Data Privacy Sharing - Practical Algorithms

Contents

(T3) Digital Watermarking

A digital watermark is a kind of marker covertly embedded in a noise-tolerant signal such as audio, video or image data. It is typically used to identify ownership of the copyright of such signal.

"Watermarking" is the process of hiding digital information in a carrier signal; the hidden information should, but does not need to, contain a relation to the carrier signal. Digital watermarks may be used to verify the authenticity or integrity of the carrier signal or to show the identity of its owners. It is prominently used for tracing copyright infringements and for banknote authentication.

Traditional Watermark

Drawing

Unlike traditional watermarking which makes the watermark obvious for all viewers, digital watermarking is usually hidden from plain view and can only be accessed by a certain algorithm to extract the watermark data.

One of the methods of digital watermarking is **steganography**, which is the practice of concealing a file, message, image, or video within another file, message, image, or video.

Steganography

Simple steganography works by storing the hidden information in the data that does not significantly change how the original file looks.

Some of the simple steganographic methods are:

- **Text Data**:
  - White space manipulation: adding of white spaces before a newline character to encode hidden information
- **Numeric Data**:
  - Least significant decimal: altering least significant decimal places to store hidden information
- **Image Data**:
  - Least significant bits: altering the least significant bits of each pixel to store hidden information

Steganography - Text in text using white space manipulation

In this example, we encode our secret message into white spaces and then hide it after the first paragraph

In [1]:
```python
# package for encoding
import codext

# open lorem ipsum file
with open('lorem_ipsum.txt') as f:
    document_text = f.read()

print('Text from file:\n')
print(document_text)
```

```
Text from file:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incidid
unt ut labore et dolore magna aliqua. Pellentesque dignissim enim sit amet. Quam adipi
scing vitae proin sagittis nisl rhoncus mattis. Arcu risus quis varius quam quisque id
diam vel quam. Turpis cursus in hac habitasse. Sodales ut etiam sit amet. Est velit eg
estas dui id ornare arcu odio ut. Volutpat maecenas volutpat blandit aliquam etiam era
t velit scelerisque in. Magna sit amet purus gravida quis blandit turpis cursus. Sit a
met massa vitae tortor.

Nisl tincidunt eget nullam non nisi. Amet consectetur adipiscing elit ut. Mauris nunc
congue nisi vitae suscipit. Ullamcorper morbi tincidunt ornare massa eget egestas. Mau
ris nunc congue nisi vitae suscipit tellus mauris a diam. Tortor at risus viverra adip
iscing at in tellus integer feugiat. Mauris ultrices eros in cursus turpis massa tinci
dunt dui ut. Dui vivamus arcu felis bibendum ut tristique et egestas. Eget aliquet nib
h praesent tristique magna. Tincidunt praesent semper feugiat nibh sed pulvinar. Condi
mentum id venenatis a condimentum vitae sapien pellentesque habitant morbi. Nulla at v
olutpat diam ut venenatis tellus. Amet commodo nulla facilisi nullam. A diam maecenas
sed enim ut. Erat nam at lectus urna duis convallis convallis tellus. Integer eget ali
quet nibh praesent tristique magna sit amet. Tincidunt lobortis feugiat vivamus at. No
n nisi est sit amet facilisis magna etiam tempor. Orci nulla pellentesque dignissim en
im sit.

Maecenas volutpat blandit aliquam etiam erat velit. Euismod in pellentesque massa plac
erat duis ultricies lacus sed. Placerat duis ultricies lacus sed turpis tincidunt id a
liquet. Lectus nulla at volutpat diam ut venenatis. A lacus vestibulum sed arcu non od
io. Nulla pharetra diam sit amet nisl suscipit adipiscing bibendum est. Faucibus ornar
e suspendisse sed nisi lacus sed. Massa tempor nec feugiat nisl pretium fusce id veli
t. Mi proin sed libero enim sed. Et ultrices neque ornare aenean euismod elementum nis
i quis eleifend. Aliquet risus feugiat in ante metus. Turpis in eu mi bibendum neque.
Ipsum consequat nisl vel pretium lectus quam id leo. Adipiscing enim eu turpis egestas
pretium aenean. Arcu ac tortor dignissim convallis aenean et. Montes nascetur ridiculu
s mus mauris. Pretium fusce id velit ut tortor pretium viverra.

Erat imperdiet sed euismod nisi porta lorem mollis aliquam. Amet facilisis magna etiam
tempor orci eu lobortis elementum nibh. Enim neque volutpat ac tincidunt vitae semper
quis lectus nulla. Nulla at volutpat diam ut venenatis tellus in metus. Maecenas accum
san lacus vel facilisis volutpat est. Posuere ac ut consequat semper viverra nam liber
o. Risus feugiat in ante metus dictum at tempor commodo ullamcorper. Etiam tempor orci
eu lobortis elementum nibh tellus molestie nunc. Feugiat scelerisque varius morbi enim
nunc faucibus a pellentesque. Sed odio morbi quis commodo. Sit amet aliquam id diam ma
ecenas ultricies mi eget. Eu non diam phasellus vestibulum lorem sed risus ultricies.
Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Condimentum vitae sapi
en pellentesque habitant morbi tristique senectus. Fringilla ut morbi tincidunt augue.
Porttitor rhoncus dolor purus non enim praesent elementum facilisis leo. Tincidunt vit
ae semper quis lectus nulla at. A scelerisque purus semper eget duis at tellus at. Con
gue quisque egestas diam in arcu cursus euismod. Sit amet venenatis urna cursus.

Lacus viverra vitae congue eu consequat ac felis donec. Condimentum mattis pellentesqu
e id nibh tortor id aliquet lectus proin. Ut morbi tincidunt augue interdum. Pellentes
que elit ullamcorper dignissim cras tincidunt lobortis feugiat vivamus at. Quis eleife
nd quam adipiscing vitae proin sagittis nisl. Sit amet nulla facilisi morbi tempus iac
ulis. Libero enim sed faucibus turpis. Enim diam vulputate ut pharetra. Aliquam etiam
erat velit scelerisque in dictum non. Nunc eget lorem dolor sed viverra ipsum. Donec a
c odio tempor orci dapibus ultrices in iaculis nunc. Lorem mollis aliquam ut porttitor
leo a diam sollicitudin.
```

In [2]:
```python
secret_message = 'This is a secret!'

encoded_secret_message = codext.encode(secret_message, 'whitespace')

print(encoded_secret_message)
```

In [3]:
```python
def hide_text_in_text(cover_text:str, hidden_text:str) -> str:
    """Hides hidden_text in cover_text using white space manipulation steganography.
    Hidden text is placed at the end of the first paragraph

    Args:
        cover_text: Text to be shown to public
        hidden_text: Text to be hidden from public
    Return:
        hidden_text mixed into the cover text
    """

    first_newline_position = min(len(cover_text), cover_text.find('\n'))

    encoded_hidden_text = codext.encode(hidden_text, 'whitespace')

    return cover_text[:first_newline_position]+ encoded_hidden_text + cover_text[firs

def extract_text_in_text(mixed_text:str) -> str:
    """Extract hidden_text from cover_text using white space manipulation steganograp

    Args:
        mixed_text: Text mixed with hidden text

    Return:
        decoded hidden text
    """

    first_newline_position = min(len(mixed_text), mixed_text.find('\n'))

    hidden_text_start_position = len(mixed_text[:first_newline_position].rstrip())

    encoded_hidden_text = mixed_text[hidden_text_start_position:first_newline_positio


    return codext.decode(encoded_hidden_text, 'whitespace')
```

In [4]:
```python
mixed = hide_text_in_text(document_text, secret_message)

print(mixed)
```


```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incidid
unt ut labore et dolore magna aliqua. Pellentesque dignissim enim sit amet. Quam adipi
scing vitae proin sagittis nisl rhoncus mattis. Arcu risus quis varius quam quisque id
diam vel quam. Turpis cursus in hac habitasse. Sodales ut etiam sit amet. Est velit eg
estas dui id ornare arcu odio ut. Volutpat maecenas volutpat blandit aliquam etiam era
t velit scelerisque in. Magna sit amet purus gravida quis blandit turpis cursus. Sit a
met massa vitae tortor.

Nisl tincidunt eget nullam non nisi. Amet consectetur adipiscing elit ut. Mauris nunc
congue nisi vitae suscipit. Ullamcorper morbi tincidunt ornare massa eget egestas. Mau
ris nunc congue nisi vitae suscipit tellus mauris a diam. Tortor at risus viverra adip
iscing at in tellus integer feugiat. Mauris ultrices eros in cursus turpis massa tinci
dunt dui ut. Dui vivamus arcu felis bibendum ut tristique et egestas. Eget aliquet nib
h praesent tristique magna. Tincidunt praesent semper feugiat nibh sed pulvinar. Condi
mentum id venenatis a condimentum vitae sapien pellentesque habitant morbi. Nulla at v
olutpat diam ut venenatis tellus. Amet commodo nulla facilisi nullam. A diam maecenas
sed enim ut. Erat nam at lectus urna duis convallis convallis tellus. Integer eget ali
quet nibh praesent tristique magna sit amet. Tincidunt lobortis feugiat vivamus at. No
n nisi est sit amet facilisis magna etiam tempor. Orci nulla pellentesque dignissim en
im sit.

Maecenas volutpat blandit aliquam etiam erat velit. Euismod in pellentesque massa plac
erat duis ultricies lacus sed. Placerat duis ultricies lacus sed turpis tincidunt id a
liquet. Lectus nulla at volutpat diam ut venenatis. A lacus vestibulum sed arcu non od
io. Nulla pharetra diam sit amet nisl suscipit adipiscing bibendum est. Faucibus ornar
e suspendisse sed nisi lacus sed. Massa tempor nec feugiat nisl pretium fusce id veli
t. Mi proin sed libero enim sed. Et ultrices neque ornare aenean euismod elementum nis
i quis eleifend. Aliquet risus feugiat in ante metus. Turpis in eu mi bibendum neque.
Ipsum consequat nisl vel pretium lectus quam id leo. Adipiscing enim eu turpis egestas
pretium aenean. Arcu ac tortor dignissim convallis aenean et. Montes nascetur ridiculu
s mus mauris. Pretium fusce id velit ut tortor pretium viverra.

Erat imperdiet sed euismod nisi porta lorem mollis aliquam. Amet facilisis magna etiam
tempor orci eu lobortis elementum nibh. Enim neque volutpat ac tincidunt vitae semper
quis lectus nulla. Nulla at volutpat diam ut venenatis tellus in metus. Maecenas accum
san lacus vel facilisis volutpat est. Posuere ac ut consequat semper viverra nam liber
o. Risus feugiat in ante metus dictum at tempor commodo ullamcorper. Etiam tempor orci
eu lobortis elementum nibh tellus molestie nunc. Feugiat scelerisque varius morbi enim
nunc faucibus a pellentesque. Sed odio morbi quis commodo. Sit amet aliquam id diam ma
ecenas ultricies mi eget. Eu non diam phasellus vestibulum lorem sed risus ultricies.
Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Condimentum vitae sapi
en pellentesque habitant morbi tristique senectus. Fringilla ut morbi tincidunt augue.
Porttitor rhoncus dolor purus non enim praesent elementum facilisis leo. Tincidunt vit
ae semper quis lectus nulla at. A scelerisque purus semper eget duis at tellus at. Con
gue quisque egestas diam in arcu cursus euismod. Sit amet venenatis urna cursus.

Lacus viverra vitae congue eu consequat ac felis donec. Condimentum mattis pellentesqu
e id nibh tortor id aliquet lectus proin. Ut morbi tincidunt augue interdum. Pellentes
que elit ullamcorper dignissim cras tincidunt lobortis feugiat vivamus at. Quis eleife
nd quam adipiscing vitae proin sagittis nisl. Sit amet nulla facilisi morbi tempus iac
ulis. Libero enim sed faucibus turpis. Enim diam vulputate ut pharetra. Aliquam etiam
erat velit scelerisque in dictum non. Nunc eget lorem dolor sed viverra ipsum. Donec a
c odio tempor orci dapibus ultrices in iaculis nunc. Lorem mollis aliquam ut porttitor
leo a diam sollicitudin.
```


In [5]:
```python
print(extract_text_in_text(mixed))
```

```
This is a secret!
```

Steganography - Image in image using least significant bits algorithm

e.g. in the pixel 8-bit value 10110011 (of one of the RGB colors), the first 4 bits 1011 are the most significant bits, while the last 4 bits 0011 are the least significant bits. We can store hidden information in the last 4 bits while not compromising too much of how the overall image looks

In [6]:
```python
import numpy as np
import matplotlib.pyplot as plt
from skimage import io


def decimalToBinary(decimal_number:int) -> str:
    """Translate decimal to binary to see bit representation in str

    Args:
        decimal_number: integer represented in decimal numeric system

    Returns:
        decimal_number represented in binary numeric system as a string data type

    """
    return bin(decimal_number).replace("0b", "")


def binaryToDecimal(binary_string:str) -> int:
    """Translate binary back to decimal

    Args:
        binary_string: string of binary bits

    Returns:
        decimal representation of binary bits in integer data type
    """
    return int(binary_string, 2)
```

8-bit RGB color codes are usually represented in a group of 3 decimal integers (with range 0-255), for example the color Red is represented as (255, 0, 0) in decimal integer representation or (11111111, 00000000, 00000000) in binary bit representation.

