Securing Scalable Visual Data in a Grid Environment

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Ingenieur
an der Naturwissenschaftlichen Fakultät
der Universität Salzburg

eingereicht von
Thomas Stütz

Betreuer
Ao. Univ.-Prof. Mag. Dr. Andreas Uhl

Salzburg, 9. Mai 2006

Contents

Chapter 1

Introduction

In bunten Bildern wenig Klarheit,
Viel Irrtum und ein Fünkchen Wahrheit,
So wird der beste Trank gebraut,
Der alle Welt erquickt und auferbaut.
Lustige Person
Goethe, **"Faust - Der Tragödie erster Teil"** [18]

This work was funded within the **Austrian Grid**, a joint project of many Austrian universities and institutions. The goal of the Austrian Grid is to start and support grid computing in Austria. "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" is a definition given by I.Forster, one of the pioneers of grid computing, in [17]. The grid can be classified as a middle-ware abstraction layer to ordinary networks. Several different grid middle-ware implementations are currently available, the most common and therefore chosen for the Austrian Grid is the **Globus Toolkit**.

1.1 Motivation

Within the Globus Toolkit there are no special extensions for secure transmission of visual data. The special treatment of visual data for secure transmission has certain advantages. The term **selective encryption** expresses that the special properties of visual data are exploited and preserved. Often the amount of encryption can be adjusted. Hence the complexity of the encryption process can possibly be reduced. The reduced amount of data which is encrypted could even make the employment of asymmetrical ciphers feasible. The usage of asymmetrical ciphers greatly simplifies the key management and the protocol layout. Unfortunately asymmetrical ciphers are generally much to slow for streaming visual data.
In heterogeneous networks, on which a grid generally is built upon, the bandwidth differs. Furthermore not only the bandwidth may differ, but the capabilities of

the **grid nodes** (the devices participating in the grid). In fact this problem will become bigger, when the usage of mobile devices and grid technology will grow. Since mobile devices' (e.g. cell phones') computing capabilities increase at high velocity, the employment of grid technology will not be unfeasible for long. Hence visual data is needed to be streamed at various bandwidths, qualities and resolutions. Non-scalable formats have to compress a different stream for every node (with nearly the same complexity), while scalable formats already contain several qualities within one stream and can very easily be converted. When using full encryption for visual data, it has to be decrypted, transcoded and again encrypted. There are many problems within the above described procedure. If the rate adaption has to be conducted at an arbitrary point (grid technology aims at highly dynamic usage) in the secure transmission, the encryption key has to be present here. This leads to enormous issues in the key distribution and accompanying protocols. Furthermore the encoding and decoding at different quality settings is timely and computationally extremely expensive, resulting in enormous delays.
How can selective encryption of scalable visual data simplify this process? Scalability of visual data means that several qualities of it are present within a single file or more generally data stream. When encrypting selectively a goal is to preserve the scalability of those data streams. A rate adaption can then be simply realized by dropping the additional high qualities of the data stream. No decrypting, no transcoding and no encrypting is needed any longer.

1.1.1 Application Scenarios

Applications using selective encryption may have different needs apart from the described advantages. These can be divided into those which have to confidentially hide all image information and those which even want to expose certain image information. The second approach is called **transparent encryption**. An application scenario is e.g. pay tv, where a low quality version should become public to seduce customers. Both confidential and transparent encryption will be discussed.

1.2 Standards for Scalable Visual Content

Several standards for visual data exist, which define compression and structure. The usage of standardized formats is favored for many reasons. Interoperability is guaranteed, the software is publicly available, legal affairs are mostly already sorted out and most visual content is already encoded in the most popular formats. This paper will focus on image data, since most of the results can be extrapolated to video data. Currently one of the most or even the most popular image standard is **JPEG**, which is therefore discussed in detail. Its successor, the **JPEG2000** standard is considered to continuously take its place. While the

JPEG standard additionally defines scalable modes, these are naturally part of the JPEG2000 compression concept.

While the major part of this work is dedicated to the analysis of selective encryption of JPEG and JPEG2000, the applicability of the proposed methods in a grid environment is evaluated in the last chapter. Furthermore a secure visual pipeline that applys selective encryption and is based on the grid tool **glogin** is described.

Chapter 2

JPEG: Modes of Operation

2.1 Baseline JPEG

The JPEG standard for the coding of digital images was designed by the Joint Photographic Experts Group and is based on a collaborative effort between **ITU** (International Telecommunication Union) and the **ISO** (International Standard Organization). Hence all the details can be found in the last recommendation of the ITU for the JPEG standard ([6]), which is freely available at `www.w3c.org`. Furthermore the **IJG** Independent JPEG Group distributes a widely used free library (**libjpeg**) for JPEG image compression, which can be seen as the **reference library**. JPEG is still the most common image compression system in the world, with a wide range of application. Therefore, dealing with this standard is - despite its age - still worth-while. Contrary to most beliefs, the JPEG Standard is not just defining one image compression method, but is a set of diverse image compression standards, of which Baseline Sequential JPEG is just one out of many. The standard covers progressive modes, extensions to the Baseline System as well as a lossless mode of operation. The lossless mode will no be dealt with, because it never became widely accepted (it is even quite hard to find a working implementation). In fact the JPEG committee became aware of the poor acceptance of the lossless JPEG mode and finally found a successor, based on HP's LOCO algorithm.
This section deals with the Baseline System. Due to its interleaved mode of operation a straightforward application of selective encryption is not possible. The progressive DCT-based and Hierarchical mode are based on the Baseline System, which is therefore discussed. Further extensions of Baseline JPEG include increased color precision (12-bit samples) and arithmetic coding.

2.1.1 The JPEG Compression Pipeline

A compression pipeline in general transforms raw source data in to a compressed form. We assume that our raw image data is grouped in **components** (e.g. for each color). These components contain height x width entries, called **pixel** where height is the image height and width is image width (in pixel). The term "height x width" is referred to as **resolution**. Each pixel has a value and the precision it is stored with is frequently called **color depth**. Common values are 8 bit and 12 bit and in the case of JPEG the only ones that are supported. Baseline Sequential JPEG is a **lossy compression method**, which means that information considered to be of minor visual importance is lost in the compression process. Hence an image can generally not be fully reconstructed from the compressed data.

The Encoding Procedure

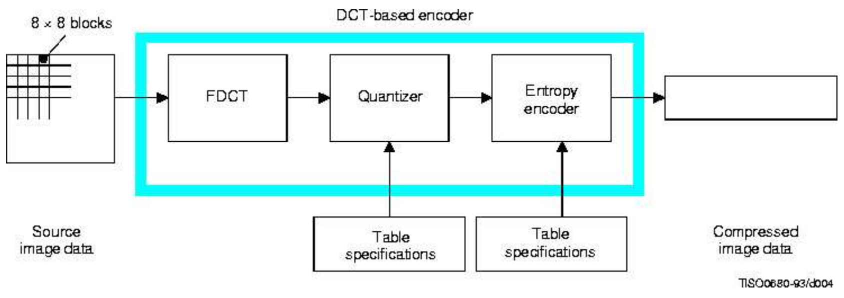


Figure 2.1: JPEG Encoding Procedure [6]

First all components are subdivided into 8x8 blocks, if height or width are not a multiple of eight padding is used. The encoder is free to choose any padding. The goal is to minimize the cost of coding. On each of these 8x8 blocks the **FDCT**, Forward Discrete Cosine Transform is applied. The resulting coefficients are then quantized. Only in this step information is lost, except numerical inaccuracies in other steps. The quantized coefficients are then entropy encoded through a Huffman encoder. The specification for this process has to be supplied in form of Huffman tables.

The Decoding Procedure

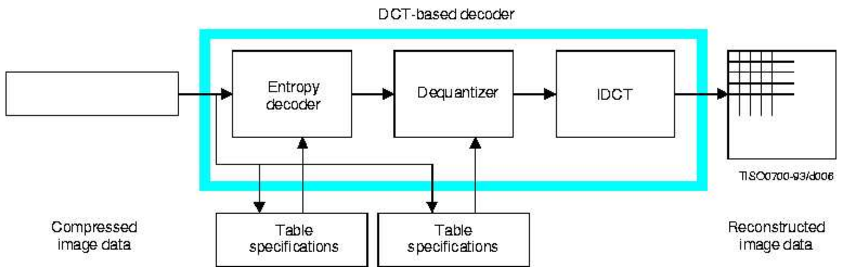


Figure 2.2: JPEG Decoding Procedure [6]

In the decoding procedure the Huffman encoded parts of compressed image are decoded using the Huffman code specified through the Huffman tables, which have to be in the compressed image. After Huffman decoding we have obtained the quantized coefficients, which are then dequantized (again the quantization tables that specify this process have to be in the compressed image). In the next step the **IDCT**, Inverse Discrete Cosine Transform, is applied, resulting in an approximation of the source image.

2.1.2 Discrete Cosine Transform (DCT)

Before the blocks of the image components are discrete cosine transformed, a **level shift** is applied (this assumes that the raw image is always unsigned). This level shift subtracts 2^{n-1} from every pixel for n-bit imagery. For the usual 8-bit image 128 is subtracted from every pixel, accordingly the range of the pixels is $(-128, 127)$. The goal of the FDCT is to map the 8x8 block of pixel to much less correlated 8x8 block of coefficients. The coefficients have less and less influence on the overall error when reconstructing the image with increasing x^* and y^* indices. This is often "referred to as energy compaction property". We thereby refer to x^* and y^* as indices of the coefficient block, whereas x and y denote the indices of the raw blocks. Numbering starts in the top left corner.
The mathematical definition of the FDCT is given below:

Definition 2.1.1 (Forward Discrete Cosine Transformation)

$$F(x^*, y^*) = \frac{c(x^*)c(y^*)}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\frac{(2x+1)x^*\pi}{16} \cos\frac{(2y+1)y^*\pi}{16}$$

where c(z)= $\begin{cases} \frac{1}{\sqrt{2}} & if\ z = 0 \\ 1 & otherwise \end{cases}$

$F(x^*, y^*)$ denotes the coefficient at position x^*, y^* in the block, which is derived from all pixel (denoted by $f(x,y)$) weighted by cosine terms. For $F(0,0)$ we obtain eight times the average of all pixel. This is easily shown, since $\cos(0) = 1$ and $c(0) = \frac{1}{\sqrt{2}}$ the remaining term is $\frac{1}{8}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)$. Hence the first coefficient ($F(0,0)$) is called **DC coefficient**, whereas all other are referred to as **AC coefficients**. The term separable means that this procedure can be separated into two operations, one for the columns and one for the rows. This is of importance for memory saving implementations. The resulting coefficients are in the range of $(-1024, 1024)$. Apart from numerical inaccuracies this process is fully reversible. For completeness the definition of the IDCT is given below:

Definition 2.1.2 (Inverse Discrete Cosine Transformation)

$$f(x,y) = \frac{1}{4} \sum_{x^*=0}^{7} \sum_{y^*=0}^{7} F(x^*,y^*)c(x)c(y) \cos \frac{(2x+1)x^*\pi}{16} \cos \frac{(2y+1)y^*\pi}{16}$$

2.1.3 Quantization

The quantization process of JPEG is quite simple to explain. Every coefficient of the DCT is integer divided by a quantization divisor. This is done for all coefficients of a block. The quantization divisors are specified in quantization

16	11	10	16	124	140	151	161
12	12	14	19	126	158	160	155
14	13	16	24	140	157	169	156
14	17	22	29	151	187	180	162
18	22	37	56	168	109	103	177
24	35	55	64	181	104	113	192
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	199

Table 2.1: The Luminance Quantization Table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Table 2.2: The Chrominance Quantization Table

tables and are the same for every block of a component. Hence precision is lost through this division and the coefficient will get zero if the quantization divisor is bigger than the coefficient itself. Quantization tables have to be specified (so it is free to define one's own quantization tables), but tables which were empirically derived are proposed. Hence these tables are very frequently used by implementations. There are two different quantization tables, one for chrominance and one for luminance. Since the topics of this paper are not focused on human visual perception, color transforms will not be discussed in detail. Although the JPEG compression is for arbitrary number of components, the most common type of image to compress is three component color image. This image is quite often represented in a RGB color space. As it is considered to be empirically proven, that those colors have different impact on the human visual system (red is perceived far more intensely than blue), it does not seem appropriate to compress every color component with the same procedure. Furthermore the colors from RGB space can be mapped to one luminance and two chrominance channels. The luminance channel is more important for human visual perception than the chroma channels. This can be seen in their quantization tables (2.1 and 2.2) too. The quality of an image can roughly be controlled through the quantization tables. Usually this is done by modifying the quantization tables through scaling. Since this procedure is not all intuitive for most users, a new term **quality factor** was invented. It is in the range of $(1 \ldots 100)$. It is converted into a scalar multiplier α as follows:

Definition 2.1.3 (Quality Factor)

$$\alpha = \begin{cases} \frac{50}{q_f} & if \quad 1 \leq q_f \leq \quad 50 \\ 2 - \frac{2q_f}{100} & if \quad 50 \leq q_f \leq \quad 100 \end{cases}$$

where q_f is the quality factor.

2.1.4 Coding

After having applied the FDCT and quantization to a block, the DC and the AC coefficients are prepared for entropy encoding (shown in figure 2.3). The DC coefficient of the previous block is used to predict the value of the current DC coefficient and the difference is encoded. The AC coefficients are ordered in to a 1-dimensional sequence, using the (so called) zig-zag scan illustrated in figure 2.3. Again the previous coefficient in the sequence is used as prediction for the current

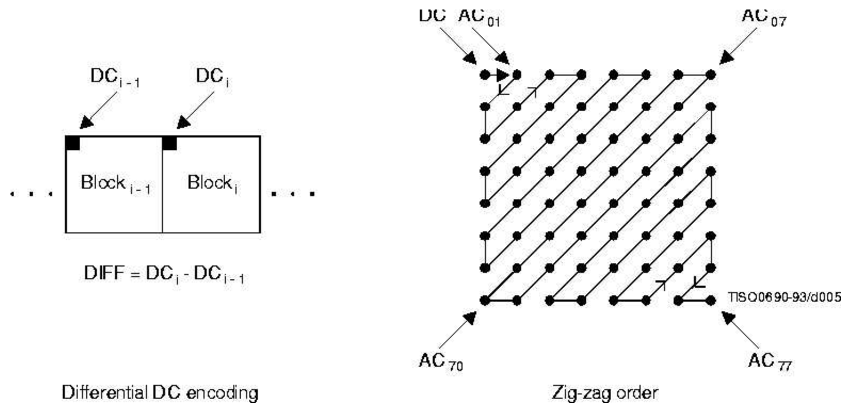


Figure 2.3: The DCT Coefficients [6]

coefficient and the difference (residual) is encoded. Two entropy encoders can be chosen according to the standard, Huffman or arithmetic encoding. Nevertheless most applications only support Huffman coding. Huffman coding is mandatory for the Baseline System while arithmetic coding is only available in the Extended

System. JPEG does not use plain Huffman codes, but a more sophisticated form of this coding technique mixed with run legth coding. First of all a special symbol **EOB**, End of Block, is employed signaling that the following residuals of the block are all zero. Considering the increasing values of the quantization table in zig-zag sequence, this is quite likely to save space. Since there are 4096 different residual values (for 8-bit raw images) to be Huffman encoded this would generally lead to quite long code words and suboptimal compression results. Hence the range of possible residual values is subdivided into pairs of symmetrical intervals (categories). Within a category codewords are assigned. For a detailed description refer to [6] at page 89.

2.2 Progressive DCT-based Mode

There are two distinct DCT-based progressive modes, namely **successive approximation** and **spectral selection**. They can be combined rather freely. While these two are already implemented in reference library, Hierarchical JPEG is not supported.

2.2.1 Spectral Selection

The DCT is applied to an 8x8 block of the image. The result is an 8x8 block of DCT coefficients, which are quantized and ordered through the Zig-Zag scan (see 2.1.4). In standard mode of operation, these coefficients are consecutively coded. More precisely the image is subdivided into its components. These components then are divided into 8x8 blocks. Starting at the first block of the first image component, it gets transformed, quantized and all coefficients are coded. Then the first block of the next image component is dealt with up to the last image component. This is done block for block, resulting in a component interleaved code stream on a block basis. Optionally one can turn off the default interleaved mode, and encode each component on individually. Spectral Selection breaks up the block borders and DCT coefficients can be grouped by their Zig-Zag scan index. The components are still subdivided into 8x8 blocks that are DCT transformed and quantized, but now one can specify which coefficients are coded and where they are placed in the file stream.

2.2.2 Successive Approximation

The quantized DCT coefficients have a binary representation, which is 8 bit for the Baseline and can be 12 bit for the Extended System. The idea of Successive Approximation is to group the coefficient data through their binary positions. More precisely, after applying DCT and quantization to the blocks of the components, we have all coefficient data ready for coding. But instead of coding all of the binary representation of the coefficient data, Successive Approximation allows us to code parts of the binary representation separately. The first 6 bit of a coefficient is the smallest fraction which the JPEG standard allows to specify. This fraction is then coded as usual, while the following bits are emitted without coding. It has to be remarked that according to the standard DC and AC coefficients have to be treated separately.

2.2.3 Compression Performance

Naturally the quality of an image compression method is determined by its coding performance. In the following sections PSNR plots of the progressive DCT-based modes are presented. The test were conducted using the user programs `cjpeg` and `djpeg` of the IJG's reference library. Quite surprisingly the coding performance of these modes is superior to standard JPEG. The test image was the classical Lena picture. The PSNR plots result from varying the JPEG quality parameter in its whole range $(1, \ldots, 100)$.

Successive Approximation

Figure 2.4 shows a comparison of standard JPEG with successive approximation mode of operation. The data is organized as follows:

- First 6 bit of all DC coefficients Huffman coded
- 1 bit more of DC coefficient data
- 1 bit more of DC coefficient data
- First 6 bit of all AC coefficients Huffman coded
- 1 bit more of AC coefficient data
- 1 bit more of AC coefficient data

Spectral Selection

Figure 2.5 shows a comparison of standard JPEG with spectral selection mode of operation. The data is organized as follows:

- First all DC coefficients Huffman coded
- AC coefficients 1 to 6
- always consecutive groups of 6 AC coefficients
- AC coefficients 42 to 49
- two more groups of 7 coefficients

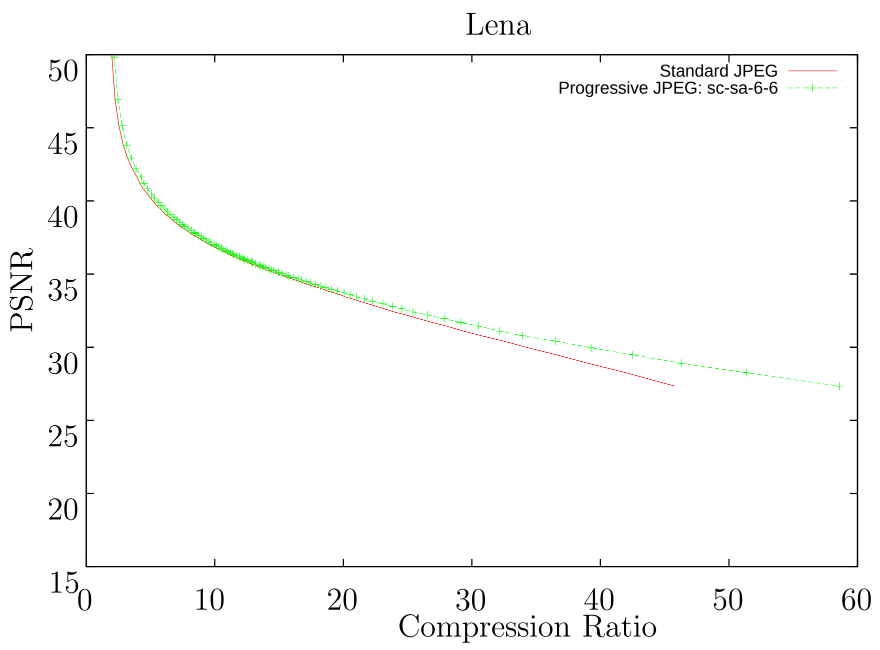


Figure 2.4: PSNR Plot of Successive Approximation

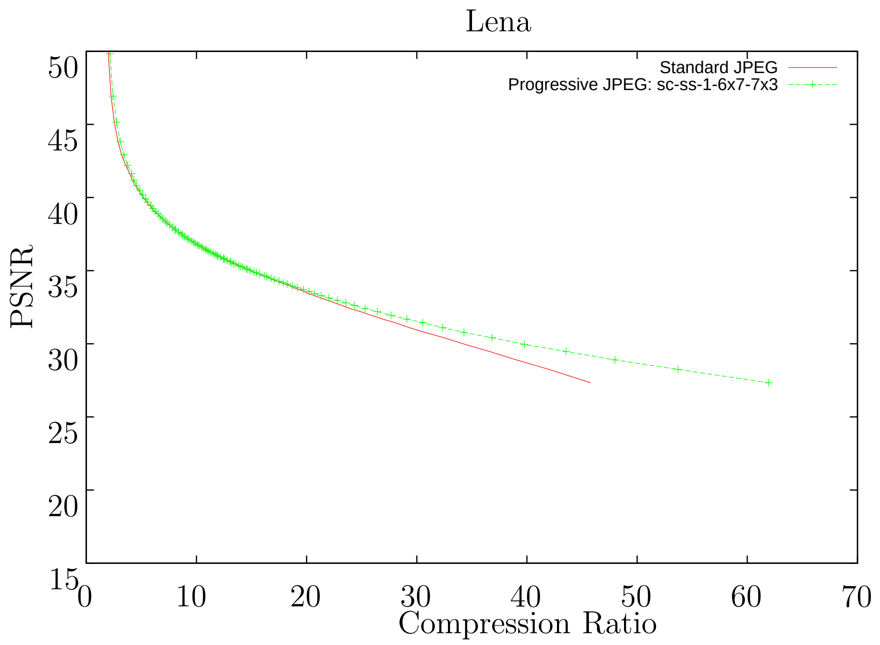


Figure 2.5: PSNR Plot of Spectral Selection

Spectral Selection and Successive Approximation

Figure 2.6 shows a comparison of standard JPEG with spectral selection and successive approximation mixed. This is libjpeg's default configuration of the DCT-based progressive mode and it organizes the data as follows:

- First 7 bit DC coefficients Huffman coded
- First 6 bit of first 5 AC coefficients
- 6 bit of the remaining AC coefficients
- Next bit of all AC coefficients
- Lowest bit of DC coefficients
- Lowest bit of all AC coefficients

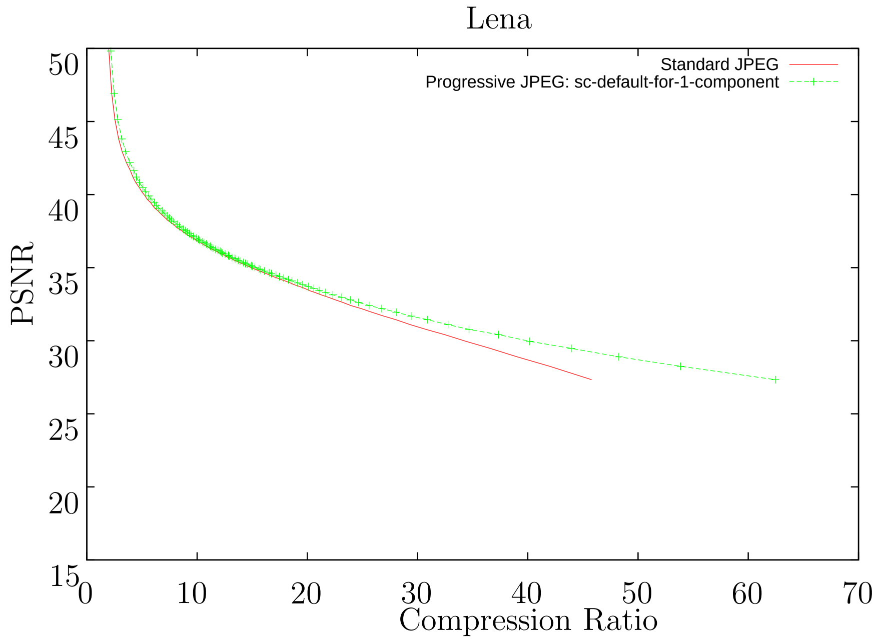


Figure 2.6: PSNR Plot of Default Progressive Mode

2.3 Hierarchical Mode

Hierarchical mode is resolution progressive. This is achieved through coding a sequence of layers in a pyramid (see figure 2.7). Therefore the original image is $(n-1)$-times sub-sampled for n resolution layers. Figure 2.8 shows subsampling of the Lena image. In the **base layer** (also referred to as layer 0) the smallest resolution subsample is encoded with one of the JPEG compression methods. In the following layers no subsamples of the image are encoded, but a differential image is constructed and encoded. For the first layer, the smallest resolution subsample is upsampled and used as a prediction. The differential image is then again JPEG encoded. For the next layers the procedure is quite similar, the image of the previous layer is reconstructed and used as a prediction. Figure 2.9 shows the result for three layered Hierarchical JPEG compression of the Lena image. The base layer is encoded with 75% JPEG quality, layer 1 and 2 are quantized with a uniform quantization table of 14. This procedure is thought to exploit spatial correlation and therefore gain coding performance in comparison to independently coding multiple resolutions. It is possible to reconstruct a Hierarchical JPEG compressed image at different resolutions. Figure 2.10 shows all reconstructions.

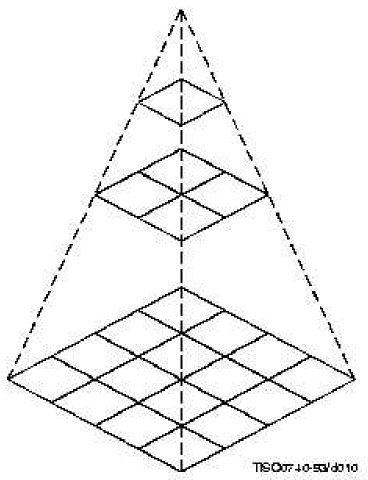


Figure 2.7: Hierarchical Multi Resolution Encoding [6]

2.3.1 Implementation

As already mentioned above implementations for Hierarchical JPEG are rare. Therefore no suitable implementation for our purpose was found and we have decided to implement a subset Hierarchical JPEG based on the **IJG's** (the Independent JPEG Group) reference library. This section is thought to give a detailed description of the subset of Hierarchical JPEG that we implemented. First of all it is limited to gray scaled images, which for academic purposes is not that a big restriction as for everyday applications. Some restrictions result from that fact that the implementation is based on IJG's reference library. The library itself is capable of the Baseline processes and in a restricted form of the Extended System. The decision between 8-bit and 12-bit samples is a compile time decision and we quite frankly have to admit that the attempt to get it to work properly with 12-bit support was of limited success. Since on most systems (probably all) the library is compiled with default 8-bit support, our implementation is limited to 8-bit images too. Further differences between the implementation and its description in the standard are discussed in 2.3.1.

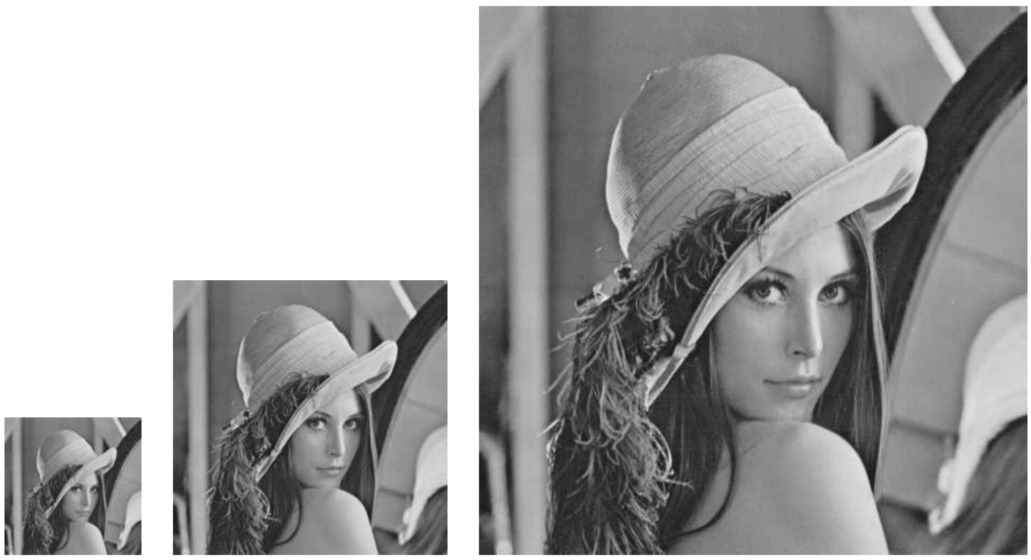

Layer 0 Layer 1 Layer 2

Figure 2.8: Subsampling

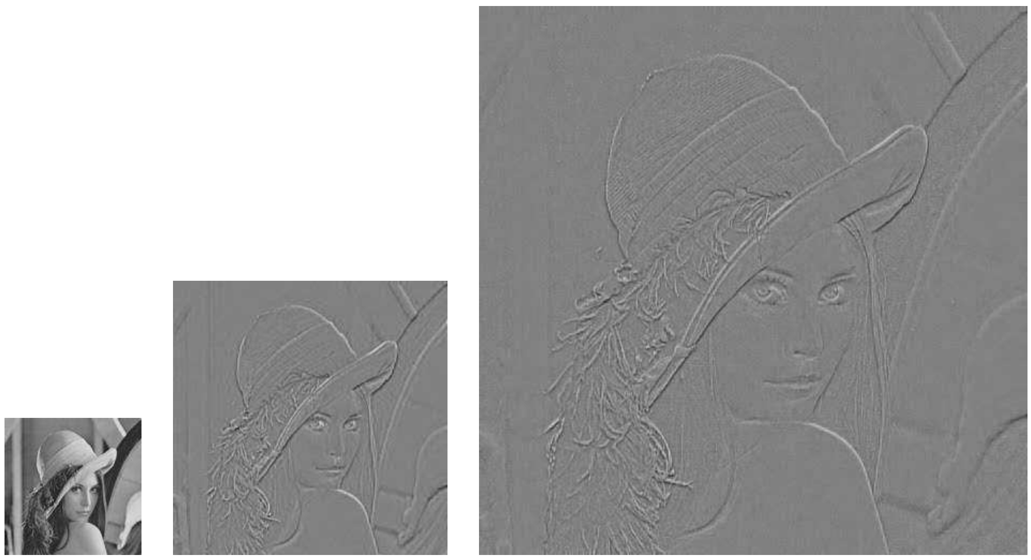

Layer 0 Layer 1 Layer 2

Figure 2.9: Differential Frames

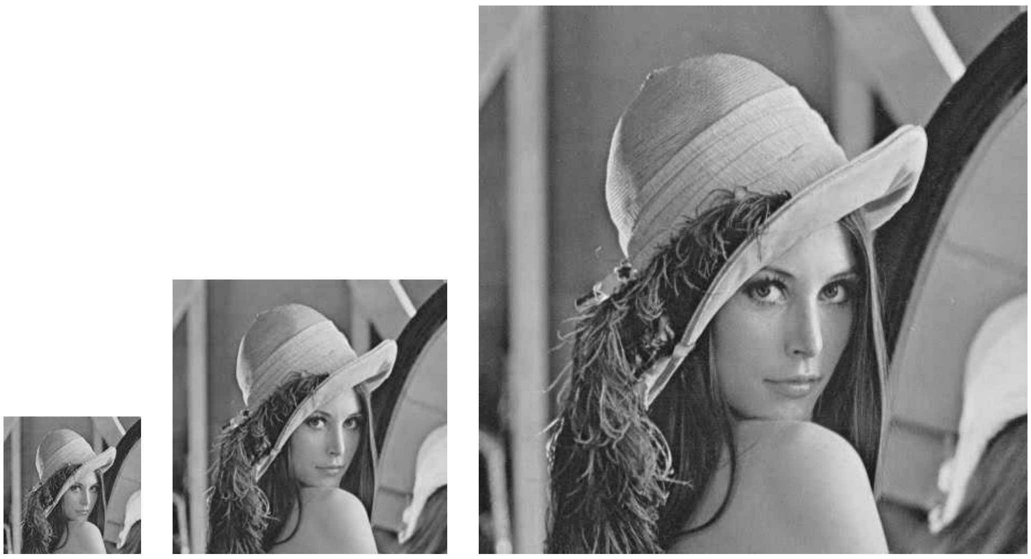


Figure 2.10: Reconstructions

Encoding

As most coding decisions are encoder choices, one can specify quite a number of parameters. First of all the number of layers can be specified and then for each layer a modifier for its quantization table. Depending on which type of quantization table is used (this can be specified too), the interpretation of this value differs. If using the default table, it is read as "quality factor" (see Definition 2.1.3), whereas when using a uniform quantization table it is the value of all quantization table entries. Furthermore baseline or progressive mode can be used. Having specified these parameters the encoding process can proceed.

1. The raw image is $(n-1)$ subsampled for n layers.
2. The base layer (smallest resolution image) is JPEG encoded.
3. The JPEG encoded base layer is upsampled.
4. The differential image between the upsample and the subsample of the same resolution is JPEG encoded (enhancement layer 1).
5. JPEG encoded enhancement and base layers are used to reconstruct a lower resolution image which is upsampled.
6. The differential image between the upsample and the subsample of the same resolution is JPEG encoded.
7. If there are more layers to be coded goto 5.

Considering that there are no applications supporting Hierarchical JPEG, there was no need to implement the correct file stream syntax of the Hierarchical mode of operation. The result of encoding is a single JPEG file for every layer. Nevertheless the specific characteristics of the Hierarchical file stream have to be reconsidered when evaluating e.g. the compression performance. Using the default settings of the reference library the difference between the sizes of the Hierarchical JPEG file and the collection of separate JPEG files can be calculated straightforwardly. Apparently the collection of separate files is bigger than the size of the single file. The reason for this is mainly the subsequent specification of the same Huffman tables in each file, which is not necessary in a single JPEG file. Since the quantization tables differ in general, they can not be subtracted. The markers signaling start and stop of an image are solely interchanged for markers signaling start and stop of inter code stream image. Hence the resulting Hierarchical JPEG file would be $215(n-1)$ bytes smaller than the separate files, where n again is number of layers. This is only true for using libjpeg's default settings. Using other Huffman tables or encoders might lead to different results, since the size of the Huffman table may vary.

Sampling Filters

The upsampling and downsampling filters that are suggested in [6] are implemented. The upsampling filter increases the resolution by a factor of two. The

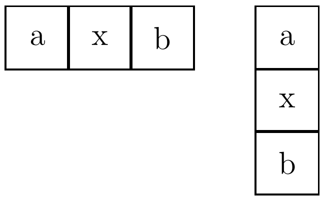


Figure 2.11: Upsampling Filter

filter can be applied vertically and horizontally. As illustrated in figure 2.11, the pixel x is the value to be interpolated. This is done by a simple bi-linear interpolation, $V_x = \frac{V_a+V_b}{2}$, where V_z is the value of pixel z. In order to downsample, one takes every second pixel of an image and replaces it with the weighted sums of it and its neighbors. The downsample filter implemented and illustrated in

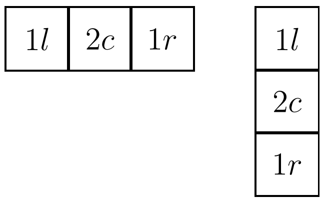


Figure 2.12: Downsampling Filter

figure 2.12 maps every second pixel to $\frac{V_l+2V_c+V_r}{4}$. Again this filter can be applied vertically and horizontally, which is done in our implementation.

Decoding

Decoding a Hierarchical JPEG file starts with identifying the base layer image in the file stream. The base layer is upsampled and the next differential image added. For every remaining differential image the previous reconstructed image is upsampled and the differential image added.

Non Standard Properties

The standard [6](p.140) demands a different coding model for differential images which is partly ignored in our implementation. However, it seems justifiable to do so. In detail the required modifications are:

1. The coding models are modified such that they are capable of coding two's complement differences.
2. The DC coefficient is coded without prediction.
3. 1 extra bit of precision for the Huffman coding of the AC coefficient is added.

The first point of the modification list is simply realized through omitting the level shift (see section 2.1.2). In our implementation this is realized through adding the level shift of 128 to every pixel in the difference image before encoding. The following two points, which both cope with the need for one extra bit of precision, are a result of the differential process. In fact the differential image's pixel are in the range of $(-255, 255)$, which corresponds to a 9 bit precision image. As we are using the reference library only 8 bit images are supported.
There are reasons why this acceptable. First the difference is not made between two totally different images, but between the upsampled low resolution version of an image and the original image, which tend to be numerically closer. Furthermore we deal with continuous-tone images. It can be shown that the upsampling and the downsampling guarantee difference values in range at least for horizontal, vertical and diagonal edges between extremely varying homogeneous regions (zero values next to 255 in the raw image). Nevertheless there are images for which the upsampling and downsampling procedure does not ensure difference values in range. A simple example where the difference values are out of range, is an image where every second row and column is zero and the rest 255. Another simple way to find such images is random distribution of zero and 255 in an image. Assuming continuous-tone imagery, these images might be considered as notoriously constructed, but they show a certain lack in the implementation, which has to be considered.
Therefore two major aspects have to be discussed, the influence on the compression performance in terms of file size and image quality.

In fact the only difference is made by the coding of the DC coefficients and a few bytes more for the AC Huffman table specification if no out of range difference appears. If an out of range difference appears, it is coded as if in range (disregarding MSB) in our implementation.
Since the Huffman table for the DC coefficients can be adjusted to this coding situation, coding performance should not vary greatly.
Considering the image quality, only cases were out of range difference happen, make a difference. When using multiple layers the error can be compensated in the consecutive layers. However, such errors can not be completely ruled out.

2.3.2 Parameter Study

Because Hierarchical JPEG is not widely used, nothing or very little is known about good parameters settings. Hence we tested the a subset of the parameter space for two layers up to six layers. For each number of layers the quality parameter q_f for the lowest resolution image is varied in the range of $1, \ldots, 100$. For all other layers the scalar multiplier α is varied in the range of $0, \ldots, 255$. The resulting PSNR plots are shown in figures 2.13 to 2.17. The best results are obtained with the fewest layers (one base layer, one enhancement layer). Furthermore the fractal nature of the results is remarkable. The results show that there are parameter constellation which are quite close to the compression performance of baseline JPEG, but there are many very bad performing parameter constellations too. Therefore further analysis had to be conducted.

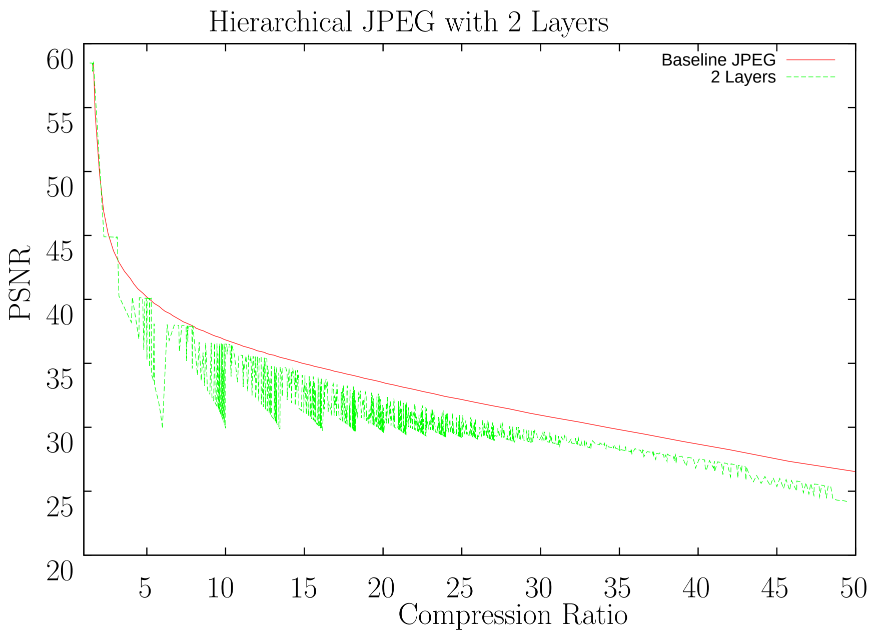


Figure 2.13: Coding Performance of Hierarchical JPEG: 2 Layers

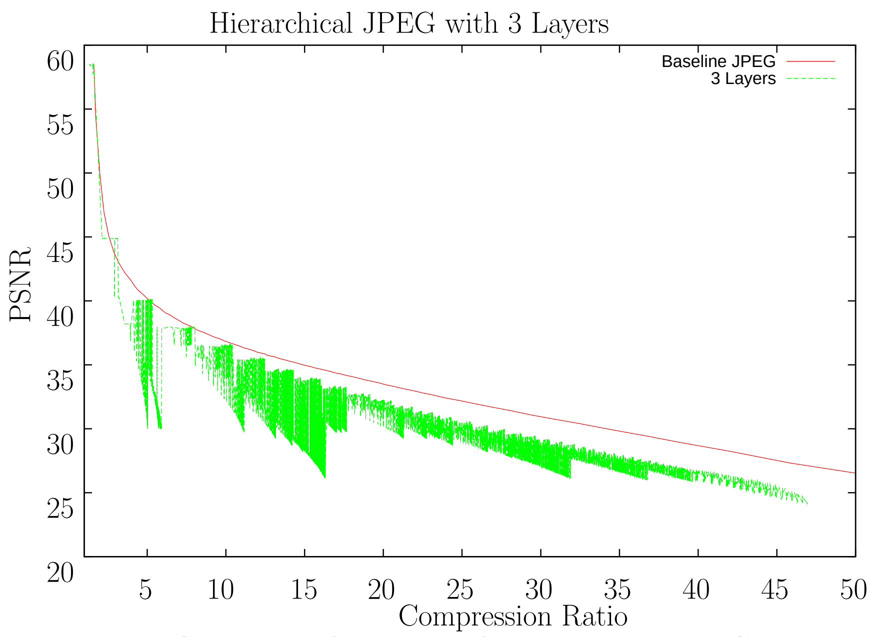


Figure 2.14: Coding Performance of Hierarchical JPEG: 3 Layers

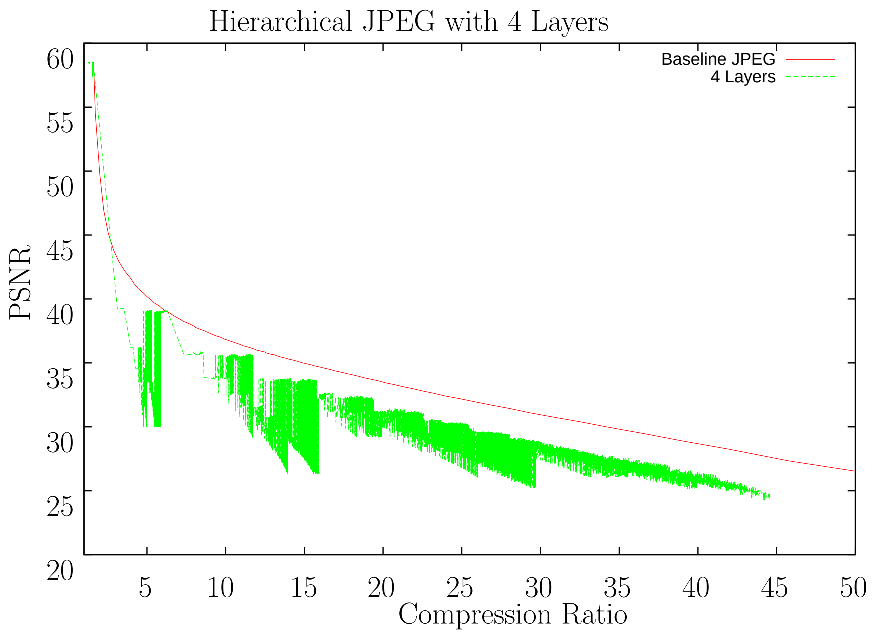


Figure 2.15: Coding Performance of Hierarchical JPEG: 4 Layers

2.3.3 Correlation of the Encoding Parameters

In figure 2.18 the result of a correlation analysis of the encoding parameters with compression ratio and PSNR is shown. The first column shows the correlation between the PSNR and different coding parameters. Results are presented for

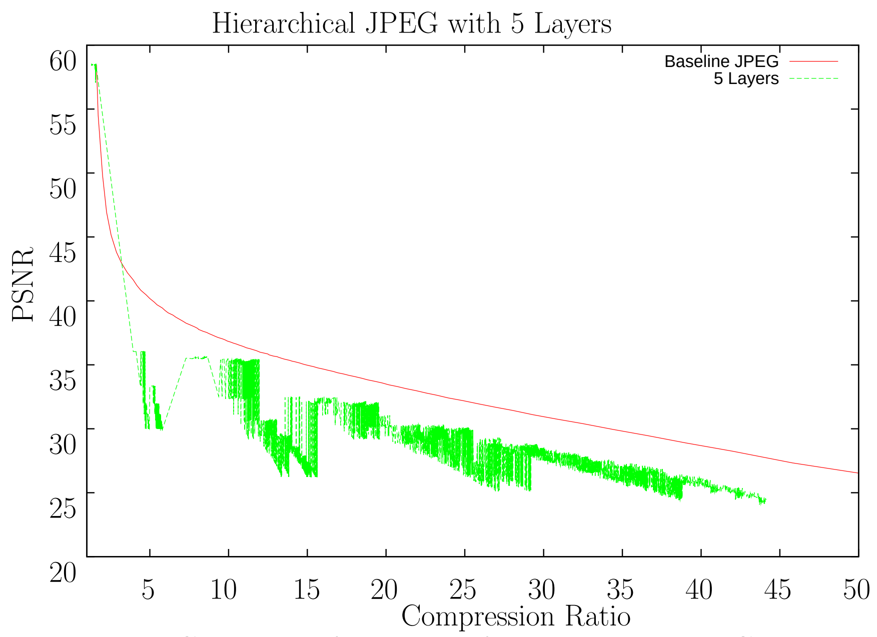


Figure 2.16: Coding Performance of Hierarchical JPEG: 5 Layers

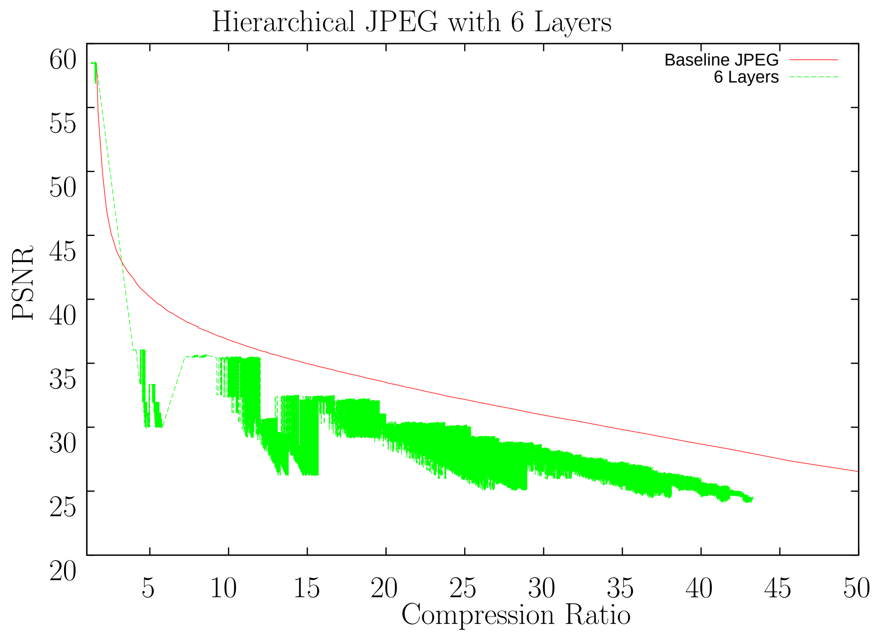


Figure 2.17: Coding Performance of Hierarchical JPEG: 6 Layers

Hierarchical JPEG with two to six layers. The first row of each plot show the correlation of the parameter "JPEG quality" with the PSNR. The following rows show the correlation of the scalar multiplier (see section 2.3) with the uniform quantization matrix of each following layer. The second column shows the correlation between the compression ratio and these parameters. The different sign of

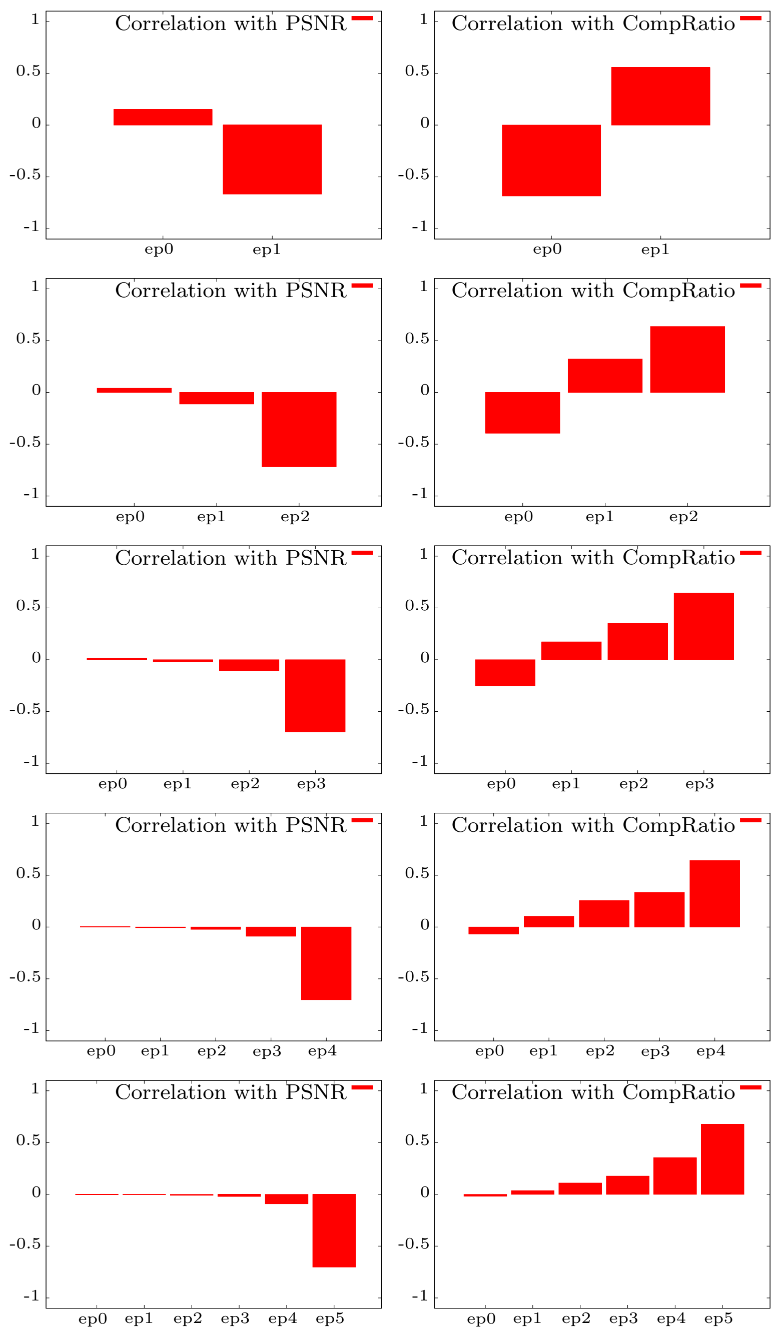


Figure 2.18: Correlation of Coding Parameters with PSNR and Compression Ratio

the coding parameter of layer 0 is explained as follows:
While the increase of JPEG quality results in a image of higher quality, an increase of the scalar multiplier reduces the quality of the image. Hence the results can be read that the lower the last coding parameter, the better the image qual-

ity. Far more uncorrelated is that a higher JPEG quality leads to a higher PSNR. The influence of the coding parameters decreases with their resolution. Another noticeable fact is that PSNR is higher correlated to higher resolutions than the compression ratio. This implies that the last layer (highest resolution) is most important for image quality while encoding the other layers at good quality can significantly decrease compression performance.

