Chapter 5

IMAGE AND VIDEO ENCRYPTION

Several review papers have been published on image and video encryption providing a more or less complete overview of the techniques proposed so far. Kunkelmann [76] and Qiao and Nahrstedt [120] provide overviews, comparisons, and assessments of classical encryption schemes for visual data with emphasis on MPEG proposed up to 1998. Bhargava et al. [14] review four MPEG encryption algorithms published by the authors themselves in the period 1997 – 1999. More recent surveys are provided by Liu and Eskicioglu [87] (with focus on shortcomings of current schemes and future issues), But [18] (where the suitability of available MPEG-1 ciphers for streaming video is assessed), and Lookabaugh et al. [85] (who focus on a cryptanalysis of MPEG-2 ciphers; in [84] the authors discuss MPEG-2 encryption as an example for selective encryption in consumer applications, the paper having broader scope though). The only monograph existing so far in this area is the PhD-thesis of Kunkelmann [74]. This book covers in-depth discussions and comparisons of MPEG encryption techniques published up to 1998, unfortunately it is written in German.

Image and video encryption are of course closely related by the fact that raw video data consists of a sequence of still images. However, compressed video like the MPEG format is composed of different types of data which can be treated in specific ways by special encryption schemes. As a consequence, we first discuss (still) image encryption techniques in each section which can be applied to still images or single frames (e.g. I-frames in MPEG) in a video. Subsequently, we describe procedures that exploit video specific properties. Additionally, due to reasons explained in the previous section, we distinguish between techniques applied during the compression stage (compression oriented schemes) and techniques applied to an already given bitstream (bitstream oriented schemes).

After describing and discussing the algorithms suggested so far in literature, we provide a final assessment for each technique. For this assessment, we evaluate the following properties:

- **Time demand:** The additional time demand required for the encryption consists of two parts – the time required to perform the actual encryption (denoted as "time(E)") and the time required to identify the parts of the data which are subject to encryption (which may include bitstream parsing or any other preprocessing technique and is denoted as "time(P)"). The time demand is rated as 0, low, medium, high, and may have the additional property of being scalable if depending on the amount of data encrypted.

- **Security:** The security of an entire image and video encryption approach has two aspects. First, the security of the cipher in use itself, and second, the importance and suitability of the data subject to encryption. The security is rated as low, medium, high, and may have the additional property of being scalable if depending on the amount of data encrypted. In accordance to the two aspects of security, two entirely different types of attacks against image and video encryption systems are possible. On the one hand, the cipher in use may be the target of an attack. In this case, common cryptanalytic results about the security of specific ciphers in general apply. On the other hand, in the case of partial or selective encryption, it is possible to reconstruct the visual content without taking care of the encrypted parts. Depending on the importance of the encrypted data for visual perception, the result may range from entirely incomprehensible to just poor or reduced quality. In any case, when conducting such a "direct reconstruction", the high frequency noise originating from the encrypted portions of the data is propagated into the reconstructed frame. In order to avoid this phenomenon, "error-concealment attacks" [174], "perceptual attacks" [85], or "replacement attacks" [104, 106] have been suggested. These types of attacks either try to conceal the quality reduction caused by encryption by treating unbreakable data as lost and then trying to minimise the impact on quality as a result of loss (error-concealment attacks) or simply replace the encrypted parts of the data by either artificial data mimicking simple non-structured content (replacement attacks) or data minimising the perceptual impact of the incorrect data (perceptual attacks).

- **Bitstream compliance:** An image or video encryption scheme is said to be bitstream compliant, if the resulting bitstream is compliant to the bitstream definition of the compression system in use. Bitstream compliance is inherent to any compression oriented encryption scheme if it is based on somehow manipulating coefficient data. On the other hand, bitstream oriented schemes often do not take care about this property at all. Bitstream compliance is rated yes or no.

- **Compressed domain processing:** An important property of image and video encryption schemes is whether they may be applied directly to a given bitstream or the bitstream needs to be decoded before being able to apply encryption. This property is denoted "Bitstream processing" and rated yes or no.

- **Compression performance affected:** As we shall see, quite a lot of image and video encryption schemes increase the file size as compared to applying compression without encryption. This property is denoted as "affecting R/D" (rate-distortion) and is rated yes, moderately, and no.

1. DCT-based Techniques

1.1 Image Encryption

1.1.1 Compression Oriented Schemes

Zig-zag Permutation Algorithm. The historically first MPEG encryption proposal is due to Tang [149] and is called "zig-zag permutation algorithm". The idea is to substitute the fixed zig-zag quantised DCT coefficient scan pattern by a random permutation list. Consequently, in the terminology introduced in the previous section this is a soft encryption approach. Additional security measures are proposed as follows:

- The 8 bit DC coefficient is to be split into two 4 bit halves, out of which the MSB part replaces the original DC coefficient and the LSB part replaces the smallest AC coefficient. The reason is that the DC coefficients could be identified immediately by their size thus revealing a low-pass approximation of the image.

- Before DC-splitting, the DC coefficients of eight 8×8-pixels blocks are concatenated, DES encrypted and written back byte-oriented.

- Instead of using one permutation list a cryptographically strong random bit generator selects one out of two permutation list for each 8×8-pixels block.

Shin et al. [140] propose a very similar system except that instead of splitting the DC coefficient the sign bits of the DC coefficients are DES encrypted and the DC coefficients are not subject to permutation.

There have been several shortcomings of the zig-zag permutation algorithm identified in literature:

- **Security**

  – Known plaintext and chosen ciphertext attacks: Permutations are known to be vulnerable against known plaintext attacks. Assuming a certain frame of the video is known (the plaintext) – by comparing original and permuted coefficients the permutation list can be retrieved. Even in case two lists have been used the correct one can be found on a per block basis by applying both and using the block with most non-zero coefficients in the upper left corner as decrypted block as shown by Qiao and Nahrstedt [100, 120]. Assuming a decoder is available to an attacker, Uehara and Safavi-Naini [158] also show the analogous weakness against a chosen ciphertext attack.

  - Ciphertext only attack: The ciphertext only attack is the weakest attack available to an adversary. In the context of the zig-zag permutation algorithm it is based on the fact that non-zero AC coefficients tend to gather in the upper left corner of the considered image block. Once the non-zero coefficients have been identified in the block, they are shifted to the upper left corner of the block and only a relatively small number of combinations among the non-zero coefficients is required to be tested. Based on some statistical analysis which involves the frequency of non-zero occurrence, Qiao and Nahrstedt give some impressive examples how well images can be approximated using these techniques [120].

- **Decrease of compression performance:** The zig-zag scan as included in the JPEG and MPEG standards orders the coefficients with respect to increasing frequency and decreasing magnitude. As a consequence, long runs of zeros occur in the high frequency areas of a block. The JPEG and MPEG Huffman tables are optimised with respect to those properties – therefore, by changing the zig-zag pattern, some compression performance is expected to be lost. In fact, Qiao and Nahrstedt [120] show a compression performance decrease of up to 45% for MPEG, Zeng and Lei report a bit overhead of 108% for a single I-frame and of 55% for an entire video for H.263 [186], whereas Kailasanathan et al. report decrease up to 20% for JPEG [66].

Shi and Bhargava [138] propose a similar though not equivalent approach. The Huffman codeword list as defined in the MPEG standard is permuted using a secret key, and subsequently used in the compression and decompression process. In order to prevent the algorithm from affecting compression performance only permutations are admissible which maintain the length of the codewords. This limits the number of possible permutations significantly (as already mentioned by the authors) and therefore reduces the available keyspace (and with it the security of the system). Note that contrasting to this approach zig-zag permutation potentially also changes the number of required Huffman codewords (due to different amount and length of zero-coefficient runs) in addition to the resulting permutation of the Huffman table.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	low	yes	no	yes

Table 5.1. Overall assessment of the Zig-zag Permutation Algorithm

Frequency-band Coefficient Shuffling. In order to limit the drop in compression efficiency as seen with zig-zag permutation, Zeng and Lei [185, 186] propose not to permute the coefficients within a single 8 × 8 pixels block but to group the coefficients from an entire set of such blocks together and perform permutation to DCT coefficients within a frequency band (i.e. with similar frequency location). This strategy reduces the bit overhead significantly, but still a file size increase of 10 - 20% can be observed [186]. Whereas the security problems as induced by the use of permutations remain valid in principle, the situation is improved as compared to the pure *zig-zag* permutation approach since additional key material may be employed to define which blocks are used to select coefficients from and the described ciphertext only attack is much more difficult since more blocks are involved.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	low	low	yes	no	moderately

Table 5.2. Overall assessment of Frequency-band Coefficient Shuffling

Scalable Coefficient Encryption. Cheng and Li [21] propose to encrypt the low-frequency DCT coefficients only and leave the high-frequency ones unencrypted in the JPEG environment. Their approach is therefore a partial encryption technique. The authors themselves mention that the security of this idea is questionable since when applied to all image blocks edge information remains visible. Kunkelmann and Reinema [77,76] apply this idea to the MPEG case, use DES or IDEA for encryption, and suggest to use a different amount of coefficients for I and P/B frames. In case the technique is applied to the DCT coefficients, care needs to be taken that the encrypted coefficients exhibit an admissible magnitude to be further processed correctly. In their latter paper Kunkelmann and Reinema apply this idea to the MPEG bitstream instead to coefficients. This of course raises the question of bitstream compliance as discussed in section 1.1.2 (chapter 5) “VLC Codeword Encryption”.

Wu and Kuo [177, 178] raise the same security problems with respect to scalable coefficient encryption as Cheng and Li and give visual examples how well edge information can by recovered from material encrypted in the described way. They point out that the concentration of signal energy to a few coefficients as done by most orthogonal transforms does not necessarily imply that the same is true for intelligibility, which is often scattered among all frequency components. Whereas this general observation questions all partial encryption techniques in the DCT domain in principle, wavelet based techniques show different characteristics in this respect.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
medium+scalable	0	low+scalable	yes	yes	no

Table 5.3. Overall assessment of Scalable Coefficient Encryption (in coefficient domain)

Coefficient Sign Bit Encryption. Zeng and Lei [185, 186] suggest to encrypt the sign bit of each DCT coefficient only (which is a partial encryption approach). The rationale behind this idea is that the sequence of sign bits already exhibits high entropy, consequently, a further increase of entropy caused by encryption (and with it a file size increase after compression) should not be expected. Experimental results involving a H.263 codec even show a small bitrate reduction when applying sign bit encryption. Shin et al. [140] propose to encrypt the DC coefficient sign bit in addition to zig-zag permutation (see above).

Shi and Bhargava [137, 14] propose VEA (Video Encryption Algorithm)) which relies as well on the basic principle to randomly change the sign bits of all DCT coefficients, however, they propose to apply this principle directly on the bitstream (which is possible in principle since coefficient sign bits are separated from the Huffman codewords in the bitstream).