Let's do a simple decimal to binary conversion to see how the decimal values look like in bits.

In [7]:
```python
print('{} in binary is {}'.format(255, decimalToBinary(255)))
print('{} in decimal is {}'.format('11111111', binaryToDecimal('11111111')))
```

```
255 in binary is 11111111
11111111 in decimal is 255
```

Let's take only the most significant bits (first 4 bits) and convert into decimal.

Let's also take the least significant bits (last 4 bits) and convert into decimal.

In [8]:
```python
print('{} in decimal is {}'.format('11110000', binaryToDecimal('11110000')))
print('{} in decimal is {}'.format('00001111', binaryToDecimal('00001111')))
```

```
11110000 in decimal is 240
00001111 in decimal is 15
```

Let's create a palette for quick access to colors to create a random bitmap image to see the effects of showing only the most significant bits and least significant bits.

We also create a palette each for the most significant bits and least significant bits version of the palette

In [9]:
```python
# create palette for quick access to colors
palette = np.array([[255,   0,   0], # index 0: red
                    [  0, 255,   0], # index 1: green
                    [  0,   0, 255], # index 2: blue
                    [255, 255, 255], # index 3: white
                    [  0,   0,   0], # index 4: black
                    [255, 255,   0], # index 5: yellow
                   ], dtype=np.uint8)

significant_bits_palette = (palette /255 * 240).astype(int)

non_significant_bits_palette = (palette /255 * 15).astype(int)

print('Palette:')
print(palette)
print('Most significant bits palette:')
print(significant_bits_palette)
print('Least significant bits palette:')
print(non_significant_bits_palette)
```

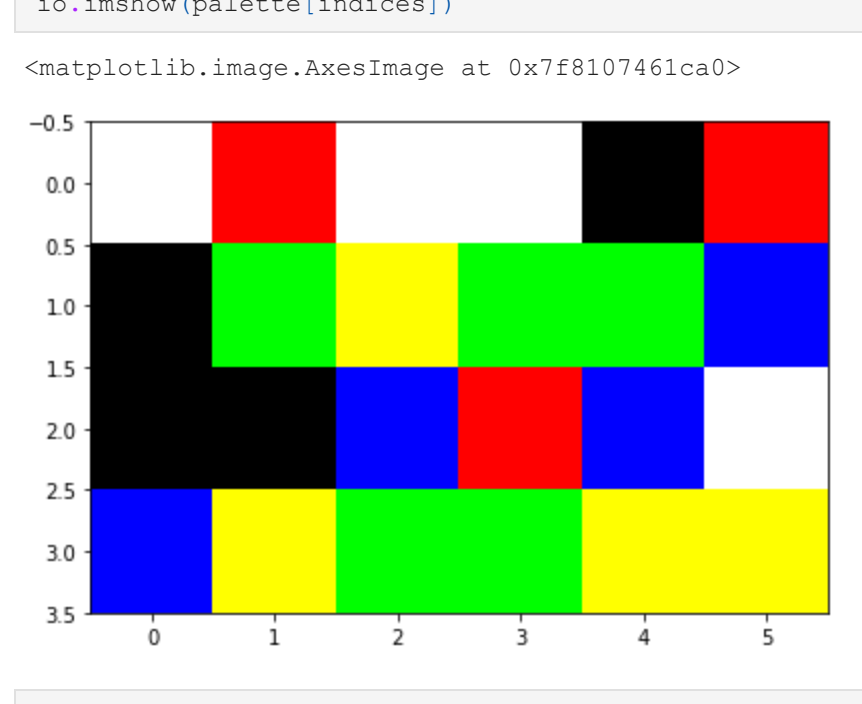
```
Palette:
[[255   0   0]
 [  0 255   0]
 [  0   0 255]
 [255 255 255]
 [  0   0   0]
 [255 255   0]]
Most significant bits palette:
[[240   0   0]
 [  0 240   0]
 [  0   0 240]
 [240 240 240]
 [  0   0   0]
 [240 240   0]]
Least significant bits palette:
[[15  0  0]
 [ 0 15  0]
 [ 0  0 15]
 [15 15 15]
 [ 0  0  0]
 [15 15  0]]
```

In [10]:
```python
# generate random array of colors using palette

m, n = 4, 6

indices = np.random.randint(0, len(palette), size=(m, n))

indices
```

Out[10]:
```
array([[3, 0, 3, 3, 4, 0],
       [4, 1, 5, 1, 1, 2],
       [4, 4, 2, 0, 2, 3],
       [2, 5, 1, 1, 5, 5]])
```

In [11]:
```python
# plot original color palette
io.imshow(palette[indices])
```

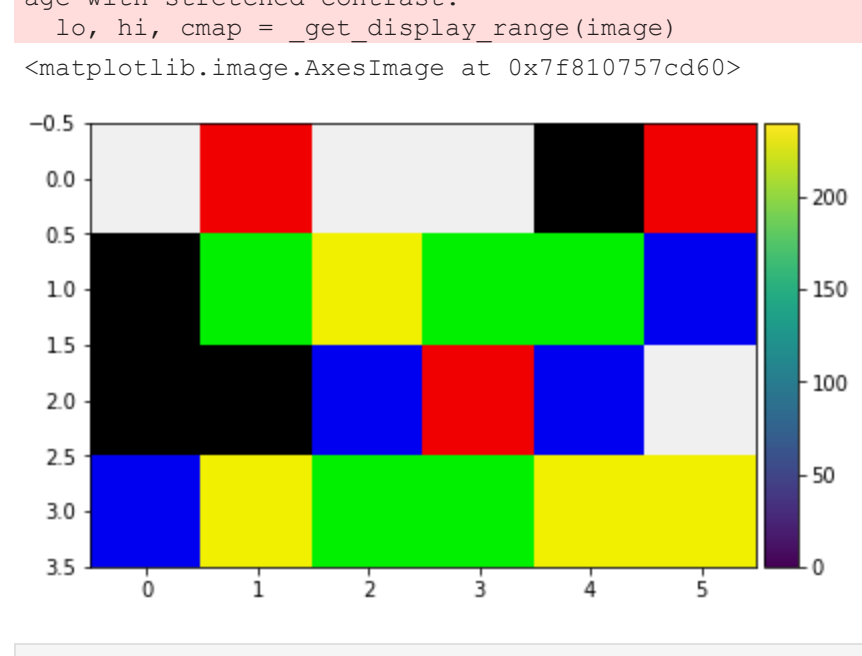
Out[11]: `<matplotlib.image.AxesImage at 0x7f8107461ca0>`



In [12]:
```python
# plot significant bits only
io.imshow(significant_bits_palette[indices])
```

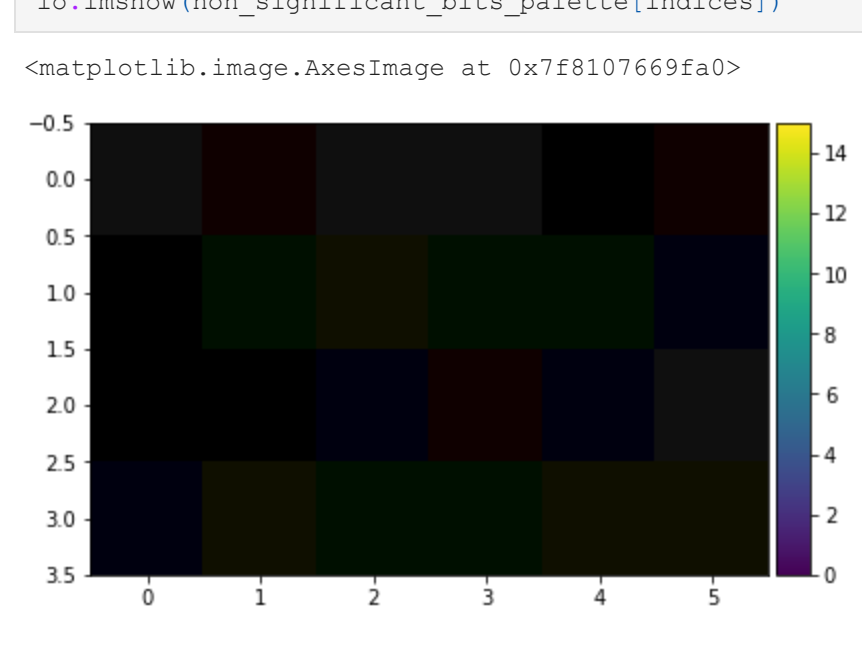
```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/skimage/
io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying im
age with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```

Out[12]: `<matplotlib.image.AxesImage at 0x7f810757cd60>`



In [13]:
```python
# plot non significant bits only
io.imshow(non_significant_bits_palette[indices])
```

Out[13]: `<matplotlib.image.AxesImage at 0x7f8107669fa0>`



Testing out with real image file

Now we try to perform the steganography on real photos.

Steganography class obtained from https://github.com/kelvins/steganography

In [14]:
```python
# PIL: Python Imaging Library
from PIL import Image

from typing import Tuple

class Steganography(object):
    @staticmethod
    def __int_to_bin(rgb: Tuple[int,int,int]) -> Tuple[str,str,str]:
        """Convert an integer tuple to a binary (string) tuple.

        Args:
            rgb: An integer tuple (e.g. (220, 110, 96))
        Return:
            A string tuple (e.g. ("00101010", "11101011", "00010110"))
        """
        r, g, b = rgb
        return ('{0:08b}'.format(r),
                '{0:08b}'.format(g),
                '{0:08b}'.format(b))

    @staticmethod
    def __bin_to_int(rgb: Tuple[str,str,str]) -> Tuple[int,int,int]:
        """Convert a binary (string) tuple to an integer tuple.
        Args:
            rgb: A string tuple (e.g. ("00101010", "11101011", "00010110"))
        Return:
            Return an int tuple (e.g. (220, 110, 96))
        """
        r, g, b = rgb
        return (int(r, 2),
                int(g, 2),
                int(b, 2))

    @staticmethod
    def __merge_rgb(rgb1: Tuple[str,str,str], rgb2: Tuple[str,str,str]) -> Tuple[str,s
        """Merge two RGB tuples.

        Args:
            rgb1: A string tuple (e.g. ("00101010", "11101011", "00010110"))
            rgb2: Another string tuple (e.g. ("00101010", "11101011", "00010110"))
        Return: An integer tuple with the two RGB values merged.
        """
        r1, g1, b1 = rgb1
        r2, g2, b2 = rgb2
        rgb = (r1[:4] + r2[:4],
               g1[:4] + g2[:4],
               b1[:4] + b2[:4])
        return rgb

    @staticmethod
    def merge(img1: Image, img2: Image) -> Image:
        """Merge two images. The second one will be merged into the first one.

        Args:
            img1: First image
            img2: Second image
        Return: A new merged image.
        """

        # Check the images dimensions
        if img2.size[0] > img1.size[0] or img2.size[1] > img1.size[1]:
            raise ValueError('Image 2 should not be larger than Image 1!')

        # Get the pixel map of the two images
        pixel_map1 = img1.load()
        pixel_map2 = img2.load()

        # Create a new image that will be outputted
        new_image = Image.new(img1.mode, img1.size)
        pixels_new = new_image.load()

        for i in range(img1.size[0]):
            for j in range(img1.size[1]):
                rgb1 = Steganography.__int_to_bin(pixel_map1[i, j])

                # Use a black pixel as default
                rgb2 = Steganography.__int_to_bin((0, 0, 0))

                # Check if the pixel map position is valid for the second image
                if i < img2.size[0] and j < img2.size[1]:
                    rgb2 = Steganography.__int_to_bin(pixel_map2[i, j])

                # Merge the two pixels and convert it to a integer tuple
                rgb = Steganography.__merge_rgb(rgb1, rgb2)

                pixels_new[i, j] = Steganography.__bin_to_int(rgb)

        return new_image

    @staticmethod
    def unmerge(img: Image) -> Image:
        """Unmerge an image.

