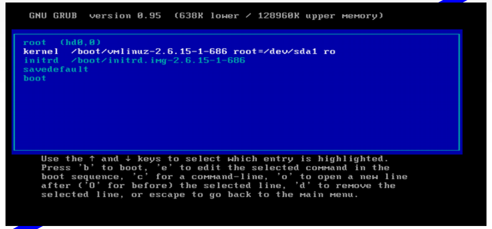
Disk & Filesystem Forensics

Davide Balzarotti
davide.balzarotti@eurecom.fr

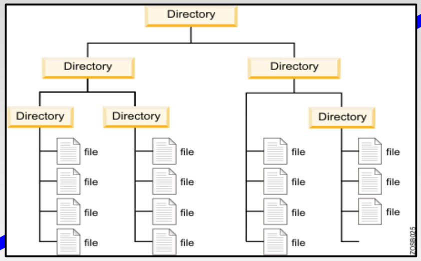
Physical Disks



```
  GNU GRUB  version 0.95  (638K lower / 128960K upper memory)

 root   (hd0,0)
 kernel  /boot/vmlinuz-2.6.15-1-686 root=/dev/sda1 ro
 initrd  /boot/initrd.img-2.6.15-1-686
 savedefault
 boot

    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press 'b' to boot, 'e' to edit the selected command in the
    boot sequence, 'c' for a command-line, 'o' to open a new line
    after ('O' for before) the selected line, 'd' to remove the
    selected line, or escape to go back to the main menu.
```

Volumes and Partitions

File Systems

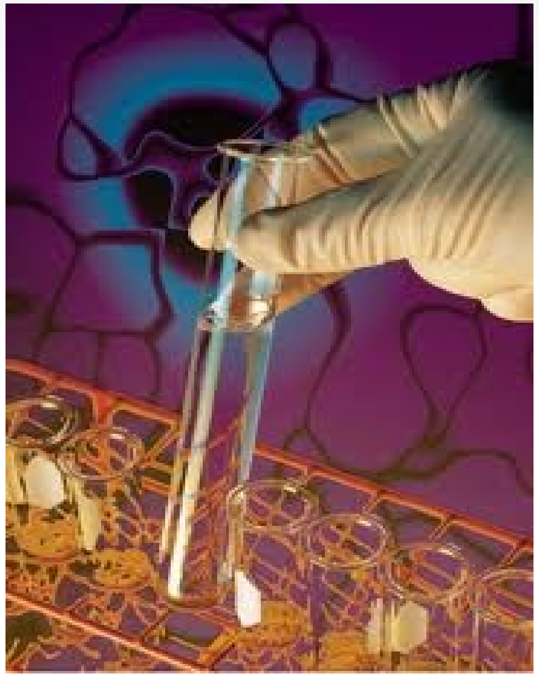


Files

Hard Disks

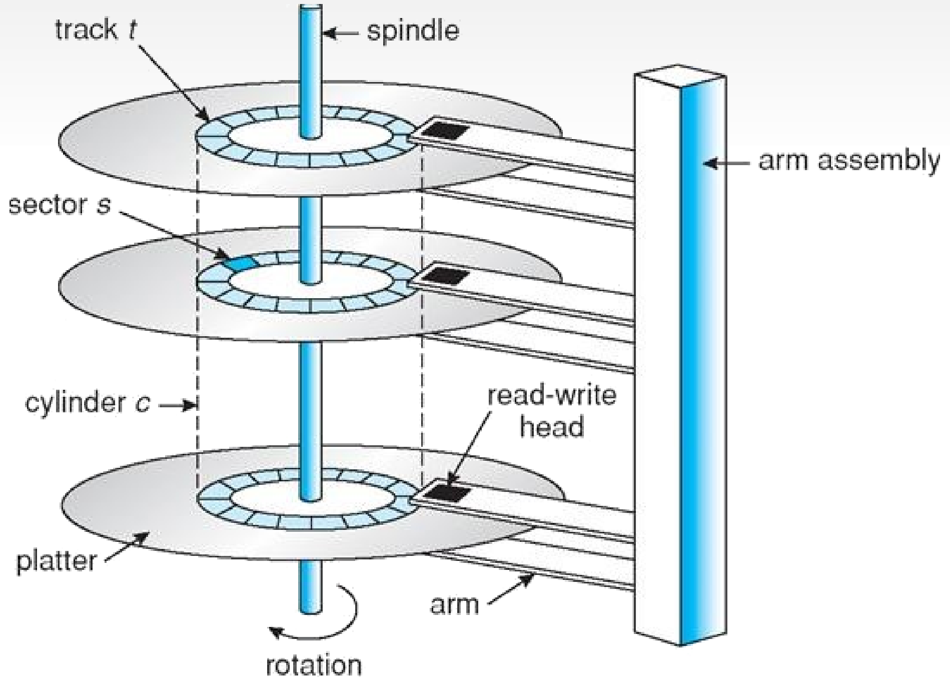
Summary

- Physical Disks
  - Disk geometry
  - Hidden and Reserved areas
  - Flash drives
  - SMART
  - Disk acquisition
- Volumes and Partitions
- File systems
- The Malware Corner



Hard Disks

- Devices that provide non-volatile data storage
- Data is stored on one or more circular platters that are stacked on top of each other and spin at the same time
- Hard disks are organized in cylinders, heads, and sectors
  (referred as *disk geometry*)



Disk Addressing

- You can check the information about your disk with

```
$ hdparm -I /dev/sda
```

- Sectors are the smaller addressable data units
- To address each sectors, Disks use a linear access scheme called *Logical Block Addressing* (LBA)
  - Each sector is assigned a unique sector number
  - The mapping between logic numbers and physical sectors is done by the disk controller

Hidden Areas

- Disks can have reserved areas that are hidden from the operating system
- Host Protected Area (HPA)
  - used by computer makers to store booting and diagnostic software, and/or a copy of the originally provided operating system for recovery purposes

    ```
    $ hdparm -N /dev/sda
    ```

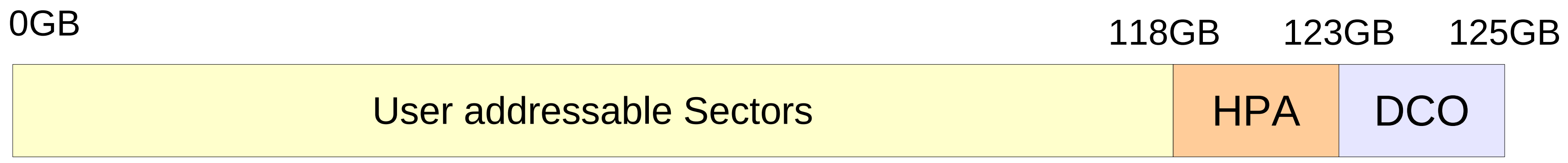
Hidden Areas

- Disks can have reserved space hidden to the operating system
- Host Protected Area (HPA)
  - used by computer makers to store booting and diagnostic software, and/or a copy of the originally provided operating system for recovery purposes

```
$ hdparm -N /dev/sda
```

- Device Configuration Overlay (DCO)
  - Introduced to allows vendors to get hard disks from different manufacturers and then configure them to have the same number of sectors

```
$ hdparm --dco-identify /dev/sda
```



Reserved Service Area

- Additional storage area that is not visible from the outside and that is managed by the disk firmware
- This area contains
  - The image of the firmware itself
  - A number of spare sectors to remap defective or damaged sectors
  - SMART (Self-Monitoring, Analysis, and Reporting Technology) logs

This area is NOT accessible through the LBA addressing scheme !!

Reserved Service Area

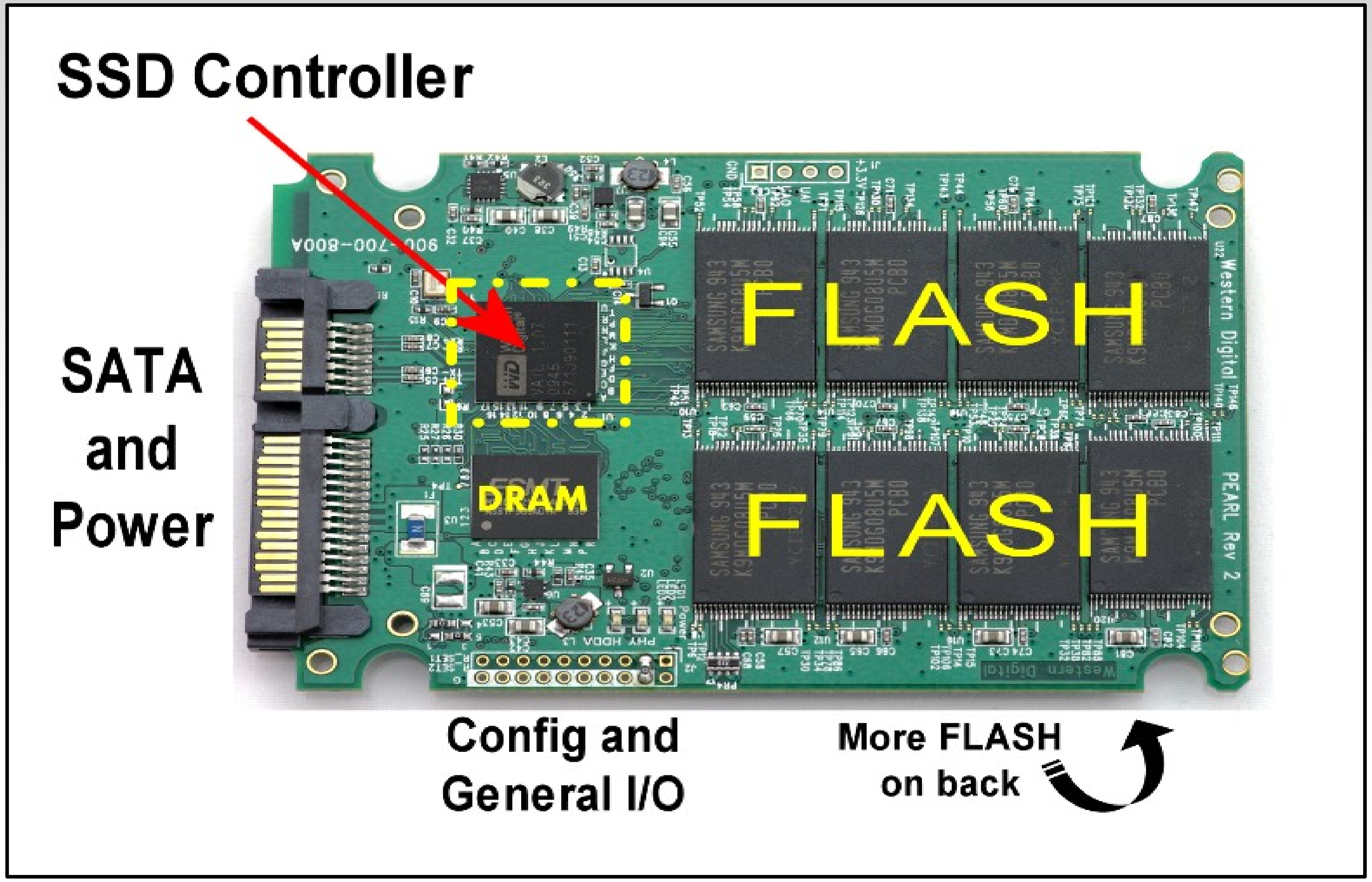
- When the firmware detects a malfunctioning sector, it remaps the logical sector to a different physical sector
  - The p-list (permanent list) contains the bad sectors encountered during the post-manufacture testing
  - The g-list (growth list) marks the sectors that are detected as bad while the drive is in use
- There is no easy (nor standard) way to access the content of these lists
  - Ad-hoc solutions may be available for certain disks models
  - Expensive hardware recovery systems exist that can access the content of the service area

Impact on Forensics

- HPA can be disabled before acquisition
- DCO can be permanently removed
  - The operation is irreversible
  - It may cause data losses (?)

- Bad sectors can contain leftover data
  - Hard to recover without the right equipment
  - A serious problem for military or disks used to store highly sensitive data

Flash Drives



Flash Drives

- They have the same *interface* of an HDD but they work in a completely different way
- Data is stored in transistors arranged in pages of 4K each
  - Each page can be written only a limited number of times (~100K)
  - Therefore, flash drives use a wear-levelling technique to distributed the data evenly across the entire disk
  - These transformations are done by the Flash Translation Layer (FTL) controller
- Read and Write (set bits to 0) are fast, but Erasing (reset bits to 1) is slow and can only be applied to entire blocks (e.g, 256K each)
  - In-place update is not possible
  - Flash disks normally implement some form of garbage collection mechanism to delete the pages that are no longer needed

Impact on Forensics

- If the garbage collector starts during an acquisition it can
  - Potentially remove interesting data
  - Change the state of the disk → hash will not match anymore !!
  - The image will contains artifacts that are no longer on the disk (and it is impossible to prove that they were ever there)

- Forensic researchers call this process “self-corrosion"
- The wear leveling algorithm makes very hard to properly remove data from a SDD disk

“*Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery?*”
Graeme B. Bell and Richard Boddington

“*Reliably Erasing Data from Flash-Based Solid State Drives*”
Michael Wei, Laura M. Grupp, Frederick M. Spada, Steven Swanson

More on Self-Corrosion

- The garbage collector is triggered by the TRIM command
  - Issued by the operating system to notify the drive about the blocks that are no longer in use
  - Performed when users delete files, format the disk, modify a partition, …

http://articles.forensicfocus.com/2014/09/23/recovering-evidence-from-ssd-drives-in-2014-understanding-trim-garbage-collection-and-exclusions/

