Telltale Watermarks for Counting JPEG Compressions

Matthias Carnein; Department of Information Systems, University of Münster; Münster, Germany
Pascal Schöttle; Institute of Computer Science, Universität Innsbruck; Innsbruck, Austria
Rainer Böhme; Institute of Computer Science, Universität Innsbruck; Innsbruck, Austria

Abstract

Telltale watermarks allow to infer how a watermarked signal has been altered on the channel by analyzing the distortion applied to the watermark. We propose a new kind of telltale watermark for digital images that tracks the number of JPEG (re-)compressions. Our watermarks leverage peculiarities in the convergence behavior of JPEG images. We show that it is possible to generate or find combinations of pixel values which form so called "counter blocks". Counter blocks cycle predictably through a fixed number of states in subsequent JPEG compression and decompression operations. By combining counter blocks with different cycle lengths in one image, we are able to track the number of JPEG (re-)compressions in extended ranges. We evaluate the accuracy of counter blocks, discuss pitfalls when embedding them, and study the construction of counter blocks with specified cycle lengths.

Introduction

When repeatedly compressing and decompressing digital images with the popular JPEG standard, we typically observe that the pixel values converge after a certain number of compressions. This phenomenon has been previously used in image forensics to estimate the number of times an image has been (re-)compressed. However, for color images, using chrominance subsampling with a linear interpolation of chrominance values, we observe that some blocks do not converge. Instead they cycle through a fixed number of states. This is because the chrominance interpolation introduces an error which can interfere with the convergence of blocks. Such cyclic blocks enable interesting application scenarios.

In this paper we explore the usage of these cyclic blocks as a new form of telltale watermark. In contrast to fragile watermarks, where detection of the watermark fails as soon as the image is illegitimately modified, telltale watermarks are used to track the kind of modification a watermarked image has undergone. Our telltale watermark is able to accurately count how often an image has been JPEG compressed. This is accomplished by placing cyclic blocks of different cycle length in an image. By observing in which state these blocks are, it is possible to determine how often the image has been (re-)compressed up to a very large number. Depending on the appearance of the blocks, they can either be embedded into the content of the image or placed as control blocks in a dedicated area. Alternatively, it is possible to find already existing cyclic blocks in a given image and use them as a completely non-intrusive watermark.

Tracking JPEG compressions up to a very large number gives way for new applications. For example, one can create a hop count based on the number of JPEG compressions or a fingerprinting algorithm where each copy is JPEG compressed a different number of times. Furthermore, one could imagine a versioning application where an image is JPEG compressed every time a new version is committed or forwarded in social media platforms. Finally, the number of compressions is interesting to the field of image forensics, as it can indicate how often an image has been opened and saved with image editing software.

We show that the cyclic blocks can track the number of compressions very accurately for up to thousands of JPEG compressions. Further, we evaluate the proposed counter blocks regarding their robustness against spill-over effects, i.e. distortions that can occur during JPEG compressions with chrominance subsampling. We show that by properly padding the counter blocks, they can be successfully protected from spill-over. Finally, we evaluate how robust the watermark is against active tampering.

The remainder of this paper is organized as follows: Section 2 introduces the notation and describes different types of digital watermarks and their use case. Additionally, the phenomenon of block convergence is described. Section 3 describes the discovered peculiarity in block convergence and uses it to construct a telltale watermark that allows to count the number of JPEG compressions. Then, Section 4 evaluates the proposed watermark regarding its robustness against compression errors and tampering. Finally, Section 5 concludes with a summary of the results and an outlook on future research.

Related Work

Notation

In the following, matrices and vectors are denoted by boldface symbols. The inverse of a matrix $\boldsymbol{x}$ is denoted $\boldsymbol{x}^{-1}$ and its transposition $\boldsymbol{x}^T$. Throughout this paper, images are JPEG compressed and decompressed various times. To indicate the number of compressions, the superscript (t) is used to denote an object of the t-th JPEG compression and decompression cycle.

Digital Watermarks

Digital watermarks have been traditionally used to protect the authenticity of media data in order to enforce copyright and indicate the ownership of a digital multimedia signal. The main security property of such watermarks is their robustness against tampering or noise. Attempts to remove or distort a *robust watermark* should lead to a degradation of quality of the media data itself to the point where it is no longer usable and all its value is lost. This ensures that the watermark always remains present in the multimedia signal and the only way to remove it without knowledge of the secret key, is to destroy the signal altogether.

Additionally, watermarks can be used to protect the integrity of media data, i.e. to verify that the multimedia signal has not been modified or tampered with. This is typically achieved by using *fragile watermarks*, which are designed to be destroyed if the media data undergoes any form of modification. This ensures that the watermark can only be detected if the signal is complete and unmodified, whereas the probability that a tampered signal contains the watermark is negligible.

Fragile watermarks are useful to prevent any form of modification on the watermarked signal. However, it is often helpful to prevent only specific operations while allowing others. For example, the author of an image likely approves that the image is compressed in order to achieve a smaller file size. However, the author most likely wants to prevent that parts of the image are removed or added. Such a selective approach can be achieved by using *semi-fragile watermarks* that distinguish between legitimate and illegitimate distortions. Semi-fragile watermarks are designed to survive legitimate distortion while being destroyed by illegitimate. Therefore, only signals that have undergone legitimate transformations will contain the watermark.

Another type of watermark that shares a similar goal as semi-fragile watermarks but allows a more informed decision making is a *telltale watermark*. Its aim is to detect *how* an image was modified rather than *whether* it was modified [5]. Telltale watermarks utilize the fact that the watermark undergoes the same transformation as the host signal [1]. By analyzing the distortion applied to the watermark, it is possible to estimate which operation and distortion was applied to the signal and by this, to decide whether the distortion can be considered legitimate or not.

Despite its usefulness, very limited research has been done on telltale watermarks. This is mostly because it is difficult to design a watermark that allows to draw meaningful conclusions from the distortion. Additionally, there appears to be disagreement on how to classify telltale watermarks among the existing literature. While some see telltale watermarks as a separate category alongside (semi-)fragile and robust watermarks [5], it was originally introduced as a form of fragile watermark [11].

The general notion and first example of a telltale watermark has been introduced by Kundur and Hatzinakos [10, 11] in 1998. The authors propose to embed a watermark in the discrete wavelet domain by quantizing selected wavelet coefficients in different subbands. By analyzing in which subbands the watermark bits have been corrupted, it is possible to hypothesize which operation was performed. This is because different operations affect different subbands. For example, JPEG compression neglects information of higher frequencies and therefore corrupts the watermark bits in those frequency ranges [10]. On the other hand, if regions are replaced or changed, the lower frequencies will also differ. Other examples of telltale watermarks are localization watermarks that allow to identify the regions of the signal that have been corrupted [5, p. 410].

We believe that the best way to classify telltale watermarks is to see them as a sub-category or extension of semi-fragile watermarks. This seems intuitive since both types of watermarks are destroyed by certain distortions. The main difference is that semi-fragile watermarks merely survive legitimate distortions, i.e. are binary detectors. Telltale watermarks, on the other hand, further allow to derive information based on the condition of the modified watermark, i.e. are parametric detectors.

