

Laboratory Activity 1

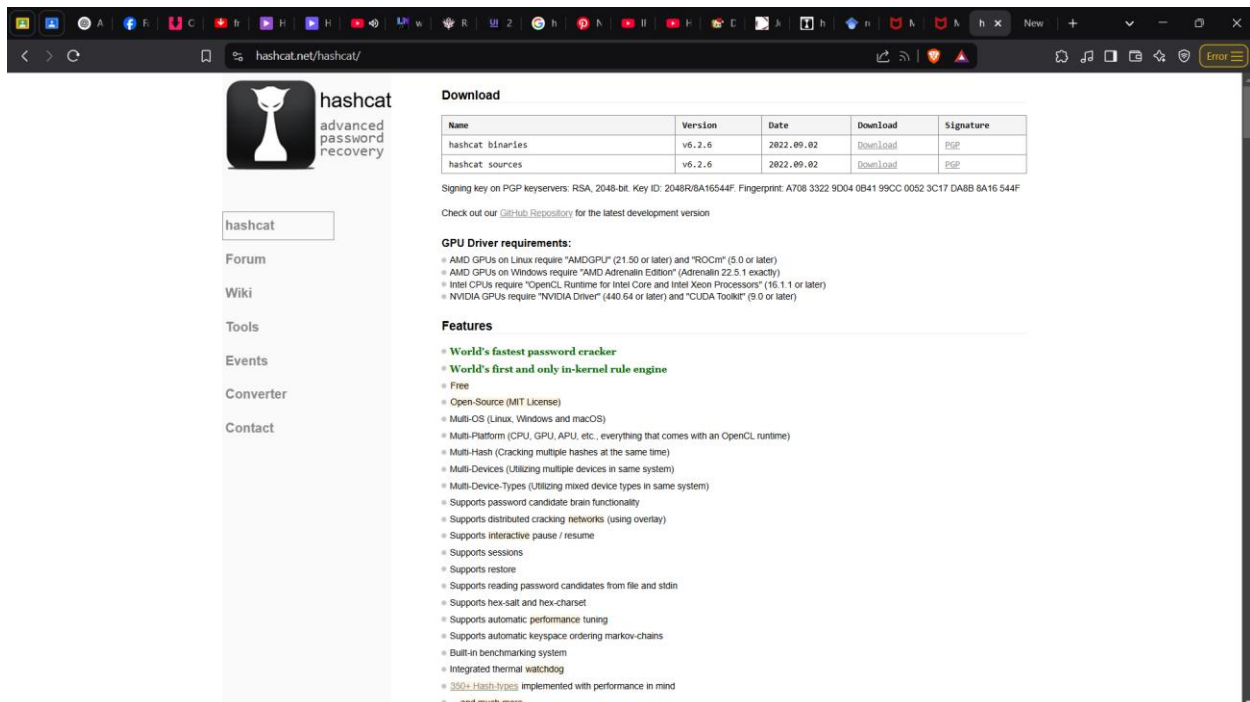
1. Download and use any software that can encrypt and decrypt a file or folder.
2. Create a simple guide/step-by-step of your work.
3. Send the software and the guide/step-by-step of your work in a pdf format.

Step 1:

I downloaded a software called john the ripper and hashcat.

Hashcat

In this part I downloaded the binary file of this one



The screenshot shows the hashcat website with a sidebar on the left containing links to Forum, Wiki, Tools, Events, Converter, and Contact. The main content area has a 'Download' section with a table of binaries and sources. Below the table is a PGP key and a link to the GitHub repository. The 'GPU Driver requirements' section lists requirements for AMD, Intel, and NVIDIA. The 'Features' section lists various capabilities of the software.

Name	Version	Date	Download	Signature
hashcat: binaries	v6.2.6	2022.09.02	Download	PGP
hashcat: sources	v6.2.6	2022.09.02	Download	PGP

Signing key on PGP keyserver: RSA, 2048-bit. Key ID: 2048RBA16544F. Fingerprint: A708 3322 9D04 0B41 99CC 0052 3C17 DA8B 8A16 544F

Check out our [GitHub Repository](#) for the latest development version

GPU Driver requirements:

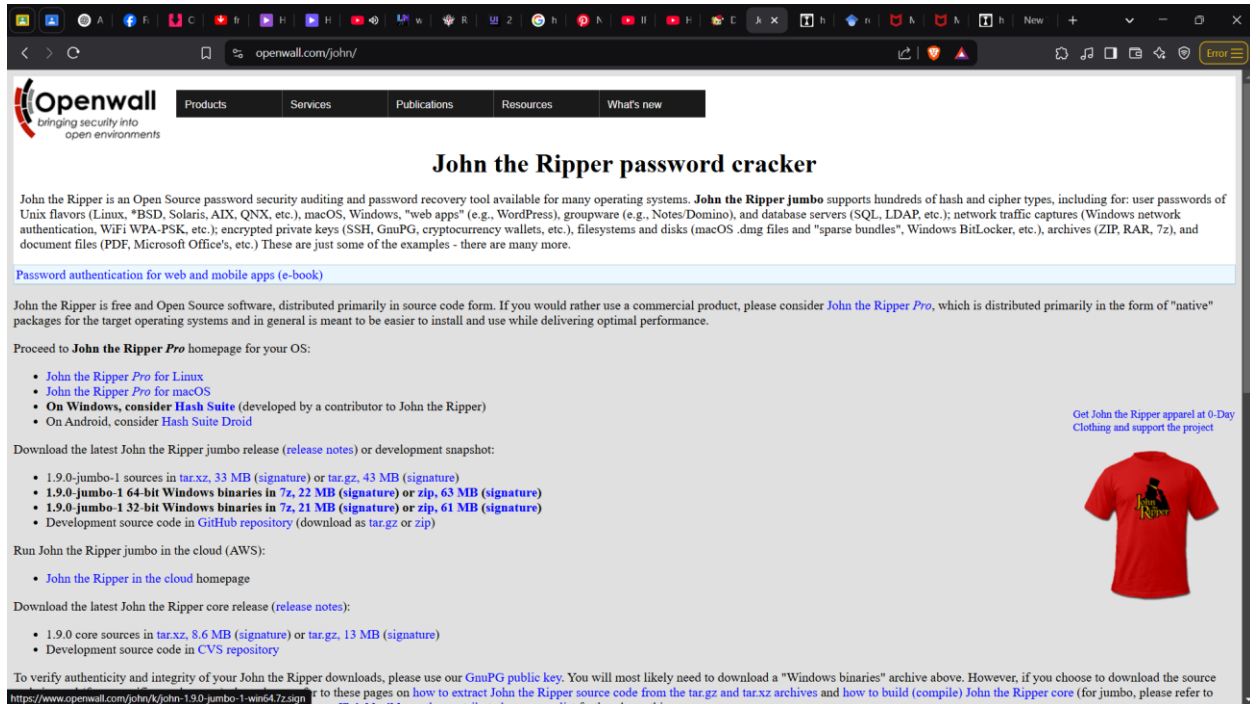
- AMD GPUs on Linux require "AMDGPU" (21.50 or later) and "ROCm" (5.0 or later)
- AMD GPUs on Windows require "AMD Adrenalin Edition" (Adrenalin 22.5.1 exactly)
- Intel CPUs require "OpenCL Runtime for Intel Core and Intel Xeon Processors" (18.1.1 or later)
- NVIDIA GPUs require "NVIDIA Driver" (440.64 or later) and "CUDA Toolkit" (9.0 or later)

Features

- World's fastest password cracker
- World's first and only in-kernel rule engine
- Free
- Open-Source (MIT License)
- Multi-OS (Linux, Windows and macOS)
- Multi-Platform (CPU, GPU, APU, etc., everything that comes with an OpenCL runtime)
- Multi-Hash (Cracking multiple hashes at the same time)
- Multi-Devices (Utilizing multiple devices in same system)
- Multi-Device-Types (Utilizing mixed device types in same system)
- Supports password candidate brain functionality
- Supports distributed cracking networks (using overlay)
- Supports interactive pause / resume
- Supports sessions
- Supports restore
- Supports reading password candidates from file and stdin
- Supports hex-salt and hex-charset
- Supports automatic performance tuning
- Supports automatic keyspace ordering markov-chains
- Built-in benchmarking system
- Integrated thermal watchdog
- 350+ hash-types implemented with performance in mind
- ... and much more

