Reading Headers

offset	size	description
0	2	signature, must be 4D42 hex
2	4	size of BMP file in bytes (unreliable)
6	2	reserved, must be zero
8	2	reserved, must be zero
10	4	offset to start of image data in bytes
14	4	size of BITMAPINFOHEADER structure, must be 40
18	4	image width in pixels
22	4	image height in pixels
26	2	number of planes in the image, must be 1
28	2	number of bits per pixel (1, 4, 8, or 24)
30	4	compression type (0=none, 1=RLE-8, 2=RLE-4)
34	4	size of image data in bytes (including padding)
38	4	horizontal resolution in pixels per meter (unreliable)
42	4	vertical resolution in pixels per meter (unreliable)
46	4	number of colors in image, or zero
50	4	number of important colors, or zero

Offsets represent the index value and size represents the number of bytes the respective detail of the image is using to store its value. For example, the starting 2 bytes are reserved for the format of the image (BM - 424D(hex)). Like this, the value of width is store between the offset 18 and 22 i.e. 4 bytes are reserved for width. Similarly, we can read any value included in headers by reading the correct bytes with the help of the above table.

This is the small part of the Hexdump of tiger.bmp.

```
umangkumar@Umangs-Air desktop % cd test/Stegano
umangkumar@Umangs-Air Stegano % xxd tiger.bmp
00000000: 424d 3884 0300 0000 0000 3600 0000 2800  BM8.......6...(.
00000010: 0000 4001 0000 f000 0000 0100 1800 0000  ..@.............
00000020: 0000 0284 0300 120b 0000 120b 0000 0000  ................
00000030: 0000 0000 0000 6379 805a 7382 4e68 764d  ......cy.Zs.NhvM
00000040: 666e 4b65 6e51 6d72 5a73 7b56 6e7a 4866  fnKenQmrZs{VnzHf
00000050: 7041 6670 4d69 7052 6c71 596f 755e 7275  pAfpMipRlqYou^ru
00000060: 5a6d 7054 6d72 576d 7552 6a74 4d68 713a  ZmpTmrWmuRjtMhq:
00000070: 535e 324f 5a33 4f59 2e43 4b35 464a 3343  S^2OZ3OY.CK5FJ3C
00000080: 4b2c 4148 2842 4c2d 4754 364c 564c 5f68  K,AH(BL-GT6LVL_h
00000090: 526a 7248 626d 4059 615c 686c 5865 6839  RjrHbm@Ya\hlXeh9
```

In the last row starting with BM8… this is the starting of header bytes and ends at the offset 54, the . before cy is the last byte of the headers.

How we store data in files.

To store data we use the bit masking technique.

Tiger.bmp is the original image and out_image.bmp is the image containing a secret message.

This is the small part of the Hexdump of out_image.bmp

```
umangkumar@Umangs-Air Stegano % xxd out_image.bmp
00000000: 424d 3884 0300 0000 0000 3600 0000 2800  BM8.......6...(.
00000010: 0000 4001 0000 f000 0000 0100 1800 0000  ..@.............
00000020: 0000 0284 0300 120b 0000 120b 0000 0000  ................
00000030: 0000 0000 0000 6278 805a 7283 4f68 764d  ......bx.Zr.OhvM
00000040: 676e 4a64 6f51 6c73 5b73 7a56 6f7b 4867  gnJdoQls[szVo{Hg
00000050: 7140 6670 4d68 7052 6d71 586e 745f 7274  q@fpMhpRmqXnt_rt
00000060: 5b6d 7054 6c72 566c 7553 6a74 4c69 713b  [mpTlrVluSjtLiq;
00000070: 535f 334e 5a32 4e59 2e42 4b34 464b 3243  S_3NZ2NY.BK4FK2C
00000080: 4b2c 4049 2942 4c2c 4754 364c 564c 5e69  K,@I)BL,GT6LVL^i
00000090: 536a 7348 626c 4059 615c 686d 5865 6838  SjsHbl@Ya\hmXeh8
000000a0: 4d52 2c3e 462e 4247 3d54 5b4c 5e6a 4a61  MR,>F.BG=T[L^jJa
```

Things are embedded in the image in the order len(password), password, len(secret message), secret message.

In this case, we use the password “csb101” (length 6) to protect the message. So first, the number 6 will be put in the image.

Every byte can be represented in 8 bits. We will mask the 8 **BITS** of the number 6 on the first 8 **BYTES**.

6 - 00000110 (8-bit representation)
The first 8 bytes in the images are:
c - 01100011
y - 01111001
. - 00101110
Z - 01011010
s - 01110011
. - 00101110
N - 01001110
h - 01101000

Now mask every bit of 6 on the last bit of bytes.

c - 01100011	mask	0	=>	01100010	- b
y - 01111001	mask	0	=>	01111000	- x
. - 00101110	mask	0	=>	00101110	- .
Z - 01011010	mask	0	=>	01011010	- z
s - 01110011	mask	0	=>	01110010	- r
. - 00101110	mask	1	=>	00101111	- /
N - 01001110	mask	1	=>	01001111	- 0
h - 01101000	mask	0	=>	01101000	- h

The first 8 bytes (cy.Zs.Nh) are replaced by (bx.zr/oh) you may have also observed it in the hex dump of out_image.bmp.
So likewise we stored every value one by one and bit by bit.
To decode we get the last bit of every byte then combine the bits then present their corresponding ASCII value.

Thank you for reading.
Umang Kumar