STEGANOGRAPHY

HIDING DATA WITHIN DATA

@manalco

@cubosensei

www.manalco.co

MANUEL ALVARADO

Web Developer, Systems Engineer, MSc. Information Security

Agenda

Basics

- Confidenciality
- Integrity
- Availability
- Cryptography

Steganography

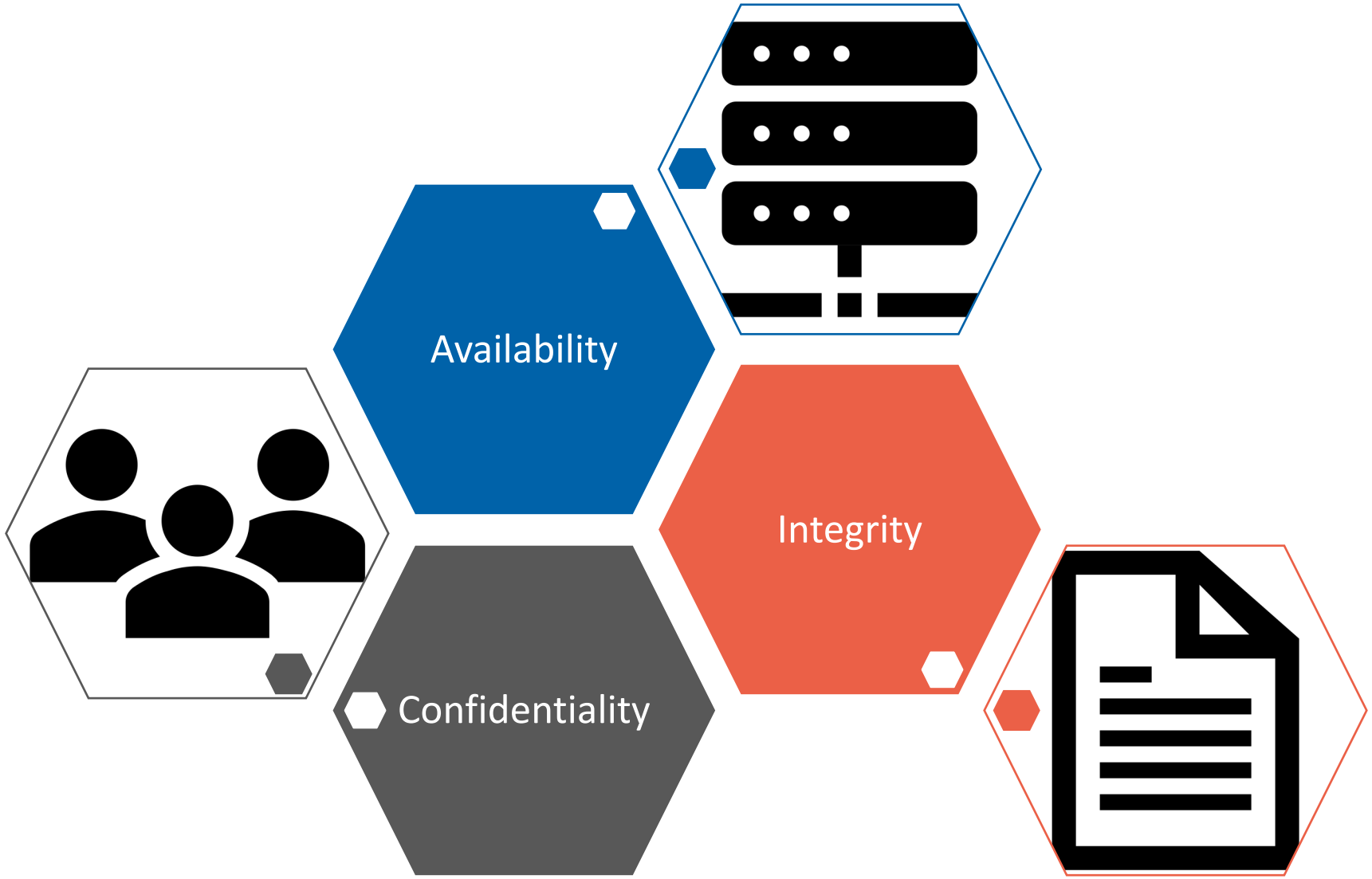
- What it is
- What it is not
- What it is used for