John the ripper

There in john the ripper I downloaded the zip file of it



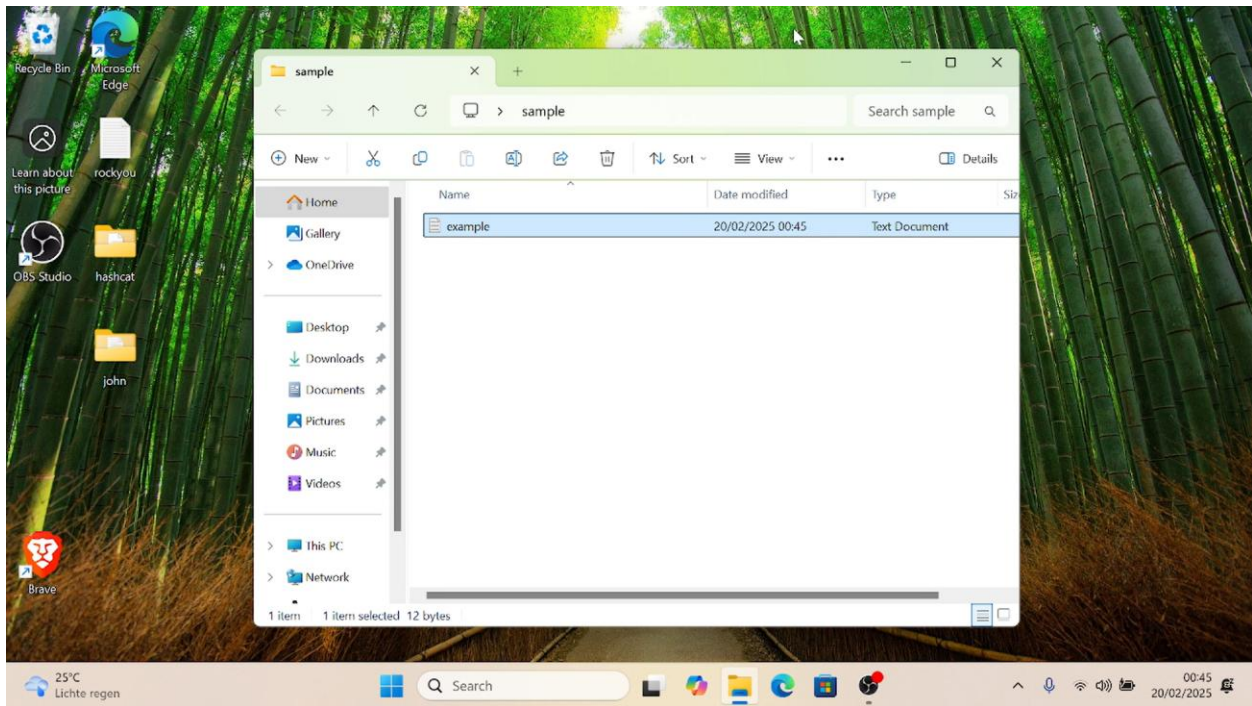
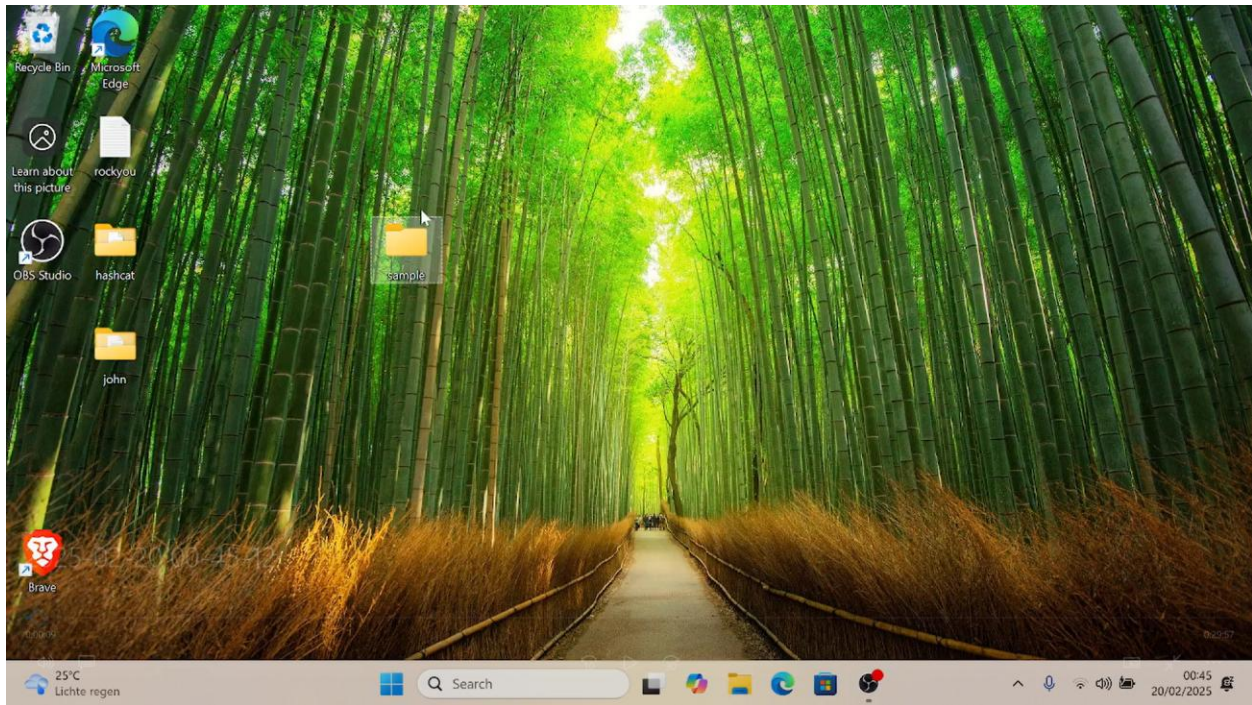
The screenshot shows the Openwall website for John the Ripper. The header includes the Openwall logo and navigation links: Products, Services, Publications, Resources, and What's new. The main heading is "John the Ripper password cracker". Below this, a paragraph describes John the Ripper as an Open Source password security auditing and password recovery tool available for many operating systems. It lists supported hash and cipher types, including user passwords of Unix flavors (Linux, BSD, Solaris, AIX, QNX, etc.), macOS, Windows, "web apps" (e.g., WordPress), groupware (e.g., Notes/Domino), and database servers (SQL, LDAP, etc.). It also mentions network traffic captures (Windows network authentication, WiFi WPA-PSK, etc.), encrypted private keys (SSH, GnuPG, cryptocurrency wallets, etc.), filesystems and disks (macOS .dmg files and "sparse bundles", Windows BitLocker, etc.), archives (ZIP, RAR, 7z), and document files (PDF, Microsoft Office's, etc.). A link to "Password authentication for web and mobile apps (e-book)" is provided. A section titled "John the Ripper is free and Open Source software, distributed primarily in source code form." explains that if a commercial product is preferred, John the Ripper Pro is available. It states that John the Ripper Pro is distributed primarily in the form of "native" packages for the target operating systems and is meant to be easier to install and use while delivering optimal performance. A link to "Proceed to John the Ripper Pro homepage for your OS:" is provided. A list of links for John the Ripper Pro is shown: John the Ripper Pro for Linux, John the Ripper Pro for macOS, On Windows, consider Hash Suite (developed by a contributor to John the Ripper), and On Android, consider Hash Suite Droid. A section titled "Download the latest John the Ripper jumbo release (release notes) or development snapshot:" lists four options: 1.9.0-jumbo-1 sources in tar.xz (33 MB signature) or tar.gz (43 MB signature), 1.9.0-jumbo-1 64-bit Windows binaries in 7z (22 MB signature) or zip (63 MB signature), 1.9.0-jumbo-1 32-bit Windows binaries in 7z (21 MB signature) or zip (61 MB signature), and Development source code in GitHub repository (download as tar.gz or zip). A section titled "Run John the Ripper jumbo in the cloud (AWS):" includes a link to "John the Ripper in the cloud homepage". A section titled "Download the latest John the Ripper core release (release notes):" lists two options: 1.9.0 core sources in tar.xz (8.6 MB signature) or tar.gz (13 MB signature), and Development source code in CVS repository. A footer note states: "To verify authenticity and integrity of your John the Ripper downloads, please use our GnuPG public key. You will most likely need to download a 'Windows binaries' archive above. However, if you choose to download the source code, please refer to these pages on how to extract John the Ripper source code from the tar.gz and tar.xz archives and how to build (compile) John the Ripper core (for jumbo, please refer to https://www.openwall.com/john/v/john-1.9.0-jumbo-1-win64.7z.sign)". On the right side of the page, there is a red t-shirt with a John the Ripper logo and the text "Get John the Ripper apparel at 0-Day Clothing and support the project".

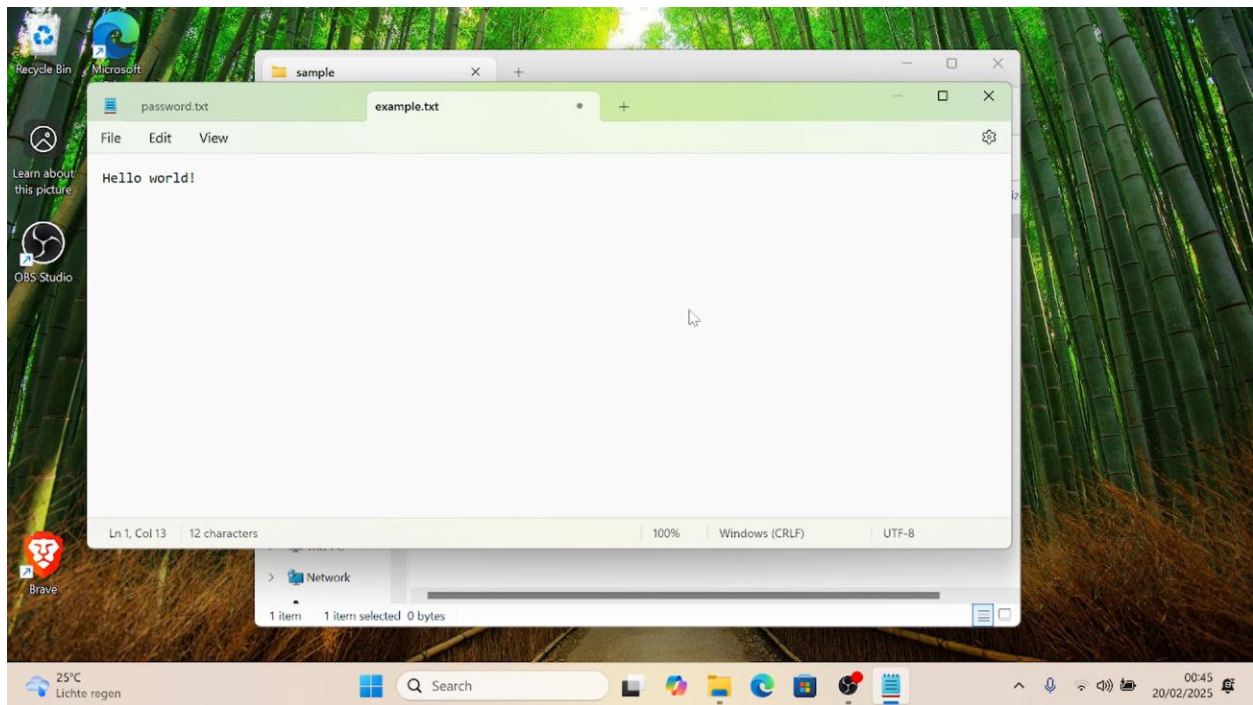
After downloading hashcat and john the ripper I proceeded into the main topic which is to encrypt a folder and on how to cracked the password on this folder using the software that I downloaded.

