**Steganography:** The word Steganography is derived from two Greek words- ‘stegos’ meaning ‘to cover’ and ‘grayfia’, meaning ‘writing’, thus translating to ‘covered writing’, or ‘hidden writing’. Steganography is a method of hiding secret data, by embedding it into an audio, video, image or text file. It is one of the methods employed to protect secret or sensitive data from malicious attacks.

**OpenStego provides two main functionalities:**

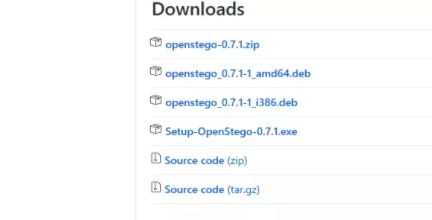
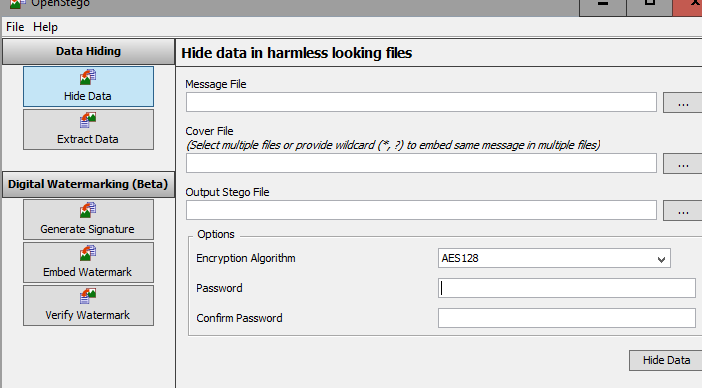
* **Data Hiding:** It can hide any data within a cover file (e.g. images).
* **Watermarking (beta):** Watermarking files (e.g. images) with an invisible signature. It can be used to detect unauthorized file copying.

**steganography \_medium = hidden message + carrier + steganography key**

In this experiment, we use OpenStego tool to hide a file inside an image

### Download OpenStego

Open your browser and navigate to the [OpenStego website](https://www.openstego.com/" \t "_blank). Click 'Downloads' at the top right. Scroll down to find the right version of OpenStego for your processor e.g. 'Setup-OpenStego-0.7.1.exe'.



**Note**: **OpenStego is written in Java so you must install the**[**JRE (Java Runtime Environment)**](http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html%20)**to load it.**

### Data Hiding

In this mode, we can either hide the data (file) inside an image or extract the data from the image. Check screenshots below to see how it can be done:

#### Steps to Hide data

**Step 1: Browse for the image file which we want to choose as the message file**

**Message file:**  We need to create the secret file which we want to remain hidden. There's no restriction on the format of 'message' files – this can be a text document or another image, however it has to be smaller than the cover file it hides inside.

**Step 2: Now choose and browse the cover file to embed the data**

**Cover file:** The cover file should be a harmless looking image, inside which the 'message' file will be stored. Make sure it's bigger than the 'message' file.

**Step 3: Provide the path for the output stego file**

**Step 4: Now choose the encryption algorithm:**

* **AES 128**
* **AES 256**

**Here , I will be use AES 128**

From here we can choose how to protect your data. The default setting (AES128) is perfectly secure, however we can increase the encryption strength to AES 256 (but it requires more processing)

**Step 5: Provide the password to hide the image: I have used the password: 1234 and confirm the password**

From here you can choose to encrypt our message file with a password before hiding it inside the cover file. Without the correct password, the data will be irretrievable

**Step 6: Click on hide data**

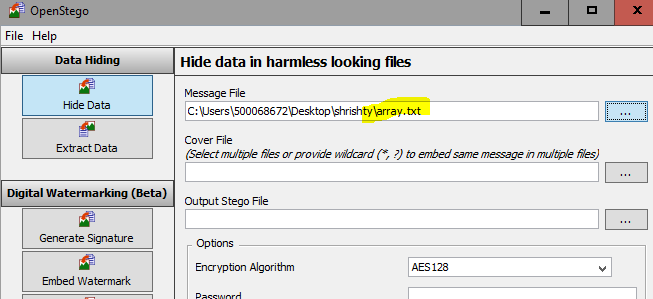
Click the ‘Hide Data’ button at the bottom-right of the screen to begin the process of concealing your message file inside the cover file. OpenStego will state it's processing the file, then display a message saying that the message has been embedded. Click ‘OK’.