Examples

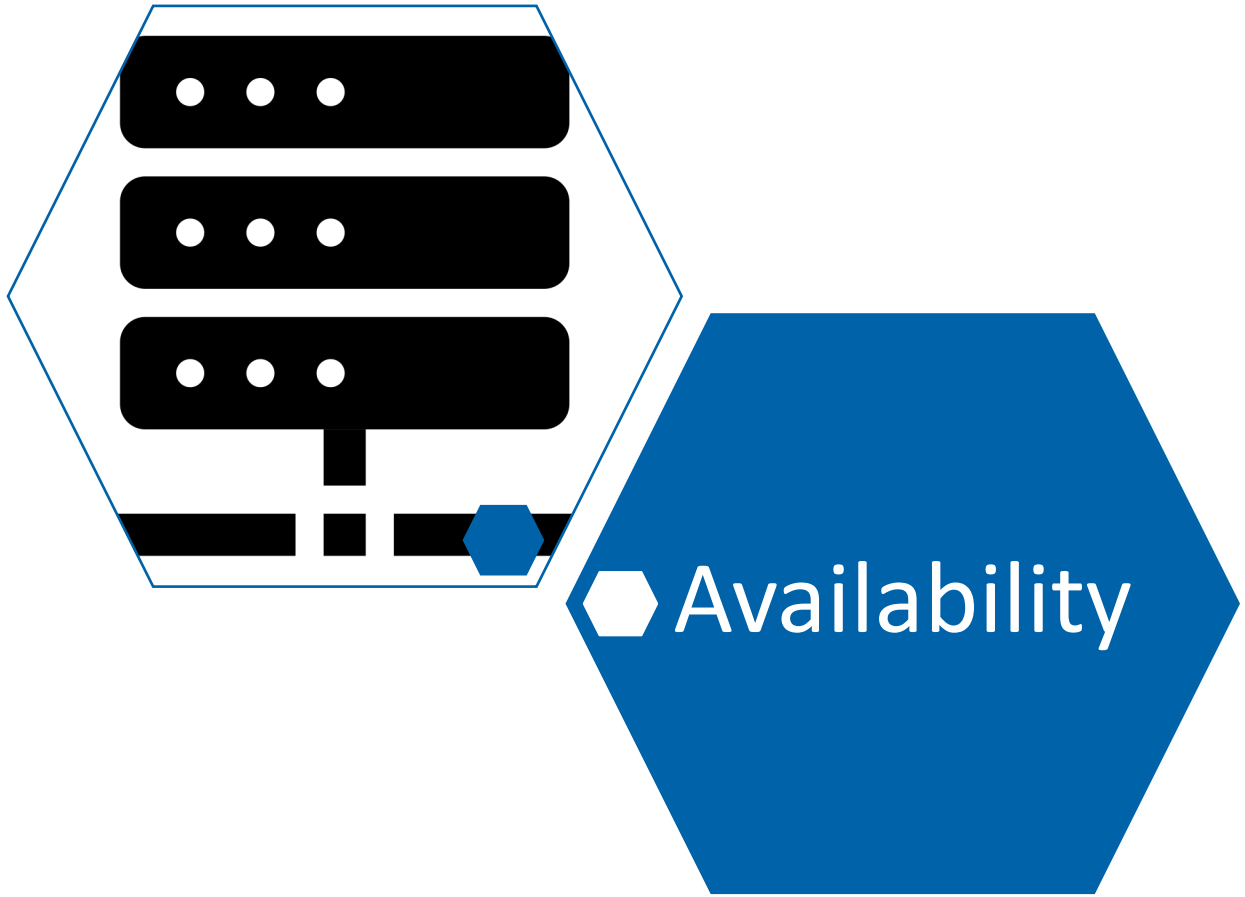
- Images
- Sound

Community

Basics



CIA Triad: Availability



- Hardware/Service oriented.
- If the resource is not available when needed it is a problem.