Here are my way on how to encrypt a folder:

Step 1:

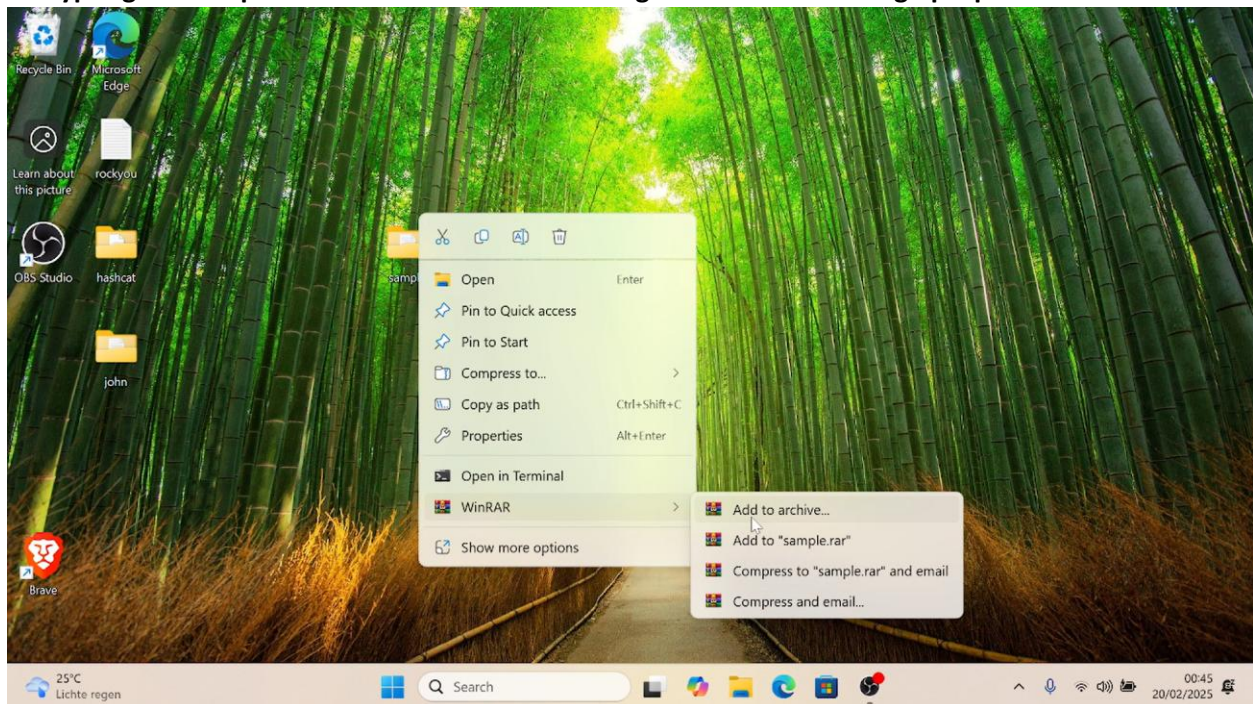
I created a folder name "sample", then putting a text document file inside of it where there consists a word "Hello world!".

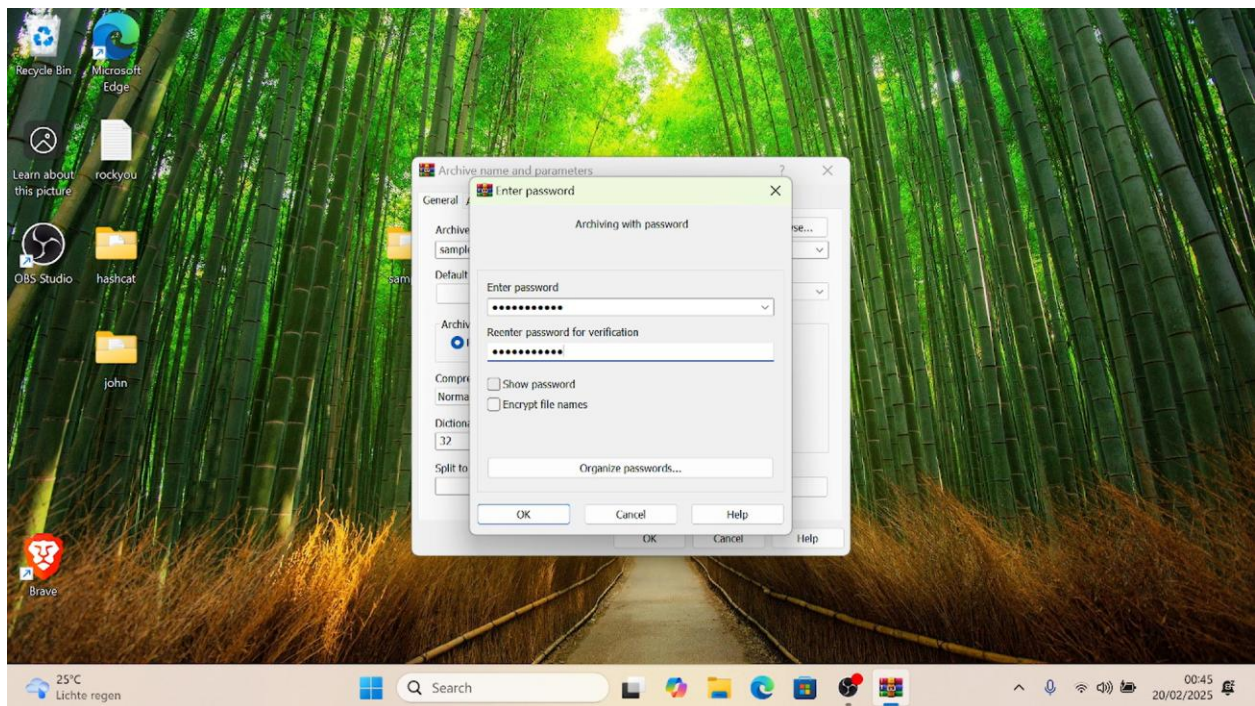
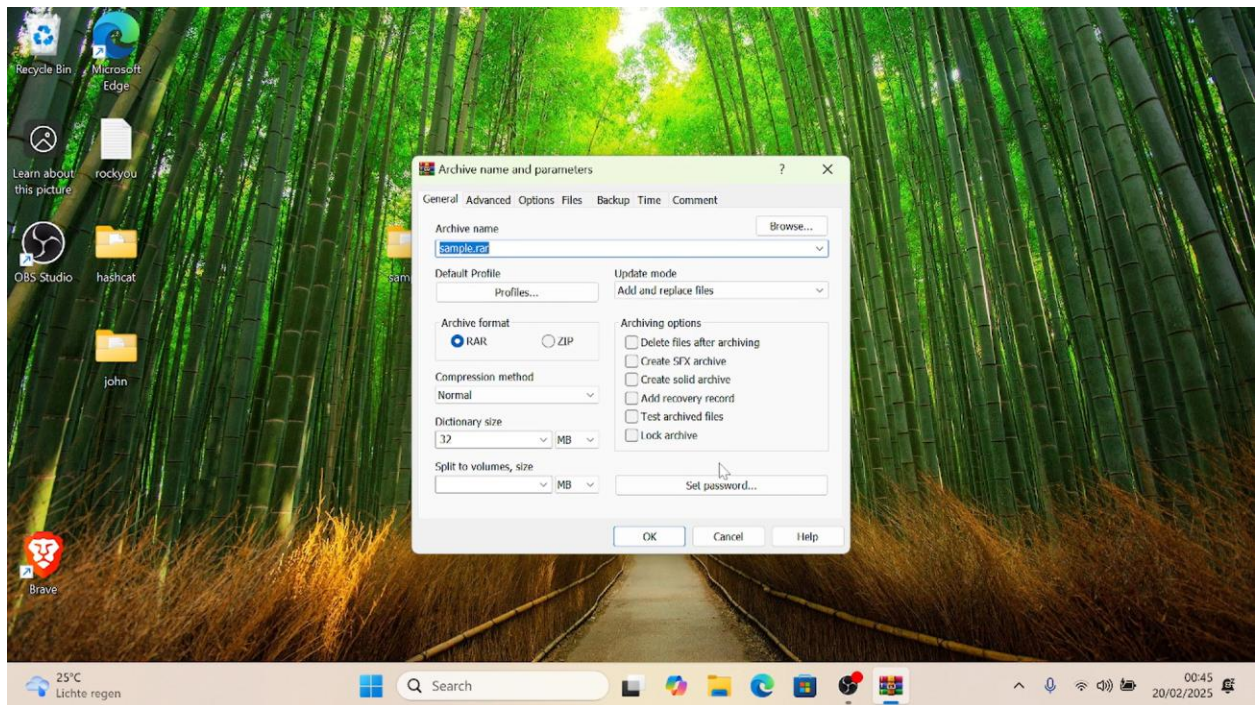