The reduction of computational amount with respect to full encryption is significant since only 13 - 20% of all data need to be encrypted. Encryption of the sign bits can be implemented in many ways: one possibility is to XOR the sign bits with a key stream coming from a cryptographically secure stream cipher, another way would be to apply a block cipher to a set of sign bits where the order of the set equals the block size. However, similar to the case of encrypting a subset of entire DCT coefficients, Wu and Kuo [177, 178] give visual examples of attacked images which have been encrypted using this idea. These examples are fairly impressive which raises severe doubts about the security of this approach.

Secret Fourier Transform Domain. An approach significantly different from those discussed so far is to conceal the transform domain into which the image data is transformed by the compression scheme. The underlying idea is that

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
medium	medium	low	yes	yes	no

Table 5.4. Overall assessment of Coefficient Sign Bit Encryption

if the transform domain is not known to a hostile attacker, it is not possible to decode the image. Fractional Fourier domains have been used in earlier work to embed watermarks in an unknown domain [37] - Unnikrishnan and Singh [162] suggest to use this technique to encrypt visual data. In particular, the input plane, the encryption plane, and the output plane of the proposed method are fractional Fourier domains related to each other by a fractional Fourier transform. While the security and the size of the corresponding keyspace is discussed (though the latter not in an explicit way), the complexity is not. The authors discuss an optical implementation, but it seems that an implementation on a digital computer would be very costly due to the transforms and the additional encryption involved. It should be noted that no type of compression is involved - the question if this scheme could be somehow integrated into a compression scheme is left untouched. While being an interesting approach in principle, it seems that this technique might be used only in very specific environments and the advantage over a classical full encryption is not obvious.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
high	high	high	no	no	yes

Table 5.5. Overall assessment of Secret Fourier Transform Domain

Secret Entropy Encoding. Based on the observation that both, cipher and entropy coder, turn the original data into redundancy-free bitstreams which cannot be decoded without certain information, Wu and Kuo [177] discuss the possibility to turn an entropy coder into a cipher. The information required for decoding is the key in the cryptographic case and the statistical model in the entropy coder case. The authors refer to earlier work where it is shown that is is extremely difficult to decode a Huffman coded bitstream without the knowledge of the Huffman codewords. Shi and Bhargava [138] (see the section on Zig-zag Permutation) suggest a codeword permutation which has a very limited keyspace. Most earlier work on using entropy coders as ciphers has been done for arithmetic coding.

Wu and Kuo propose to use multiple Huffman coding tables (MHT) out of which a specific one is selected based on random decisions for encoding a given symbol. In order to cope with maintaining the compression ratio, it is suggested to use a different set of training images for each table. Huffman tree mutation can also be used to create a large number of Huffman tables out of 4 initially given tables. In their analysis, the authors state that their approach is vulnerable against the chosen plaintext attack, but not against known plaintext and ciphertext only attacks. In their later work [178], the authors suggest to enhance the security of their scheme by inserting additional bits at random positions and by doing this in different ways for different portions of the data.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	high	no	no	moderately

Table 5.6. **Overall assessment of Secret Entropy Encoding**

1.1.2 Bitstream Oriented Schemes

Header Encryption. The most straightforward to encrypt an image or video bitstream is to encrypt its header. Bitstream compliance is immediately lost and the image or video cannot be displayed any longer using a common player. However, in case an attack is conducted against this kind of encryption, most header informations can simply be guessed or extracted from the syntax of the bitstream itself. As a consequence, the security of such a scheme depends on the type of header data encrypted – if the protected header data can not be guessed or computed by analysing the bitstream, and if this data is crucial for the decoding of the visual data, this approach could be secure. We will discuss this approach for the DCT-video case in some detail in section 1.2.1 (chapter 5) and in section 2.1.4 (chapter 5) for the wavelet case. In any case, header encryption is an interesting approach since it requires minimal encryption effort only.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low+scalable	low	low+scalable	no	yes	no

Table 5.7. **Overall assessment of Header Encryption**

Permutations. Permutations are a class of cryptographic systems well suited for designing soft encryption schemes and have been as well proposed to be applied at the bitstream level, however, all these schemes are extremely vulnerable against a known plaintext attack as described in the context of the Zig-Zag Permutation Algorithm. The entities subject to permutation (*basic shuffling units [174]*) are different when comparing the suggestions made so far.

1 **Bytes** Qiao and Nahrstedt [120] discuss the *Pure Permutation Algorithm* where single bytes of an MPEG video stream are permuted. Depending on the security requirements the permutation lists in use can be made longer or shorter. Whereas this approach is extremely fast in terms of the actual encryption process and in terms of parsing the bitstream to identify the data subject to encryption, the semantics of the MPEG stream are entirely destroyed and no bitstream compliance is obtained.

2 **VLC codewords** Based on their earlier Frequency-band Coefficient Shuffling idea [185], Wen et al. [174] and Zeng et al. [184] propose to shuffle VLC run-level codewords corresponding to single non-zero DCT coefficients. Codewords from different 8×8 pixels blocks are grouped together according to their codeword index and permuted with one permutation list. The number of groups and the range of codeword indices within one group can be adjusted according to security requirements. A problem with this approach is that different 8×8 pixels blocks usually contain a different number of non-zero coefficients which can be resolved by controlling the “last field” of each block. Format compliance may be guaranteed by truncating codewords eventually exceeding 64 coefficient per 8×8 pixels blocks, but there is of course significant processing overhead as compared to the Pure Permutation Algorithm induced by identifying VLC codewords and grouping of codewords. Kankanhalli and Guan [67] independently propose exactly the same idea, they further increase the security of their system by additionally flipping the last bit of the codewords randomly and apply corrections if the prefix of the subsequent codeword is affected.

3 **Blocks and Macroblocks** In the same papers, the authors also discuss the use of 8×8 pixels blocks and macroblocks as the basic shuffling units. Whereas in the case of macroblocks format compliance is guaranteed, in the case of 8×8 pixels blocks care must be taken about the different VLC tables used to encode inter and intra coded blocks, i.e. these blocks need to be permuted separately or the corresponding flag in the bitstream needs to be adjusted if the type of block was changed by permutation. Obviously, bitstream compliance may be achieved easily using this approach and the processing overhead is also significantly smaller as compared to the codeword permutation case. However, there is an important security problem inherent to this method. This approach is equal to a permutation of (smaller or larger) image blocks in the spatial domain and it is widely accepted that such a technique is vulnerable to a ciphertext only attack. It is only neces-

sary to group blocks with corresponding or similar boundaries together to get a good approximation of the frame.

Bytes					
time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	low	no	yes	no
VLC Codewords					
time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	medium	low	yes	yes	no
Blocks and Marcoblocks					
time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	low	low	yes	yes	no

Table 5.8. Overall assessment of Permutations applied at the bitstream level

To cope with the problem of the known plaintext attack vulnerability of permutations in general, the permutations used for encryption need to be changed frequently. This adds significant load in the area of key management and key distribution which is particularly bad since these operations often involve public-key cryptography. In order to avoid such problems, Wen, Zeng, et al. [174, 184] propose an on the fly generation of permutation lists using parts of the bitstream which are not involved in the permutations (e.g., in case of VLC codeword permutation DES encrypted DCT sign bits could be used, which are not required to be permuted due to their high "natural" entropy). In the following a corresponding example is given [174, 184]:

- Encrypt DCT sign bits (key K_F).
- Generate a random bitsequence R_L of length L (using a stream cipher controlled by key K_L), with $L > bitlength \times K$ for all $bitlength \times K$, where K is the number of codewords in a codeword table and $bitlength$ is approximately $log_2(K)$.
- For each set of codewords to be permuted, concatenate the encrypted sign bits to R'.
- R' is encrypted using K_F which results in the output R, which is repeated $bitlength$ times to result in Rr.
- $Rc = R_L \quad XOR \quad Rr$.
- Rc is partitioned into K non-overlapping segments with each $bitlength$ bits. The permutation table maps each index input value i ($0 \leq i \leq K-1$) to the i-th segment of Rc.

Note that the techniques of Wen et al. [174] have been adopted by the MPEG-4 IPMP standard.

Tosun and Feng [157] discuss interesting properties of permutations in the context of wireless video transmission. It is shown that permutations can be the basis of *errorpreserving video encryption* algorithms in the sense that after a transmission error the incorrect information is not propagated to other parts of the data as it would be the case in classical encryption schemes (caused by the avalanche effect). This is of course desirable in wireless video transmission where large amounts of channel errors occur. Limiting the error after decryption to the location where the error occurred during transmission reduces the task of error concealment considerably as compared to the case where e.g. an entire 128 bit block is destroyed (as it would be the case with AES).

One-time pad VEA. Qiao and Nahrstedt [119, 120] propose another partial encryption VEA (**V**ideo **E**ncryption **A**lgorithm)) which operates on MPEG streams at the byte level. Odd-numbered and even-numbered bytes form two new byte streams, the *Odd List* and the *Even List.* These two streams are XORed, subsequently the Even List is encrypted with a strong cipher (DES was suggested at that time). The result of the XOR operation and the encrypted Even List are the resulting cipher text. As a consequence, the DES encrypted half of the byte stream serves as a one-time pad for the other half which makes the system fairly secure, because there exists low correlation between bytes and pairs of bytes in MPEG streams (this is confirmed experimentally [119, 120]). This exploits the fact that both compression and encryption decorrelate the data. In order to further increase the security the following improvements are suggested:

- The fixed odd-even pattern is replaced by two randomly generated byte lists (where this is controlled by a $128 - 256$ bit key).
- Each set of 32 bytes is permuted, 8 different permutation lists are employed which are used in fixed order.
- The generation of the two byte lists is changed for every frame (of a video).

This algorithm exhibits very high security but this is achieved at a relatively high computational cost (as compared to full DES encryption, only 47% of the overall computations are saved). Additionally, operating at the byte level the semantic of the MPEG stream is entirely destroyed, bitstream markers may be emulated, and there exists no bitstream compliance after encryption of course.

Tosun and Feng [156] suggest to apply this idea recursively thereby reducing the necessary amount of encryption by one half in each recursion. The Even List which would be subject to encryption in the original scheme is instead again split into an **Odd$_2$** List and an **Even$_2$** List which are XORed. Now, the

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
high	low	high	no	yes	no

Table 5.9. Overall assessment of One-time pad VEA

Even_2 List (one quarter of the original data) can be encrypted and the procedure terminates or it is again split into an Odd_3 List and an Even_3 List. This strategy of course reduces the encryption effort significantly, but the security is also weaker since the one-time pad is no longer one-time in a strict sense [85]. Bitstream compliance is equally destroyed as in the original scheme.

