Fraunhofer FKIE

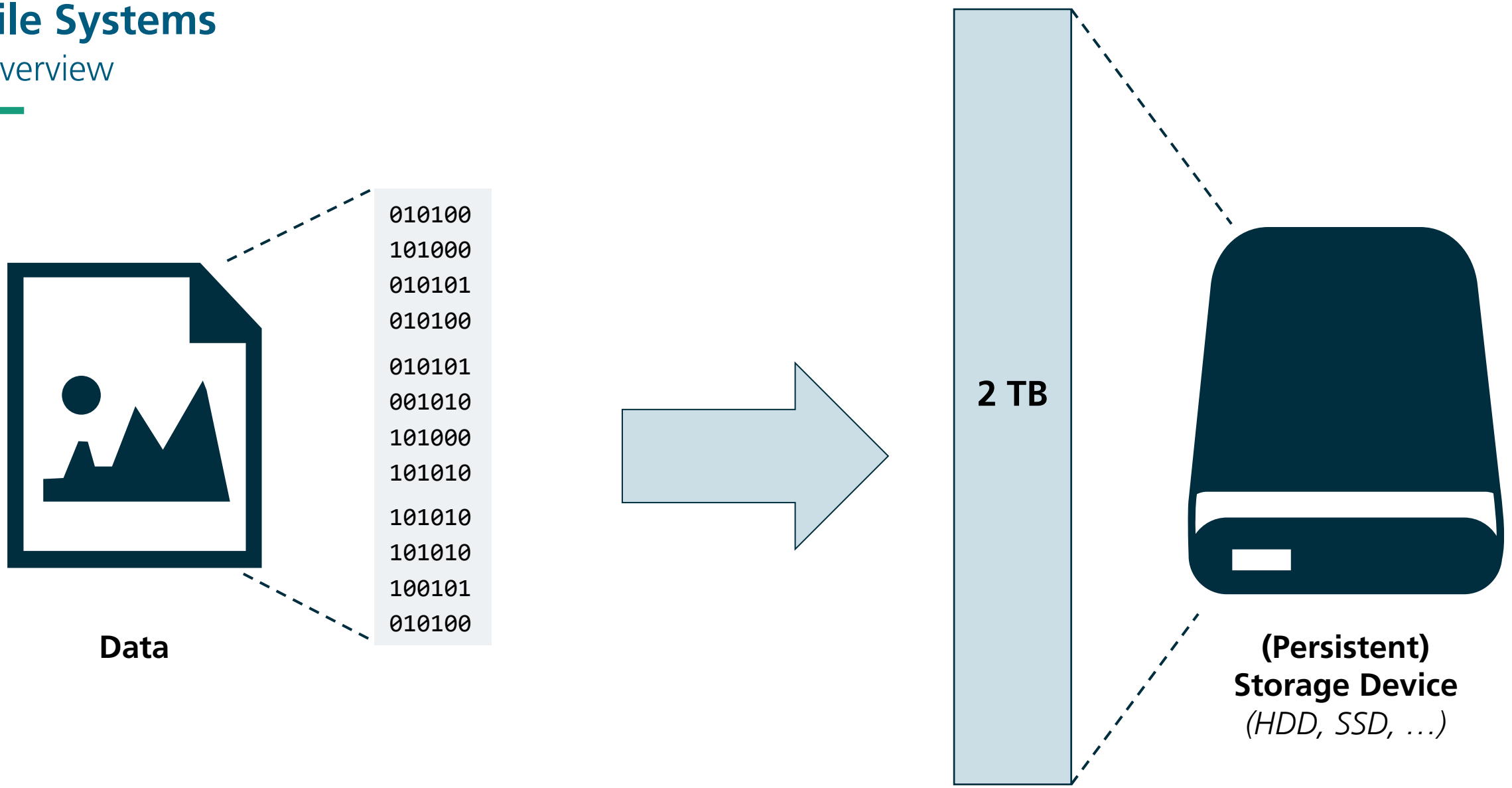
Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE

Anton Schwietert, Jan-Niclas Hilgert*

Data Hiding in File Systems

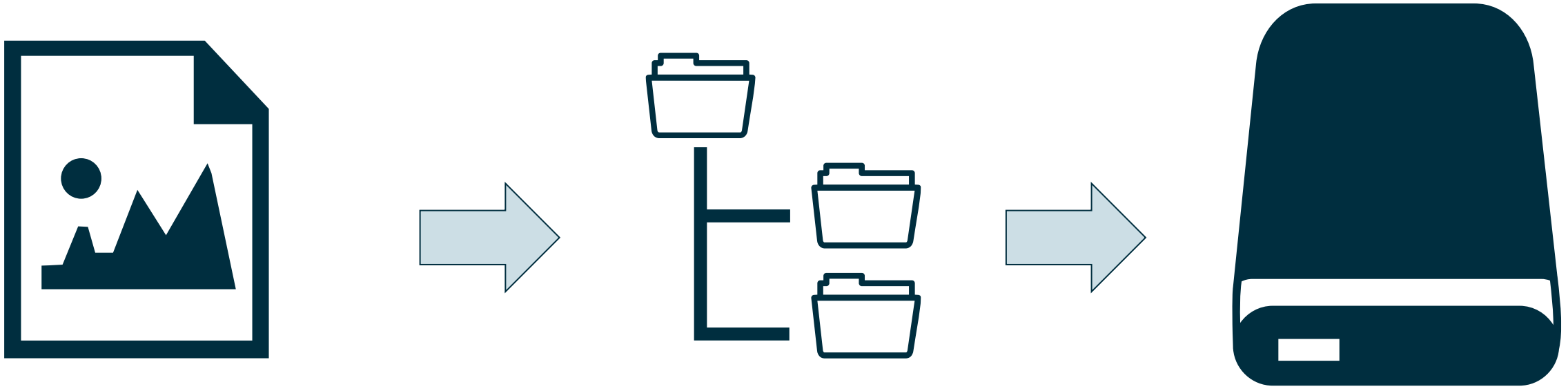
Current State, Novel Methods, and a Standardized Corpus

File Systems
Overview



Fraunhofer FKIE

File Systems
Overview

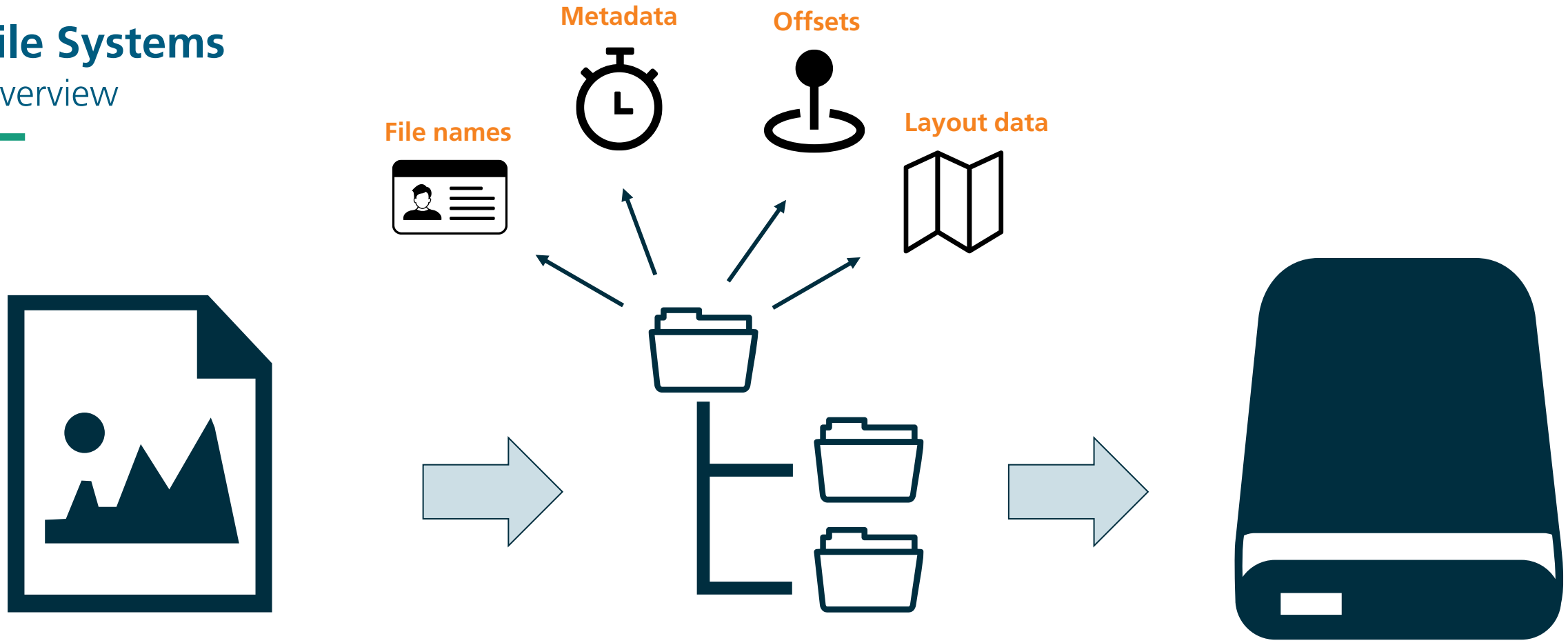


Data

File System

(Persistent) Storage Device
(HDD, SSD, …)