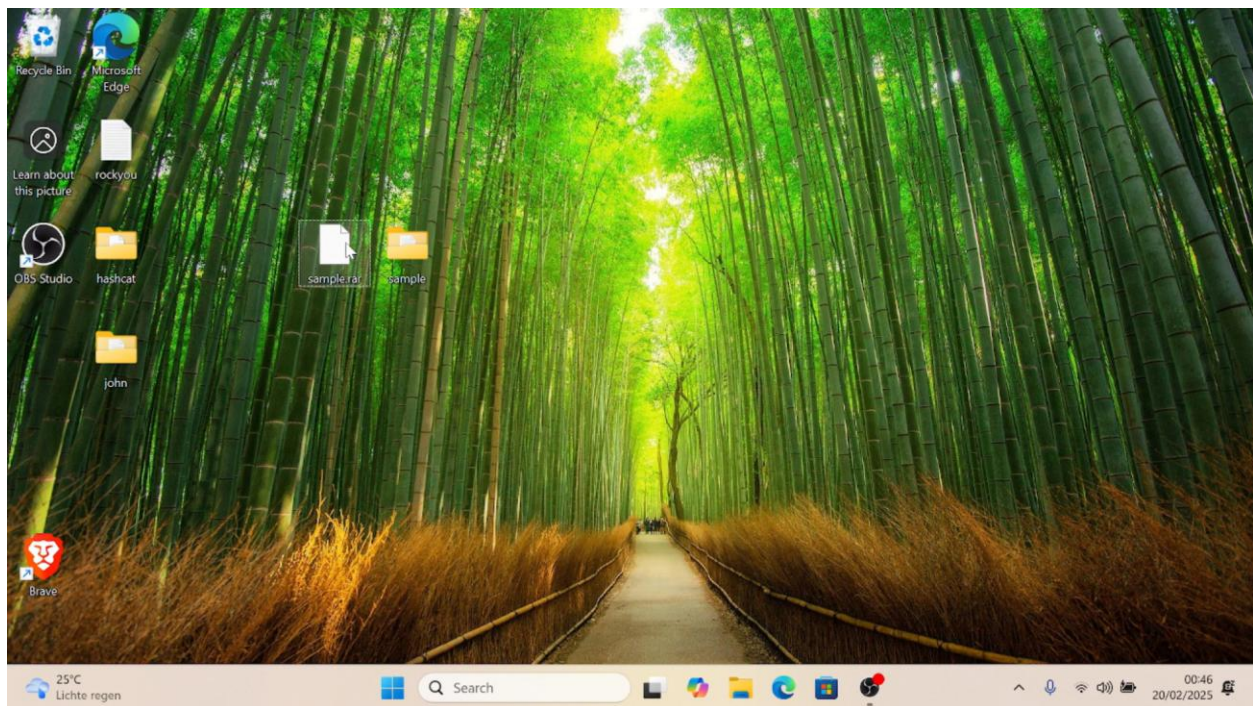
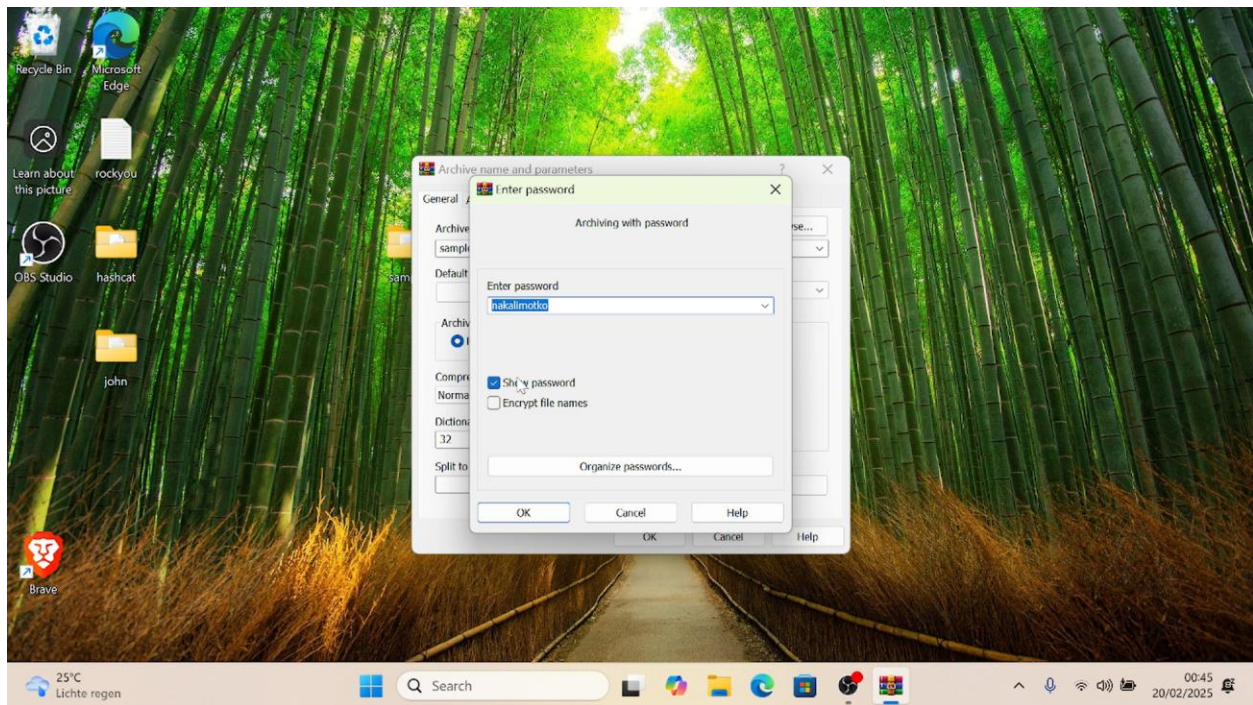


Step 2:

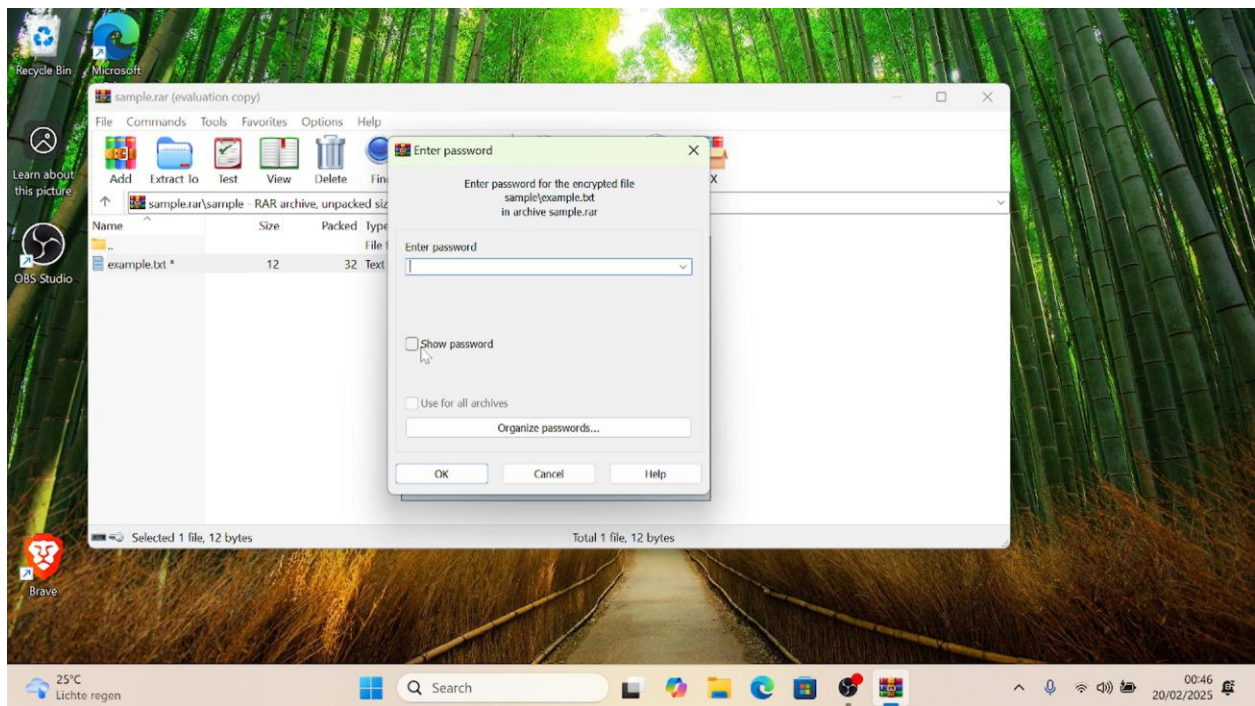
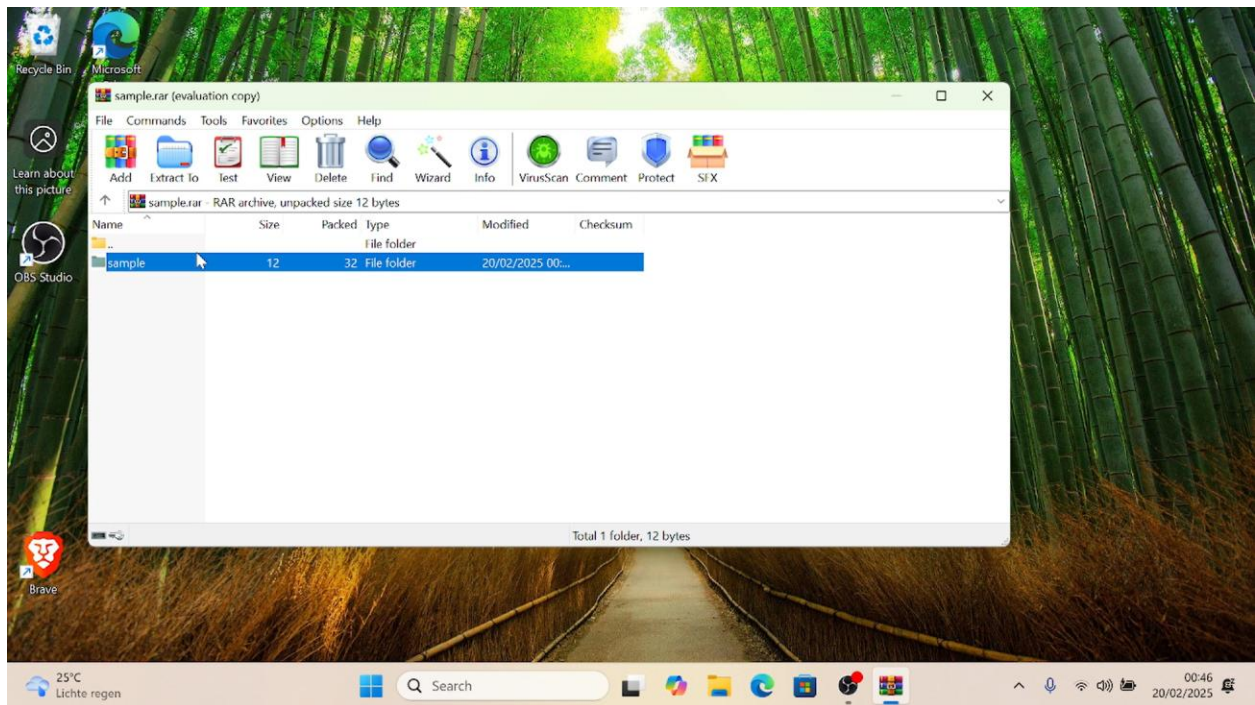
Encrypting the sample file that I created earlier using WinRAR then setting up a password there.