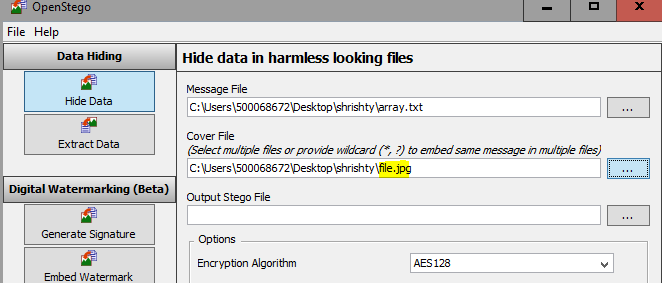
**Note: Next, make sure to erase the original 'message' and 'cover' file securely. This way there'll be no way to retrieve your secret file without using OpenStego and the password if we set one.**

**hide some text file in an image file.**

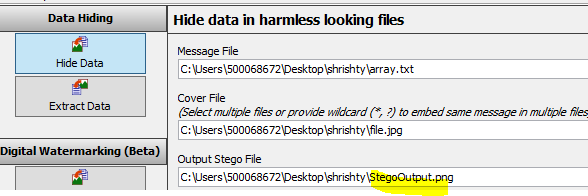
**I will follow the above mentioned 6 steps to hide the data:**

Step1 : The message file is array.txt

Step2: The cover file is an image file : **file.jpg**

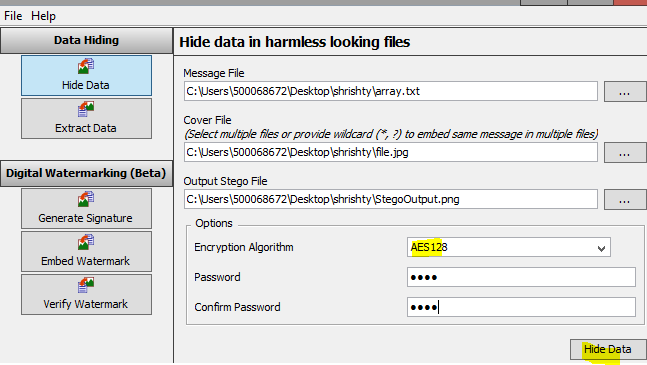


Step3 : To provide the output stego file:**StegoOutput.png**

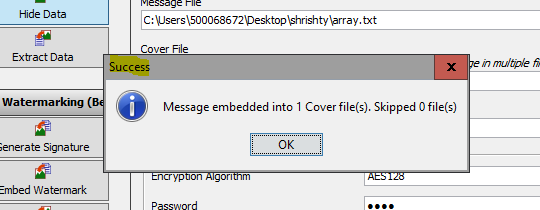


Step 4,5,6: choose the encryption algorithm to hide the data:

Here ,**AES 128** is used and then password to encrypt the data is provided. **My password for hiding the text file is : 1234** and then click on hide data.



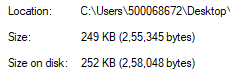
Now , this success msg pop-up confirms that our message file is embedded into the cover file.

