Image Steganography- CONT.

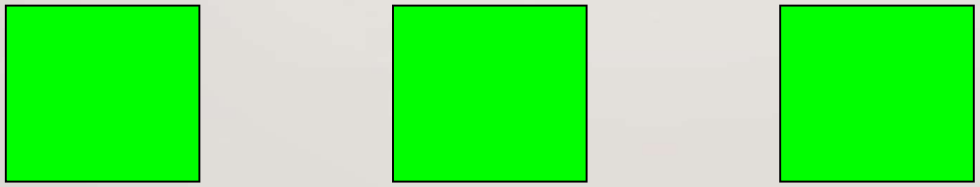
- An image is represented as an N*M (in case of greyscale images) or N*M*3 (in case of colour images) matrix in memory, with each entry representing the intensity value of a pixel.
- In image steganography, a message is embedded into an image by altering the values of some pixels, which are chosen by an encryption algorithm.
- The recipient of the image must be aware of the same algorithm in order to known which pixels he or she must select to extract the message.

LEAST-SIGNIFICANT BIT (LSB) EMBEDDING - TERMINOLOGY

- *Message* = the secret information we want to hide
- *Cover image* = image used to hide the message in
- *Stego-image* = the cover image with the message embedded

LEAST-SIGNIFICANT BIT (LSB) - CONCEPT

- Which color is different?



- In (R,G,B) left and right are (0,255,0)
- Center one is (0,254,0)
- We can use the LSB to hold info, since it looks the same either way!

LSB 24-BIT BITMAPS

- In 24-bit bmps, each pixel represented by 3 bytes (RGB)
- Use lsb of each byte to hold a bit of message
- Example: LSB 24-bit Bitmaps

  Message = 'f' = $0110\ 0110_2$

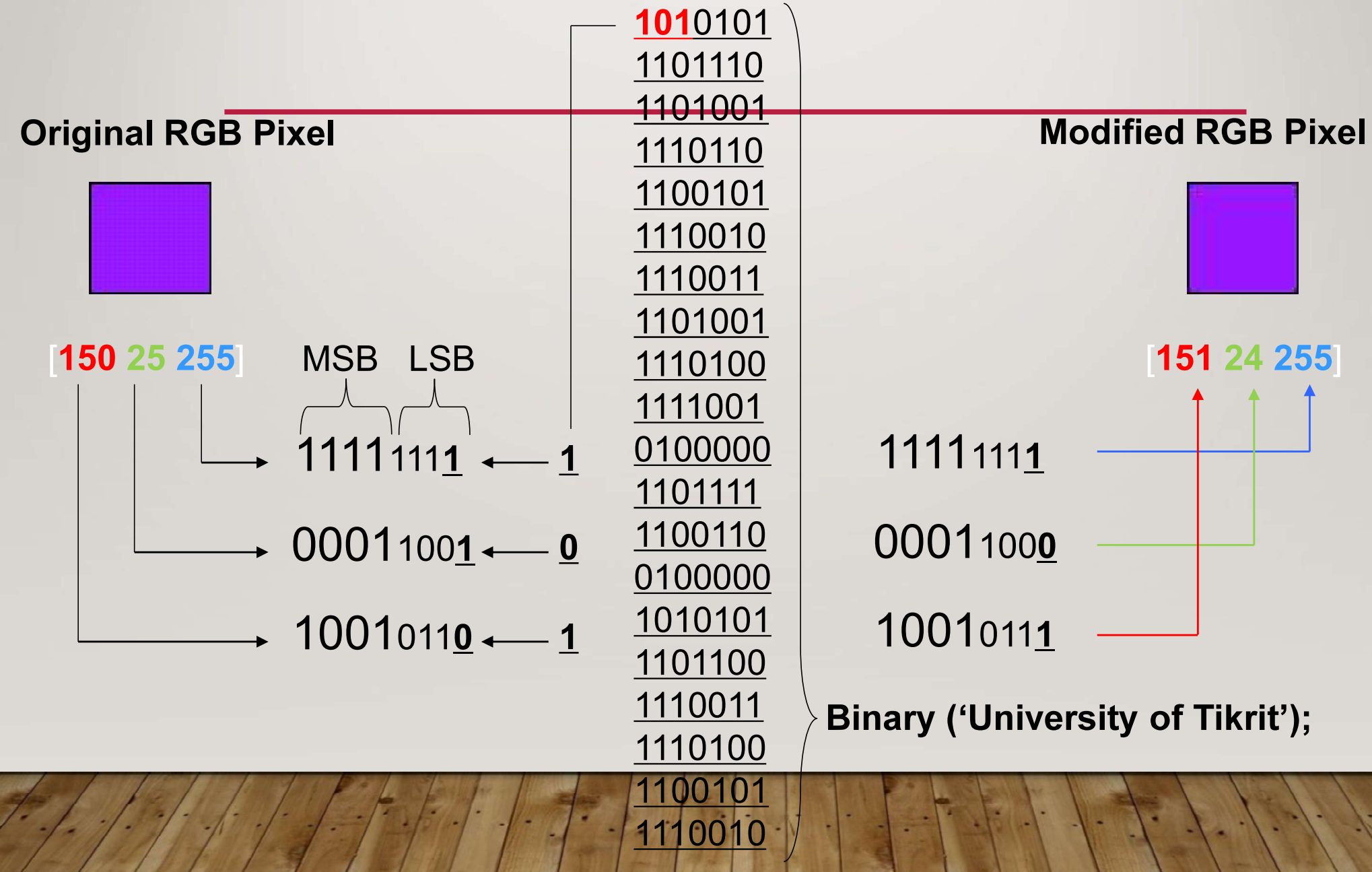
  Cover Image:

  FF FF FF 00 00 00 FF FF …

  Stego-image:

  FE FF FF 00 00 01 FF FE …

Simple Example



LSB - programming- Grayscale

- The pseudo-code below can be used to explain the processi ng to embed a text message in a grayscale image by replaci ng the LSB of each pixel:

```
pic=cover image
msg=secret message
n = number of chars in msg
for i=1 to n
  get char from msg
  for each bit in char
    get a pixel from pic
    if the bit=1
      insert a 1 in the least significant bit of the pixel
    else
      insert a 0 in the least significant bit of the pixel
      replace the pixel in pic
  end for
end for
```

LSB - programming- Color CONT.

