Errorless Robust JPEG Steganography using Outputs of JPEG Coders

Jan Butora, Pauline Puteaux, Patrick Bas

▶ **To cite this version:**

Jan Butora, Pauline Puteaux, Patrick Bas. Errorless Robust JPEG Steganography using Outputs of JPEG Coders. IEEE Transactions on Dependable and Secure Computing, 2023, pp.1-13. 10.1109/TDSC.2023.3306379 . hal-04181480

HAL Id: hal-04181480
https://hal.science/hal-04181480v1

Submitted on 16 Aug 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Errorless Robust JPEG Steganography using Outputs of JPEG Coders

Jan Butora, Pauline Puteaux and Patrick Bas

***Abstract*—Robust steganography is a technique of hiding secret messages in images so that the message can be recovered after additional image processing. One of the most popular processing operations is JPEG recompression. Unfortunately, most of today's steganographic methods addressing this issue only provide a probabilistic guarantee of recovering the secret and are consequently not errorless. That is unacceptable since even a single unexpected change can make the whole message unreadable if it is encrypted. We propose to create a robust set of DCT coefficients by inspecting their behavior during recompression, which requires access to the targeted JPEG compressor. This is done by dividing the DCT coefficients into 64 non-overlapping lattices because one embedding change can potentially affect many other coefficients from the same DCT block during recompression. The robustness is then combined with standard steganographic costs creating a lattice embedding scheme robust against JPEG recompression. Through experiments, we show that the size of the robust set and the scheme's security depends on the ordering of lattices during embedding. We verify the validity of the proposed method with three typical JPEG compressors and the *Slack* instant messaging application. We benchmark its security for various embedding payloads, three different ways of ordering the lattices, and a range of Quality Factors. Finally, this method is errorless by construction, meaning the embedded message will always be readable.**

***Index Terms*—robust steganography, recompression, lattice embedding, JPEG**

I. Introduction

With the wide usage of social networks and sharing platforms, the classical setup of steganography, which implies a lossless channel between Alice (the steganographer who embeds a payload) and Bob (the steganographer who decodes a payload), is meaningless in a lot of practical scenarios. This is due to the fact that the transmission channel involves a transcoding of the stego content (a JPEG recompression, for example) which can be seen as a noisy channel between Alice and Bob. In such a case, the errorless decoding of the payload is not possible anymore if Alice uses classical embedding schemes designed for lossless transmission (*e.g.* in the JPEG domain with the use of J-Uniward [1], UERD [2], J-Mipod [3], ...). Moreover, note that if the embedded payload is encrypted – which is usually the case for security reasons – decoding the embedded message is not possible as soon as one bit of the payload is changed.

A. Prior Works on Robust Steganography

The domain of robust steganography aims at keeping the main constraint of steganography (*i.e.* to embed an undetectable payload) but also adds the constraint of robustness, which can be defined as minimizing the bit error rate on the decoded payload after a lossy transmission channel. Note that this second constraint (robustness) is very similar to the one defined in watermarking. For this reason, secure watermarking [11] could hypothetically be considered as an option to embed robust and undetectable payload [12]. However, the proposed schemes in the watermarking literature never considered either steganalysis to benchmark undetectability or large embedding rates. In watermarking, the payload characterizes an identifier of several dozens of bits, not a message of several kilobits.

In [4], Cleaves and Ker both study the impact of lossy transmission combined with syndrome trellis code (STC) [13]. They show that the STC replicates the errors associated with the channel on the decoded payload. They propose to use a dual-STC scheme combined with Reed-Solomon (RS) Error Correcting Codes (ECC) to reduce the error rate while minimizing the embedding distortion. This scheme is, however, not benchmarked w.r.t. steganalysis.

Zhang *et al.* proposed to mix a watermarking scheme [5] based on the modifications of DCT coefficients [14] and a steganographic scheme [1] to favor embedding on coefficients which are both robust and secure by weighting the cost related to J-UNIWARD. Unfortunately, the proposed scheme is very detectable (*e.g.* $P_E = 5\%$ at quality factor QF = 75 and 0.1 bpnzAC). The payload also needs to be protected using RS-ECC, and the error rate is still essential for large QF (19% at QF = 95).

In [15], Qiao *et al.* propose to select robust cover elements, which are defined as robust because they are not equal to zero after double compression. Unfortunately, this scheme suffers from at least two drawbacks: 1) Alice has to transmit the set of robust elements to Bob as side information, and 2) the payload is still subject to errors, and the detectability is, by a large amount, more important than the non-robust scheme.

Tao *et al.* propose the idea to generate an *intermediate image* after embedding by the mean of "coefficient adjustment" [6]. This image is a modified version of the stego image, and the modifications are computed to cancel the modifications due to an ideal JPEG coding scheme. A similar idea to invert the JPEG compression scheme was developed by Lu *et al.* by combining coefficient adjustment with an auto-encoder predicting the input image [7]. In both cases, the changes made to the intermediate image increase the detectability of the modified stego image. The practical implementation of finding a perfect intermediate image is also questioned in [7] due to convergence issues. The authors adopt another strategy using the auto-encoder, but it cannot cancel all the modifications due to coding.

Recently, Zhao *et al.* [8], proposed successive compressions of the cover image to reduce the number of changes after

Reference	Strategy	Errorless	Side information	ECC	Filtering before recompression
[4]	Dual STC	No	No	Yes	No
[5]	Watermarking	No	Yes	Yes	No
[6] and [7]	Coefficient adjustment	No	No	No	No
[8]	Successive recompressions	No	No	Yes	No
[9]	Non-robust set selection	No	No	Yes	Yes
[10]	Sign modification	No	Yes	No	No
Ours	Robust set selection	Yes	No	No	Yes

TABLE I
COMPARISON BETWEEN DIFFERENT SCHEMES IN THE FRAMEWORK OF STEGANOGRAPHY ROBUST TO JPEG COMPRESSION.

embedding and compression. This method is effective but has the disadvantage of generating (recompressed) cover images that are different from natural ones, hence more detectable. The proposed scheme also uses BCH-ECC to decrease the error rate after embedding.

One of the most recent works, Sign Steganography Revisited (SSR) [10], uses the sign of the DCT coefficients to communicate the secret message. The method is checking for every DCT mode if there is a coefficient that will change its sign during recompression. If that is the case, it will prohibit embedding into this DCT mode. To this end, the steganographer needs to additionally communicate the robustness of all 64 DCT modes as a side-information.

The last line of research on this topic is the scheme called MINImizing Channel Error Rate (MINICER) [9]. The algorithm first recompresses the cover images, then checks if an embedding change would create a so-called 'overflow', which means that there is a pixel with value outside of the interval [0,255] after decompression. If so, the algorithm considers such embedding change as non-robust by setting its embedding cost to infinity. The recompressed cover image is then embedded with modified costs and specific DCT coefficients are changed back to the original single-compressed cover value in order to create a single-compressed stego image. This algorithm can also deal with filtering before recompression.

While both SSR and MINICER, could potentially provide small bit error rates in some limited scenarios, we will show in Section V-D that in a practical setting, they are, in fact, not robust.

It is important to note that, due to the practical context of steganography, the scheme's robustness has to be extremely high, as only one erroneous bit can jeopardize the whole transmission. Errorless steganographic schemes are consequently recommended. Regarding robustness to image scaling, several works address this issue for different downsampling kernels, such as the nearest neighbor kernel or the bilinear kernel, where only pixels that are respectively preserved (see Zhang *et al.* [16]), or contribute the most (see Zhu *et al.* [17]), are modified. To the best of our knowledge, errorless robust JPEG steganography has not been investigated before.

B. Outline of the Paper

This paper proposes an errorless steganographic scheme in the JPEG domain robust to JPEG recompression. As summed up in table I, the advantages of this scheme are to be errorless (it can be guaranteed that generated stego will not produce any error at the embedding), so not convey any side-information for payload extraction, to not rely on the use of Error Correcting Codes (ECC). Moreover, the embedding enables to guarantee robustness even when filtering is applied before recompression.

Section II presents both the security setup (*i.e.* the knowledge of the embedder) and the coding setup (*i.e.* the JPEG coding process). Section III details an algorithm proposed to use the output of JPEG coders to extract a set of robust coefficients according to a specific scanning strategy. Section IV presents the embedding and decoding algorithms together with a strategy to spread the payload into 64 lattices. Section V presents results on the detectability of the proposed scheme and compares it with naive embedding. A robustness analysis is also proposed when JPEG coding uses rate-optimization strategies.