Byte-Encryption. Griwodz et al. [55, 56] propose to randomly destroy bytes in an MPEG stream for free distribution, while the original bytes at the corresponding positions are transferred in encrypted form to legitimate users. This is actually equivalent to encrypting bytes at random positions. The authors find that encrypting 1 % of the data is sufficient to make a video undecodable or at least unwatchable. However, the cryptanalysis given is entirely insufficient. Consider the worst case where only MPEG header data is encrypted by chance using this approach. It is well known that header data may be reconstructed easily provided the encoder in use is known. Additionally, no attack scenario is considered but only the case of playing the protected video in a standard decoder is covered. In order to guarantee a certain level of security, a higher amount of bytes need to be encrypted and care needs to be taken about which bytes are encrypted. Wen et al. [174] describe a more general approach as part of the MPEG-4 IPMP standard, named *Syntax Unaware Runlength-based Selective Encryption* (SURSLE). This algorithm encrypts X bits, the next Y bits are left in plain-text, the next Z bits encrypted again, and so on. In addition to the abovementioned security problems, both approaches partially destroy the MPEG bitstream syntax (which is the main security approach of these schemes) and potentially emulate important MPEG markers causing a decoder to crash (which is again desired).

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
medium+scalable	low	low+scalable	no	yes	no

Table 5.10. Overall assessment of Byte Encryption

Alattar et al. [6] consider a somehow related approach by encrypting only every other basic encryption unit as opposed to other techniques where all such units are protected, although this is done in the context of a inter-frame encryption approach.

VLC Codeword Encryption. Whereas Byte Encryption does not take the syntax of the video into account, VLC codeword encryption does. Contrasting to the permutation of such codewords, strong encryption should be applied in this case. In case bitstream compliance after encryption is not the aim, Byte Encryption applied to a significant fraction of the bitstream is a better choice since VLC codewords need not be identified and therefore bitstream parsing may be avoided to a large extent. In case bitstream compliance after encryption is the aim, when encrypting VLC codewords we face the problem the the encryption of a concatenation of VLC codewords leads not necessarily to a concatenation of valid codewords. For example, given the codewords 0, 10, 110, 111, and a possible concatenation 010, a possible encryption of 010 may lead to 001 which is no valid codeword concatenation and would therefore destroy bitstream compliance. Wen et al. [173, 174] propose a solution to this problem, which has also been adopted for the MPEG-4 IPMP standard. The technique works as follows for a VLC table with $N = 2^k$ entries. Before encryption, a fixed length k bit index I is assigned to each codeword in the VLC table. After a concatenation of VLC codewords is obtained which should be encrypted, a bit string S is constructed by concatenating the indices I of the corresponding codewords. S is encrypted with a secure cipher which results in S'. S' is than mapped back to codewords using the same index-to-codewords map used before for constructing S. The result will be a different concatenation of valid VLC codewords, which are inserted back into the bitstream at the positions of the original codewords. Non power-of-two VLC tables can be treated by decomposing them into several power-of-two tables. While this scheme guarantees a standard compliant bitstream it does not preserve the size of the bitstream. In general, the original and "encrypted" concatenation of codewords will not have the same size (although the number of codewords is equal) – the examples in [174] exhibited an overhead of about 9% of the original filesize. Of course, the computational overhead to identify codewords, build the index lists, perform the mappings, and to reinsert the data is significant.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
medium+scalable	high	medium+scalable	yes	yes	moderately

Table 5.11. Overall assessment of VLC Codeword encryption

1.2 Video Encryption

Video encryption based on DCT methods is focused on standardised formats like MPEG-1,2,4 or H.26X, therefore all these techniques try to take advantage of the corresponding data formats and bitstreams. Whereas all the techniques discussed subsequently could as well be applied during the compression stage, they are mostly discussed in the context of directly manipulating the bitstream data (after compression has taken place).

1.2.1 Bitstream Oriented Schemes

Most schemes for video encryption combine various ideas and are neither purely frame-based nor purely motion-based. Therefore, we will first discuss the main ideas and their properties, subsequently we will describe some complete proposals as given in literature. Note that all techniques described in the section on image encryption may be applied to single frames or the bitstream of videos as well.

Header Encryption. As the second-lowest security level in their SECMPEG scheme (see below), Meyer and Gadegast [97] propose to encrypt all (MPEG) header data of the MPEG sequence layer, group of picture layer, picture layer, and slice layer. Lookabaugh et al. [85] discuss in detail which types of header data are interesting candidates for being encrypted. The data suggested by Meyer and Gadegast to be selectively encrypted turns out to be hardly suited for that purpose (except for the `quantizer_scale_code` field in the slice header). They suggest to encrypt the `macroblock_type` field in the macroblock header since this data covers only about 3.5% of the entire bitstream and its encryption poses severe challenges to a decoder since this field specifies how the following bits are to be parsed and it is a VLC field which may cause the decoder to get out of sync with the bitstream in case it is not correct. Wen et al. [174] investigate the encryption of the *Dquant* parameter (difference of quantisation step size QP between current and previous macroblock), which is a very simple approach since many macroblocks simply use the default settings which makes this parameter easy to attack.

For an overall assessment, see section 1.1.2 (chapter 5).

Encryption of I-Frames. Spanos and Maples [145] and Li et al. [83] independently propose to encrypt I-frames only in 1995. Since P and B-frames are reconstructed based on predictions obtained from I-frames, the main assumption is that if these are encrypted, P and B-frames are expected to be protected as well. This very simple idea has still been suggested in 2003 for a combined DVD watermarking-encryption scheme by Simitopoulos et al. [141] and for a wireless multimedia home network by Taesombut et al. [148]. There are several problems associated with this approach. First, the percentage of the bitstream which is comprised of encoded I-frames is about 25-50% which means that this approach does not reduce the computational complexity to a satisfying extent. Second, the motion in the video remains visible, especially when replacing the encrypted I-frames by uniform frames. A strategy to cope with this would be to increase the number of I-frames which is on the one hand good for security, on the other hand degrades compression performance. For example, increasing the share of I-frames from 1/3 to 1/6 in the "Miss America" and "Flowers" sequences raises the video file size by 50% [4]. Third, and even more severe, the I-blocks contained in P and B-frames resulting from poor prediction results even deliver texture information of I-frames when collected over several frames. This especially affects high motion sequences since in this case P and B-frames contain many I-blocks. Agi and Gong [4] noted this problem in 1996 and suggested to encrypt also these I-blocks. This technique is found as well in one mode of the SECMPEG scheme, in one mode of the combined watermarking encryption scheme of Wu and Wu [179], and has been also suggested in 1999 by Alattar and Al-Regib [5], apparently unaware of all the previous work in this direction. While the security as compared to pure I-frame encryption is increased, the amount of data to be encrypted increases as well (up to 40 - 80%) and the problem of still visible motion content remains unresolved. Additionally, the bitstream parsing effort to identify all I-blocks in P and B-frames is significant. In order to reduce to computational overhead, Alattar et al. [6] suggest to encrypt every other I-block (reducing the encryption percentage to 20 - 40% which is still a lot), however, still the entire approach remains quite insecure.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
medium	medium	low	yes	yes	no

Table 5.12. Overall assessment of I-frame Encryption

Encryption of Motion Vectors. Motion vectors comprise about 10% of the entire data of an MPEG video [85], therefore, restricting the encryption to motion vectors might be an interesting idea. However, from a security viewpoint encryption motion vectors alone can never be sufficient since all texture information from I-frames remains in plain text. Consequently, a video with very low temporal rate would be in plaintext in any case. I-frame material needs to be secured additionally to provide reasonable security. Similar to the DCT coefficient case, motion vector sign bits or VLC codewords can be protected. Some examples are provided in the more comprehensive schemes described below.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	medium	low	yes	yes	no

Table 5.13. Overall assessment of Motion Vector Encryption

SECMPEG. SECMPEG defines a new bitstream including a header structure which makes it incompatible with respect to common MPEG players. Besides its data integrity and authentication functionality (which we do not discuss here), five levels of security are defined:

1 No encryption.

2 Header data from the sequence layer down to the slice layer is encrypted.

3 Encrypt same data as in level 2 and the low frequency DCT coefficients of all blocks in I-frames.

4 Encrypt all I-blocks (also those in P and B-frames).

5 Encrypt the entire video.

As can be seen, three different basis techniques are combined into SECMPEG and all their properties and restrictions apply correspondingly: Header encryption, I-frame encryption, and Scalable Coefficient encryption.

Alattar et al. [6] propose as well a scheme with scalable complexity and security using three levels:

1 Encrypt all data associated with every n-th I-macroblock.

2 Encrypt all data associated with every n-th I-macroblock and all header data of every n-th P and B-encoded macroblocks.

3 Encrypt all data associated with every n-th I-macroblock and all header data of P and B-encoded macroblocks.

Again, Header encryption and I-frame encryption is combined. As a third example for an explicitly scalable scheme we describe the security levels of the combined watermarking encryption scheme by Wu and Wu [179]:

1 Encrypt the DC coefficient of the luminance component of all I frames.

2 Encrypt the luminance and chrominance DC coefficients of all I frames.

3 Encrypt all luminance and chrominance coefficients of all I frames.

4 Encrypt the DC coefficient of the luminance component of all I macroblocks.

5 Encrypt the luminance and chrominance DC coefficients of all I macroblocks.

6 Encrypt all luminance and chrominance coefficients of all I macroblocks.

7 Encrypt the data of all frames (but no header data).

Contrasting to the other two suggestions no header data is encrypted which enables the scheme in principle to deliver compliant bitstreams. Additionally, an explicit distinction between luminance and colour component is made, where securing the luminance component is of course more important from a perceptual viewpoint. The scheme by Wu and Wu again combines I-frame encryption and Scalable Coefficient encryption.

MVEA and RVEA. Shi and Bhargava [14, 139, 136] suggest to improve their former algorithm VEA by encrypting the sign bits of the motion vectors in addition to the sign bits of the DC coefficient of I-blocks. This technique is denoted MVEA (if sign bits are randomly changed by a secret key – which is vulnerable to a plaintext attack) or RVEA (if sign bits are encrypted by DES or IDEA). Both, DC coefficients and motion vectors are differentially encoded which causes significant impact when the corresponding sign bits are changed. The authors state that by encrypting motion vector data the content of P or B-frames does not have to be protected further in principle. The give a fixed scan order through the data of a macroblock, at most 64 sign bits are encrypted per macroblock. For I-macroblocks, first the luminance and chrominance DC coefficient sign bits are encrypted, subsequently the lowest frequency AC coefficient sign bits and so on. For P and B-macroblocks, the first sign bits to be encrypted are the sign bits of motion vector data, then the scan proceeds as in the case of I-macroblocks. About 10% of the entire video data consists of sign bit data which makes the approach interesting from the less computations viewpoint.