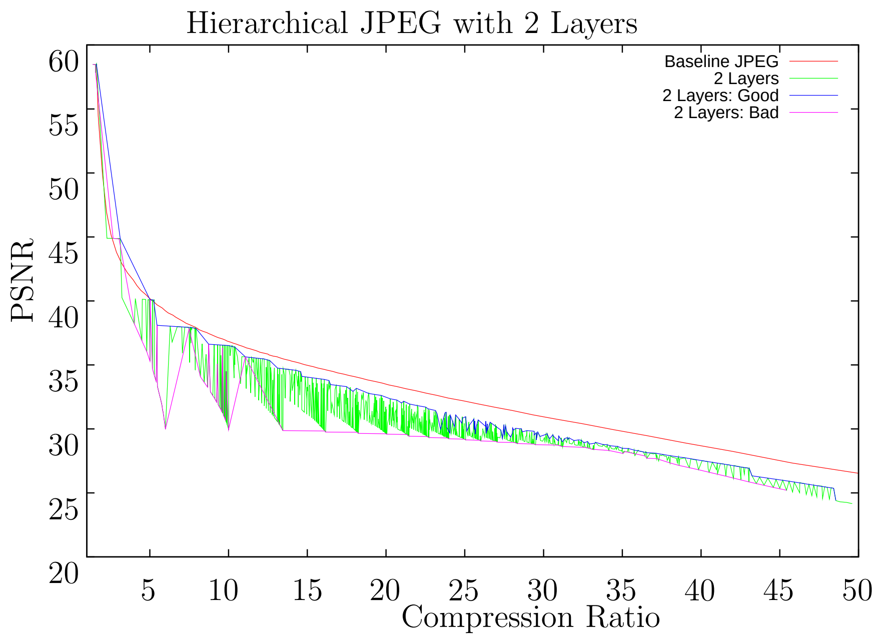


Figure 2.19: Filtering the Results for 2 Layers

2.3.4 Further Investigations

The basic idea of our further investigation has been to filter out the "good" and the "bad" parameter constellations and then conduct again a statistical investigation into how parameters differ in these groups. Since the plain goal of this whole investigation is to find quite appropriate parameters for Hierarchical JPEG, it is not evaluated and discussed with mathematical strength and sincerity. At first we constructed a very simple filter for "good" constellations (implemented as shell script). It sorts all tested parameters according to their PSNR. Starting at the highest PSNR value, it considers n following entries of same quality and picks out the one with the best compression ratio. This is quite appropriate because we have many entries with the same or very close PSNR values. The filter for the "bad" constellations is almost the same, but we take those with the worst compression ratio. The results are not that bad considering the triviality of the filter. Figures 2.19,2.20 and 2.21 show the result for two,three and six layers. The following tables show the results of the statistical analysis of the filtered results. It turns out that the bad constellations tend to have a highest resolution image

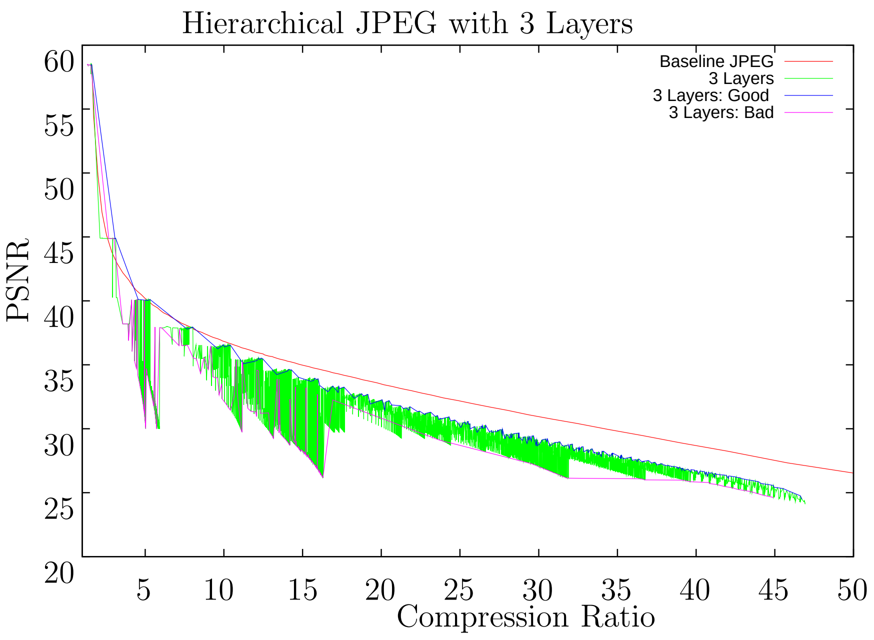


Figure 2.20: Filtering the Results for 3 Layers

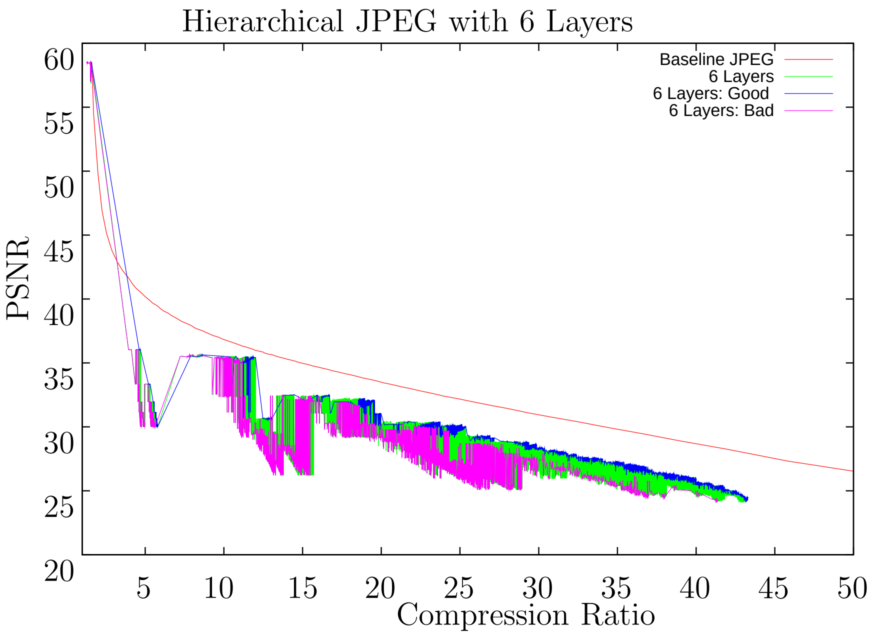


Figure 2.21: Filtering the Results for 6 Layers

of bad quality and low resolution images of higher quality, whereas for good constellations approximately the opposite is true. Good Constellations tend to have a high quality highest resolution image and the lower resolution layers of worse quality.

Good Constellations of Two Layers

	PSNR	Ratio	ep0	ep1
Mean	31.32	25.86	22.38	92.97
Std Deviation	4.77	9.78	15.70	53.18

Bad Constellations of Two Layers

	PSNR	Ratio	ep0	ep1
Mean	31.50	16.09	70.44	148.38
Std Deviation	5.22	11.74	34.84	81.91

Good Constellations of Three Layers

	PSNR	Ratio	ep0	ep1	ep2
Mean	30.45	28.50	22.32	161.94	113.42
Std Deviation	5.39	11.11	18.78	60.71	72.19

Bad Constellations of Three Layers

	PSNR	Ratio	ep0	ep1	ep2
Mean	30.50	13.29	87.09	62.93	149.80
Std Deviation	5.56	8.21	30.75	59.93	77.91

Good Constellations of Six Layers

	PSNR	Ratio	ep0	ep1	ep2	ep3	ep4	ep5
Mean	31.69	25.39	15.22	143.20	145.35	141.19	132.48	119.94
Std Deviation	8.89	12.40	35.92	74.74	75.54	78.91	74.84	79.99

Bad Constellations of Six Layers

	PSNR	Ratio	ep0	ep1	ep2	ep3	ep4	ep5
Mean	31.69	17.60	66.59	76.31	65.34	97.09	114.43	132.60
Std Deviation	8.89	9.65	47.16	88.79	86.98	83.63	82.78	82.44

Table 2.3: Parameter Analysis for Hierarchical JPEG

2.3.5 Conclusion

Based on the results of our experiments, we are able to suggest good coding parameters for Hierarchical JPEG. Figures 2.22, 2.23 and 2.24 show the resulting PSNR plots. The labels in the figures have to be read as follows:
Each label is of the form "n Layers q_f% α" , where n is the number of layers, q_f is the quality factor of the base layer and α is the scalar multiplier for the uniform quantization table consisting soley of ones. The best coding performance of Hierarchical JPEG is achieved, when the base layers are of low quality, which is of course in most cases not the goal of Hierarchical JPEG. Nevertheless figures 2.22, 2.23 and 2.24 show that there is range in the coding parameters (especially in the high quality region), where suitable compromises between coding performance and possible needs of an application (the main advantage of Hierarchical JPEG are its multi resolution capabilities) can be found. Figures 2.25 and 2.26 show the Hierarchical JPEG frames and reconstructions with lower quality for the lower

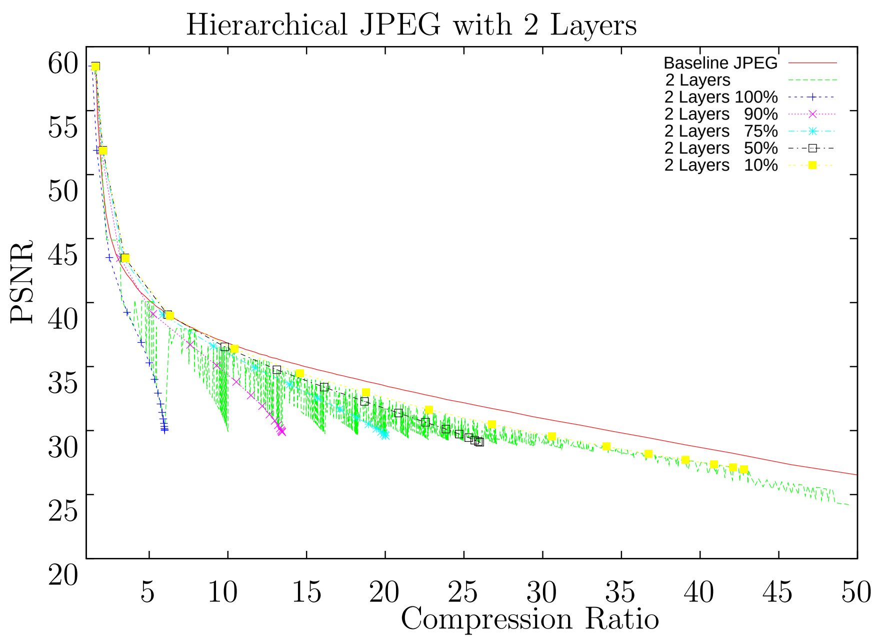


Figure 2.22: Results for Hierarchical JPEG: 2 Layer

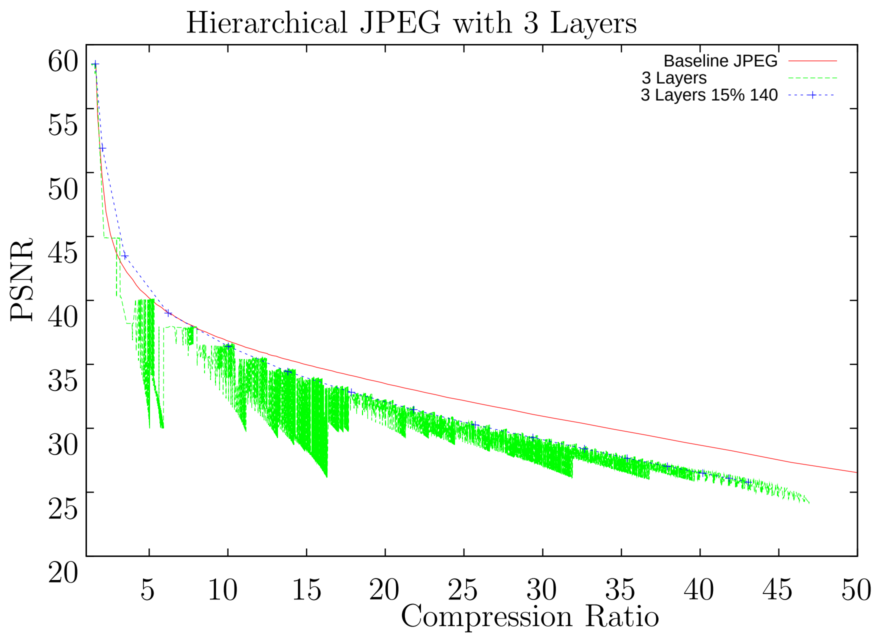


Figure 2.23: Results for Hierarchical JPEG: 3 Layer

resolutions. The results can be compared to figures 2.9 and 2.10

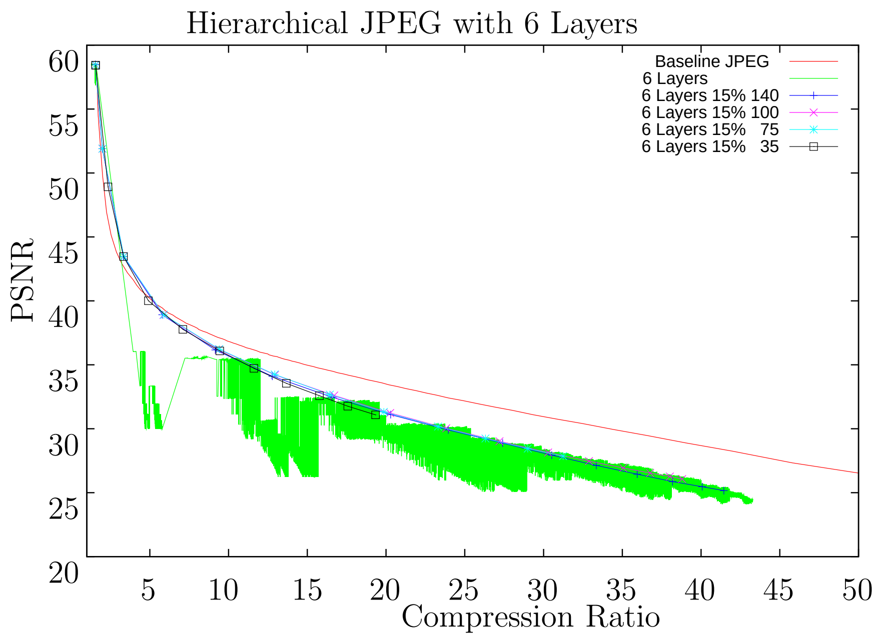


Figure 2.24: Results for Hierarchical JPEG: 6 Layer

Combining Hierarchical JPEG and Progressive Mode

It is possible to combine Hierarchical JPEG with the sequential progressive modes. This leads to a significant improvement of the coding performance, even superiour to those of Baseline JPEG in some cases. Figure 2.27 shows the result for two layers and some parameter constellations. Despite the graph seems quite complicate to read, it is not. It is always a pair of curves, of which the better results from progressive Hierarchical JPEG. Figure 2.28 shows the result for three layers and the "best" constellation.

2.4 Characteristics of Coding Processes

In order to give an final overview of the discussed JPEG coding processes tables summarizing their characteristics are presented [6].

Baseline process (required for all DCT-based decoders)
• DCT-based process • Source image: 8-bit samples within each component • Sequential • Huffman coding: 2 AC and 2 DC tables • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans

A standard compliant implementation of the Baseline System can only cope with

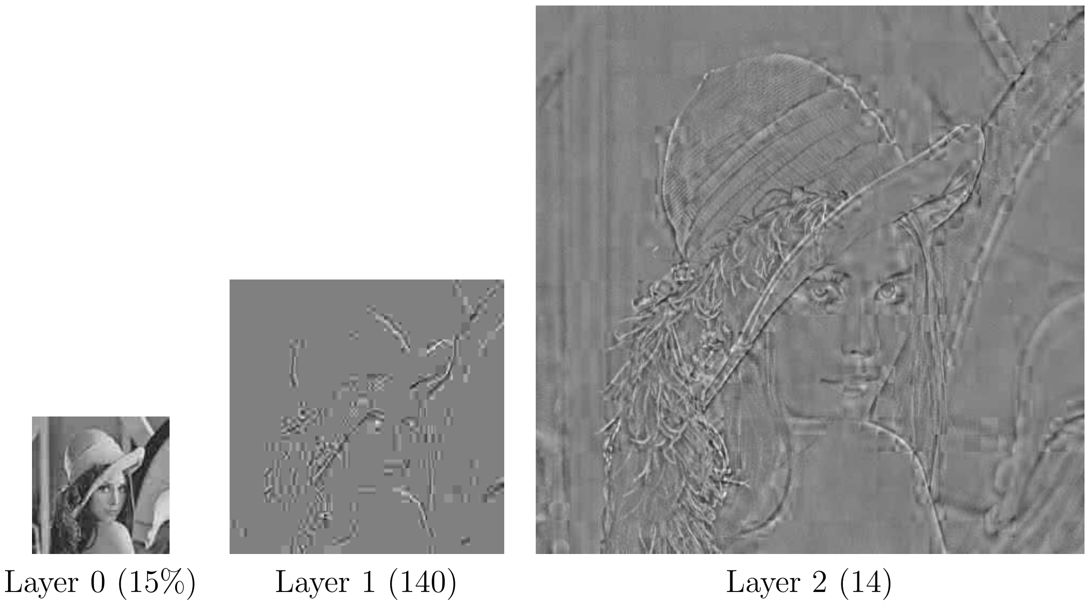


Figure 2.25: Differential Frames

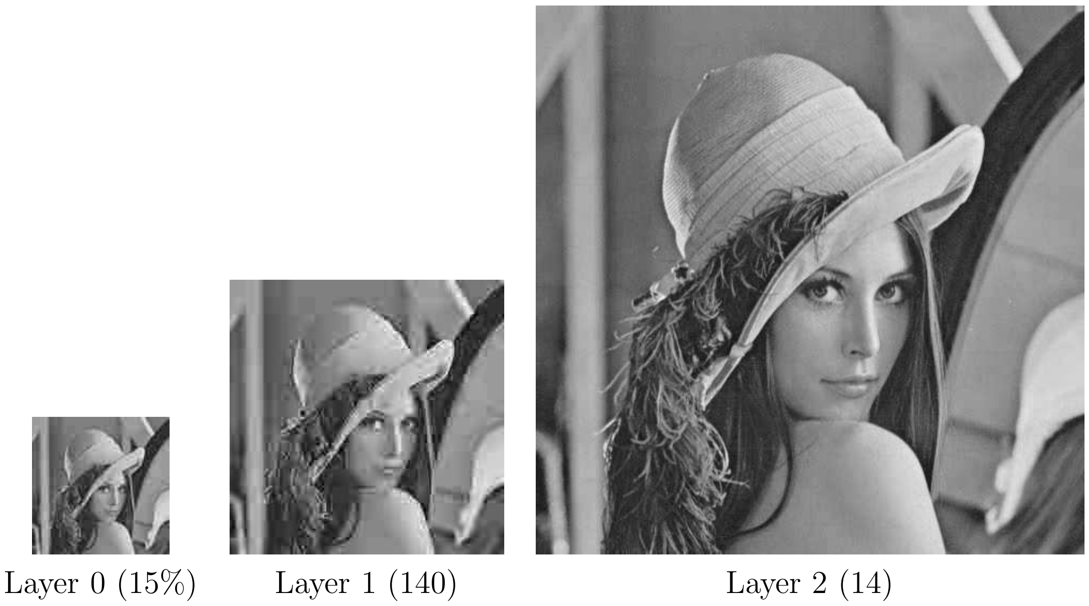


Figure 2.26: Reconstructions

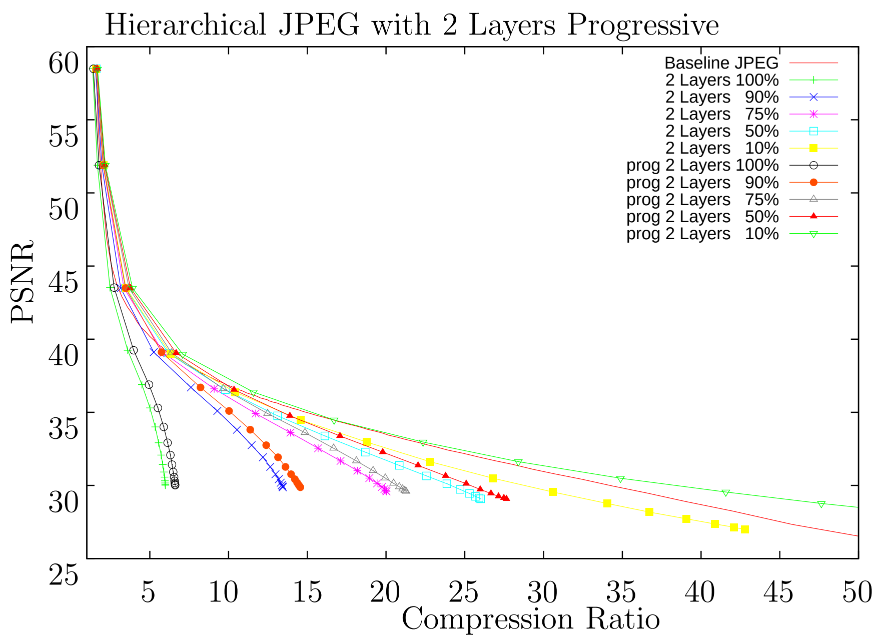


Figure 2.27: Hierarchical JPEG: 2 Layer Progressive

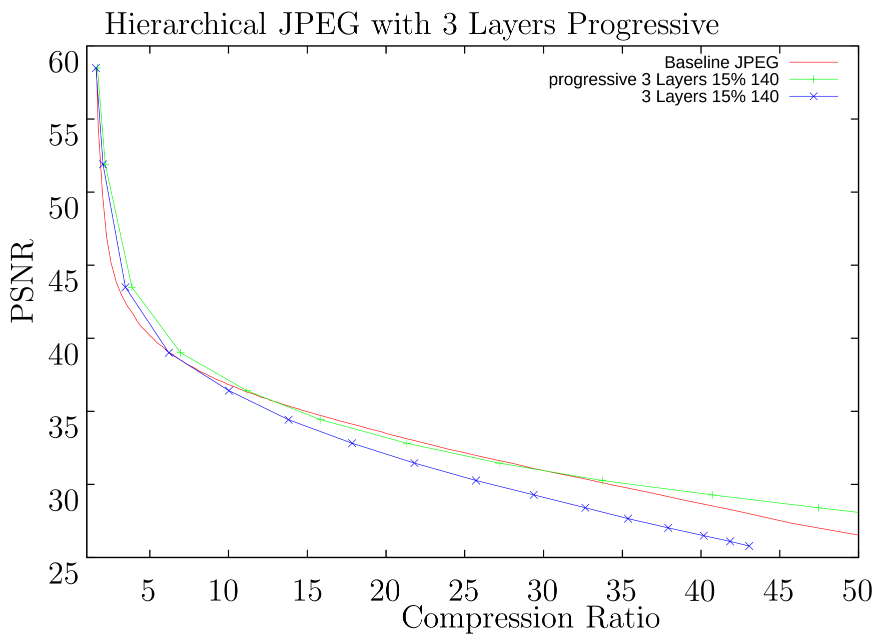


Figure 2.28: Hierarchical JPEG: 3 Layer Progressive

8-bit raw images. Furthermore it only Huffman coding is supported and at most four different tables can be specified. The components can be coded interleaved or non interleaved.

Extended DCT-based processes
• DCT-based process
• Source image: 8-bit or 12-bit samples
• Sequential or progressive
• Huffman or arithmetic coding: 4 AC and 4 DC tables
• Decoders shall process scans with 1, 2, 3, and 4 components
• Interleaved and non-interleaved scans

The Extended System supports 12-bit raw images, arithmetic coding and the progressive modes,successive approximation and spectral selection, too

Hierarchical processes
• Multiple frames (non-differential and differential)
• Uses extended DCT-based or lossless processes
• Decoders shall process scans with 1, 2, 3, and 4 components
• Interleaved and non-interleaved scans

The Hierarchical process is fully based on the Extended System and therefore comes in many variations.

Lossless processes
• Predictive process (not DCT-based)
• Source image: P-bit samples ($2 \leq P \leq 16$)
• Sequential
• Huffman or arithmetic coding: 4 DC tables
• Decoders shall process scans with 1, 2, 3, and 4 components
• Interleaved and non-interleaved scans

For completeness the properties of the lossless process is given. It is the only process which is not based on the DCT.

Chapter 3

Selective Encryption of JPEG

Various methods for encrypting DCT-based imagery have been proposed in the literature. Some can not be adopted for JPEG, ensuring at least partly code stream compliance. A distinctive feature of DCT based encryption methods is whether they are directly applied within the compression pipeline or on the bit stream of an image file. Within the compression pipeline a permutation of the zig-zag scan was proposed by [46] and [43]. This method leads to a standard compliant code stream, but also increases the file size and does not offer much security. Furthermore the encryption of the sign bits of the DC coefficients was proposed in [42]. Other methods like substituting and permuting the Huffman coefficients in a cryptographic secure manner. (can be considered a generalization of classic ciphers), are computationally expensive and increase the file size. All methods have to be assessed by the amount of security they offer and by their influence on the coding performance, computational cost, ability to scale and code stream compliance. Since discussing all proposed methods is out of the scope of this thesis, the following references to prior work which deals with selective encryption of DCT based data [8, 13, 16, 25, 34, 52, 53] and [14] are given.

Our realized and tested approach is a different one:
The Hierarchical and progressives modes of JPEG organize the image data in a scalable fashion. Different parts of the image data contribute to different image qualities. This is discussed in chapters 2.2 and 2.3. Our approach uses the grouping of these modes to selectively encrypt parts of the code stream. These parts are encrypted with modern cryptographic ciphers. In our implementation **AES** (details can be found in [11, 30] and [12]) is used. The computational complexity of this procedure is very low, since it can be applied to the encoded bit stream without much parsing and without any transcoding. To avoid marker sequences in the encoded bit stream, which would undermine the code stream compliance, the byte stuffing procedure is used. A JPEG marker always start with two bytes `0xff`. The next two bytes identify the marker. Only `0x00` signals that the previous `0xff` did not belong to a marker. Therefore if during encryption an `0xff` is produced, `0x00` has to be appended in order to avoid generating marker sequences. The result is a mostly code stream compliant encrypted bit stream.

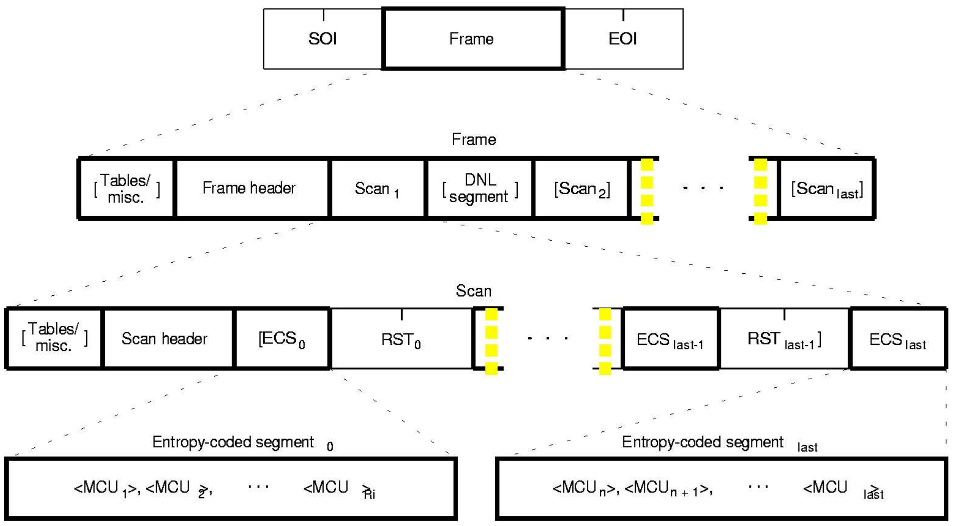


Figure 3.1: Code Stream Syntax [6]

Only the Huffman decoder will very likely complain about too few or too many coefficients in the encrypted scan, since this method does not take care of the correct number of coefficients in a scan. But most decoders are able to decode the encrypted JPEG files (basically all we tried). Figure 3.1 shows the basic JPEG code stream syntax. Progressive JPEG's code stream is organized in **scans** .
A scan can contain data of

- one or more components
- a specific part of the DCT coefficient binary representation
  (see Successive Approximation in chapter 2.2.1)
- a range of DCT coefficient data
  (see Spectral Selection in chapter. 2.2.1)

The DCT coefficient data can consist of the entire DCT coefficient or specific parts of the binary representation of a DCT coefficient. So Spectral Selection and Successive Approximation can be mixed freely.
Hierarchical JPEG's code stream is organized through frames. While in non Hierarchical JPEG files only one frame is present, in Hierarchical JPEG there is one for every layer. These frames are organized in the same way as the standard frame. Therefore Hierarchical JPEG and the other progressive modes can be combined as well.

In the following sections this method of selective encryption of JPEG files will be evaluated for the progressive and the Hierarchical modes of JPEG, but first alternative measures for image quality will be discussed.

3.1 Security Metrics

The most popular measure for image quality is still the well-known PSNR. However, the peak signal compared to the overall off the squared differences, does not fully (up to not at all) comply with human perception of similarity. Two images with different colors, but the same edges and contours, are very likely to be considered similar thus having a very low PSNR. For our purposes, where we want to measure how good our methods hide information, other methods of measuring image quality need to be considered too.

3.1.1 LSS/ESS

LSS (Luminance Similarity Score) and **ESS** (Edge Similarity Score) are two alternative measures for image quality. While the LSS measure is somehow similar interpreted as the PSNR, the ESS is something completely different. It measures the similarity of dominating edges on a block basis. The exact definition is taken from [29].

Definition 3.1.1 (LSS)

$$LSS = \frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i) \tag{3.1}$$

$$f(x_k, y_k) = \begin{cases} 1 & if|x_k - y_k| < \frac{\beta}{2} \\ -\alpha \text{ round}\left(\frac{|x_k - y_k|}{\beta}\right) & otherwise. \end{cases} \tag{3.2}$$

Two equally sized images denoted by x and y are compared on block basis and the images are therefore split up into N equally sized blocks. For each of these blocks the average luminance values, denoted by x_k, y_k with $k \in \{1, \ldots, N\}$, are calculated. If the average luminance values of two blocks do not vary more than $\frac{\beta}{2}$, they are considered equal and f is set to 1. Otherwise f is set to the scaled negative difference. Hence the two parameters α and β control the sensitivity of the score. The score of two similar images will always be 1, while the minimum, which is reached for two images with maximal numerical distance in each point (black and white), is $-\alpha\frac{max\_color}{\beta}$. In our tests, if not explicitly stated otherwise, α and β are set to 0.1 and 3. Hence the LSS of our 8 bit imagery is in the range of 1 to -8.5.

Definition 3.1.2 (ESS)

$$ESS = \frac{\sum_{i=1}^{N} w(e_{x_i}, e_{y_i})}{\sum_{i=1}^{N} c(e_{x_i}, e_{y_i})} \tag{3.3}$$

$$w(e_{x_b}, e_{y_b}) = \begin{cases} 0 & if\ e_{x_b} = 0\ or\ e_{y_b} = 0 \\ |\cos(\phi(e_{x_b}) - \phi(e_{y_b})| & otherwise. \end{cases} \tag{3.4}$$

$$c(e_{x_b}, e_{y_b}) = \begin{cases} 0 & if\ e_{x_b} = 0\ and\ e_{y_b} = 0 \\ 1 & otherwise. \end{cases} \tag{3.5}$$

The edge similarity score measures the degree of resemblance of the edge and the contour information between two images. The original image is split up in blocks and for each of the blocks a dominant edge is detected. The edges are detected with the Sobel operator. The dominant edge is quantized into one of eight equally spaced directions. These directions are indexed from one to eight, a zero is assigned if no edge was detected (the index of the dominant edge of the i-th block of an image x is denoted by e_{x_i}). For the dividend in 3.3 the sum of the absolute values of the cosine of the difference of the two angels ($\phi(e_{x_b})$ and $\phi(e_{y_b})$) is computed. This difference only computed when both of the blocks contain edge information, otherwise w is set to zero. The divisor in 3.3 is to scale the ESS into $[0, 1]$. A value near zero indicates no edge similarity, while a near one indicates high similarity.

For both measures different block sizes might lead to different results.

3.2 Replacement Attack

Considering that we use cryptographic secure cipher, attacks against the cipher are unfeasible. Nevertheless the image quality of the direct reconstruction of the encrypted image must not be considered as the best quality an attacker can achieve. Identifying encrypted data is quite easy due to its statistical deviation from unencrypted data. This leads to a rather powerful attack against selective encryption, the replacement of the encrypted parts through average gray scaled image data. This method leads to a significant improvement of image quality, since most of the noise induced by encryption is removed. It can be considered an upper bound of image quality an attacker can achieve. If a whole scan is encrypted an replacement attack is quite easy to conduct. One simply has to set the whole scan to average gray. In case of libjpeg it is sufficient to leave the whole scan empty. If parts of a scan are encrypted, the replacement attack has to be more elaborate. The partially encrypted encoded coefficient data, has to be scanned for deviations to the statistical properties of encoded coefficient data and violations against Huffman encoding rules. Most likely are too many codewords or too few. Having identified the encrypted parts they have to be replaced with the corresponding data for an average gray pixel.

3.3 Selective Encryption of Progressive JPEG

In this chapter selective encryption of progressive JPEG will be evaluated. Metrics for image quality are LSS,ESS and the PSNR. Furthermore for every encryption method the appropriate replacement attack will be given.

3.3.1 Successive Approximation

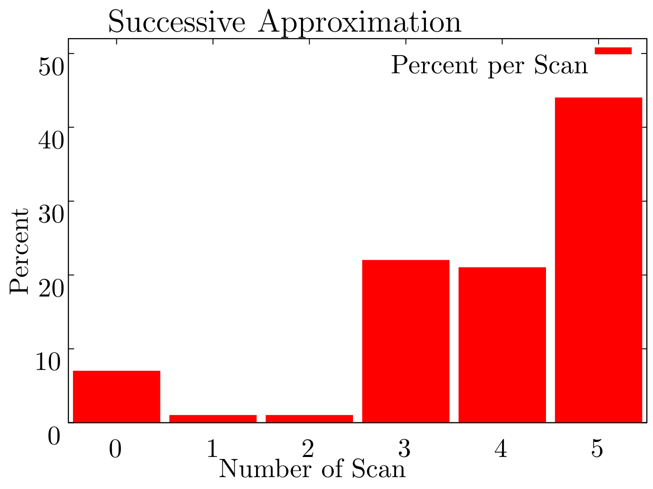


Figure 3.2: Successive Approximation: Distribution of Data

Our investigations on encryption of progressive JPEG using successive approximation will often encrypt whole scans. Therefore an overview of the amount of data per scan is given in figure 3.2. To get an impression how much and which image information is given in a specific scan figure 3.5 shows the reconstruction of the original image based on a specific scan. Figure 3.3 shows the reconstructed images of selectively encrypted JPEG files. The first image (in the top left corner) is the result of encrypting the first scan, which are first six bits all DC coefficients. The second image of the first row is the result of encrypting the second scan,which is the next bit of all DC coefficients. The scan encrypted is always indicated on top of each picture. The image metrics PSNR, LSS and ESS are given below each picture. LSS and ESS are computed with a block size of eight. Table 3.1 shows the result for LSS and ESS with block sizes four,eight and sixteen.