JPEG Block Convergence

Digital watermarks can generally be embedded in any digital signal. In this paper, we focus on the common case of digital images. More precisely we investigate images that are compressed with the popular image compression standard JPEG. JPEG specifies a series of lossy and lossless compression steps. In the simple case of grayscale images, the image is split into non-overlapping 8×8 blocks and each block is individually transformed into the frequency domain by using a Discrete Cosine Transformation (DCT). The resulting coefficients are then quantized and rounded. The decompression operation performs each step in reversed order and truncates the values to the range of $[0, 255]$. To formalize this transformation, we can follow the notation in [3, pp. 32 sq.] and assume that $\boldsymbol{x}_{\boxplus}^{(t)}$ is a matrix of serialized blocks after the t-th JPEG compression of an image. Each column contains the intensity values of one block in column-major order. With this, the compression and decompression operation can be expressed using the following recursion:

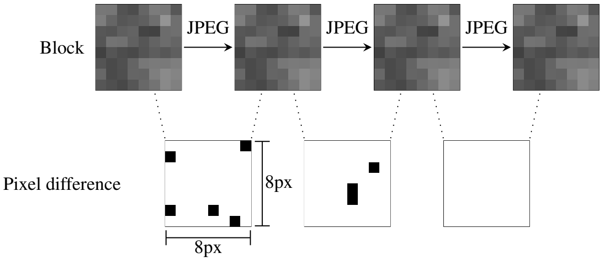


Figure 1. *Example of block convergence in greyscale images,* $q = 100$

$$\boldsymbol{x}_{\boxplus}^{(t+1)} = \mathrm{tr}\left(\left[\boldsymbol{D}^T\left(\boldsymbol{q}\left[\boldsymbol{q}^{-1}\left(\boldsymbol{D}\cdot\boldsymbol{x}_{\boxplus}^{(t)}\right)\right]\right)\right]\right), \qquad (1)$$

where $\boldsymbol{D}$ denotes the 2D-DCT transformation matrix, $\boldsymbol{q}$ the quantization matrix, $[\cdot]$ denotes rounding and $\mathrm{tr}(\cdot)$ truncates to the value range.

During repeated JPEG compression and decompression of images a phenomenon called *block convergence* can be observed. It describes that the intensity values of an image converge during repeated compression and decompression under the same settings. The analysis of block convergence has been first utilized by Lai and Böhme [12] in 2013 in order to estimate the number of times a grayscale image has been JPEG compressed with quality factor 100. The authors define a block as *stable* after t iterations if its values in $t+1$ equal its values in t. The ratio of stable blocks can then be used to estimate the number of compressions by comparing it to reference values. For this analysis, the authors excluded blocks that only contain a single value. These blocks are called *flat* and are stable from the beginning.

Formally, we can express the phenomenon of block convergence for a block $\boldsymbol{x}_i$, i.e. for the i-th column of $\boldsymbol{x}_{\boxplus}$ as:

$$\forall i, \exists t : \boldsymbol{x}_i^{(t)} = \boldsymbol{x}_i^{(t+1)}. \qquad (2)$$

The conjecture is that every grayscale JPEG block will eventually become stable during its compression and decompression lifecycle. Further compression and decompression will then no longer alter the block.

Figure 1 illustrates this concept using an 8×8 block from an image of the BossBase image database [2][1]. Since the marginal pixel differences between the blocks are difficult to see for the

[1] BossBase image “1.pgm”, rows 481–488, columns 41–48, stable after 2 compressions

human eye, we also report the pixel positions that change from one iteration to the next. While white denotes pixels that have not change, black denotes pixels that have. The figure shows that the first two JPEG compressions of the block lead to information loss and only a slightly different version of the block can be reconstructed. Therefore the block changes with each compression and decompression. However, with the third compression, the block is stable and the pixel values do not change anymore. Therefore the exact block can be reconstructed, despite JPEG compression. This also holds true for all subsequent compressions of this block.

Preliminary work of this paper extended the analysis of block convergence to color images where color conversion as well as chrominance sub- and upsampling influence the convergence [4]. Color conversion separates chrominance and luminance information by converting the color values from the *RGB* to the YC_bC_r model. This separation allows to value chrominance information differently than luminance information. Usually, color information is stored in a lower resolution since variations in color are less perceptible to the human eye. Extending the previously introduced notation, we assume that $\boldsymbol{x}_{\boxplus}^{(t)}$ is now a matrix of serialized JPEG blocks that successively contain the intensity values of the three YC_bC_r channels [3, pp. 32 sq.]. In other words, each column corresponds to one block and holds the intensity values, first of the Y channel, then of the C_b and C_r channel in column-major order. With this notation we can construct transformation matrices to formally express the JPEG compression for color images. The quantized DCT coefficients can be calculated, as:

$$\boldsymbol{y}_{\boxplus}^{(t+1)} = \left[\boldsymbol{q}^{-1}\left(\boldsymbol{D}\left(\boldsymbol{H}_{\text{sub}}\left(\boldsymbol{C}_{YC_bC_r}\boldsymbol{x}_{\boxplus}^{(t)}\right)\right)\right)\right]. \tag{3}$$

Here, $\boldsymbol{D}$ is used for the DCT transformation and $\boldsymbol{C}_{YC_bC_r}$ denotes the color conversion matrix. Additionally, $\boldsymbol{H}_{\text{sup}}$ denotes the linear filter to subsample the chrominance information and $\boldsymbol{q}$ denotes the quantization matrix.

By using the quantized DCT coefficients, we can express the decompression operation to reconstruct the pixel values:

$$\boldsymbol{x}_{\boxplus}^{(t+1)} = \text{tr}\left(\left[\boldsymbol{C}_{RGB}\left(\boldsymbol{H}_{\text{up}}\left[\boldsymbol{D}^{T}\left(\boldsymbol{q}\boldsymbol{y}_{\boxplus}^{(t+1)}\right)\right]\right)\right]\right), \tag{4}$$

where $\boldsymbol{C}_{RGB}$ transforms the image back to the *RGB* color space and $\boldsymbol{H}_{\text{up}}$ upsamples the chrominance information.

The extension to color has shown that the analysis of block convergence needs to be slightly adapted for the colored case [4]. First, convergence needs to be analyzed for all channels. We cannot consider a block converged until none of the intensity values of the red, green and blue channel change. Second, the analyzed block-size needs to be adapted to the subsampling rate. With chrominance subsampling, fewer values are stored for the chrominance channels. Chrominance blocks therefore represent a larger area. For example, with the most common subsampling rate 4:2:0, each chrominance block represents a 16 × 16 area. Hence, block convergence needs to be analyzed for macro-blocks, i.e. the area covered by a chrominance block. However, we can also observe that despite these adaptions, Equation (2) does not always hold for the colored case. In the following section, we evaluate this exception to block convergence and utilize it to create a new kind of telltale watermark.

