Steganography

"En cryptographie, aucune règle n'est absolue."
("In cryptography, no rule is absolute")
Etienne Bazeries (1901)

- The word **Steganography** is derived from the Greek word ***Steganos***, which means covered or hidden, combined with ***graphy***, which means writing or drawing.

- Steganography is the art and science of communicating in a way that conceals any existence of the communication of information.

- The technique involves concealing the actual information within a "carrier", which can be an image, video, or audio file. It is implemented in a manner that prevents a third party from detecting the existence of the concealed message or file.

- A distinction must be made between cryptography (Greek *kryptos*, hidden) and steganography (Greek *steganos*, covered).

- Steganography can be considered to be the hidden cousin of cryptography. Cryptography provides privacy, whereas Steganography is intended to provide secrecy.

- Privacy is required for example when we use a credit card for purchase on the Internet. This involves Cryptographic techniques that reduce the plaintext to a coded segment of gibberish that only the recipient with the correct decryption keys can read.

- However, though the code may be unbreakable, the coded message transmission is visible, and therefore can be intercepted and possibly cracked.

- Steganography on the other hand provides true secrecy (and can incorporate Cryptography as well), where no one is even aware of the fact that messages are being sent.

- The origins of Steganography go back to 440 BC when Herodotus mentions an example in his *Histories* (The History, The Seventh Book – Polymnia, p259). Demaratus (King of Sparta) sent a warning about an impending attack on Greece by the Persian king Xerxes.

- He wrote the message directly on the wooden backing of a wax tablet before applying its beeswax surface. Wax tablets were commonly used in those times as reusable writing surfaces.

- The following is another example of Steganographic communication from Herodotus (The History, The Fifth Book – Terpsichore, p166):

  “For Histiaios, desiring to signify to Aristagoras that he should revolt, was not able to do it safely in any other way, because the roads were guarded, but shaved off the hair of the most faithful of his slaves, and having marked his head by pricking it, waited till the hair had grown again; and as soon as it was grown, he sent him away to Miletos, giving him no other charge but this, namely that when he should have arrived at Miletos he should bid Aristagoras shave his hair and look at his head: and the marks, as I have said before, signified revolt.”

- The first reported publication of a book on Steganography in 1499 (*Polygraphiae*) is attributed to the monk Johannes Trithemius who developed his so-called "Ave-Maria-Cipher" with which one can hide information in a Latin praise of God.

- One of Trithemius' schemes was to conceal messages in long invocations of the names of angels, with the secret message appearing as a pattern of letters within the words. For example, as every other letter in every other word:

  padiel a**p**o**r**s**y** mesarpon o**m**e**u**a**s** peludyn m**a**l**p**r**e**a**x**o

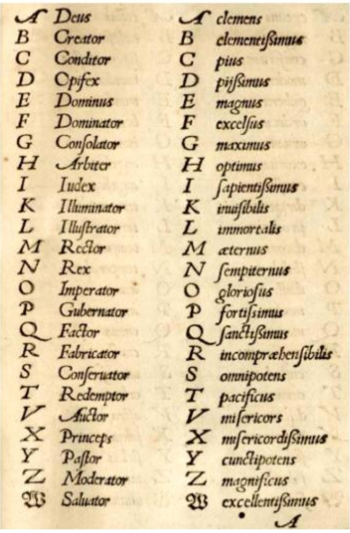
  which reveals "**prymus apex**"

- The "Ave Maria" cipher is a book that contains a series of tables, each of which has a list of words, one per letter. To code a message, the message letters are replaced by the corresponding words.

- If the tables are used in order, one table per letter, then the coded message will appear to be an innocent prayer.

- The cipher is a table of 384 parallel columns of Latin words. By taking words representing plaintext letters it is possible to construct ciphertexts that look like innocent religious litanies.

- Consider for example the following table from his text:

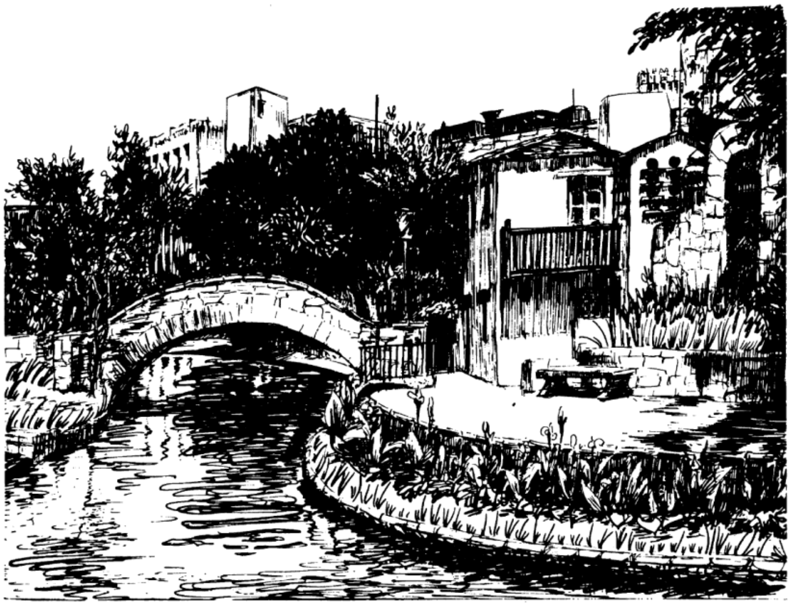


A	Deus	A	clemens
B	Creator	B	clementiſsimus
C	Conditor	C	pius
D	Opifex	D	piiſsimus
E	Dominus	E	magnus
F	Dominator	F	excelſus
G	Conſolator	G	maximus
H	Arbiter	H	optimus
I	Iudex	I	ſapientiſsimus
K	Illuminator	K	inuiſibilis
L	Illuſtrator	L	immortalis
M	Rector	M	æternus
N	Rex	N	ſempiternus
O	Imperator	O	glorioſus
P	Gubernator	P	fortiſsimus
Q	Factor	Q	ſanctiſsimus
R	Fabricator	R	incompræhenſibilis
S	Conſeruator	S	omnipotens
T	Redemptor	T	pacificus
V	Auctor	V	miſericors
X	Princeps	X	miſericordiſsimus
Y	Paſtor	Y	cunctipotens
Z	Moderator	Z	magnificus
W	Saluator	W	excellentiſsimus

A

- Using the above table, the plaintext “IACR” will encrypted as “Judex clemens conditor incomprehensibilis”.

- For yet another historical example, consider the following semagram (semantic symbol or picture) which was sent as postcard from one military officer to another in 1945:



- The hidden message is in Morse code. Can you identify it?

Steganography in the Digital Age

- Steganography has traditionally been used by the military and criminal classes. But it is increasingly being adopted by the commercial sector. Examples of business applications are watermarking and copyright protection of documents and images.

- Criminals, organized crime, and terrorist groups are increasingly using modern technology such as encryption and steganography to conduct their activities.

- Steganography is a method of hiding digital information in a way that evades detection and that makes the job of law enforcement organizations even more difficult.

- With these emerging new techniques, a hidden message is indistinguishable from white (random) noise. Even if the message is suspected, there is no proof of its existence. To actually prove there was a message, and not just randomness, the code needs to be cracked or the random number seed guessed. This feature of modern steganography is called "plausible deniability."

- Of course, there are a number of peaceful applications for steganography as well. The simplest and oldest are used in map making, where cartographers sometimes add a tiny fictional street to their maps, allowing them to prosecute copycats. A similar trick is to add fictional names to mailing lists as a check against unauthorized resellers.

- Newer applications use steganography like a watermark, to protect a copyright on information. Photo collections sold on digital media often have hidden messages embedded in them, which allows detection of unauthorized use. The same technique when applied to DVDs is even more effective, since the industry builds DVD recorders to detect and disallow copying of protected DVDs.

- Even biological data, stored on DNA, may be a candidate for hidden messages, as biotech companies seek to prevent unauthorized use of their genetically engineered material. The technology is already in place for this: three New York researchers successfully hid a secret message in a DNA sequence and sent it across the country.

Steganography Techniques - Least Significant Bit Insertion

- Steganography requires two data components: the cover or carrier, and the information to be concealed. The cover is the medium into which the information is embedded and concealed.

- The process can be summarized as follows:

  cover_medium + hidden_data + stego_key = stego_medium

- The ***cover_medium*** is the carrier file in which data to be concealed, the *hidden_data*, is embedded. The data to be concealed can also be encrypted using the ***stego_key***.

- The resulting file is the ***stego_medium*** (which will be the same type of file as the cover_medium). The cover_medium (and, thus, the stego_medium) are typically image or audio files.

- The selection of a carrier medium is critical to the effectiveness and security of the Steganographic technique.

- Steganography techniques fall into two categories: Image Domain and Transform Domain.

- Image Domain techniques embed the message within the intensity of the pixel directly whereas transform domain techniques transform the image first, then the message is embedded into the image.

- Image Domain techniques encompass bit-wise methods that use bit insertion and noise manipulation. Transform domain techniques involve the manipulation of algorithms and image transforms.

- A simple implementation of an Image Domain technique is Least Significant Bit (LSB) embedding. We will use the LSB method to embed secret data within a larger, carrier image.

- Each color channel R, G, and B in the RGB colorspace is represented by a number, and this number is represented by a number of bits. A bit-plane refers to all the bits at a single bit position across a complete image.

- Consider the number 42, whose 8-bit binary representation is "00101010". Starting from the right, we have a "0" in the zeroth bit-plane, a "1" in the first, a "0" in the second, and so on for all eight bits. In an image, a bit-plane refers to the 0 or 1 value at a given position for all pixels, laid out in the same format. In LSB Steganography, the least significant bit-planes are manipulated.

- LSB embedding is a very simple strategy to implement steganography. The technique replaces some of the information in a given pixel with information from the data in the image.

- While it is possible to embed data into an image on any bit-plane, LSB embedding is performed on the least significant bit(s). This minimizes the variation in colors that the embedding creates.

- For example, embedding into the least significant bit changes the color value by one. Embedding into the second bit-plane can change the color value by 2. If embedding is performed on the least significant two pixels, the result is that a color in the cover can be any of four colors after embedding.

- LSB embedding always results in a loss of information (resolution) in the cover image. This is an effect of embedding directly into a pixel.

- The size of an image file is directly related to the number of pixels and the resolution. A typical 640x480 pixel image using a palette of 256 colors (8-bit resolution) would require a file about 307 KBytes in size (640 • 480 • 1 Byte), whereas a 1024x768 pix high-resolution 24-bit color image would result in a 2.36 MByte file (1024 • 768 • 3 bytes).

- In order to minimize the image file sizes a number of compression schemes have been developed over time, notably Bitmap (BMP), Graphic Interchange Format (GIF), and Joint Photographic Experts Group (JPEG) file types.

- GIF and 8-bit BMP files employ what is known as **lossless** compression, a scheme that allows the software to exactly reconstruct the original image. JPEG, on the other hand, uses **lossy** compression, which means that the expanded image is very nearly the same as the original but not an exact duplicate.

- While both methods conserve storage space, lossless compression is much better suited to applications where the integrity of the original information must be maintained, such as steganography. While JPEG can be used for Steganographic applications, it is more common to embed data in GIF or BMP files.

- The key innovation in Steganography is to select an innocuous carrier (video or audio) that contains a lot random information, i.e., white noise.

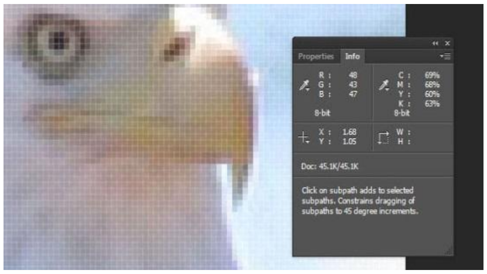
- White noise manifests itself in the form of dirt specks on images or noise hiss in an audio signal. Steganography in essence inserts the secret message into a carrier to make it appear as random as the white noise.

- The most popular methods use digitized photographs mainly because digitized photographs harbor plenty of white noise. A digitized photograph is stored as an array of colored pixels.

- In a typical 24-bit color image, each pixel has three components (Red, Green and Blue) with each component using 8-bits to determine the color intensities. These 8-bit values often range from 0 - 255. Each number is stored in binary as eight bits, with the most significant bit (MSB) equal to 128 (2^7) in decimal, and the least significant bit (LSB) equal to 1 (2^0).

- To understand how the pixel values of a digital image vary, consider the following image:

- The following images show the difference in RGB values for adjacent pixels (in the pupil area):



R : 48
G : 43
B : 47
8-bit

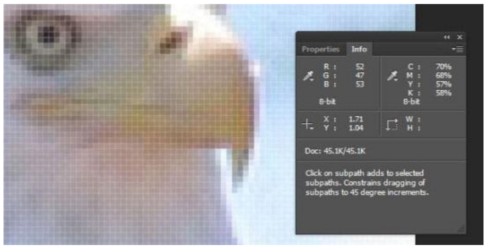
C : 69%
M : 68%
Y : 60%
K : 63%
8-bit

X : 1.68
Y : 1.05

W :
H :

Doc: 45.1K/45.1K

Click on subpath adds to selected subpaths. Constrains dragging of subpaths to 45 degree increments.



R : 52
G : 47
B : 53
8-bit

C : 70%
M : 68%
Y : 57%
K : 58%
8-bit

X : 1.71
Y : 1.04

W :
H :

Doc: 45.1K/45.1K

Click on subpath adds to selected subpaths. Constrains dragging of subpaths to 45 degree increments.

- A difference of one or two in the intensities is imperceptible; in fact a digitized picture can still look quite good even if the least significant four bits of intensity are altered - a change of up to 16 in the color's value.

- This provides plenty of space to conceal secret data. Text is usually stored using 7 bit ASCII characters. If we use the least significant bit of the red, green, and blue values for a pixel, we can get three bits per pixel.

- If we used 3 pixels we could store 9 bits which is enough for one character, with two bits left over. Thus a 640 x 480 pixel image can hold 102,400 characters.

- A modern smartphone is capable of taking images with 8 Mega Pixels or over 8,000,000 pixels. This carrier image would be large enough to conceal 2,666,666.7 characters (complete version of Moby Dick, which is 2,242,843 characters).

- Note that a Grey scale image can only store 1 bit of information per pixel.

LSB Implementation

- As an example of LSB insertion, let us assume that we have three adjacent pixels (nine bytes) with the following three RGB components, each encoded using 8 bits:

	Pixel 1	Pixel 2	Pixel 3
Red	11010110	11100100	11001111
Green	00010111	00011010	00010001
Blue	00011100	00110000	00010011

- Now let assume that we wish to conceal the following set of 9 bits: 111000110 within the 9 bytes. The process can be summarized as follows:

	Pixel 1	Pixel 2	Pixel 3
Red	11010111	11100101	11001111
Green	00010110	00011010	00010000
Blue	00011101	00110001	00010010

- Notice that in this example 7 bits were actually flipped (shaded bits), the other 3 remain unchanged. The percentage of bits actually flipped will vary but LSB tends to result in an average of 50% bits flipped.