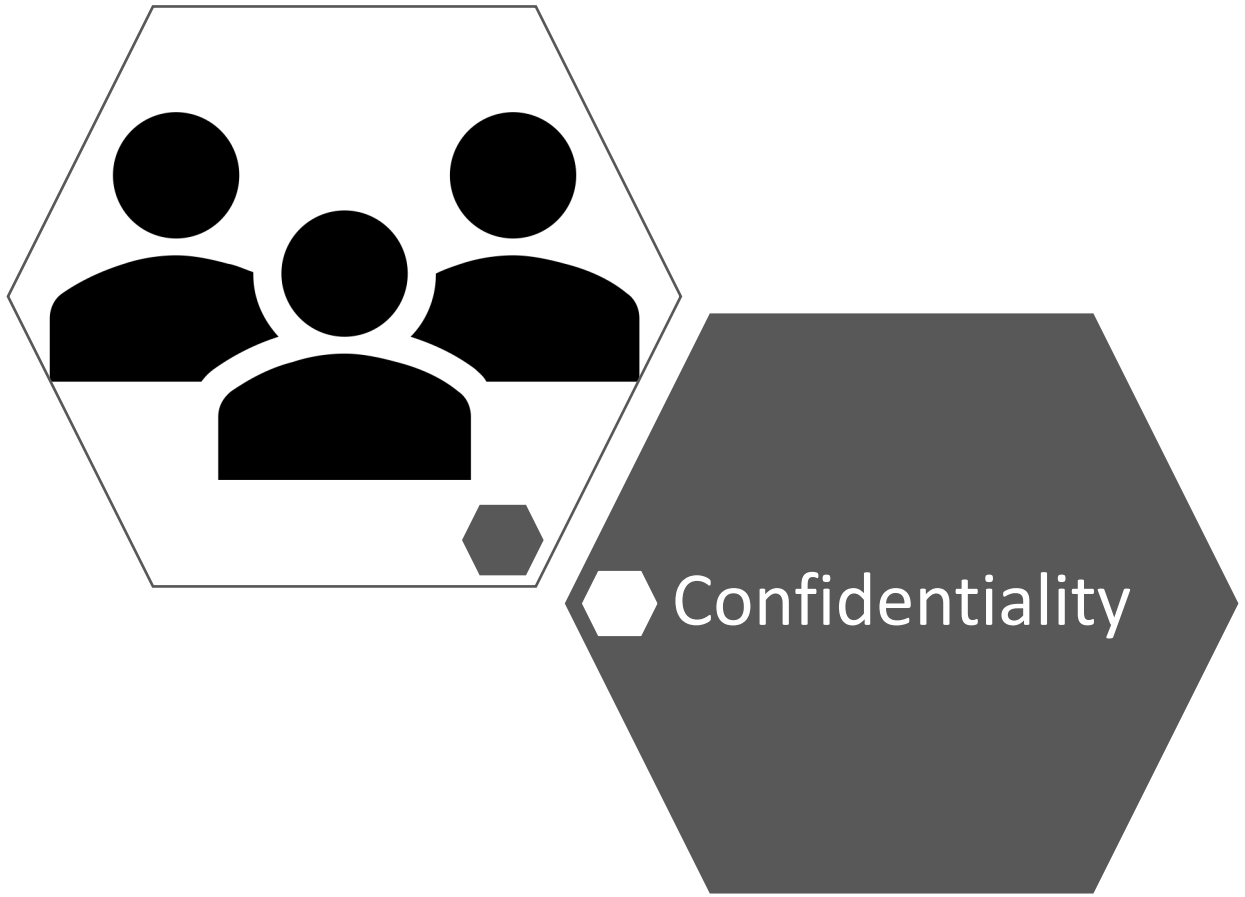
CIA Triad: Integrity

Integrity

Only authorized users are able to _____ the asset.

- Update/Modify
  - Only in determined ways.

