

# Conestoga College

<b>Course</b>	INFO8965: Computer and Network Security
<b>Activity</b>	Cryptography
<b>Student Name</b>	
<b>Date performed</b>	

## Objectives

- Understand how Steganography works
- Learning about hashing algorithms and password cracking
- Learn how to use steghide and hashcat tools

## Resources

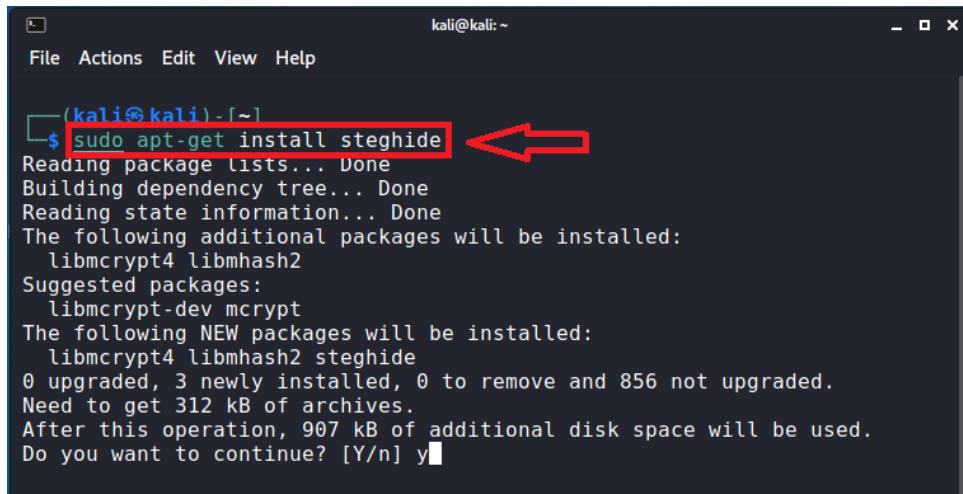
- PC / Laptop
- Virtual Box: <https://www.virtualbox.org/wiki/Downloads>
- Linux Kali
- steghide, hashcat

## (A) Steganography

The following tasks need to be performed on Kali Linux. Open Terminal in Kali Linux. Install “**steghide**” on Kali by using the apt-get install command.

```
sudo apt-get install steghide
```

Follow the prompt and press “y” when asked for.

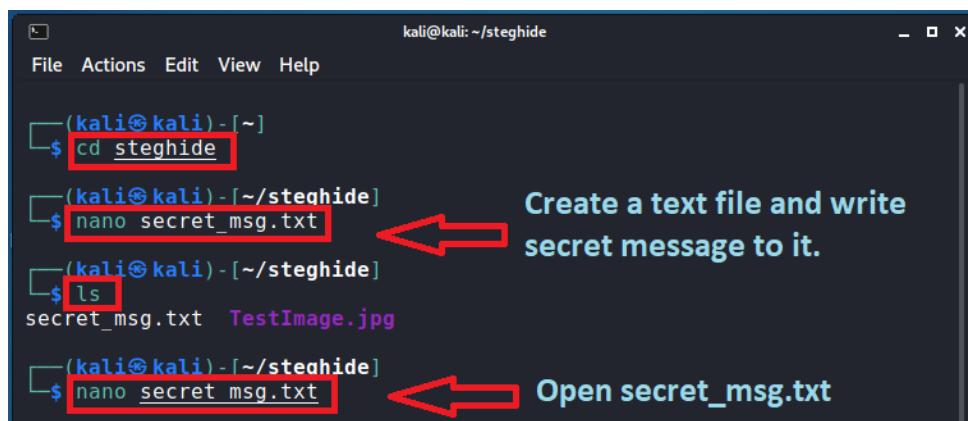


```
kali@kali:~$ sudo apt-get install steghide
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libmcrypt4 libmhash2
Suggested packages:
  libmcrypt-dev mcrypt
The following NEW packages will be installed:
  libmcrypt4 libmhash2 steghide
0 upgraded, 3 newly installed, 0 to remove and 856 not upgraded.
Need to get 312 kB of archives.
After this operation, 907 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Once the app is installed you can use it to hide a message inside an image as discussed in class. Type “**steghide --help**” to see the list of option associated with this command.

Create a folder named **steghide**. Use change-directory command ‘**cd**’ to navigate to this folder. [Hint: If required, use **pwd** to see your current working director]

- Open firefox and download any image in this folder. Name it **TestImage.jpg**.
- Create a text file using **nano** command as shown below. Write your name and some secret text in it.
- Press **Ctrl + O** to save the file. When asked, give it a name **secret\_msg.txt**. Press **Ctrl + X** to close it and return to terminal.



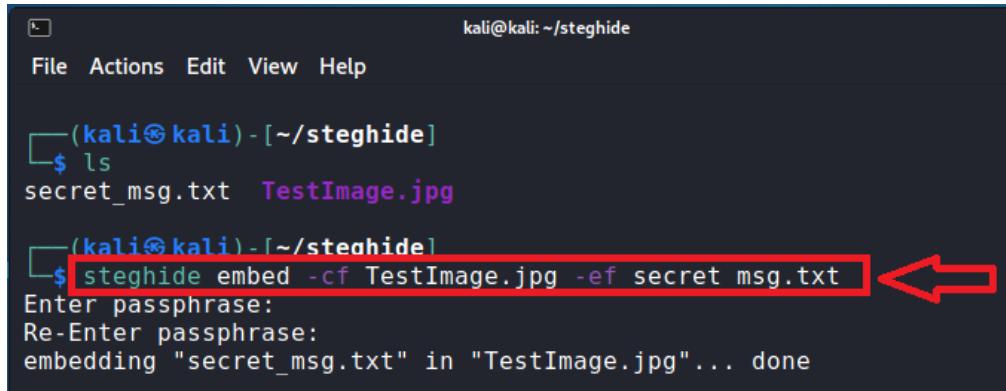
```
kali@kali:~/steghide$ cd steghide
(kali@kali:~/steghide)$ nano secret_msg.txt
(kali@kali:~/steghide)$ ls
secret_msg.txt  TestImage.jpg
(kali@kali:~/steghide)$ nano secret_msg.txt
```

Create a text file and write secret message to it.

Open secret\_msg.txt

Use the following command to embed the text file in the image. You also need to choose a password to encrypt the message.