        Args:
            img: The input image.
        Return: The unmerged/extracted image.
        """

        # Load the pixel map
        pixel_map = img.load()

        # Create the new image and load the pixel map
        new_image = Image.new(img.mode, img.size)
        pixels_new = new_image.load()

        # Tuple used to store the image original size
        original_size = img.size

        for i in range(img.size[0]):
            for j in range(img.size[1]):
                # Get the RGB (as a string tuple) from the current pixel
                r, g, b = Steganography.__int_to_bin(pixel_map[i, j])

                # Extract the last 4 bits (corresponding to the hidden image)
                # Concatenate 4 zero bits because we are working with 8 bit
                rgb = (r[4:] + '0000',
                       g[4:] + '0000',
                       b[4:] + '0000')

                # Convert it to an integer tuple
                pixels_new[i, j] = Steganography.__bin_to_int(rgb)

                # If this is a 'valid' position, store it
                # as the last valid position
                if pixels_new[i, j] != (0, 0, 0):
                    original_size = (i + 1, j + 1)

        # Crop the image based on the 'valid' pixels
        new_image = new_image.crop((0, 0, original_size[0], original_size[1]))

        return new_image
```

Original image (doge.jpg)

Original image

Hidden image (nyancat.jpeg)

Hidden image

In [15]:
```python
merged_image = Steganography.merge(Image.open('doge.jpg'), Image.open('nyancat.jpeg'))
merged_image.save('merged.jpg')
```

Merged image

merged image

(T5) Email Data Leakage Prevention - Example using regular expression filtering

Data Leak Prevention (DLP) is a feature that prevents certain information to leave the organization via email. What information this is, is defined in the configuration of the DLP system. Typically, it includes credit card numbers, bank account numbers, excessive amounts of email addresses or other personal information in one email message, etc.

In this example, we will be exploring regular expression (regex) filtering as a means of checking whether an outgoing mail contain personal information (e.g. NRIC, email addresses, telephone numbers)

A regular expression (shortened as regex or regexp) is a sequence of characters that define a search pattern.

Useful websites on regular expressions:

- regex testing: https://regexr.com/
- regex cheatsheet: https://www.rexegg.com/regex-quickstart.html
- python regex package: https://docs.python.org/3/library/re.html
- email regex (RFC 5322 Official Standard): http://emailregex.com/
- regex debugger with railroad diagram generator: https://www.debuggex.com/

Regular Expression basics

Character set notation: [...] or [^...]

- *example 1: [abc] means that only the characters 'a', 'b' or 'c' would be matched*
- *example 2: [a-zA-Z] means that all characters from 'a' to 'z' and 'A' to 'Z' would be matched*
- *example 3: [|-|'] means that only '-' or ''' symbols would be matched*
- *example 4 (negation set): [^abc] means any character that is not 'a', 'b' or 'c' would be matched*

Special character sets:

- **\w , \d , \s**: for word (i.e. any Unicode letter, ideogram, digit, or underscore), digit (i.e. 0 to 9), whitespace (i.e. ' ') respectively
- **\W , \D , \S**: for not word, not digit, not whitespace respectively
- **.** : any character

Repetition notation: {...}

- *example 1: [abc]{3} means that only the characters 'a', 'b' or 'c' that occurred consecutively 3 times (e.g. 'abc', 'bbc', 'aaa') would be matched*
- *example 2: [abc]{2,4} means that only the characters 'a', 'b' or 'c' that occurred consecutively 2-4 times (e.g. 'ab', 'bbc', 'aaaa') would be matched*
- *example 3: [abc]{2,} means that only the characters 'a', 'b' or 'c' that occurred consecutively at least 2 times (e.g. 'ab', 'bbc', 'aaaa', 'bbbbbbbbbbbbbbac') would be matched*

Special repetition notation:

- *****: equivalent to {0,}
- **+**: equivalent to {1,}

Capturing group: (...)

- used to group multiple tokens together for extracting substrings or for back reference
- *example 1: [STFG][0-9]{3}([0-9]{4}[A-IZJ]) would only capture the last 4 digits + 1 check alphabet instead of the full string matched.*

Non-capturing group: (?:...)

- used to group multiple tokens together but capture the substring
- *example 1: (?:+65)[0-9]{8} would match all 8 digit strings with additional +65 prefix*

In [16]:
```python
import re

# elaborate more on the regex components basics
nric_pattern = re.compile('[STFG][0-9]{7}[A-IZJ]')

# add email samples to test regex to filter all personal data. see existing corpuses

nric_pattern.search('asdf asdf adsf ad ad S1234567A asdfadf')
```

Out[16]: `<re.Match object; span=(21, 30), match='S1234567A'>`

Singapore Telephone Regex Railroad Diagram

SG-Telephone-Regex-Railroad-Diagram

In [17]:
```python
telephone_num_pattern = re.compile(r'(?:\+65 {0,1}){0,1}[0-9]{4} {0,1}[0-9]{4}')

telephone_num_pattern.findall('asd asdf adsf adf adf +65 12345678 +65 1213 1314 asdfa
```

Out[17]:
```
['+65 12345678', '+65 1213 1314', '+6532145678', '91231234', '9123 3214']
```

General Email Regex Railroad Diagram (RFC5322)

General-Email-Regex-Railroad-Diagram

In [18]:
```python
# do note to add \ in front of special characters (that have meaning in python) that
# eg ' and "
# use r"" for the string to interpret the characters raw so that you don't have to do

email_address_pattern = re.compile(r"(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&
                                   )



email_address_pattern.findall('asdf user%example.com@example.org  asd "john..doe"@exam
```

Out[18]:
```
['user%example.com@example.org',
 '"john..doe"@example.org',
 'user.name+tag+sorting@example.com',
 '"test_asdf"@testing.com',
 'qwe`~asdf@asdfasdf.com',
 'john.smith@[123.123.123.132]',
 'one+two-123@test-test.org.sg']
```

Debugging regular expressions

Note: numbers after LITERAL or RANGE refer to the ASCII decimal number reference for the character

Refer to http://www.asciitable.com/ for more information on ASCII character mapping

In [19]:
```python
some_faulty_pattern = re.compile('[+-=]',re.DEBUG) # use re.DEBUG flag to show how th

some_faulty_pattern.findall('1 - 1 = 0\n 1 + 1 = 2')
```

```
IN
  RANGE (43, 61)

 0. INFO 8 0b100 1 1 (to 9)
      in
 5.     RANGE 0x2b 0x3d ('+'-'=')
 8.     FAILURE
 9: IN 5 (to 15)
11.   RANGE 0x2b 0x3d ('+'-'=')
14.   FAILURE
15: SUCCESS
```

Out[19]:
```
['1', '-', '1', '=', '0', '1', '+', '1', '=', '2']
```

In [20]:
```python
rectified_pattern = re.compile('[+\-=]',re.DEBUG) # use re.DEBUG flag to show how the

rectified_pattern.findall('1 - 1 = 0\n 1 + 1 = 2')
```

```
IN
  LITERAL 43
  LITERAL 45
  LITERAL 61

 0. INFO 14 0b100 1 1 (to 15)
      in
 5.     CHARSET [0x00000000, 0x20002800, 0x00000000, 0x00000000, 0x00000000, 0x0000000
0, 0x00000000, 0x00000000]
14.     FAILURE
15: IN 11 (to 27)
17.   CHARSET [0x00000000, 0x20002800, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000]
26.   FAILURE
27: SUCCESS
```

Out[20]:
```
['-', '=', '+', '=']
```

Testing on a real life email dataset

There is a corpus of real emails archived from Enron that is available for public use. We use the dataset to test our PII filtering function.

In [21]:
```python
from typing import Dict
import pprint
import tarfile

pp = pprint.PrettyPrinter(indent=4)
```

In [22]:
```python
# define a pii_filter function
def pii_filter(text_data: str, regex_dict: Dict = None) -> Dict:
    """Function that filters for pii in text

    Args:
        text_data: textual data in string type
        regex_dict: optional dictionary of regex rules to use

    Return:
        dictionary of matched items based on each key in regex_dict
    """
    if regex_dict is None:
        regex_dict = {
#             'nric': '[STFG][0-9]{7}[A-IZJ]',
            'phone': '(?:[0-9]{3}|\([0-9]{3}\))?[-. ]?(?:[0-9]{3})[-. ]?(?:[0-9]{4})',
#             'phone': '(?:\+65 {0,1}){0,1}[0-9]{4} {0,1}[0-9]{4}',
            'email': r"(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-
        }

    pii_dict = {}

    for k,v in regex_dict.items():
        pattern = re.compile(v)
        pii_dict[k]= pattern.findall(text_data)

    return pii_dict

pp.pprint(pii_filter('asdf user%example.com@example.org 713-853-5660 (713) 853-5660 as
```

```
{   'email': [   'user%example.com@example.org',
                 '"john..doe"@example.org',
                 'user.name+tag+sorting@example.com',
                 '"test_asdf"@testing.com',
                 'qwe`~asdf@asdfasdf.com',
                 'john.smith@[123.123.123.132]',
                 'one+two-123@test-test.org.sg'],
    'phone': ['713-853-5660', '(713) 853-5660']}
```

Download subset of enron email corpus for testing

In [23]:
```python
# download takes very long so please use local copy once downloaded.
# Please do not untar the full corpus in an unprotected computer as there are some em

# from io import StringIO
# import requests
# import os

#################### DOWNLOAD FULL EMAIL CORPUS #######################
# url = "https://www.cs.cmu.edu/~./enron/enron_mail_20150507.tar.gz"
# with open('enron_mail_20150507.tar.gz', 'wb') as f:
#     f.write(requests.get(url).content)
#########################################################################

# emails = []

# url = "https://www.cs.cmu.edu/~./enron/enron_mail_20150507.tar.gz"
# enron_mail_data = StringIO(requests.get(url).content)


# # save a subset of emails to local
# python_folder_path = os.path.abspath('')

# if not os.path.exists('enron_emails'):
#     os.makedirs('enron_emails')

# download_path = os.path.join(python_folder_path,'enron_emails')


# with tarfile.open(enron_mail_data, 'r|*') as email_tar:
#     for member in email_tar:
#         if member.name.startswith('maildir/blair-l/meetings___nng_customer_mtg/') a
#             full_file_path = os.path.join(download_path, member.name)
#             if not os.path.exists(os.path.dirname(full_file_path)):
#                 os.makedirs(os.path.dirname(full_file_path), exist_ok=True)
#             with open(full_file_path, 'wb') as f:
#                 f.write(email_tar.extractfile(member).read())
```

Extract data from email files

In [24]:
```python
import os

python_folder_path = os.path.abspath('')

download_path = os.path.join(python_folder_path,'enron_emails')

emails = []

for root, dirs, files in os.walk(download_path, topdown=False):
    for filename in files:
        with open(os.path.join(root, filename), 'rb') as f:
            full_content = f.read().decode('ascii')
            emails.append({
                'filename': filename,
                'content': full_content,
                'header': str(full_content.split('.pst\r\n')[0]),
                'main_message': str(full_content.split('.pst\r\n')[1])
            })


print('Example email content\n')
print(emails[0]['main_message'])
```

```
Example email content



Shelley, Larry Berger has put together the 4 PAA's based on these employeef efforts

 on the testing of the new Flowing Gas Documents.  They did a very good job and I thin
k

it would be good tor recognize them for their work on testing while they continued to

keep their day to day activities under control as well.

Diana Porter and Jean Adams are for $150

Kathy Sturr and Renee Perry are for $250

If you approve, please process these PAA's.  If you have any questions, please let me
know.
```

Thanks. Lynn

In [25]:
```python
scan_results = []
for email in emails:
    scan_results.append(
        {
            'filename': email['filename'],
            'scan_results': pii_filter(email['main_message'])
        }
    )