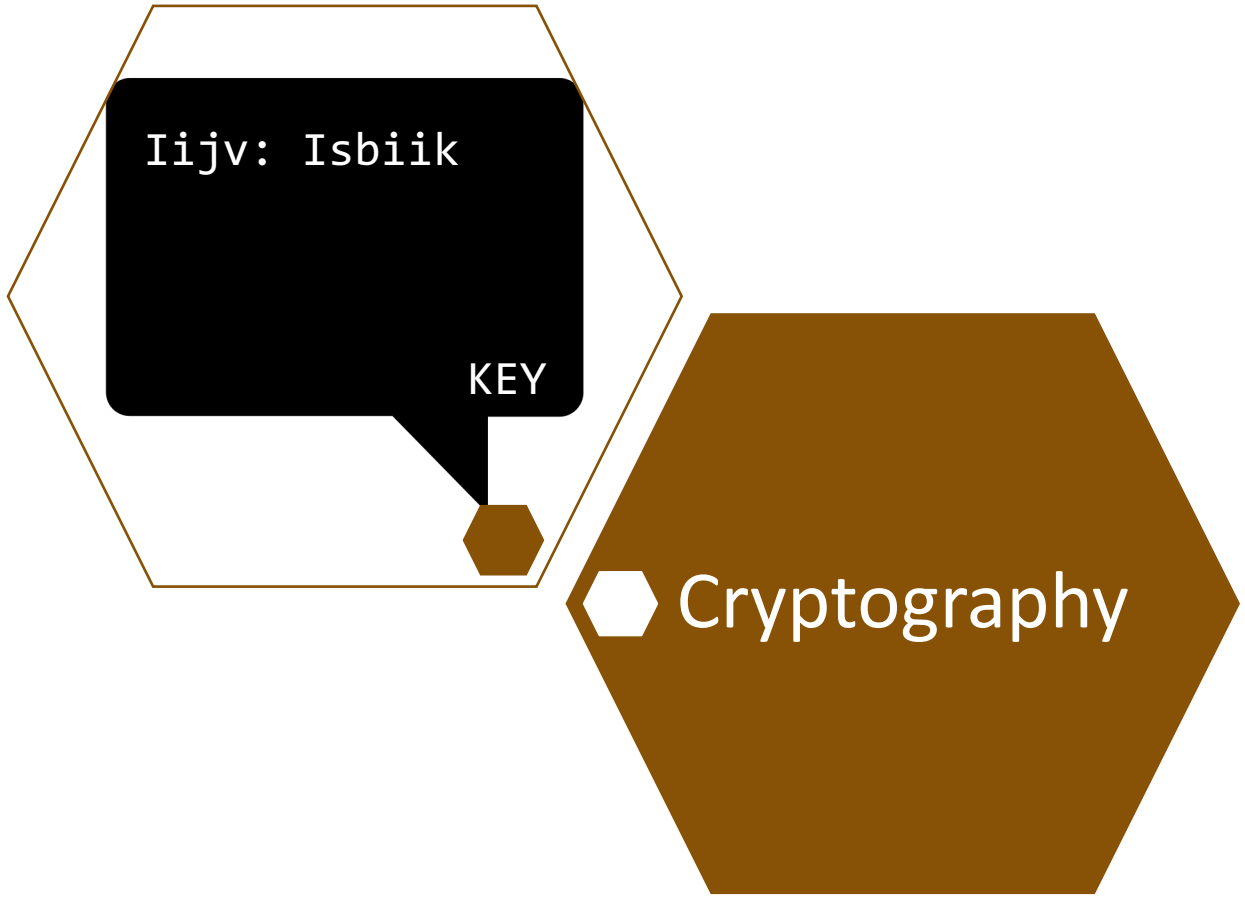
CIA Triad: Confidentiality



Only authorized users are able to _____ the asset.

- Access
- Read
- Print
- Even know of its existence

Cryptography



- Secure communications.
- Information confidentiality.
- Enigma Machine.

STEGANOGRAPHY

(NOW FOR REAL)

What it is

PRACTICE

"The practice of concealing messages or information within other non-secret text or data."

-Oxford Dictionary

ART

"Steganography is the art of hiding information within a disclosed element."

-?

What it is NOT

METHOD

"A particular procedure for accomplishing or approaching something, especially a systematic or established one."

-Oxford Dictionary

TECHNIQUE

"A way of carrying out a particular task, especially the execution or performance of an artistic work or a scientific procedure."

