Digital Forensics

Federico Conti

2024/25

Contents

Forensic Acquisition

Acquisition is the process of cloning or copying digital data evidence.

- forensically sound (integrity and non-repudiation)
  - the copies must be identical to the original
  - the procedures must be documented and implemented using known methods and technologies, so that they can be verified by the opposite party
- a critical step
  - proper handling of data ensures that all actions taken on it can be checked, repeated, and verified at any time
  - incomplete or incorrect handling of data has the potential to compromise the entire investigation

It is generally recommended to avoid conducting analysis on the original device (namely, best evidence).

Creating a forensic image is typically considered the most effective method for preserving digital evidence (One or more, usually two).

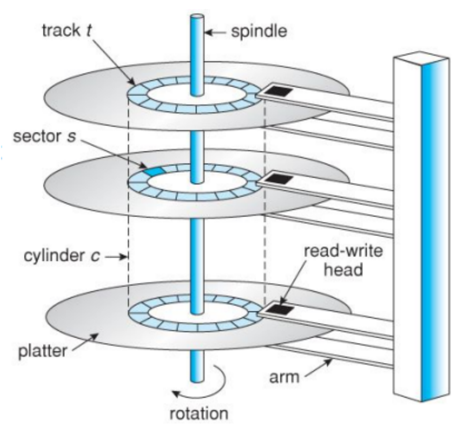
Accessing the original media only once during the acquisition phase can help minimize the risk of altering or damaging the evidence

HDD and SSD technologies

HDD

A **hard disk** is a sealed unit containing a number of platters in a stack:

- each platter has two wothey moverking surfaces
- each working surface is divided into a number of concentric rings called tracks
  - the collection of all tracks that are the same distance from the edge of the platter is called a cylinder.
- each track is further divided into sectors
  - a sector is the smallest unit that can be accessed on a storage device
  - traditionally containing 512 bytes of data each, but recent hard drives have switched to 4KB sectors (Advanced Format)
  - a cluster is a group of sectors (from 1 to 64 sectors) that make up the smallest unit of disk allocation for a file within a file system



The data on a hard drive is read by read-write hands. The standard configuration uses one head per surface and they moves simultaneously from one cylinder to another.

A **low level format** is performed on the blank platters to create data structures for tracks and sectors

- creates all of the headers and trailers marking the beginning and ends of each sector
- header and trailer also keep the linear sector numbers (cf. LBA later), and error-correcting codes (ECC)

All disks are shipped with a few bad sectors (additional ones can be expected to go bad slowly over time).

- disks keep spare sectors to replace bad ones
- ECC calculation is performed with every disk read or write: if an error is detected but the data is recoverable, then a *soft error* has occurred if the data on a bad sector cannot be recovered, then a *hard error* has occurred. A bad sector can be replace with a spare one (but any information written is usually lost)

Older hard drives used a system called **CHS (Cylinder-Head-Sector)** addressing to locate data on the disk. This method relied on:

- Cylinders (C) → The track number (a ring of data on a disk platter).
- Heads (H) → The read/write head that accesses a platter's surface.
- Sectors (S) → The smallest unit of storage on a track.

Eample: CHS (100, 2, 30) would mean: Cylinder 100; Head 2 (indicating the second platter side); Sector 30 (the exact location on that track).

CHS had physical limitations → It could only handle disks up to 504 MB (later ECHS extended this to 8 GB).

Instead of using three values (CHS), **Logical Block Addressing(LBA)** assigns a single number to each sector. LBA starts at 0 and counts up sequentially. The operating system and file system treat the disk as a continuous array of sectors, making it easier to manage.

Example: LBA 123456: The 123,456th sector on the disk.

SSD

A **Solid-State Drive (SSD)** is a non-volatile storage device, meaning it retains data even when the power is off. Unlike Hard Disk Drives (HDDs), which use spinning magnetic platters, SSDs rely on flash memory chips to store data.