```
steghide embed -cf TestImage.jpg -ef secret_msg.txt
```

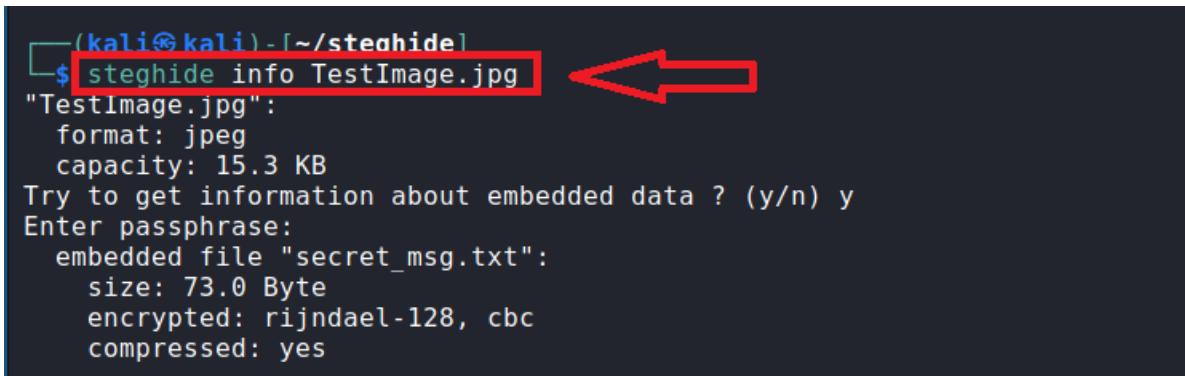


A terminal window titled "kali@kali: ~/steghide". The command \$ steghide embed -cf TestImage.jpg -ef secret\_msg.txt is highlighted with a red box and a red arrow pointing to it from the left. The output shows the command being run, followed by prompts for a passphrase and confirmation of embedding success.

```
kali@kali: ~/steghide
File Actions Edit View Help
(kali㉿kali)-[~/steghide]
$ ls
secret_msg.txt  TestImage.jpg
(kali㉿kali)-[~/steghide]
$ steghide embed -cf TestImage.jpg -ef secret_msg.txt
Enter passphrase:
Re-Enter passphrase:
embedding "secret_msg.txt" in "TestImage.jpg"... done
```

You can find out information about the steganography by using the info options.

```
steghide info TestImage.jpg
```



A terminal window titled "kali@kali: ~/steghide". The command \$ steghide info TestImage.jpg is highlighted with a red box and a red arrow pointing to it from the left. The output provides details about the embedded file, including its size, encryption method, and compression status.