pp.pprint(scan_results)
```

```
[   {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '1',
        'scan_results': {   'email': ['audrey.robertson@enron.com'],
                            'phone': ['(713) 853-5849', '(713) 646-2551']}},
    {   'filename': '3',
        'scan_results': {'email': [], 'phone': ['(713) 853-5754']}},
    {'filename': '2', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '1',
        'scan_results': {   'email': ['isc.hotline@enron.com'],
                            'phone': ['713.345.4727']}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '7',
        'scan_results': {   'email': [   "'fkelly@gbmdc.com",
                                         "'sstojic@gbmdc.com",
                                         "'rnuschler@akingump.com",
                                         "'cmoore@akingump.com"],
                            'phone': []}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '2', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '2', 'scan_results': {'email': [], 'phone': ['001-2002']}},
    {   'filename': '92',
        'scan_results': {'email': ["'gharvey@gibbs-bruns.com"], 'phone': []}},
    {   'filename': '51',
        'scan_results': {'email': [], 'phone': ['(713) 853-6600']}},
    {   'filename': '93',
        'scan_results': {'email': ["'gharvey@gibbs-bruns.com"], 'phone': []}},
    {   'filename': '20',
        'scan_results': {'email': ["'gharvey@gibbs-bruns.com"], 'phone': []}},
    {'filename': '18', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '27', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '9', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '11', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '7',
        'scan_results': {'email': ["'charlie_graham@enron.com"], 'phone': []}},
    {   'filename': '16',
        'scan_results': {   'email': [   "'gharvey@gibbs-bruns.com",
                                         "'gharvey@gibbs-bruns.com",
                                         'cott@gibbs-bruns.com'],
                            'phone': []}},
    {   'filename': '89',
        'scan_results': {'email': ["'gharvey@gibbs-bruns.com"], 'phone': []}},
    {'filename': '87', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '6',
        'scan_results': {   'email': [],
                            'phone': [   ' 0658605',
                                         ' 0658606',
                                         '(620)562-4325']}},
    {   'filename': '28',
        'scan_results': {'email': [], 'phone': ['(713) 853-5691']}},
    {   'filename': '17',
        'scan_results': {   'email': [],
                            'phone': ['918-732-1355', '918-732-1325']}},
    {   'filename': '1',
        'scan_results': {'email': ["'gharvey@gibbs-bruns.com"], 'phone': []}},
    {'filename': '10', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '19', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '26', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '8', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '21',
        'scan_results': {   'email': [],
                            'phone': ['402-398-7061', '402-398-7122']}},
    {   'filename': '81',
        'scan_results': {'email': ["'gharvey@gibbs-bruns.com"], 'phone': []}},
    {'filename': '88', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '91', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '90', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '79', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '77', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '84', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '24', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '23',
        'scan_results': {   'email': [],
                            'phone': ['402-398-7061', '402-398-7122']}},
    {   'filename': '4',
        'scan_results': {'email': [], 'phone': ['(402) 398-7061']}},
    {   'filename': '15',
        'scan_results': {'email': ['cott@gibbs-bruns.com'], 'phone': []}},
    {   'filename': '3',
        'scan_results': {   'email': [   "'ronnie.brinkman@enron.com",
                                         'raham@enron.com',
                                         'raham@enron.com',
                                         'ile@enron.com',
                                         'oseley@enron.com'],
                            'phone': ['(402) 398-7061']}},
    {'filename': '12', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '85', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '82', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '78',
        'scan_results': {   'email': [   'gharvey@gibbs-bruns.com',
                                         'gharvey@gibbs-bruns.com'],
                            'phone': []}},
    {   'filename': '2',
        'scan_results': {   'email': [],
                            'phone': ['(620) 562-7694', '(620) 792-3991']}},
    {'filename': '13', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '5', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '14', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '22',
        'scan_results': {   'email': ["'hwoodso@enron.com"],
                            'phone': [   '402-398-7061',
                                         '402-398-7122',
                                         '(918) 732-1303',
                                         '(918) 588-7466']}},
    {   'filename': '25',
        'scan_results': {'email': [], 'phone': ['(402) 398-7061']}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '2', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '32',
        'scan_results': {   'email': ['gblair@enron.com'],
                            'phone': [   ' 0179010',
                                         '0050371',
                                         '(713) 345-4727']}},
    {   'filename': '35',
        'scan_results': {   'email': ['houlvlm@ups.com', 'houlvlm@ups.com'],
                            'phone': ['281-847-6769', '800-432-2877']}},
    {   'filename': '34',
        'scan_results': {   'email': [   'avidson@socalgas.com',
                                         'lblair@enron.com',
                                         'ohman@enron.com',
                                         'ohman@enron.com',
                                         'pdavidson@socalgas.com'],
                            'phone': []}},
    {'filename': '33', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '20',
        'scan_results': {   'email': [   'houlvlm@ups.com',
                                         'arcia@enron.com',
                                         'houlanb@ups.com',
                                         'houlmbb@ups.com',
                                         'glenn.lewis@enron.com',
                                         'arcia@enron.com',
                                         'houlvlm@ups.com'],
                            'phone': []}},
    {'filename': '18', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '27', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '9', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '11',
        'scan_results': {'email': [], 'phone': ['713-853-5787']}},
    {'filename': '7', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '29', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '16',
        'scan_results': {   'email': [],
                            'phone': [   '(403) 920-6341',
                                         '(505) 564-4015',
                                         '(505) 599-2357',
                                         '(505) 599-2334']}},
    {'filename': '6', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '28',
        'scan_results': {   'email': [   'jbowen@instrometinc.com',
                                         'tjlcpl@pipeline.com'],
                            'phone': []}},
    {   'filename': '17',
        'scan_results': {   'email': ['chris.sebesta@enron.com'],
                            'phone': ['402-398-7064', '402-398-7413']}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '10', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '19', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '26',
        'scan_results': {   'email': ['houlvlm@ups.com', 'houlvlm@ups.com'],
                            'phone': ['281-847-6769', '800-432-2877']}},
    {'filename': '8', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '21', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '38', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '36', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '31',
        'scan_results': {   'email': ['gblair@enron.com'],
                            'phone': [   ' 0179010',
                                         '0046227',
                                         '(713) 345-4727']}},
    {   'filename': '30',
        'scan_results': {   'email': [   'obinhold@ftenergy.com',
                                         "'lynn.blair@enron.com",
                                         'lynn.blair@enron.com',
                                         'brobinhold@ftenergy.com'],
                            'phone': ['720-548-5722']}},
    {'filename': '37', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '39', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '24', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '23', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '4', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '15', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '3', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '12',
        'scan_results': {'email': [], 'phone': ['(402) 398-7061']}},
    {'filename': '40', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '2', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '13', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '5',
        'scan_results': {'email': ['ulligan@enron.com'], 'phone': []}},
    {'filename': '14', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '22', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '25', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '135', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '307',
        'scan_results': {   'email': [],
                            'phone': ['713-767-5406', '505-355-2468']}},
    {'filename': '551', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '61',
        'scan_results': {   'email': [   'athaway@dom.com',
                                         'athaway@dom.com',
                                         'lair@enron.com',
                                         'athaway@dom.com',
                                         'athaway@dom.com',
                                         'ryselius@dom.com',
                                         'ewis@dom.com',
                                         'arlo@dom.com',
                                         'ee@dom.com',
                                         'athaway@dom.com',
                                         'oyer@dom.com',
                                         'ryselius@dom.com',
                                         'ewis@dom.com',
                                         'arlo@dom.com',
                                         'ee@dom.com',
                                         'nng@enron.com',
                                         'greg_hathaway@dom.com',
                                         'enron.messaging.administration@enron.com'],
                            'phone': ['2001090', '7841582']}},
    {   'filename': '95',
        'scan_results': {'email': [], 'phone': ['(713) 853-3072']}},
    {'filename': '338', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '556', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '300', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '132', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '59', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '569',
        'scan_results': {   'email': ['-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '92', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '66', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '336',
        'scan_results': {'email': [], 'phone': ['915-685-4103']}},
    {   'filename': '104',
        'scan_results': {   'email': [],
                            'phone': ['(713)853-1966', '(713)303-2985']}},
    {   'filename': '560',
        'scan_results': {'email': [], 'phone': ['281.775.8844']}},
    {'filename': '309', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '50', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '68', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '593',
        'scan_results': {'email': ['help@gossipflash.com'], 'phone': []}},
    {   'filename': '567',
        'scan_results': {'email': ['help@gossipflash.com'], 'phone': []}},
    {   'filename': '103',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {   'filename': '331',
        'scan_results': {'email': [], 'phone': ['915-685-4103']}},
    {'filename': '57', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '558', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '168', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '391', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '533', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '365', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '157',
        'scan_results': {'email': [], 'phone': ['720.548.5470']}},
    {'filename': '150', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '362', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '534',
        'scan_results': {   'email': [   'nergyreleases@yahoo.com',
                                         'info@fea.com',
                                         'sponsor@enerfax.com',
                                         'subscribegold@enerfax.com',
                                         'advertising@enerfax.com'],
                            'phone': [   '281-681-3301',
                                         '386-2700',
                                         '976-6111',
                                         '510-548-6200',
                                         '281-362-7979',
                                         '281-296-9922',
                                         '888-792-2592']}},
    {   'filename': '396',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '159', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '32',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {   'filename': '502',
        'scan_results': {   'email': [   'nergyreleases@yahoo.com',
                                         'info@fea.com',
                                         'sponsor@enerfax.com',
                                         'subscribegold@enerfax.com',
                                         'advertising@enerfax.com'],
                            'phone': [   '281-681-3301',
                                         '386-2700',
                                         '976-6111',
                                         '510-548-6200',
                                         '281-362-7979',
                                         '281-296-9922',
                                         '888-792-2592']}},
    {'filename': '166', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '354', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '192', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '398', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '35',
        'scan_results': {   'email': [],
                            'phone': [   '800-973-6766',
                                         '800-973-6766',
                                         '713-345-5552',
                                         '713-345-8525']}},
    {   'filename': '195',
        'scan_results': {   'email': [],
                            'phone': ['(713) 345-3951', '(832) 661-1933']}},
    {   'filename': '353',
        'scan_results': {'email': ['mary.darveaux@enron.com'], 'phone': []}},
    {'filename': '161', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '505', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '566',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '330', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '102', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '592', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '69', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '559',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {'filename': '56', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '105', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '337', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '561', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '51', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '308', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '58', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '557',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {   'filename': '133',
        'scan_results': {   'email': [],
                            'phone': ['402-398-7061', '402-398-7122']}},
    {   'filename': '301',
        'scan_results': {   'email': ['embassy9@home.com'],
                            'phone': ['915-688-3838', '915-685-4103']}},
    {'filename': '67', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '93',
        'scan_results': {'email': [], 'phone': ['(713) 853-3072']}},
    {   'filename': '568',
        'scan_results': {'email': ['help@infobeat.com'], 'phone': []}},
    {   'filename': '306',
        'scan_results': {'email': [], 'phone': ['402-398-7468']}},
    {'filename': '134', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '550', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '94',
        'scan_results': {'email': [], 'phone': ['(713) 853-3072']}},
    {'filename': '339', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '60',
        'scan_results': {   'email': [   'athaway@dom.com',
                                         'athaway@dom.com',
                                         'lair@enron.com',
                                         'athaway@dom.com',
                                         'athaway@dom.com',
                                         'ryselius@dom.com',
                                         'ewis@dom.com',
                                         'arlo@dom.com',
                                         'ee@dom.com',
                                         'athaway@dom.com',
                                         'oyer@dom.com',
                                         'ryselius@dom.com',
                                         'ewis@dom.com',
                                         'arlo@dom.com',
                                         'ee@dom.com',
                                         'nng@enron.com',
                                         'greg_hathaway@dom.com',
                                         'enron.messaging.administration@enron.com'],
                            'phone': ['2001090', '7841582']}},
    {   'filename': '399',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {   'filename': '34',
        'scan_results': {'email': ['eia@enron.com'], 'phone': []}},
    {'filename': '160', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '352', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '504', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '194', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '33',
        'scan_results': {   'email': ['jlloyd@socalgas.com'],
                            'phone': ['(213) 244-3717']}},
    {'filename': '158', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '193', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '503',
        'scan_results': {'email': ['help@gossipflash.com'], 'phone': []}},
    {'filename': '355', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '167',
        'scan_results': {   'email': [],
                            'phone': ['713-345-4745', '402-398-7454']}},
    {'filename': '363', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '151', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '535', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '169', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '532',
        'scan_results': {'email': [], 'phone': ['(713) 853-3072']}},
    {'filename': '156', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '364',
        'scan_results': {   'email': [   "'steven.january@enron.com",
                                         "'bradley.holmes@enron.com",
                                         "'grace.blair@enron.com"],
                            'phone': []}},
    {'filename': '390', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '686', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '216', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '672', 'scan_results': {'email': [], 'phone': ['0000000000']}},
    {'filename': '229', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '675', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '211', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '681', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '478', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '227', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '471',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {'filename': '643', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '485', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '218', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '688', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '482', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '644', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '476', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '220', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '449', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '280', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '422',
        'scan_results': {   'email': ['fran.fagan@enron.com'],
                            'phone': ['713.853.5219']}},
    {   'filename': '610',
        'scan_results': {'email': [], 'phone': ['713-345-8923']}},
    {   'filename': '274',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '628', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '273', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '617', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '425', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '287', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '621', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '413',
        'scan_results': {   'email': [],
                            'phone': ['713-345-4745', '402-398-7454']}},
    {'filename': '245', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '289',
        'scan_results': {'email': ['help@infobeat.com'], 'phone': []}},
    {'filename': '619', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '242', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '414',
        'scan_results': {'email': [], 'phone': [' 0722015', '713-646-7410']}},
    {   'filename': '626',
        'scan_results': {   'email': [],
                            'phone': ['713 853-5996', '(402) 398-7130']}},
    {'filename': '477', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '645',
        'scan_results': {   'email': ['randy.bryan@enron.com'],
                            'phone': ['0000000000', ' 0000000', '0007659']}},
    {   'filename': '221',
        'scan_results': {   'email': ['alpin@enron.com'],
                            'phone': ['402-398-7334']}},
    {'filename': '483', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '448', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '484',
        'scan_results': {   'email': [   'avidson@socalgas.com',
                                         'lblair@enron.com',
                                         'ohman@enron.com',
                                         'ohman@enron.com',
                                         'pdavidson@socalgas.com'],
                            'phone': []}},
    {'filename': '226', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '642',
        'scan_results': {'email': ['swalker@sidrichgas.com'], 'phone': []}},
    {'filename': '470', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '689', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '219', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '680',
        'scan_results': {'email': ['swalker@sidrichgas.com'], 'phone': []}},
    {'filename': '674', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '446',
        'scan_results': {   'email': [   'cormanerb@cs.com',
                                         'rick.dietz@kingwoodcable.com'],
                            'phone': []}},
    {'filename': '210', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '479', 'scan_results': {'email': [], 'phone': ['2757723']}},
    {'filename': '217', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '673', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '687',
        'scan_results': {'email': [], 'phone': ['713-853-6933']}},
    {'filename': '228', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '618', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '288',
        'scan_results': {'email': ['help@gossipflash.com'], 'phone': []}},
    {'filename': '243', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '627', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '415', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '412', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '620',
        'scan_results': {'email': [], 'phone': ['952-887-1782']}},
    {   'filename': '244',
        'scan_results': {   'email': ['karen.clapper@enron.com'],
                            'phone': ['713-853-6735', '713-646-8000']}},
    {'filename': '629', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '286', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '272', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '424', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '616', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '611', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '423', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '275',
        'scan_results': {   'email': [   'nergyreleases@yahoo.com',
                                         'info@fea.com',
                                         'sponsor@enerfax.com',
                                         'subscribegold@enerfax.com',
                                         'advertising@enerfax.com'],
                            'phone': [   '281-681-3301',
                                         '386-2700',
                                         '976-6111',
                                         '510-548-6200',
                                         '281-362-7979',
                                         '281-296-9922']}},
    {'filename': '281', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '257',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {   'filename': '401',
        'scan_results': {   'email': ['info@rigzone.com'],
                            'phone': ['281-345-4040']}},
    {'filename': '633', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '268', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '634', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '406', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '250', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '439',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {   'filename': '292',
        'scan_results': {   'email': ['gblair@enron.com'],
                            'phone': [   '800-642-1617',
                                         '800-774-1500',
                                         '800-227-7368',
                                         ' 5037126']}},
    {'filename': '266', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '602',
        'scan_results': {   'email': ['randy.bryan@enron.com'],
                            'phone': ['0000000000', ' 0000000', '0007659']}},
    {'filename': '430', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '259', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '437',
        'scan_results': {'email': ['help@gossipflash.com'], 'phone': []}},
    {'filename': '605', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '261',
        'scan_results': {'email': ['eia@enron.com'], 'phone': []}},
    {'filename': '295', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '408', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '651', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '463',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '235', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '497', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '669',
        'scan_results': {   'email': ['hillips@entergykoch.com'],
                            'phone': ['713-544-4017']}},
    {'filename': '490', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '232', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '464', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '656',
        'scan_results': {   'email': [],
                            'phone': ['505-392-4943', '580-338-3929']}},
    {'filename': '499', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '452', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '660', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '204', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '658', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '203', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '667', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '455',
        'scan_results': {   'email': ['chris.sebesta@enron.com'],
                            'phone': ['402-398-7064', '402-398-7413']}},
    {   'filename': '294',
        'scan_results': {'email': [], 'phone': ['714-408-6837']}},
    {'filename': '604', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '436', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '260', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '409', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '267',
        'scan_results': {   'email': ['hdostello@houston.rr.com'],
                            'phone': [   '713 853 1411',
                                         '888 853 9797',
                                         '713 345 4745',
                                         '402 398 7454']}},
    {'filename': '431', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '603', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '293', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '258', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '407', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '635', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '251', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '438', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '256',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {   'filename': '632',
        'scan_results': {'email': [], 'phone': ['5274062486', '4873664']}},
    {   'filename': '400',
        'scan_results': {'email': ['help@gossipflash.com'], 'phone': []}},
    {   'filename': '269',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {'filename': '659', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '202', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '454',
        'scan_results': {'email': [], 'phone': ['(713)853-6816']}},
    {'filename': '666', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '498', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '661', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '453', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '205', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '668', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '233', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '657', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '465',
        'scan_results': {   'email': [   'nergyreleases@yahoo.com',
                                         'info@fea.com',
                                         'sponsor@enerfax.com',
                                         'tsavaiano@summitenergy.com',
                                         'subscribegold@enerfax.com',
                                         'advertising@enerfax.com'],
                            'phone': [   '281-681-3301',
                                         '386-2700',
                                         '976-6111',
                                         '510-548-6200',
                                         '281-362-7979',
                                         '281-296-9922',
                                         '888-792-2592']}},
    {   'filename': '491',
        'scan_results': {   'email': ['dennis.lee@enron.com'],
                            'phone': [' 1000000', '(713) 853-1715']}},
    {'filename': '496', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '462', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '650', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '234', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '346',
        'scan_results': {'email': ['mary.darveaux@enron.com'], 'phone': []}},
    {'filename': '174', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '510',
        'scan_results': {   'email': [   "'mike.bodnar@enron.com",
                                         'chneider@alliant-energy.com',
                                         'chneider@alliant-energy.com',
                                         'icken@alliant-energy.com',
                                         'orr@alliant-energy.com',
                                         'aldwell@alliant-energy.com',
                                         'jdoering@navigantconsulting.com',
                                         'jdoering@navigantconsulting.com'],
                            'phone': ['(608) 252-3078', '(608) 252-3130']}},
    {'filename': '180', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '379', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '20', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '187', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '18', 'scan_results': {'email': [], 'phone': ['986-1996']}},
    {'filename': '517', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '173', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '341', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '27', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '9', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '145',
        'scan_results': {'email': [], 'phone': ['720.548.5470']}},
    {'filename': '377', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '521',
        'scan_results': {   'email': [],
                            'phone': ['713-345-4745', '402-398-7454']}},
    {'filename': '11', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '348', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '526', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '370', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '142',
        'scan_results': {   'email': [],
                            'phone': ['713-345-4745', '402-398-7454']}},
    {'filename': '384', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '7', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '29', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '519',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {   'filename': '16',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {'filename': '189', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '129', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '42',
        'scan_results': {   'email': ['alpin@enron.com'],
                            'phone': ['402-398-7334']}},
    {'filename': '89', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '572',
        'scan_results': {'email': [], 'phone': ['(713) 853-5754']}},
    {   'filename': '116',
        'scan_results': {'email': ['news@oxfordprinceton.com'], 'phone': []}},
    {'filename': '324', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '586',
        'scan_results': {   'email': [   'notice@enerfax.com',
                                         'bgwozd@ziffenergy.com',
                                         'info@fea.com',
                                         'nergyreleases@yahoo.com',
                                         'dvertise@enerfax.com',
                                         'gastorage@aol.com',
                                         'gastorage@aol.com',
                                         'enerfax-subscribe@egroups.com',
                                         'aily@relay.netatlantic.com'],
                            'phone': [   '888-736-5780',
                                         '800-853-6252',
                                         '281-362-7979',
                                         '281-296-9922',
                                         '681-3301',
                                         '386-2700',
                                         '510-548-6200',
                                         '713-266-7771',
                                         '(212)515-3700',
                                         '888-792-2592',
                                         '915 698 3699',
                                         '915 698 2859']}},
    {   'filename': '45',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '581', 'scan_results': {'email': [], 'phone': ['2000000349']}},
    {   'filename': '323',
        'scan_results': {   'email': ['chris.sebesta@enron.com'],
                            'phone': ['402-398-7064', '402-398-7413']}},
    {'filename': '111', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '575',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {   'filename': '588',
        'scan_results': {'email': ['esell@midamerican.com'], 'phone': []}},
    {   'filename': '73',
        'scan_results': {'email': [], 'phone': ['412-279-9298']}},
    {'filename': '118', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '87', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '543', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '315', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '127', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '80', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '120',
        'scan_results': {'email': [], 'phone': ['713-345-3858']}},
    {'filename': '312', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '544', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '6', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '385',
        'scan_results': {'email': [], 'phone': ['501-788-8907']}},
    {'filename': '28', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '527',
        'scan_results': {'email': ['lp@quickinspirations.com'], 'phone': []}},
    {   'filename': '143',
        'scan_results': {'email': [], 'phone': ['720.548.5470']}},
    {'filename': '371', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '17',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {'filename': '188', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '518', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '376', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '144', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '520',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {   'filename': '1',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '349', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '10', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '340', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '172', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '186', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '19', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '529', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '26', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '8', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '181', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '175', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '347', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '511', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '21', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '378', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '75',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '81', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '313',
        'scan_results': {   'email': ['embassy9@home.com'],
                            'phone': ['915-688-3838', '915-685-4103']}},
    {   'filename': '121',
        'scan_results': {   'email': [   'wellness@enron.com',
                                         'wellness@enron.com'],
                            'phone': ['713-745-4000']}},
    {'filename': '545', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '119', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '86', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '72',
        'scan_results': {   'email': [],
                            'phone': [   '(270) 688-6968',
                                         '(270) 688-6637',
                                         '(270) 688-6944']}},
    {   'filename': '589',
        'scan_results': {   'email': [   'chneider@alliant-energy.com',
                                         'chneider@alliant-energy.com',
                                         'icken@alliant-energy.com',
                                         'orr@alliant-energy.com',
                                         'aldwell@alliant-energy.com',
                                         'jdoering@navigantconsulting.com',
                                         'jdoering@navigantconsulting.com'],
                            'phone': ['(608) 252-3078', '(608) 252-3130']}},
    {'filename': '126', 'scan_results': {'email': [], 'phone': ['2757723']}},
    {'filename': '314', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '44',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '110', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '322', 'scan_results': {'email': [], 'phone': ['1000559']}},
    {'filename': '574', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '580', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '43',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {   'filename': '128',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '587', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '573',
        'scan_results': {   'email': ['dennis.lee@enron.com'],
                            'phone': [' 4000002', '(713) 853-1715']}},
    {'filename': '88', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '325', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '117',
        'scan_results': {   'email': [   'info@princetonlive.com',
                                         'news@oxfordprinceton.com'],
                            'phone': ['(609) 520-9099']}},
    {'filename': '508', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '198',
        'scan_results': {   'email': [],
                            'phone': ['(713) 345-3951', '(832) 661-1933']}},
    {'filename': '361', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '153',
        'scan_results': {'email': [], 'phone': ['(713) 853-3072']}},
    {'filename': '537', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '38',
        'scan_results': {   'email': ['alpin@enron.com'],
                            'phone': ['402-398-7334']}},
    {   'filename': '395',
        'scan_results': {'email': [], 'phone': ['2746140', '2746140']}},
    {'filename': '359', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '392', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '530', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '154', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '366', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '36', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '539',
        'scan_results': {   'email': [   'christina.cheddar@dowjones.com',
                                         'carol.remond@dowjones.com',
                                         'michael.rapoport@dowjones.com'],
                            'phone': [   '201-938-5400',
                                         '201-938-5166',
                                         '201-938-2074',
                                         '201-938-5166',
                                         '201-938-5976']}},
    {'filename': '196', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '162', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '350', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '506',
        'scan_results': {   'email': [],
                            'phone': [   '1231306',
                                         '1231306',
                                         '1231306',
                                         '1231306',
                                         '1231306',
                                         '1231306']}},
    {'filename': '368', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '31', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '357', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '165', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '191', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '555', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '131', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '303', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '91', 'scan_results': {'email': [], 'phone': ['2000000316']}},
    {   'filename': '65',
        'scan_results': {   'email': [   'athaway@dom.com',
                                         'athaway@dom.com',
                                         'ryselius@dom.com',
                                         'ewis@dom.com',
                                         'arlo@dom.com',
                                         'ee@dom.com',
                                         'athaway@dom.com',
                                         'oyer@dom.com',
                                         'ryselius@dom.com',
                                         'ewis@dom.com',
                                         'arlo@dom.com',
                                         'ee@dom.com',
                                         'nng@enron.com',
                                         'greg_hathaway@dom.com'],
                            'phone': ['2001090', '7841582']}},
    {   'filename': '304',
        'scan_results': {'email': [], 'phone': ['713-853-3000']}},
    {'filename': '136', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '552', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '62',
        'scan_results': {   'email': [   'athaway@dom.com',
                                         'athaway@dom.com',
                                         'lair@enron.com',
                                         'athaway@dom.com',
                                         'athaway@dom.com',
                                         'ryselius@dom.com',
                                         'ewis@dom.com',
                                         'arlo@dom.com',
                                         'ee@dom.com',
                                         'athaway@dom.com',
                                         'oyer@dom.com',
                                         'ryselius@dom.com',
                                         'ewis@dom.com',
                                         'arlo@dom.com',
                                         'ee@dom.com',
                                         'nng@enron.com',
                                         'greg_hathaway@dom.com',
                                         'enron.messaging.administration@enron.com'],
                            'phone': ['2001090', '7841582']}},
    {'filename': '599', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '96',
        'scan_results': {'email': [], 'phone': ['(713) 853-3072']}},
    {'filename': '109', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '590',
        'scan_results': {   'email': ['qi-help@quickinspirations.com'],
                            'phone': []}},
    {'filename': '564', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '332',
        'scan_results': {'email': [], 'phone': ['713-853-1411']}},
    {'filename': '100', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '54',
        'scan_results': {   'email': ['dennis.lee@enron.com'],
                            'phone': [   '-694-8210',
                                         '-861-1739',
                                         '(713) 853-1715']}},
    {'filename': '107', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '335', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '563', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '98',
        'scan_results': {   'email': [   "'kvanpelt@cmsenergy.com",
                                         "'jhickman@nisource.com",
                                         "'jtyoung@nisource.com",
                                         'ris_g._king@dom.com',
                                         "'theresa.hess@enron.com",
                                         'ynn.blair@enron.com',
                                         "'randy.young@gulfsouthpl.com",
                                         "'tom_gwilliam@iroquois.com",
                                         "'michael.d.rasmuson@williams.com",
                                         "'janie.k.nielsen@williams.com",
                                         'lark@neg.pge.com',
                                         "'jarregu@mail.pnm.com",
                                         'illis@reliantenergy.com',
                                         "'tjwilliams@reliantenergy.com",
```