MVEA and RVEA combine Coefficient Sign bit encryption, Scalable Coefficient encryption, and Motion Vector encryption.

Techniques in MPEG-4 IPMP. In their simulations under the MPEG-4 IPMP framework Wen et al. [174] investigate three different configurations with increasing security:

1 Encrypt the FLC coded DCT sign bit, the parameter DQUANT (two bits determining the difference between the quantisation parameter used for the previous macroblock and the current one), and the I-macroblock DC value.

2 Encrypt the VLC motion vector field.

3 Encrypt the data of both previous suggestions.

This proposal combines several techniques as well: I-frame encryption, Coefficient Sign Bit encryption, Header encryption, Motion vector encryption, and Scalable Coefficient encryption. Whereas the first two options are said to be useful for entertainment purposes only, the third configuration provides satisfying results. However, the authors discuss error concealment attacks against some of the proposed parameters: setting DQUANT and the motion vector field simply to zero, and the DC coefficients to a constant value significantly improves the reconstruction of an encrypted video. The authors therefore propose to combine their suggested encryption configurations with permutations to achieve a higher security level.

1.3 Our Implementations of selective MPEG-encryption

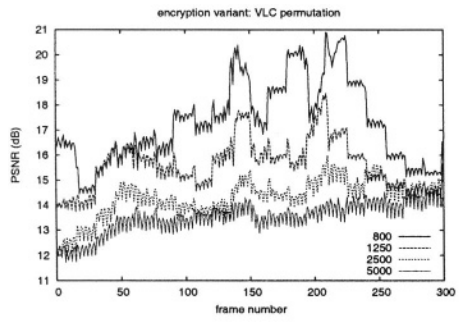
1.3.1 General Information

For our assessment on the effects of the various encryption method proposed in the literature we recreated a number of them, following their description to a higher or lesser degree. We integrated these encryption schemes into an open-source MPEG encoder called `mpeg2enc` from the `mjpegtools` package available at `http://sourceforge.net/projects/mjpgtools/`. This enables us to compare them in the same environment with the same sequences and settings, like the target bitrate. These experiments may be performed online at `http://www.ganesh.org/book/`. We conducted our experiments at 4 different target bitrates (800, 1250, 2500, 5000 kbps (kilo Bits per second)), and we used 5 different sequences (Bowing, Surf Side, Coast Guard, Akiyo, Calendar). Sample frames of these images are shown in the appendix on page 139. Although we performed our experiments with all 5 sequences we do not intend to show results from the surf side sequence since there are two major differences with the other sequences which make it hardly comparable: the sequence is very short, and it is no colour sequence.

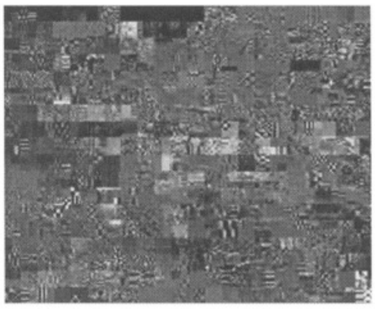
1.3.2 VLC Table Codeword Permutation

The coefficients in a transformed 8*8 DCT block are scanned in zig-zag-order, each non-zero value is encoded using a predefined entry in a VLC (variable length code) table. The actual value does not only depend on this non-zero value, but also on the number of coefficients with 0 value immediately before this value. The entries in this table can be as long as 16 bits, and to be decodable any codeword must not be the prefix of another codeword (this requirement is also called “Fano condition”).

A possible way to encrypt a bitstream is to exchange the entries in the VLC table. When a value/runlength pair is encoded a different codeword is retrieved from the table than it would be in the unencrypted case. Several consequences follow: when random codewords are used they are in general not prefix-free.



(a) resulting quality for 4 different bitrates



(b) frame #180, bitrate 5000 kb/s

Figure 5.1. VLC encryption results with test sequence #1

So the easiest way to generate another set of valid codewords is to use the existing ones and to permute them. In this scenario the encryption key is the seed value for a PRNG which generates to permutation. This approach also gives to opportunity to produce a standard-compliant bitstream. The next problem is that the codewords have been chosen to be as good as possible with respect to compression performance, that means that for combinations which are very likely to appear short codewords are assigned to. An unrestricted permutation disturbs this order and the resulting bitstream is longer than necessary. The countermeasure to this problem is to restrict the permutation further: Just codewords of the same length are allowed to be exchanged. On the other hand this leads to a decreased security: The number of short codewords is limited, but they are very probable. An attacker can expect a high number of them and he can try all possible combinations of them to reconstruct the image, and he can ignore the longer ones. This results in a image lacking the high frequencies. This method is related to the “Zigzag permutation algorithm” on page 47, “Secret Entropy Coding” on page 51, and “VLC Codeword permutation” on page 53.

Experiments. This encryption scheme is similar to the VLC codeword permutation described in section 1.1.2 (chapter 5). As you can see in figure 5.1(a) the quality of the resulting encrypted bitstream is below 20dB PSNR in general, with an average of 15dB. In figure 5.1(b) you can see one of the “best” frames, nothing recognisable is left. We can also determine a slight dependence of the quality of the resulting video on the target bitrate.

1.3.3 Macroblock Permutation

Each frame within a video consists of the same number of macroblocks, each macroblock contains such data as quantised coefficients from the Y, U and V bands, or motion vectors. A variant to encrypt videos is to exchange the macroblocks within a frame. The key for this encryption method is used as a seed value for a PRNG which generates a permutation.

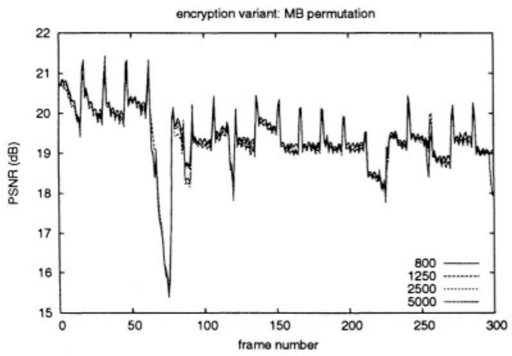
This is an encryption variant which is annoying but not very secure. The reason is that based on the correlation of border pixels the originally neighbouring macroblocks can be regained. This effect becomes even easier when there are more frames available which are permuted using the same order. So to keep this approach reasonably secure it is necessary to change the key as often as possible, at best with every frame. However the initial insecurity still exists. This variant the variant described in the next section are similar to "Blocks and macroblocks" on page 53.

Experiments. Because the basic shuffling unit here is the macroblock it can be expected that although the entire visual information is present the content can not by recognised instantly. The frames are degraded because the information is not on the expected place. Sequence 3 shows better results than the other sequences (compare figures 5.2(a) and 5.2(c)). This can be easily explained because when a block containing water is exchanged with another block containing water the difference is not very significant (see figure 5.2(b)). Other sequences show "worse" results, e.g. sequence 5 (see figures 5.2(c) and 5.2(d)). The spikes shown on the graphs indicate the I-framesI-frame, the two frames shown were coded as I-frame. When looking at the results of MB permutation encryption we are unable to make out any significant dependency of the resulting quality on the target bitrate, therefore the decision to show the sample frames for bitrate 1250 was random.

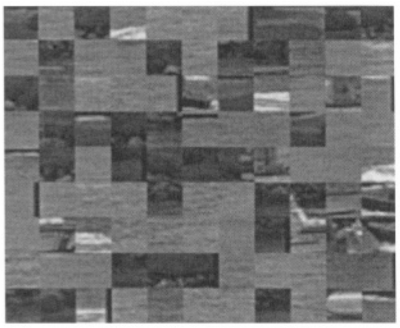
1.3.4 DCT Block Permutation

Similar to the approach where complete macroblocks are exchanged it is possible to exchange the individual 8*8 DCT coefficient blocks. By permuting the smaller blocks the confusion being created is bigger, and reconstruction becomes more difficult but not impossible. Additionally the algorithm does not discriminate between DCT blocks from the Y plane and DCT blocks from the U and V planes. Again examples with the "best" results (=high PSNR) are shown, in this case sequence 4, see figure 5.3.

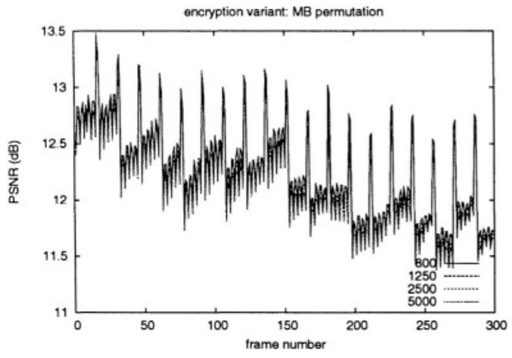
Experiments. Figure 5.3(a) shows that the compression rate does not influence the performance of the encryption. Although the image looks indecipherable the same reconstruction method can be applied as in the macroblock permutation method shown above: blocks which are neighbours in the origi-



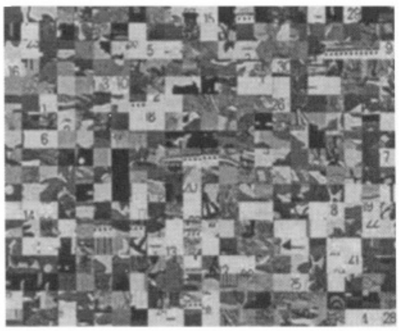
(a) resulting quality for sequence 3



(b) frame #48, bitrate 1250 kb/s (sequence 3)



(c) resulting quality for sequence 5



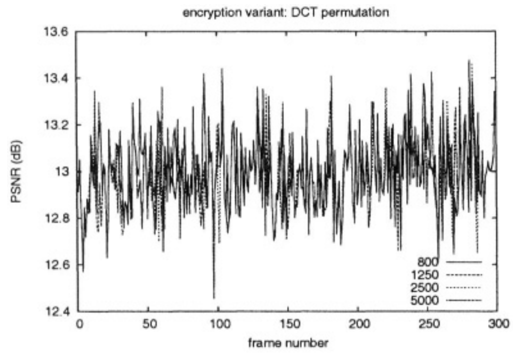
(d) frame #48, bitrate 1250 kb/s (sequence 5)

Figure 5.2. MB permutation, results with test sequences #3 and #5

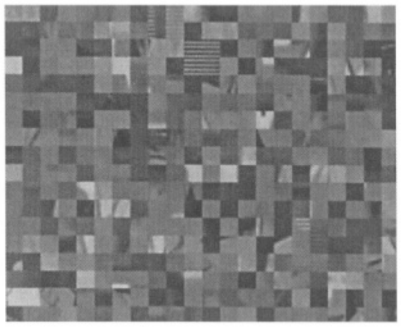
nal frame have similar pixels at their boundaries, using this correlation many blocks can be recovered. More correlations (respectively a scale factor of 2) exist between the blocks of the U and V plane and the Y plane blocks. The reason that the figure 5.3(b) looks almost random is because some of the 8*8 pixel Y plane blocks end up as 16*16 pixel U or V plane blocks, and vice versa. Therefore a human eye can hardly detect any features.

