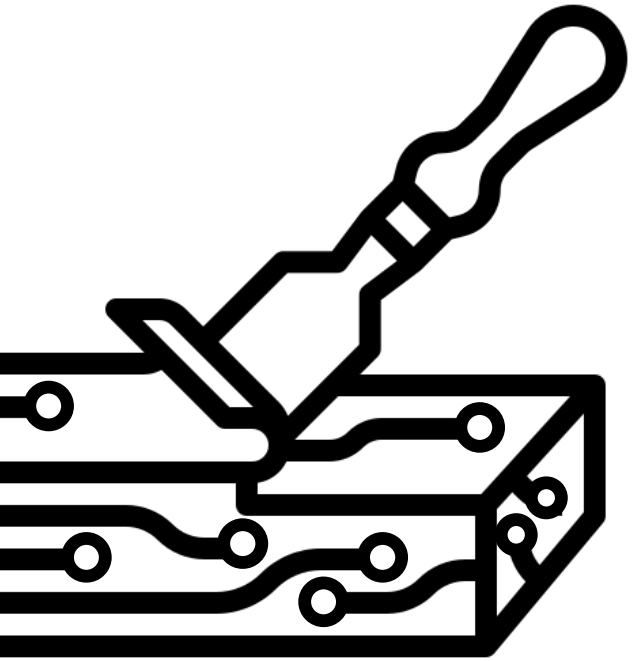
→ no file system



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving



Use

file format information

instead of

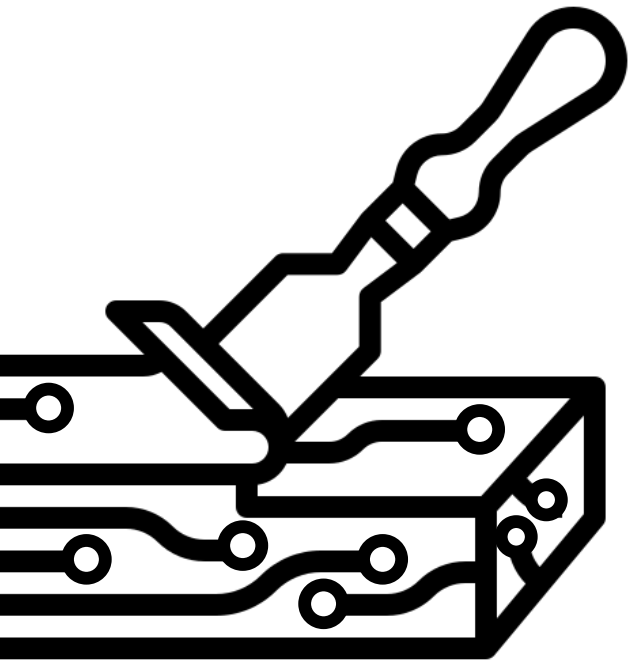
file system information

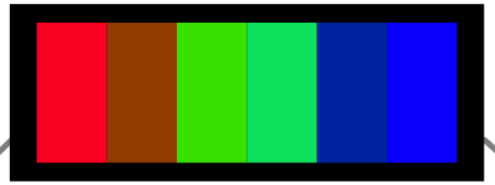
to recover files.

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving





	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							

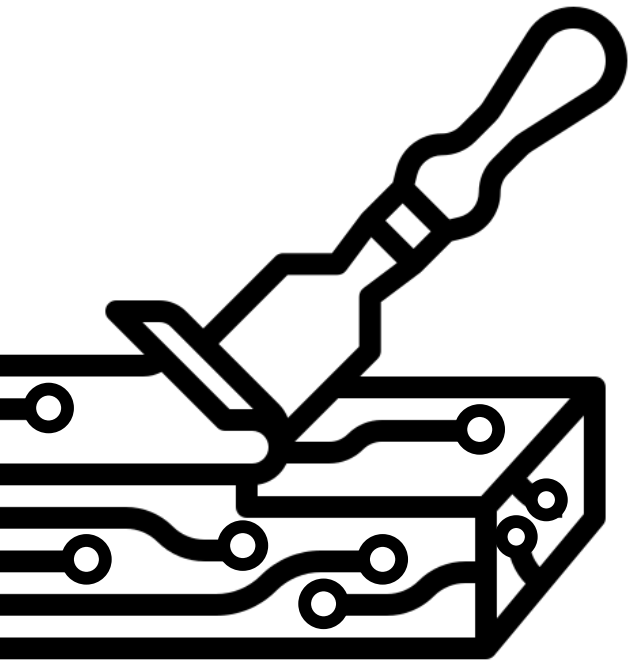
SEGMENTS	FIELDS	VALUES
START OF IMAGE	marker	FFD8
APPLICATION0 (DEFAULT HEADER)	marker/length	FFE0/16
	identifier	JFIF\0
	version	1.1
	units	1 (dpi)
	density	72x72
	thumbnail	0x0
DEFINE QUANTIZATION TABLE	marker/length	FFDB/67
	destination	0 (luminance)
	table (8x8)	{1} (100% quality)
DEFINE QUANTIZATION 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

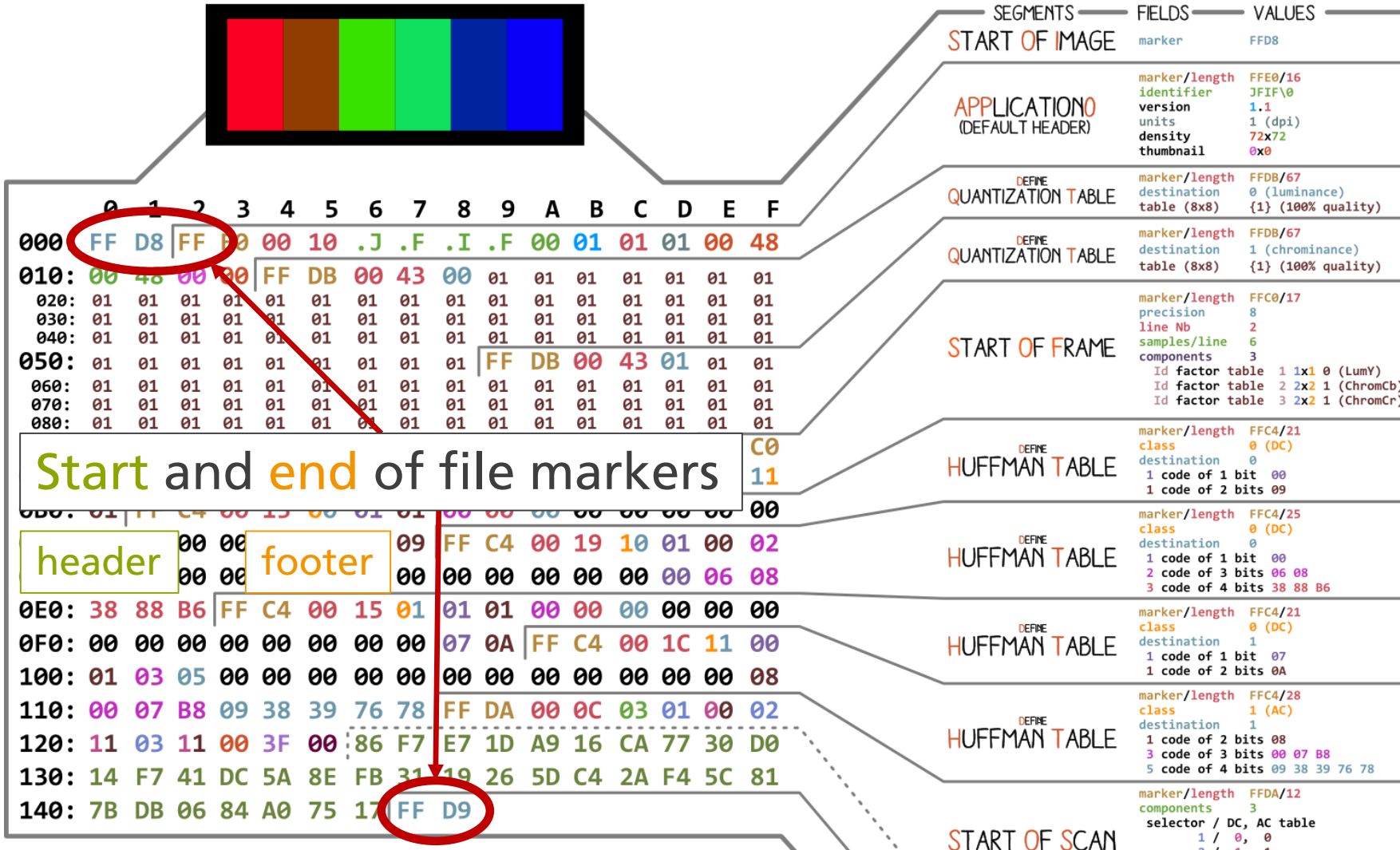
ANGE ALBERTINI
http://pics.corkami.com

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving



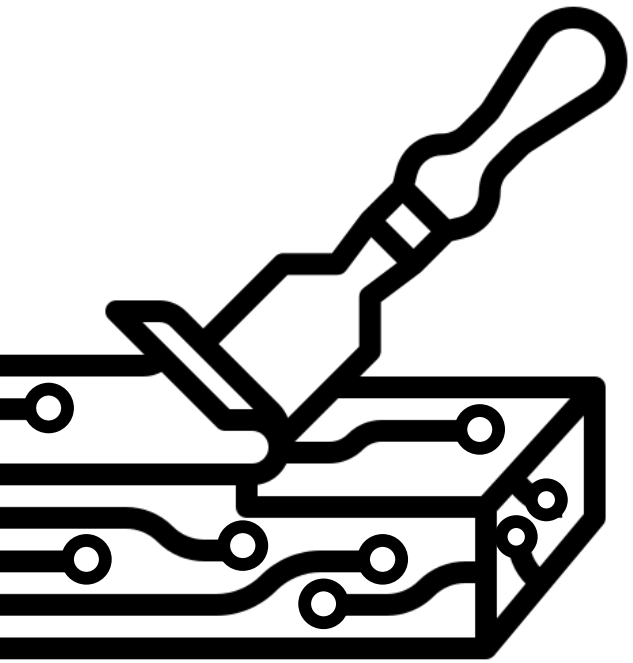


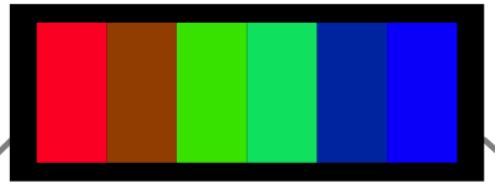
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving

Internal file structure





```
     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
```

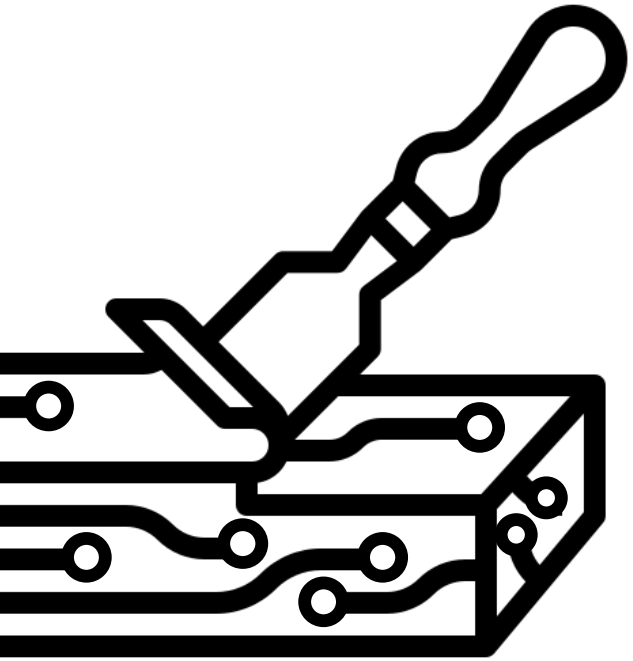
SEGMENTS	FIELDS	VALUES
START OF IMAGE	marker	FFD8
APPLICATION0 (DEFAULT HEADER)	marker/length	FFE0/16
	identifier	JFIF\0
	version	1.1
	units	1 (dpi)
	density	72x72
	thumbnail	0x0
DEFINE QUANTIZATION TABLE	marker/length	FFDB/67
	destination	0 (luminance)
	table (8x8)	{1} (100% quality)
DEFINE QUANTIZATION 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

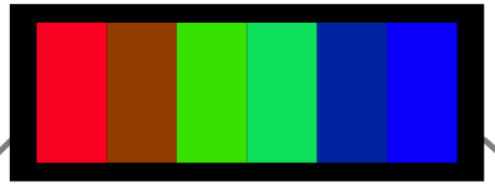
ANGE ALBERTINI
http://pics.corkami.com

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving





File syntax

```
      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
```

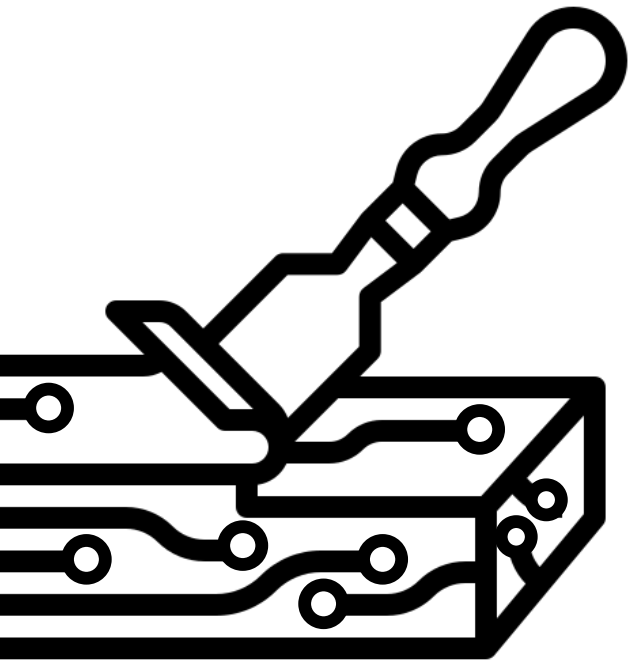
SEGMENTS	FIELDS	VALUES
START OF IMAGE	marker	FFD8
APPLICATION0 (DEFAULT HEADER)	marker/length	FFE0/16
	identifier	JFIF\0
	version	1.1
	units	1 (dpi)
	density	72x72
	thumbnail	0x0
DEFINE QUANTIZATION TABLE	marker/length	FFDB/67
	destination	0 (luminance)
	table (8x8)	{1} (100% quality)
DEFINE QUANTIZATION 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

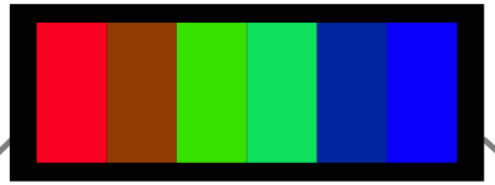
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving

File characterisics





```
     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 00 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
```

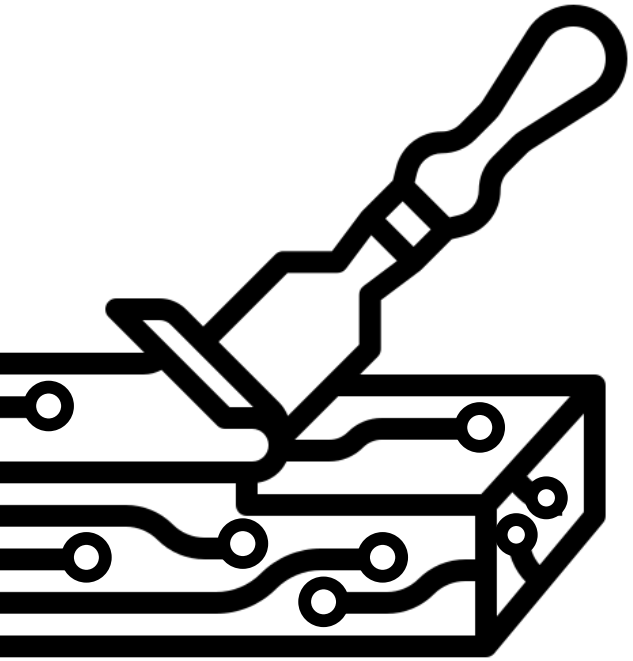
SEGMENTS	FIELDS	VALUES
START OF IMAGE	marker	FFD8
APPLICATION0 (DEFAULT HEADER)	marker/length	FFE0/16
	identifier	JFIF\0
	version	1.1
	units	1 (dpi)
	density	72x72
	thumbnail	0x0
DEFINE QUANTIZATION TABLE	marker/length	FFDB/67
	destination	0 (luminance)
	table (8x8)	{1} (100% quality)
DEFINE QUANTIZATION 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

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving

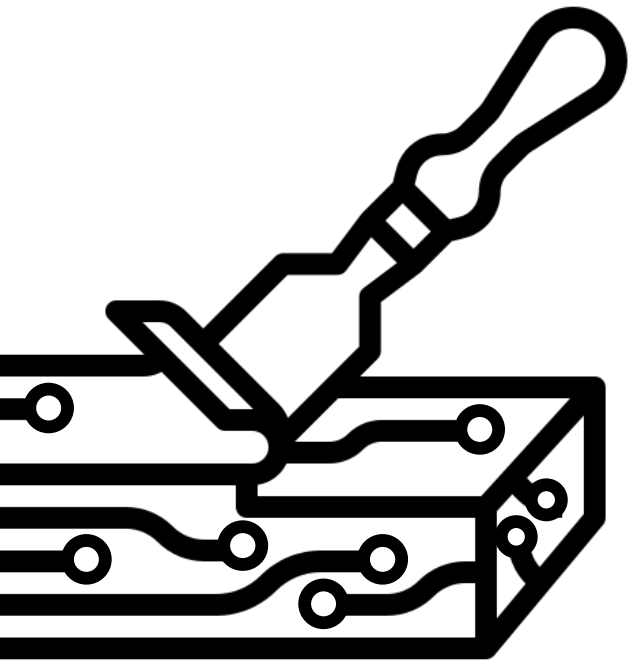


File content/semantics

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving



Internal file structure

Start and end of file markers

File characterisics

File syntax

File content/semantics

All of this can be used to carve files

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

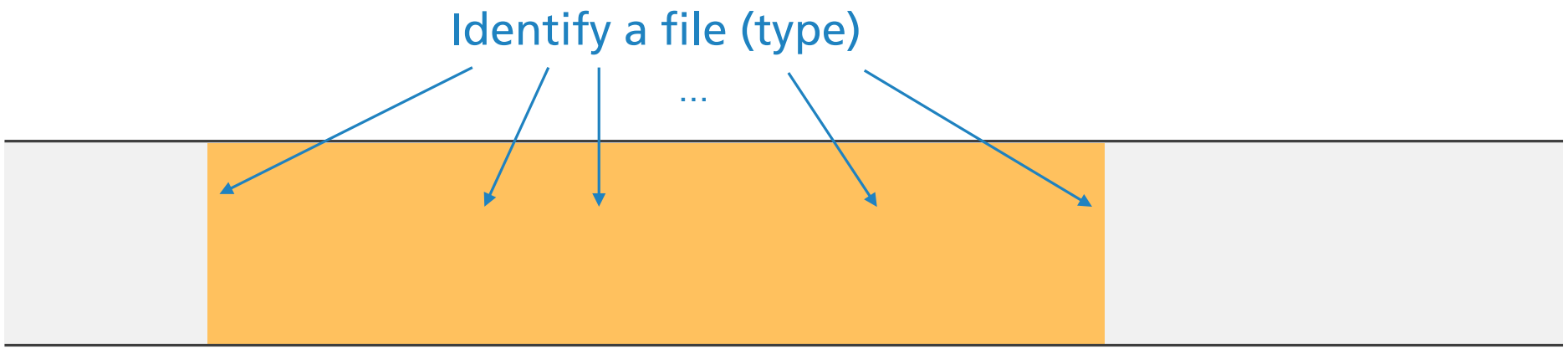
File Carving. Easy as 1…

Image of a storage device

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving. Easy as 1, 2…



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

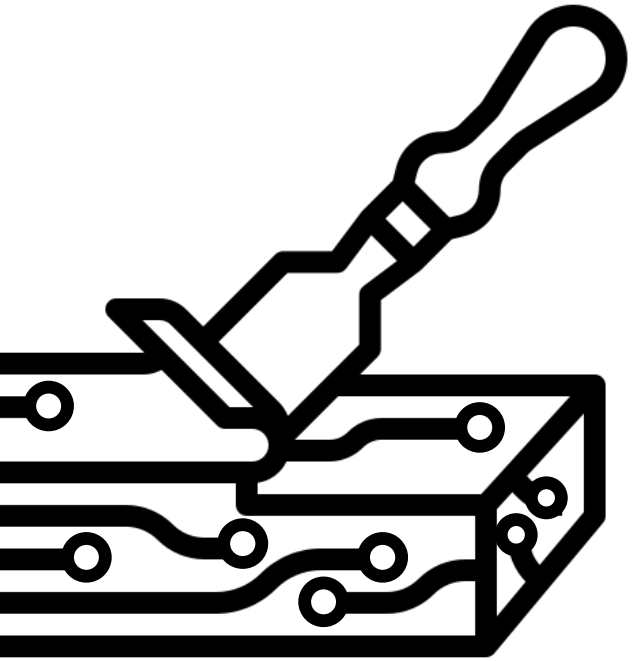
File Carving. Easy as 1, 2, 3.

File in unallocated space

Image of a storage device

Carved file

File Carving



Internal file structure

Start and end of file markers

File characterisics

File syntax

File content/semantics

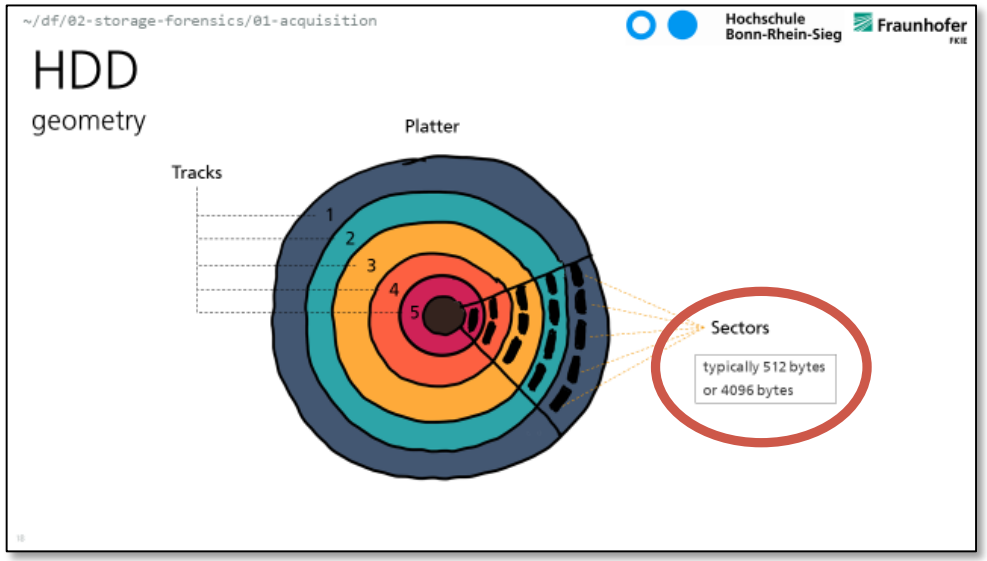
Storage device characteristics

All of this can be used to carve files

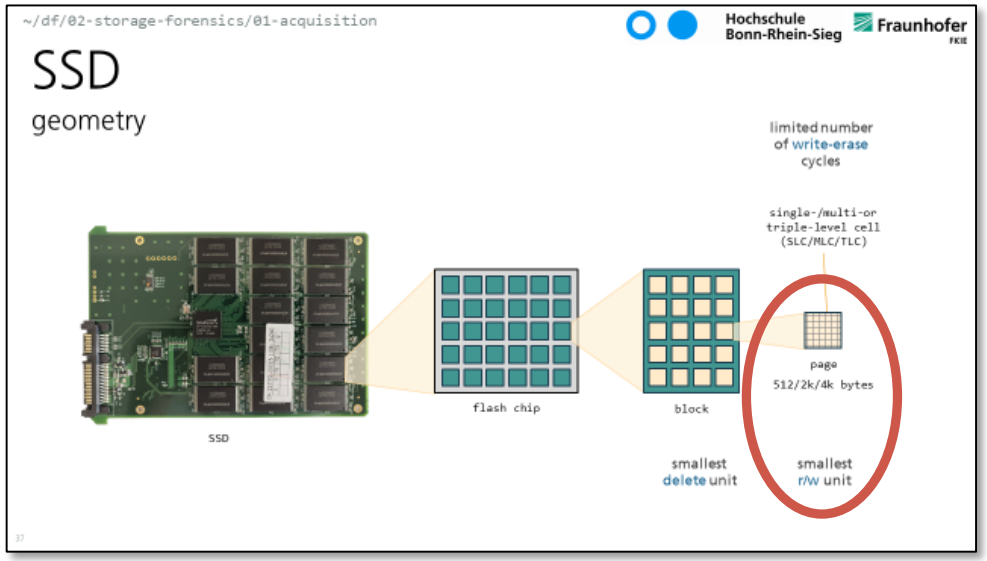
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving

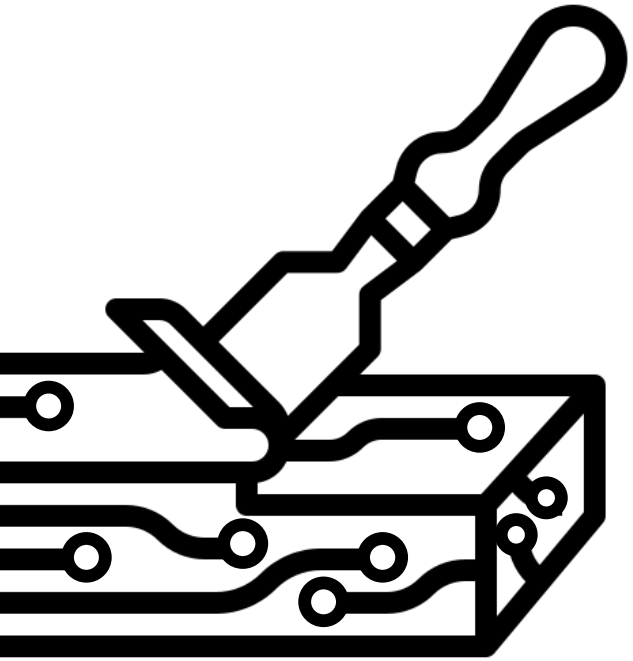


Typical HDD sector sizes:
512 or 4k bytes



Typical SSD page sizes:
512, 2k, or 4k bytes

File Carving



Internal file structure

Start and end of file markers

File characterisics

File syntax

File content/semantics

Storage device characteristics

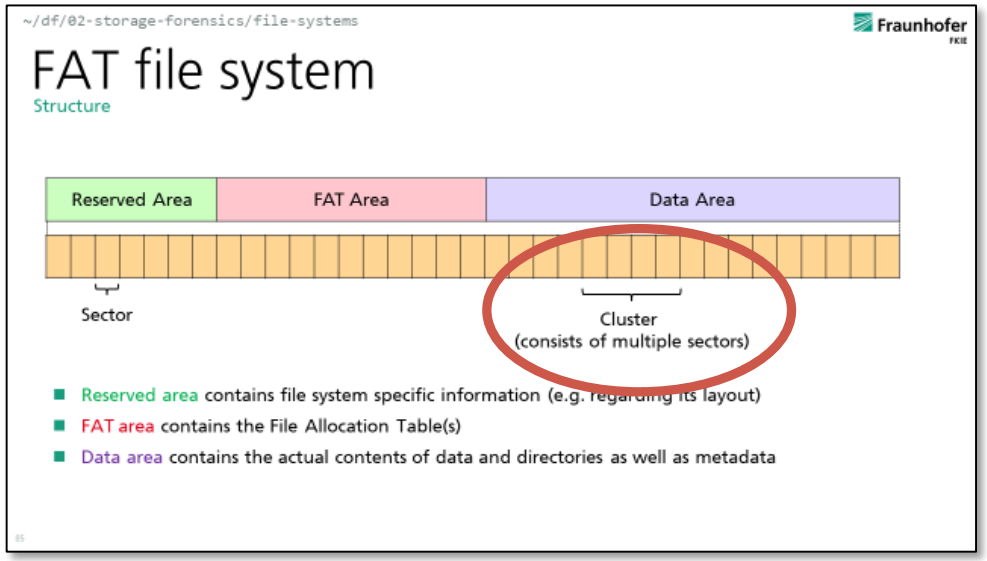
File system characteristics

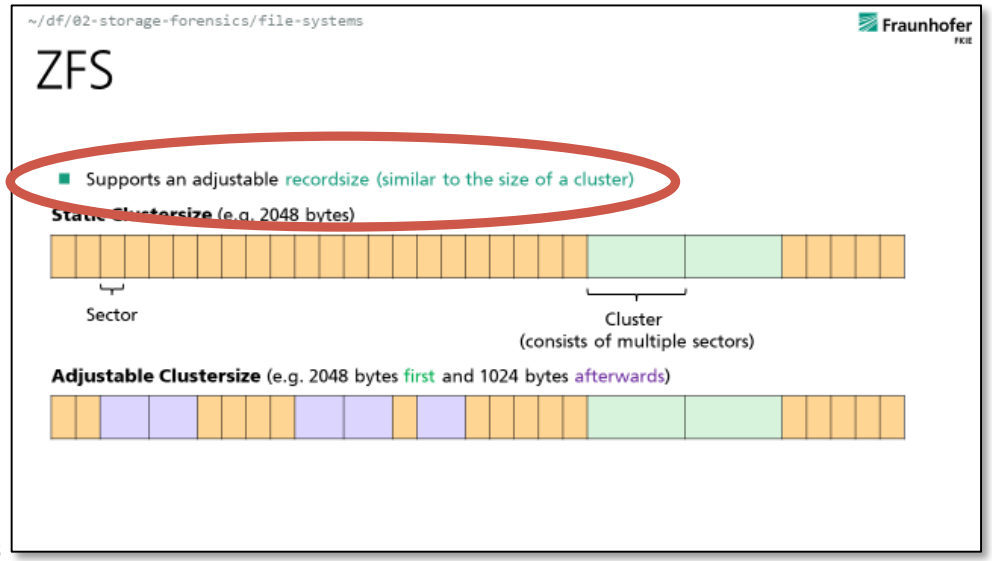
All of this can be used to carve files

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving





File system have fixed cluster sizes.

But sometimes they don't.

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving. Easy as 1…

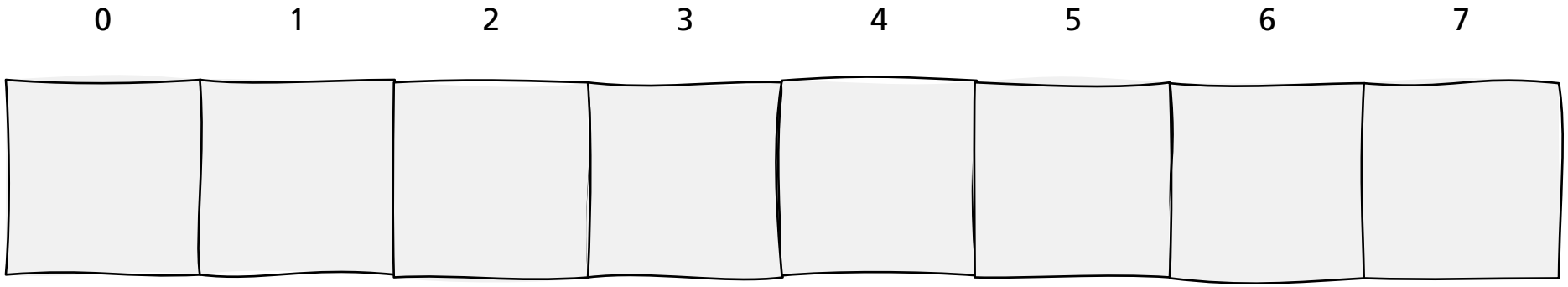
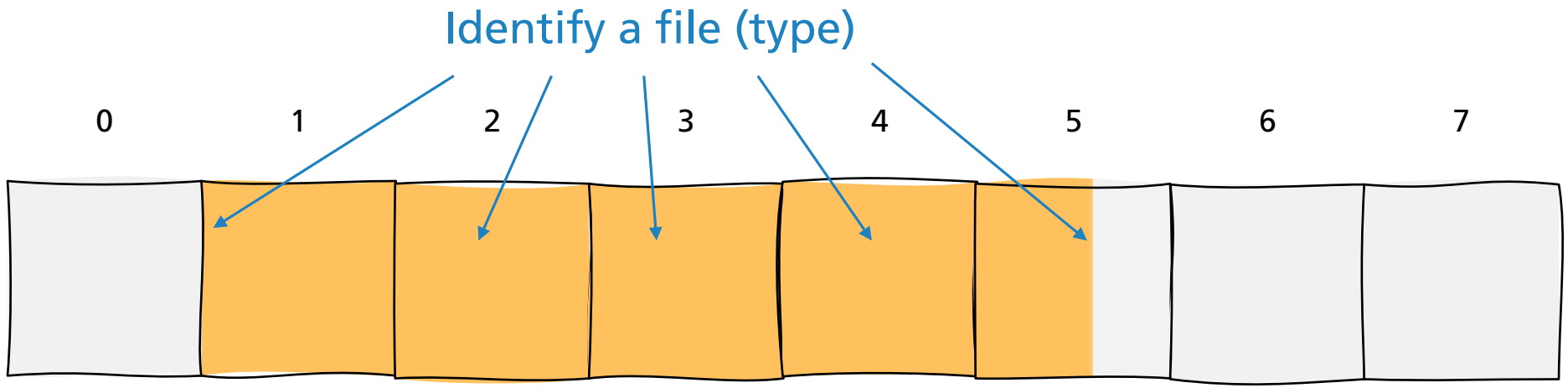


Image of a storage device

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving. Easy as 1, 2…



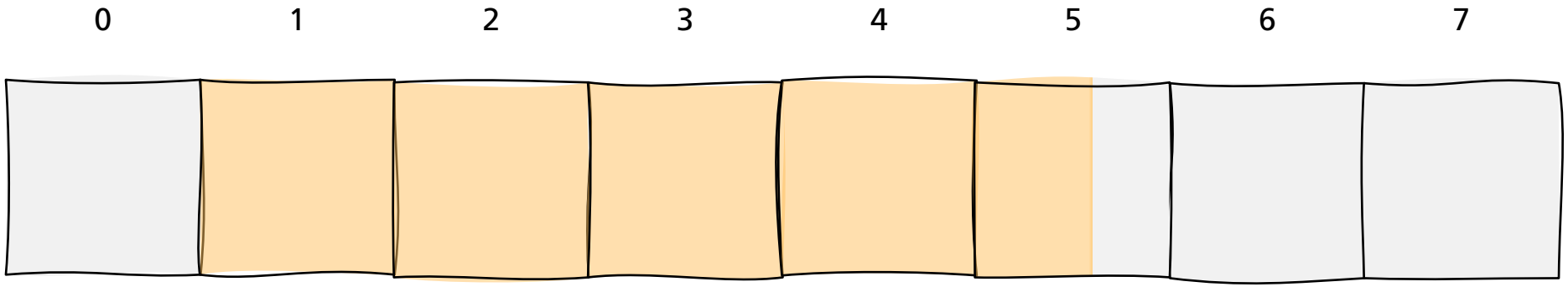
File in unallocated space

Image of a storage device

Hochschule Bonn-Rhein-Sieg

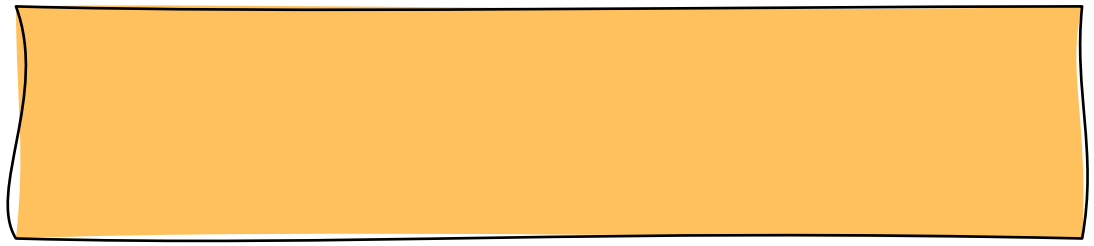
Fraunhofer FKIE

File Carving. Easy as 1, 2, 3.



File in unallocated space

Image of a storage device



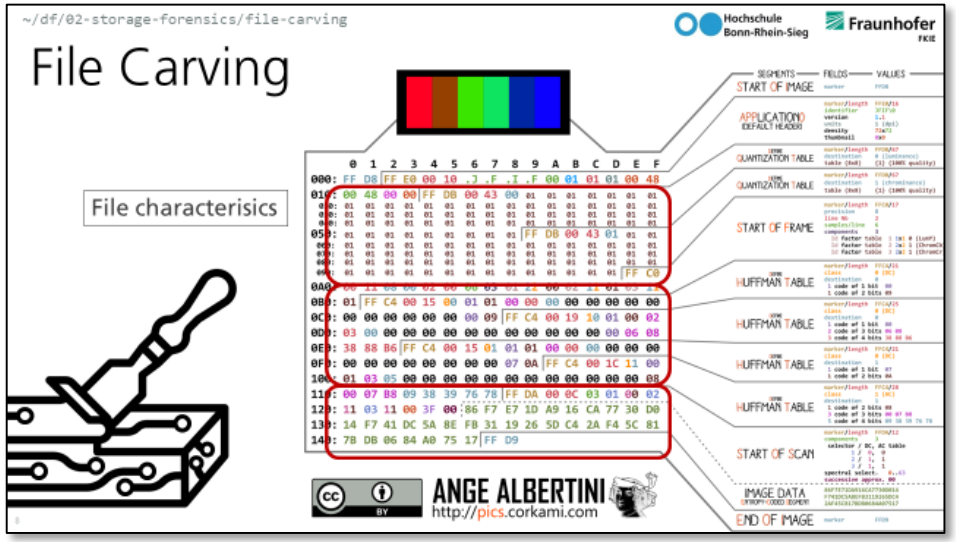
Carved file

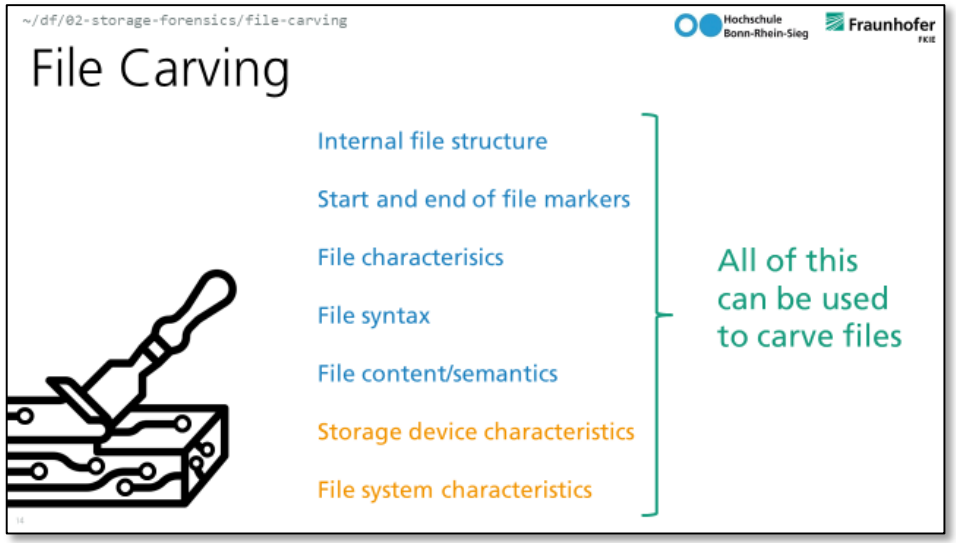
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving

If it's that easy, why do we need this?