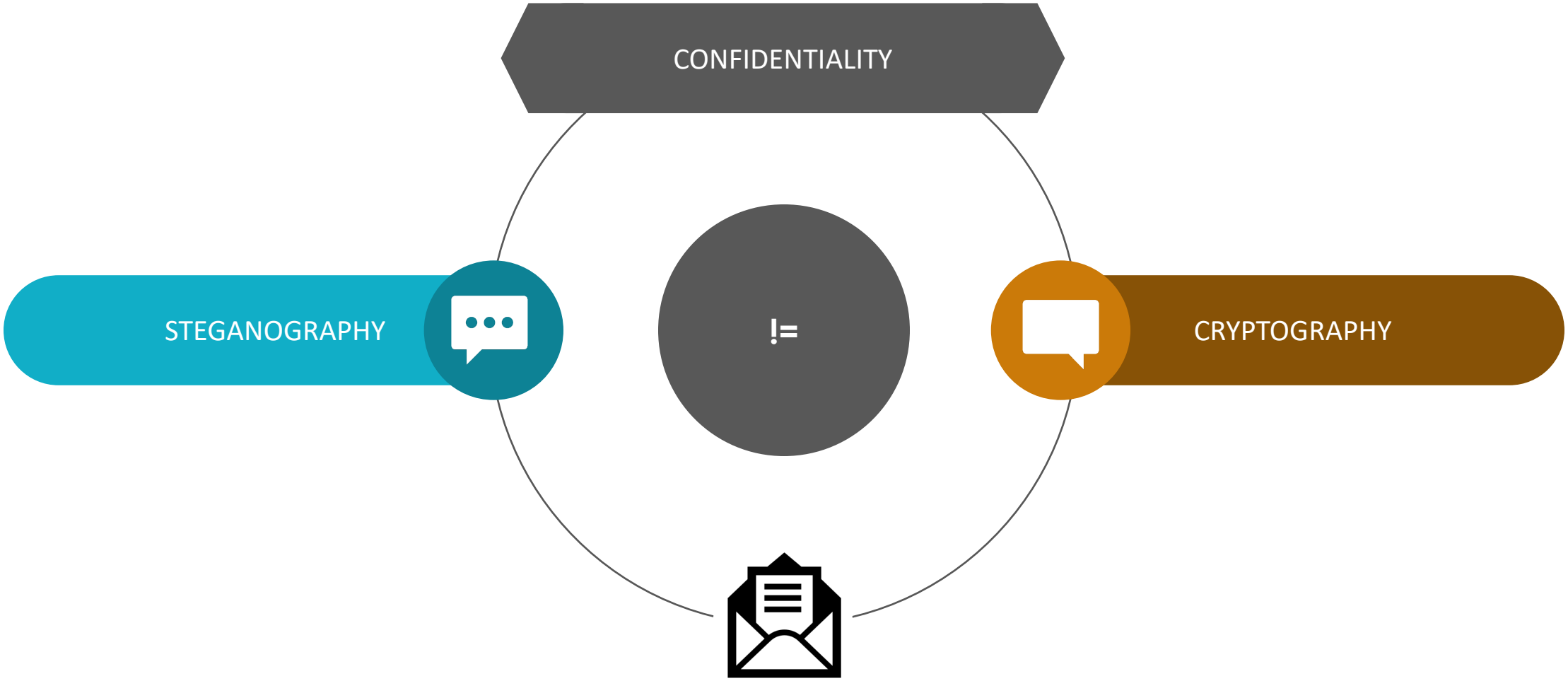
-Oxford Dictioinary

CRYPTOGRAPHY

"Is the practice and study of techniques for secure communication."

-Wikipedia

What it is NOT



What it is for

ACTUAL MESSAGE FROM GERMAN SPY IN WWII

Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by products, ejecting suets and vegetable oils.

What it is for

ACTUAL MESSAGE FROM GERMAN SPY IN WWII

Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by products, ejecting suets and vegetable oils.

TAKING THE SECOND LETTER FROM EACH WORD

Pershing sails from NY June 1.

Hiding Data

STEGANOGRAPHY WITH IMAGE FILES

An image is just a set of pixels.

- Each pixel represents RGB with 3 bytes.
  - (0,0,0) is black and (255,255,255) is white.
- So it makes it possible to hide the info in the **Least Significant Bit (LSB)** and the variation may be imperceptible.

STEGANOGRAPHY WITH AUDIO FILES

MIDI files have 16 channels (0-15) for tracks. Usually a file does not use all the channels, so it makes it possible to code a message in an empty channel.

Examples

Hiding Data: Least Significant Bit

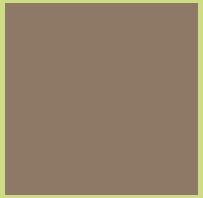
ORIGINAL

Having two pixels from an RGB image

MODIFIED

It is possible to hide the message 110101 in the least significant bit (red)

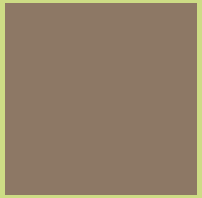
R=132, G=134, B=137
(10000100, 10000110, 10001001)



R=141, G=121, B=101
(10001101, 01111001, 01100101)

110101

R=133, G=135, B=136
(10000101, 10000111, 10001000)



R=141, G=120, B=101
(10001101, 01111000, 01100101)

Examples

Hiding Data: Inside Image Files

IMAGE WITHOUT MESSAGE

The original file

IMAGE WITH MESSAGE

This image has the message:

`HELLO WORLD`

Examples

Hiding Data: Inside Image Files

```
from stegano import lsb

secret = lsb.hide("~/mario1.png", "HELLO WORLD")
secret.save("~/mario2.png")
```

ENCODE

```
from stegano import lsb

clear_message = lsb.reveal("~/mario2.png")
print(clear_message)
```

DECODE

Examples

Hiding Data: Inside Image Files

```
import bitarray
import base64
import sys
from PIL import Image

file_name = sys.argv[1]
message = sys.argv[2]
encoded_message = base64.b64encode(message)

ba = bitarray.bitarray()
ba.frombytes(encoded_message.encode('utf-8'))
bit_array = [int(i) for i in ba]

im = Image.open(file_name)
width, height = im.size
pixels = im.load()
```

```
i = 0
for x in range(0,width):
    r,g,b = pixels[x,0]
    #Default values in case no bit has to be modified
    new_bit_red_pixel = 255
    new_bit_green_pixel = 255
    new_bit_blue_pixel = 255

    if i<len(bit_array):
        r_bit = bin(r)
        r_last_bit = int(r_bit[-1])
        r_new_last_bit = r_last_bit & bit_array[i]
        new_bit_red_pixel = int(r_bit[:-1]+str(r_new_last_bit),2)
        i += 1

    if i<len(bit_array):
        g_bit = bin(g)
        g_last_bit = int(g_bit[-1])
        g_new_last_bit = g_last_bit & bit_array[i]
        new_bit_green_pixel = int(g_bit[:-1]+str(g_new_last_bit),2)
        i += 1

    if i<len(bit_array):
        b_bit = bin(b)
        b_last_bit = int(b_bit[-1])
        b_new_last_bit = b_last_bit & bit_array[i]
        new_bit_blue_pixel = int(b_bit[:-1]+str(b_new_last_bit),2)
        i += 1

    pixels[x,0] = (new_bit_red_pixel,new_bit_green_pixel,new_bit_blue_pixel)

im.save(file_name.replace('.png', '_encoded.png'))
```