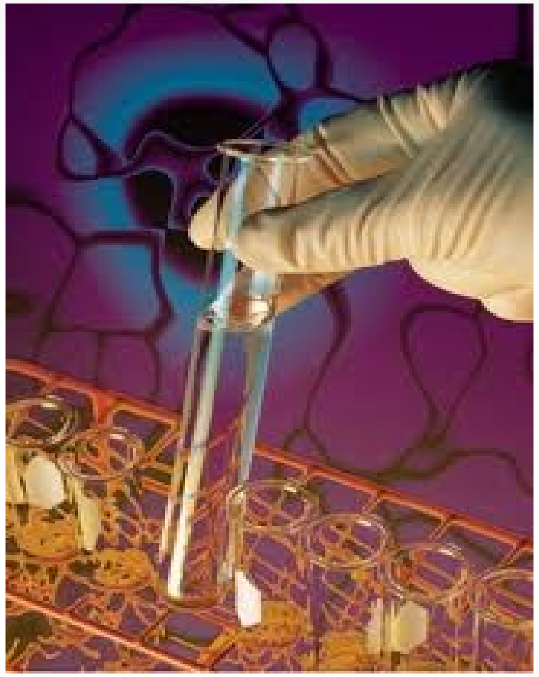
More on Self-Corrosion

- The garbage collector is triggered by the TRIM command
  - Issued by the operating system to notify the drive about the blocks that are no longer in use
  - Performed when users delete files, format the disk, modify a partition, …
- So, the data is not lost if the TRIM command is not issued, or if the TRIM protocol is not supported by any link of the chain
  - Linux added TRIM support in kernel 2.6.33 but it is not active by default (require the “discard” mount option)
  - Ext3 does not support automatic trimming
  - Microsoft added TRIM support in Windows 7 only for ordinary (SATA) drives and only for the NTFS filesystem
    - No support for PCI-Express SSDs, for USB external disks
    - No support for RAID

http://articles.forensicfocus.com/2014/09/23/recovering-evidence-from-ssd-drives-in-2014-understanding-trim-garbage-collection-and-exclusions/

Summary

- Physical Disks
  - ✔ Disk geometry
  - ✔ Hidden and Reserved areas
  - ✔ Flash drives
  - SMART
  - Disk acquisition
- Volumes and Partitions
- File systems
- The Malware Corner



SMART

- Allows the disk to perform self-tests and track performance, statistics, and errors
  - Implemented by most of the disk since 1995
- Can be turned off, but some disks will keep collecting information anyway :(
- The implementation and the interpretation of each value is vendor specific
  - E.g., some disks measure the time in minutes, some in hours
  - Be very careful when you use SMART information in forensic

(Forensic aside, it is very useful to check the SMART error logs in case of disk failure)

SMART

```
$ smartctl -x /dev/sda

=== START OF INFORMATION SECTION ===
Model Family:     Samsung based SSDs
Device Model:     SAMSUNG SSD PM800 2.5" 128GB
Serial Number:    S0DTNEAZ708153
LU WWN Device Id: 5 0000f0 056424431
Firmware Version: VBM25D1Q
User Capacity:    128,035,676,160 bytes [128 GB]
Sector Size:      512 bytes logical/physical
Device is:        In smartctl database
ATA Version is:   7
ATA Standard is:  ATA/ATAPI-7 T13 1532D revision 1
Local Time is:    Wed Feb 22 11:39:16 2012 CET
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

SMART

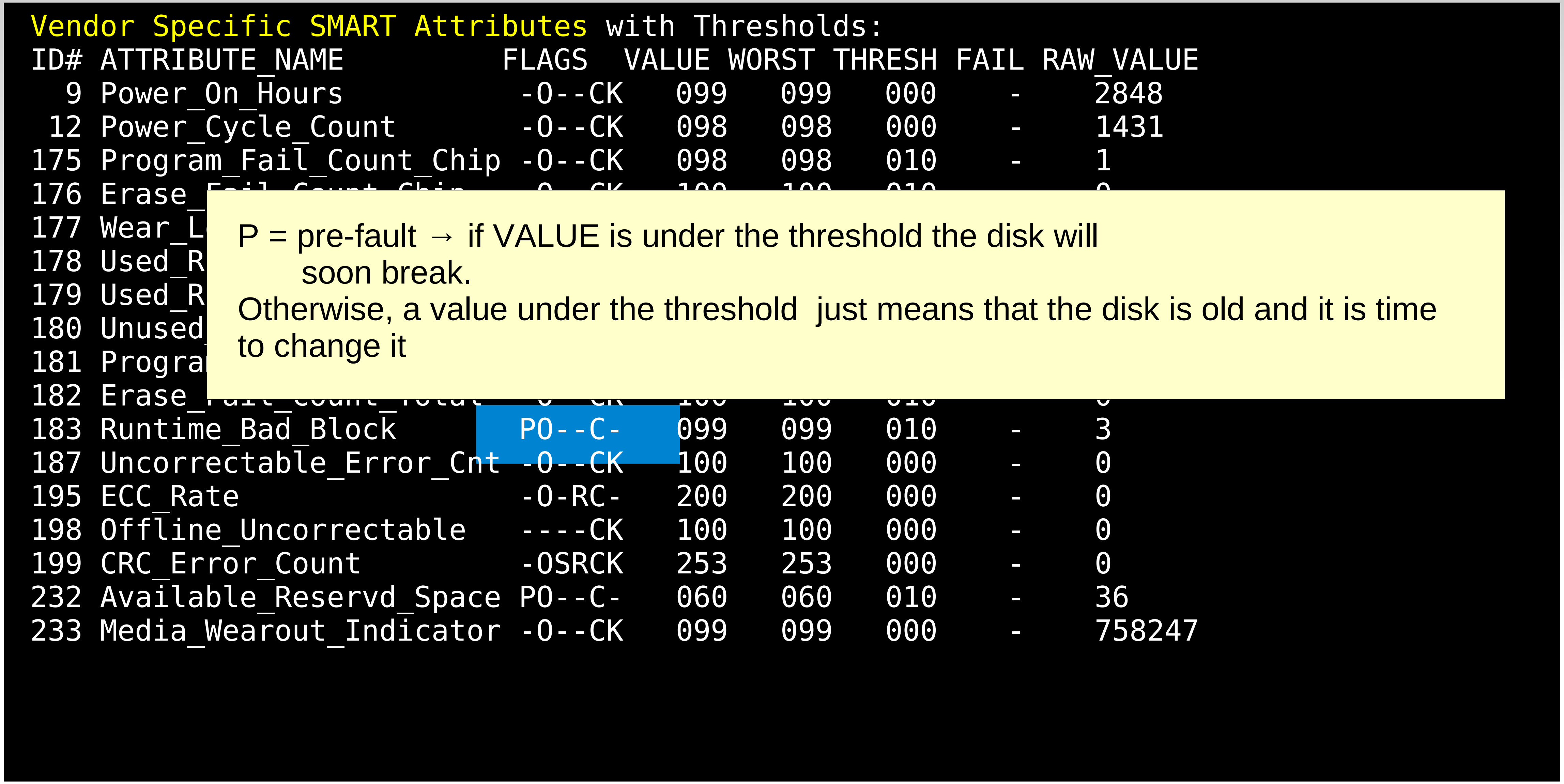
```
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAGS    VALUE WORST THRESH FAIL RAW_VALUE
  9 Power_On_Hours           -O--CK   099   099   000    -    2848
 12 Power_Cycle_Count        -O--CK   098   098   000    -    1431
175 Program_Fail_Count_Chip -O--CK   098   098   010    -    1
176 Erase_Fail_Count_Chip   -O--CK   100   100   010    -    0
177 Wear_Leveling_Count     PO--C-   097   097   017    -    140
178 Used_Rsvd_Blk_Cnt_Chip  PO--C-   060   060   010    -    24
179 Used_Rsvd_Blk_Cnt_Tot   PO--C-   092   092   010    -    294
180 Unused_Rsvd_Blk_Cnt_Tot PO--C-   092   092   010    -    3546
181 Program_Fail_Cnt_Total  -O--CK   099   099   010    -    3
182 Erase_Fail_Count_Total  -O--CK   100   100   010    -    0
183 Runtime_Bad_Block       PO--C-   099   099   010    -    3
187 Uncorrectable_Error_Cnt -O--CK   100   100   000    -    0
195 ECC_Rate                -O-RC-   200   200   000    -    0
198 Offline_Uncorrectable   ----CK   100   100   000    -    0
199 CRC_Error_Count         -OSRCK   253   253   000    -    0
232 Available_Reservd_Space PO--C-   060   060   010    -    36
233 Media_Wearout_Indicator -O--CK   099   099   000    -    758247
```

SMART

```
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAGS    VALUE WORST THRESH FAIL RAW_VALUE
  9 Power_On_Hours           -O--CK   099   099   000    -    2848
 12 Power_Cycle_Count        -O--CK   098   098   000    -    1431
175 Program_Fail_Count_Chip -O--CK   098   098   010    -    1
176 Erase_Fail_Count_Chip    -O--CK   100   100   010    -    0
177 Wear_Le
178 Used_Rs
179 Used_Rs
180 Unused_
181 Program
182 Erase_Fail_Count_Total   -O--CK   100   100   010    -    0
183 Runtime_Bad_Block        PO--C-   099   099   010    -    3
187 Uncorrectable_Error_Cnt -O--CK   100   100   000    -    0
195 ECC_Rate                 -O-RC-   200   200   000    -    0
198 Offline_Uncorrectable    ----CK   100   100   000    -    0
199 CRC_Error_Count          -OSRCK   253   253   000    -    0
232 Available_Reservd_Space PO--C-   060   060   010    -    36
233 Media_Wearout_Indicator -O--CK   099   099   000    -    758247
```

RAW = real value (unknown unit)
VALUE = value normalized by the manufacturer on a 0-254 scale
THRESH = below this value, there is a problem

SMART

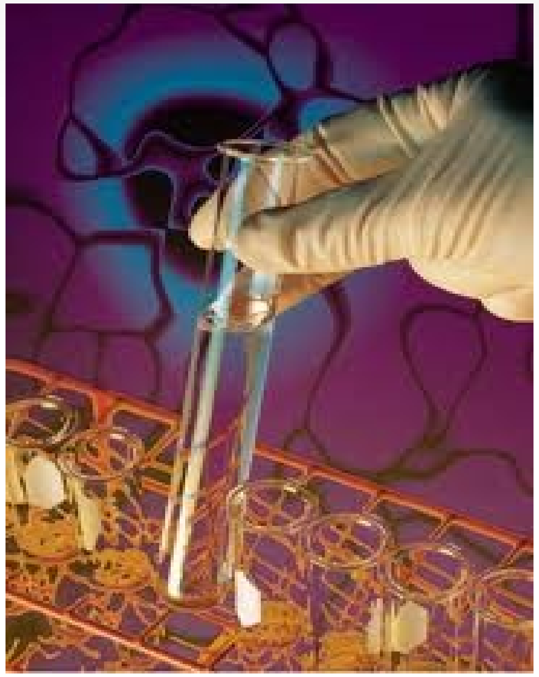


```
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAGS    VALUE WORST THRESH FAIL RAW_VALUE
  9 Power_On_Hours           -O--CK   099   099   000    -    2848
 12 Power_Cycle_Count        -O--CK   098   098   000    -    1431
175 Program_Fail_Count_Chip -O--CK   098   098   010    -    1
176 Erase_Fail_Count_Chip   -O--CK   100   100   010    -    0
177 Wear_L
178 Used_R
179 Used_R
180 Unused
181 Progra
182 Erase_Fail_Count_Total  -O--CK   100   100   010    -    0
183 Runtime_Bad_Block        PO--C-   099   099   010    -    3
187 Uncorrectable_Error_Cnt -O--CK   100   100   000    -    0
195 ECC_Rate                 -O-RC-   200   200   000    -    0
198 Offline_Uncorrectable    ----CK   100   100   000    -    0
199 CRC_Error_Count          -OSRCK   253   253   000    -    0
232 Available_Reservd_Space PO--C-   060   060   010    -    36
233 Media_Wearout_Indicator -O--CK   099   099   000    -    758247
```

P = pre-fault → if VALUE is under the threshold the disk will soon break.
Otherwise, a value under the threshold just means that the disk is old and it is time to change it

Summary

- Physical Disks
  - ✔ Disk geometry
  - ✔ Hidden and Reserved areas
  - ✔ Flash drives
  - ✔ SMART
  - Disk acquisition
- Volumes and Partitions
- File systems
- The Malware Corner



Acquisition

- The goal is to obtain a perfect copy of the disk
- Also record available metadata
  - Disk serial number and manufacturer
  - SMART information
  - Who, when, and where the image was made
- Acquisition is a crucial part of the investigation process
- The National Institute of Standards and Technology (NIST) performs tests on common acquisition tools
  - Results are publicly available online:

    http://nij.ncjrs.gov/App/publications/Pub_search.aspx?category=99&PSID=55

Acquisition

- Logical acquisition
  - Copy of the file system files preserving timestamps
  - Granularity: **file**
  - You can only get what the filesystem wants to show (e.g., no deleted files and unallocated space)
- Physical acquisition
  - Exact copy of a storage device
  - Granularity: **sector**
  - You can only get what the disk wants to show

Acquisition Procedure

- Remove the disk and connect it to your system through a physical write blocker
  - or use an hardware duplicator if you have one
- Software RAID is not a big issue, but the presence of strange RAID controllers may complicate your ability to access / interpret the data.
  In this case:
  - Boot the system in a safe environment (e.g., using a Linux live CD)
  - Perform a forensic image through the network
  - Imaging a single volume (e.g., `/dev/sda1`) could be better than imaging the whole disk (e.g., `/dev/sda`)

Disk Image Formats (raw)

- RAW
  - Easier to collect and analyze with all tools
  - Very large
  - Does not store metadata
- Split RAW
  - Solve the size problem by splitting the raw image in multiple files
  - Not all tools support it

Disk Image Formats (forensic containers)

- Expert Witness Format (EWF)
  - Native format of the Encase forensics application
  - Compress data and store some metadata
  - Checksum of each 32KB chunk + hash of the entire image
  - Record position of bad blocks (filled with zeros)
  - Not supported by all tools
- AFF
  - Open source, extensible design
  - Not very well supported
  - Compress data, support encryption and signatures

dd

- Low-level copy tool, part of the GNU Coreutils package
- Can be used to create RAW images of a file or a device

```
$ dd  if=/dev/sda  of=disk.raw  <options>
```

`bs=4096`	(block size) number of bytes read and written at the same time (default=512)
`skip=64`	skip the first 64 blocks in the source file
`count=10`	copy only 10 blocks (instead of until the end of the file)

- Sending a INFO (or USR1) signal to a running `dd` process makes it print I/O statistics to standard error

