Nolan Cassidy & Joseph Gregory

Protecting Against Copyright Infringement Using Data-Labeling

Introduction:

Once files are digitized they can easily be shared with the world. Some images, audio, and movies are not meant to be shared with everyone. Matter of fact it can be devastating for people who confidential information or monetary losses. The uses of data labeling can help put a stop to the copyright infringement problem the internet has created.

Our idea is to incorporate watermarks into the image, audio, and videos. Not a normal visible watermark, instead we want to use a digital fingerprint that is hidden in the data of the file. This will allow the owner to customize each file to have the users data encoded in. While this is not apparent to the user except for a slight file increase, it will allow for the owner to determine who the original user was if a leak were to take place.

The way of encoding data in a hidden manner is already know as steganography. Steganography can be done many ways but all result in data being hidden in a file. LSB is a simple algorithm and the one we used in our tests to see if this is applicable to copyright infringement. It works by taking the least significant bit or bits and replaces them with the bits of the encoded message. Another method is called masking which uses more of a scatter approach. By changing whole areas of lighting barely it allows masking to make changes that are not noticeable to the human eye but also increase the file size much more drastically. But the data is more ingrained into the file making it harder for the message to be tampered with or lost. The last way is to perform a transformation which is when you perform some sort of math or just add the data to the end. This pattern is applied to the file and leads to less of a size increase but harder to pull off.

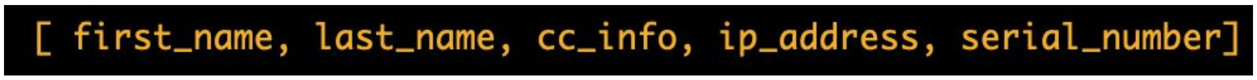
Overall a mix of steganography and cryptography is an easy fix for copyright infringement. While some methods may be harder to make, affect the size differently, or change the original obviously they all lead to helping track back the file to the original owner. The choice of algorithm is truly based of the needs of the owner. Are you more interested in security, then use transforming which is the hardest to tamper with but may increase the size dramatically. But if you are a musician you wouldn't want you files largened so a LSB would be a more beneficial choice.

Below we discuss how it is possible to collect data on a user. Then show and experiment with tools created using python to encode our metadata into the file. At the end we make sure the data was able to be decoded and compare how much size and physical changes were made. This is made to show that any one or business can incorporate this into their security and products. With more time we would have loved to dive into more algorithms and types of files like videos and text. Data labeling is showing to be extremely valuable in protecting assets in a way never thought of imagined.

Metadata:

In this day in age, everything is purchased or shared virtually online. Files are one of the most commonly shared file types among users. More specifically, artists distribute their files on multiple outlets online for users to experience. Platforms such as Soundcloud and iTunes deny the ability to download songs for free. By allowing files to be obtained without a purchase, the artist receives no money for the work they produced.

Once a purchase has been made, the user now owns the song and can listen as they please. This purchase agreement does not allow the user to distribute or resell the file. Still, this does not stop the traction of illegal online distribution. By inserting specific metadata to each audio file in a unique way, this can help trace back the origin of the initial leak of the file.
One way to hide the data is by taking a list of strings that will be hashed:



```
[ first_name, last_name, cc_info, ip_address, serial_number]
```

This information is stored only on the client side of the purchase to prevent possible data breaches on the server side. The information is then hashed, stored with the correct serial number and sent into the database that is only accessed by the correct hashed serial number. The hashing will make sure if anyone does extract the number from the file, its information will still not be able to be decoded.

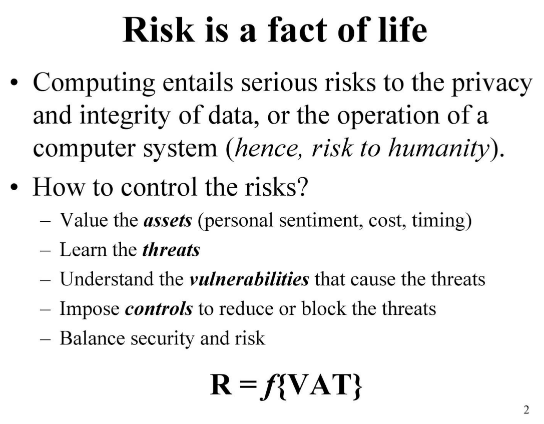
Images & Documents:

Images and Documents are shared on a daily basis online. Many of these files are not meant to be shared with the world and it can be devastating if they are leaked. Some examples are; A teacher or writer having a textbook/writings leaked. A photographer or artist having their work posted online. Or even a photo shared with friends on a private social media site. In the end of the day if something gets out that is not meant to, it can cost the owner immense amounts of money and trouble.