File Systems
Overview

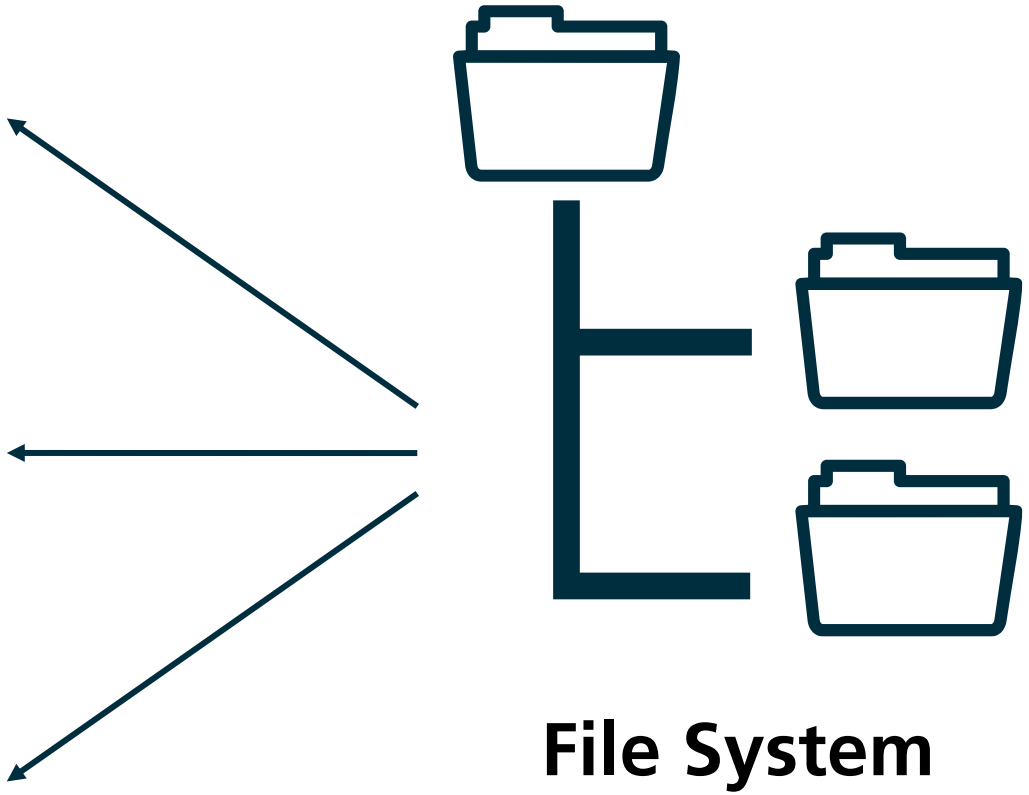


File Systems
Forensic Analysis

Find and **list files** and their **metadata**

Find hidden files

Recover deleted files

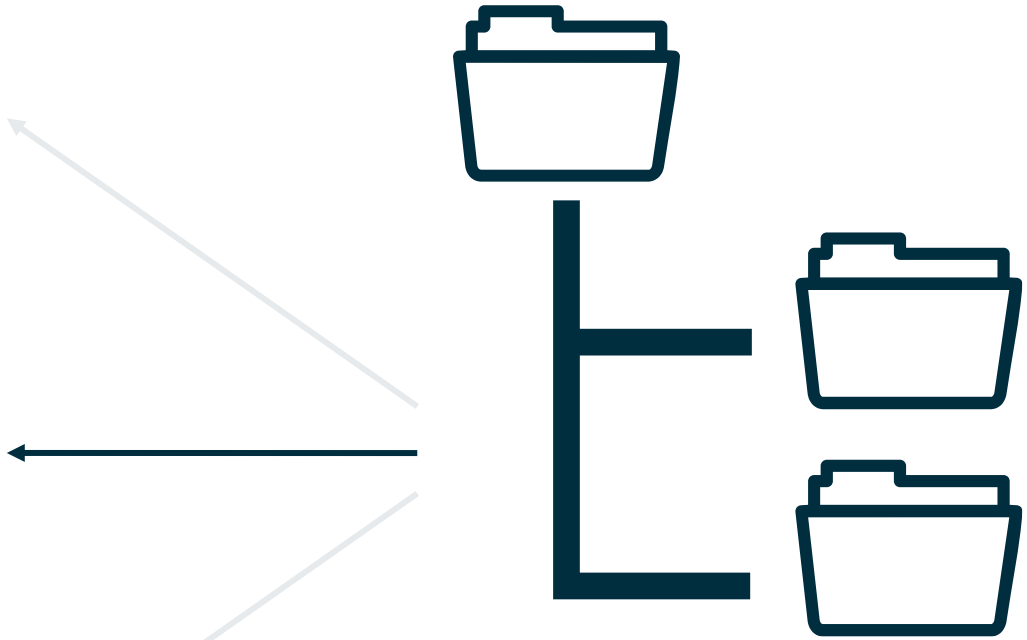


File Systems
Forensic Analysis

Find and **list files** and their **metadata**

Find hidden files

Recover deleted files



File System

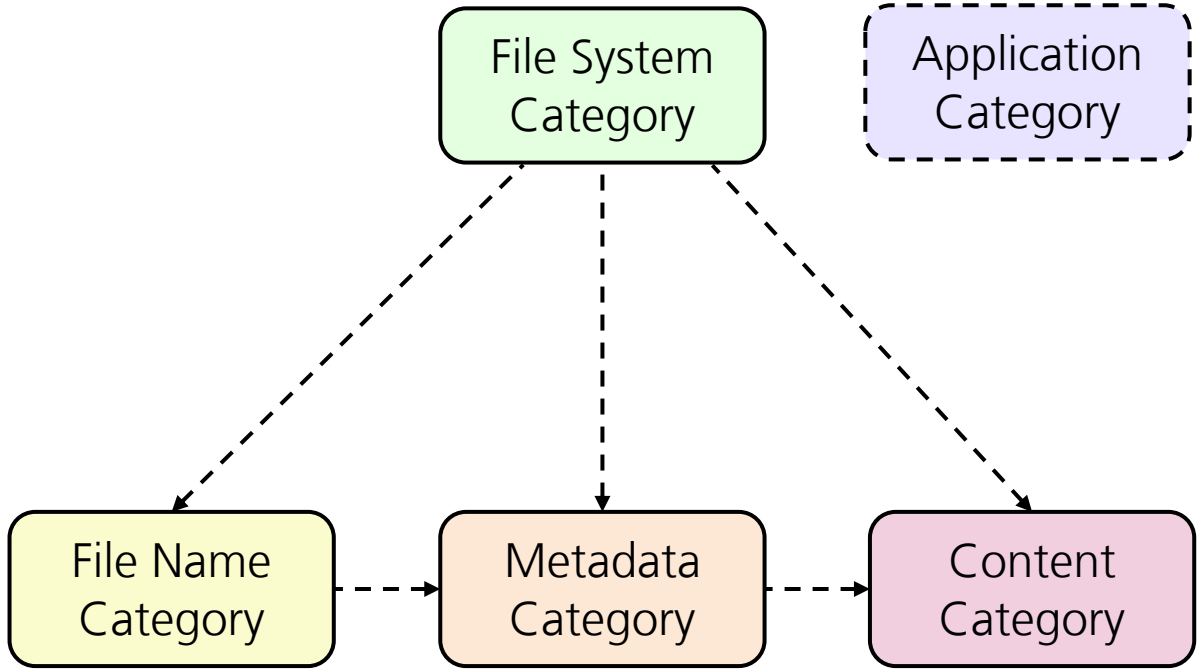
File Systems
Forensic Analysis

Find
hidden files

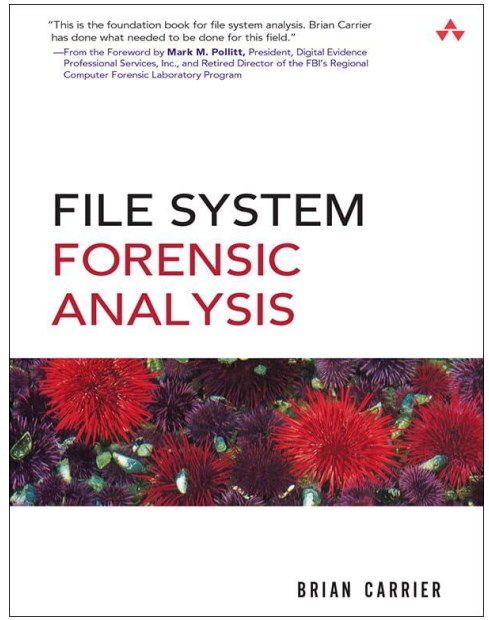
Where and how
can data be hidden
within a file system?

Data Hiding in File Systems

Where?