II. CONSIDERED SETUPS

A. Notations

A bold capital letter is considered as a matrix, and a lowercase letter denotes a coefficient. (i, j) denotes the pixels or coefficients coordinates i and j.

B. Security Setup

Robust steganographic schemes can be split into two categories, schemes which assume that the compression parameters are either known or unknown. The scheme that we present here belongs to the first category. Its security setup is illustrated on the top diagram of Fig. 1.

Alice, the embedder, sends a stego image, denoted $\mathbf{S}_1$, on a platform that compresses $\mathbf{S}_1$ into $\mathbf{S}_2$ using a JPEG coder. Bob, the receiver, downloads the image from the platform and tries to decode the payload. Because we assume that Alice's cover, denoted $\mathbf{C}_1$, is also in the JPEG format, the image is consequently double-compressed. Note that the setup is equivalent to classical (lossless) steganography if the platform does not recompress the uploaded image. In the following, we consider that the platform does compress the uploaded image.

We assume that Alice knows both the coding scheme and the coding parameters used by the platform. Practically, this can be done by inspecting the uploaded-downloaded images. The JPEG quantization matrix is public, and the coding scheme can often be identified by comparing the uploaded/downloaded image with the outputs of different coders.

Note that this setup follows the Kerckhoffs' principle, which states that anything not related to the secrecy of the application (here, the fact that a payload is potentially transmitted to Bob)

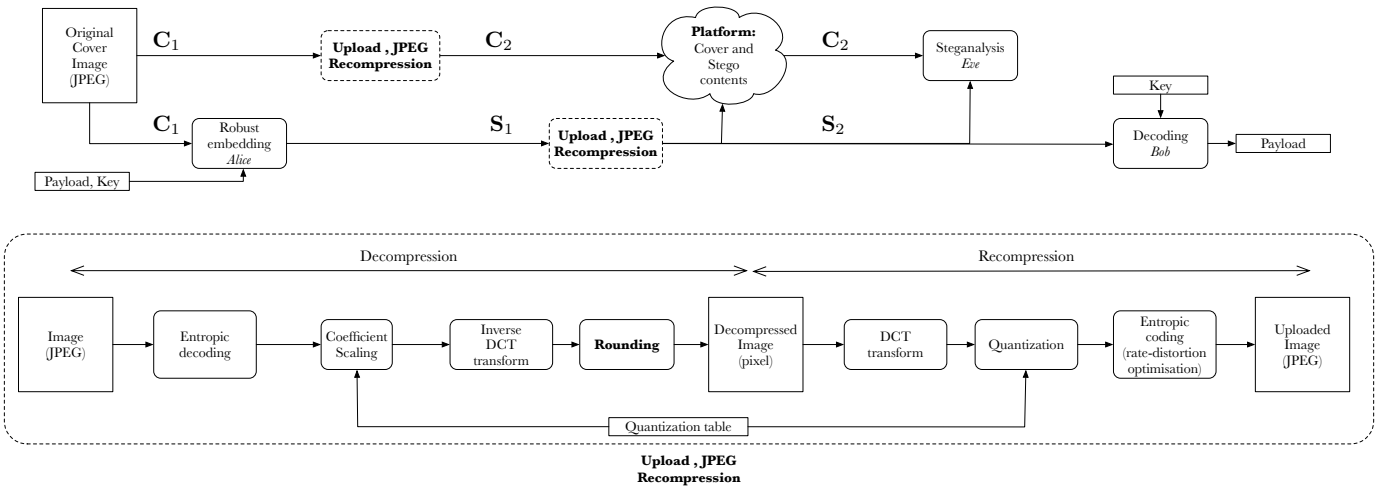

Fig. 1. Considered setups for robust steganography after JPEG compression: the top diagram represents the whole chain of processes and the different players (Alice, Bob, and Eve), see also Section II-B. The bottom diagram depicts the different operations involved in the uploading process, see also Section II-C.

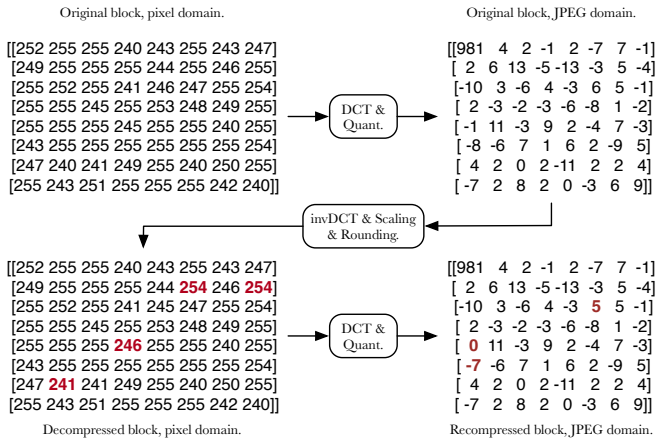

Fig. 2. Example of DCT coefficient changes after decompression and recompression (QF = 100). Changes in the pixel or JPEG domain are displayed in bold and dark red.

should be considered public. Within this setup, the steganalyst Eve has access to the platform. She can also download images to try to differentiate an uploaded cover content (denoted $\mathbf{C}_2$) from an uploaded stego content ($\mathbf{S}_2$).

In order to minimize the difference between the uploaded image (respectively $\mathbf{C}_1$ or $\mathbf{S}_1$) and the downloaded image (respectively $\mathbf{C}_2$ or $\mathbf{S}_2$), we assume that the uploaded image is already coded with the same coding parameters than the downloaded image. We assume first that JPEG compression is the only process done during the upload on the platform, which means that the uploaded image is already resized correctly in order to prevent any resizing operation. This assumption is rather practical since it is usually the case on many platforms such as Facebook, WhatsApp, FlickR [18], [19] or *Slack*. Note that in section III-E we also consider the case where a size-invariant filter is applied to the image before compression.

C. JPEG Coding Setup

Firstly, we recall the main features of the JPEG coding scheme. Without loss of generality, we assume that the original image is coded as greyscale. Still, the same methodology regarding the embedding mechanisms described in Sections III and IV can be applied on color channels, with or without sub-sampling.

- The original image is first decomposed into disjoint blocks of size 8×8 pixels.

- Each block is then transformed into 64 DCT coefficients using the specific DCT type-II transform.

- Each coefficient is then quantized according to a quantization matrix specific to the coding algorithm and each DCT mode.

Note that there is a specific relation between the JPEG quality factor QF and the quantization matrix, but depending on the implementation of the coder, this relation can be different. For example, the Libjpeg library[1], associated with the `convert` command, uses the classical relationship proposed by the standard (see [20], Section IV). On the contrary, the `mozjpeg` library[2] uses *ad hoc* quantization tables.

- For each block, the coefficients are scanned using a zigzag order.

- Depending on the coding scheme, the lossless entropic coding scheme can be different: the quantized coefficients are either directly coded using Run Length Coding and Huffman Coding, or a rate-distortion optimization procedure is applied to change the magnitude of several coefficients to increase the coding rate.

Note that in the first case, the coefficients and blocks are independently coded, but in the second case, there is an interplay between the possible coefficient values and the length of the produced code. The potential use of the rate-distortion procedure depends on the implementation. For example, the Libjpeg library does not implement by default any rate-

[1]http://libjpeg.sourceforge.net
[2]https://github.com/mozilla/mozjpeg

distortion procedure, but the `mozjpeg` library implements by default a Viterbi algorithm relying on a trellis.

The upload of a JPEG image on the platform consists in decompressing and then recompressing the image (see Fig. 1, bottom). Before recompression, the JPEG image is decoded by first performing entropic decoding, coefficient scaling according to the quantization matrix, and inverse DCT transform. This is a lossy process because once decompressed, the pixel values are rounded to integer values between 0 and 255. This rounding operation in the pixel domain can modify a fraction of the DCT coefficients after recompression. Such an effect will be particularly significant (but not only) on blocks containing pixels initially clipped to 255 since, after decompression, the clipping may change the magnitude of DCT coefficients (see example depicted in Fig. 2).

D. An example: Slack instant messaging

In this section, we want to give a real-world example of where robust steganography is needed. As a representative, we chose the *Slack* application for instant messaging[3]. During our experiments tested on MacOS Version 4.31.156, we learned that given a JPEG image, *Slack* does, in fact, recompress the image before transmitting it to the receiver. As such, standard steganographic tools cannot be used, as the recompression would destroy the embedded message. This paper aims to propose a steganographic scheme that is robust to such a process. We learned that the recompression in *Slack* produces the exact same DCT coefficients and quantization tables as if produced by `convert` (see Section IV) without specifying any compression quality parameters.

```
convert input_name.jpg output_name.jpg
```

This process is equivalent to the use of the closest standard quantization table of `input_name.jpg` during the generation of `output_name.jpg`. This allows us to simulate the *Slack* recompression by simply locally calling the `convert` compressor. All the results reported for `convert` are thus directly applicable to the communication on the *Slack* social network.