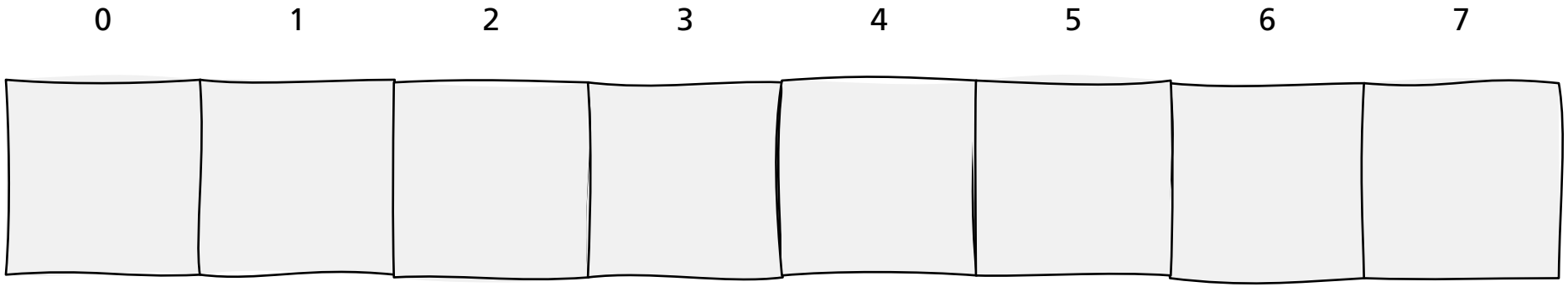




Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

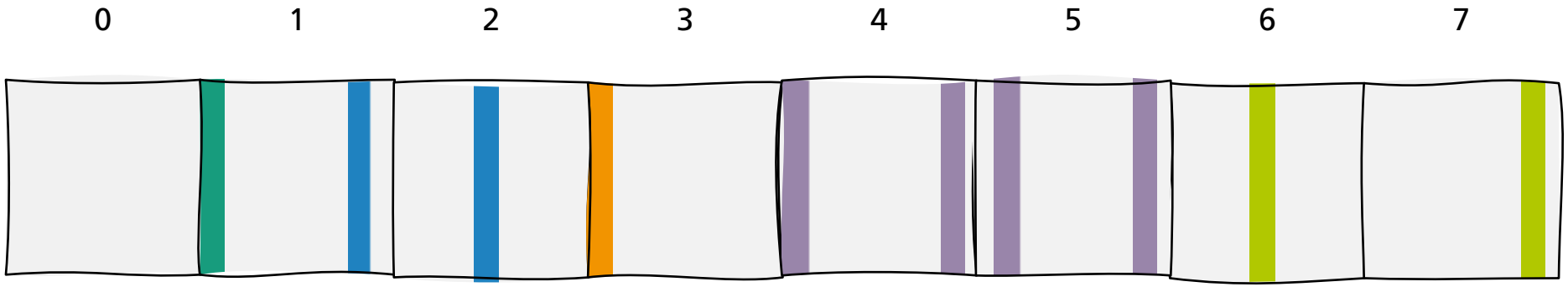
Searching for Signatures



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Searching for Signatures



Signatures can be:

headers

internal

footers

Headers typically start at the beginning of a block.

But not always.

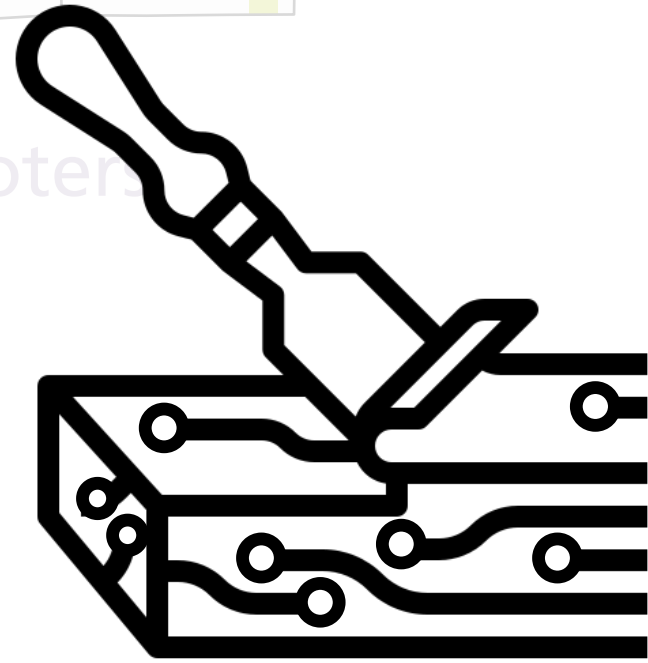
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Searching for Signatures

Now, that we have our signatures:

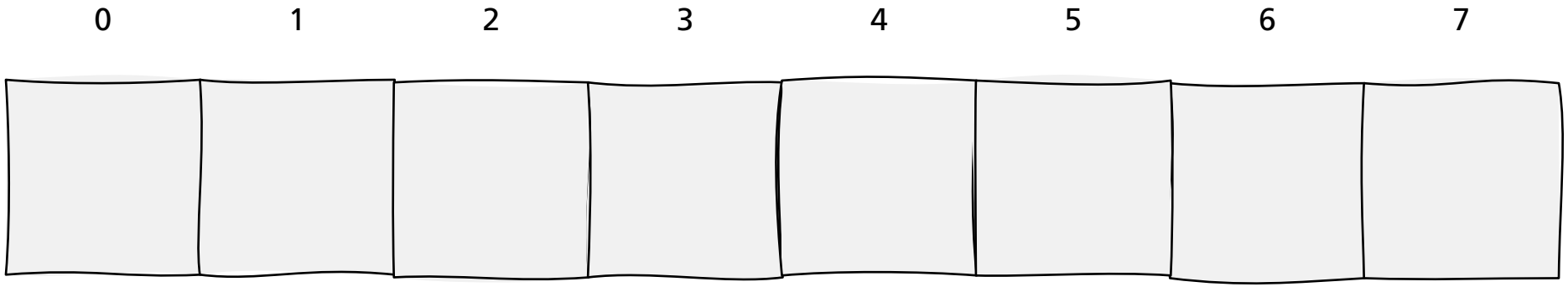
Let's carve!



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

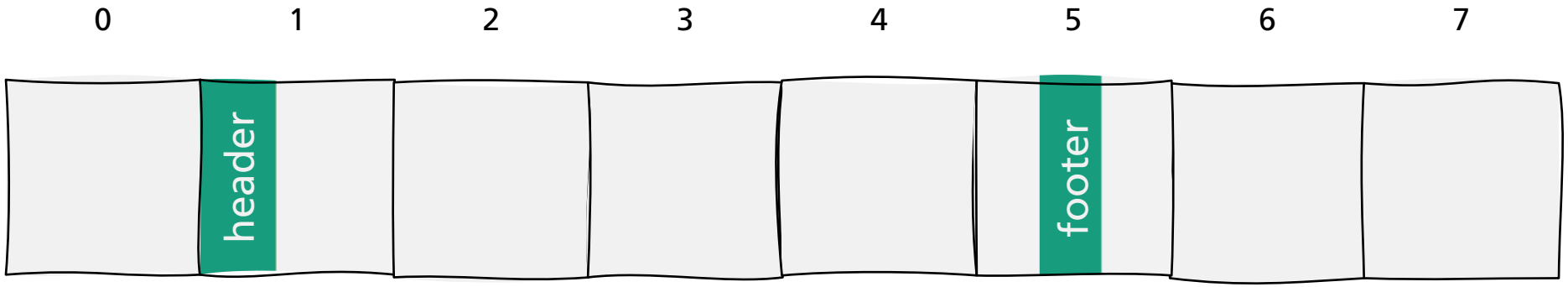
Header-to-Footer Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

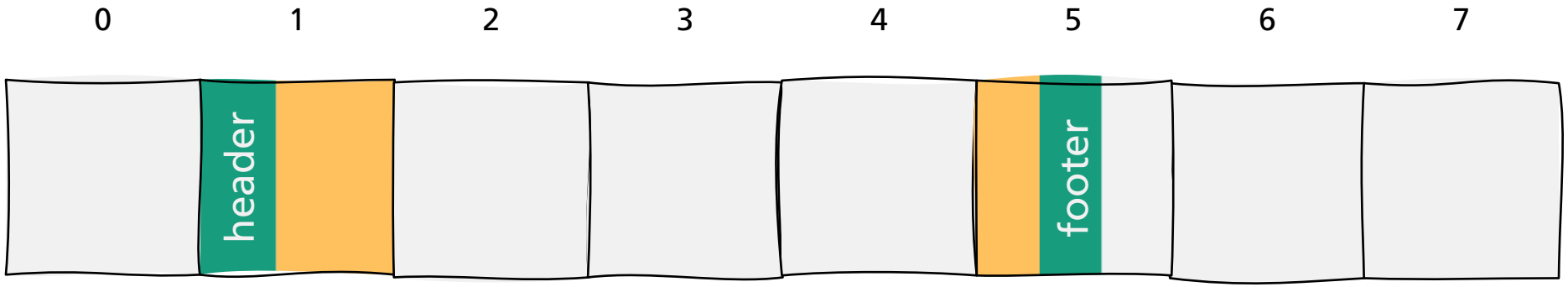
Header-to-Footer Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

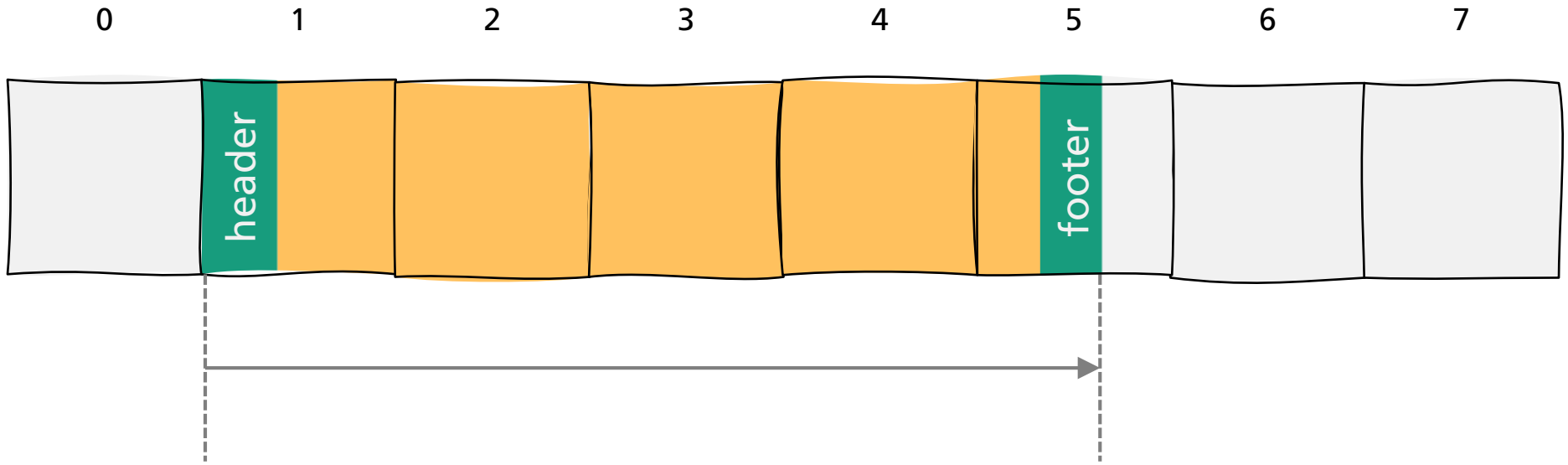
Header-to-Footer Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

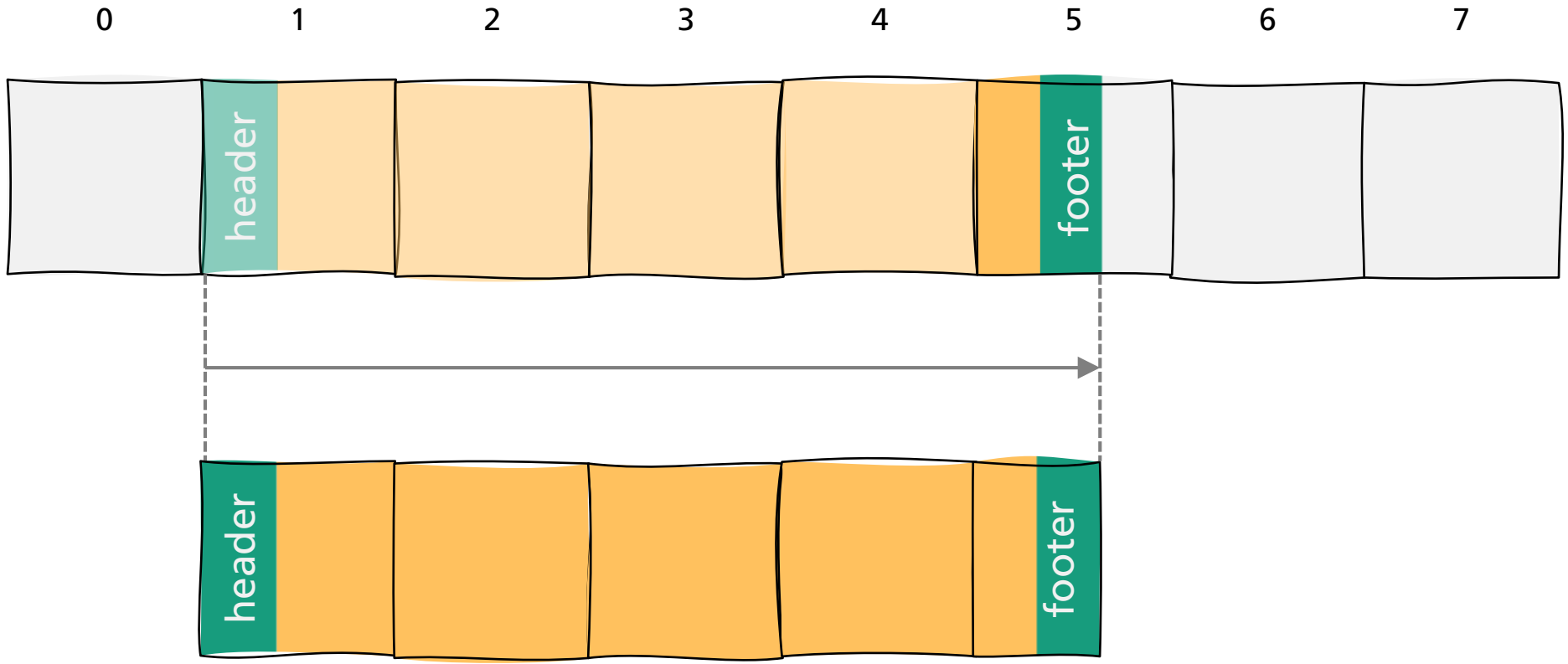
Header-to-Footer Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

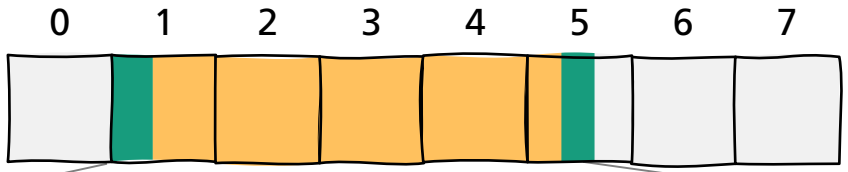
Header-to-Footer Carving



Hochschule Bonn-Rhein-Sieg

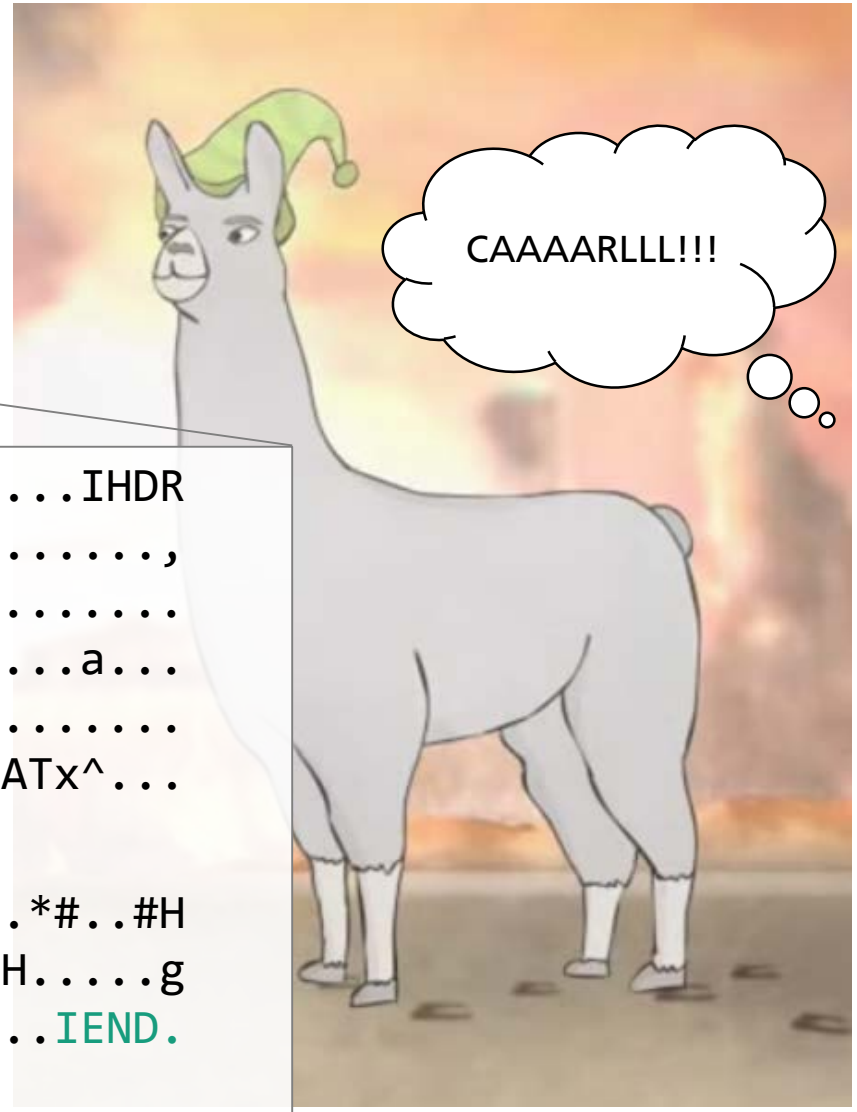
Fraunhofer FKIE

Header-to-Footer Carving



carl.png

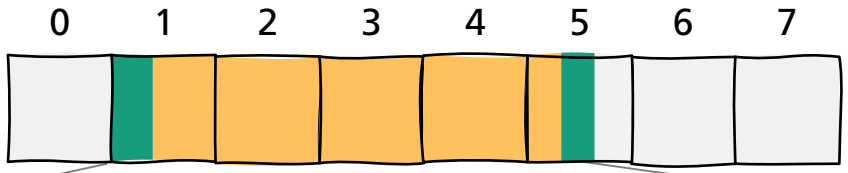
```
b0000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452  .PNG........IHDR
b0000010: 0000 01c6 0000 0256 0806 0000 00c2 ec27  .......V.......,
b0000020: f700 0000 0173 5247 4200 aece 1ce9 0000  .....sRGB.......
b0000030: 0004 6741 4d41 0000 b18f 0bfc 6105 0000  ..gAMA......a...
b0000040: 0009 7048 5973 0000 0ec2 0000 0ec2 0115  ..pHYs..........
b0000050: 284a 8000 00ff a549 4441 5478 5ec4 fde7  (J.....IDATx^...
…
b003fc40: 876d 0928 331e ba5b 00ca 2a23 d6c5 2348  .m.(3..[..*#..#H
b003fc50: c444 bcf8 fcc0 afe9 2148 ffe5 97ff 0367  .D......!H.....g
b003fc60: 21e2 a34c 530f 5500 0000 0049 454e 44ae  !..LS.U....IEND.
b003fc70: 4260 82                                  B`.
```



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Header-to-Footer Carving



carl.jpg

```
b0000000: ffd8 ffe0 0010 4a46 4946 0001 0102 0025  ......JFIF.....%
b0000010: 0025 0000 ffe1 18a6 4578 6966 0000 4949  .%......Exif..II
b0000020: 2a00 0800 0000 0600 1a01 0500 0100 0000  *...............
b0000030: 5600 0000 1b01 0500 0100 0000 5e00 0000  V...........^...
b0000040: 2801 0300 0100 0000 0300 0000 3101 0200  (...........1...
b0000050: 0d00 0000 6600 0000 3201 0200 1400 0000  ....f...2.......
…
b0023fb0: 8116 e66f fefb fc5a 97f7 d613 f183 6a6d  ...o...Z......jm
b0023fc0: 4747 cb95 11b4 f7d3 d120 f25e 56a9 b95b  GG....... .^V..[
b0023fd0: 7bfb be32 3945 24d6 e6bf 2ff9 8aa0 3d66  {..29E$.../...=f
b0023fe0: 0ff7 1048 1710 91c4 f11e 99ff d9         ...H.........
```

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

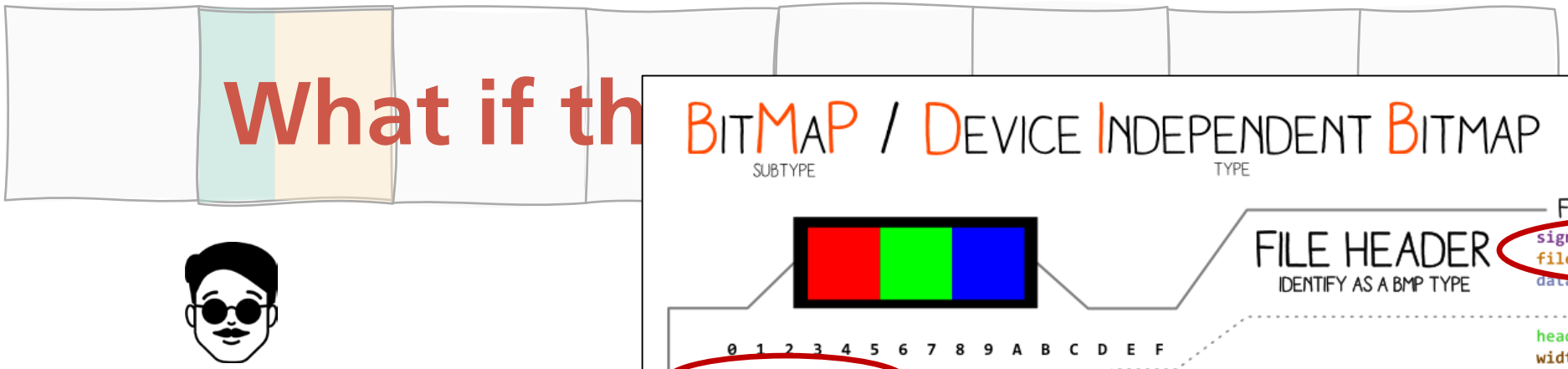
Header-to-Footer Carving

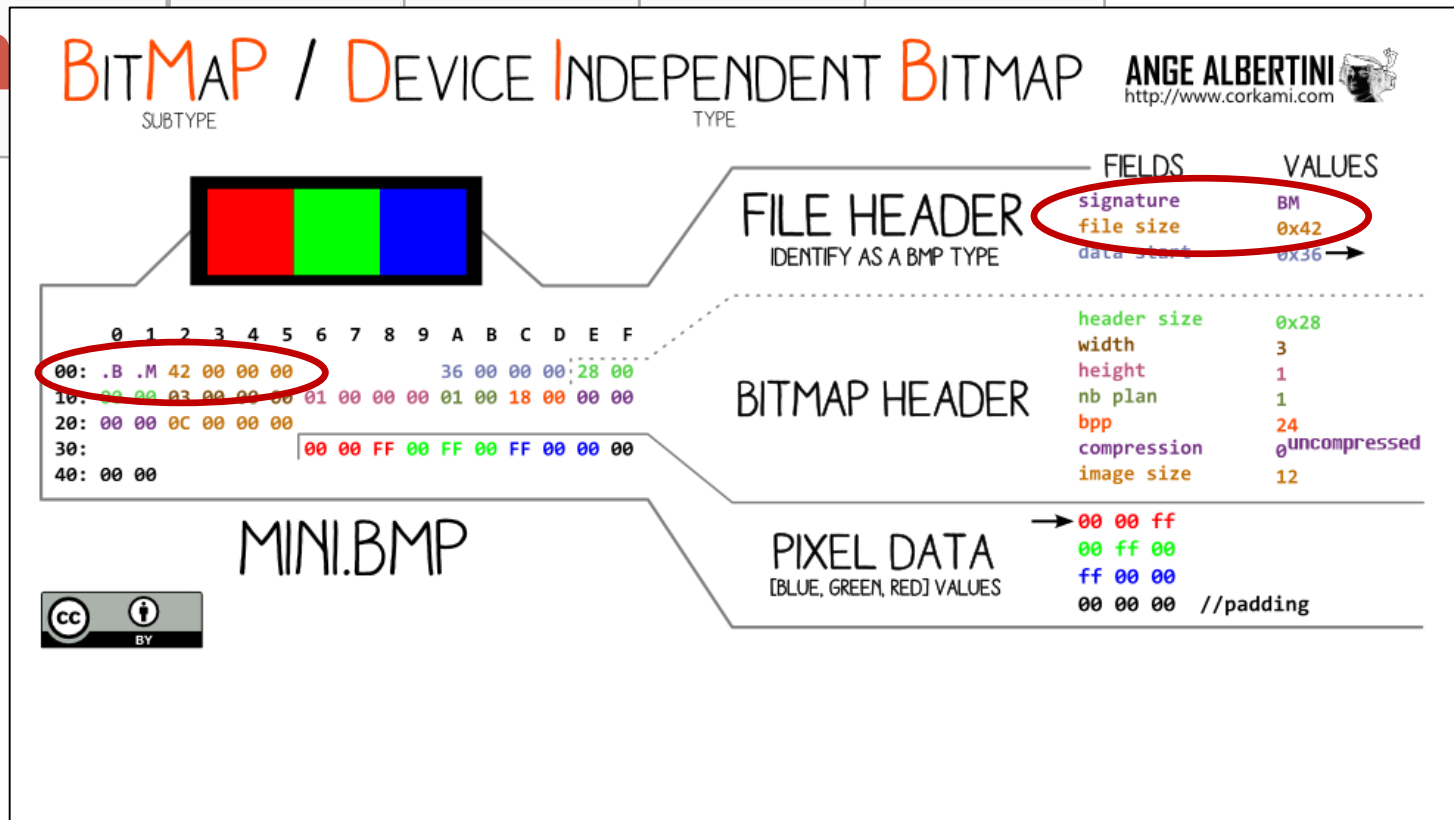
That is, what if the file format does not have a footer?

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Header-Embedded-Length Carving

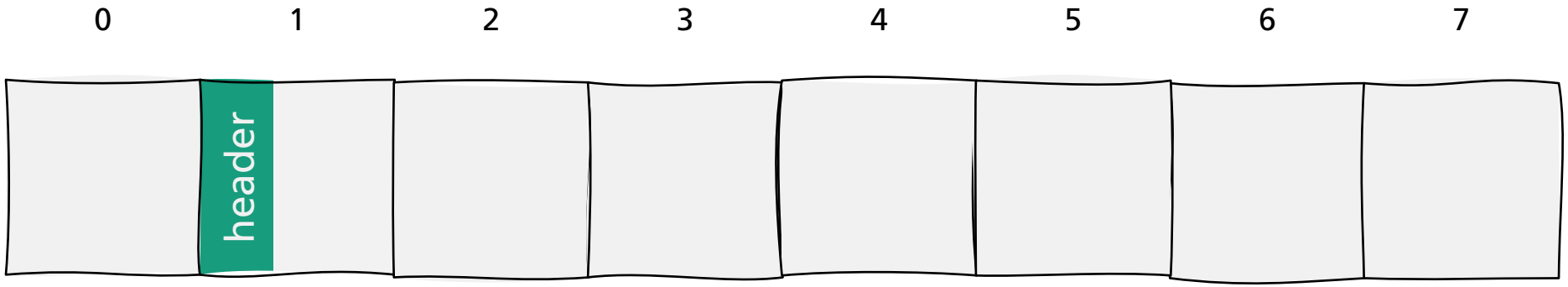




Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

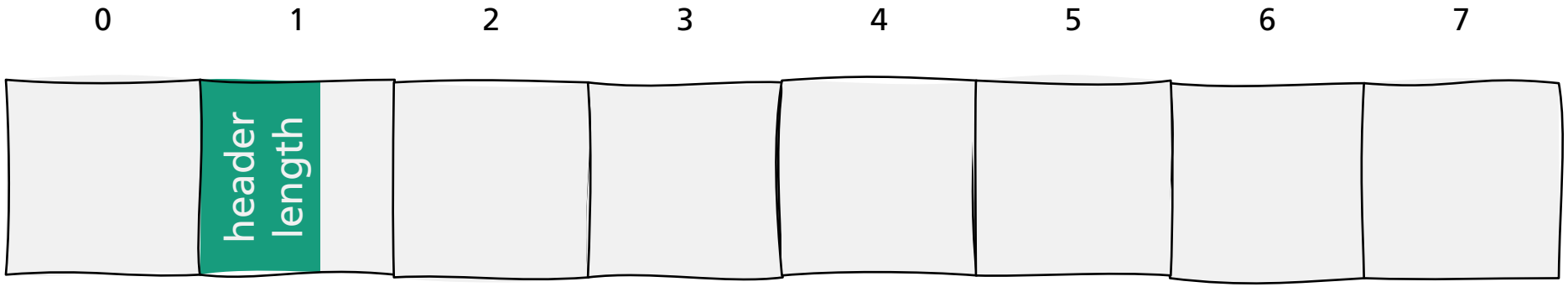
Header-Embedded-Length Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

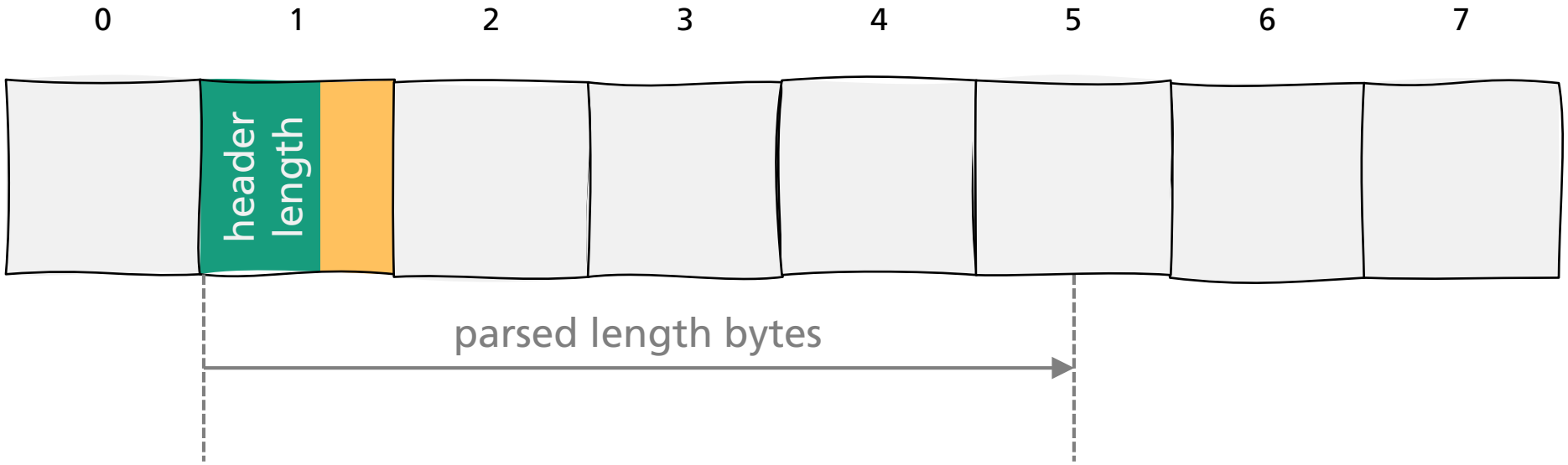
Header-Embedded-Length Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

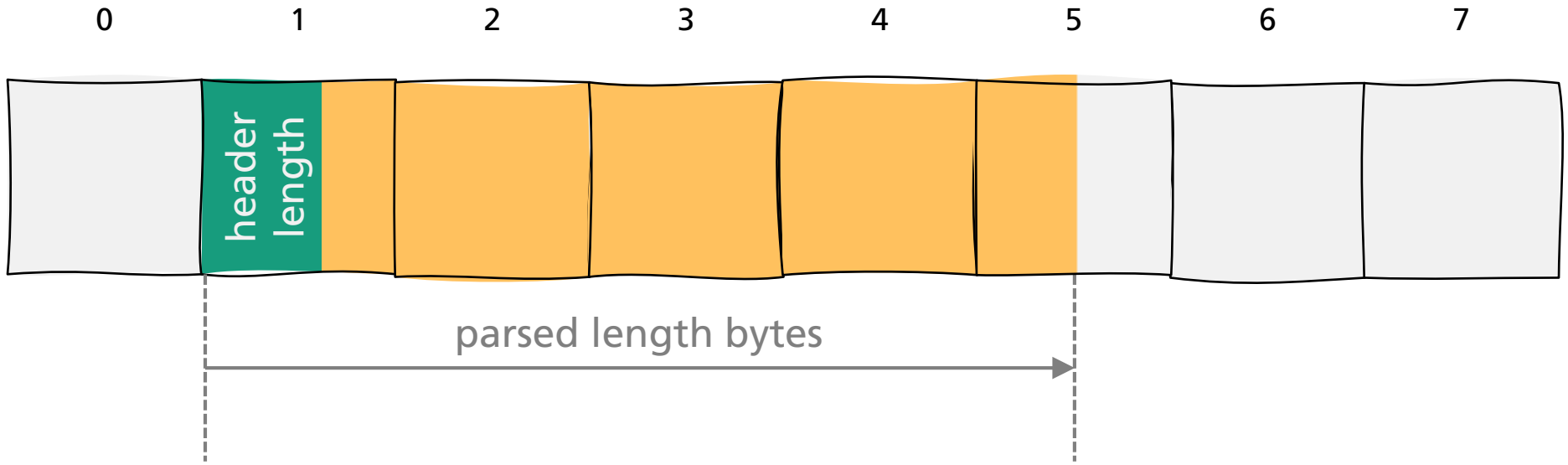
Header-Embedded-Length Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

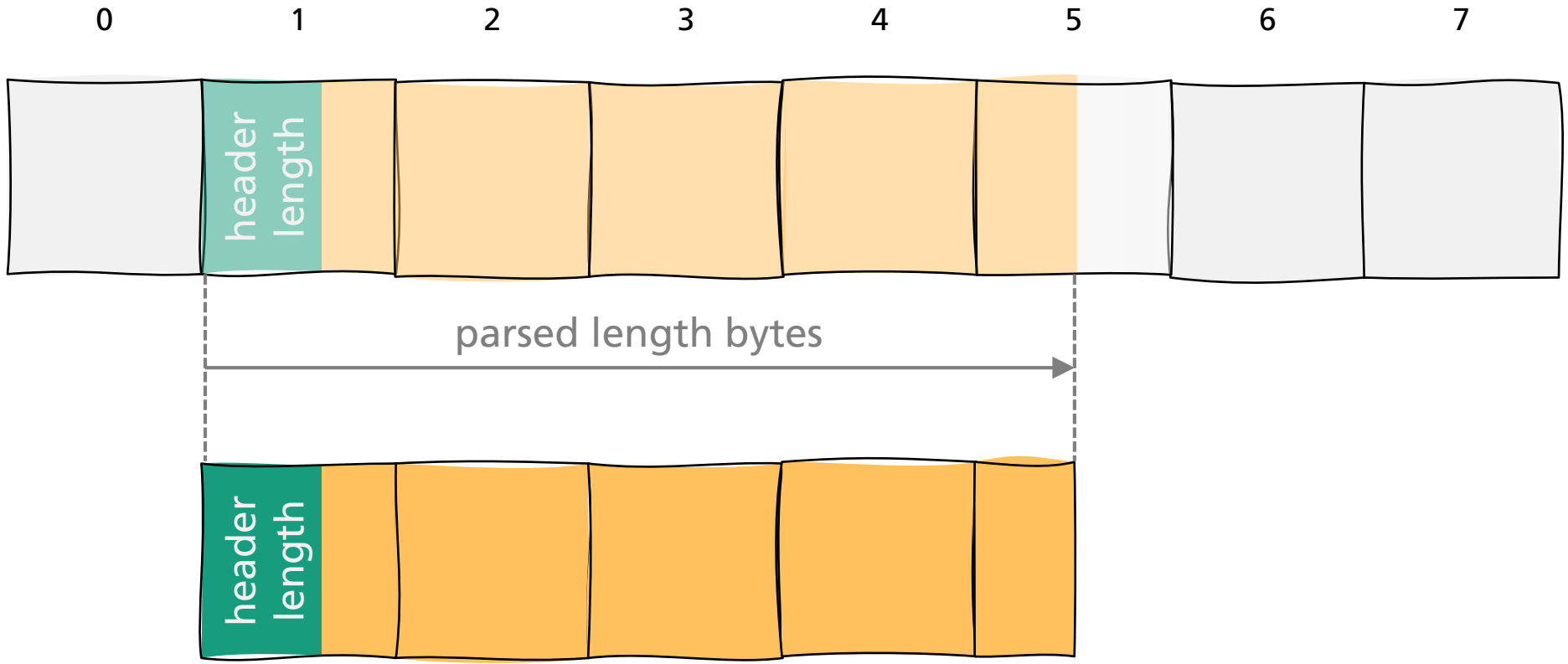
Header-Embedded-Length Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