- The smallest unit of an SSD is a page, which is composed of several memory cells: the page sizes are 2KB, 4KB, 8KB, 16KB or larger (usually they are 4 KB in size).
- Several pages on the SSD are summarized to a block: the block size typically varies between 256KB (128 pages * 2KB per page) and 4MB (256 pages * 16KB per page)

Operation	**Description**
Read and Write	An SSD can read and write data at the page level.
Write	Writing is only possible if other pages in the block are empty. Otherwise, it leads to write amplification.
Erase Data	An SSD can only erase an entire block at once due to the physical and electrical characteristics of the memory cells.

Modifying data requires a **Program/Erase (P/E) cycle**.

- During a P/E cycle, an entire block containing the targeted pages is written to memory
- The block is then marked for deletion, and the updated data is rewritten to another block

The erase operation does not happen immediately after data is marked for deletion. Instead, the SSD performs it asynchronously when necessary to optimize performance.

Garbage Collection helps free up space efficiently while minimizing interruptions to read/write operations.

Every flash memory cell has a limited number of P/E cycles before it wears out Wear leveling is a technique used by SSDs to distribute writes evenly across all memory blocks.

Unlike HDDs, SSDs cannot simply overwrite deleted files. If deleted data is not managed properly, unnecessary data copies can cause write amplification, increasing wear on the SSD. TRIM command allows the OS to inform the SSD which pages are no longer needed, enabling it to manage space more efficiently.

Forensic Investigators Face a Big Problem with SSDs|

TRIM permanently deletes data by triggering the garbage collector, making data recovery impossible. The garbage collector operates independently within the SSD controller, meaning:

- Even a hardware write blocker (a forensic tool to prevent data changes) cannot stop it.
- Data can change midway or between acquisitions, complicating forensic investig

Interface standards and protocols

Disks are accessed using standard interfaces that define how they physically and logically connect to a computer system. Each standard improves data transfer speeds and fixes limitations from older technologies. A disk interface consists of two key components:

- Physical Connection → Defines the type of cable and connector used to attach the disk to the system.
- Logical Connection → Defines the protocol that controls how data is transferred between the disk and the computer.

ATA (Advanced Technology Attachment):

a widely used interface standard with multiple versions:

- ATA-1, ATA-2, ..., ATA-8 → Each new version improves speed and capacity.
- ATAPI (ATA Packet Interface) → Allows removable media (CD/DVD drives) to be connected using ATA but still uses SCSI commands for data transfer.

Type	**Description**
PATA (Parallel ATA)	Older, also known as IDE (Integrated Drive Electronics). Uses ribbon cables with multiple pins.
SATA (Serial ATA)	Modern standard introduced in ATA/ATAPI-7. Uses thin serial cables, improving speed and efficiency.
SATA 3.0	Supports speeds up to 6 Gbit/s (SATA-600). Common in modern HDDs and SSDs.
ATA-8	Introduced optimizations for SSDs, including TRIM support.

SCSI (Small Computer System Interface):

now replaced by SAS (Serial Attached SCSI) that is based on the SCSI standard, but uses a serial interface to connect storage. Better scalable and faster (supports data transfer rates of up to 24 Gbit/s).

NVMe (Non-Volatile Memory Express):

Uses a PCIe interface to connect storage devices to a computer. Commonly used for high-performance solid-state drives since it supports data transfer rates of up to 32 GB/s.

USB (Universal Serial Bus):

- uses a serial interface to connect storage devices to a computer
- mass storage is the standard protocol used for storage devices
- commonly used for external hard drives, flash drives, and other portable storage devices
- USB 3.1 Gen 1 standard supports speeds up to 5 Gbit/s, while the USB 3.1 Gen 2 standard, supports speeds up to 20 Gbit/s.

ATA specification

Introduced in ATA-3, **hard disk passwords** are an optional security feature designed to restrict unauthorized access to a hard drive. However, it is a lock mechanism, not encryption, meaning data remains unencrypted and could be accessed by other means.