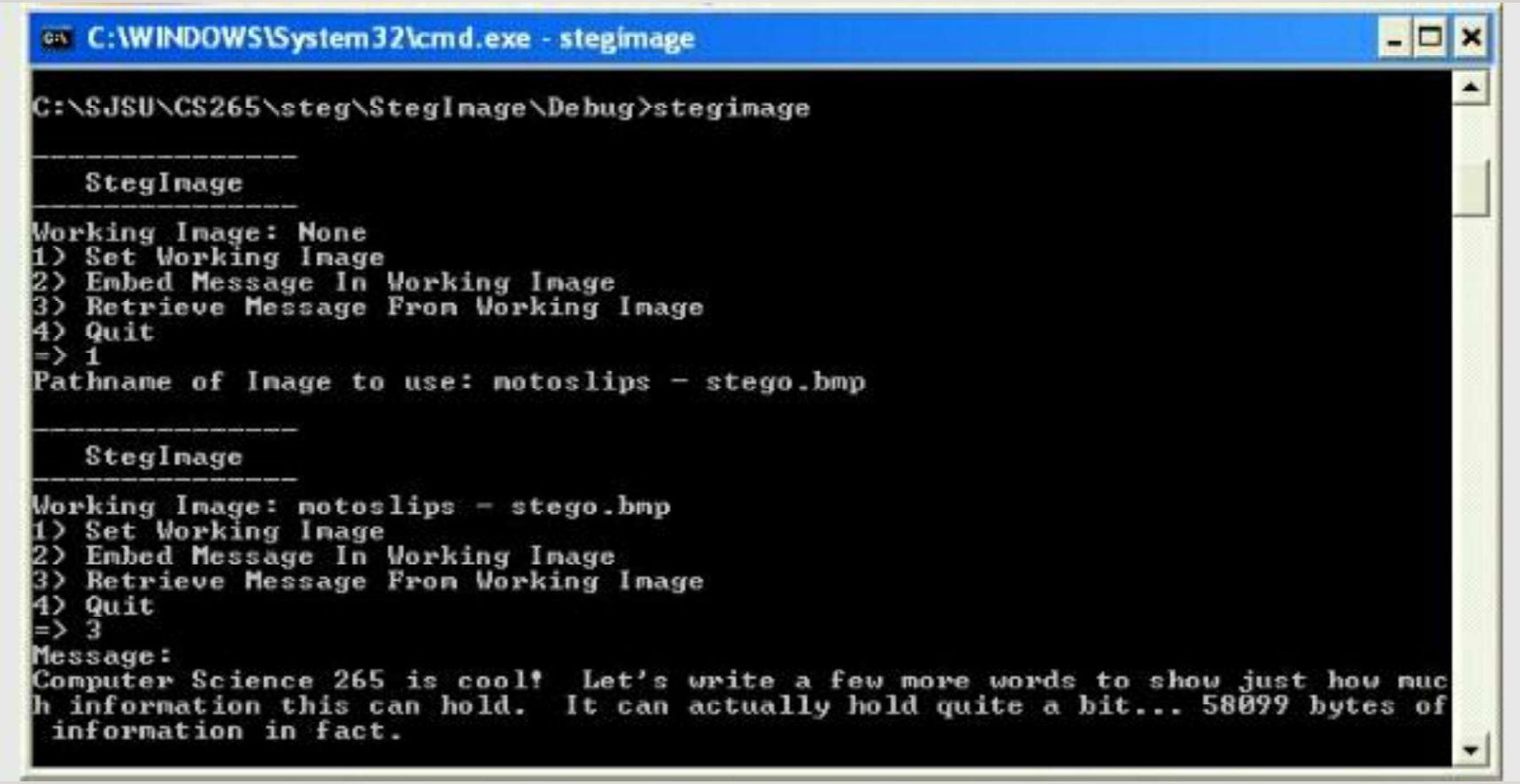
- The pseudo-code below can be used to explain a simple LS B insertion algorithm used with 24-bit images:

```
pic=cover image
msg=secret message
n=number of chars in msg
for i=1 to n
  get char from msg
  for each 3 bits in char
    get a pixel from pic
    get the red value of the pixel
```

LSB - programming CONT.

```
    if the first bit=1
      insert a 1 in the least significant bit of the red value
    else
      insert a 0 in the least significant bit of the red value
      get the green value of the pixel
    if the second bit=1
      insert a 1 in the least significant bit of the green
      value
    else
      insert a 0 in the least significant bit of the green
      value
      get the blue value of the pixel
    if the third bit=1
      insert a 1 in the least significant bit of the blue
      value
    else
      insert a 0 in the least significant bit of the blue
      value
      replace the value in pic
    end for
end for
```

LSB - Implementation...CONT.



8 bits

- Don’t hold direct color values
- Do hold offsets into a palette
- Can’t just change lsb, because adjacent colors in palette ma y not be similar