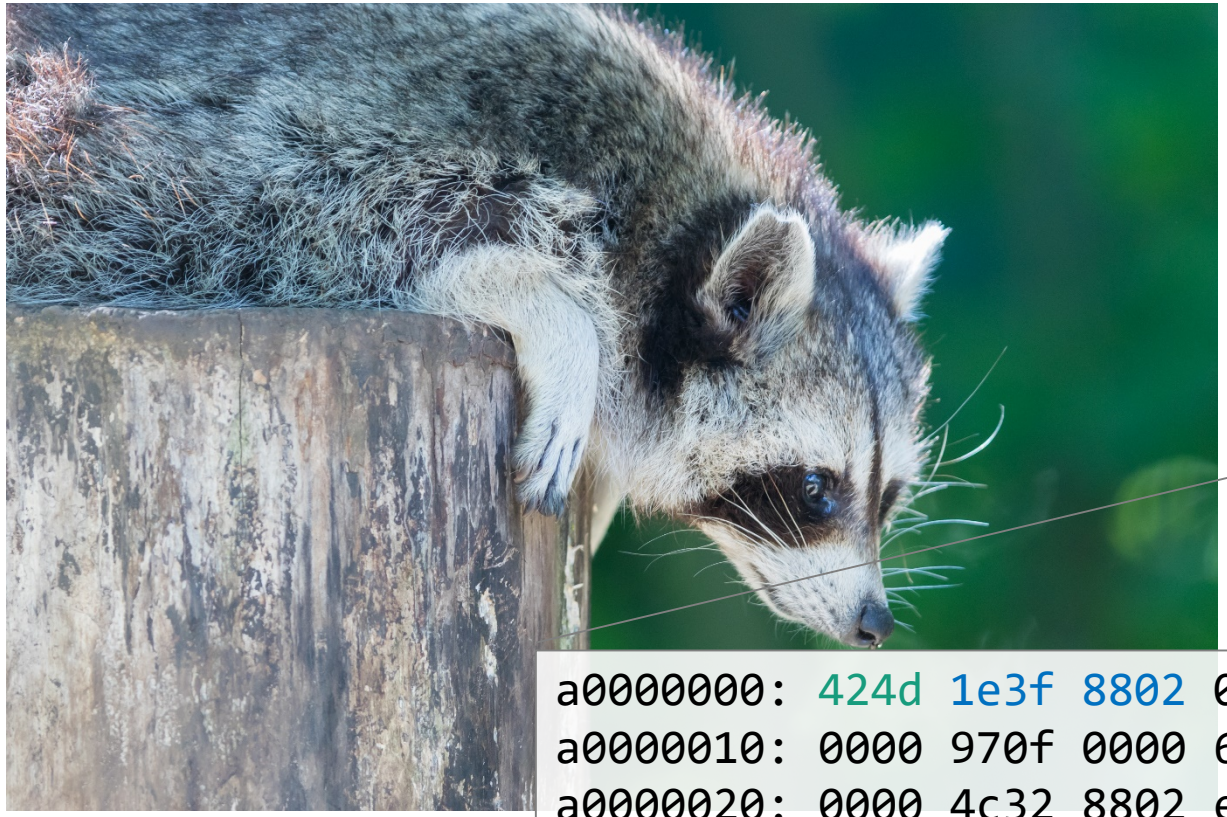
Header-Embedded-Length Carving

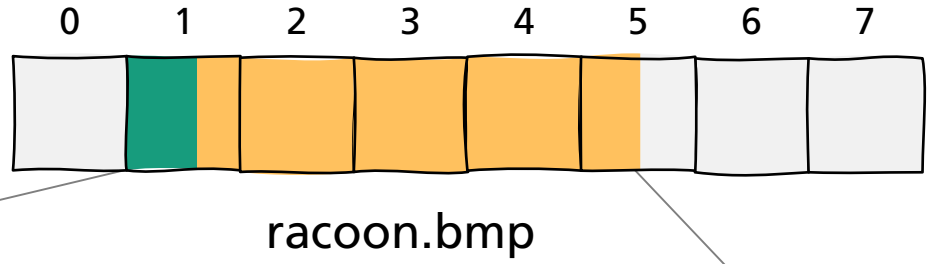


Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Header-Embedded-Length Carving





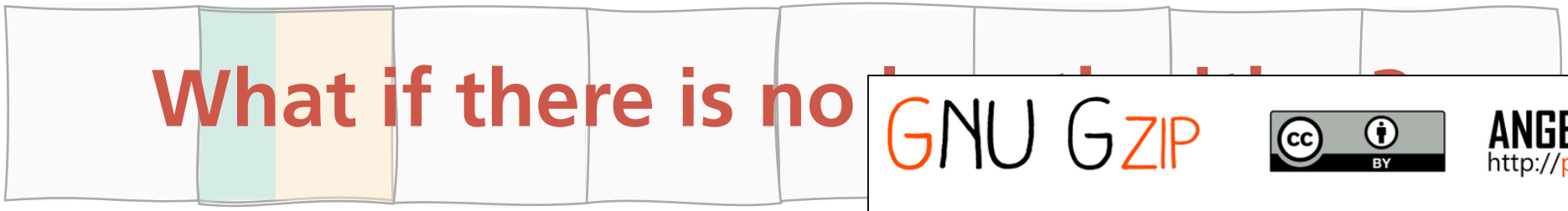

```
a0000000: 424d 1e3f 8802 0000 0000 8a00 0000 7c00  BM.?..........|.
a0000010: 0000 970f 0000 650a 0000 0100 2000 0300  ......e..... ...
a0000020: 0000 4c32 8802 e824 0000 e824 0000 0000  ..L2...$...$....
...
a2883f00: f819 f8a8 f938 f9c7 fa57 fae7 fb77 fc07  .....8...W...w..
a2883f10: fc98 fd29 fdba fe4b fedc ff6d ffff       ...)...K...m..
```

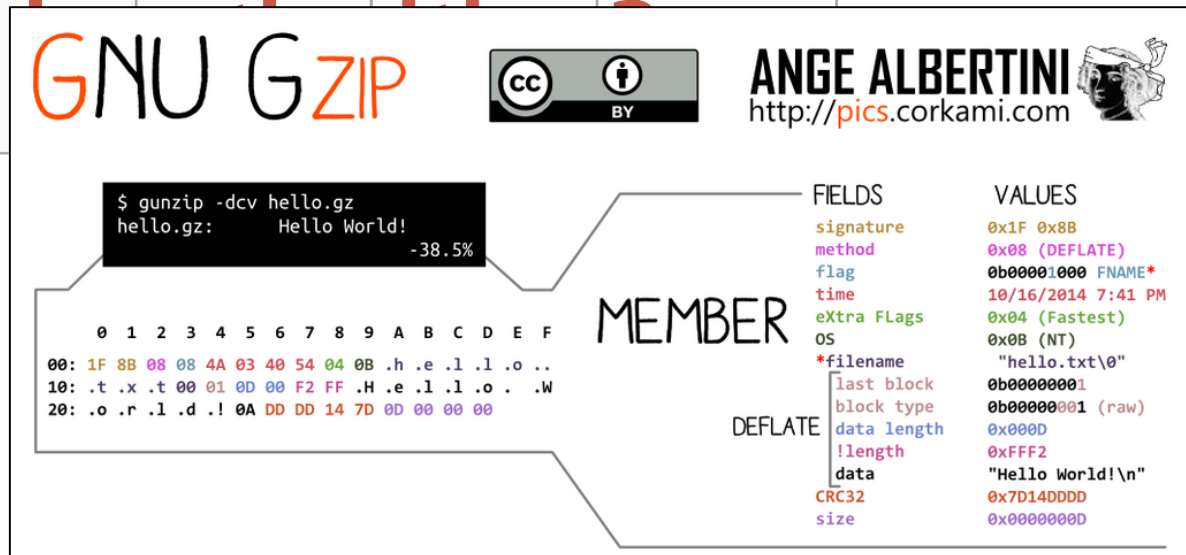
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Header-Embedded-Length Carving

Header-Maximum-Size Carving



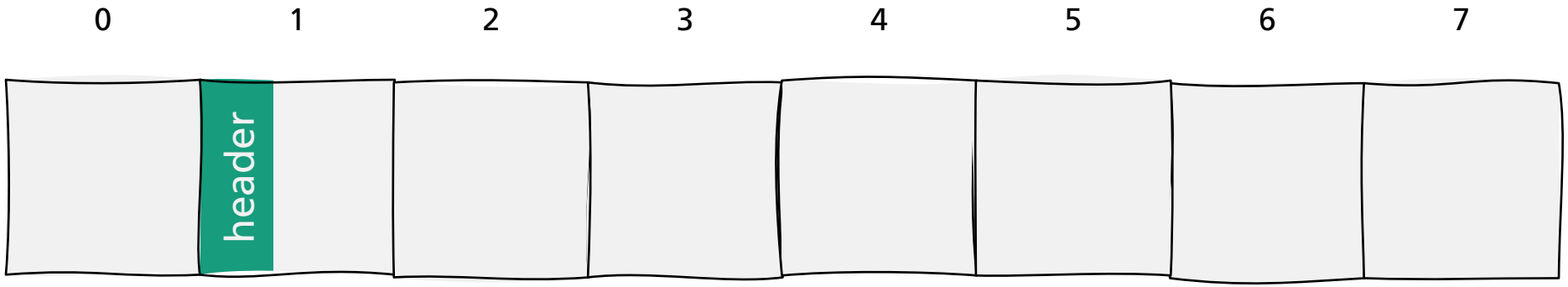



```
$ gunzip carved-01.gz
gzip: carved-01.gz: decompression OK, trailing garbage ignored
```

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

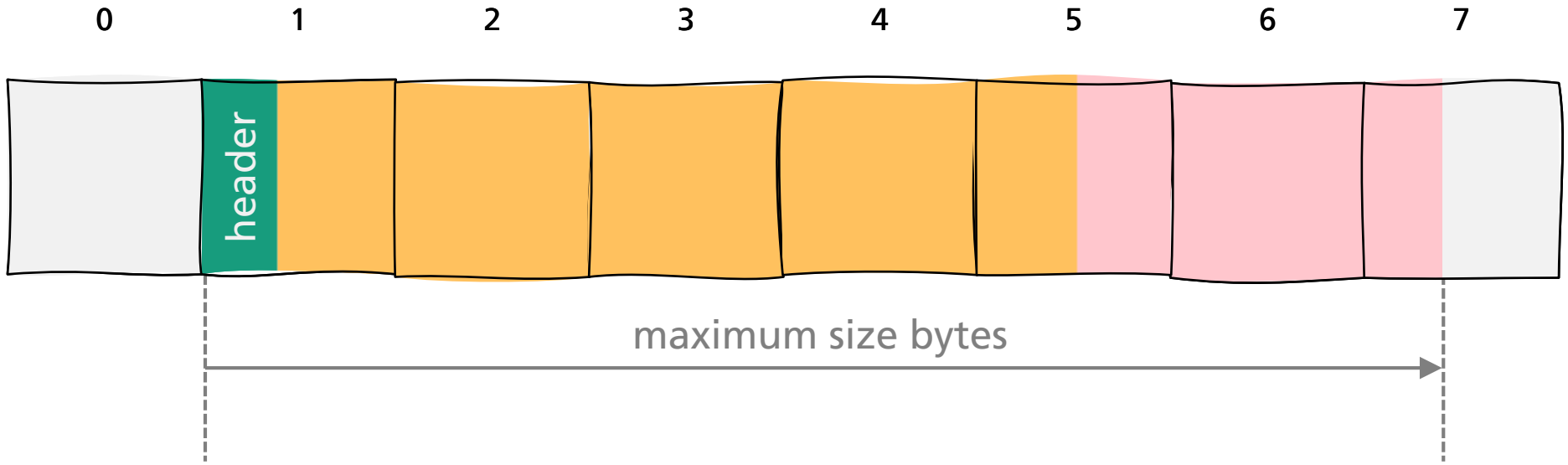
Header-Maximum-Size Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

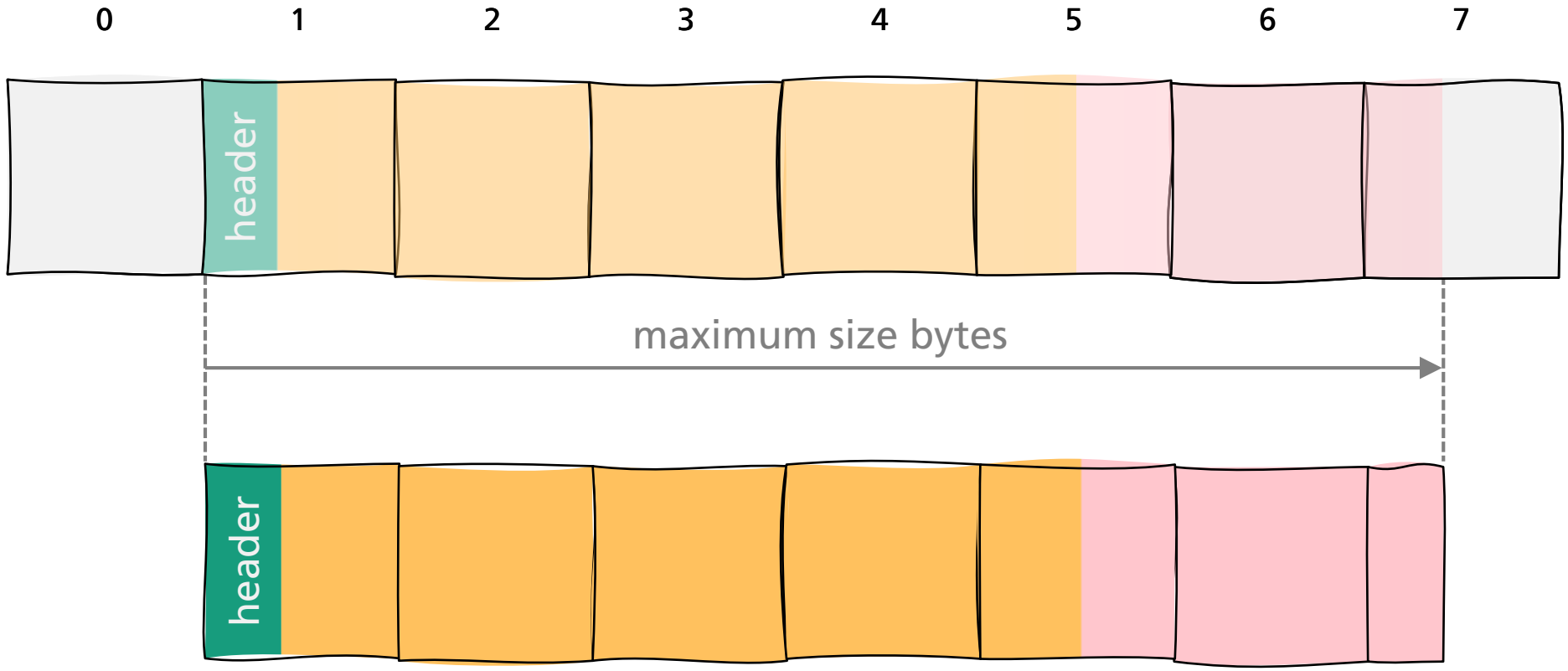
Header-Maximum-Size Carving



Hochschule Bonn-Rhein-Sieg

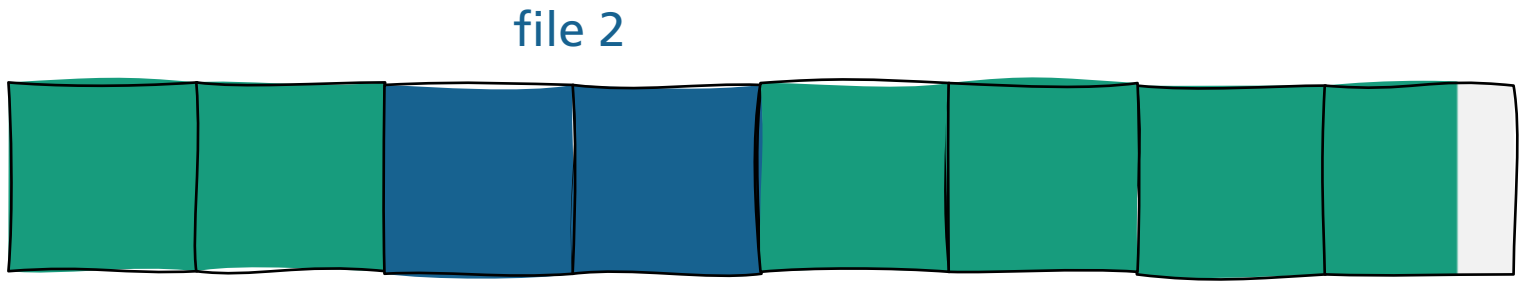
Fraunhofer FKIE

Header-Maximum-Size Carving



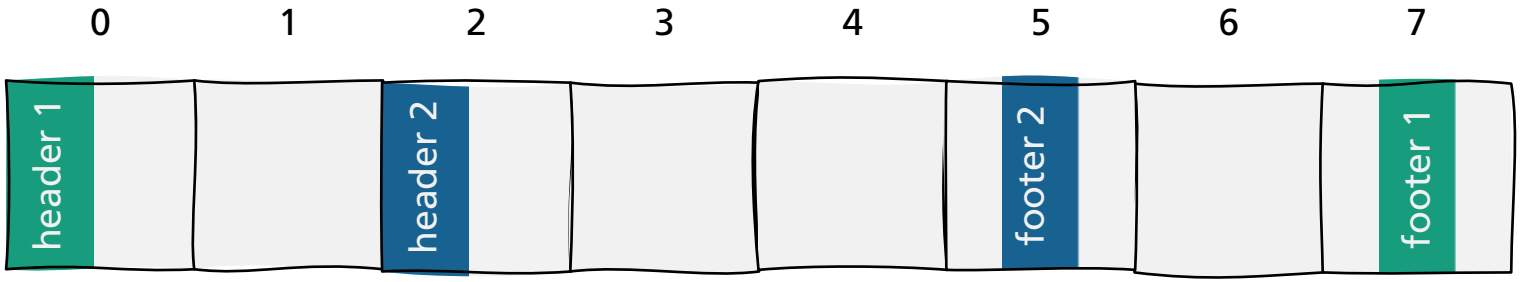
File Carving

It could be all so simple…



…but then there's fragmentation.

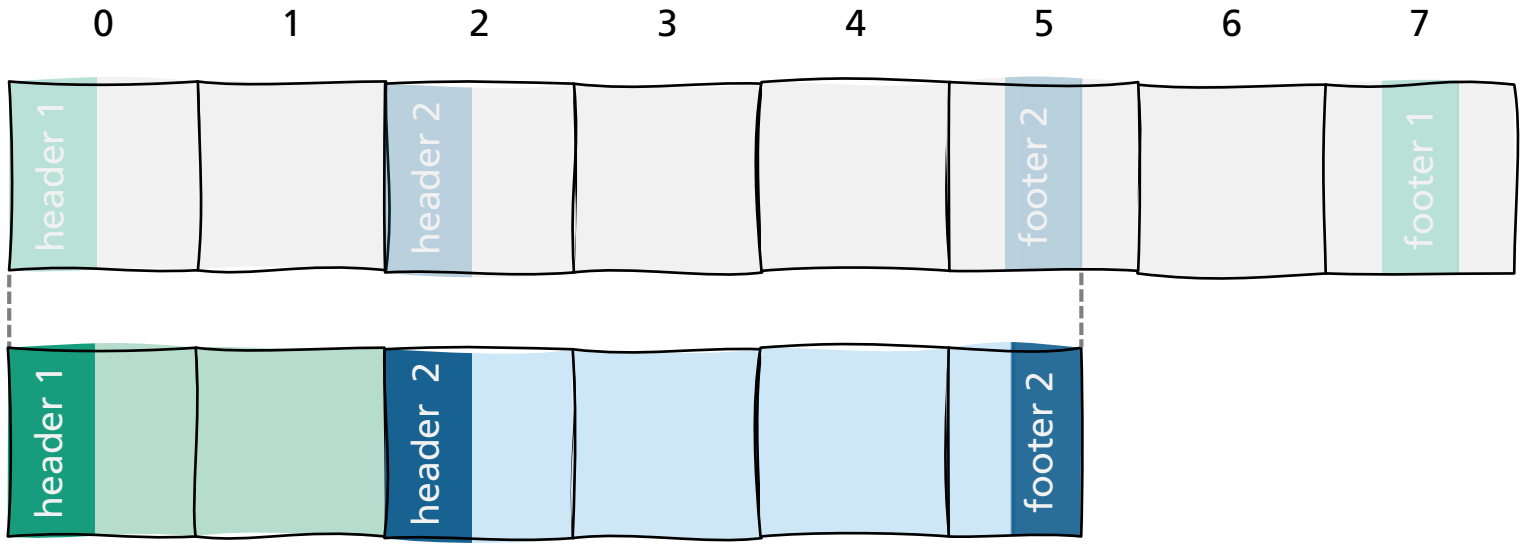
Fragmentation



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

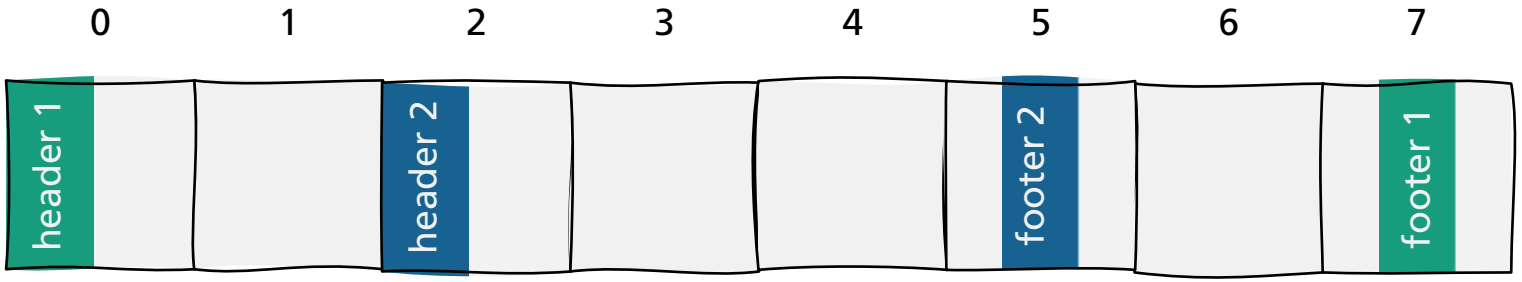
Fragmentation



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

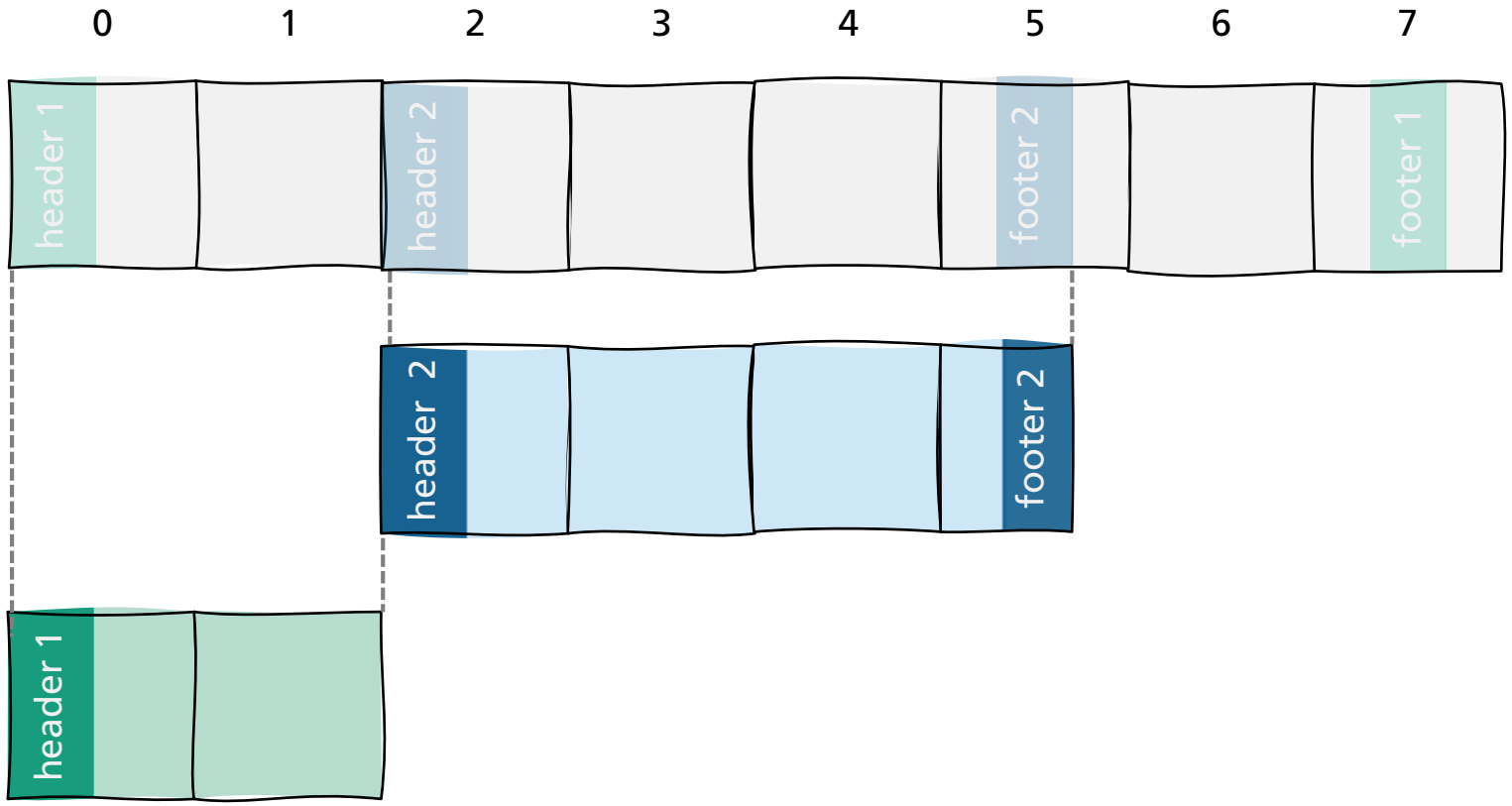
Fragmentation



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

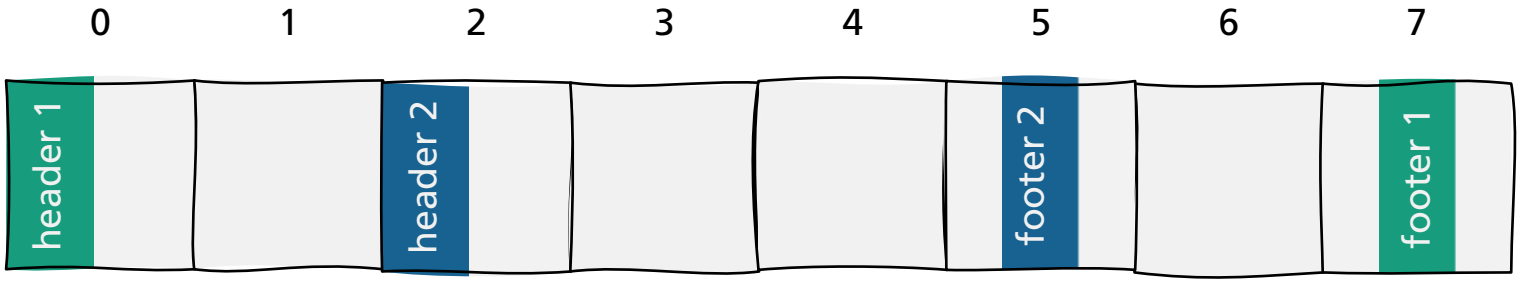
Fragmentation



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

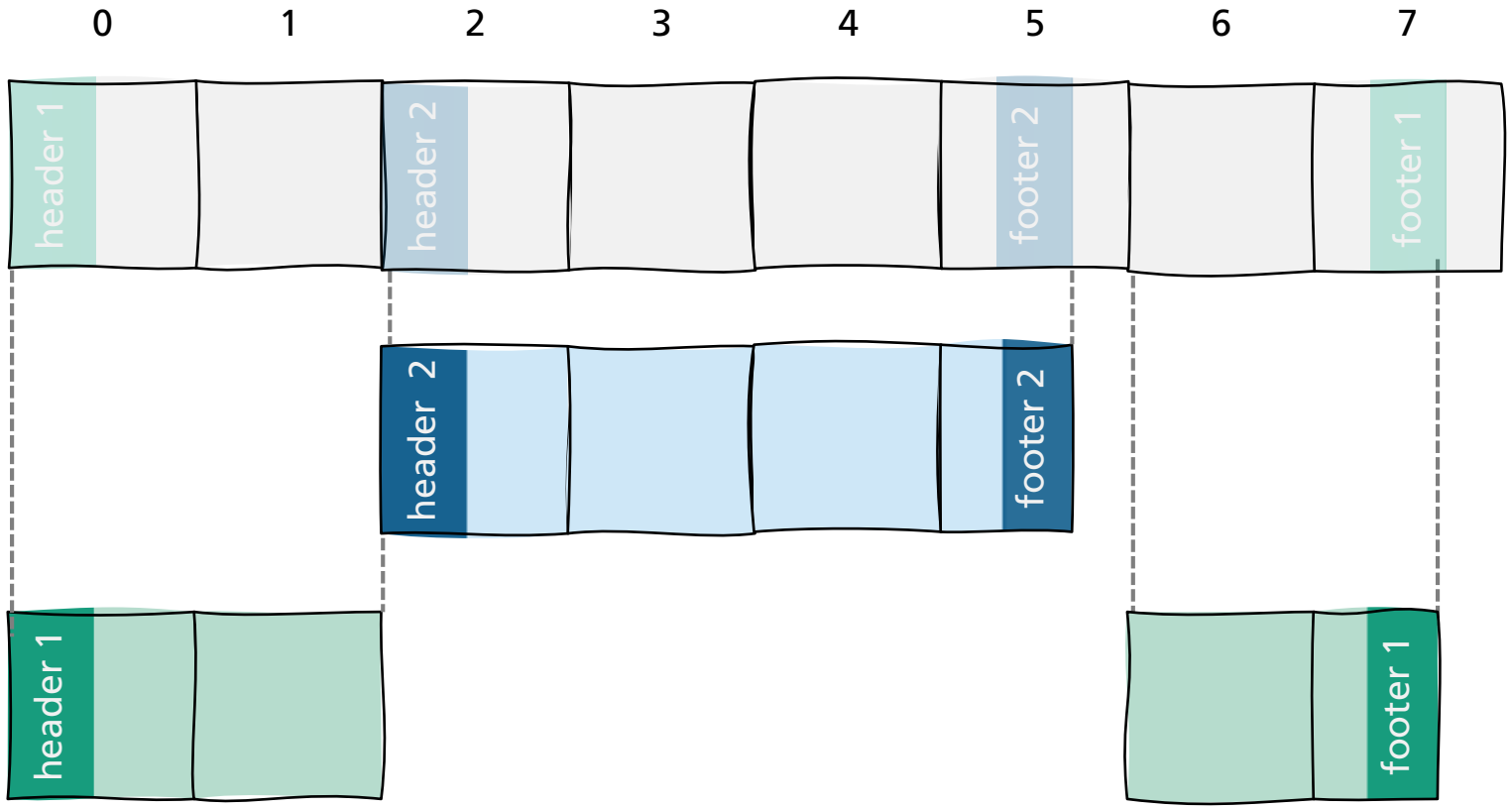
Fragmentation



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

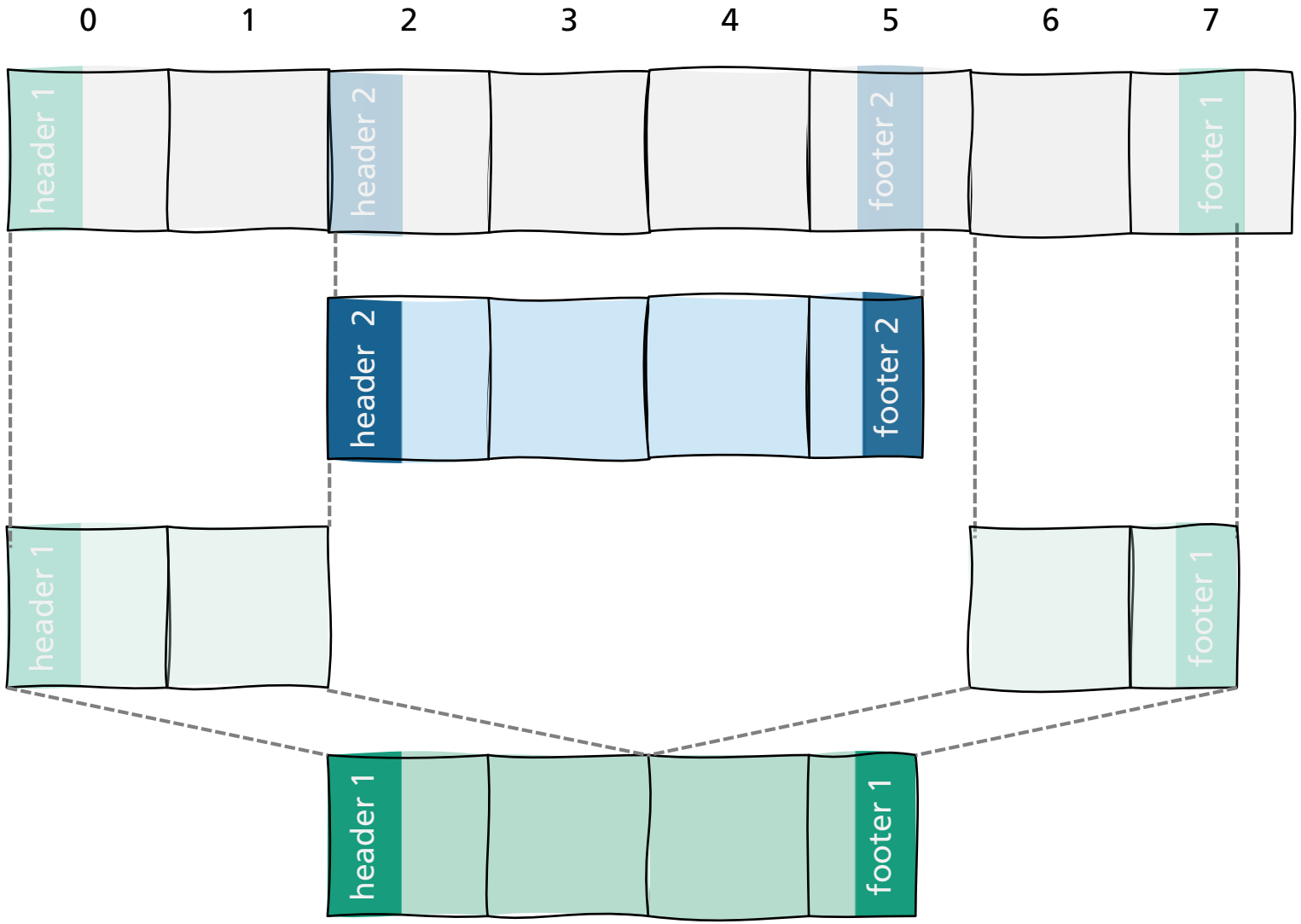
Fragmentation



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

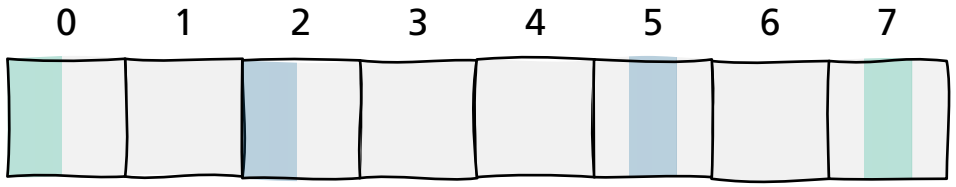
Fragmentation



Hochschule Bonn-Rhein-Sieg

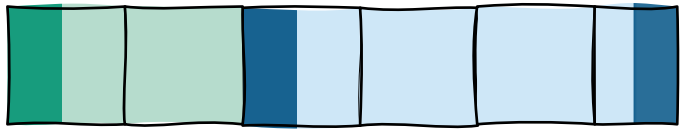
Fraunhofer FKIE

Fragmentation



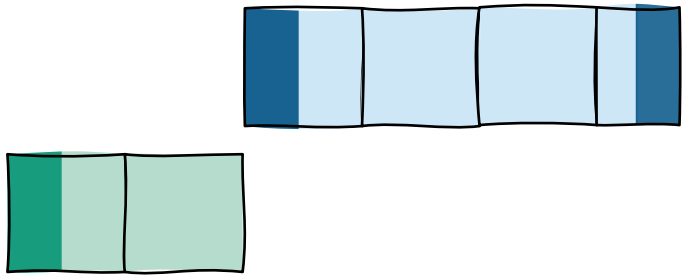
Foremost

http://foremost.sourceforge.net



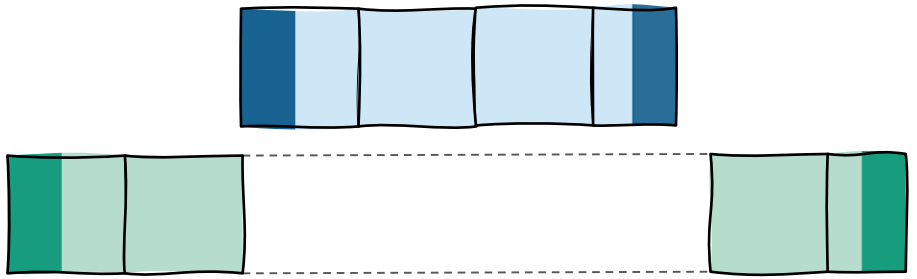
Scalpel

https://github.com/sleuthkit/scalpel



PhotoRec

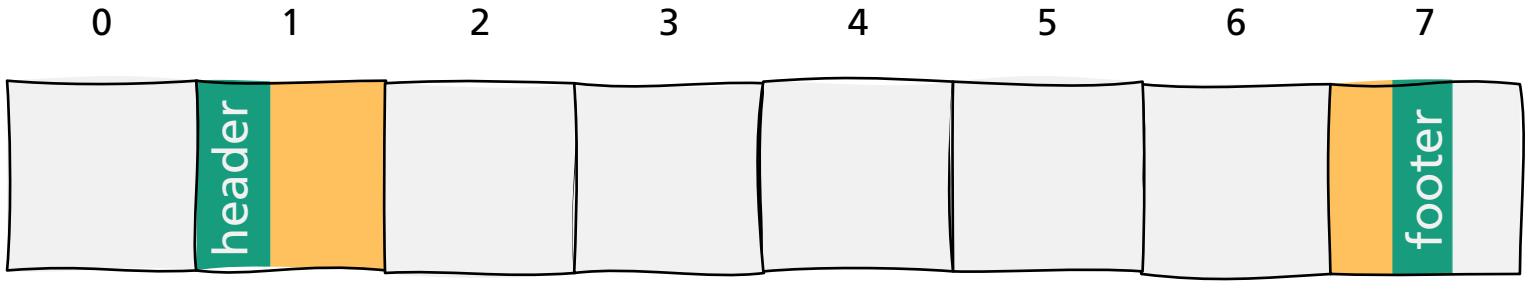
https://www.cgsecurity.org/wiki/PhotoRec



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

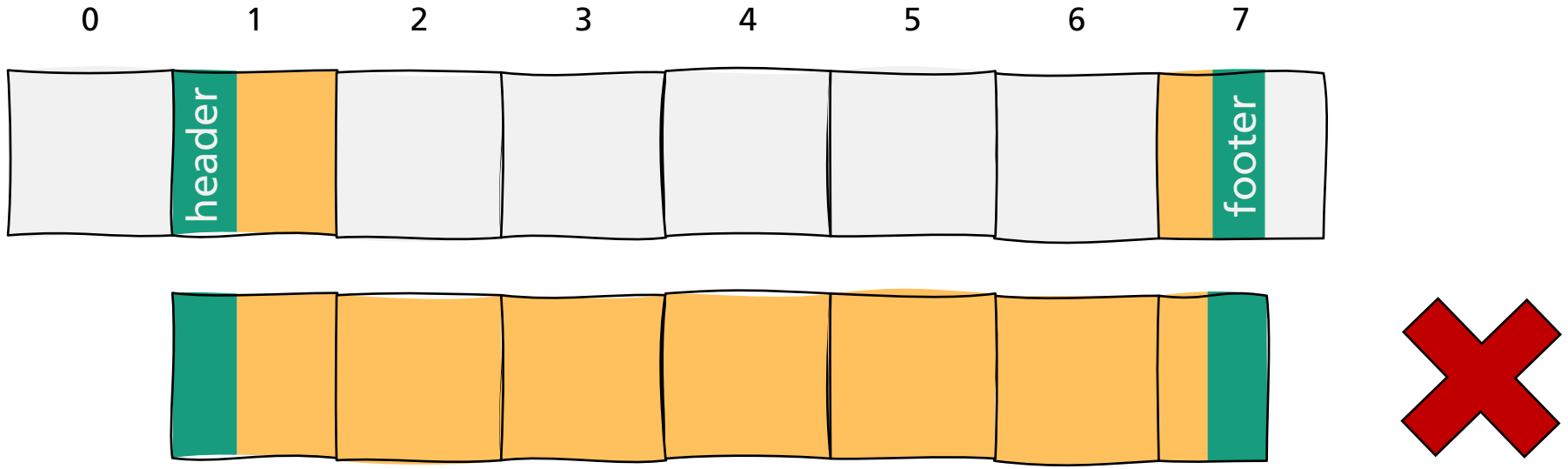
Bifragment Gap Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Bifragment Gap Carving