1.3.5 Motion Vector Permutation

Similar to the macroblock permutation and the DCT block permutation it is possible to permute the motion vectors which are assigned to distinctive



(a) resulting quality for 4 different bitrates



(b) frame #1, bitrate 2500 kb/s

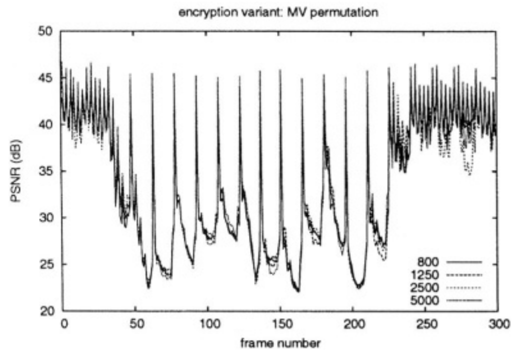
Figure 5.3. DCT block permutation, results with test sequence #4

macroblocks. Within a predicted frame each macroblock can be either an I-block or a predictive block, motion vectors are just assigned to the latter. These vectors can be permuted according to an order provided by a PRNG, where the seed is the key.

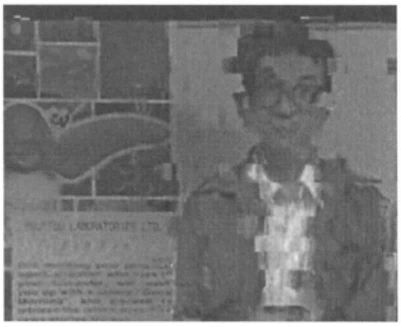
The distortion which results from this encryption method is very light, since it affects no I-frames and no I-blocks, and since in many cases many motion vectors within a frame share the same overall direction. However the effects increase with the number of successive P- or B-frames, and they vanish with the next I-frame (obviously). This variant is related to the MVEA and RVEA methods.

Experiments. In this experiment we exchanged the motion vectors. Obviously this method leaves I-blocks and I-frames unaltered. Since the motion vectors are just exchanged within the current frame, but not encrypted or altered otherwise, a global motion (such as a camera pan movement) leads to very small distortions — even when exchanged all motion vectors point to the same general direction. This effect can be partially observed at test sequence #3 (coast guard).

Viewers of such an encrypted bitstream still know about what is going on since they see the keyframes and the visually sensible degradation of the first and last predicted frames is less than in the middle of a GOP (group of pictures). So it is annoying for viewers but not unobservable. Figure 5.4(a) shows the PSNR plot of the encrypted Bowing sequence, during phases with no apparent motion the quality of both I-frames and B/P-frames is approximately the same, but when there is significant motion then the quality drops at the B- and



(a) resulting quality for 4 different bitrates



(b) frame #100, bitrate 5000 kb/s

Figure 5.4. Motion vector permutation, results with test sequence #1

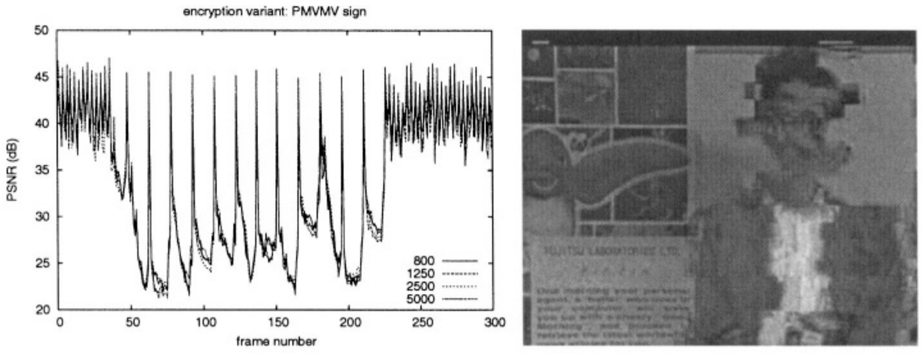
P-frames. Figure 5.4(b) is an example for a predicted frame with a medium amount of motion, the PSNR is 27.2dB, the minimum, average and maximum values are 22.2, 33.3 and 46.7 dB, respectively. This behaviour is the reason why this technique has been suggested to be used as a transparent encryption technique (compare section 4 (chapter 5)).

1.3.6 Motion Vector Sign Change

Motion vectors are signed values, and therefore another possibility for light encryption is to change to sign bits of these vectors. The actual motion vectors are not stored in the bitstream, they are predicted based upon previous motion vectors, and just the prediction error (residual) is stored. So we have two variants for sign encryption: change the signs of the prediction, and change the signs of the residual.

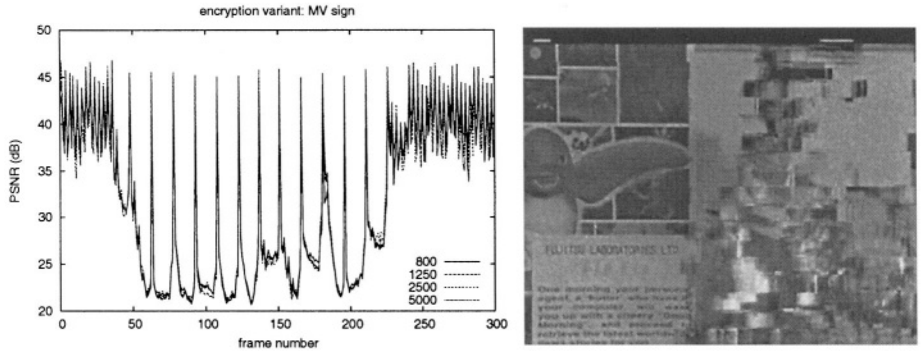
Again the decision which signs are changed and which are left as they were is based on the output from a PRNG, again with the key as its seed value. And again the distortion generated by this encryption method is very light. A countermeasure is even easier as an attacker has just to try all 2^2 (in directions x and y) or 2^4 ($x + y$ again, and two fields in the interlaced case) sign combination of either the predicted motion vectors or on the residuals, and then choose the best one — based on the correlation with the neighbour blocks.

Experiments. The encryption of the sign bits is done with the help of a pseudo-random sequence of bits. The algorithm works similar to the encryption method explained above, the permutation of motion vectors. Therefore the effects are similar: I-blocks are unaffected, small amounts of motion cause



(a) resulting quality for 4 different bitrates (b) frame #100, bitrate 5000 kb/s

Figure 5.5. Motion vector prediction sign change, sequence #1



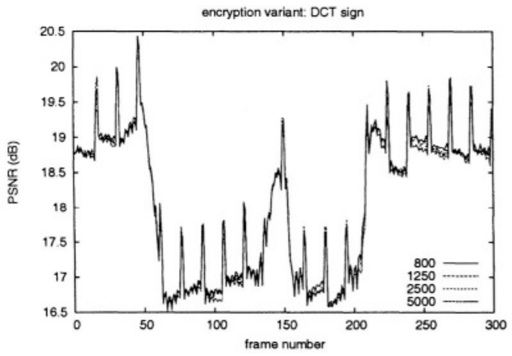
(a) resulting quality for 4 different bitrates (b) frame #100, bitrate 5000 kb/s

Figure 5.6. Motion vector residual sign change, sequence #1

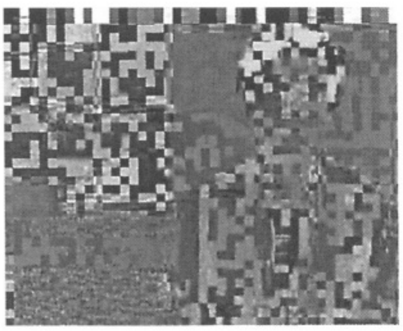
minor distortions,... The quality figures show that the second variant, the encryption of the actually stored/transmitted sign of the residual results in slightly higher distortions (compare figures 5.5(a) and 5.6(a)).

1.3.7 DCT Coefficient Sign Bit Encryption

Apart from the motion vectors the transform coefficients are signed values as well. So it is possible to change their signs in a pseudo-random manner as well. There are several variation possible: the most simple variation is to change the sign bits of all blocks. Other variations are to change just the sign bits of I-



(a) resulting quality for 4 different bitrates



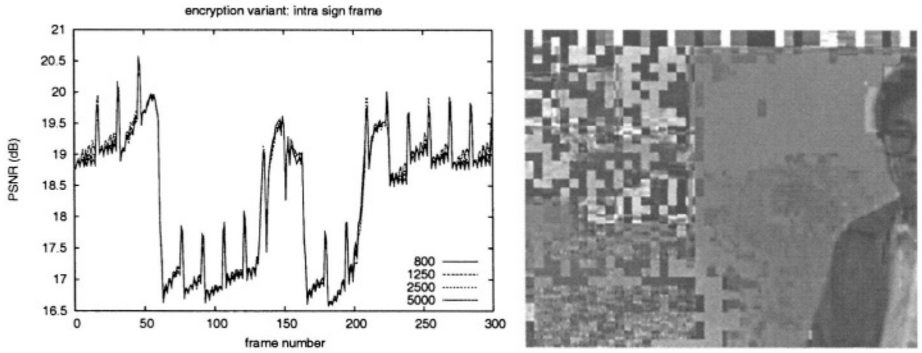
(b) frame #100, bitrate 5000 kb/s

Figure 5.7. DCT coefficient sign change, results with test sequence #1

blocks, either just the I-blocks which are located in I-frames, or additionally the I-blocks in predicted frames. The latter can be seen as a complementary option to the various motion vector encryption methods, as they do not change any I-block. For more information see page 50.