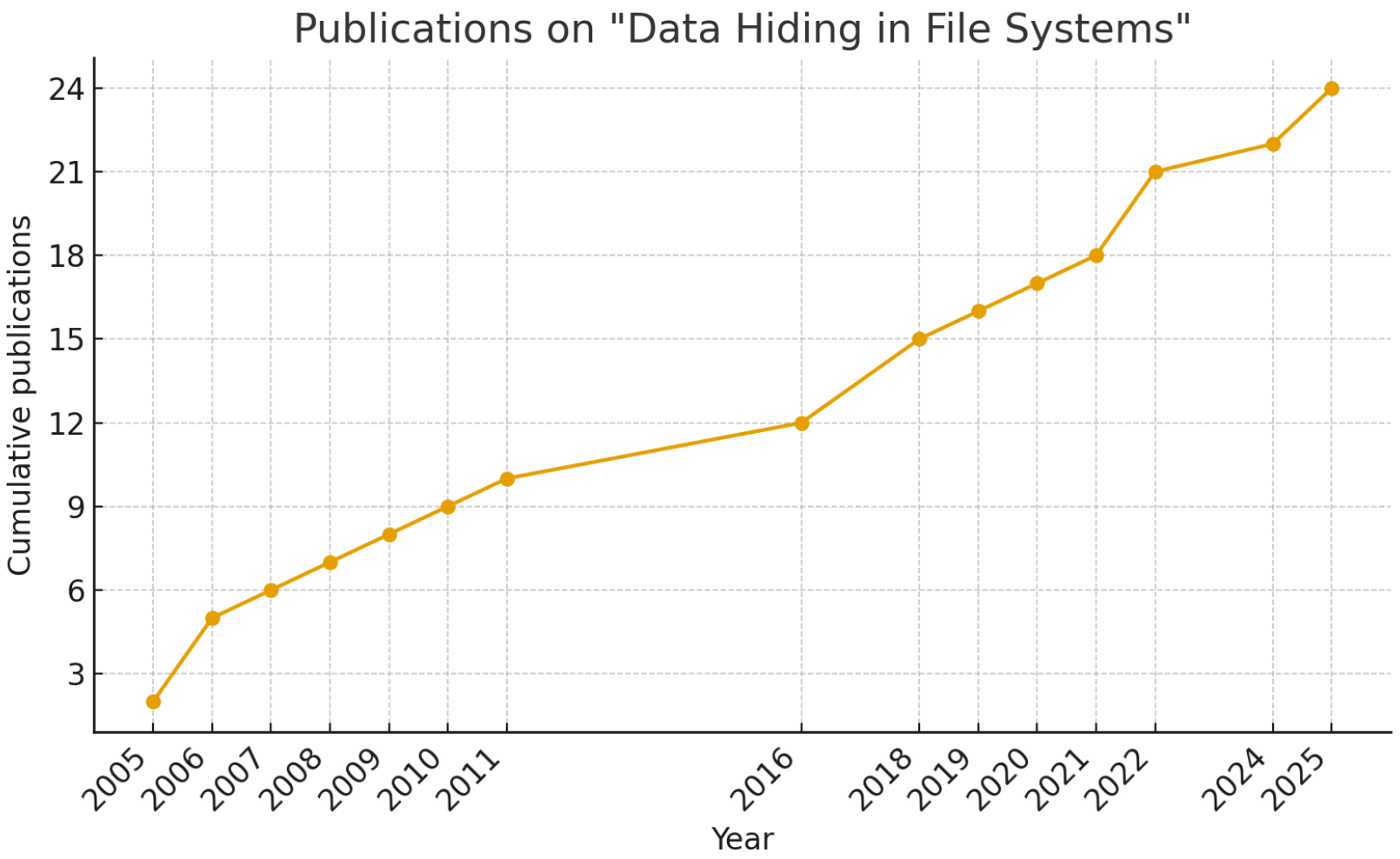
Data Categories of File Systems



Fundamental work about the forensic analysis of file systems

Data Hiding in File Systems

How?



Data Hiding in File Systems

How?

README

slack_hider

This is a simple python tool that r
of files to slack space locations.
Code is for evaluation purposes

README MIT license

fishy

`fishy` is a toolkit for filesystem based data hiding techniques, implemented in Python. It collects various common exploitation methods, that make use of existing datastructures on the filesystem layer, for hiding data from conventional file access methods. This toolkit is intended to introduce people to the concept of established anti-forensic methods associated with data hiding.

This document will provide some basic information about `fishy`. For a more in-depth documentation, you can visit the github wiki or use the documentation within the repository.

README

ext4-timestamp-magic

PoC implementation for nanosecond timestamp-based data hiding in the ext4 file system

In the meantime the ext4 Timestamp Hiding Tool is integrated into our fishy framework. Please have a look on:
https://github.com/dasec/fishy

Data Hiding in File Systems

Contributions

Current State

Novel Methods

Standardized Corpus

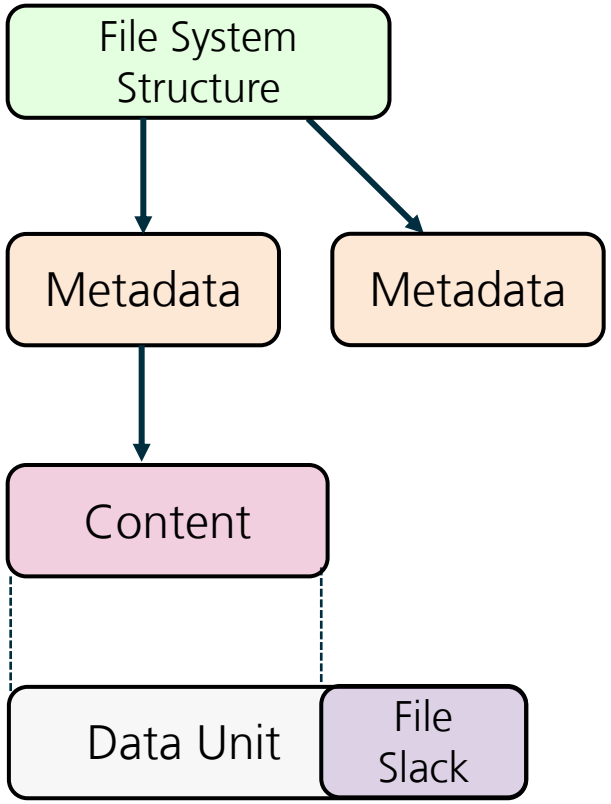
Data Hiding in File Systems

Current State

	ext2/ext3	ext4	APFS	FAT	NTFS	XFS	exFAT	Btrfs
File System Category								
Superblock	2	9	22			7		20
Group Descriptor (Table)	2,16	9						
Boot Sector	2	9		1				
Content Category								
File Slack	6,17,19	8,24		5,8,14,18,19	3,8,17,18			20
Deleted Files				5	5			
Alternate Data Streams					1,3,5,16,17,18, 19			
Bad Blocks	1	8		1,5,8,14	1,3,8,17			
Block Bitmap		9						
Additional Data Units					17			
Metadata Category								
Metadata Entries	2,6	8,9	10, 22		3	7	4, 23	
MFT/FAT		8		8,13	1, 3			
`$DATA` Attribute					3			
Timestamps		8,9,11	10		8,12	7	4,23	20
Extended Attributes	17					17		
Allocation Bitmap		9						
Symbolic Link Slack	21	21			21	21		
File Name Category								
File Names	6,19				19			
Directory Entries		9		15				

Data Hiding in File Systems

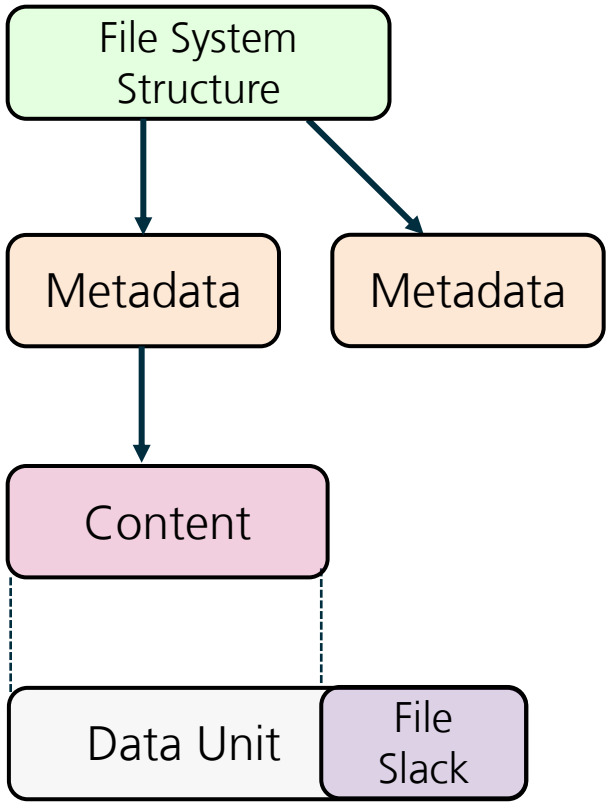
Slack Space



Content Category

Data Hiding in File Systems

Slack Space



Hiding in Slack Space — the adversary hides data in the slack space, the unused allocated bytes of file (see 2(c)). They conclude that over time a lot of files see very little change during syste
hiding. Som
hiding in sl

B. Slack space

Storage blocks on disk often only contain one APFS data structure, with this datastructure not using all the space of a block. The remainder is referred to as slack [5]. For our research we extended the AFRO [16] open source tool to also parse the remainder of a block. We implemented this parsing for container superblocks, volume superblocks and their respective object maps. With AFRO we were able to parse and catalogue the contents of the slack space. More on this in Section IV.