```
$ dd if=/dev/zero of=/dev/null & pid=$!
$ kill -s INFO $pid;
```

Errors During Acquisition

- Dealing with bad blocks
  - ✗ Skip them
    (very bad, the total size will not mach and you dont know why)
  - ✗ Stop the procedure and report the error
    (you still need to acquire the data after the defective sector)
  - ✔ Mark the position, replace them with zeros, and continue the acquisition

- Acquisition using `dd`

`bs=512`	copy one sector at the time
`conv=noerror,sync`	continue after read error, padding with NULL bytes each block (i.e, replace bad sectors with zeros)
`iflag=direct`	direct I/O, bypassing the kernel cache

Forensic Versions of `dd`

- `dd` was not designed with forensics in mind
- Specific versions of `dd` exist to support:
  - On the fly piecewise hashing
  - Error log to keep track of defective sectors
  - Progress report
  - Split output in multiple raw files
- `dcfldd` (fork of `dd`)
- `dc3dd` (patch for `dd`)

The Sleuth Kit

- Opensource digital forensic tool composed by a library and several command line tools
  - Supports RAW, split RAW, EWF, and AFF image formats
- Tools are organized in several layers:
  - image-tools, volume-tools, filesystem-tools, block-tools, inode-tools, file-tools, journals-tools
- Image tools:

  `img_stat` – display some statistics about the image file

  `img_cat` – print the image content in RAW format

Booting an Image

- Sometimes it is convenient to boot the OS contained in a disk image using an Emulator (e.g., QEMU, KVM, VirtualBox, VmWare, ...)
- `xmount` creates on-the-fly a virtual file system using FUSE (Filesystem in Userspace) that contains a virtual representation of the input image
  - Input formats include: dd, ewf, aff
  - Virtual formats include: raw, Virtualbox, and VmWare
- `xmount` also supports *virtual write access* redirected to a cache file
- Windows images do not boot if the hardware is changed
  - Use `opengates` to "open" the hardware limitation of Windows installations (works with all version up to Windows 7)

Volumes & Partitions

Partitions & Volumes

- Disks are normally organized in smaller logical units, called partitions (or slices)
  - Partitions are not required to fill all the entire space available on the disk

Partitions & Volumes

- Disks are normally organized in smaller logical units, called Partitions (or slices)
    - Partitions are not required to fill all the entire space available on the disk
- Volumes are disk areas presented to the OS as a single container with a given filesystem
    - Most of the partitions are associated to a single volume but...

      … Partitions may exist without a volumes (e.g., ext3 partitions under Windows)

      … Volumes may exist without partitions (e.g., a floppy disk)
    - "*Extended Partitions*" may contain multiple volumes
- Volumes are the ones managed by `mount` in Unix or associated to a letter drive in Windows

Disk Partitioning

- Different ways exist to divide a disk into partitions
  - Apple partition map (APM)
  - BSD Disklabel
  - IBM PC Partition Table stored in a Master Boot Record (**MBR**)
    - Standard for most Linux/Windows systems for over 25 years
    - 512 bytes located in the first logical sector of the disk
    - Contains a partition table, a bootstrapping code, and an optional disk signature
  - GUID Partition Tables (GPT)
    - INTEL standard proposed to replace MBR
    - Used natively by EFI/UEFI firmware

MBR - Partition Table

- A Partition Table can contains a maximum of 4 primary partitions (or 3 primary and 1 extended)
  - Partitions are expressed in [Cylinders, Heads, Sectors] and they are limited to 2^{32}-1 sectors (2 TB)
  - Extended partitions contains Extended Boot Records to manage the subdivision in a number of logical partitions

- The original IBM bootstrap procedure consisted in:
  - The BIOS loads and executes a real-mode code stored in the MBR
  - The code can load an additional second stage boot loader from disk
  - The code finds the partition with the active flag set and loads and executes its first sector (named ***Volume Boot Record***) that contains Operating System specific code
  - Many extensions and alternative implementations exist

GPT vs MBR

- MBR can only handle 2T disks (or 16TB with 4k sectors) and four primary partitions

- **GPT** (part of the UEFI specification) introduces
  - Fault-tolerance by keeping copies of the partition table in the first and last sector on the disk
  - Almost unlimited disk size (2^{64} sectors)
  - Primary partitions only, but a large number of that (>128)
  - Unique identifiers for each partition

Sleuth Kit Volume Layer

`mmstat` – display the partitioning scheme adopted

`mmls` – list the partitions (e.g., the partition table)

`mmcat` – print the content of a volume to the standard output

```
$ mmls ext-part-test-2.dd
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
 Slot     Start         End            Length         Description
00:   Meta     0000000000   0000000000   0000000001   Primary Table (#0)
01:   -----    0000000000   0000000062   0000000063   Unallocated
02:   00:00    0000000063   0000052415   0000052353   DOS FAT16 (0x04)
03:   00:01    0000052416   0000104831   0000052416   DOS FAT16 (0x04)
04:   00:02    0000104832   0000157247   0000052416   DOS FAT16 (0x04)
05:   Meta     0000157248   0000312479   0000155232   DOS Extended (0x05)
06:   Meta     0000157248   0000157248   0000000001   Extended Table (#1)
```

Lost and Damaged Partitions

- `sigfind` – look for byte patterns in a file
  - Part of the Sleuth Kit
  - Contains pre-defined patterns for many file-systems and partition types
  - Can be used to find lost or removed partitions
    (but then they must be extracted manually)

- `testdisk` – scan and repair disk partitions
  - Can list the partitions and fix them if they are malformed

- `gpart` – scan the disk and “guess” the partition table

Mount a Disk Image

- Associate a file to a loop device
  - The device can then be mounted like any other device
  - Or you can play with it without mounting it

```
$ losetup -o32256 /dev/loop0 /path/to/c.img
$ losetup -d /dev/loop0
```

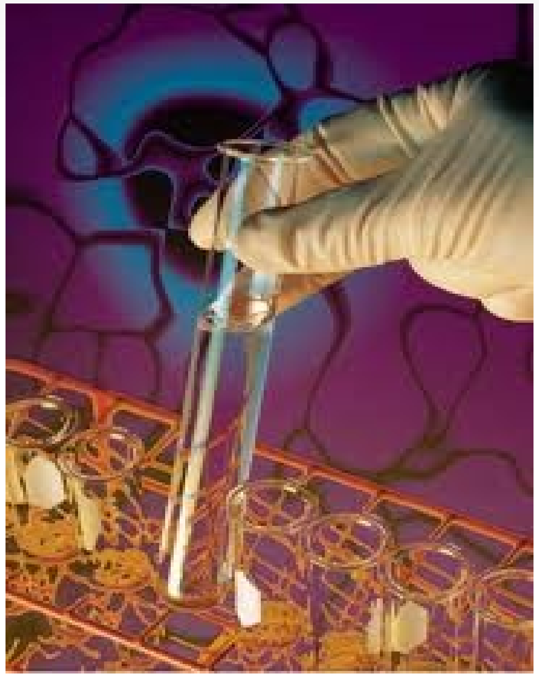
- Mount a file

