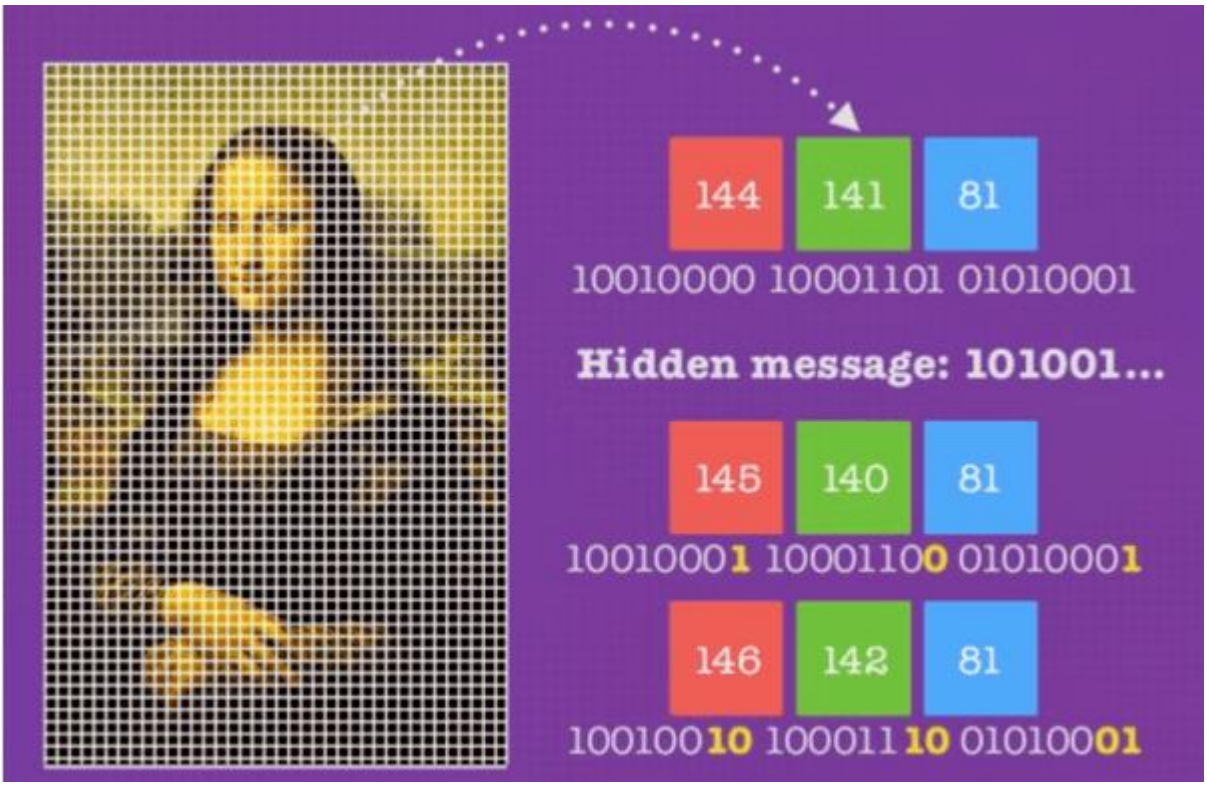
Image Steganography (LSB)

Khebchi Abdallah G2 ISI

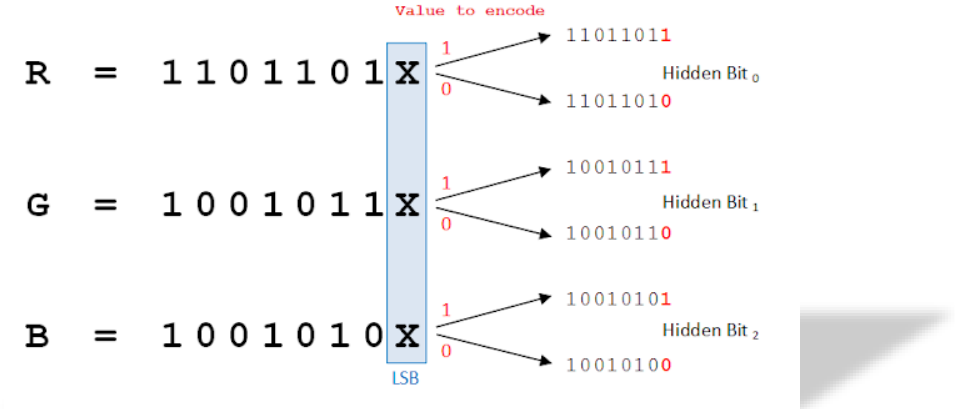
1. Introduction

La stéganographie est le processus consistant à cacher un message secret dans un message plus grand de manière à ce que quelqu'un ne puisse pas connaître la présence ou le contenu du message caché. Bien que la stéganographie ne soit pas à confondre avec le cryptage, qui est le processus consistant à rendre un message inintelligible, la stéganographie tente de cacher l'existence de la communication.