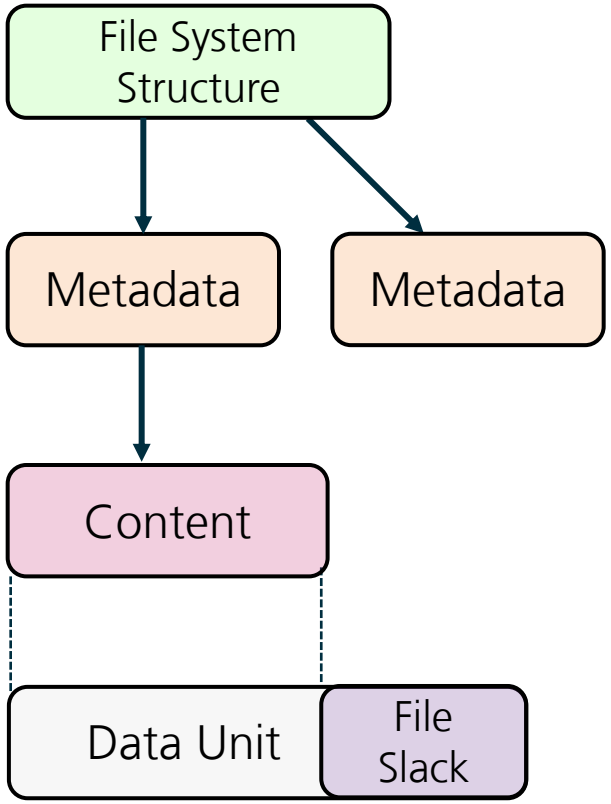
F. File Slack Space Hiding

Since the file system has so
files. Shu-fen et al [6] describ
similar to hiding files in free s
space sizes available. To find
size of the files in the system.
the remaining space in the fi
encrypted in order to restore the file. As such, there will be no directory entry for the hidden file because of the manual storage of file segments. Again, the locations of the file segments could be decrypted if the configuration file can be found. Also, the hidden file contents will easily be seen following the end of the file.

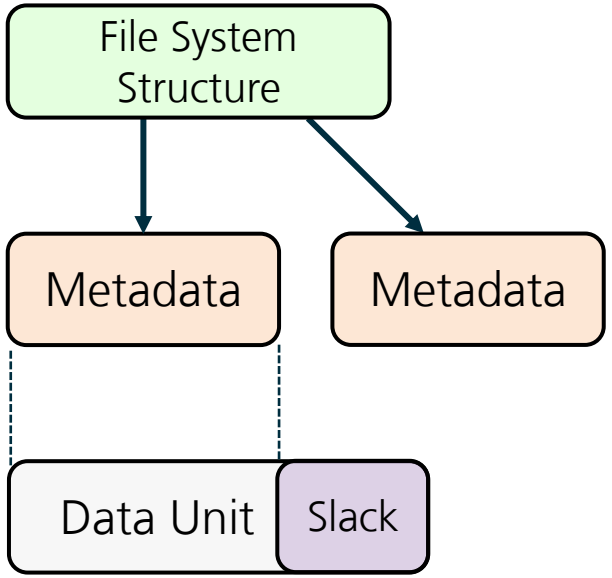
Content Category

Data Hiding in File Systems

Slack Space



Content Category



Metadata Category

Data Hiding in File Systems

Slack Space

5.3 Inode: Slack Space/Extended Attributes

The ext2 and ext3 filesystems have an inode size of 128 bytes and leave no space for data hiding. The default size of an inode record in ext4 is 256 bytes [10]. It can even be set to the filesystem block size using the `mkfs.ext4 [-I inode size]` option at format time. The extra 128 bytes are divided into a range of fixed fields and a range of extended attributes as shown in Figure 7. Each inode contains the field `i_extra_isize`, which records the additional number of bytes for the fixed fields beyond the original 128 bytes. The extra space between the end of the inode structure and the end of the inode record is meant to store extended

Content Category

Metadata Category

Data Hiding in File Systems

Slack Space

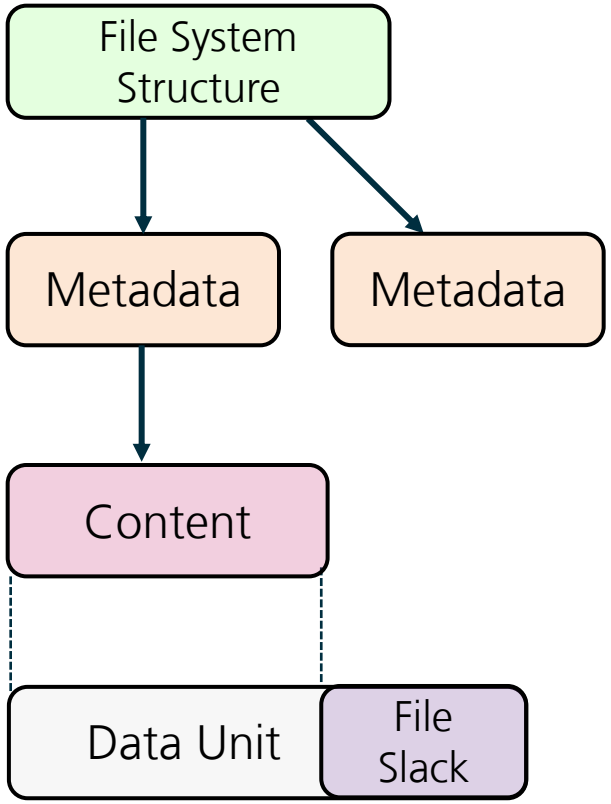
FS	Block Size			Checksum	
	default	min	max	meta	data
ext2/3	1/4 KiB	1 KiB	4 KiB[1]	✗	✗
ext4	4 KiB	1 KiB	64 KiB	✗	✗
XFS	4 KiB	512 B	64 KiB[2]	✓	✗
NTFS	4 KiB	512 B	2048 KiB[3]	✗	✗
FAT	*	512 B	512 KiB[4]	✗	✗
exFAT	*	512 B	32MB	✓	✗
Btrfs	4 KiB	512 B	64 KiB	✓	✓
APFS	4 KiB	4 KiB	64 KiB	✓	✗
ZFS	128KB	512 B	16 MiB	✓	✓