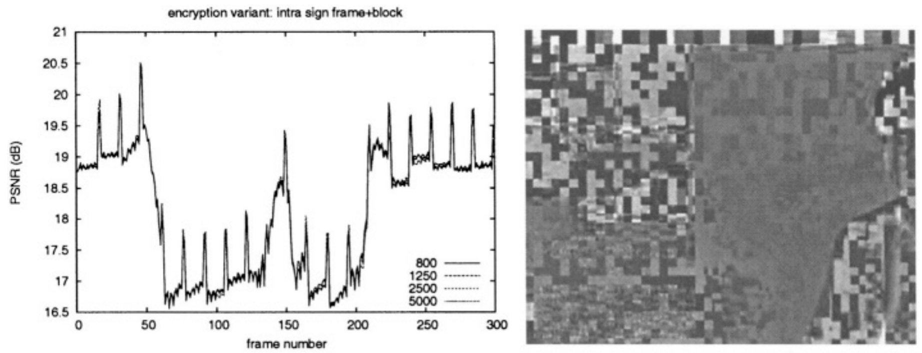
Experiments. We performed three different experiments as described above: in the first test we allowed to change the sign bits of all DCT blocks. A result can be DCT block seen in figure 5.7, this time the example frame is one of the frames resulting in a low PSNR value. The visual impression is that the colours and the luminance are changed, but it is still possible to get an idea about the contents. This is because the absolute values of the coefficients do not change, this means that blocks with a large amount of high frequency are still blocks with a large amount of high frequency, and blocks with a small high frequency amount are as flat as before: the objects are distorted, but the areas containing edges are recognisable. It is possible, too, that in almost-static scenes which cover more than one GOP averaging might cancel out the strongest effects.

In the other two experiments shown in figures 5.8 and 5.9 the signs of the coefficients of I-blocks were changed in a pseudo-random manner. The difference is that in figure 5.8 I-blocks which are embedded in P- or B-blocks were not encrypted. This makes a significant difference when there is a large amount of motion in the frame, so that the MPEG encoder decides it is better to include I-blocks than residual blocks. In figures 5.8(a) and 5.9(a) we see that the overall performance difference is not very large, just at frames with large motion (the person enters or leaves the view, or bows) the second half of



(a) resulting quality for 4 different bitrates (b) frame #55, bitrate 5000 kb/s

Figure 5.8. I-frame DCT block sign change, results with test sequence #1



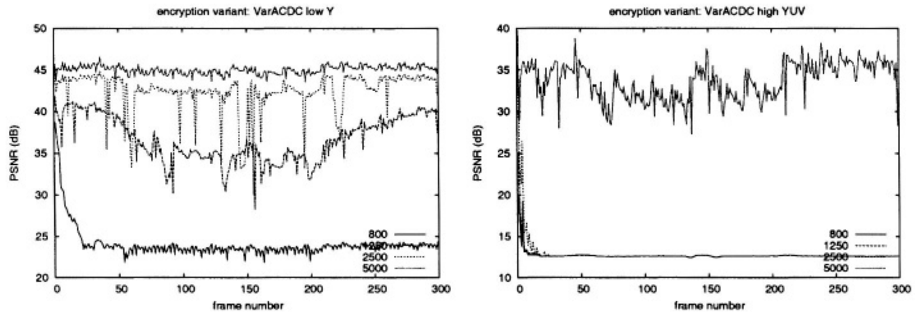
(a) resulting quality for 4 different bitrates (b) frame #55, bitrate 5000 kb/s

Figure 5.9. I-frame + I-block DCT block sign change, results with test sequence #1

the figures show slightly lower PSNR. The visual difference between figures 5.8(b) and 5.9(b) is obvious: in the second case everything looks "encrypted" whereas in the first case the person in motion can be clearly viewed.

1.3.8 DCT Coefficient Mangling

Before the transformed values in an 8*8 block are encoded it is possible to modify them. Individual bits are XOR-encoded with PRNG values. This is prone to generate longer bitstreams since some significant bits which were originally 0 are now changed to 1. To minimise this effect our approach is first



(a) resulting quality, low impact

(b) resulting quality, high impact

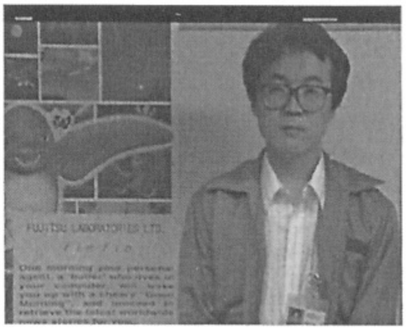
Figure 5.10. DC and AC coefficient mangling, low vs. high impact, sequence #1

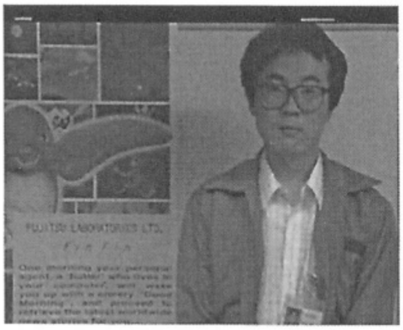
to change just values which are non-zero anyway and second to change just some less significant bits of these values, so that they change just within their order of magnitude.

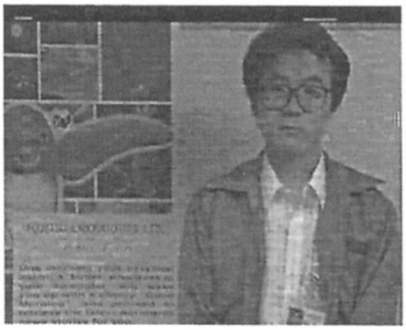
As it is not known beforehand which values are 0 and which are not, which of these non-zero positions contain large numbers and which do not, one must come up with a predefined scheme which is likely to match these requirements. On the other hand we know that in the low-frequency parts of the DCT block the probability is high to find such values. So we use the following algorithm: Take each of the first n coefficients in zig-zag order, and XOR the lowest m bits with a PRNG value. This can happen for both the Y blocks as well as for the U and V blocks. Carefully choosing n and m allows for a fine-grained tuning of the amount of encryption. This method is related to "Scalable Coefficient Encryption" as shown on page 49.

Experiments. The current implementation allows to change bits defined by the same bitmask of the first n coefficients. Usually the bitmask is set to values like 1,3,7 to allow the lowest 1,2,3 bits to be changed. A more flexible approach would change more bits at the first, the DC coefficient, and less and less bits with increasing number of the AC coefficients. A third parameter determines whether to change only some coefficients of blocks in the Y-plane, or additionally the coefficients in the U- and V-planes. With these 3 parameters it is possible to control the degree of the remaining visibility.

We performed the experiments with two sets of parameters. The first set, named "low impact", changes the lowest 2 bits of the first 4 coefficients of the Y-plane. The "high impact" setting changes the lowest 3 bits of the first 8 coefficients of the Y-, U- and V-planes. When we perform these experiments


(a) frame #100, bitrate 5000 kb/s


(b) frame #100, bitrate 2500 kb/s


(c) frame #100, bitrate 1250 kb/s

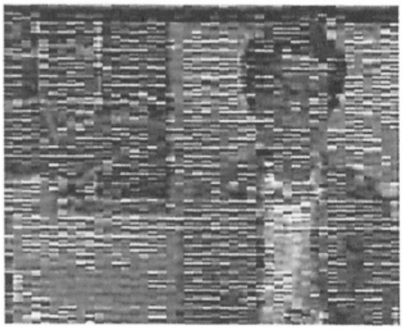
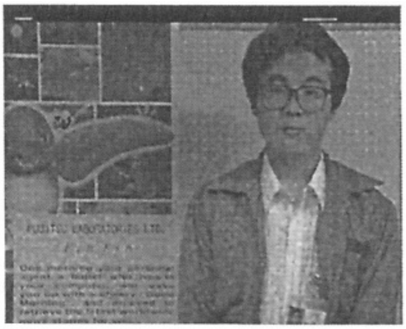

(d) frame #100, bitrate 800 kb/s

Figure 5.11. DC and AC coefficient mangling with low impact, sequence #1


(a) frame #100, bitrate 5000 kb/s

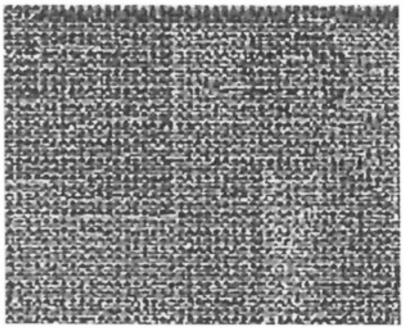
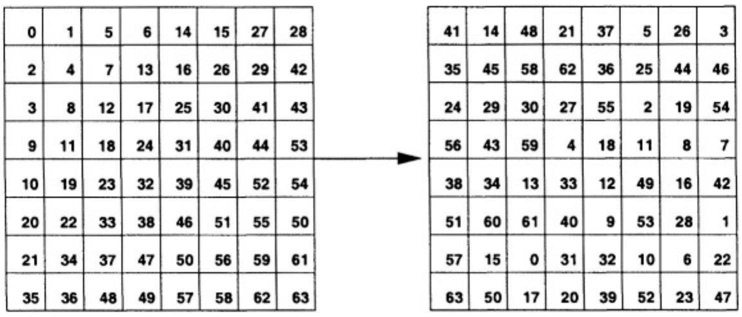

(b) frame #100, bitrate 2500 kb/s

Figure 5.12. DC and AC coefficient mangling with high impact, sequence #1



0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	50
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

41	14	48	21	37	5	26	3
35	45	58	62	36	25	44	46
24	29	30	27	55	2	19	54
56	43	59	4	18	11	8	7
38	34	13	33	12	49	16	42
51	60	61	40	9	53	28	1
57	15	0	31	32	10	6	22
63	50	17	20	39	52	23	47

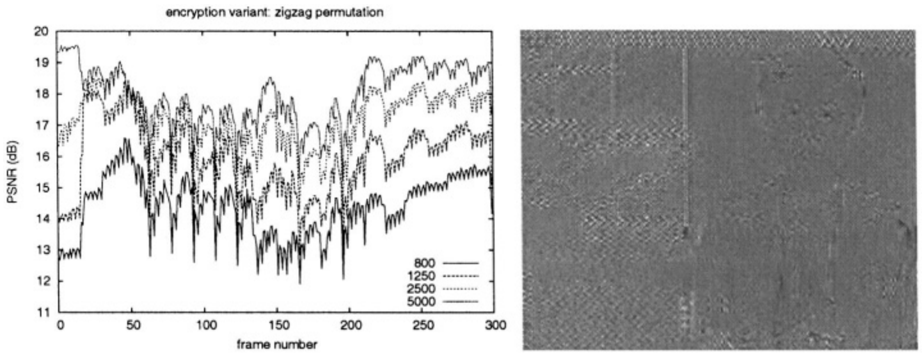
Figure 5.13. Modified Scan Order (example)

with varying compression rates we see a strong dependence between the effects of the encryption, the amount of changed bits, and the compression rate. With a high compression rate (i.e. a low bitrate) the effect of changed few bits is better observable than with a low compression rate. With an increasing number of affected bits the effects can be observed on longer bitstreams as well. So we see that it is important to tune the impact of this encryption scheme after the compression rate has been set.