III. Robust Embedding

A. Overview of the Embedding Algorithm

This section presents the robust embedding strategy. For better readability, we outline the general idea of the embedding scheme and only then describe the details of every mechanism involved.

First, we define a robust coefficient and how to extract the set of robust coefficients. Since one embedding change can potentially affect the robustness of many DCT coefficients from the same 8×8 block, we divide the image into 64 non-overlapping lattices (one per DCT mode) and perform the embedding iteratively on every lattice separately. We assume for simplicity that every 8×8 block is coded independently during the JPEG compression (we shall see in Section V the impact of this assumption when it is not the case).

[3]https://slack.com

Then, we show how to use the robustness of the coefficients during embedding. And finally, we look at how much payload shall be embedded in each of the 64 lattices.

The scheme of the embedding procedure is shown in Fig. 3.

B. Robustness

For ease of understanding, we introduce several definitions.

Definition III.1 (Processed modes)**.** Given k-th DCT mode (with a pre-defined ordering of modes), $k \in \{1, \dots, 64\}$, denote $\mathcal{P}_k = \{1, \dots, k-1\}$, $\mathcal{P}_1 = \emptyset$ the set of all modes that have already been processed by the algorithm.

Definition III.2 (Pseudo-stego)**.** The k-th pseudo-stego is the cover image with already embedded lattices from $\mathcal{P}_k$.

We will describe the process for a single 8×8 block of single-compressed DCT coefficients of the k-th pseudo-stego $\mathbf{c} = \{c_n\}_{n=1}^{64} \in \mathbb{Z}^{64}$. Let $i_n \in \mathbb{Z}^{64}$ be a vector containing $i \in \mathbb{Z}$ at n-th coordinate and zeros elsewhere. Let $R(\mathbf{c}, i_n) \in \mathbb{Z}^{64}$ denote the recompressed DCT coefficients of $(\mathbf{c} + i_n)$.

Definition III.3 (Robust coefficient)**.** We say a coefficient c_k is robust towards an embedding change $i \in \{-1, 1\}$, if during recompression it:

(R1): Does not change processed modes:

$$\forall l \in \mathcal{P}_k : R(\mathbf{c}, i_k)_l = R(\mathbf{c}, \mathbf{0})_l.$$

(R2): Preserves a change by i:

$$R(\mathbf{c}, i_k)_k = c_k + i.$$

(R3): Preserves no change:

$$R(\mathbf{c}, \mathbf{0})_k = c_k.$$

The sets of all robust coefficients towards $+1$ and -1 are denoted $\mathcal{R}_k^+$ and $\mathcal{R}_k^-$ respectively.

If a coefficient does not belong to a robust set, we say it is non-robust. The set of all non-robust coefficients is denoted $\mathcal{R}_k^0$.

Note that without (R1), embedding in the k-th lattice would destroy the message encoded in a lattice $l \in \mathcal{P}_k$, which would make the secret message unreadable.

(R2) states that the embedding change needs to survive the recompression.

(R3) gives us a choice during the embedding, whether to change the coefficient or not. We want to point out that a coefficient can belong to both sets R_k^+ and R_k^-.

We illustrate these conditions graphically with a 3×3 DCT block (for the sake of simplicity) in Fig. 4.

From the construction of the robust sets, the non-robust coefficients are coefficients c_k, such that either $R(\mathbf{c}, \mathbf{0})_k \neq c_k$, or $R(\mathbf{c}, i_k)_k \neq c_k + i$, $i \in \{-1, 1\}$. In either case, it is essential to note that even though we cannot have control over the recompressed coefficient, we can compute its value $R(\mathbf{c}, \mathbf{0})_k$. We can see that this was, in fact, the case for one of the already

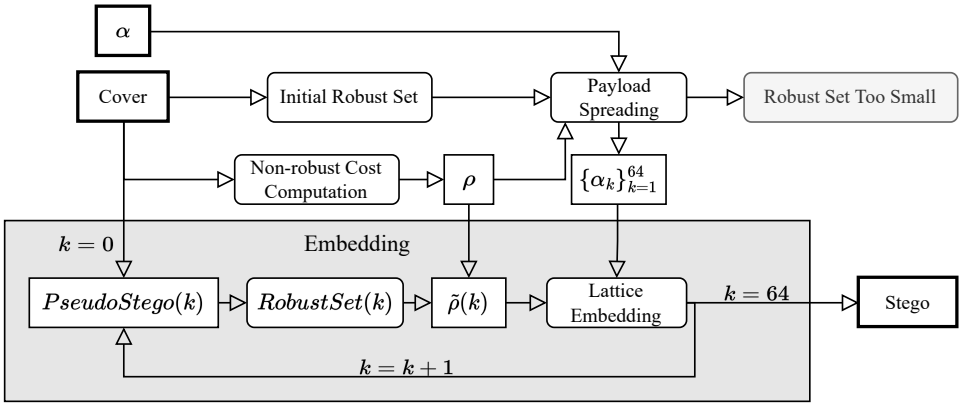

Fig. 3. Main steps of the embedding algorithm.

processed modes (0,1) in Fig. 4: its value changed from 3 to 2, but it does not prevent us from correctly embedding. For practical implementation with STCs, we perform standard embedding on the robust sets (with their possible embedding values) and do not change the non-robust set. This is done by setting their corresponding embedding costs to wet costs. Since we do have access to $R(\mathbf{c}, \mathbf{0})$, we are still able to encode the message in the trellis. Section IV explains the embedding mechanism in further detail.

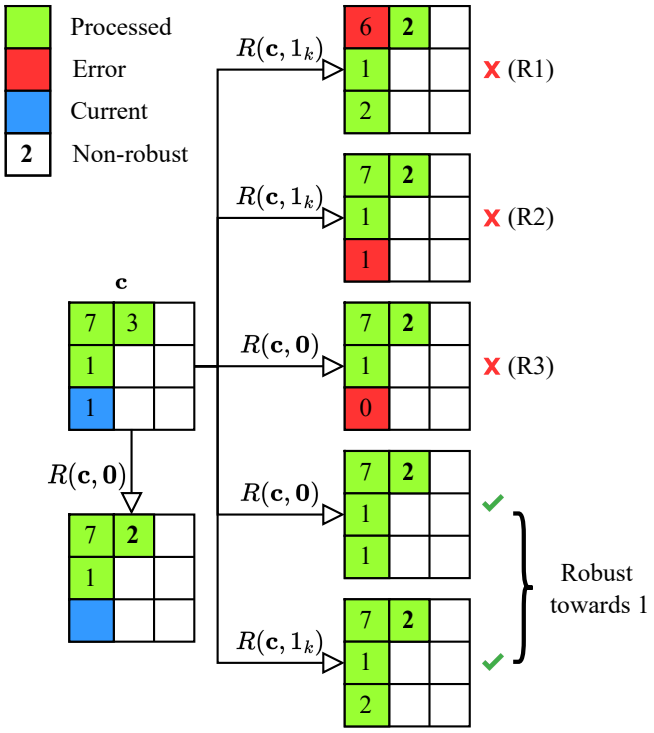

Fig. 4. Mechanism for deciding robustness of a DCT coefficient. Left: DCT coefficients $\mathbf{c}$ and their recompressed values, Right: Possible effects of recompression on the current and processed lattices. The first three situations, each associated with one condition R1, R2, or R3, assign the coefficient to a non-robust set. Only if the last two situations arise is the coefficient robust towards 1.

C. Lattice Embedding

To embed a given lattice, we need to compute the robust costs. Let ρ^+, ρ^- denote the embedding costs (computed from a given steganographic algorithm) of changing a DCT coefficient c by $+1$ or -1. The robust costs $\tilde{\rho}^+, \tilde{\rho}^-$ are created by updating the original costs:

$$\begin{cases} \tilde{\rho}^{\pm} = \rho^{\pm}, & c \in \mathcal{R}^+ \cap \mathcal{R}^-, \\ \tilde{\rho}^- = \infty, & c \in \mathcal{R}^+, \\ \tilde{\rho}^+ = \infty, & c \in \mathcal{R}^-, \\ \tilde{\rho}^{\pm} = \infty, & c \in \mathcal{R}^0. \end{cases} \tag{1}$$

In other cases, we keep $\tilde{\rho}^+ = \rho^+$ and $\tilde{\rho}^- = \rho^-$.

Let α_k be the portion of the total payload we desire to embed in the k-th lattice. For practical embedding, we would provide the embedding costs and payload to STCs to perform the embedding. However, in this work, we simulate the optimal embedding. Therefore we find the optimal change rates:

$$\beta_k^{\pm} = \frac{e^{-\lambda \tilde{\rho}_k^{\pm}}}{1 + e^{-\lambda \tilde{\rho}_k^+} e^{-\lambda \tilde{\rho}_k^-}}, \tag{2}$$

where $\lambda > 0$ is the Lagrange multiplier ensuring that we embed the desired payload

$$\sum_{n=1}^{N} H_3(\beta_n^+, \beta_n^-) = \alpha_k,$$

and $H_3(\beta^+, \beta^-)$ is the ternary entropy function

$$H_3(\beta^+, \beta^-) = \\ -(1-\beta^+-\beta^-)\log(1-\beta^+-\beta^-) - \beta^+ \log \beta^+ - \beta^- \log \beta^-.$$

Having the optimal change rates, the coefficients from this lattice are embedded. That concludes the embedding of the given lattice, and we move on to the next.

D. Payload Spreading

Since our proposed method embeds iteratively 64 non-overlapping lattices, we need to decide *a priori* what portion of the embedding payload is carried in every lattice. Therefore, we compute the so-called "Initial robust set" for every lattice

– the robust set without any embedding change. Let α denote the total payload in bits we want to embed. Having the robust sets $\mathcal{R}^+, \mathcal{R}^-, \mathcal{R}^0$, we update the embedding costs (Eq. (1)) and compute the optimal change rates (Eq. (2)) for the whole image. From these change rates, we compute the proportion of payload α_k in every lattice:

$$\alpha_k = \sum_{n=1}^{N} H_3(\beta^+_{n,k}, \beta^-_{n,k}),$$

where $\beta^{\pm}_{n,k}$ is the n-th change rate in the k-th lattice. At this stage, it is possible that we cannot communicate the desired payload because of the small size of the robust set. In such a situation, we have no choice but to use a different image or a smaller secret message.

We are also well aware that there could be another potential issue during actual embedding with this approach of spreading the payload among lattices. In particular, the robust set in the k-th lattice could be of a very different size after embedding the previous lattices. As a result, we would not have enough robust coefficients to use for embedding a prescribed payload α_k. However, in practice, we observed that the size of robust sets in each lattice changes in a negligible way, see Fig. 8.

E. Filtering before recompression

The presented algorithm can be easily extended to be robust to a filtering operation (*e.g.* blurring, sharpening, ...) occurring before the re-compression. This processing can be done on the platform to improve the rendering of the image once published.

Assuming that the filter window size is smaller or equal to 8×8 pixels, this filtering operation can propagate changes between one block and its 8 neighbors, but not between non-adjacent blocks. We need in this case to embed separately in 9 extra macro-lattices, *i.e.* sets of JPEG blocks that are distant by 16 pixels in one or two directions as depicted in Fig. 5. The final number of lattices in this case is consequently multiplied by 9.

One might wonder why 4 macro-lattices are not enough. The answer relies on the fact that each mode is affected by the same $\{-1, 0, +1\}$ change and recompression. Consequently, if after one embedding operation, one change occurs in a block belonging to a previously visited macro-lattice, there is an ambiguity to know which of the two neighboring blocks is responsible for the change.

F. Number of calls to the compressor

In the end, with only recompression the embedding requires $3 \times 64 \times 2$ calls to the compressor : one for each embedding modification in the set $\{-1, 0, 1\}$ multiplied by the 64 DCT modes multiplied by the number of steps necessary to perform the embedding, *i.e.* the estimation of the robust set for payload spreading and the payload embedding. If the robustness has to deal with filtering and compression, then the number of calls equals $9 \times 3 \times 64 \times 2$. Note that a call to a JPEG compressor is rather fast and the (UERD) embedding operation robust to the compressor takes about 10s on a Macbook Pro with M1 chip on a 512×512 image and the `convert` compressor, and 90s when filtering is involved. Note that the filtering makes the scheme 9 times slower because we need to consider 9 times more lattices, as explained in the previous section.

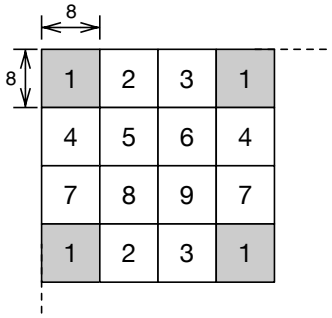

Fig. 5. The 9 macro-lattices used to be robust to filtering.

IV. Practical Implementation

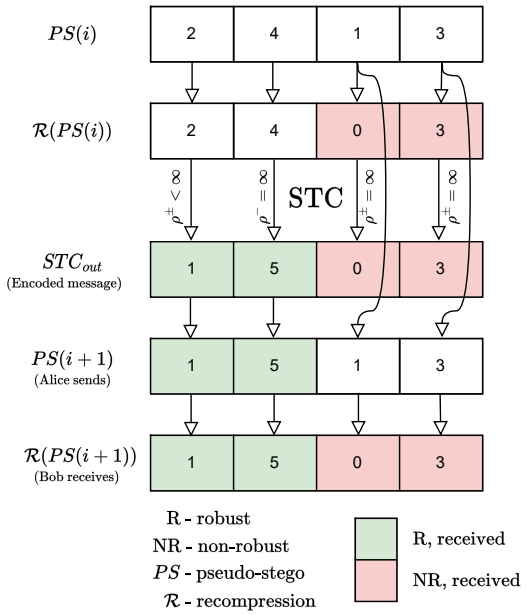

Fig. 6. Practical embedding of i-th lattice with Syndrom-Trellis Codes: Alice sends a bitstream which takes into account the recompression $\mathcal{R}(.)$ in order to be compliant with both the STC coding and decoding processes. The costs ρ are also chosen in order to deal with non-robust modifications.

In this section, we want to detail the technicalities necessary for actual embedding. First, JPEG compression is significantly impacted by the compressor. We consider three different JPEG compressors in this work: ImageMagick's `convert`, `mozjpeg` (by default with rate-distortion optimization), and `mozjpeg` without optimization (`-notrellis` option). As mentioned earlier in Section II-C, the main difference between these compressors is the quantization tables used. Specifically, `mozjpeg` uses stronger quantization (bigger quantization steps) than `convert`. This harsher quantization positively affects the robustness of an image. In Fig. 7, we show the average initial robust size over 10 images compressed with the three compressors across a range of quality factors and with the random scanning strategy (see Section IV-A for more

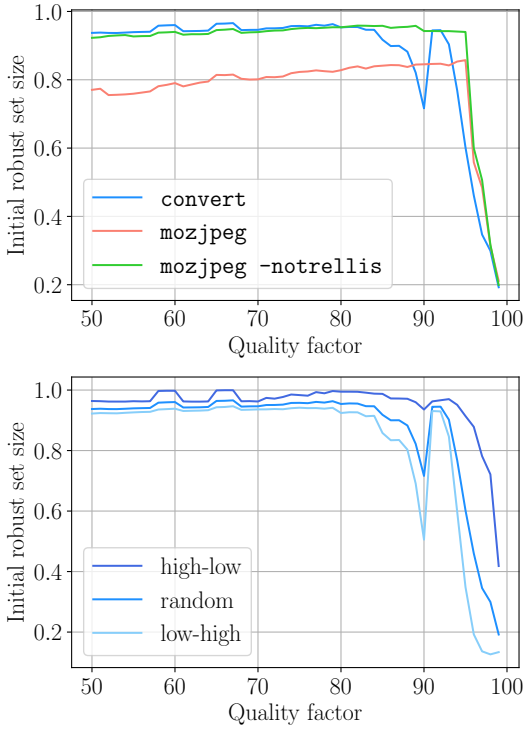

Fig. 7. Relative robust set size average over 10 cover images. Top: Random scanning strategy, bottom: `convert`.