```
                              'phone': ['415.973.4455']}},
    {   'filename': '13',
        'scan_results': {'email': [], 'phone': ['415.973.1386']}},
    {   'filename': '5',
        'scan_results': {   'email': [   'lair@enron.com',
                                         '2@pge.com',
                                         'theresa.hess@enron.com',
                                         '2@pge.com',
                                         'aureen.riter@elpaso.com',
                                         "'william.griffith@elpaso.com",
                                         "'adele.zuroff@elpaso.com",
                                         "'kvanpelt@cmsenergy.com",
                                         "'ernesto.ochoa@elpaso.com",
                                         'oyal@elpaso.com',
                                         'ynn.blair@enron.com',
                                         'acey@enron.com',
                                         "'randy.young@gulfsouthpl.com",
                                         "'michael.d.rasmuson@williams.com",
                                         "'janie.k.nielsen@williams.com",
                                         "'jarregu@mail.pnm.com",
                                         "'pdavidson@socalgas.com",
                                         'shikawa@socalgas.com',
                                         "'mark.gracey@elpaso.com",
                                         "'tom_gwilliam@iroquois.com",
                                         "'jay.story@neg.pge.com",
                                         'ris_g._king@dom.com',
                                         'illis@reliantenergy.com',
                                         "'tjwilliams@reliantenergy.com",
                                         "'csuarez@reliantenergy.com",
                                         "'jhickman@nisource.com",
                                         "'jtyoung@nisource.com",
                                         'tilwell@sppc.com',
                                         "'gsuga@tuscaroragas.com",
                                         "'scotth@questar.com",
                                         "'klburch@duke-energy.com",
                                         "'mlmccain@duke-energy.com",
                                         "'ken_schubert@transcanada.com",
                                         "'dale.m.davis@williams.com",
                                         'enron.messaging.administration@enron.com'],
                            'phone': [   '415.973.1386',
                                         '(713) 853-7637',
                                         '888-627-7024',
                                         '888-627-7024',
                                         '415.973.1386']}},
    {   'filename': '14',
        'scan_results': {'email': [], 'phone': ['415.973.1386']}},
    {   'filename': '18',
        'scan_results': {   'email': [   'athaway@dom.com',
                                         'athaway@dom.com',
                                         'ercaldo@enron.com',
                                         'enron.messaging.administration@enron.com'],
                            'phone': []}},
    {'filename': '9', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '11', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '7', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '16',
        'scan_results': {   'email': [],
                            'phone': [   '816-756-1500',
                                         '816-756-1635',
                                         '713-853-5660']}},
    {   'filename': '6',
        'scan_results': {   'email': ['steven.january@enron.com'],
                            'phone': ['713-853-5660']}},
    {'filename': '17', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '10',
        'scan_results': {'email': [], 'phone': ['713-853-5660']}},
    {'filename': '8', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '4', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '15',
        'scan_results': {   'email': [],
                            'phone': [   '(713) 646-8000',
                                         '(713) 853-5660',
                                         '713-853-5660']}},
    {'filename': '3', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '12',
        'scan_results': {'email': [], 'phone': ['713-853-5660']}},
    {'filename': '2', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '13',
        'scan_results': {'email': ['dave.tyler@semcoenergy.com'], 'phone': []}},
    {   'filename': '5',
        'scan_results': {   'email': [   'jmosher@oneok.com',
                                         'tjones@oneok.com',
                                         'dennis.lee@enron.com'],
                            'phone': ['(713) 853-1715']}},
    {   'filename': '14',
        'scan_results': {'email': [], 'phone': ['713-853-5660']}},
    {   'filename': '1',
        'scan_results': {'email': ['mary.darveaux@enron.com'], 'phone': []}},
    {'filename': '8', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '2', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '5', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '1',
        'scan_results': {   'email': ['ipayit@enron.com', 'ipayit@enron.com'],
                            'phone': ['(713) 853-7127']}},
    {'filename': '4', 'scan_results': {'email': [], 'phone': ['713-345-4727']}},
    {'filename': '18', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '9', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '11', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '7',
        'scan_results': {   'email': ['8774890708@pagenetmessage.net'],
                            'phone': [' 8774890']}},
    {'filename': '6', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '1', 'scan_results': {'email': [], 'phone': []}},
    {   'filename': '10',
        'scan_results': {   'email': [],
                            'phone': ['713-857-2103', '281-531-0385']}},
    {   'filename': '19',
        'scan_results': {   'email': [],
                            'phone': ['877-550-7985', '402-896-2585']}},
    {   'filename': '8',
        'scan_results': {   'email': [],
                            'phone': ['605-272-5680', '605-272-5296']}},
    {'filename': '4', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '3', 'scan_results': {'email': [], 'phone': ['713-853-6414']}},
    {   'filename': '12',
        'scan_results': {   'email': ['cynthia.biggs@travelpark.com'],
                            'phone': [   '402 397 8000',
                                         '402 926 4036',
                                         ' 005 7033',
                                         '713 853-5842',
                                         '800 523-6586',
                                         '215 245-4707']}},
    {'filename': '2', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '13', 'scan_results': {'email': [], 'phone': []}},
    {'filename': '5', 'scan_results': {'email': [], 'phone': ['605-272-5680']}},
    {   'filename': '1',
        'scan_results': {'email': ['ulligan@enron.com'], 'phone': []}},
    {   'filename': '3',
        'scan_results': {'email': ["'sstojic@gbmdc.com"], 'phone': []}}]
```

(T7) Hashing with salt

- cryptographic package: https://www.pycryptodome.org/en/latest/src/hash/hash.html
- reverse lookup MD5 hash: https://md5.gromweb.com/

In [26]:
```python
from Crypto.Hash import MD5
```

In [27]:
```python
# MD5.new() only accepts binary values so by putting a 'b' in front of the string, it
easy_password_hash = MD5.new(b'password')