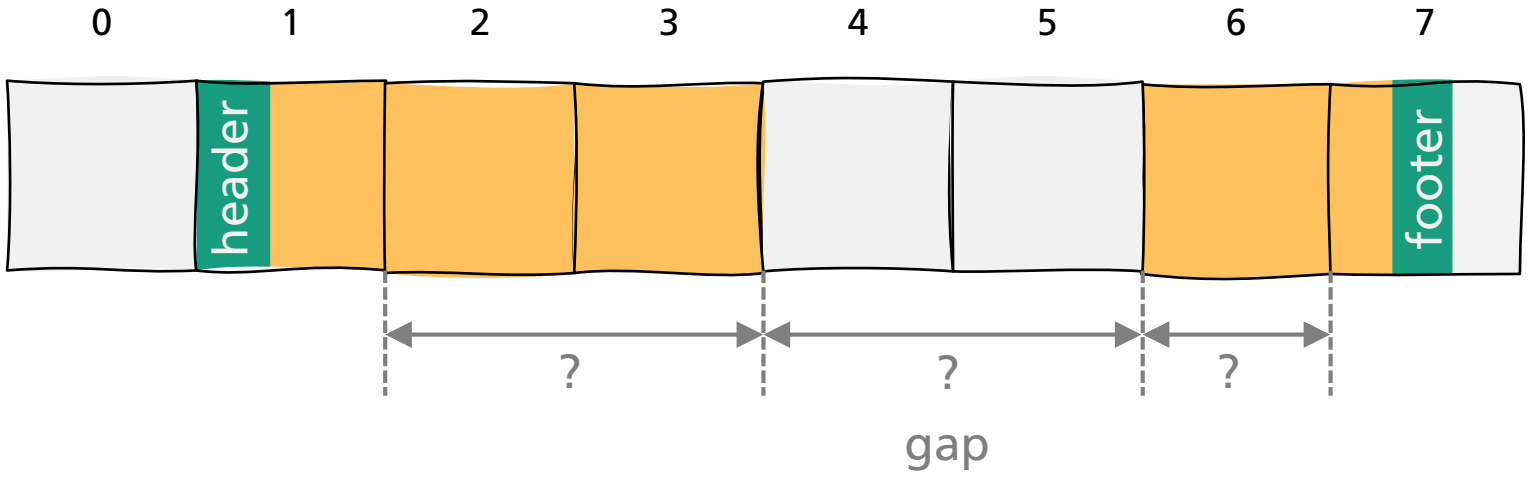
Fast object validation

- Container structures
- Decompression
- Semanctics
- Manual
- …

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

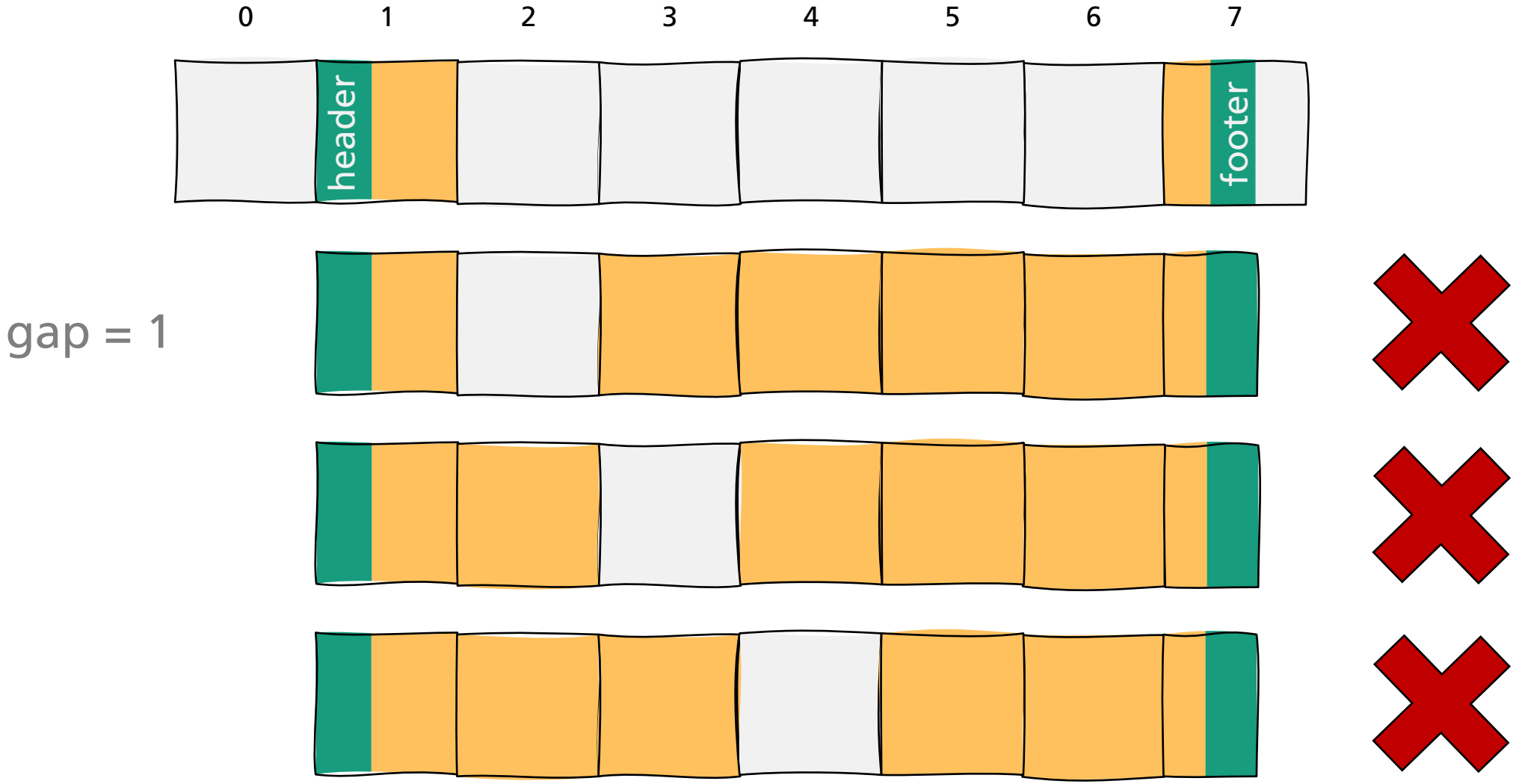
Bifragment Gap Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

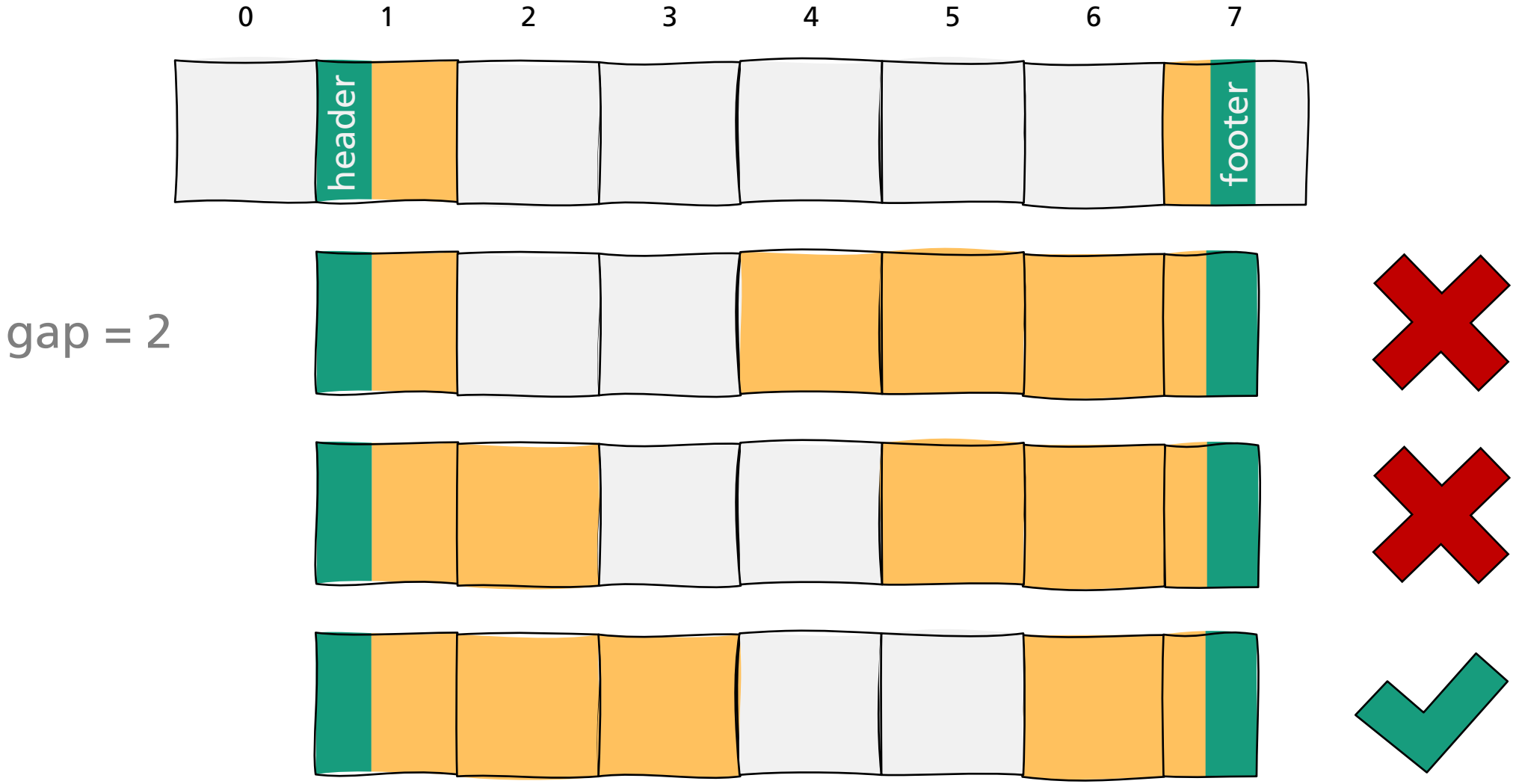
Bifragment Gap Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Bifragment Gap Carving

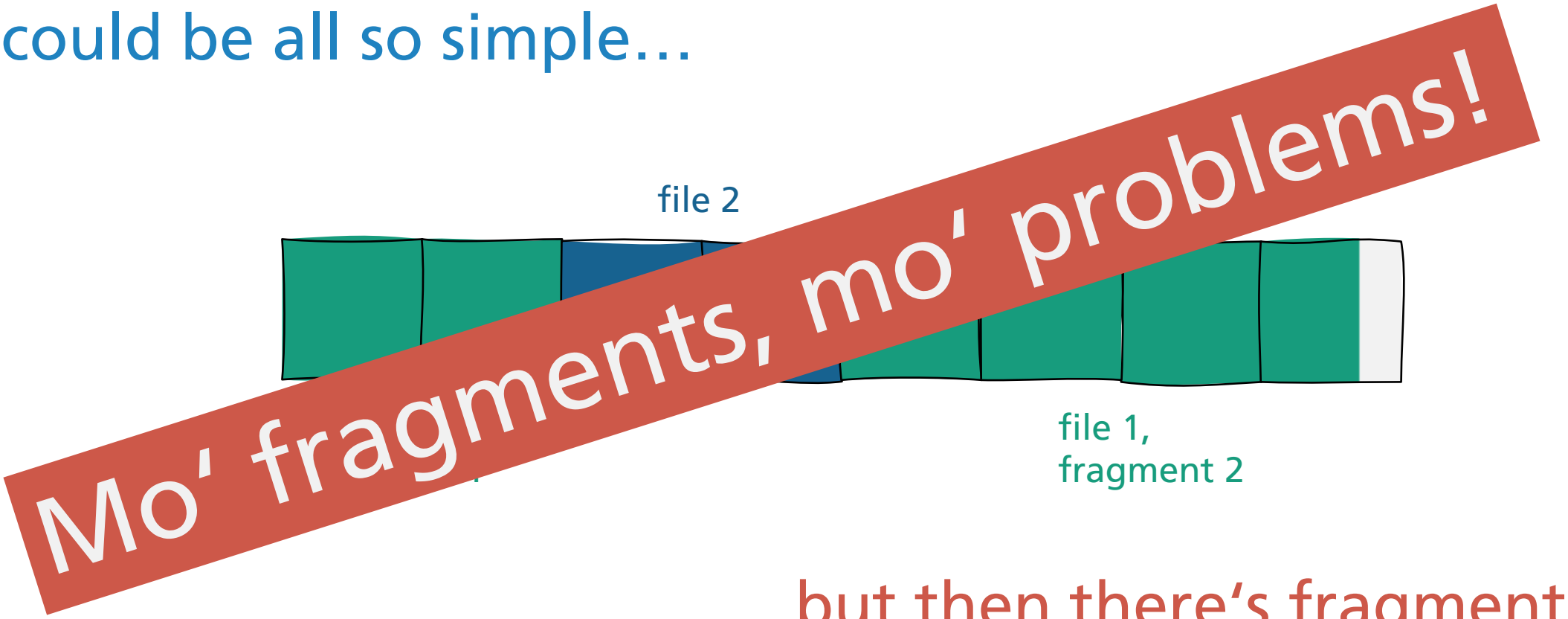


Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving. Mo‘ Problems.

It could be all so simple…

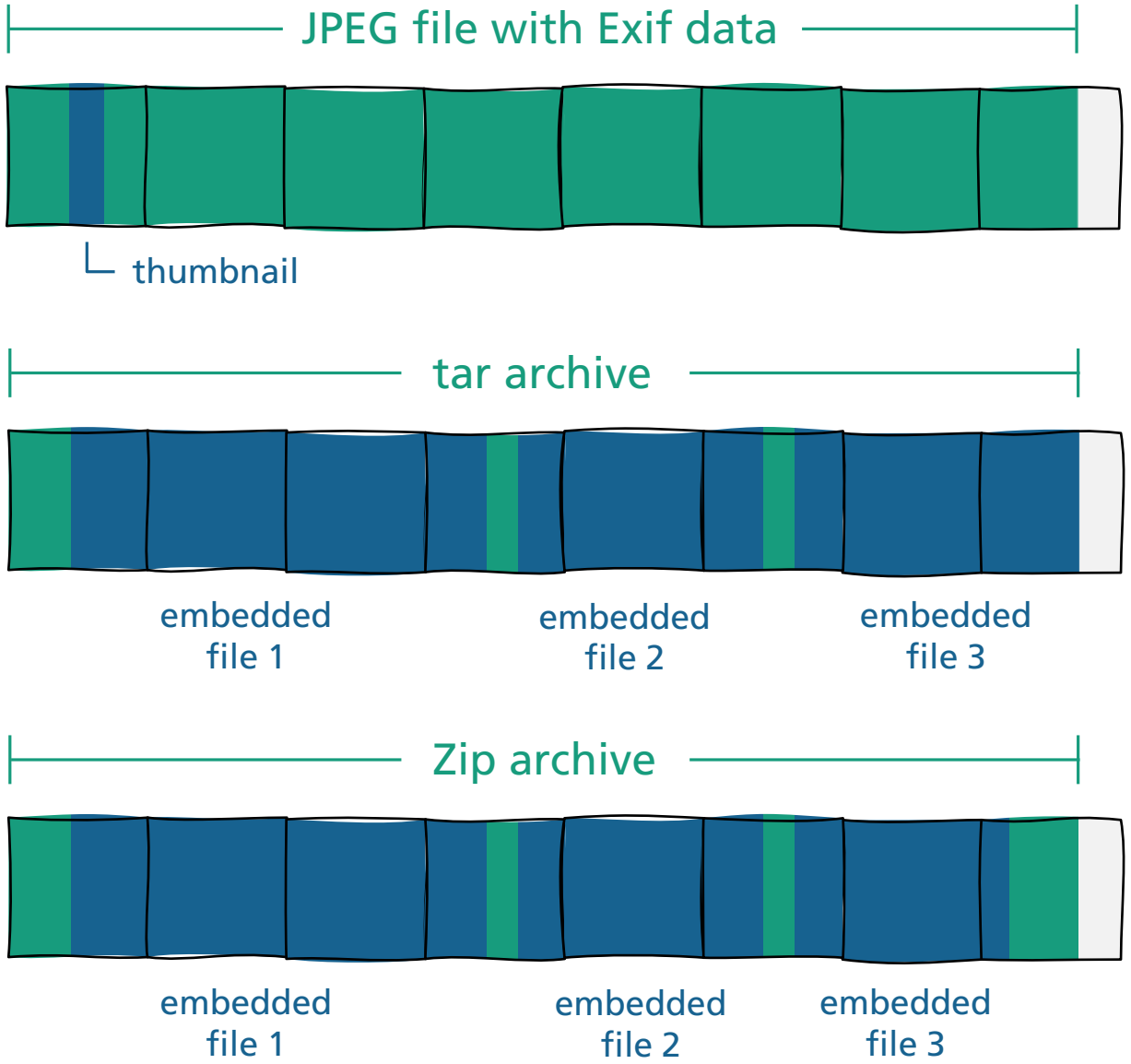


…but then there‘s fragmentation.

And embedded files.

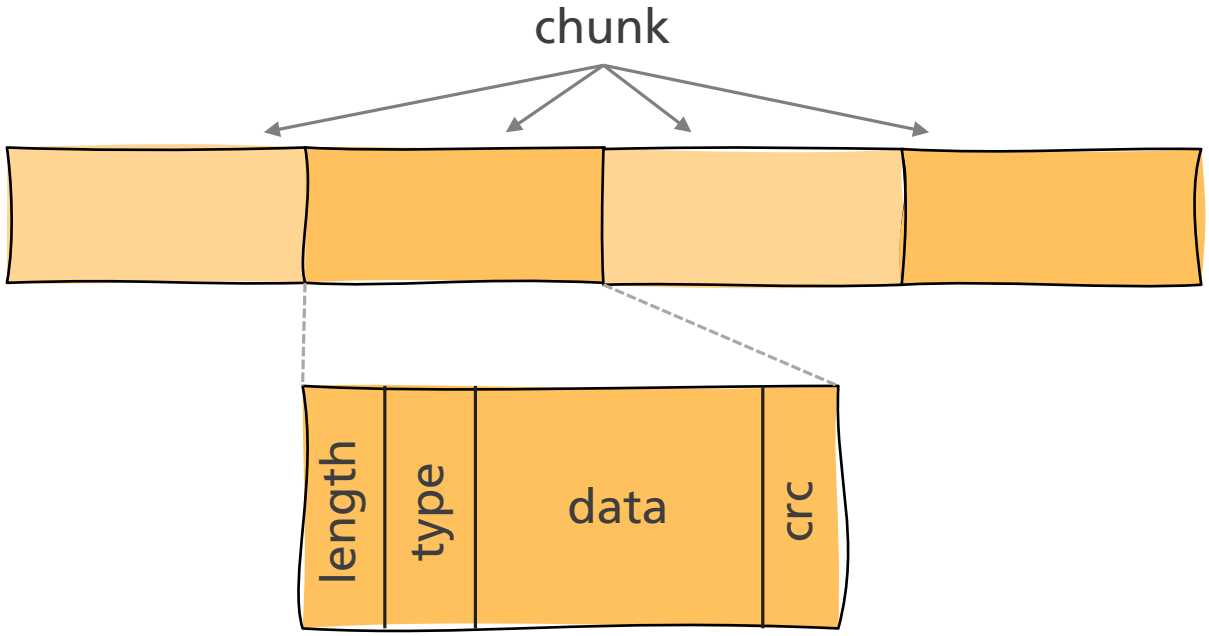
Interlude: Embedded Files

What is a file?



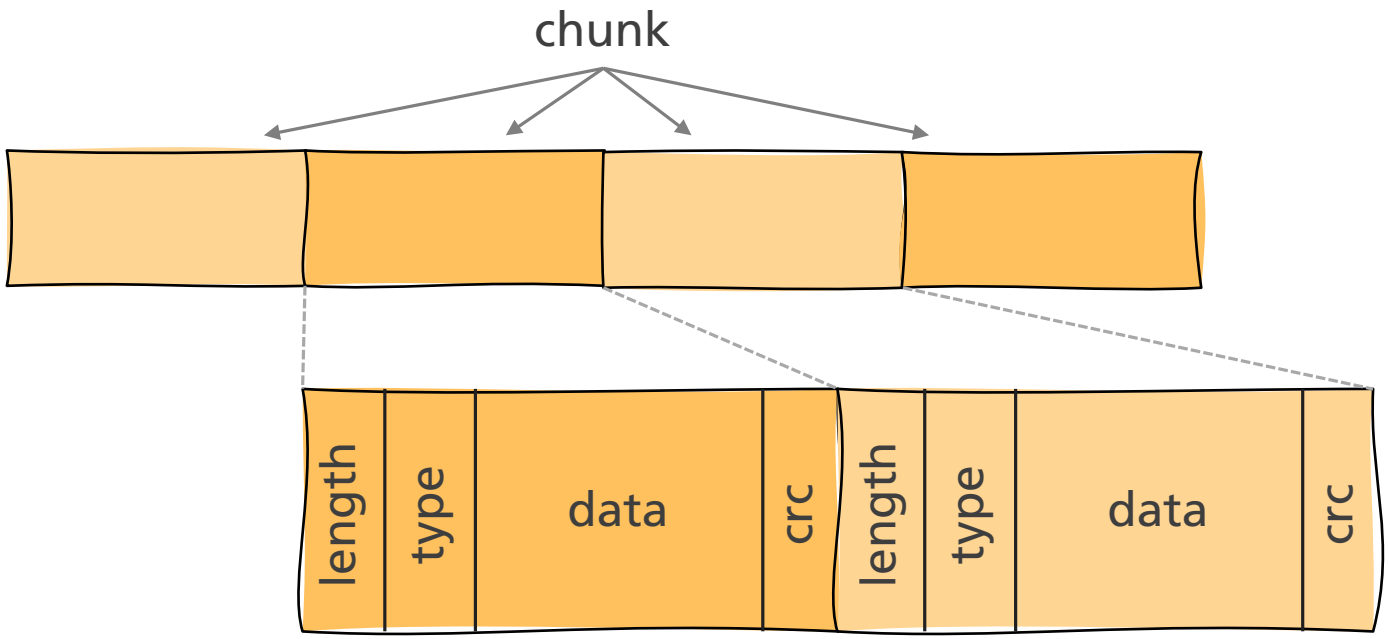
Syntax-based Carving

using the example of PNG



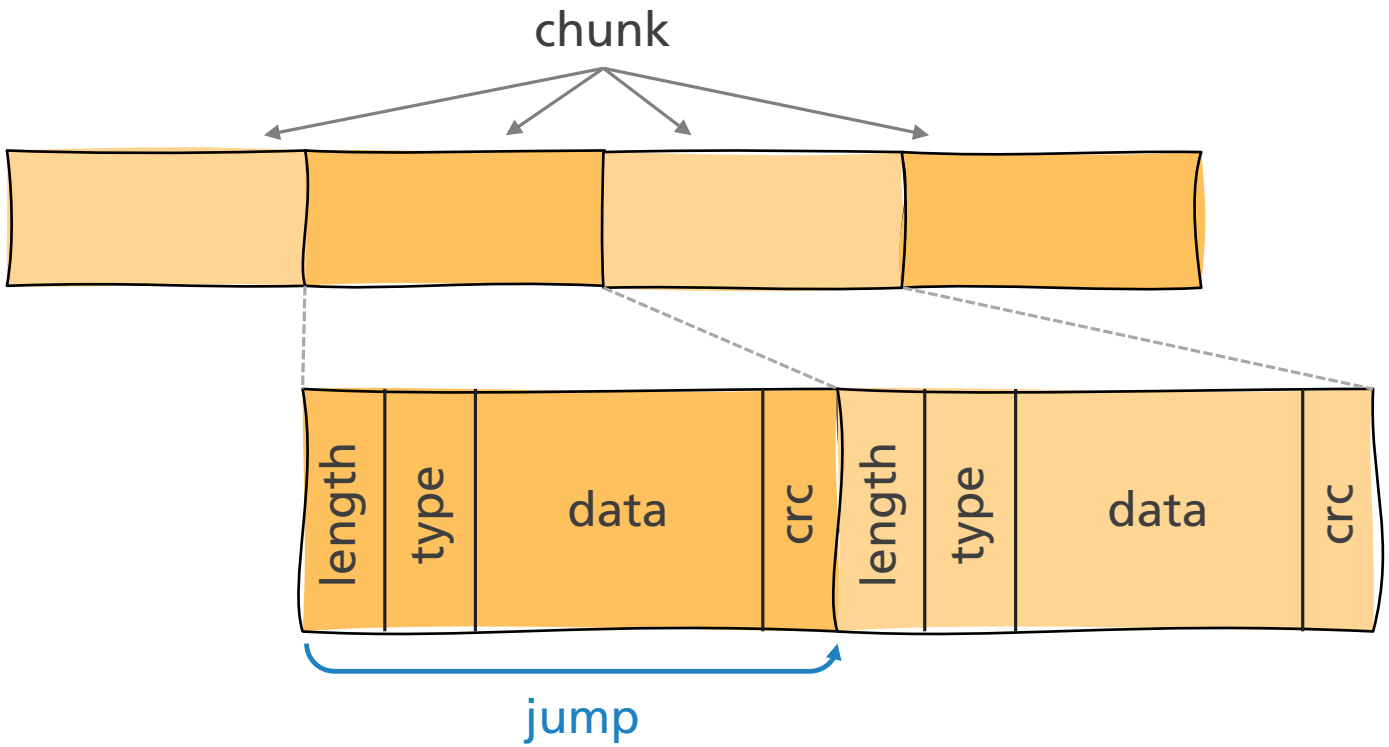
PNG file (simplified)

Syntax-based Carving



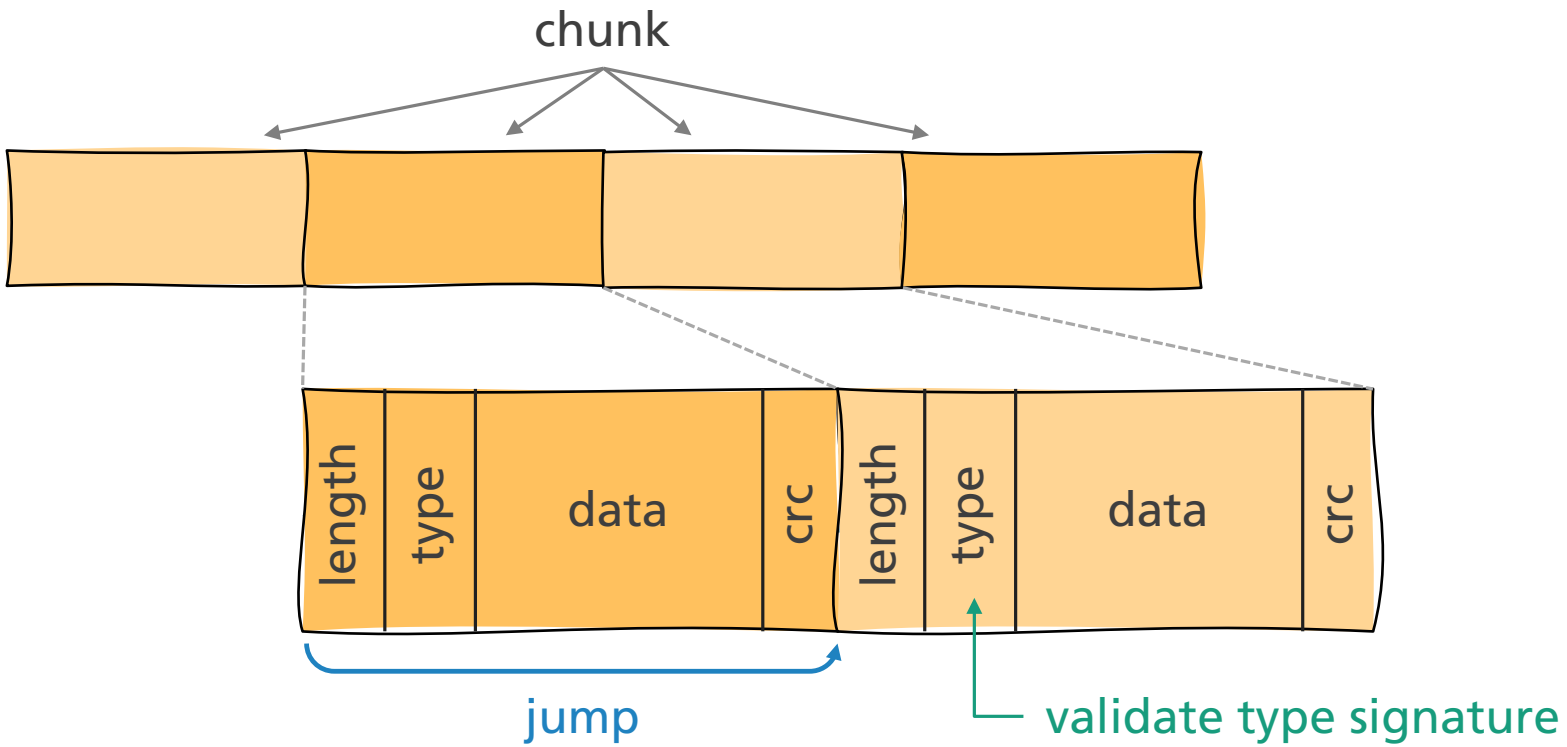
PNG file (simplified)

Syntax-based Carving



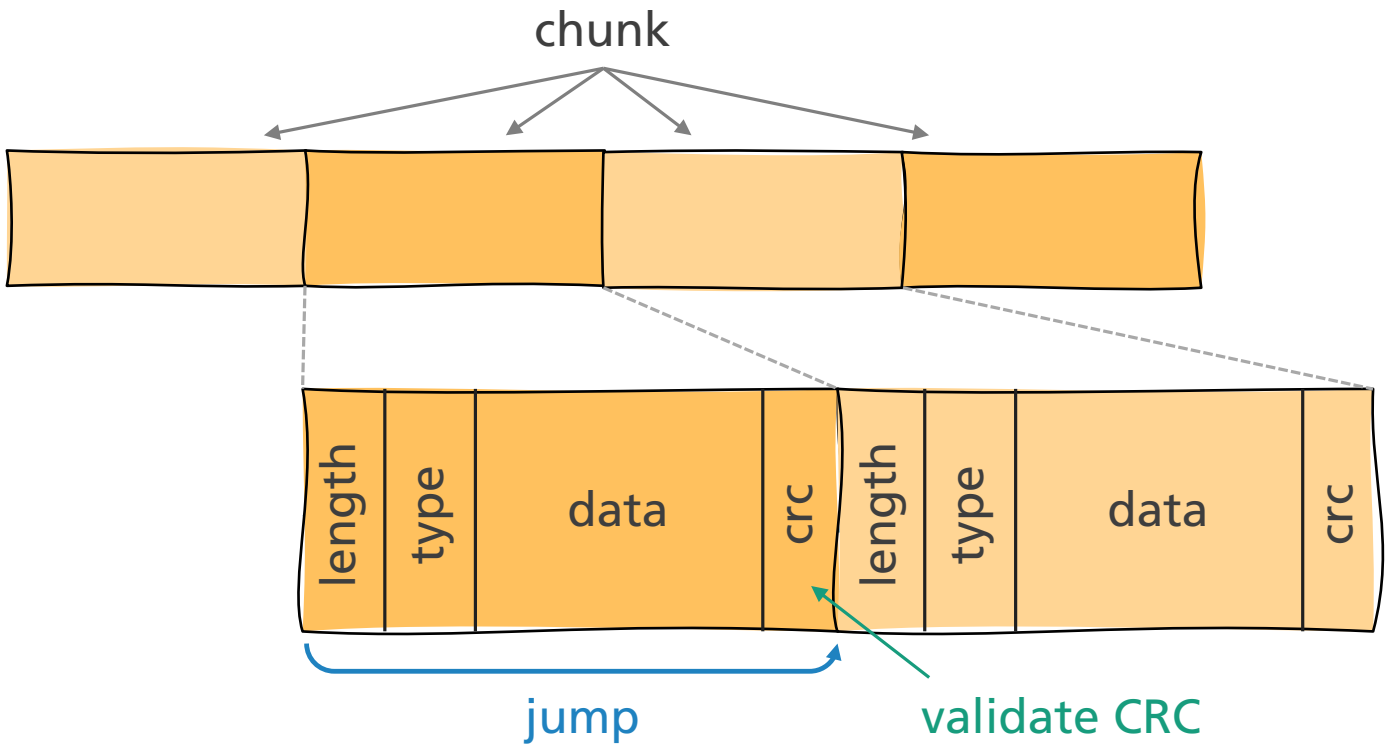
PNG file (simplified)

Syntax-based Carving



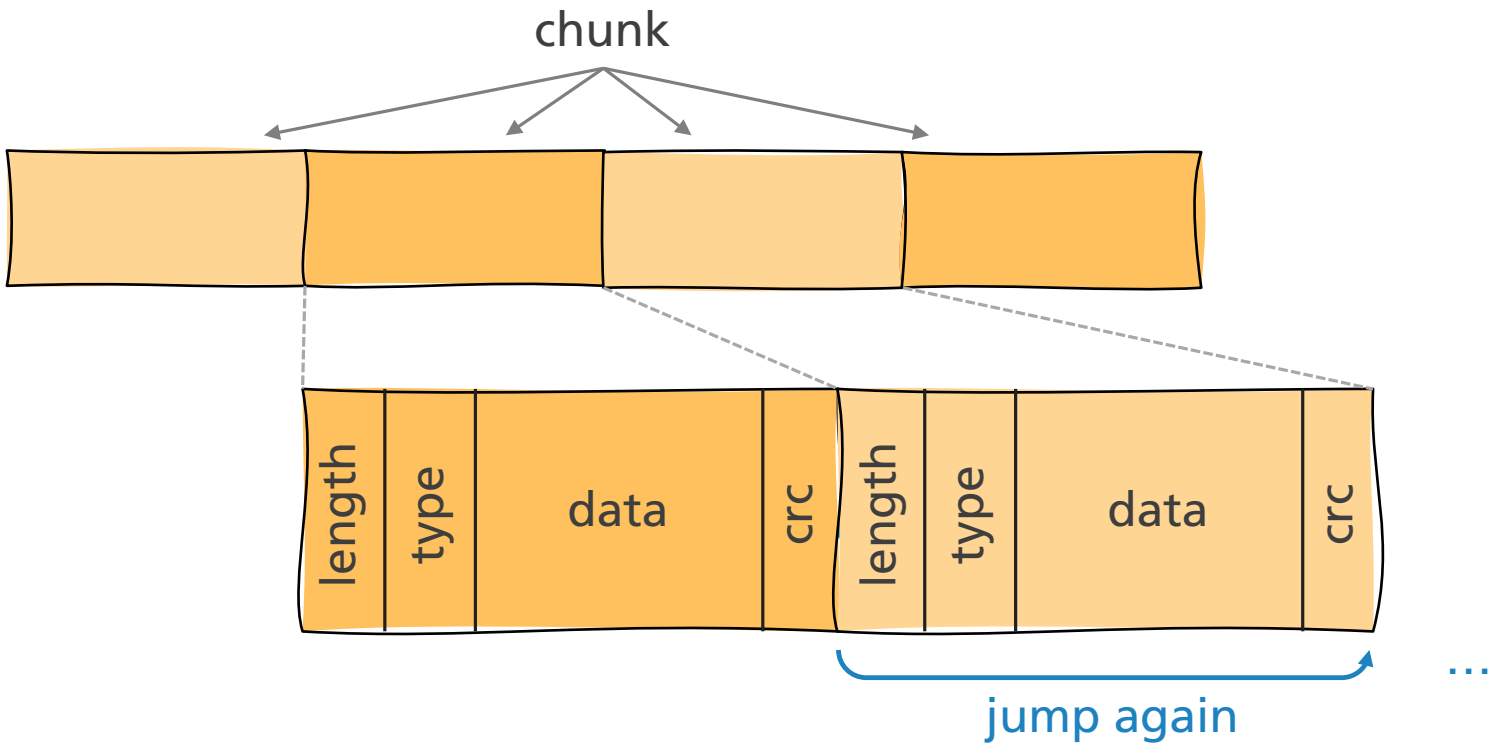
PNG file (simplified)

Syntax-based Carving



PNG file (simplified)

Syntax-based Carving

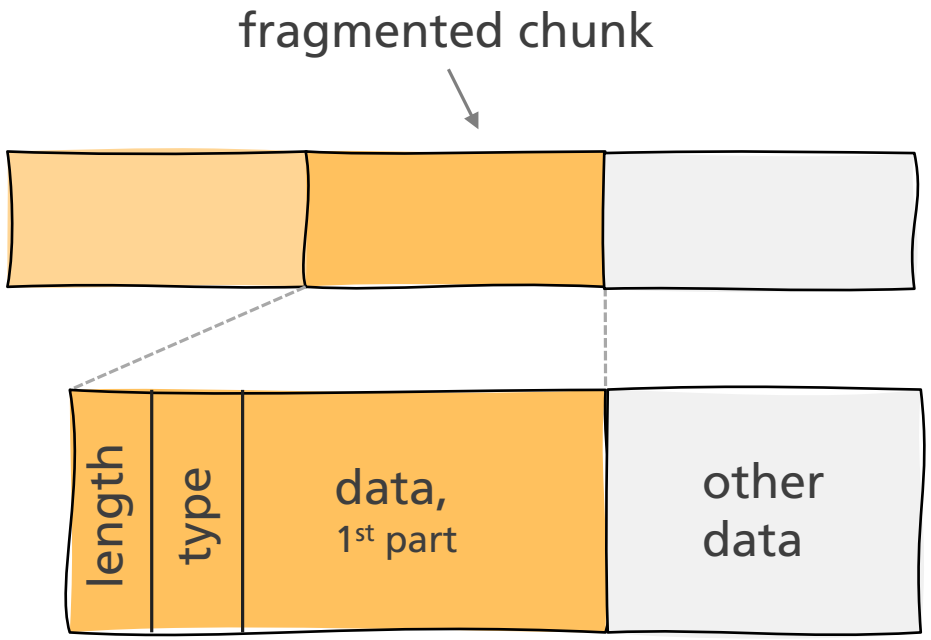


PNG file (simplified)

