Multimedia security and privacy

Section 1

(1.1) Explain the difference in the usage of and requirements to digital robust watermarking steganography and tamper proofing.

- Digital data hiding or steganography:
  - Embed information in any multimedia file
  - Position of embed information is secrete
  - Only decodable if you possess the embedding key
  - Invisible from the perceptual and stochastic point of view
  - Message is content independent
- General algorithm:
  - Requirements: media file x, private keys (k1, k2), secrete message m
  1. $\mathrm{m}_{enc} = encrypt_{k1}(m)$, for example with one time pad
  2. Convert m_{enc} values, 0 => -1, 1 => +1
  3. Generate positions (where to hide) using k2
  4. Embed encrypted data m_{enc} into the corresponding positions
- Goal:
  - Extract $m^{\wedge}$ from the multimedia file, such that: $\Pr[m = m^{\wedge}] = 1$

(1.1)

- Digital robust watermarking:
  - Embed copyright information in any multimedia file
  - Position of embed information is secrete
  - Only detectable if you possess the embedding key
  - Invisible from the perceptual point of view
  - Message is content independent
- General algorithm:
  - Requirements: media file x, private key k1
  1. Generate positions (where to hide) using k1
  2. Mark x, with mark k into generated positions
- Goal:
  - Detect if there is a watermark => yes or no answer

(1.1)

- Tamper proofing:
  - Embed information in any multimedia file
  - Position of embed information is secrete
  - Only detectable and recoverable if you possess the embedding key
  - Invisible from the perceptual point of view
  - Message is content dependent
- General algorithm:
  - Requirements: media file x, private key k1, data m
  1. Compute: $\mathrm{m}_{dep} = p(m|x)$
  2. Convert m_{dep} values, 0 => -1, 1 => +1
  3. Generate positions (where to hide) using k1
  4. Embed data m_{dep} into the corresponding positions
- Goal:
  - Detect if there were any modifications to your embedded multimedia file => verification
  - Recover original data without embedded data => recovery

(1.1)

- Attacks:
  - Different types of digital data hiding will be attacked differently
  - In digital data hiding, attacks will try to (1) detect if a an multimedia file carries a hidden message, (2) cryptanalysis to get k and m
  - In digital robust watermarking, attackers will try to break the robustness, in other words, they want to (1) remove the copyright information without losing monetary value
  - In tamper proofing, the attacks are content modifications, we want to (1) detect if there was a lossy compression applied

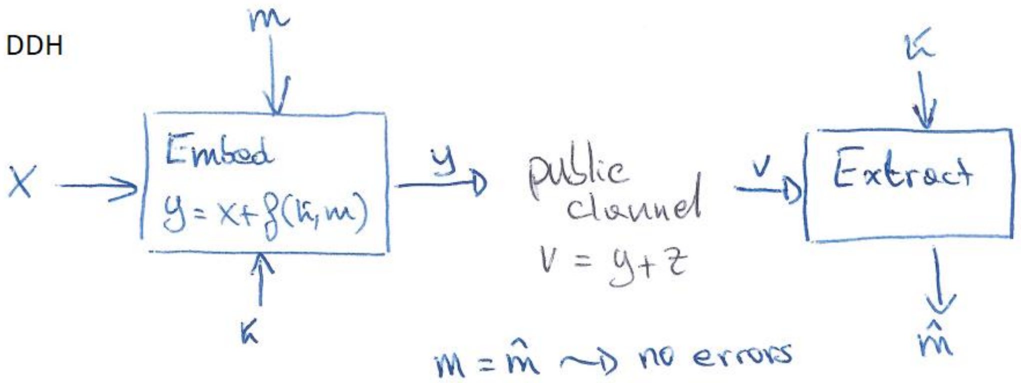
(1.2) Explain the difference between the watermarking and data hiding. Explain the block diagrams and the difference between the decoding and detection problems.

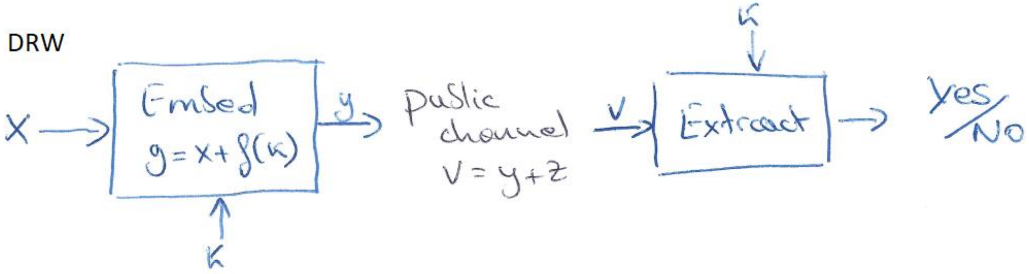
- Digital data hiding or steganography:
  - Embed information in any multimedia file
  - Position of embed information is secrete
  - Only decodable if you possess the embedding key
  - Invisible from the perceptual and stochastic point of view
  - Message is content independent
- General algorithm:
  - Requirements: media file x, private keys (k1, k2), secrete message m
  1. $\mathrm{m}_{enc} = encrypt_{k1}(m)$, for example with one time pad
  2. Convert m_{enc} values, 0 => -1, 1 => +1
  3. Generate positions (where to hide) using k2
  4. Embed encrypted data m_{enc} into the corresponding positions
- Goal:
  - Extract $m^{\wedge}$ from the multimedia file, such that: $\Pr[m = m^{\wedge}] = 1$

(1.2)

- Digital robust watermarking:
  - Embed copyright information in any multimedia file
  - Position of embed information is secrete
  - Only detectable if you possess the embedding key
  - Invisible from the perceptual point of view
  - Message is content independent
- General algorithm:
  - Requirements: media file x, private key k1
  1. Generate positions (where to hide) using k1
  2. Mark x, with mark k into generated positions
- Goal:
  - Detect if there is a watermark => yes or no answer

(1.2)





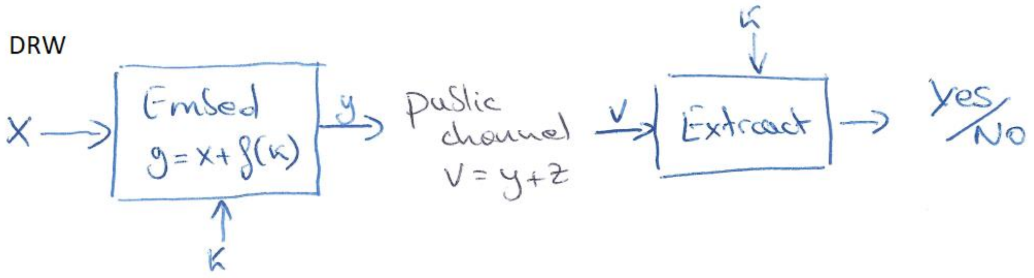
Attacks:

Different types of digital data hiding will be attacked differently

In digital data hiding, attacks will try to (1) detect if a an multimedia file carries a hidden message, (2) cryptanalysis to get k and m

In digital robust watermarking, attackers will try to break the robustness, in other words, they want to (1) remove the copyright information without losing monetary value

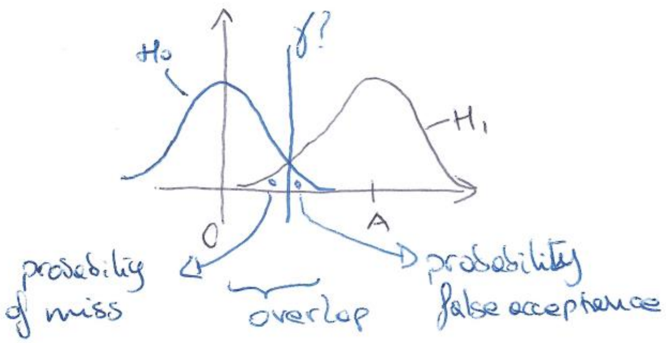
(1.3) Explain the watermark detection problem. Explain the difference between the Neyman-Pearson and Bayesian hypothesis testing. Explain the different types of errors.

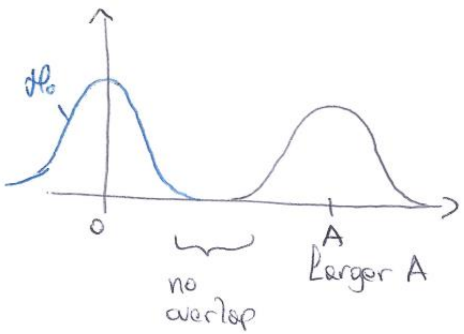


- When trying to detect if an image has been watermarked, there are two possible outcomes:
    - it hasn't been watermarked (H_0):
        1. $v = x + z$
        2. $v = x + \gamma w' + z$ $\rightarrow w'$ is a watermark, but not ours
        - $H_0: v \sim f_{v|H_0}(v|H_0)$
    - it has been watermarked (H_1):
        1. $v = x + \gamma w + z$
        - $H_1: v \sim f_{v|H_1}(v|H_1)$
    - We will treat x as random noise: $x + z = z$

(1.3)

- Let's we assume we are an authorized entity, and possess the secret embedding key k.
- From k we can compute: $w = f(k)$
- Let's solve for 1D → $v[i] = w[i] + z[i]$:
  - $H_0: v = z$ can be seen as random noise → $H_0 \sim N(0, \sigma_z^2)$
  - $H_1: v = w + z$, if $w = A$ → $\mathrm{H}_1 \sim N(A, \sigma_z^2)$
  - There may or not be an overlap between these 2 distributions
  - The larger A is, the smaller the overlap, but the it becomes less robust!
  - Because of this overlap, we need to set a decision threshold γ





- Now we could do this technique N times, and decide between H_0 and H_1 by averaging the N 1D decisions

➢ Would be as efficient as random guessing because A is rather small

(1.3)

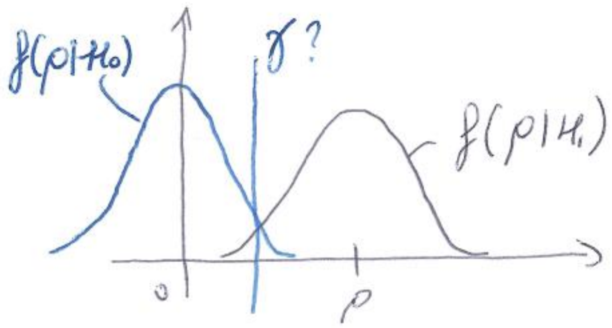
- Let’s we assume we are an authorized entity, and possess the secret embedding key k.
- From k we can compute: $w = f(k)$
- Let’s solve for ND:
    - Goal → compute likelihood ratio of watermark:

$$\rho(v) = \frac{\mathrm{f}_{\rho|H_0}(\rho|H_0)}{\mathrm{f}_{\rho|H_1}(\rho|H_1)}$$

    - It is easily computable as: (we can do this because it is a gaussian distribution)

$$\rho = \frac{1}{N}\sum v[i] * w[i], \qquad i = 0..N$$

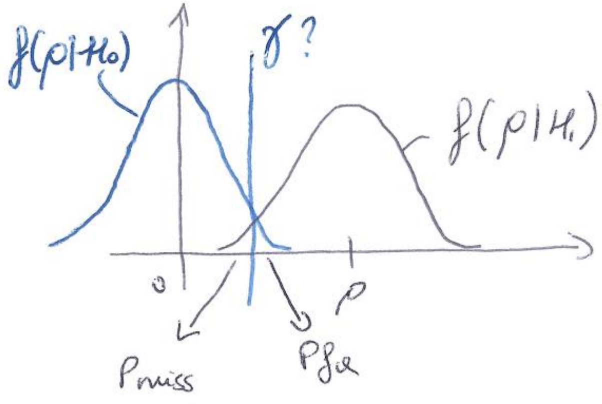
    - Now we have two distributions:
        1. $H_0 : v = z$, is distributed as → $\mathrm{f}_{(\rho|H_0)}$
        2. $H_1 : v = w + z$, is distributed as → $\mathrm{f}_{(\rho|H_1)}$



(1.3)

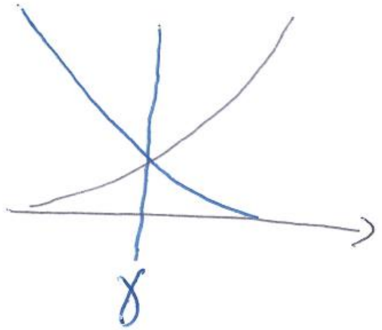
- From these 2 distributions, we can now set the threshold
- There are 2 techniques

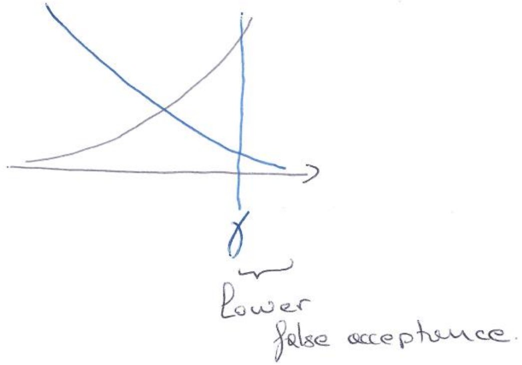
$$P_m = \int_{-\infty}^{\gamma} f(\rho|H_1)\, d\rho$$



$$P_{fa} = \int_{\gamma}^{+\infty} f(\rho|H_0)\, d\rho$$

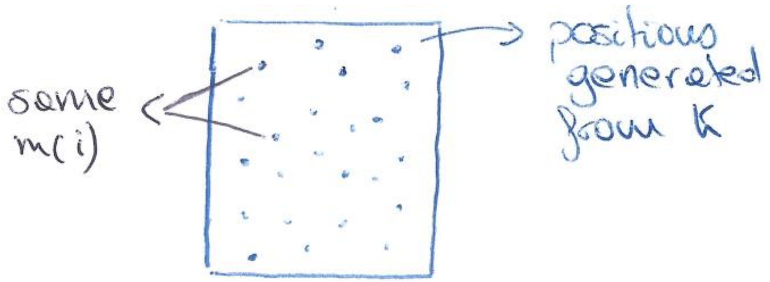
1. Bayesian:
   - we want to equally minimize both P_m and P_{fa}
2. Neyman-Pearson:
   - we choose a choose an acceptable value for either of the two errors, and don't care what happens to the other value





(1.4) Explain the geometrical synchronization problem in the digital watermarking.

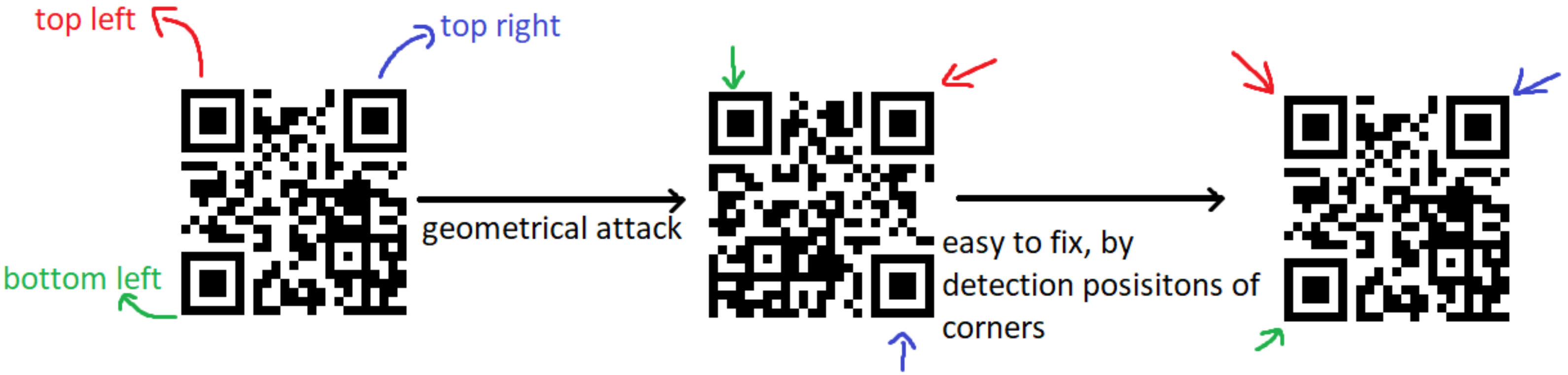
- Any person that holds the secret key k, they can generate the locations of the watermark



- However, if there is the slightest geometrical deviation, the pixels won’t be in the same place as before, for example:
    1. User isn’t hold the device/image in the same position
    2. Geometrical distortions were applied by an attacker

➢ Transform domain won’t help us either, as it isn’t invariant to geometrical operations

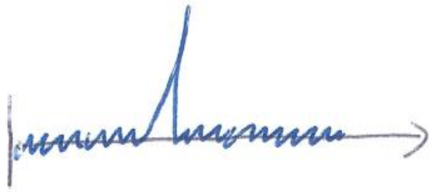
➢ QR codes fix this problem by having synchronization spots, known to the device:



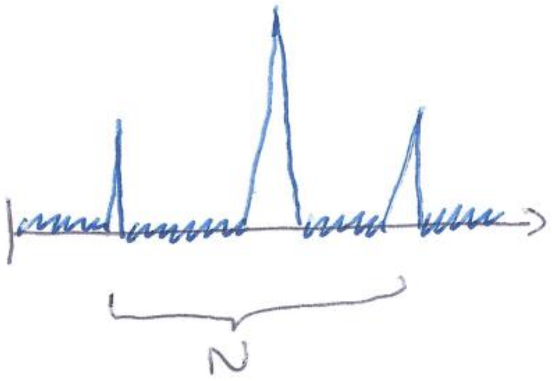
(1.4)

- Sadly this method won’t be useful for digital robust watermarking, as it would be child’s play to simply crop out the synchronization corners

- Before fixing our problem, lets talk about signals, and auto correlation:
- Let w be a random generated signal of length N
- If we correlate signal w with itself we get the following plot:

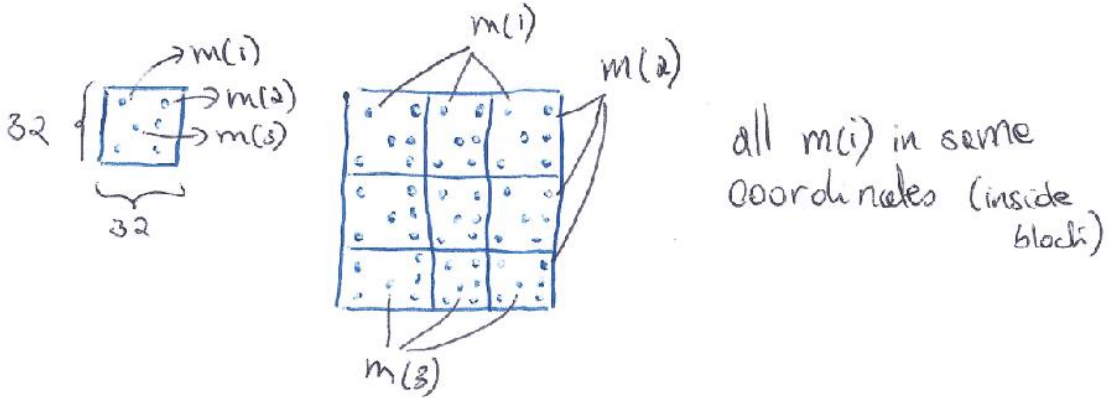


- Now lets make w'that is w twice (w' is periodical), w' is of length 2N, and period N
- If we correlate w' with itself we get the following plot:

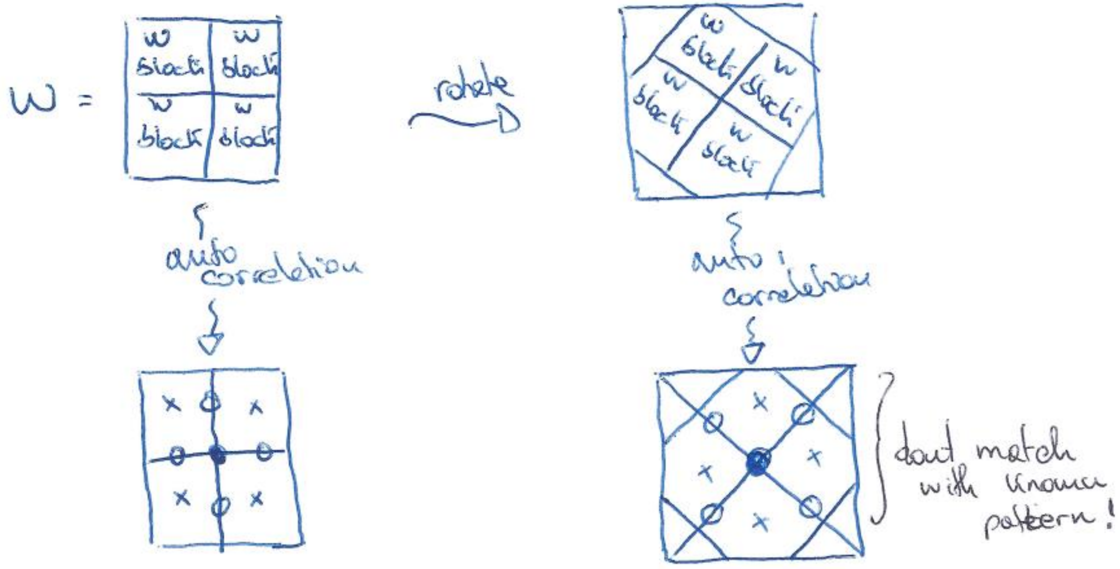


- From this plot we can actually “read” the length of w

- We will simply apply the same mechanism to our watermark, but with 2D signals!
- For robustness we already repeat the same value many times, now we will simply make it periodical
- We start by splitting x in R blocks, a good size is 32x32
- We create a new w of the same size as the blocks, generated from k



- Lets say we have 4 blocks of repetition, (in practice we will have more)
- Auto correlation will look like this:



(1.4)

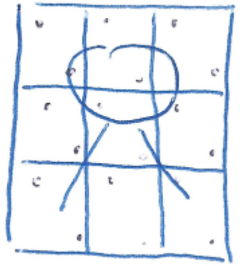
- For flipping and cropping, we need to do one more thing
  - We add “reference” bits to our watermark
  - These bits are generated from the key k
  - So they are known to the decoder
  - By cross correlating w and $w^\wedge$, we can see what is missing → $w^\wedge$ is the estimated watermark

- Detection algorithm → we receive $v = y + z = x + w + z$
  1. $x^\wedge = denosie(v)$
  2. $w^\wedge = y - x^\wedge$
  3. auto-correlate($w^\wedge$) → fixes rotation, scaling, shearing, aspect ratio
  4. Compute $w = f(k)$
  5. Cross-correlate($w^\wedge, w$) → fixes cropping, flipping

(1.4)

- With all this in place, we are unfortunately weak to an averaging attack!

$y = x + w$

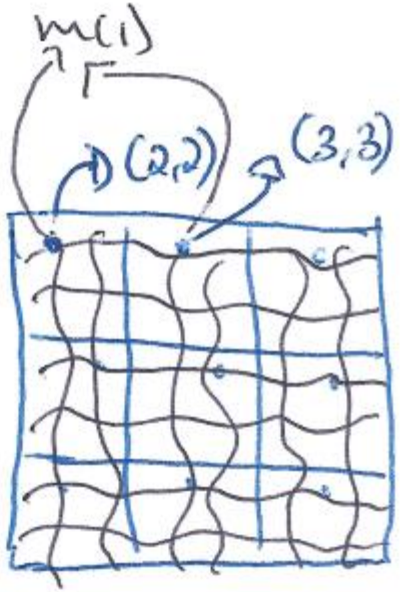


→ compute denoise$(y) = \hat{x}$ → compute $y - \hat{x} = \hat{w}$

→ split $\hat{w}$ into blocks of 32x32 → average 32x32 blocks

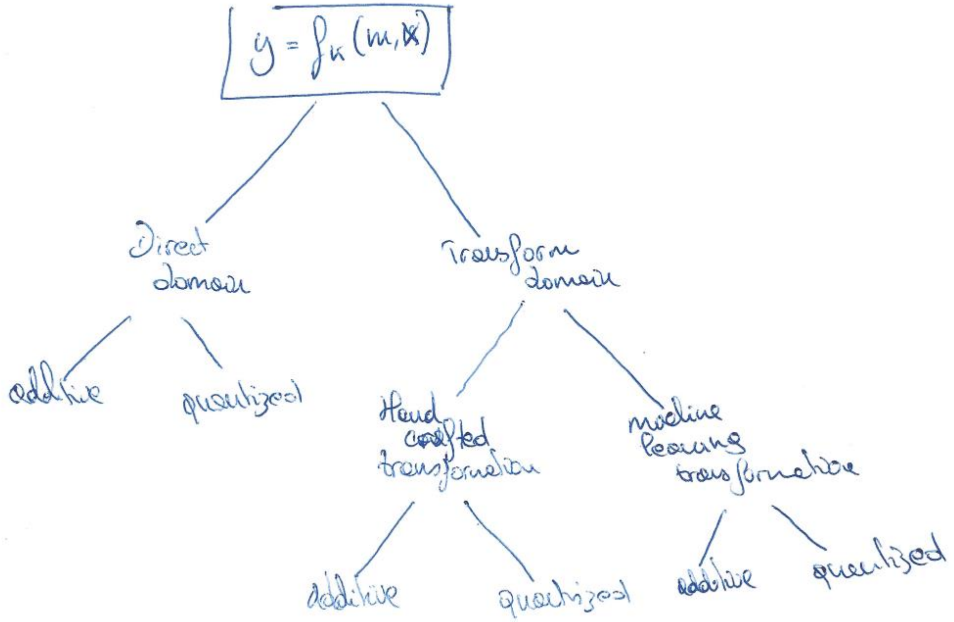
→ better approx of w!

- To fix this problem we simply add fluctuations to our periodical watermark
- Since it is just small fluctuations we keep all the desired properties
- The attacker will no longer be able to perform averaging attack on us



(1.5) Explain the difference between additive and quantization embedding/modulation

- Both techniques can be used in the same scenarios
    1. Direct domain
    2. Transform hand crafted domain
    3. Transform machine learning domain



- Direct domain can be seen as a transform domain with transform matrix Identity

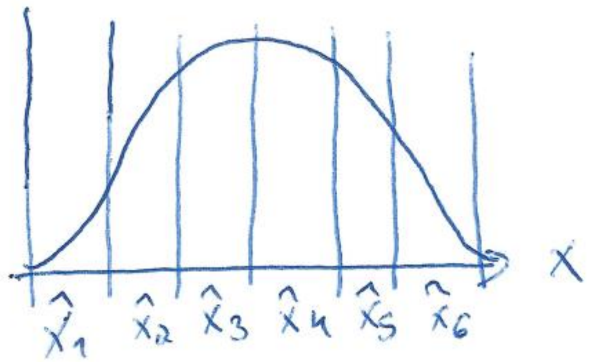
(1.5)

- Additive embedding in direct domain:
  1. Compute the watermark matrix $w = f(m, k)$
  2. Compute mask matrix ϕ using edge detection filter → Laplacian filter
  3. Set γ, usually between 1 and 7
  4. Compute $y = f_k(x, m) = x + \gamma\phi \times w$ → with × point to point multiplication

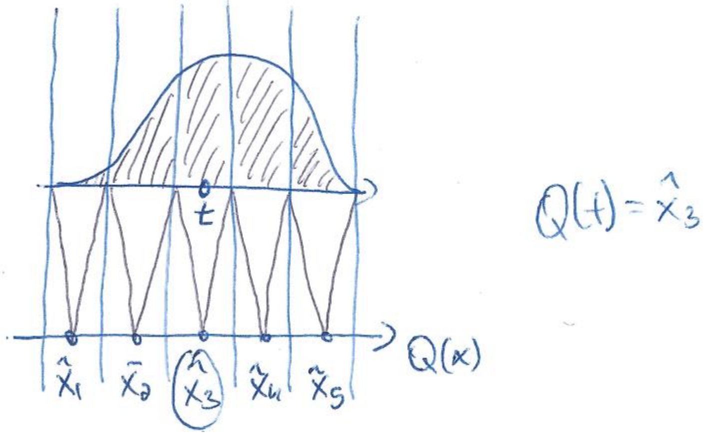
- Additive embedding in transform domain:
  1. Compute $\mathrm{x}_{\mathrm{tr}} = Transform(x)$
  2. Compute the watermark matrix $w = f(m, k)$
  3. Compute mask matrix ϕ using edge detection filter → Laplacian filter
  4. Set γ, usually between 1 and 7
  5. Compute $y = f_k(x, m) = Transform^{-1}(x_{tr} + \gamma^{\sim}\phi^{\sim} \times w^{\sim})$ → with × point to point multiplication

(1.5)

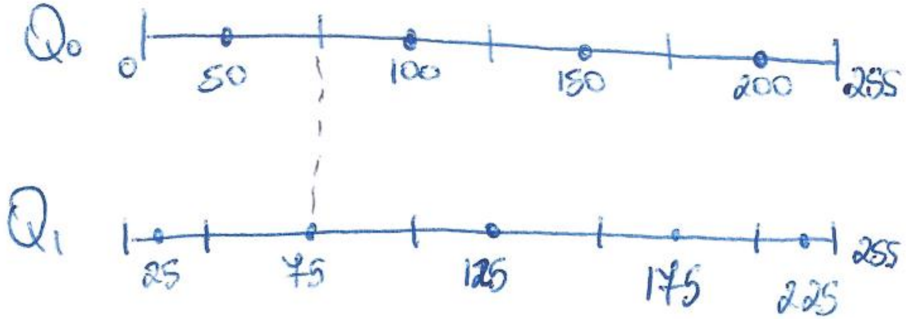
- Quantization modulation:
  - What are quantifiers? Quantifiers are defined by:
    1. Their intervals, can be uniform or not
    2. Their centroids → $x^{\wedge}$
    - ➢ Each interval has a corresponding centroid



  - Let Q be a quantifier:
    - If we want to quantify the value t via the quantifier Q
    - We note it $Q(t)$
    - Which is equal to the centroid $x_i^{\wedge} \in x^{\wedge}$ corresponding to the interval where value t is



  - With this method key k is used to generate both locations and quantifiers
    - We need to generate 2 quantifiers Q_0 and Q_1
    - If we want to embed $m(i) = 0$, we use Q_0
    - If we want to embed $m(i) = 1$, we use Q_1

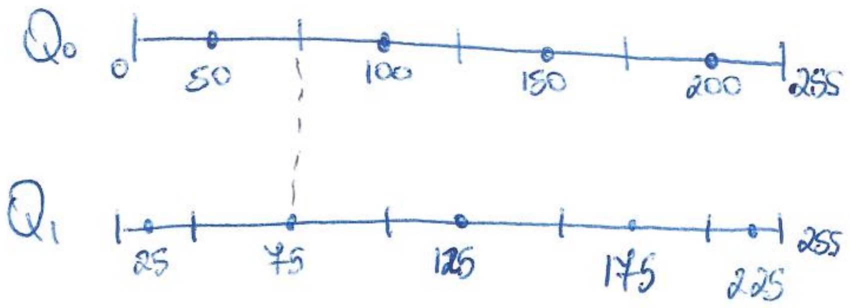


  - Quantifiers Q_0 and Q_1 are “complementary” so their centroids are as far away as possible

(1.5)

- Lets try we want to embed $y(i,j) = Q_i\big(x(i,j)\big) = Q_i(110)$

  - If $m = 0 \;\rightarrow\; y(i,j) = 100$
  - If $m = 1 \;\rightarrow\; y(i,j) = 125$



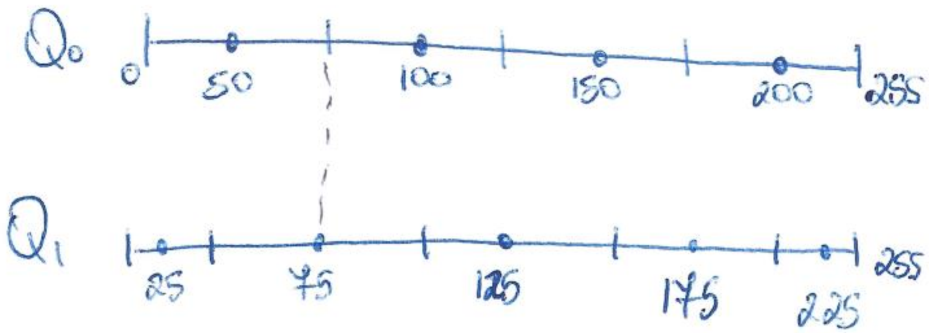
- Just do this for all secrete embedding positions for complete embedding

- Decoding:
  1. Compute $Q_0,\;\; Q_1$ and secret locations using k
  2. Go through each secrete position
     1. Decode $m\text{^} = 0$ or $m\text{^} = 1$ depending on the closest centroid from both quantifiers

  - If $v(i,j) = 120 \rightarrow m = 1$
  - If $v(i,j) = 60 \rightarrow m = 0$

(1.5)

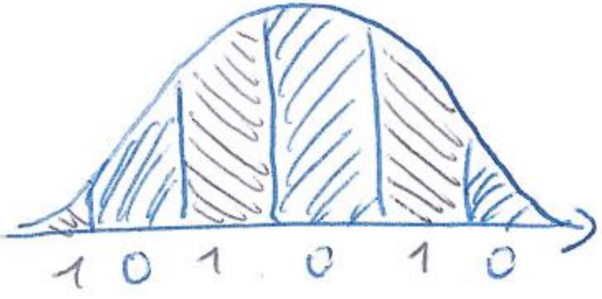
- Lets inspect possible attacks!



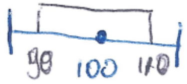
1. We encode → $Q_i\ (110) \rightarrow m = 0\ \rightarrow\ y(i,j) = 100$
2. Attacker adds noise → $v(i,j) = y(i,j) + 15 = 115$
3. We decode $m = 1$!!

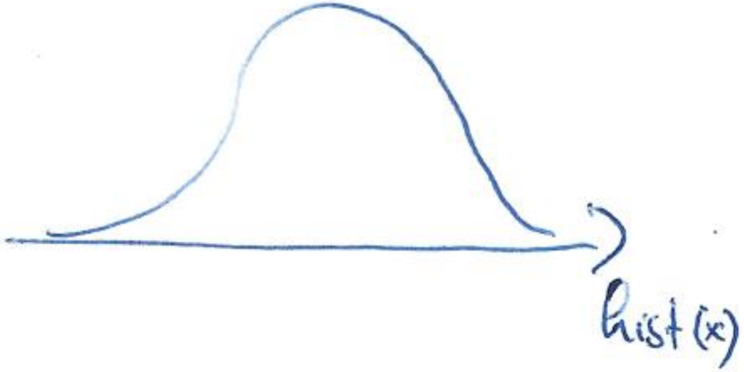
- If attacker adds too much noise, image looses quality (15 is too much!)

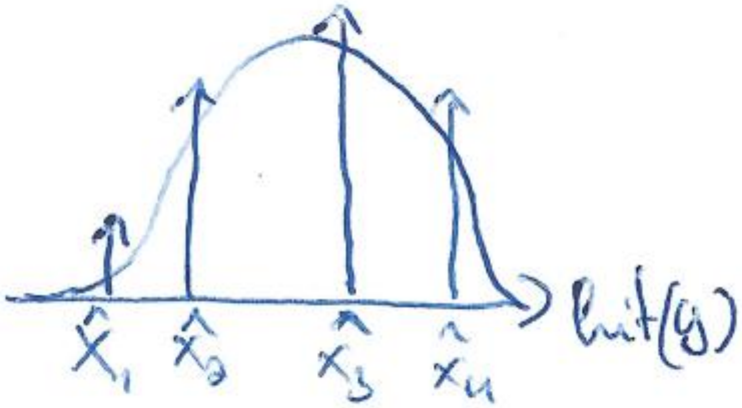
  (because he has to add this much random noise everywhere!!)



- If attacker doesn't add enough, it won't interfere with decoding







(1.5)

- Which method is best ?

  - Additive watermark:
    1. Good extraction in flat signals
    2. Not so good extraction in edges → because it relies on denoising!
  - Quantization watermark:
    1. Doesn't rely on denoising for watermark extraction!

(1.6) Explain digital watermarking in the transform domain. What are the main advantages?

- Why do we need transform domain?
  - ➢ It is impossible/very hard to establish statistical relationship between all values in x
  - ➢ Which means it is a very bad domain for compression!

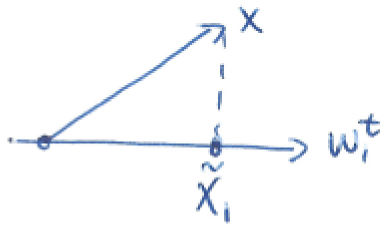
- To go to the transform domain, we simply project x onto a transform matrix T

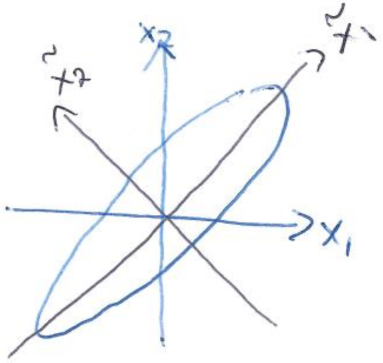
$$\begin{bmatrix} | \\ \tilde{x} \\ | \end{bmatrix} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_N \end{bmatrix} = \begin{array}{c} 1 \\ \\ \\ N \end{array} \overset{N}{\begin{bmatrix} - T_1^t - \\ - T_2^t - \\ \vdots \\ - T_N^t - \end{bmatrix}} \begin{bmatrix} | \\ x \\ | \end{bmatrix}_N$$

- By doing this projection we achieve some desired properties:
  1. Statistical independence between all $x_i^{\sim}$ (because of multiplication with random values)
  2. Energy compaction → $var(x_i^{\sim}) \geq var(x_{i+1}^{\sim})$

- This energy compaction allows us to keep ~10% of $x^{\sim}$, without loosing the essential information!
- Since T is of size (NxN), we can compute → $x \cong T^{-1}x^{\sim}$, with unique solution

(1.6)





$var(x_1) = var(x_2)$

$\rightarrow var(\tilde{x}_1) > var(\tilde{x}_2)$

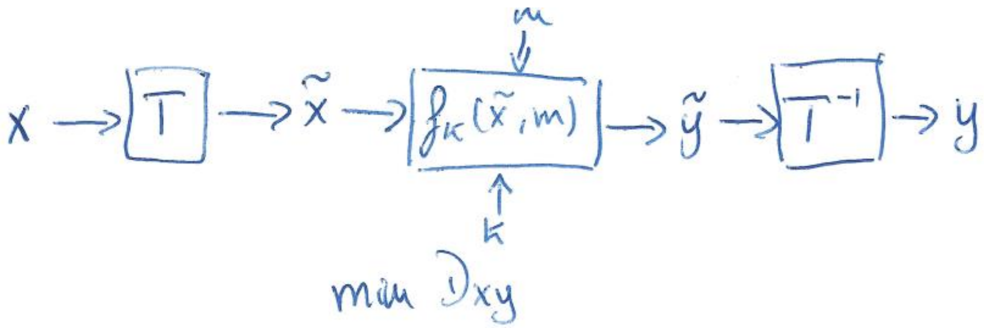
- Lets say we want to compress a given multimedia file x
- We do the following:
    1. Compute $x^{\sim} = Tx$
    2. Keep only a (%) of $x^{\sim}$
    3. Send/store $x^{\sim}$
    4. Compute $x^{\wedge} = T^{-1}x^{\sim}$ → Goal $|x^{\wedge} - x| < \sigma$

- It is a lossy compression!

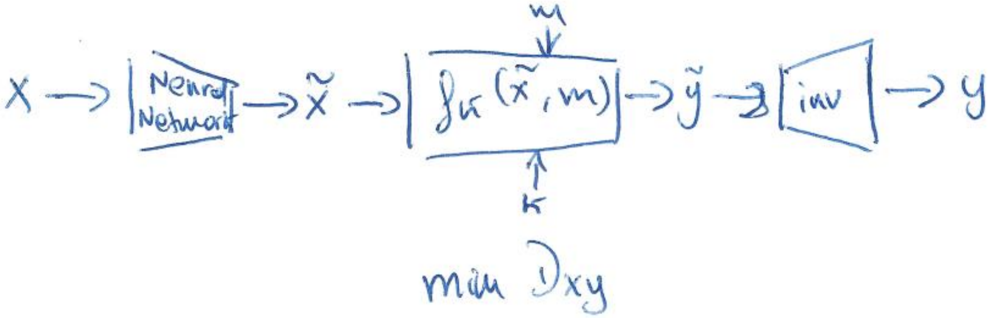
➢ In digital robust watermarking, we will apply the watermark in the transform domain!!

(1.6)

- How can we compute this transformation matrix T ?
- There are 2 methods



1. Hand crafted → data independent
2. Machine Learning → data dependent



- ML will offer better transformations, with adapted projections! (better feature extraction)
- ML costs time to train
- ML transform domain is more invariant to distortions
- HC transformations are Linear