```
$ mount -o loop,ro,offset=start_sector*sector_size file mountpoint
```

FileSystems

Summary

- ✓ Physical Disks
- ✓ Volumes and Partitions
- File systems
  - General concepts
  - EXT
  - Slack space
  - (Un)Deleting files in EXT2/3
  - Data erasure
  - NTFS
  - Encryption
- The Malware Corner



Filesystem

- The way computers organize, name, store, and manipulate files, directories, and their metadata
  - Designed to organize data in a way that can be easily stored and retrieved by users and applications
- Several dozen FSs exist, often associated to different OSs
  - File Allocation Table (FAT) and New Technology File System (NTFS) for Windows
  - Second, Third, and Forth Extended Filesystems (ext2, ext3, ext4) for Linux
- Note that a FS can be smaller than the volume that contains it
  - The difference (if exists) is called volume slack and it can be used to hide data

Some Common Features

- Data is stored in allocation units (called blocks)
  - Normally the size if fixed and it is decided when the FS is created
- Over time, files and free space get fragmented
  - The file content is stored in non-consecutive blocks
  - It severely impact certain FS (e.g., FAT), but it is less of an issue in others (e.g., ext3)
- Filesystems often keep a list of bad blocks (found to be defective when reading) and do not use them to store information

Journaling

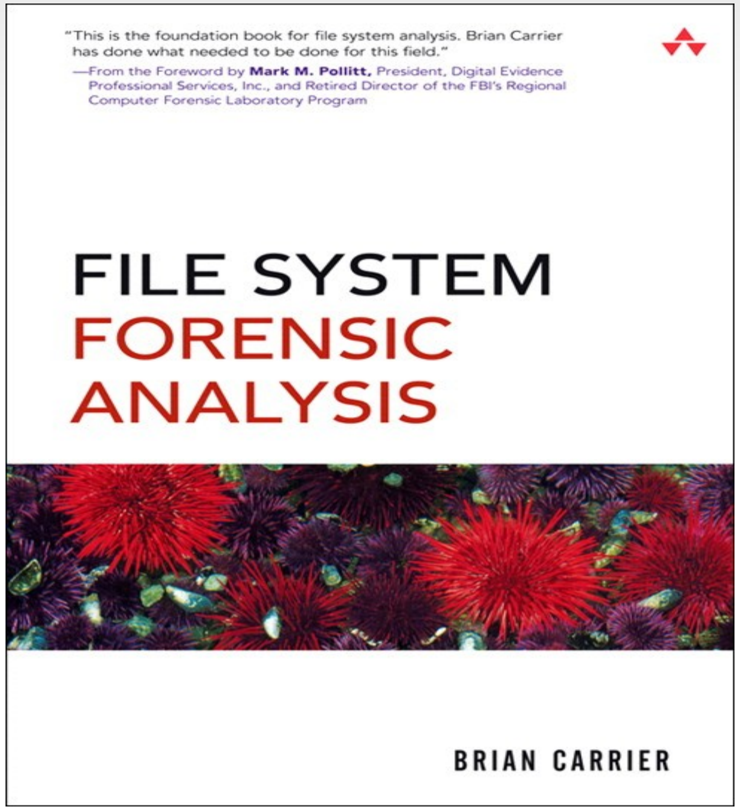
- Problem:
  - In case of a system crash (or sudden loss of power) inconsistencies can lead to data corruption
- Solution:
  - Maintain a journal to log changes of the file system in a circular buffer
  - Commit the journal to the filesystem at periodic intervals
  - In case of a crash, the journal can be used as a checkpoint to recover unsaved information
- Typically only metadata is journaled, and data blocks are written directly to the disk (for performance reasons). However data-journaling is also available as an option

Why FS Forensics?

- Why performing FS forensics when you can mount the FS and examine the files?
  - To find information hidden *outside* traditional files
  - To recover *deleted* files

Why FS Forensics?

- Why performing FS forensics when you can mount the FS and examine the files?
  - To find information hidden *outside* traditional files
  - To recover *deleted* files
- Sleuthkit provides many tools to help the FS forensic process
- However, you really need to understand the file system internals to do a good job
  - Example?
    Do you know that the 3rd sector of a FAT32 FS is not used and <u>should</u> contain only zeros?


"This is the foundation book for file system analysis. Brian Carrier has done what needed to be done for this field."
—From the Foreword by **Mark M. Pollitt,** President, Digital Evidence Professional Services, Inc., and Retired Director of the FBI's Regional Computer Forensic Laboratory Program

FILE SYSTEM
FORENSIC
ANALYSIS

BRIAN CARRIER

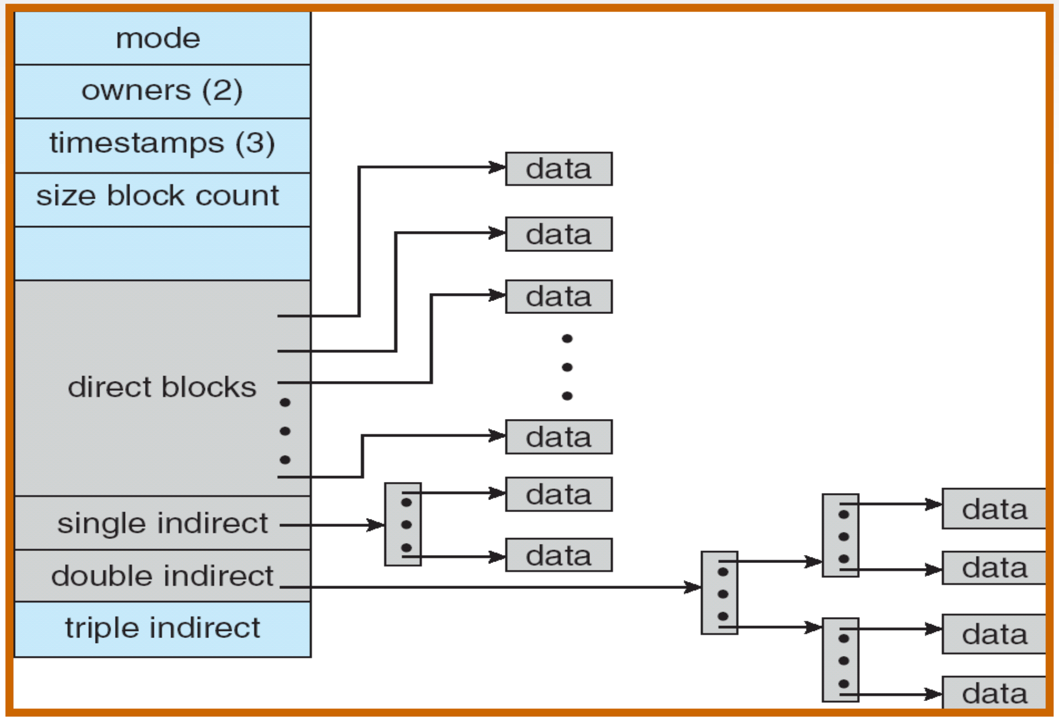
EXT2 Filesystem

- The space is split in blocks, organized into block groups
  - If possible, metadata and data of a single file are contained in the same block group

- Each block group contains
  - A copy of the superblock (metadata of the filesystem)
  - A block group descriptor table (metadata of the group)
  - A block bitmap (specify which block is free or in use)
  - An inode bitmap (specify which inode is free or in use)
  - An inode table   (array of inodes, the number is specified in the superblock)
  - The actual data blocks

Index Nodes

- An index node (inode) stores information about files and directories
  - Ownership, permissions, file size, file type, timestamps, ...
  - Not the filename (that is saved in the directory)
  - The list of blocks that contain the actual data



Beyond Ext2

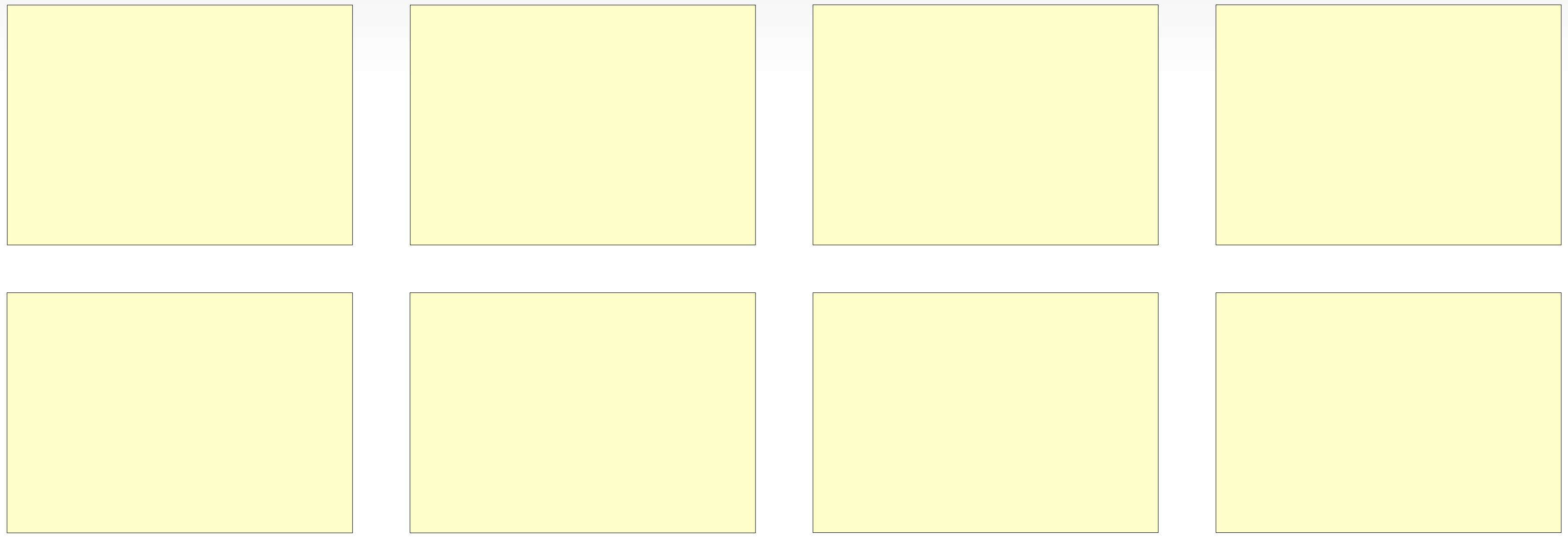
- Ext2 was introduced in 1993
- Ext3 was introduced in 2001 and added
  - A journal