- The implementation of the bit insertion (or flipping) is quite straightforward. The flowing code fragments illustrate this (in C):

```
Byte|= 1;        // Set LSB of Byte to 1
Byte &= 0xFE;    // Set MSB of Byte to 1
```

- The following code illustrates some example functions to process bytes:

```
typedef unsigned char byte;      // C not Java!

byte GetLSBs (byte *Byte)        // Get the LSB
{
        int i;
        byte BitPosition = 0;
        for (i = 0; i < 8; i++)
        {
                BitPosition >>= 1;
                if (*Byte & 1)
                        BitPosition |=  0x80;
                Byte++;
        }
        return BitPosition;
}

void SetLSBs (byte *Byte, byte Byte0)    // Set the LSB
{
        int i;
        for (i = 0; i < 8; ++i)
        {
                if (Byte0 & 1)
                        *Byte |= 1;
                else
                        *Byte &= 0xFE;
                ++Byte;
                        Byte0 >>= 1;
  }
}
```

- In order to extract the information from an image the 8-bit binary equivalent of each RGB color component of pixels must first be obtained.

- The LSB of this binary number represents a one bit of the hidden information that was embedded. Each of such bits are then stored in an output file to generate the original image.

- LSB insertion simple but it has a disadvantage of being susceptible to image processing operations such as cropping and compression.

- Another drawback of this technique is that if the original image was in GIF or BMP file format (lossless compression techniques) and then converted to a JPEG file format (a lossy compression technique), and then converted back to the original format, the data in the LSBs will not be recoverable.

- There are many libraries available for manipulating image files (reading, writing, processing, etc) depending on the language that will be used to implement the application.

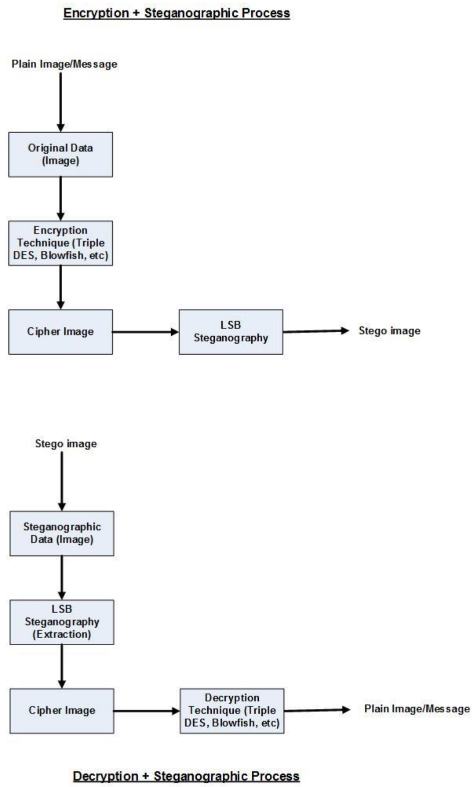
- Some of the more popular ones are:
  - **Pillow –** Python library for image manipulation
  - **ImageMagick** (this is a command line tool but it does provide a C++ API (**Magick++**))
  - **ImageMagick and the RMagick gem** – Ruby
  - **Qt** also provides a **QImage** class (very easy to design UIs as well)
  - **CImg** (http://cimg.sourceforge.net/) - defines classes and methods to manage images in your own C++ code.
  - **libjpeg** (http://libjpeg.sourceforge.net/) – good C library for manipulating jpeg images
  - **Java Advanced Imaging (JAI) Image I/O tools** (http://www.oracle.com/technetwork/java/current-142188.html) -

Combining Cryptography into Steganography

- It is relatively easy to incorporate encryption techniques into steganography in order to achieve a higher level of security.

- The general technique is to apply an encryption technique to the "secret" data before applying the steganographic technique to it.