```
(kali㉿kali)-[~/steghide]
$ steghide info TestImage.jpg
"TestImage.jpg":
  format: jpeg
  capacity: 15.3 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "secret_msg.txt":
    size: 73.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
```

Send the file to your friend or you can extract the information yourself using the **extract** option as mentioned in the command below. Enter the passphrase you chose before and it will extract the text file.

Navigate to **steghide** directory and execute the following commands (one-by-one). In Linux Shell, a single-line comment starts with hashtag (#) symbol with no white spaces and lasts till the end of the line.

**Note:** Take a screen capture of the terminal and paste it in the **Output #1** (2 Marks).

Commands	Purpose
<code>clear ls rm secret_msg.txt steghide extract -sf TestImage.jpg ls cat secret msg.txt</code>	# Clear screen # List # Remove file # Extract hidden information # List # Display text file on the terminal

The following screen capture shows expected results for Output # 1.

```
(kali㉿kali)-[~/steghide]
└─$ ls
pexels-photo-1632790.jpeg  secret_msg.txt

(kali㉿kali)-[~/steghide]
└─$ rm secret_msg.txt

(kali㉿kali)-[~/steghide]
└─$ steghide extract -sf pexels-photo-1632790.jpeg
Enter passphrase:
wrote extracted data to "secret_msg.txt".

(kali㉿kali)-[~/steghide]
└─$ ls -l
total 1036
-rw-r--r-- 1 kali kali 1053616 Sep 25 23:22 pexels-photo-1632790.jpeg
-rw-r--r-- 1 kali kali      40 Sep 25 23:22 secret_msg.txt

(kali㉿kali)-[~/steghide]
└─$ cat secret_msg.txt
Name: Ali Hassan
Code: D8I$dsf#sd49uir5
```

Answer the following questions and write your response in **Output # 2**.

- Which encryption method is used in this example? (1 marks)
- Observe your original image and compare it with the one that has embedded hidden message. Can you spot any difference? (1 marks)
- Perform online research to find out how safe steganography is. Can it be easily detected. (2 marks)

## (B) Hashcat on Kali Linux (5 marks)

Use this link to <https://emn178.github.io/online-tools/md5.html>

Let us choose a 7-character password. The password should be of this format:

**Info896**

For this example, Let's choose the course code for this class and restrict it to just the first seven characters. First an upper-case character, followed by three lower-case characters and then three numeric digits. The reason behind this is, this is a very popular method of creating passwords where the requirements are to have at least one capital letter and numbers. People tend to make the first letter capital and then append numbers at the end. Reflect on your passwords to see if this is true or not.

---

If we can specify a part of the password as known value, the number of tries in the Brute force method goes down and it takes much less time in comparison to a pure brute force method. Now input this value (Info896) in the input field and copy the hash value. It should be Exactly like this:

```
909e0e6f01d9aa5b7be1be1606d65251
```

Open a terminal window on Kali Linux and save this hash value in a file named “**hashtest**” using the following command.

```
echo "909e0e6f01d9aa5b7be1be1606d65251" > hashtest
```

Run any of the following command(s) to see documentation of **hashcat** tool, which is an advanced CPU-based password recovery utility.

```
hashcat --help  
man hashcat
```

You will get a long list of options, but we are only interested in the following:

\* **General:**

```
-m, --hash-type=NUM Hash-type, see references below  
-a, --attack-mode=NUM Attack-mode, see references below
```

\* **Attack modes:**

```
3 = Brute-force
```

\* **Hash types:**

```
0 = MD5
```

Based on these options, the start of our command looks like the following, where MD5 (-m 0) and brute-force attack mode (-a 3) are specified. These options limit the search space for brute force search. (**Do not execute the command yet**).

```
hashcat -m 0 -a 3
```

Run **hashcat --help** again to see the documentation and note the built-in charsets, which will look like the following:

\* **Built-in charsets:**

?	Charset
l	abcdefghijklmnopqrstuvwxyz [a-z]
u	ABCDEFGHIJKLMNOPQRSTUVWXYZ [A-Z]
d	0123456789 [0-9]
h	0123456789abcdef [0-9a-f]
H	0123456789ABCDEF [0-9A-F]
s	!"#\$%&'()*+,./:;?=:@[\]^_`{ }~
a	?l?u?d?s
b	0x00 - 0xff

---

We can use these built-in charsets to specify a custom charset as follows:

```
-1 ?u -2 ?u?l?d -3 ?d
```

Here, three different character sets are defined. The “?” followed by either “l”, “u”, “d” characters are used to access the built-in charsets available (see “hashcat --help” output above for more details).

This implies our custom charsets -1, -2 and -3 are equal to:

Charset	Characters contained in the Charset
-1	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
-2	a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9
-3	0 1 2 3 4 5 6 7 8 9

After defining the three charsets, we need to specify which one to use for each character of the password as follows:

```
?1?2?2?2?3?3?3
```

In this pattern, ?1 indicates that the first character must be selected from the first character set, which consists of uppercase letters. It is followed by ?2?2?2, specifying that the next three characters can be a combination of uppercase letters, lowercase letters, and digits. Finally, the last character belongs to dataset 3, which includes digits only.

Putting this all together, the full command to execute is as follows:

```
hashcat -m 0 -a 3 -1 ?u -2 ?1?u?d -3 ?d hashtest ?1?2?2?2?3?3?3 --force
```

The term "hashtest" refers to the file created earlier, which contains the hashed password. After some processing time, Hashcat will crack the hashed password and generate output (as shown below) that reveals the password. The processing time can vary depending on the processing power of the machine.