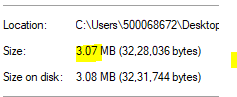


File.png StegoOutput.png

**hiding text or lower resolution image file, the output file’s size increases slightly.**

Before After



**Extracting the data**

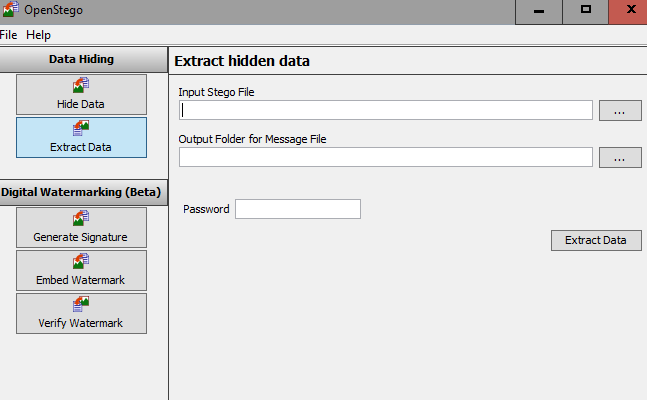
**Steps to extract the hidden data :**

**Step 1: Browse the location of the input stego file**

**Step 2; Choose an output folder for the message file**

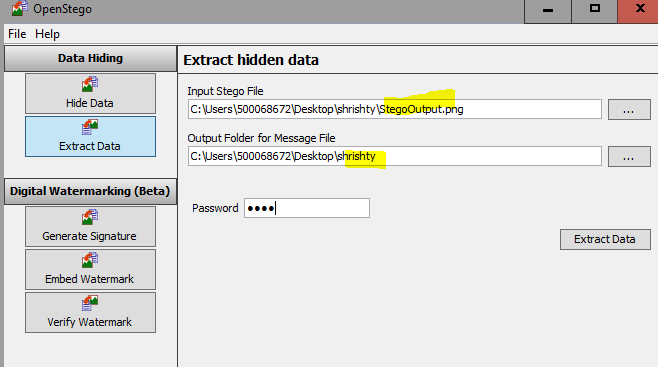
**Step 3: Provide the same password which we used while hiding the data**

**Step 4: Click on extract data.**

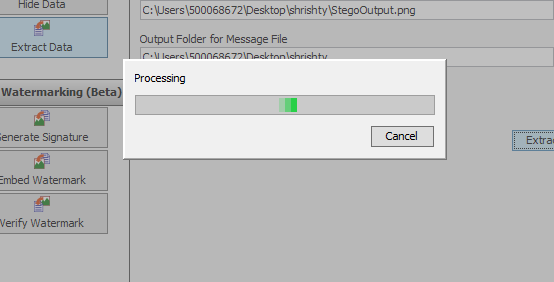


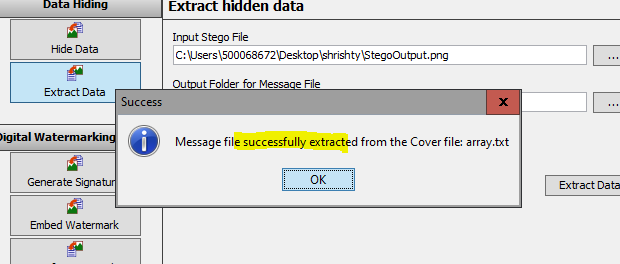
**Now lets try to extract the text file which we have hidden:**

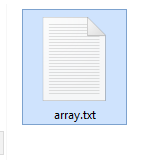
* In the **‘**Input Stego File**’** field, we enter the path of the output stego.
* In the **‘**Output Folder of Message File**’** field, we enter the folder name where we want to see our message file.
* Then provide the path of the output folder ,where we want the message file
* Now provide the password which we used to hide the data: 1234
* Click on extract data



It processes the file.

]