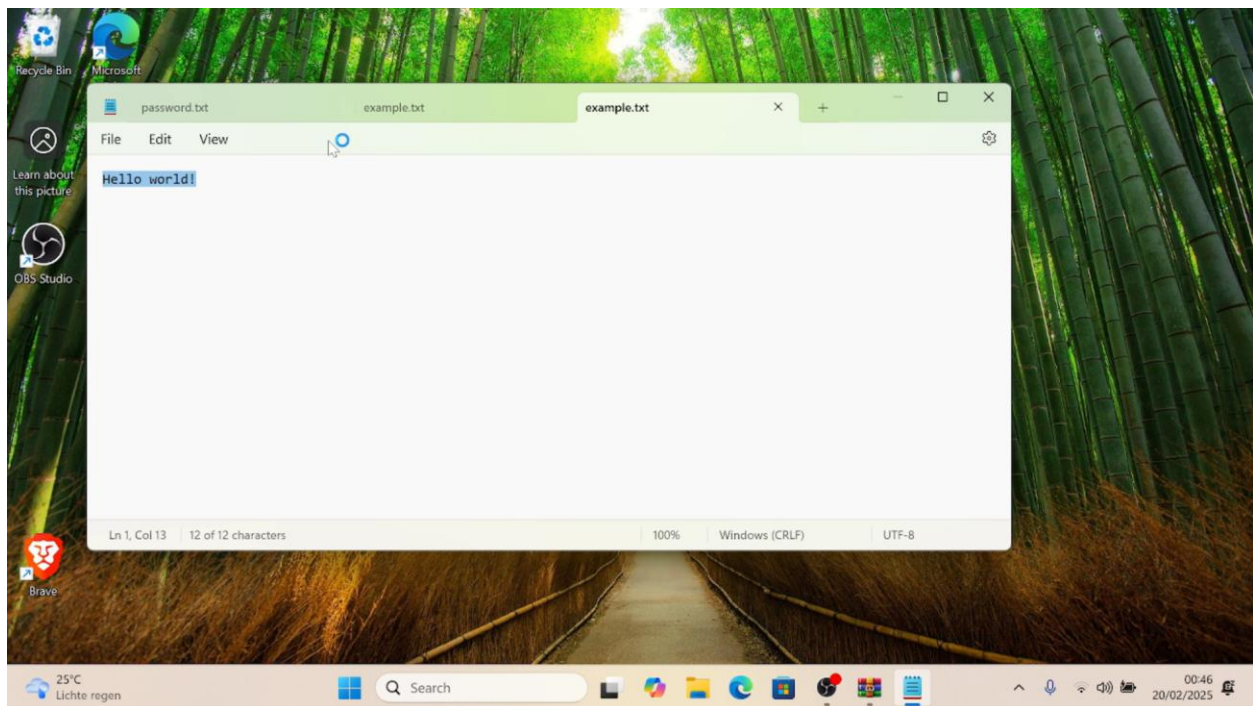
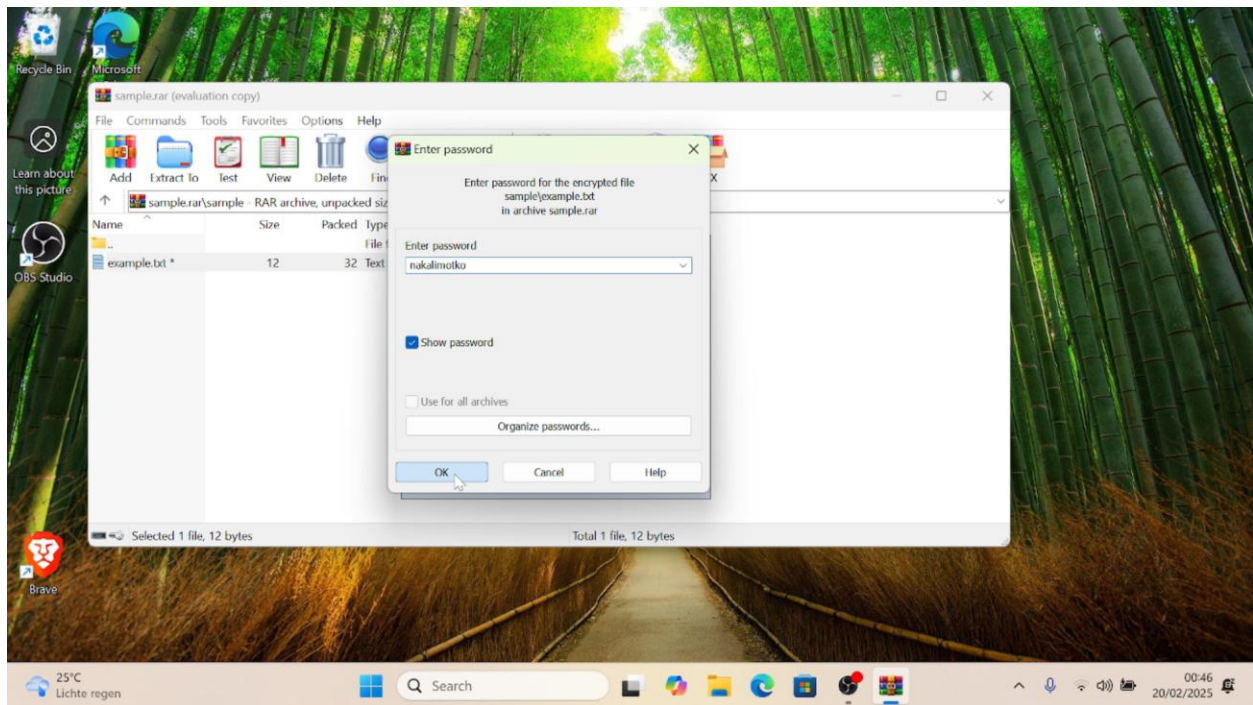






Then I checked the file to see if it was really encrypted.





