STEGANOGRAPHY

Stanko Krtalic Rusendic

@monorkin
github.com/Stankec

Ruby on Rails Engineer @ Q Alliance


01
WATERMARKS

I hate watermarks. They are too intrusive.

Or they aren’t intrusive enough.

There is no good watermarking solution!

02

STEGANOGRAPHY

I wanted to solve this issue, and while researching I ran into Steganography.

steganography

/stɛgə'nɒgrəfɪ/

noun

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

03 IMAGES

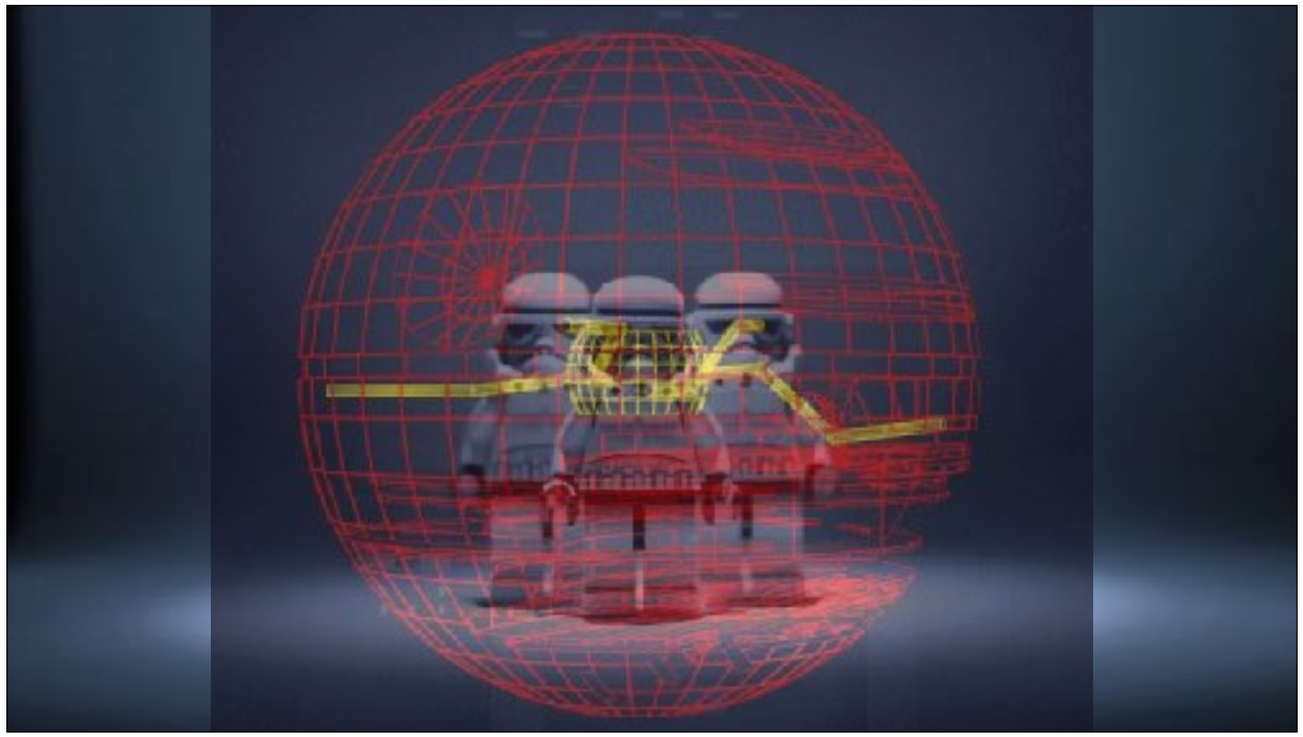
Images can be stored in several formats on your PC.

If these stormtroopers want to hide data they have a few options.

They can use the image’s EXIF data, COMMENT fields, or just append the data after the image data.

EXIF, iTXt (PNG), comment (JPEG)