  - An online file system growing mechanism
  - An HTree indexing for larger directories
- Ext4 was introduced in 2008 and added
  - Support for larger files and filesystems size
  - Improved timestamps (nanoseconds and longer in the future)
  - Extents to index large amount of blocks in one shot
- F2FS introduced in 2013 for flash file systems …..

Slack Space

- Scenario:
  - The filesystem has a block size of 4 Kilobyte

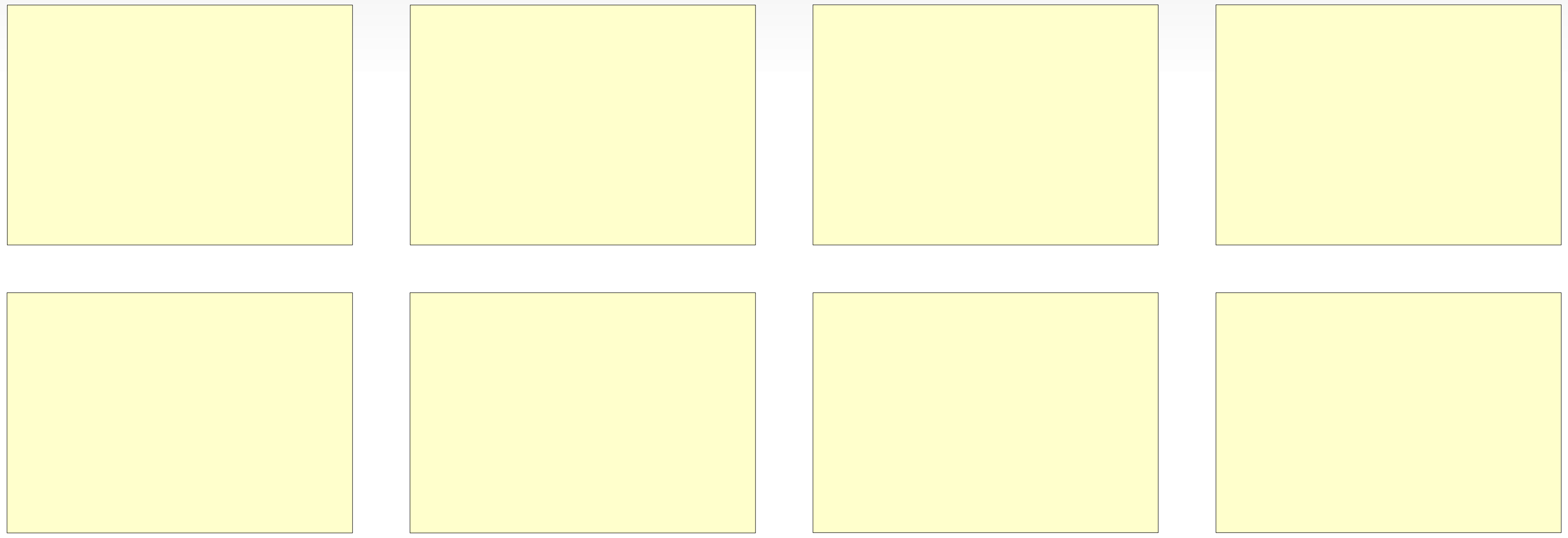
Slack Space

- Scenario:
  - The filesystem has a block size of 4 Kilobyte
  - The disk has a sector size of 512 byte



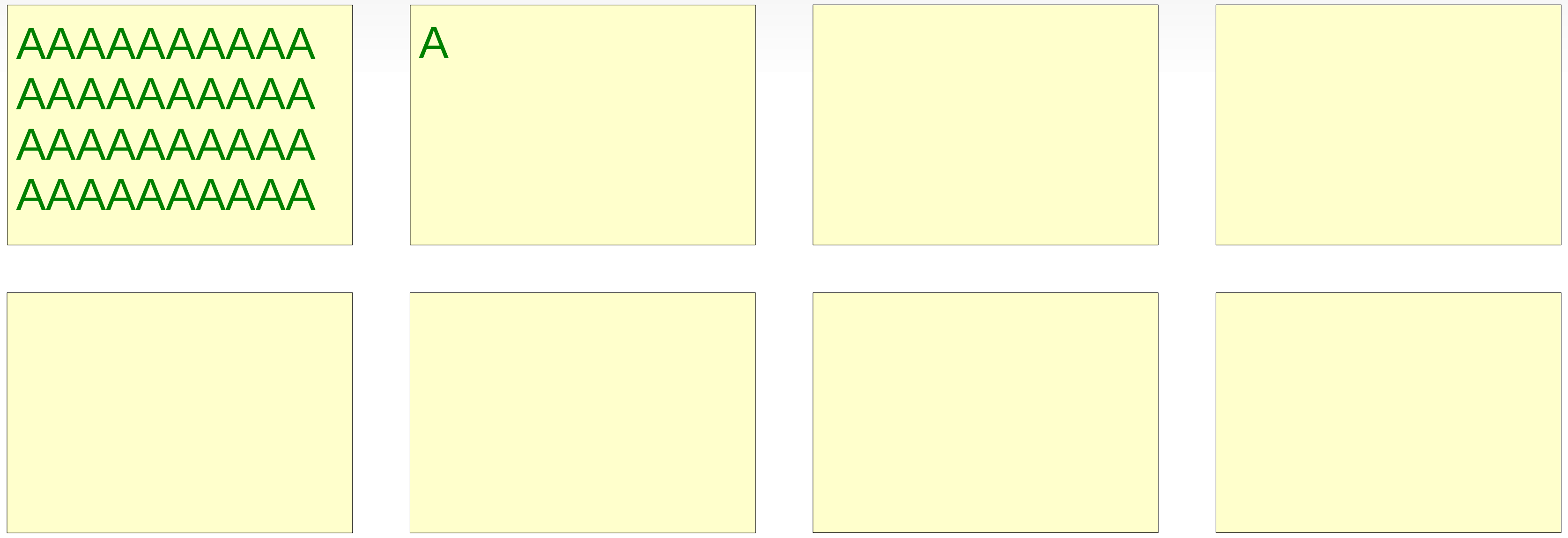
Slack Space

- Scenario:
  - The filesystem has a block size of 4 Kilobyte
  - The disk has a sector size of 512 byte
  - The user create a file containing 513 bytes (all “A” characters)



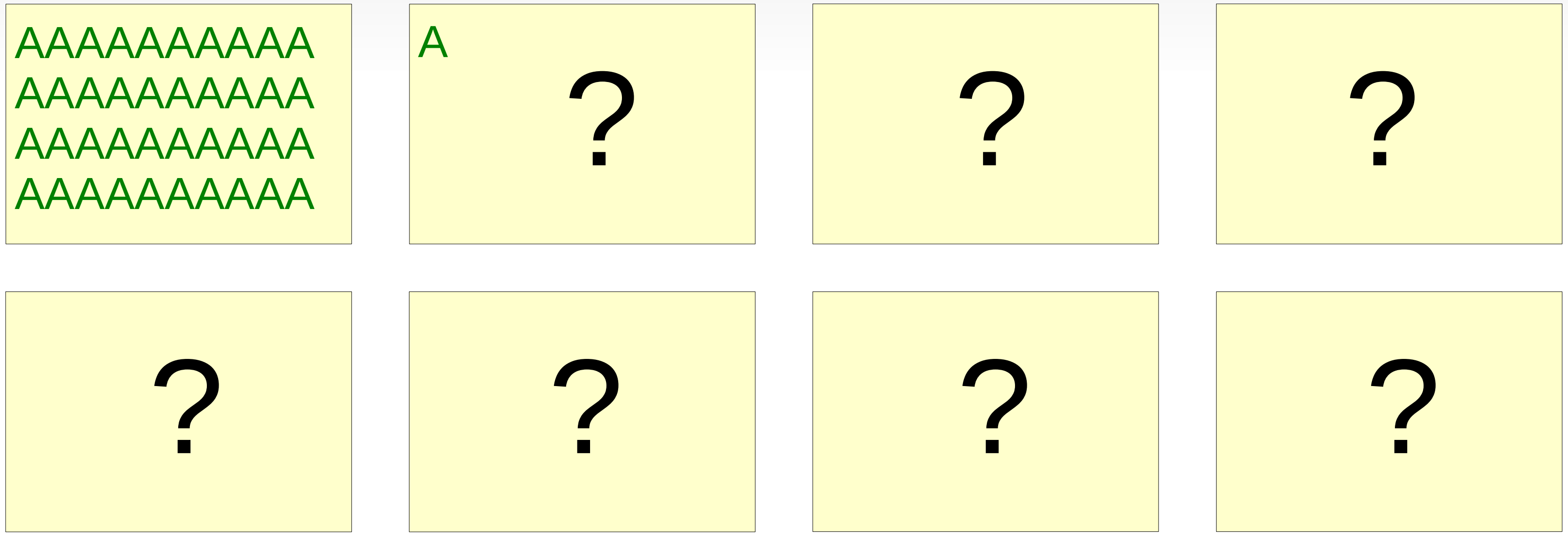
Slack Space

- Scenario:
  - The filesystem has a block size of 4 Kilobyte
  - The disk has a sector size of 512 byte
  - The user create a file containing 513 bytes (all “A” characters)



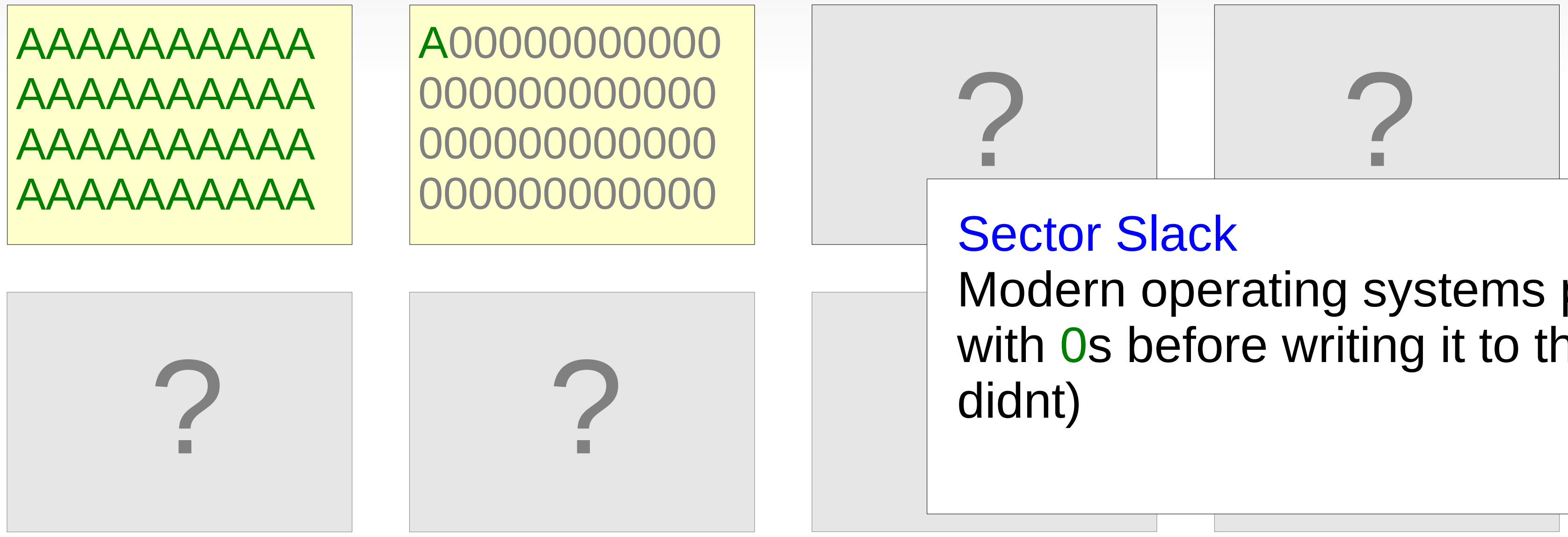
Slack Space

- Scenario:
  - The filesystem has a block size of 4 Kilobyte
  - The disk has a sector size of 512 byte
  - The user create a file containing 513 bytes (all "A" characters)



Slack Space

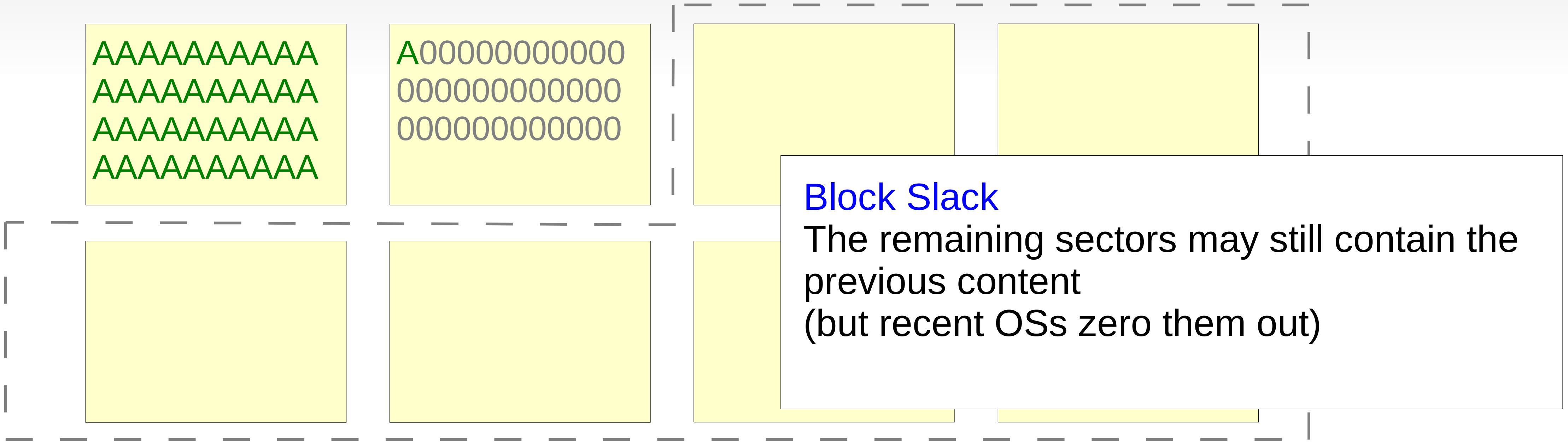
- Scenario:
  - The filesystem has a block size of 4 Kilobyte
  - The disk has a sector size of 512 byte
  - The user create a file containing 513 bytes (all “A” characters)



Sector Slack
Modern operating systems pad each sector with 0s before writing it to the disk (but DOS didnt)

Slack Space

- Scenario:
  - The filesystem has a block size of 4 Kilobyte
  - The disk has a sector size of 512 byte
  - The user create a file containing 513 bytes (all “A” characters)

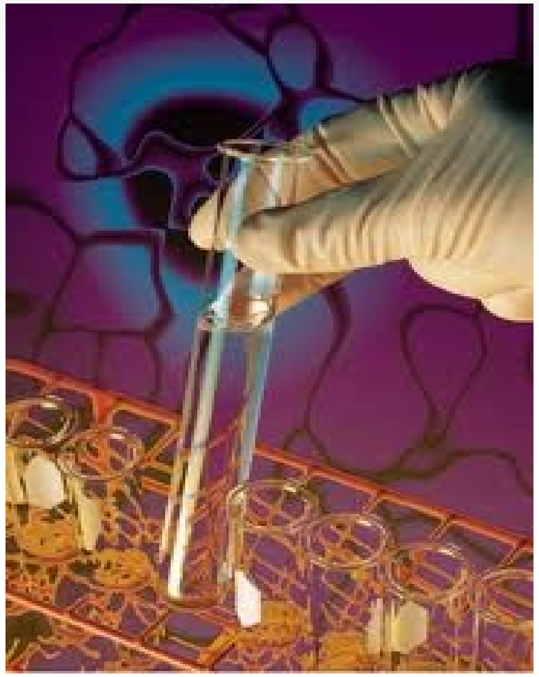


Block Slack
The remaining sectors may still contain the previous content
(but recent OSs zero them out)

*supported by the slacker tool (by Metasploit)

Summary

- ✔ Physical Disks
- ✔ Volumes and Partitions
- ✔ File systems
  - ✔ General concepts
  - ✔ EXT
  - ✔ Slack space
  - (Un)Deleting files in EXT2/3
  - Data erasure
  - NTFS
  - Encryption
- The Malware Corner



Deleted Files

- **DELETED** [Metadata → Data]
  The file has been unlinked and it is no longer presented in the directory listing.
  The content is still intact and the metadata are available and still point to the file

Deleted Files

- **DELETED** **[Metadata → Data]**
  The file has been unlinked and it is no longer presented in the directory listing.
  The content is still intact and the metadata are available and still point to the file

- **ORPHANED** **[Metadata , Data]**
  Both the content and the metadata are still available, but they are no longer linked together

Deleted Files

- **DELETED** [Metadata → Data]
  The file has been unlinked and it is no longer presented in the directory listing.
  The content is still intact and the metadata are available and still point to the file

- **ORPHANED** [Metadata , Data]
  Both the content and the metadata are still available, but they are no longer linked together

- **UNALLOCATED** [Data]
  The file metadata are no longer available, but the content has not yet been overwritten.
  Full recovery may be possible using a carving tool

Deleted Files

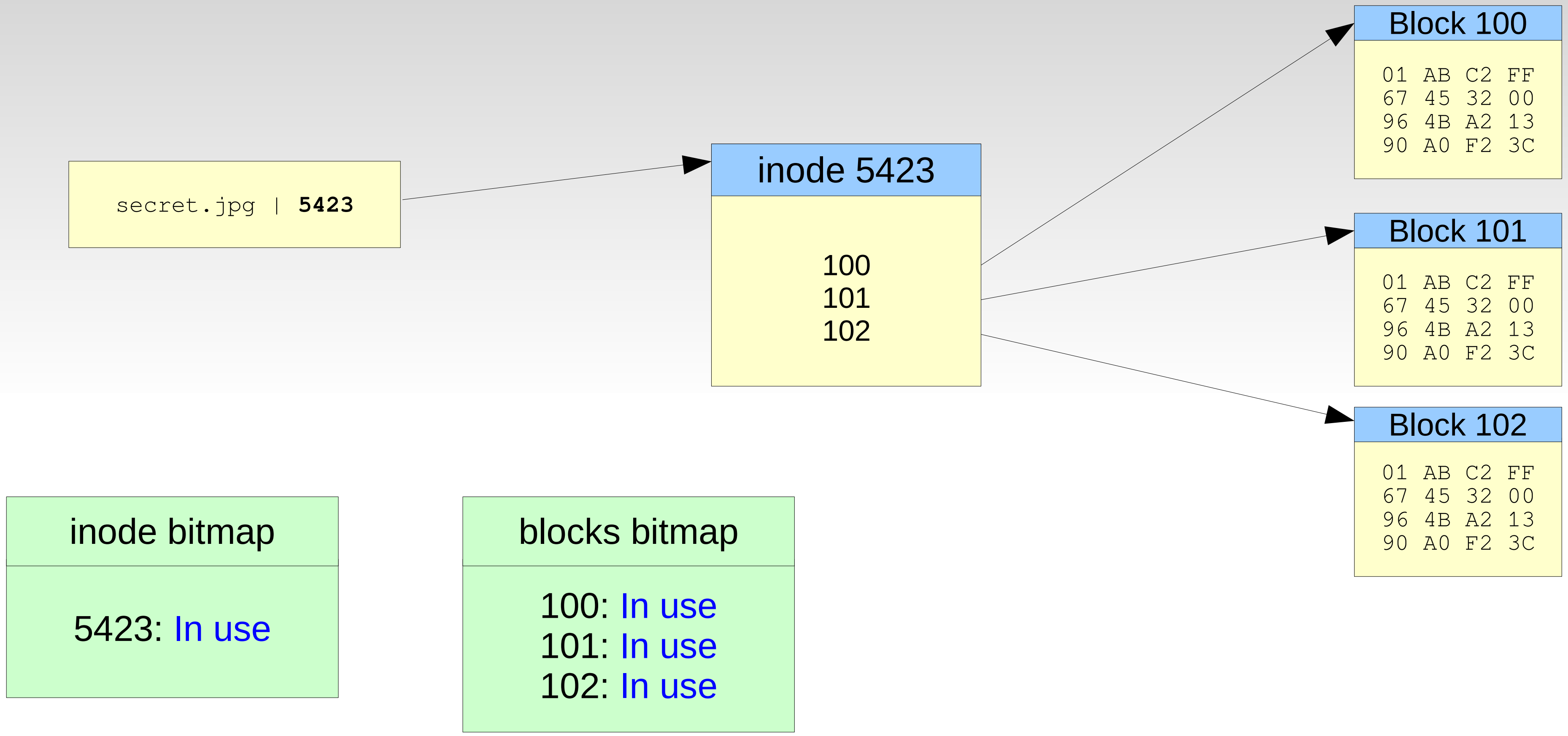
- **DELETED** **[Metadata → Data]**
  The file has been unlinked and it is no longer presented in the directory listing.
  The content is still intact and the metadata are available and still point to the file

- **ORPHANED** **[Metadata , Data]**
  Both the content and the metadata are still available, but they are no longer linked together

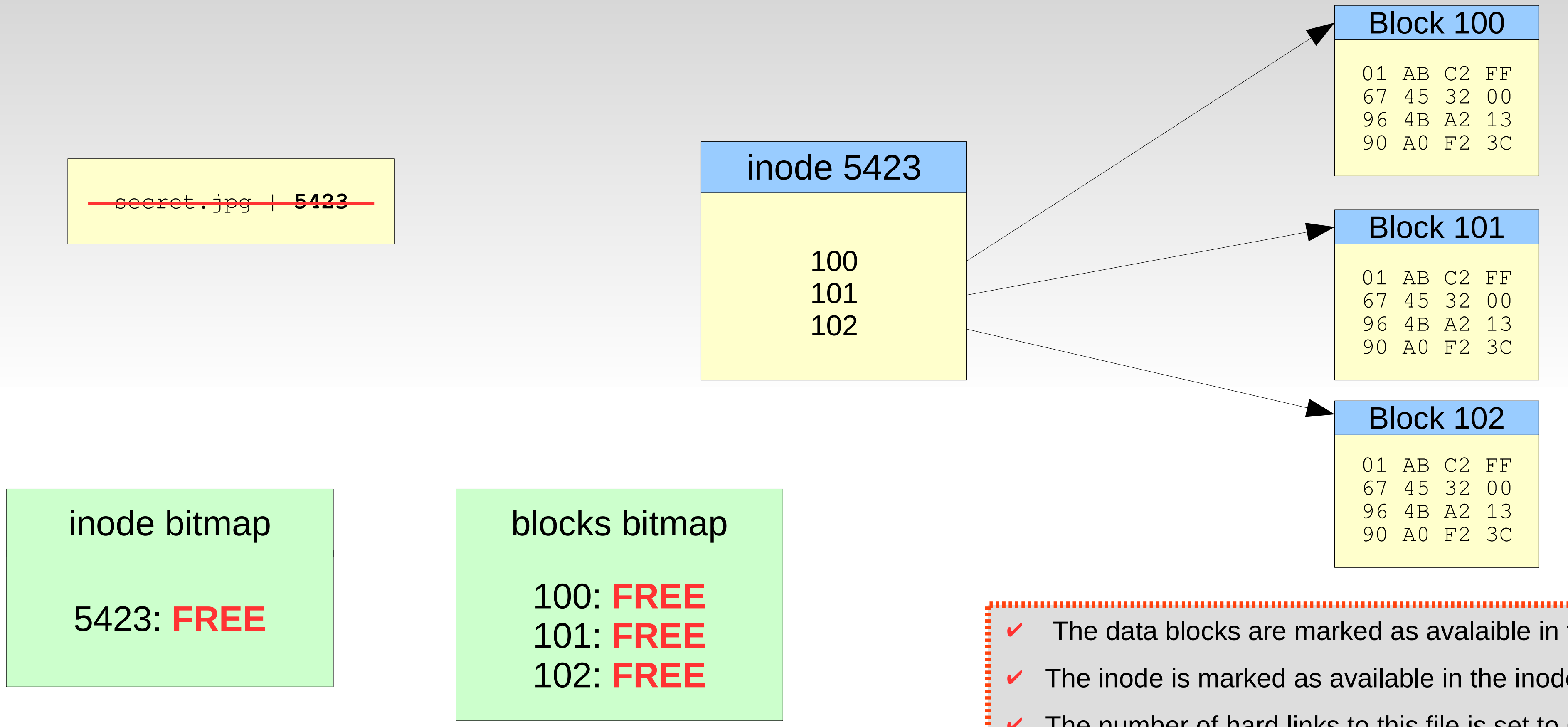
- **UNALLOCATED** **[Data]**
  The file metadata are no longer available, but the content has not yet been overwritten.
  Full recovery may be possible using a carving tool