easy_password_hash.hexdigest()
```

Out[27]: '5f4dcc3b5aa765d61d8327deb882cf99'

Avalanche effect

In cryptography, the avalanche effect is the desirable property of cryptographic algorithms wherein if an input is changed slightly (for example, flipping a single bit), the output changes significantly (e.g., half the output bits flip).

This property provides the following benefits:

- prevent people from making predictions about the input, being given only the output
- makes it obvious that the data's integrity is affected even if it is only a 1 bit change.

In [28]:
```python
print('The MD5 hash of \'{}\' is {}'.format('password', MD5.new(b'password').hexdigest
print('The MD5 hash of \'{}\' is {}'.format('passwore', MD5.new(b'passwore').hexdigest

for i in range(10):
    plaintext_password = 'password{}'.format(i)
    print('The MD5 hash of \'{}\' is {}'.format(plaintext_password, MD5.new(plaintext_
```

```
The MD5 hash of 'password' is 5f4dcc3b5aa765d61d8327deb882cf99
The MD5 hash of 'passwore' is a826176c6495c5116189db91770e20ce
The MD5 hash of 'password0' is 305e4f55ce823e111a46a9d500bcb86c
The MD5 hash of 'password1' is 7c6a180b36896a0a8c02787eeafb0e4c
The MD5 hash of 'password2' is 6cb75f652a9b52798eb6cf2201057c73
The MD5 hash of 'password3' is 819b0643d6b89dc9b579fdfc9094f28e
The MD5 hash of 'password4' is 34cc93ece0ba9e3f6f235d4af979b16c
The MD5 hash of 'password5' is db0edd04aaac4506f7edab03ac855d56
The MD5 hash of 'password6' is 218dd27aebeccecae69ad8408d9a36bf
The MD5 hash of 'password7' is 00cdb7bb942cf6b290ceb97d6aca64a3
The MD5 hash of 'password8' is b25ef06be3b6948c0bc431da46c2c738
The MD5 hash of 'password9' is 5d69dd95ac183c9643780ed7027d128a
```

Reverse lookup hashes

Try searching for '5f4dcc3b5aa765d61d8327deb882cf99' in the internet. Are you able to find the original text from the hash?

Rainbow table attacks makes use of precomputed hashes tables to quickly identify your password from hashes.

To circumvent such attacks, add salt before hashing to significantly increase the size of the precomputed hash table needed to do reverse lookup of hashes. The longer the salt length, the larger the required precomputed hash table

Salt generation

Salt are additional characters that are randomly generated and added to the back of the original string before hashing to make dictionary attacks harder to execute (as the dictionary of known passwords would have to be multiple times bigger than dictionaries for unsalted password)

In [29]:
```python
import random

def generate_salt(char_set: str = None, length: int = 16)-> str:
    """Generates a random string of alphanumeric characters as salt

    Args:
        char_set: string containing the characters used for salt generation
        length: the length of salt required
    Return:
        randomly generated salt based on char_set and length

    """
    if char_set is None:
        char_set = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

    chars=[]
    for i in range(length):
        chars.append(random.choice(char_set))

    return "".join(chars)
```

In [30]:
```python
salt = generate_salt()

print(salt)
```

```
oZ3noyFLUxaWF6pr
```

In [31]:
```python
# append salt to the original easy password
easy_password_hash.update(salt.encode('ascii'))

easy_password_hash.hexdigest()
```

Out[31]: '562cc314626b51d93f15de1a40bfbb5e'

Reverse lookup hashes with salt

Try searching for the hash generated by adding salt to the value. Are you able to find the original text from the hash?

(T8) Field level encryption - Example using Symmetric Key Encryption

In [32]:
```python
import pandas as pd
import requests
import io
```

Get data from data.gov.sg

Note: to perform the download, you need to unrestrict the download rate limit for jupyter notebook by re-opening it with command:

`jupyter notebook --NotebookApp.iopub_data_rate_limit=1.0e10`

In [33]:
```python
api_response = requests.get('https://data.gov.sg/api/action/package_show?id=acra-info
                           ).json()
resources = api_response['result']['resources']

data_set_url = ''

for resource in resources:
    if resource['name'] == 'ACRA Information on Corporate Entities (\'X\')':
        data_set_url = resource['url']
        print('Data set URL:\n{}'.format(data_set_url))

acra_csv_data = io.StringIO(requests.get(data_set_url).content.decode('utf-8'))

acra_csv_df = pd.read_csv(acra_csv_data, dtype=str)

acra_csv_df.head()
```

```
Data set URL:
https://storage.data.gov.sg/acra-information-on-corporate-entities/resources/acra-info
rmation-on-corporate-entities-x-2020-09-03T10-32-09Z.csv
```

Out[33]:

	business_constitution_description	primary_ssic_description	primary_user_described_activity	street_name
0	Sole Proprietor	PUBS	na	CIRCULAR ROAD
1	Partnership	WHOLESALE TRADE OF A VARIETY OF GOODS WITHOUT ...	na	SENJA ROAD
2	na	CAFES AND COFFEE HOUSES	SNACKS AND BEVERAGE KIOSK OR SHOP	ROBINSON ROAD
3	Sole Proprietor	PASSENGER LAND TRANSPORT N.E.C. (EG PRIVATE CA...	na	CHOA CHU KANG CRESCENT
4	na	OTHER HOLDING COMPANIES	INVESTMENT	CECIL STREET

5 rows × 94 columns

If internet access is not available. Use local copy of dataset.

In [34]:
```python
# read ACRA companies table starting with letter X (taken from data.gov.sg)
# https://data.gov.sg/dataset/acra-information-on-corporate-entities

# acra_csv_df = pd.read_csv('acra-information-on-corporate-entities-x.csv', dtype=str

# acra_csv_df.head()
```

In [35]:
```python
from Crypto.Cipher import AES
from Crypto.Hash import MD5
from typing import Tuple

def aes_encrypt(value:bin, key:bin)-> Tuple[bin, bin]:
    """Function to perform AES encryption using SIV mode (as SIV mode doesn't require

    Args:
        value: plaintext value to be encrypted
        key: encryption key used to perform encryption