Using the following python source online we were able to begin testing ways of encoding and decoding information into the files. Link for Image Steganography:
https://github.com/raffg/steganography

For the purposes of encoding data we are exploring the comparison of the file and the hidden message to ensure it has not changed much. Here are the results using the LSB algorithm.

For our first example let us say a teacher wants to make each file unique to each individual student. Here is an unmodified slide from class on the left. On the right is the slide with the initials and id of the student, in this case “NC951555049”.

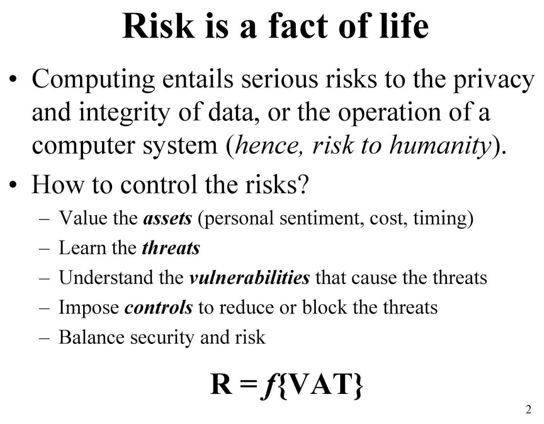


Risk is a fact of life

- Computing entails serious risks to the privacy and integrity of data, or the operation of a computer system (*hence, risk to humanity*).
- How to control the risks?
  - Value the ***assets*** (personal sentiment, cost, timing)
  - Learn the ***threats***
  - Understand the ***vulnerabilities*** that cause the threats
  - Impose ***controls*** to reduce or block the threats
  - Balance security and risk

R = *f*{VAT}

2



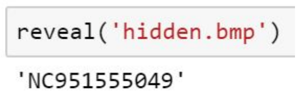
Risk is a fact of life

- Computing entails serious risks to the privacy and integrity of data, or the operation of a computer system (*hence, risk to humanity*).
- How to control the risks?
  - Value the ***assets*** (personal sentiment, cost, timing)
  - Learn the ***threats***
  - Understand the ***vulnerabilities*** that cause the threats
  - Impose ***controls*** to reduce or block the threats
  - Balance security and risk

R = *f*{VAT}

2

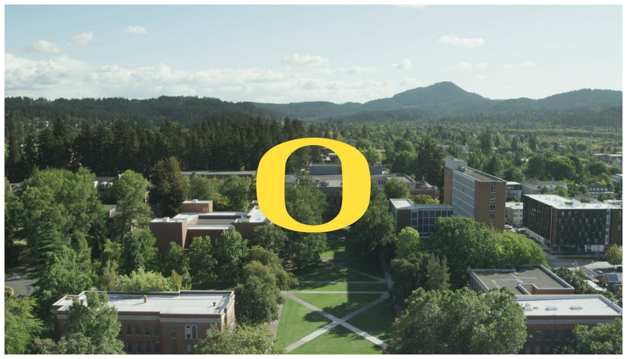
When decoded the information is still able to be retrieved.



```
reveal('hidden.bmp')
'NC951555049'
```

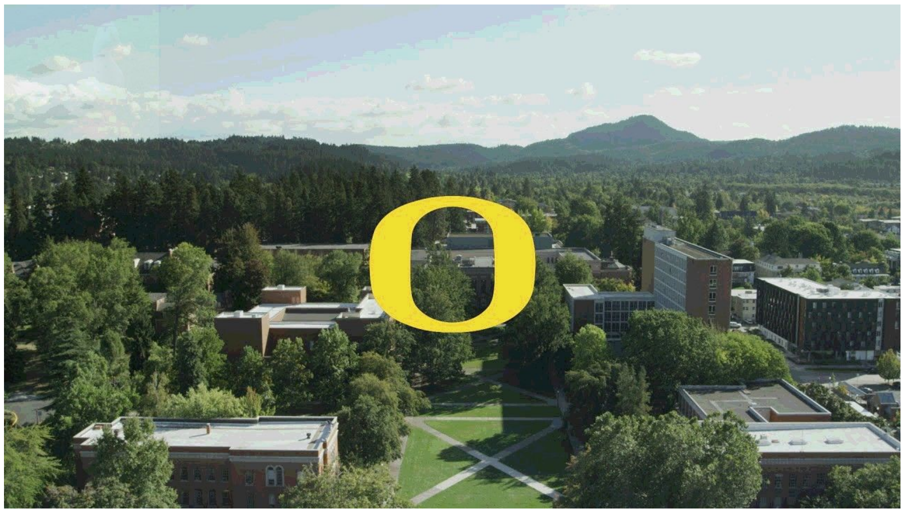
The file size of the original was 379 KB and the modified version was 488 KB, that means our 11 digit string added 109 KB to the file but the visual elements are not noticeable. Many teachers express their concerns of having textbooks,slides, and classwork being leaked online. If each student’s copy was marked this would allow teachers to find the culprit.

