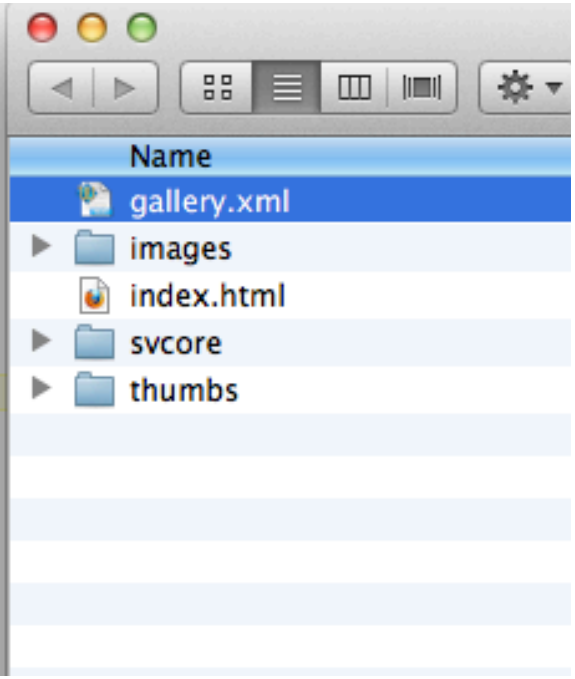
Wrap Up from Last Lecture

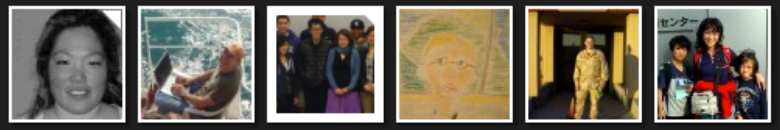
- In the Simpleviewer XML file we entered metadata for the class picture
  - Added the picture file to images
  - Added tiny picture file to thumbs

```
<image imageURL="images/shipcomputing1.JPG"
 thumbURL="thumbs/shipcomputing1.JPG"
 linkURL="images/shipcomputing1.JPG"
 linkTarget="_blank">
 <caption><![CDATA[On Board]]></caption>
</image>
<image imageURL="images/class.jpg"
 thumbURL="thumbs/class.jpg"
 linkURL="images/class.jpg"
 linkTarget="_blank">
 <caption><![CDATA[CSE120]]></caption>
</image>
<image imageURL="images/portraint.JPG"
 thumbURL="thumbs/portraint.JPG"
 linkURL="images/portraint.JPG"
 linkTarget="_blank">
 <caption><![CDATA[Karalina's Art]]></caption>
</image>
```



The Result

Faces



SIMPLEVIEWER

Computing Is Pretty Strange

Steganography: An Amazing Thing To Do with Bits

Lawrence Snyder
University of Washington, Seattle

Steganography

- The process of hiding information
- Two Greek roots meaning:

  "stego" == "roof" "stega" == "cover"

Why Hide Information?

- Most common reason to hide information is to avoid being caught with it
  - Military and spy documents
  - Repressive governments restricting news/info
  - Avoid others "snooping" – privacy
- Hiding is different than encryption … uses the fact that the searcher doesn't know it's there

Illustrate A Way To Do It

- The Plan …
  - hide “subversive” protest picture in “calendar art”

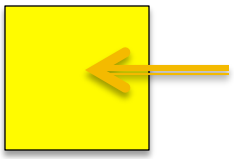
Guest Image

Host Image

Recall Properties of Pixels

- Pixels are made of RGB color

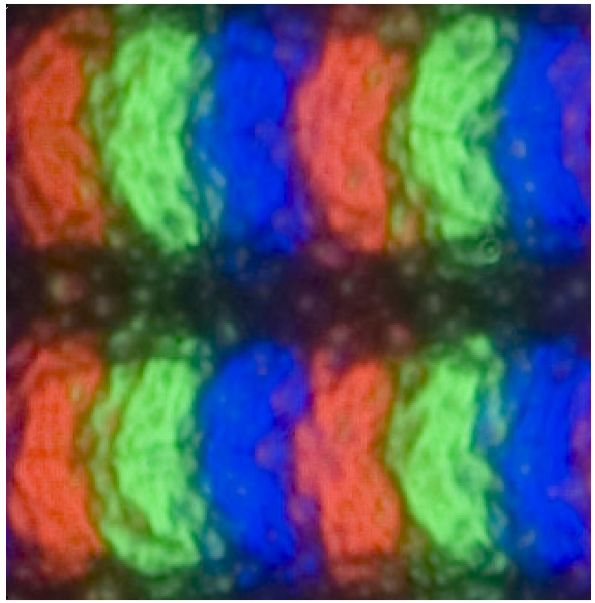
- Light intensity (brightness) of the 3 lights is given by RGB bytes:



R: 1111 1111
G: 1111 1111
B: 0000 0000

- We can manipulate the bits and change the picture

2 x 2 Pixels



Manipulating Bits

- A handy fact about binary numbers is that a sequence of bits

  `11010110101110`

  can be shifted right or left by dividing or multiplying by a power of 2
  - Dividing by 2^n shifts right n bit positions
  - Multiplying by 2^n shifts left n bit positions

  `11010110101110` $\div 2^2 =$ `00110101101011`

  `11010110101110` $\times 2^2 =$ `01011010111000`

Step 1: Reduce Bits of Guest

- We don't need all of the bits in RGB to get a decent picture

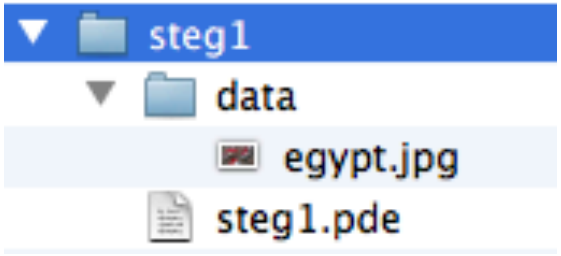
All bits

1011 0100 1101 0011 0001 1100

Left 2 bits of each color

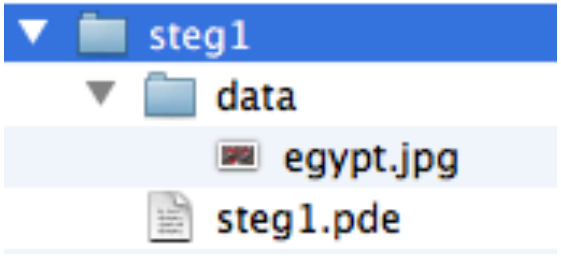
1011 0100 1101 0011 0001 1100

Check Removing Bits




```
PImage baseIm, fewerBits;
int i = 0;
int wid=450;
int hi=300;
color c;
int fact=1;
```

Check Removing Bits




```
PImage baseIm, fewerBits;
int i = 0;
int wid=450;
int hi=300;
color c;
int fact=1;
```

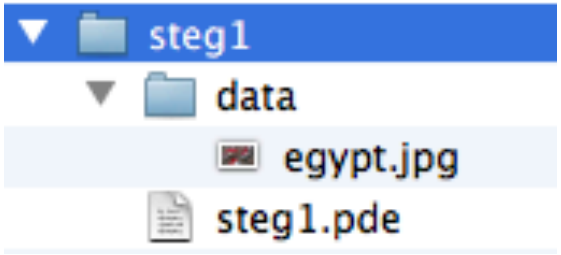
```
void setup( ) {
  size(wid, hi);
  baseIm = loadImage("egypt.jpg");
}

void draw( ) {
  image(baseIm, 0, 0);
  loadPixels();
  for (int i = 0; i<wid*hi; i++){
    c = color( fact*(int(red(pixels[i])/fact)),
               fact*(int(green(pixels[i])/fact)),
               fact*(int(blue(pixels[i])/fact)));
    pixels[i] = c;
  }
  updatePixels( );
}

void mousePressed( ) {
    fact = 2 * fact;
}
```

1101 0110

0011 0101

1101 0100

Check Removing Bits




```
PImage baseIm, fewerBits;
int i = 0;
int wid=450;
int hi=300;
color c;
int fact=1;
```

```
void setup( ) {
  size(wid, hi);
  baseIm = loadImage("egypt.jpg");
}

void draw( ) {
  image(baseIm, 0, 0);
  loadPixels();
  for (                    <wid*hi; i++){
    c =                    *(int(red(pixels[i])/fact)),
                       fact*(int(green(pixels[i])/fact)),
                       fact*(int(blue(pixels[i])/fact)));
    pixels[i] = c;
  }
  updatePixels( );
}

void mousePressed( ) {
    fact = 2 * fact;
}
```

Just Do It!

1101 0110

0011 0101

1101 0100

Step 2: Replace Bits In Host

- Put guest bits into right 2 bits of host

1111 0100 1101 0011 1011 1101

1011 0100 1101 0011 0001 1100

1111 0110 1101 0011 1011 1100

Processing Code For Guest→Host

```
PImage crowd, fog;
int i = 0;
int srcw=512;
int srch=346;
int wid=450;
int hi=300;
color c, cprime;

void setup( ) {
  size(srcw, srch);
  crowd = loadImage("egypt.jpg");
  fog = loadImage("fog.jpg");
  image(fog,0,0);
  for (int i=0; i<srcw; i++){
    for(int j=0; j<srch; j++) {
      c = get(i,j);
      if (i<wid && j<hi) {
        cprime=crowd.get(i,j);
        cprime=color(4*(int(red(c))/4) + (int(red(cprime))/64),
                     4*(int(green(c))/4) + (int(green(cprime))/64),
                     4*(int(blue(c))/4) + (int(blue(cprime))/64));
        set(i,j, cprime);
      } else {
        set(i,j,c);
      }
    }
  }
}
```

Encoding Code

```
void draw( ) {
  if (mousePressed) {
    saveFrame("stegFog.png");
  }
}
```

Code To Save Result on Click

Compare fog.jpg with stegFog.png

fog.jpg

stegFog.png

Really?
Just Do It!