from https://www.boiteaklou.fr/Steganography-Least-Significant-Bit.html

ENCODE

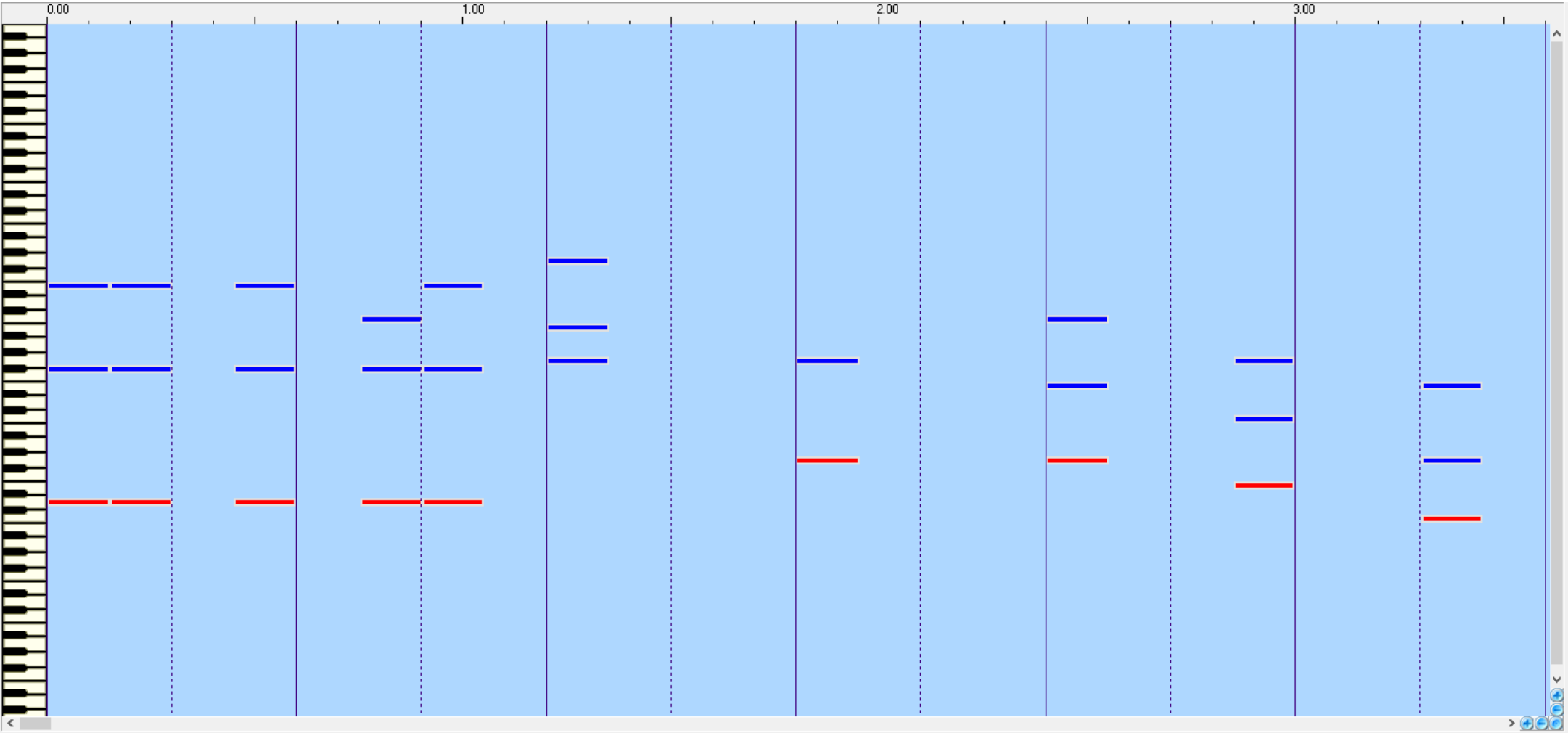
Hiding Data: Inside Image Files

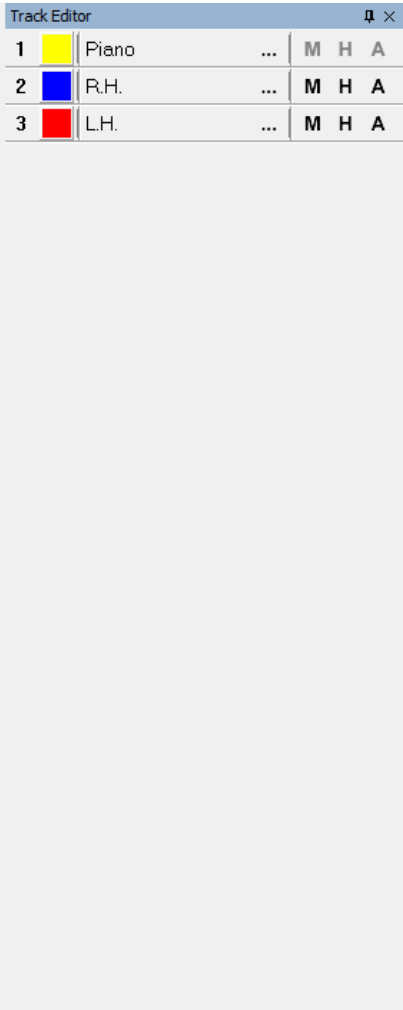
```python
import base64
from PIL import Image

image = Image.open("out.png")

extracted = ''

pixels = image.load()
for x in range(0,image.width):
    r,g,b = pixels[x,0]
    # Store LSB of each color channel of each pixel
    extracted += bin(r)[-1]
    extracted += bin(g)[-1]
    extracted += bin(b)[-1]

chars = []
for i in range(len(extracted)/8):
    byte = extracted[i*8:(i+1)*8]
    chars.append(chr(int(''.join([str(bit) for bit in byte]), 2)))

flag = base64.b64decode(''.join(chars))
print flag
```

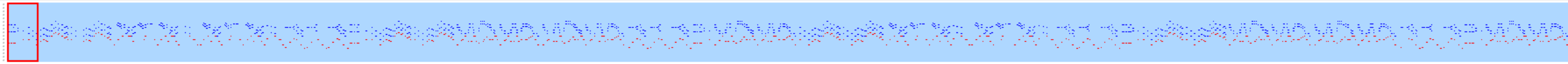
from https://www.boiteaklou.fr/Steganography-Least-Significant-Bit.html