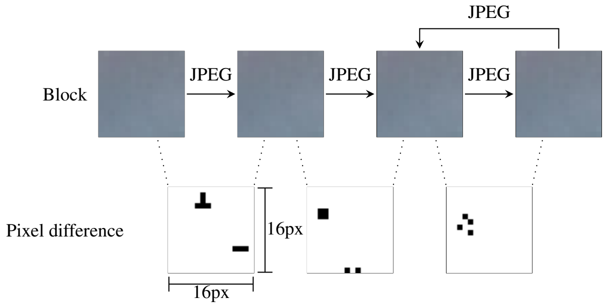
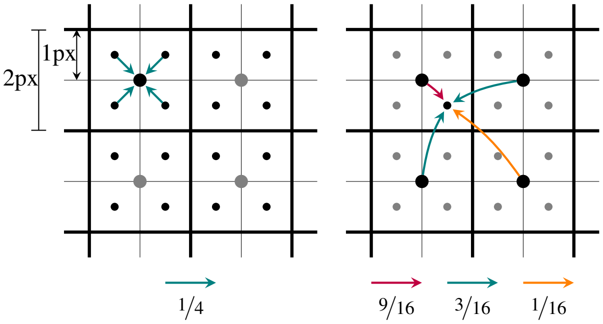


Figure 2. *Exemplary macro-block that does not converge but leaves two cyclic final states,* $q = 100$



(a) Simple subsampling

(b) Fancy upsampling

Figure 3. *Typical sub- and upsampling methods [4, 9]*

Contribution

Cyclic Blocks

Generally, block convergence also occurs in color images and the blocks for all color channels converge during repeated compression and decompression. However, when using chrominance subsampling it is possible that chrominance upsampling introduces an error into the decompression that interferes with this convergence path. If this error reverts a block to an earlier state, a cycle is created where the macro-block never becomes stable. Instead it cycles through a fixed number of states with each compression and decompression. This section evaluates when the upsampling error occurs and investigates how to use this peculiarity to create a unique watermark that counts the number of JPEG compressions.

An example of the convergence path for a cyclic 16 × 16 macro-block in an image of the RAISE image database [6] is shown in Figure 2[2]. Again, we also report the pixel positions that change from one block to the next. With the first two compressions the block converges as expected but after the second compression it ends in a cycle where it alternates between two different states and never converges.

The occurrence of cyclic blocks is highly dependent on the used sub- and upsampling algorithm. The above example was observed when using a combination of "simple subsampling" and "fancy upsampling" as available in the popular JPEG library `libjpeg` up to version `6b`. Figure 3 visualizes this sub- and upsampling procedure for the popular subsampling rate 4:2:0. "Simple subsampling" merely calculates and stores the average of a 2 × 2

[2] RAISE image "r000da54ft.TIF", rows 2385–2400, columns 433–448 cyclic after 8 compressions

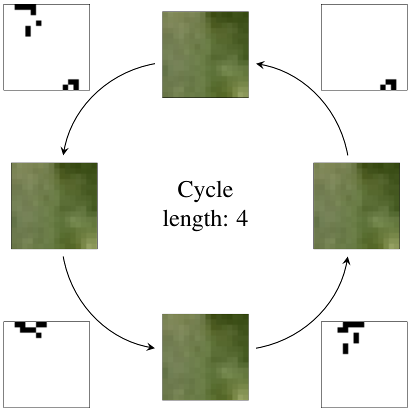


Figure 4. *Example of a counter block with cycle length four, $q = 100$*

pixel area as shown in Figure 3(a). To upsample the image, "fancy upsampling" employs a linear interpolation where the stored values are weighted by proximity. While the the nearest chrominance pixel is weighted with $^{9}/_{16}$, the two orthogonally adjacent pixels are weighted with $^{3}/_{16}$ and the diagonally adjacent pixel with $^{1}/_{16}$, as shown in Figure 3(b) [4, 9].

There exist several alternative approaches for sub- and upsampling, most notably simple upsampling and DCT scaling [7, 16]. Even though it is theoretically possible that the error introduced by other upsampling methods creates cyclic blocks, we have not observed any cyclic behavior when applying these methods. Additionally, there is a lot of controversy surrounding the usefulness of DCT scaling [14, 8], and the combination depicted in Figure 3 is by far the most popular approach to chrominance subsampling. The vast majority of popular JPEG libraries use this approach by default, e. g. `libjpeg 6b` [9], `libjpeg-turbo` [15] and `mozjpeg` [13].

While the previous example shows a block that alternates between two different states, much longer cycles are possible. This depends on the block and the error introduced by chrominance upsampling. The cycle length of a block describes the number of different states that the block traverses before reaching its initial state again. As an example, Figure 4 shows a cyclic block with cycle length four[3]. Again, the pixel difference between the blocks are reported. Note that each modified pixel is changed at least twice, such that the pixel returns to its original value and the block remains the same after a full cycle.

Counter Blocks

Since cyclic blocks constantly change based on the number of JPEG compressions, they can be used to construct a watermark that counts the number of JPEG compressions. By placing several blocks of different cycle length into an image, we can observe in which state the blocks are and use this information to accurately identify how often the image has been (re-)compressed up to a large number. Since these blocks allow us to count the number of compressions the image has undergone, we name this type of watermark "counter blocks".

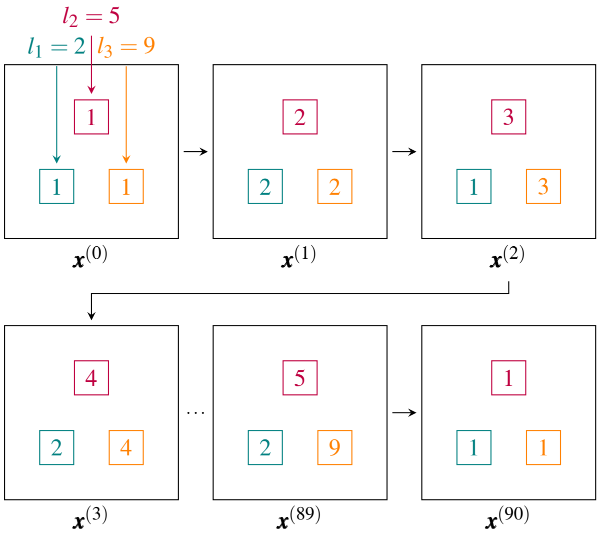


Figure 5. *States of counter blocks exactly identify the number of compressions due to different cycle lengths*

Figure 5 illustrates how different cycle lengths uniquely identify the number of compressions. As an example, three counter blocks with cycle length $l_1 = 2$, $l_2 = 5$ and $l_3 = 9$ and their respective states for every compression are shown. All blocks start in their initial state and cycle through their respective states with each compression. While blocks with shorter cycle length require fewer compression to complete a cycle, longer cycle lengths require more compressions. Because of this, the combination of states uniquely identifies the number of compressions. In this example, the blocks $\boldsymbol{x}^{(0)}$ and $\boldsymbol{x}^{(90)}$ are the first blocks that share the same combination of states. Therefore, these three blocks allow to identify up to 90 different compressions.