There are two passwords:

1. User Password → Set by the owner.
2. Master Password → was designed so an administrator can gain access in case the user password was lost (every hard disk is initially supplied with an undocumented master password).

If passwords are being used, there are two modes that the disk can operate:

1. high security mode: the user and master password can unlock the disk
2. maximum-security mode: the user password can unlock the disk but the master password can unlock the disk after the disk content have been wiped

A protected HD will require the SECURITY_UNLOCK command to be executed with the correct password before any other ATA command. After the password has been entered, the disk works normally until the disk is powered on

Some ATA commands are still enabled on the HD when it is locked (so it may show as a valid disk when it is connected to a computer); however, trying to read data from a locked disk will produce an error

Setting the Password:

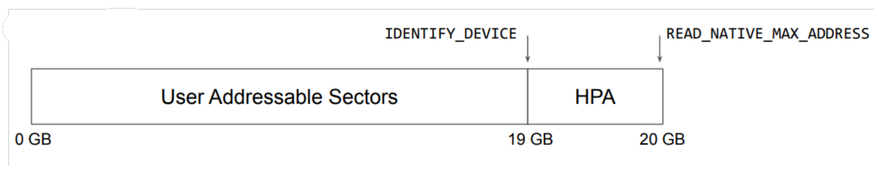
- Can be configured via BIOS settings.
- Linux users can manage HDD passwords using tools like hdparm.

Introduced in ATA-4, the **Host Protected Area (HPA)** is a hidden storage section on a hard disk that is not accessible to the operating system. It is used mainly by hardware vendors for system recovery files, diagnostics, or factory reset tools. A HPA is at the end of the disk and when used, it can be accessed by reconfiguring the hard disk.

Two ATA commands that return maximum physical addressable sectors

- READ_NATIVE_MAX_ADDRESS: return the maximum physical address
- IDENTIFY_DEVICE: return only the number of sectors that a user can access

To create an HPA, the SET_MAX_ADRESS command is used to set the maximum address to which the user should have access (to remove it, use SET_MAX_ADDRESS = READ_NATIVE_MAX_ADDRESS).



the SET_MAX_ADRESS command support different settings, e.g.,

- volatility bit: the HPA exist after the hard disk is reset or power cycled (otherwise the effect is permanent)
- locking command: prevents modification to the maximum address until next reset

when the BIOS requires to read/write some data in the HPA it uses SET_MAX_ADRESS with volatility bit and locking.

It is possible to protect settings with a password (different from HD passwords).

The **Device Configuration Overlay (DCO)** was introduced in ATA-6 and allows manufacturers to limit the apparent capabilities of a hard disk. This feature enables backward compatibility with older systems but can also be exploited to hide data.

A computer uses the IDENTIFY_DEVICE command to check an HDD's specifications (size, features, supported commands). If a DCO is applied, the IDENTIFY_DEVICE command will not show the actual full disk size or features.

This is achieved using two special ATA commands:

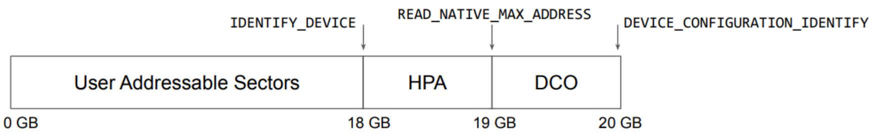
- DEVICE_CONFIGURATION_SET → Modifies the DCO settings to restrict visible disk space or disable features.
- DEVICE_CONFIGURATION_RESET → Restores the original settings, removing the DCO restrictions.

Example: A 2 TB hard drive can be made to appear as a 500 GB drive, hiding the remaining 1.5 TB from the operating system.

The DCO and HPA can co-exist on the same HDD (but DCO must be set first).

