# Naive steganography

Ho Dac Hung

# Contents

- LSB embedding
- Steganography in palatte images

# 1. LSB embedding

- Arguably, LSB embedding is the simplest steganographic algorithm. It can be applied to any collection of numerical data represented in digital form.

# 1. LSB embedding

- Let us assume that $x[i] \in X = \{0,1,2,\dots,2^{n_c}-1\}$ is a sequence of integers.
- Depending on the image format and the bit depth chosen for representing the individual values, each $x[i]$ can be represented using nc bits $b[i,1], b[i,2], \dots, b[i,n_c]$,

$$x[i] = \sum_{k=1}^{n_c} b[i,k]2^{n_c-k}$$

# 1. LSB embedding

- LSB embedding, as its name suggests, works by replacing the LSBs of $x[i]$ with the message bits $m[i]$, obtaining in the process the stego image $y[i]$.

# 1. LSB embedding

```
Path = Perm(n);
y = x;
m = min(m, n);
for i = 1 to m {
    y[Path[i]] = x[Path[i]] + m[i] – x[Path[i]] mod 2;
}
```

# 1. LSB embedding

```
Path = Perm(n);
for i = 1 to m {
    m[i] = y[Path[i]] mod 2;
}
```

# 1. LSB embedding

- The amplitude of changes in LSB embedding is 1. because natural images contain a small amount of noise due to various noise sources present during image acquisition the LSB plane of raw, never-compressed natural images already looks random.

# 1. LSB embedding

| | Frequency of occurrence |
|---|---|
| Bits | 0.49942, 0.50058 |
| Pairs | 0.24958, 0.24984, 0.24984, 0.25073 |
| Triples | 0.1246, 0.1250, 0.1247, 0.1251, 0.1250, 0.1249, 0.1251, 0.1256 |

# 1. LSB embedding

- The data are consistent with the claim that the LSB plane is random. Even though this is not a proof of randomness, the argument is convincing enough to make us intuitively believe that any attempts to detect the act of randomly flipping a subset of bits from the LSB plane are doomed to fail.
- This seemingly intuitive claim is far from truth because LSB embedding in images can be very reliably detected.

# 1. LSB embedding

- Even if the LSB plane of covers was truly random, it may still be possible to detect embedding changes due to flipping LSBs if, for example, the second LSB plane b[i, nc − 1] and the LSB plane were somehow dependent! In the most extreme case of dependence, if b[i, nc − 1] = b[i, nc] for each i, detecting LSB changes would be trivial. All we would have to do is to compare the LSB plane with the second LSB plane.

# 1. LSB embedding

- The embedding operation of flipping the LSB can be written mathematically in many different ways:

$$\begin{aligned}\mathrm{LSBflip}(x) &= \begin{cases} x+1 & \text{when } x \text{ even} \\ x-1 & \text{when } x \text{ odd} \end{cases} \\ &= x + 1 - 2(x \bmod 2) \\ &= x + (-1)^x .\end{aligned}$$

# 1. LSB embedding

- LSB embedding also induces $2^{n_c} - 1$ disjoint LSB pairs on the set of all possible element values $\{0,1,2,\dots,2^{n_c}-1\}$,

$$\{0,1\},\{2,3\},\dots,\{2^{n_c}-2,2^{n_c}-1\}$$

# 1. LSB embedding

- Note that if x[i] is in LSB pair {2k, 2k + 1}, it must stay there after embedding because the pair elements differ only in their LSBs ($2k \leftrightarrow 2k + 1$). This simple observation is the starting point of many powerful attacks on LSB embedding.

# 1. LSB embedding

- For any steganographic method, it is often valuable to mathematically express the impact of embedding on the image histogram. Many steganographic techniques introduce characteristic artifacts into the histogram and these artifacts can be used to detect the presence of secret messages.

# 1. LSB embedding

- Let $h[j] = 0,1, \ldots, 2^{n_c} - 1,$ be the histogram of elements from the cover image

$$h[j] = \sum_{i=1}^{n} \delta(x[i] - j)$$

# 1. LSB embedding

- We will assume that Alice is embedding a stream of m random bits.
- We denote by $\alpha = m/n$ the relative payload Alice communicates.

# 1. LSB embedding

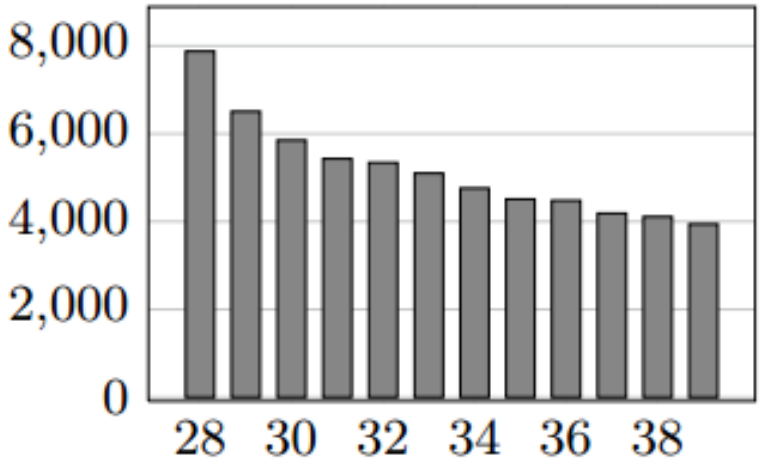
- Because during LSB embedding the pixel values within one LSB pair $\{2k, 2k+1\}$, $k = 0,1, \ldots, 2^{n_c} - 1$, are changed into each other but never to any other value, the sum $h_\alpha[2k] + h_\alpha[2k+1]$ stays unchanged for any α and thus forms an invariant under LSB embedding.

