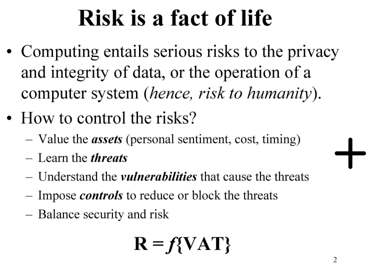
Copyright Infringement & Data-Labeling

Nolan Cassidy
Joseph Gregory

Data Labeling to fight against Copyright

- We wanted to stop people from commiting the terrible act of copyright infringement for artists, teachers, musicians, photographers, etc.
- Steganography:
    - After first encrypting a message with cryptography. We encode the data of a message into an original file such as an image
    - This modifies the file but in a way that is not visible to the user
    - Lot of algorithms to choose from, all of which affect the size and look of the original
- By making each file unique the owner can track back to the origin who created a leak



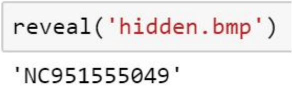
Risk is a fact of life

- 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}

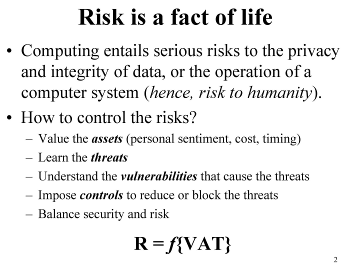
2

+



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

=



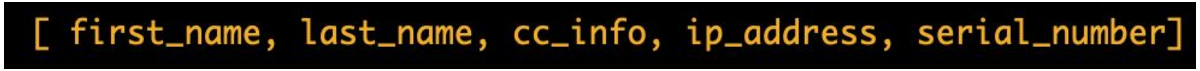
Risk is a fact of life

- 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

User Metadata

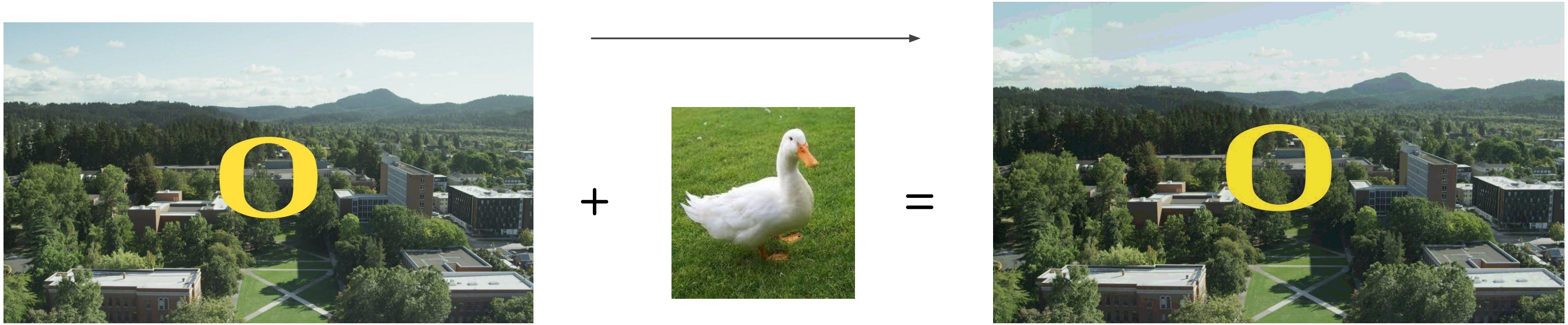


```
[ first_name, last_name, cc_info, ip_address, serial_number]
```

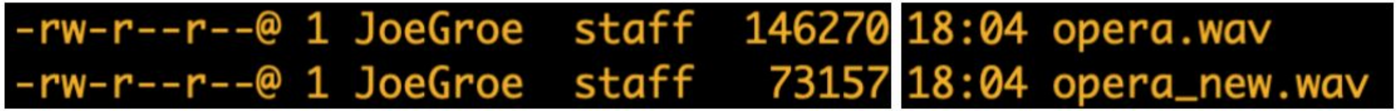
- Stored only on the client side of the purchase
- Prevents possible data breaches on the server side
- The information is hashed and stored with the correct serial number
- Information will not be decoded if anyone extracts the serial number

Image Data Labeling

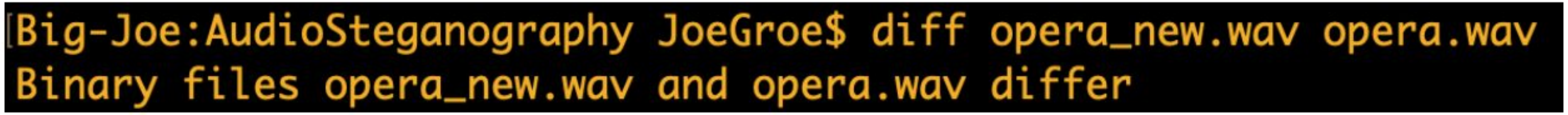
- Performed experiments on files, trying to encode different size messages into the the images
    - Tested string size 11, string size 1000, and a whole image
- To our surprise for all of the varying modifications the size of the file only grew by 100 KB and the outputs were barely changed except for when hiding a whole image inside of an image
- Here is our biggest test:



Audio Data Labeling



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

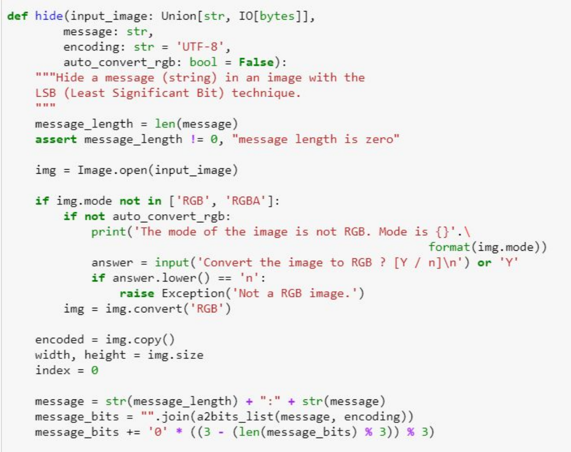


```
[Big-Joe:AudioSteganography JoeGroe$ diff opera_new.wav opera.wav
Binary files opera_new.wav and opera.wav differ
```

- Smaller output file
- Differ in bits
- Little to no alteration in audio

Conclusion

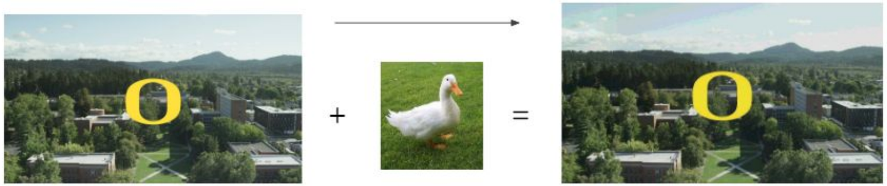
- The question still remains: is it impossible to stop people from illegally distributing files over the internet?
- No matter what if someone wants the can alter the file enough to get rid of any hidden information.
- But for anyone who was not aware the data is hidden then they would have no way of 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.



```
def hide(input_image: Union[str, IO[bytes]],
         message: str,
         encoding: str = 'UTF-8',
         auto_convert_rgb: bool = False):
    """Hide a message (string) in an image with the
    LSB (Least Significant Bit) technique.
    """
    message_length = len(message)
    assert message_length != 0, "message length is zero"

    img = Image.open(input_image)

    if img.mode not in ['RGB', 'RGBA']:
        if not auto_convert_rgb:
            print('The mode of the image is not RGB. Mode is {}'.\
                                                             format(img.mode))
            answer = input('Convert the image to RGB ? [Y / n]\n') or 'Y'
            if answer.lower() == 'n':
                raise Exception('Not a RGB image.')
        img = img.convert('RGB')

    encoded = img.copy()
    width, height = img.size
    index = 0

    message = str(message_length) + ":" + str(message)
    message_bits = "".join(a2bits_list(message, encoding))
    message_bits += '0' * ((3 - (len(message_bits) % 3)) % 3)
```

Protecting Against Copyright Infringement Using Data-Labeling

Nolan Cassidy & Joseph Gregory



The Problem:

Images, Text, Audio, Images and much more is shared online against the permission of the owner.

The Solution:

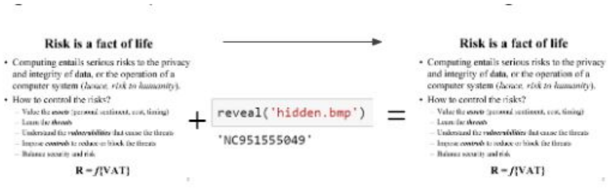
Use Data-labeling to hide metadata of the person it belongs to. If leaked the data can be extracted to determine the culprit.

Method:

First we will encrypt the metadata to ensure it cannot be read if found. Then we use steganography to place the data into the file hiddenly.

Image Example:

This example shows how a teacher could store a students id into a file. This way if it is share online or with classmates it can be traced back.



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)

+

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

Audio:

Tested storing data within the audio file to compare the file sizes. Promising results showed that this can work with minimal distortion and possible optimization of storage space.

Steganography:

Using Least Significant Bit method (LSB) we were able to store this data hiddenly in the files. Other algorithms exist. With our research we learned they vary in ease, size change, and physical change. Choosing which is best is a case by case scenario.

Meta Data:

Hashing a list containing certain details about the user, hashing the information and sending it stored with the serial number on the client side can prevent possible data breach on the server side. Can only be accessed with hashed serial number.e.g. [first, last, cc_info, serial_num]