The DEVICE_CONFIGURATION_IDENTIFY command return the actual features and size of a disk:

- we can detect DCO if `DEVICE_CONFIGURATION_IDENTIFY` $\neq$ `IDENTIFY_DEVICE`



At least three different methods for detecting HPA on Linux: `dmesg`, `hdparm`, and `disk_stat` (https://wiki.sleuthkit.org/index.php?title=Disk_stat)

Disk encryption: Bitlocker

BitLocker is a full-disk encryption feature in Windows that protects data using a multi-layered encryption system.

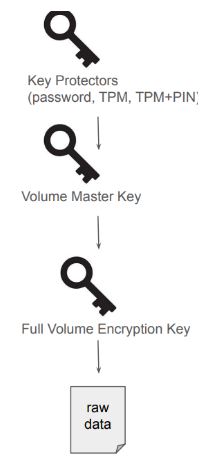
It makes use of symmetric encryption (by default, AES-128).

On modern systems, it is coupled with a Trusted Platform Module (TPM):

- the main functions of TPM are the generation, storage and secure management of cryptographic keys
- on a computer without TPM a password can be used (then BitLocker encryption will be just as secure as the password you set)

Bitlocker uses different symmetric key:

1. row data is encrypted with the Full Volume Encryption Key (FVEK)
2. FVEK is then encrypted with the Volume Master Key (VMK)
3. VMK is in turn encrypted by one of several possible methods depending on the chosen authentication type (that is, key protectors or TPM) and recovery scenarios



The use of intermediate key (VMK between FVEK and any key protectors) allows changing the keys without the need to re-encrypt the raw data in a case a given key protector is compromised or changed.

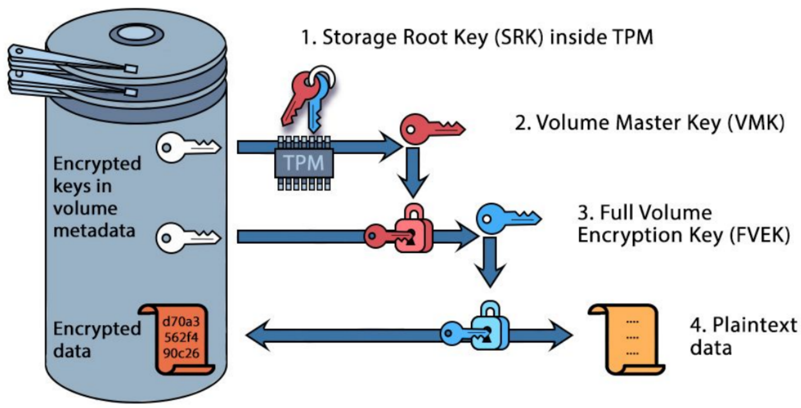
- When changing a key protector, a new VMK will be created and used to encrypt the old FVEK with the new VMK

BitLocker supports multiple key protector options, depending on the security needs and device type.

TPM only

- The TPM module (a hardware security chip) decrypts the VMK using a Storage Root Key (SRK) stored in the TPM.
- The SRK is only released if Secure Boot passes, ensuring the device boots with its original OS and configuration.
- The BitLocker volume is unlocked automatically during boot, before the user logs in.

View article.



TPM + PIN

- The TPM module will only release the VMK if the user enters a correct PIN during the pre-boot phase.
- If too many incorrect PIN attempts occur, the TPM will lock access to the encryption key, preventing brute-force attacks.

Key Takeaways:

- BitLocker is excellent for protecting against physical threats like device theft or unauthorized hard drive access.
- It does NOT protect against malware, ransomware, or unauthorized logins by users on the same computer.
- TPM + PIN is the most secure option to prevent unauthorized access, even if a device is stolen.

BitLocker poses a problem for forensic investigators, as all information on the drive will be encrypted, and therefore unreadable. Some methods for breaking BitLocker password are:

- the RAM dump/hibernation file/page file attack: this attack is universal, and works regardless of the type of protector. It dumps from the computer's volatile memory (and possibly in the page/hibernation file) the VMK that is loaded unencrypted while the volume is mounted
- BitLocker recovery keys: in many situations recovery keys are be stored in the user's Microsoft Account. Extracting those keys from their account allows instantly mounting or decrypting protected volumes regardless of the type of protector

Toolset and examples

A **loop** device is a special kind of block device that does not map to a physical hardware device (such as a hard disk) but instead maps to a regular file stored within a filesystem.