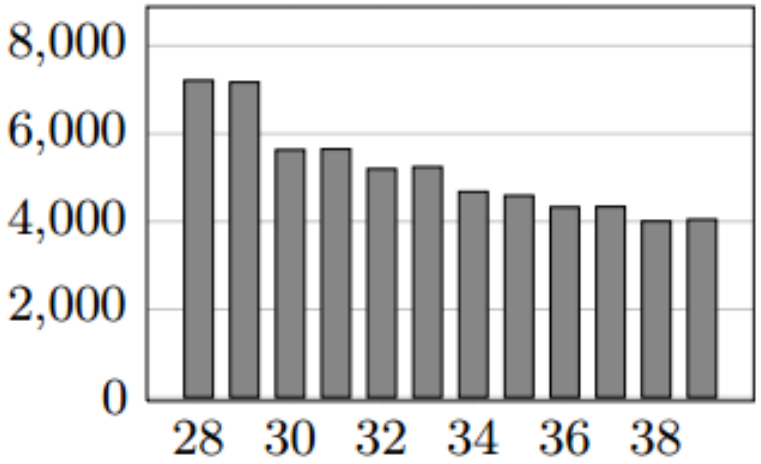
# 1. LSB embedding

- We say that LSB embedding has a tendency to even out the histogram within each bin. This leads to a characteristic staircase artifact in the histogram of the stego image, which can be used as an identifying feature for images fully embedded with LSB embedding. This observation is quantified in the so-called histogram attack

# 1. LSB embedding

# 1. LSB embedding



# 1. LSB embedding

- In a fully embedded stego image (α = 1), we expect:

$$h_\alpha[2k] \approx \bar{h}[2k] = \frac{h_\alpha[2k] + h_\alpha[2k+1]}{2}$$

# 1. LSB embedding

- The histogram attack amounts to the following composite hypothesistesting problem:

$$H_0\colon\ h_\alpha \sim \bar{h}$$
$$H_1\colon\ h_\alpha \nsim \bar{h}$$

- Which we approach using Pearson's chi-square test. This test determines whether the even grayscale values in the stego image follow the known distribution $\bar{h}$.

# 1. LSB embedding

- The chi-square test first computes the test statistic S

$$S = \sum_{k=0}^{d-1} \frac{(h_\alpha[2k] - \bar{h}[2k])^2}{\bar{h}[2k]}$$

# 1. LSB embedding

- One can intuitively see that if the even grayscales follow the expected distribution, the value of S will be small, indicating the fact that the stego image is fully embedded with LSB embedding. Large values of S mean that the match is poor and notify us that the image under inspection is not fully embedded.

# 2. Steganography in palatte images

- Palette images, such as GIF or the indexed form of PNG, represent the image data using pointers to a palette of colors stored in the header. It is possible to hide messages both in the palette and in the image data.

# 2. Steganography in palatte images

- Reordering the image palette and reindexing the image data correspondingly is a modification that does not change the visual appearance of the image. Thus, it is possible to hide short messages as permutations of the palette.

# 2. Steganography in palatte images

- For most palette images, it is not possible to apply LSB embedding directly to the image colors because new colors that are not in the palette would be obtained and, once the total number of unique colors exceeds 256, the image could no longer be stored as a GIF.

# 2. Steganography in palatte images

- One simple solution to this problem is to preprocess the palette before embedding by decreasing the color depth to 128, 64, or 32 colors. This way, when the LSBs of one, two, or three color channels, respectively, are perturbed, the total number of newly created colors will be at most 256.

# 2. Steganography in palatte images

- Many problems associated with applying LSB-like embedding to palette images can be somewhat alleviated by presorting the palette colors so that neighboring colors are close and applying simple LSB embedding to the pointers.

# 2. Steganography in palatte images

- There were attempts to eliminate this problem by ordering the palette using more sophisticated algorithms that would minimize the differences between neighboring palette colors in the palette. The task of finding the optimal ordering leads to the traveling-salesman problem, which is known to be NP-complete

# 2. Steganography in palatte images

- It should be clear that we do not really need to order the palette to obtain the minimal embedding distortion. In fact, all that is needed is to assign parities (bits 0 and 1) to each palette color so that the closest color to each palette color has the opposite parity.

# 2. Steganography in palatte images

- Let the image palette contain $N_P$ colors represented as RGB triples, $c[j] = (r[j], g[j], b[j])$ , with parities $P[j] \in \{0,1\}, j = 1, \dots, N_P$ . We define the isolation, $s[j]$, for each color $c[j]$ as the distance from $c[j]$ to its closest neighbor from the palette, $s[j] = min_{j \neq j'} d(c[j], c[j'])$.

# 2. Steganography in palatte images

1. Calculate the matrix of distances between all pairs of colors $d_{RGB}(k,l) = d_{RGB}(c[k], c[l])$. Set $P = \{\emptyset\}$.
2. Order all distances $d_{RGB}(k,l), k > l$, to a non-decreasing sequence $D = d_{RGB}(k_1, l_1) \leq d_{RGB}(k_2, l_2) \leq \cdots$ For a unique order, resolve ties, for example, using alphabetical order.
3. Iteratively repeat Step 4 until $P$ contains all $N_P$ palette colors.

# 2. Steganography in palatte images

4. Choose the next distance $d_{RGB}(k,l)$ in $D$, such that either $c_k \notin P$ or $c_l \notin P$.If no such $d_{RGB}(k,l)$ can be found, this means that P already contains all $N_P$ colors and we are done.