    Return:
        ciphertext and digest in binary data type

    """
    # convert key into MD5 as SIV mode requires minimum 32 byte key
    hash_key = MD5.new(key).hexdigest().encode('ascii')
    cipher = AES.new(hash_key, AES.MODE_SIV)

    return cipher.encrypt_and_digest(value)

def aes_decrypt(ciphertext:bin, digest:bin, key:bin) -> bin:
    """Function to perform AES decryption using SIV mode (as SIV mode doesn't require

    Args:
        ciphertext: ciphertext to be decrypted
        digest: digest produced from encryption
        key: encryption key used to perform decryption

    Return:
        ciphertext and digest in binary data type

    """
    # convert key into MD5 as SIV mode requires minimum 32 byte key
    hash_key = MD5.new(key).hexdigest().encode('ascii')
    cipher = AES.new(hash_key, AES.MODE_SIV)

    return cipher.decrypt_and_verify(ciphertext, digest)
```

Perform encryption on simple message

In [36]:
```python
secret_key = b'secret'

message = b'secret msg'

ciphertext_1, digest_1 = aes_encrypt(message, secret_key)

print('My encrypted message is:\n{}'.format(ciphertext_1))
print('\nMy encrypted message digest is:\n{}'.format(digest_1))

print('\nMy decrypted secret message is:\n{}'.format(aes_decrypt(ciphertext_1, digest_
```

```
My encrypted message is:
b'\xe9\xf9\xc4\x83\xa45\xf4AC\x1d'

My encrypted message digest is:
b'>Z\xf9\xe6\xe8}\x85\x043\xb7\x05\xb4\xac\xca\x84\x8f'

My decrypted secret message is:
b'secret msg'
```

Perform encryption on field 'entity_name'

In [37]:
```python
encrypted_df = acra_csv_df.copy()

encrypted_df['entity_name'] = encrypted_df['entity_name'].apply(lambda x: aes_encrypt

encrypted_df.head()
```

Out[37]:

	business_constitution_description	primary_ssic_description	primary_user_described_activity	street_name
0	Sole Proprietor	PUBS	na	CIRCULAR ROAD
1	Partnership	WHOLESALE TRADE OF A VARIETY OF GOODS WITHOUT ...	na	SENJA ROAD
2	na	CAFES AND COFFEE HOUSES	SNACKS AND BEVERAGE KIOSK OR SHOP	ROBINSON ROAD
3	Sole Proprietor	PASSENGER LAND TRANSPORT N.E.C. (EG PRIVATE CA...	na	CHOA CHU KANG CRESCENT
4	na	OTHER HOLDING COMPANIES	INVESTMENT	CECIL STREET

5 rows × 94 columns

Perform decryption on field 'entity_name'

In [38]:
```python
encrypted_df['entity_name'].apply(lambda x: aes_decrypt(x[0], x[1], secret_key).decod
```

```
Out[38]:
0                          X
1                X & B TRADING
2        X & E PASSION PTE. LTD.
3               X & F SERVICES
4    X & H INTERNATIONAL PTE. LTD.
Name: entity_name, dtype: object
```

(T9) Tokenization

In [39]:
```python
# create functions for generating and storing tokens
import random

class TokenDict:
    """Class for storing randomly generated tokens

        Typical usage example:
            token_dict = TokenDict()

    """
    def __init__(self):
        # protected attributes
        self.__token_dict = {}

    def generate_token(self) -> str:
        NUMBERS = "0123456789"
        chars=[]
        for i in range(16):
            chars.append(random.choice(NUMBERS))

        token = "".join(chars)
        return token

    def tokenize_value(self, real_value:str) -> str:
        if self.retrieve_token(real_value) is not None:
            raise ValueError('value is already in token dictionary')
        token=self.generate_token()
        while token in self.__token_dict.keys() or token==real_value:
            token = self.generate_token()
        self.__token_dict[token]=real_value
        return token

    def retrieve_value(self, token:str) -> str:
        return self.__token_dict[token]

    def retrieve_token(self, real_value:str) -> str:
        for k,v in self.__token_dict.items():
            if v == real_value:
                return k
        return None
```

Initialise token dictionary class and enter values into the dictionary

In [40]:
```python
token_dict_1 = TokenDict()

token_1 = token_dict_1.tokenize_value('1234567812345678')
print('First Token: {}'.format(token_1))
token_2 = token_dict_1.tokenize_value('8888888888888880')
print('Second Token: {}'.format(token_2))
```

```
First Token: 1963812867436780
Second Token: 9256726201869807
```

Retrieve token from value and vice versa

In [41]:
```python
print('Retrieve second value from second token: {}'.format(token_dict_1.retrieve_value

print('Retrieve first token from first value: {}'.format(token_dict_1.retrieve_token(
```

```
Retrieve second value from second token: 8888888888888880
Retrieve first token from first value: 1963812867436780
```

Try to extract dictionary from token dictionary class

In [42]:
```python
# prefix attribute with __ to protect it from access outside of the class

try:
    new_dict = token_dict_1.__token_dict
except AttributeError as e:
    print('Attribute Error: {}'.format(str(e)))
```

```
Attribute Error: 'TokenDict' object has no attribute '__token_dict'
```

(T13) Data File Integrity Verification - Example using Asymmetric Key Encryption

- cryptographic package: https://www.pycryptodome.org/en/latest/src/public_key/rsa.html

Public-key cryptosystems are convenient in that they do not require the sender and receiver to share a common secret in order to communicate securely (among other useful properties).

However, they often rely on complicated mathematical computations and are thus generally much more inefficient than comparable symmetric-key cryptosystems. In many applications, the high cost of encrypting long messages in a public-key cryptosystem can be prohibitive.

The asymmetric encryption algorithm only allows encryption of data smaller than the key length. To break large files into chunks smaller than key length to perform asymmetric encryption would be computationally inefficient.

However, this can be addressed by hybrid systems by using a combination of both, where a symmetric session key is used to encrypt the file and the asymmetric encryption be done on the symmetric session key.

Define functions

In [43]:
```python
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Signature import pkcs1_15
from Crypto.Hash import MD5
import base64
from typing import Tuple

def generate_keys() -> Tuple[RSA.RsaKey, RSA.RsaKey] :
    """generate asymmetric key pairs

    Args: None
    Return:
        private key, public key

    """
    modulus_length = 1024

    key = RSA.generate(modulus_length)

    pub_key = key.publickey()

    return key, pub_key

def asym_encrypt(plaintext_message: bin, public_key: RSA.RsaKey) -> bin:
    """Asymmetric encryption function

    Note: encryption can only be done using public key for security reasons.
    For signing, please refer to Crypto.Signature

    Args:
        plaintext_message: plaintext message that you want to encrypt
        public_key: recipient's public key to be used for encrypting plaintext message

    Return:
        ciphertext_message

    """
    encryptor = PKCS1_OAEP.new(public_key)
    encrypted_msg = encryptor.encrypt(plaintext_message)
    encoded_encrypted_msg = base64.b64encode(encrypted_msg)
    return encoded_encrypted_msg

def asym_decrypt(ciphertext_message: bin, private_key: RSA.RsaKey) -> bin:
    """Asymmetric decryption function

    Note: decryption can only be done using private key for security reasons.
    For signing, please refer to Crypto.Signature

    Args:
        ciphertext_message: encrypted message sent from sender encrypted with recipien
        private_key: recipient's private key to be used for decrypting ciphertext mess


    """
    encryptor = PKCS1_OAEP.new(private_key)
    decoded_encrypted_msg = base64.b64decode(ciphertext_message)
    decoded_decrypted_msg = encryptor.decrypt(decoded_encrypted_msg)
    return decoded_decrypted_msg


def generate_session_key(char_set: str = None, length: int = 16)-> str:
    """Generates a random string of alphanumeric characters as session key for AES enc

    Args:
        char_set: string containing the characters used for session key generation
        length: the length of session key required
    Return:
        randomly generated salt based on char_set and length

    """
    if char_set is None:
        char_set = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

    chars=[]
    for i in range(length):
        chars.append(random.choice(char_set))

    return "".join(chars)
```

Perform encryption and decryption on simple message

In [44]:
```python
private, public = generate_keys()
message = b'Hello world'
encoded = asym_encrypt(message, public)
asym_decrypt(encoded, private)
```

Out[44]: b'Hello world'

Perform public key cryptography example on files

In [45]:
```python
# prepare key pairs

# sender key pairs
s_private, s_public = generate_keys()

# receiver key pairs
r_private, r_public = generate_keys()
```

Sender's perspective

Has knowledge of:

- sender's private key
- sender's public key
- receiver's public key

Perform encryption on file

As asymmetric encryption can only work for messages smaller than the asymmetric key length, we need to use symmetric encryption to encrypt the file and use the asymmetric encryption to encrypt the symmetric encryption key (also known as the **session key**).

In [46]:
```python
# prepare files and filenames
filename = 'acra-information-on-corporate-entities-x.csv'
signed_md5_filename = 'signed_md5'
encrypted_data_filename = 'encrypted_file'
encrypted_key_filename = 'encrypted_key'

# generate symmetric key for file encryption. This is also known as a session key as
symm_session_key = generate_session_key().encode('ascii')
```

Start symmetric encryption using session key and write to file

In [47]:
```python
# read file to be encrypted. We will use the same data.gov.sg acra companies dataset.
with open(filename, 'rb') as f:
    data = f.read()

# write encrypted data into file
with open(encrypted_data_filename, 'wb') as f:
    cipher, digest = aes_encrypt(data, symm_session_key)
#     print(len(digest))
#     print(len(cipher))
    f.write(digest)
    f.write(cipher)

print('File {} is encrypted using AES and saved as {}'.format(filename,encrypted_data_
```

```
File acra-information-on-corporate-entities-x.csv is encrypted using AES and saved as
encrypted_file
```

Start asymmetric encryption of session key using recipient's public key and write into file

=> only recipient's private key can be used to decrypt the encrypted session key

In [48]:
```python
# write encrypted session key into file
with open(encrypted_key_filename, 'wb') as f:
    f.write(asym_encrypt(symm_session_key, r_public))


print('Session is encrypted using recipient\'s public key and saved as {}'.format(enc
```

```
Session is encrypted using recipient's public key and saved as encrypted_key
```

Create file hash for file integrity check by recipient and sign it with sender's private key

In [49]:
```python
# create file digest
md5_digest = MD5.new(data).digest()


# write md5 hash of data to file
with open(signed_md5_filename, 'wb') as f:
#     print(len(md5_digest))
#     print(len(pkcs1_15.new(s_private).sign(MD5.new(md5_digest))))
    f.write(md5_digest)
    f.write(pkcs1_15.new(s_private).sign(MD5.new(md5_digest)))


print('Hash digest of file is signed using sender\'s private key and saved as {}'.for
```

```
Hash digest of file is signed using sender's private key and saved as signed_md5
```

Receiver's perspective

Has knowledge of:

- receiver's private key
- receiver's public key
- sender's public key

Decrypt session key using receiver's private key

In [50]:
```python
# read encrypted session key from file and decrypt it
with open(encrypted_key_filename, 'rb') as f:
    decrypted_file_symm_key=asym_decrypt(f.read(), r_private)

print('Successfully decrypted session key. \nSession Key:{}'.format(decrypted_file_sym
```

```
Successfully decrypted session key.
Session Key:b'r2vtYzkSAfPrruV5'
```

Use session key to decrypt file

In [51]:
```python
decrypted_filename = 'decrypted.csv'

# read encrypted data with decrypted symmetric key
with open(encrypted_data_filename, 'rb') as f:
    sym_digest = f.read(16)
    f.seek(16)
    sym_data = f.read()
    decrypted_data = aes_decrypt(sym_data, sym_digest, decrypted_file_symm_key)

# write decrypted data into csv file
with open(decrypted_filename, 'wb') as f:
    f.write(decrypted_data)

decrypted_df = pd.read_csv(decrypted_filename, dtype=str)
decrypted_df.head()
```

Out[51]:

	business_constitution_description	primary_ssic_description	primary_user_described_activity	street_name
0	Sole Proprietor	PUBS	na	CIRCULAR ROAD
1	Partnership	WHOLESALE TRADE OF A VARIETY OF GOODS WITHOUT ...	na	SENJA ROAD
2	na	CAFES AND COFFEE HOUSES	SNACKS AND BEVERAGE KIOSK OR SHOP	ROBINSON ROAD
3	Sole Proprietor	PASSENGER LAND TRANSPORT N.E.C. (EG PRIVATE CA...	na	CHOA CHU KANG CRESCENT
4	na	OTHER HOLDING COMPANIES	INVESTMENT	CECIL STREET

5 rows × 94 columns

Validate sender identity

In [52]:
```python
# read signed md5 digest from file
with open(signed_md5_filename, 'rb') as f:
    received_md5_digest = f.read(16)
    f.seek(16)
    received_digest_signature = f.read()

# verify signature of md5 digest to validate sender identity
try:
    pkcs1_15.new(s_public).verify(MD5.new(received_md5_digest),received_digest_signatu
    print('Signature is valid. Sender identity confirmed')
except (ValueError, TypeError):
    print('Signature is not valid. Sender identity invalid')
```