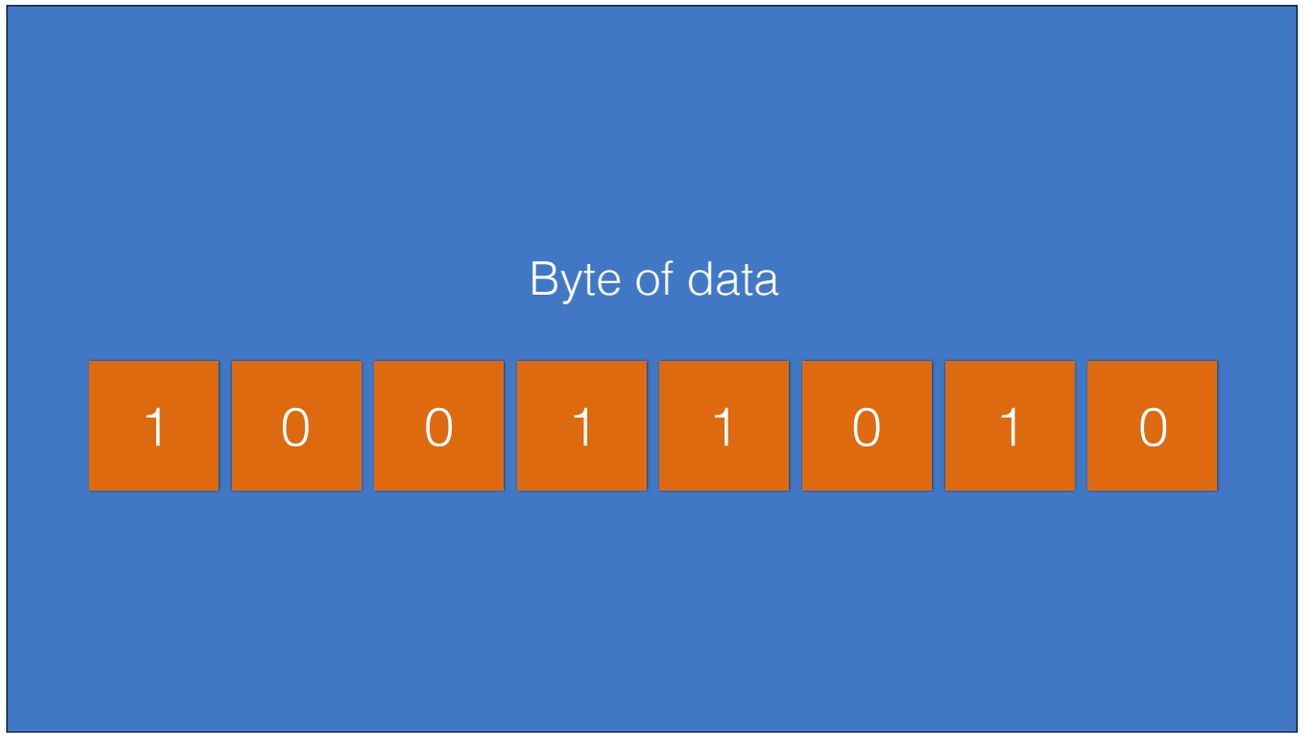
Are extremely easy to spot and to strip



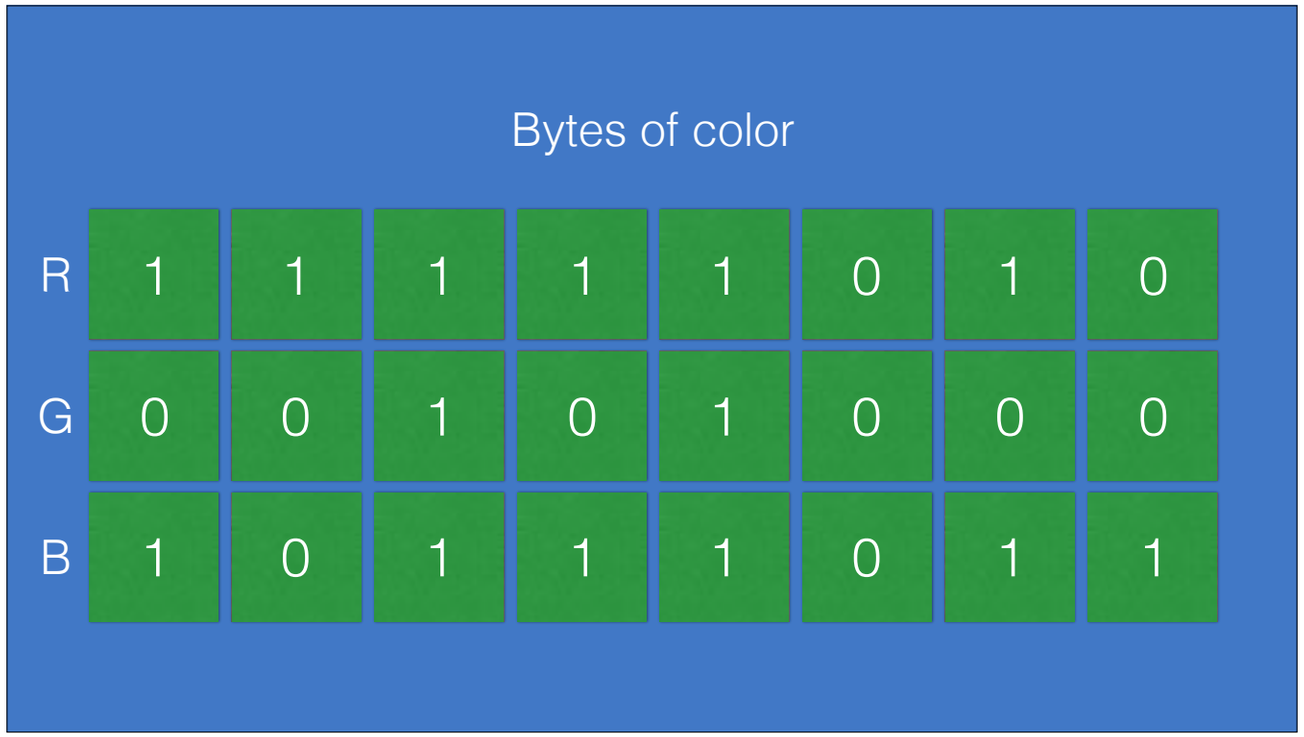
We need to find a way to store the death star plans so that those guys can’t find them!