- **OVERWRITTEN** **[ ]**
  The file content has been partially overwritten. Partial recovery may still be possible

Deleting Files on Ext2

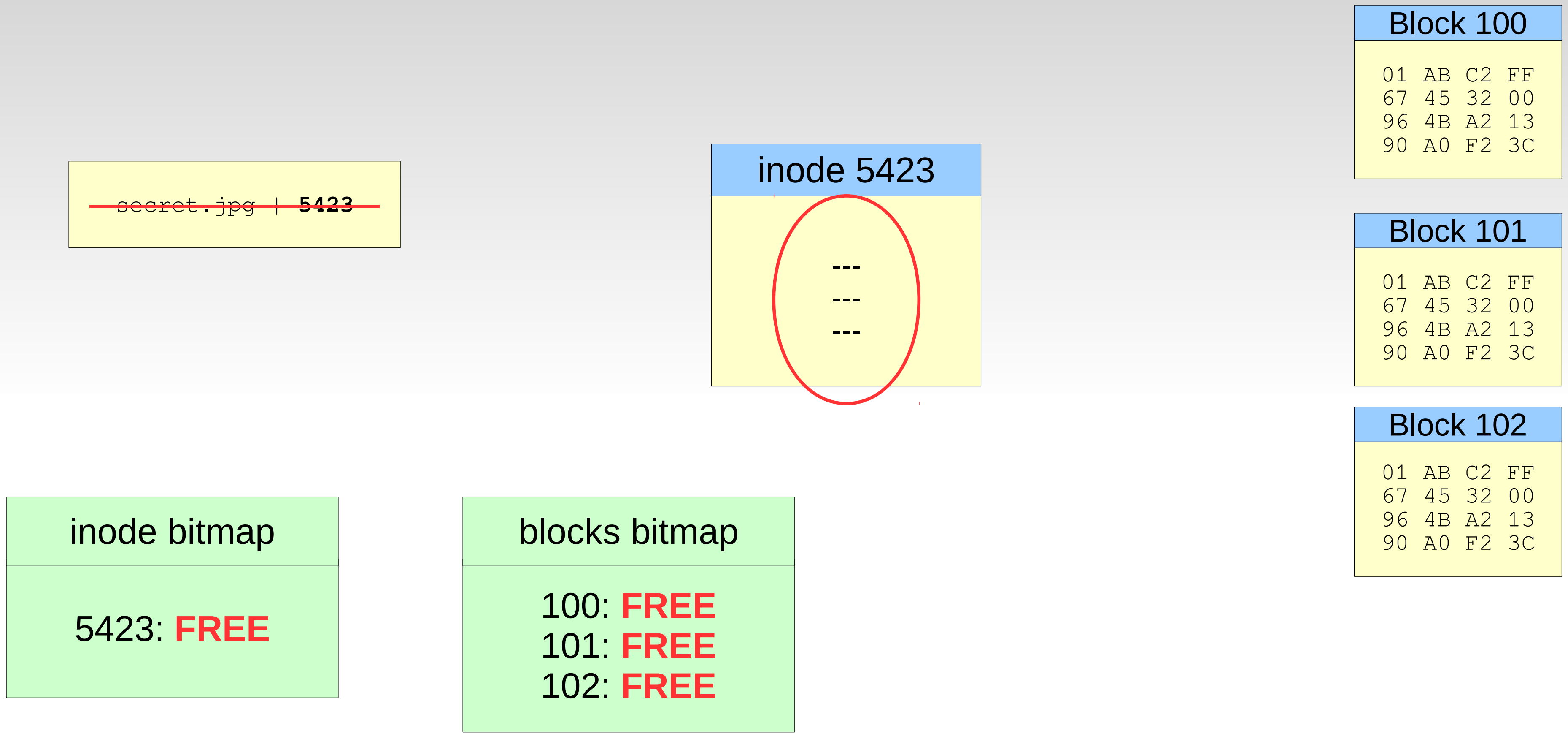


Deleting Files on Ext2



- ✔ The data blocks are marked as avalaible in the block bitmap
- ✔ The inode is marked as available in the inode bitmap
- ✔ The number of hard links to this file is set to 0 in the file's inode
- ✔ The deletion time is set in the file's inode
- ✔ The file's name in the directory entry is invalidated

Deleting Files on Ext3



Block 100
```
01 AB C2 FF
67 45 32 00
96 4B A2 13
90 A0 F2 3C
```

Block 101
```
01 AB C2 FF
67 45 32 00
96 4B A2 13
90 A0 F2 3C
```

Block 102
```
01 AB C2 FF
67 45 32 00
96 4B A2 13
90 A0 F2 3C
```

Undeleting Files on Ext3

- It is a hard and error-prone process
- Approach I: Carving
  - Remember that inodes and data blocks are allocated in the same block group (if possible)
  - Extract all the FREE blocks in the group of the inode of the deleted file
  - Carve on the extracted blocks

Undeleting Files on Ext3

- It is a hard and error-prone process
- Approach I: Carving
  - Remember that inodes and data blocks are allocated in the same block group (if possible)
  - Extract all the FREE blocks in the group of the inode of the deleted file
  - Carve on the extracted blocks
- Approach II: Carving + Locality + Indirect blocks (`frib` from FireEye)
  - Use carving like in (I) to locate the first block of a file. The first 12 are likely consecutive
  - The first block after the data one is very likely to contain the first indirect block
  - Extract the blocks pointer and then repeat (

Undeleting Files on Ext3

- It is a hard and error-prone process
- Approach I: Carving
  - Remember that inodes and data blocks are allocated in the same block group (if possible)
  - Extract all the FREE blocks in the group of the inode of the deleted file
  - Carve on the extracted blocks
- Approach II: Carving + Locality + Indirect blocks (`frib` from FireEye)
  - Use carving like in (I) to locate the first block of a file. The first 12 are likely consecutive
  - The first block after the data one is very likely to contain the first indirect block
  - Extract the blocks pointer and then repeat (
- Approach III: Journal-based recovery (`extundelete`)
  - The block in which an `inode` is located is recorded in the journal when an inode is updated
  - If you are lucky, you can find an old version (with the data pointers still intact) of the deleted file's `inode` in the journal because another file was updated before the deletion

Recycling vs Deleting vs Wiping

- Some systems move the file to a different place (e.g., the recycled bin) when the user press delete
  - The file content is NOT deleted from disk

Recycling vs Deleting vs Wiping

- Some systems move the file to a different place (e.g., the recycled bin) when the user press delete
    - The file content is NOT deleted from disk
- Traditional delete operations (`rm` in Linux or `shift-del` in Windows) remove the file entry from the FS directory and mark the space occupied by the file as available
    - The file content is NOT immediately deleted from disk

Recycling vs Deleting vs Wiping

- Some systems move the file to a different place (e.g., the recycled bin) when the user press delete
  - The file content is NOT deleted from disk
- Traditional delete operations (`rm` in Linux or `shift-del` in Windows) remove the file entry from the FS directory and mark the space occupied by the file as available
  - The file content is NOT immediately deleted from disk
- Wiping relies on a software or a hardware device to completely re-write every single bit of the destination
  - The file content is deleted from disk
  - Metadata and other copies of the content (e.g., autosave) may still exist
    Wiping a file is not the same as if the file never existed
  - E.g.,