Content Category

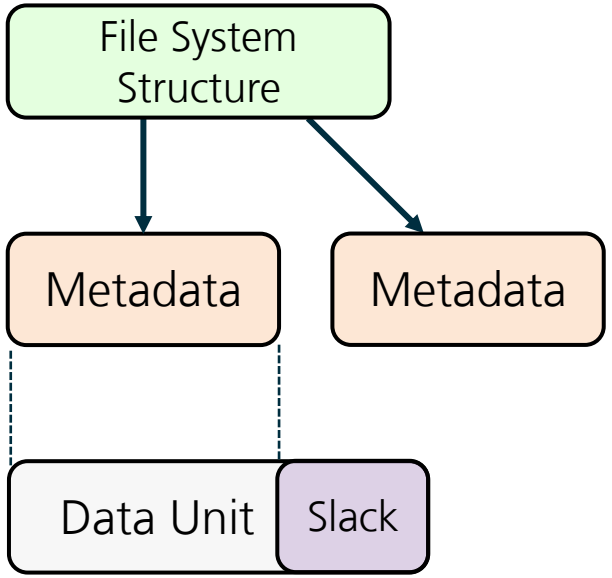
Metadata Category

Data Hiding in File Systems

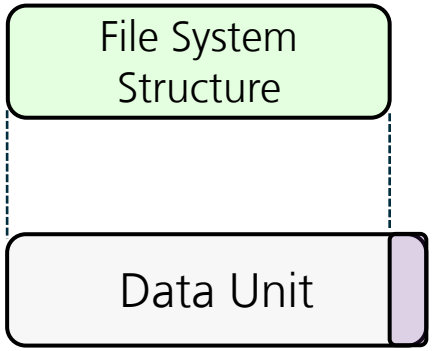
Slack Space



Content Category



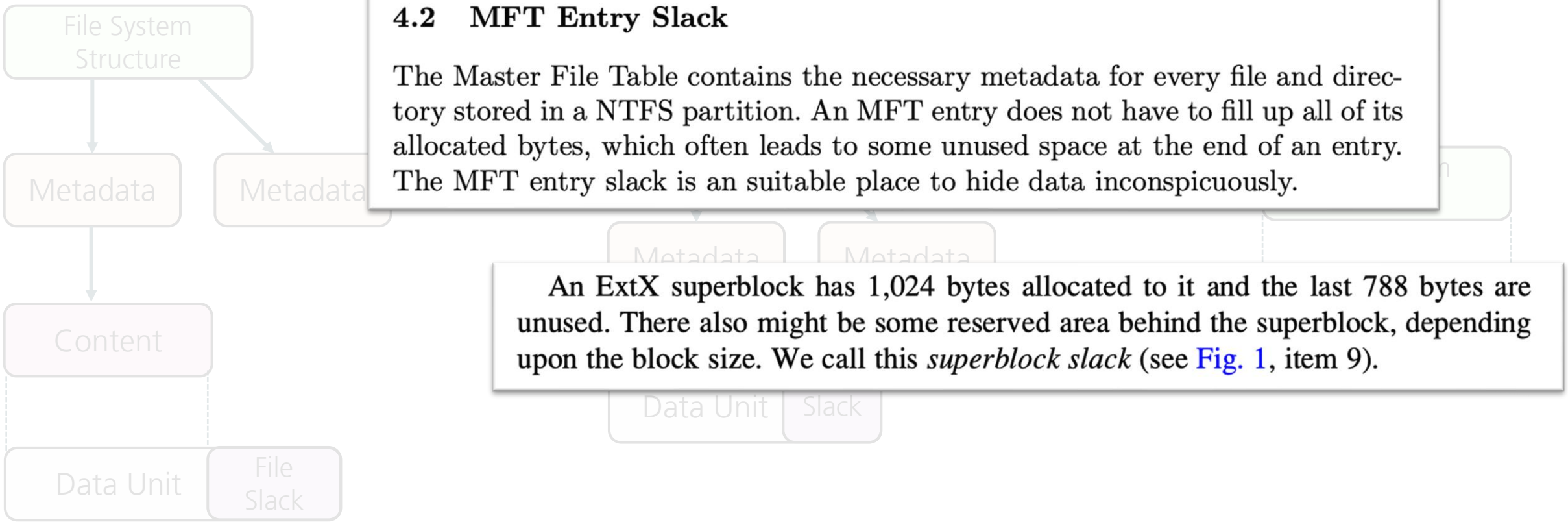
Metadata Category



File System Category

Data Hiding in File Systems

Slack Space



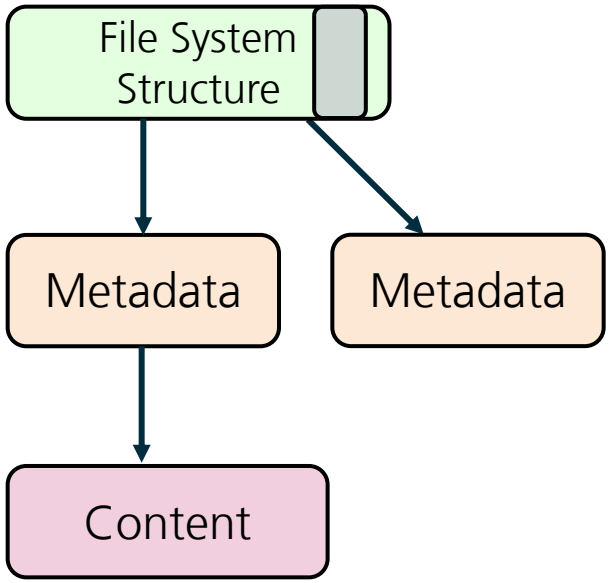
4.2 MFT Entry Slack

The Master File Table contains the necessary metadata for every file and directory stored in a NTFS partition. An MFT entry does not have to fill up all of its allocated bytes, which often leads to some unused space at the end of an entry. The MFT entry slack is an suitable place to hide data inconspicuously.

An ExtX superblock has 1,024 bytes allocated to it and the last 788 bytes are unused. There also might be some reserved area behind the superblock, depending upon the block size. We call this *superblock slack* (see Fig. 1, item 9).

Data Hiding in File Systems

Reserved Areas



Hiding Data in Reserved Portions of Superblocks

Every superblock has at least 384 bytes reserved for updates to the file system specification. Since this group of bytes does not currently have a purpose, it can be used to hide data. Also, a superblock is supposed to fit in only 1K of space, but it uses an entire block on the file system (up to 4K). Therefore, it is often possible to hide data in the extra 3K (max) of space.

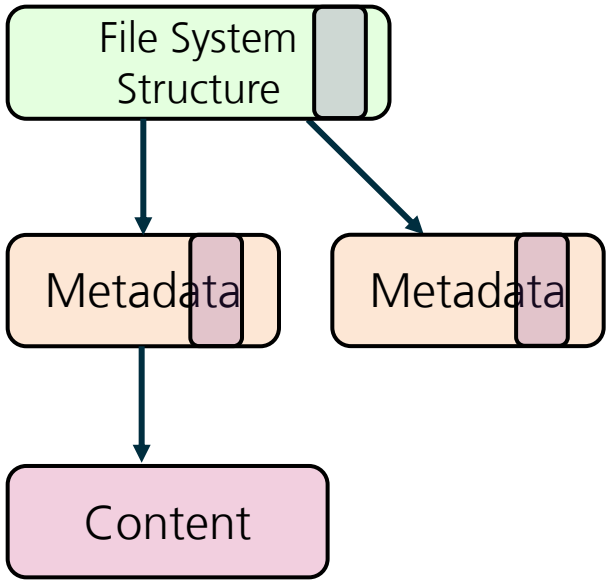
Procedure:

1. Create a file named `secret_data.txt` with size less than 384 bytes.
2. Insert a floppy disk into the computer.
3. Execute the command: `# mke2fs -O none /dev/fd0` to format the disk as Ext2 with no extra features.
4. Execute the command: `# dd if=secret_data.txt of=/dev/fd0 seek=1664` to hide the data.

Alerts: None by `e2fsck`, EnCase, FTK and iLook.

Data Hiding in File Systems

Reserved Areas



The `osd2` field has multiple meanings depending on the filesystem creator:

Linux:

Offset	Size	Name	Description
0x0	__le16	l_i_blocks_high	Upper 16-bits of the block count. Please see the note attached to i_blocks_lo.
0x2	__le16	l_i_file_acl_high	Upper 16-bits of the extended attribute block (historically, the file ACL location). See the Extended Attributes section below.
0x4	__le16	l_i_uid_high	Upper 16-bits of the Owner UID.
0x6	__le16	l_i_gid_high	Upper 16-bits of the GID.
0x8	__le16	l_i_checksum_lo	Lower 16-bits of the inode checksum.
0xA	__le16	l_i_reserved	Unused.

https://docs.kernel.org/filesystems/ext4/inodes.html

Data Hiding in File Systems

Misuse of File System Structures

Alternate Data Streams

Block Allocation Manipulation

Timestamps

Data Hiding in File Systems

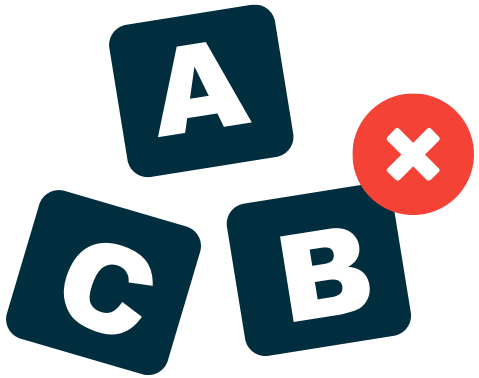
Misuse of File System Structures

Alternate
Data Streams

Block Allocation Manipulation

Timestamps

> It is possible to manipulate the file system metadata that identifies bad blocks (e.g., the File Allocation Table in a FAT file system or $BadClus in NTFS) so that usable blocks are marked as bad and therefore will no longer be accessed by the operating system. Such *metadata manipulation* (see Fig. 1, item 7) will produce blocks that can store hidden data.



Adding Bad Clusters

Data Hiding in File Systems

Misuse of File System Structures

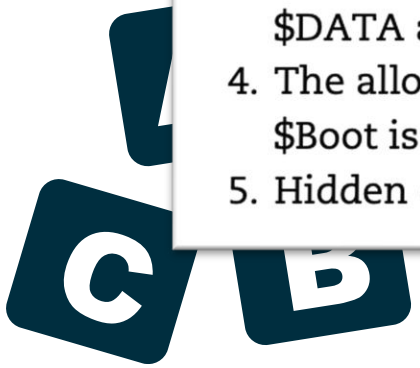
Alternate Data Streams

Block Allocation Manipulation

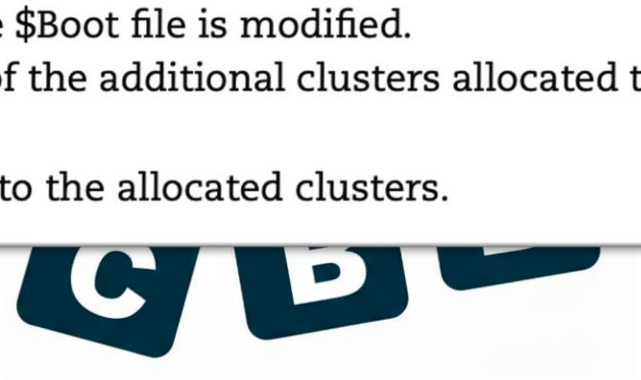
Timestamps

We applied this method to the $Boot file, but it is equally applicable to all others. The following procedure was used to create test data:

1. Cluster run list of the $DATA attribute of the $Boot file is modified.
2. The last VCN value of the $DATA attribute of the $Boot file is modified.
3. The real size, allocated size and compressed size of the $DATA attribute of the $Boot file is modified.
4. The allocation status of the additional clusters allocated to $Boot is set to 1.
5. Hidden data is pasted to the allocated clusters.