```
Signature is valid. Sender identity confirmed
```

Validate file integrity

In [53]:
```python
# perform md5 hash on data and compare with received md5 hash, to verify file integri
decrypted_data_digest = MD5.new(decrypted_data).digest()

if received_md5_digest == decrypted_data_digest:
    print('File integrity verified')
else:
    print('File integrity failed')
```

```
File integrity verified
```

Format preserving encryption

In [54]:
```python
# library for format preserving encryption using ffx method
import pyffx

# create fpe object to perform encryption and decryption
e = pyffx.Integer(b'secret-key', length=16)
```

In [55]:
```python
fpe_ciphertext = e.encrypt('1234567812345678')

fpe_ciphertext
```

Out[55]: 9845110889077199

In [56]:
```python
e.decrypt(fpe_ciphertext)
```

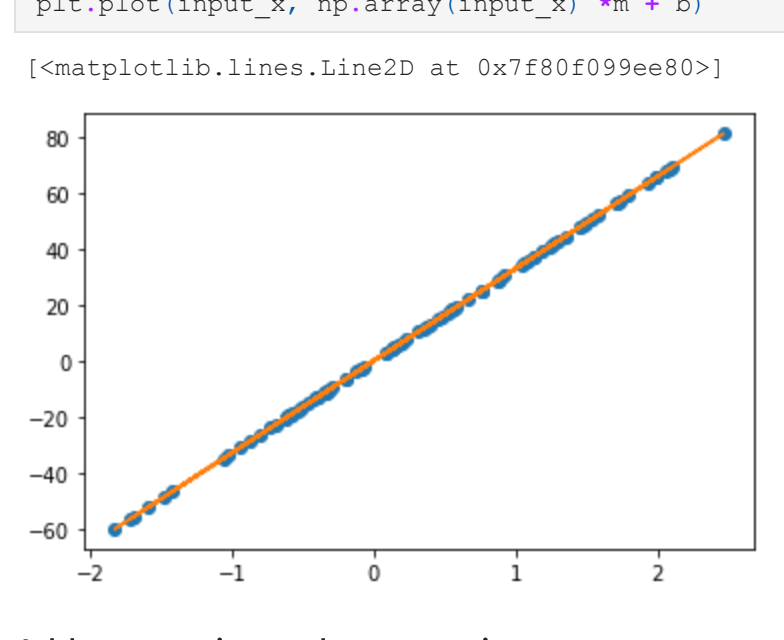
Out[56]: 1234567812345678

Synthetic Data Generation

Generate data points for linear regression

In [57]:
```python
from sklearn import datasets
import matplotlib.pyplot as plt
import numpy as np

# generate regression dataset
inputs, outputs = datasets.make_regression(n_features=1, noise=0.0)

# reshape the x values
input_x = np.concatenate(inputs, axis = 0)

# get gradient and coefficients
m, b = np.polyfit(input_x, outputs, 1)

# scatter plot
plt.plot(input_x, outputs, 'o')


# plot regression line
plt.plot(input_x, np.array(input_x) *m + b)
```

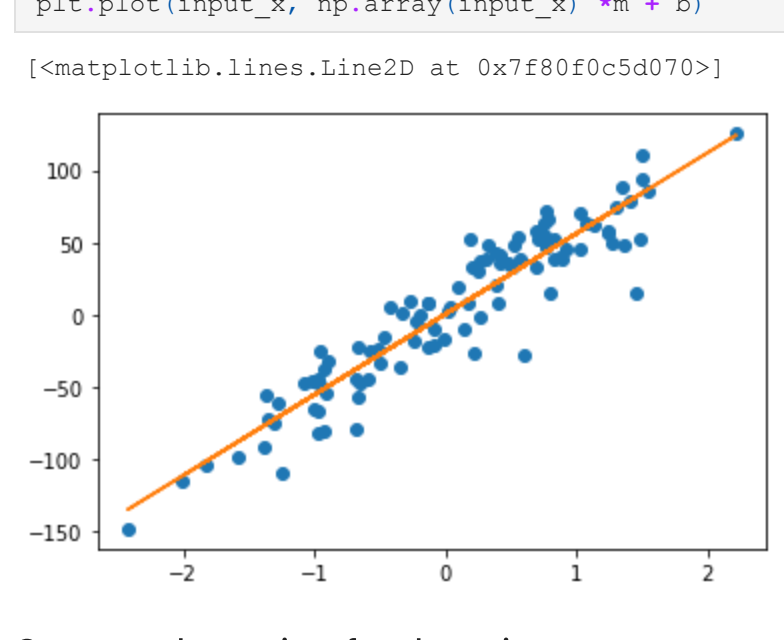
Out[57]: [<matplotlib.lines.Line2D at 0x7f80f099ee80>]



Add some noise to the regression

In [58]:
```python
# generate regression dataset
inputs, outputs = datasets.make_regression(n_features=1, noise=20.0)

# reshape the x values
input_x = np.concatenate(inputs, axis = 0)

# get gradient and coefficients
m, b = np.polyfit(input_x, outputs, 1)

# scatter plot
plt.plot(input_x, outputs, 'o')


# plot regression line
plt.plot(input_x, np.array(input_x) *m + b)
```

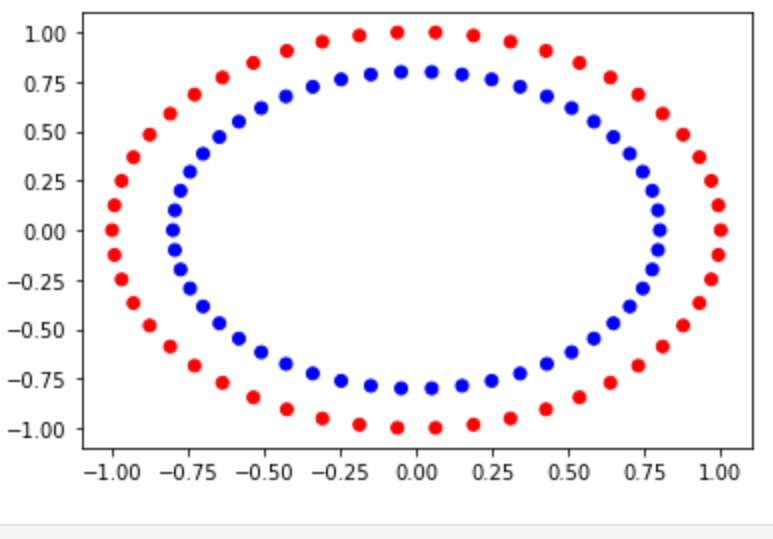
Out[58]: [<matplotlib.lines.Line2D at 0x7f80f0c5d070>]



Generate data points for clustering

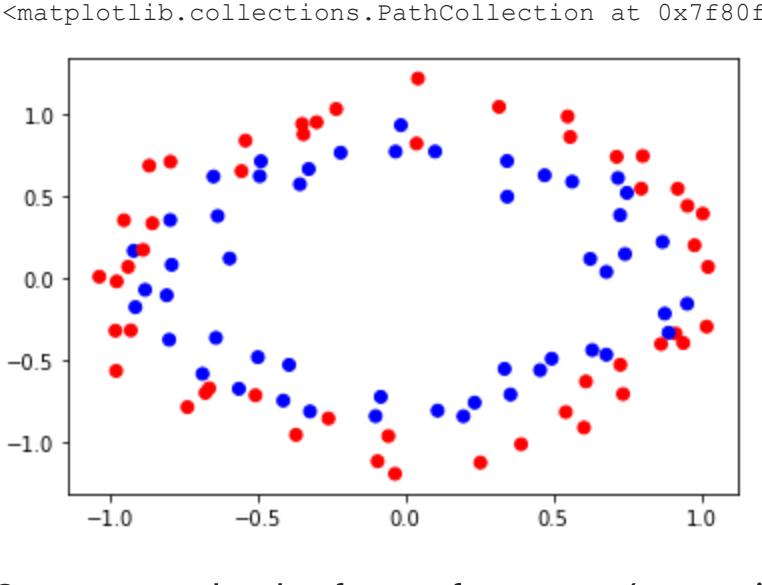
In [59]:
```python
# generate circles dataset
data, labels = datasets.make_circles()

# split axis
x, y = zip(*data)

# define color map for labels
LABEL_COLOR_MAP = {0 : 'r',
                   1 : 'b'}

#
label_color = [LABEL_COLOR_MAP[l] for l in list(labels)]


plt.scatter(x, y, c=label_color)
```

Out[59]: <matplotlib.collections.PathCollection at 0x7f80f0eef280>



In [60]:
```
data, labels = datasets.make_circles(noise=0.09)

x, y = zip(*data)

LABEL_COLOR_MAP = {0 : 'r',
                   1 : 'b'}


label_color = [LABEL_COLOR_MAP[l] for l in list(labels)]


plt.scatter(x, y, c=label_color)
```

Out[60]: `<matplotlib.collections.PathCollection at 0x7f80f0a3dac0>`



Generate pandas dataframe of customer (categorical/attribute) data

- Note 1: when doing package installation, do downgrade faker package `pip install Faker==2.0.5` if not you will get an error as described in https://github.com/tirthajyoti/pydbgen/issues/16
- Note 2: if you encounter `SSL: CERTIFICATE_VERIFY_FAILED error` do follow instructions here to fix

`If you're using macOS go to Macintosh HD > Applications > Python3.6 folder (or whatever version of python you're using) > double click on "Install Certificates.command" file.`

In [61]:
```
import pydbgen
# https://github.com/tirthajyoti/pydbgen
from pydbgen import pydbgen
myDB=pydbgen.pydb()

synth_df = myDB.gen_dataframe(100,['ssn','name','city','phone','date', 'zipcode', 'jo

synth_df.head()
```

Out[61]:

	ssn	name	city	phone-number	date	zipcode	job_title
0	363-68-3795	Marcus Nguyen	Scott City	693-379-2636	1999-05-23	93035	Environmental consultant
1	581-34-8870	Kimberly Davis	Gracey	958-745-9607	1997-11-30	81349	Civil Service fast streamer
2	033-88-0392	Carol Young	La Moille	475-242-0524	1973-03-11	79846	Armed forces technical officer
3	874-59-7154	Thomas Johnson	Miami Lakes	297-233-7340	1976-11-26	86730	Environmental education officer
4	454-93-1995	Jessica Hess	Fern Creek	524-985-2616	1994-10-17	02340	Scientist, marine

Homomorphic Encryption

Drawing

Fully homomorphic encryption is where any function applied on the ciphertext when decrypted would give the same results as the function applied on the plaintext value.

Where only a few types of functions are supported, the type of encryption is called partially homomorphic encryption.

In this example we explore the partially homomorphic encryption cryptosystem called paillier.

In [62]:
```
from phe import paillier
import pprint
import numpy as np

pp = pprint.PrettyPrinter(indent=4)

# generate key pair for encryption and decryption
public_key, private_key = paillier.generate_paillier_keypair()

# perform encryption
secret_number_list = [3.141592653, 300, -4.6e-12]
encrypted_number_list = [public_key.encrypt(x) for x in secret_number_list]

# perform functions
encrypted_number_list_multiply = [x * 5 for x in encrypted_number_list]
encrypted_number_list_addition = [x + 5 for x in encrypted_number_list]
encrypted_number_list_mean = np.mean(encrypted_number_list)


# show results
print('Original data:\n {}'.format(secret_number_list))

print('\nMultiplication example (x * 5):')
pp.pprint(encrypted_number_list_multiply)
print('\ndecrypts to ')
pp.pprint([private_key.decrypt(x) for x in encrypted_number_list_multiply])

print('\nAddition example (x + 5):')
pp.pprint(encrypted_number_list_addition)
print('\ndecrypts to ')
pp.pprint([private_key.decrypt(x) for x in encrypted_number_list_addition])


print('\nMean example:\n{}\n\ndecrypts to\n{}'.format(
    encrypted_number_list_mean, private_key.decrypt(encrypted_number_list_mean)))
```

```
Original data:
 [3.141592653, 300, -4.6e-12]

Multiplication example (x * 5):
[   <phe.paillier.EncryptedNumber object at 0x7f80f293a670>,
    <phe.paillier.EncryptedNumber object at 0x7f80f293a460>,
    <phe.paillier.EncryptedNumber object at 0x7f80f293a6a0>]

decrypts to
[15.707963265, 1500, -2.2999999999999998e-11]

Addition example (x + 5):
[   <phe.paillier.EncryptedNumber object at 0x7f80f293a790>,
    <phe.paillier.EncryptedNumber object at 0x7f80f293a8b0>,
    <phe.paillier.EncryptedNumber object at 0x7f80f293a850>]

decrypts to
[8.141592653, 305, 4.9999999999954]

Mean example:
<phe.paillier.EncryptedNumber object at 0x7f80f293a970>

decrypts to
101.04719755099846
```

In [ ]: