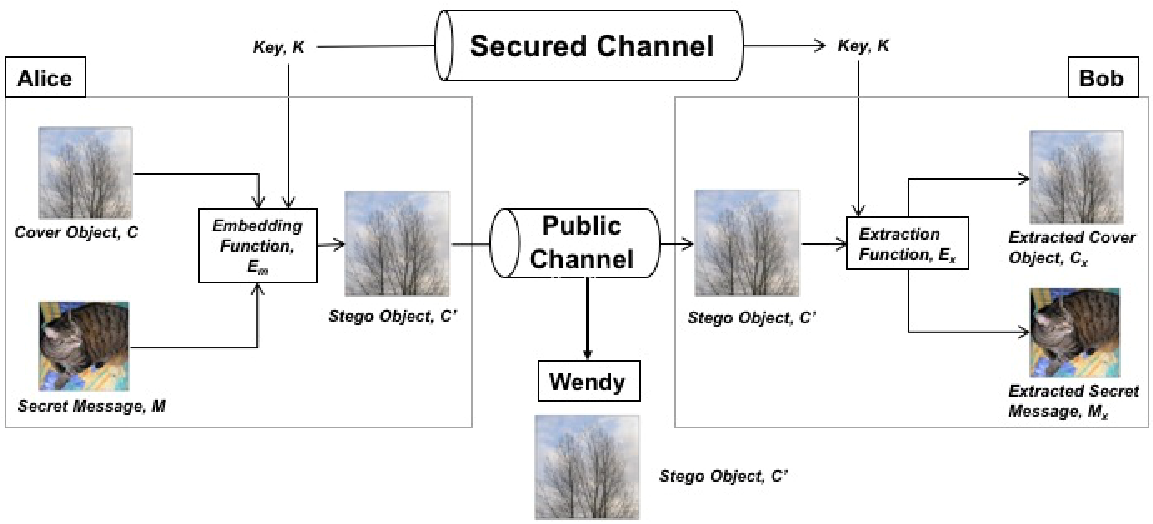
Tutorial #8

Security in Computing COSC2356/2357

Q1: Discuss the general model of secured data hiding (i.e. steganography).

Answer:

Modern Steganography: General Model



Embedding Function, $E_m : C \oplus K \oplus M \rightarrow C'$

Extraction Function, $E_x(C', K) : C' \oplus K \rightarrow C_x, M_x$

The modern formulation of steganography is often given in terms of the **prisoner's problem** where ***Alice*** and ***Bob*** are two inmates who wish to communicate in order to hatch an escape plan.

However, all communication between them is examined by the warden, ***Wendy***, who will put them in solitary confinement at the slightest suspicion of covert communication.

Alice wishing to send a ***secret message (m)*** to Bob, "*embeds*" ***m*** into a ***cover-object (C)*** using a ***stego-key (K)***, and obtains a ***stego-object (S)***.

The ***stego-object (S)*** is then sent through the public channel and ***stego-key (K)*** is sent to Bob using a private channel.

Cover-object (C): refers to the object used as the carrier to embed messages into. Generally, less important in the communication.

Stego-object (S): refers to the object which is carrying a **hidden message**. So, given a **cover object** and a message. The goal of the steganographer is to produce a **stego object** which would carry the message.

Stego-Key (K): refers to the secret information (i.e. secret key) that will be used to extract secret message. For example, location of **hidden message** in the stego-object.

Embedding Function (E_M): refers to the algorithm used by **Alice (sender)** to hide secret message into the cover-object and produce stego-object.

Extraction Function (E_X): refers to the algorithm used by **Bob (receiver)** to extract secret message from the stego-object.

Generally, the ***embedding function (E_M)*** and ***extraction function (E_X)*** are public, i.e. known by **Wendy (warden)**

Q2: Why steganography is important? Discuss.

Answer:

Steganography is used to provide **privacy** and **authenticity** of sensitive data.