Generally, the number of compressions that can be tracked by counter blocks depends on the cycle lengths of the used blocks. With only a single counter block, the maximum number of detectable compressions is the cycle length of the block. With more counter blocks in an image, this number increases to the Least Common Multiple (LCM) of all cycle lengths. Therefore, the cycle lengths of the counter blocks should be chosen to have a large LCM.

Block Discovery

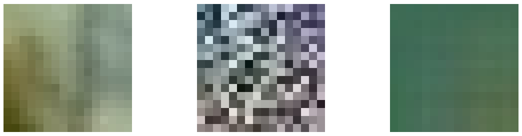
Several approaches can be used in order to find appropriate counter blocks. In general, the solution space for cyclic blocks is defined by a very complex system of equations. The system can be formed by using Equation (4) and solving for the values of $\boldsymbol{x}^{(0)}$ to meet the following condition:

$$\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(l+j)} \qquad \forall j \in \{0, \ldots, l-1\} \tag{5}$$

By choosing a cycle length l, a cyclic block of desired cycle length can be found.

Unfortunately, the formed system of equations can be very complex and difficult to solve. For this reason we also evaluate a more practical solution to find counter blocks, namely an exhaustive search. By evaluating the convergence behavior for a multitude of blocks, many counter blocks with different cycle

[3] RAISE image "r0176fbcat.TIF", rows 945–960, columns 385–400, cyclic after 5 compressions



(a) Image (b) Random (c) Smooth

Figure 6. Types of counter blocks

lengths can be identified. Generally, different sources can be used for such an exhaustive search. As an example, the blocks of natural images can be analyzed. An example of a cyclic block found in the RAISE image database [6] is shown in Figure 6(a)[4]. A problem with an exhaustive search over image data is that a lot of blocks are flat. Since flat blocks do not converge at all they do not qualify as counter blocks. Depending on the image content, these blocks can be very common which makes it difficult to find blocks with long cycle lengths.

As an alternative, it is also possible to analyze the convergence of randomly generated blocks. This is possible, since the convergence path of blocks has been reported to be widely independent of the image content [12]. For this reason, the distribution of cyclic blocks between image and random data is very similar. An example of a randomly generated counter block is shown in Figure 6(b). A problem with such blocks is the unnatural look which makes them more intrusive when embedded into an image

Lastly, we also introduce randomly generated blocks with less variance. For this we randomly choose a base color of the block. Then, we randomly permute the pixel values but require that the difference between neighboring pixels is at most one per channel. This creates a much smoother block that has a more natural look. This can even make it possible to embed the block directly into the image content. Additionally, such smooth blocks have a lot of favorable properties as the following sections will reveal. To avoid confusion, we will refer to these blocks as “smooth” in the remainder of this paper. An example of a smooth counter block is shown in Figure 6(c).

When constructing counter blocks, it has to be considered that the cyclic behavior of blocks dependens on the specific compression settings. This means that counter blocks are only cyclic under the settings that they have been constructed for. These settings include the choice of quality factor, (Fast-)DCT implementation, choice of subsampling rate and algorithm, among others. If any of the setting is changed, the blocks have a different convergence path and thus loose their cyclic behavior. Further, the choice of settings also influences the frequency of different cycle length. For example, we focus on the highest quality factor 100 throughout this paper. Cyclic blocks also exist for lower quality factors, however, typically have a shorter cycle length. This is because compressions with a lower quality factor tend to converge much faster.

To evaluate how frequent cyclic blocks occur, we perform an exhaustive search over $1,000,000$ 16×16 blocks for each block type. As image-data we use blocks of images taken from the RAISE image database [6]. Figure 7 shows the distribution of cycle lengths for all evaluated blocks on a logarithmic scale. As we can see from the dashed line, the distribution is almost log-linear. Here, a cycle length of one denotes stable blocks that do not cycle between different states. The results show that the vast majority of blocks become stable. However, roughly 7.5% of blocks show cyclic behavior, i.e. have a cycle length higher than one. This holds true for all three evaluated block types and confirms that the convergence path is mostly independent from the image content as discovered in [12]. About 66% of these cyclic blocks alternate between two different states, i.e. have a cycle length of two. We do observe much longer cycles, however, these cycles are considerable less common.

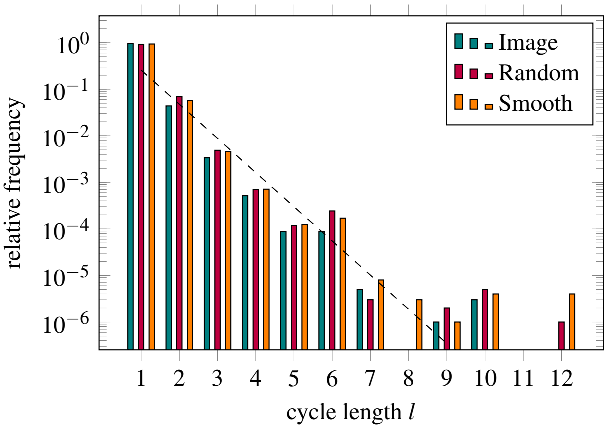


Figure 7. Distribution of cycle lengths for 16×16 blocks

The longest cycle in our experiment alternates between twelve different states. Counter blocks with such a long cycle are very useful for our watermark since they allow us to track a very high number of compressions. With all cyclic blocks we discovered, it is possible to uniquely identify up to 2520 JPEG compressions. With a more elaborate construction and search of cyclic blocks even much higher numbers might become possible.

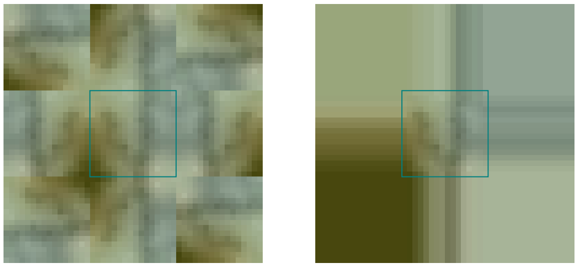
Padding

Up until now, we observed the convergence behavior for isolated blocks without the influence of surrounding blocks. Since our aim is to embed blocks as a watermark into images, we need to investigate which influence surroundings can have on the cyclic behavior of counter blocks. In JPEG, the influence between neighboring blocks is typically very limited since each block undergoes a separate transformation as highlighted by Equations (3) and (4). This means that the compression and decompression of blocks is mostly independent. However, when using “fancy upsampling” the decompression is no longer confined within a block. This can be intuitively visualized by assuming that the center line in Figure 3(b) is a block boundary that separates two macro-blocks. In this case the outermost values of the blocks are interpolated using information from both blocks.

The influence that blocks have on their neighbors is called “spill-over” [4] since changes to a block can spill into their neighboring blocks, causing them to change. These effects are problematic when embedding counter blocks, since spill-overs can change the counter blocks and cause them to alter or lose their cyclic behavior. This renders the block useless for our watermark, since the number of compressions can no longer be identified.