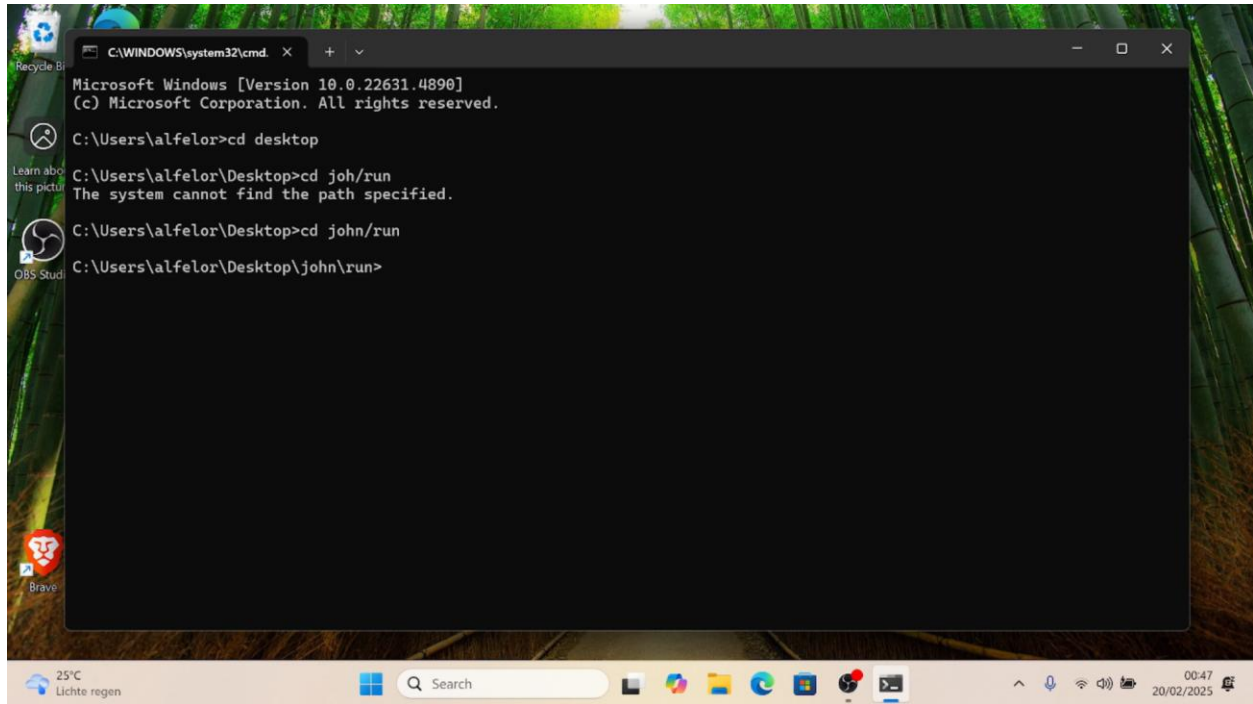
The encryption was succesfull.

In this part this is the decrypting process.

In decrypting I am using a command prompt (cmd), because the software that I downloaded earlier didn't have a Graphical User Interface (GUI).

Step 1:

Locating the location of "john" which are the name of the software that I've downloaded earlier.



A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd". The window shows the following commands and output:

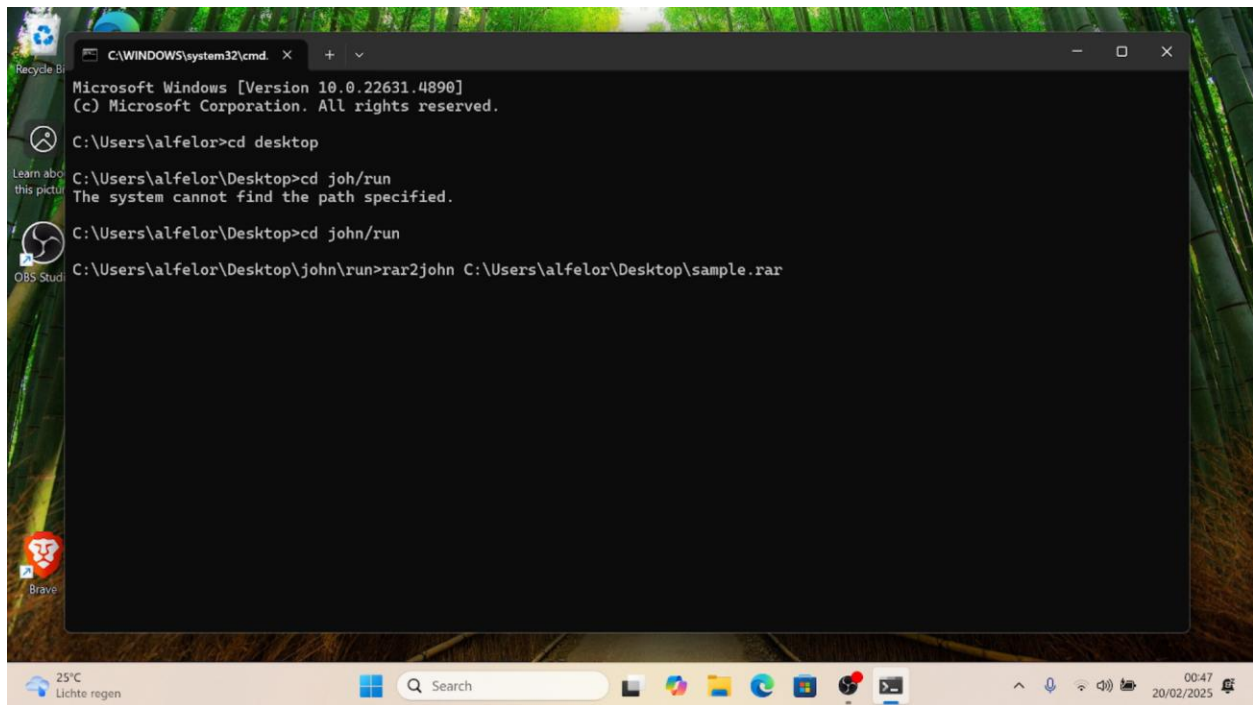
```
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alfelor>cd desktop
C:\Users\alfelor\Desktop>cd joh/run
The system cannot find the path specified.
C:\Users\alfelor\Desktop>cd john/run
C:\Users\alfelor\Desktop\john\run>
```

The background of the desktop shows a green forest scene. The taskbar at the bottom displays the date and time as 00:47 on 20/02/2025, along with various system icons and a search bar.

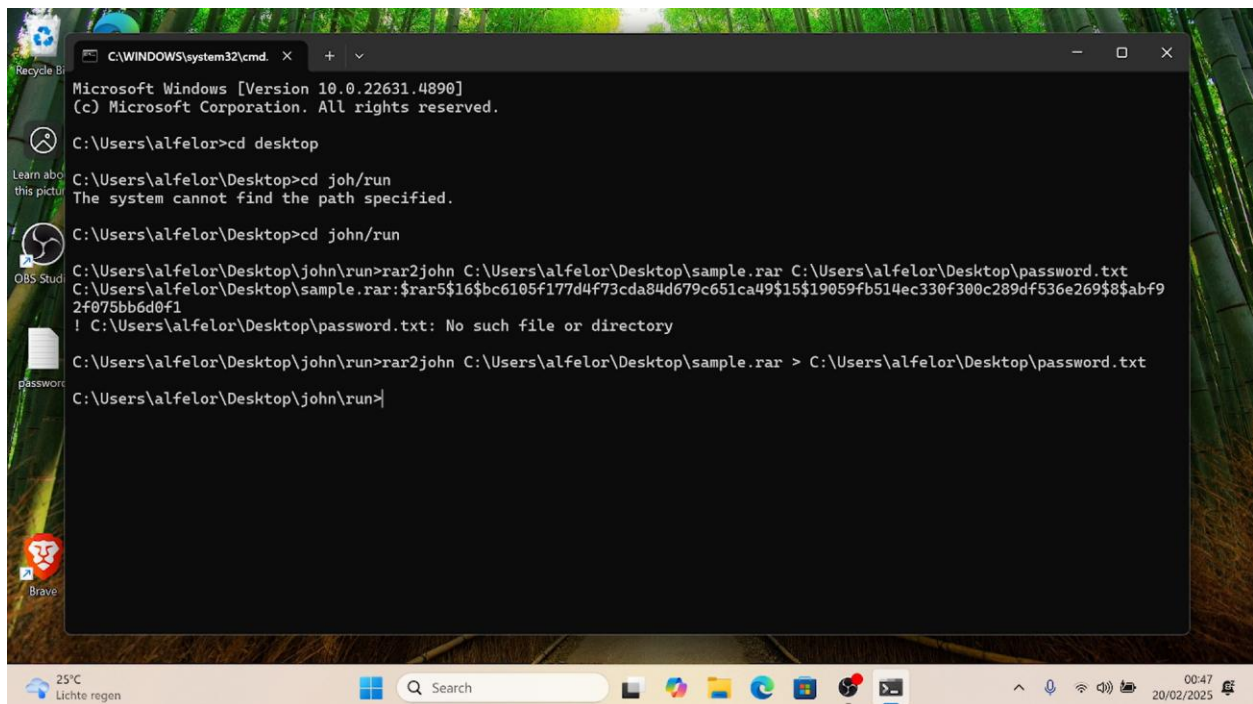
Step 2:

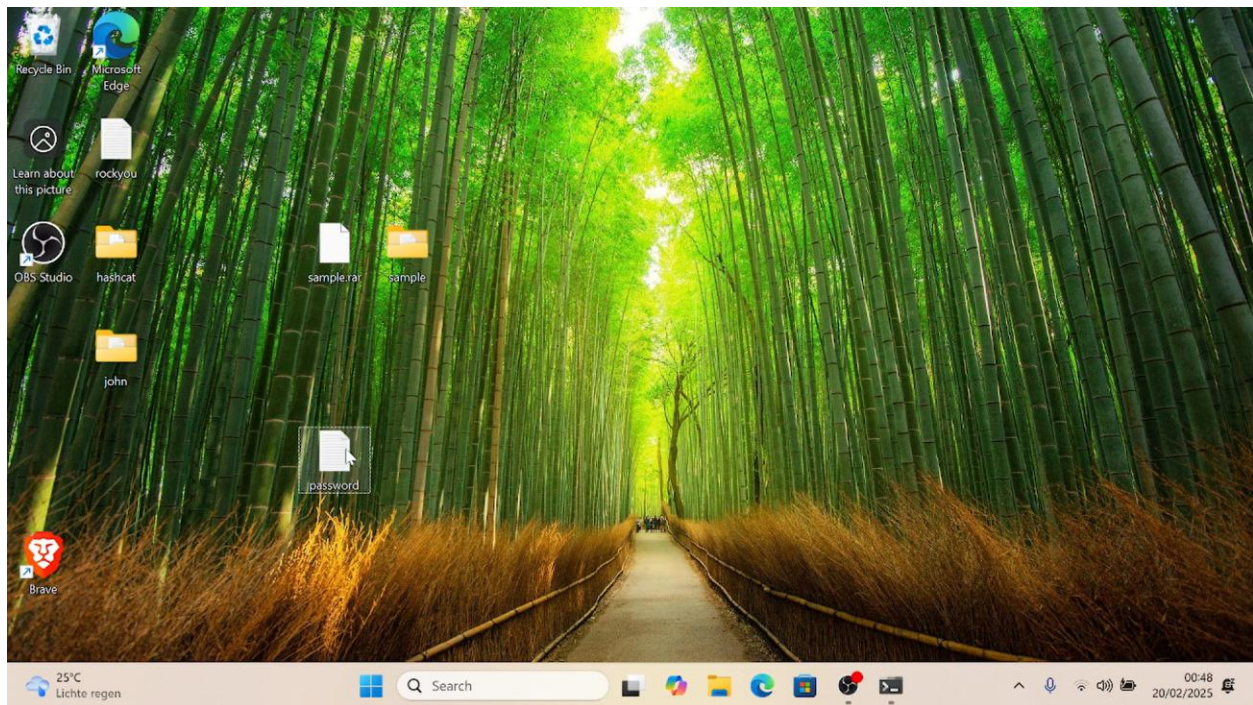
"rar2john" This command is to get the hashed password of the sample.rar that I've encrypt earlier. Then put the designated location of the folder that I want to get there hashed password.