Sensitive data can be hidden in insensitive data before storing in *Cloud data centre (CDC)* **(provides privacy)**

For example, patient personal information can be hidden in patient medical record to provide privacy

Data owners identification information can be hidden in data so that a data consumer can verify if the data is actually originated by the owner **(provides authenticity)**

For example, sensor ID can be hidden in signal (eg. ECG data) before sending it to *CDC* so that sensor data consumer (eg. doctor) can verify if the data is actually originated from that sensor or not.

Q3: Say, Alice wants to hide a *secret binary message (M = 1010)* in an integer number 512876. Discuss, how the message 'M' can be hidden in the above integer (i.e. embedding procedure). What is the stego integer number and stego key?

[Hints: Use the online "Decimal to Binary Converter" to convert the number to binary and binary to decimal: http://www.binaryconvert.com/convert_unsigned_int.html

Use the online "Binary to Decimal Converter" to convert the number to binary and binary to decimal:

https://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html

Select 4 random bits by your own to hide message bits.]

Answer:

- Here, length of secret message *M=1010* is 4 *(i.e. length = 4) and cover data = {*512876*}*
- *Embedding Procedure:*
  - The number in cover data is converted to *32-bit binary string:*

    00000000000001111101001101101100

  - Alice will hide each bit of the *secret binary message* in randomly selected locations of the binary strings.
  - Assume that the left most bit location is '1' and right most bit location is '32'. Now, the randomly selected locations are (from left): 18th, 20th , 27th and 30th bits of 00000000000001111101001101101100. *(i.e. locations = {18,20,27,30})*
  - Therefore, Stego-Key (S_K) becomes:

    S_K = <{18,20,27,30}>

  - The selected locations are highlighted as follows:

    00000000000001111101001101101100

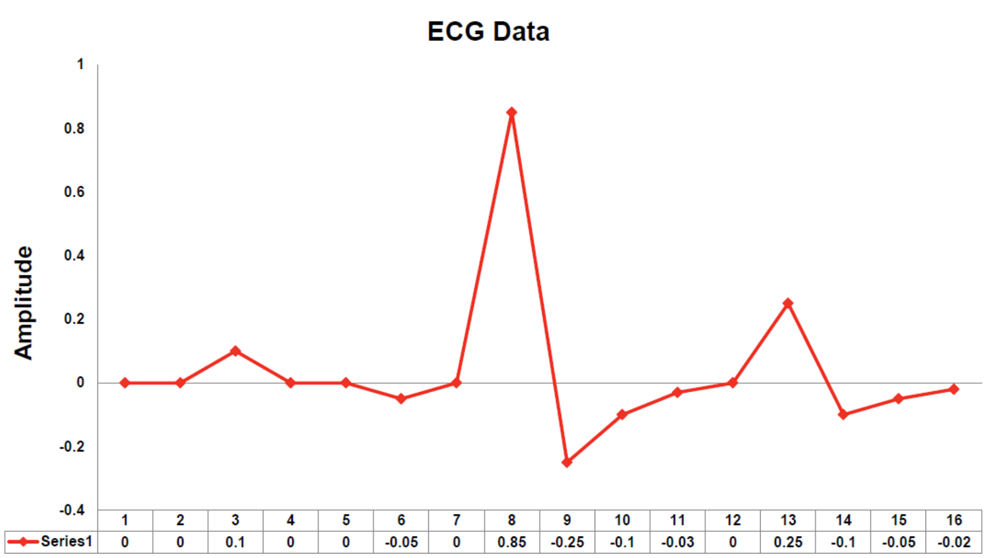
    In the above bit string, the left most bit of ***M*** is embedded in the 18th location of, the next bit is embedded in 20th location, and so on. We get the new bit string as follows:

    00000000000001111100001101101000

    The decimal value of the above binary string is **508776**

    Therefore, the stego integer number is 508776 and stego-key, S_K= <{18,20,27,30}>