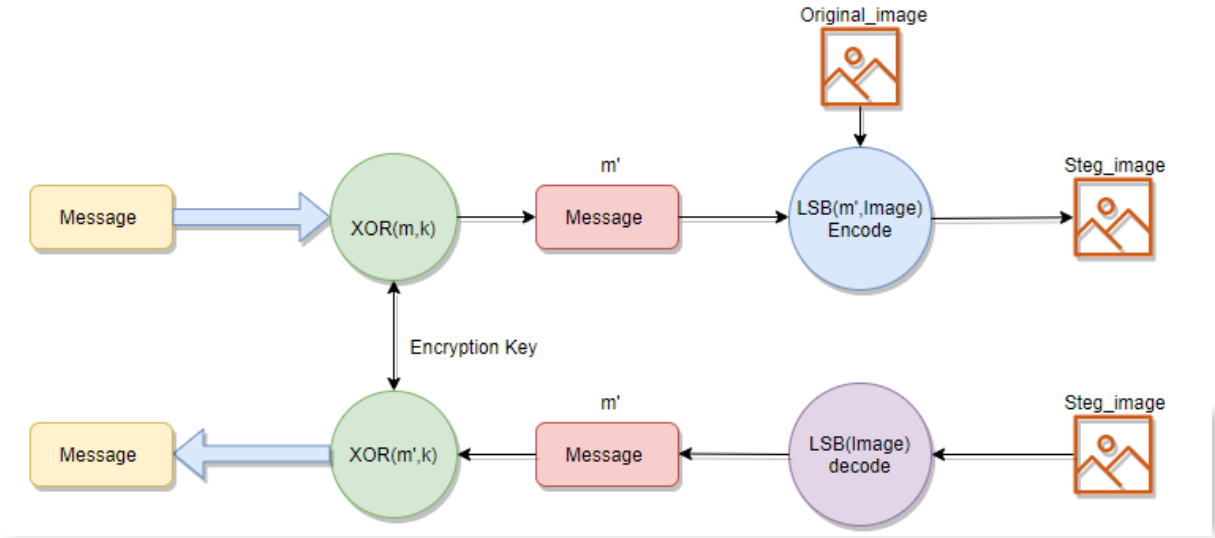


2. LSB

The **Least Significant Bit** (**LSB**) **method** est un moyen d'intégrer des informations secrètes. En remplaçant la valeur de pondération minimale d'un pixel échantillonné par des bits binaires de données d'informations secrètes.



LSB method with XOR encryption diagram



3. **LSB Encoding implementation**

Step 1 : Convertir chaque caractère en bits

```
func Byte2Bin(b []byte) (bin string) {
    for _, bb := range b {
        bin = fmt.Sprintf("%s%.8b", bin, bb)
    }
    return
}
```

Step2 : Convertir l'image en RGBA format

```
func Image2Rgba(im image.Image) *image.RGBA {
    bs := im.Bounds()
    n := image.NewRGBA(image.Rect(0, 0, bs.Dx(), bs.Dy()))
    draw.Draw(n, n.Bounds(), im, bs.Min, draw.Src)
    return n
}
```

Step 3 : pour chaque point RGBA dans l'image, remplacez le dernier bit par le bit de message

```
// convert image into rgb pattern
rgbIm := utils.ImToRgba(im)
for y := 0; y < h; y++ {
    for x := 0; x < w; x++ {
        col := rgbIm.RGBAAt(x, y)
        if index+1 <= l {
            col.R = addBite(col.R, bm[index])
        }
        if index+2 <= l {
            col.G = addBite(col.G, bm[index+1])

        }
        if index+3 <= l {
            col.B = addBite(col.B, bm[index+2])
        }
        rgbIm.SetRGBA(x, y, col)
        index += 3

    }
}
```

4. **Lsb Decoding implementation :**

   Step 1 : Convertir l'image en RGBA format

   Step2 : Extraire le message des pixels de l'image en utilisant l'encodage utf8.

```go
w, h := im.Bounds().Dx(), im.Bounds().Dy() // get image dims
rgbObj := utils.ImToRgba(im) // image → RGBA
var (
    msgL     int = 0 // message length
    msgSl    string
    charBuff int   = 0 // bit extracted from pixle
    n        uint8 = 0 // encoding  charBuff
)
for y := 0; y < h; y++ {
    for x := 0; x < w; x++ {
        rgbaColorPos := rgbObj.RGBAAt(x, y) // get RGBA point
        for _, c := range []uint8{rgbaColorPos.R, rgbaColorPos.G, rgbaColorPos.B} {
            lsb := c & 1 // get LSB
            charBuff += int(lsb << (7 - n)) // bit shifting into the buffer
            if n == 8 {
                b := byte(charBuff)
              // get message length from header
                if b > 47 && b < 58 && len(message) == 0 {
                    msgSl += string(b)
                } else {
                    msgL, _ = strconv.Atoi(msgSl)
                    if len(message) == msgL {
                        goto fin
                    }
                    message += string(b)
                }
                // another char finding and decoding.
                n, charBuff = 0, 0
            }
            n++
        }}}
```

5. **Image Steganography LSB Avantages et inconvénients:**

   Advantages :

   - Cette méthode est très rapide et facile à mettre en œuvre par rapport aux autres méthodes de stéganographie d'images.
   - L'image de sortie présente une très légère différence par rapport à l'image d'entrée.
   - Cette méthode constitue les bases de nombreux autres algorithmes complexes.

   Inconvénients :

   - Ce type de codage des données est faible car il peut être facilement décodé en prenant les LSB de l'image et en obtenant le message au format binaire. (le cryptage est nécessaire)
   - Lors de l'intégration du message dans plusieurs LSB, la qualité de l'image peut diminuer en fonction du nombre de pixels modifiés. (taille du message et de l'image)