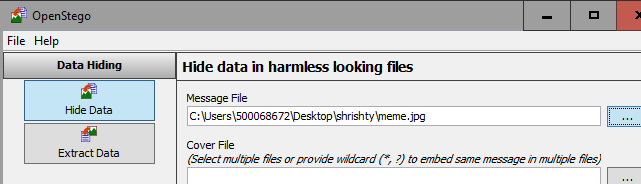
 **The extracted file:**

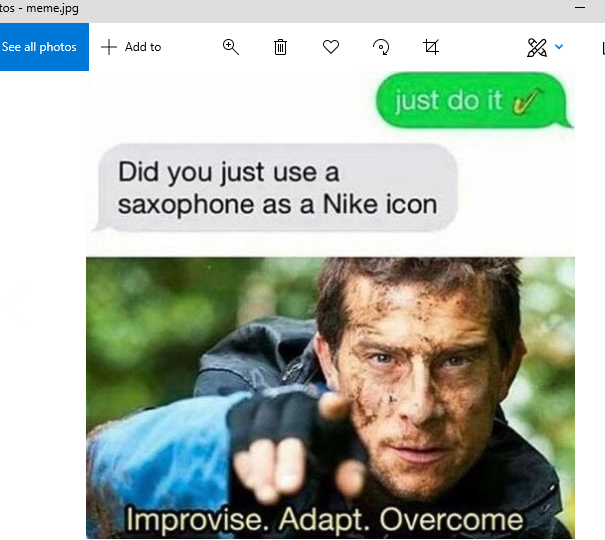
**Let’s try hiding some different files in different extensions.**

Hiding a image file in another image

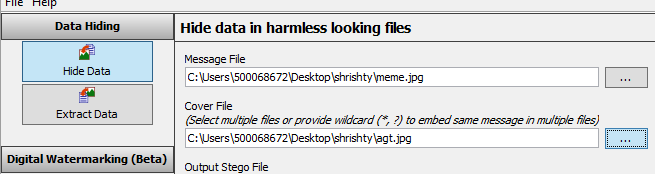
Follow the steps for hiding the data as mentioned earlier.

**Message file:meme.jpg**



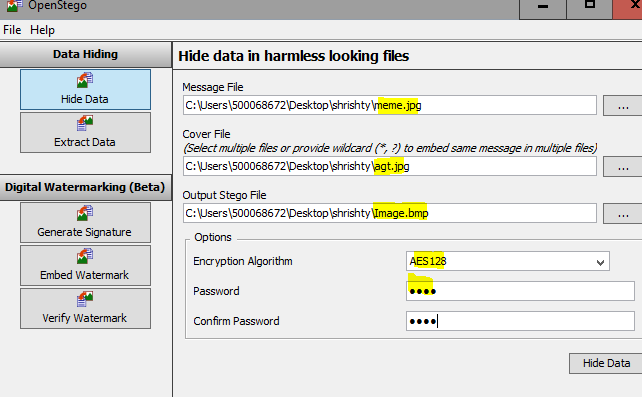


**The cover file:agt.png**



’

Provide the path for the output stego file and choose the algorithm and then provide the password

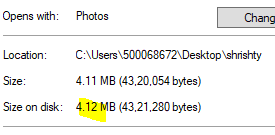
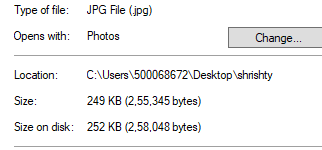


If we open the file, we can see that the image is the same as the cover media and no change can be seen.

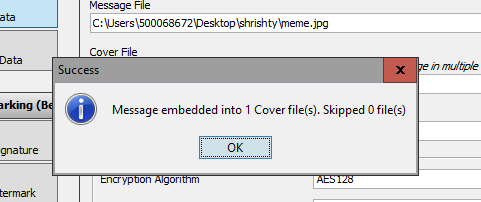


But checking the properties of the image file, the size of the file tells us that something is hidden :

BEFORE AFTER



We get the output message as ‘Message embedded’.