To protect counter blocks from spill-over, it is necessary

[4] RAISE image “r002fc3e2t.TIF”, rows 433–448, columns 257–272, cyclic after 15 compressions



(a) Mirrored blocks (b) Repeat border values

Figure 8. *Padding strategies (illustrated using the block from Figure 6(a))*

to reduce the error introduced by chrominance subsampling at the block boundaries. This can be achieved by surrounding the block with content that is similar at the block boundaries. An intuitive choice is to protect the block using the mirrored block itself, such that the values at the block boundaries perfectly match. An example of this padding strategy is shown in Figure 8(a), using the block from Figure 6(a) to illustrate the concept. The marked block in the middle is the original block which is surrounded by mirrored versions of itself. Even though this can reduce the error at the block boundaries, it comes with several drawbacks. On the one hand, some of the mirrored blocks will also show cyclic behavior since the DCT-transformation is invariant to certain rotations. This creates varying spill-over on the block since some padding blocks cycle while others do not. These varying conditions are difficult to manage and increase the probability of undesired spill-over. On the other hand, cyclic padding blocks themselves are particularly susceptible to spill-over effects and would need to be protected as well. This strategy therefore shifts the initial problem to the padding blocks instead of solving the problem.

An alternative padding strategy is to repeat the outermost value of the row or column for an entire macro-block. This strategy creates a macor-block where the entire row or column consists of the value at the block boundary. Therefore, this strategy also matches the pixel values at the block boundaries but creates a more robust block that is much less susceptible to spill-over or other compression errors. The repeated color values make the block much easier to compress, which reduces the error rate. Figure 8(b) illustrates this padding strategy. Again, the original block is marked in the middle and the block is surrounded by one macro-block in each direction.

Note that the counter block can also be padded with more than a single macro-block in each direction. The more padding blocks are used, the longer it takes for spill-overs to affect the counter block since the spill-over has to propagate through several layers of padding blocks. Nevertheless, this only delays the problem of spill-over. For this reason, it is desirable to create conditions where a single layer of blocks is able to protect the counter block.

Embedding

Once the block is adequately protected from spill-over, it can be embedded into an image as a watermark. To do so, a dedicated image area can be chosen where the selected counter blocks are placed. As an example, the bottom of the image can be reserved for the watermark. In our case, we pad the bottom of the image with 80 pixels of a neutral gray background. This gives enough room to place the padded 48 × 48 blocks and leaves one macro-block of space to the image content. Note that this is not required for the watermark, but solely for visual purposes. Then, we place several counter blocks of different cycle length centered and evenly spread on the gray area. Since our watermark utilizes the blocking structure of JPEG, each counter block needs to be aligned to the grid-structure. This ensures that each counter block forms a new macro-block and retains its cyclic behavior.

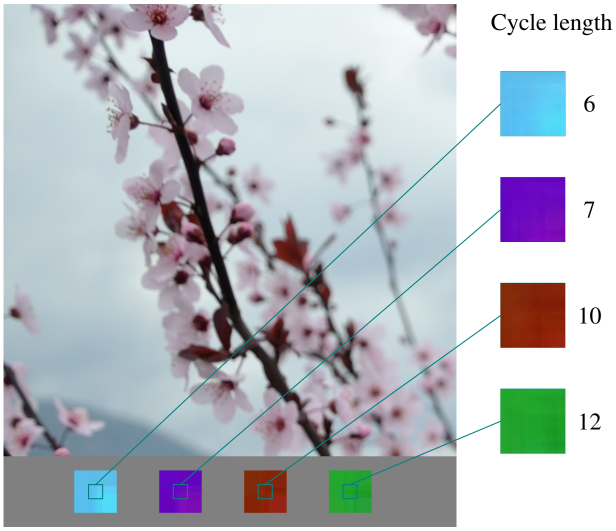


Figure 9. *Image marked with four padded counter blocks*

Figure 9 exemplarily shows an image that has been marked with the proposed watermark. We chose four different counter blocks of length $l_1 = 6$, $l_2 = 7$, $l_3 = 10$ and $l_4 = 12$. All blocks were found with an exhaustive search, using the "smooth" block type. The blocks are padded by repeating-the border value as introduced in the previous section. The counter blocks and their respective cycle lengths are shown on the right.

When the image is JPEG compressed and decompressed, the placed counter blocks will cycle through their respective states. By observing in which state each counter block is, the exact number of compressions can be identified. In the example above, the chosen counter blocks allow to uniquely identify up to 420 consecutive compressions.

One drawback of this embedding strategy is that the watermark is visible and influences the appearance of the image. With subsampling rate 4:2:0, one counter block requires at least 16 × 16 pixels. In order to track the number of compressions up to a high number, several blocks are required. Further, to protect the block from spill over, the block should be padded adequately, which requires at least 48 × 48 pixels per counter block. To avoid this, we can also search for already existing cyclic blocks in a given image. This allows to create a completely non-intrusive and invisible watermark. The amount of cyclic blocks in a given image depends on the number compressions that the image has undergone. Typically the first cyclic blocks appear after several compressions. For an uncompressed image barely any blocks show cyclic behavior. In these scenarios we need a "warm up" period of several compressions in order to find possible counter blocks.

Figure 10 shows an example of a non-intrusive watermark. Exemplarily, four cyclic blocks with a cycle length of up to three

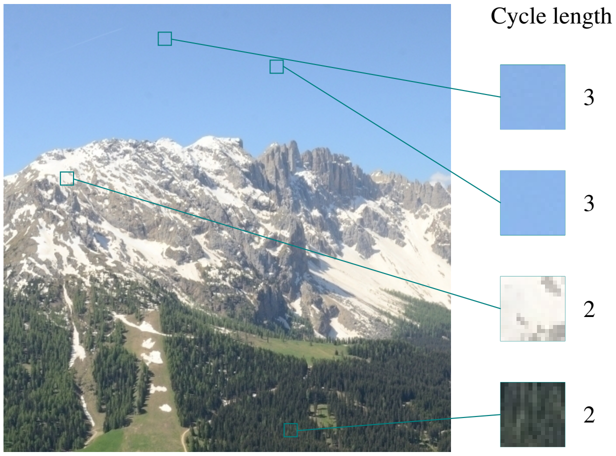


Figure 10. *Example of non-intrusive counter blocks*

are marked. Again, the counter blocks and their respective cycle lengths are shown on the right. However, due to the shorter cycle lengths, the chosen counter blocks only allow to track up to six compressions. Generally, it is also possible to find cyclic blocks with longer cycles. However, given the rarity of long cycle lengths (cf. Figure 7), this approach is mostly limited to track fewer compressions or to very large images.

Given the rarity of cyclic blocks, it is interesting to evaluate how large an image needs to be, such that chosen cycle lengths occur with sufficiently high probability. In other word, we want to state the probability of cyclic blocks as a function of the image size. To do so, we utilize the empirical distribution from Figure 7 and denote the probability of a cycle length l as p_l. Given n different blocks, the probability that none of the blocks will have a cycle length of exactly l can be expressed as $(1-p_l)^n$. Consequently, we can calculate the probability P of the complementary event, i.e. that at least one block will have cycle length of exactly l:

$$P = 1 - (1 - p_l)^n \tag{6}$$

By solving the equation for n, we can calculate how many blocks are required, such that the occurrence of a cycle length l is at least P:

$$n \geq \log_{1-p_l} 1 - P. \tag{7}$$

For example, if we want to observe a counter block of cycle length exactly 2 with at least 90% probability, we can calculate the required number of blocks as:

$$n \geq \log_{1-0.056} 1 - 0.90 = 39.96, \tag{8}$$

where $p_2 = 0.056$ as shown in Figure 7. This means we require at least $40 \cdot 16 \cdot 16 = 10240$ pixels in the image for the probability of a cyclic block to be sufficiently high.