Another experiment is shown on the cover-page of this book. Starting with the upper left image and advancing in scan-line order we encrypt an increasing number of coefficients while we keep the number of affected bits constant: 1,3,6,10,21 coefficients with the 4 least significant bits. When we start at the lower right image and continue left and upwards the number of affected coefficients is constant (15), but the number of affected coefficient bits increases, 1 at the lower right image, then 2,3,4. We meet at the centre with an image which originally was merged from the images of the two authors (see appendix), and then of 21 coefficients the 4 least significant bits were scrambled.

1.3.9 Zigzag Order Permutation

After the quantisation in a 8*8 DCT coefficient block the coefficients are encoded together with runs of zeros. The order starts with the DC coefficient, continues with the low frequency AC coefficients and ends with the highest frequency coefficient, the order is a zig-zag curve in the 8*8 block. A method to encrypt the data is to use a different order to encode the data, an example is shown in figure 5.13. Based on a seed for a PRNG the coefficients are permuted before the zig-zag scan is performed. The drawback of this approach is that with the perturbation of the coefficients the natural order does not exist any more, generating uncommon patterns of zeros. This means that VLC codewords with a higher length must be issued, this leads to a lower compression ratio. See also section 1.1.1 (chapter 5).



(a) resulting quality for 4 different bitrates (b) frame #100, bitrate 5000 kb/s

Figure 5.14. Zig-zag order change, sequence #1

Experiments. When the order how the coefficients are coded is changed the effect is that a decoder which is unaware of this encryption sees high-frequency coefficients at a much higher rate than usual. Usually the coefficients are clustered near the DC coefficient, but after this zigzag permutation they are spread out evenly across the whole 8*8 block. The effects can be seen in figure 5.14(b): much high frequency elements, and just a very basic outline of the real frame contents. Since this method affects all block types the differences within a sequence are negligible, and the effects (grey high frequency noise) always similar.

1.4 Encryption of Scalable or Embedded Bitstreams

Several techniques described so far clearly show that selectively encrypting visual data implies a significant processing overhead especially when bitstream compliance has to be maintained. In case a selective encryption process requires a multimedia bitstream to be parsed in order to identify the parts to be subjected to encryption, the problem of high processing overhead occurs in general. For example, in order to selectively protect DC and large AC coefficients of a JPEG image (as suggested by some authors), the file needs to be parsed for the EOB symbols 0x00 to identify the start of a new 8×8 pixels block (with two exceptions: if 0xFF is followed by 0x00, 0x00 is used as a stuffbit and has to be ignored and if AC63 (the last AC-Coefficient) not equals 0 there will be no 0x00 and the AC coefficients have to be counted). Under such circumstances, selective encryption will not help to reduce the processing demands of the entire application [114].

A possible solution to this problem is to use the visual data in the form of scalable bitstreams. In such bitstreams the data is already organised in layers according to its visual importance and the bitstreams do not have to be parsed to identify the parts that should be protected by the encryption process. However, this approach can only be taken if the data has not already been compressed into a non-scalable format (or an expensive format conversion involving partial decompression and scalable recompression needs to be applied). Additionally, the question of rate-distortion performance of the underlying scalable compression schemes is crucial since a higher bitrate of the scalable bitstream would again imply a higher overall encryption effort.

The basic idea of all these schemes is to create a base layer (which contains a low quality version of the visual data) and one or more enhancement layers (which contain the data required to upgrade the quality of the base layer). In an embedded bitstream the first part simply corresponds to the base layer and subsequent parts of the bitstream may be used to create enhancement layers. In this setting, base layer encryption is an interesting and efficient way to provide confidentiality to visual data. Encrypting the enhancement layers on the other hand is called "transparent encryption" (which is discussed in detail in section 4 (chapter 5)) and serves a different purpose, for example it may be used in a "try and buy" scenario.

The MPEG-2 and MPEG-4 scalability profiles provide three types of scalability:

- SNR Scalability: the base layer contains a full resolution but strongly quantised version of the video, the enhancement layers consist of DCT coefficient differences to weaker quantised versions of the data.

- Resolution Scalability: the base layer is a low resolution version of the video (usually generated by repeated weighted averaging and subsequent downsampling), the enhancement layers contain the difference between different resolutions of the data.

- Temporal Scalability: the base layer is a version of the video with reduced frame rate, the enhancement layers simply contain the frames required to achieve higher frame rates.

Additionally, a way has been defined in the context of ATM networks and DVB to partition MPEG-2 data into more and less important parts in order to enable unequal error protection functionality, where leading DCT coefficients and motion vector data constitute the important part and high frequency DCT coefficients the less important part. However, special MPEG units supporting this functionality are required.

In order to overcome the limitations with respect to the small number of possible enhancement layers in those schemes, the MPEG,MPEG-4 FGS (fine granular scalability) mode has been defined. After the creation of a base layer and a single enhancement layer, the latter is encoded in a bitplane oriented mode thereby creating a potentially high number of layers.

MPEG scalability profiles have not found wide acceptance due to several reasons, reduced coding efficiency in case of using a high number of enhancement layers and higher encoding complexity among them. The obvious advantages in the context of confidential video transmission might change this in the future. Additionally, the MPEG committee has recently launched a call for proposals for a scalable video codec which should overcome the problems of MPEG-2 and MPEG-4.

Kunkelmann [74] claims the MPEG-2 data partitioning scheme to be best suited for a base layer encryption approach without giving empirical evidence. Kunkelmann and Horn [76] compare the results of a base layer encryption scheme applied to a spatial domain pyramid vector quantisation codec to a non scalable MPEG-1 partial encryption technique and find it to be superior from the compression and security viewpoint. Tosun and Feng [155] define three layers in an MPEG video which consist of DC and low frequency AC coefficients (base layer), middle frequency AC coefficients (middle layer), and high frequency AC coefficients (enhancement layer). The number of coefficients in these respective layers may be altered adaptively, encryption is applied to base and middle layers only. Only the base layer is guaranteed to be transmitted. In subsequent work [156] the authors focus on wireless transmission and additionally apply forward error correction and an iterative generalisation of the VEA one time pad algorithm which reduces the amount of encrypted data significantly. Eskicioglu and Delp [41] suggest to use Shamirs (t,n)-threshold scheme for a secret sharing based key management scheme in the context of multicasting encrypted multiple layers of scalable video, Eskicioglu et al. [43] provide simulation results for that approach. Yuan et al. [183] propose an encryption scheme for MPEG-4 FGS which encrypts the base layer and the sign bits of the DCT coefficients in the enhancement layer.

While the papers discussed so far focus more on the aspect of reducing the amount of preprocessing for selective encryption by the use of scalable bitstreams, another group of papers focuses onto the networking and streaming aspect. In particular, when streaming visual data over networks with varying bandwidth, bitrate reduction might need to be performed at the network nodes where the networks changes its bandwidth. This poses two problems:

- Encryption/decryption load: In a setting using a conventional bitstream which requires transcoding for rate adaptation, encrypted visual data needs to be decrypted, transcoded, and re-encrypted again. It is evident that this puts severe computational load onto the network node.

- Key management: Even more severe, in order to be able to perform the abovementioned operations, the network node must get access to the key material required for the encryption and decryption process. This implies a significant key management challenge since this might affect several network nodes along the transmission path.

Encrypting scalable bitstreams helps to solve this problem since rate adaptation can be facilitated without the need to decrypt the data – bitrate can simply be reduced by dropping enhancement layer data no matter if encrypted or not. Of course, header data needs to be present in unencrypted form to indicate the regions of a bitstream where enhancement data is located.

Venkatramani et al. [163] describe a very general system architecture where secure adaptive streaming is supported no matter if the underlying data is in scalable format or not. The headers are left unencrypted and in case of non-scalable material several resolutions or quality levels are provided as separate bitstreams which are selected by a streaming server according to the clients' properties. Wee and Apostopoulos [171, 170] introduce a concept denoted as "secure scalable streaming" which provides the abovementioned properties based on scalable compression schemes, network packetisation techniques, and "progressive encryption techniques". The latter are encryption algorithms which match nicely to scalable codecs by allowing encrypted streams to be truncated and decrypted without sacrificing security, like block ciphers in CBC mode or stream ciphers. Based on information stored in the non-encrypted header data, even rate-distortion optimal transcoding may be achieved in encrypted form. Among several wavelet-based codecs, MPEG-4 FGS is discussed as one possible scalable codec to be employed in the system.

1.4.1 Experimental Comparison of Layered Encryption Techniques for DCT-coded Data

In this section we systematically compare the different possibilities how to organise DCT-coded visual data into several quality layers with respect to their applicability to selective encryption (compare also [46]). These experiments may be performed online at `http://www.ganesh.org/book/`.

Since no MPEG software is publicly available which implements all scalability modes, we use the progressive JPEG modes from the JPEG extended system [110]. As we shall see, the different progressive JPEG modes perfectly simulate the types of MPEG scalability. In JPEG, the terminology is changed from layers to scans.

- Hierarchical progressive mode (HP): an image pyramid is constructed by repeated weighted averaging and downsampling. The lowest resolution approximation is stored as JPEG (i.e. the first scan), reconstructed, bilinearly upsampled, and the difference to the next resolution level is computed and stored as JPEG with different quantisation strategy (similar to P and B frames in MPEG). This is repeated until the top level of the pyramid is reached. This mode corresponds well to MPEG-2 resolution scalability.

- Sequential progressive modes
  - Spectral selection (SS): the first scan contains the DC coefficients from each block of the image, subsequent scans may consist of a varying number of AC coefficients, always taking an equal number from each block. This mode is very similar to the abovementioned DVB/MPEG-2 data partitioning scheme.
  - Successive approximation (SA): the most significant bits of all coefficients are organised in the first scan, the second scan contains the next bit corresponding to the binary representation of the coefficients, and so on. Since quantisation is highly related to reducing the bit depth of coefficients, this mode behaves similarly to SNR scalability.

The JPEG standard also allows to mix different modes – an important example is to use the DC coefficient as first scan, the subsequent scans contain the binary representation of the AC coefficients as defined by successive approximation (we denote this mode as mixed (MM)). The three modes allow a different amount of scans. Whereas spectral selection offers a maximum of 64 scans, the hierarchical progressive mode is restricted to 5 or 6 sensible scans (given a $2^8 \times 2^8$ pixels image). Successive approximation mostly uses a maximum of 10 scans (depending on the data type used for coefficient representation). Similar to the scalability profile of MPEG-2, the JPEG progressive modes are not used very much and are poorly supported and documented in commercial software. Although providing much better functionality for transmission based applications, the compression performance could be expected to decrease using JPEG progressive modes. As a matter of fact, compression performance is at least as good as for the baseline system and often better (Fig. 5.15 shows the rate distortion performance of the Photoshop baseline and progressive JPEG versions). However, the computational demand for encoding and decoding is of course higher.

This also serves as an excellent example how poorly documented the progressive JPEG modes are – there is no hint in the Photoshop documentation what type of progressive mode is employed. All subsequently used images are in 8bpp 512^2 pixels format.