Now before trying to extract the data, first delete the image and the cover file

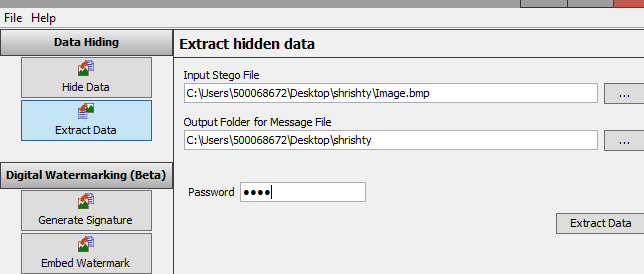
**Extracting data**

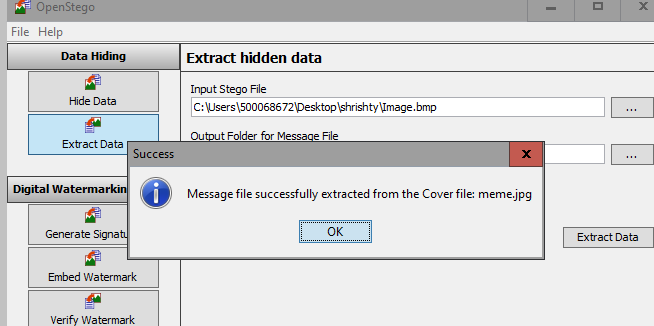
To retrieve the hidden message from the stego image, we can click on the ‘Extract Data’ tab from the left side of the window.

In the **‘**Input Stego File**’** field, we enter the path of the output stego.

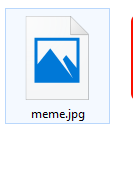
● In the **‘**Output Folder of Message File**’** field, we enter the folder name where we want to see our message file.

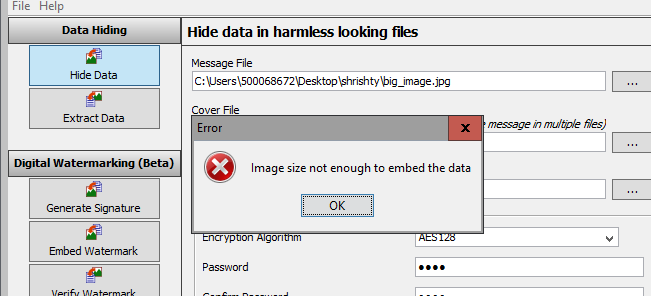
● Enter password (1234) and click on ‘Extract Data’.





**Extracted successfully**:

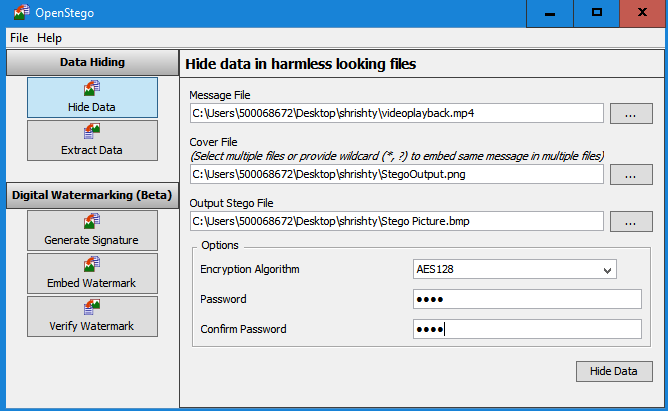
:

**If we try to hide file of size 10.2 MB inside an image of 17.6 kB, we get the error message: ‘Image size not enough to embed the data’** 

