Data-Hygiene

> [!toc] Table of contents

Whether it's network surveillance, digital forensics, or house searches: surveillance is always about data that could potentially be used against you. That's why it's important to regularly ask yourself what data is really necessary:

- Do we need to take notes for this meeting?
- If everyone was at the meeting, notes of it may not be necessary for some things.
- Do I need to bring my mobile phone with me?
- Do I need to text my friends about the cool thing I just did?
- Bragging has been the downfall of many!

If there is no data, no one can access it. However, the assessment of a few people that certain documents are no longer necessary and that they can be destroyed may be deeply regretted a few years later. Still, depending on the stored data, simply storing documents under the bed or on an unencrypted USB drive might be too risky. So, how can data be stored securely? In any case, only in very few cases on paper!

> [!warning] Warning {static}
>
> If you have created “incriminating material” - Get rid of
>
> it ASAP!

However, most people are probably aware that simply deleting files does not mean that the data is irretrievably lost. Not even when Windows warns you that emptying the recycle bin will really make everything disappear into a black hole forever.

Deleting data securely

> [!tip] TL;DR {static}

> The safest way to delete data, is when the drive is encrypted.
>
> In those cases, every forensic tool still need the encryption password before they can read anything - even if you just deleted the files "normally".

To illustrate what happens when files are deleted “normally,” here is a metaphor:

> [!technical] SSDs vs HDDs
>
> The following scenario only applies to a limited extent to
>
> common types of storage, such as classic HDD hard drives! There are additional things to consider for flash storage such as SD cards, USB sticks, or SSDs. More on this under “Special features” below.

How Files are "deleted" - Anna & Arthur's shared apartment

Anna & Arthur live in a shared apartment. Their names and addresses are listed in the address book (unlike a phone book, everything is sorted by address here). The apartment is the storage medium (hard drive, USB stick, SD card, etc.) and Anna & Arthur are the data on that storage medium. The good old paper phone book (these huge books, where every ones landline number and home address could be looked up at) is the so called *address management system* of the storage medium.

If you want to find Arthur, you enter Arthur's address. The computer then goes to the address, fetches Arthur from his apartment, and displays him on the screen. This is normal operation when data is stored in memory and is being used.

Unfortunately, during the last action, Arthur's mask slipped down over his nose, he was identified, and now he has to leave quickly: The data must be deleted.

If you now click on “delete,” this file will be moved to the recycle bin. Nothing is really deleted when moved to the recycle bin; just think of it as a “files to be deleted” folder.

So you empty the trash can too. What has happened now? Has Arthur disappeared?

No, you have only deleted Arthur's name from the address book. Arthur himself is still sitting on his couch waiting for something to happen: The data is still physically on the storage medium. It is just no longer indexed in the memory's address directory.

If the cops look in the address book, they won't find Arthur's name anymore. But if they simply search street by street, door by door, they will eventually come across Anna & Arthur's shared apartment, where Arthur is still sitting.

The solution? Overwriting the data: Anna & Arthur need random new tenants.

> [!tip] Overwrite data! {static}
>
> In summary: Data is only truly deleted when the addresses in the memory where it was stored have been overwritten by other random data.

However, this process is not standard in any common operating system (whether PC or mobile phone), as these only delete the address entries for the files. This therefore requires additional actions.

Special characteristics

- **Addressing:** With flash memory such as SD cards, USB sticks, or SSDs, the operating system does not know exactly which bits the data is actually stored on. There is no clear connection between physical bits and externally addressable sector addresses. Therefore, these bits cannot simply be overwritten because it is not clear which ones should be overwritten.
- **Overprovisioning**: In addition, these types of memory block certain address spaces from external write access, known as “reserved blocks.” This overprovisioning has three main functions: error correction, optimization of write speed, and preservation of the storage medium's service life.

> [!technical] Technical Details - Overprovisioning
>
> - Error correction: If individual storage cells become defective (e.g., due to wear), the controller can fall back on this reserve to prevent data from being stored “corrupted.”
> - Write speed: Since the reserve blocks are already available “empty,” cells do not always have to be deleted before they can be rewritten. The controller can thus directly access empty cells and write to them immediately.
> - Lifespan: By rotating the data on the memory cells, overprovisioning prevents individual cells from remaining in the same state for a very long time. This typically causes these cells to become asymmetrical in terms of their “on” and “off” states. They therefore tend to tip in one direction or the other. This leads to errors during write operations

> because a transistor that has been "on" for years, for example, is now told to switch to “off” with an extremely short pulse. However, this may not happen because it has been “on” for so long.

Therefore, it is not sufficient to overwrite memory cells with random bits using conventional methods. This leaves the reserve blocks untouched, from which old data can be reconstructed in case of doubt. The ATA specification provides two commands for this: `SECURITY ERASE UNIT` and `ENHANCED SECURITY ERASE UNIT`. The former overwrites with zeros, the latter with random bytes. If these commands are applied to an SSD, the reserve blocks will also be overwritten. Command line tools are available for this purpose in both Linux and Windows, but they can be a little hacky. Most SSD manufacturers such as Samsung, Kingston, Western Digital, and others provide their own tools for this purpose, which can be used.

These tools basically do nothing more than apply these commands to SSDs with their own (proprietary) firmware.

Deleting encrypted data

A more efficient method is encryption. The following applies to both rotating disks (HDDs) and SSDs:

When the data carrier is encrypted, a key is generated and stored in the header of the memory. You will be asked to set a password for the encryption. This password is then used to encrypt the key stored in the header - not the data itself.

Every data read or write operation is symmetrically decrypted or encrypted using the key.

Due to the mathematical properties of modern encryption algorithms, the bit states on the physical data carrier cannot be distinguished from random bits. An encrypted data carrier therefore looks exactly the same as one that has been randomly written to.

To securely delete this data, only the key in the header of the data carrier needs to be deleted and overwritten. This not only saves a lot of time (it takes only a few minutes), but also preserves the life of the data carrier. Completely overwriting a 1TB HDD can easily take more than 5 hours.

More detailed information can be found here.

> [!info] Summary {static}
>
> - Data on unencrypted data carriers: deleted data leaves traces that can be recovered. Therefore, data must be overwritten with random bits

(preferably several times) when deleted.
- Data on encrypted storage devices: These can only be decrypted using the key in their header. This key is secured with a password. If only this key is deleted and overwritten, the data can no longer be recovered.

Version #2
Erstellt: 2026-02-15 16:16:18 UTC von ESC-IT Migration Bot
Zuletzt aktualisiert: 2026-03-18 13:28:09 UTC von ESC-IT Migration Bot