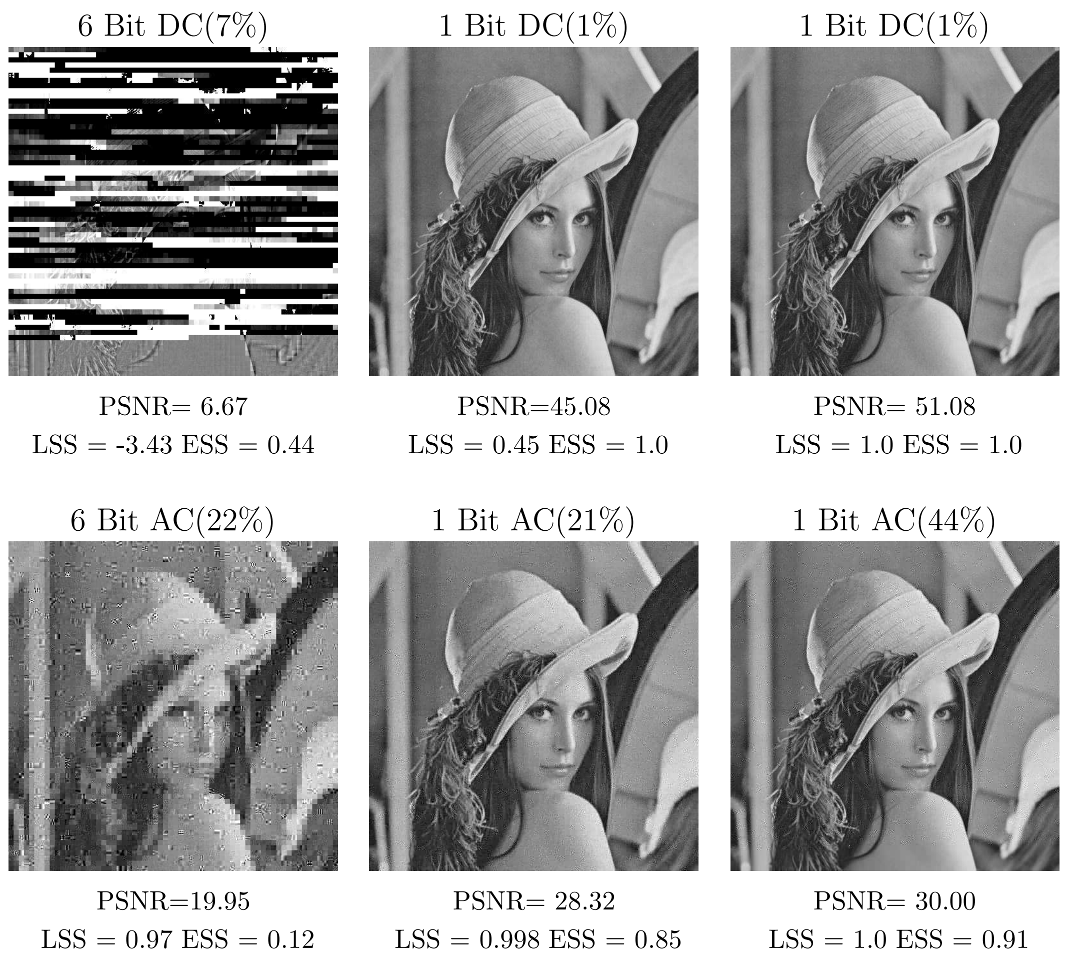


Figure 3.3: Encryption with Successive Approximation

Blocksize	LSS SA 0	ESS SA 0	LSS SA 1	ESS SA 1	LSS SA 2	ESS SA 2
4 pixel	-3.44	0.42	0.45	1.00	1.0	1.0
8 pixel	-3.43	0.44	0.45	1.00	1.0	1.0
16 Pixel	-2.46	0.20	0.92	0.97	1.0	1.0

Blocksize	LSS SA_3	ESS SA 3	LSS SA 4	ESS SA 4	LSS SA 5	ESS SA 5
4 Pixel	-0.16	0.10	0.42	0.55	0.76	0.72
8 Pixel	0.97	0.12	1.0	0.85	1.0	0.91
16 Pixel	0.97	0.12	1.0	0.85	1.0	0.91

Table 3.1: Different Blocksizes for LSS and ESS for Encryption with Successive Approximation

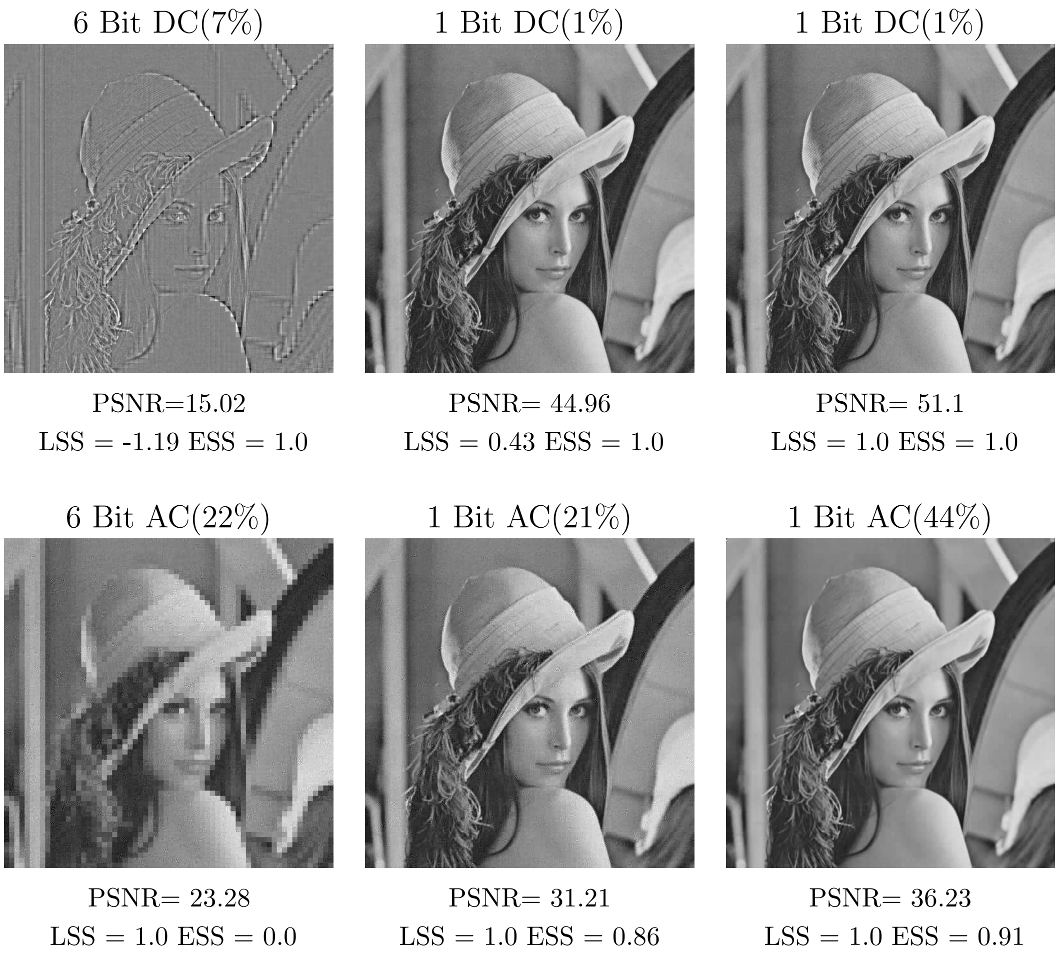


Figure 3.4: Replacement Attack against Encryption with Successive Approximation

Blocksize	LSS SA 0	ESS SA 0	LSS SA 1	ESS SA 1	LSS SA 2	ESS SA 2
4 pixel	-1.20	1.00	0.43	1.00	1.00	1.00
8 pixel	-1.20	1.00	0.43	1.00	1.00	1.00
16 Pixel	-1.14	0.44	0.62	0.97	1.00	0.99

Blocksize	LSS SA_3	ESS SA 3	LSS SA 4	ESS SA 4	LSS SA 5	ESS SA 5
4 Pixel	0.04	0.02	0.50	0.63	0.88	0.79
8 Pixel	1.00	0.00	1.00	0.86	1.00	0.91
16 Pixel	1.00	0.64	1.00	0.94	1.00	0.96

Table 3.2: Different Blocksizes for LSS and ESS: Replacement Attack against Encryption with Successive Approximation

The position in the table is the same as in the figure. Hence "LSS SA 0" are the LSS values for the image resulting from first six bit of DC coefficients encrypted, "LSS SA 1" for the next DC bit encrypted and similarly for the rest. DC coeffi-

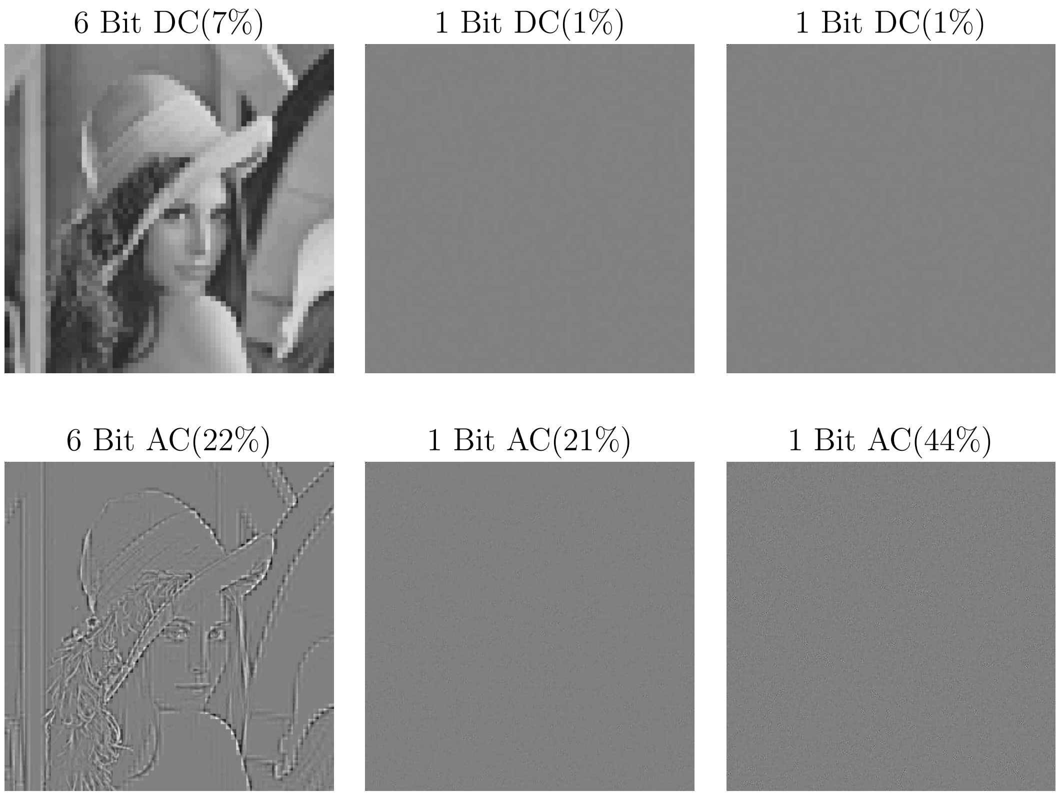


Figure 3.5: Image Information of Successive Approximation Scans

cients do contain the most image information, also serious distortion is generated by encrypting the first six bit of AC data.

The encryption of the first additional bit of the DC coefficients leads to some loss of luminance information, but to no severe distortion for human visual perception. The second additional bit does not even have an impact on the image quality metrics. Furthermore the additional bits of the AC coefficients have very limited influence on the image quality, despite their high percentage on overall data.

Figure 3.4 shows the result of a replacement attack against the encrypted images in figure 3.4. The LSS and ESS for different block sizes are given in table 3.2 The position in the table corresponds to the position in figure 3.4 similarly as in table 3.1 Generally the reduction of noise due to the replacement attack is both perceived and measured. The biggest improvement is achieved when replacing the encrypted 6 bit DC coefficients. While the replacement of the 6 bit AC coefficient visually reduces noise, the ESS indicates lower image quality. After encrypting the 6 bit AC data nearly no edge information is any longer present. As it turns out, random edges do significantly perform better than nearly no edges.

Successive Increasing Encryption

Instead of encrypting whole scans it is also possible to encrypt just parts of a scan. We evaluated the PSNR, LSS and ESS of the image of Lena starting at the beginning of the coefficient data and successively increasing the amount of data encrypted. Additionally an appropriate replacement attack was conducted.

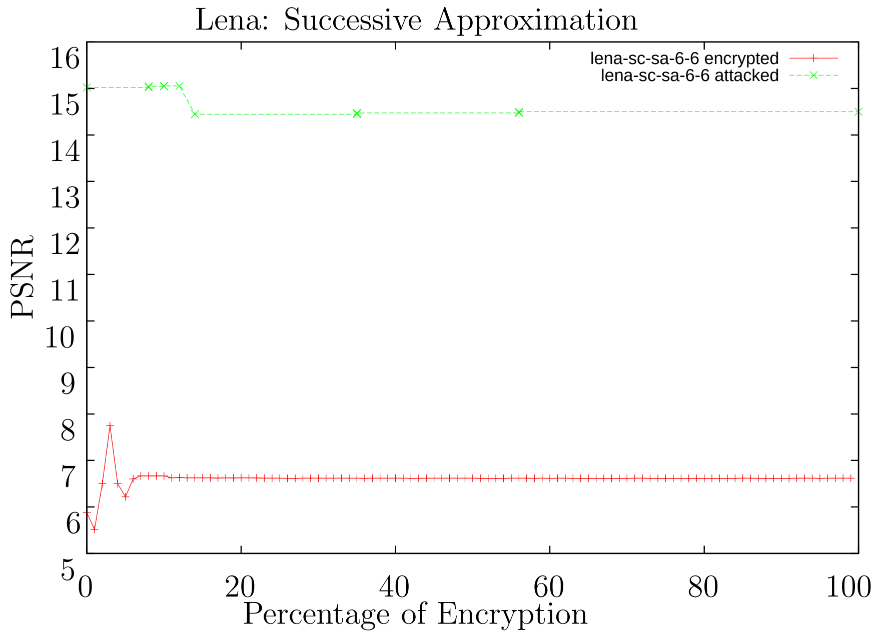


Figure 3.6: Successive Approximation: PSNR of Encrypted

In figure 3.6 the resulting PSNR for encryption and the corresponding attack is shown in dependence of the percentage encrypted or replaced. The better PSNR of the replacement attack is simply explained: The average color of the original image is 123.7 (computed with `imgcmp -m mae -f lena.jpg -F black.jpg`), therefore if we replace every encrypted coefficient with 128 this PSNR will result in a higher PSNR value. Nevertheless this can not be taken as significant quality improvement. The resulting LSS and ESS plots are presented in figure 3.7. Both metrics clearly show that at least after encrypting the DC coefficient data all significant color information is lost. Nevertheless to hide all of the edge and contour information the first 6 bits of the AC coefficients have to be encrypted too.

Encryption at Different Locations

Figures 3.8 and 3.9 show the results of encrypting one percent of coefficient data at different locations in the file. A replacement attack was not conducted, since this method does not operate on whole scans but on parts of scans. Nevertheless a replacement attack would be possible, but much harder to conduct (see section

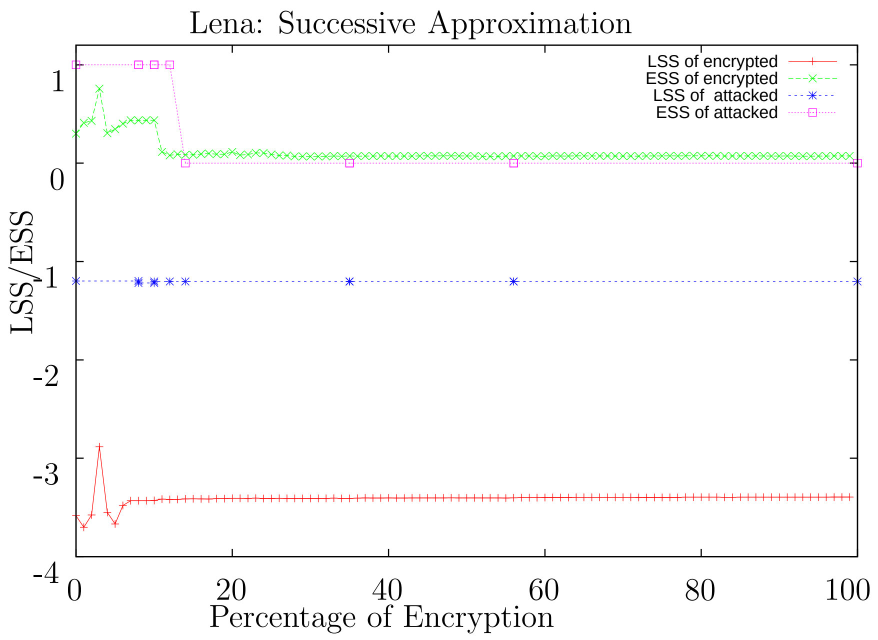


Figure 3.7: Successive Approximation: LSS/ESS of Encrypted

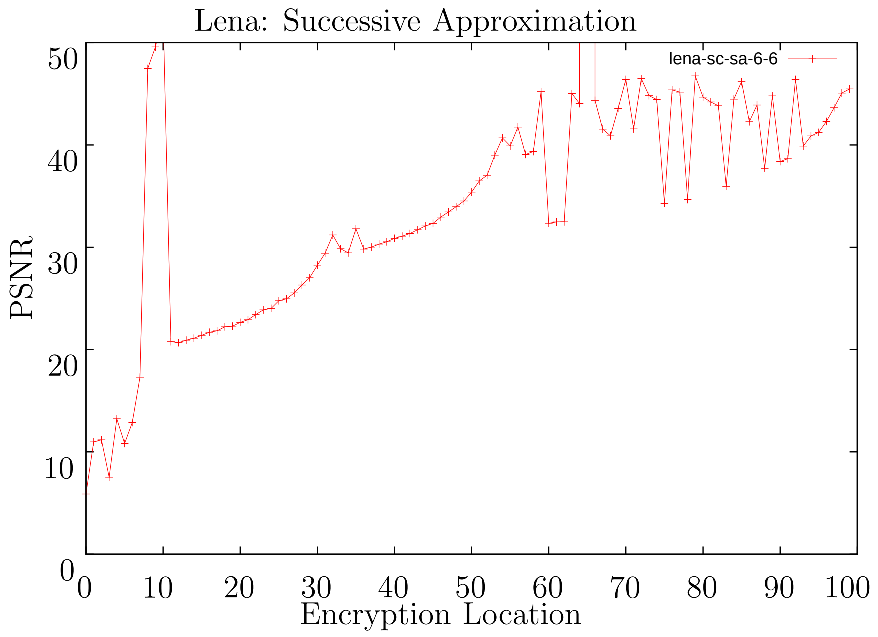


Figure 3.8: Successive Approximation: PSNR for Different Encryption Locations

3.2). However, the results of a replacement attack can be roughly estimated through the replacement of the whole scan containing the one percent encrypted (see figure 3.4) and the original image. The results show that encryption at the

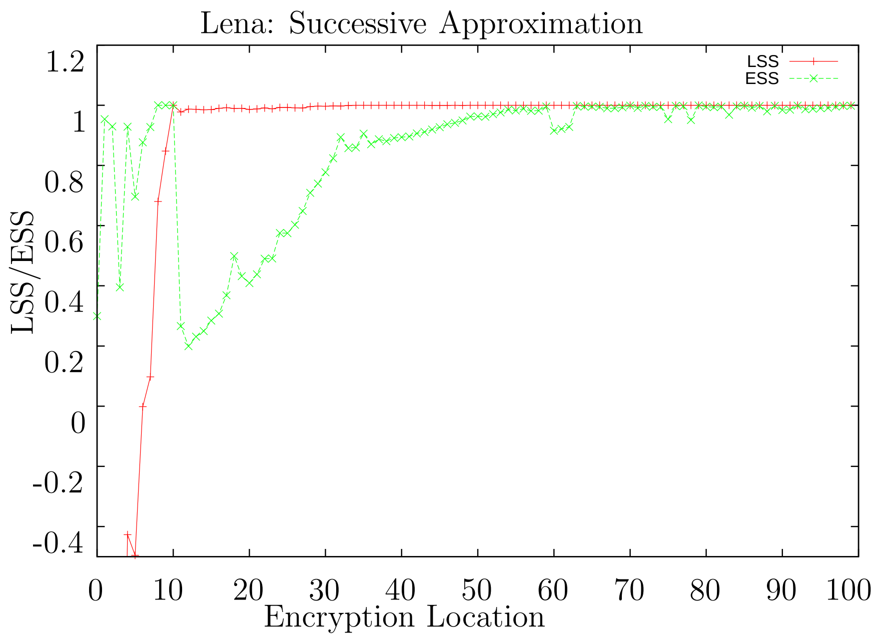


Figure 3.9: Successive Approximation: LSS/ESS for Different Encryption Locations

beginning of the data has much more influence on the image quality than at the end. A special case are the two one bit DC scans, which do not significantly influence the image quality. It can be seen at eight to ten percent in figures 3.9 and 3.8 that all three metrics indicate a significant improvement of image quality in this area.

3.3.2 Spectral Selection

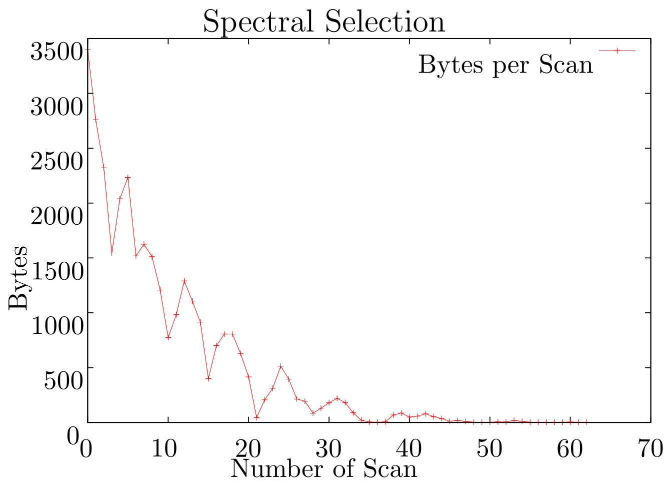


Figure 3.10: Spectral Selection: Distribution of Data

Similar investigations as about encryption with progressive JPEG using successive approximation were conducted about encryption with spectral selection. Figure 3.10 shows the distribution of encoded coefficient data per scan. Due to the standard quantization table the coefficient data decreases in zig-zag scan sequence. In a scan all coefficients of a certain position in the zig-zag scan are encoded. Basically the DC coefficients contain all color information, while the AC coefficients contain the edge and contour information (see figure 3.11). For blocksize 16 the LSS is 1.0 and ESS is 0.63 for the reconstruction based on DC coefficients. For the reconstruction based on the AC coefficients the LSS is -1.14 and the ESS is 0.44. The first seven AC coefficients contain over 50% of the overall AC coefficient data (in the Lena image). Figure 3.12 shows reconstructions based on the first 50% and the last 50% of coefficient data. The PSNR and luminance values are not of interest because no color information is present, but ESS is of interest. These values indicate that more edge and contour information is located in the first 50% of AC coefficient data. Nonetheless the second 50% percent contain enough image information to guess the original content of the image. Furthermore selected scans were encrypted and the corresponding replacement attack conducted. Figure 3.13 shows the results and table 3.3 gives additional information for the LSS and ESS evaluated with different block sizes. The position in the table corresponds to the position in the figure. Starting in the top left corner "LSS SS 0" and "ESS SS 0" are additional values for the reconstructed image where all DC coefficients (scan 0) have been encrypted and ending in the right corner at the bottom with "LSS SS 5" and "ESS SS 5" adding values for the reconstructed image with all AC coefficients at position 20 in zig-zag scan encrypted.

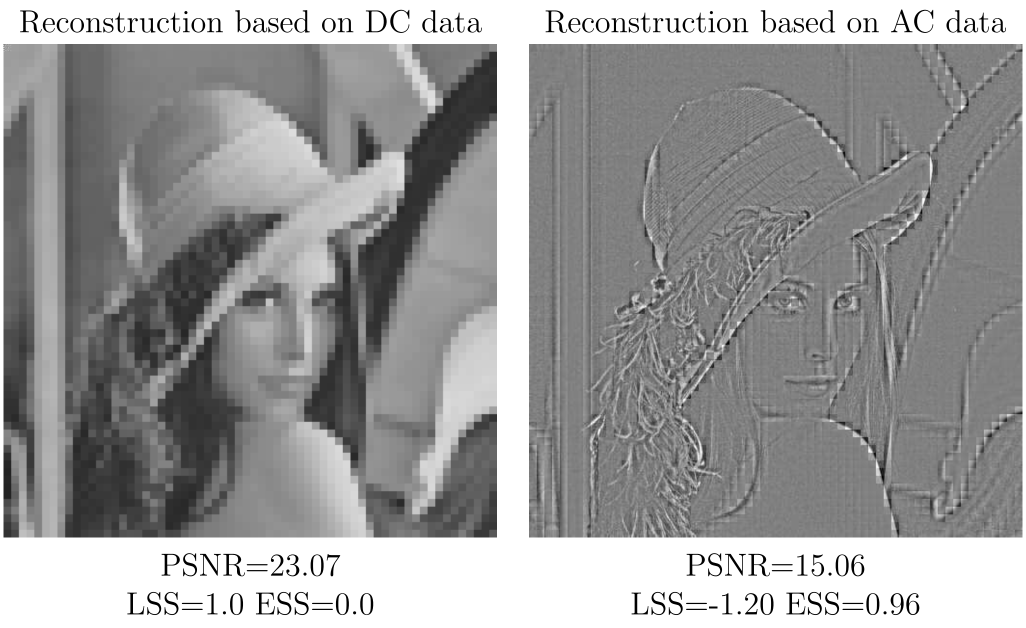

Reconstruction based on DC data
Reconstruction based on AC data
PSNR=23.07
LSS=1.0 ESS=0.0
PSNR=15.06
LSS=-1.20 ESS=0.96

Figure 3.11: Reconstructions Based on DC and AC Coefficient Data

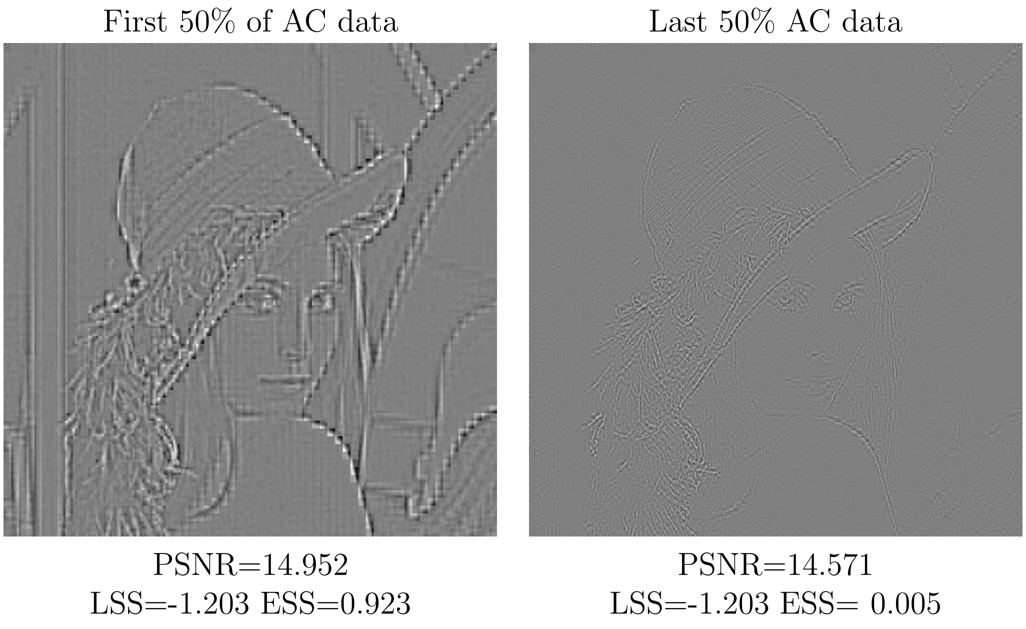


Figure 3.12: Reconstructions Based on AC Coefficient Data

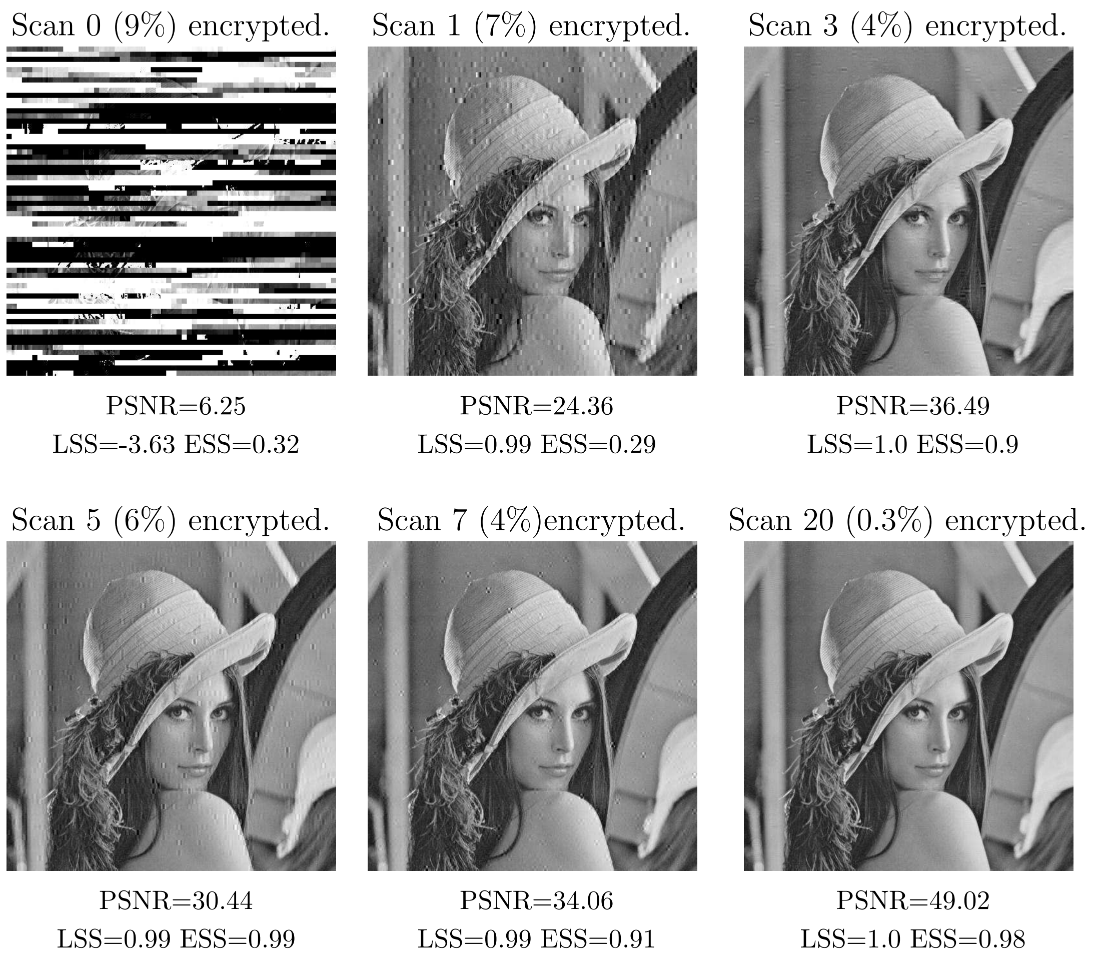


Figure 3.13: Encryption with Spectral Selection

Blocksize	LSS SS 0	ESS SS 0	LSS SS 1	ESS SS 1	LSS SS 2	ESS SS 2
4 pixel	-3.67	0.31	-0.06	0.48	1.00	0.72
8 pixel	-3.64	0.33	0.99	0.30	1.00	1.00
16 Pixel	-2.33	0.15	0.99	0.64	1.00	0.96

Blocksize	LSS SS 3	ESS SS 3	LSS SS 4	ESS SS 4	LSS SS 5	ESS SS 5
4 Pixel	1.00	0.45	1.00	0.64	1.00	0.98
8 Pixel	1.00	0.99	1.00	0.91	1.00	0.99
16 Pixel	1.00	0.91	1.00	0.99	1.00	1.00

Table 3.3: Different Blocksizes for LSS and ESS: Spectral Selection

The results clearly show the decreasing influence of the AC coefficients.

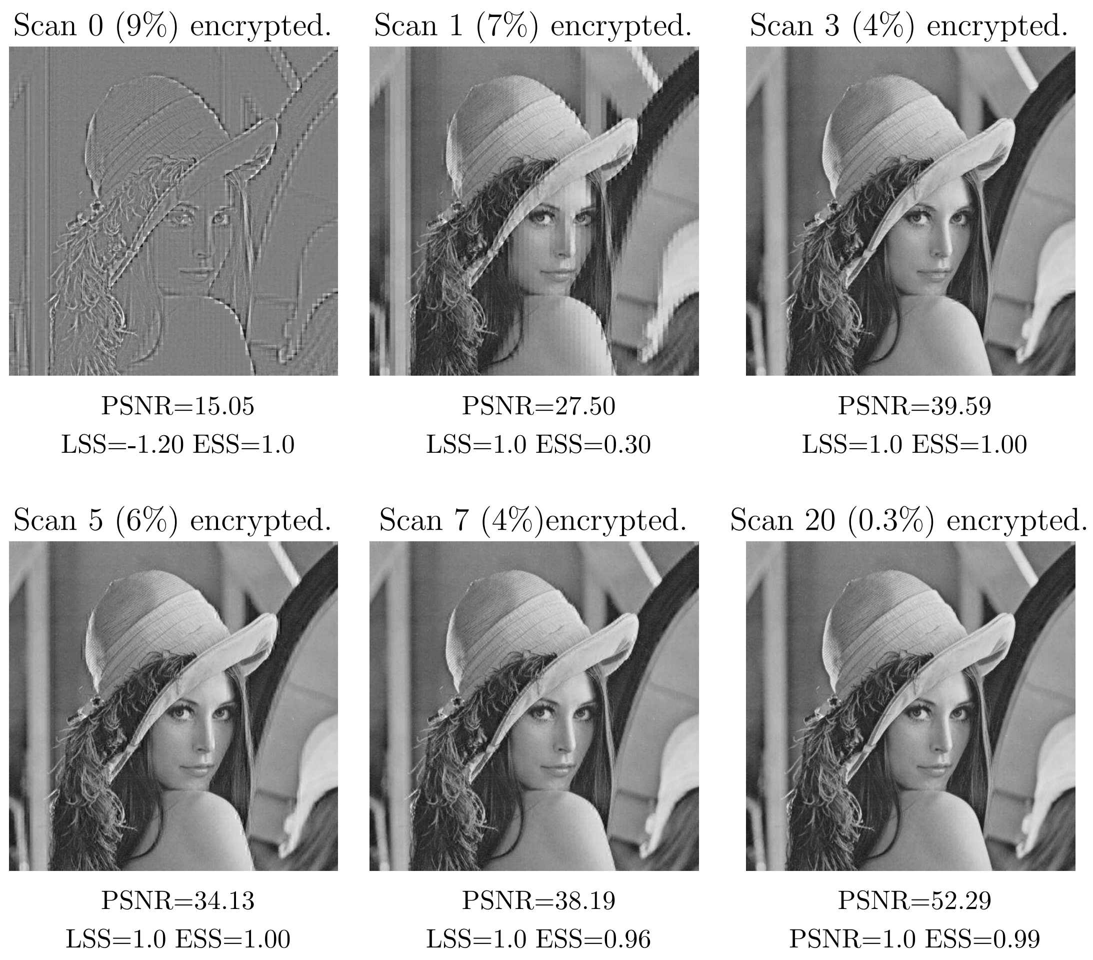


Figure 3.14: Replacement Attack against Encryption with Spectral Selection

Blocksize	LSS SS 0	ESS SS 0	LSS SS 1	ESS SS 1	LSS SS 2	ESS SS 2
4 pixel	-1.20	1.00	0.23	0.62	1.00	0.83
8 pixel	-1.20	1.00	1.00	0.30	1.00	1.00
16 Pixel	-1.15	0.43	1.00	0.74	1.00	0.98
Blocksize	**LSS SS 3**	**ESS SS 3**	**LSS SS 4**	**ESS SS 4**	**LSS SS 5**	**ESS SS 5**
4 Pixel	1.00	0.63	1.00	0.79	1.00	0.98
8 Pixel	1.00	1.00	1.00	0.96	1.00	0.99
16 Pixel	1.00	0.94	1.00	0.99	1.00	1.00

Table 3.4: Different Blocksizes for LSS and ESS: Replacement Attack with Spectral Selection

Figure 3.14 and table 3.4 show the results for the corresponding replacement attacks.

Successive Increasing Encryption

Similarly to the procedure with successive approximation, more and more of the JPEG file was encrypted and the corresponding PSNR, LSS and ESS evaluated. Once again it turns out that the PSNR is often inappropriate metric for encrypted visual data. Since the DC coefficients contain most of the color information and those are encrypted first, the PSNR and LSS are indicating constantly poor image quality. Therefore the ESS values are of superior interest. They show a significant improvement of the image quality through a replacement attack. The ESS graph would indicate that about 26% of the coefficient data has to be encrypted in order that nearly no edges match. However this has to be interpreted carefully, because as shown in figure 3.12 a reconstruction based on the last 50% of AC coefficient data has low ESS value, but does contain enough image information that a human can perceive the content.

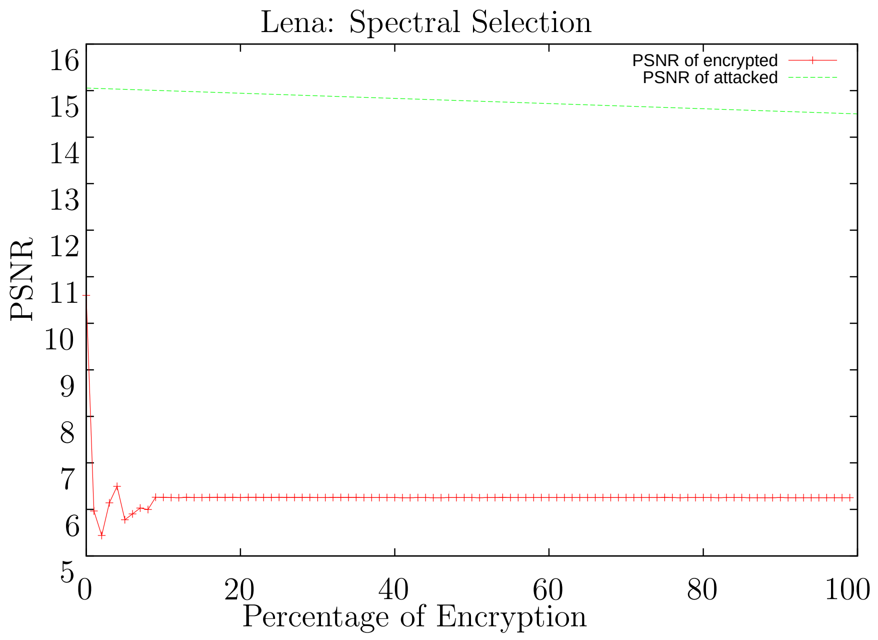


Figure 3.15: Spectral Selection: PSNR of Encrypted

Encryption at Different Location

Similarly to the procedure applied to successive approximation, one percent of the file is encrypted and only the location varies. The results (see figures 3.17 and 3.18) show that the visually more important parts are located at the beginning.

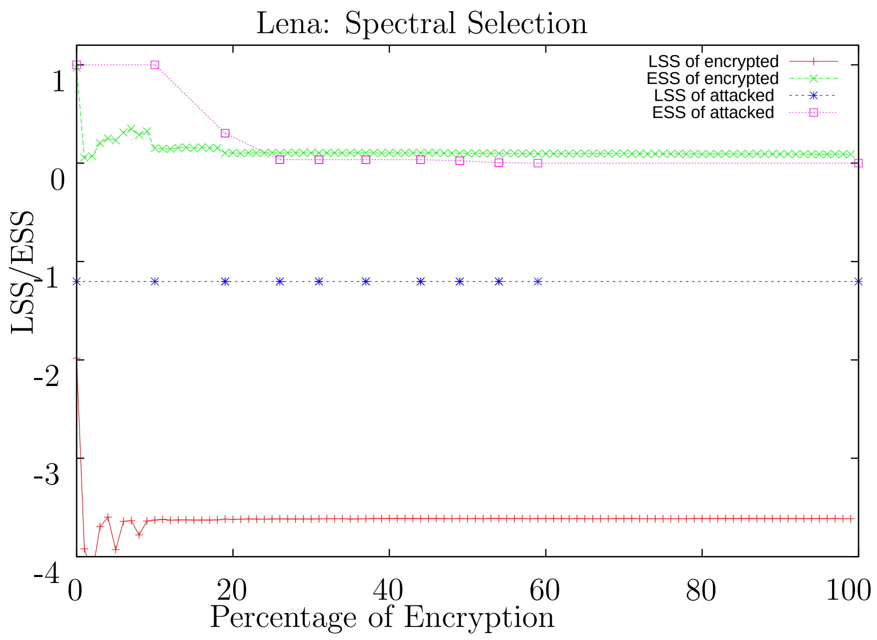


Figure 3.16: Spectral Selection: LSS/ESS of Encrypted

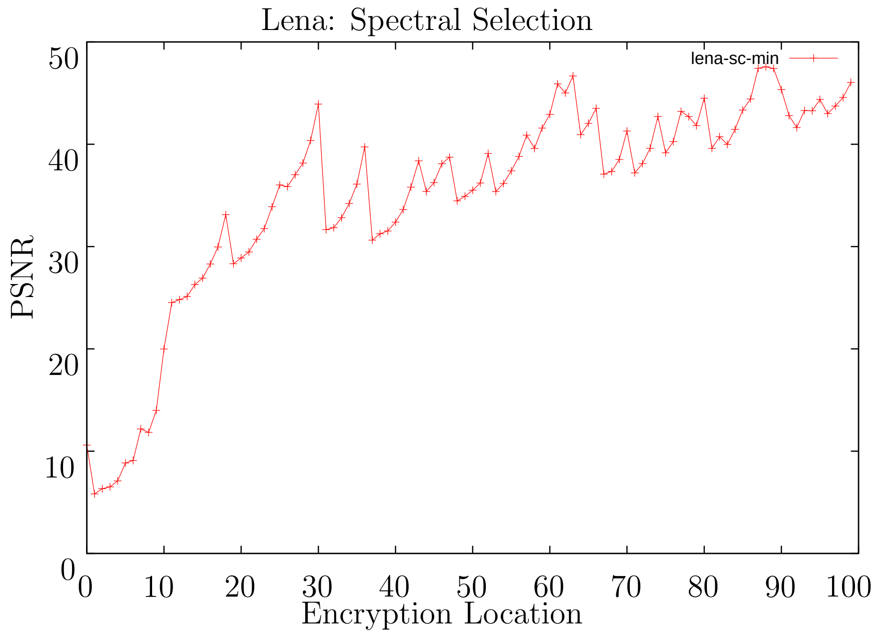


Figure 3.17: Spectral Selection: PSNR for Different Encryption Locations

3.4 Selective Encryption of Hierarchical JPEG

The Hierarchical JPEG coding procedure's main parameter is the number of layers. Therefore the selective encryption of Hierarchical JPEG is discussed on

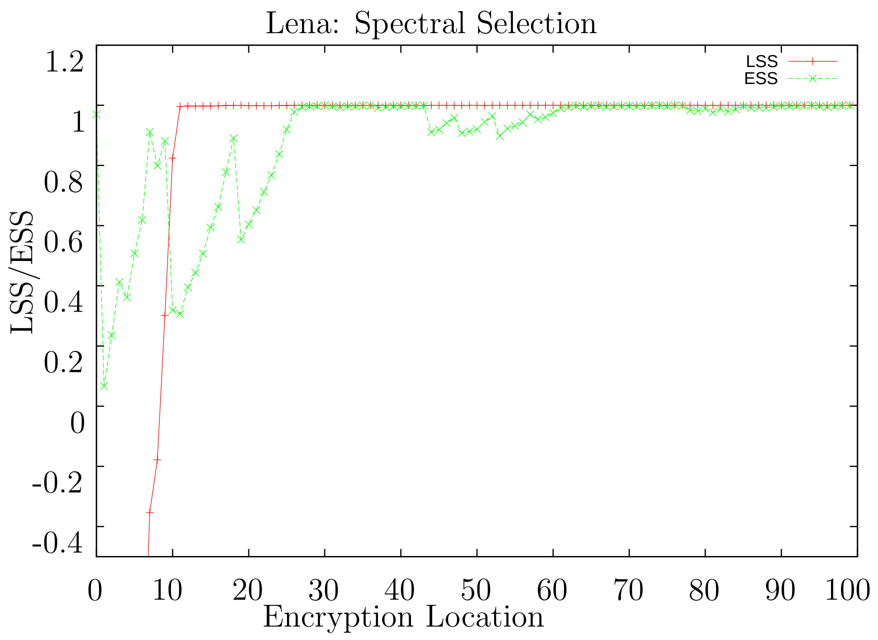


Figure 3.18: Spectral Selection: LSS/ESS for Different Encryption Locations

the basis of the number of layers. Since the coding performance is better for a lower number of layers, Hierarchical JPEG with two and three layers are discussed separately. In order to discuss Hierarchical JPEG with more layers the case of six layers is presented. The base layer of 512x512 test picture encoded is just 16x16 pixel for Hierarchical JPEG with 6 layers. Encrypting the enhancement layers will result in a subsample of the original image plus the noise of the encrypted layers. This noise can easily be replaced by ignoring the encrypted enhancement layers, which basically is a replacement attack. But in the case of Hierarchical JPEG the replacement attack is very easy to apply, since every layer is encoded in a frame, which is basically an ordinary JPEG file without start of image and end of image markers.

3.4.1 Encrypting Hierarchical JPEG with Two Layers

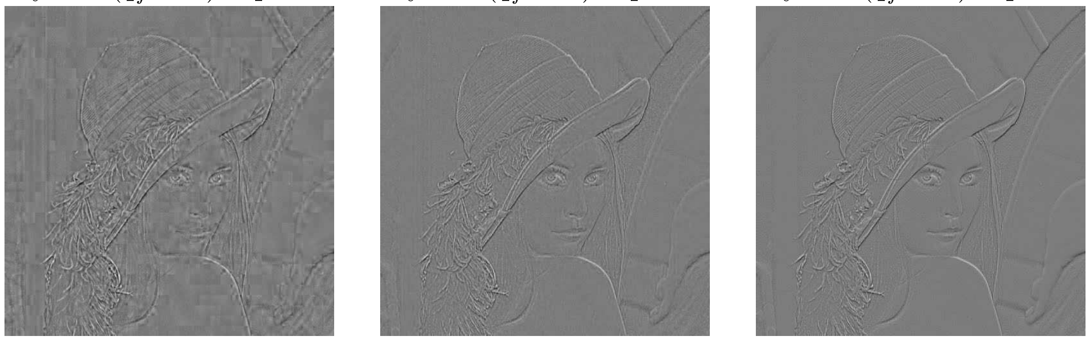
Figure 3.19 shows the reconstruction of a Hierarchical JPEG file with two layers and the base layer encrypted. The image is unrecognizable, but as figure 3.20 shows a replacement attack reveals much of the edge information. Furthermore the replacement attack is conducted for different quality settings of the encrypted base layer. The better the quality settings of the encrypted base layer has been, the better the reconstructions look. Interestingly in this case the ESS values do not correspond with human perception and indicate higher quality for the reconstructions with lower quality settings for the base layer. Most of the luminance information is contained in the base layer.

Layer 0 encrypted

PSNR=6.73
LSS=-3.41 ESS=0.18

Figure 3.19: Encryption of Hierarchical JPEG with 2 Layers

Layer 0 (q_f=10) replaced | Layer 0 (q_f=55) replaced | Layer 0 (q_f=95) replaced



PSNR=14.73
LSS=-1.22 ESS=0.29

PSNR=14.74
LSS=-1.21 ESS=0.16

PSNR=14.73
LSS=-1.21 ESS= 0.14

Figure 3.20: Replacement Attack Against Encryption of Hierarchical JPEG with 2 Layers

3.4.2 Encrypting Hierarchical JPEG with Three Layers

Figure 3.21 shows reconstructions of Hierarchical JPEG with three layers. The corresponding replacement attacks are given in figure 3.22. Most of the luminance information is contained in the base layer, while edge and contour information is contained in every enhancement layer. Therefore the PSNR and LSS values are very low if the base layer is encrypted.

3.4.3 Encrypting Hierarchical JPEG with Six Layers

When encrypting Hierarchical JPEG with 6 layers one has plenty of possibilities. However, not all can be presented. Nevertheless tables 3.5, 3.6,3.7 and 3.8 give a rather complete overview. The first two summarize the results for Hierarchical

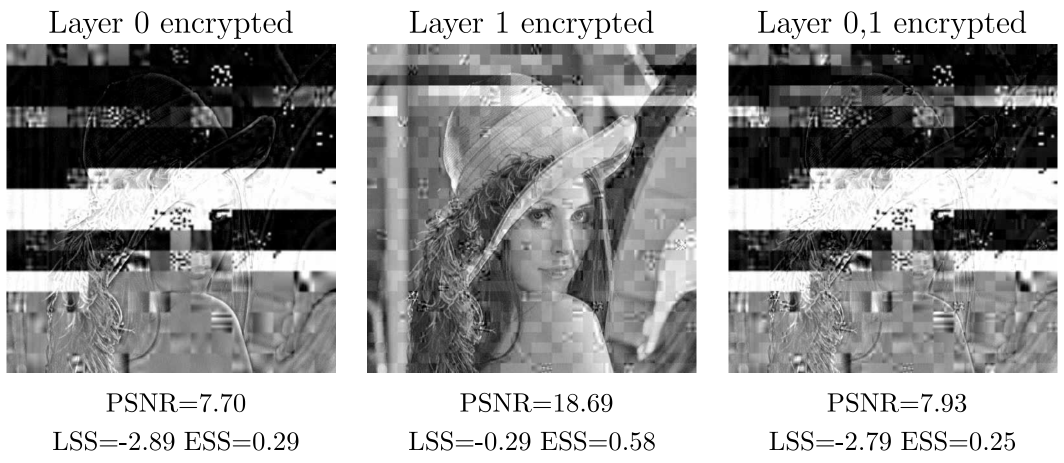


Figure 3.21: Encryption of Hierarchical JPEG with 3 Layers

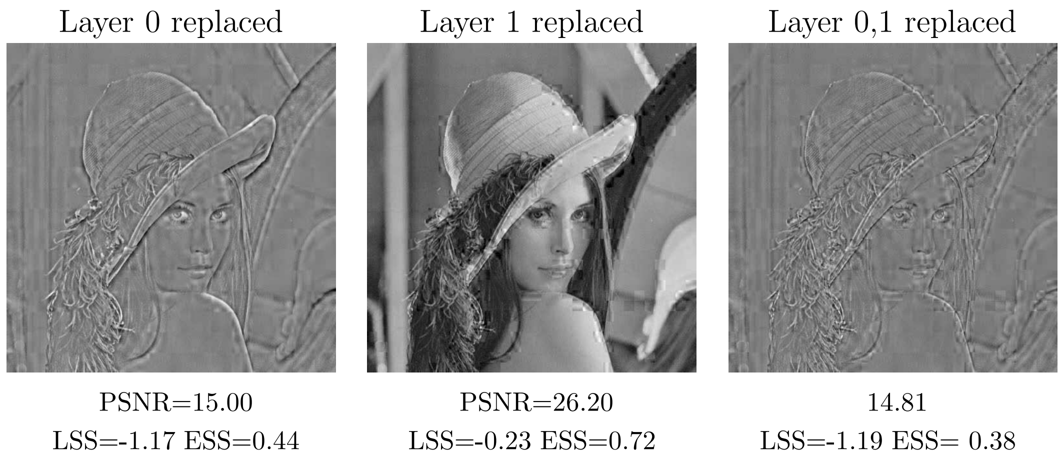


Figure 3.22: Replacement Attack Against Encryption of Hierarchical JPEG with 3 Layers

JPEG optimized for compression ($q_f = 10$ for the base layer, α is set to 140 for all but the last enhancment layer where α is set to 10), while the next two give an overview for good quality enhancment layers ($q_f = 75$ for the base layer, α is set to 10 for all other layers). The images presented are all based on Hierarchical JPEG optimized for good compression. It seems appropriate to start encrypting at the base layer and continue up to the penultimate layer. Additionally the last enhancement layer is encrypted. The results are shown in figure 3.23. The corresponding replacement attack is presented in figure 3.24. Once more the replacement attack proves to be a powerful tool to reconstruct encrypted images at higher quality. The luminance information is contained in the first two layers. The first four layers have to be encrypted to achieve that the reconstructed image is unrecognizable. Nevertheless the replacement attacks show that there is perceivable edge and contour information even if encoding up to the penultimate layer. Replacing the ultimate layer reveals a low quality subsample of the original.