Hochschule Bonn-Rhein-Sieg

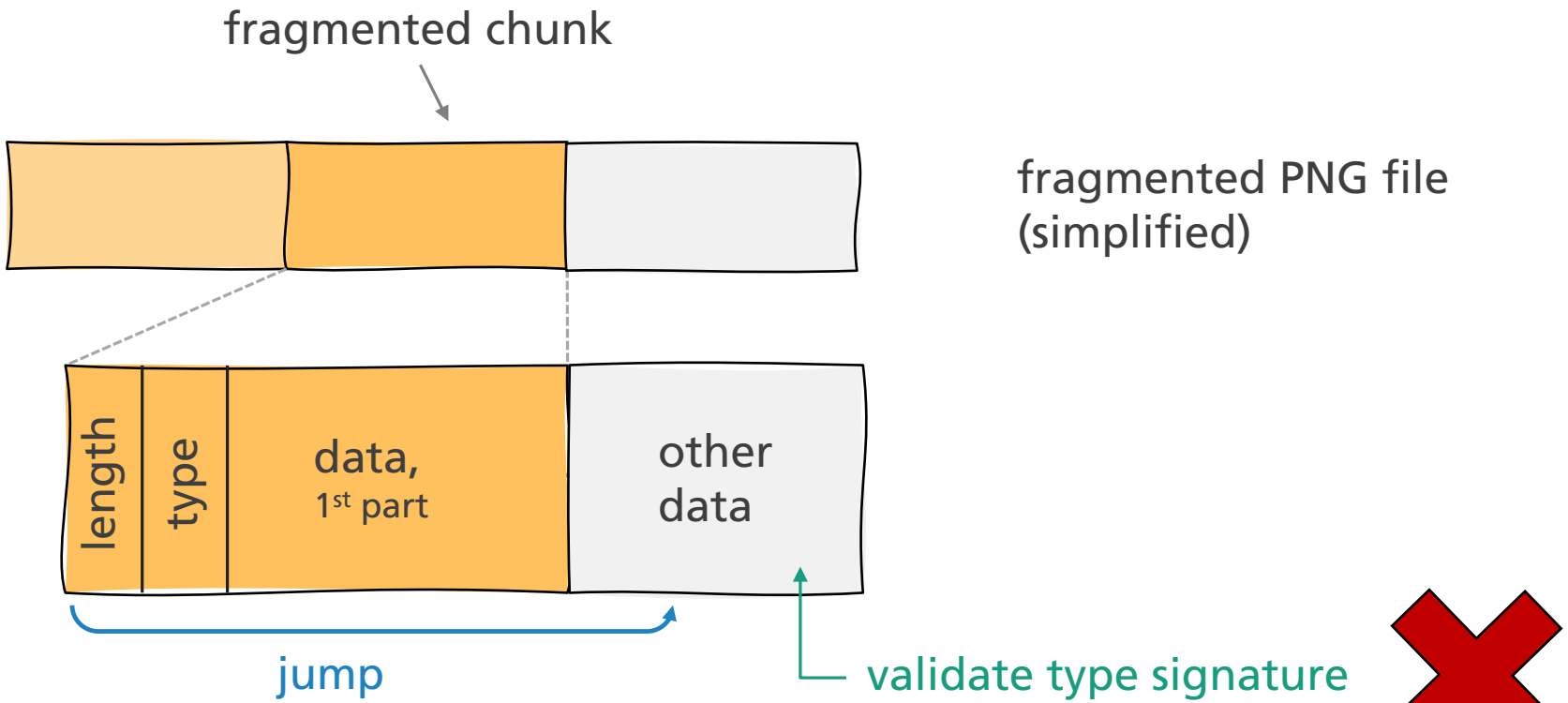
Fraunhofer FKIE

Syntax-based Carving



fragmented PNG file
(simplified)

Syntax-based Carving

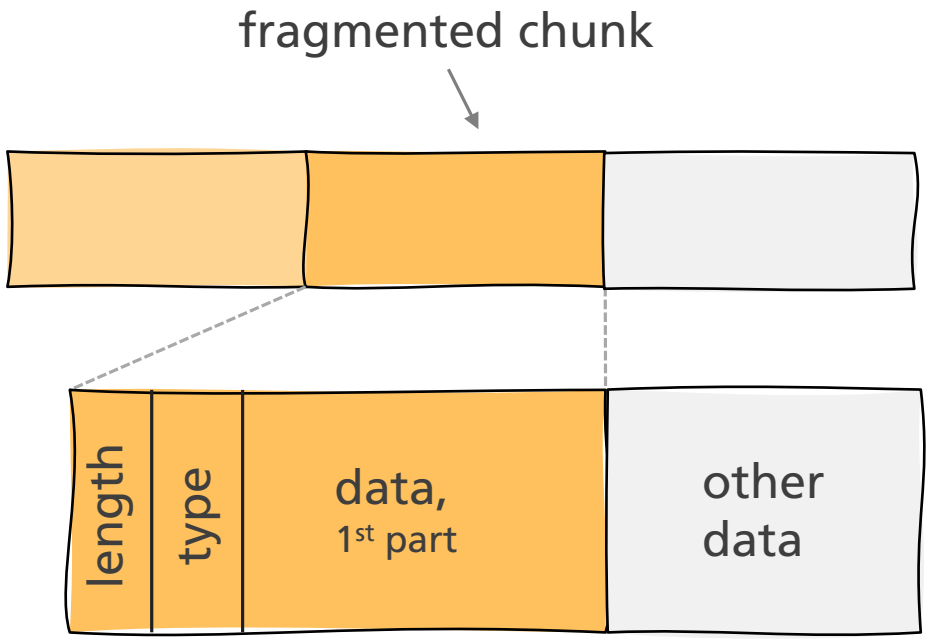


fragmented PNG file
(simplified)

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Syntax-based Carving



fragmented PNG file
(simplified)

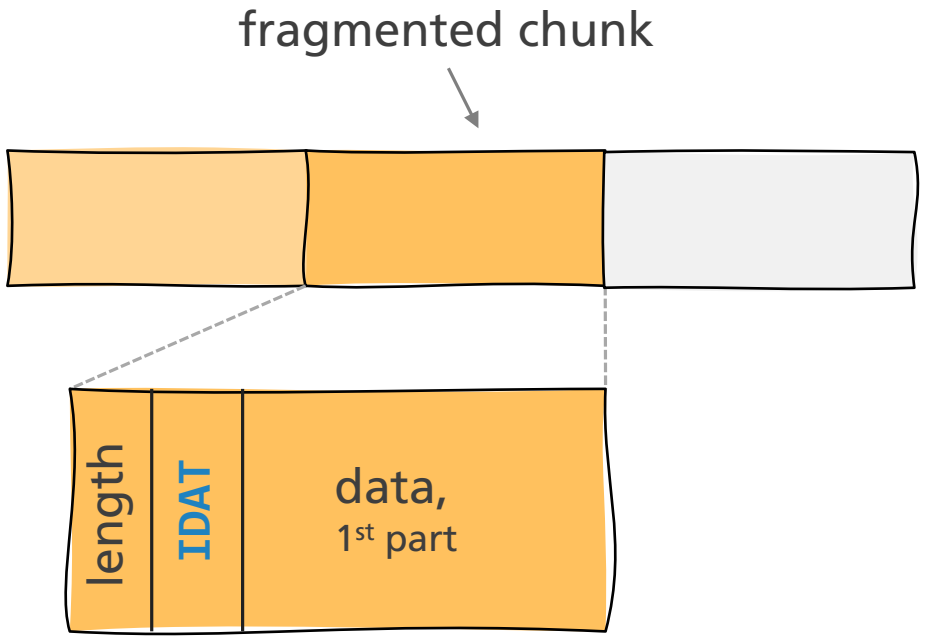
How do we find the next fragment?

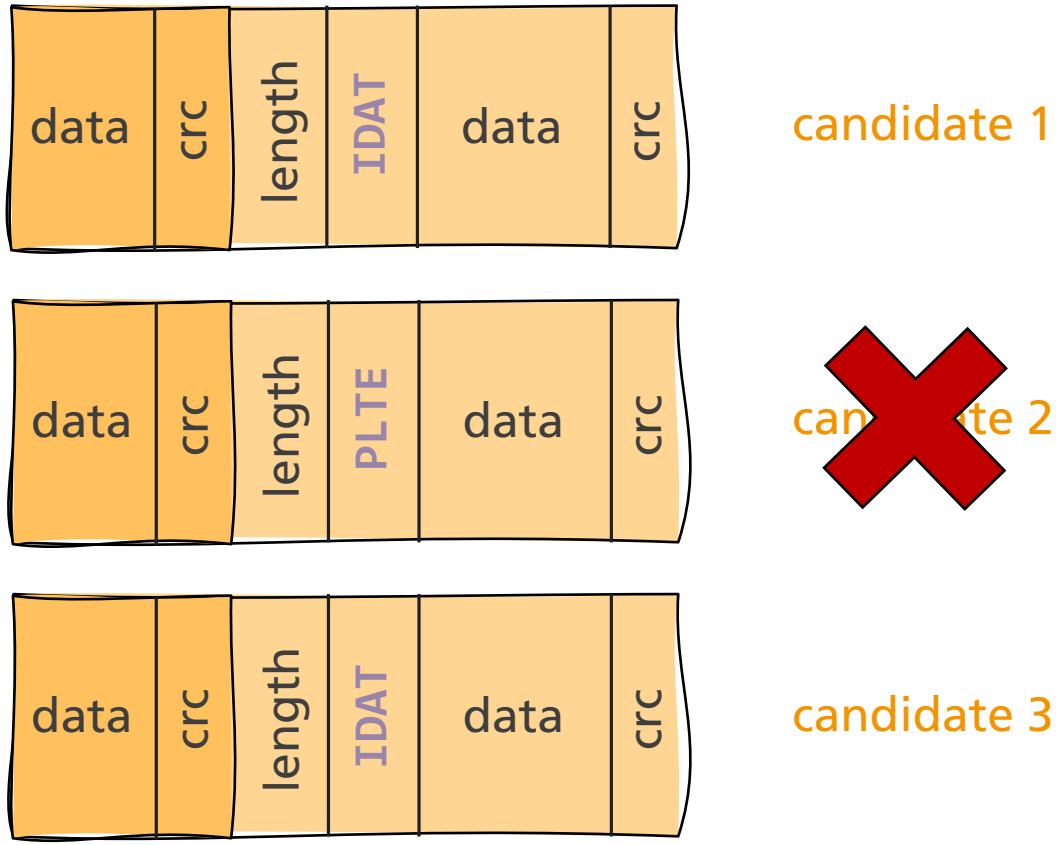
Use file format syntax and
storage device properties!

Syntax-based Carving

How do we find the next fragment?

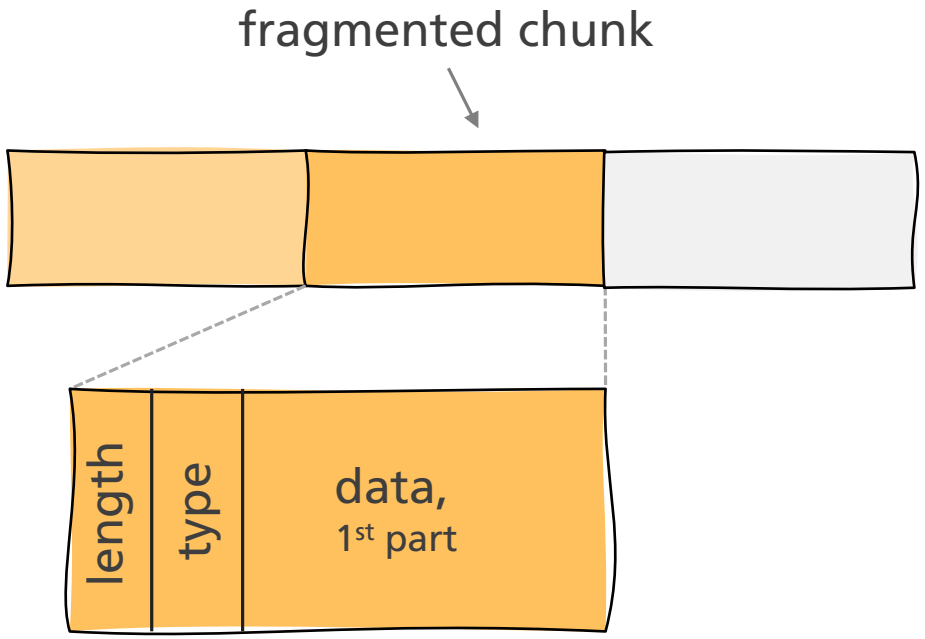
File format syntax and storage device properties!





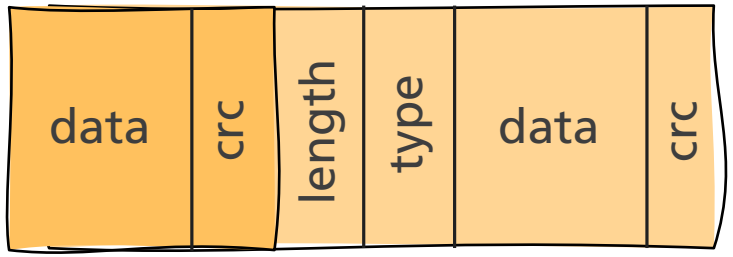
Not a valid candidate because of chunk ordering constraints.

Syntax-based Carving

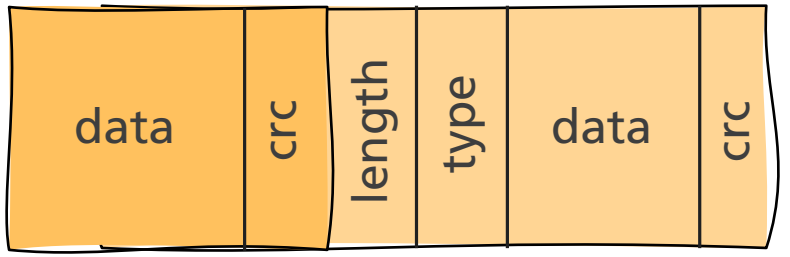


How do we find the next fragment?

File format syntax and storage device properties!



candidate 1



candidate 2

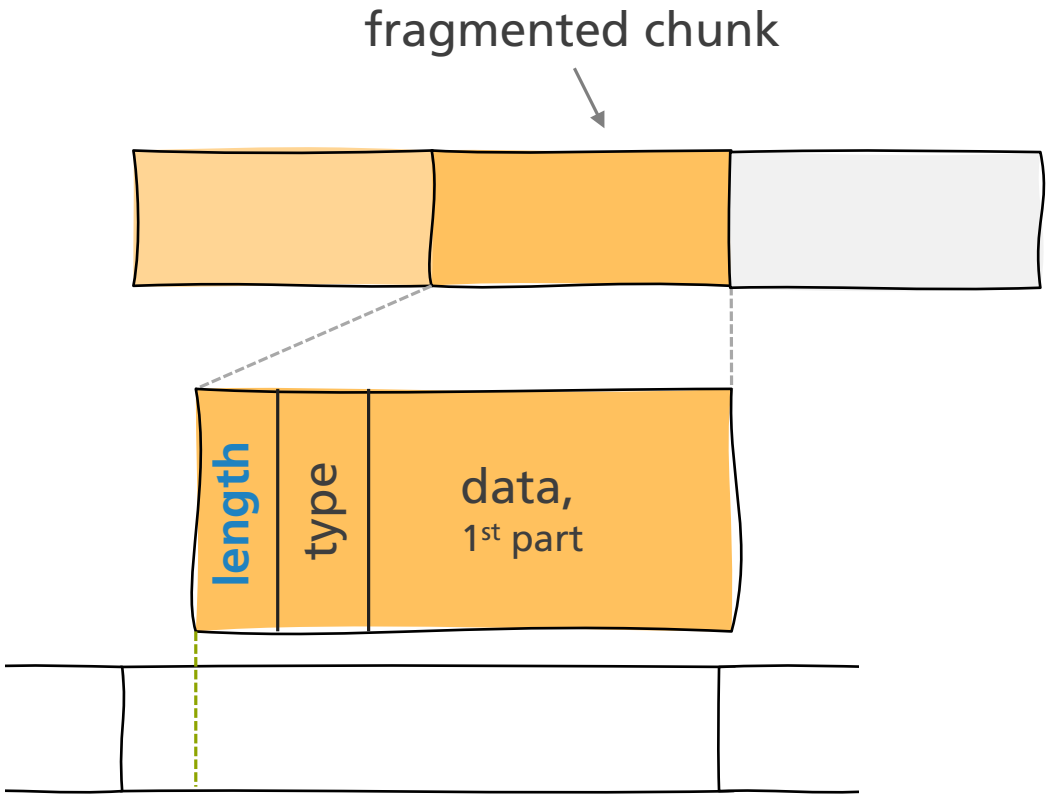
Hochschule Bonn-Rhein-Sieg

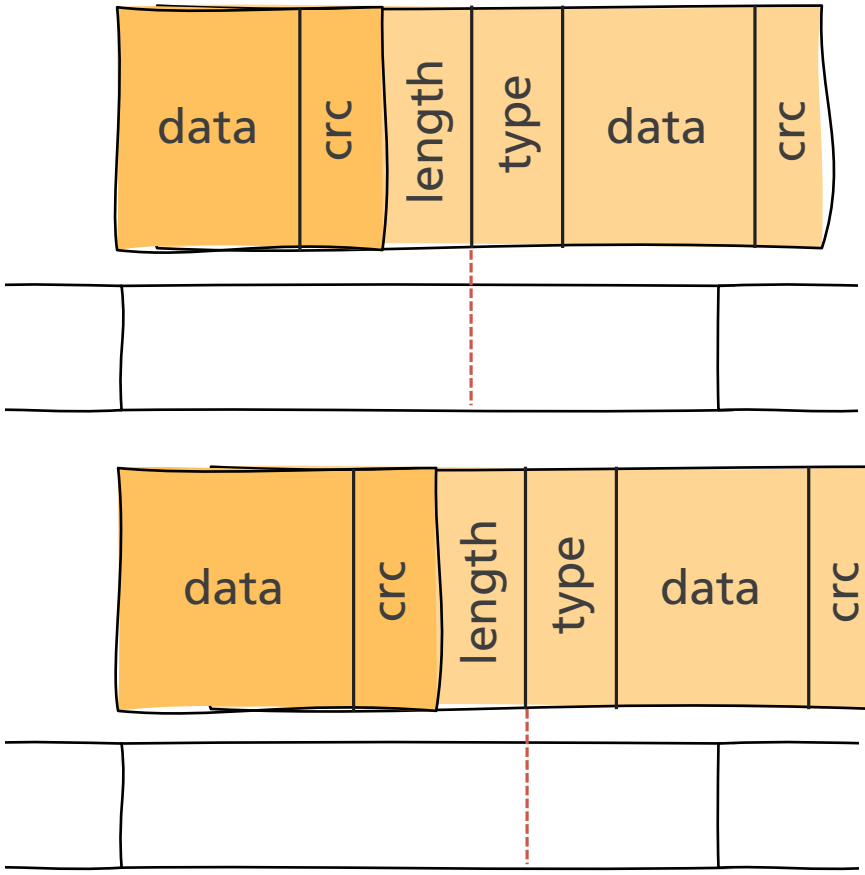
Fraunhofer FKIE

Syntax-based Carving

How do we find the next fragment?

File format syntax and storage device properties!

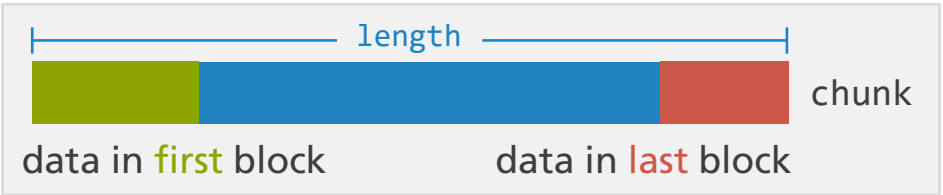




candidate 1

candidate 2

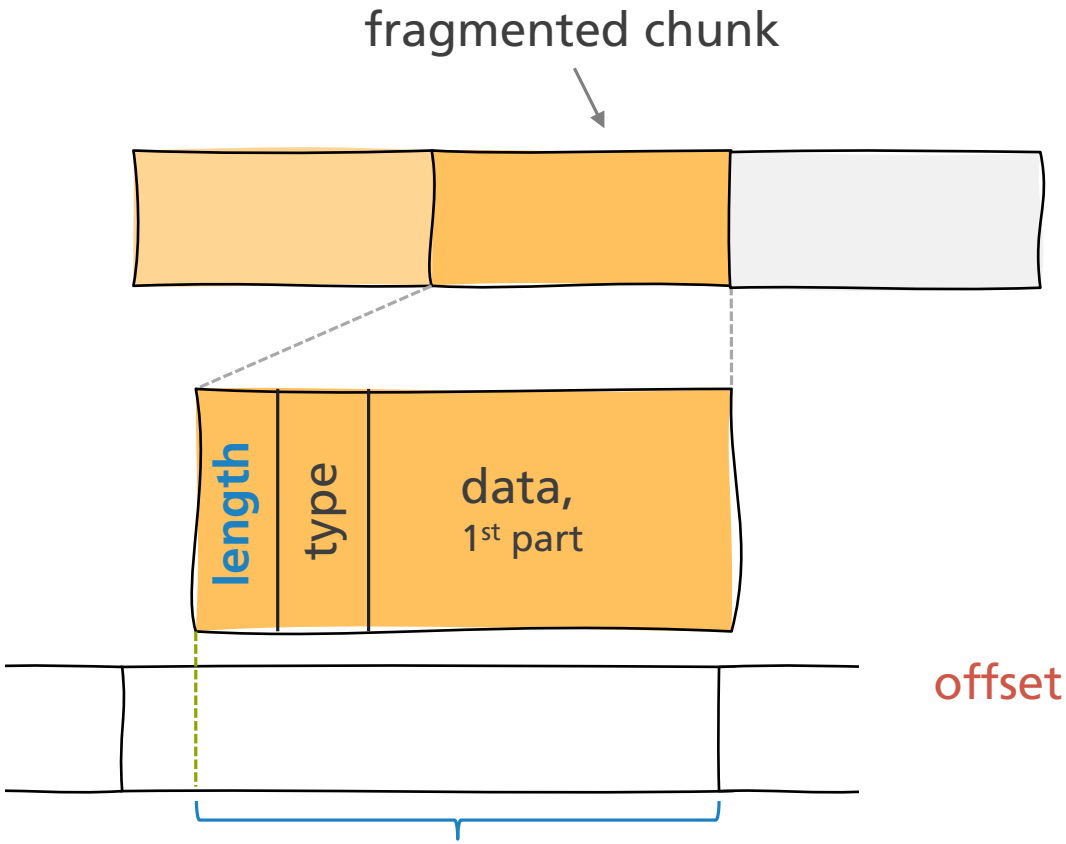
storage device/fs blocks

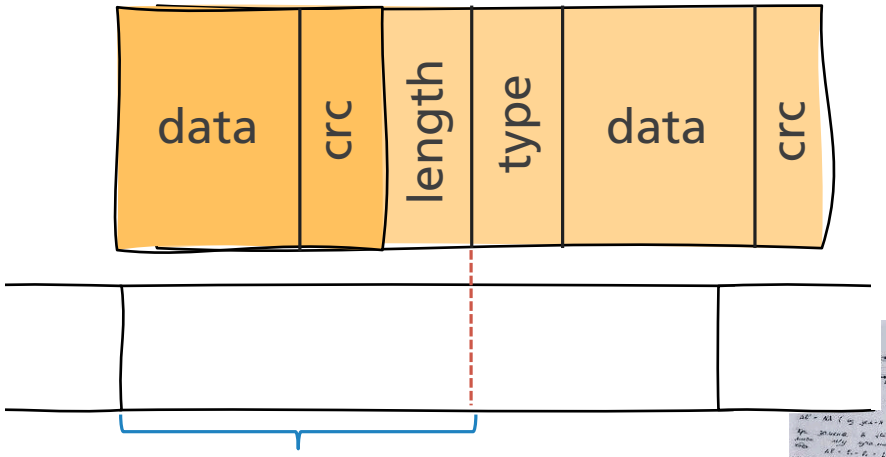


Syntax-based Carving

How do we find the next fragment?

File format syntax and storage device properties!



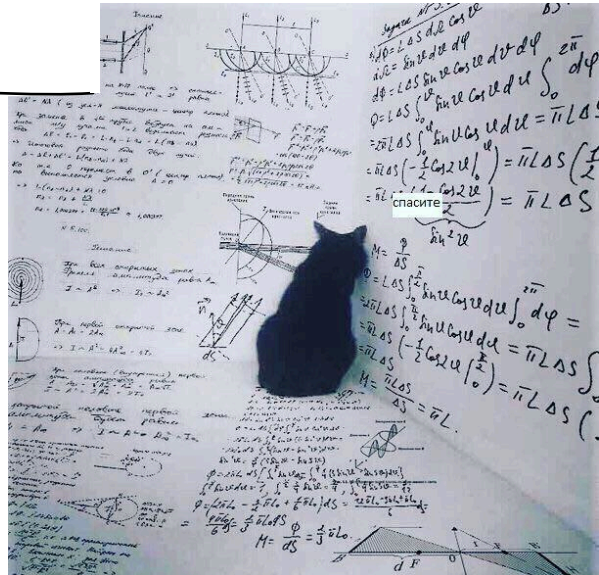


candidate 1

offset % block size = Δ_b

Δ_f = block size – (offset % block size)

Valid successor fragments: (chunk size – Δ_b – Δ_f) % block size = 0

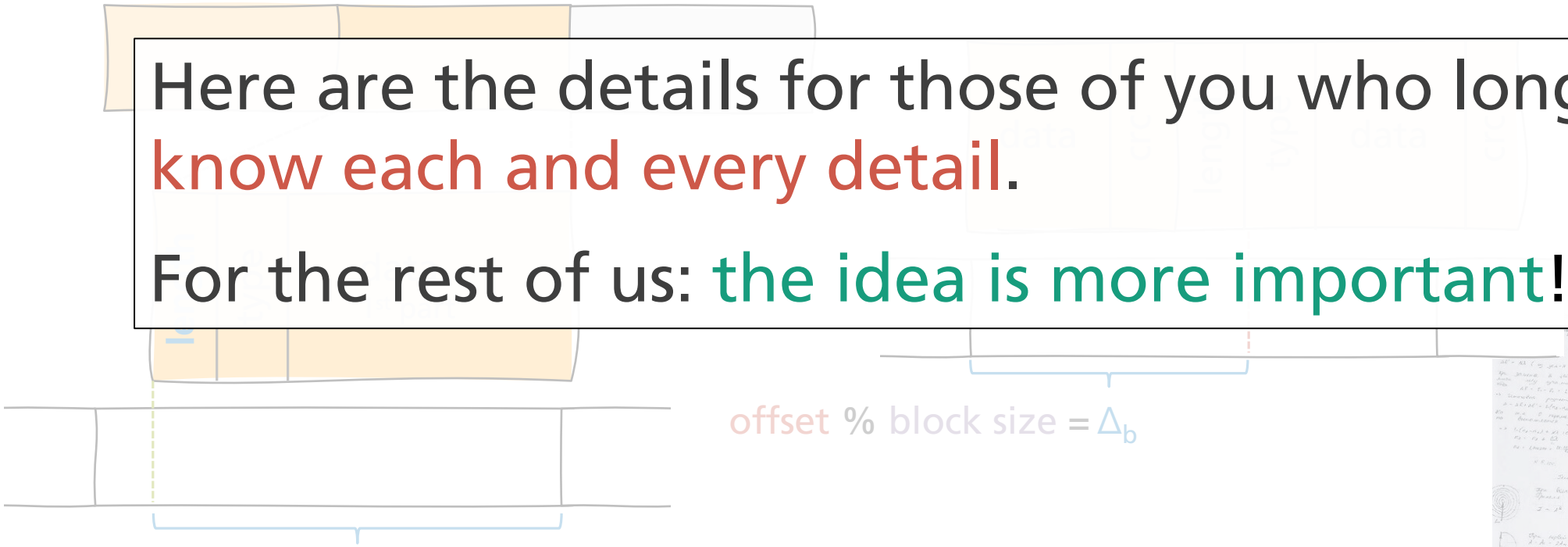


Syntax-based Carving

How do we find the next fragment?

File format syntax and storage device properties!

fragmented chunk



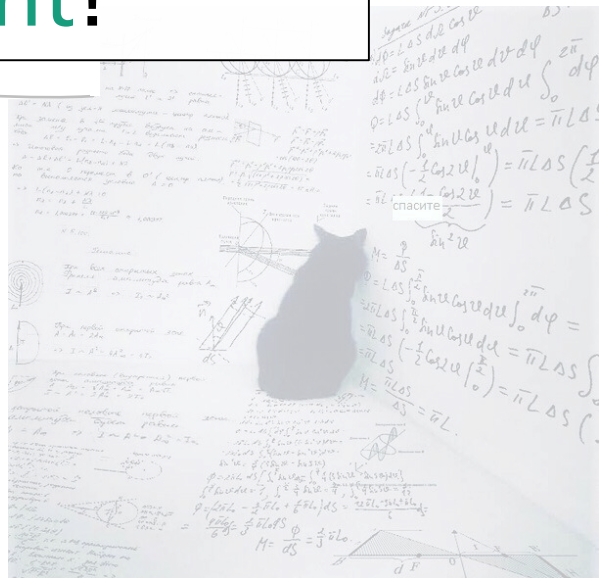
Here are the details for those of you who long to know each and every detail.

For the rest of us: the idea is more important!

offset % block size = Δ_b

Δ_f = block size – (offset % block size)

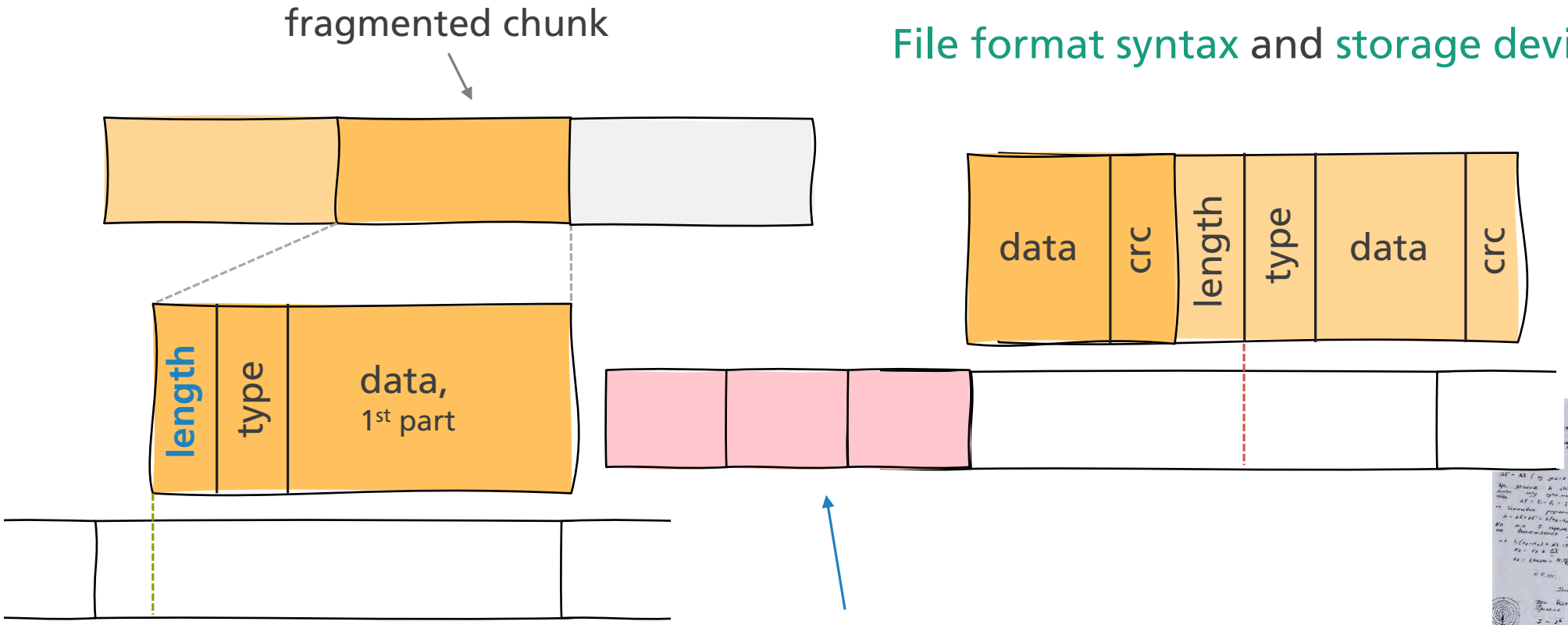
Valid successor fragments: (chunk size – Δ_b – Δ_f) % block size = 0



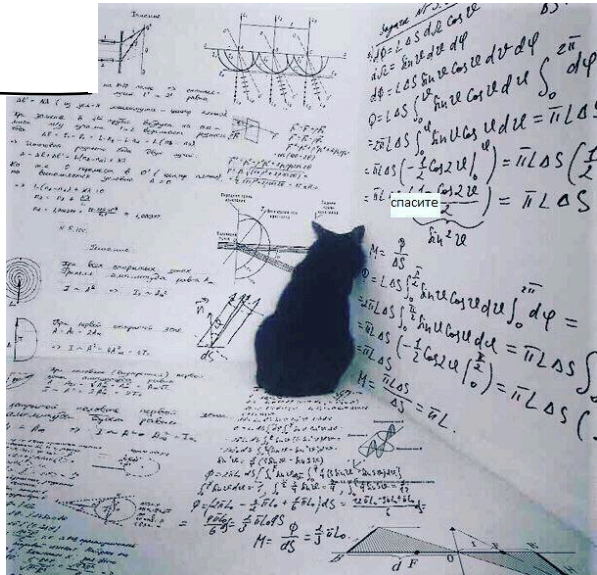
Syntax-based Carving

How do we find the next fragment?

File format syntax and storage device properties!



Use bifragment gap carving or similar strategies to generate block combinations.

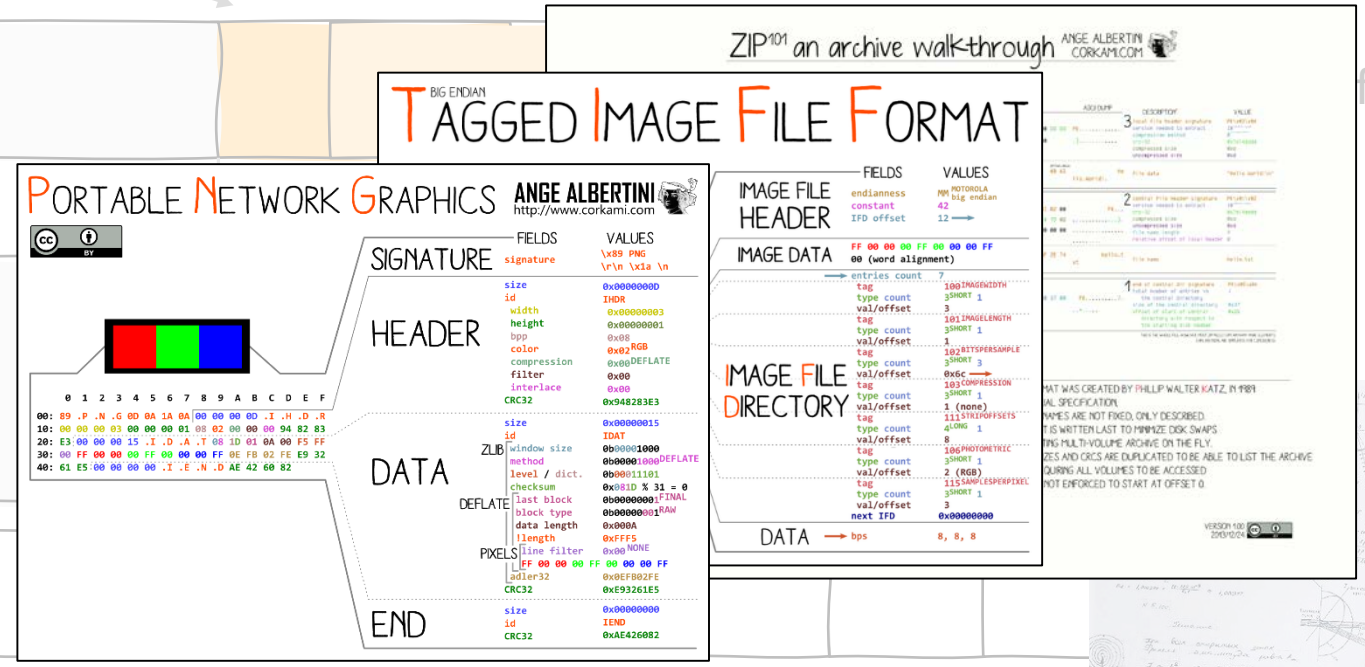


Syntax-based Carving

fragmented chunk

File formats can be very complex!

And there are lots of 'em!