The second example we wanted to try encoding a larger message into the image. For something sold online like and image or artwork, a plethora of information can be collected. This can include but not limited to; names, social, ip address, email, credit card, purchase number, date, age, location data, etc. The list is endless so the meta data being stored could possible grow to something very large. The next example shows the original on the left and the modified image on the right which contains 1000 character long serial number.



To our delight and surprise even with the longer string the file change was not noticeable to the human eye except for some dullness to the colors. Also once again the size of the file grew from 144 KB to 212 KB. Overall this means that this could also be a great application for photographers and artists who want to share their work but do not want it to leave the source.

The final example we wanted to go to the extreme and see some changes in our image so instead of text I encoded this image of the duck into the oregon image.



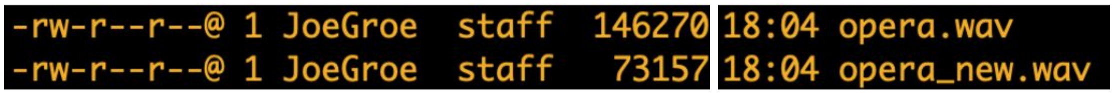
Believe it or not the photo of the duck is actually encoded into the image on the right. As you can see some of the image has been altered, for example the sky has discoloration of blues. But overall the image is still very much in tact. The final size of the modified image was also only 228 KB which is not a lot surprisingly.

Audio:

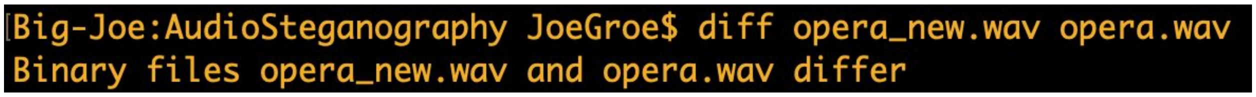
We found an open source audio steganography program created by Saprative Jana that takes in a desired WAV file and adds a message within the audio file without much distortion or manipulation of the original audio file. The program works by accepting a filename and the programs asks for a specific message to be stored within the file. Once stored, the programs outputs the audio file with a message hidden within it.

You can find the source code here: https://github.com/saprative/AudioSteganography

The one problem with hiding metadata within an audio file is the additional memory that needs to be allocated. Below is the output of two audio files and the conclusion that both files differ, despite containing the same audio.



```
-rw-r--r--@ 1 JoeGroe  staff  146270 18:04 opera.wav
-rw-r--r--@ 1 JoeGroe  staff   73157 18:04 opera_new.wav
```



```
[Big-Joe:AudioSteganography JoeGroe$ diff opera_new.wav opera.wav
Binary files opera_new.wav and opera.wav differ
```

opera.wav is the original input file with opera_new.wav as the output file. What is interesting here, is that the program uses the file's least significant bit to input the secret message. If the information that is being added to a file is small, it will actually decrease the size of the file. This allows for the file size to actually be less than the original one. However, if the information being stored is large, it will increase the size of the file.

Conclusion:

Throughout our research for this project, we have learned how viable steganography can be for storing important information that should not be seen by ordinary users. For the examples we tested, by replacing the least significant bits of the file it shows that it is possible to minimize storage usage while also creating extra security for an artists' work. Hiding metadata within files is not limited to only pictures and audio files but can also be performed in videos and text files.

Any file type can store data behind the walls of the product itself. Using the least significant bit method works well for most file types and is the easiest approach for anyone wanting to get into steganography. We also found that JPEG images are the most difficult to perform this method. The reason for it is that JPEG files are lossy compressions, which means that the bit information can be altered or manipulated easily which can lead to data loss within the file. So, choosing the proper file format is important when performing steganography.

The question still remains: is it impossible to stop people from illegally distributing files over the internet? No matter what if someone wants the can alter the file enough to get rid of any hidden information. But for anyone who was not aware the data is hidden then they would have no way of recognizing it. The answer is difficult to find, but with the help of steganography, people with a small background of programming can easily help protect their work with easy to use programs like the ones we used for the project. By adding data within their files, it can be helpful to track files to see how, when and where the file originated from.