```
$ dcfldd pattern=00 of=/dev/sd
$ shred /tmp/foo
```

Controversy on Data Erasure

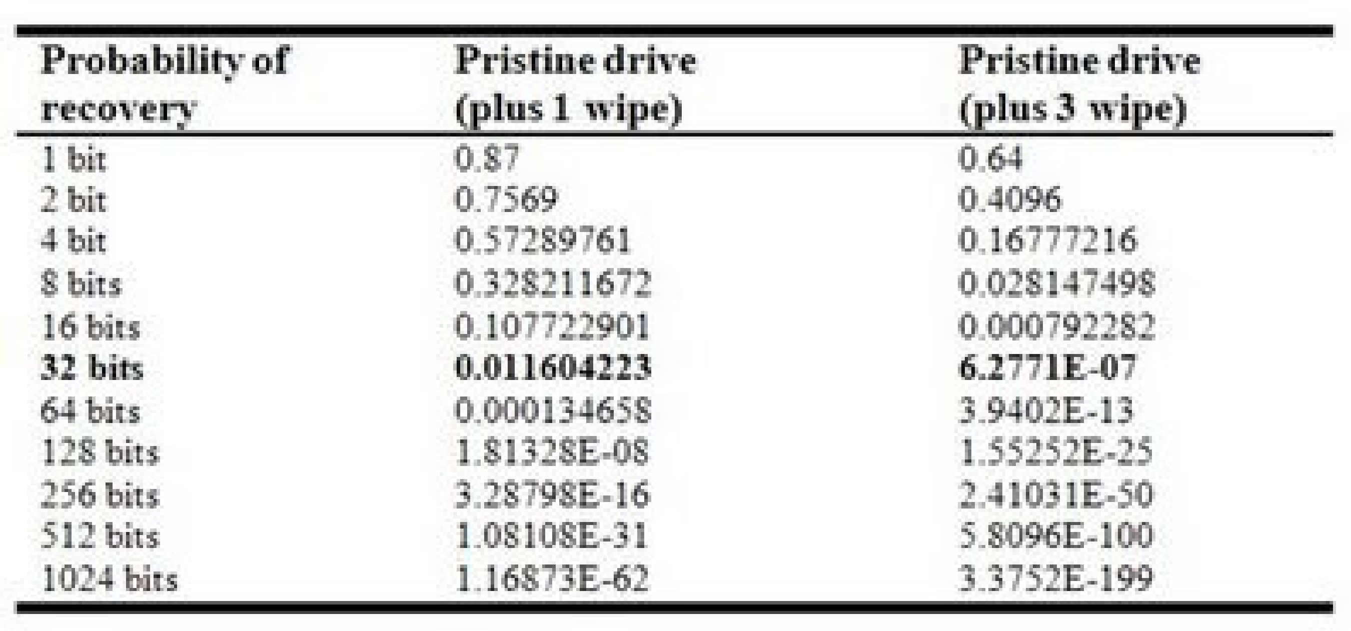
- In older hard drives when a 1 (bit) is written to the disk the resulting magnetic charge is closed to 0.95 (if the previous state was 0) or 1.05 (if the previous state was 1)
  - Using a Magnetic Force Microscope it is therefore possible to recover the previous state of the bit
- Therefore, it was often recommended in the past to overwrite each byte multiple times
  - **DoD**: overwrite with a character, its complement, and finally a random value
  - **Canadian Police Standard**: overwrite with 1, then 0, and repeat a minimum of three times
  - **Guttman method**: overwrite 35 times with random patterns !!!

Controversy on Data Erasure

"In many instances, using a magnetic force microscope to determine the prior value written to the hard drive was less successful than a simple coin toss."

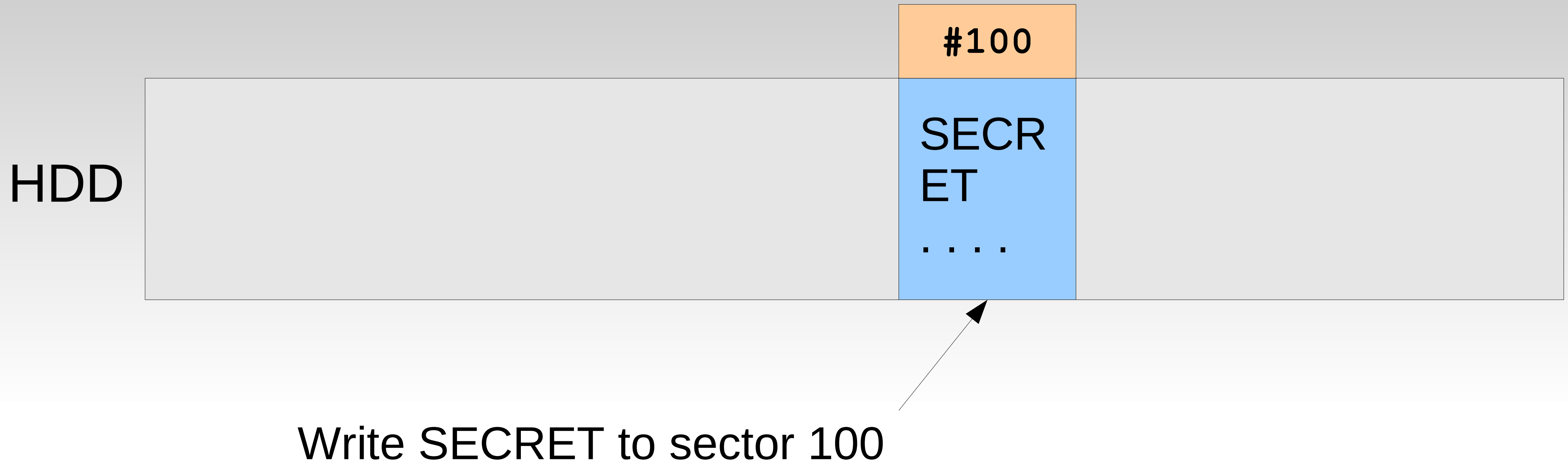
– SANS Computer Forensic

In reality, there is good chance of recovery any individual bit from a drive, but the chances of recovery any reasonable amount of data is negligible

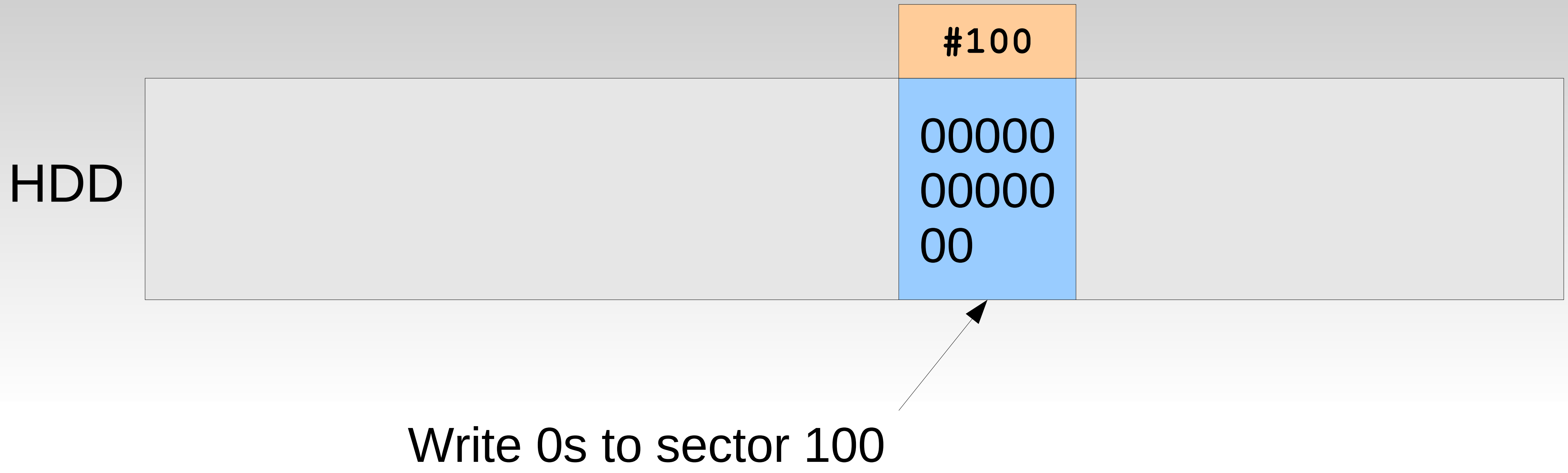


Probability of recovery	Pristine drive (plus 1 wipe)	Pristine drive (plus 3 wipe)
1 bit	0.87	0.64
2 bit	0.7569	0.4096
4 bit	0.57289761	0.16777216
8 bits	0.328211672	0.028147498
16 bits	0.107722901	0.000792282
32 bits	**0.011604223**	**6.2771E-07**
64 bits	0.000134658	3.9402E-13
128 bits	1.81328E-08	1.55252E-25
256 bits	3.28798E-16	2.41031E-50
512 bits	1.08108E-31	5.8096E-100
1024 bits	1.16873E-62	3.3752E-199

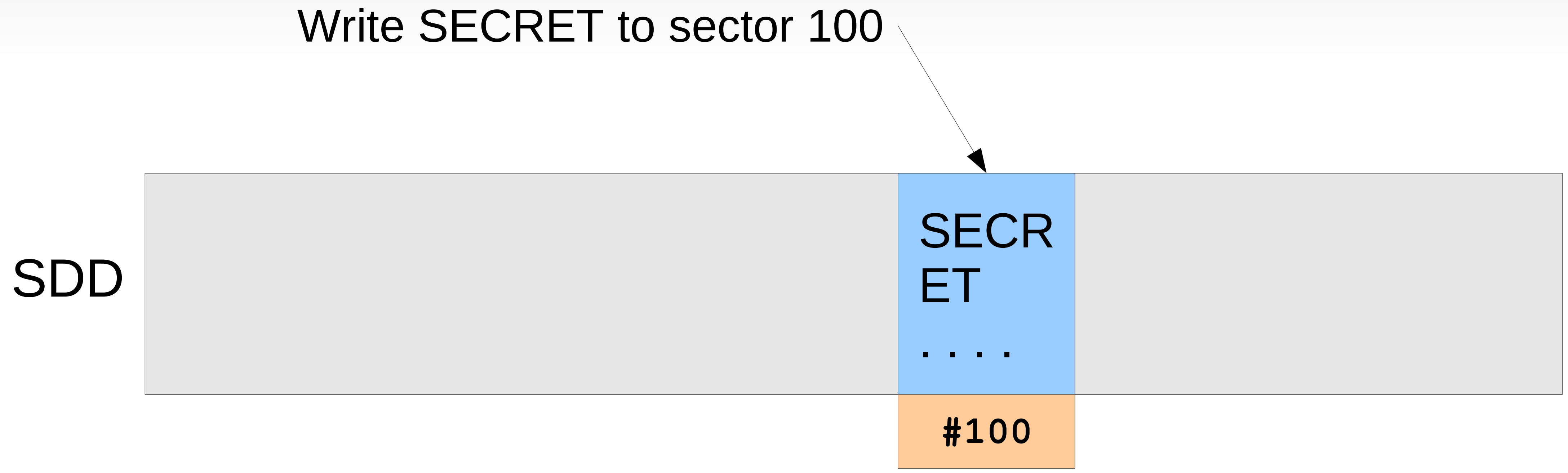
HDD vs SDD



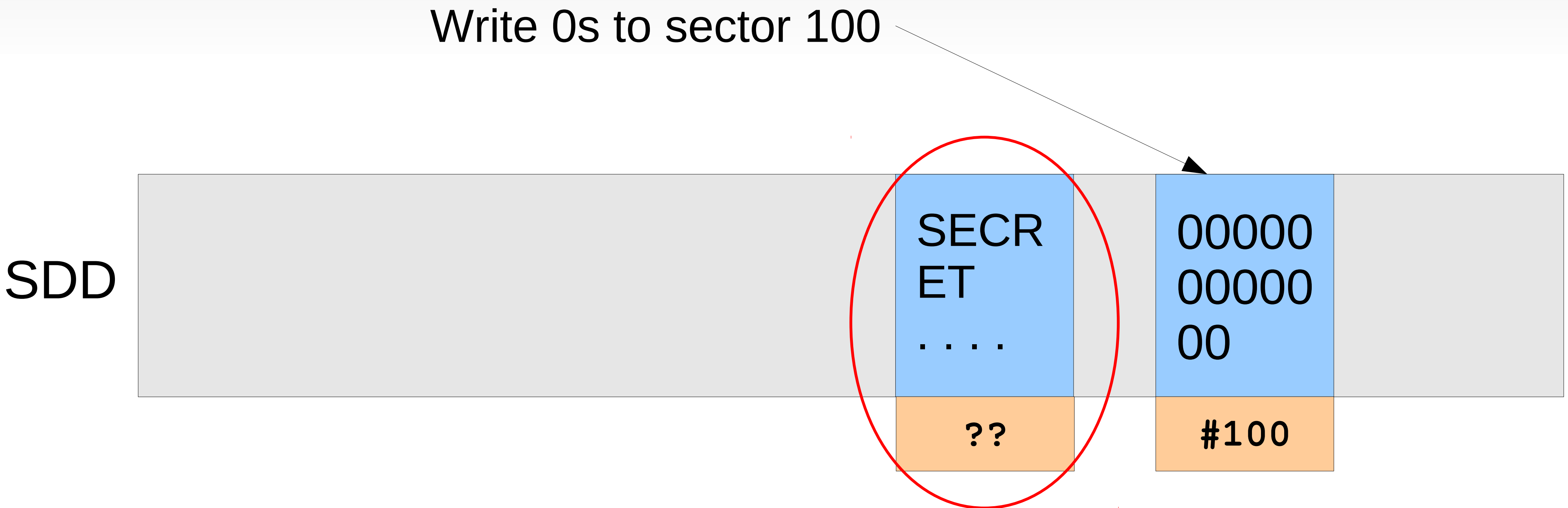
HDD vs SDD



HDD vs SDD



HDD vs SDD



Erasing Data from SDD

- For more info on how effective data overwriting is to sanitize SSD drive refer to:

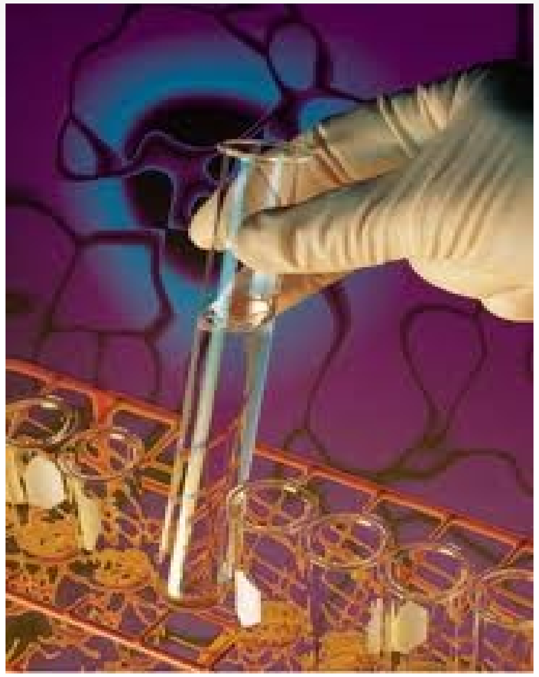
  “*Reliably Erasing Data From Flash-Based Solid State Drives*”

  - Usenix Conference on File and Storage Technologies - 2011

- Most drives support optional ATA commands to erase the entire disk content.
  - A recent study reported that 3 out of 7 disk tested failed to properly complete the operation

Summary

- ✓ Physical Disks
- ✓ Volumes and Partitions
- ✓ File systems
  - ✓ General concepts
  - ✓ EXT
  - ✓ Slack space
  - ✓ (Un)Deleting files in EXT2/3
  - ✓ Data erasure
  - NTFS
  - Encryption
- The Malware Corner



New Technology File System

- NTFS is a journaling file system first introduced in 1993 to replace FAT
  - Standard Windows filesystem since Windows NT/2000
  - Still in use today with Windows 10
  - Proprietary FS: no official specification was released by MS
  - Support for transparent compression or encryption

- NTFS partitions start with a NTFS Boot Sector which includes general info
  (bytes per sector, sectors per cluster, ...)
- Every piece of information is stored in a file
- Every file has a corresponding entry in the *Master File Table* (MFT)

NTFS

- The MFT is a dynamically-growing table made of records
  - Each record is 1024 bytes (by default)
  - There is one record for each file + records for special Metafiles
  - Each record can contain several Attributes
    (e.g., data, file name, standard info, object id, security descriptor, index root, …)
  - Attributes can be
    - Resident → they contain their data
    - Non resident → point to a cluster run (sequence of consecutive clusters identified by the first + the number of <u>clusters</u>) that contains the actual data

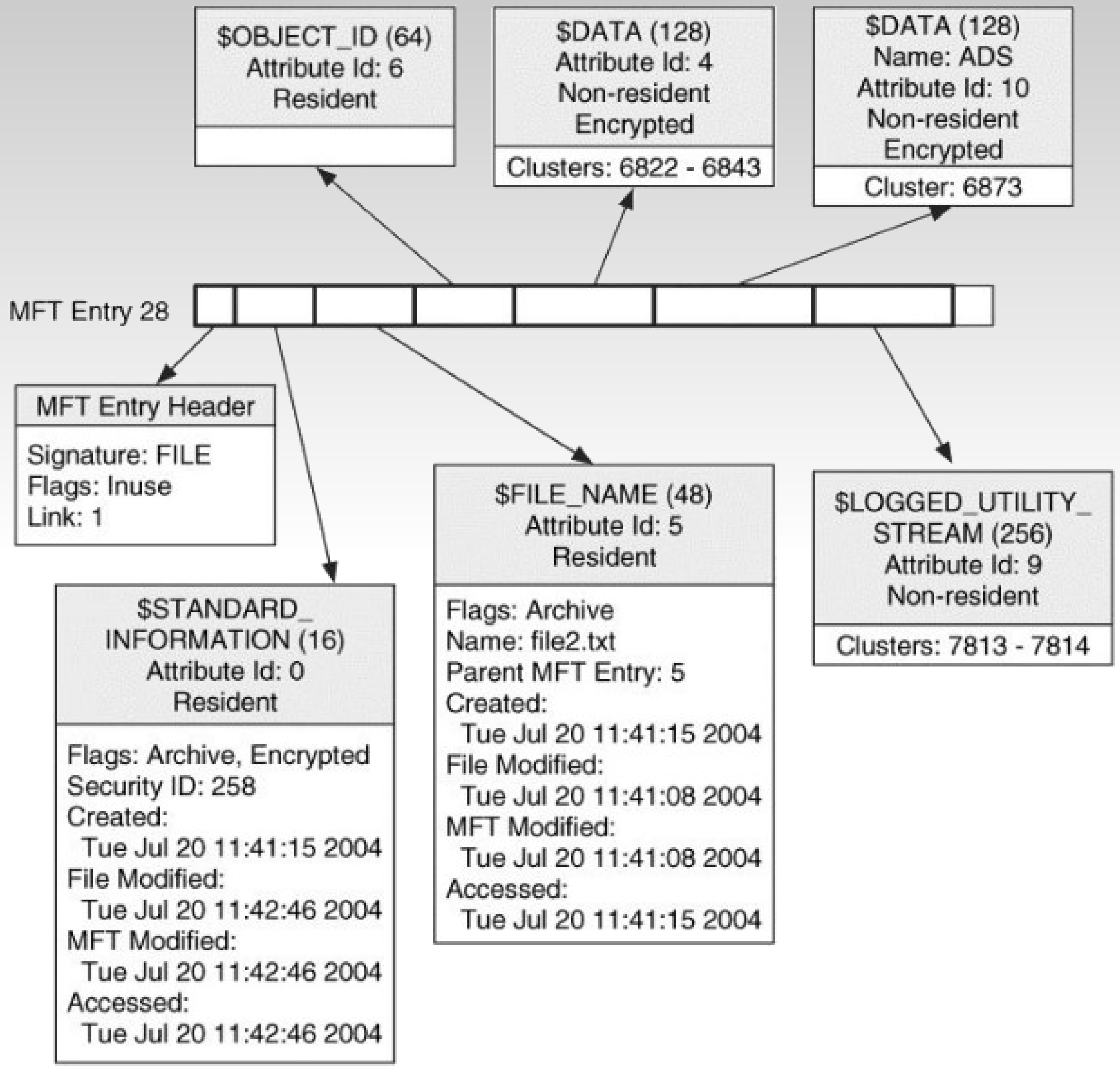
MS way to call blocks

Metafiles

`$MFT`	→ the MFT itself has an entry in the MFT
`$MFTMirr`	→ backup copy of the MFT
`$LogFile`	→ journaling information
`$Root`	→ points to the system root directory
`$Bitmap`	→ allocation status of each cluster
`$BadClus`	→ malfunctioning cluster
`$Secure`	→ security descriptors of the filesystem
`$Extend\$UsnJrnl`	→ extended journal (from Vista) which stores info on every action that happens to a file: rename, move, create, delete

`...`

Attributes



Source: Brian Carrier

Attributes

- Standard Information Attribute (SIA) – `resident`
  - File owner, flags (hidden, system, …)
  - MACE timestamps: Modify, Access, Creation, and last MFT Entry modification
- Filename Attribute (FNA) – `resident`
  - Unicode file name, parent directory, and MACE timestamps (may be different)
- Data Attribute
  - Small files (~700bytes) are stored in the attribute
  - One (unnamed) main data attribute for each file
  - Optional (named) data attributes used to specify alternate streams

Alternate Data Stream (ADS)

- More than one data stream can be associated with the same filename
  - Only the main stream is used to compute the file size
  - Alternative streams are not visible in Explorer (from Windows 7: `dir /r`)
  - Sometimes (but rarely) used by normal software:
    - `Zone.Identifier` are ADS added by Internet Explorer to mark files downloaded from external sites as possibly unsafe to run

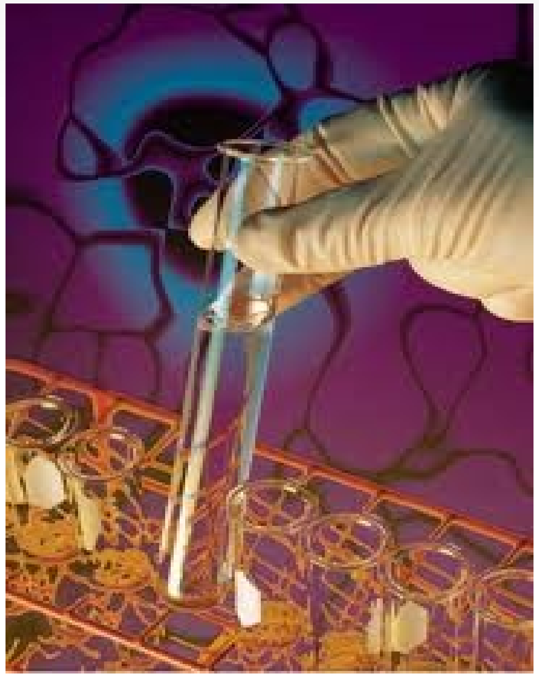
```
C:\>type secretpicture.jpg > innocuous.txt:secret.jpg