Δ_f = block size – (offset % block size)

Δ_b = offset % block size

Valid successor fragments: (chunk size – Δ_b – Δ_f) % block size = 0

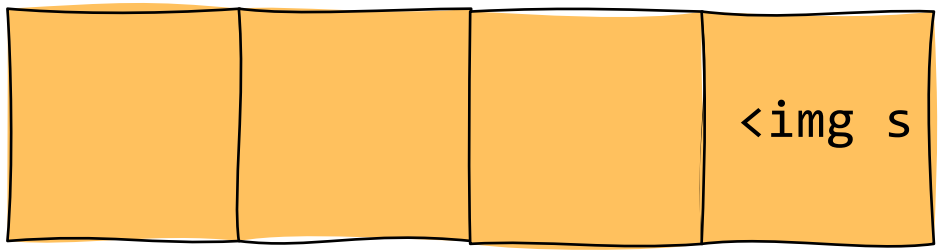
Syntax-based Carving

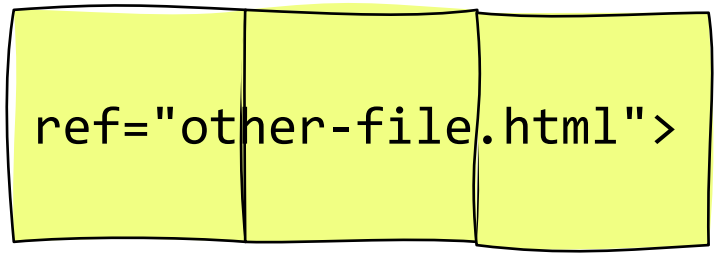
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

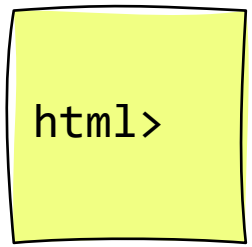
Content-based Carving

base fragment

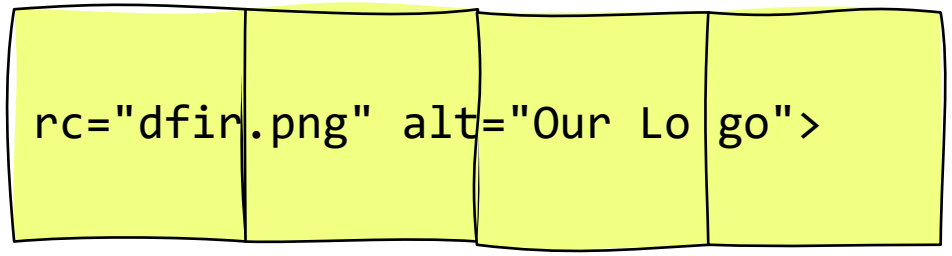




candidate 1



candidate 2



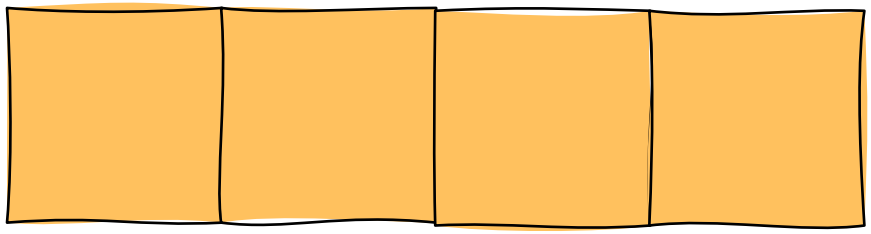
cand. 3

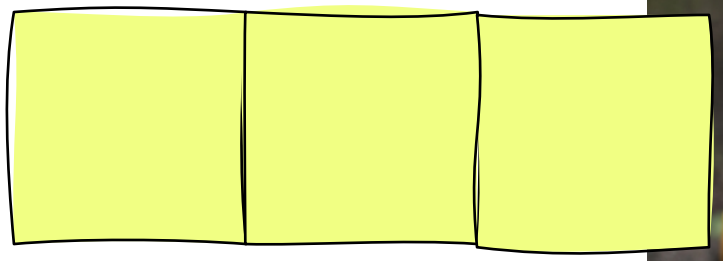
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

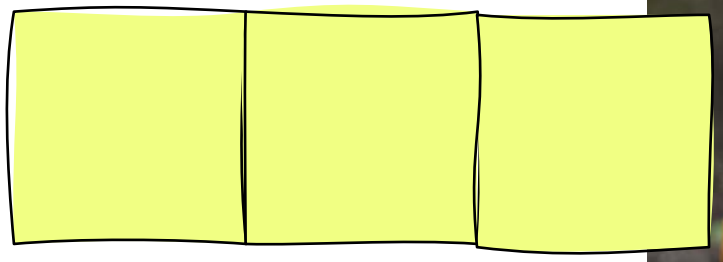
Content-based Carving

base fragment





candidate 1

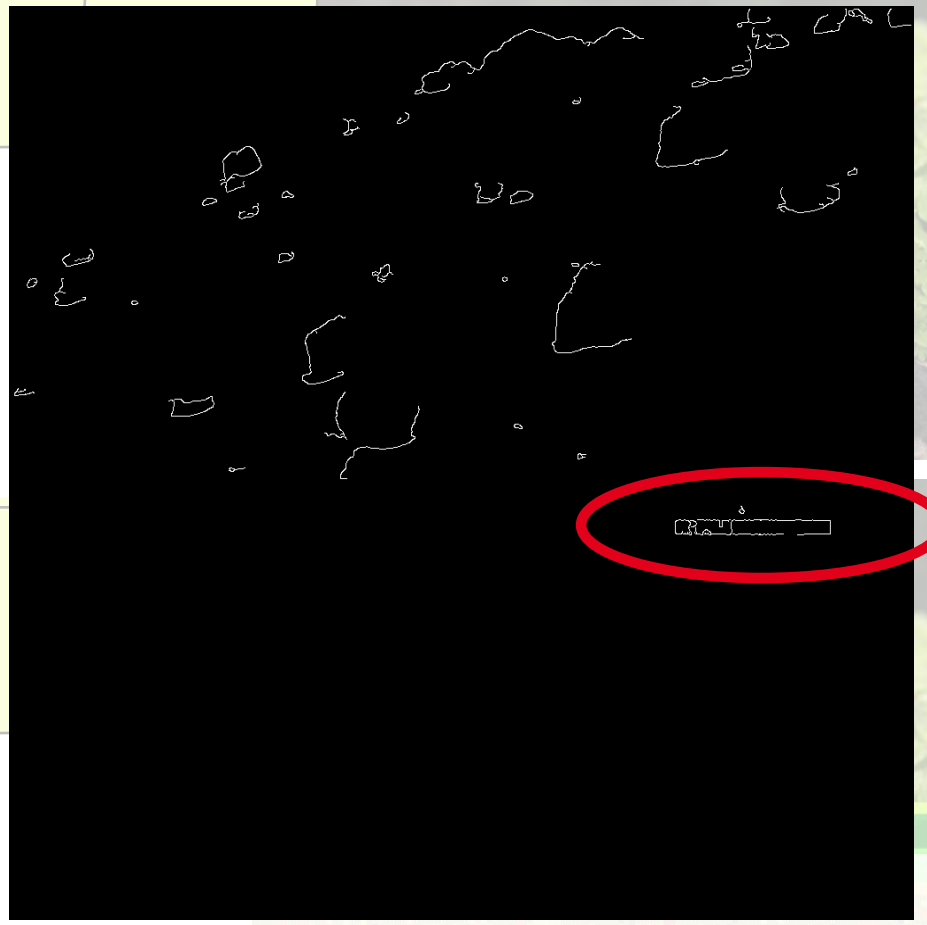


candidate 2

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Content-based Carving

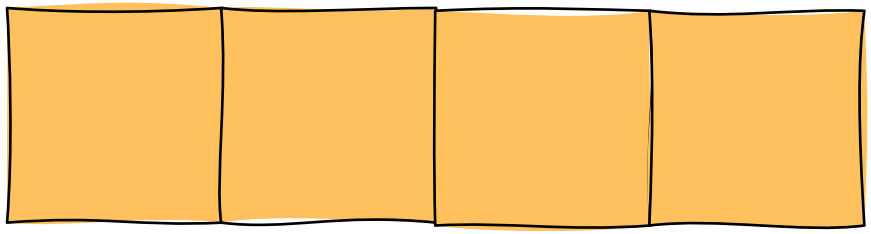


Hochschule Bonn-Rhein-Sieg

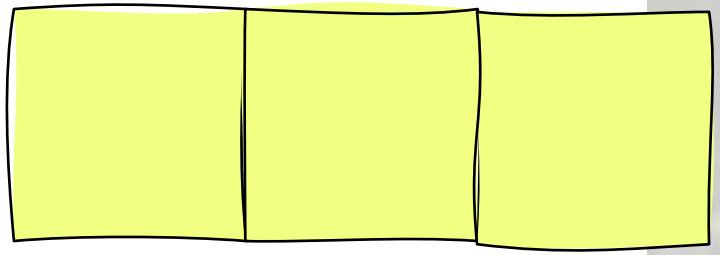
Fraunhofer FKIE

Content-based Carving

base fragment

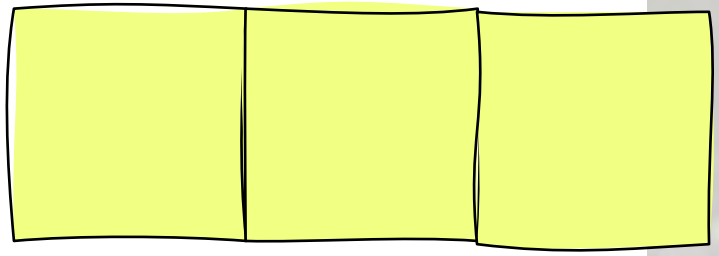


candidate 1

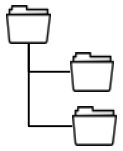


Trying a lot of candidates is expensive!

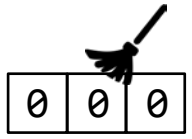
candidate 2



Search Space Reduction



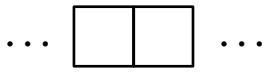
File system information:
allocated files, allocation patterns, …



Wiped/clean areas

Used blocks

Collation



Fragment growing

Try to eliminate as many blocks a possible!

Limitations

encryption

compression

fragmentation

slow

false positives (aka garbage)

no metadata

preprocessing ¯_(ツ)_/¯

use all available information

use separate carving phases

search space reduction

validation

internal metadata

File Carving Got 99 Problems.

→ Fragmentation with a lot of fragments

→ Unfortunate fragmentation

→ Fragments in arbitrary order

→ Missings fragments

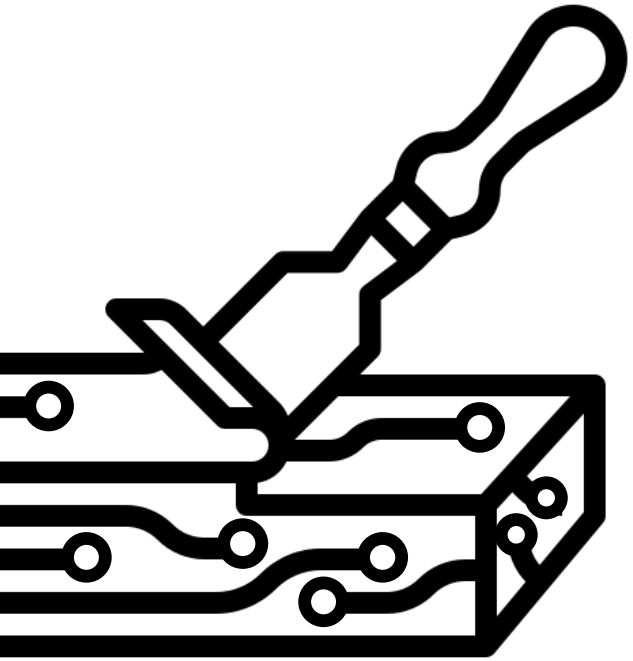
→ No (fixed) block size

→ …

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

File Carving



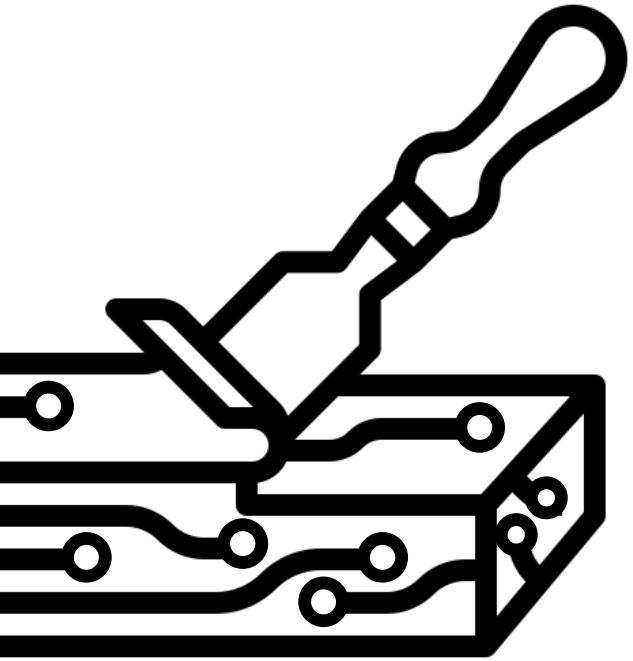
Can help when you really need all files (of a certain type).

Can be really time consuming.

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Record Carving



When you don't need the whole file:
data carving or record carving.

Only restore relevant parts or entries of a file (type).

E. g. Exif data, email addresses, …

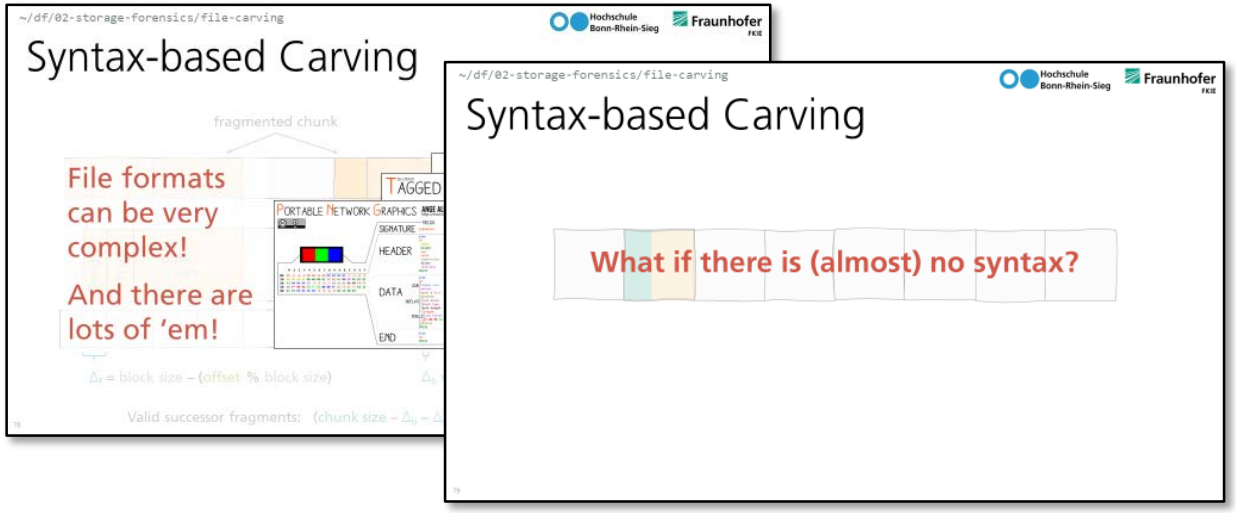
→ We'll have a look at `bulk_extractor` later on.

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Any Questions So Far?

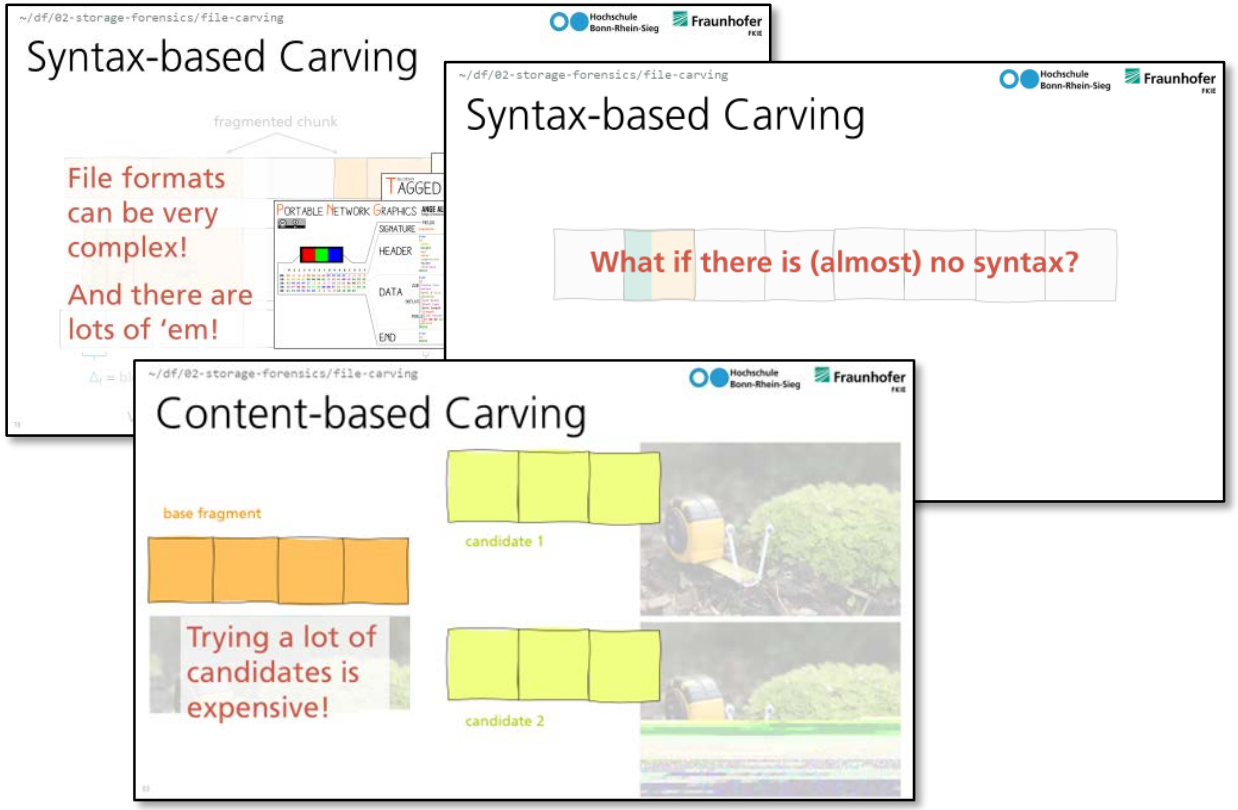
File Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

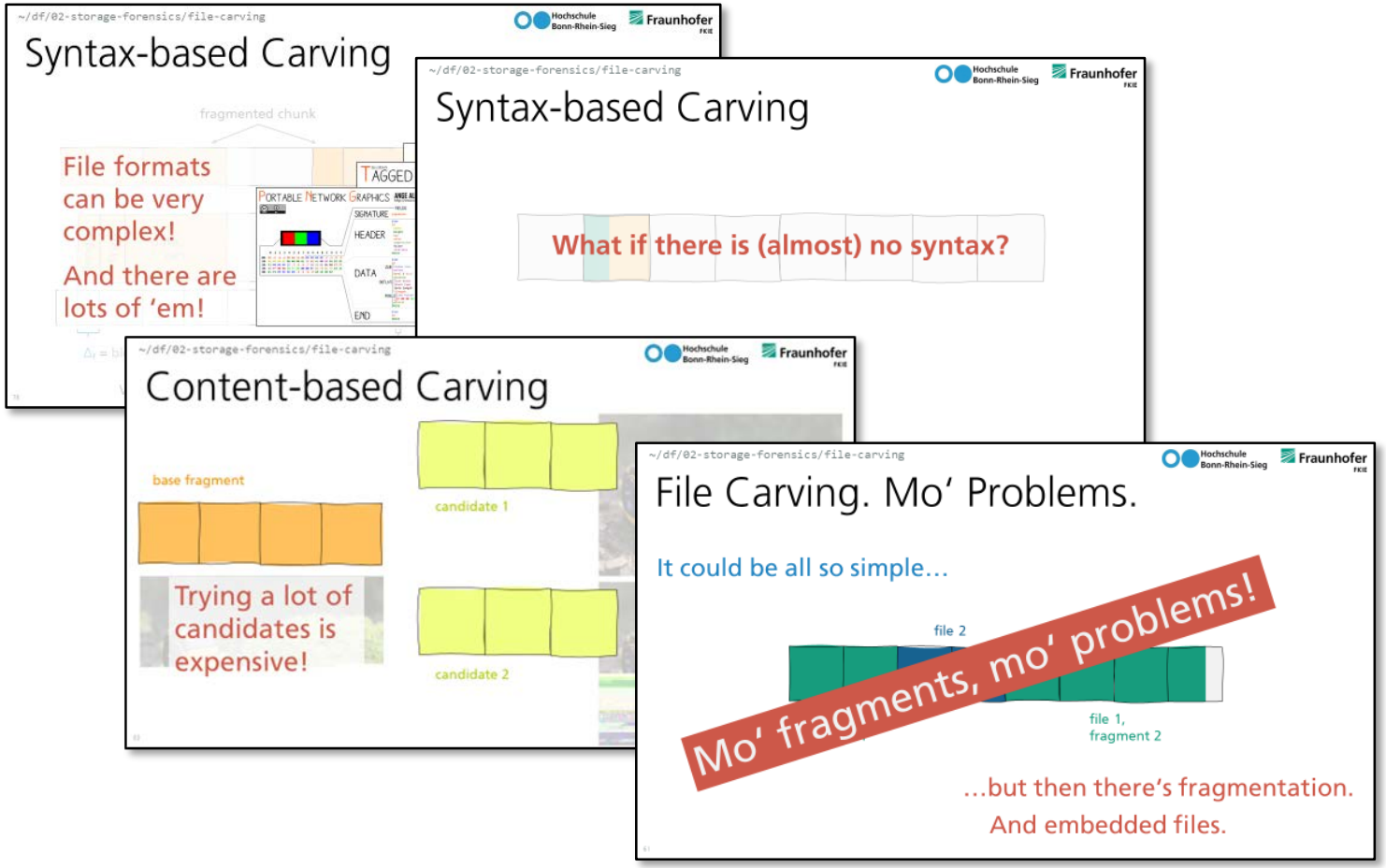
File Carving



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

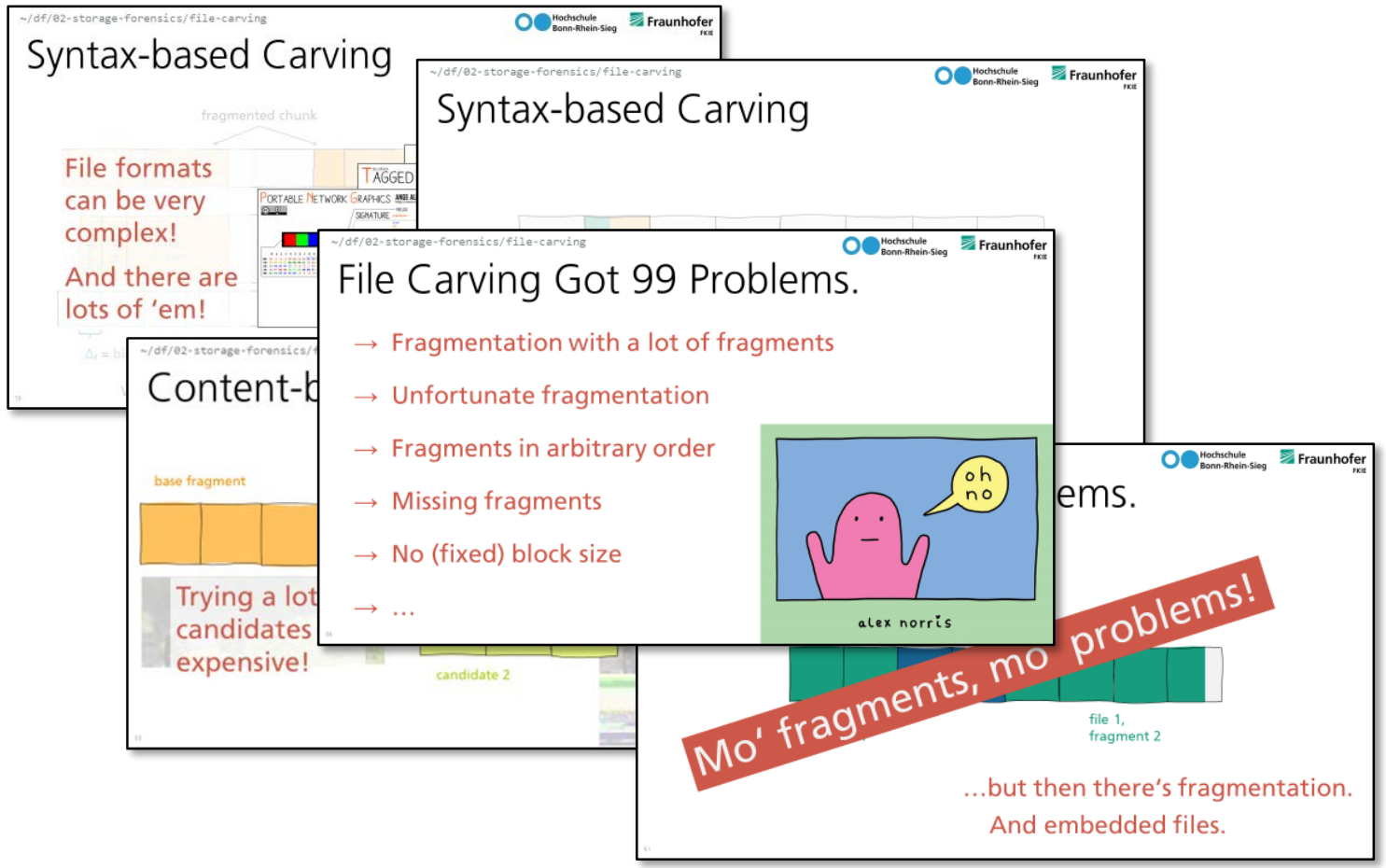
File Carving



Hochschule Bonn-Rhein-Sieg

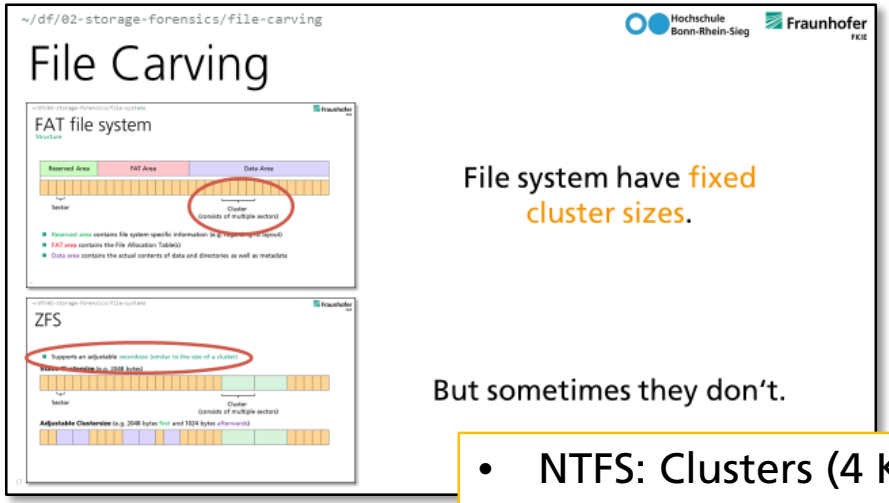
Fraunhofer FKIE

File Carving

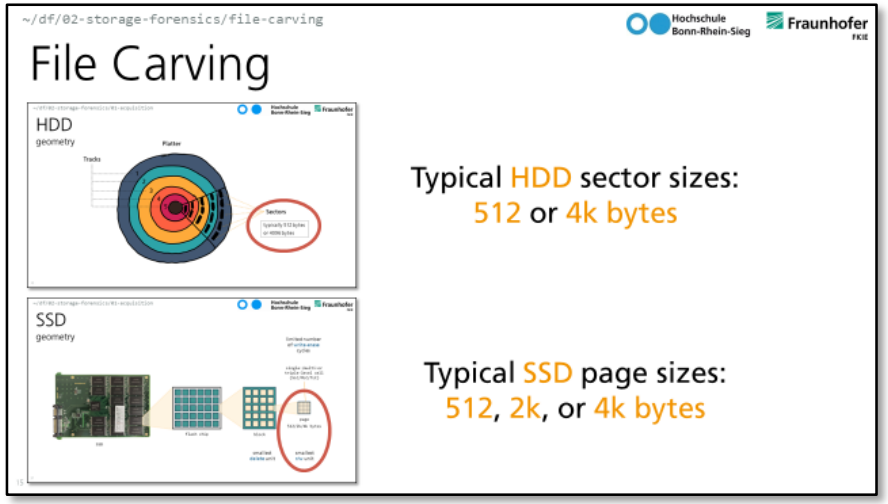


Data Units and Disk Sectors

- Modern file systems store files in *data units* of multiple disk sectors
  - Allocated *data units* aligned with disk sector boundaries



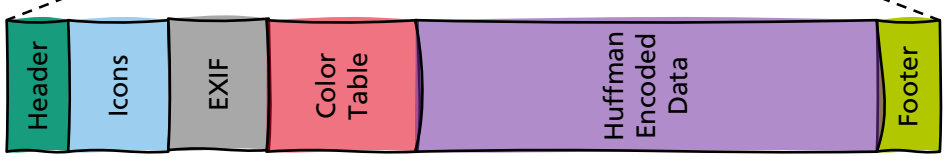
- NTFS: Clusters (4 KiB)
- Ext3: Blocks (1, 2, 4 KiB)
- Ext4: Blocks (4 KiB)
- …



Data Units and Disk Sectors

File Hash Values

carl.jpg



21893 Bytes

a87ca5123242111647b713630c8dcb97 (MD5)

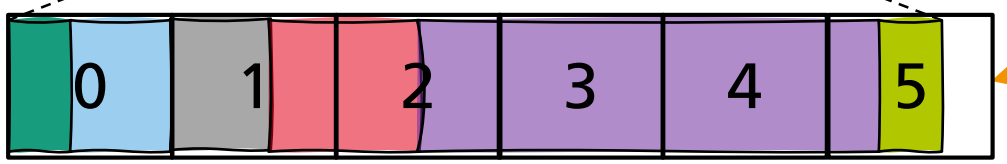
Clearly identifies image file!

Data Units and Disk Sectors

File Blocks

- Files can be viewed as sequence of data units

carl.jpg



With zero-padding!

21893 Bytes

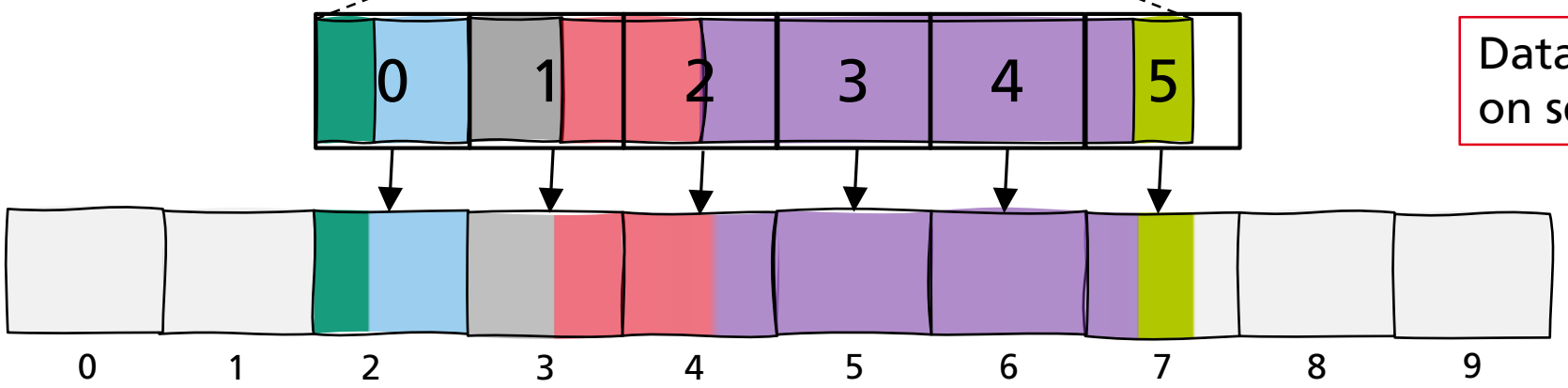
a87ca5123242111647b713630c8dcb97 (MD5)

Data Units and Disk Sectors

File Blocks and Sector Boundaries

- Data units are stored in (multiple consecutive) sectors

carl.jpg



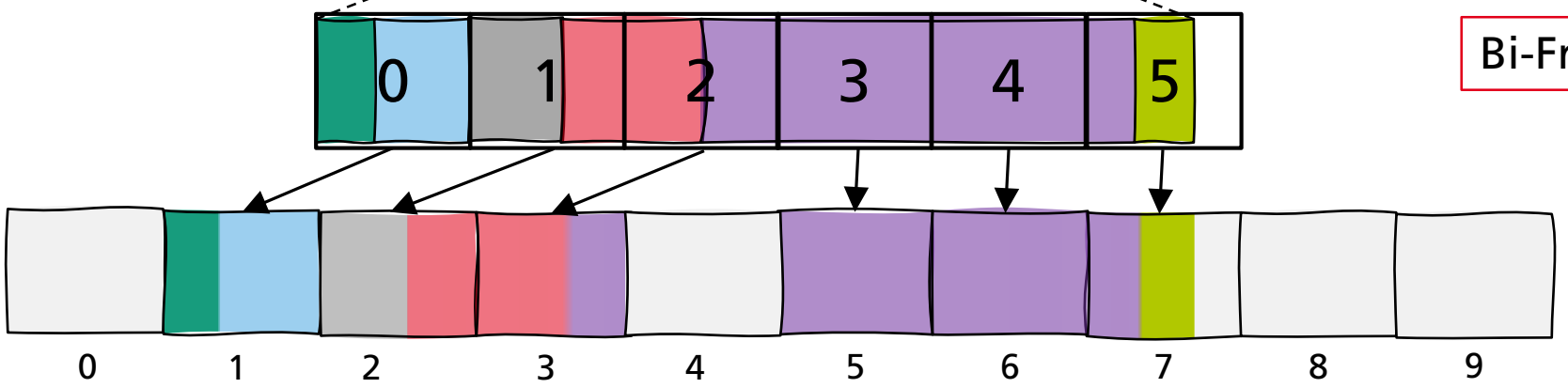
Data units aligned on sector boundaries

Data Units and Disk Sectors

Fragmentation

- Data units are stored in (multiple consecutive) sectors

carl.jpg

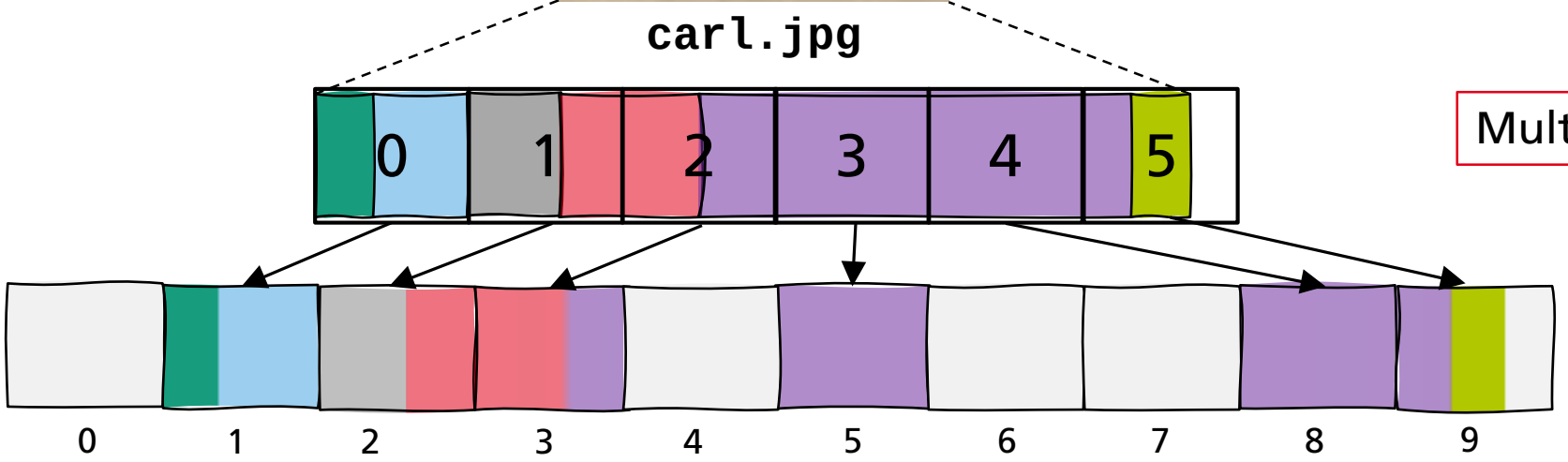


Bi-Fragmentation

Data Units and Disk Sectors

More Fragmentation

- Data units are stored in (multiple consecutive) sectors



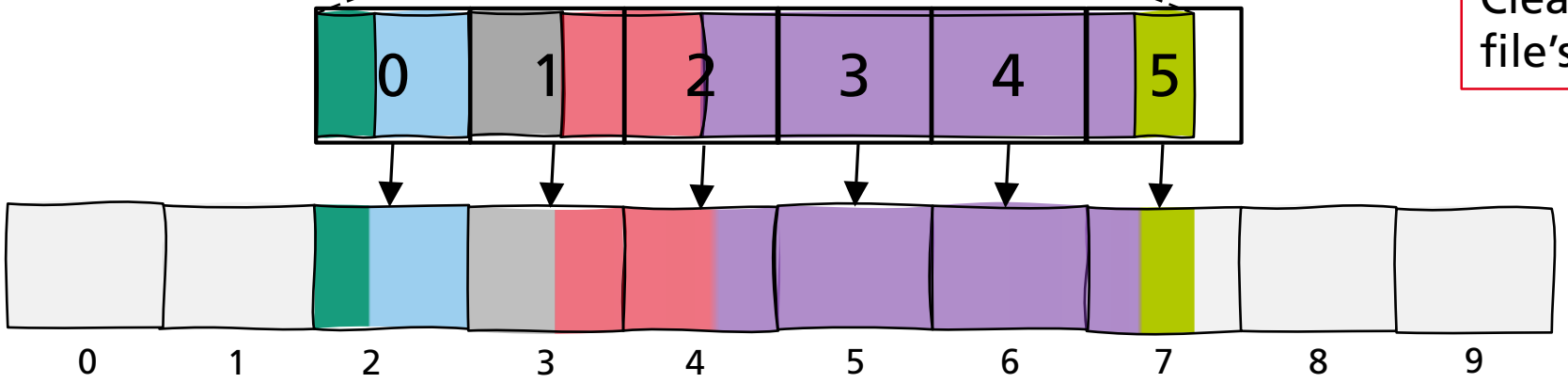
Multiple Fragments

Data Units and Disk Sectors

File Block Hash Values

- Hashing can be applied to each data unit

carl.jpg



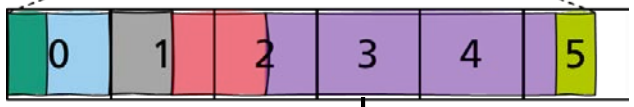
Block	*MD5 value*
0	cfcd208495d565ef66e7dff9f98764da
1	c4ca4238a0b923820dcc509a6f75849b
2	c81e728d9d4c2f636f067f89cc14862c
3	6d7fce9fee471194aa8b5b6e47267f03
…	…

Clearly identifies the file's data units!

Hash-based Carving

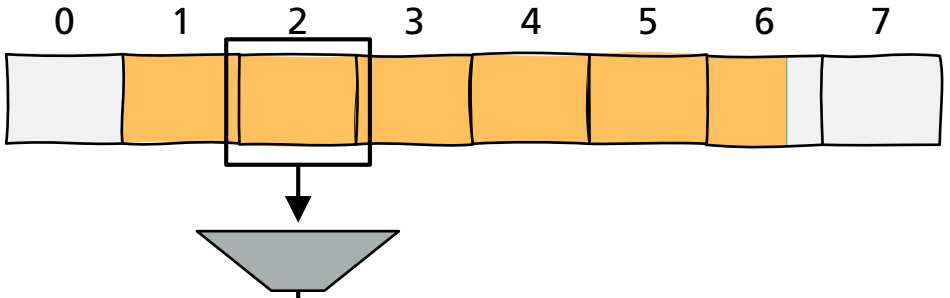
Idea

carl.jpg



(1) Compute hash values of fixed-size *file blocks* of target files

(2) Compute *sector hashes* of equal size on search media



(3) Search for matches between sector hashes and file block hashes

Block	*Byte range*	*MD5 value*
0	0 - 4095	cfcd208495d565ef66e7dff9f98764da
1	4096 - 8191	c4ca4238a0b923820dcc509a6f75849b
2	8192 - 12287	c81e728d9d4c2f636f067f89cc14862c
3	12288 - 16384	6d7fce9fee471194aa8b5b6e47267f03
…	…	…

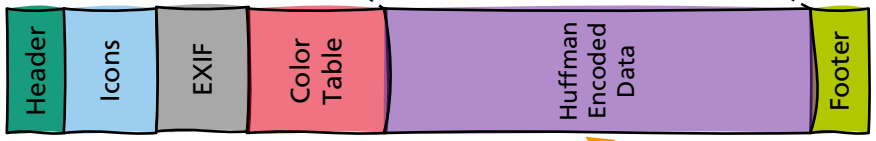
(4) Determine *file fragments* on search media & try to recover them!