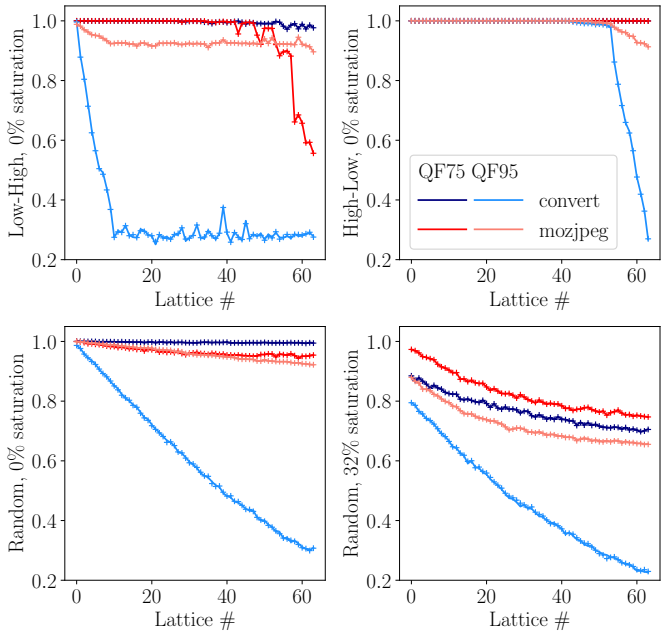

Fig. 8. Proportion of robust coefficients in every lattice during embedding. The solid line corresponds to a cover image (Initial robust set), while the crosses mark a stego image embedded with 0.8 bpnzAC. The first three figures are from a single BOSSBase image of size 512×512 without any saturated pixels across three scanning strategies. The last figure shows the robustness of an image with 32% of its pixels saturated and random scan.

details about scanning strategies). We can see that for quality factors above 80, the images compressed with `convert` have a smaller robust set than those compressed with `mozjpeg` without the rate-distortion optimization. This is especially true around quality factor 90, where `convert` has a sudden drop in the robust set size. Strangely enough, `convert`'s robust size jumps back 93% for qualities 91 and 92. Although we are not sure why exactly this phenomenon is happening, we are convinced it is related to the quantization tables because we are unaware of another substantial difference between the compressors. Additionally, we can notice that disabling `mozjpeg`'s rate-distortion optimization increases the robust set size, mainly for qualities below 95.

A. Scanning Strategies

Secondly, we can notice that in the definition of *Processed modes*, we assumed a given ordering of DCT modes. We consider three natural scanning strategies:

1) Low-High: Scan modes in a zig-zag manner from low to high-frequency modes (as done in JPEG),
2) High-Low: Reverse the order of Low-High, and
3) Random: Randomly assign ordering of modes in every 8×8 block.[4]

We will see in Section V that these three strategies will affect the empirical security of the embedding scheme.

Next, a scanning strategy dictates the size of the robust set in every lattice. In Fig. 8, we show the relative size of the robust set across all 64 lattices. The top two plots show the Low-High and High-Low scanning strategies for an image without saturated pixels, while the bottom plots show the Random scanning strategy for a non-saturated and a greatly saturated image. We can make several interesting observations from this figure. Robustness at QF 95 is generally smaller than for QF 75. This is to be expected, as bigger quantization steps (lower quality) provide more robustness against recompression. Similarly, since `mozjpeg`'s quantization steps are mostly bigger than those of `convert`, `mozjpeg` yields on average better robustness. We observe that the Random scan linearly decreases the robustness as we proceed through the lattices. It is important to realize that in the Random strategy, one lattice does not correspond to a DCT mode, as every block is scanned in a different random order. This ensures a relatively equal payload across all lattices, which allows us to skip the computation of the Initial robust set. In contrast, the first two scanning strategies assign one lattice per DCT mode, and it can be seen (especially for QF 95, `convert`) that most of the payload is carried in the low-frequency modes because their robustness decreases drastically. Finally, the saturation of pixels reduces the robustness due to the clipping of pixels into the dynamic range $[0, 255]$ during recompression. During our experiments, all non-saturated images exhibit a similar trend, while robustness tends to decrease with increasing saturated area. Interestingly, we can also notice that the size of the Initial robust set in every lattice is not very different from the robust set of an image embedded with 0.8 bpnzAC. This justifies our payload allocation based on the Initial robust set (see Section III-D). If it were not the case, there would be a

[4]The pseudorandom key used for generating the permutations can be a part of the secret key.

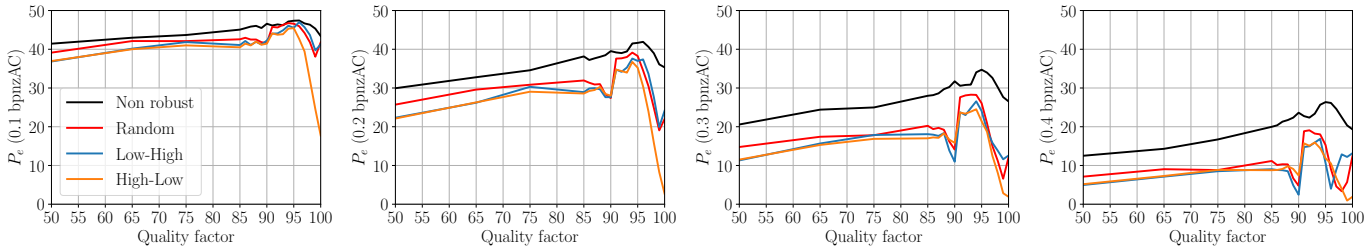

Fig. 9. Detection errors with `convert`.

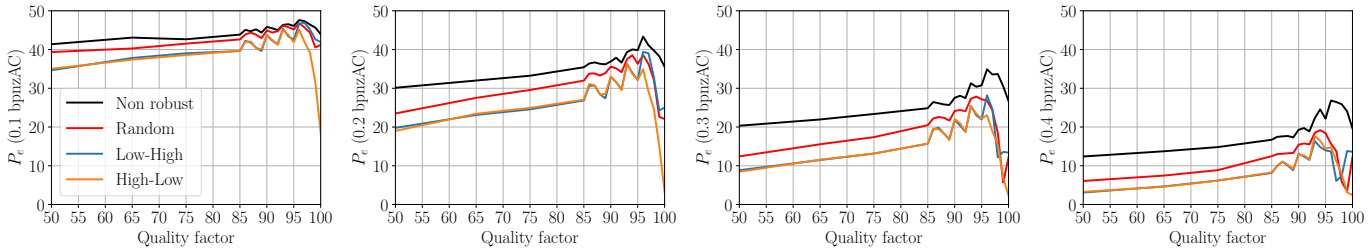

Fig. 10. Detection errors with `mozjpeg` without rate-distortion optimization.

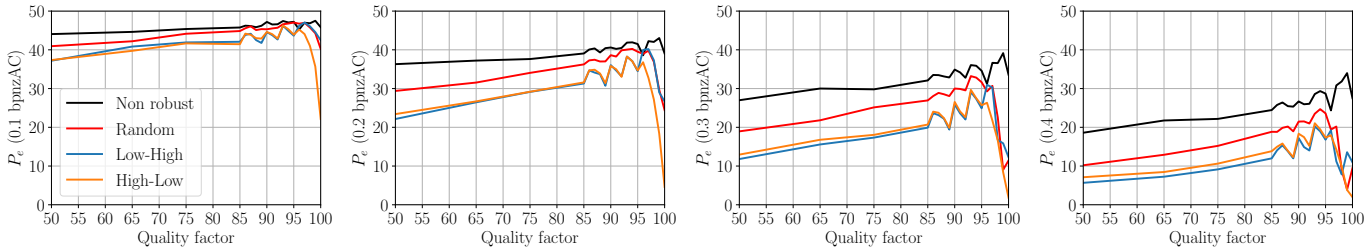

Fig. 11. Detection errors with `mozjpeg` with rate-distortion optimization.

risk of having a lattice with a robust set that is too small for the desired payload.

B. Syndrome-Trellis Coding

Lastly, we explain how to use the proposed methodology for practical embedding with Syndrome-Trellis Codes (STC). Fig. 6 shows Alice's action on a single lattice. Given i-th pseudo-stego, she will take the i-th lattice as a vector and inspect its recompressed values to asses the coefficients' robustness. She will then perform embedding on the recompressed lattice according to Section III-C. However, she cannot simply send the output of the coding mechanism because the channel's recompression can potentially change the non-robust coefficients, which would prevent Bob from reading the secret message. Instead, Alice puts the original non-robust coefficients (before recompression) back into the lattice, which yields the $(i+1)$-th pseudo-stego. This way, it is ensured that after the recompression, Bob will be able to decode the secret message.

For the actual STC implementation, we used a python wrapper of a C++ implementation, with the height constraint $h = 10$.[5] With this particular implementation, we observed an undesirable behavior when the message size is 'too small' (*e.g.* less than 0.5% of the changeable elements), namely the codes would make an embedding change of magnitude 2. While this would not happen often, it would ultimately destroy the communicated message. Since this seems to be the STC implementation flaw, we discarded the experiments affected by this phenomenon. From a practical point of view, this is not an issue, as the steganographer can easily verify if the coding mechanism created such an embedding change and either embed the cover image again or discard this combination of image/message for transmission.

[5]https://github.com/daniellerch/pySTC

V. Results

The goal of this section is to benchmark the different characteristics of the proposed scheme within the framework of robust steganography. The considered setup will be the same as the one depicted in Fig. 1 and recalled in Section II-B, *i.e.* the steganographer will embed its payload on a single-compressed image and submit it on the platform that will compress the stego image. On the other side, the steganalyst will observe contents that are published on the platform, *i.e.* contents which are double-compressed, either in their cover or stego version. The lossy coding algorithm and used parameters by the steganographer and the platform will be identical. The considered figures of merit are the following:

Emb. rate (bpnzAC)	Scan	Coder	94	95	96	97	98	99	100
0.1	all	all	100	100	100	100	100	100	100
0.2	Random and High-Low	all	100	100	100	100	100	100	100
0.2	Low-High	convert	100	100	99.98	99.72	94.53	89.95	99.97
0.2	Low-High	mozjpeg, no optim	100	100	100	99.98	98.22	88.25	99.89
0.2	Low-High	mozjpeg, optim	100	100	99.98	99.98	99.44	92.91	99.92
0.3	Low-High	convert	100	99.13	99.05	71.18	44.03	25.09	30.07
0.3	High-Low	convert	100	100	100	100	100	100	20.21
0.3	Random	convert	100	100	100	100	100	87.09	27.12
0.3	Low-High	mozjpeg, no optim	100	100	99.08	99.05	58.13	28.99	30.24
0.3	High-Low	mozjpeg, no optim	100	100	100	100	100	100	19.93
0.3	Random	mozjpeg, no optim	100	100	100	100	99.99	98.31	27.55
0.3	Low-High	mozjpeg, optim	100	100	99.96	99.89	72.74	45.64	49.37
0.3	High-Low	mozjpeg, optim	100	100	100	100	100	100	63.5
0.3	Random	mozjpeg, optim	100	100	100	100	99.98	99.39	57.82
0.4	Low-High	convert	100	99.95	84.53	34.9	18.8	10.18	7.96
0.4	High-Low	convert	100	100	100	100	100	100	4.82
0.4	Random	convert	100	100	100	100	99.98	33.12	6.65
0.4	Low-High	mozjpeg, no optim	100	100	99.94	96.71	23.03	11.85	7.89
0.4	High-Low	mozjpeg, no optim	100	100	100	100	100	100	4.83
0.4	Random	mozjpeg, no optim	100	100	100	100	99.97	59.59	6.63
0.4	Low-High	mozjpeg, optim	100	100	99.92	98.07	39.73	22.34	19.97
0.4	High-Low	mozjpeg, optim	100	100	100	100	100	100	34.83
0.4	Random	mozjpeg, optim	100	100	100	99.98	99.97	76.0	26.61

TABLE II
EMBEDDING SUCCESS RATES FOR DIFFERENT JPEG QUALITY FACTORS (IN %) WITH CONVERT AND MOZJPEG WITH AND WITHOUT RATE-DISTORTION OPTIMIZATION.

- The *practical security* of the robust scheme for different scanning strategies and JPEG compressors. It is also important to compare it w.r.t. the naive embedding which is not robust but maximizes the security and can be considered as a baseline.
- The *embedding success rate*, *i.e.* the probability to be able to embed the prescribed payload. The size of the robust set being limited, for high-quality factors, the robust set may be too small on several images.
- The *impact of the compressor*, such as `mozjpeg` or `convert` (used by the *Slack* application, see Section II-D), which can use specific quantization matrices or can optimize the rate-distortion trade-off and which can change quantization values before applying Huffman coding.

These different features are evaluated using a classical steganography/steganalysis setup:

- Images from BOSSBase [21] in greyscale format are used as a source of covers. Images in the pixel/PGM format are used as pre-covers and then compressed with the appropriate compressor.
- Regarding steganography, UERD [2] is chosen as the embedding scheme because it offers a good tradeoff between complexity and practical security. Different payload sizes ranging from 0.1 bpnzAC to 0.4 bpnzAC are adopted.
- Regarding steganalysis, DCTR features [22] are combined with the regularized linear classifier [23]. 5,000 pairs of images are used for training, and 5,000 pairs for testing.
- The classical probability of error P_e minimizing the sum of false positive and false negative rates during training is reported as a measure of practical security.

Different quality factors, ranging from 50 to 100, are reported to analyze the evolution of the mentioned features w.r.t. JPEG quantization. Note, however, that there is a substantial discrepancy between the quantization matrices used by `libjpeg/convert` and `mozjpeg`, the quantization steps used by `mozjpeg` being always greater or equal to the quantization steps used by `convert`.

Note also that, except when explicitly mentioned in the case of rate-distortion optimization, the proposed scheme is errorless. Consequently, no error-correcting codes need to be used, and no error transmission probability needs to be reported.

A. Practical security

Fig. 9, Fig. 10 and Fig.11 present respectively the detection error P_e for respectively `convert`, `mozjpeg` without (option `-notrellis` added) and `mozjpeg` with the rate-distortion trade-off.

Several remarks can be drawn from this extensive set of experiments.

a) On the impact of the scan strategy: The strategy of randomly picking the DCT modes for each lattice offers a gain of practical security w.r.t. to scans starting with low frequencies or high frequencies except for very high-quality factors (*i.e.* ≥ 95). For quality factors below 85, the gain associated with the random scan is between 2 and 5% w.r.t. the other scans. However, one can choose the scan starting with low-frequency modes for high-quality factors. It is also interesting

to notice that starting with low frequencies is, on average, a better strategy than scanning the high frequencies first, even if the size of the robust set is far larger with the second option (see Fig. 8). We can see that there is a tradeoff between the size of the robust set and the modes it considers as robust.

On one side, the high-frequency modes are more robust since they are associated with bigger, hence more conservative, quantization steps; on the other side, they are more detectable.

Note also that, on average, the random-scan strategy has to be preferred because of its higher security and its possibility to spread the payload without first computing the robust set (see Section IV).

b) On the gap between robust and non-robust embedding: When comparing with the most favorable embedding strategy, we can observe that, at 0.1 bpnzAC, the gap is very small (*i.e.* below 3% in terms of detection error) but becomes substantial (*i.e.* reaching more than 10% for few quality factors) for larger embedding rates and high-quality factors. This is not surprising since we have seen in Section IV that the size of the robust set tends to decrease w.r.t. the quality factor, which means that the embedding algorithm has to perform more embedding changes. Indeed, for the same payload size, the number of embedding changes decreases w.r.t. the number of changeable coefficients. The larger detectability is also due to the fact that the non-robust coefficients initially associated with small embedding costs in a non-robust setting cannot be modified anymore with the proposed scheme. They are lost for embedding.

c) On the impact of the quality factor: For the `convert` coder, we can notice a "bump" for quality factors between 90 and 100, which is associated with an increase of detectability w.r.t. the non-robust scheme. For the `mozjpeg` coder, we can also observe oscillations of the overall detectability. We hypothesize that these two phenomena are due to the interplay between DCT modes quantized with specific steps. From Fig. 7, we can observe that the oscillations in terms of detectability are on par with the ones coming from the size of the robust sets. Note also that part of the non-monotony of the detectability is also due to the quantization tables only and was already observed for plain JPEG steganography [24].

d) On the impact of the coder: If, as reported above, the different quantization tables used by the two coders are associated with different detectabilities, we can also notice that whenever the `mozjpeg` coder uses rate-distortion optimization, the practical detectability is smaller. If we noticed that the payload size is smaller (the number of 0s increases by about 10% after the optimization), it is probably not the only reason since the same decrease of 0s is observed between `convert` and `mozjpeg` without optimization.

We hypothesize that the produced cover images with optimization are also more "secure" sources since they have less isolated non-zero modes due to the optimization process.

e) On the impact of filtering: In Fig. 12 we show how the maximum embedding capacity changes if we additionally process the decompressed image before recompression. We used `convert` at two different quality factors and a Low-High scanning strategy. Two operations are considered: 3×3 blurring with Gaussian kernel and 3×3 sharpening (both available in `convert`). The capacity was computed assuming optimal coder allowing to communicate 1 bit per coefficient robust towards one embedding change and $\log_2 3$ bits per coefficient robust towards both embedding changes. We can see that both processing operations decrease the capacity from up to $\log_2 3$ bits per coefficient (bpc) to less than 0.35 bpc at QF 75, and at QF 95, the maximum capacity decreases from $0.3-0.45$ bpc to less than 0.06 bpc, preventing the sender from using bigger messages. Moreover, since the attainable payloads are so small, the security suffers as well, because the steganographer simply cannot commit to embedding changes associated with small embedding costs. At 0.1 bpnzAC, the probability of error is 3% for QF 75 and 1% for QF 95 respectively. However, despite increased detectability and limited embedding capacity, the robustness of the method is still guaranteed.

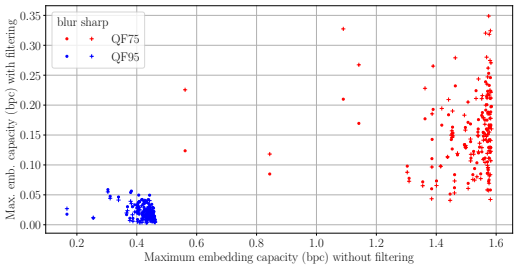

Fig. 12. Maximum embedding capacity evolution with extra spatial filtering over 100 randomly selected images with `convert` and Low-High scanning strategy.

f) Detectability with SOTA detectors: Because of the increased complexity of Deep Learning steganalyzers, we provide only limited evaluation with JIN-SRNet [25], [26]. In Table III, we show the detection error rates for the proposed scheme with a Low-High scanning strategy and original (non-robust, single-compressed) UERD. Interestingly enough, the proposed scheme is in most cases more secure than the original UERD in single-compressed images. We believe this is due to preventing some 'easy-to-catch' embedding changes during the recompression process. Most importantly, we see that our robust methodology does not increase the detectability even with a state-of-the-art detector.

g) Detectability before and after recompression: As a side-experiment, we also performed steganalysis on Cover or Stego images before recompression (this would be equivalent to a practical scenario where Eve has access to inputs $(\mathbf{C_1}, \mathbf{S_1})$ of the platform in Fig. 1), and we did not notice any gap between the practical security of the scheme before or after recompression.

B. Embedding success rate

We analyze here when the embedding fails, *i.e.* when the payload size is too small or too large to not modify only robust coefficients.

1) Impact of a small robust set: Because the size of the robust set can be limited, the maximum embedding capacity for ternary embedding can be smaller than $\log_2(3) \simeq 1.58$ bit

Method	QF	Emb. rate (bpnzAC)			
		0.1	0.2	0.3	0.4
Proposed	75	0.1392	0.0525	0.0195	0.0088
	95	0.2742	0.1420	0.0403	0.0085
UERD (non-robust)	75	0.1155	0.0428	0.0197	0.0075
	95	0.2423	0.1195	0.0735	0.0330

TABLE III
DETECTION ERROR OF SRNET WITH CONVERT AND LOW-HIGH SCANNING STRATEGY.

Method	QF	Emb. rate (bpnzAC)					
		0.1	0.2	0.3	0.4	0.5	0.6
Proposed	75	100	100	100	100	100	100
	92	100	100	100	100	100	99
	95	96	95	92	92	80	73
	100	0	0	0	0	0	0
[9]	75	0	0	0	0	0	0
	92	0	0	0	0	0	0
	95	0	0	0	0	0	0
	100	0	0	0	0	0	0
[10]	75	0	0	0	0	0	0
	92	0	0	0	0	0	0
	95	0	0	0	0	0	0
	100	0	0	0	0	0	0

TABLE IV
STC EMBEDDING SUCCESS RATES (IN %) FOR DIFFERENT EMBEDDING SCHEMES WITH CONVERT AT QUALITY FACTORS 75, 92, 95, AND 100. THE PROPOSED METHOD WAS COMBINED WITH THE RANDOM SCANNING STRATEGY. HEIGHT CONSTRAINT OF THE STC WAS SET TO $h = 10$.

per coefficient. If the prescribed embedding rate (chosen to be in bpnzAC) is larger than the maximum achievable rate, we consequently report an embedding failure. Table II reports the embedding success rate computed on the BOSSBase database for the different coders, different scanning strategies, embedding rates ranging from 0.1 bpnzAC to 0.4 bpnzAC, and different quantization factors. We can draw several conclusions:

- For QF ≤ 94, the embedding success rate reaches 100%, but the higher the quality factor, the less robust the embedding is. This is due to the fact that, in this range of quality factors, the size of the robust set decreases.
- The Low-High scan is less robust than the Random scan, which is less robust than the High-Low scan. This is coherent with the size of the robust sets, which follow the same trend (the robust set associated with the Low-High is smaller than the robust set associated with the Random scan, which is smaller than the one associated with the High-Low) plotted in Fig. 7
- The `mozjeg` coder with rate-distortion optimization is more robust than the same coder without optimization, which is, in turn, more robust than the `convert` coder. Again, this is coherent with the hierarchy on the robust set size, plotted in Fig. 7.

2) Impact of STCs: Since the above analysis is assuming optimal coding mechanism, we now investigate the effect of a practical coding scheme, the STCs. Table IV shows embedding success rate across different methods while using `convert` coder. We consider the embedding to be successful, only if the whole secret message can be retrieved from the recompressed image. We can see that our method starts failing only for high qualities and with increasing payload. This is due to the small robust set size, which in turn forces the (sub-optimal) STC to embed into a non-robust coefficient. This is in line with the sudden drop in robust set size for qualities above 92 depicted in Fig. 7. Note that this could be already verified at the sender's side and the steganographer can thus avoid sending a non-robust stego image. This problem is inherently related to the use of STCs and the fact that for very low or very high payload sizes, wet costs can be used.

C. Impact of rate-distortion strategies

This last experiment investigates to what extent the rate-distortion strategy, which is used by the `mozjpeg` coder to decrease the file size by changing DCT coefficients values and to decrease their associated Huffman code length, is detrimental to the robustness of the scheme. This is conducted out of curiosity since we know in advance that the proposed scheme is not robust to change of coefficient after quantization. Fig. 13 shows the ratio of images that can convey the payload when this option is activated.

Here we can see two behaviors:

1) For quality factors ≤ 95, starting with low-frequency coefficients increases the correct extraction rate significantly. For quality factors > 95, starting with the High-frequency coefficient offers the best extraction rate. Moreover, there is an overall drop in extraction rate for quality factors > 95, which is caused by reduced robust set size (see Fig. 7).
2) The larger the embedding rate, the smaller the number of images having a correct extraction. However, we can note that in a favorable setting (Low to High scan and QF below 95 at 0.1 bpnzAC), the ratio of the correctly extracted payload is larger than 60%.

D. Comparison with prior art

In Table V, we compare the security of the proposed method at JPEG qualities 75 and 95 to those of MINICER [9] and SSR [10]. To have a fair security comparison without any issues caused by STCs, we only used simulated embedding. Because the SSR method changes the sign of DCT coefficients during embedding, we see very small detection errors across all payloads for both quality factors. The MINICER, on the other hand, has a security performance comparable to the proposed method at QF 75 and is even more secure at the higher quality. We explain this by the fact that MINICER assigns wet costs to all the DC modes during its cost update. This is a detail worth commenting upon because we observed in our experiments that the DC mode is producing the most robust coefficients. We, therefore, believe that MINICER does this update simply for security reasons. We will not follow this logic because we are not designing a new steganographic scheme, but instead are giving a general methodology for robust steganography. To this end, our main objective is the robustness of a given embedding scheme.

Finally, we would like to point out that even though MINICER is less detectable than our method, Table IV shows that both [9] and [10] are not robust in any of the studied

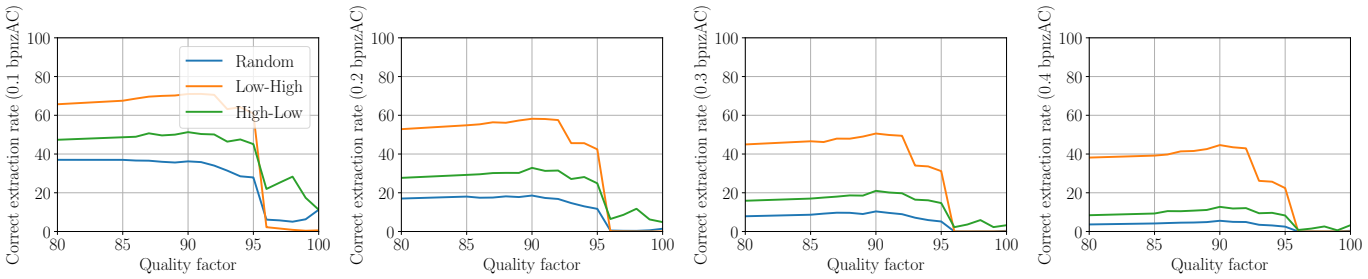

Fig. 13. Correct extraction rate (in %) for `mozjpeg` with rate-distortion optimization for different embedding rates.

cases, while our method is fully robust at the lowest quality and more than 90% robust even at quality 95 for payloads below 0.5 bpnzAC. This can be explained by studying the two methods in a little bit more detail. We learned that [10] relies on the fact that embedding changes do not change the sign of the DCT coefficients. It is practically not true because many embedding changes are performed on coefficients equal to ± 1. Consequently, after recompression, the sign of these coefficients can change with a high probability to zero. More importantly, the robust set computed by the method is very small because instead of robust coefficients, the method uses robust DCT modes, where a DCT mode is robust if all of its coefficients are robust. As such we were not able to robustly embed a payload bigger than 0.01 bpnzAC. The method [9], on the other hand, assumes that DCT coefficients are changed during recompression only in blocks containing saturation. This is however not the only case. We observed that steganographic embedding changes combined with recompression can also cause other coefficients from the same DCT block to change, which gravely affects the robustness of [9].

Note that the scheme proposed in the paper does not have these drawbacks since, thanks to the lattice embedding, a coefficient can be modified if and only if all the previously modified coefficients do not change (see condition R1 in Section III-A).

Method	QF	Emb. rate (bpnzAC)			
		0.1	0.2	0.3	0.4
Proposed	75	0.5000	0.3999	0.2365	0.1732
	95	0.4997	0.4583	0.2926	0.1386
[9]	75	0.4579	0.4050	0.2154	0.1340
	95	0.4999	0.4574	0.4030	0.3618
[10]	75	0.0451	0.0047	0.0019	0.0025
	95	0.0140	0.0082	0.0076	0.0022

TABLE V
DETECTION ERROR WITH `convert`. THE PROPOSED METHOD WAS COMBINED WITH THE RANDOM SCANNING STRATEGY.

VI. Conclusions and Perspectives

In this work, we introduced a methodology for JPEG steganography robust against subsequent recompression. Although the JPEG compressor has to be assumed known, it does not present any obstacles because, in a typical channel, a social media, we can easily access the compressor. Moreover, we noticed that the recompression pipeline present in the professional social network *Slack* produces the very same results as the `convert` compressor, which validates this assumption.

First, we introduced the notion of the robustness of a DCT coefficient. We showed that this could be done by dividing the image into 64 non-overlapping lattices and performing 64 consecutive recompressions (associated with ± 1 modifications) of an image, one per lattice. We introduced three ordering of the lattices: Low to High, High to Low, and Random. We showed that these three strategies offer different robust sets. Then we combined the coefficients' robustness with steganographic costs from a non-robust stego algorithm in a straightforward way to robustify the algorithm. Additionally, it was shown how this could be done in a practical setting with Syndrome-Trellis Codes.

In the last part of the paper, we evaluate the security of our method with machine-learning steganalysis. We observe that the security is affected by everything in the system: Quality Factor, compressor, and the scanning strategy of the lattices. We link the differences in security to different sizes of the robust sets. Moreover, we can observe security loss compared to the non-robust version of the stego algorithm, which is expected because many coefficients with small embedding costs will not be usable for robust embedding. Lastly, unlike any of the preceding works on robust steganography, our method is truly errorless, giving us guarantees on the readability of the embedded secret message. The only exception to this is `mozjpeg` which allows rate-distortion optimization. On the other hand, we have seen that, if successfully embedded, the rate-distortion optimization increases the security of the underlying scheme.

In the future, we plan to derive theoretical bounds on the embedding capacity in the noisy recompression channel. The source code of the proposed robust embedding is available from https://janbutora.github.io/downloads/.

Acknowledgements

This work was granted access to the HPC (High-Performance Computing) resources of IDRIS (Institut du Développement et des Ressources en Informatique Scientifique) under the allocation 2022-AD011012855 made by GENCI (Grand Equipement National de Calcul Intensif). This work received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 101021687 (project "UNCOVER").

REFERENCES

[1] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP Journal on Information Security*, vol. 2014, no. 1, pp. 1–13, 2014.

[2] L. Guo, J. Ni, W. Su, C. Tang, and Y.-Q. Shi, "Using statistical image model for jpeg steganography: Uniform embedding revisited," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2669–2680, 2015.

[3] Q. Giboulot, R. Cogranne, and P. Bas, "Detectability-based JPEG steganography modeling the processing pipeline: the noise-content trade-off," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2202–2217, Jan. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03096658

[4] C. Kin-Cleaves and A. D. Ker, "Adaptive steganography in the noisy channel with dual-syndrome trellis codes," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2018, pp. 1–7.

[5] Y. Zhang, X. Luo, C. Yang, D. Ye, and F. Liu, "A framework of adaptive steganography resisting jpeg compression and detection," *Security and Communication Networks*, vol. 9, no. 15, pp. 2957–2971, 2016.

[6] J. Tao, S. Li, X. Zhang, and Z. Wang, "Towards robust image steganography," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 594–600, 2018.

[7] W. Lu, J. Zhang, X. Zhao, W. Zhang, and J. Huang, "Secure robust jpeg steganography based on autoencoder with adaptive bch encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[8] Z. Zhao, Q. Guan, H. Zhang, and X. Zhao, "Improving the robustness of adaptive steganographic algorithms based on transport channel matching," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1843–1856, 2018.

[9] K. Zeng, K. Chen, W. Zhang, Y. Wang, and N. Yu, "Improving robust adaptive steganography via minimizing channel errors," *Signal Processing*, vol. 195, p. 108498, 2022.

[10] X. Wu, T. Qiao, Y. Chen, M. Xu, N. Zheng, and X. Luo, "Sign steganography revisited with robust domain selection," *Signal Processing*, vol. 196, p. 108522, 2022.

[11] P. Bas, F. Teddy, F. Cayre, G. Doërr, and B. Mathon, *Watermarking Security: Fundamentals, Secure Design and Attacks*. Springer, Jan. 2016.

[12] F. Cayre and P. Bas, "Kerckhoffs-based embedding security classes for WOA data-hiding," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, March 2008.

[13] T. Filler, J. Judas, and J. Fridrich, "Minimizing additive distortion in steganography using syndrome-trellis codes," *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 3, pp. 920–935, 2011.

[14] W. Luo, G. L. Heileman, and C. E. Pizano, "Fast and robust watermarking of jpeg files," in *Proceedings Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*. IEEE, 2002, pp. 158–162.

[15] T. Qiao, S. Wang, X. Luo, and Z. Zhu, "Robust steganography resisting jpeg compression by improving selection of cover element," *Signal Processing*, vol. 183, p. 108048, 2021.

[16] Y. Zhang, X. Luo, J. Wang, C. Yang, and F. Liu, "A robust image steganography method resistant to scaling and detection," *Journal of Internet Technology*, vol. 19, no. 2, pp. 607–618, 2018.

[17] L. Zhu, X. Luo, Y. Zhang, C. Yang, and F. Liu, "Inverse interpolation and its application in robust image steganography," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 6, pp. 4052–4064, 2021.

[18] A. Castiglione, G. Cattaneo, and A. De Santis, "A forensic analysis of images on online social networks," in *2011 third international conference on intelligent networking and collaborative systems*. IEEE, 2011, pp. 679–684.

[19] R. Caldelli, R. Becarelli, and I. Amerini, "Image origin classification based on social network provenance," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1299–1308, 2017.

[20] J. Butora and J. Fridrich, "Revisiting perturbed quantization," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 125–136.

[21] P. Bas, T. Pevny, and T. Filler, "Bossbase," http://exile.felk.cvut.cz/boss, May 2011.

[22] V. Holub and J. Fridrich, "Low-complexity features for jpeg steganalysis using undecimated dct," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 219–228, 2015.

[23] R. Cogranne, V. Sedighi, J. Fridrich, and T. Pevnỳ, "Is ensemble classifier needed for steganalysis in high-dimensional feature spaces?" in *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*. IEEE, 2015, pp. 1–6.

[24] J. Butora and J. Fridrich, "Effect of jpeg quality on steganographic security," in *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, 2019, pp. 47–56.

[25] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, 2018.

[26] J. Butora, Y. Yousfi, and J. Fridrich, "How to pretrain for steganalysis," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 143–148.