<u>**Q4:**</u> Assume that Alice wants to hide a secret binary message (**M = 00110**) in an ECG Signal with **16 samples** as shown below:



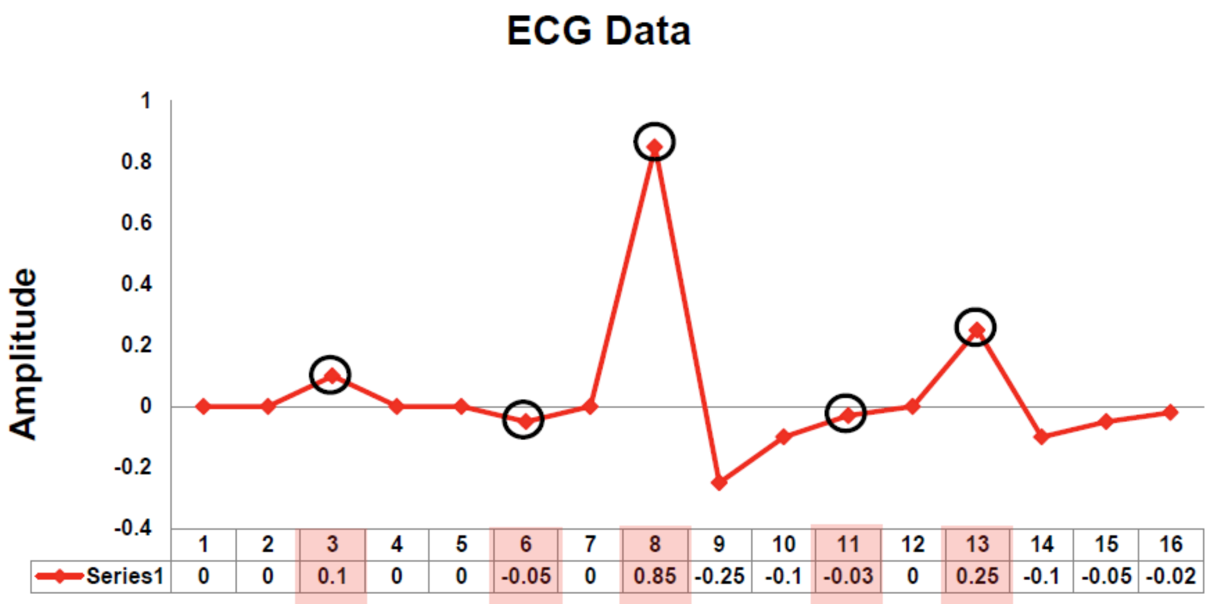
Hence, the **cover data** becomes: **C = {0, 0, 0.1, 0, 0, -0.05, 0, 0.85, -0.25, -0.1, -0.03, 0, 0.25, -0.1, -0.05, -0.02}**. Show how Alice can hide **secret binary message (M)** within **cover data (C)**. What would be the **stego data (S)** and **stego key (S_K)** that would be sent to Bob? Discuss, how Bob can extract secret message from the above five stego integers.

[Hints: Convert each ECG sample into 32-bit binary string using online calculator and use LSB of corresponding binary string to hide a bit of secret message].

Answer:

Here, length of secret message ***(M = 00110)*** is **5** *(i.e. length = 5) and cover data = {0, 0, 0.1, 0, 0, -0.05, 0, 0.85, -0.25, -0.1, -0.03, 0, 0.25, -0.1, -0.05, -0.02}*

- ***Embedding Procedure:***
  - Alice will **randomly** select **five** samples of cover ECG data in ascending order.
  - Let, randomly selected samples are: 3rd, 6th, 8th, 11th, and 13th elements of ECG signal**.** (i.e. locations = {3, 6, 8, 11, 13}). [See the following Figure.]