Adding Bad Clusters



Adding Clusters to a file

Data Hiding in File Systems

Misuse of File System Structures

Alternate
Data Streams

Block Allocation
Manipulation

Timestamps

```
~$ stat README
File: README
Size: 3996            Blocks: 8          IO Block: 4096   regular file
Device: 805h/2053d    Inode: 2763742     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/     jnh)   Gid: ( 1000/     jnh)
Access: 2022-10-06 16:13:46.068708287 +0200
Modify: 2022-10-06 14:13:40.836580656 +0200
Change: 2022-10-06 14:13:40.750485053 +0200
Birth: -
```

Data Hiding in File Systems

Misuse of File System Structures

Alternate
Data Streams

Block Allocation
Manipulation

Timestamps

```
~$ stat README
File: README
Size: 3996            Blocks: 8          IO Block: 4096   regular file
Device: 805h/2053d    Inode: 2763742     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/     jnh)   Gid: ( 1000/     jnh)
Access: 2022-10-06 16:13:46.068708287 +0200
Modify: 2022-10-06 14:13:40.836580656 +0200
Change: 2022-10-06 14:13:40.750485053 +0200
Birth: -
```

068708287 836580656 750485053

Fraunhofer FKIE

Data Hiding in File Systems

Misuse of File System Structures

Alternate
Data Streams

Block Allocation
Manipulation

Timestamps

```
~$ stat README
File: README
Size: 3996            Blocks: 8          IO Block: 4096   regular file
Device: 805h/2053d    Inode: 2763742     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/     jnh)   Gid: ( 1000/     jnh)
Access: 2022-10-06 16:13:46.068708287 +0200
Modify: 2022-10-06 14:13:40.836580656 +0200
Change: 2022-10-06 14:13:40.750485053 +0200
Birth: -
```

```
68708287836580656750485053
D F R W S A P A C 2 0 2 5
```

Data Hiding in File Systems

Misuse of File System Structures

Alternate
Data Streams

Block Allocation
Manipulation

Timestamps

FS	Time-stamps	Size (bit)	Nano part	Location	Checksum
ext2/3	m,a,c	32	0	inode	✗
ext4	m,a,c,cr	64	32	inode	✓
NTFS	m,a,c,cr	64	24	MFT record	✗
exFAT	m,a,cr	32	0	Directory entry	✗
FAT	m,a,cr*	32,16*	0	Directory entry	✗
APFS	m,a,c,cr	64	32	inode	✓
XFS	m,a,c,cr	64	32	inode	✓
ZFS	m,a,c,cr	64	32	znode	✓
Btrfs	m,a,c,cr	64	32	inode	✓
HFS+	m,a,c,cr	32	0	Catalog file entry	✗