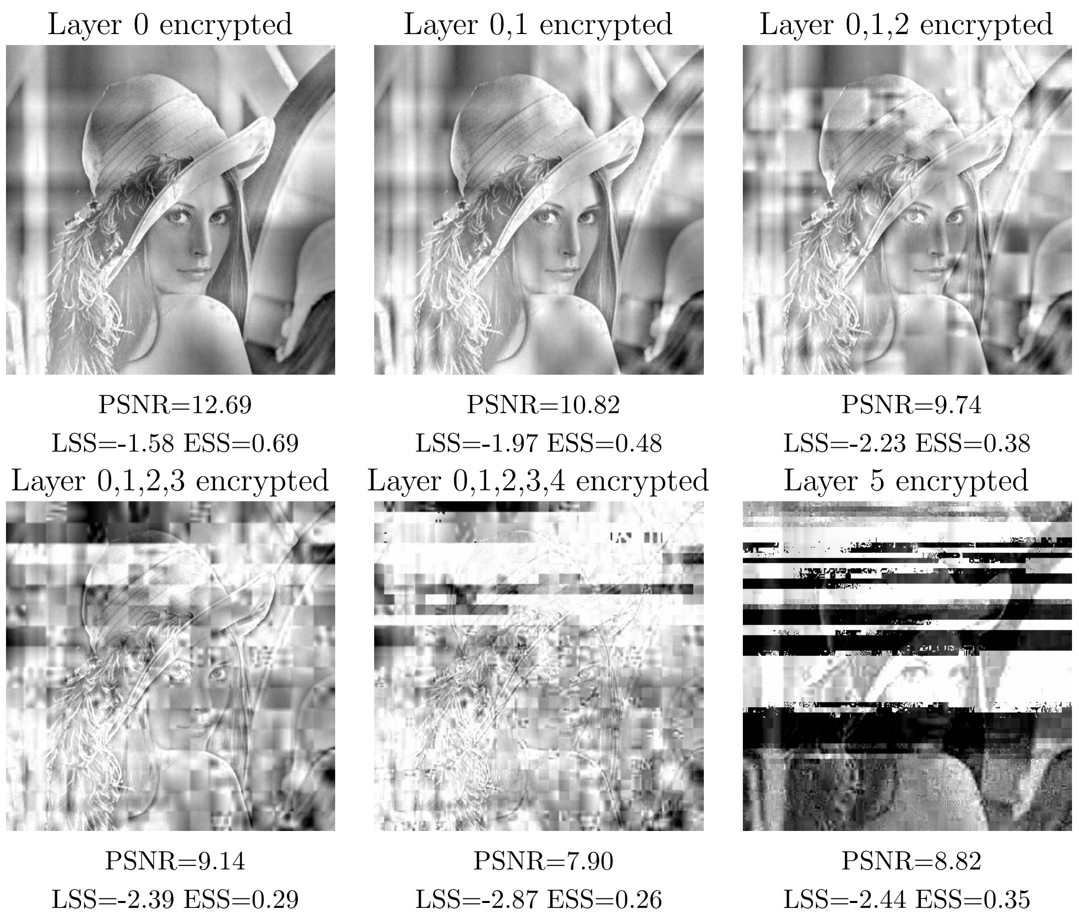


Figure 3.23: Encryption of Hierarchical JPEG with 6 Layers

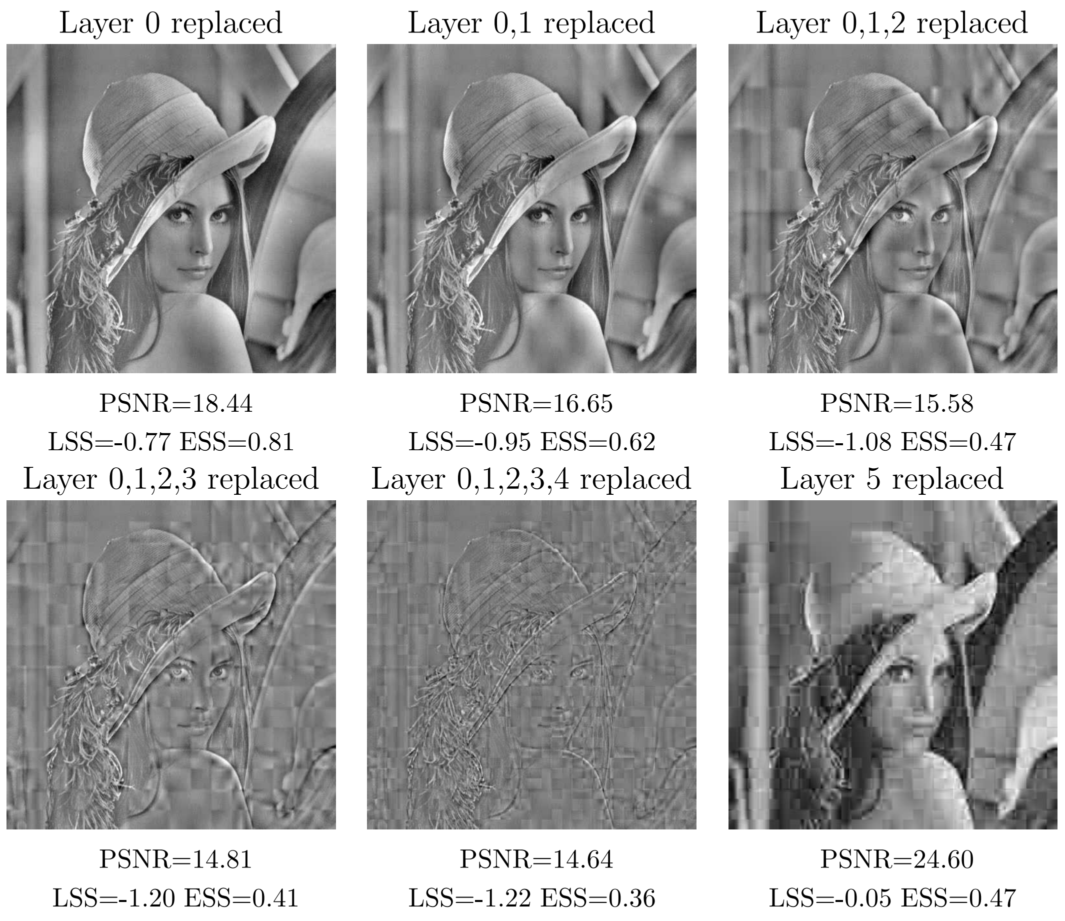


Figure 3.24: Replacement Attack against Encryption of Hierarchical JPEG with 6 Layers

3.5 Application Scenarios

Selective encryption might be considered for various application related reasons. Nonetheless the needs of the applications can be separated in those which have to confidentially hide most or all image information and those cases in which one can accept that some information can be retrieved from the encrypted material. In some cases even a low quality version should become public while the high quality version is only available to those who can decrypt it.

3.5.1 Confidentiality

In this section the possibilities for using selective encryption of JPEG data for ensuring visual data confidential encryption will be discussed. Basically to fully ensure visual data confidential encryption is quite impossible without encrypting nearly all of the file.

Layers encrypted	0	1	2	3	4	5	0,1	1,2	2,3	3,4	4,5
PSNR direct	12.69	18.42	17.85	11.82	13.78	8.82	10.82	15.53	11.48	13.84	8.05
PSNR attacked	18.44	19.45	21.00	21.87	22.90	24.60	16.65	18.10	19.12	20.04	18.93
LSS direct	-1.58	-0.68	-0.74	-1.51	-0.98	-2.44	-1.97	-1.05	-1.73	-1.16	-2.89
LSS attacked	-0.77	-0.56	-0.34	-0.19	-0.38	-0.05	-0.95	-0.73	-0.57	-0.50	-0.70
ESS direct	0.69	0.56	0.45	0.39	0.42	0.35	0.48	0.39	0.33	0.31	0.28
ESS attacked	0.81	0.61	0.53	0.49	0.57	0.47	0.62	0.45	0.41	0.43	0.39

Table 3.5: The Results for Hierachical JPEG with 6 Layers: Optimized for Compression

Layers encrypted	0,1,2	1,2,3	2,3,4	3,4,5	0,1,2,3,	1,2,3,4	2,3,4,5	0,1,2,3,4,	1,2,3,4,5
PSNR direct	9.74	10.79	12.86	7.93	9.14	11.86	7.80	7.90	7.66
PSNR attacked	15.58	16.84	18.01	14.06	14.81	16.15	14.93	14.64	14.87
LSS direct	-2.23	-1.93	-1.41	-2.90	-2.39	-1.62	-2.93	-2.87	-2.96
LSS attacked	-1.08	-0.86	-0.70	-1.20	-1.20	-0.92	-1.09	-1.22	-1.11
ESS direct	0.38	0.31	0.28	0.23	0.29	0.27	0.21	0.26	0.20
ESS attacked	0.47	0.40	0.35	0.14	0.41	0.33	0.02	0.36	0.00

Table 3.6: The Results for Hierachical JPEG with 6 Layers: Optimized for Compression (2)

Layers encrypted	0	1	2	3	4	5	0,1	1,2	2,3	3,4	4,5
PSNR direct	16.65	20.52	20.03	14.54	8.51	8.80	14.51	16.71	12.88	6.96	8.98
PSNR attacked	18.42	22.92	25.06	26.90	26.64	23.94	16.53	19.41	21.33	22.46	20.00
LSS direct	-0.98	-0.54	-0.52	-1.21	-2.88	-2.36	-1.26	-0.88	-1.44	-3.43	-2.24
LSS attacked	-0.74	-0.34	-0.12	0.08	-0.22	-0.46	-0.98	-0.57	-0.30	-0.33	-0.70
ESS direct	0.64	0.51	0.40	0.43	0.47	0.53	0.43	0.33	0.30	0.23	0.30
ESS attacked	0.77	0.66	0.62	0.67	0.68	0.79	0.60	0.54	0.48	0.50	0.59

Table 3.7: The Results for Hierachical JPEG with 6 Layers: High Quality Layers

Layers encrypted	0,1,2	1,2,3	2,3,4	3,4,5	0,1,2,3	1,2,3,4	2,3,4,5	0,1,2,3,4,	1,2,3,4,5
PSNR direct	12.98	11.78	6.68	8.88	9.91	6.63	8.64	6.44	8.51
PSNR attacked	15.54	17.94	19.38	18.54	14.99	16.97	16.95	14.74	15.44
LSS direct	-1.47	-1.68	-3.46	-2.25	-2.15	-3.55	-2.36	-3.0	-2.44
LSS attacked	-1.10	-0.70	-0.51	-0.76	-1.18	-0.80	-0.87	-1.21	-1.01
ESS direct	0.31	0.27	0.18	0.18	0.25	0.15	0.14	0.13	0.12
ESS attacked	0.51	0.43	0.31	0.43	0.40	0.19	0.21	0.15	0.00

Table 3.8: The Results for Hierachical JPEG with 6 Layers: High Quality Layers (2)

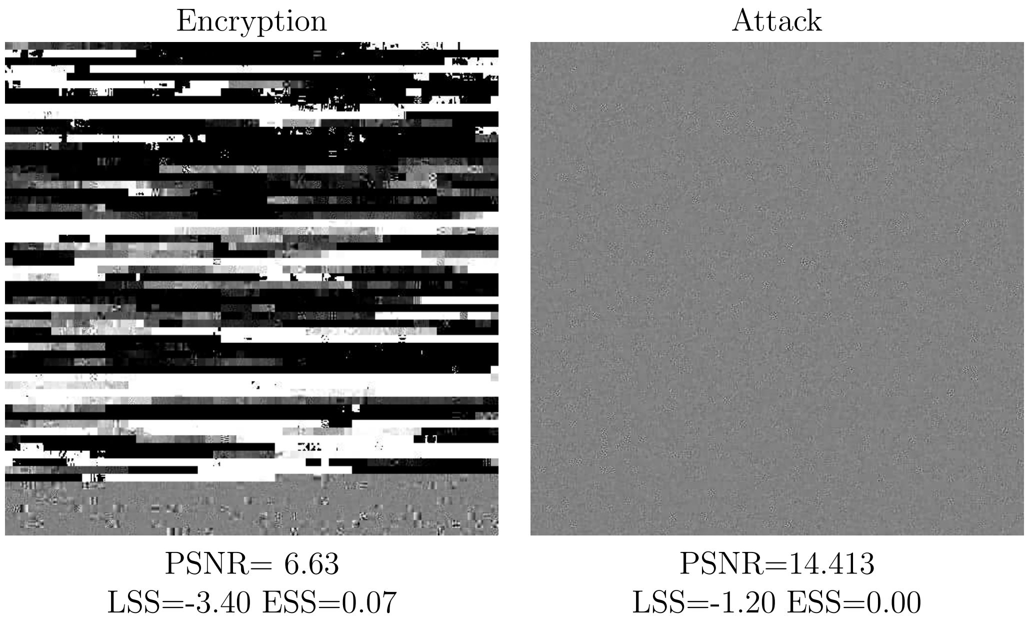


Figure 3.25: Successive Approximation for Confidentiality

Spectral Selection

Considering a JPEG file using spectral selection, first of all we have to encode all DC coefficients, because otherwise a subsampled image would be available to the attacker. Furthermore figure 3.12 shows that even the last 50% of AC data contain enough image information to draw concrete conclusions about the image content. Hence strictly more than 55% have to be encrypted in order to guarantee at least some confidentiality.

Hierarchical JPEG

The application of selective encryption of Hierarchical JPEG for confidential encryption is quite limited. In fact every enhancement layer contains sufficient edge and contour information to get at least an idea of the image content. Therefore every layer needs to be encrypted to guarantee that no content revealing information can be reconstructed.

Successive Approximation

The first 6 bit of DC coefficients have to encrypted. Furthermore the 6 bit of AC coefficients have to be encrypted, because they contain most of the edge and contour information. Applying this method to the Lena image leads to good results (see figure 3.25). In this example 30% of image data are encrypted. Figure 3.26 shows an example with extreme edges between surfaces of the same color. One can see that in this case it is necessary to encrypt the two DC bits as well.

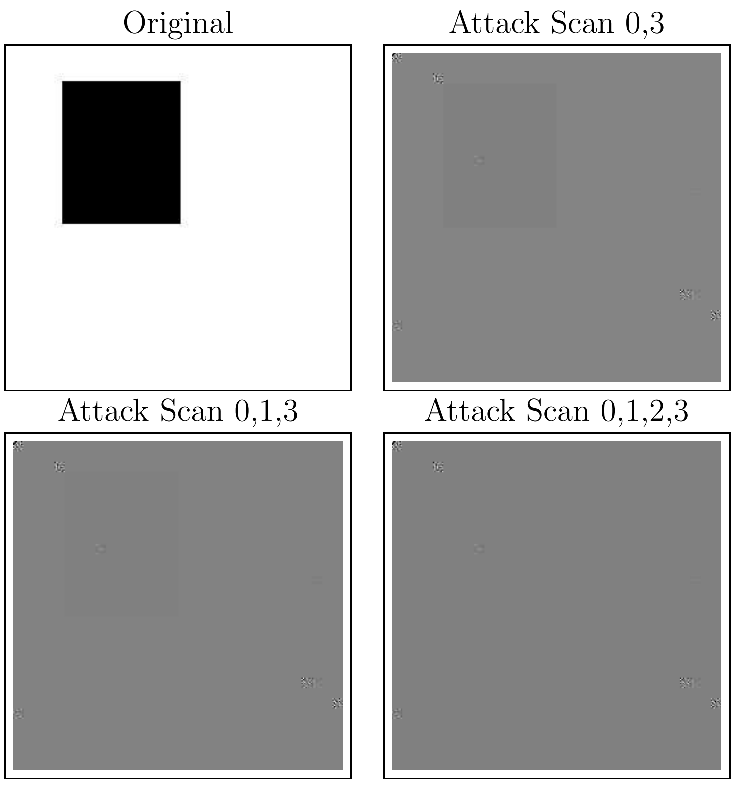


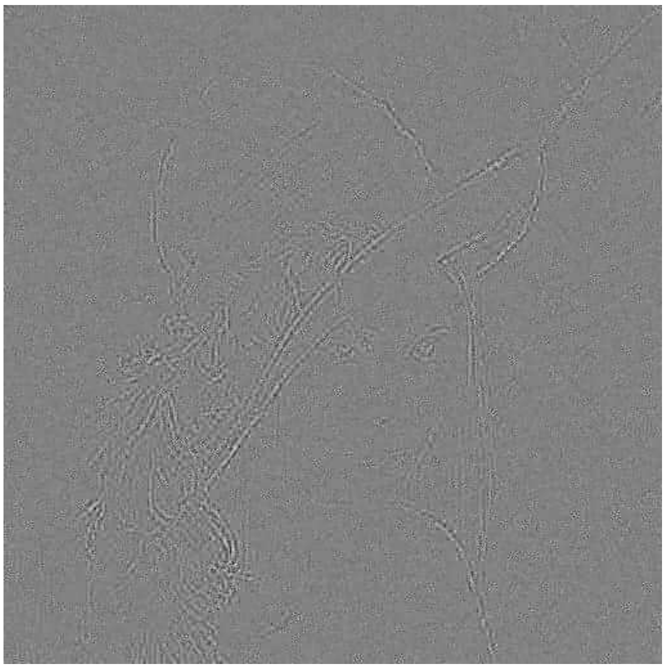
Figure 3.26: Successive Approximation for Confidentiality (2)

Conclusion

In order to ensure high confidentiality large parts of the JPEG files have to be encrypted. Most promising seems the use of selective encryption of JPEG with successive approximation. Nevertheless the results do depend on the unencrypted image source. However encrypting 6 bit DC and AC coefficients mostly leads to good results. Furthermore if the goal is to encrypt even less one might combine this approach with spectral selection, but it is very likely that edges and contours will show up. Figure 3.27 shows the result if the last third of the 6 bit AC coefficient data is not encrypted (this only makes 7% of the encoded coeffiecient data). One can clearly recognize some of the most important edges of the original picture, although still 23% of coefficient data is encrypted. Hence the benefits of this combination are quite limited.

3.5.2 Transparent Encryption

The main purpose of transparent encryption is not to fully hide all information of an image, but even the opposite, namely to make a specific controlled amount of image information public. Therefore there are two major requirements that have to be met:



PSNR=14.47
LSS=-1.20 ESS=0.04

Figure 3.27: Spectral Selection and Successive Approximation for Confidentiality

- To hide a specific amount of image information
- To show a specific amount of image information.

While the first requirement is a generalization of the confidentiality encryption approach - the condition of full encryption of all image information is extended to a specific amount - , the second requirement, namely to explicitly demand a certain image quality, is completely new.
Classic approaches for this purpose mainly feature the full encryption of the refinement information. In the case of spectral selection this has to be understood as the encryption of the AC coefficients starting from the last until the desired image quality is reached. For successive approximation one would start at the last bit of AC coefficient data and successively encrypting more and more scans up to the desired level of quality. Considering Hierarchical JPEG, this method would encrypt enhancement layers, starting from the last highest resolution layer until the desired image quality is reached.
However, when this approach is applied much of the image data has to be encrypted. Therefore it is of interest whether the amount of encrypted data can be reduced if one considers another encryption approach. The goal is to have adequate results but less encryption effort. Therefore visually more important data, which is not located in the last portions of enhancement information, is encrypted

first. Both approaches will be evaluated in the next sections. Once more not only the reconstructions of the encrypted files are relevant, but also possible attacks. Therefore the corresponding replacement attack of each encryption will be given and discussed as well. When encrypting with the goal of transparent encryption, the noise induced by the encryption has to be considered. The encryption of DC coefficients will mostly lead to totally noised images, while a replacement attack reveals that most of the edge and contour information is still present. Hence the encryption of the DC coefficient data is not suitable for transparent encryption.

Spectral Selection: the Classic Approach

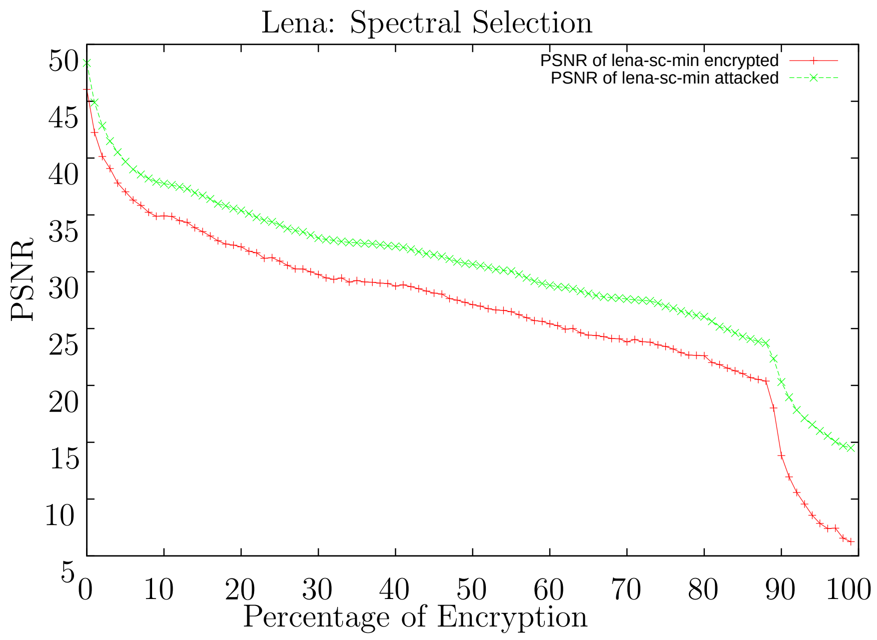


Figure 3.28: PSNR of Encrypted JPEG

Since the DC coefficient data can not be encrypted, the worst quality which can be achieved is the reconstruction based on the DC coefficients (see figure 3.11). When the standard approach is employed 90% of the coefficient data have to be encrypted to achieve this quality. The only possibility to cut down quality would be to increase the DC entry in the quantization table. Since quantization table are designed in order to achieve the best coding results, this will very likely decrease coding performance. The figures 3.28 and 3.29 show the PSNR , LSS and ESS of successively more and more encrypted JPEG file and the corresponding replacement attack. While the PSNR values indicate a significant improvement of image quality due to a replacement attack, the corresponding LSS/ESS values are more difficult to interpret. The LSS can not improve in the range from 1 to about 87, as it is constantly 1 for both. Since the DC coefficients which contain most color information are still unencrypted this is not unexpected. Starting

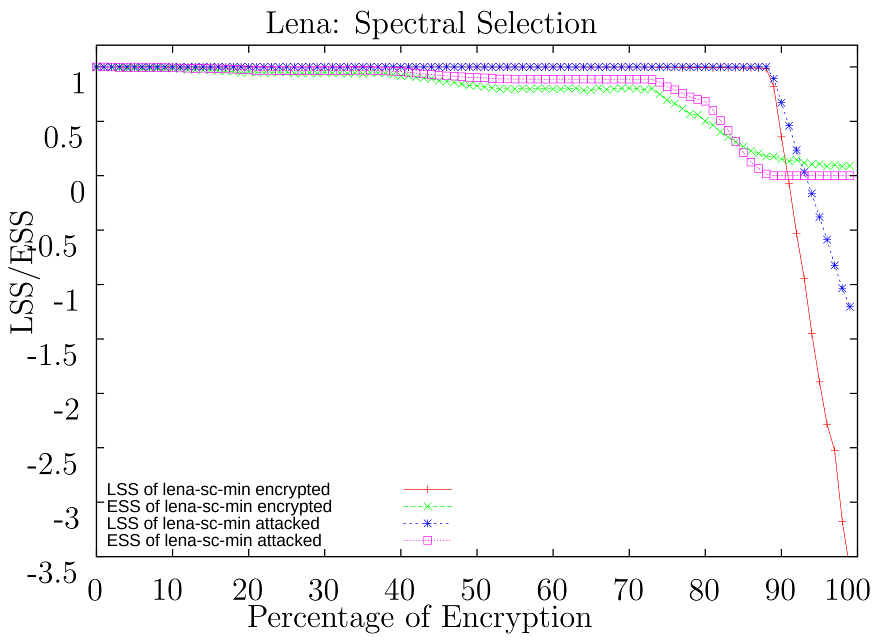


Figure 3.29: LSS/ESS of Encrypted JPEG

at about 87% the LSS of the replacement gets better and better, due to the fact that we replace whole parts of the image with gray which is closer to the average of the blocks luminance than a nearly random value. The ESS of the replacement attack is superior to that of the encryption until most of the edge information (at about 83%) is encrypeted/replaced. Then the images resulting of the replacement attack contain no more edge information in the 8x8 blocks. As it turns out random edge information does significantly perform better than no edge information in terms of ESS. The corresponding attack does not tremendously improve image quality. The reason is that the error propagation in a JPEG file is always from the previous coefficient in zig-zag scan to the next (see section 2.1.4). Hence when starting the encryption at the end the error does not propagate to unencrypted areas and therefore diminish the impact of the replacement attack. Nevertheless the replacement attack leads to an improvement of image quality and most artefacts resulting of the encryption can be suppressed. A high percentage has to be encrypted in order to have at least some loss of image quality. More than 40% have to be encrypted that even with the replacement attack a significant loss of image quality is perceivable. About 70% have to be encrypted obtain an image that consists mostly of noised 8x8 blocks, but the edge information of the first AC coefficients is still present. At 90% all AC coefficient data is encrypted. Figure 3.30 shows selected JPEG files, encrypted with this method.

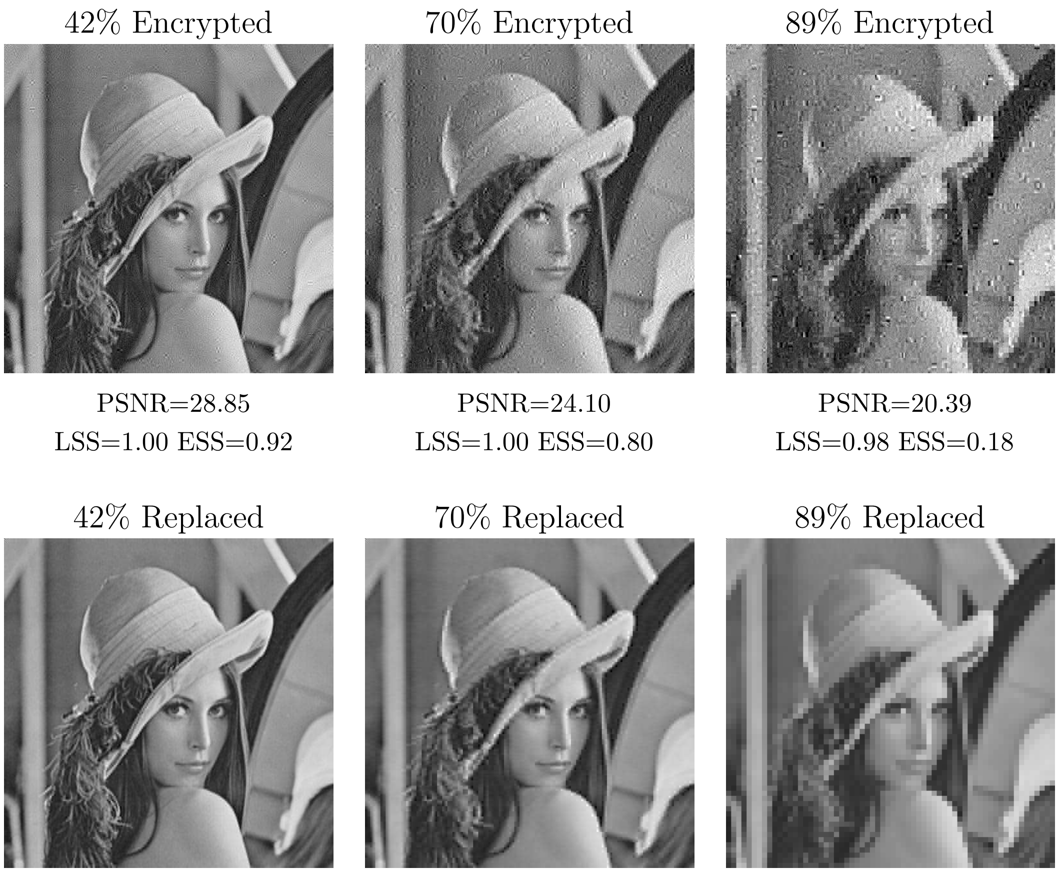
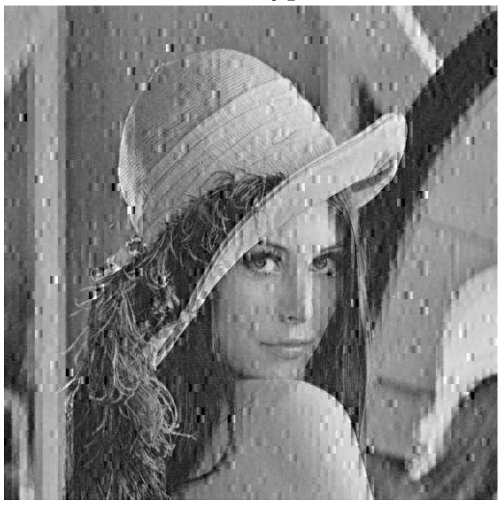


Figure 3.30: Spectral Selection: The Classic Approach

Spectral Selection: the New Approach

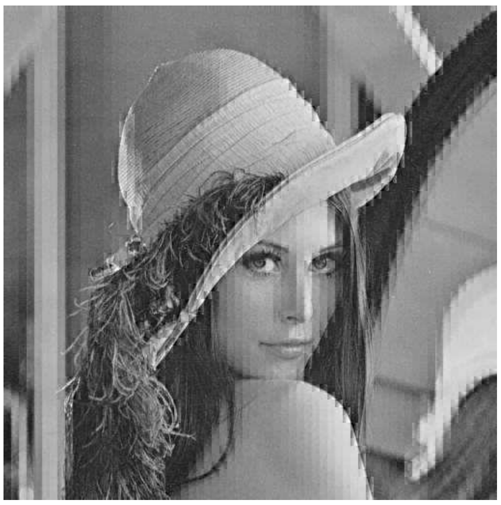
The PSNR and LSS/ESS plots of the classic approach (see figures 3.28 and 3.29) indicate that there is space for improvement. Since the last layers do not contribute much to the image quality, it is reasonable not to start encrypting at the end of the data, but at a specific point after the DC data according to the required image quality. The PSNR and LSS/ESS plots (see figures 3.17 and 3.18) can give an overview, how a specific encryption location influences the image quality of the reconstruction. For most applications starting right after the DC data will be appropriate, in order to minimize the image quality and the encryption rate. Figure 3.31 shows the results of encrypting the first AC coefficient (7% of the file size), which leads to nearly the same image quality as encrypting 70% starting from the end (see figure 3.30). If even worse image quality is desired the amount of encryption has to be increased. However, the advantages of the new method will become less when the desired quality is very close to full encryption of all AC coefficient data. The encryption of 23% percent of the leading AC coefficients results in a image quality similar to encrypting about 89% with the classical approach (see figures 3.30 and 3.32).

7% Encrypted



PSNR=24.18
LSS=0.99 ESS=0.30

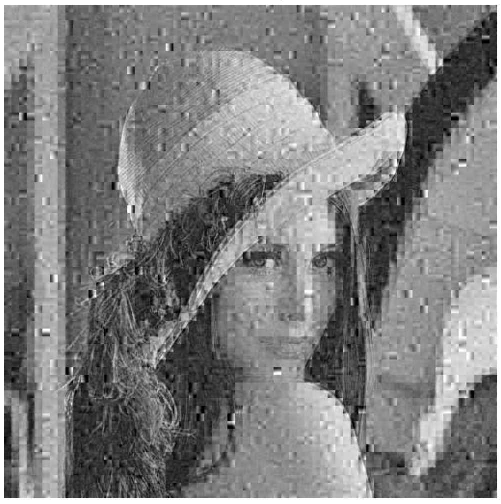
7% Attacked



PSNR=27.14
LSS=1.00 ESS=0.31

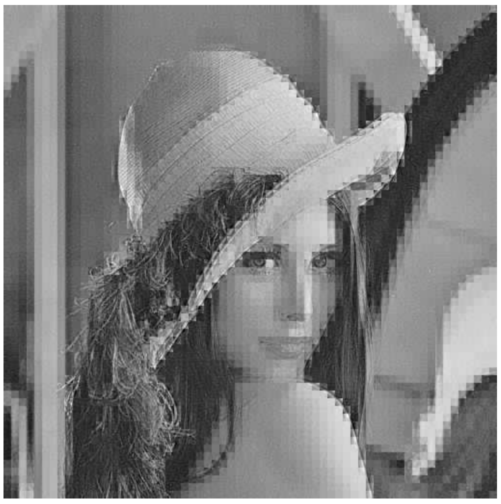
Figure 3.31: Spectral Selection: The New Approach

23% Encrypted



PSNR=22.00
LSS=0.99 ESS=0.18

23% Attacked



PSNR=25.15
LSS=1.00 ESS=0.034

Figure 3.32: Spectral Selection: The New Approach

Successive Approximation: the Classic Approach

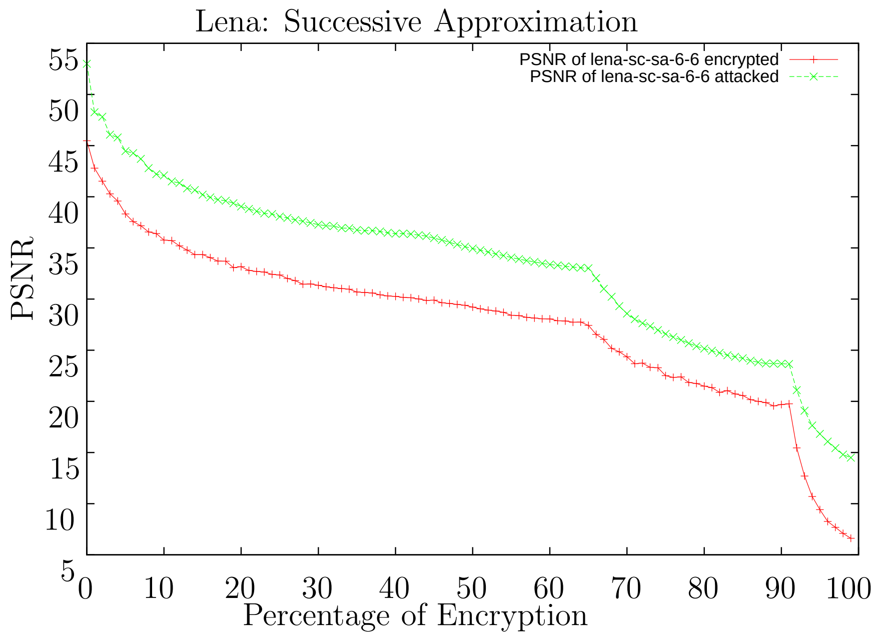


Figure 3.33: PSNR of Encrypted JPEG

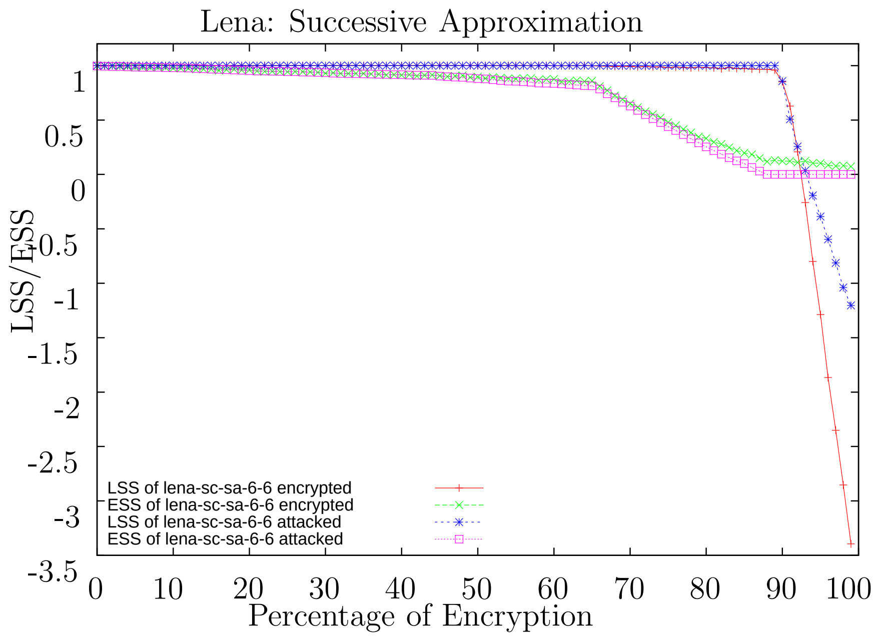


Figure 3.34: LSS/ESS of Encrypted JPEG

As shown in figure 3.5 almost all image information is contained in two scans, the 6 bit DC coefficient scan containing about 7% coefficient data and the 6 bit

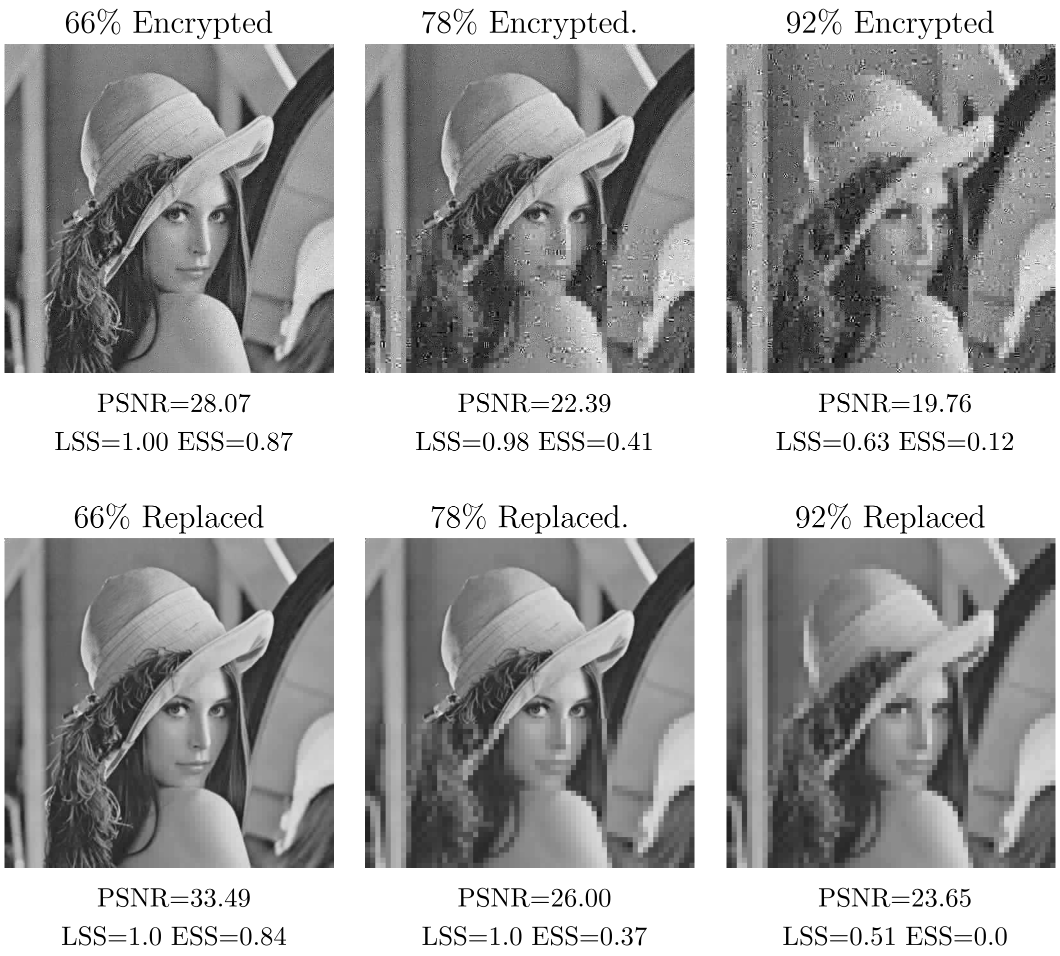


Figure 3.35: Successive Approximation: The Classic Approach

AC coefficient scan containing about 22% of coefficient data. Therefore much of the coefficient data has to be encrypted - approximately 66%, these are the two 1 bit planes of the AC coefficients and some minor AC coefficient data - to obtain a significant loss of image information. Both PSNR and ESS (see figures 3.33 and 3.34) show discontinuities at a little bit more than 60%, this is were the 6 bit AC coefficients start. However more than 65% of coefficient data have to be encrypted to obtain a significant decrease of image quality. Selected encrypted files are shown in figure 3.35. In the image with 78% encrypted one can see that the upper half of the 6 bit AC coefficients is still unencrypted while the lower half is already encrypted.

Successive Approximation: the New Approach

The PSNR and LSS/ESS plots of the classic approach (see 3.33 and 3.34) indicate that their is space for improvement. Since the last layers do not contribute much to the image quality, it is reasonable not to start encrypting at the end of the data, but at a specific point after the DC data according to the required image quality. The PSNR and LSS/ESS plots (see figures 3.8 and 3.9) can give an overview, which location has what influence. For most applications starting right after the DC data is appropriate. The scan containing the 6 bit AC coefficient data mainly influences the image quality. However, when encrypting this scan with 22% this will hide image information subsequently from the top left corner to the right left corner, line by line. For most applications this will not be the goal and this scan has to be split up with spectral selection. Figure 3.36 shows results for the encryption of the leading 5 AC coefficients (only the 6 MS-bits, which represent 14% of coefficient data, are encrypted), which leads to similar image quality as encryption of about 90% with the classical approach (see figure 3.35). In this case the new approach can significantly reduce the encryption rate for the same image quality.

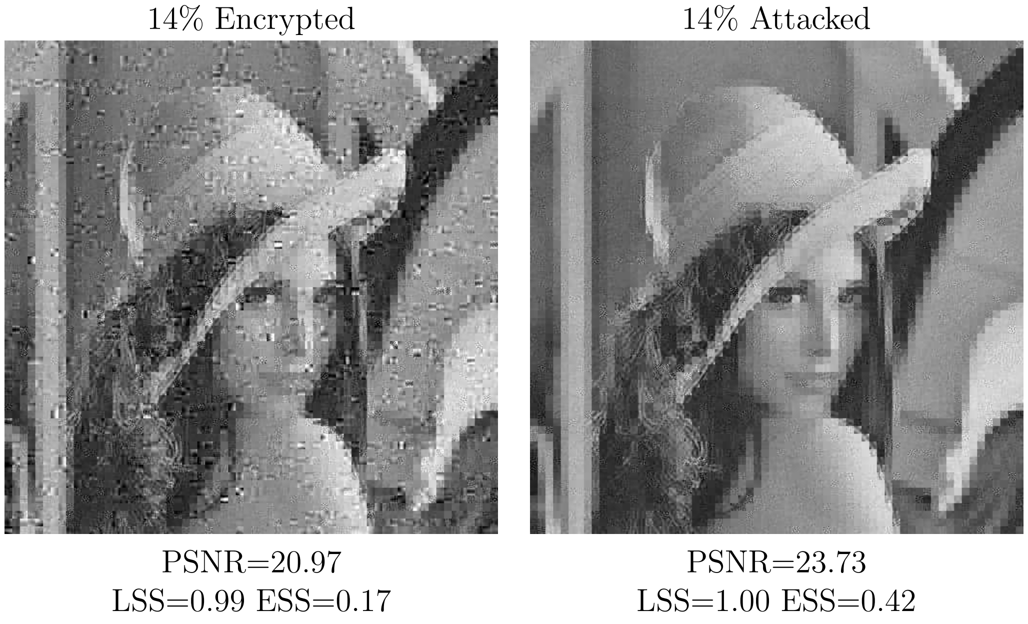


Figure 3.36: Successive Approximation: The New Approach

Hierarchical JPEG: the Classic Approach

Hierarchical JPEG can offer a great flexibility in its coding parameters. While for the progressive sequential JPEG modes the worst image quality is limited (due to the DC coefficients), this is not true for Hierarchical JPEG. The noise induced by encryption is major problem of selective encryption of Hierarchical JPEG. Combining the Hierarchical mode and progressive modes leads to an even finer scalability, but the previous results will not apply fully because of the different nature of the differential images. When applying the classical approach to Hierarchical JPEG, the enhancement layers are encrypted. Since the quality

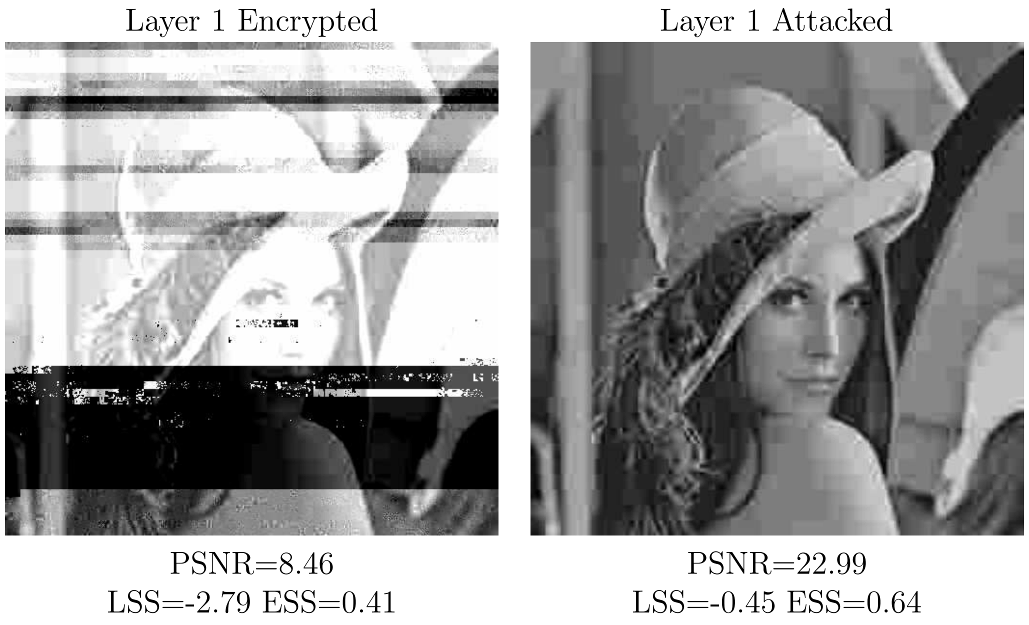


Figure 3.37: Hierarchical JPEG with Low Quality Base Layer: The Classic Approach

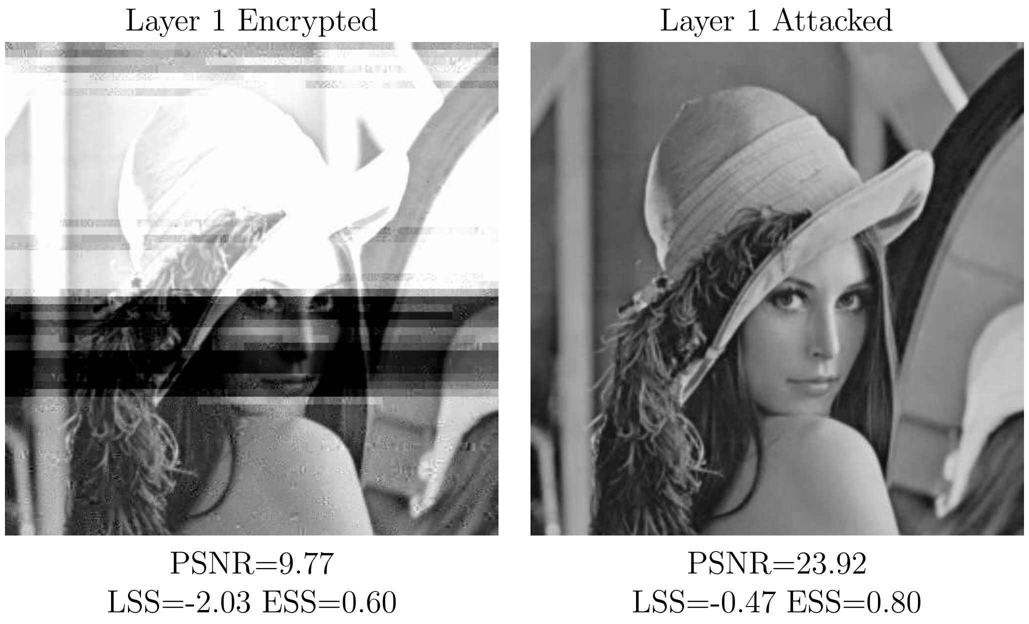


Figure 3.38: Hierarchical JPEG with High Quality Base Layer: The Classic Approach

of the base layer can be freely adjusted, the quality of the image resulting from a replacement attack can be determined a priori by the compression settings. However depending on the quality of the base layer (worse quality leads to better compression), the base layer's size varies. Despite the better quality base layer the image quality of these files is the same. For two layers (base layer and enhancement layer) the base layer has between 30% (high quality) and 6% (low quality) of the file size. Hence 70% to 94% have to be encrypted, which nevertheless is quite the same amount of bytes. In figure 3.37 the case with a low quality base layer is shown. The results for a high quality base layer are illustrated in figure 3.38. It can be clearly seen that the quality of the reconstruction depends on the quality of the base layer. Furthermore the noise is rather high and there is an enormous gap between the direct reconstruction and the result obtained through a replacement attack. When the number of layers is increased a higher and higher percentage of data has to be encrypted. This approach is quite flexible, but needs high percentage of encryption and suffers severely from distortion and noise.