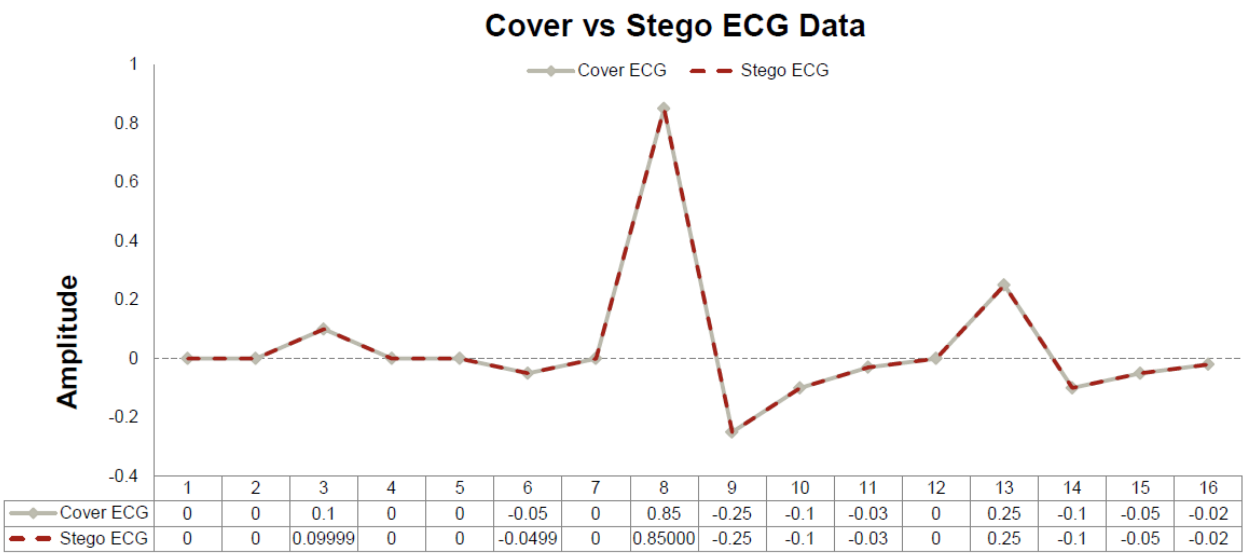
  - Each selected sample in cover ECG data is converted to 32-bit binary string (using IEEE-754 Single Precision) to obtain 5 binary strings. Use the following link for converting Floating Point Number into Binary String: https://www.h-schmidt.net/FloatConverter/IEEE754.html

  - Each secret bit is embedded in LSB of corresponding binary string and converted to floating point number as follows:

Sample Number	***Sample Value***	***Binary String***	***Secret Bit to Hide in LSB***	***Stego Binary***	***Stego Number***
3	0.1	00111101110011001100110011001101	0	00111101110011001100110011001100	0.099999994
6	-0.05	10111101010011001100110011001101	0	10111101010011001100110011001100	-0.049999997
8	0.85	00111111010110011001100110011010	1	00111111010110011001100110011011	0.8500001
11	-0.03	10111100111101011100001010001111	1	10111100111101011100001010001111	-0.03
13	0.25	00111110100000000000000000000000	0	00111110100000000000000000000000	0.25

- Therefore**, Stego-Key (S_K)** becomes: **S_K = <length, locations> = <5, {3,6, 8,11,13}>**
- Hence, stego data:
  S = {0, 0, 0.099999994, 0, 0, - 0.049999997, 0, 0.8500001, -0.25, -0.1, -0.03, 0, 0.25, -0.1, -0.05, -0.02}
- **S_K** and **S** are sent to Bob.

A comparison of cover ECG signal (C) and Stego ECG signal (S) is shown below:



As it is extremely difficult to distinguish the difference between the cover ECG signal (C) and Stego ECG signal (S), the existence of the secret message cannot be detected.