Data Hiding in File Systems

Contributions

Current State

Novel Methods

Standardized Corpus

Data Hiding in File Systems

Contributions

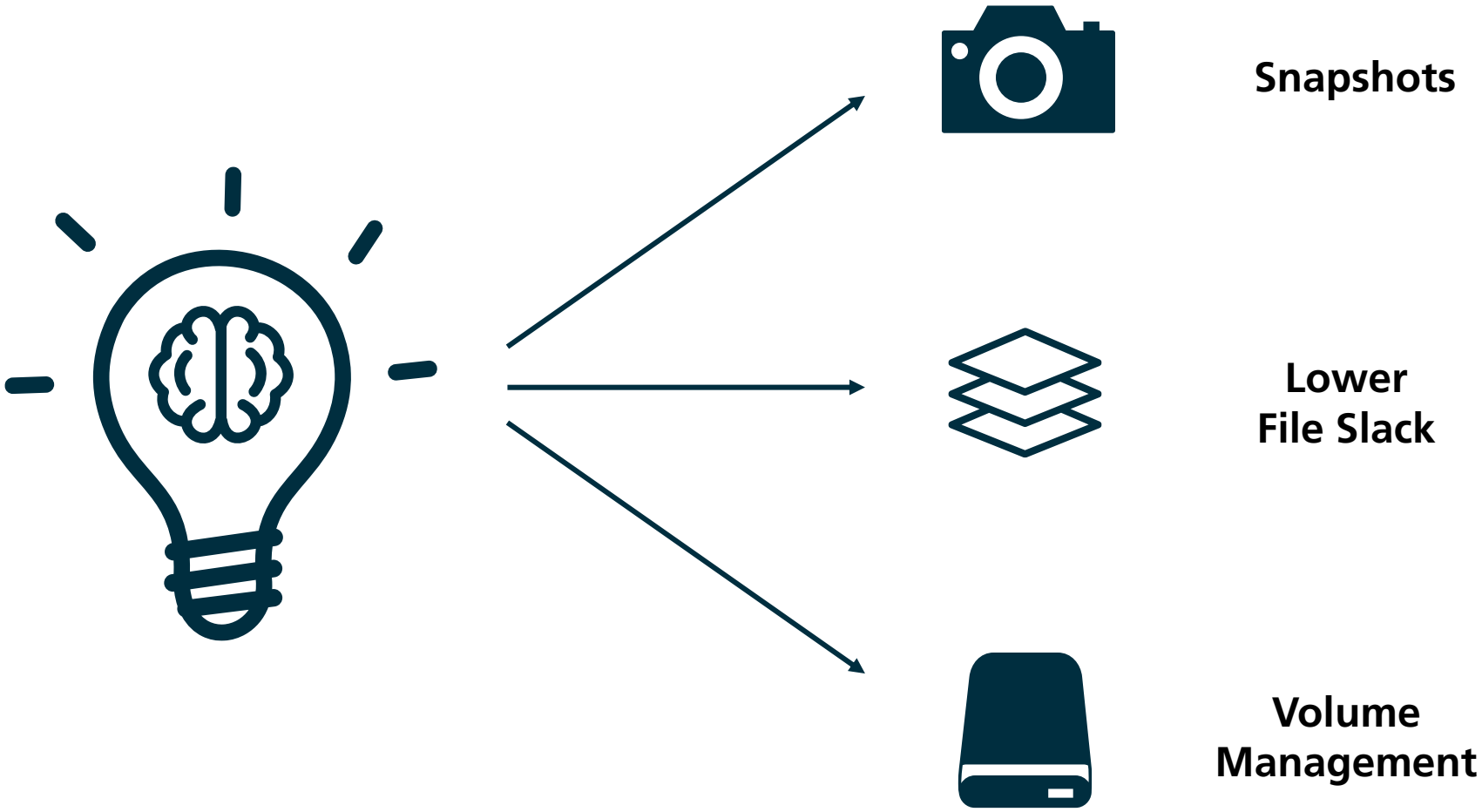
Current State

Novel Methods

Standardized Corpus

Data Hiding in File Systems

Contributions

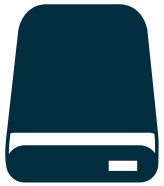


Data Hiding in File Systems

Novel Methods

Snapshots

Lower File Slack



Volume Management

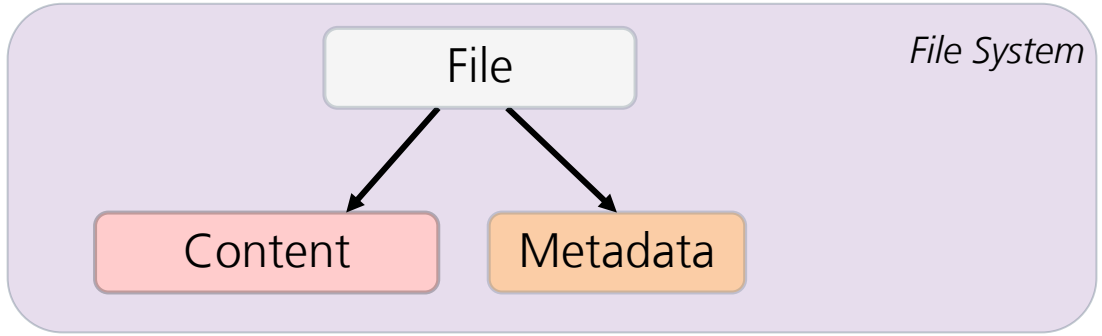
Data Hiding in File Systems

Novel Methods

Snapshots

Lower File Slack

Volume Management



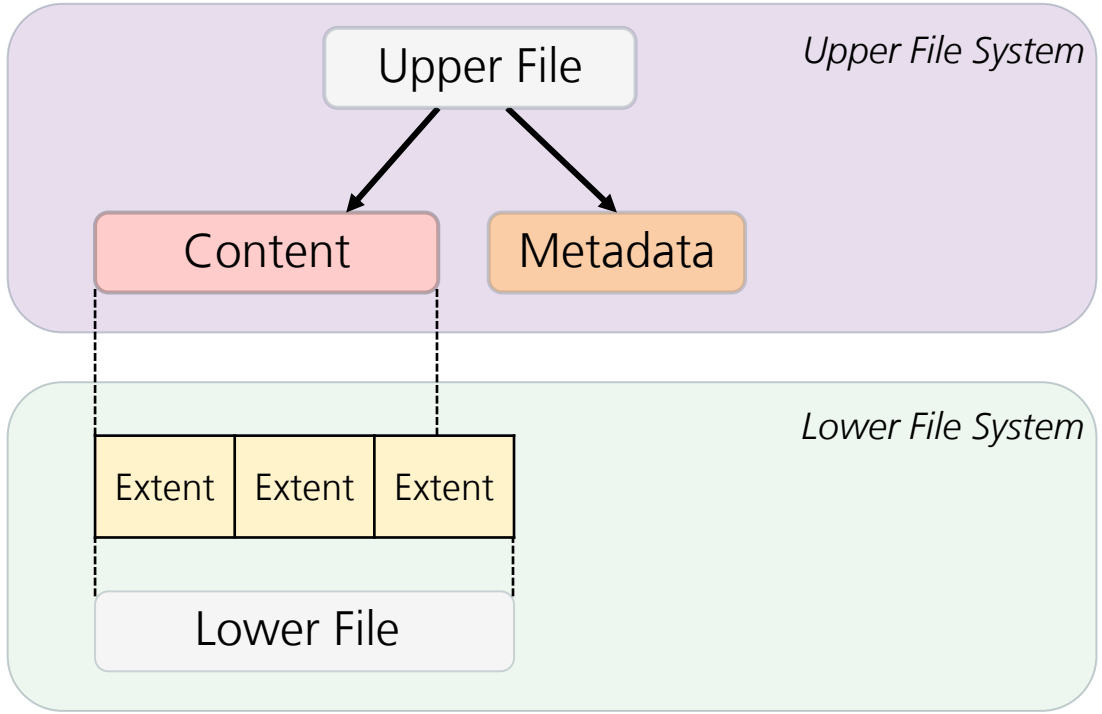
Data Hiding in File Systems

Novel Methods

Snapshots

Lower File Slack

Volume Management



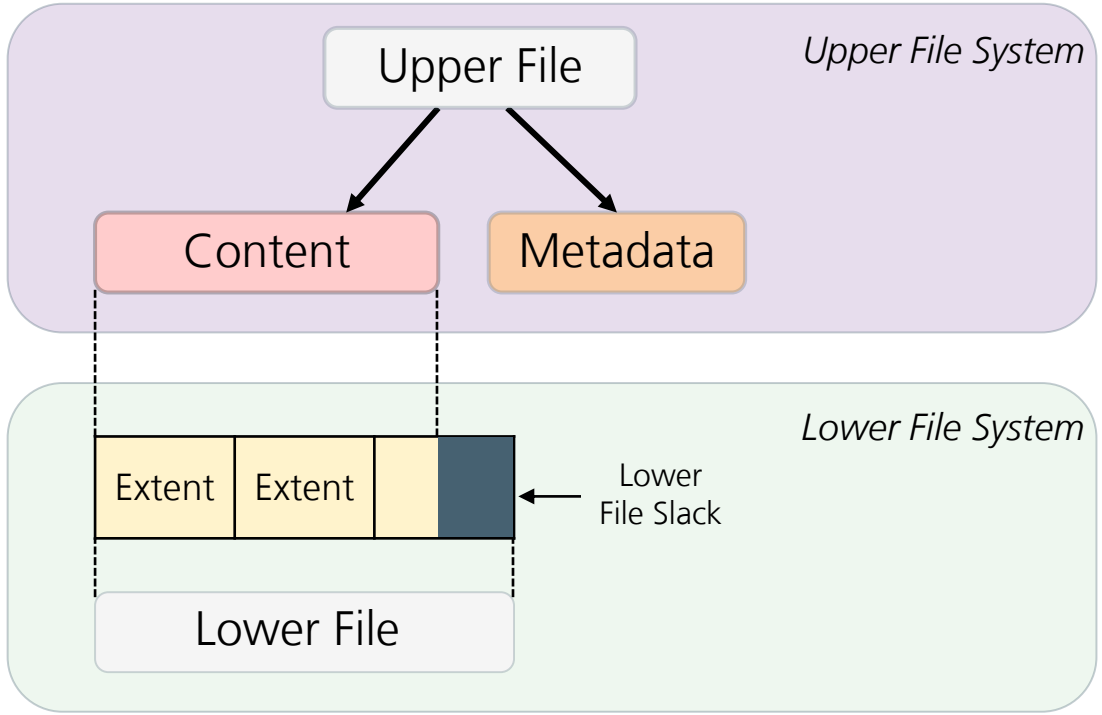
Data Hiding in File Systems

Novel Methods

Snapshots

Lower File Slack

Volume Management



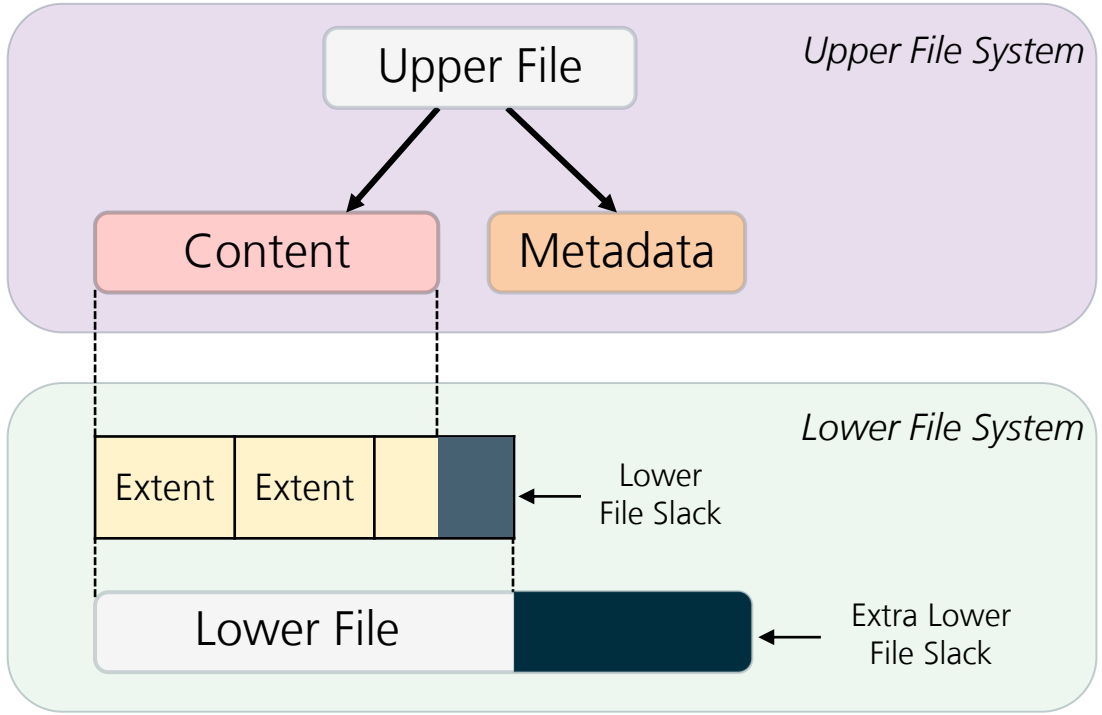
Data Hiding in File Systems

Novel Methods

Snapshots

Lower File Slack

Volume Management



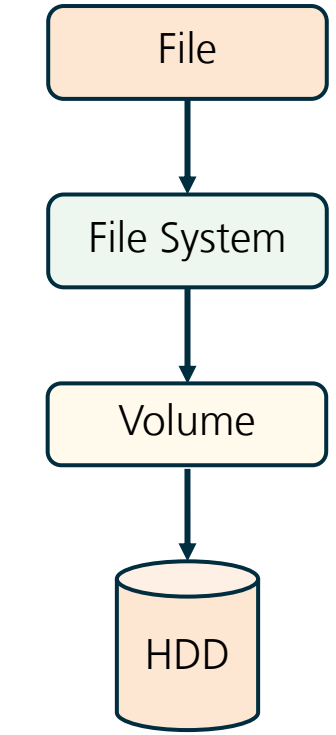
Data Hiding in File Systems

Novel Methods

Snapshots

Lower
File Slack

**Volume
Management**

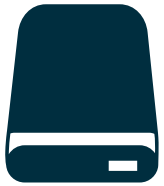


Data Hiding in File Systems

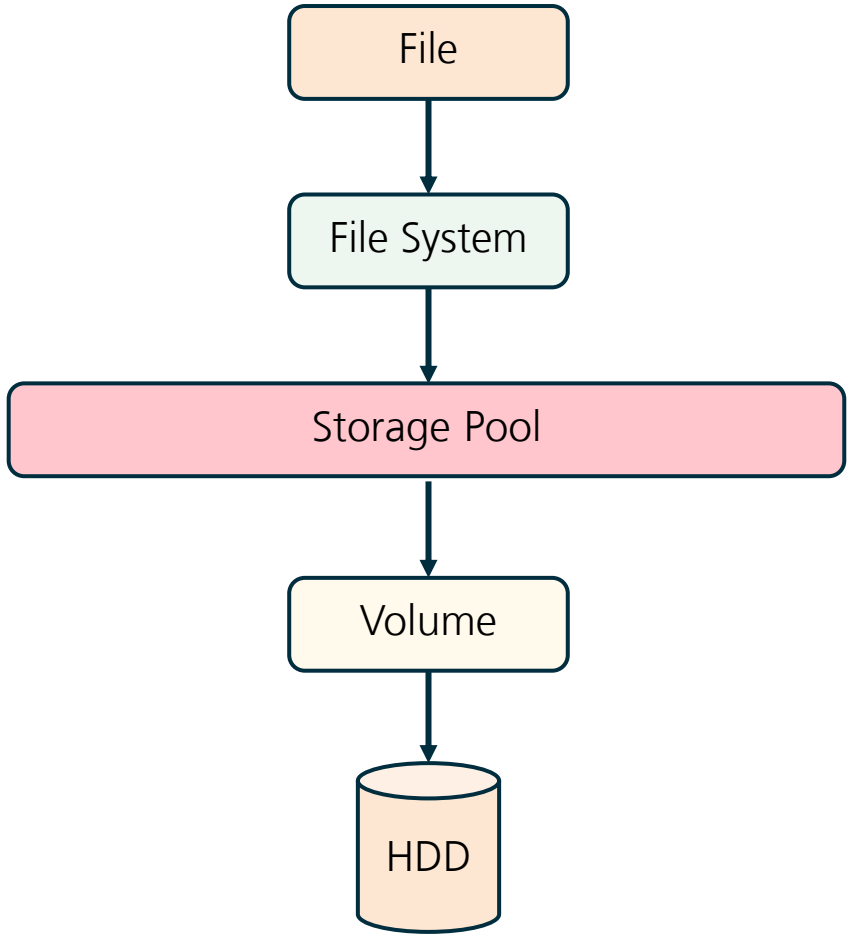
Novel Methods

Snapshots

Lower
File Slack



Volume Management

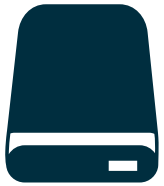


Data Hiding in File Systems

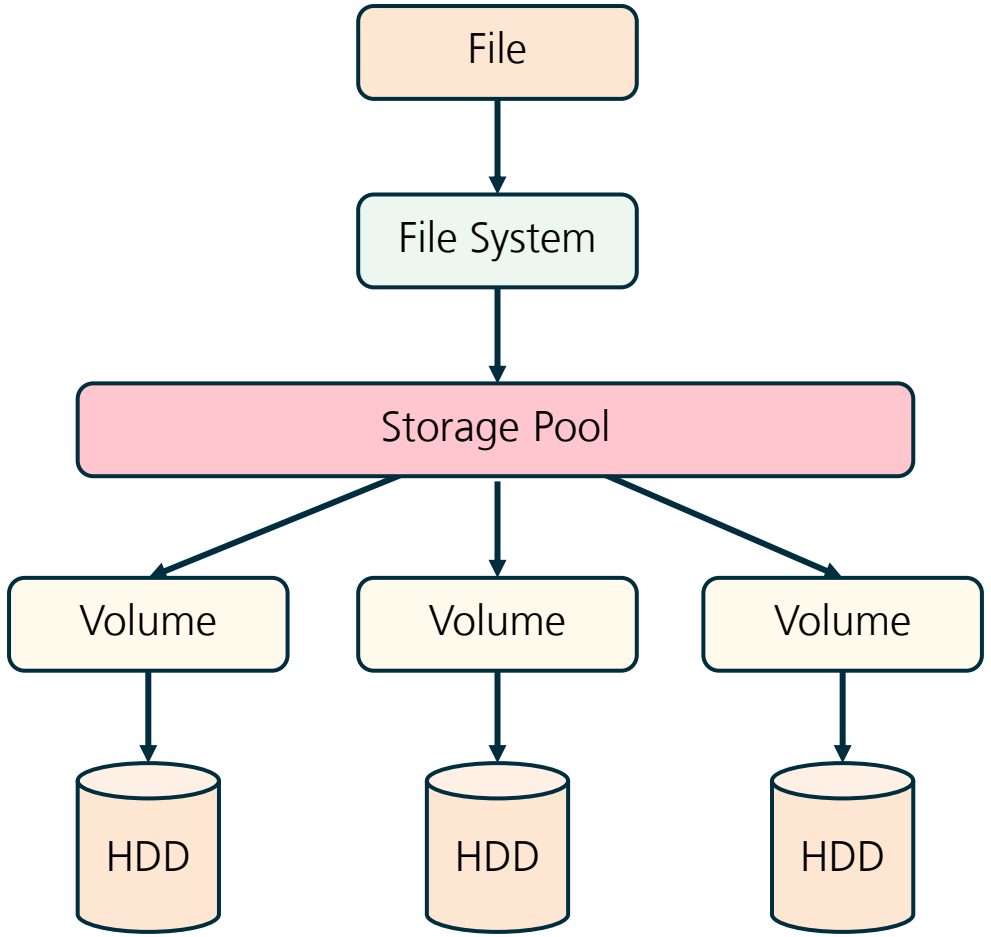
Novel Methods

Snapshots

Lower
File Slack



Volume Management



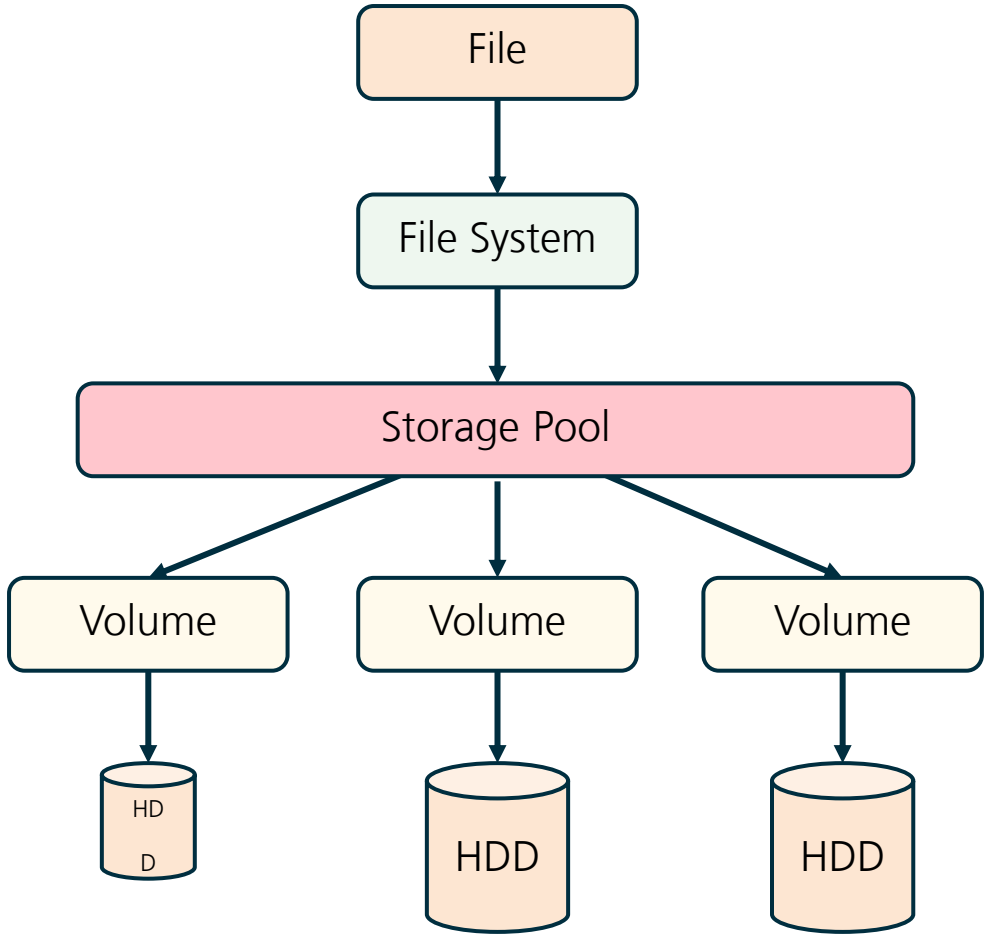
Data Hiding in File Systems

Novel Methods

Snapshots

Lower
File Slack

Volume Management



Data Hiding in File Systems

Contributions

Current State

Novel Methods

Standardized Corpus