- The following diagram summarizes the process:


Encryption + Steganographic Process
Plain Image/Message
Original Data (Image)
Encryption Technique (Triple DES, Blowfish, etc)
Cipher Image
LSB Steganography
Stego image
Stego image
Steganographic Data (Image)
LSB Steganography (Extraction)
Cipher Image
Decryption Technique (Triple DES, Blowfish, etc)
Plain Image/Message
Decryption + Steganographic Process

QR Steganography

- Quick Response Code or better known as QR Code is a two dimensional barcode that allows high speed data encoding and decoding capabilities. It was invented by Denso-Wave a Toyota subsidiary in 1994 in order to track various automobile parts during the vehicle manufacturing process.

- Generally QR Codes are used for distributing small information such as URLs, a phone numbers, and small text messages.

- The QR code design has several desirable features:

    - High capacity encoding of data, specifically its capacity to encode 7089 numeric characters
    - Small printout size
    - Chinese/Japanese character capability
    - Dirt and damage resistance
    - A structure append feature - enables QR code to subdivide data into several compartments that contain data
    - QR codes are omni-directional, which means these codes can be read from any direction

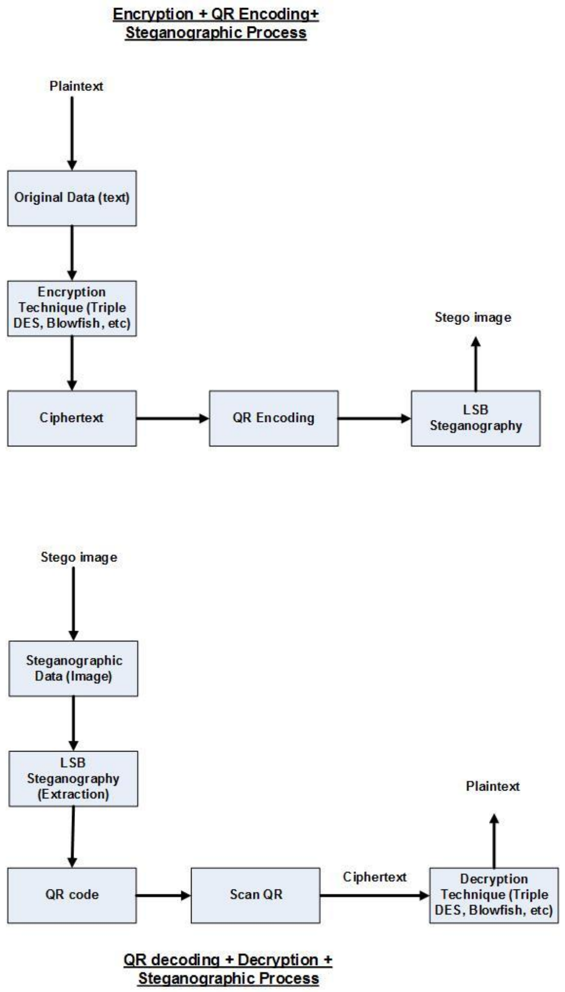
- The code itself looks like black separable components that are arranged in a square against a white background. There are several good QR generating applications available for the Android, Linux, and Windows platforms.

- This arrangement can then be scanned by the QR scanner or QR scanner app of a smartphone.

- The following is an example of a QR code. If scanned with a smartphone it will direct the device to a well-known web site:

- The general technique here is to first encrypt the message using any cryptographic algorithm, then encode it using QR, and finally conceal it in an image or audio carrier file using LSB steganography techniques.

- The following diagram summarizes this process:


Encryption + QR Encoding+ Steganographic Process
Plaintext
Original Data (text)
Encryption Technique (Triple DES, Blowfish, etc)
Ciphertext
QR Encoding
LSB Steganography
Stego image
Stego image
Steganographic Data (Image)
LSB Steganography (Extraction)
QR code
Scan QR
Ciphertext
Decryption Technique (Triple DES, Blowfish, etc)
Plaintext
QR decoding + Decryption + Steganographic Process

- Once the QR Code has been embedded into a cover image, only the communicating parties will be aware of the existence of the QR Code within the image.