**Hide data using video file**

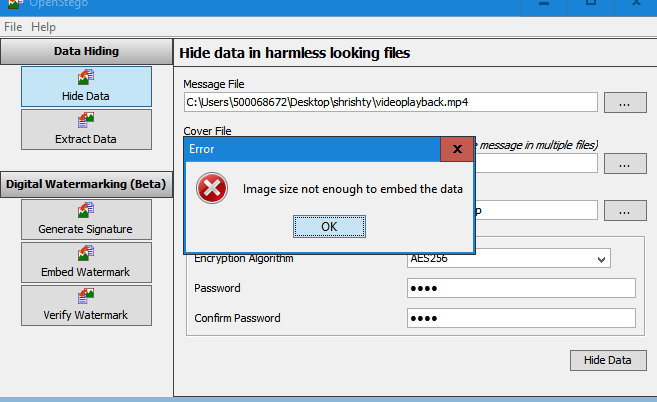
Message file : videoplayback.mp4

Cover file : StegoOutput.png



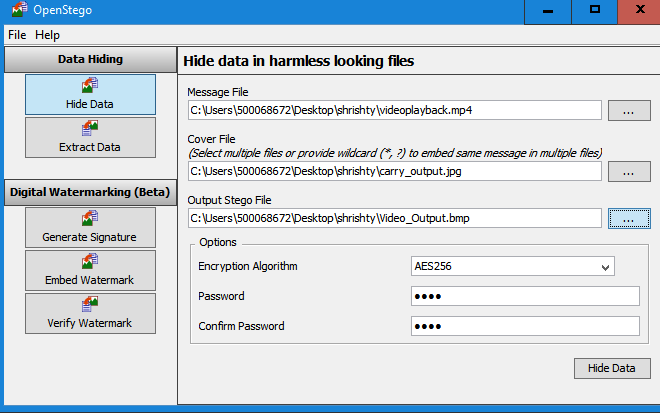
**We will see error because our video file is of larger size:**

**Image size wasn’t large enough to hide the mp3 file**

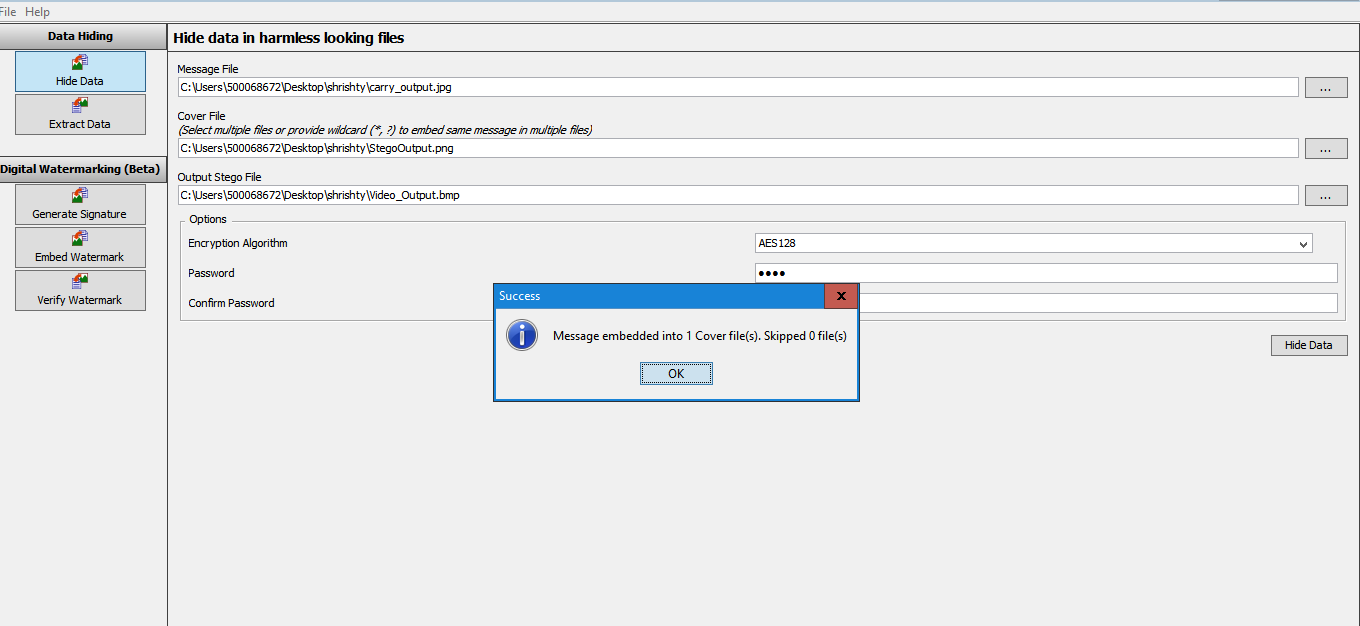


**Note**: **The cover file size needs to be larger than the message file. Large documents, mp4 ,mp3 files require larger cover files.**