```
909e0e6f01d9aa5b7be1be1606d65251:Info896
Session.....: hashcat
Status.....: Cracked
Hash.Mode....: 0 (MD5)
Hash.Target....: 909e0e6f01d9aa5b7be1be1606d65251
Time.Started....: Wed Sep 25 23:40:22 2024, (1 min, 45 secs)
Time.Estimated ...: Wed Sep 25 23:42:07 2024, (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Mask.....: ?1?2?2?2?3?3?3 [7]
Guess.Charset....: -1 ?u, -2 ?l?u?d, -3 ?d, -4 Undefined
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 33470.3 kH/s (7.06ms) @ Accel:256 Loops:1024 Thr:1 Vec:4
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 3791659008/6196528000 (61.19%)
Rejected.....: 0/3791659008 (0.00%)
Restore.Point....: 37888/62000 (61.11%)
Restore.Sub.#1 ...: Salt:0 Amplifier:18432-19456 Iteration:0-1024
Candidate.Engine.: Device Generator
Candidates.#1....: UIOr205 → DIVc996
Hardware.Mon.#1 ..: Util:100%

Started: Wed Sep 25 23:40:05 2024
Stopped: Wed Sep 25 23:42:08 2024
```



Execute the following command to view results of the previous operations.

```
cat ~/.hashcat/hashcat.potfile
```

Use the following command to find hashcat.potfile:

```
sudo find / -name "hashcat.potfile"
```

Create a hash value for your **custom password** using the webservice mentioned above. Your seven-character password should be of the following pattern:

```
[Uppercase_First_character_of_Your_FullName +
Lowercase_Next_ThreeCharacters_Of_Your_FullName +
Last_Three_Digits_Of_Your_StudentID]
```

For Example: My full name is Ali Hassan and if my student ID is 123, then my custom password will be Alih123.

Use hashcat tool to crack your custom hashed password and paste the screen capture in **Output # 3** (4 marks).

```
ATTENTION! Pure (unoptimized) backend kernels selected.  
Pure kernels can crack longer passwords, but drastically reduce performance.  
If you want to switch to optimized kernels, append -O to your commandline.  
See the above message to find out about the exact limits.
```

```
Watchdog: Temperature abort trigger set to 90c
```

```
Host memory required for this attack: 0 MB
```

```
c2e701af04ebc3e886ea3826c3473216:Alih123
```

```
Session.....: hashcat  
Status.....: Cracked  
Hash.Mode....: 0 (MD5)  
Hash.Target....: c2e701af04ebc3e886ea3826c3473216  
Time.Started....: Wed Sep 25 23:53:13 2024, (1 sec)  
Time.Estimated ... : Wed Sep 25 23:53:14 2024, (0 secs)  
Kernel.Feature ... : Pure Kernel  
Guess.Mask.....: ?1?2??2?3?3?3 [7]  
Guess.Charset....: -1 ?u, -2 ?l?u?d, -3 ?d, -4 Undefined  
Guess.Queue.....: 1/1 (100.00%)  
Speed.#1.....: 37109.9 kH/s (10.43ms) @ Accel:384 Loops:1024 Thr:1 Vec:4  
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)  
Progress.....: 40344576/6196528000 (0.65%)  
Rejected.....: 0/40344576 (0.00%)  
Restore.Point....: 384/62000 (0.62%)  
Restore.Sub.#1 ... : Salt:0 Amplifier:4096-5120 Iteration:0-1024  
Candidate.Engine.: Device Generator  
Candidates.#1....: NPrd888 → Q22b012  
Hardware.Mon.#1..: Util:100%
```

```
Started: Wed Sep 25 23:53:11 2024
```

```
Stopped: Wed Sep 25 23:53:15 2024
```

**Figure: hashcat tool to crack hashed password**

### (C) Submission Check List

	To Do	Done
1	Please use the provided <b>Lab report template</b> to complete attempt. Make sure the form in the cover page is filled up.	
2	Provide all the required outputs.	
3	Give appropriate title or captions (small description) to all the output boxes. (5% marks per output will be deducted, if they are not properly captioned)	
4	Submit your report as instructed.	