

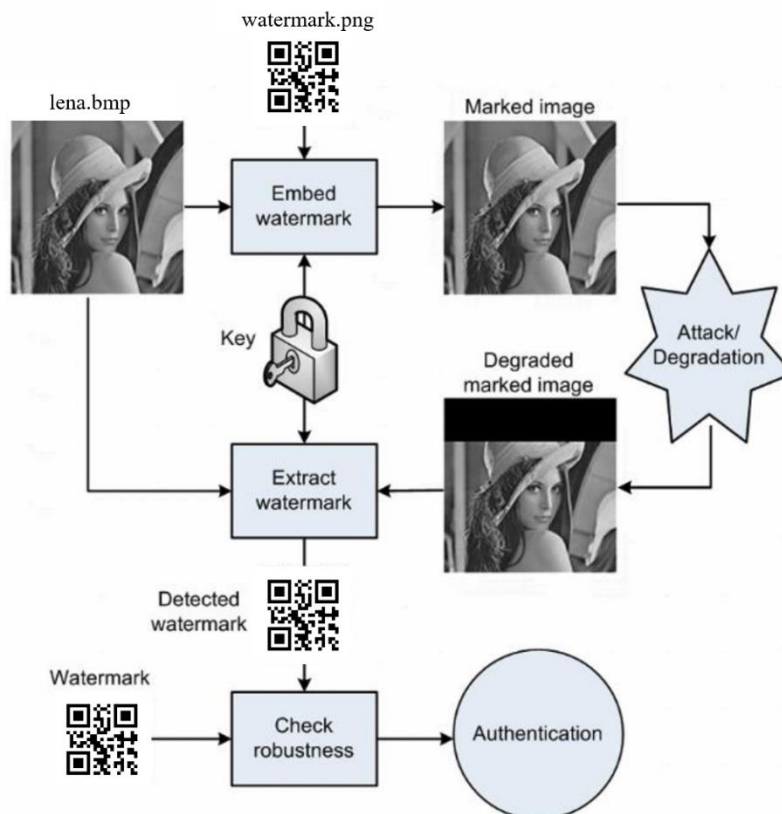
## Homework 4

310611095 許立暘

M093781 趙宇涵

### Problem Description

此次作業我們需要在一張照片（lena.bmp）中隱藏一個浮水印照片（watermark.png），其中需自行設計隱藏的方法，接受三種階段的攻擊後將浮水印取出。

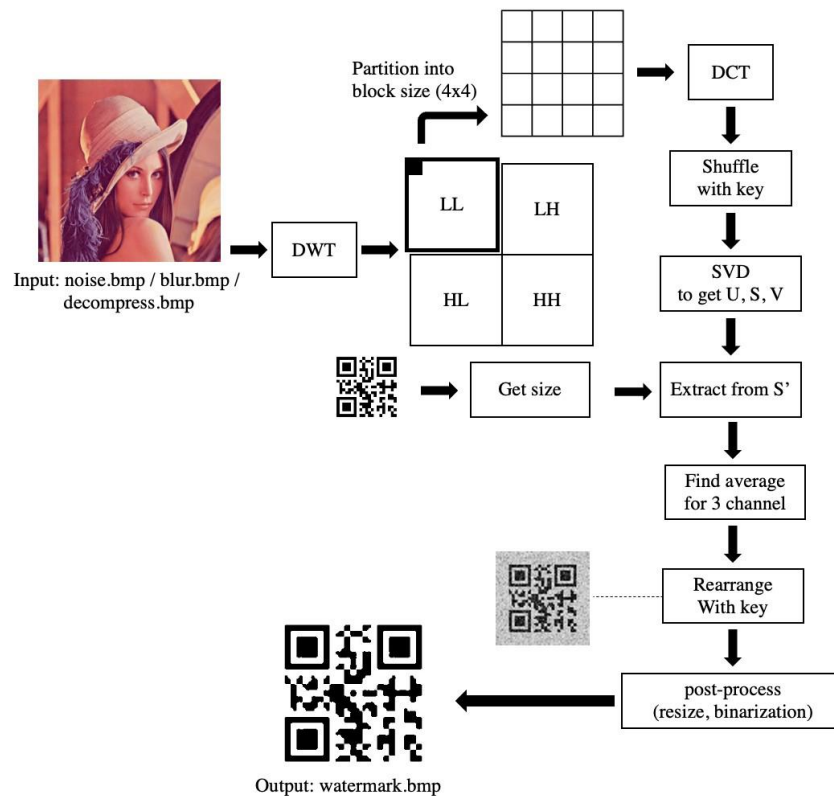
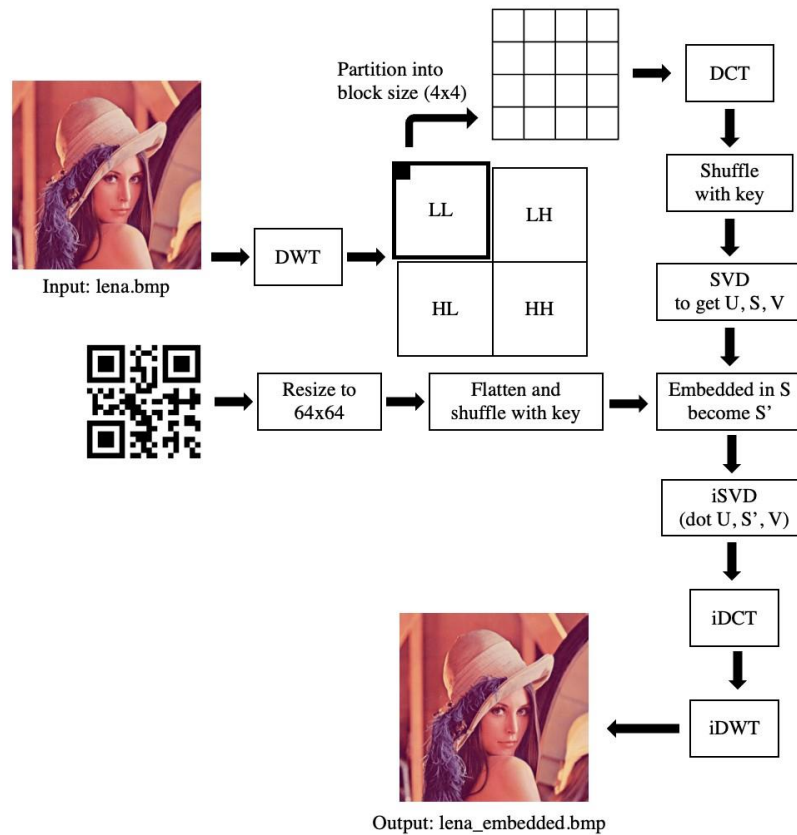


在隱藏浮水印的過程中，我們可以設計一個 key，用以紀錄隱藏的方法或是位置，更進一步加強隱藏浮水印的強度。且該 key 同時也可以用在提取浮水印時，透過這個 key 來找尋我們隱藏在其中的浮水印。