C:\>mspaint innocuous.txt:secret.jpg
```

(Un)Deleting Files on NTFS

- When a file is deleted on a NTFS filesystem...
  - ..its entry in the MFT is marked accordingly (by changing one byte)
  - ..all clusters are marked as free in the `$BitMap`
  - The metadata and cluster pointers are not overwritten

Summary

- ✔ Physical Disks
- ✔ Volumes and Partitions
- ✔ File systems
  - ✔ General concepts
  - ✔ EXT
  - ✔ Slack space
  - ✔ (Un)Deleting files in EXT2/3
  - ✔ Data erasure
  - ✔ NTFS
  - Encryption
- The Malware Corner

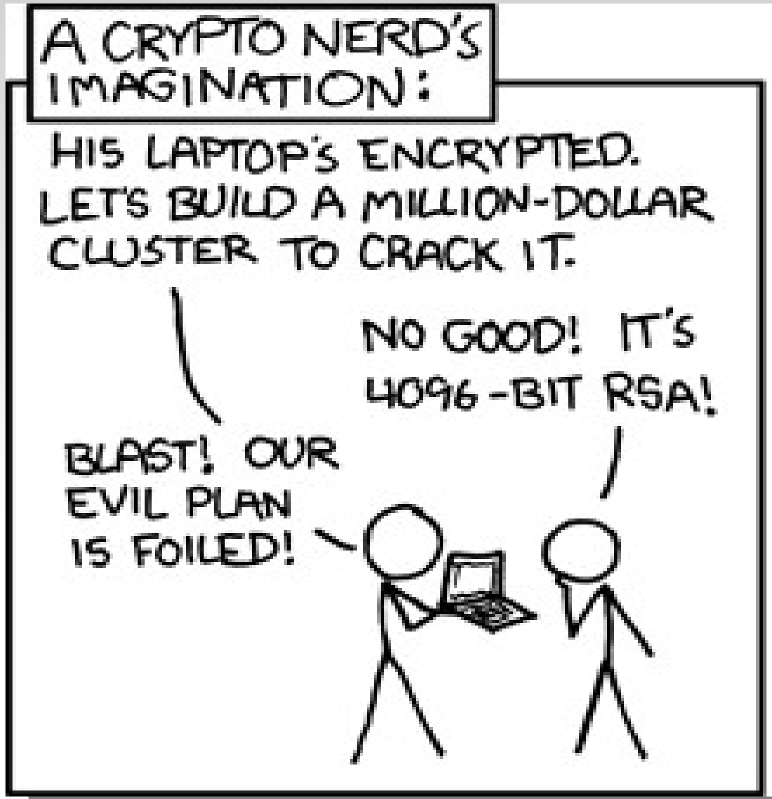


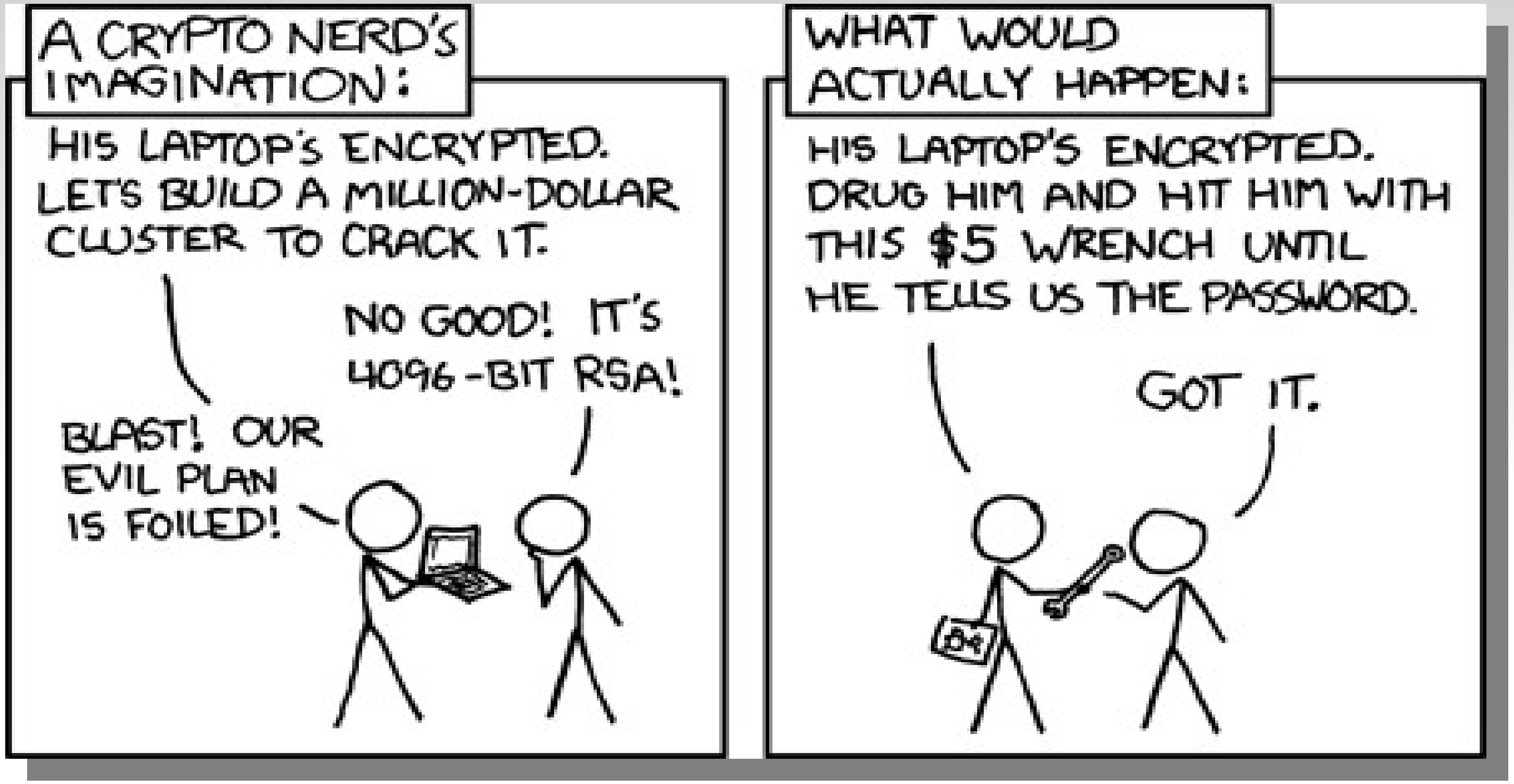
Disk Encryption

- Increasingly common “*data at rest*” technique to protect information
  - Fulldisk Encryption
    - Can be implemented in hardware (by the disk itself) or in software
    - All data is <u>decrypted</u> when the system runs
  - Volume Encryption
    - Only a certain volume is encrypted
    - All data is decrypted when the volume is in use
  - File & Filesystem Encryption
    - Usually protect the file content (not its name, timestamps,...)
    - Sometimes can be applied to entire directories
    - Particular information can be decrypted only when needed

Encrypted data is only as secure as the decryption key

- Memory analysis
  - Volume and FS-level encryption often require the OS to have access to the decryption key
  - "*Lest We Remember: Cold Boot Attacks on Encryption Keys*" USENIX Security 2008
- Bruteforce
- Intimidation (rubber-hose cryptanalysis)
  - Often used at border controls and in federal investigation
  - Still unclear if the court can force a user to decrypt his data
- Keyloggers (black bag operations)
  - Require a proper search warrant
  - Used by law enforcement agencies


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

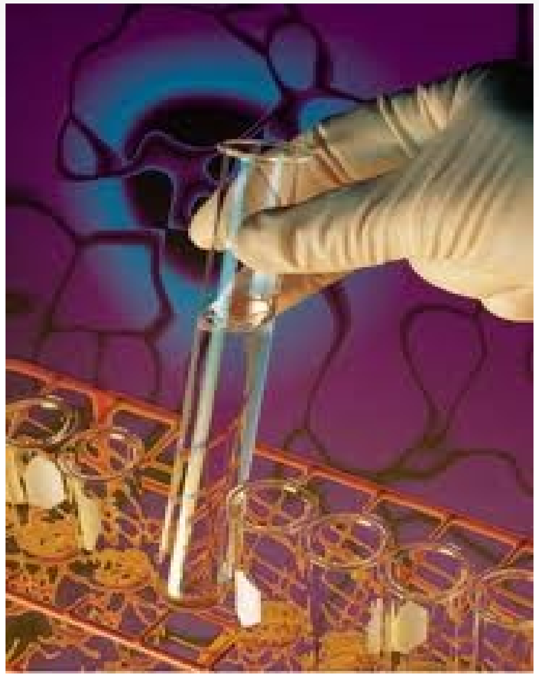
Deniable Encryption

- Goal: make impossible to prove even that there is some encrypted data without the user cooperation
- Hidden volumes
  - An encrypted container volume is initialized with random encrypted data
  - A filesystem is created in the container volume and populated with files
  - A second encrypted volume is created in the unused space of the container volume
- Possession of deniable encryption tools may be considered suspicious during an investigation

The Malware Corner

Summary

- ✔ Physical Disks
- ✔ Volumes and Partitions
- ✔ File systems
- ✔ The Malware Corner
  - Physical Hard Disk infection
  - Bootkits
  - Hiding data outside the file system
  - Hiding files in the filesystem
  - Ramsonware



Malware & Disks

- Almost every piece of malware is stored *somewhere* on the disk of the infected machine

  (except for *disk-less malware* → see memory forensics)

- This is why traditionally AntiVirus software were designed to routinely scan the filesystem

Physical Hard Disk Infection

- Idea: infect the firmware of the disk
- First example by *Zaddach et al.[1]* in 2013.
  Few months later we discovered the NSA had been using "disk implants" for several years. Also used by the `EquationDrug` and `Grayfish` APTs
- Result:
  - The malicious code is stored in the *Reserved Service Area*, where only the firmware itself (which is infected) can reach
  - The malware can survive any attempts to remove it or to re-install the system
  - The malware can easily use the disk to infect the OS
- A nightmare from a forensic point of view

J. Zaddach, A. Kurmus, D. Balzarotti, E. O. Blass, A. Francillon, T. Goodspeed, M. Gupta, I. Koltsidas.
"Implementation and implications of a stealth hard-drive backdoor" ACSAC 2013

Boot-kits

- Malware designed to interfere with the boot process of a system and execute code *before* the OS kernel takes control
  - Bootkits typically infect the MBR or the VBR
  - A copy of the original boot code is stored with the bootkit configuration in some hidden sectors – usually at the end of the disk (80%) or between partitions (20%)
  - At boot, they modify the OS kernel (e.g., by loading unsigned modules) or they pass parameters to it to disable security features
- Very popular in the early days, but rapidly reappearing in the last decade

“Nice Boots!-A Large-Scale Analysis of Bootkits and New Ways to Stop Them” DIMVA 2015

Hiding Data outside the Filesystem

- Outside the partitioned space:
  - Sectors that do not belong to a partition often add up to several megabytes
  - Used by bootkits to store their code, but also by some malware to keep their data hidden outside the file system
- Inside hidden storage:
  - Volume created in a hidden file
  - Typically encrypted

Hiding Files inside the Filesystem

- Malicious code loaded in the operating system (Rootkits) is often used to hide files and directory in the file system
  - Remember the SONY Rootkit?
    It was designed to hides every file, process, or registry key beginning with `$sys$`

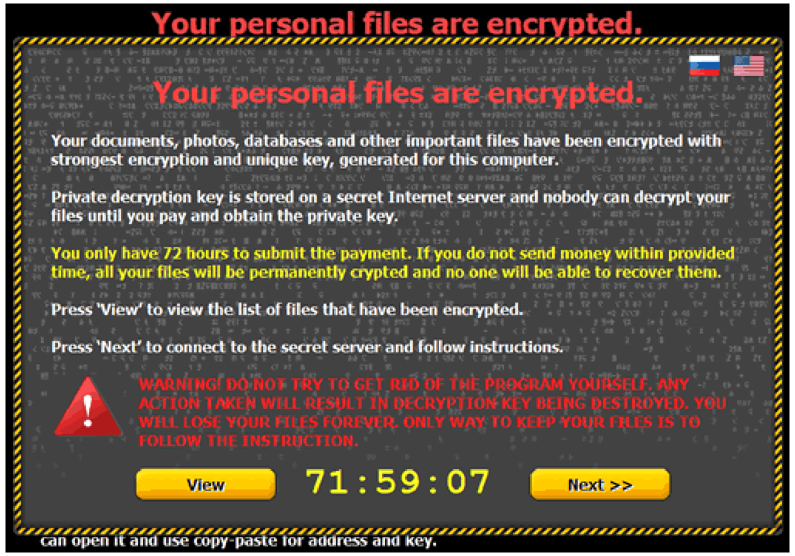
- Rootkits typically intercept requests to list files in a directory or read the content of a file and return fake information to hide components from view
  - One more reason why a physical acquisition is better than a logical one

Examples

- ZeroAccess
  - It replaces an existing driver with the malicious one
  - A rootkit intercepts read or write operations to the infected driver and return the content of the original clean copy of the file
  - Sets up a RC4-encrypted hidden volume in the computer's file system
- TDL4
  - MBR Rootkit
  - Encrypted partition (using its own rudimentary FS) at the end of the disk
- Gapz
  - Recent VBR bootkit
  - Uses a hidden storage (FAT32 volume encrypted with AES)
- Rovnix
  - First bootkit to target the VBR
  - Uses a hidden encrypted partition placed on unpartitioned disk space

Ransomware

- Encrypt files on disk not for protection, but to extort money
- We conducted an in-depth study of ~1400 ransomware files collected from 2006 and 2014
  - Early versions were very simple (e.g., just lock the screen) or contained naive crypto mistakes (e.g., used a hardcoded key)
  - Recent version are more sophisticated and generate the encryption key remotely



Your personal files are encrypted.

Your personal files are encrypted.

Your documents, photos, databases and other important files have been encrypted with strongest encryption and unique key, generated for this computer.

Private decryption key is stored on a secret Internet server and nobody can decrypt your files until you pay and obtain the private key.

You only have 72 hours to submit the payment. If you do not send money within provided time, all your files will be permanently crypted and no one will be able to recover them.

Press 'View' to view the list of files that have been encrypted.

Press 'Next' to connect to the secret server and follow instructions.

WARNING! DO NOT TRY TO GET RID OF THE PROGRAM YOURSELF. ANY ACTION TAKEN WILL RESULT IN DECRYPTION KEY BEING DESTROYED. YOU WILL LOSE YOUR FILES FOREVER. ONLY WAY TO KEEP YOUR FILES IS TO FOLLOW THE INSTRUCTION.

71:59:07

can open it and use copy-paste for address and key.

A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, E. Kirda
"*Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks*"

Ready to Play?

Demo

- In which we recover hidden and deleted data from a EXT3 linux filesystem using
  - fls
  - fstat
  - ffind
  - ils -e
  - icat
  - istat
  - ifind
  - blkcat
  - blkls
  - dumpe2fs -b