DECODE

Hiding Data: Inside Audio Files







Examples

Hiding Data: Inside Audio Files

```
import sys
import midi

file_name = sys.argv[1]
message = sys.argv[2]
midi_file = midi.read_midifile(file_name)
track = midi.Track( )
channels = {}
channel = 0
```

```
#Look for a free channel
for i in midi_file:
    for j in i:
        try:
            j.channel
        except:
            pass
        else:
            if channels.has_key(j.channel):
                channels[j.channel] += 1
            else:
                channels[j.channel] = 1

for c in range(0,15):
    if not channels.has_key(c):
        channel = c
        break
```

ENCODE

Examples

Hiding Data: Inside Audio Files

```
message = message.upper()
message = "-msg-"+message

#Encode the message in the selected channel
for character in list(message):
    note = midi.NoteOnEvent(tick=0, channel=channel, data=[(ord(character)), 0])
    track.append(note)

#write the new MIDI file
midi_file.append(track)
file_name = file_name.replace('.mid', '_encoded.mid')
midi.write_midifile(file_name, midi_file)

print("Message has been coded into : "+file_name+" in channel "+str(channel))
```

ENCODE

Examples

Hiding Data: Inside Audio Files

```
import sys
import midi

midi_file = midi.read_midifile(sys.argv[1])
message = ""
channels = {}
channel = -1
```

DECODE

Hiding Data: Inside Audio Files

```
#Look for used channels
for i in midi_file:
    for j in i:
        try:
            j.channel
        except:
            pass
        else:
            if channels.has_key(j.channel):
                channels[j.channel] += 1
            else:
                channels[j.channel] = 1

for c in channels:
    initializer = ""
    for track in midi_file:
        for event in track:
            if isinstance(event, midi.NoteOnEvent) and event.channel == c and len(initializer) < 5:
                initializer += chr(event.data[0])
            else:
                if initializer == "-msg-":
                    channel = c
                    break
        if channel >= 0:
            break
    if channel >= 0:
        break
```

DECODE

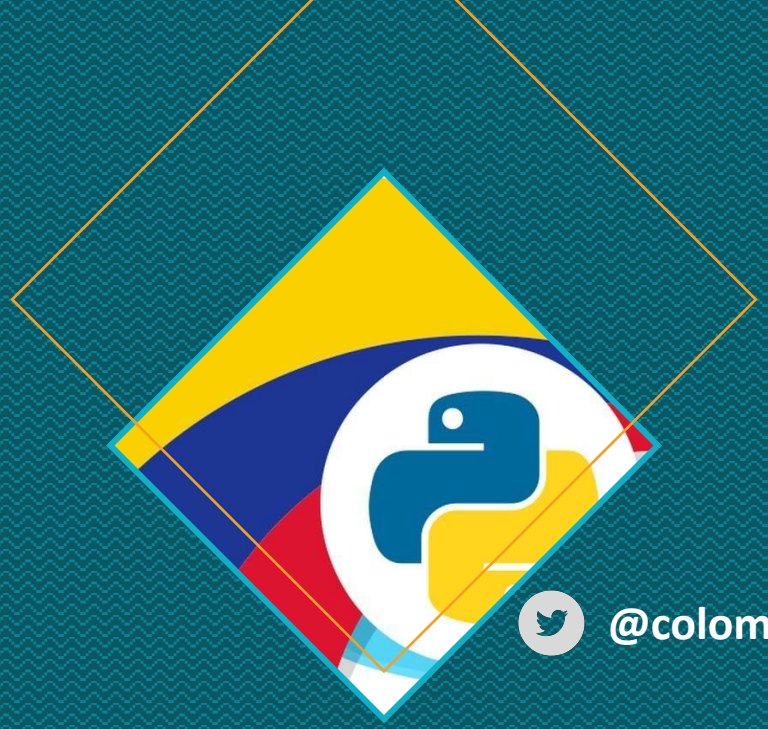
Examples

Hiding Data: Inside Audio Files

```
for track in midi_file:
    for event in track:
        if isinstance(event, midi.NoteOnEvent) and event.channel == channel:
            message += chr(event.data[0])

message = message.replace('-msg-', '')

print("Message is: '"+message+"'")
```

DECODE

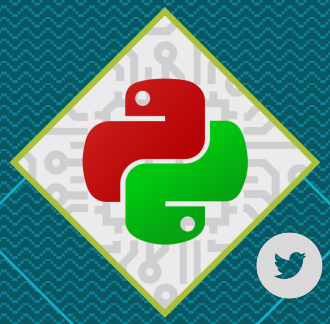
Examples



@colombiapython

Python Community

@pythoncali



@pythontulua

@scipyla


Thank You