**Let’s try using large size cover file**

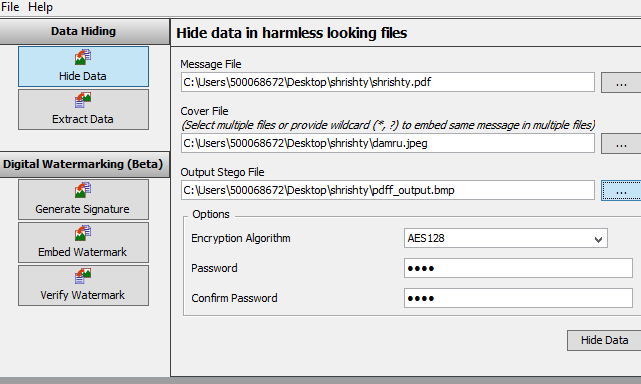


We get the output message as ‘Message embedded’



**Hiding a PDF in an image**.

**Hide the data**: Follow the steps 1-6 mentioned above to hide the data

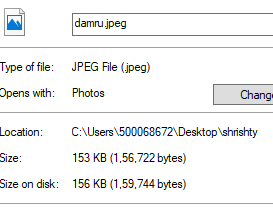
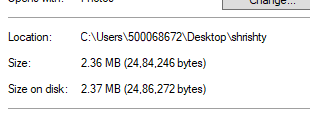


Visually the image is exactly the same.

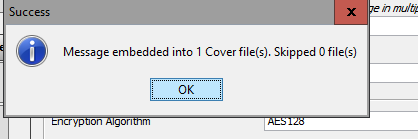
damru.jpeg pdff\_output.bmp

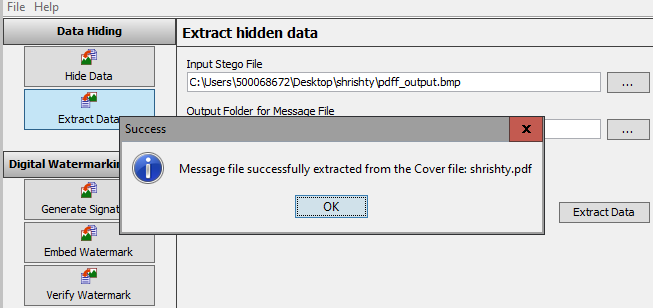
But if we check the file property and check the size of the file, the output file is larger then the cover file though they both look same from outside.

**Successfully embedded message will be displayed**.



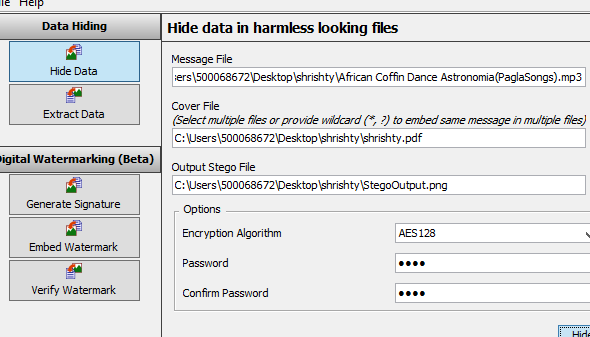
**Extract the hidden data** : Follow the steps(1-4) mentioned above to extract the data.