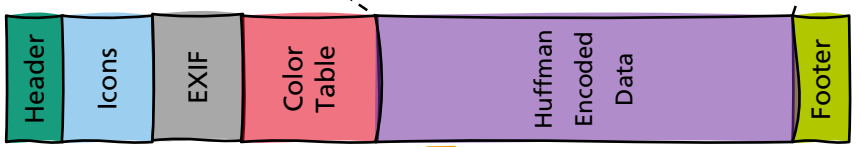
(Non-)Distinct File Blocks

Distinct Blocks

- Some blocks are likely distinct for each file

Only part of files directly maps to visible portion





Different files will have different Huffman encoded data

(Non-)Distinct File Blocks

Non-Distinct Blocks

- Other blocks might occur in more than one file

Only part of files directly maps to visible portion

EXIF and color tables are generated by camera and are likely the same.

(Non-)Distinct File Blocks

Distinct Blocks

> *Distinct file block:*
>
> file block which does not occur anywhere more than once except in a copy of the original file

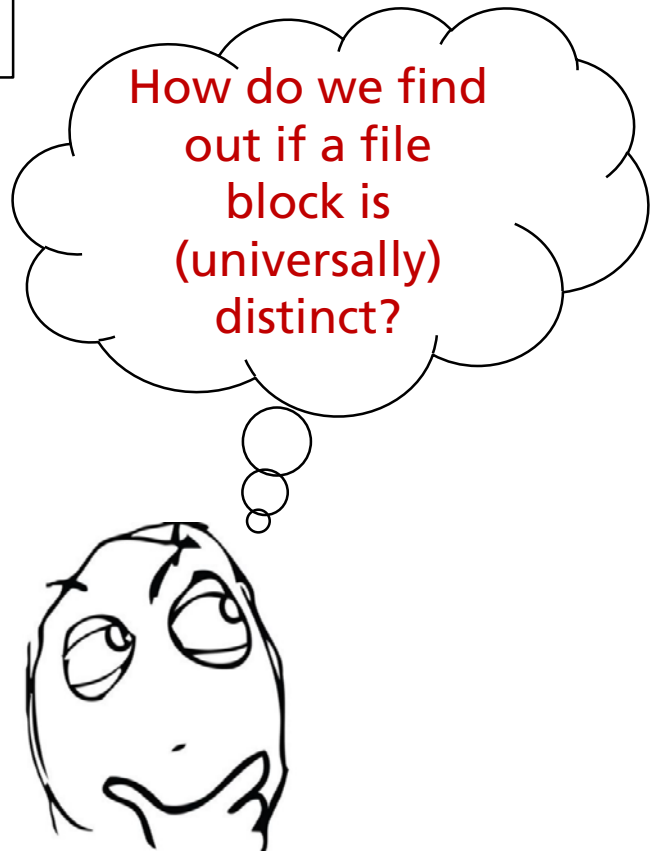
- In theory, hash-based carving lets us find *file fragments*
  - Fragmented files, files deleted and partially overwritten,…
  - Worst case: single file block
- Hash-based carving heavily relies on distinct file blocks
  - Distinct file block on search media strong evidence that file was present
  - Non-distinct blocks complicate file carving

(Non-)Distinct File Blocks

Distinct Blocks

> *Distinct file block:*
>
> file block which does not occur anywhere more than once except in a copy of the original file

- In theory, hash-based carving lets us find *file fragments*
  - Fragmented files, files deleted and partially overwritten,…
  - Worst case: single file block
- Hash-based carving heavily relies on distinct file blocks
  - Distinct file block on search media strong evidence that file was present
  - Non-distinct blocks complicate file carving



(Non-)Distinct File Blocks

Block Frequencies

How do we find out if a file block is (universally) distinct?

- Compare large file datasets (GOVDOCS, OpenMalware 2012, NSRL RDS) to find *singleton*, *pair*, *common* blocks

Table 1. Incidence of singleton, paired, and common sectors in three file corpora.

No. of blocks	Govdocs	OpenMalware 2012	2009 NSRL RDS
Block size: 512 bytes			
Singleton	911.4 M (98.93%)	1,063.1 M (88.69%)	N/A
Pair	7.1 M (.77%)	75.5 M (6.30%)	N/A
Common	2.7 M (.29%)	60.0 M (5.01%)	N/A
Block size: 4 kibibytes			
Singleton	117.2 M (99.46%)	143.8 M (89.51%)	567.0 M (96.00%)
Pair	0.5 M (.44%)	9.3 M (5.79%)	16.4 M (2.79%)
Common	0.1 M (.11%)	7.6 M (4.71%)	7.1 M (1.21%)

[Young, Foster, Garfinkel, Fairbanks. Distinct Sector Hashes for Target File Detection, *2012*]

(Non-)Distinct File Blocks

Block Frequencies

How do we know if a file block is (universally) distinct?

- Compare large file datasets (GOVDOCS, OpenMalware 2012, NSRL RDS) to find *singleton*, *pair*, *common* blocks

All kinds of user-generated content

- Photos
- Videos

Usually high in entropy

Table 1. Incidence of singleton, paired, and common sectors in three file corpora.

No. of blocks	Govdocs	OpenMalware 2012	2009 NSRL RDS
Block size: 512 bytes			
Singleton	911.4 M (98.93%)	1,063.1 M (88.69%)	N/A
Pair	7.1 M (.77%)	75.5 M (6.30%)	N/A
Common	2.7 M (.29%)	60.0 M (5.01%)	N/A
Block size: 4 kibibytes			
Singleton	117.2 M (99.46%)	143.8 M (89.51%)	567.0 M (96.00%)
Pair	0.5 M (.44%)	9.3 M (5.79%)	16.4 M (2.79%)
Common	0.1 M (.11%)	7.6 M (4.71%)	7.1 M (1.21%)

[Young, Foster, Garfinkel, Fairbanks. Distinct Sector Hashes for Target File Detection, *2012*]

(Non-)Distinct File Blocks

Block Frequencies

How do we know if a file block is (universally) distinct?

- Compare large file datasets (GOVDOCS, OpenMalware 2012, NSRL RDS) to find *singleton*, *pair*, *common* blocks

All kinds of user-generated content

- Photos
- Videos

Usually high in entropy

Table 1. Incidence of singleton, paired, and common sectors in three file corpora.

No. of blocks	Govdocs	OpenMalware 2012	2009 NSRL RDS
Block size: 512 bytes			
Singleton	911.4 M (98.93%)	1,063.1 M (88.69%)	N/A
Pair	7.1 M (.77%)	75.5 M (6.30%)	N/A
Common	2.7 M (.29%)	60.0 M (5.01%)	N/A
Block size: 4 kibibytes			
Singleton	117.2 M (99.46%)	143.8 M (89.51%)	567.0 M (96.00%)
Pair	0.5 M (.44%)	9.3 M (5.79%)	16.4 M (2.79%)
Common	0.1 M (.11%)	7.6 M (4.71%)	7.1 M (1.21%)

[Young, Foster, Garfinkel, Fairbanks. Distinct Sector Hashes for Target File Detection, *2012*]

Blocks with repeating byte patterns

- NUL-bytes (0x00)
- Internal data structures (EXIF, Microsoft SAT,…)

Usually low in entropy

(Non-)Distinct File Blocks

4-byte Ramp Pattern

```
8102 0000 8202 0000 8302 0000 8402 0000
8502 0000 8602 0000 8702 0000 8802 0000
8902 0000 8a02 0000 8b02 0000 8c02 0000
8d02 0000 8e02 0000 8f02 0000 9002 0000
```

64 bytes from exemplary .xls-file shows the “ramp” structure of the Microsoft Compound Document File Allocation Table.

[Garfinkel, McCarrin. Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb, 2015.]

- Incrementing 4-byte integers in *File Allocation Tables (FATs)* of the *Microsoft Compound Document Format*
- Low chance of match between any specific MS Office files…
  - … but high chance of matches between large sets of office files

(Non-)Distinct File Blocks

Identify Common Blocks

- Determining if blocks are universally distinct is difficult…
  - Even though blocks are *locally distinct* in large database, chance matches to other files on search media occur
  - The larger the database, the more common blocks we discover

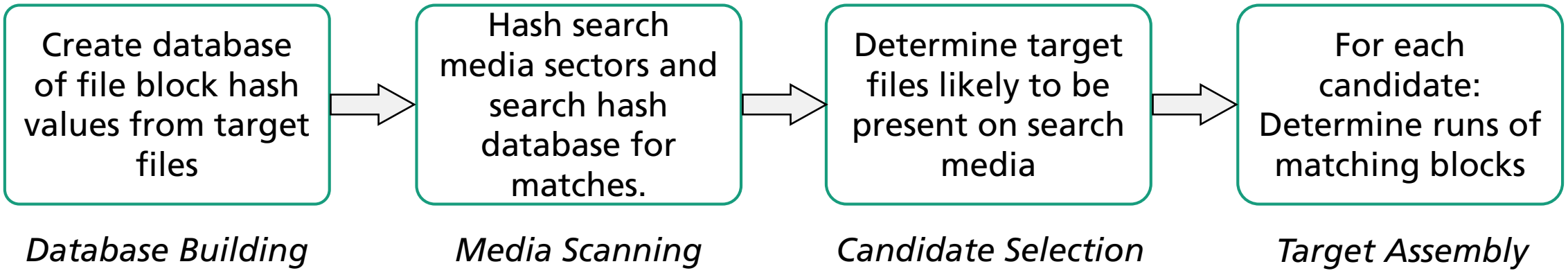
- Legitimate reasons for the same block appearing multiple times
  - E.g. different versions of the same video (extended and normal)

More reasonable: Identify *common blocks* using *frequency* and *byte patterns*

Hash-based Carving

Carving Process

Identify target files on search media via hash matches between sector hashes and fixed-size file block hashes



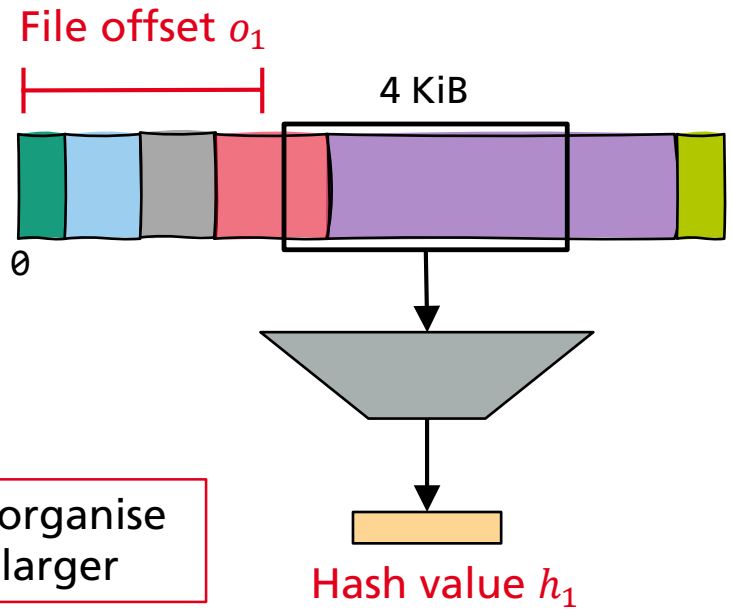
Tools:

- `hashdb` – High-speed hash database (100k lookups/s)
- `bulk_extractor` – Open source tool for media scanning (Optional)

Database Building

- Compute hash values of each 4 KiB target file block and store values in hashdb database

```
~$ bulk_extractor -E hashdb -S hasdb_mode=import –o targets.hdb carl.jpg
```



Most file systems organise in 4 KiB blocks or larger

Hash-value h_i -> (Source file, file offset o_i)

Key	*Value*
h_0	(carl.jpg, o_0)
h_1	(carl.jpg, o_1)
	…

Key-value store

targets.hdb

Note: Internally hashdb uses *file ids* to identify files. We are using the file name just for demonstration purposes.

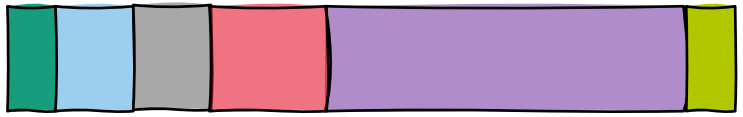
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

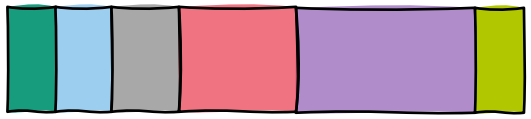
Database Building

Example

carl.jpg



racoon.jpg



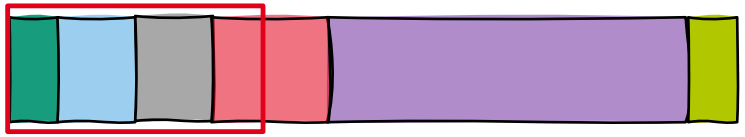
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

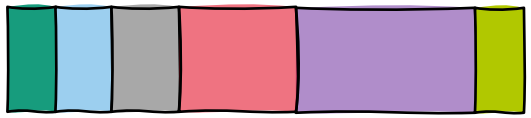
Database Building

Example

carl.jpg



racoon.jpg



12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)

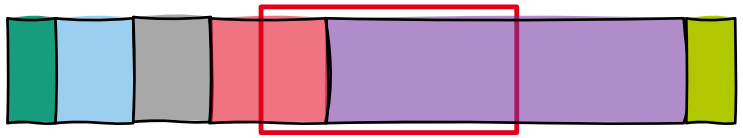
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

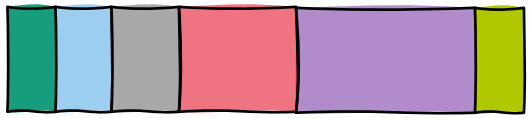
Database Building

Example

carl.jpg



racoon.jpg



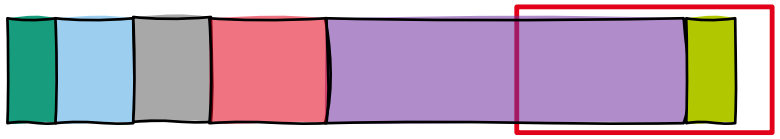
```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
```

Hochschule Bonn-Rhein-Sieg

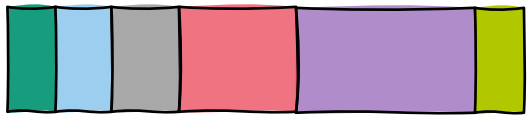
Fraunhofer FKIE

Database Building

carl.jpg



racoon.jpg



```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
c7698f7bae9378423a34e46a25852e6a (1, carl.jpg, 8192)
```

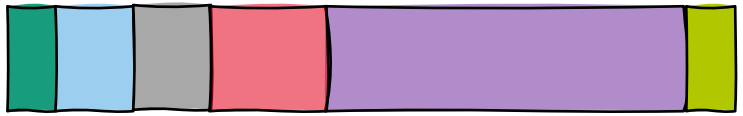
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

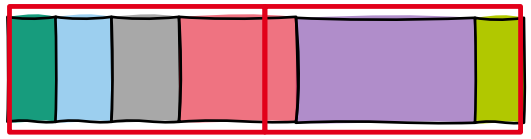
Database Building

Example

carl.jpg



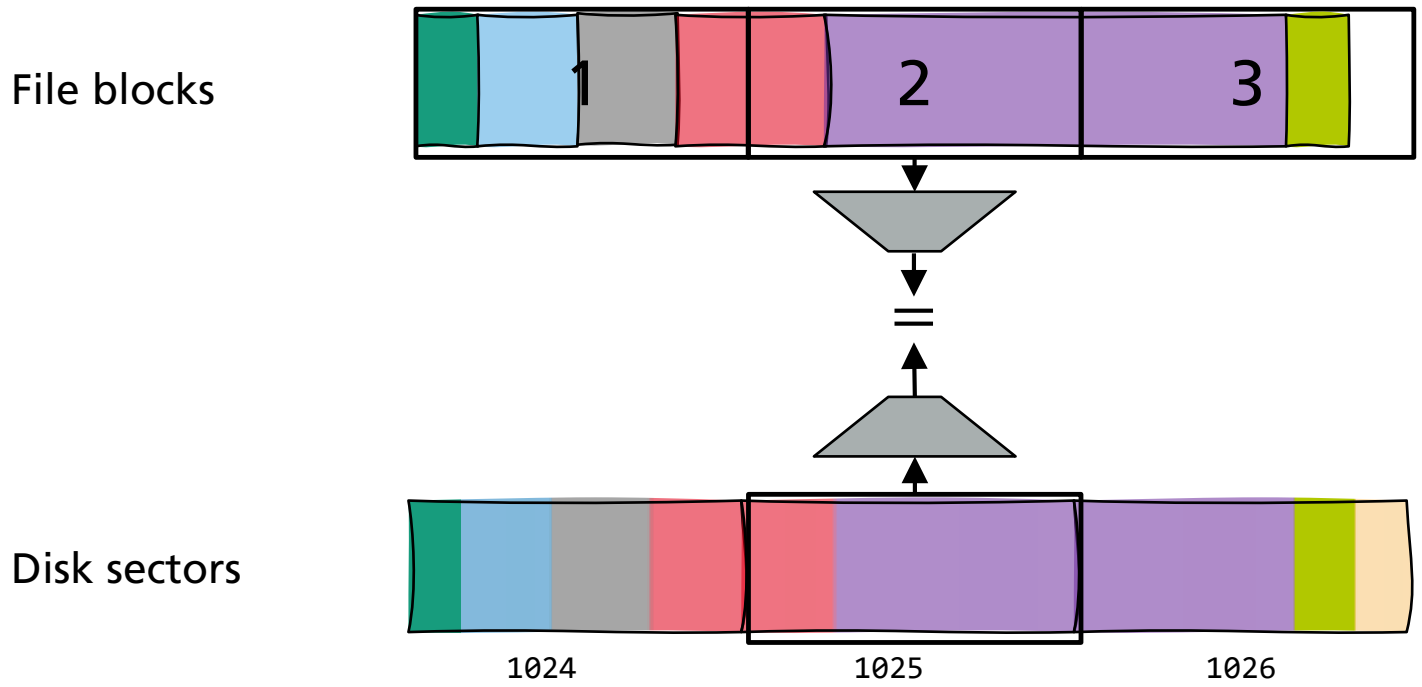
racoon.jpg



```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
c7698f7bae9378423a34e46a25852e6a (1, carl.jpg, 8192)
fed73beb2ed647c4f5ebcaf550146b91 (2, racoon.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (2, racoon.jpg, 4096)
```

Sector Size and Alignment Issues

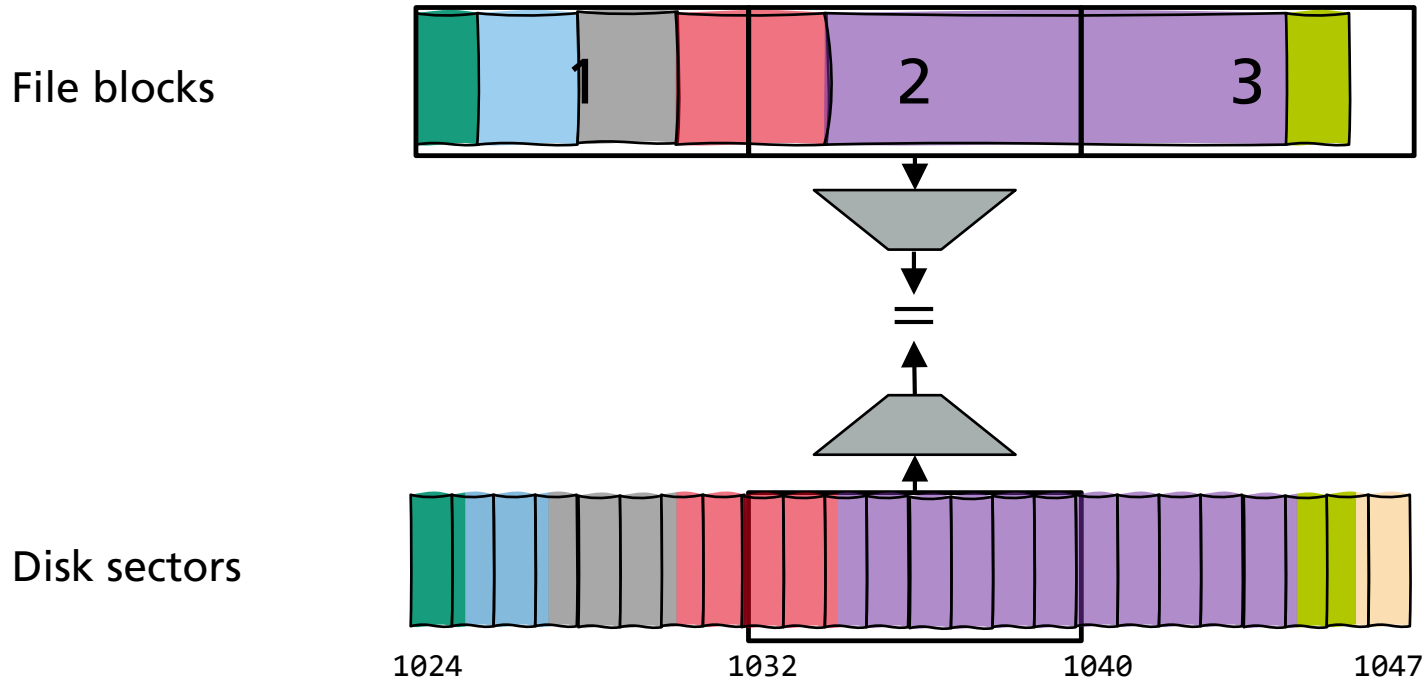
- Hash 4 KiB blocks of search media sectors and search hash database for matches
- Sector hashes need to be aligned with file block hashes
  - True if hashed sectors aligned with fs allocation blocks



In case of 4 KiB sector size:
Automatically aligned

Sector Size and Alignment Issues

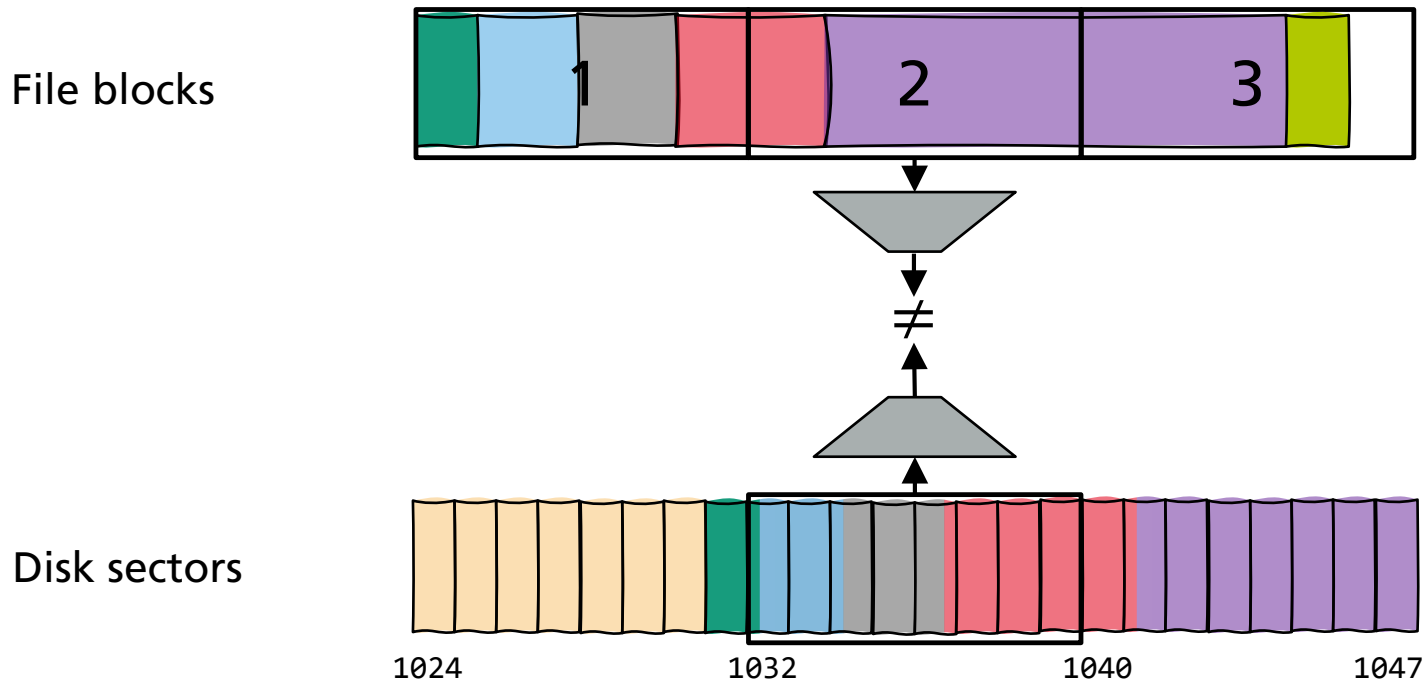
- Hash 4 KiB blocks of search media sectors and search hash database for matches
- Sector hashes need to be aligned with file block hashes
  - True if hashed sectors aligned with fs allocation blocks



In case of 512 bytes sector size:
Not necessarily aligned

Sector Size and Alignment Issues

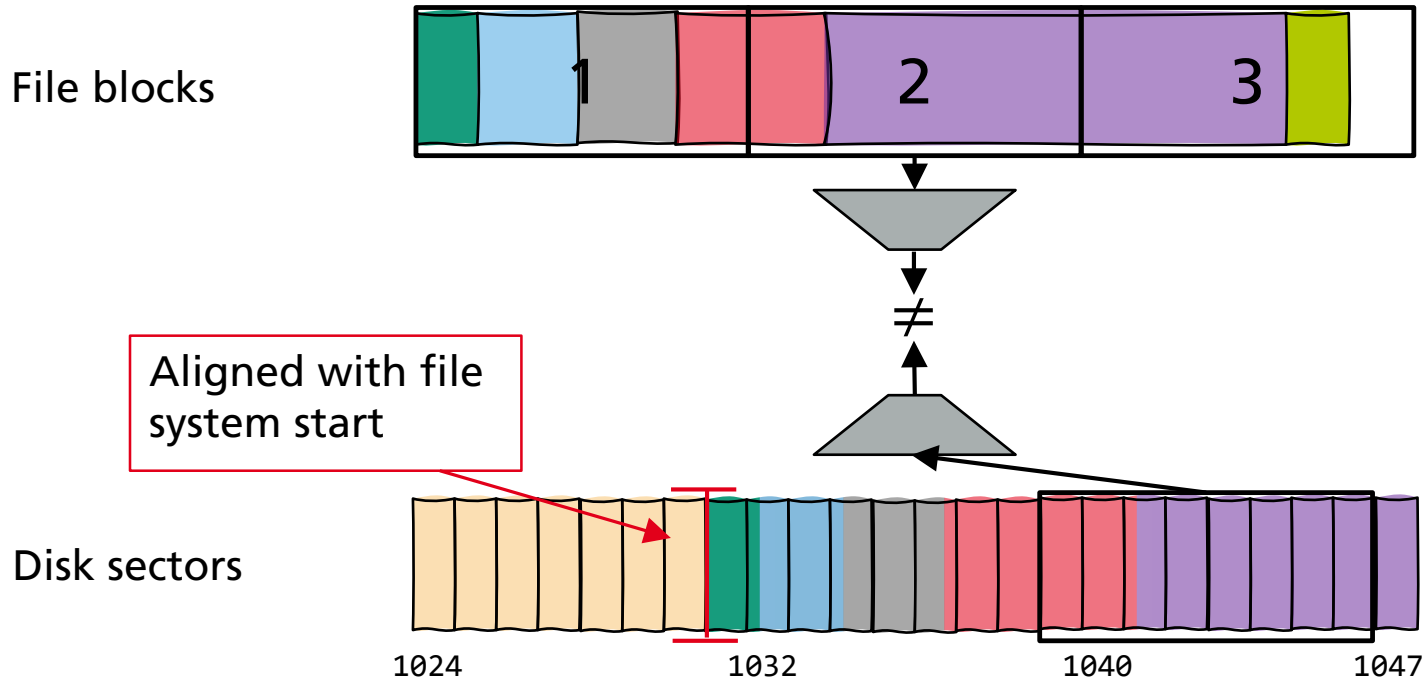
- Hash 4 KiB blocks of search media sectors and search hash database for matches
- Sector hashes need to be aligned with file block hashes
  - True if hashed sectors aligned with fs allocation blocks



In case of 512 bytes sector size:
Not necessarily aligned

Sector Size and Alignment Issues

- Hash 4 KiB blocks of search media sectors and search hash database for matches
- Sector hashes need to be aligned with file block hashes
  - True if hashed sectors aligned with fs allocation blocks



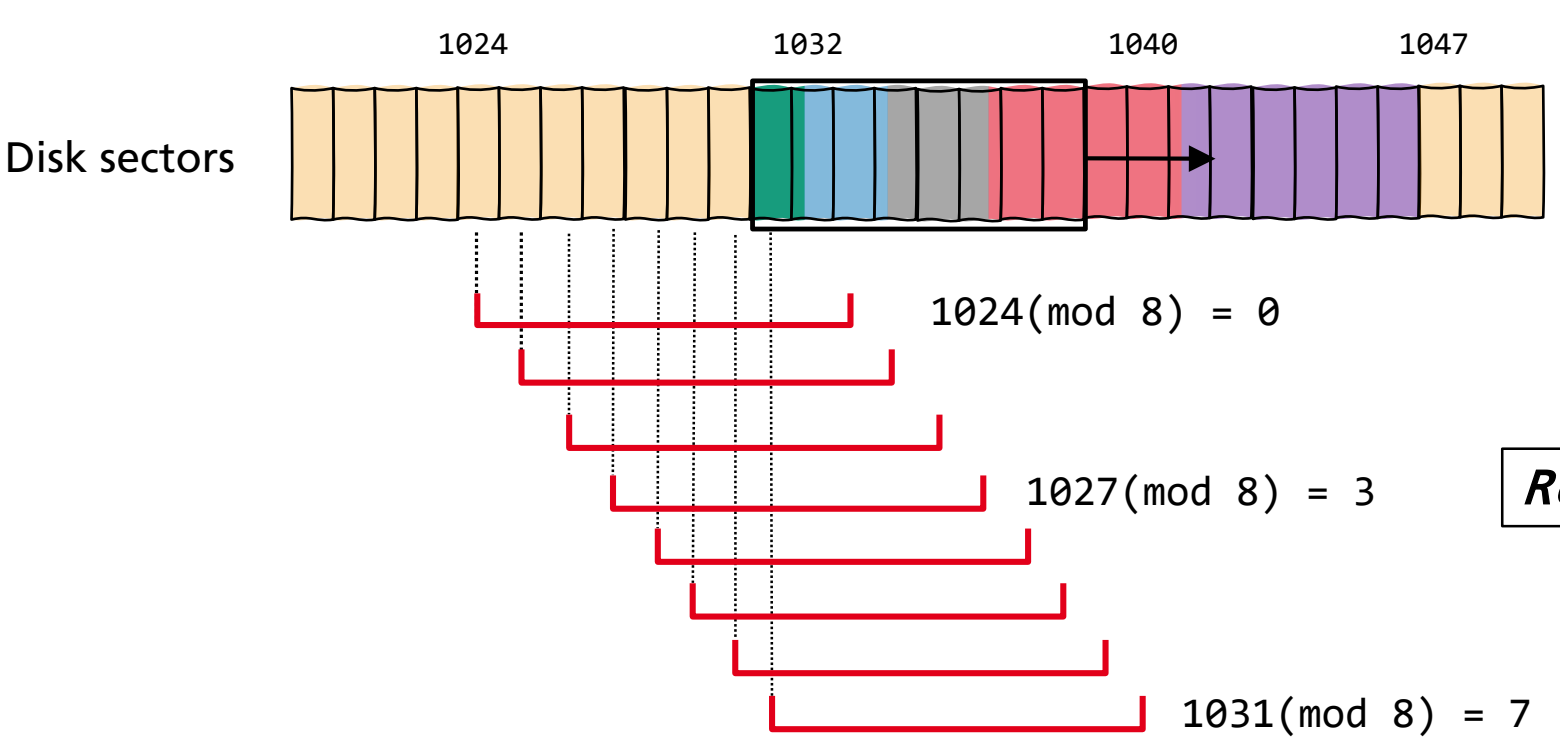
In case of 512 bytes sector size:
Not necessarily aligned

Align sector hashes with file system start

But what if file system starting point is unknown or untrustworthy?

Sector Size and Alignment Issues

- In case of 512 byte sectors: Use 4 KiB sliding window for sector hashes, moving 1 sector at a time



Sliding window accounts for different file system offsets

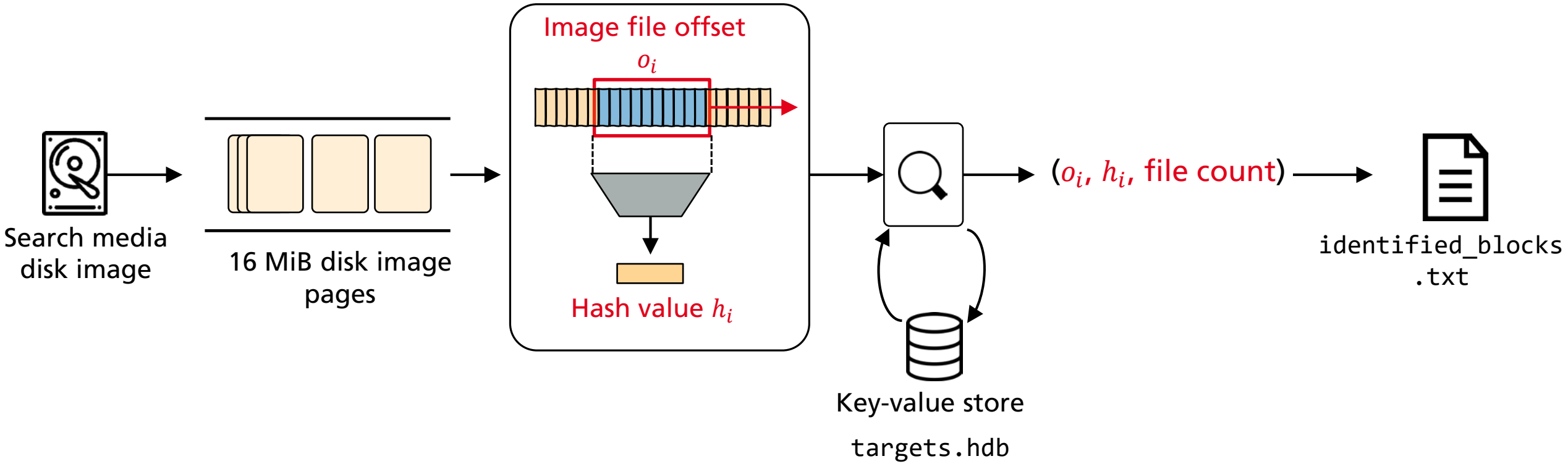
Same file system offset → Same sector number (mod 8)

Result: 8 distinct sets of sector hashes

Media Scanning

- Hash disk sectors of search media with 4 KiB sliding window and search file block hash database for matches

```
~$ bulk_extractor -S hashdb_mode=scan -S hashdb_scan_path_or_socket=targets.hdb -E hashdb
-o identified_blocks.txt suspect_drive_image.E01
```



Hochschule Bonn-Rhein-Sieg

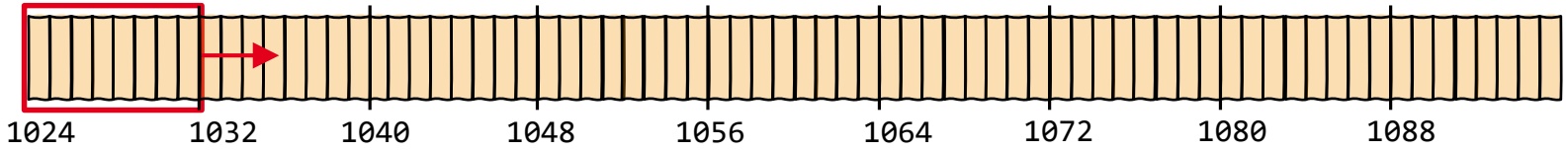
Fraunhofer FKIE

Media Scanning

Example Continued

```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
c7698f7bae9378423a34e46a25852e6a (1, carl.jpg, 8192)
fed73beb2ed647c4f5ebcaf550146b91 (2, racoon.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (2, racoon.jpg, 4096)
```

Search media image

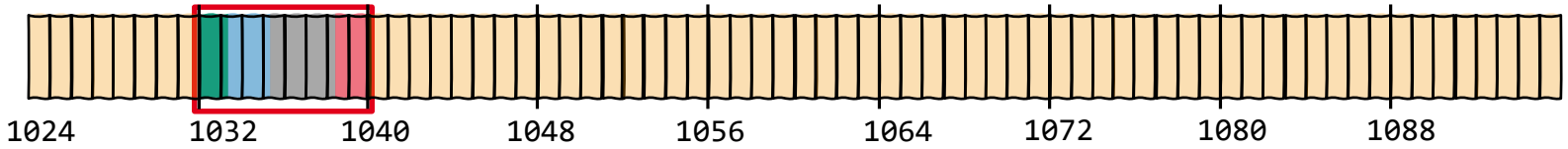


Media Scanning

Example Continued

```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
c7698f7bae9378423a34e46a25852e6a (1, carl.jpg, 8192)
fed73beb2ed647c4f5ebcaf550146b91 (2, racoon.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (2, racoon.jpg, 4096)
```

Search media image



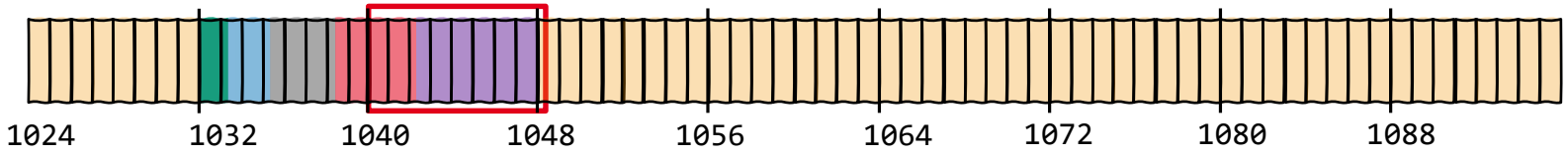

```
1032 12470fe406d44017d96eab37dd65fc14 count: 1
```

Media Scanning

Example Continued

```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
c7698f7bae9378423a34e46a25852e6a (1, carl.jpg, 8192)
fed73beb2ed647c4f5ebcaf550146b91 (2, racoon.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (2, racoon.jpg, 4096)
```

Search media image




```
1032 12470fe406d44017d96eab37dd65fc14 count: 1
1040 e5a93371cfc7eab4a88221dd1f6c1a3c count: 1
```

Hochschule Bonn-Rhein-Sieg

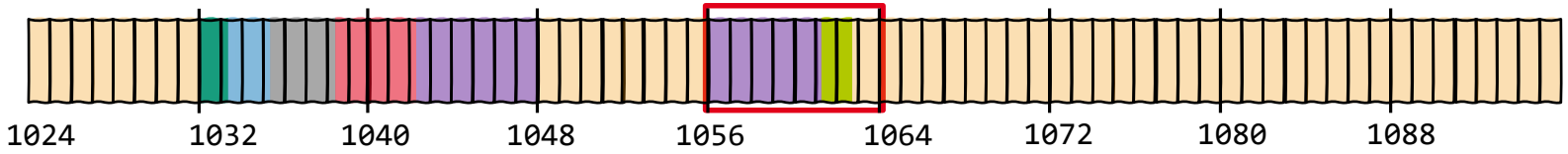
Fraunhofer FKIE

Media Scanning

Example Continued

```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
c7698f7bae9378423a34e46a25852e6a (1, carl.jpg, 8192)
fed73beb2ed647c4f5ebcaf550146b91 (2, racoon.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (2, racoon.jpg, 4096)
```

Search media image




```
1032 12470fe406d44017d96eab37dd65fc14 count: 1
1040 e5a93371cfc7eab4a88221dd1f6c1a3c count: 1
1056 c7698f7bae9378423a34e46a25852e6a count: 1
```

Hochschule Bonn-Rhein-Sieg

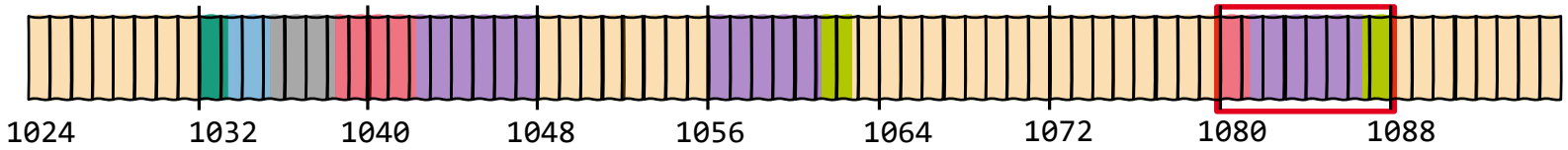
Fraunhofer FKIE

Media Scanning

Example Continued

```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
c7698f7bae9378423a34e46a25852e6a (1, carl.jpg, 8192)
fed73beb2ed647c4f5ebcaf550146b91 (2, racoon.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (2, racoon.jpg, 4096)
```

Search media image




```
1032 12470fe406d44017d96eab37dd65fc14 count: 1
1040 e5a93371cfc7eab4a88221dd1f6c1a3c count: 1
1056 c7698f7bae9378423a34e46a25852e6a count: 1
1080 e5a93371cfc7eab4a88221dd1f6c1a3c count: 1
```

Candidate Selection

Probative Blocks

- Identify candidate files likely to be present on search media
  - Remember*: Common blocks* make file reassembly harder

> **Assumption**: File was present if we find a single distinct block

- Some blocks are *locally distinct* in target dataset, but are built from common structures
  - Incrementing binary numbers (Microsoft FAT), Whitespaces,…

- Canidiate selection requires to filter-out *non-probative* blocks
  - Evaluate *locally distinct blocks* if they provide *byte patterns* and *common structures*

Testing for Probative Blocks

Ramp Test

- Filter-out blocks with incrementing 4-byte binary numbers

```
8102 0000 8202 0000 8302 0000 8402 0000
8502 0000 8602 0000 8702 0000 8802 0000
8902 0000 8a02 0000 8b02 0000 8c02 0000
8d02 0000 8e02 0000 8f02 0000 9002 0000
```

64 bytes from exemplary .xls-file shows the “ramp” structure of the Microsoft Compound Document File Allocation Table.