Hierarchical JPEG: the New Approach

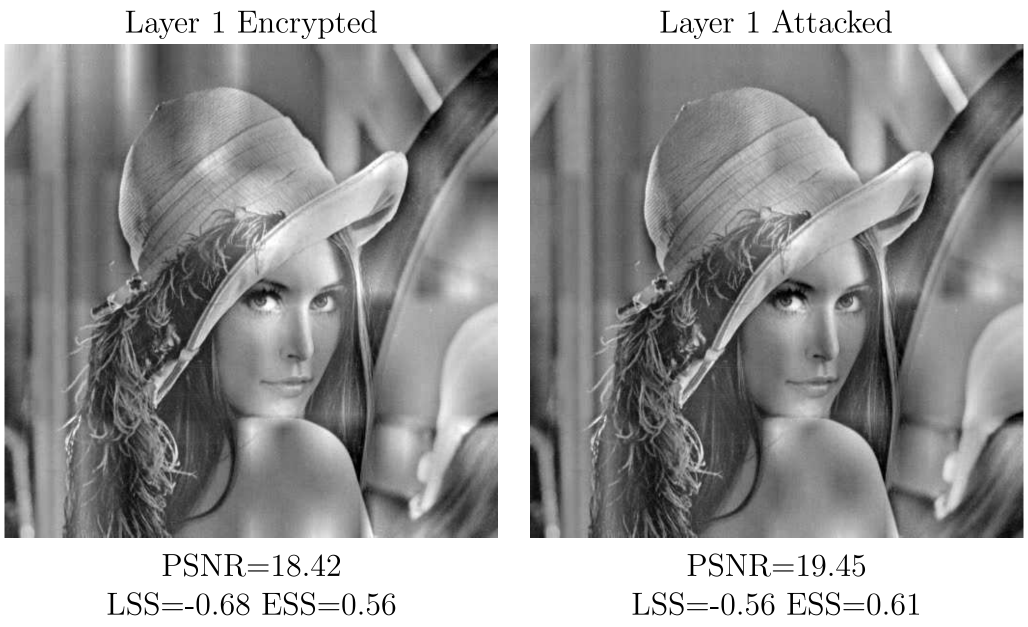


Figure 3.39: Hierarchical JPEG with 6 Layers: The New Approach

Despite the fact that the most influential layer always is the last enhancement layer the lower layers do contain information which is crucial for reconstructing the image. Basically all the luminance information is contained in the lower layers. In figure 3.21 the image in the middle shows the result of encrypting the first enhancement layer (three layers, two enhancement layers). A significant loss of image quality is achieved by encrypting only 1.6% of image data. Even more image data is lost when encrypting the base layer (1.6% of image data), but it is questionable if this meets the required image quality. As figure 3.22 reveals, a replacement attack improves image quality due to removing encryption artefacts,

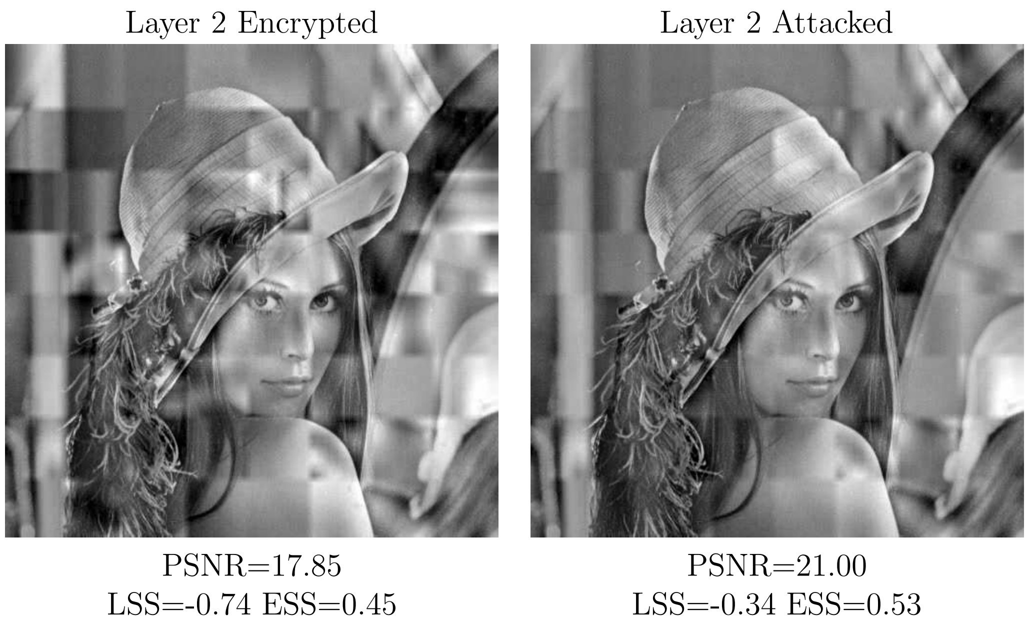


Figure 3.40: Hierarchical JPEG with 6 Layers: The New Approach (2)

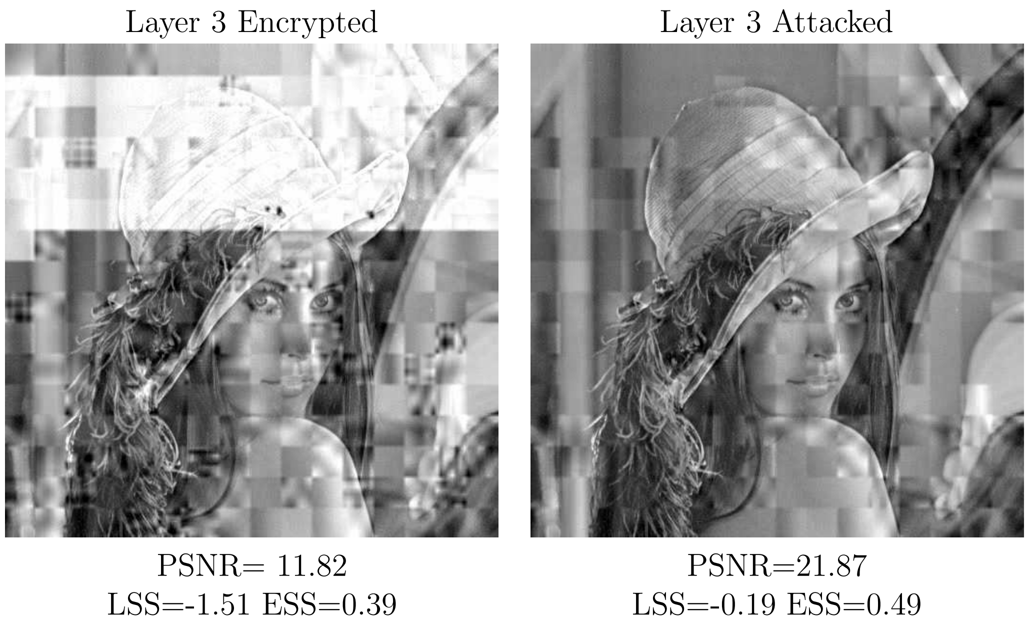


Figure 3.41: Hierarchical JPEG with 6 Layers: The New Approach (3)

but the image information contained in the encrypted parts is lost. The possibly disturbing encryption noise can reduced by using more layers (see figure 3.23). With more layers it is possible to remove all color information without inducing too much noise. This is neither possible with spectral selection nor successive

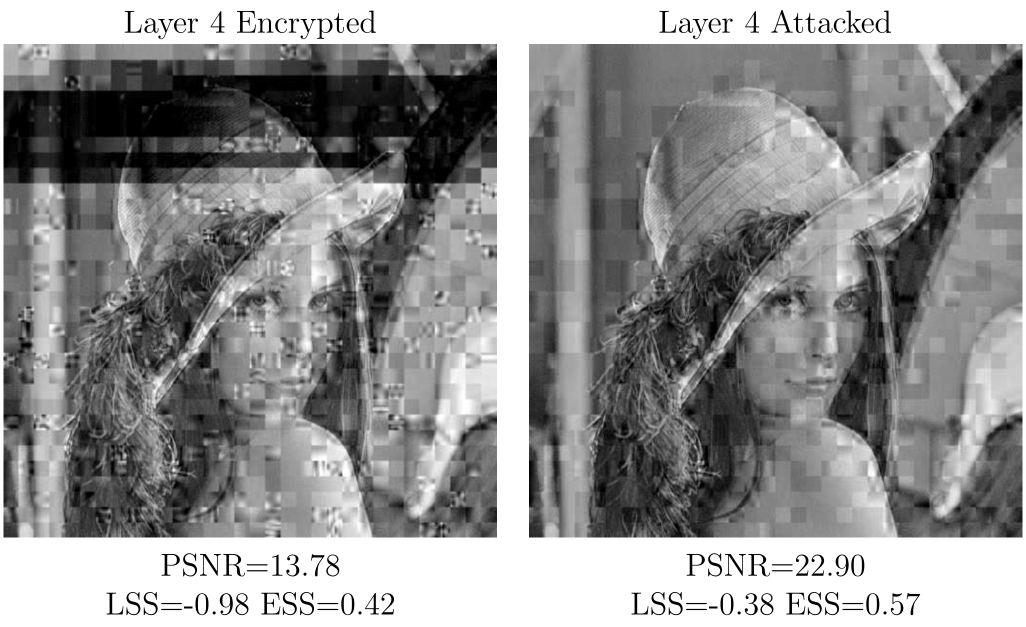


Figure 3.42: Hierarchical JPEG with 6 Layers: The New Approach (4)

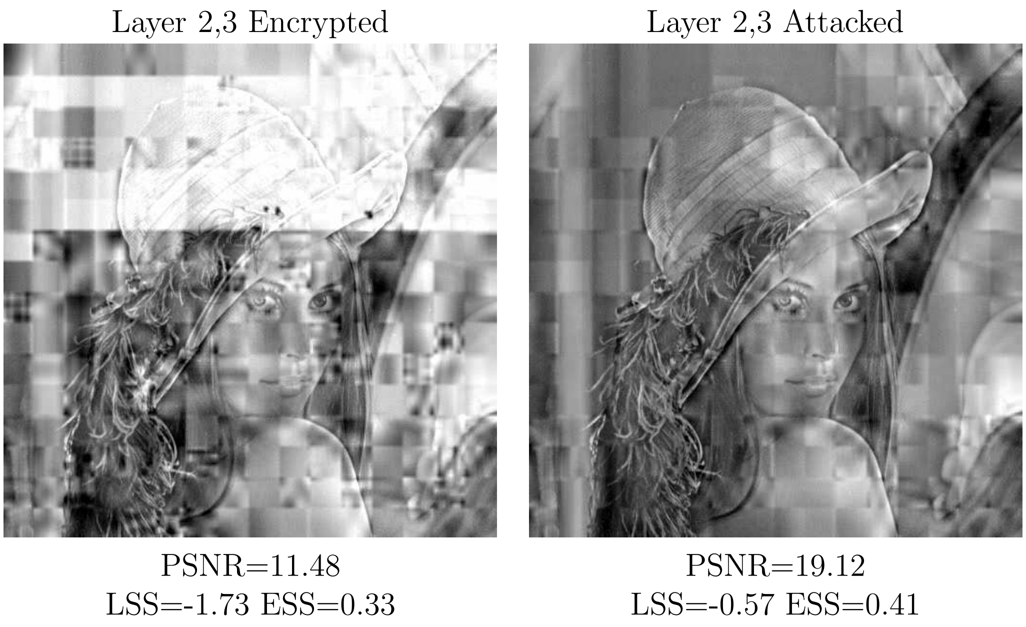
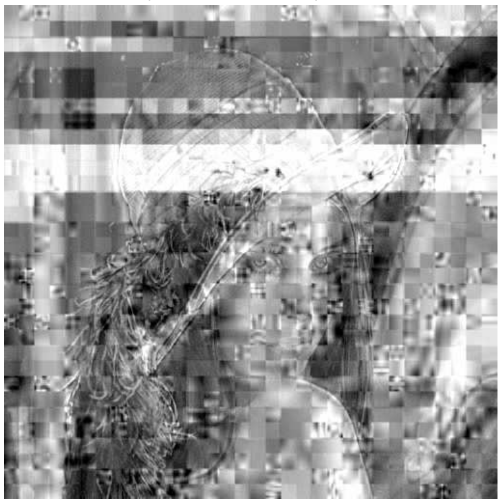


Figure 3.43: Hierarchical JPEG with 6 Layers: The New Approach (5)

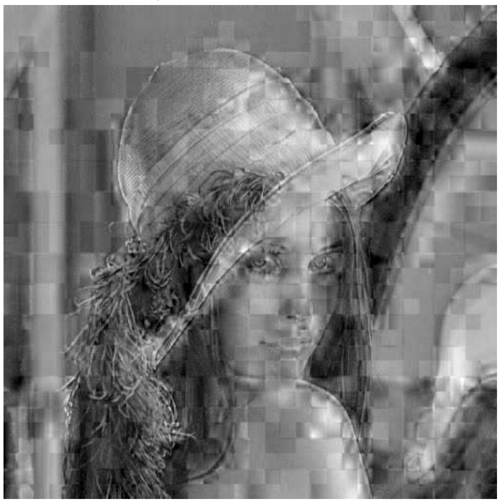
approximation. For the compression of the Lena image with recommended parameters for good compression performance the percentage of the layers on the overall size is 0.62%,0.67%, 0.80%, 1.29%, 2.55%, 94.05% (starting with base layer and ending with the last enhancement layer). These parameters are $q_f = 10$

Layer 3,4 Encrypted



PSNR=13.84
LSS=-1.16 ESS=0.31

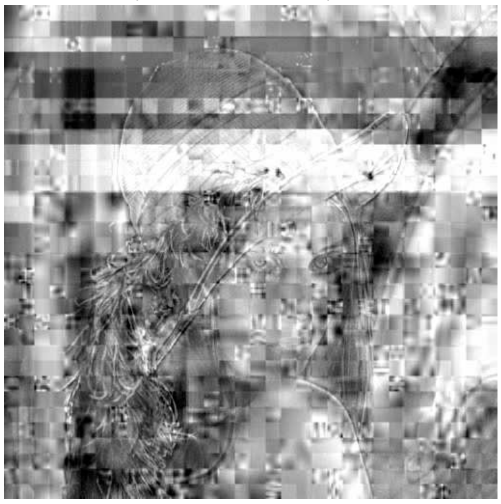
Layer 3,4 Attacked



PSNR=20.04
LSS=-0.50 ESS=0.43

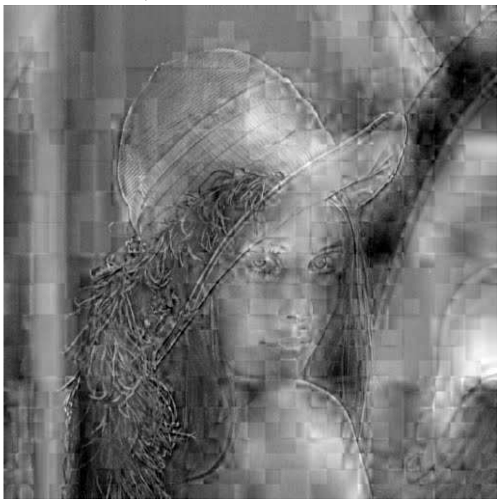
Figure 3.44: Hierarchical JPEG with 6 Layers: The New Approach (6)

Layer 2,3,4 Encrypted



PSNR=12.86
LSS=-1.41 ESS=0.28

Layer 2,3,4 Attacked



PSNR=18.01
LSS=-0.70 ESS=0.35

Figure 3.45: Hierarchical JPEG with 6 layer: The New Approach

for layer 0, $\alpha = 140$ for all other enhancement layers except the last, where α is set to 10. Summarizing the first four layers contain only 3.27% of the image data, but heavily decrease the image quality (see figure 3.23). If one wants to preserve more luminance information the lowest layers must not be encrypted. In figures 3.39 to 3.45 all possibilities of encryption - omitting the base layer - of

Hierarchical JPEG with 6 layers are shown. In fact these figures also contain all possibilities for Hierarchical JPEG with fewer layers, since the unencrypted base layer and the next unencrypted enhancement layers can be reconstructed. This is similar to Hierarchical JPEG with fewer layers. E.g. if the base layer and layer 1 are not encrypted, the layer 1 can be reconstructed and we obtain Hierarchical JPEG with one layer less. The results clearly show that the image quality can severely be reduced by encrypting very little of the file. By using selective encryption of progressive JPEG the noise induced by encryption could possibly be reduced. The results presented all focus on Hierarchical JPEG optimized for good compression, this leads to low quality low resolution layers. In tables 3.7 and 3.8 the results for high quality low resolution layers are summarized.

Conclusion

The results show that the goals of transparent encryption can be met to some degree. When applying transparent encryption to spectral selection and successive approximation there is a lower bound for the image quality. This lower bound is derived from the image information contained in the DC coefficients. It is not possible to obtain a worse image quality without encrypting some DC coefficients and therefore destroying all image information of those blocks. It could be shown that instead of encrypting all enhancement layers the encryption of certain parts leads to comparable results with less encryption effort. The gap between encryption and replacement attack is quite big when applying transparent encryption to Hierarchical JPEG. When applying the new approach the luminance information of an image can be removed.
However, when the proposed methods are applied to real world applications, the exact requirements and conditions have to be evaluated first. Afterward one has to verify that those requirements and conditions can be met with the proposed methods. Since Hierarchical JPEG is not very widely used, a condition might be the use of the progressive-sequential modes, which are limited in the decrease of image quality. The results presented could serve as guidelines.

Chapter 4

JPEG2000: An Introduction

JPEG2000 was designed to replace the widely used JPEG standard. Part-1 of the JPEG2000 standard was first published in December 2000. Until now its usage is still limited, but slowly receiving more and more attention and popularity and even some mobile phones with cameras (e.g., the Nokia N70 3G / GSM Phone) are currently capable of displaying JPEG2000 files. The digital cinema specification applies JPEG2000 for the compression of the frames [5]. SANYO WB2000 is a networked digital video camera which does 640x480 @ 30 fps and outputs JPEG2000 images and Thomson's Infinity Digital Media Camcorder also is capable of recording JPEG2000 images. Analog Devices produce a JPEG2000 chip (ADV202) which is sold for about 30$ and is able to encode standard NTSC video at 40 fps. Among the intended features of JPEG2000 are

- better compression performance,
- a lossless mode within the same compression pipeline,
- multi-resolution capabilities,
- improved scalable capabilities,
- improved error resilience and
- improved rate/distortion control.

Hence one can state that JPEG2000 was designed to be the ultimate multi-functional, multi-purpose image compression standard. Therefore (and possibly because patents had to be bypassed as well) it is quite a complex system, whose fundamental building blocks are the **DWT** (Discrete Wavelet Transform) and Taubmann's **EBCOT** (Embedded Block Coding with Optimal Truncation). However, the standard is far too complex and extensive to give a detailed description within the scope of this thesis. The very interested reader is referred to Taubman's book [47], while the interested reader can get a shorter (and freely available) overview in [7]. It has been written by Michael Adams the author of **JasPer** an open source JPEG2000 implementation, that has been included in the JPEG-2000 Part-5 standard (i.e., ISO/IEC 15444-5), as an official reference implementation of the JPEG-2000 Part-1 codec. There is another reference implementation in JAVA called **JJ2000** as well.

4.1 The JPEG2000 Compression Pipeline

Figure 4.1 shows the JPEG2000 compression pipeline. In the first step, the preprocessing, the image components are divided into tiles and color transformation can be done either the fully reversible (**RCT**) or the irreversible (**ICT**) way. By default there is only one tile, the whole image. The tiles are then independently

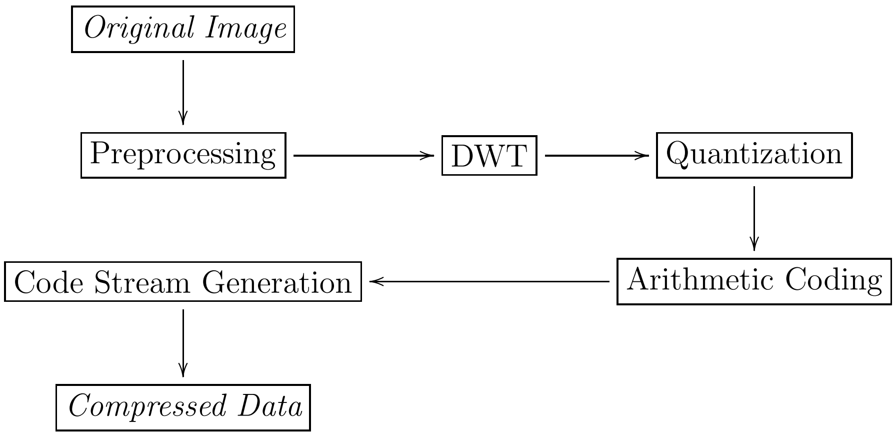


Figure 4.1: The JPEG2000 Compression Pipeline

wavelet transformed. Section 4.2.4 gives a short introduction to the DWT. The DWT coefficients are then quantized, through uniform dead-zone quantization. The DWT coefficients are then divided into 64x64 sized blocks (**code blocks**) which are independently coded on a bit plane basis. Within the coding procedure every bit plane of a code block is assigned a quality level. In the last step the **code stream** is generated. JPEG2000 offers an extremely flexible code stream. It offers a progressive behavior in four different directions, which are Quality, Resolution, Spatial Location and Components. All these quality layers are realized through the ordering of the so called **packets**. A packet contains encoded coefficient data of a certain component, a certain tile, a certain quality layer, a certain resolution and a certain **precinct**. A precinct is the result of yet another partition scheme for the DWT coefficients, which groups code blocks belonging to the same spatial region. If quality progression is desired, the packets are ordered according to the quality level to which they contribute. If resolution progression is desired, the low resolution packets are sent first and higher resolutions by and by (hence the low resolution image is available first). For spatial progression the packets are grouped according to the spatial region to which they belong (e.g. line by line). If component progression is desired the packets are grouped by the component they contribute to. In this case it is possible to receive a gray scaled (luminance) image first and then add color step by step. As already mentioned the building block of the code stream is a packet, which will therefore play a major role in selective encryption. In the following sections only the DWT and the code stream will be discussed in detail. The DWT is discussed because it is a completely new transformation offering many features which make JPEG2000 superior to classic JPEG. The code stream syntax and organization are crucial for both selective encryption and robust hashing of JPEG2000 files.

4.2 The Wavelet Transform

As for JPEG2000 a detailed description (especially including all the mathematical details) is not possible within the scope of this thesis. Several excellent monographs, articles, and books have been published about wavelet transformation ([9, 21, 22, 26, 27, 40]). The following section only tries to summarize some of the key concepts and give an impression of how the discrete wavelet transform works. Wavelets have their origins in pure mathematics, but nevertheless are frequently applied to various problems. However, to fully understand wavelets (up to the level of proving that they work) a good mathematical background and time is required. Nevertheless the result, the DWT, is rather simple to explain and to implement. Furthermore its complexity (in an optimized form, known as the **fast DWT**) is only $O(n)$, whereas a Fourier Transformation is $O(n \log n)$ There is a strong relationship between subband transforms and wavelet transforms. It is so close that the terms are often used interchangeably.

4.2.1 Subband Transforms

Subband transforms do not process the data in blocks. Hence block artefacts are omitted. An input signal $x[n]$ is transformed to an output signal $y[n]$, whereby $y[t]$ for a specific t can depend on all $x[n],\ n \in \mathbb{Z}$. The way this is done is defined by so called **filters**. Usually this is restricted to those $x[n]$ which are near t. In case of odd-length filters it is restricted to those $x[n]$ with $|n - t| <= w$ where $2w + 1$ is the window size (filter length). Therefore this is also called **sliding window transform** and all elements of $x[n]$ within the window of t are transformed to $y[t]$. The rules of the transformation are given through the filters. The filter applied in the forward transformation is called **analysis filter** (h_x), whereas the one applied to reverse the transformation is called **synthesis filter** (g_x). The output signal $y[t]$ can also depend on the position t. The term channel thereby denotes the number of cases. In JPEG2000 a low pass filter and a high pass filter are applied alternately. Since a separable transform is used it can be applied to rows and columns iteratively. Hence this is a two channel separable subband transform. The next subsection will try to summarize some of the mathematical properties of a wavelet transform.

4.2.2 Mathematical Background

In image transforms we usually consider discrete signals $x[k]$ (the image). Nonetheless the wavelet transformation originally is concerned with the representation of functions through wavelet bases. The functions of interest are in $\mathcal{L}^2$, the Hilbert space of square-integrable (finite energy) functions. A wavelet basis is defined by a function $\psi(t)$, modified by two parameters, the dilatation a and the translation b, both $\in \mathbb{R}$.

Definition 4.2.1 (Wavelet Basis)
is a family of functions $\{\psi(t)_{(a,b)})\}$ *with:*

$$\psi(t)_{a,b} = |a|^{\frac{-1}{2}} \psi(\frac{t-b}{a})$$

It can be shown that countable bases of wavelets are sufficient so that any signal/function $x(t) \in \mathcal{L}^2$ can be written as linear combination of the wavelets. In this case the notation is changed. The translation b is set to $n2^m$ and the dilatation a to 2^m, where n and m are in $\mathbb{Z}$. Furthermore the wavelet basis is denoted by $\psi_n^{(m)}$.

Definition 4.2.2 (Countable Wavelet Basis)
is a family of functions $\{\psi(t)_n^{(m)}\}$ *with:*

$$\psi(t)_n^{(m)} = \sqrt{2^m}\psi(2^{-m}t - n)$$

such that $\psi(t)_n^{(m)}$ *are linearly independent and span* $\mathcal{L}^2$.

Then any signal $x(t)$ can be written

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} y^{(m)}(n)\psi_n^{(m)}(t)$$

The $y^{(m)}(n)$ are a sequence of real numbers.

These results are quite fundamental, but not really applicable. However, through the **MRA** (Multi Resolution Analysis), introduced by Mallat [28], we have a tool for interpreting and constructing wavelet bases.

Multi Resolution Analysis

MRA on $\mathcal{L}^2$ is defined as a set of subspaces ($\ldots \subset \mathcal{V}^0 \subset \mathcal{V}^{-1} \subset \ldots$) of $\mathcal{L}^2$, satisfying the following additional properties:

MR1 $\bigcup_{m\in\mathcal{Z}} \mathcal{V}^m = \mathcal{L}^2$

MR2 $\bigcap_{m\in\mathcal{Z}} \mathcal{V}^m = \{0\}$

MR3 $x(t) \in \mathcal{V}^0 \iff x(2^{-m}t) \in \mathcal{V}^m$

MR4 $x(t) \in \mathcal{V}^0 \iff x(t-n) \in \mathcal{V}^0$

MR5 $\exists \{\varphi_n\}_{n\in\mathbb{Z}}$, orthonormal basis for $\mathcal{V}^0$, such that $\varphi(t) = \varphi(t-n)$.

Usually the function $\varphi(t)$ is referred to as the **scaling function**. It can be shown that the scaling function can be used to construct an orthonormal wavelet basis. Properties MR3 and MR5 indicate that every subspace $\mathcal{V}^m$ has an orthonormal basis, $\{\varphi_n^{(m)}\}_{n\in\mathbb{Z}}$. It can be shown that the scaling function $\varphi(t)$ can be written as linear combination of the orthonormal basis $\{\varphi_n^{-1}\}_{n\in\mathbb{Z}}$ of $\mathcal{V}^{-1}$ for some sequence $g_0[n]$.

$$\varphi(t) = \sqrt{2}\sum_{n=-\infty}^{\infty} g_0[n]\varphi_n^{-1}(t) = \sqrt{2}\sum_{n=-\infty}^{\infty} g_0[n]\varphi(2t-n)$$

It can be shown that the sequence $g_0[n]$ fulfills the conditions required for low-pass synthesis filter and that $g_1[n]$ fulfills the conditions required for high-pass filter of a two channel orthonormal subband transform. Furthermore we can express the mother wavelet ψ as linear combination of:

$$\psi(t) = \sqrt{2}\sum_{n=-\infty}^{\infty} g_1[n]\varphi(2t-n)$$

where $g_1[n]$ is defined by a sequence of weights $g_0[n]$ derived through:

$$g_1[n] = (-1)^{n+1} g_0[-(n-1)]$$

Hence we can construct a wavelet basis applying MRA, defined by a scaling function φ.

4.2.3 Discrete Wavelet Transform

In the discrete case the input signal $x(t)$ is characterized at some resolution, e.g. $\mathcal{V}^0$, by the discrete sequence $y_0^{(0)}[n]$ such that

$$x^{(0)}(t) = \sum_{n\in\mathbb{Z}} y_0^{(0)}[n]\varphi(t-n)$$

This can be decomposed into

$$x^{(0)}(t) = \sum_{i\in\mathbb{Z}} y_0^{(1)}[i]\varphi_n^{(1)}(t) + \sum_{n\in\mathbb{Z}} y_1^{(1)}[n]\varphi_n^{(1)}(t)$$

where

$y_0^{(1)}$ is derived through applying the corresponding low-pass analysis filter h_0 for the synthesis filter g_0 to the sequence y_0^0 and

$y_1^{(1)}$ is derived through applying the corresponding high-pass analysis filter h_1 for the synthesis filter g_1 to the sequence y_0^0.

Hence $y_0^{(1)}$ is denoted as low-pass band and $y_1^{(1)}$ as high-pass band. These operations can be applied recursively resulting in some subset of a tree structure. In the case of JPEG2000 this decomposition is applied iteratively up to a certain level (**wavelet decomposition level**) to the low-pass band ($y_0^{(n)}$).

4.2.4 The DWT in JPEG2000

In JPEG2000 Part-1 the **CDF 9/7** (Cohen-Daubechies-Feauveau) filters are used for the lossy mode, while for the lossless mode a spline based 5/3 filter is used. 9/7 indicates that the low-pass filter has length 9 and the high-pass filter length 7. It is applied to rows and columns separately. In fact after the DWT the subbands are interleaved. Since the DWT is iteratively applied to the low-pass band, the low-pass coefficients are sorted into the top left corner. Figure 4.2 shows the decomposition in the four resulting subbands, where the low-pass band of horizontal and the vertical DWT (**$\mathbf{LL}_1$ band**) is in the top left corner. The high-pass band is in the bottom right corner **$\mathbf{HH}_1$ band**. The LL_1 band is a subsampled version of the original, while the high-pass bands contain the edge information. Figure 4.3 visualizes the notation. Applying the DWT to the LL_1

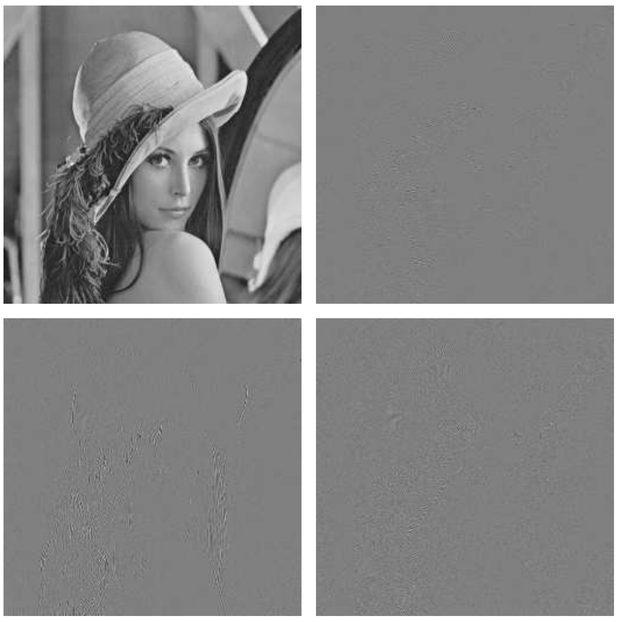


Figure 4.2: Discrete Wavelet Transform: Fist Iteration

band yields to a further partition of the image. The result is shown in figure 4.4. Figure 4.5 shows the result of a wavelet decomposition of 3 levels. The JPEG2000 standard supports decomposition levels from 0 to 32, where typical values are in the range of 4 to 9.

4.3 The Code Stream

At the first stage of the JPEG2000 compression pipeline is the partition of the image components into tiles, which are then transformed with the DWT. After the

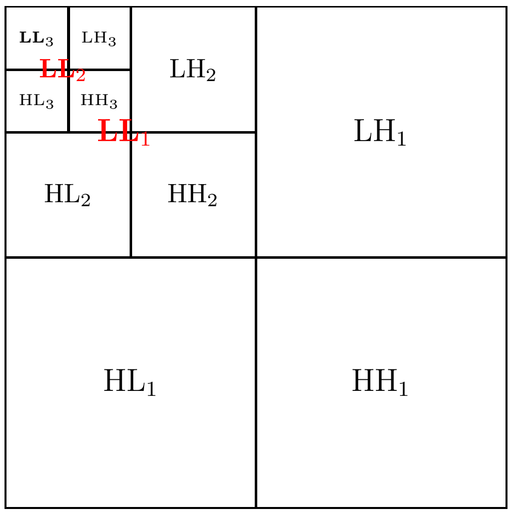


Figure 4.3: A 3 Level Decomposition

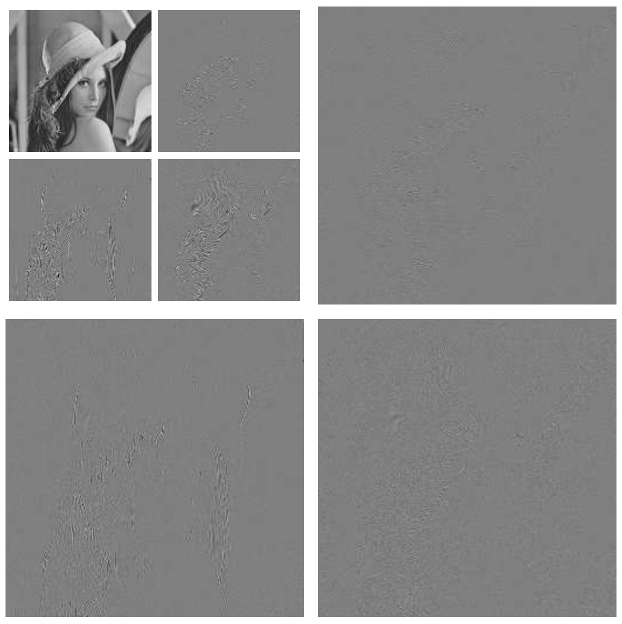


Figure 4.4: Discrete Wavelet Transform: Second Iteration

DWT, JPEG2000 applies a modern adaptive quantization and coding procedure. Details can be found in [47]. The most important entity in JPEG2000 stream is a packet.

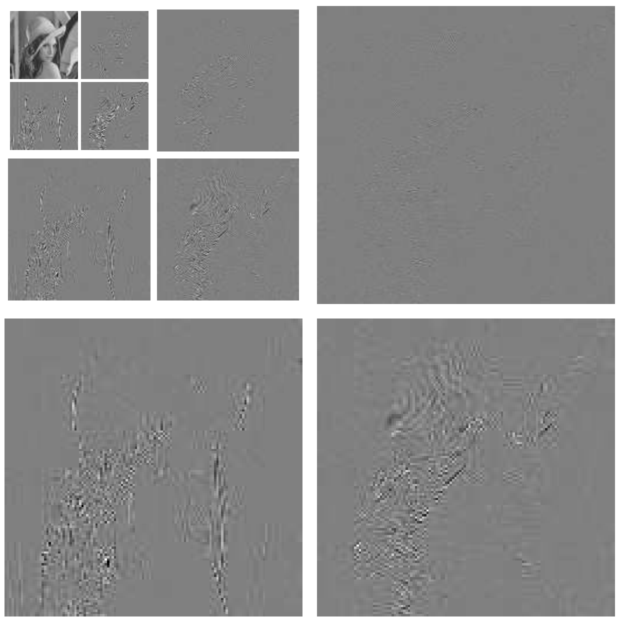


Figure 4.5: Discrete Wavelet Transform: Third Iteration

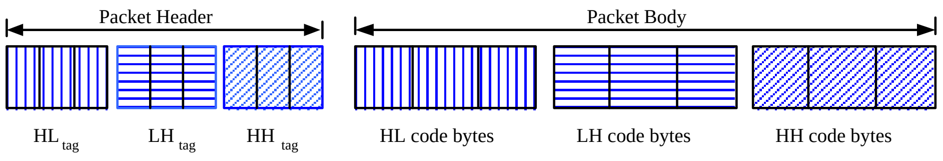


Figure 4.6: Packet Structure

4.3.1 JPEG2000 Packets

As discussed in section 4, a JPEG2000 code stream packet contains data of a component, a tile, a quality, a resolution and a certain precinct. The localization is possible because the DWT preserves it. All progression is achieved through the order of the packets. As JPEG2000 is rather sophisticated, so is its code stream. Associated with each tile is a single **pack stream** (consisting only of packets), which may be segmented into tile parts. The packet headers are not indicated by marker sequences as in JPEG (see section 3), but by a **tag tree** encoding scheme. This coding procedure is applied to two dimensional integer arrays and is able to exploit the redundancy between neighbouring elements. A detailled description can be found in [47] (p.384-89). The packet headers contain the packet and header length, which are necessary to find the next packet header and the packet body. In more detail, the packet header contain information about the length of all **CCPs** (code block contributions) within the scope of the packet. The sum of all CCP lengths is the length of the packet. As figure

4.6 indicates, the contributions are ordered by subband. Therefore the whole code stream has to be parsed and the tag tree built up consecutively. Hence the parsing for packets is quite a time consuming and computationally intensive task. However, JPEG2000 is not just sophisticated in its complexity but also in its encoding options. The procedure described above is rather error-prone. Every bit lost or added will destroy the rest of the JPEG2000 code stream, every packet header bit (containing packet or header length) flipped will destroy the rest of the code stream. Since the transmission over noisy channels was considered in the JPEG2000 standardization, there is an error-resilient mode where packet header markers are included.

4.3.2 Markers

The two markers **SOP**, start of packet with hexadecimal code `0xff91` and **EPH**, end of packet header with hexadecimal code `0xff92` indicate the start of the header information and the start of the packet body information. Generally JPEG2000 marker sequences are two bytes , of which the first is `0xff`. Values in excess of `0xff8f` are not allowed in the paket body bytes. There are JPEG2000 markers signaling the start and end of the image, of a tile and diverse properties. Markers must not occur in the compressed code stream of the packet body. When parsing for markers, one has to be careful since all values can be taken within the main, tile and tile-part headers.

4.4 JPEG2000 Compression Performance

JPEG2000 performs significantly better than all JPEG modi. Figure 4.7 shows the results achieved with **JJ2000**'s default settings and the settings for lossless mode. Note that even in the lossless mode a decoding at certain bit rates is possible. JJ2000 is a collaborative project among JJ2000 partners (Canon, Ericsson and the Ecole Polytechnique Federale de Lausanne) for a JAVA^{TM} implementation of the JPEG 2000 standard (ISO/IEC 15444). The default settings include lossy mode 5 level wavelet decomposition, usage of one tile and quality (layer) progression. Although the results shown in figure 4.7 reveal a superior coding performance, the real improvement is pointed out by figure 4.8, which impressively demonstrates the low bit rate abilities of JPEG2000. It is also noticeable that the lossless mode is superior to JPEG and close to the default settings (approximately 1dB).

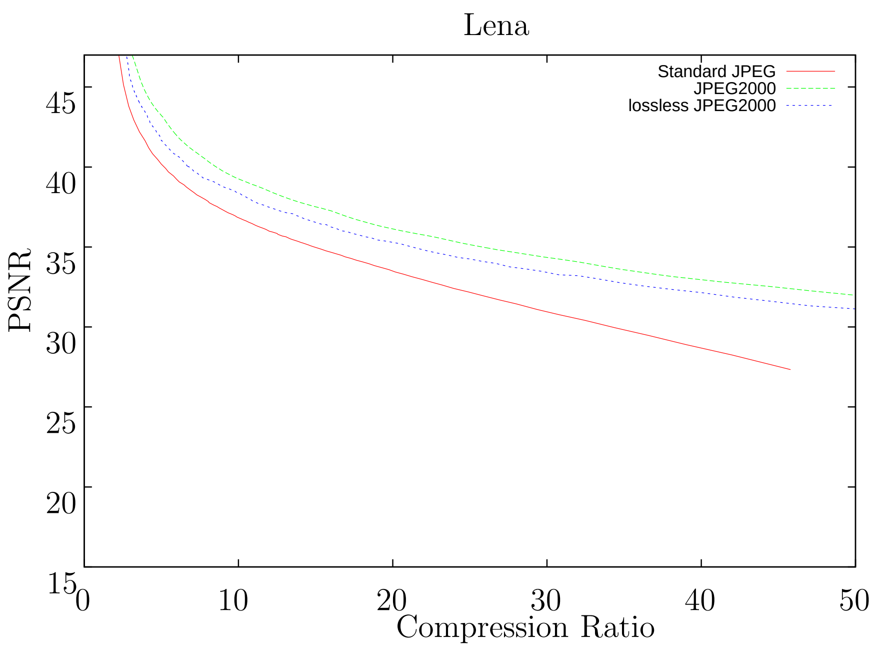


Figure 4.7: JPEG2000 Coding Performance

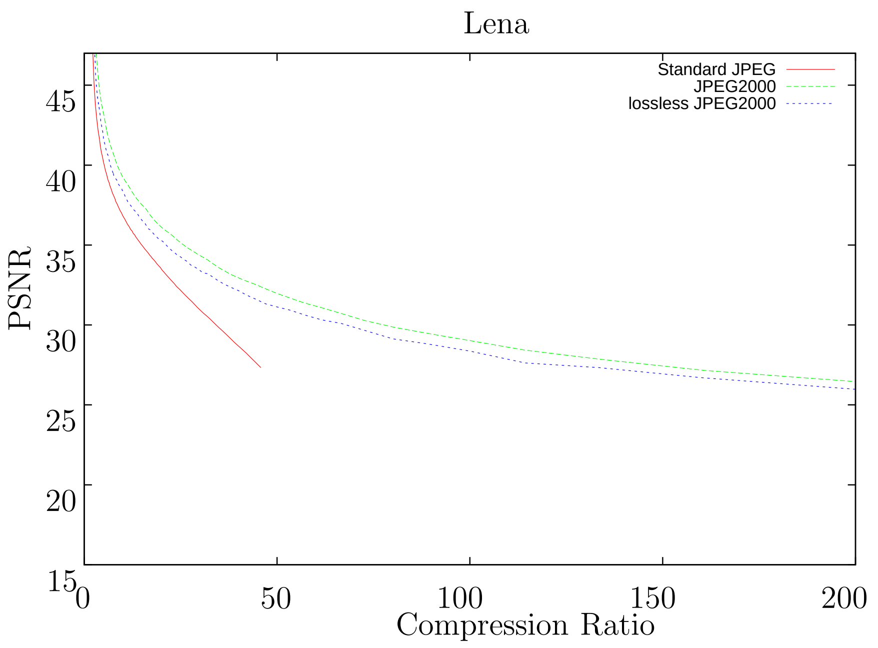


Figure 4.8: JPEG2000 Coding Performance: Low Bit Rates

Chapter 5

Selective Encryption of JPEG2000

In the following section approaches for selective encryption of JPEG2000 are discussed.
Sections 5.1 and 5.2 summarize previous work, then in section 5.3 the encryption of the pack stream is discussed and new approaches are suggested. The topic is partly covered by the emerging JPSEC standard (JPEG2000 Part 8), which is currently under construction. Contributions have been made by [49], [54] and [51]. While [51] wants to solve content based encryption with a new file syntax, [49] and [54] try to preserve format compliance with the JPEG2000 standard.
The format-compliant approach is more generic, since all software that is capable of processing JPEG2000 files can deal with a format-compliant encrypted file. Hence the focus of the following section is solely on format-compliant encryption. The results presented are based on the work of Norcen and Uhl ([31] and [32]), whose support is gratefully acknowledged.
Depending on the encryption approach other useful and convenient properties of code stream can be preserved, such as scalability and error resilience. Various methods have been proposed. The previous work can be divided into two groups: methods which are applied within the compression pipeline and methods operating solely on the compressed code stream. Both are discussed in the following sections.