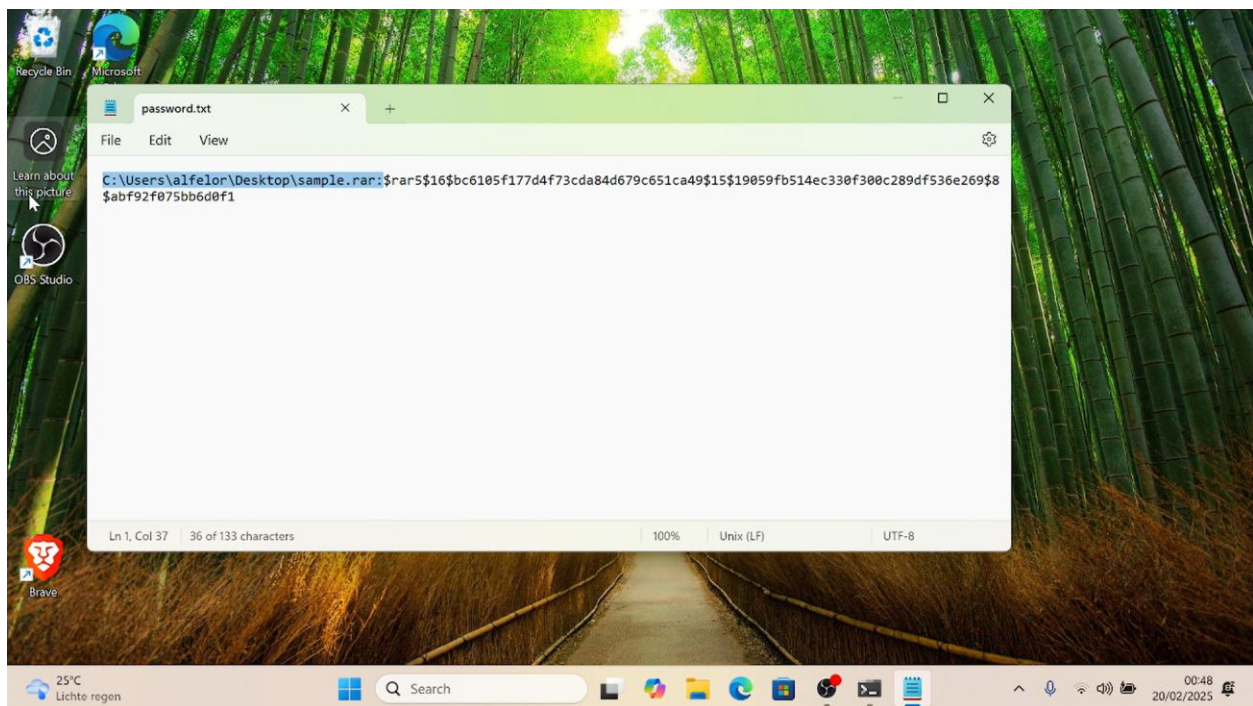
Step 3:

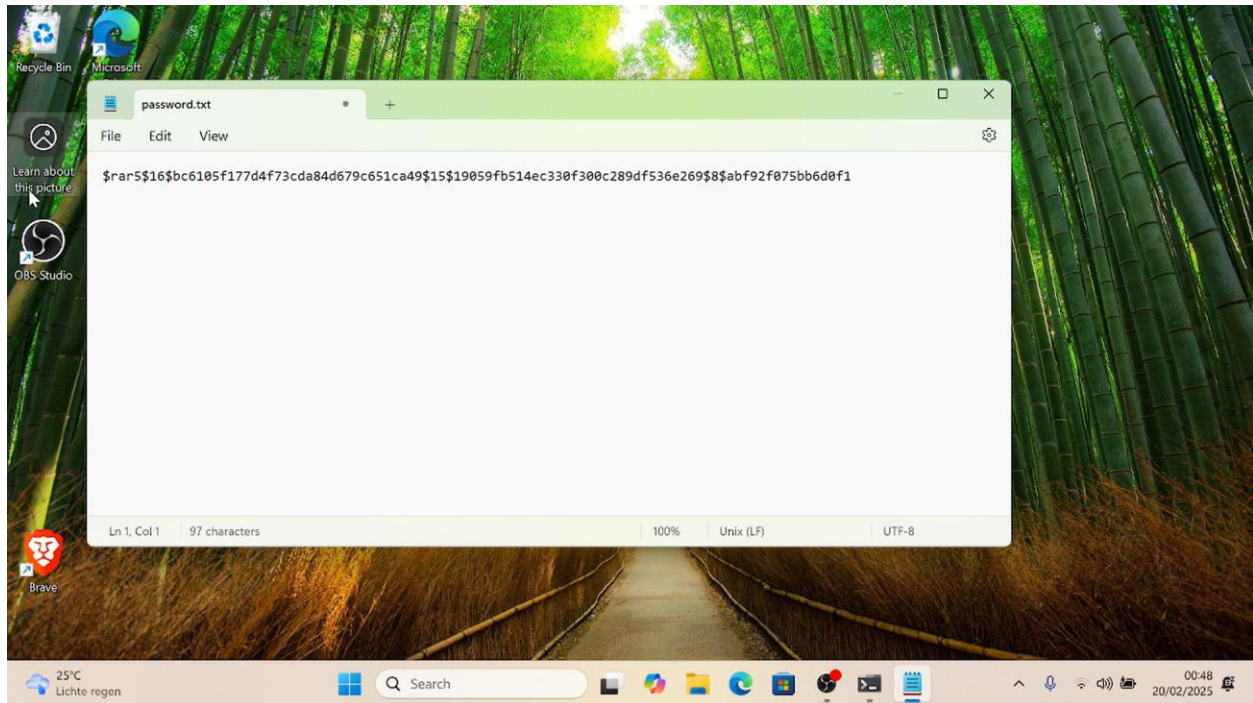
Putting a destination as to where I want to put the hashed password of the folder that I hashed using john software by naming it a password.txt, in this part I just put this one in the desktop so that it is easy to locate..





In this part I highlight the part that are need to remove, because this is not needed, the only part that are needed is the text that are not highlight there. That was the hashed password of the file that I encrypt before.

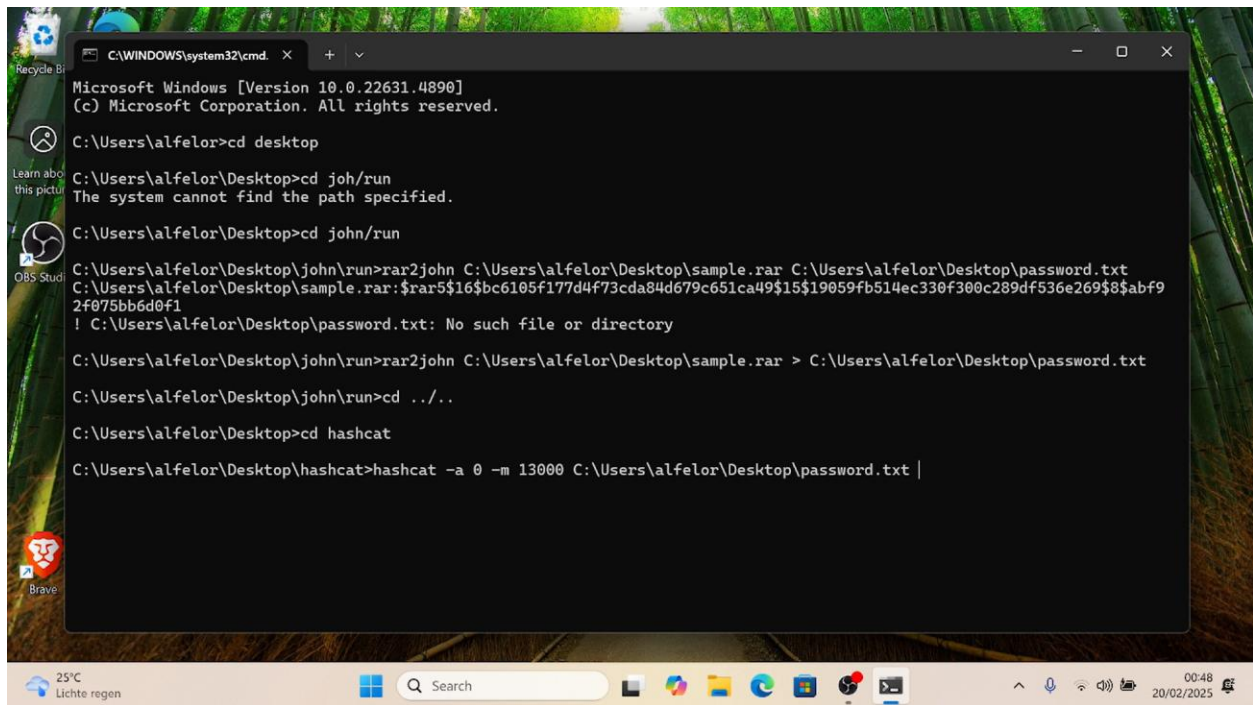




Step 4:

In this part I'm using the hashcat software to cracked the password that are hashed earlier using the john software.