- useful to access a forensic image
- read/only can be forced
- the offset parameter could be useful to directly access a volume
- (can be used to simulate a block device to be acquired)

Key losetup Commands:

- `losetup -a`→ Shows the status of all loop devices.
- `losetup -d [device]` → Detaches a loop device.
- `losetup -f` → Finds the first available (unused) loop device.
- `losetup -o [offset]` → Starts reading data at a specific offset in the file.
- `losetup -r /dev/loop0 [srcfile]` → Sets up a read-only loop device.

dd is the precursor of all acquisition tools, allowing for the acquisition of data bit by bit in raw format.

Key `dd` Options:

- `if=` → Input file (or device to copy from).
- `of=` → Output file (or device to write to).
- `bs=` → Block size (how much data to read/write at a time).
- `conv=` → Specifies conversion options (e.g., noerror to continue on errors).

Example

```
dd if=/dev/sda of=/mnt/dest/image.dd bs=512
```

losetup and dd

In this exercise, we will simulate a block device using a compressed forensic image and interact with it as if it were a real disk.

```
wget https://github.com/enricorusso/DF_Exs/raw/main/acquisition/image.dd.gz
gunzip image.dd.gz

# Before setting up the loop device, find an available one using
losetup -f

sudo losetup -r /dev/loop1 image.dd # Now, set up the image as a read-only loop device

sudo dmesg | tail # To verify that the loop device was correctly attached, check system logs
[84058.342422] loop1: detected capacity change from 0 to 2033664

sudo fdisk -l /dev/loop1 # To inspect the loop device and view partition details

sudo partx -a /dev/loop1 # Since the loop device represents an entire disk, Linux does not automaticall

mkdir -p /mnt/forensic_image
sudo mount -o ro /dev/loop1p1 /mnt/forensic_image

ls -l /mnt/forensic_image

#clean
sudo umount /mnt/forensic_image
sudo losetup -d /dev/loop1
```

In digital forensics, verifying the integrity of a forensic image is crucial to ensure that the data remains unchanged during analysis. This is done by calculating cryptographic hash values (MD5 and SHA1) before and after mounting the image → Before using the forensic image, compute its MD5 and SHA1 hashes.

```
md5sum image.dd
446144a4af914d7e55603b6042f20db1  image.dd

sha1sum image.dd
99540f5aaa170afbab722729e980fd6dc34ff323
image.dd

md5sum /dev/loop1
446144a4af914d7e55603b6042f20db1
/dev/loop1

sha1sum /dev/loop1
99540f5aaa170afbab722729e980fd6dc34ff323
/dev/loop1
```

The dd tool does not calculate hashes during acquisition, so forensic best practices require manually computing hashes before and after imaging to ensure data integrity.

```
# Instead of separately computing hashes before and after, we can stream data from dd to tee, simultane
sudo dd if=/dev/loop1 bs=512 | tee image.dd | hashdeep -c md5,sha1 > image.src_hash # #apt install hash

# dd if=/dev/loop1 bs=512 → Reads data from the loop device.
# tee image.dd → Writes data to image.dd while also passing it to the next command.
# hashdeep -c md5,sha1 → Computes MD5 and SHA1 hashes as data is written.
# > image.src_hash → Saves the computed hashes to image.src_hash.

#Finale verification hash
md5sum image.dd
```

A faulty disk

In this exercise, we simulate a faulty disk by mapping a logical block device and introducing bad sectors. We then attempt to acquire it using dd, handling errors properly to maintain forensic integrity.