5.1 Encryption within the Compression Pipeline

Several format-compliant encryption schemes are applied within the compression pipeline:

- Encryption of the sign bits of the wavelet coefficients [19, 45, 59]
- Permutation of bit planes and codeblocks [45]
- Permutation of coefficients [45, 59]

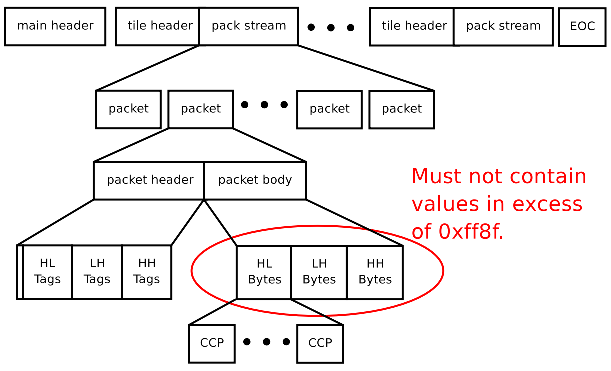


Figure 5.1: Restrictions of the Packet Body

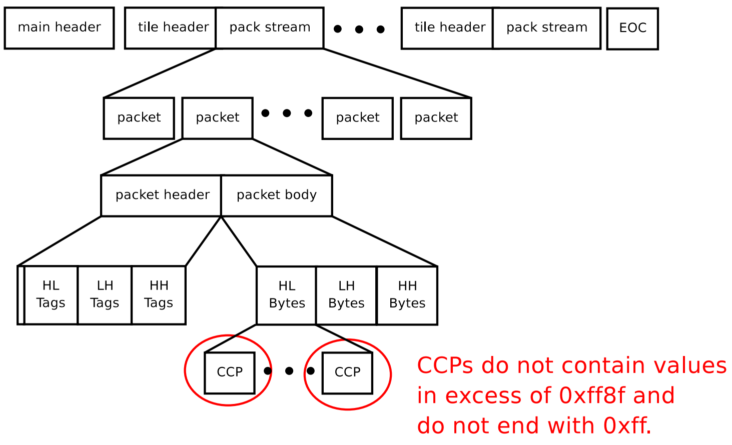


Figure 5.2: Restrictions of the CCPs

- Wavelet packets (the subband partition is kept secret) [35–37]
- Secret wavelet filters [33] [24]

However, there are good reasons for splitting up compression and encryption. Since compression is extremely expensive compared to the sole encryption (in terms of computing power), every encryption/decryption process becomes costly too. Therefore format-compliant encryption that operates on the compressed code stream is discussed more extensively.

5.2 Encryption of the Compressed Code Stream

All these approaches try to encrypt the compressed code stream compliantly. Thereby they yield to minimize the computational effort for encryption while maintaining a high level of security and preserving useful code stream properties, such as scalability. Figures 5.1 and 5.2 illustrate the restrictions induced by the code stream syntax of JPEG2000, namely that the packet body must not contain values in excess of `0xff8f` and that additionally CCPs must not end with a `0xff` byte. However, the last requirement solely avoids marker sequences at CCP borders ([47] p.495) and is therefore of minor importance for code stream compliance.

The tags in the packet header contain the information about the length of the CCPs, the packet body contains the actual arithmetically coded data.

In [32] it is pointed out that there are coding settings which make the identification of the packet headers and bodies easy and computationally inexpensive. This is done by inserting marker sequences (SOP and EPH).

Many format-compliant encryption approaches, as proposed by Norcen and Uhl [32], Wu and Deng [56,57], Wu and Ma [54], Conan [49], Kiya [23], Canon Inc. [20], Lian [44], Grosbois [19], Dufaux, e.a. [15] and Mao [29], operate solely on the packet bodies.

Only Mao suggested an approach that applies a bit stuffing procedure in [55] (taken from the summary of previous work in [56]). Byte or bit stuffing requires the packet header to be corrected and thus imposes an additional computational effort. Encryption is quite likely to produce two byte sequences in excess of `0xff8f`. Whenever such a sequence is produced a zero bit is inserted. As this changes the overall length of the packet body, the packet header has to be updated.

One has to take care that the proposed scheme is actually reversible. In [56] it is mentioned that Canon inc. proposed an encryption scheme [20], which was later found to be irreversible by Wu, Ma and Deng [10]. The influence of the encryption of different subbands, bit planes and code passes on image quality was evaluated in [44]. On basis of these evaluations they suggest to divide the compressed coefficients into a low frequency part and a high frequency part. For the low frequency part the five most significant bit planes (6-10th) of all coefficients are encrypted, while only the cleanup pass of the five most significant bit planes is encrypted.

In the following sections we will discuss approaches from the technical point of view: How is the encryption of the packet body realized? We start with discussing the iterative encryption approach of Wu and Deng [56] and continue with the approaches of Conan [49] and Kiya [23], which are very similar and process the packet body byte per byte. The approach of Wu and Ma [29] also works on a byte basis, but offers more security. In [15] another approach working on a byte basis is described. It offers even higher encryption rate and we will see in section 5.3 that this is even the maximum encryption rate without producing superfluous information. However, a new approach that uses a standard block cipher (AES) with a modified OFB mode will be presented.

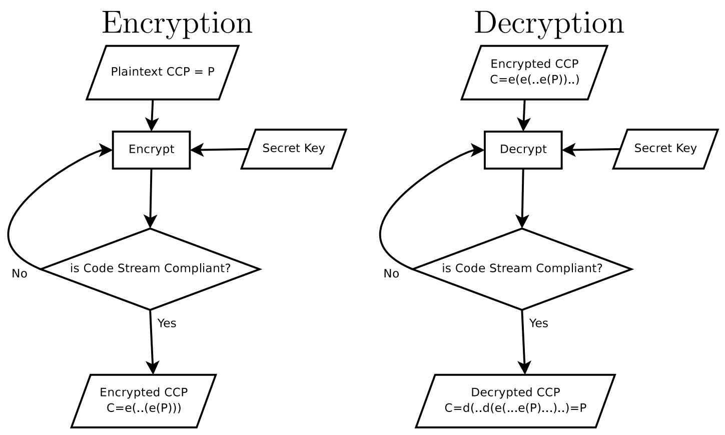


Figure 5.3: Iterative Encryption

5.2.1 Iterative Encryption

A reversible approach based on recursive encryption which works on CCPs was proposed by Wu and Deng in [56]. The CCPs are recursively encrypted until they are code stream compliant. The basic encryption algorithm is the following:
For all CCPs:

1. Encrypt the CCP.
2. Check if it contains a two byte sequence in excess of 0xff8f.
   If yes goto 1 and re-encrypt the encrypted CCP.
3. Check if it ends with 0xff.
   If yes goto 1 and re-encrypt the encrypted CCP.
4. Output the code stream compliant encrypted CCP.

Accordingly, the decryption is done in the following way.
For all CCPs:

1. Decrypt the CCP.
2. Check if it contains a two byte sequence in excess of 0xff8f.
   If yes goto 1 and re-decrypt the decrypted CCP.
3. Check if it ends with 0xff.
   If yes goto 1 and re-decrypt the decrypted CCP.
4. Output the code stream compliant decrypted CCP.

Figure 5.3 shows the flowchart diagrams for encryption and decryption. The feasibility of this approach is discussed in section 5.3.3, where we will also show that in general this approach can not be extended to the whole packet body. Hence it is limited to be applied to CCPs, the borders of which can only be determined through a rather costly tag tree decoding.
Iterative encryption requires that the condition for iteration (in our case code stream compliance) is met by both plain and cipher text. Accordingly only encryption schemes in which the ordering of the encryption does not play a role

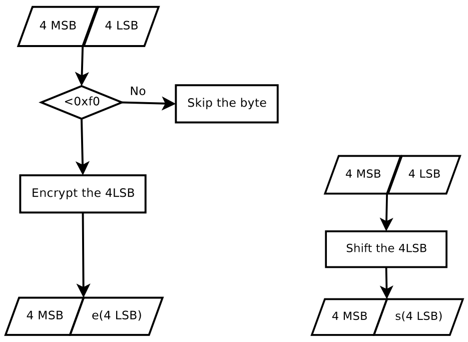


Figure 5.4: Overview of the Approach by Kiya

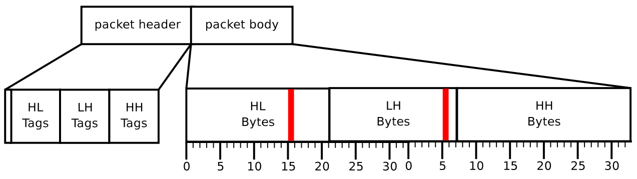


Figure 5.5: Overview of the Approach by Kiya (2)

$(d(e(e(p))) = e(p))$ can be applied. This is met by ECB, but not CFB, OFB, ... In [56] Wu and Deng propose generating a key stream K with the same length as the CCP (n) and add it to the packet body P and take this modulo 256^n to obtain the encrypted CCP C. To decrypt they compute $P = C - K \mod 256^n$.

5.2.2 The Approach of Conan and Kiya

In the approach proposed by Conan [49] only the 4 LSBits are encrypted if the value is less than `0xf0`. A similar approach was presented by Kiya [23]. Additionally to the encryption of the 4 LSBits of bytes below `0xf0`, they suggest a bit shift operation on all 4 LSBits. Furthermore they introduce a parameter m and encrypt/shift only one randomly chosen half byte out of m. Figure 5.4 shows a flowchart of both the encryption and the shifting method, while figure 5.5 illustrates how the parameter m affects the encryption process. The parameter m is set to 33, with the effect that one out of 33 bytes is encrypted.
Figures 5.6 and 5.7 show the result for different m for shifting and encryption. It is noticeable that shifting the 4LSBs requires a significantly higher encryption rate to hide image information than the encryption of the 4 LSBs. Shifting the 4 LSBs does not even hide all image information for the highest security setting (m=1). To achieve a rather similar result for encryption of the 4 LSBs we can only encrypt one out of 10 bytes. It is remarkable, that the error concealment attack does not increase the image quality dramatically. For error concealment the encoder inserts an error resilience segmentation symbol in the MQ codeword at the end

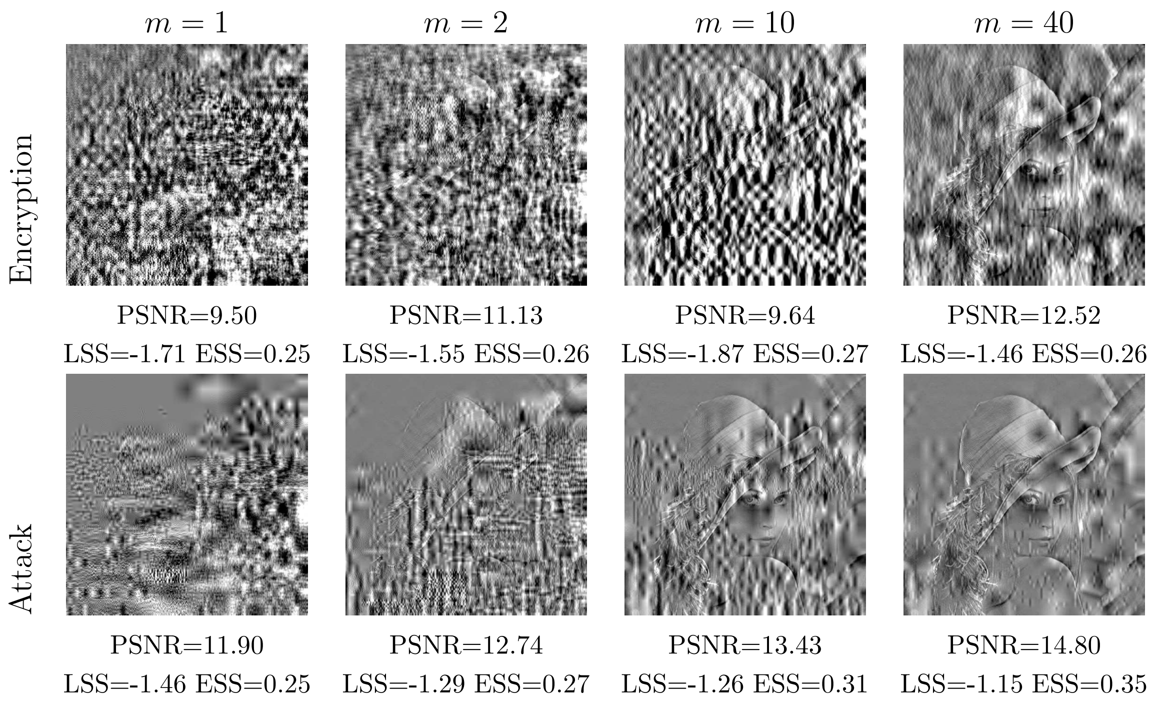


Figure 5.6: Kiya: Shift of the 4 LSB

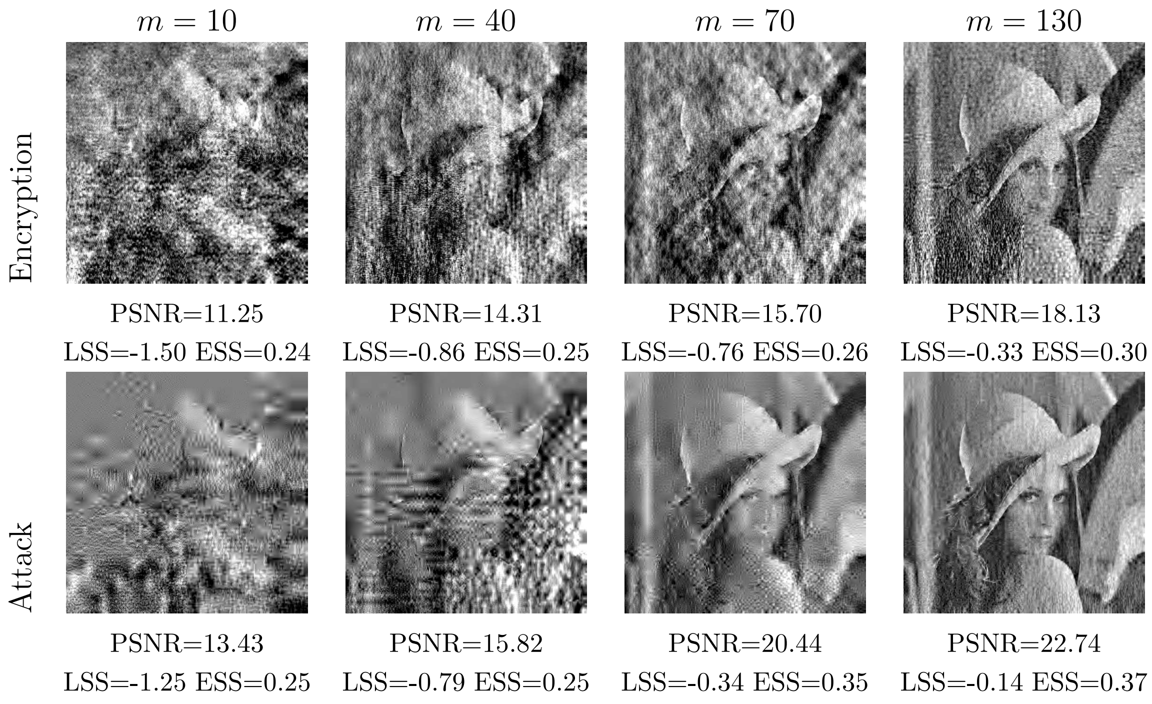


Figure 5.7: Kiya: Encryption of the 4 LSB

of each bit-plane (cleanup pass). The decoders can use this information to detect and conceal the errors induced by encryption. This error concealment method therefore can detect the bitplane which is corrupted and discards all following code block contributions. Hence the distortion introduced is kept minimal. In the Kiya encryption scheme we randomly choose one out of m bytes for encryption/shifting. Hence the code blocks contributions are rather randomly affected. There is no special bias on visually important regions or parts where the error propagation would have a large influence, e.g., the lowest resolution bands. Error propagation takes place, but in a rather reduced fashion (see figures 5.7 and 5.6).
However, the security of both approaches is rather weak, since only half of the content is processed at most. The security of he shift operation is even more doubtful than that of encryption, since the 4 LSBs are mapped to one out of only 4 possibilities and, even worse, one of these possibilities is the identity. For both approaches gradient attacks are at least theoretically possible, especially for large enough ms.

5.2.3 Two Approaches of Wu and Ma

Wu and Ma proposed two approaches [54] which are capable of encrypting packet bodies to 99.22% in average. Both preserve the `0xff` byte and the byte after it. The first of their two approaches applies a stream cipher, while the second works with a block cipher.

An Encryption Scheme based on a Stream Cipher

A stream cipher is used to generate a key stream. The `0xff` bytes are discarded and we obtain a modified key stream S. s_i denotes the i-th byte of the key stream, m_i denotes the i-th byte of the packet body (plain text) and c_i denotes the i-th ciphertext byte. The encryption works byte per byte on the packet body in the following way:

```
if ($m_1$=0xff), then $c_1 = m_1$;
else $c_1 = m_1 + s_1 \mod$ 0xff;
for $i = 2$ to (packet_body_length)

    if ($m_i =$ 0xff or $m_{i-1}$=0xff), then $c_i = m_i$;
    else $c_i = m_i + s_i \mod$ 0xff;
```

Every byte that is not a `0xff` byte or the successor of a `0xff` byte is encrypted to the range [`0x00`,`0xfe`]. In section 5.3.2 this and other encryption methods are discussed. The decryption algorithm works quite similarly:

```
if ($c_1$=0xff), then $m_1 = c_1$;
else $m_1 = c_1 - m_1 \mod$ 0xff;
for $i = 2$ to (packet_body_length)

    if ($m_i =$ 0xff or $m_{i-1}$=0xff), then $c_i = m_i$;
```

else $m_i = c_i - s_i \mod$ `0xff`;

This approach avoids producing values in excess of `0xff8f`, since no `0xff`s are produced and all two byte sequences (`0xff`, X) are preserved.

An Encryption Scheme based on a Block Cipher

This approach preserves all two byte sequences (`0xff`, X) as well, and basically works as follows:

1. Select all bytes of the packet body that are not successors of or a `0xff` byte.
2. Split the selected bytes into blocks and encrypt them iteratively until no `0xff` is contained in the ciphertext.
3. Replace the selected bytes in the original block with the encrypted bytes.

When the number of selected bytes is not a multiple of the block size, ciphertext stealing [41] is proposed. In average the probability that all encrypted bytes are unequal `0xff` is 93.9% for 128-bit blocksize. However, this approach does not contain any feedback mechanisms.
Both approaches are computationally inexpensive and offer quite a high encryption ratio.

5.2.4 An Approach by Dufaux, e.a.

The goal of the proposed approach [15] has been conditional access control, which is quite similar to transparent encryption. The user has only access to certain qualities of an image (usually the low quality version is free as for transparent encryption). Their encryption approach uses a PRNG, namely the SHA1 PRNG with a seed of 64 bit. The encryption procedure is the following:

1. If the packet body byte b_i equals `0xff`, it is preserved.
2. If the previous packet body b_{i-1} byte has been `0xff`, the cipher byte $c_i = b_i + k_i \mod \texttt{0x8f} + 1$, where k_i derived from the PRNG and is in $[\texttt{0x00}, \texttt{0x8f}]$.
3. If the previous packet body b_{i-1} byte has not been `0xff`, the cipher byte $c_i = b_i + k_i \mod \texttt{0xff}$, where k_i derived from the PRNG and is in $[\texttt{0x00}, \texttt{0xfe}]$.

The proposed method to obtain a number in the right range is iterative generation of a random number until it is in the right range. This approach encrypts even more than the approach of Wu and Ma [54].

5.2.5 Confidentiality

It could be shown in [32] that encrypting 20% at the beginning of a JPEG2000 file is enough to hide all significant image information. These rule of thumb is based on JPEG2000 files encoded with the JJ2000 encoder either lossy with a target bit rate of 2 bpp or lossless. Figure 5.8 shows the direct reconstruction and the corresponding error resilience attack - basically the same as the replacement

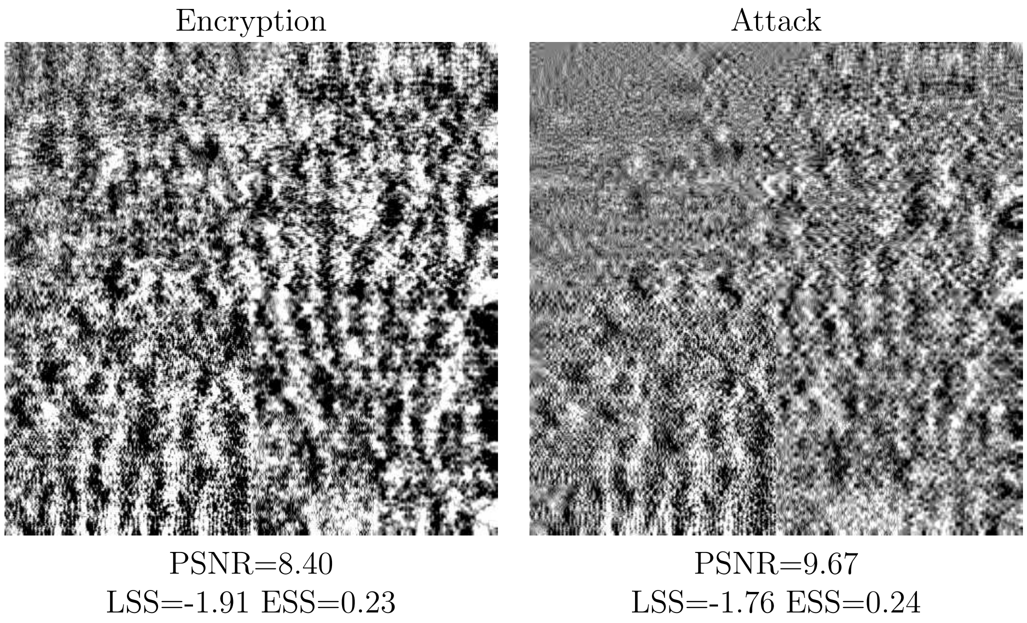


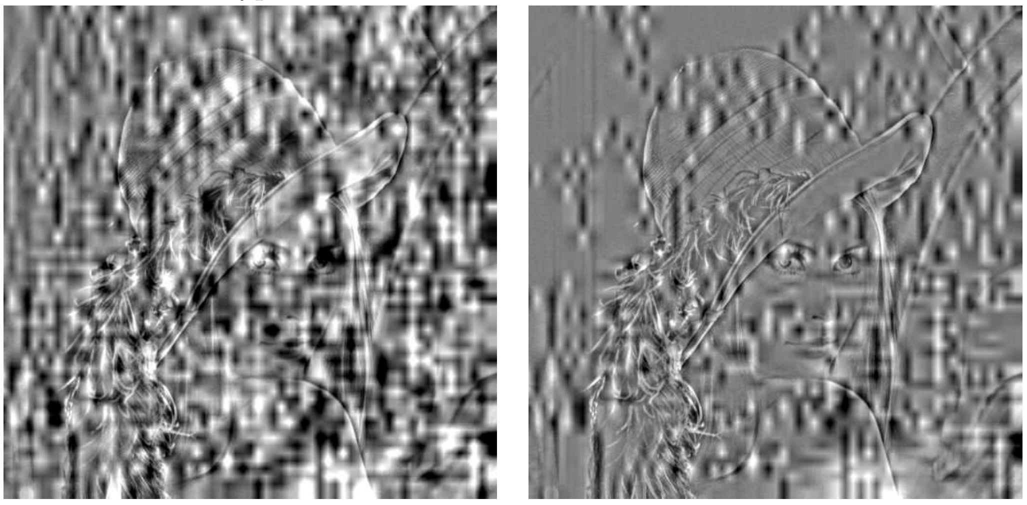
Figure 5.8: Confidentiality with JPEG2000: 20% Encrypted

attack. The term "error resilience" is more appropriate for JPEG2000, since this attack is conducted with JPEG2000's error resilience methods. In order to have an example where too little information was encrypted figure 5.9 shows an example with only 1% encrypted.
This approach can be realized by using either one of the two approaches of Wu and Ma or the CCP based approach of Deng and Wu. For the approach of Conan and Kiya the results can not be applied since the encryption ratio is relatively low (1/2 at most).

Generalization

It could be shown that the encryption of 20% of a JPEG2000 file encoded with JJ2000 at 2 bpp (rate = 0.25) or lossless is sufficient to hide all signifcant image information. It is interesting to see whether this holds for different encoding parameters, e.g., lower rates or less quality layers. JJ2000 default setting produces 22 layers at a rate of 0.25. It turns out that the usage of quality layers is crucial in order that the encryption of 20% hides all signifcant image information. In figure 5.10 the attacked images (encoded with no quality layers) reveal image information, especially for the high compression. If there are many quality layers the code blocks contribute to many different packets. In this way the encryption of a packet effects the decoding of several other packets. But if there are no quality layers (that means there is only one) there is only a single contribution of a code block to a single packet. Therefore no error propagation takes place and nearly all packet bodies have to be encrypted to assure confidential encryption.
When using quality layers this works for lower bit rates as well up to a rate of

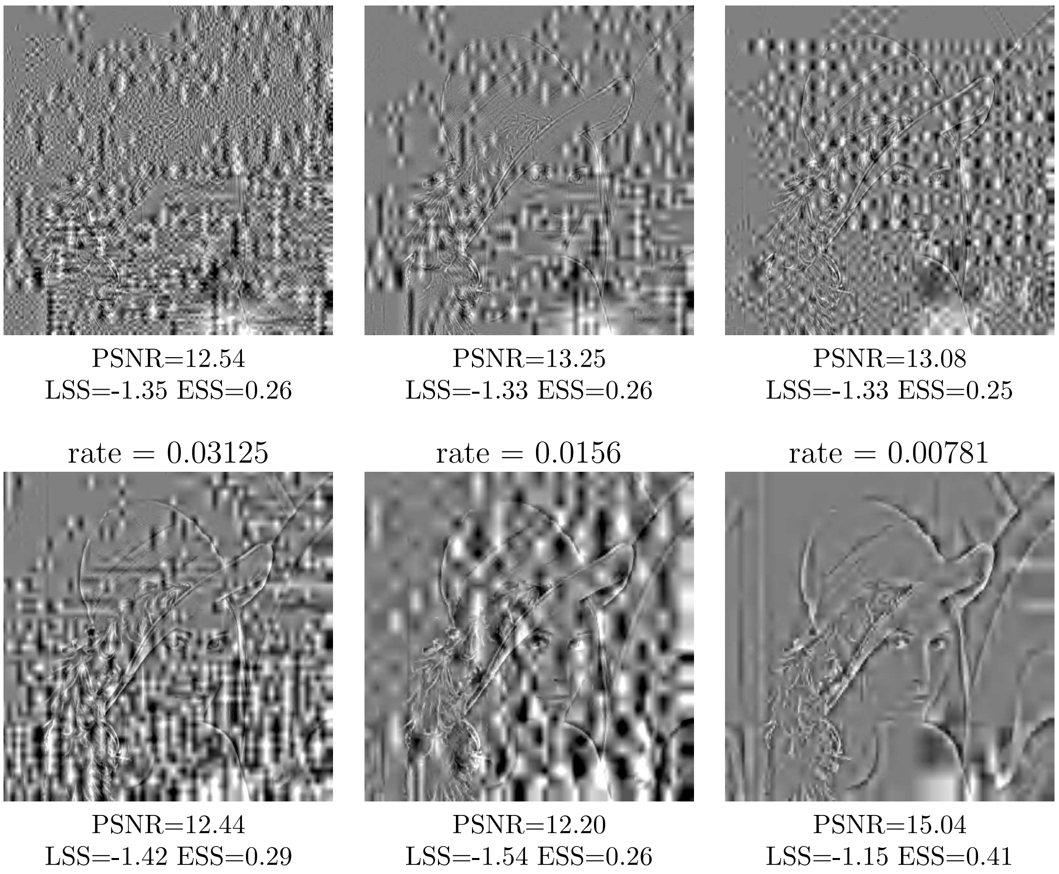
Encryption Attack


PSNR= 11.53
LSS= -1.57 ESS=0.26

PSNR=13.46
LSS=-1.31 ESS=0.29

Figure 5.9: Confidentiality with JPEG2000: 1% Encrypted

rate = 0.25 rate = 0.125 rate = 0.0625



PSNR=12.54
LSS=-1.35 ESS=0.26

PSNR=13.25
LSS=-1.33 ESS=0.26

PSNR=13.08
LSS=-1.33 ESS=0.25

rate = 0.03125 rate = 0.0156 rate = 0.00781

PSNR=12.44
LSS=-1.42 ESS=0.29

PSNR=12.20
LSS=-1.54 ESS=0.26

PSNR=15.04
LSS=-1.15 ESS=0.41

Figure 5.10: 20% Encrypted with no Quality Layers (Attacks)

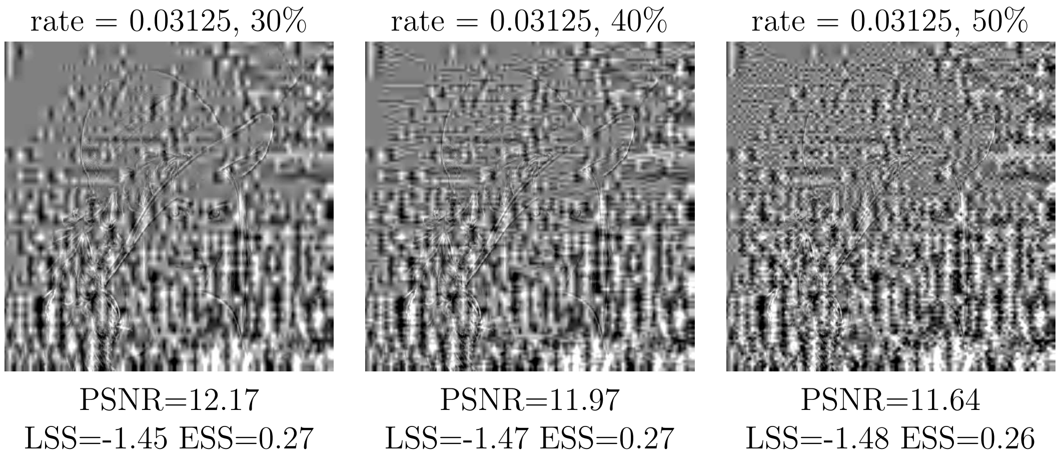


Figure 5.11: 30-50% Encrypted with no Quality Layers (Attacks)

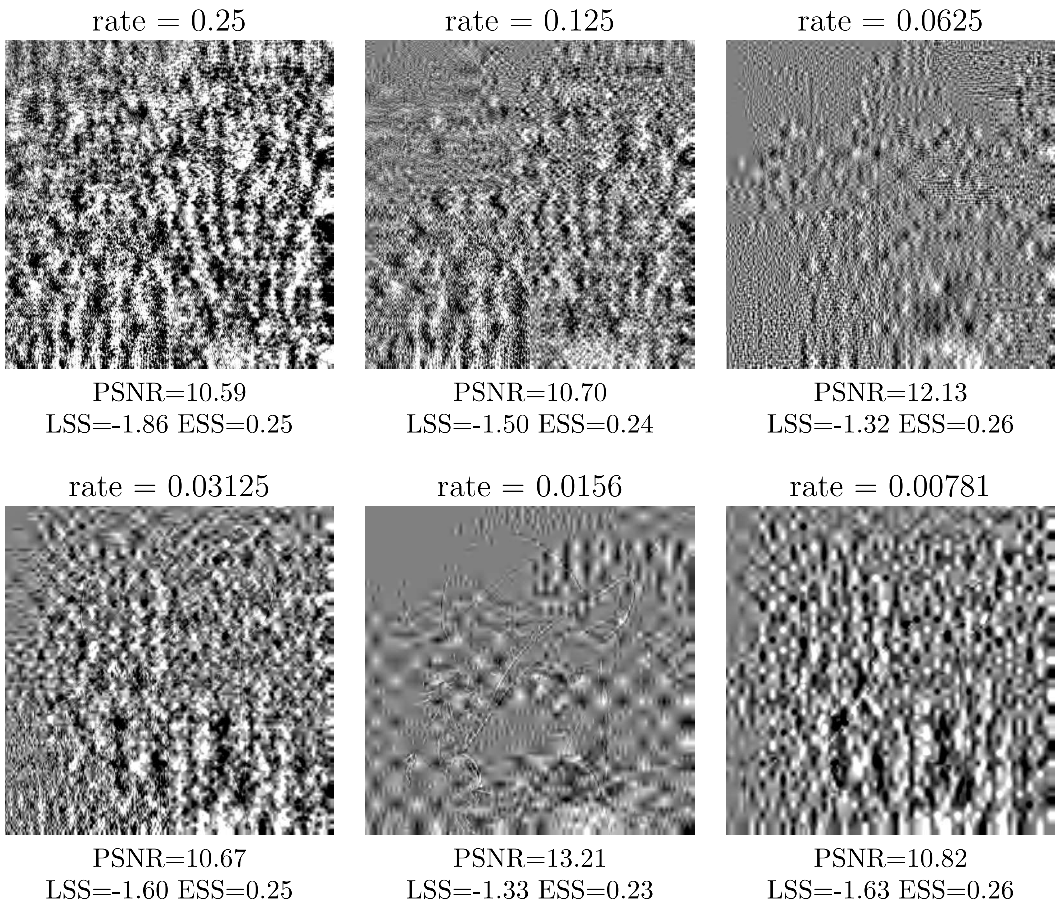


Figure 5.12: 20% Encrypted with Quality Layers (Attacks)

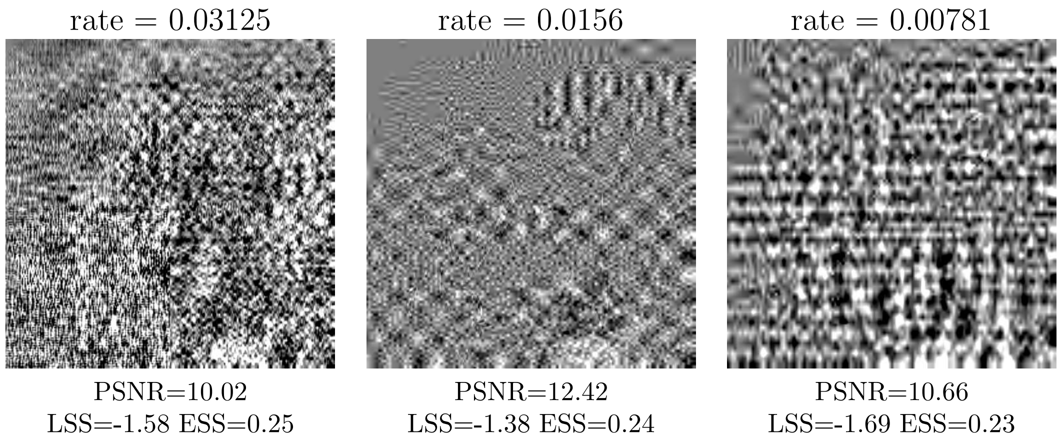


Figure 5.13: 30% Encrypted with Quality Layers (Attacks)

0.0625, where images with fewer quality layers start to reveal sparse image information (only high frequency edge information, which one might only recognize if one knows the original image). Figure 5.12 shows the result of encrypting 20% for various rates.
Starting at the rate of 0.0625 it is also necessary to encrypt more and more of the compressed image. If no quality layers are specified, more than 50% of of the highly compressed images have to be encrypted. If quality layers are used this can be reduced to about 30% (see figure 5.13).
The encryption of 20% is sufficient for rates down to 0.0625 (0.5 bpp); for even smaller rates the encryption of 35% of the file stream is recommended. Note that the figures show error concealment attacks which an arbitrary attacker might possibly never be able to conduct. In order to conduct an error concealment attack, the according information needs to be added by the encoder, this is in general not the case. The attackers only possibility to obtain a better image quality is, to write an JPEG2000 decoder which is capable of ignoring an arbitrary amount of packet body data. Then the attacker is capable of getting close to the result of an error concealment attack. If this attack is not conducted manually with human interaction, suitable quality metrics have to be found. Nevertheless the error concealment attack delivers the maximum of image quality one can deduce from the encrypted JPEG2000 file and will not be easily be achieved by an attacker.

5.3 Analysis of Packet Body Encryption

When encrypting the image content contained in a JPEG2000 pack stream – this means encrypting the packet bodies –, we have two opportunities:

- Encrypt CCPs
- Encrypt packet bodies

Both preserve scalability on a packet basis, while the first even preserves it on a CCP basis. A disadvantage of the CCP approach is that the CCP borders have to be identified, which imposes an overhead. Both possibilities are technically very similar since the only restriction when encrypting a CCP is that no `0xff` is produced at the end. This is not of great importance for the code stream compliance of the cipher (as long as no markers are produced). Apart from this requirement both are the same.
In general we seek a reversible mapping between the code words (of the packet bodies or the CCPs) that is computationally inexpensive and cryptographically secure.
Since the parsing of the packet bodies can be done quite easily and conveniently through marker sequences, approaches based on packet bodies are preferred. As the packet body length is not limited, memory inexpensive approaches have to split up the packet bodies into blocks. The code stream compliance is explained in terms of byte dependencies and therefore in section 5.3.2 we discuss all approaches based on byte-wise encryption. We will show that the `0xff` bytes have to be preserved when applying byte-wise encryption and that the optimal solutions are similar to the approach of Dufaux, e.a. [15]. Furthermore approaches with 100% encryption rate are proposed, but these have the great disadvantage of producing superfluous information that has to be stored outside the packet bodies.
In section 5.3.3 approaches based on fixed-size blocks are discussed. Additionally in section 5.3.4 approaches working on the whole packet body are discussed and one approach is presented.
The question if the CCP based iterative encryption approach [56] can be extended to whole packet bodies is answered in section 5.3.5.
In order to assess the security and feasibility of encryption approaches that operate on the packet body, it will be necessary to determine some properties of those code words, e.g., the number of possible code stream compliant code words for a packet body. This is done in section 5.3.1.

5.3.1 "Countable" Properties of the Pack Stream

We are interested in determining the number of possible code words for a given length. Furthermore we are interested in how the number of `0xff` bytes influences the number of code words. This leads to the following definitions:

Definition 5.3.1 ($\mathtt{CW}_n$)
The number of possible code words for a CCP with n bytes. This is also the number of code words for a pack stream of length n, disregarding the CCPs.

Definition 5.3.2 (C_n)
The set of possible code words for a CCP with n bytes. This is also the set of code words for a pack stream of length n, disregarding the CCPs.

Definition 5.3.3 ($\mathtt{CW}_n^j$)
The number of possible code words for a CCP with n bytes that contain j `0xff`s. This is also the number of code words for a packet body of length n that contains j `0xff`s, disregarding the CCPs.

It turns out that to calculate these numbers we need to know the number of possibilities how to place k `0xff`s in a code word. There is the restriction that no value in excess of `0xff8f` has to be present in a code word, hence no two `0xff`s have to be placed consecutively. Figure 5.14 illustrates this for code words of the length 4. The undashed two boxes are the so called frames.

Definition 5.3.4 ($\phi(n, k)$)
The number of possibilities to disjointly place k frames with two elements (two adjacent elements) in a n element sequence.

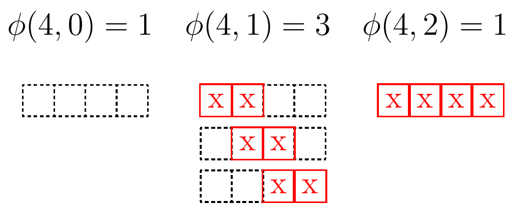


Figure 5.14: An Example of $\phi(4, k)$

Lemma 5.3.1 (Recursive Calculation of $\phi(n, k)$)
The $\phi(n, k)$ can be computed recursively with:

$$\begin{aligned} \phi(n,0) &= 1 \\ \phi(n,j) &= \textstyle\sum_{k=2(j-1)}^{n-2} \phi(k, j-1) \end{aligned}$$

Proof sketch 1 (Lemma 5.3.1)
No frame can be placed in exactly one way.
One frame can be placed $n-1$ ways on a n-element sequence.

$$\phi(n,1) = \sum_{k=2(1-1)}^{n-2} \phi(k, 1-1) = n-1$$

Considering j frames, we place the first frame in front, which leads to

$$\phi(n-2, j-1) \textit{ possibilities.}$$

We proceed with placing the first frame one position back, which leads to

$$\phi(n-3, j-1) \textit{ possibilities.}$$

This can be continued until there are less positions available than the $j-1$ frames would occupy. Since $j-1$ frames occupy $2(j-1)$ positions, this leads to

$$\phi(2(j-1), j-1) \textit{ possibilities.}$$

Summing all up, we obtain:

$$\phi(n,j) = \sum_{k=2(j-1)}^{n-2} \phi(k, j-1)$$

Lemma 5.3.2 (Calculation of $\phi(n,k)$)
Easier $\phi(n,k)$ can be computed with:

$$\phi(n,j) = \binom{n-j}{j}$$

Proof sketch 2 (Lemma 5.3.2)
In order to prove lemma 5.3.2 we have to show that its definition of $\phi(n,j)$ fulfills the equations in lemma 5.3.1. For $j = 0$ we can easily show equality.

$$\binom{n-j}{0} \quad = 1 \quad = \phi(n,0) \tag{5.1}$$

$$\tag{5.2}$$

Now we can apply the recursion for the binomial coefficients for $n - j > j > 0$.

$$\begin{aligned}
\binom{n-j}{j} &= \binom{n-j-1}{j-1} + \binom{n-j-1}{j} && (5.3)\\
&= \binom{n-j-1}{j-1} + \binom{n-j-2}{j-1} + \binom{n-j-2}{j} && (5.4)\\
&\vdots\\
&= \binom{n-j-1}{j-1} + \binom{n-j-2}{j-1} + \ldots + \binom{j+2}{j-1} + \binom{j+1}{j-1} + \binom{j}{j} && (5.5)
\end{aligned}$$

Since $\binom{j}{j} = 1 = \binom{j-1}{j-1}$, we obtain:

$$\begin{aligned}
&= \binom{j-1}{j-1} + \binom{j}{j-1} + \ldots + \binom{n-j-1}{j-1} && (5.6)\\
&= \binom{2(j-1)-(j-1)}{j-1} + \ldots + \binom{n-2-(j-1)}{j-1} && (5.7)\\
&= \sum_{k=2(j-1)}^{n-2} \binom{k-(j-1)}{j-1} = \sum_{k=2(j-1)}^{n-2} \phi(k, j-1) && (5.8)
\end{aligned}$$

Lemma 5.3.3 (Calculation of $\mathtt{CW}_n^j$)

$$\mathtt{CW}_n^j = \phi(n,j) 255^{n-2j} 144^j$$

Lemma 5.3.4 (Calculation of $\mathtt{CW}_n$)

$$\mathtt{CW}_n = \sum_{j=0}^{\lfloor n/2 \rfloor} \phi(n,j) 255^{n-2j} 144^j$$

Proof sketch 3 (Lemma 5.3.3 and Lemma 5.3.4)

The number of all possible code words can be composed by the number of code words containing a certain number of `0xff`*s. Every* `0xff` *restricts the maximum value of the following byte to 143. Hence the number of possible code words with j* `0xff` *at a fixed position is*

$$255^{n-2j}144^{j}.$$

Every `0xff` *byte requires that the next byte is in the range of $[0, 143]$ and therefore further reduces the number of bytes which can take values in $[0, 255]$ to $n-2$. Additionally the number of different possibilities to place the k* `0xff`*s have to be taken into account. Since 2* `0xff`*s must not be adjacent this is equivalent to $\phi(n, k)$.*
Therefore the number of code words with j `0xff`*s is*

$$\mathtt{CW}_n^j = \phi(n, j)255^{n-2j}144^{j}.$$

To obtain all code words of length n, we simply sum all $\mathtt{CW}_n^j$ up - starting with no `0xff` *up to $\lfloor n/2 \rfloor$* `0xff`*s - and obtain*

$$\mathtt{CW}n = \sum_{j=0}^{\lfloor n/2 \rfloor} \phi(n, j)255^{n-2j}144^{j}.$$

Since the presented definition of $\mathtt{CW}_n$ is complex and time consuming, another recursive definition[1] can be used:

Lemma 5.3.5 (Another recursive definition of $\mathtt{CW}_n$)

$$\begin{aligned} \mathtt{CW}_1 &= 255 \\ \mathtt{CW}_2 &= 255 * 255 + 144 \\ \mathtt{CW}_n &= 255 * \mathtt{CW}_{n-1} + 144 * \mathtt{CW}_{n-2} \end{aligned}$$

The great advantage of this definition is that it is a linear recurrence relation which can be easily solved. The solution is a polynom of degree n which can rapidly be calculated.

Lemma 5.3.6 (A direct definition of $\mathtt{CW}_n$)

$$\mathtt{CW}_n = \frac{32}{7289 + 85\sqrt{7289}} \left(\frac{96}{-85 - \sqrt{7289}} \right)^n + \frac{-7257 - 85\sqrt{7289}}{7289 + 85\sqrt{7289}} \left(\frac{3}{2}(85 + \sqrt{7289}) \right)^n$$

Proof sketch 4 (Lemma 5.3.5)

The number of codewords with length 1 is 255.
The number of codewords with length 2 is put together by those that start with a `0xff` *(144) and those that don't (255*255).*
Generally all code words of length n can be constructed by adding a byte with 255 possibilities to the code words with length $n-1$ and by adding a `0xff` *and a byte with 144 possibilities to the code words with length $n-2$.*
Two constraints have to be met that the above consideration holds.