“hashcat -a 0 -m 13000” This is the command that I use in my process (-a) is attack (0) is direct and (-m) mode which means my attack is a direct mode and (13000) I use this one because this is the algorithm of RAR5.



```
C:\WINDOWS\system32\cmd. X + v
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alfelor>cd desktop
C:\Users\alfelor\Desktop>cd joh/run
The system cannot find the path specified.

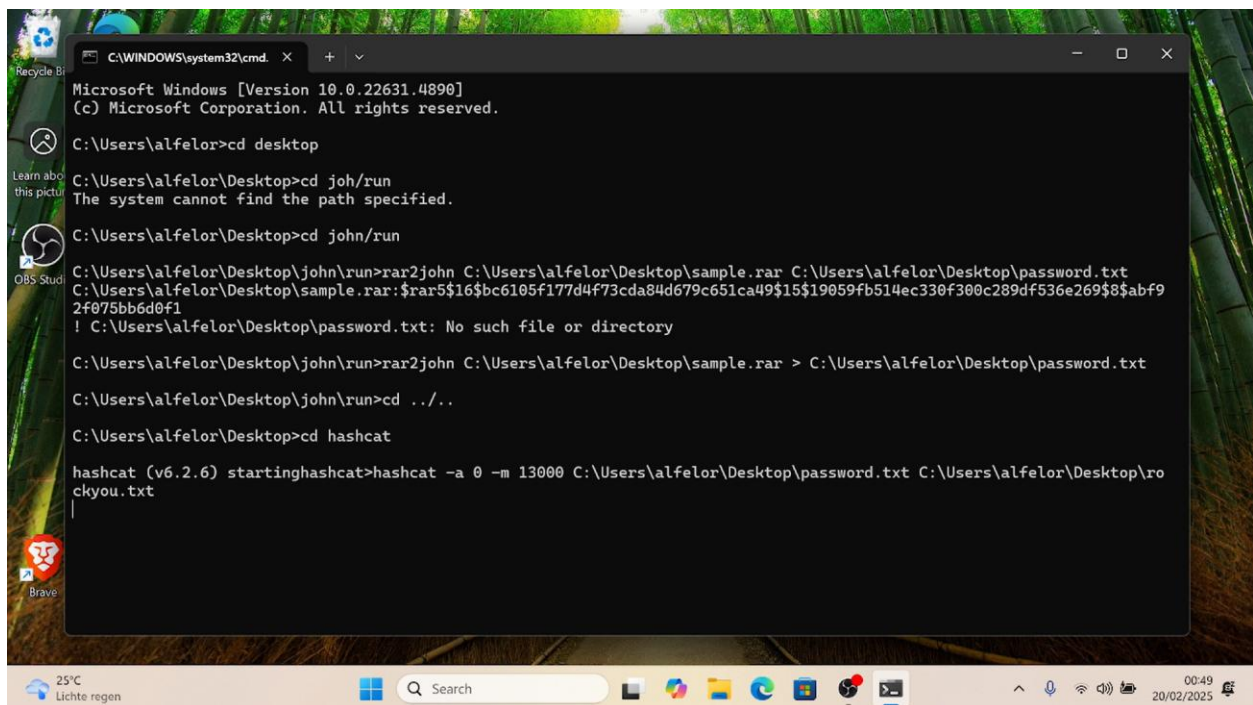
C:\Users\alfelor\Desktop>cd john/run
C:\Users\alfelor\Desktop\john\run>rar2john C:\Users\alfelor\Desktop\sample.rar C:\Users\alfelor\Desktop\password.txt
C:\Users\alfelor\Desktop\sample.rar:$rar5$16$bc6105f177d4f73cda84d679c651ca49$15$19059fb514ec330f300c289df536e269$8$abf9
2f075bb6d0f1
! C:\Users\alfelor\Desktop\password.txt: No such file or directory

C:\Users\alfelor\Desktop\john\run>rar2john C:\Users\alfelor\Desktop\sample.rar > C:\Users\alfelor\Desktop\password.txt

C:\Users\alfelor\Desktop\john\run>cd ../../
C:\Users\alfelor\Desktop>cd hashcat
C:\Users\alfelor\Desktop\hashcat>hashcat -a 0 -m 13000 C:\Users\alfelor\Desktop\password.txt |
```

Step 5:

There I use a world list to speed up decrypting process this one is only optional.



```
C:\WINDOWS\system32\cmd. X + v
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alfelor>cd desktop
C:\Users\alfelor\Desktop>cd joh/run
The system cannot find the path specified.

C:\Users\alfelor\Desktop>cd john/run
C:\Users\alfelor\Desktop\john\run>rar2john C:\Users\alfelor\Desktop\sample.rar C:\Users\alfelor\Desktop\password.txt
C:\Users\alfelor\Desktop\sample.rar:$rar5$16$bc6105f177d4f73cda84d679c651ca49$15$19059fb514ec330f300c289df536e269$8$abf9
2f075bb6d0f1
! C:\Users\alfelor\Desktop\password.txt: No such file or directory

C:\Users\alfelor\Desktop\john\run>rar2john C:\Users\alfelor\Desktop\sample.rar > C:\Users\alfelor\Desktop\password.txt

C:\Users\alfelor\Desktop\john\run>cd ../../
C:\Users\alfelor\Desktop>cd hashcat
hashcat (v6.2.6) startinghashcat>hashcat -a 0 -m 13000 C:\Users\alfelor\Desktop\password.txt C:\Users\alfelor\Desktop\ro
ckyou.txt
|
```

```
C:\WINDOWS\system32\cmd. x + v
C:\Users\alfelor\Desktop\john>run>cd ../../
C:\Users\alfelor\Desktop>cd hashcat

hashcat (v6.2.6) startinghashcat>hashcat -a 0 -m 13000 C:\Users\alfelor\Desktop\password.txt C:\Users\alfelor\Desktop\r
ockyou.txt
OpenCL API (OpenCL 3.0 ) - Platform #1 [Intel(R) Corporation]
=====
* Device #1: Intel(R) HD Graphics 620, 1568/3222 MB (805 MB allocatable), 24MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-SIMD-LOOP

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 309 MB

Starting self-test. Please be patient...|
```

Just wait there until the password was decrypt, and just press {s} to see the status.

```
Host memory required for this attack: 309 MB

Dictionary cache built:
* Filename..: C:\Users\alfelor\Desktop\rackyou.txt
* Passwords.: 1222832
* Bytes.....: 10518511
* Keyspace...: 1222832
* Runtime....: 0 secs

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit =>

Session.....: hashcat
Status.....: Running
Hash.Mode.....: 13000 (RAR5)
Hash.Target.....: $rar5$16$bc6105f177d4f73cda84d679c651ca49$15$19059f...b6d0f1
Time.Started.....: Thu Feb 20 00:49:18 2025 (3 secs)
Time.Estimated...: Thu Feb 20 01:23:37 2025 (34 mins, 16 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (C:\Users\alfelor\Desktop\rackyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 595 H/s (7.87ms) @ Accel:1 Loops:32 Thr:256 Vec:1
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.....: 0/1222832 (0.00%)
Rejected.....: 0/0 (0.00%)
Restore.Point...: 0/1222832 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:7456-7488
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 -> horoscope

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => |
```


There the decrypting process was done, and there are the password of the file that I encrypt earlier.

```
C:\WINDOWS\system32\cmd. X + v
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (C:\Users\alfelor\Desktop\rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 570 H/s (7.57ms) @ Accel:1 Loops:32 Thr:256 Vec:1
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.....: 774144/1222832 (63.31%)
Rejected.....: 0/774144 (0.00%)
Restore.Point....: 774144/1222832 (63.31%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:13600-13632
Candidate.Engine.: Device Generator
Candidates.#1....: tekelodiego -> supapet

$rar5$16$bc6105f177d4f73cda84d679c651ca49$15$19059fb514ec330f300c289df536e269$8$abf92f075bb6d0f1:nakalimotko

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 13000 (RAR5)
Hash.Target.....: $rar5$16$bc6105f177d4f73cda84d679c651ca49$15$19059fb514ec330f300c289df536e269$8$abf92f075bb6d0f1
Time.Started....: Thu Feb 20 00:49:18 2025 (24 mins, 48 secs)
Time.Estimated...: Thu Feb 20 01:14:06 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (C:\Users\alfelor\Desktop\rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 570 H/s (7.56ms) @ Accel:1 Loops:32 Thr:256 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 847872/1222832 (69.34%)
Rejected.....: 0/847872 (0.00%)
Restore.Point....: 841728/1222832 (68.83%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:32768-32799
Candidate.Engine.: Device Generator
Candidates.#1....: newme06 -> musicall

S|
```

That's it for my encrypting and decrypting.