Encryption & Steganography: Amazing Things To Do with Bits

Lawrence Snyder
University of Washington, Seattle
Encryption

- Encryption is the process of “scrambling” data so it is difficult (impossible?) to understand it.
- We encrypt data to keep it private.
- Every site that you use as https:// is encrypted.
- Familiar example: Caesar cipher:

\[
\begin{align*}
C: & \text{ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z} \\
E: & \text{ D E F G H I J K L M N O P Q R S T U V W X Y Z A B C}
\end{align*}
\]

- What would Julius be encrypted to?
Encryption

- Encryption is the process of “scrambling” data so it is difficult (impossible?) to understand it.
- We encrypt data to keep it private.
- Every site that you use as https:// is encrypted.
- Familiar example: Caesar cipher:

  C:  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
  E:  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z  A  B  C

- What would Julius be encrypted to? Mxolxv
More Typically ...

- The fixed shift of an alphabet is easy to break

Alternate:

- Sender uses a key, k, to multiply clear byte sequences (recall they’re numbers) by k
  -- Send encrypted result – looks like gibberish --

- Receiver divides by k to decrypt getting clear
Example

- Let the clear be: “MEET @ 9” and k=13
- Break clear text into 2-letter sequences:
  - ME  ET  $@  9
- Interpret text as numbers
  - 7769  6984  3264  3257
- Multiply by key:
  - $7769 \times 13 = 100997$
  - $6984 \times 13 = 090792$
  - $3264 \times 13 = 042432$
  - $3257 \times 13 = 042341$
- Send encrypted (6-digit) number
- Receiver does the reverse process ...
The problem with “private key” encryption: the two sides have to meet to agree on key

Public Key fixes this: The receiver publishes (on Web site, say) a (very very special) key, K

More importantly, the theory it uses means that NO practical amount of computing can break the code

Here’s what you do ...
Public Key Process

- Sender breaks up the message into blocks as before
- Sender cubes each block – yup, raises to the 3\textsuperscript{rd} power – and mods it by K, i.e. \((<\text{text}>^3)\%K\)
- Transmit results
- Receiver raises each remainder to a high power determined by prime numbers & known only to him
- Receiver mods by K, too, which are – surprisingly – the original blocks!
- The receiver assembles the message
- Thanks to Euler and Diffie & Hellman

This Is Amazing!!!
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”
Step 1: Reduce Bits of Guest

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

All bits

Left 2 bits of each color

1011 0100 1101 0011 0001 1100

1011 0100 1101 0011 0001 1100

© 2011 Larry Snyder, CSE
Step 2: Replace Bits In Host

- Put guest bits into right 2 bits of host
Compare fog.jpg with stegFog.png

Really? Just Do It!
Let’s Look At Them

... and then we’ll see the details
Processing Code For Guest→Host

```cpp
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);
            }
        }
    }
}

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

Encoding Code
How Does It Work

- After the pictures are loaded

```java
int 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
```c
Plmage 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);
                }
            }
        }
    }
}
```
How Does It Work

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

```c
  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

3/7/11  © 2011 Larry Snyder, CSE
How Much Is Coded Like Original?

- Run A Test ... www.tineye.com

5 Results

Searched over 1,8825 billion images in 0.013 seconds.
for file: fog.jpg

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

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/...