- ***Embedding Procedure:***
  - Bob receives **S_K** and **S** from Alice and wants to extract secret message from stego data (**S**).
  - Bob knows from **S_K** that the length of secret message is **5**
  - According to **S_K**, samples that contain a secret bit are: **3rd, 6th, 8th, 11th, and 13th** elements of stego ECG signal (**S**). (i.e. locations = {3, 6, 8, 11, 13}).
  - Each selected sample in stego ECG signal (**S**) is converted to 32-bit binary string and LSB of corresponding binary string is extracted to construct secret message.
  - Secret bits are extracted as follows:

Sample Number	***Sample Value***	***Binary String***	***Secret Bit from LSB***
3	0.099999994	00111101110011001100110011001100	0
6	-0.049999997	10111101010011001100110011001100	0
8	0.8500001	00111111010110011001100110011011	1
11	-0.03	10111100111101011100001010001111	1
13	0.25	00111110100000000000000000000000	0

  - Therefore, extracted secret message, **M_X =** ***00110***

Task-1 (Hiding Secret Message within HTML file)

Say, Alice has a HTML file “Text_Cover.html” that looks like as below when opened in a Web Browser:

The source of the HTML file is as follows:

```
<font color=#000000>"The time has come," the Walrus said,</font><br>

<font color=#000000>"To talk of many things: </font><br>

<font color=#000000>Of shoes and ships and sealing wax </font><br>

<font color=#000000>Of cabbages and kings </font><br>

<font color=#000000>And why the sea is boiling hot </font><br>

<font color=#000000>And whether pigs have wings." </font><br>
```

Assume that Alice has a secret message **M = 010000010100001001000011**. Now, she hides the **M** in the HTML file to produce **stego HTML file**. Next, Alice sends the stego HTML file to Bob. Bob extracts the secret message **M** from the stego HTML file.

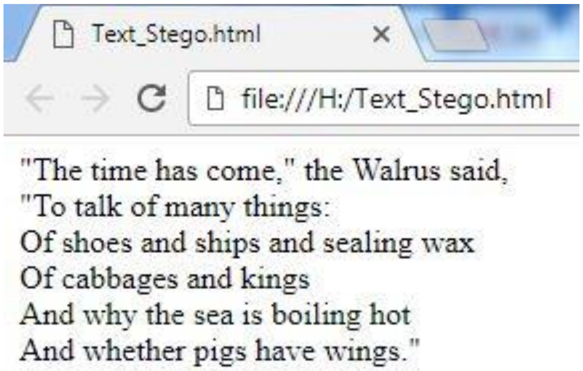
Embedding Procedure (By Alice):

- Here, length of secret message is **24** *(i.e. length = 24) and cover data = {“*Text_Cover.html*”}*
- Alice fragments 24 bits secret message into 4 segments of binary strings. Each segment has **6 bits** (as ***font-color*** takes **6 bit binary value**).
- The segments are: ***010000, 010100, 001001, 000011***
- Alice sets message segments as the font-color of 1^{st}, 2^{nd} , 5^{th} and 6^{th} lines as follows:

```
<font color=#010000>"The time has come," the Walrus said,</font><br>

<font color=#010100>"To talk of many things: </font><br>

<font color=#000000>Of shoes and ships and sealing wax </font><br>

<font color=#000000>Of cabbages and kings </font><br>

<font color=#001001>And why the sea is boiling hot </font><br>
```

```
<font color=#000011>And whether pigs have wings." </font><br>
```

The stego HTML file (**Text_Stego.html**) is obtained as follows:



- Therefore, **Stego-Key (S_K)** becomes: ***$S_K = <lines> = <\{1,2,5,6\}>$***

Alice sends **Text_Stego.html** and ***$S_K = <\{1,2,5,6\}>$*** to Bob.

Compare the Cover HTML file and Stego HTML file. Do you find any difference?

Extraction Procedure:

- Bob retrieves the colour codes from the lines of source files of **Text_Stego.html** as per given in S_K.
- The retrieved segments are: 010000, 010100, 001001, 000011
- The secret message is obtained as: **M_X = 010000010100001001000011**