How Does It Work

- After the pictures are loaded

```
cprime=color(4*(int(red(c))/4) + (int(red(cprime))/64),
             4*(int(green(c))/4) + (int(green(cprime))/64),
             4*(int(blue(c))/4) + (int(blue(cprime))/64));
```

Clear right 2 bits of host

Extract left 2 bits of guest

New combined color

Recover The Image

```
PImage fog;
int flip = 0;
int srcw=512;
int srch=346;
int wid=450;
int hi=300;
color c, cprime;

void setup( ) {
  size(srcw, srch);
  fog = loadImage("stegFog.png");
  image(fog,0,0);
 }

 void draw( ) {
   if (mousePressed) {
     for (int i=0; i<srcw; i++){
     for(int j=0; j<srch; j++) {
      c = get(i,j);
      if (i<wid && j<hi) {
         cprime=color(64*(int(red(c))%4),
                      64*(int(green(c))%4),
                      64*(int(blue(c))%4));
         set(i,j, cprime);
      } else {
        set(i,j,c);
      }
     }
    }
   }
 }
```

Recover The Image

```
PImage fog;
int flip = 0;
int srcw=512;
int srch=346;
int wid=450;
int hi=300;
color c, cprime;

void setup( ) {
  size(srcw, srch);
  fog = loadImage("stegFog.png");
  image(fog,0,0);
 }

 void draw( ) {
   if (mousePressed) {
     for (int i=0; i<srcw; i++){
     for(int j=0; j<srch; j++) {
      c = get(i,j);
      if (i<wid && j<hi) {
         cprime=color(64*(int(red(c))%4),
                      64*(int(green(c))%4),
                      64*(int(blue(c))%4));
         set(i,j, cprime);
      } else {
        set(i,j,c);
      }
     }
    }
   }
 }
```

Just Do It!

How Does It Work

- Read in the file, and then on mouse click, pull out the bits and make a picture

```
cprime=color(64*(int(red(c))%4),
             64*(int(green(c))%4),
             64*(int(blue(c))%4));
```