An additional benefit of the non-intrusive watermark is that the cyclic blocks tend to be well padded, for several reasons. First of all, blocks in natural images are often surrounded by similar content. In the example above, the image features a large blue sky where pixel values are very similar. This reduces the influence of spill-over effects. Secondly, when analyzing the image for cyclic blocks only already well padded blocks will show cyclic behavior. Therefore, this watermark does not require any further padding.

Successful protection from spill-over for one cycle

	Block type		
Padding	Image	Random	Smooth
None	00.00%	00.00%	00.00%
Mirrored blocks	65.80%	00.60%	04.00%
Repeat border values	93.20%	28.40%	53.40%

Evaluation

This section validates the proposed telltale watermark regarding its robustness. For our experiments we utilize the very popular JPEG library `libjpeg 6b` [9] with quality factor 100 and the default settings, i.e. "slow" DCT algorithm, "simple subsampling" with subsampling rate 4:2:0 and "fancy upsampling". All experiments are performed using 8156 never-compressed color images obtained from the RAISE image database [6].

Robustness Against Spill-Over

First, we evaluate whether the presented padding strategies can protect the counter blocks from spill-over effects. For this we compare all block types (cf. Figure 6) combined with all padding strategies (cf. Figure 8). We embed the padded counter blocks into a randomly selected image in a dedicated gray area with at least one macro-block between the padded watermark and the the image content (cf. Figure 9). Then, we JPEG compress the entire image and observe whether the convergence of the counter block differs in comparison to its isolated convergence path. Note that in this setup the choice of image has a rather low influence on the watermark. Since we are embedding the watermark in a dedicated area with a gray background, most of the spill-overs on the block come from those neighboring blocks. Nevertheless, since we are possibly tracking hundreds of JPEG compressions, the spill-over created by the image content is able to propagate through the entire image and may also affect the watermark.

Table 1 shows the percentage of images that were successfully protected from spill-overs for an entire cycle length. The results show that blocks without padding are always affected by spill-overs. Not one of the tested blocks remained the same after a full cycle, regardless of block-type. This can be easily explained, since the watermark is surrounded by a vastly different area. In our setup, we choose a dedicated area with a neutral gray color to embed our watermark. This increases the subsampling error and causes the outermost values of the block to change after one compression and decompression cycle. With the next compression the introduced change propagates through the entire block due to the DCT and IDCT transformation. This result highlights that the right padding strategy is essential when embedding counter blocks.

Note that protection for one cycle does not prevent that blocks are affected by spill-overs during subsequent compressions. Spill-overs are propagating very slowly through an image and even the slightest change in one block can cause a chain reaction that can slowly affect the entire image. Nevertheless, changes due to spill-over are most probable after the very first compressions and changes after an entire cycle length are less common. Therefore, the protection during the first cycle gives a good indicator on how well the block is protected.

When using mirrored blocks to pad the watermark, the results are mixed. With blocks taken from images, the strategy is able to protect the majority of blocks. However, when using randomly generated blocks, the protection rate is much lower and almost no blocks can be protected. The major problem of mirrored blocks is their own high variance and weak stability. This makes them prone to compressions and subsampling errors. Depending on the rotation, the padding blocks can even be cyclic themselves which leads to changing spill-overs on the watermark which are harder to manage.

By far the most successful padding strategy in our experiment is repeating the border values of the block. With blocks from image data, almost all blocks could be successfully protected from spill-over. On the one hand, this strategy replicates the outermost values, which reduces the error through interpolation at the block-boundaries. On the other hand, this strategy reduces the variance of the padding blocks by often repeating the same color value. Either an entire row or an entire column consists of the same color. This reduces the variance within the padding blocks which makes them less susceptible to compression errors. Unfortunately, for randomly generated blocks the success rate of this padding strategy is much lower due to their high variance. To some degree, this can be mitigated by using smooth blocks, where a base color is chosen randomly but the variance remains low. Again, this makes the padding blocks more robust against compression errors and thus improves the protection rate.

Overall, repeating the border values has shown the best performance. Nevertheless, the creator of the watermark can evaluate how robust the watermark is, beforehand. Therefore, counter blocks and padding can always be chosen accordingly to prevent spill-over.

Robustness Against Tampering

Next, we evaluate the robustness of counter blocks against active tampering. In line with our previous definition of a telltale watermark as a subcategory of semi-fragile watermarks, counter blocks are destroyed by illegitimate distortions. So, if an active attacker changes the compression setting or replaces pixel values in order to influence the convergence behavior or states of the embedded counter blocks, they indicate this kind of illegitimate modification. For example, the attacker can slightly adjust the luminance of the image or JPEG compress it with different settings. In both cases the convergence path for all blocks is set back and the cyclic behavior of counter blocks is lost.

If the counter blocks are known to the attacker, e. g. because they are embedded in a dedicated area (cf. Figure 9), the attacker may remove or distort the counter blocks. Again, the analyst detects that the counter blocks have been transferred into a non-valid state, but it remains unclear which operations were performed.

Since the attacker knows the initial state of (her copy of) the counter blocks she might replace the counter blocks with an earlier or later state in order to pretend that the image has been compressed a different number of times. In this scenario the analyst might remain oblivious that the image has been tampered with. To prevent an attacker from manipulating the counter blocks we can hide them, e. g. by using our non-intrusive watermark (cf. Figure 10) or by embedding a block in similar image content. In this case, the attacker cannot easily forge the state of the counter blocks, because it is not easy to locate the blocks. Additionally, if the counter blocks are spread evenly within the image, it prevents the attacker from local tampering, as she is oblivious if she destroys one of the counter blocks.

Lastly, the attacker can also estimate the counter blocks, e. g., by performing an exhaustive search over the whole image in order to find cyclic blocks. Again, this might allow to manipulate the states of the counter blocks. Additionally, by calculating the LCM of all counter blocks, the attacker can fool the analyst by compressing the image more times than are detectable. With counter blocks alone, this kind of attack cannot be prevented.

Overall, counter blocks are not robust against an active attack, thus fulfilling the fragile demand of a telltale watermark. By hiding the counter blocks, it is possible to prevent that an attacker can forge a different number of compressions but it remains difficult to prevent that the watermark is removed from the image.

Conclusion

This work proposes a new kind of telltale watermark that allows to track the number of JPEG compressions an image has undergone. Telltale watermarks are a rare type of digital watermarks, typically seen as a sub-category of semi-fragile watermarks. While semi-fragile watermarks aim to detect *whether* a digital signal has been modified, telltale watermarks aim to detect *how* it was modified [5]. They utilize the fact that the watermark undergoes the same distortion as the image and infer the type of modification from the distortion of the watermark. Our proposed watermark changes based on the number of JPEG compressions and thus allows to accurately identify the number of compressions the image has undergone.