[Garfinkel, McCarrin. Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb, 2015.]

- Microsoft Compound Document File Allocation Tables (FATs)
  - High chance of collision within few thousand MS Office files

True if 50% of the file block bytes match the ramp pattern

Testing for Probative Blocks

Whitespace Test

- Blank lines of 100 spaces, terminated by newline character

```
0000000: 2020 2020 2020 2020 2020 2020 2020 2020
0000010: 2020 2020 2020 2020 2020 2020 0a20 2020
0000020: 2020 2020 2020 2020 2020 2020 2020 2020
0000030: 2020 2020 2020 2020 2020 2020 2020 2020
0000040: 2020 2020 2020 2020 2020 2020 2020 2020
0000050: 2020 2020 2020 2020 2020 2020 2020 2020
0000060: 2020 2020 2020 2020 2020 2020 2020 2020
0000070: 2020 2020 2020 2020 2020 2020 2020 2020
0000080: 200a 2020 2020 2020 2020 2020 2020 2020
0000090: 2020 2020 2020 2020 2020 2020 2020 2020
```

XMP section of a JPEG file containing whitespaces used for padding.

[Garfinkel, McCarrin. Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb, 2015.]

- Commonly found in XMP sections of JPEG files produced with Adobe Photoshop

True if 75% of the file block bytes contain whitespaces

Testing for Probative Blocks

Histogram Test

- Blocks with repeating or alternating 4-byte values

```
0000 6400 0000 01ff ffff 9c00 0000 0100
0000 6400 0000 01ff ffff 9c00 0000 0200
0000 0000 0000 0100 0000 6400 0000 01ff
ffff 9c00 0000 0100 0000 6400 0000 01ff
ffff 9c00 0000 0100 0000 6400 0000 01ff
ffff 9c00 0000 0100 0000 6400 0000 01ff
ffff 9c00 0000 0100 0000 6400 0000 01ff
ffff 9c00 0000 0100 0000 6400 0000 01ff
```

Repeating byte pattern found in a QuickTime file and a Powerpoint file.

[Garfinkel, McCarrin. Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb, 2015.]

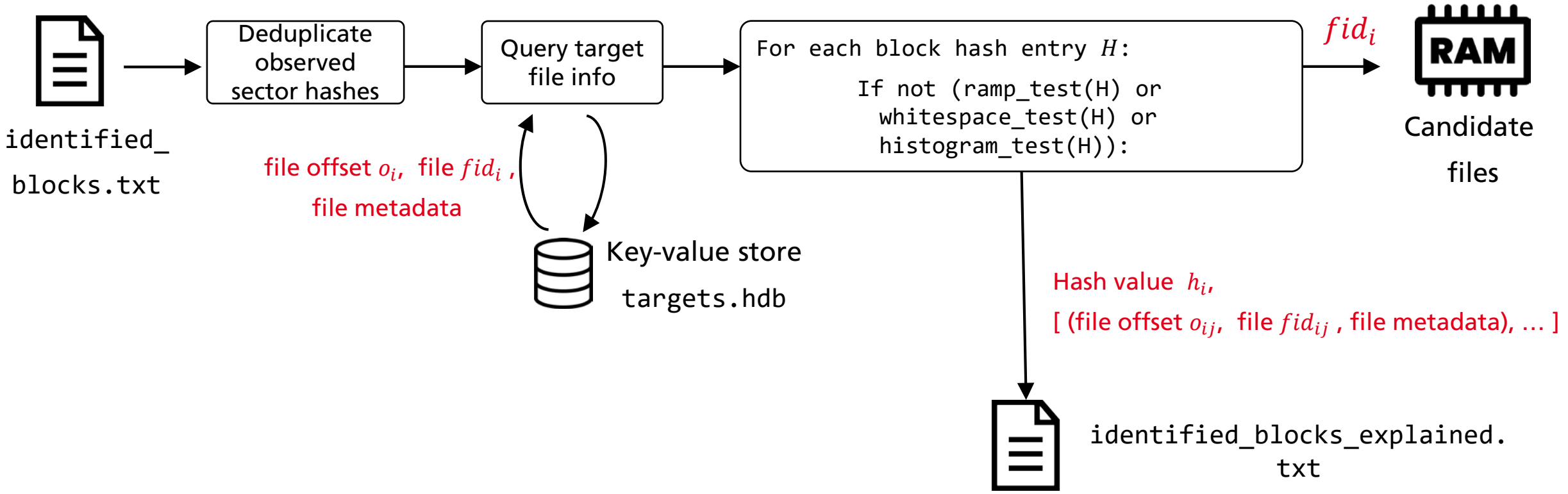
- Compute histogram of 1024 4-byte integers in 4 KiB file blocks

True if any 4-gram takes more than 25% of the block

Candidate Selection

- Identify candidate files likely to be present on search media

```
~$ hashdb explain_identified_blocks identified_blocks.txt > identified_blocks_explained.txt
```



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Candidate Selection

Example Continued

```
1032 12470fe406d44017d96eab37dd65fc14 count: 1
1040 e5a93371cfc7eab4a88221dd1f6c1a3c count: 1
1056 c7698f7bae9378423a34e46a25852e6a count: 1
1080 e5a93371cfc7eab4a88221dd1f6c1a3c count: 1
```

```
12470fe406d44017d96eab37dd65fc14 (1, carl.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (1, carl.jpg, 4096)
c7698f7bae9378423a34e46a25852e6a (1, carl.jpg, 8192)
fed73beb2ed647c4f5ebcaf550146b91 (2, racoon.jpg, 0)
e5a93371cfc7eab4a88221dd1f6c1a3c (2, racoon.jpg, 4096)
```

```
12470fe406d44017d96eab37dd65fc14 count: 1
e5a93371cfc7eab4a88221dd1f6c1a3c count: 1
c7698f7bae9378423a34e46a25852e6a count: 1
e5a93371cfc7eab4a88221dd1f6c1a3c count: 1
```

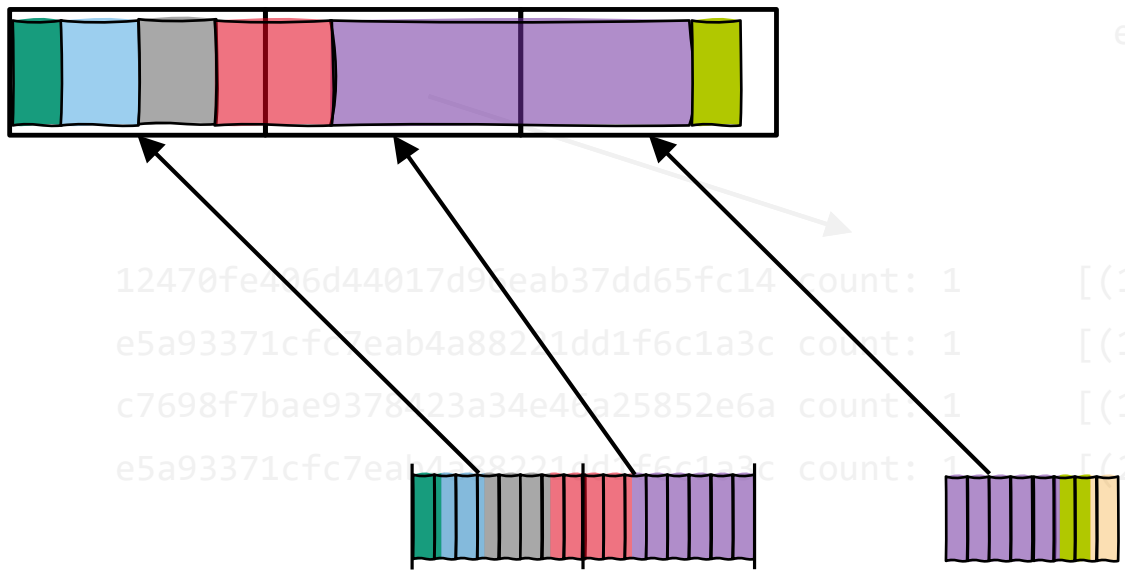
```
[(1, carl.jpg, 0, metadata)]
[(1, carl.jpg, 4096, metadata)]
[(1, carl.jpg, 8192, metadata)]
[(2, racoon.jpg, 4096, metadata)]
```

Candidate Selection

Example Continued

carl.jpg

racoon.jpg

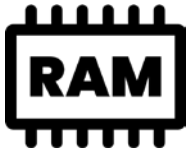


Candidate Selection

Example Continued

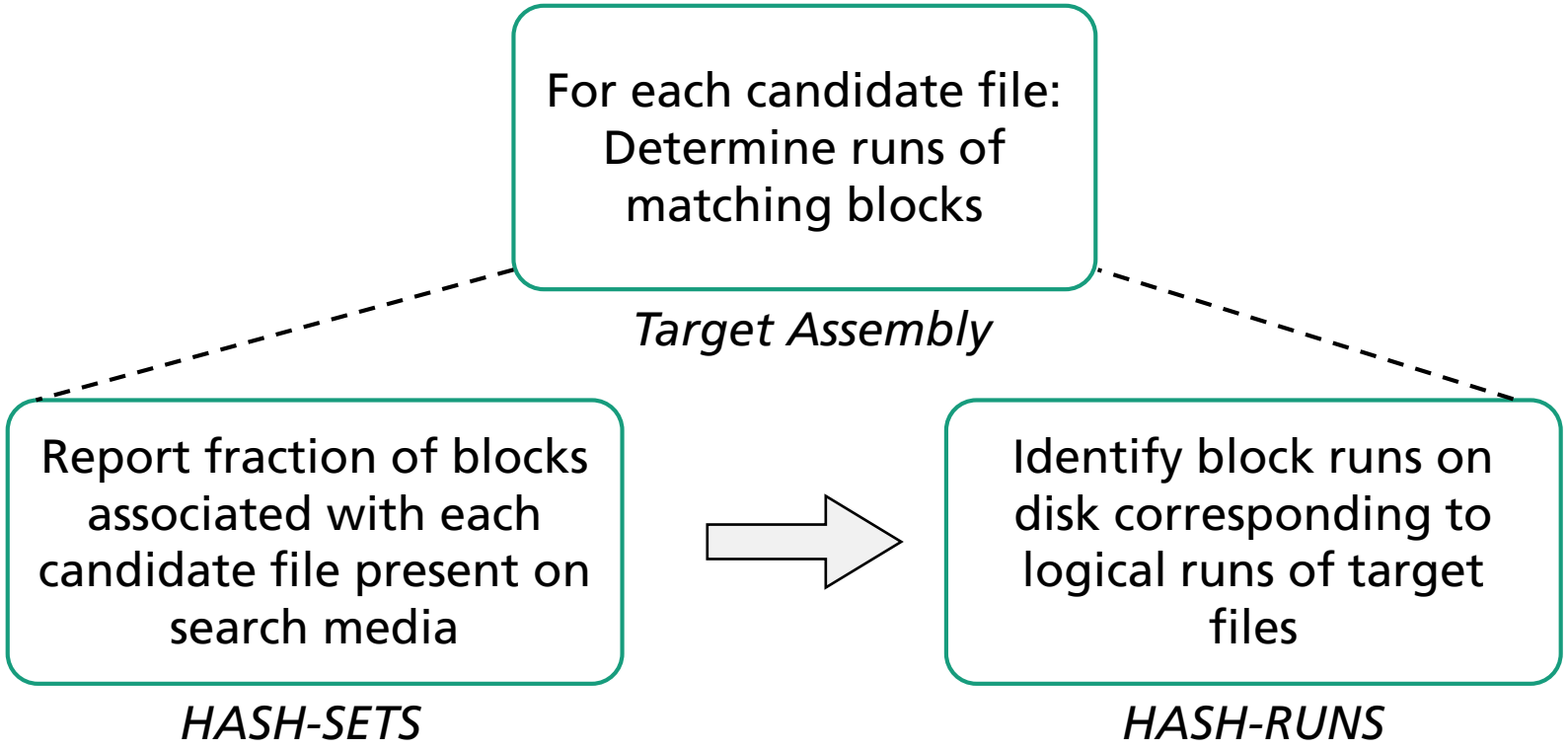
```
12470fe406d44017d96eab37dd65fc14 count: 1     [(1, carl.jpg, 0, metadata)]
e5a93371cfc7eab4a88221dd1f6c1a3c count: 1     [(1, carl.jpg, 4096, metadata)]
c7698f7bae9378423a34e46a25852e6a count: 1     [(1, carl.jpg, 8192, metadata)]
e5a93371cfc7eab4a88221dd1f6c1a3c count: 1     [(2, racoon.jpg, 4096, metadata)]
```

(1, carl.jpg, metadata)
(2, racoon.jpg, metadata)



Target Assembly

- After candidate files have been selected…



HASH-SETS

Hash value h_i,

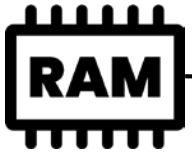
[(target file fid_{ij}, target file offset o_{ij}, metadata), …]

identified_blocks
_explained.txt

```
For each block hash entry H:
    For each target file fid_ij that matched:
```

fid_{ij}

```
If file fid_ij in candidate_files:
    (fid_i, score + 1, metadata) with
    fid_i == fid_ij
```



Candidate files

Compute `fraction` of discovered candidate files:

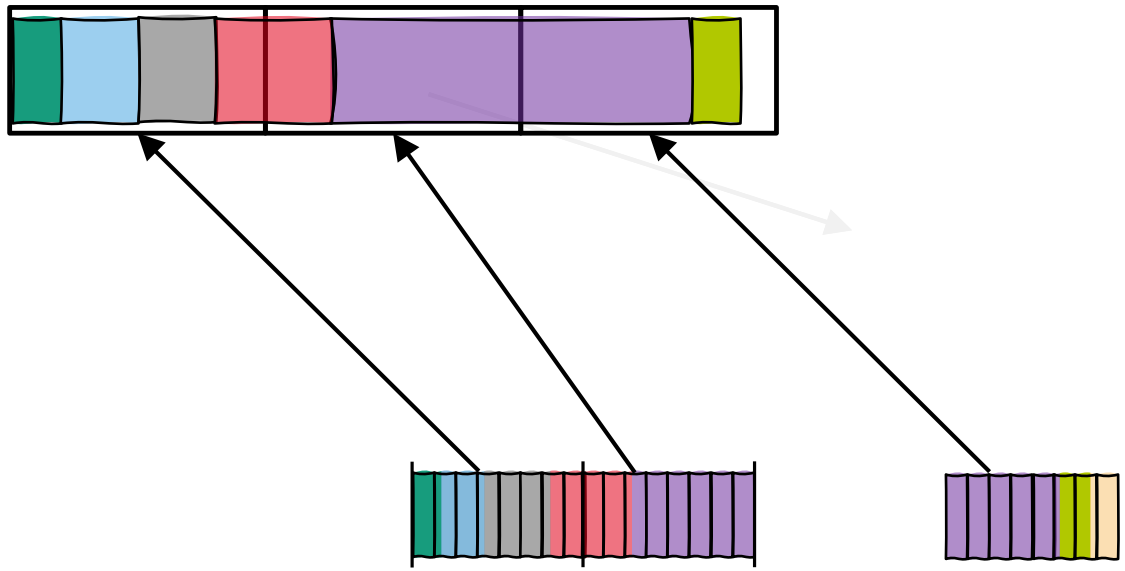
```
score / (metadata.filesize // 4096)
```

$(fid_i, \texttt{score}, \texttt{fraction})$

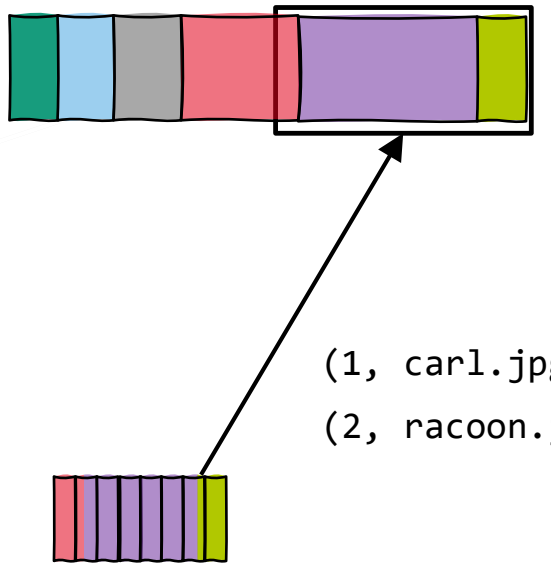
HASH-SETS

Example Continued

carl.jpg



racoon.jpg



(1, carl.jpg, 3, metadata)

(2, racoon.jpg, 1, metadata)

(1, carl.jpg, 3, 1)

(2, racoon.jpg, 1, 0.5)

HASH-RUNS

identified_blocks
.txt

identified_blocks
_explained.txt

```
(1, carl.jpg): {
     12470fe406d44017d96eab37dd65fc14: [ 1032, { 0 } ],
     e5a93371cfc7eab4a88221dd1f6c1a3c: [ 1040, { 1 } ],
     c7698f7bae9378423a34e46a25852e6a: [ 1056, { 2 } ]
},
(2, racoon.jpg): {
     e5a93371cfc7eab4a88221dd1f6c1a3c: [ 1080, { 1 } ]
}
```

For each candidate file:

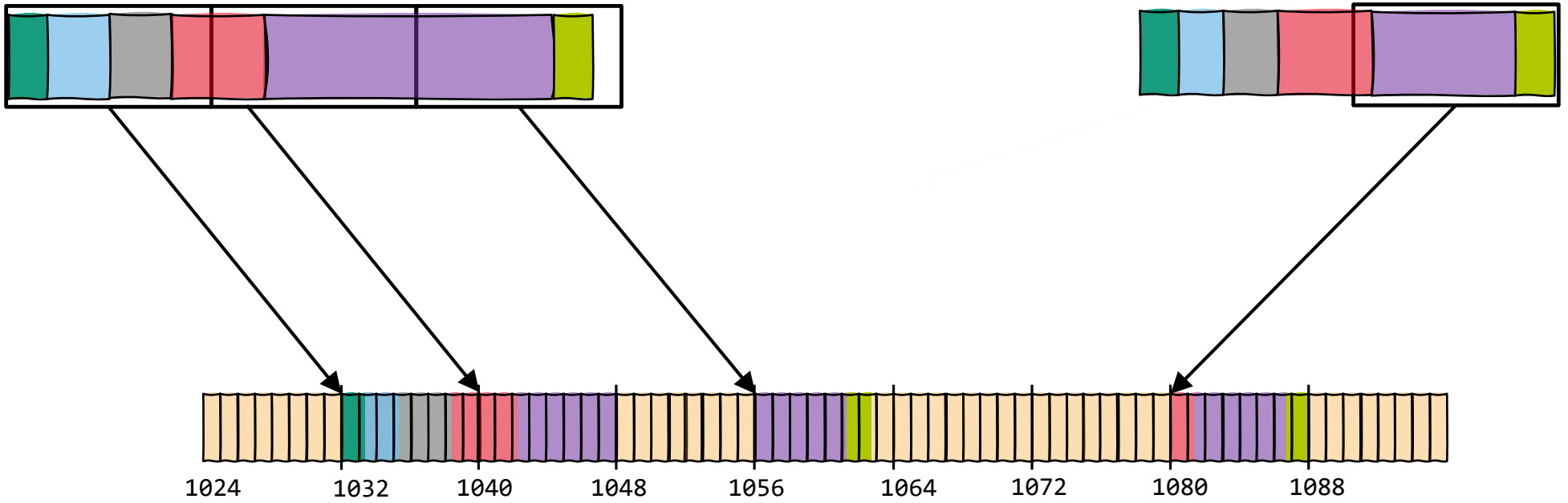
- Set of sector hashes associated with each file
- Block numbers where hash value appears in candidate files
- Disk block where each hash value was found on the disk image

HASH-RUNS

Example Continued

carl.jpg

racoon.jpg



HASH-RUNS

Block Runs

- Merge sequential disk and file blocks into block runs

```
(1, carl.jpg): {
     [ 1032, { 0 } ],
     [ 1040, { 1 } ],
     [ 1056, { 2 } ]
},
(2, racoon.jpg): {
     [ 1080, { 1 } ]
}
```

For each file: Merge consecutive entries `A`, `B` if

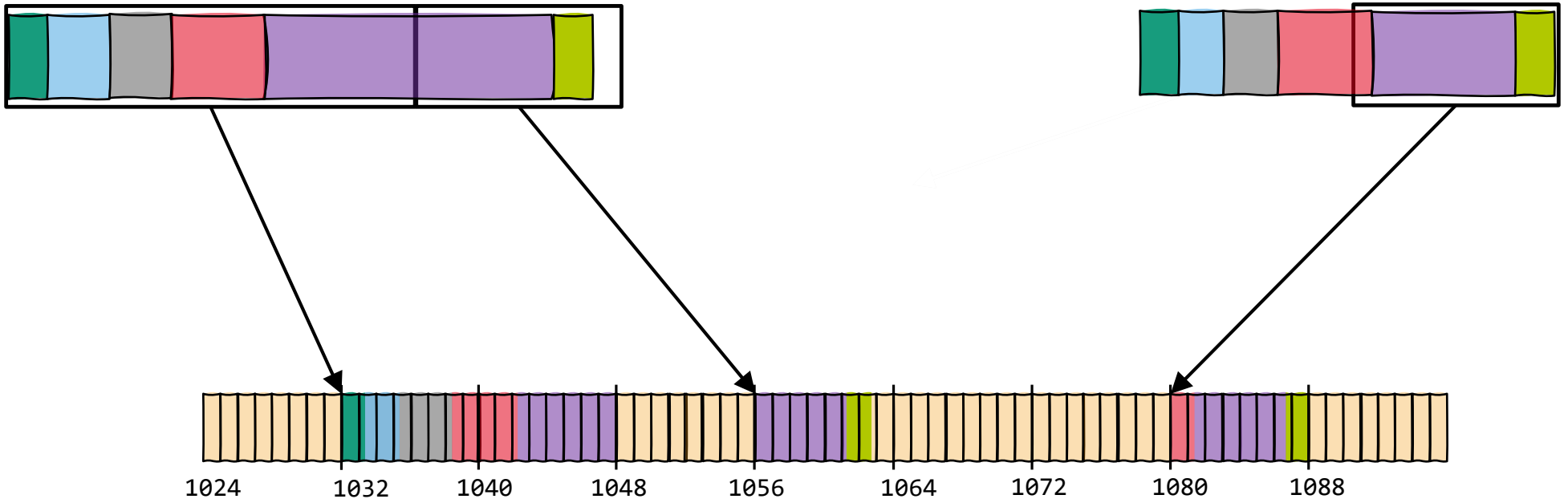
- `diskBlockB` = `diskBlockA + 8`
- `fileBlocksB` holds element which is 1 greater than `fileBlocksA`

HASH-RUNS

Example Continued

carl.jpg

racoon.jpg



Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

HASH-RUNS

Merging Block Runs

- Combine block run elements

```
[ carl.jpg, 2, 1032, 0, 1 ]
[ carl.jpg, 1, 1056, 2, 2 ]
[ racoon.jpg, 1, 1080, 0, 0 ]
```

Merge block runs if

- Number of bytes in sector gap between runs matches number of bytes in logical blocks
- Sectors in gap contain only NULLs

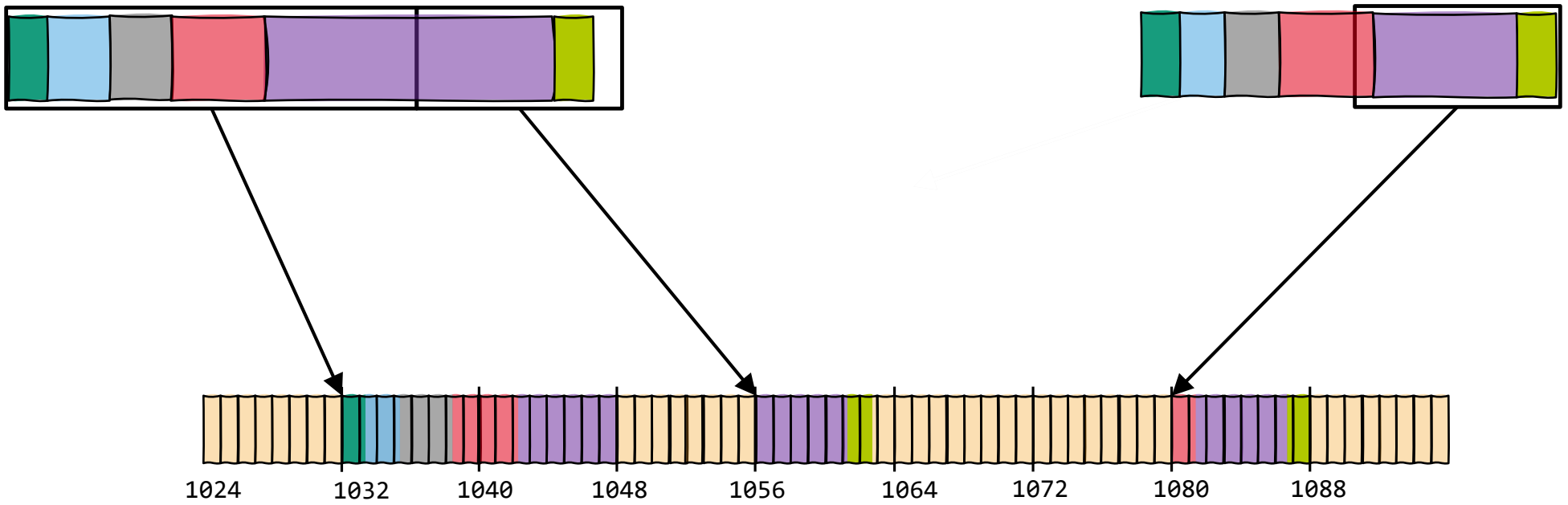
HASH-RUNS

Example Continued

carl.jpg

```
[ carl.jpg, 2, 1032, 0, 1 ]
[ carl.jpg, 1, 1056, 2, 2 ]
[ racoon.jpg, 1, 1080, 0, 0 ]
```

racoon.jpg



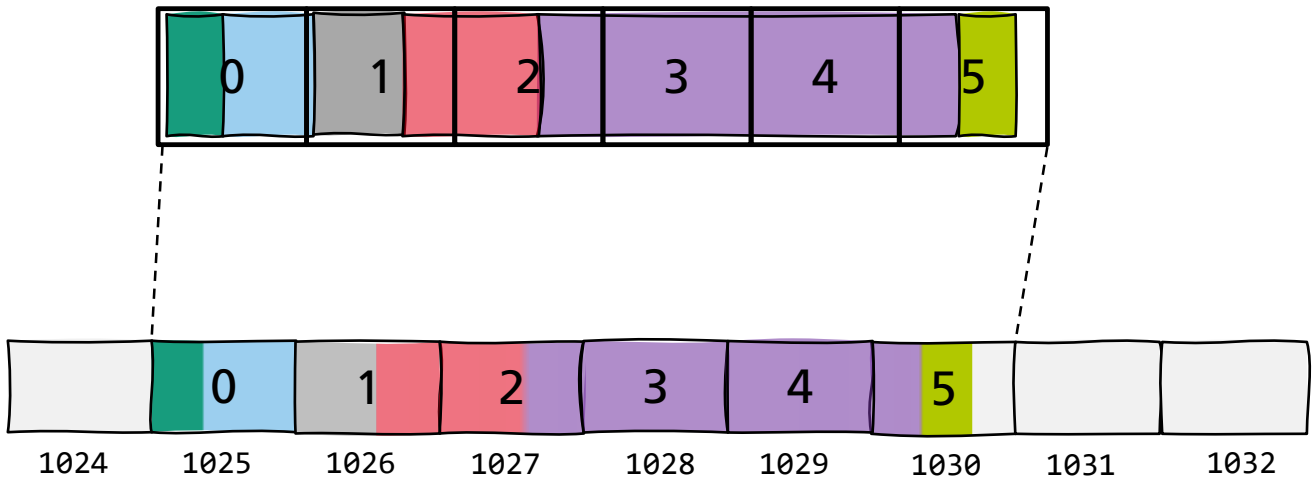
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Matching Scenarios

Copies

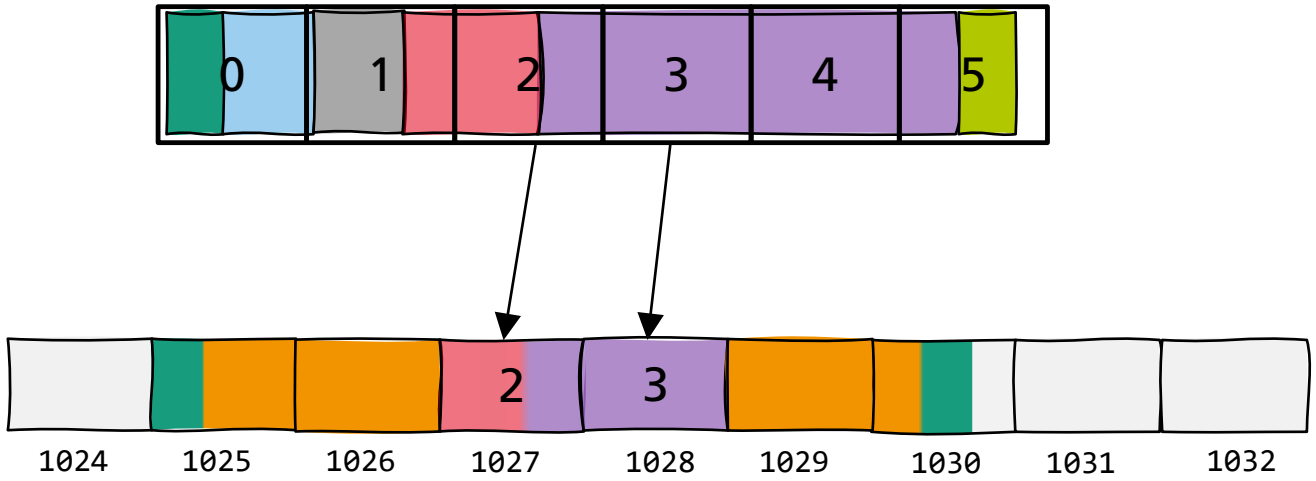
- *Copies of intact target files* present on search media
  - File can be allocated, deleted, or in free space



Matching Scenarios

Deleted and Partially Overwritten

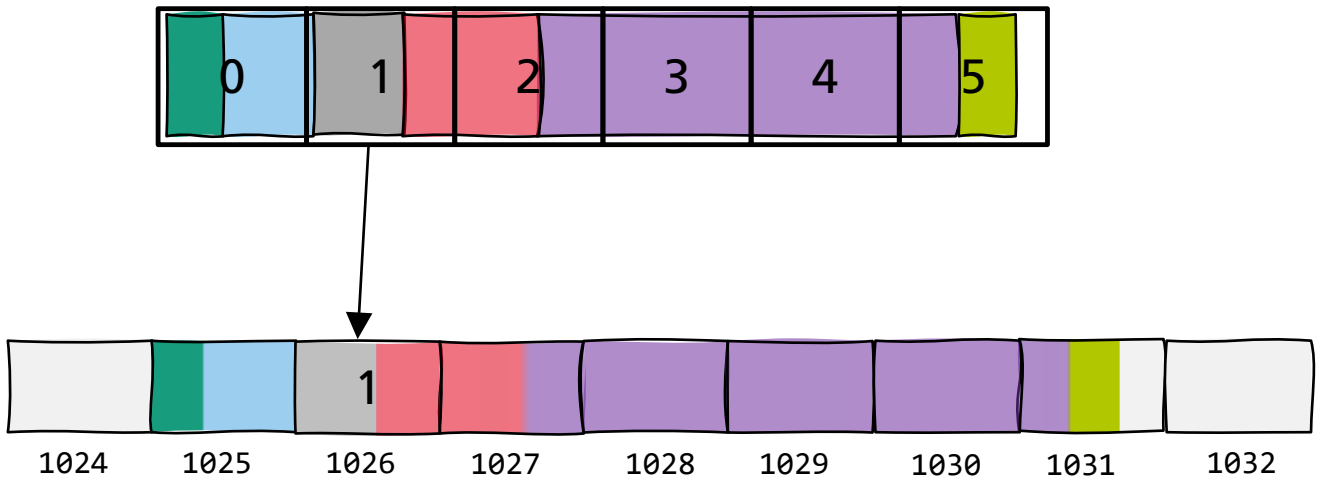
- *Deleted and partially overwritten* files
  - Still some fragments left…



Matching Scenarios

Similar File Blocks

- Files that are *similar* to target files
  - Identification of blocks shared between two files



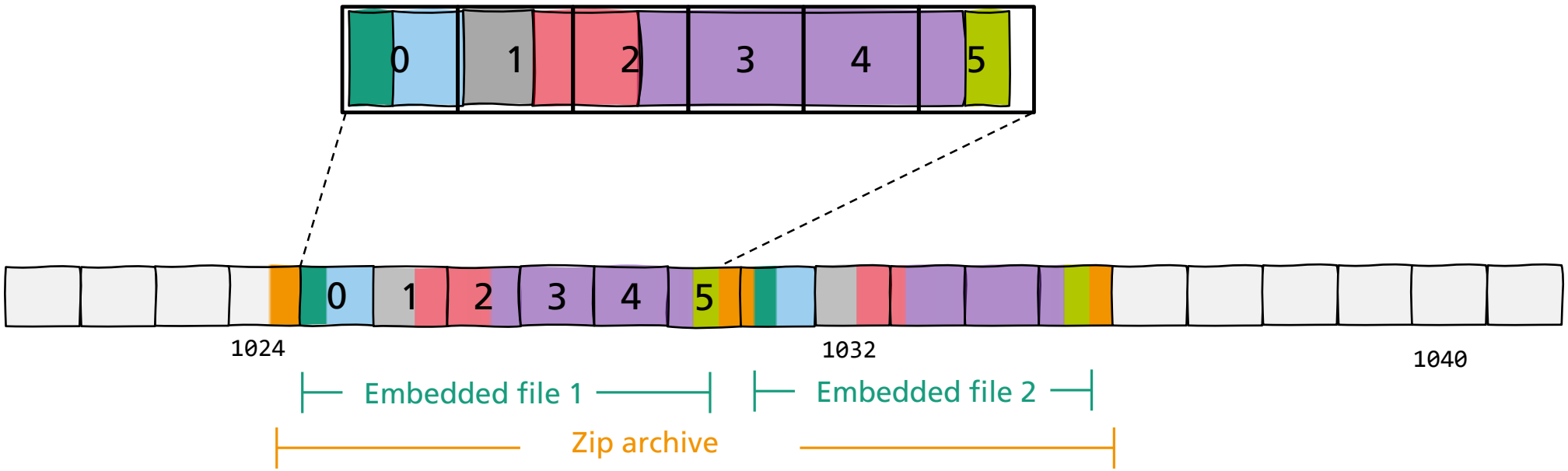
Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Matching Scenarios

Embedded files

- Target files *embedded in carrying files*
  - …when the file is embedded on even sector boundary



File Carving Got 99 Problems.

→ Fragmentation with a lot of fragments

→ Unfortunate fragmentation

→ Fragments in arbitrary order

→ Missing fragments

→ No (fixed) block size

→ …

File Carving Got 99 Problems.

→ Fragmentation with a lot of fragments

→ Unfortunate fragmentation

→ Fragments in arbitrary order

→ Missing fragments

→ No (fixed) block size

→ …

File Carving Got 99 Problems.

→ Fragmentation with a lot of fragments

→ Unfortunate fragmentation

→ Fragments in arbitrary order

→ Missing fragments

→ No (fixed) block size

→ …

File Carving Got 99 Problems.

→ Fragmentation with a lot of fragments

→ Unfortunate fragmentation

→ Fragments in arbitrary order

→ Missing fragments

→ No (fixed) block size

→ …

File Carving Got 99 Problems.

→ Fragmentation with a lot of fragments

→ Unfortunate fragmentation

→ Fragments in arbitrary order

→ Missing fragments

→ No (fixed) block size

→ …

File Carving Got 99 Problems.

→ Fragmentation with a lot of fragments

→ Unfortunate fragmentation

→ Fragments in arbitrary order

→ Missing fragments

→ No (fixed) block size

→ …

Limitations

- Encryption
  - Single files or complete file system

- Compression

- Depends on reference dataset
  - If target file is not part of reference data, file cannot be reassembled (or recognized)

- Potentially slow
  - …depends on target media size and reference dataset size
  - *Sector Sampling* allows significant speed-up

Hochschule Bonn-Rhein-Sieg

Fraunhofer FKIE

Any Questions?