1. We create a logical "faulty" device (1Kb) with the command dmsetup*

   - `8 8 error` → [starting sector; add sector] maps the next 8 sectors of 512 byte (8 to 16) of the bad_disk device to an error area. This means that any attempt to read or write to bad_disk sectors 8 to 16 will generate an error.
   - `/dev/loop1` is the origin and must be initialized with a .dd (`sudo losetup /dev/loop0 image.dd`/)

   ```
   sudo dmsetup create bad_disk << EOF
   0 8 linear /dev/loop0 0
   8 8 error
   16 2033648 linear /dev/loop0 16
   EOF
   ```

2. Scan the simulated bad sectors.

   ```
   sh sudo badblocks -b 512 -v /dev/mapper/bad_disk
   ```

3. To ensure that all reads go directly to the faulty device (and are not cached), we disable readahead. Then, check the block device size:

   ```
   sudo blockdev --setra 0 /dev/mapper/bad_disk
   sudo blockdev --getsz /dev/mapper/bad_disk
   ```

4. Now, try acquiring the faulty disk with dd

   ```
   sudo dd if=/dev/mapper/bad_disk of=bad.dd bs=512
   ```

   Problem: dd stops when it hits a bad sector, preventing a complete acquisition.

5. To log bad sectors and replace them with zeros, use `conv=sync,noerror`

   - `tee bad.dd` → Writes the output to bad.dd while streaming it to `hashdeep` for hashing.

   ```
   sudo dd if=/dev/mapper/bad_disk bs=512 conv=sync,noerror
   | tee bad.dd | hashdeep -c md5,sha1 > bad_image.src_hash
   ```

6. After acquisition, compare the hash of bad.dd to the hash calculated during acquisition in `bad_image.src_hash`

   ```
   hashdeep -c md5,sha1 bad.dd
   ```

dc3dd

An enhanced version of dd designed specifically for digital forensics. It was developed by the DoD Cyber Crime Center (DC3) and includes several critical forensic features missing in standard dd.

Example

```
sudo dc3dd if=/dev/mapper/bad_disk of=bad.dd ssz=512 log=image.log hlog=hash.log hash=md5 hash=sha1
```

Image file formats

When acquiring digital evidence, the choice of image format is crucial for integrity, compatibility, and efficiency in analysis. There are two main categories of forensic image formats.

The output from dd acquisition is a raw image * it contains only the data from the source device * all the descriptive data about the acquisition (e.g., hashes values, dates, or times) need to be saved in a separate file

An embedded image contains data from the source device and additional descriptive data (metadata).

- **Expert Witness Format (EWF)**

  Joachim Metz (Google) created the libewf project, open source (https://github.com/libyal/libewf, apt install ewf-tools). It provides a library and set of tools to manage the ewf format.

  - `ewfacquire`: reads storage media data from devices and write files to EWF files.
  - `ewfexport`: exports storage media data in EWF files to (split) RAW format or a specific version of EWF files.
  - sudo systemctl restart guymager ewfinfo```: shows the metadata in EWF files.
  - `ewfmount`: FUSE mounts EWF files.
  - `ewfrecover`: special variant of ewfexport to create a new set of EWF files from a corrupt set.
  - `ewfverify`: verifies the storage media data in EWF files

- **Advanced Forensic Format (AFF)**

  Open Source format developed by Dr. Simson L. Garfinkel

  - Provide compressed or uncompressed image files
  - No size restriction for disk-to-image files
  - Provide space in the image file or segmented files for metadata (unlimited number)
  - Digital signatures
  - Encryption with decryption on-the-fly
  - No patents

  Still lacks wide adoption (software available at https://github.com/sshock/AFFLIBv3).

Guymager

Guymager is a graphical (Qt-based) forensic imager. It is capable of producing image files in EWF, AFF and dd format (apt install guymager).

- AFF is disabled by default (`sudo nano /etc/guymager/guymager.cfg → set AffEnabled=true → sudo systemctl restart guymager`)

FTK Imager

FTK Imager is a data preview and imaging tool used to acquire data (evidence) in a forensically sound manner by creating copies of data without making changes to the original evidence (https://www.exterro.com/ftk-product-downloads).