[1]Thanks to cbrown@brownsystems.com and CHHeckmann@gmail.com for their posts in math.sci

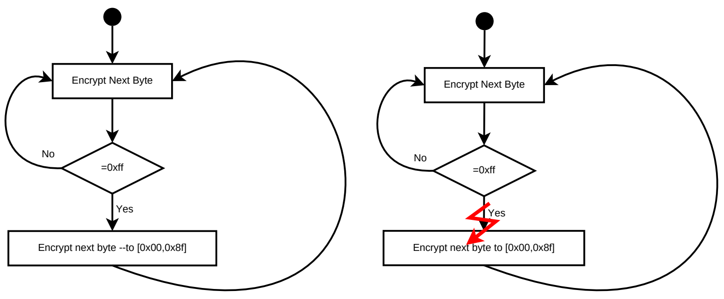

Encrypt Next Byte
No
=0xff
Yes
Encrypt next byte --to [0x00,0x8f]
Encrypt Next Byte
No
=0xff
Yes
Encrypt next byte to [0x00,0x8f]

Figure 5.15: Encryption on a Byte Basis: Problems

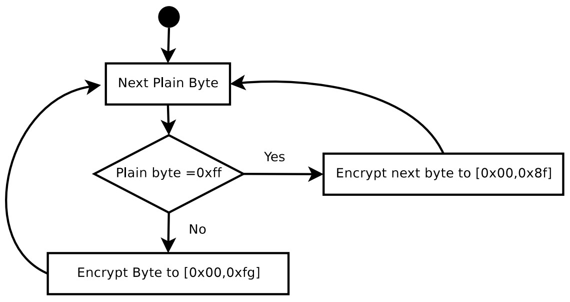


Figure 5.16: Encryption on a Byte Basis: Solution

1. *All code words of length n can be constructed this way.*
   Let $c_n \in C_n$ be an arbitrary but fixed code word. Then the first $n-1$ byte are either in C_{n-1} or end with `0xff`*. In the first case they are in the set of code words constructed by concatenating the code words with length $n-1$ with a byte with 255 possibilities. In the second case the first $n-2$ bytes must be code stream compliant and therefore in C_{n-2} (otherwise c_n could not be code stream compliant). Hence c_n can be constructed by appending* `0xff` *and a byte with 255 possibilities to code word in C_{n-2}.*
2. *No code word is counted more than once.*
   The two sets C_{n-1} and that resulting from appending a `0xff` *to a code word in C_{n-2} are disjoint, since $c \in C_{n-1}$ must not end with a* `0xff`*.*

5.3.2 Encryption on a Byte Basis

An optimal cipher can be considered to behave like a random function, hence encryption byte per byte does not preserve code stream compliance.
An approach to change this could be to change the encryption function, such that whenever a `0xff` is produced it is not followed by a byte larger than `0x8f`. This would yield to 2 encryption functions, one from $[0, \texttt{0xff}]$ to the same range and another applied after a `0xff` only mapping to $[0, \texttt{0x8f}]$.
This is not possible because an arbitrary byte can be mapped to `0xff` and the next plain byte may take an arbitrary value in $[0, \texttt{0xff}]$. This byte has to be mapped to $[0, \texttt{0x8f}]$, but there is no bijective function between sets of different cardinality. Figure 5.15 illustrates how this happens, while figure 5.16 shows the

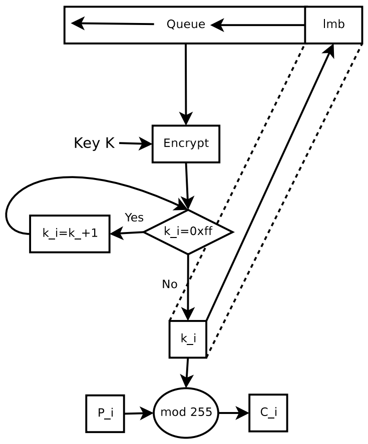


Figure 5.17: A Modified OFB Mode

proposed solution.
In fact the only possibility to avoid the production of too many bytes (when processing byte per byte) is to preserve the `0xff` byte. Actually, we map every byte in [`0x00`, `0xfe`] to [`0x00`, `0xfe`], leave every `0xff` unchanged and map every byte after `0xff` to [`0x00`, `0x8f`].
Therefore one encryption method that encrypts from [`0x00`, `0xfe`] to [`0x00`, `0xfe`] and another that encrypts from [`0x00`, `0x8f`] to [`0x00`, `0x8f`] are needed.
In the next section possible solutions are discussed.

Modified Encryption

The problem of encryption methods that do not work like standard ciphers on bits and bytes has already come up in the field of format-compliant JPEG and MPEG encryption [52,53]. The proposed approach is capable of encrypting arbitrary code words. Thereby the code words are grouped into sets with cardinality 2^k. Since every number has a binary representation, this is always possible. Then every code word in a set is assigned number ($\in [0, 2^k - 1]$) in a fixed length binary representation. Encryption is performed in the following way:

- The code words belonging to certain set are selected, mapped to their fixed length number and those fixed length numbers are then concatenated.
- The concatenation of the fixed length numbers is the encrypted using a standard cipher, such as AES in CFB mode.
- The result is then mapped back to the original code words.

This approach shuffles code words in a set (cryptographically secure). In the case of encryption of [`0x00`, `0xfe`], this has to be divided into sets with cardinality 128,64,32,16,8,4,2 and 1. Hence one byte would always stay the same. For the

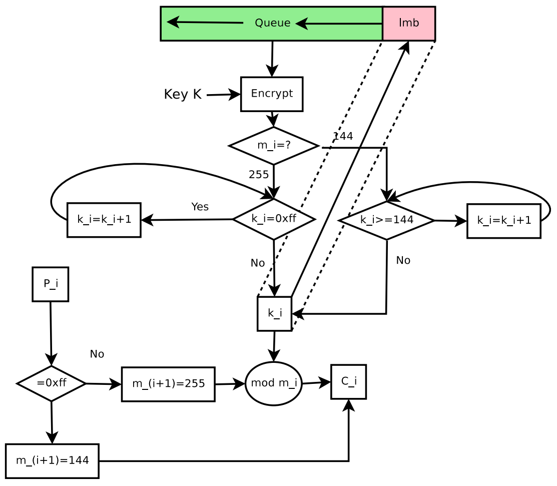


Figure 5.18: A Modified OFB Mode for Packet Body Encryption

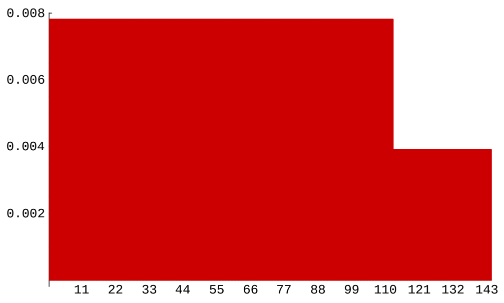


Figure 5.19: The Bias for Encryption Modulo 0x8f+1

encryption of bytes in [`0x00`, `0x8f`] this has to be divided into sets with cardinality 128 and 16. This approach is rather complex and furthermore the goal is to encrypt the pack stream byte per byte, which is not possible with this approach. Other approaches have to be considered.

For encryption of bytes in [`0x00`, `0xfe`] a modified CFB/OFB mode -similar to the stream cipher approach of [54]- leads to a simpler solution. The OFB mode encrypts a queue (that is initially set to an unique initialization vector) and xors the left most queue byte (k_i) with the plain text byte (p_i) to obtain a cipher text byte (c_i). The usual xor of the feedback mode is replaced by an addition of k_i (randomized through encryption) and the plain text packet body byte modulo 255 (see figure 5.17). Additionally all k_i that are equal to `0xff` have to be discarded, because otherwise a bias would be introduced, since the plain text would be mapped to itself for both 0 and `0xff`. If the feedback (indicated by the dashed lines) is taken from the cipher text (c_i), this mode would be equivalent to the CFB mode. In case of encryption of bytes in [`0x00`, `0x8f`], simply applying

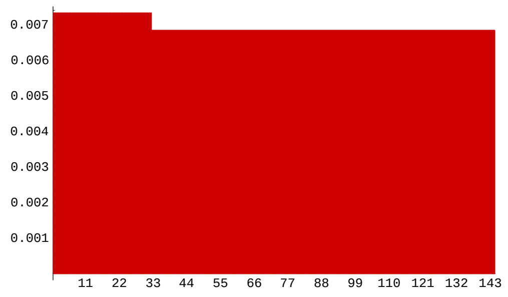


Figure 5.20: The Bias for Encryption Modulo 0x8f+1 with k_i $\in [0, 2^{11} - 1]$

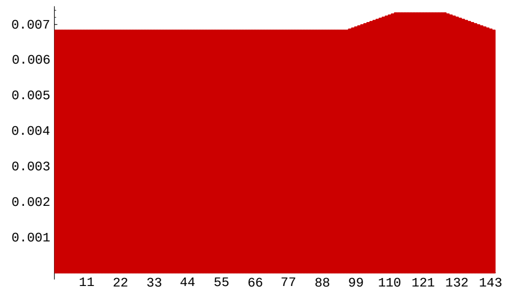


Figure 5.21: The Bias for a Modified Encryption Modulo 0x8f+1

addition modulo `0x8f` + 1 leads to the situation that is illustrated in figure 5.19 for the plain byte `0x00`. The distinct outcomes of encryption (from 0 to 143) and their relative frequencies (y-axis) are plotted. Perfect security would require that every possible outcome of encryption has the same probability.

One can see that a bias is introduced for the first 112 values.

The figure illustrates the situation where the outcome of k_i is assumed to be perfectly equally distributed. To increase security, the bias has to be reduced or omitted.

If every k_i in excess of `0x8f` is discarded (every k_i is discarded with a probability 112/256), we obtain an unbiased outcome. Figure 5.18 the modified OFB encryption algorithm is illustrated. According to the previous plain text byte the k_i are discarded. If the previous plain text byte was a `0xff`, then every k_i in excess of `0x8f` is discarded, else all k_i equal to `0xff` are discarded. The modul is also adjusted according to the previous plain text byte. If the previous plain text byte was a `0xff`, then the modul is set to `0x8f`+1 (144), else the modul is set to `0xff` (255). This approach is very similar to the approach in [15], but uses a modified OFB mode instead of a PRNG.

Other approaches yield to reduce the bias. The bias can be reduced by modifying the addition modulo `0x8f`+1. This could be simply achieved by increasing the

range of the k_i. Figures 5.20 shows the outcome for p_i=0 and k_i in [0,2047]. However, if we replace k_i+p_i mod `0x8f`+1 by k*_i+k_i+p_i mod `0x8f`+1, where k*_i is a half byte of the queue, we obtain less bias as well. In figure 5.21 this is illustrated for p_i=0. The bias introduced is rather the same (the minimum probability of all values divided by the maximum probability of all values is 0.9333 for both), but the complexity of the modulo operation is reduced, since the numbers of which the rest modulo 144 is calculated are only between [0,255+127] instead of [0,2047].
The modified addition approaches and the discarding of the bytes do not differ greatly in security and encrypt 99.61% (255/256) of the packet body bytes in average.

Encryption on a Byte Basis Producing Superfluous Information

If the packet body is encrypted using standard ciphers, values in excess of `0xff8f` are produced. One can sacrifice a code word (e.g. `0xff8f`) to signal that a non code stream compliant byte has been produced and store the value elsewhere (e.g. comment field of the header).
There is an improvement of this approach, but it requires limited random access. Furthermore we need an encryption function that encrypts bytes in the range of [`0x00`, `0x8f`] to the same range. Whenever a `0xff` is found in the plain text, the next byte is encrypted to [`0x00`, `0x8f`] and its value and position stored in the plain queue. Whenever a `0xff` is produced in the cipher text and a stored value is available, it is inserted and the cipher byte is inserted at the stored position. If no stored byte is available from the plain queue, the value and the position are stored in the cipher queue. This procedure reduces the information that has to be stored outside the encrypted pack stream. It can be applied over package borders.
Since this approach does not consider the CCPs, `0xff`s are a little bit more likely in the cipher text, thus likely producing more bytes.

5.3.3 Encryption on a Block Basis

Changing the size of the processing unit does not change the dependency on a byte basis. It only becomes more complex and more difficult to see. Hence every block based approach must lead to superfluous information or again preserve at least the `0xff` bytes.
However, one would suggest applying the iterative encryption [56] approach to blocks of the pack stream. This is not possible, because in order to assure code stream compliance no sequences in excess of `0xff8f` must be produced at the block borders. Hence either a `0xff` byte has to be avoided at the end of a block or the value of the first byte must not excess `0x8f`. Both conditions are not met by the plain text block and therefore the plain text can not be distinguished from an encrypted cipher text that needs to be decrypted once more.

However, iterative encryption and encryption on a block basis can be combined in order to reduce the superfluous information. If a block ends with `0xff`, its length is increased by 1 byte and this information is recorded. Hence every block produces one bit of superfluous information. In fact it is far more likely that the last byte of a block does not end with a `0xff`, hence this bit sequence can be coded very efficiently. Since CCPs must not end with a `0xff`, its probability in the packet body should be below $\frac{1}{256}$.

5.3.4 Encryption of the Entire Packet Body

If we can ensure enough memory or a maximum packet body length (through according encoding settings, e.g., specifying many small intermediate layers), encryption approaches for the whole packet body become feasible. It is possible to encrypt the whole packet body in a way such that we have a reversible mapping between code words with a fixed number of `0xffs` (C_n^j). One can get an impression of the cryptographic complexity of that approach by counting all possible mappings between these code words (C_n^j). There are $\mathtt{CW}_n^j!$ bijective mappings between those code words. In lemma 5.3.3 and lemma 5.3.2 we showed that there are $\binom{n-j}{j}255^{n-2j}144^j$ code words of length n with j `0xffs` ($\mathtt{CW}_n^j$).
An encryption procedure that is a secure mapping applies encryption byte per byte as discussed in section 5.3.2. and then reversibly and securely changes the positions of the (`0xff`, x).

5.3.5 Feasibility of the Iterative Encryption Approach

When applying the iterative encryption approach of [56], it is interesting how many iterations are required. In fact the number of iterations also determines if this approach can be extended to whole packet bodies. If the number of iterations does not increase too much with the number of plain text bytes, the iterative encryption could be applied to packet bodies as well. Since $\mathtt{CW}_n$ is the number of possible codewords, the probability of randomly generating a code stream compliant encryption is $p_n = \frac{\mathtt{CW}_n}{256^n}$. Let $q_n = 1 - p_n$ then the expected number of rounds can be written as:

$$E_n = p_n + 2q_np_n + 3q_n^2p_n + \ldots = \sum_{k=0}^{+\infty}(k+1)q_n^kp_n = \frac{1}{p_n}$$

Figure 5.22 shows the expected number of rounds and figure 5.23 shows the probability of a code stream compliant encryption for 1,2,3 and 4 rounds (both for up to 512 byte). The expected number of rows is 2.5 at most, hence the encryption of the CCPs works properly.
In figures 5.24 and 5.25 the results for up to 5120 byte are shown.
Actually, the encryption of packets with iterative encryption is not feasible, as the number of expected rounds increases exponentially. At about 4000 byte the expected rounds exceed 1000.

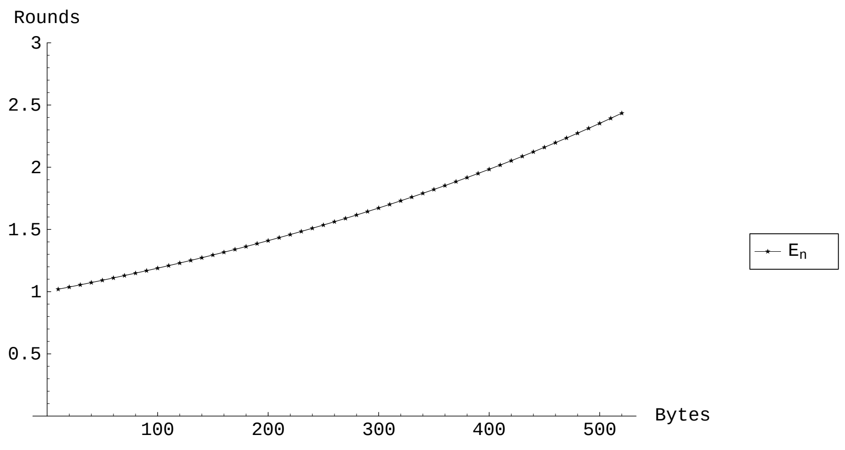


Figure 5.22: Expected Rounds for up to 512 Byte

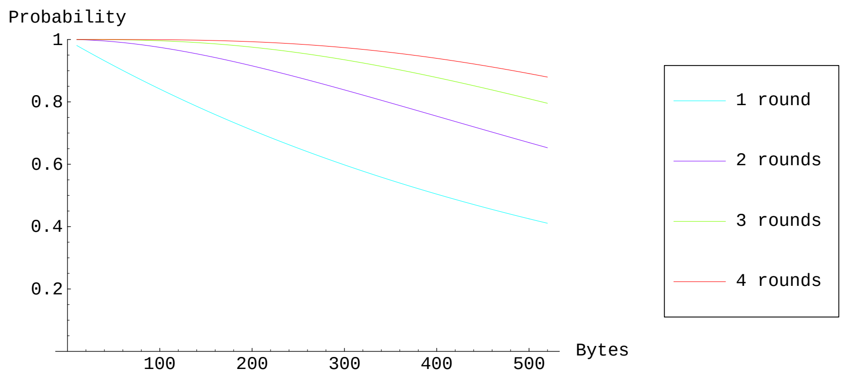


Figure 5.23: Probability of a Format-Compliant Ciphertext for up to 512 Byte

But if we can assure a maximum packet size ($\leq$ 1623) this approach remains feasible, as the number of expected rounds is below 16. In OFB and CFB mode [41] the queue with the size of a block (usually 16 byte for state-of-the-art ciphers, such as AES) is encrypted for a single byte. Hence the overhead of iterative encryption and OFB/CFB mode does not greatly vary when the length of the packet bodies can be kept below 1623.

5.4 Conclusion

A high percentage of the JPEG2000 packet body can be encrypted on a byte basis, which makes the encryption process extremely lightweight in terms of computational effort. When encrypting JPEG2000 packet bodies without producing superfluous information 99.61% of the packet bodies can be processed. The com-

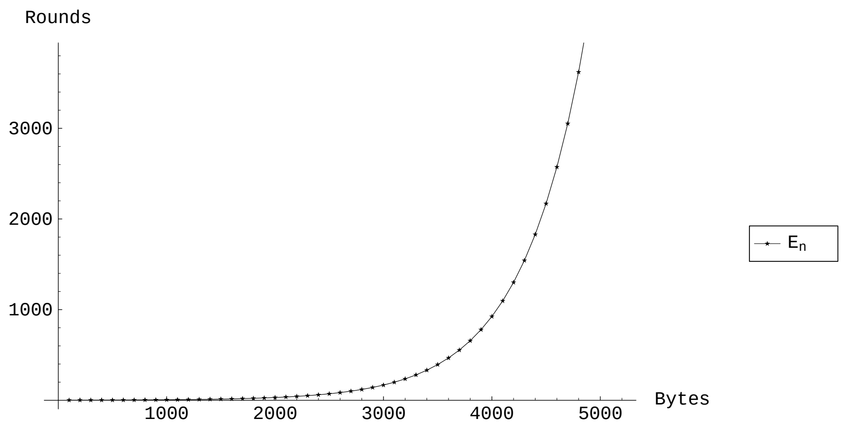


Figure 5.24: Expected Rounds

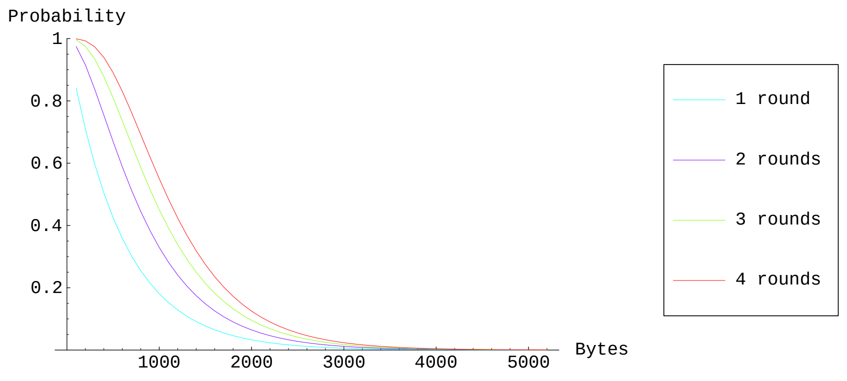


Figure 5.25: Probability of a Format-Compliant Ciphertext for up to 5120 Byte

putational effort is very close to common OFB/CFB modes. For most applications this far more than sufficient. However, the packet body structure and the position of the `0xff` are a strong fingerprint of the image. Hence if one has the original image one can identify it in the encrypted domain. To fully encrypt the package body it has to be encrypted as whole, split up into the CCPs or superfluous information is produced. If one can assure a maximum packet body length through according encoding settings, the iterative encryption and the encryption of the packet body as a whole become feasible.

Chapter 6

A Secure Visual Pipeline in a Grid Environment

The secure visual pipeline has been developed with the Grid software **Globus** [2] and will serve as middleware extension for the Austrian Grid users. The Grid is a rather wide and widely used term, which has not yet been exactly defined. However, we will not enter the discussion what the Grid really is, but present a short overview how the Grid or more precisely the Grid software Globus is used in our secure visual pipeline. Basically the Globus toolkit enables a distributed user and authorization management based on **CAs** (Certification Authorities) and X.509 certificates [38]. According to the authorization granted through a CA grid users can utilize grid resources, like computation time and storage. This usage of a grid resource may also be the simple log on and usage of an ordinary work station, which is capable of user authentication via Globus CA accounts.

The Globus toolkit still has many limitations, e.g., there is absolutely no interactivity, because its origins are in batch processing. This issue has been discussed in [39] and a solution has been presented, namely the tool **glogin**, which enables direct interactive communication between Grid nodes. Furthermore this tool offers functionality similar to ssh. So there is a solution for text based applications to be used easily in a Grid environment, but there is nothing comparable for applications with a graphical user interface. Highly complex and time-consuming simulations, which are calculated in the Grid, but have to be visualized on a simple grid node have been major motivations for Grid computing. Although the previous example was the original motivation, a possible application of Grid technology could be the access control and distribution of multimedia content, e.g., digital television or network gaming. Hence a flexible tool for the transmission of visual data in Grid environment is needed.

6.1 GVid - Video Coding and Encryption for Advanced Grid Visualization

The **GVid** software is a result of a joint project of the Institute of Graphics and Parallel Processing (GUP) at the Joh. Kepler University Linz and the Department of Computer Sciences at the University Salzburg, which included Thomas Köckerbauer, Dieter Kranzlmüller, Martin Pollack, Herman Rosmanith, Thomas Stütz and Andreas Uhl.

6.1.1 The Structure of GVid

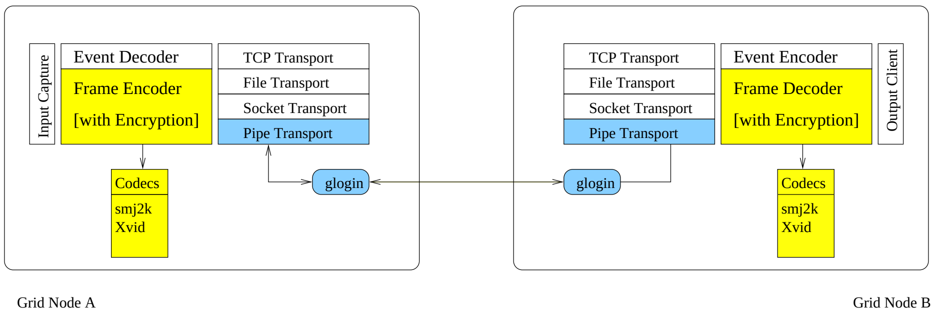


Figure 6.1: GVid Component Overview

One goal of the project was to be capable to use as many applications as easily as possible within the Grid.
Therefore several input adapters exist, that are responsible of acquiring the visual data of the application. Currently a freeGLUT [1], a vtk [3] and a X11 based input adapter are implemented. The X11 input adapter enables every X11 application to be transmitted over the Grid by simply grabbing its window. In this fashion a Grid based remote desktop can easily be realized by grabbing the root window. A more sophisticated approach is a modified freeGLUT library, which enables Grid transmission for every application linked against it. A similar approach is implemented for the vtk.
The frame encoder is capable of using several compression codecs, currently XviD (a MPEG4 based codec) [4], **smj2k** (a JPEG2000 based codec with selective encryption capabilities) and **smpjpg** (progressive JPEG with selective encryption, which is still work in progress) are supported. In the frame encoder also selective encryption is realized (for smj2k and smpjpg). In sections 6.1.2 and 6.2 the visual quality and the computational complexity of these approaches are discussed.
The transport layer offers several options, one of them is capable of using the Grid infrastructure. It uses the tool glogin for the transportation of both video and event data.
The output client is capable of visualizing the application and capturing all driven user events. Figure 6.1 illustrates the structure.

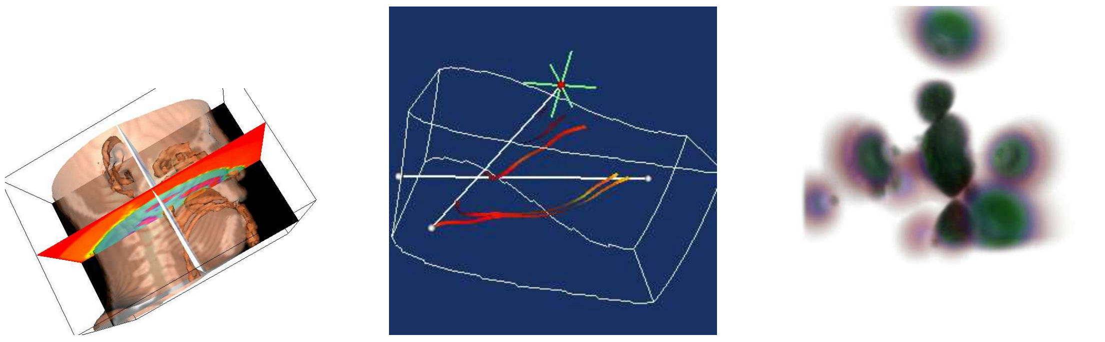


Figure 6.2: Screenshots of Demo Applications: Medical, Streamlines and Raycast

6.1.2 A Comparison of smj2k and XviD within GVid

When using GVid with XviD encoding it is expected to have higher encoding efficiency, because the codec also considers temporal redundancy for coding. Therefore we had a look at the compression performance of both codecs in order to verify this assumption.
In order to compare two video codecs, we have to assess their performance and features. The perceptual quality of visual data is usually measured with the PSNR. In fact the correlation between perceived image quality and PSNR is rather loose. Hence several attempts have been made to find an image metric which comes closer to the quality assessment of the human visual system. We applied two approaches, namely LSS/ESS (luminance similarity score and edge similarity score as described in section 3.1.1) and SSIM, the structural similarity index [58].
The SSIM is an image metric, which tries to measure image quality through structural similarity and ranges from 0 (no similarity) to 1 (high similarity).
For doing our codec comparison we used three different typical visualization applications shipped with the downloadable distribution of vtk. All of them are navigable by the user the same way. Some typical screenshots from these applications can be seen in figure 6.2.

- Medical
  This example reads a volume dataset of a human skull, extracts two isosurfaces that represent the skin and bone, creates three orthogonal planes and displays them. The rendered images are very colorful and offer a good mixture of sharp edges and diffuse areas.
- Streamlines
  With this example one can use two LineWidgets to seed and manipulate streamlines. The images produced by this demo are characterized by a flat background and a few very sharp and thin lines, challenging a high frequency response from the codec.
- Raycast
  The demo creates the opposite scenario to the Streamlines demo, a quite

diffuse volume rendered image of an iron proton.

In order to create a fair testing environment for the codecs we once recorded what we think of as typical interaction with all three applications, consisting of rotation of the whole scene, followed by panning, a full zoom in and a zoom out operation. We then created a special setup, allowing us to repeatedly reproduce this exact interaction with the applications at different quality and bandwidth settings for comparing the codecs' quality results.

6.1.3 Results

For both codecs (XviD and smj2k) the image quality has been evaluated for two different parameter settings. Several image quality measurements have been evaluated, namely PSNR, LSS/ESS and SSIM. Both codecs have been evaluated for various bit rates. XviD is tested without motion estimation (labeled `xvid_npf` in the plots) and with motion estimation (labeled `xvid` in the plots). In the case of motion estimation only p-frames are used.
The parameter setting for XviD with no motion detection is useful for having a comparable codec for smj2k, because both do not use temporal redundancy. On the other hand there are scenarios (e.g., transmission over noisy channels) where the error propagation of XviD with motion estimation is a drawback.
Furthermore if we assume that the transmission channel is limited (which is always the case) and only used for a single video transmission (which is not always the case), only the maximum bandwidth used is of interest, because if the maximum bandwidth exceeds the limit of the transmission channel, lacks occur and resynchronization has to take place. Thus with these assumptions, the motion compensation does not bring much benefit, because in the worst case nearly no or no temporal redundancy can be exploited. MotionJPEG2000 is tested with (`smj2k`) and without quality layers (`smj2k_noilyrs`). Furthermore the reversible integer wavelet transform has been used. The usage of quality layers reduces the percentage of data to be encrypted. Additionally SOP and EPH markers have been inserted, because the selective encryption approach implemented in smj2k is based on these markers.

The Results for the Medical Demo

The medical demo has quite a lot of movement in it. The first 40 frames show a 90 degree rotation of a human skull around both x- and z-axis. In the next 30 frames the top view of the skull is moved up and down. Then it is zoomed into the top view until frame 110. For the next 110 frames it is zoomed out. The PSNR plot surprisingly shows a head to head lead of MotionJPEG2000 with no quality layers (`smj2k_noilyrs`) and XviD with p-frames (`xvid`) (see figure 6.3). The worst performance shows XViD with no p-frames (`xvid_npf`) and `smj2k`. However, very low bit rates could only be achieved with the smj2k codec with no quality layers. The LSS shows the XviD codec in the lead (with and without

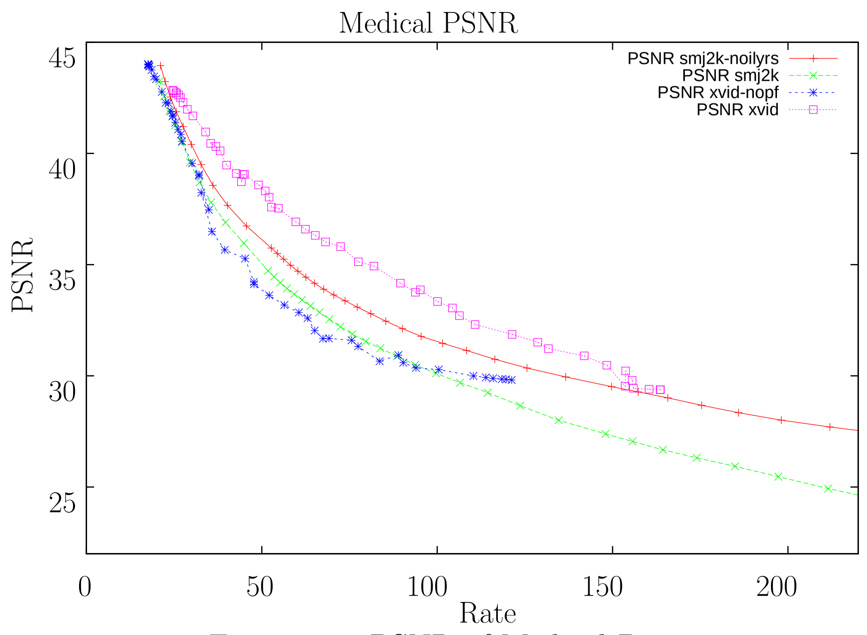


Figure 6.3: PSNR of Medical Demo

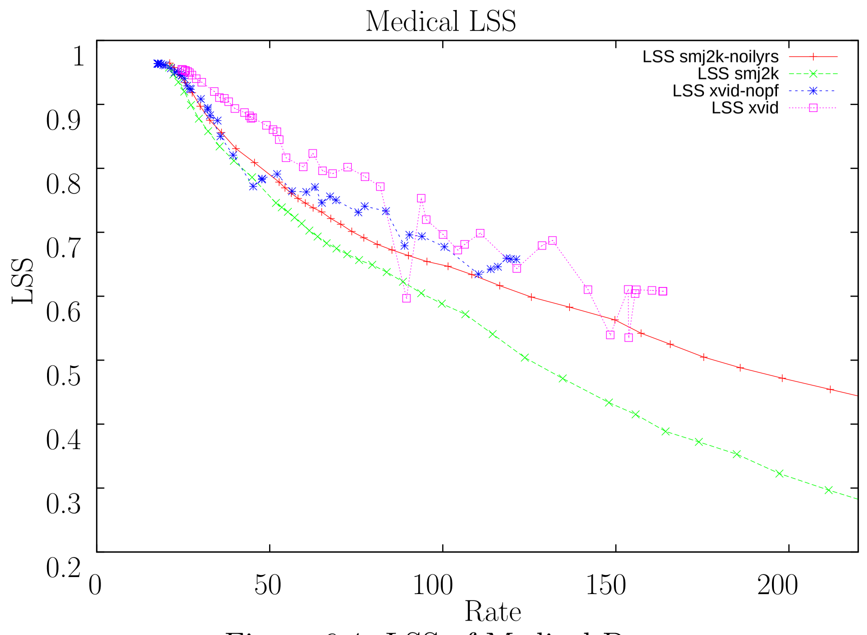


Figure 6.4: LSS of Medical Demo

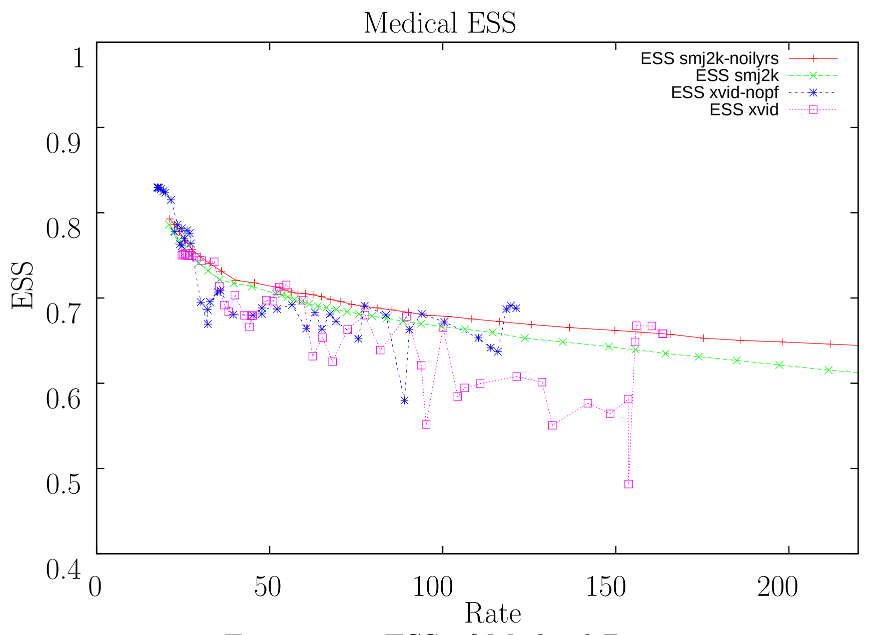


Figure 6.5: ESS of Medical Demo

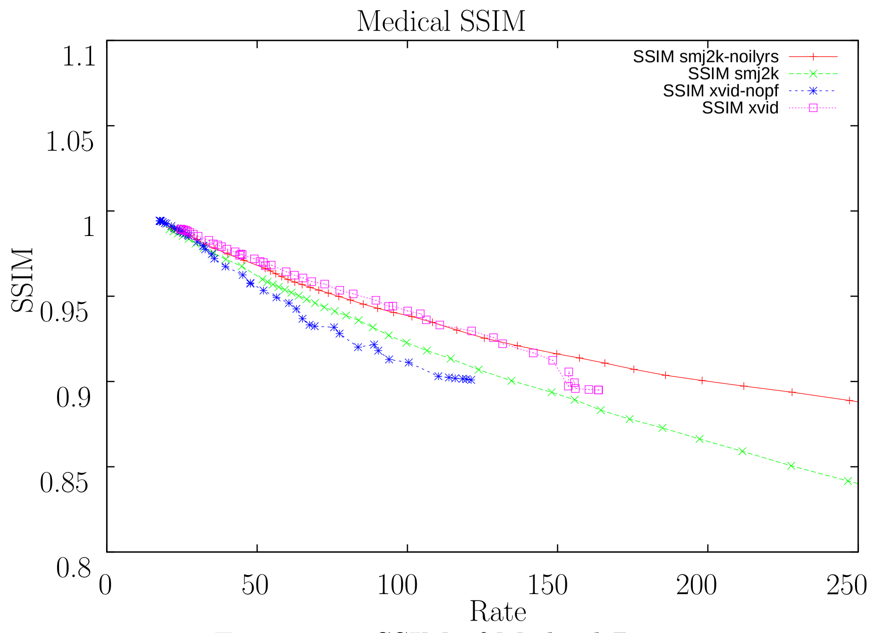


Figure 6.6: SSIM of Medical Demo

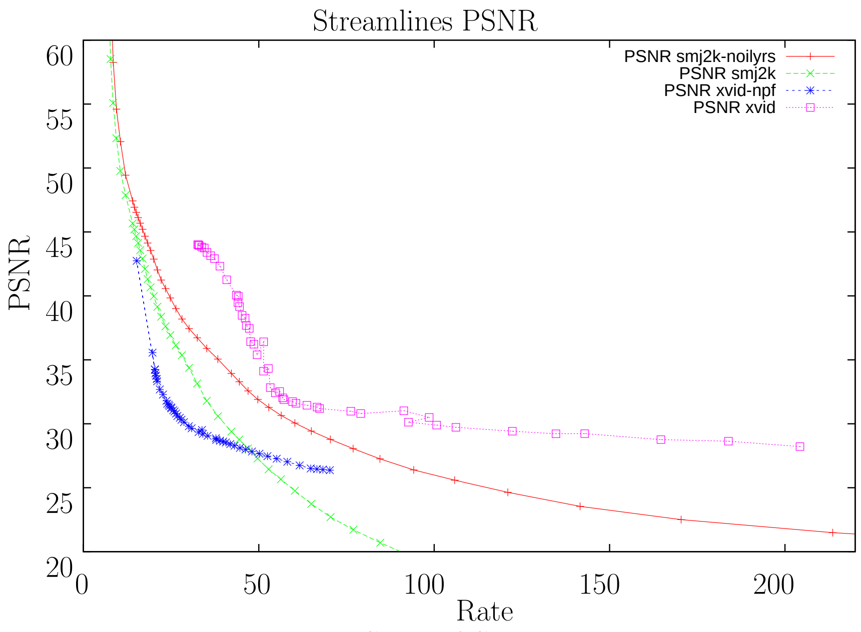


Figure 6.7: PSNR of Streamlines Demo

p-frames) up to the high compression ratios only supported by the smj2k codec (see figure 6.4). In contrast to LSS the ESS rates the smj2k codecs higher than the XviD codecs (see figure 6.5). Notable is the discontinuous behavior of the ESS especially for XviD with motion estimation. A rating quite similar to that of the PSNR is shown by the SSIM (see figure 6.6).

The Results for the Streamlines Demo

Contrary to the medical demo the streamlines demo has far less motion in it. In the first 115 frames only a small fraction of the image is changed at all. Then within 25 frames a rotation (affecting much of the image) takes place. In the next 80 frames again only a small fraction of the image is altered. Hence the PSNR plot looks rather different compared to the one for the medical demo (see figure 6.7). The XviD codec with motion compensation shows the best performance, followed by the `smj2k_noilyrs` but with a considerable gap in between. While `smj2k` is better at the lower compression rates, it is even outperformed by `xvid-npf` in some higher compression cases. While the LSS only shows advantages for smj2k codecs at very low compression rates, it assesses both very low image qualities for higher compression (see figure 6.8). The ESS basically confirms the assertion of the LSS (see figure 6.9). However, the SSIM is again quite similar to the PSNR (see figure 6.10), rating `smj2k_noilyrs` after XviD with motion compensation. Summarizing, the streamlines demo is a video source with plenty of temporal redundancy, hence XviD with motion compensation is definitely best suited for this and comparable sources.