The JPEG compression standard divides an image into several blocks to improve the performance of the algorithm. During repeated JPEG compression and decompression these block usually converge, a phenomenon known as block convergence. Additionally, JPEG usually reduces the color information in color images since the human eye is less sensitive to information loss in color. However, when using a linear interpolation to upsample the chrominance information, a peculiarity in block convergence can be observed. It is possible that blocks never converge but instead cycle through a fixed number of different states. The cycle length of cyclic block can vary depending on the block which we utilize to count the number of compressions. By placing several of these "counter blocks" with different cycle lengths in an image, we can observe the states of the counter blocks and use this information to uniquely identify the number of compressions up to a very large number. In our experiments, we were able to track more than two thousand consecutive JPEG compressions of an image. Since our approach utilizes the JPEG blocking structure, the watermark needs to be aligned to the block-grid of the image. When the image is illegitimately modified, e. g. by using a different compression or by applying a global filter, the watermark is fragile and destroyed.

The proposed watermark can be useful in image forensics where the compression history of an image is an important indicator to identify forgeries. By marking an image with the proposed watermark, the number of (re-)compressions can be accurately tracked. Due to its semi-fragile nature, the watermark is destroyed by tampering and thus indicates illegal modifications. Additionally, counter blocks give rise to new application scenarios, due to the very high number of compressions that can be tracked. For example, the watermark can be used as a hop-counter or to count how often an image has been forwarded in a social network. In these scenarios, each compression counts an intermediate step of the image. Lastly, a fingerprinting algorithm can be constructed, where each copy of an image has been compressed a different number of times. With gentle assumptions about colluders, this allows to attribute each copy to the individual.

Counter blocks can be found by solving a complex system of equations or through an exhaustive search over image or random data. To protect the cyclic behavior of counter blocks when embedded into an image, they need to be shielded from the influence of surrounding blocks. This is because the chrominance interpolation is not confined within a macro-block and thus interpolation errors can occur at the block boundaries. To protect blocks from these errors, they need to be surrounded with similar content at the block boundaries. By far the best protection we found was to repeat the outermost value of the row or column for an entire macro-block. With this strategy the majority of blocks kept their cyclic behavior even after embedding.

Even though the cycle length of randomly generated blocks follows a similar distribution, random blocks are much more difficult to protect from spill-over. This is due to the high variance between pixel values which increases the error introduced by chrominance subsampling. To mitigate this, it is better to choose a random base color and randomly permute some of the pixel values to reduce the variance in the randomly generated blocks.

One drawback of the presented watermark is that it is a visible watermark that needs to be embedded somewhere in the image. In our experiments we usually chose a dedicated gray area at the bottom of the image. However, to make it less visible it can also be embedded into similar looking image content. Alternatively, it is also possible to look for already existing cyclic blocks in a given image. Even though this allows to create an invisible watermark it is typically limited to detect fewer compressions due to the rarity of longer cycle lengths.

Overall the existence of counter blocks is a nice phenomenon but limited to very constrained application scenarios. The most crucial limitation is the high sensitivity to the exact JPEG compressions settings as well as the dependence on high quality compression. Even though cyclic blocks exist with lower quality factors, they are considerable less common and typically of shorter cycle length. As an example, less than 1% of blocks show cyclic behavior with quality factor 99 and most cycles have a cycle length shorter than five.

Finally, future research should focus on cyclic blocks that are less sensitive to specific compression settings. The proposed counter blocks can be removed from an image by other compressions settings such as a different quality factor or different subsampling algorithms. Counter blocks that are less sensitive to these settings can be easier applied in practice. Additionally, it is likely that counter blocks exist in other lossy compression standards that do block-wise wise transformation followed by rounding to integers.

Acknowledgments

Part of this research and the associated conference presentation have been supported by Archimedes Privatstiftung, Innsbruck.

References

[1] Bassem Abdel-Aziz and Jean-Yves Chouinard. “On Perceptual Quality of Watermarked Images – An Experimental Approach”. *Digital Watermarking*. Ed. by Ton Kalker, Ingemar Cox, and YongMan Ro. Vol. 2939. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 277–288.

[2] Patrick Bas, Tomáš Filler, and Tomáš Pevný, Break our steganographic system — the ins and outs of organizing BOSS. Information Hiding (13th International Workshop). Ed. by Tomáš Filler, Tomáš Pevný, Scott Craver, and Andrew Ker. Vol. 6958. Lecture Notes in Computer Science. Berlin Heidelberg: Springer-Verlag, pp. 59–70. (2011).

[3] Rainer Böhme, Advanced Statistical Steganalysis. 1st. Springer-Verlag, Berlin Heidelberg, 2010.

[4] Matthias Carnein, Pascal Schöttle, and Rainer Böhme, Forensics of High-Quality JPEG Images with Color Subsampling. Proceedings of the 7^{th} IEEE International Workshop on Information Forensics and Security (WIFS). Rome, Italy. (2015).

[5] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker, Digital Watermarking and Steganography. 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[6] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato, RAISE: A Raw Images Dataset for Digital Image Forensics. Proceedings of the 6th ACM Multimedia Systems Conference (MMSys ’15). Portland, Oregon: ACM, pp. 219–224. (2015). URL: `http://mmlab.science.unitn.it/RAISE/` (visited on 01/09/2016).

[7] Rakesh Dugad and Narendra Ahuja, A fast scheme for image size change in the compressed domain. IEEE Transactions on Circuits and Systems for Video Technology 11, 4, pp. 461–474. (2001).

[8] Hardwarebug. IJG swings again, and misses. 2010. URL: `http://hardwarebug.org/2010/02/01/ijg-swings-again-and-misses/` (visited on 01/09/2016).

[9] Independent JPEG Group. libjpeg. URL: `http://www.ijg.org/` (visited on 01/09/2016).

[10] Deepa Kundur and Dimitrios Hatzinakos, Digital watermarking for telltale tamper proofing and authentication. Proceedings of the IEEE 87, 7, pp. 1167–1180. (1999).

[11] Deepa Kundur and Dimitrios Hatzinakos, Towards a telltale watermarking technique for tamper-proofing. International Conference on Image Processing (ICIP 98). Vol. 2, pp. 409–413. (1998).

[12] ShiYue Lai and Rainer Böhme, Block convergence in repeated transform coding: JPEG-100 forensics, carbon dating, and tamper detection. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3028–3032. (2013).

[13] Mozilla. Mozilla JPEG Encoder Project. URL: `https://github.com/mozilla/mozjpeg` (visited on 01/09/2016).

[14] The libjpeg-turbo Project. A Study on the Usefulness of DCT Scaling and SmartScale. URL: `http://www.libjpeg-turbo.org/About/SmartScale` (visited on 01/09/2016).

[15] The libjpeg-turbo Project. libjpeg-turbo. URL: `http://libjpeg-turbo.virtualgl.org/` (visited on 01/09/2016).