- This steganographic technique is ideal for security applications such as applications for exchanging short confidential messages, phone numbers, account numbers, and embedding signature/copyright information in an image.

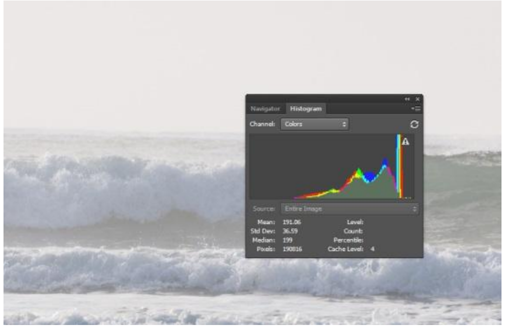
- Consider a situation where an artist or musician embeds an encrypted QR Code establishing his or her ownership into their image. This will allow the legitimate owner to claim ownership in cases of plagiarism and copyright violations.

Steganography Examples and Issues

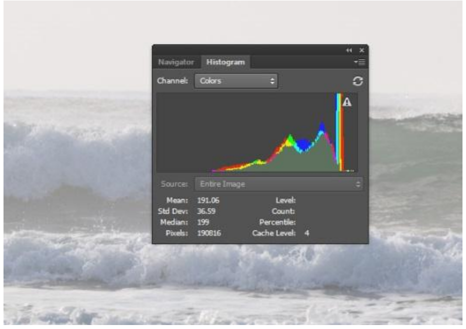
- Concealing data within a cover or carrier is quite simple using LSB insertion, moreover the cover image will not much. The untrained eye will be unable to discern any anomalies in the image.

- Consider the two images below. The one on the left is the original carrier image. The one on the right is the carrier with the full text of Moby Dick in pdf format embedded in it.

- The statistics of both images reveals virtually no difference between the two:



```
Channel: Colors
Source: Entire Image
   Mean: 191.06          Level:
Std Dev: 36.59           Count:
 Median: 199        Percentile:
 Pixels: 190816    Cache Level: 4
```



```
Channel: Colors
Source: Entire Image
   Mean: 191.06          Level:
Std Dev: 36.59           Count:
 Median: 199        Percentile:
 Pixels: 190816    Cache Level: 4
```

- However, someone who does put in the effort to detect steganographic activity will most likely be trained in steganalysis and will be able to detect LSB stego using more advanced statistical analysis.

- The original intensity values in the original cover image were white noise, i.e. random. The new values are strongly patterned, because they represent significant information of the secret image. This is the precisely the type anomalies which will be detectable using a statistical attack.

- The next obvious step is to achieve good steganography is make the message look random before hiding it.

- The easiest solution is simply to encode the message before hiding it. Using a strong cryptographic technique (Triple DES, Blowfish, etc), the coded message will appear just as random as the picture data it is replacing.

- Another approach is to spread the hidden information randomly over the photo. "Pseudo-random number" generators take a starting value, called a seed, and produce a string of numbers which appear random.

- To spread a hidden message randomly over a cover picture, use the pseudo-random sequence of numbers as the pixel order. Descrambling the photo requires knowing the seed that started the pseudo-random number generator.

- **Steganalysis** is the art and science of detecting and breaking steganography. One method of this analysis is to examine the color palette of a graphical image. In most images, there will be a unique binary encoding of each individual color.

- If the image contains hidden data, however, many colors in the palette will have duplicate binary encodings since, for all practical purposes, we can't count the LSB. If the analysis of the color palette of a given file yields many duplicates, we might safely conclude that the file has hidden information.

- However, this raises an obvious problem; which files would be analyzed? Suppose the hidden message is concealed in a cover image and posted at any number of sites the Internet (Facebook, Craigslist, etc, etc). How could we possibly search for possible stego files on the Internet?