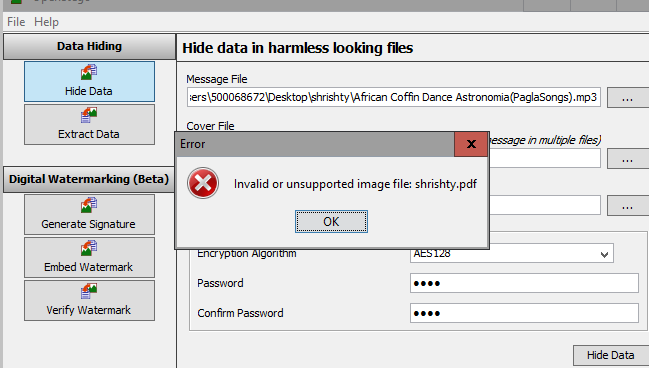


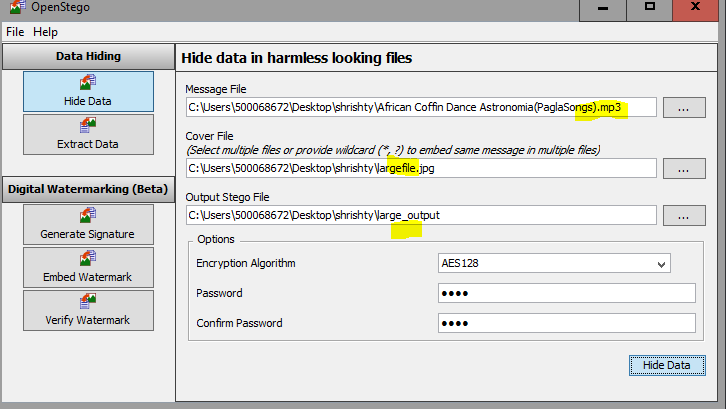


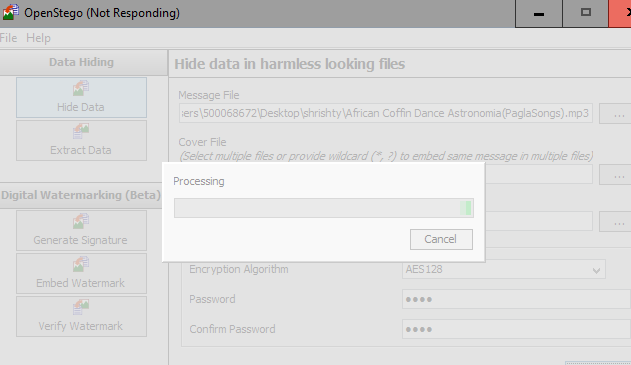
**Successfully we were able to hide pdf file in an image file and extract it later on using openstego.**

**Hide mp3 file in pdf then we will see an error:**

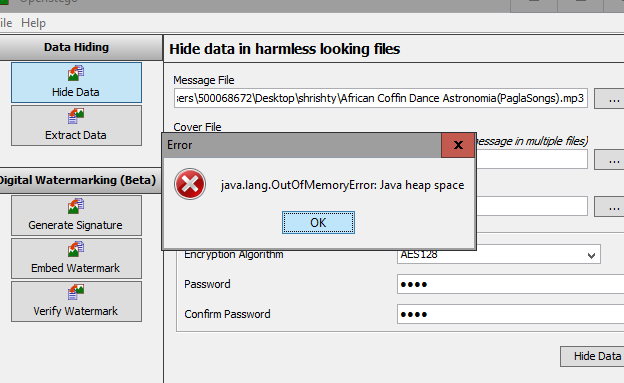
****

**This could be one of the limitation of openstego**

** Hide mp3 in a large image file:**

**NOTE: Hiding larger data requires more processing time.**

Even in a large enough file we couldn’t hide the file.



**Observation**

* There's no restriction on the format of 'message' files – this can be a text document or another image, however it has to be smaller than the cover file it hides inside.

**SIZE OF COVER FILE >= SIZE OF HIDDEN MESSAGE FILE**

* If our file is very large, OpenStego supports splitting it across a number of cover files.
* While extracting data, the size output file comes out to be equal to that of the hidden message file. This showcases the authenticity of the OpenStego tool.
* While hiding text or lower resolution image file, the output file’s size increases slightly.
* Make sure to erase the original 'message' and 'cover' file securely. This way there'll be no way to retrieve your secret file without using OpenStego and the password if we set one

**Limitations:**

* Even in a large enough file we couldn’t hide the file.(eg in mp3 file)
* We can’t hide large audio or video files in an image.
* The processing time was much higher. So the processing time increases as the file size increases.

**Exception:**

* If we try to hide file of size 10.2 MB inside an image of 17.6 kB, we get the error message: ‘Image size not enough to embed the data’