最後將提取出的浮水印與原先的浮水印做比較，分析這個方法的強韌性，觀察其是否有能力抵抗攻擊。

## Methodology

以下為這次作業的流程圖，上圖為把浮水印藏入原圖的流程，下圖為把浮水印擷取出來的過程：



這次的作業中，為了抵抗助教的三種攻擊，普通的 LSB(Least Significant Bit) 方法很容易失敗，因此這裡使用了老師上課提到過的 DWT-DCT-SVD 方法來實作，以下簡單說明三者的功能以及目的：

### 一、Discrete Wavelet Transform (DWT)

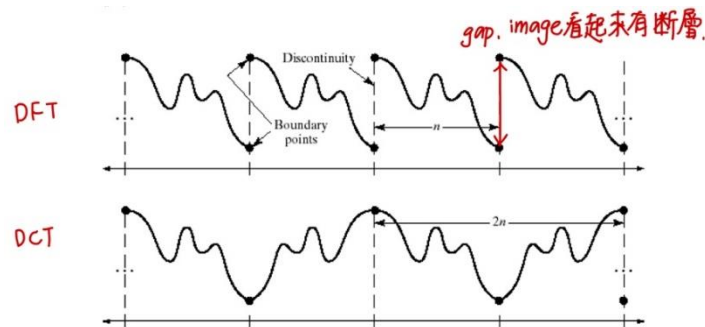
DWT 主要的目的是分離出影像中高頻與低頻成分，且分別對水平方向與垂直方向做分離。

因大部分的攻擊（例如：高斯模糊或壓縮）都是針對高頻區域做攻擊，因此我們的目標是將浮水印藏在低頻，也就是流程圖中 LL 的區域，這也是我們欲使用 DWT 的主因。

### 二、Discrete Cosine transform (DCT)

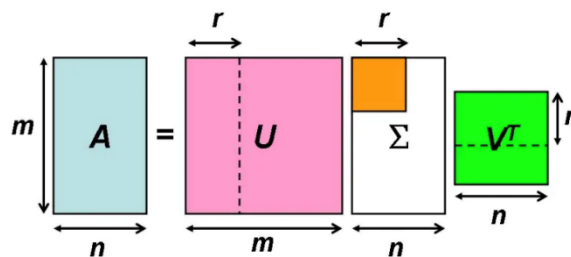
DCT 方法目的是把原本空間域的影像轉換為頻域，功能與 Discrete Fourier Transform 相似，不同的是 cosine 可以減少離散成分之間因 gap 而產生的影響。如下圖所示。

因此在這次我們使用的是 DCT 方法。



### 三、Singular Value Decomposition (SVD)

SVD 為線性代數中分離矩陣的一種方法，主要可以將矩陣拆分為  $U$ ,  $\Sigma$ ,  $V$  三個維度不同的矩陣，如下圖所示：



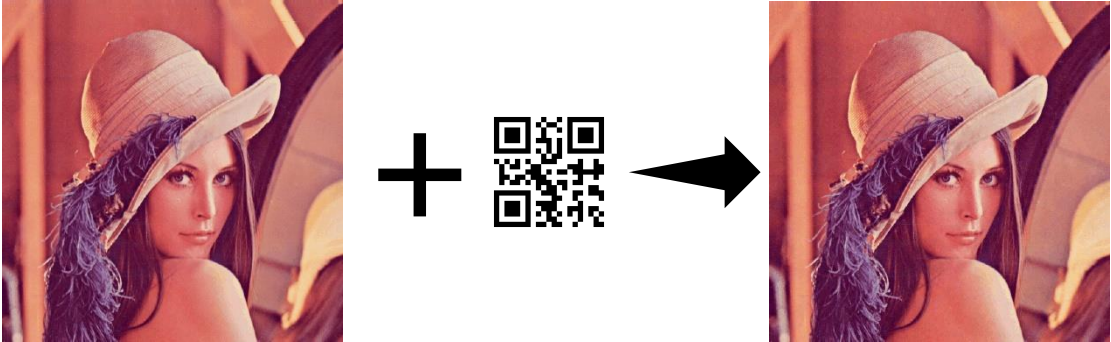
進行 SVD 分離後，我們便可以把浮水印的資訊藏在對角矩陣  $\Sigma$  當中，而此處我們設定兩個參數  $d1$ ,  $d2$ ，分別表示藏在  $\Sigma$  中第一維與第二維的資訊，此處若  $d1/d2$  的數值越大，加入浮水印後的影像失真度越大，但是也越能夠抵抗攻擊（robustness 越強），我們在這次作業中設定了兩組參數：

|     | d1 | d2 | d1/d2 |
|-----|----|----|-------|
| 參數一 | 36 | 20 | 1.8   |
| 參數二 | 80 | 20 | 4     |

後者的  $d1/d2$  分數較高，預計這組參數較能抵禦外部攻擊。

Conclusion & Discussion

透過以上的處理，我們可以將 lena.bmp 加上一張浮水印 watermark.png。其中，lena.bmp 的大小為 1024\*1024，watermark.png 的大小為 290\*290，如下所示：



而由於乘載量的緣故，因此無法於 lena.bmp 放下完整尺寸的 watermark.png，因此此處採用 resize 的手段將 watermark.png 壓縮到 64\*64，再將其藏入 lena.bmp。

如 methodology 中所述，我們調整兩組參數，首先選用參數一，此狀況下隱藏浮水印的效果較好，隱藏浮水印後的圖片更接近原圖；而之後我們也選用參數二，此種狀況下雖然隱藏浮水印的效果較差，但是具有更高的強韌性，對於還原浮水印時效果更好。

| 參數一 |         | 參數二    |  |
|-----|---------|--------|--|
|     |         |        |  |
|     |         |        |  |
|     | PSNR    | SSIM   |  |
| 參數一 | 37.9067 | 0.9962 |  |
| 參數二 | 31.6945 | 0.9854 |  |

從放大的細節中可以看出，參數二參數的背景顆粒感較重，而參數一參數則



較接近原圖。使用科學方法進行比較，無論是 PSNR 與 SSIM 皆是參數一分數較高，因此推測參數一參數隱藏浮水印能力較強。而不論是上述哪種參數，僅僅使用肉眼無法將隱藏的浮水印找出，因此驗證此方法可以將浮水印隱藏於圖片中。

首先，我們先針對參數一進行還原：



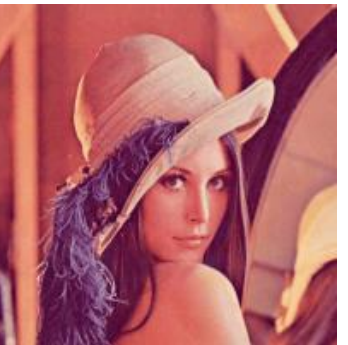

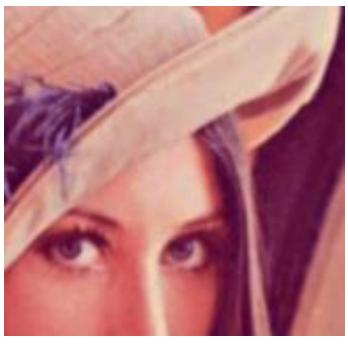
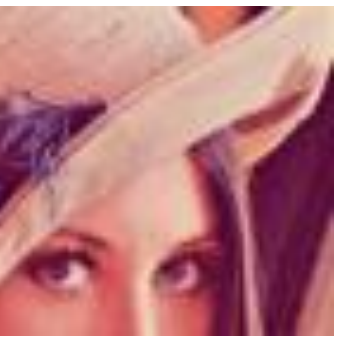
我們對沒有被攻擊的圖片進行還原，試圖提取其中的浮水印，其結果如下：



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

由於我們先將浮水印壓縮至 64\*64，因此還原的結果也是 64\*64，必須使用 resize 函式將其轉化為 290\*290。而這個壓縮再轉換的過程中便出現失真，在黑白的交界帶出現了灰色地帶，進一步使用二值化（閾值為 0.7）將其轉化為黑白二色之圖片。轉化前 BER=0.7648，轉化後為 0.9478。由此可以得出：此方法確實可以將浮水印從照片中還原，而且主要損失由圖片縮小再放大時造成。

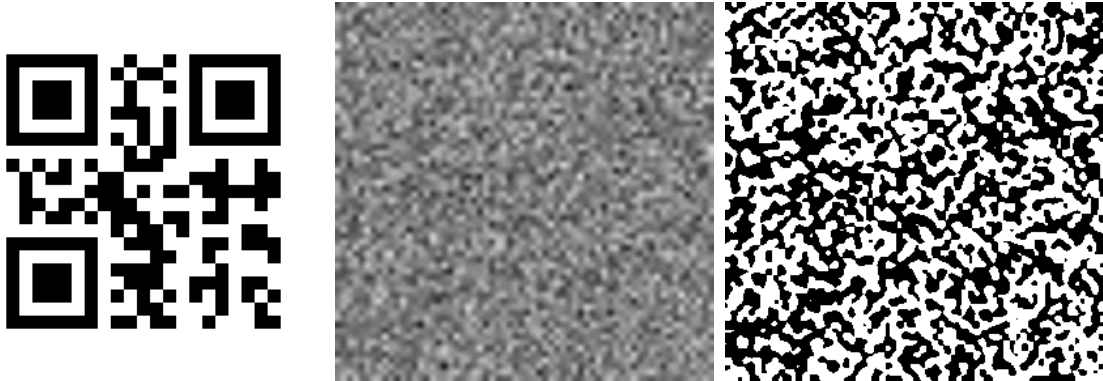
接著採用三種等級的攻擊，分別為 noise、blur、compress，結果如下所示：

| Noise   | Blur  | Compress  |
|---|---|---|
|  |  |  |
|  |  |  |

## 1. Noise

助教指定的 noise 形式為 gaussian noise，參數為  $\text{mean} = 0$ ， $\text{sigma} = 0.1$ 。

此時我們可以得到的還原的浮水印，而此時大小仍為  $64 \times 64$ ，因此使用 `resize` 將尺寸還原成  $290 \times 290$ 。



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

由於中圖還原結果尚有保留許多灰色區域，因此使用二值化（閾值 = 0.5）將其轉換為黑白二色。

由此結果可以看出，在 noise 的攻擊下，浮水印完全被破壞，肉眼看不出任何相似之處。進一步計算  $\text{BER} = 0.500$ ，更加印證了此浮水印已完全被破壞。而右圖的二值化結果，BER 也僅有 0.5253，此結果幾乎與隨機創造的黑白圖片與浮水印相比之 BER 值相同，因此可以得到 noise 已將浮水印完全破壞的結論。

而為了驗證是否為 noise 效果太過強烈導致於浮水印完全被破壞，我們將 gaussian noise 參數稍作調整， $\text{mean} = 0$ ， $\text{sigma} = 0.05$ 。在此參數下，我們可以很好的還原浮水印：



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

為了將中圖的灰色區域去除，使用二值化（閾值 = 0.5），將其轉換為黑白二色的圖片。

此結果可以看出，二值化之前的圖片便已可以隱約看出原始浮水印的痕跡，計算  $\text{BER} = 0.5076$ 。而二值化結果， $\text{BER} = 0.8910$ ，甚至使用手機也可以掃描出 QC code 的內容。因此可以得證此方法可以經得起的 noise 攻擊，但若是 noise 的攻擊太過強烈，浮水印則會被完全破壞。

## 2. Blur

助教指定的 blur 形式為 gaussian blur，參數為大小 =  $7 \times 7$ ，sigma = 2。

經過上述的方法，我們可以將浮水印還原，此時還原出來的浮水印大小為  $64 \times 46$ ，使用 resize 函式將其還原成  $290 \times 290$ 。而由於還原結果具有灰色部分，因此採用閾值為 0.45 的二值化，將圖片轉化為黑白二色之圖片，如下圖所示：



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

由上圖可以看出，尚未二值化之前的圖片便已可以看出浮水印，此時 BER = 0.5092。由於灰色部分很多，因此 BER 值並不好看。而進行二值化之後，BER = 0.9468，且使用手機可以讀取 QR code 內容，還原浮水印的能力非常好。

## 3. Compress

助教指定的 compress 參數為改變 size 至  $200 \times 200$ 。此處我們先將 compress 之後的圖片還原成  $1024 \times 1024$ ，再進行浮水印的圖取。

使用上述的方法，我們可以將浮水印還原出來，結果如下：



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

起初還原的圖片大小為  $64 \times 64$ ，此處使用 resize 將浮水印轉化為  $290 \times 290$ 。而此時可以看出浮水印大致輪廓，此時 BER = 0.4981。而進行二值化的結果，BER 上升至 0.9251。使用手機掃描，也可以得到 QR code 內部的訊息。以此觀察，此方法可以很好的提取將被 compress 的圖片中的浮水印。

接著，我們使用參數二，觀察是否具有更好的抗攻擊能力。



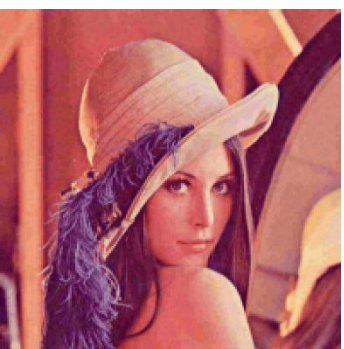
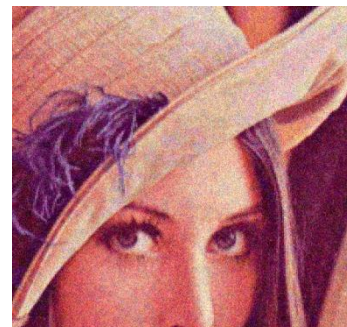
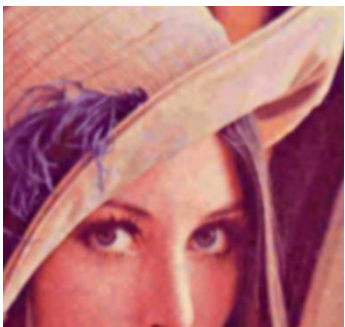
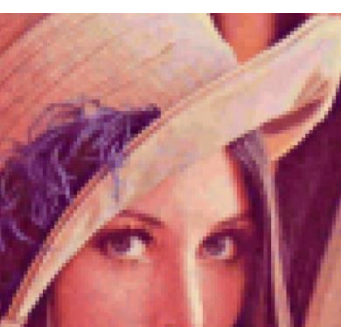
我們對沒有被攻擊的圖片進行還原，試圖提取其中的浮水印，其結果如下：



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

由於乘載量的緣故，因此隱藏的浮水印大小事先被壓縮至 64\*64，而提取之後再進行 resize 成 290\*290。而由比較上圖可以看出，QR code 有被模糊化的趨勢，其原因應該是由於圖片壓縮與放大造成的，使用二值化（閾值 = 0.5）將圖片轉為黑白二色的結果，使得其邊緣變得銳利，更接近原先的圖樣。原先 BER=0.7703，二值化後 BER=0.9542。使用手機也可以輕易讀取內容，還原相當成功。

接著使用助教的三種攻擊模式進行攻擊，分別得到以下幾種結果：

| Noise   | Blur  | Compress   |
|---|---|--|
|  |  |  |
|  |  |  |



## 1. Noise

使用助教指定的 gaussian noise，參數為  $\text{mean} = 0$ 、 $\text{sigma} = 0.1$ ，可以得到以下的還原結果：



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

由圖中可以看出，即使是使用參數一無法還原的 noise 參數，參數二在初步還原時便已可以隱約看出 QR code，而二值化（閾值 = 0.5）過後可以得到一個相對清晰的圖案，使用手機可以讀取其訊息。在經過 noise 的攻擊下，使用參數二可以很好的還原浮水印，達到參數一無法達到的效果。還原結果  $\text{BER} = 0.5177$ ，二值化後  $\text{BER} = 0.9167$ 。

## 2. Blur

使用助教指定的 gaussian blur，參數為大小 =  $7 \times 7$ ， $\text{sigma} = 2$ 。



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

使用參數二，可以從上圖的還原結果看到 QR code 的大致輪廓，而且二值化（閾值 = 0.5）過後的浮水印清晰可見，使用手機也可以完整的讀取其內容，浮水印提取非常成功。還原結果  $\text{BER} = 0.5144$ ，二值化過後  $\text{BER} = 0.9509$ 。

### 3. Compress

由於經過 compress 之後，圖片大小為 200\*200，因此要將圖片 decompress 為 1024\*1024，再進行浮水印的還原，其還原結果如下：



左圖為原始 watermark.png，中圖為還原結果，右圖為二值化。

從中間圖中可以看出 QR code 的大致外觀，除了背景的灰色部分略多以外其圖案算是非常明顯，因此使用二值化（閾值 = 0.5）進行處理。處理完之後的浮水印外觀清晰，使用手機也可以讀取其內容，影像還原度甚高。還原結果 BER = 0.5192，二值化過後 BER = 0.9461。

數據整理：

參數一：

|               | PSNR    | SSIM   |
|---------------|---------|--------|
| Lena embedded | 37.9067 | 0.9962 |

| BER      | 還原結果   | 二值化結果  |
|----------|--------|--------|
| 未經攻擊     | 0.7648 | 0.9478 |
| 強烈 Noise | 0.500  | 0.5253 |
| 輕微 Noise | 0.5076 | 0.8910 |
| Blur     | 0.5092 | 0.9468 |
| Compress | 0.4981 | 0.9251 |

參數二：

|               | PSNR    | SSIM   |
|---------------|---------|--------|
| Lena embedded | 31.6945 | 0.9854 |

| BER      | 還原結果   | 二值化結果  |
|----------|--------|--------|
| 未經攻擊     | 0.7703 | 0.9542 |
| Noise    | 0.5177 | 0.9167 |
| Blur     | 0.5144 | 0.9509 |
| Compress | 0.5192 | 0.9461 |

無論是從上表，或是從圖片中都可以看出，使用參數二的浮水印還原效果遠大於參數一的還原效果，因此可以認定參數二強韌性更強。但是若是觀察隱藏浮水印的 lena 圖片，可以看出參數二的圖片較多顆粒感，比起參數一的近乎沒改變原圖效果，參數二在此略輸一籌。但是考慮到隱藏浮水印的最終目的是要能夠經得住攻擊並且完整讀取，因此參數二才是更適合隱藏浮水印的參數。

## Reference

- [1] Manie Kansal Manie Kansal, Gursharanjeet Singh, B V Kranthi, “DWT, DCT and SVD based Digital Image Watermarking”, 2012 International Conference on Computing Sciences, 2012.
- [2] Linglong Tan, Yihong He, Fengzhi Wu, Dong Zhang, “A Blind Watermarking Algorithm for Digital Image Based on DWT”, Journal of Physics: Conference Series, 2020.
- [3] Ali Benoraira, Khier Benmahammed, Noureddine Boucenna, “Blind image watermarking technique based on differential embedding in DWT and DCT domains”, EURASIP Journal on Advances in Signal, 2015.
- [4] Nawaf Hazim Barnouti, Zaid Saeb Sabri, Khaldoun L. Hameed, “Digital Watermarking Based on DWT (Discrete Wavelet Transform) and DCT (Discrete Cosine Transform)”, International Journal of Engineering & Technology, 2018.
- [5] Mahbuba Begum, Mohammad Shorif Uddin, “Digital Image Watermarking Techniques: A Review”, 2020.