04 LSB

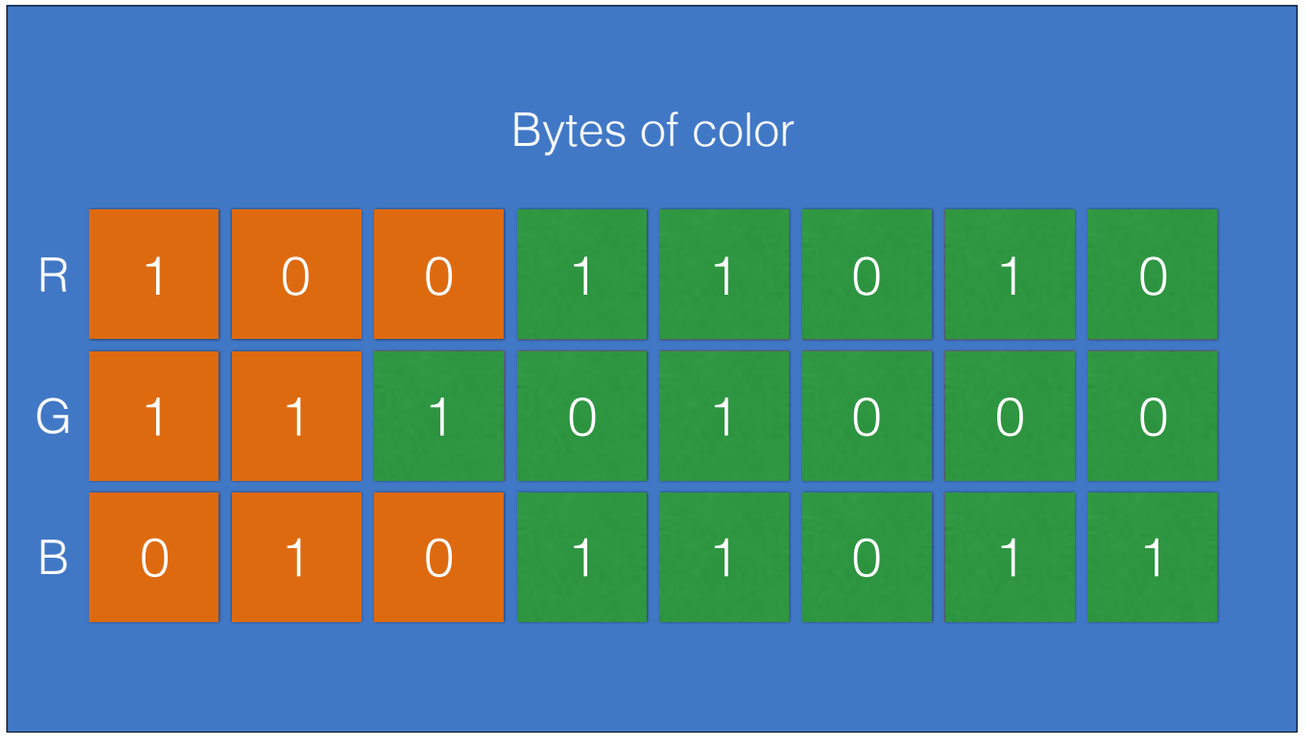
Leas Significant Bit Steganography to the rescue!



We can represent one colour component with 8 bits - 1 byte



To represent a color we need 3 bytes



We can use 8 bits of our 3 bytes to encode a byte of our message. This means that we will lose the original data, but in a way which makes it indistinguishable from regular image data.

The human eye is most sensitive to green and red. This is an evolutionary thing. When we were hunter-gatherers it was important to be able to detect movement and berries. Therefore we will use 2 bits of green, three bits of red and three bits of blue.

Since we are modifying the least significantnt bytes, we are changing the value by a maximum of 8. If you open up Photoshop, Sketch or your browser and pick a colour, e.g. 130, 130, 130 and change each value by 8 you will notice that you can’t notice the difference.