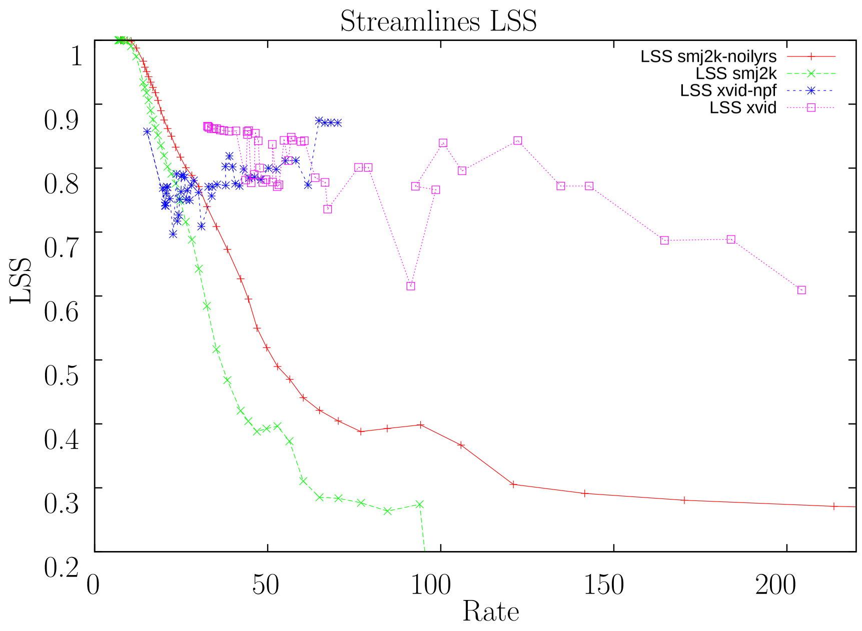


Figure 6.8: LSS of Streamlines Demo

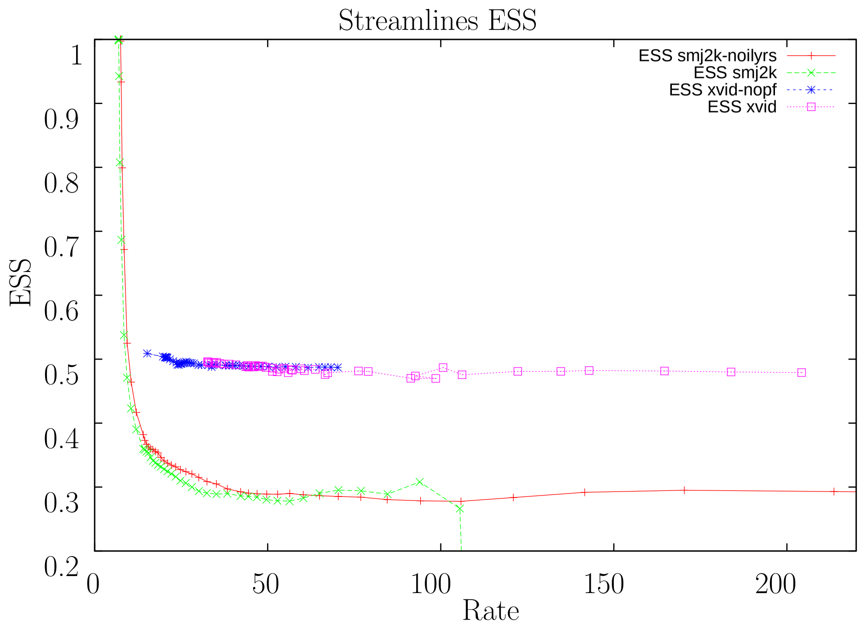


Figure 6.9: ESS of Streamlines Demo

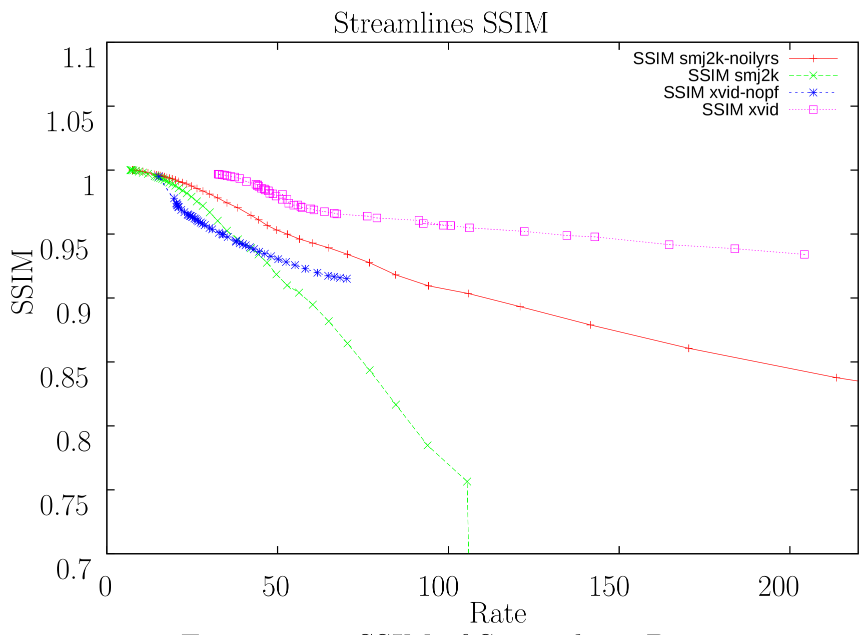


Figure 6.10: SSIM of Streamlines Demo

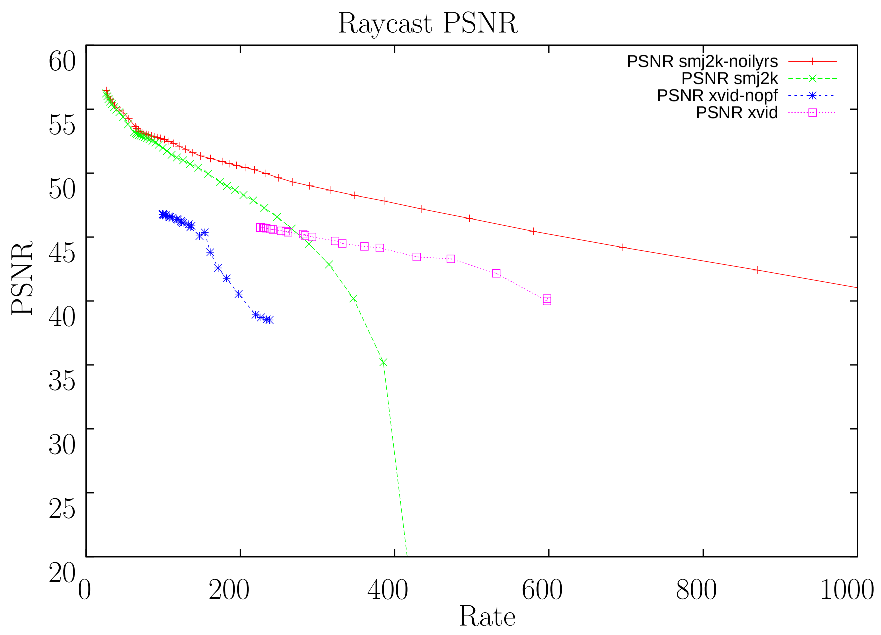


Figure 6.11: PSNR of Raycast Demo

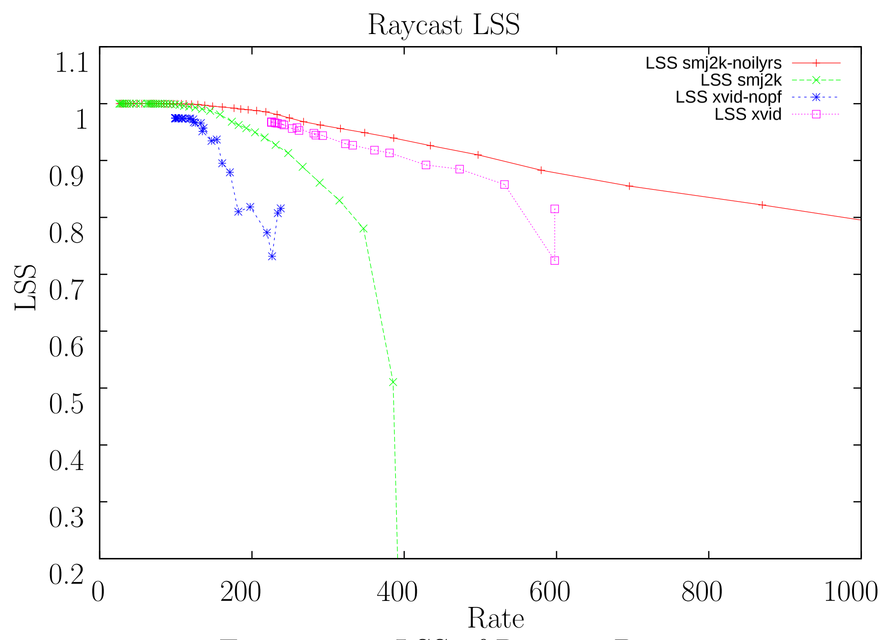


Figure 6.12: LSS of Raycast Demo

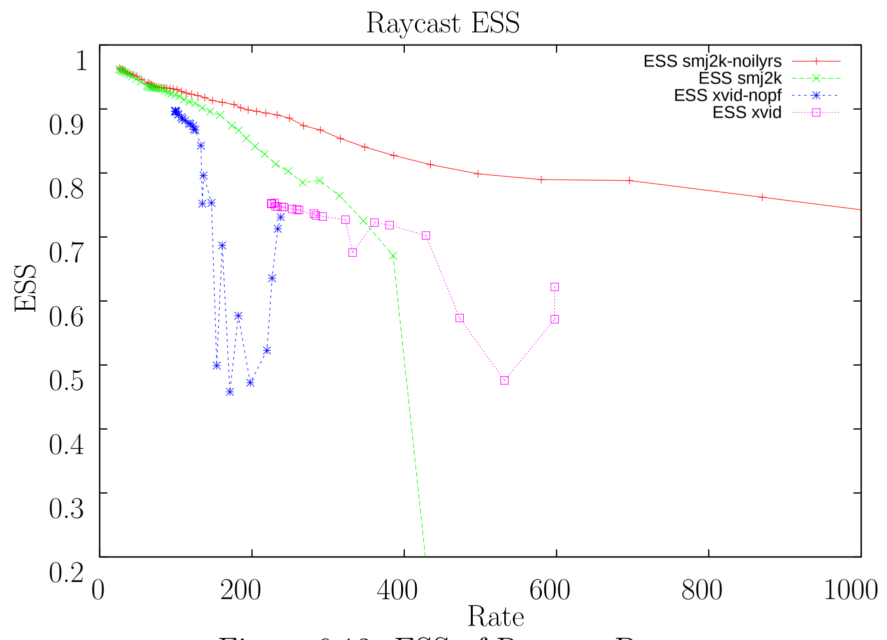


Figure 6.13: ESS of Raycast Demo

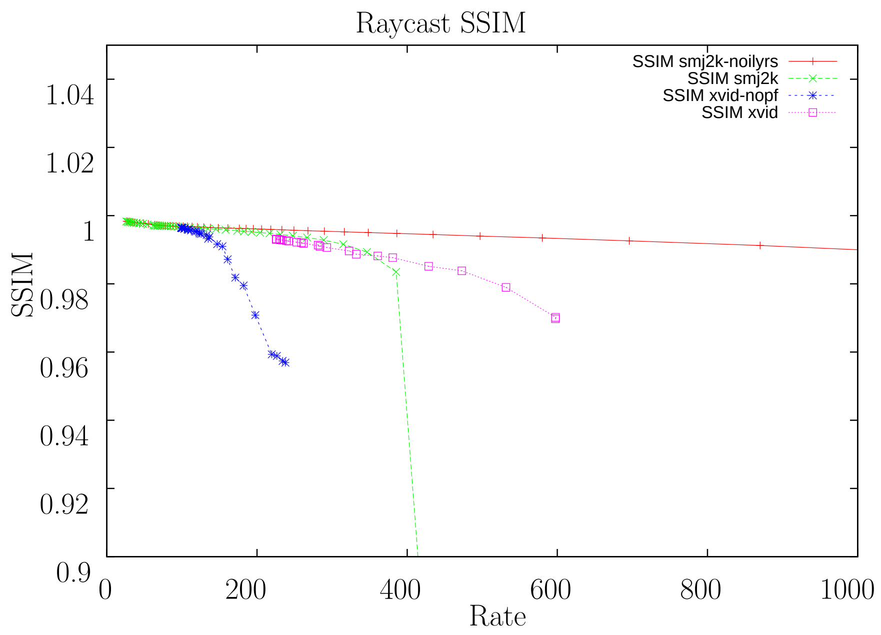


Figure 6.14: SSIM of Raycast Demo

The Results of the Raycast Demo

The raycast demo is the video sequence containing most motion. Blurry objects are rotated in the first 30 frames, then it is zoomed to these objects for 20 frames, followed by 30 frames of rotation and 30 frames of translational movements. Finally it is zoomed out for 50 frames. The PSNR shows `smj2k_noilyrs` in the lead, followed by smj2k (with quality layers) at low compression rates (see figure 6.11). For higher compression rates `xvid` performs better than `smj2k` (with quality layers). The worst performance is achieved by `xvid_npf`. These results are confirmed by the LSS and ESS plots (see figures 6.12 and 6.13). The SSIM once again is quite similar to the PSNR and also confirms its assertions (see figure 6.14).

6.1.4 Summary

The results have shown that the video source is the major factor concerning the compression efficiency of the codecs. However, the MotionJPEG2000 codec (especially with no quality layers) is surprisingly well performing considering the fact that it does not use temporal redundancy at all. It turned out that the encoding settings optimized for encryption have a negative influence on the compression performance especially for high compression. Hence there is a certain trade off between reduced encryption effort and bandwidth consumption. Under the assumption of a dedicated transmission channel only the worst case of maximum bandwidth consumption is relevant and the codecs have to be judged on the worst case scenario (high frame complexity, no temporal redundancy).

The smj2k codec is based on the open source JPEG2000 reference implementation JasPer, which is not yet thoroughly optimized, like the XviD codec, which is developed by a big community that is constantly optimizing the code. A closer look at the computational complexity of both selective encryption and compression is taken in section 6.2

6.2 Performance Evaluation

Currently three image/video compression methods (smj2k, smpjpg and XviD) are (smpjpg is still work in progress) implemented in GVid. In the following we give a short performance analysis of the three underlying video compression methods and different encryption approaches.

6.2.1 Codec Performance

The codecs are tested with 165 frames of the medical demo with two resolution (640x480 and 320x240). The medical demo was chosen because it is representative for the future usage of the GVid software. The experiments where conducted on an Acer Aspire 1620 laptop with a Pentium 4 3Ghz, a cache of 512 KB and 512 MB memory. The runtime was evaluated three times and the average **fps** (frames per second) were evaluated on the basis of these measurements. The summary of this evaluation is given in tables 6.1, 6.2, 6.3 and 6.4. The smj2k codec

JPEG2000	JPEG2000 (qlyrs)	PJPEG 85	PJPEG 20	XviD (a,s)
49.02	100.07s	20.10s	17.68s	5.58s
49.34	91.70s	15.73s	19.19s	5.45s
49.43	97.09s	13.18s	13.75s	5.60s
3.35fps	1.71fps	10.10fps	9.78fps	29.77fps

Table 6.1: Runtime Performance of Different Codecs: 640x480

XviD (a)	XviD (s)	XviD	XviD (I)	XviD (I,a)
7.37s	14.08s	15.16s	5.93s	3.29s
6.75s	12.73s	12.94s	5.58s	3.31s
7.24s	13.18s	14.97s	5.59s	3.30s
23.17fps	12.38fps	13.49fps	28.94fps	73.99fps

Table 6.2: Runtime Performance of Different Codecs: 640x480 (2)

is based on the open source implementation JasPer (Version 1.701) and tested with two encoding parameter sets. The first denoted by JPEG2000 in the tables 6.1 and 6.3 is with the following settings: no quality layers, no tiling, 64x64 code blocks, 6 level DWT, SOP and EPH marker sequences, integer mode and a rate of

JPEG2000	JPEG2000(qlyrs)	PJPEG 85	PJPEG 20	XviD (a,s)
13.37s	37.16s	3.02s	2.65s	1.59s
14.05s	34.79s	3.00s	2.57s	1.54s
13.73s	34.98s	2.96s	2.52s	1.56s
12.03fps	4.63fps	55.12fps	63.95fps	105.54fps

Table 6.3: Runtime Performance of Different Codecs: 320x240

XviD (a)	XviD (s)	XviD	XviD (I)	XviD (I, a)
1.73s	3.19s	4.31s	1.74s	0.89s
1.76s	3.17s	3.88s	1.75s	0.99s
1.78s	3.19s	5.46s	1.82s	0.99s
93.93fps	51.83fps	36.26fps	93.22fps	172.47fps

Table 6.4: Runtime Performance of Different Codecs: 320x240 (2)

0.8. The second denoted by JPEG2000 (qlyrs) uses the same settings but many quality layers (approximately 80).
The results show that the runtime performance of the JasPer implementation of JPEG2000 does depend on the number of quality layers. So many quality layers significantly reduce the runtime performance. 80 quality layers are indeed very many and for the about 20 quality layers that are necessary for selective encryption the impact on compression performance not that significant. The target bit rate has nearly no influence as long as the quality layers are not altered. However, the smaller the bit rate the smaller is the influence of the quality layers on the coding performance.
The smjpg codec is based on the libjpeg (Version 6b). In section 3.5.1 we could show that confidential encryption of progressive JPEG is only possible with successive approximation. Hence the default progression order is used, which includes successive approximation. In section 2.2.3 the default progression order for single component images is given. But in this application scenario we have to deal with color images. Libjpeg's default progression order for color images is the following:

- Initial DC scan for Y,Cb,Cr (lowest bit not sent)
- First AC scan: send first 5 Y AC coefficients, minus 2 lowest bits
- All Cr AC coefficients, minus lowest bit
  (chroma data is usually too small to be worth subdividing further)
- All Cb AC coefficients, minus lowest bit
  (chroma data is usually too small to be worth subdividing further)
- Remaining Y AC coefficients, minus 2 lowest bits
- Next-to-lowest bit of all Y AC coefficients
- Lowest bit of all DC coefficients

- Lowest bit of Cr AC coefficients
- Lowest bit of Cb AC coefficients
- Lowest Y AC bit

The progressive JPEG codec tested was evaluated for two different quality factors (80 and 20). The runtime performance of libjpeg's progressive JPEG implementation does not change dramatically for different quality factors and therefore bit rates.
The XViD codec is tested with the test program `xvid_encraw` which is part of the source release of version 1.02. Several coding settings have been evaluated. In tables 6.2 and 6.4 XviD (a,s) denotes XviD with assembler enabled and single pass mode (fastest), XviD (a) is with assembler turned on, XviD (s) is single pass mode and no assembler and XviD is without assembler and not single pass, XviD (I) encodes only I-frames and XviD (I,a) encodes only I-frames with assembler turned on. No B-frames are used at all.

Analysis

The fastest codec for all settings is XviD. However, only with assembler turned on, XviD is capable of realtime compression for resolution of 640x480. The results of the XviD codec dramatically show how much optimization is still possible, because technically there is not much difference between progressive JPEG and XviD with just I-frames. Hence the progressive JPEG codec can very likely be implemented much faster using heavy optimization and assembler as in the XviD codec. Without assembler the XviD codec is for all settings for a resolution of 640x480 below 29fps. For a resolution of 320x240 progressive JPEG becomes capable of realtime compression too.
The slowest codec is JPEG2000 with quality layers. It is at no resolution and encoding parameter capable of realtime compression at reasonable high frame rates. There is still much optimization work to do, such that JPEG2000 can be used for realtime compression at reasonably high frame rates. The difference between progressive JPEG and XviD shows how thorough optimization can improve performance drastically. Also the usage of special hardware can be considered.

6.2.2 Encryption Performance

Several selective encryption methods have been presented for progressive JPEG and JPEG2000 in the previous chapters, but the topic of actual performance benefit has not yet been thoroughly discussed. While for JPEG only one encryption method has been given (see section 3), several methods have been proposed for JPEG2000. All encryption methods will be evaluated for confidential encryption. The encryption performance depends on the amount of compressed data a codec produces. The perfect testing scenario would require that all codecs produce exactly the same image/video quality. However, this is not feasible on the one hand because we have no perfect image quality metric, on the other hand because the codecs can not be adjusted to a certain quality, which in fact would mean that one would have to extensively search for parameters which lead to the same image/video quality. Hence we tried to approximately generate equal image/video quality and we give the overall PSNR. For progressive JPEG the quality factor is set to 75 and JPEG2000 is compressed at a bit rate of 0.02 for 640x480 and at a bit rate of 0.025 for 320x240. These settings provide very high image/video quality, despite the low PSNR values.

Encryption Methods

All selective encryption methods have to be compared to traditional encryption. Therefore two test programs were implemented. For every codec these traditional methods are evaluated too. One encrypts the whole stream on a block basis in ECB mode and in the end cipher text stealing (as described in [41]) is applied. This implementation can be assumed to be the fastest possible encryption technique. The other encrypts the whole stream byte per byte in OFB mode. For every byte, the queue is encrypted. The queue consists of 16 bytes, thus for every cipher byte 16 bytes have to be AES encrypted. Therefore the complexity of the OFB mode should be approximately 16 times more than that of the ECB mode, disregarding other tasks like IO. The results show a difference of a factor 10. Both programs are command line tools which work on files, hence all file IO is measured with these methods.
The selective encryption approach for progressive JPEG is described in section 3 and in the experiments the first five scans were encrypted (all DC minus one bit, Y AC coefficients minus 2 bit and all Cr,Cb coefficients minus 1 bit). In fact since this setting encrypts even more per component than the single component case that was evaluated in section 3.5.1, it can be considered secure. A quality factor of 75 leads to a compressed stream of 4563200 byte for 640x480 and 1610696 byte for 320x240.
XviD is evaluated single pass and multi pass (the assembler does not play a role for the compression ratio). XviD with single pass leads to a compressed stream of 1013450 byte for 640x480 with an average PSNR of 38.32 db for the first component and about 35 db for the other components. The stream length for a resolution of 320x240 is 728680 byte and the PSNR is 42.15 db for the first component and about 40 db for the other components.
For 640x480 XviD with default settings produces a stream of 6505761 byte with a PSNR of 49.34 db for the first component and about 48 db for the other components. For 320x240 the default settings result in a stream of 2055554 byte with a PSNR of 47.76 db for first component and about 47 db for for the other components. XviD with only I-frames result in a stream of 1546749 byte (640x480) with PSNR 36.69 db, 32.21 db and 34.1 db and 955817 byte (320x240) with PSNR values of 39.24 db, 36.61 db and 36.65 db. JPEG2000 produces a stream of 3033321 byte (640x320) and 947465 byte (320x240). Tables 6.5 and 6.6

	640x480	320x240
XviD (s)	1013450	728680
XviD	6505761	2055554
XviD (I)	1546749	955817
smjpg 75	4563200	1610696
smj2k	3033321	947465

Table 6.5: Stream Length in Bytes for All Codecs

	640x480			320x240		
XviD (s)	38.3	35.3	34.3	42.2	40.4	39.6
XviD	49.3	48.2	47.7	47.8	47.4	46.2
XviD (I)	36.7	32.2	34.1	39.2	36.6	36.7
smjpg 75	37.6	35.4	34.5	35.4	32.34	31.2
smj2k	40.3	38.7	38.1	35.3	33.8	33.3

Table 6.6: PSNR for All Codecs

PJPEG 75			XviD (s) (38db)		XviD (49db)		XviD (I)(37db)	
smjpg	ECB	0FB	ECB	OFB	ECB	OFB	ECB	OFB
0.51s	0.256s	2.704s	0.067s	0.646s	0.347s	3.840s	0.127s	1.320s
0.31s	0.253s	2.795s	0.066s	0.603s	0.357s	3.873s	0.127s	1.214s
0.31s	0.258s	3.181s	0.070s	0.603s	0.354s	3.872s	0.126s	1.225s
438.05fps	649.60fps	57.03fps	2438.42fps	267.28fps	467.864fps	42.72fps	1302.63fps	127.94fps

Table 6.7: Selective Encryption of the Medical Demo: 640x480

PJPEG 75			XviD (s) (42db)		XviD (49db)		XviD (I)(39db)	
smjpg	ECB	0FB	ECB	OFB	ECB	OFB	ECB	OFB
0.13s	0.104s	1.219s	0.056s	0.435s	0.122s	1.342s	0.125s	1.244s
0.26s	0.098s	0.966s	0.051s	0.438s	0.120s	1.222s	0.118s	1.219s
0.22s	0.096s	0.970s	0.055s	0.513s	0.131s	1.215s	0.126s	1.240s
811.48fps	1661.07fps	156.89fps	3055.56fps	357.14fps	1327.08fps	130.99fps	1341.46fps	133.68fps

Table 6.8: Selective Encryption of the Medical Demo: 320x240

Selective Encryption of MotionJPEG2000									
ECB	0FB	mem_ofb	strm_ofb	WuMa	Kiya1	Kiya2	KiyaS	strm_zero	mem_zero
0.185s	1.792s	0.13s	0.51s	0.70s	1.31s	0.63s	1.36s	0.41s	0.04s
0.175s	1.842s	0.10s	0.61s	0.38s	1.44s	0.70s	1.19s	0.26s	0.06s
0.174s	1.831s	0.13s	0.42s	0.55s	1.15s	0.65s	1.47s	0.30s	0.07s
926.97fps	90.58fps	1375.00fps	321.43fps	303.68fps	126.92fps	250.00fps	123.13fps	510.31fps	2911.76fps

Table 6.9: Selective Encryption of the Medical Demo: 640x480 (2)

Selective Encryption of MotionJPEG2000									
ECB	0FB	mem_ofb	strm_ofb	WuMa	Kiya1	Kiya2	KiyaS	strm_zero	mem_zero
0.062s	0.563s	0.050s	0.19s	0.19s	0.40s	0.25s	0.30s	0.08s	0.02s
0.061s	0.585s	0.080s	0.20s	0.20s	0.30s	0.15s	0.34s	0.07s	0.04s
0.062s	0.570s	0.070s	0.21s	0.20s	0.30s	0.19s	0.38s	0.12s	0.03s
2675.68fps	288.13fps	2475.00fps	825.00fps	838.98fps	495.00fps	839.98fps	485.00fps	1833.33fps	5500.00fps

Table 6.10: Selective Encryption of the Medical Demo: 320x240 (2)

gives a summary of the stream length' and average PSNR. Despite the relatively low PSNR the visual quality is high. The low PSNR might be a result of the artificial source.
Several selective encryption approaches and different implementations have been evaluated. Basically two different implementations exist, one is rather generic and uses the abstraction layer for streams (jas_stream_t) as implemented in JasPer, the other simply works on a pointer to the memory where the JPEG2000 frame is temporarily stored. While the first approach is very generic and works on every file, socket, pipe, etc. the second is only applicable when the JPEG2000 frame is stored in memory. Furthermore the first approach works byte per byte even on unseekable streams. To assess the overhead introduced by processing the stream two test encryption methods have been written. Those methods do not encrypt any byte but parse for the packet bodies and write zeros in the packet bodies. Hence they give the time for the parsing and processing overhead for the selective encryption approaches. They are labeled strm_zero and mem_zero in the tables 6.9 and 6.10.
The approaches Kiya and Conan are evaluated with m set to 1 (Kiya1) and m set to 2 (Kiya2) for encryption of the half byte and with m set to 1 for shifting of the half byte (KiyaS). They are based on the stream implementation.
Furthermore the approach of Wu and Ma is evaluated, but only 30.0% for 640x480 and 31.3% for 320x240 are encrypted. The same ratio is encrypted for mem_ofb and strm_ofb. Both encryption methods securely encrypt the JPEG2000 stream in a modified OFB fashion. The bias is omitted by simply discarding bytes in the queue that are either `0xff` or in excess of `0x8f` after a `0xff`. Afterward addition modulo `0xff` or `0x8f`+1 (after a `0xff`) is applied.

Analysis

The results show that selective encryption can be realized very efficiently, but that care has to be taken with the implementation. It turns out that the abstraction layer with JasPer streams is rather costly and encryption can be far more efficiently implemented directly on the memory pointer. The comparisons of the mem_zero and strm_zero encryption codecs in tables 6.9 and 6.10 shows the difference between the two implementations. However, the more general approach is the usage of streams. The selective encryption approaches with the memory pointer implementation even outperform the ECB encryption. But the ECB encryption also contains all the file IO, which is not necessary for the memory pointer implementation. The OFB JPEG2000 encryption approaches could also be a bit more optimized, such that they get closer to ECB performance.
The approaches of Kiya and Conan are less efficient because more data has to be encrypted to guarantee confidential encryption. The approach of Wu and Ma shows the same performance than the strm_ofb, which is not surprising because it is implemented rather similarly.
Summarizing the results show that selective encryption can be implemented

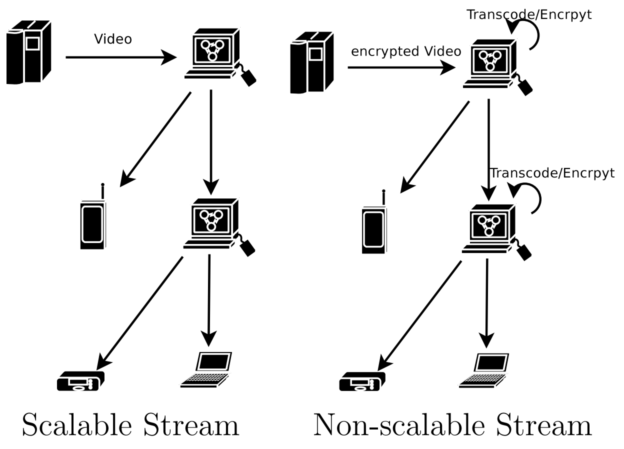


Figure 6.15: Application Scenario

rather efficiently, but a careful optimized implementation is necessary. Abstraction is possibly to expensive for the runtime performance.
Comparing these results to the compression performance results, the encryption is only a very small part of the secure visual pipeline's runtime. Hence, the actual benefit from optimizing the encryption process is very limited. Actually this is the reason why most of the codecs are implemented with the JasPer streams. The benefit of applying the encryption codecs to all sorts of data types (files, sockets, pipes, etc.) is greater than the actual benefit from the improved runtime performance. However, with thorough optimization of the encryption methods one could possibly even outperform conventional ECB encryption.
The great advantage of selective encryption is that rate adaption can be conducted in the encrypted domain. Such scenarios are analyzed in the next section.

6.3 Complex Distribution Scenarios

Several applications like TV broadcasting, collaborative applications where several users work together, passive viewers of an application (control/learning/etc.) exist where the visual data is distributed between more than one client. A complex distribution scenario is illustrated in figure 6.15. With the selectivly encrypted scalable stream rate adaption is possible in the encrypted domain, while the non-scalable stream has to be decrypted and transcoded. Currently no multi client capability is implemented within the GVid software, but the implementation effort is not tremendously great to add this functionality. Figure 6.16 shows a simple multi client extension where multiple connections to the clients are created. Another possibility is distribution and rate adaption with an external application, denoted as distributor in figure 6.17. This distributor may be a simple network device with very low computational power, e.g., a wireless sender that is capable to adapt the rate according to the current bandwidth. In the case of selective encryption of scalable media this simple device would be capable of

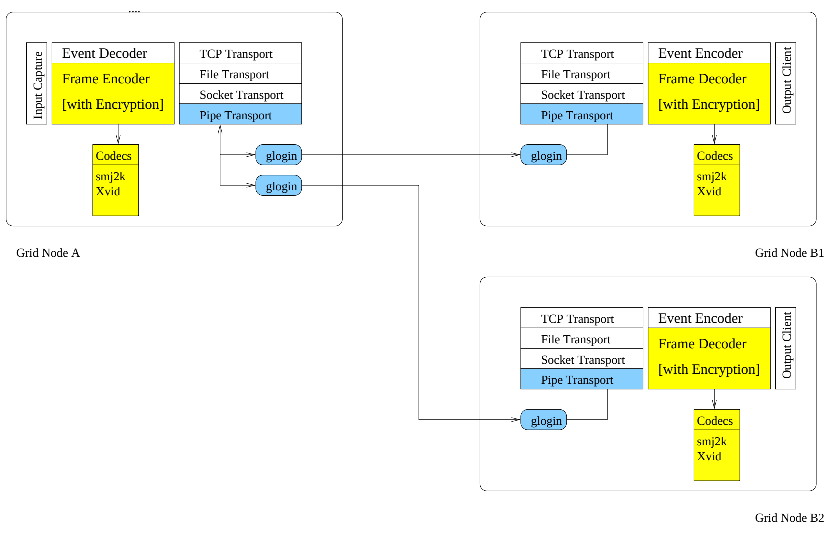


Figure 6.16: GVid Planned Component Overview

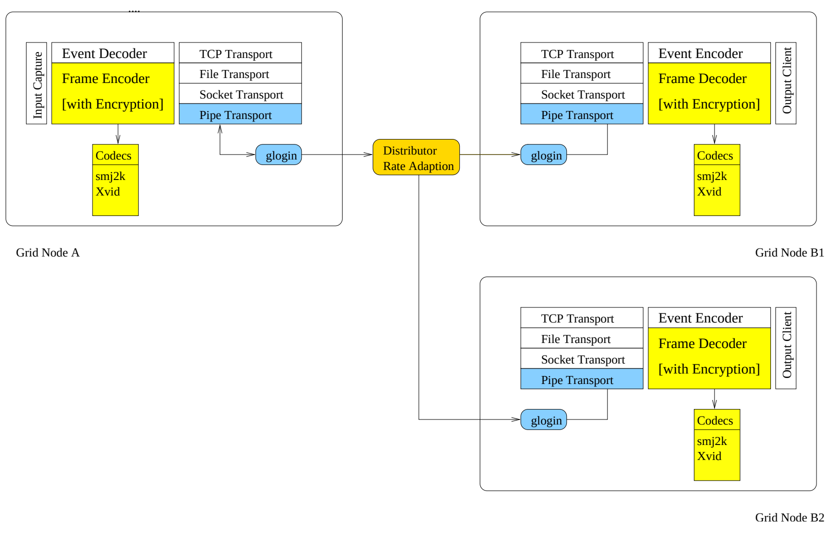


Figure 6.17: GVid Planned Component Overview (2)

rate adaption. Since rate adaption of selectively encrypted scalable media needs no key, no decryption and no transcoding. One the one hand we can compare different compression codecs, either scalable or non scalable with selective encryption or conventional encryption, on the other hand we can compare selective encryption to conventional encryption for a certain codec. The computationally complex MotionJPEG2000 codec for example is the standard for Digital Cinema. Hence for Digital Cinema the high computational effort for MotionJPEG2000 is no longer a topic because it is requested by the standard. Therefore even if the codec is not competitive to XviD, the question if selective encryption brings benefits for this codec is of interest. Furthermore some scenarios require lossless compression that only JPEG2000 is capable of.
Different clients require streams at different rates, either because of their capabilities or their connection bandwidth. Figure 6.15 illustrates complex distribution scenarios.
In the following we analyze the following scenarios:

Scenario 1 For scalable streams with selective encryption rate adaption is done at a so called distributor which only needs a negligibly small effort for this task.

Scenario 2 For scalable streams with conventional encryption the distributor has to decrypt the stream (the decryption key is required), adapt the rate which again needs only a negligible effort for transcoding. Then the different streams have to be encrypted again.
If no distributor is used the decryption can be left out.

Scenario 3 For non-scalable streams rate adaption at the distributor has to decrypt and transcode the stream and reencrypt the resulting streams.
This scenario is also applicable for scalable streams, if the distributor does no intelligent transcoding (e.g, through packet dropping).

If no distributor is used the initial compression and encryption and the decryption at the distributor can be left out.

Hence for n clients with different rates we have the following time consumption at the server and distributor:

Scenario 1 $t^{(c,s)}_{server} = t^{(c)}_{0,comp} + t^{(s)}_{enc}$

Scenario 2 $t^{(c,s)}_{server} = t^{(c)}_{0,comp} + t^{(s)}_{0,enc} + t^{(s)}_{0,dec} + \sum_{i=1}^{n} t^{(s)}_{i,enc}$

Scenario 3 $t^{(c,s)}_{server} = t^{(c)}_{0,comp} + t^{(s)}_{0,enc} + t^{(s)}_{0,dec} + t^{(c)}_{0,decomp} + \sum_{i=1}^{n} t^{(c)}_{i,comp} + \sum_{i=1}^{n} t^{(s)}_{i,enc}$

Where $t^{(c,s)}_{server}$ is the time taken at the grid node producing the stream and the distributor, c denotes the codec used and s the encryption method, $t^{(c)}_{i,comp}$ the time for compression , $t^{(c)}_{i,decomp}$ the time for decompression, $t^{(s)}_{i,enc}$ the time for encryption and $t^{(s)}_{i,dec}$ the time for decryption of the i-th stream (with a certain bit rate). The clients need the following time for all scenarios:

$$t^{(c,s)}_{clients} = \sum_{i=1}^{n} t^{(s)}_{i,dec} + \sum_{i=1}^{n} t^{(c)}_{i,decomp}.$$

6.3.1 Performance Benefits of Selective Encryption for Scalable Media

Hence the difference of selective versus conventional encryption for a certain scalable codec is in the reduced encryption amount. In the GVid, where compression currently takes place, quite a lot of clients have to be present that selective encryption is a significant benefit. But if we think of cases like video streaming where the media is already compressed the complexity reduction is enormous. The reduced complexity of the key management has to be taken into account for those selective encryption of scalable media.
At the server we have the difference between scenario 1 and scenario 2, which is

$$\Delta t = t^{(s2)}_{enc} - t^{(s1)}_{enc} + t^{(s2)}_{0,dec} + \sum_{i=1}^{n} t^{(s2)}_{i,enc}.$$

The difference between selective encryption and conventional encryption of a single frame is rather small (confirmed by the results in section 6.2.2). The client side is basically the same under the assumption that selective decryption and conventional decryption do not differ enormously and decompression is the far more complex and time consuming task.

JPEG2000

The fastest encryption methods process 927fps (1.08ms per frame for ECB) and 1375fps (0.72ms per frame for mem_ofb) for 640x480 and 2676fps (0.37ms for ECB) and 2475fps (0.40ms per frame for mem_ofb) 320x240. Hence approximately about 1ms per frame is the encryption time for a 640x480 frame and less than 0.5ms for a 320x240 frame. Since encryption and decryption have the same complexity the computational benefit of is below $(n+1)$ms for n clients and source of 640x480 compressed with a rate below 0.02. If for low latency or other reasons OFB mode has to be applied, the encryption gets more complex and only 90.6fps (11.04ms per frame for 640x480) and 288.1fps (3.47 ms per frame for 320x240) can be processed. In this case the computational benefit is below $(n + 2)$ 11ms.

Progressive JPEG

Selective encryption of progressive JPEG is capable of processing 438.1fps (2.28ms per frame for 640x480) and 811.5fps (1.23ms per frame for 320x240), while ECB implementation can encrypt 650.0fps (1.54ms per frame for 640x480) and 1661.1fps (0.6ms per frame for 320x240). The OFB implementation encrypts

57.0fps (17.54ms per frame for 640x480) and 156.9fps (6.37ms pre frame 320x240). Hence less than $1.54 - 2.28 + (n + 1)1.54$ms can be saved per frame when selective encryption is applied instead of the ECB implementation. If we compare the selective encryption approach to the OFB implementation the reduction is below $17.54 - 2.28 + (n + 1)17.54$ms.
If the resolution and the bit rates increase the advantages of selective encryption increase.

6.3.2 Comparison of All Codecs

If we want to compare different codecs with or without selective encryption, we had to evaluate the three scenarios for all codecs with all encryption methods and various numbers of clients. However, if we compare different compression methods the compression is the main performance factor. Non-scalable compression implementations like XviD have to compress a single stream for every bit rate with nearly the same complexity, while scalable compression methods like progressive JPEG and JPEG2000 can simply drop parts of the file stream (scans in JPEG and packets in JPEG2000).
The JPEG2000 compression takes 10 times longer than the XviD (a,s) compression, and gets even more costly for higher resolutions. Furthermore the decompression of JPEG2000 has nearly the same complexity as the compression, while the XviD decompression is even less complex than its compression. Hence if compression is included in the comparison, selective encryption of JPEG2000 can not lead to performance benefits because the compression is still to costly. However, for scenarios where the compression does not play a role selective encryption of JPEG2000 may lead to runtime advantages.
Comparing progressive JPEG and XviD is more interesting because XviD (a,s) is only 3 times faster. Hence at about 3 clients progressive JPEG brings a benefit at the server side. The benefits from selective encryption are rather reduced, because the XviD codec produces less data (see table 6.5) and therefore the encryption time of progressive JPEG is 10 times more than that of XviD.
Comparing JPEG2000 and progressive JPEG it is notable that the progressive JPEG compression is about 3 times faster. The encryption effort of the different selective encryption approaches does not differ much. Hence the performance of progressive JPEG with selective encryption is superior. But JPEG2000 offers much more flexibility and functionality, the most important one is the bit rate control.

6.3.3 Conclusion

Selective encryption reduces the computational effort in multi client scenarios. To be competitive the runtime performance of the scalable compression methods has to be improved.

Nevertheless the performance analysis leaves out many benefits that the usage of both scalable media and selective encryption have to offer. This includes reduced complexity for key management, reduced bandwidth and storage consumption for different bit rate versions [50]. Also the possibility of rate adaption is a big advantage for streaming to e.g. mobile devices where the rate adaption has to take place in realtime and probably by simple wireless sender device.
Considering these advantages even the rather equal performance of the selective encryption methods may be enough to satisfy its usage because of the enhanced functionality.
Furthermore advanced application scenarios like transparent encryption can be realized.

List of Figures

List of Tables

Bibliography

[1] FreeGLUT - The Free OpenGL Toolkit. online presentation, December 2005.

[2] The Globus Toolkit. online presentation, December 2005.

[3] VTK - The Visualization Toolkit. online presentation, January 2006.

[4] The XviD project homepage. online presentation, January 2006.

[5] Digital Cinema Initiatives, LLC (DCI). Digital cinema system specification v1.0. online presentation, July 2005.

[6] ISO/IEC JTC1 SC29 Working Group 1. Digital compression and coding of continuous-tone still images — requirements and guidelines, September 1992. IS 10918-1, ITU-T T.81.

[7] M.D. Adams. The JPEG-2000 still image compression standard. ISO/IEC JTC 1/SC 29/WG 1 N 2412, September 2001.

[8] H. Cheng and X. Li. On the application of image decomposition to image compression and encryption. In P. Horster, editor, *Communications and Multimedia Security II, IFIP TC6/TC11 Second Joint Working Conference on Communications and Multimedia Security, CMS '96*, pages 116–127, Essen, Germany, September 1996. Chapman & Hall.

[9] C.K. Chui, L. Montefusco, and L. Puccio. *Wavelets: Theory, Algorithms and Applications.* Academic Press, San Diego, 1994.

[10] Y. Wu D. Ma and R. Deng. Communications in JPEG2000 security group (JPSEC). In *Anaylsis of Canon Encryption Scheme*, November 2003.

[11] J. Daemen and V. Rijmen. The block cipher rijndael. In J.-J. Quisquater and B. Schneier, editors, *Smart Card Research and Applications*, volume 1820 of *Lecture Notes in Computer Science*, pages 288–296. Springer Verlag, 2000.

[12] J. Daemen and V. Rijmen. Rijndael, the advanced encryption standard. *Dr. Dobb's Journal*, 26(3):137–139, March 2001.

[13] Jana Dittmann and Ralf Steinmetz. Enabling technology for the trading of MPEG-encoded video. In *Information Security and Privacy: Second Australasian Conference, ACISP '97*, volume 1270, pages 314–324, July 1997.

[14] Ahmet Eskicioglu and Edward J. Delp. An integrated approach to encrypting scalable video. In *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME '02*, Laussanne, Switzerland, August 2002.

[15] J. Apostolopoulos F. Dufaux, S. Wee and T. Ebrahimi. JPSEC for Secure Imaging in JPEG 2000. In *SPIE Proc. Applications of Digital Image Processing XXVII.* SPIE, August 2004.

[16] Mark M. Fisch, Herbert Stögner, and Andreas Uhl. Layered encryption techniques for DCT-coded visual data. In *Proceedings (CD-ROM) of the European Signal Processing Conference, EUSIPCO '04*, Vienna, Austria, September 2004. paper cr1361.

[17] I. Forster. What is the GRID? a three point checklist. *GRIDToday*, July 2002.

[18] J.W. Goethe. *Faust - Der Tragödie erster Teil.* Reclam, 1971.

[19] Raphaël Grosbois, Pierre Gerbelot, and Touradj Ebrahimi. Authentication and access control in the JPEG 2000 compressed domain. In A.G. Tescher, editor, *Applications of Digital Image Processing XXIV*, volume 4472 of *Proceedings of SPIE*, pages 95–104, San Diego, CA, USA, July 2001.

[20] Canon inc. Encryption tool for JPEG2000 access control. ISO/IEC JTC 1/SC 29/WG 1 N 2983, 2003.

[21] B. Jawerth and W. Sweldens. An overview of wavelet based multiresolution analyses. *SIAM Review*, 36(3):377–412, 1994.

[22] B. Jawerth and W. Sweldens. Biorthogonal smooth local trigonometric bases. *J. Fourier Anal. Appl.*, 2(2):109–133, 1995.

[23] H. Kiya, D. Imaizumi, and O. Watanabe. Partial-scrambling of image encoded using JPEG2000 without generating marker codes. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'03)*, volume III, pages 205–208, Barcelona, Spain, September 2003.

[24] T. Köckerbauer, M. Kumar, and A. Uhl. Lightweight JPEG 2000 confidentiality for mobile environments. In *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME '04*, Taipei, Taiwan, June 2004.

[25] Thomas Kunkelmann. Applying encryption to video communication. In *Proceedings of the Multimedia and Security Workshop at ACM Multimedia '98*, pages 41–47, Bristol, England, September 1998.

[26] S. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Trans. on Acoust. Signal Speech Process.*, 37(12):2091–2110, 1989.

[27] S. Mallat. Multiresolution approximations and wavelet orthonormal bases of L^2. *Trans. Amer. Math. Soc.*, 315(1):69–87, 1989.

[28] S. Mallat. Zero crossings of a wavelet transform. *IEEE Trans. on Inf. Theory*, 37(4):1019–1033, 1991.

[29] Y. Mao and M. Wu. Security evaluation for communication-friendly encryption of multimedia. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'04)*, Singapore, October 2004. IEEE Signal Processing Society.

[30] National Institute of Standards and Technology. FIPS-197 - advanced encryption standard (AES), November 2001.

[31] R. Norcen, M. Podesser, A. Pommer, H.-P. Schmidt, and A. Uhl. Confidential storage and transmission of medical image data. *Computers in Biology and Medicine*, 33(3):277 – 292, 2003.

[32] Roland Norcen and Andreas Uhl. Selective encryption of the JPEG2000 bitstream. In A. Lioy and D. Mazzocchi, editors, *Communications and Multimedia Security. Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, CMS '03*, volume 2828 of *Lecture Notes on Computer Science*, pages 194 – 204, Turin, Italy, October 2003. Springer-Verlag.

[33] A. Pommer and A. Uhl. Wavelet packet methods for multimedia compression and encryption. In *Proceedings of the 2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 1–4, Victoria, Canada, August 2001. IEEE Signal Processing Society.

[34] A. Pommer and A. Uhl. Application scenarios for selective encryption of visual data. In J. Dittmann, J. Fridrich, and P. Wohlmacher, editors, *Multimedia and Security Workshop, ACM Multimedia*, pages 71–74, Juan-les-Pins, France, December 2002.

[35] A. Pommer and A. Uhl. Selective encryption of wavelet packet subband structures for obscured transmission of visual data. In *Proceedings of the 3rd IEEE Benelux Signal Processing Symposium (SPS 2002)*, pages 25–28, Leuven, Belgium, March 2002. IEEE Benelux Signal Processing Chapter.

[36] A. Pommer and A. Uhl. Selective encryption of wavelet packet subband structures for secure transmission of visual data. In J. Dittmann, J. Fridrich, and P. Wohlmacher, editors, *Multimedia and Security Workshop, ACM Multimedia*, pages 67–70, Juan-les-Pins, France, December 2002.

[37] A. Pommer and A. Uhl. Selective encryption of wavelet-packet encoded image data — efficiency and security. *ACM Multimedia Systems (Special issue on Multimedia Security)*, 9(3):279–287, 2003.

[38] W. Ford R. Housley, W. Polk and D. Solo. Rfc 3280: Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. online presentation, 2002.

[39] Herbert Rosmanith and Dieter Kranzlmüller. glogin - a multifunctional, interactive tunnel into the grid. In *5th IEEE/ACM International Workshop on Grid Computing*, Pittsburg, USA, November 2004.

[40] M.B. Ruskai, G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer, and L. Raphael, editors. *Wavelets and their Applications.* Jones and Bartlett, 1992.

[41] B. Schneier. *Applied cryptography (2nd edition): protocols, algorithms and source code in C.* Wiley Publishers, 1996.

[42] C. Shi and B. Bhargava. A fast MPEG video encryption algorithm. In *Proceedings of the Sixth ACM International Multimedia Conference*, pages 81–88, Bristol, UK, September 1998.

[43] Changgui Shi, Sheng-Yih Wang, and Bharat Bhargava. MPEG video encryption in real-time using secret key cryptography. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, pages 2822–2829, Las Vegas, Nevada, 1999.

[44] Dengfeng Zhang Shiguo Lian, Jinsheng Sun and Zhiquan Wang. A selective image encryption scheme based on JPEG2000 codec. In Y. Nakamura K. Aizawa and S. Satoh, editors, *aSmart Card Research and Applications*, volume 3332 of *Lecture Notes in Computer Science*, pages 65–72. Springer Verlag, 2004.

[45] Zhiquan Wang Shiguo Lian, Jinsheng Sun. Perceptual cryptography on JPEG2000 compressed images or videos. In *4th International Conference on Computer and Information Technology*, Wuhan, China, September 2004.

[46] L. Tang. Methods for encrypting and decrypting MPEG video data efficiently. In *Proceedings of the ACM Multimedia 1996*, pages 219–229, Boston, USA, November 1996.

[47] D. Taubman and M.W. Marcellin. *JPEG2000 — Image Compression Fundamentals, Standards and Practice.* Kluwer Academic Publishers, 2002.

[48] A. Uhl and A. Pommer. *Image and Video Encryption. From Digital Rights Management to Secured Personal Communication*, volume 15 of *Advances in Information Security.* Springer-Verlag, 2005.

[49] Y. Sadourny V. Conan and S. Thomann. Symmetric block cipher based protection: Contribution to JPSEC. ISO/IEC JTC 1/SC 29/WG 1 N 2771, October 2003.

[50] S. Wee and J. Apostolopoulos. Secure scalable video streaming for wireless networks. In *Proceedings of the 2001 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001)*, May 2001.

[51] S.J. Wee and J.G. Apostolopoulos. Secure transcoding with JPSEC confidentiality and authentication. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'04)*, Singapore, Singapore, October 2004.

[52] Jiangtao Wen, Mike Severa, Wenjun Zeng, Max Luttrell, and Weiyin Jin. A format-compliant configurable encryption framework for access control of multimedia. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing, MMSP '01*, pages 435–440, Cannes, France, October 2001.

[53] Jiangtao Wen, Mike Severa, Wenjun Zeng, Max Luttrell, and Weiyin Jin. A format-compliant configurable encryption framework for access control of video. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6):545–557, June 2002.

[54] H. Wu and D. Ma. Effiecient and secure encryption schemes for JPEG2000. In *Proceedings of the 2004 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2004)*, pages 869–872, May 2004.

[55] M. Wu and Y. Mao. Communication-friendly encryyption of multimedia. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, October 2002.

[56] Yongdong Wu and Robert H. Deng. Compliant encryption of JPEG2000 codestreams. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'04)*, Singapure, October 2004. IEEE Signal Processing Society.

[57] R. Deng Y. Wu and D. Ma. ImAccess: A method for JPEG2000 access control. ISO/IEC JTC 1/SC 29/WG 1 meeting, Seoul, March 2003.

[58] H.R. Sheikh Z. Wang, A.C. Bovik and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.

[59] Wenjun Zeng and Shawmin Lei. Efficient frequency domain selective scrambling of digital video. *IEEE Transactions on Multimedia*, 5(1):118–129, March 2003.