[16] Carlos L. Salazar and Trac D. Tran, On resizing images in the DCT domain. International Conference on Image Processing (ICIP ’04). Vol. 4, pp. 2797–2800. (2004).

Author Biography

Matthias Carnein is a member of the IT Security Research Group at the University of Münster, Germany. He received his BSc and MSc degrees in Information Systems from the University of Münster, Germany in 2013 and 2015, respectively. His research focuses on image forensics, steganography and machine learning.

Pascal Schöttle is a member of the Security and Privacy Lab at Uni-

versität Innsbruck, Austria. He received his MSc degree in IT Security from Ruhr-University Bochum and his Ph.D. degree in computer science from the University of Münster, Germany. His research interests focus on multimedia security and steganography in particular, and include asymmetric cryptography and network anomaly detection.

Rainer Böhme is Professor of Security and Privacy at the Institute of Computer Science, Universität Innsbruck, Austria. A common thread in his scientific work is the interdisciplinary approach to solving exigent problems in information security and privacy, specifically concerning cyber risk, digital forensics, cyber crime, and crypto finance. Prior affiliations in his academic career include TU Dresden and Westfälische Wilhelms-Universität Münster (both in Germany) as well as the International Computer Science Institute in Berkeley, California.

Appendix

In order to make our results reproducible, we have reported the block positions of all natural image blocks used throughout this paper. In addition, Table 2 reports the pixel values of the counter block with cycle length 12 in Figure 9. The values are reported as hexadecimals for each channel. We have observed the cyclic behavior by using the JPEG library `libjpeg 6b` [9], compiled on a Windows system with an Intel Core i7-3770 processor using MinGW and the GCC compiler. The block has been compressed using the highest quality factor 100 and the default compression settings, i.e. subsampling rate 4:2:0, simple subsampling, fancy upsampling as well as the "slow" DCT implementation.

Pixel values of counter block with cycle length 12

22	21	21	22	23	22	22	22	22	22	22	22	24	23	23	22
21	21	21	21	22	21	22	20	1f	1f	1f	1e	21	1f	20	1f
24	21	23	23	25	21	21	1f	1d	1c	1c	1b	1e	1d	1f	1d
25	22	22	25	25	25	23	21	21	1f	1f	1d	1f	1e	21	20
24	21	22	23	23	24	22	20	21	21	20	1f	1f	1e	21	23
23	20	1f	22	22	23	23	22	22	22	22	21	22	21	23	25
22	1e	1c	1d	1e	1e	20	20	1e	1f	1e	1f	1e	1c	1f	1f
22	20	1e	1f	1e	1d	1e	1d	1c	1d	1b	1b	1c	19	1c	1c
24	22	21	22	20	20	23	1f	1d	1e	1d	1c	1c	1b	1d	1d
24	22	21	22	21	20	21	1f	1d	1f	1d	1a	1c	19	1a	1b
22	20	21	22	1f	1d	1f	1e	1d	1c	1b	18	1a	17	1b	1c
24	22	21	23	20	1d	1f	1f	1d	1e	1d	19	1a	17	1a	1c
24	25	22	25	23	1e	1f	1f	1e	20	20	1b	1c	1a	1c	1e
24	23	24	25	23	1c	1f	1e	1d	1e	1b	18	15	16	18	19
28	26	25	27	24	20	23	21	20	20	1e	1b	19	19	1b	1c
28	26	25	28	29	22	26	24	23	24	23	1f	20	1d	22	25

(a) Red

a8	a7	a5	a6	a7	a6	a8	a8	a8	a8	a6	a6	a5	a4	a4	a2
a5	a5	a5	a5	a6	a5	a6	a6	a5	a5	a3	a2	a2	a0	a1	9f
a6	a3	a4	a5	a7	a5	a5	a3	a3	a2	a0	9f	9f	9e	9e	9c
a6	a3	a3	a6	a7	a7	a7	a5	a7	a5	a3	a1	a0	9f	a0	9f
a5	a2	a3	a4	a5	a6	a6	a4	a7	a7	a4	a3	a0	9f	a1	a0
a4	a1	a0	a3	a3	a4	a7	a6	a8	a8	a6	a5	a3	a1	a3	a2
a6	a2	a0	a1	a2	a2	a4	a4	a4	a5	a2	a0	9e	9c	9f	9e
a6	a4	a2	a3	a2	a1	a2	a1	a2	a3	9f	9c	9c	99	9c	9b
a5	a3	a2	a3	a1	a1	a4	a3	a3	a4	a1	9d	9c	9b	9d	9e
a5	a3	a2	a3	a2	a1	a2	a3	a3	a5	a1	9e	9c	99	9b	9c
a6	a4	a2	a3	a3	a1	a3	a3	a3	a3	a0	9c	99	96	9c	9d
a5	a3	a2	a4	a4	a1	a4	a4	a4	a5	a2	9c	9b	98	9b	9d
a4	a5	a1	a4	a4	a1	a4	a4	a5	a7	a5	a0	9f	9a	9c	9f
a4	a3	a1	a1	a4	9f	a2	a3	a3	a4	9f	9c	98	96	98	99
a5	a3	a0	a3	a3	a0	a3	a4	a4	a6	a2	9f	9b	99	99	9a
a5	a3	a0	a4	a5	a1	a6	a7	a7	aa	a7	a3	a0	9d	9d	a0

(b) Green

2d	2c	2b	2c	2f	2c	2d	2b	2f	2f	2e	2e	2e	2e	30	31
2b	2b	2b	2b	2c	2b	2c	2b	2a	2c	2b	2a	2b	2a	2d	2e
2c	29	2d	2b	2d	2b	2b	29	28	29	28	28	29	28	2b	29
2f	2c	2c	2f	2d	2b	2d	2b	2c	2c	2b	2a	2a	29	2b	2a
2f	2c	2d	2d	2b	2c	2e	2c	2e	2e	2c	2b	29	28	2a	2a
2e	2b	2a	2d	2c	2d	2f	2e	2f	2f	2e	2d	2c	2a	2c	2c
2f	2b	29	2a	2b	2b	2d	2d	2b	2c	2b	29	27	25	28	29
2f	2d	2b	2c	2b	2a	2b	2a	29	2a	28	25	25	22	25	26
2f	2d	2e	2f	2b	2b	2d	2b	2a	2b	29	26	25	24	26	27
2f	2d	2e	2f	2c	2b	2b	2b	2a	2c	29	26	25	22	24	25
2e	2d	2e	2f	2c	2a	2c	2c	2a	2a	29	25	24	21	26	27
2e	2c	2c	30	2d	2a	2d	2d	2c	2d	2d	28	27	24	27	27
2b	2c	2c	31	30	2d	2f	2f	2f	31	32	2d	2d	29	2b	2b
29	2a	2c	2f	30	2b	2e	2e	30	31	2f	2c	28	27	29	2a
2d	2d	2e	33	32	2f	32	32	34	35	34	31	2e	2c	2e	2f
2d	2d	2e	34	35	30	37	37	37	39	3a	36	33	30	35	38

(c) Blue