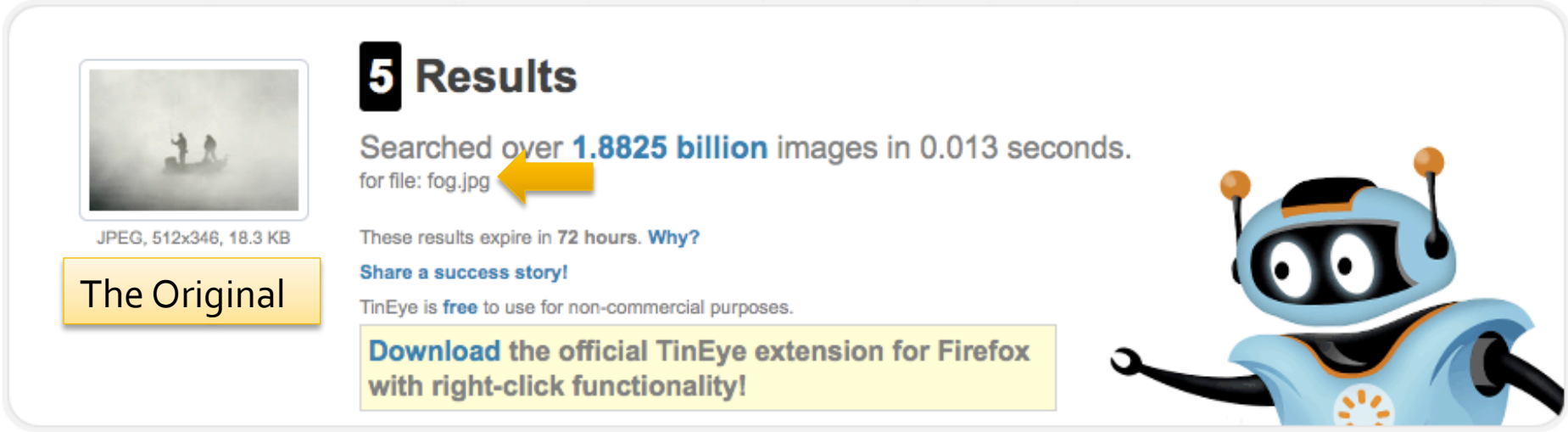
Remove right 2 bits

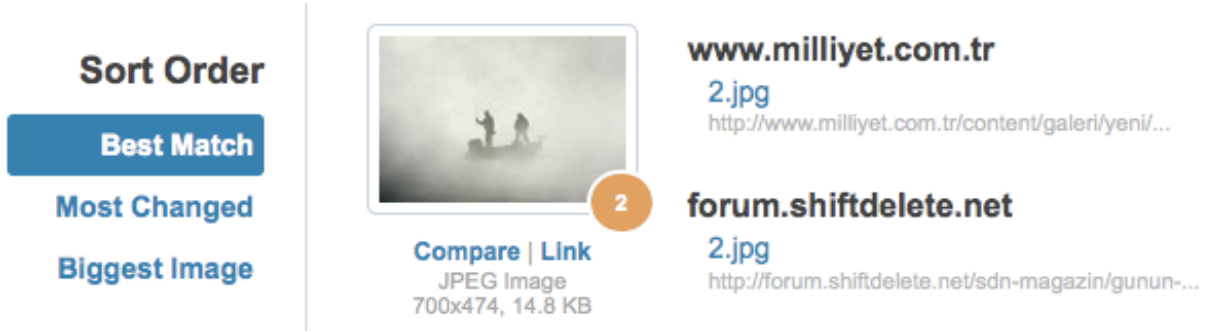
Make them left 2 bits for each color

New color

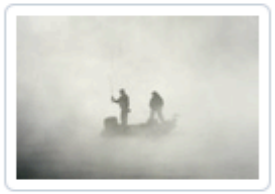
How Much Is Coded Like Original?

- Run A Test … www.tineye.com





Check The "Steganized" File



PNG, 512x346, 144.4 KB

Steganized

5 Results

Searched over **1.8825 billion** images in 2.609 seconds.
for file: stegFog.png

These results expire in **72 hours**. **Why?**

Share a success story!

TinEye is **free** to use for non-commercial purposes.

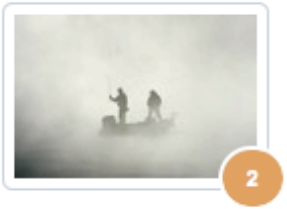
Download the official TinEye extension for Firefox with right-click functionality!

Sort Order

Best Match

Most Changed

Biggest Image



2

Compare | **Link**
JPEG Image
700x474, 14.8 KB

www.milliyet.com.tr
2.jpg
http://www.milliyet.com.tr/content/galeri/yeni/...

forum.shiftdelete.net
2.jpg
http://forum.shiftdelete.net/sdn-magazin/gunun-...

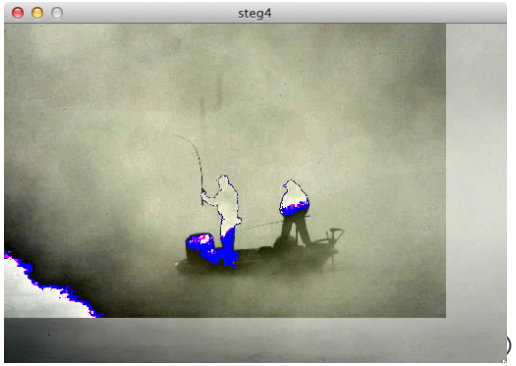
Summary

- Put guest bits into right 2 bits of host

1111 0100 1101 0011 1011 1101

1011 0100 1101 0011 0001 1100

1111 0110 1101 0011 1011 1100



Watch It:
Push the bits out the left side, slowly revealing the guest

Summary

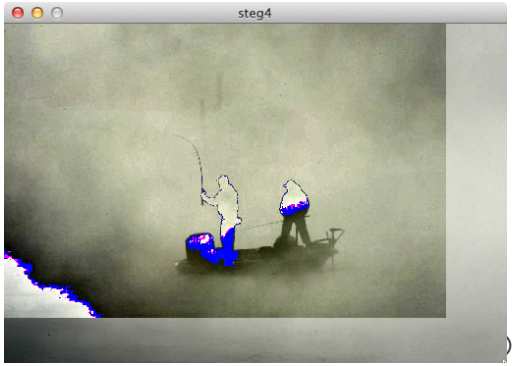
- Put guest bits into right 2 bits of host

Just Do It!

1111 0100 1101 0011 1011 1101

1011 0100 1101 0011 0001 1100

1111 0110 1101 0011 1011 1100



Watch It:
Push the bits out the left side, slowly revealing the guest