1byte of data = 1pixel

```ruby
def build_file_package(file)
  file_name = File.basename(file)
  data = file.read

  [
    file_name.length,
    file_name.unpack('C*'),
    [data.length].pack('q*').unpack('C*'),
    data.unpack('C*')
  ].flatten
end

def infuse_components_with_byte(store, r, g, b, byte)
  store << (byte ? (r & 248 | ((byte >> 5) & 7)) : r)
  store << (byte ? (g & 252 | ((byte >> 3) & 3)) : g)
  store << (byte ? (b & 248 | (byte & 7)) : b)
end
```

Ruby provides the `pack` and `unpack` methods for manipulating byte data. Unpack converts a string into an array of integers representing the value, while pack does the opposite.

As you can see from the slide, the described logic is very simple to implement in Ruby.

```
# image.ppm
P3
4 4
15
#r  g  b    r  b  g    0  0  0   15  0 15
 0  0  0    0  0  0    0  0  0    0  0  0
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

If we use a file format like PPM then we can easily modify the image data with Ruby.
PPM consists of a simple header and an array of triples representing R G B values.

05

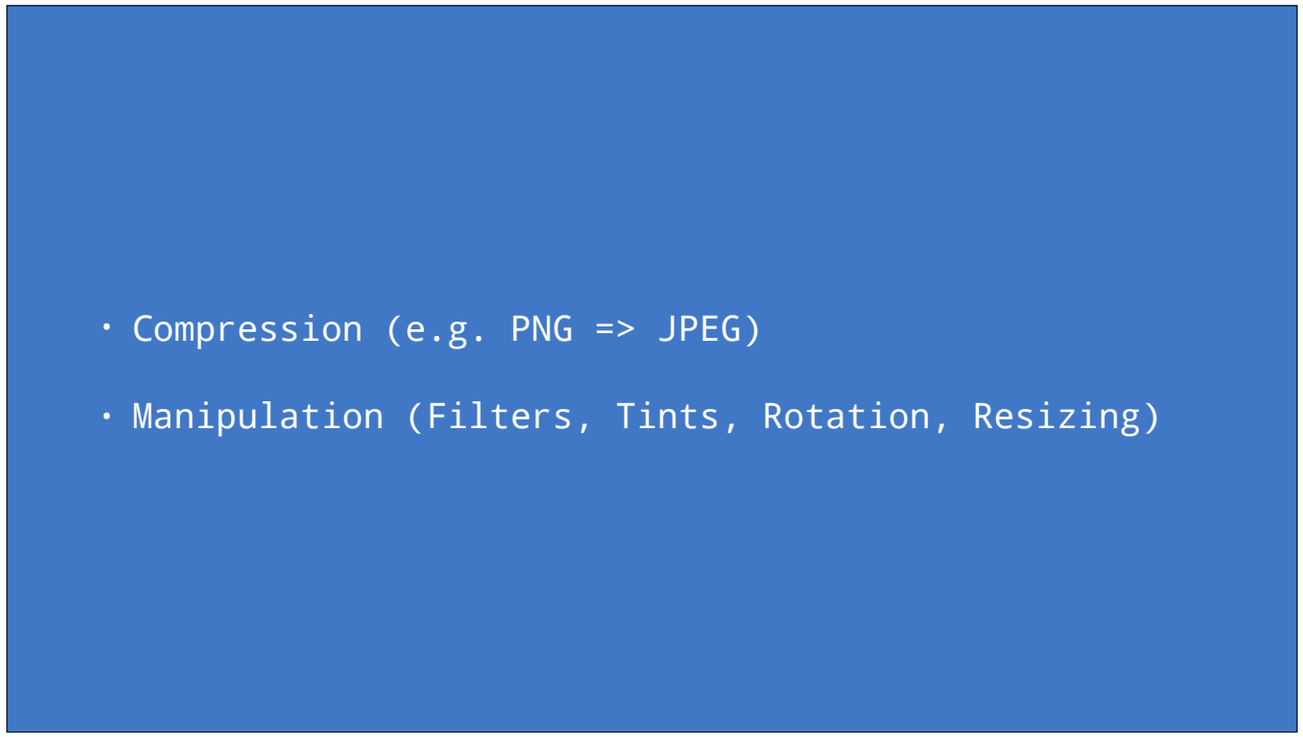
RESULT


Audio encoded in my image?
Count me surprised.

The left image has no data encoded, the right image has a book encoded in it.

06

PROBLEMS



This method is sensitive to compression, though there are methods to make it resilient to compression they are not commonly implemented.
Also image manipulations can damage data integrity.

07 PLANS

- Opensource all algorithms
- Remove dependency on ImageMagick