Data Hiding in File Systems

Contributions

Current State

Novel Methods

Standardized Corpus

Data Hiding in File Systems

Standardized Corpus

Scenario	Description
Scenario 1	Data hidden in general file slack space.
Scenario 2	Data hidden in file timestamps.
Scenario 3	Data hidden in bad units.
Scenario 4	Data hidden in additional data units.
Scenario 5	Data hidden in slack space structures.
Scenario 6	Data hidden in reserved areas of structures.
Scenario 7	Data hidden in hidden snapshots.
Scenario 8	Data hidden in lower file slack.
Scenario 9	Data hidden in the slack space of physical members in pooled file systems.

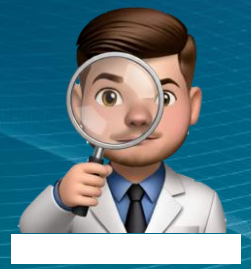
https://github.com/fkie-cad/hide-and-seek-dataset

Data Hiding in File Systems

Conclusion

- **Data hiding techniques** for file systems …
  - … exist and are practical
  - … have been actively researched for years
  - … follow a few recurring patterns (slack space, reserved/unused areas, metadata/structure misuse)
  - … are highly file-system specific
- **Novel data hiding techniques** for file systems …
  - … will keep appearing as file systems evolve and add features/structures
  - … remain file-system specific, exploiting new features and structures
- **Detection methods** …
  - … are essential and must be file-system aware
  - … should be evaluated on standardized, reproducible datasets covering diverse techniques and file systems
  - … need to be developed

Thank you for your attention!



Dr. Jan-Niclas Hilgert
Cyber Analysis & Defense
jan-niclas.hilgert@fkie.fraunhofer.de

Fraunhofer Institute for Communication,
Information Processing and Ergonomics FKIE
Zanderstrasse 5 | 53177 Bonn

www.fkie.fraunhofer.de