As discussed before, the basic idea of selectively encrypting visual data in layered representation for providing confidentiality is to simply encrypt the base layer or the scans containing the perceptually most relevant information. In this case, the enhancement layers or remaining scans may be expected to contain data which is useless on its own although given in plaintext. Of course, this is not true in case of temporal scalability since the enhancement layer contains entire frames. As a consequence, temporal scalability can not be used for layered encryption.

Decoding a partially encrypted image by treating the encrypted data as being unencrypted leads to images severely degraded be noise type patterns (which originate form the encrypted parts, see Figs. 5.16.a and 5.17.a). Using these images to judge the security of the system leads to misinterpretations since a hostile attacker can do much better. In particular, an attacker could simply ignore the encrypted parts (which can be easily identified by statistical means) or replace them by typical non-noisy data. This kind of attack is called "error-concealment" [174] or "replacement attack" [112] in the literature.

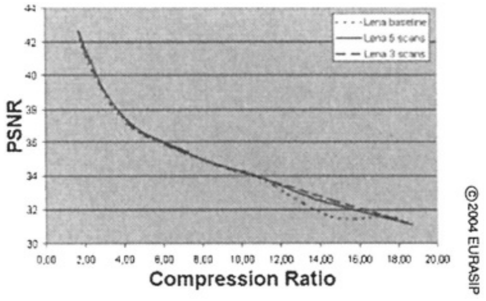
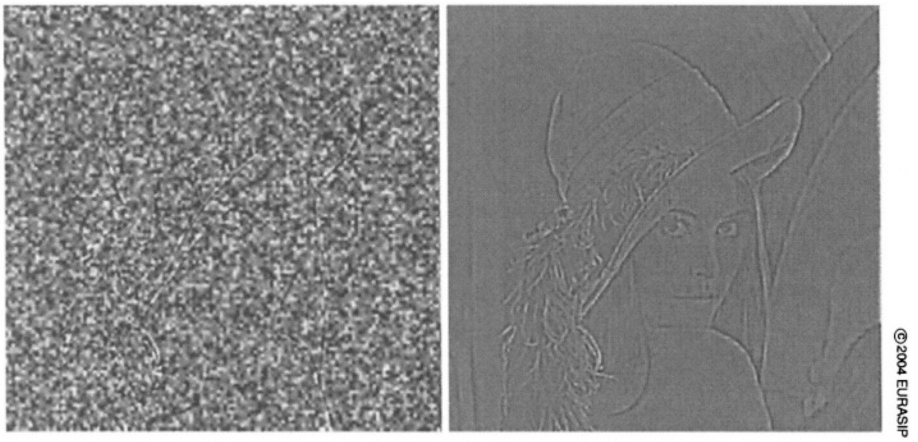


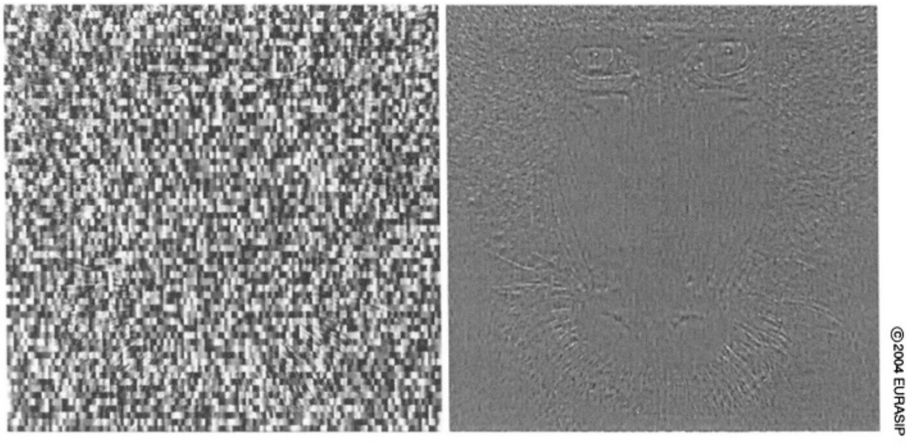
Figure 5.15. Compression performance (Lena image with 512^2 pixels and 8bpp) of Photoshops' baseline JPEG and progressive JPEG (with 3 and 5 scans).



(a) direct reconstruction (b) replacement attack

Figure 5.16. Lena image; a three level pyramid in HP mode is used with the lowest resolution encrypted

Figs. 5.16.b and 5.17.b clearly show that there can be still information left in the unencrypted parts of the data after selective encryption has been applied – in case of direct reconstruction this is hidden by the high frequency noise pattern. As a consequence, in order to facilitate a sound evaluation and comparison of the four modes to be considered, they are evaluated after a replacement attack has been mounted.



(a) direct reconstruction (b) replacement attack

Figure 5.17. Mandrill image; SS mode is used with DC and first AC coefficient encrypted

In order to be able to compare the different JPEG progressive modes for their suitability to follow the selective encryption approach, we set the amount of data to be encrypted to approximately 10 and 30%, respectively. Since we use a 10 bit representation for quantised DCT coefficients, the percentages can be exactly achieved in SA mode by encrypting the corresponding number of bitplanes. For HP, we get 31.25% of the original data encrypted by building a three level pyramid and encrypting the lowest resolution plus the first residual, and 8.3% by building a six level pyramid and encrypting the lowest resolution plus the three next residuals. SS facilitates protection of 29.7% and 9.3% of the data by encrypting 19 or 6 coefficients, respectively. Finally, we achieve encryption of 31.09% and 11.4% in the case of MM by scrambling the DC coefficients and one bitplane or three bitplanes, respectively.

The replacement attack is conducted as follows: for HP, the encrypted first scan is replaced by data originating from an equally sized image with constant gray value and eventually encrypted residuals are replaced by constant zero residuals. For SS an encrypted bitplane is replaced by a constant 0 bitplane, and for SA the encrypted coefficients are replaced by zeros.

Table 5.14 shows the PSNR values of the different techniques applied to the Lena and Mandrill image. Note that in contrast to a compression application, a method exhibiting low PSNR values is most desirable (since this implies low

	HP	SS	SA	MM
Lena, 10% enc.	14.8	14.6	7.0	6.8
Lena, 30% enc.	14.7	14.5	6.2	6.4
Mandrill, 10% enc.	17.5	16.8	7.5	7.3
Mandrill, 30% enc.	17.0	16.2	6.4	6.4

Table 5.14. **Objective quality (PSNR in dB) of reconstructed images**

image quality and therefore good resistance against the replacement attack). HP and SS show very similar results (at about 14.5 - 17.5 dB depending on the image and percentage of encryption) as well as do SA and MM at a much lower level (at 6.4 - 7.5 dB). However, it is interesting to note that there is not much numerical difference between the encryption of 10% and 30% of the data. As a consequence, we expect to perform SA and MM much better in terms of security as compared to HP and SS. In Fig. 5.18 we visually compare the reconstructed images underlying the numerical data of Table 5.14.

The numerical results are clearly confirmed by visual inspection. Whereas HP and SS clearly exhibit still remaining high frequency information (which are much clearer in the HP case), almost no information is visible for SA and MM where the images are dominated by noise. This noise comes from the fact that on average 50% of the coefficients (no matter if high or low frequency) have been altered at their MSB in the binary representation which results in those randomly looking images. Note that the replacement attack is not effective in the case of SA and MM since no matter if directly reconstructed or under the replacement attack always on average 50% of the coefficients are altered at their MSB position. Although the results of SA and MM look rather satisfying from a security point of view, there is still visual information related to the original image left. Fig. 5.19 shows that this remaining information may be enhanced using simple image processing operations which leads to the conclusion that obviously MM is the most secure variant of our investigated selective encryption schemes and is the only one that can be securely operated at a level of encrypting 10% of the data. The additionally encrypted DC coefficient makes MM much more resistant against reconstruction as compared to SA.

Increasing the amount of encrypted data up to 30% does not leave any perceptually relevant information in the remaining data in the case of SA and MM. Little information is left in case of SS applied to the Lena image, HP still reveals some edge and texture information. The Mandrill image contains much high frequency information which is still visible but not recognisable after encrypting 30% for both, the HP and SS modes.

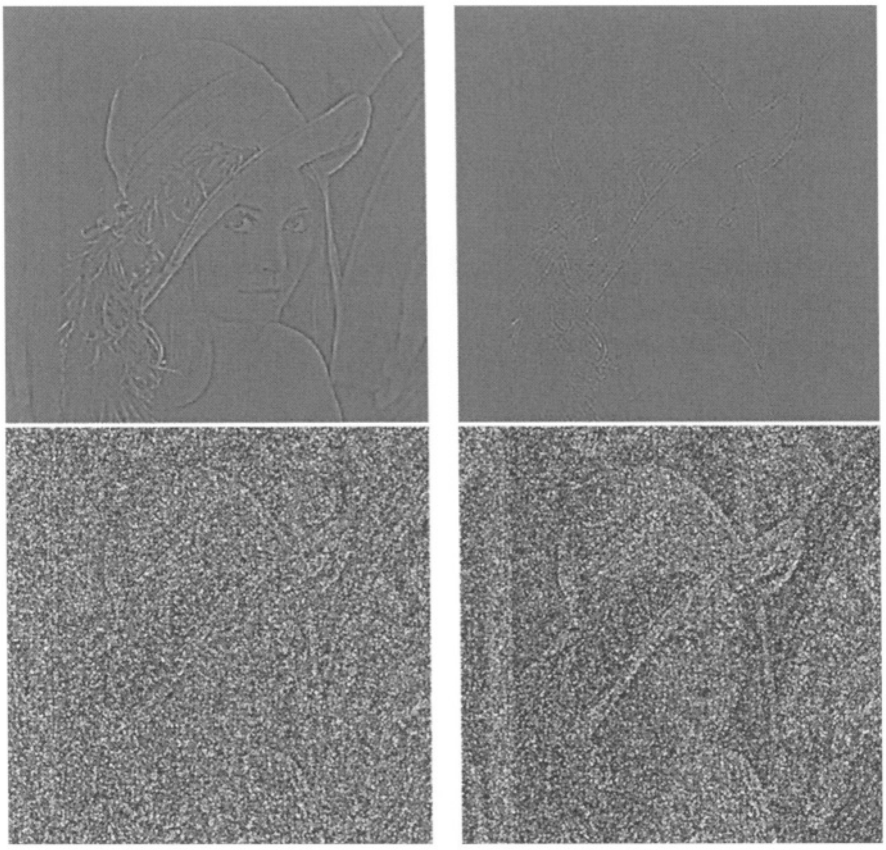


Figure 5.18. Subjective quality of reconstructed Lena image, 10% of the data encrypted (HP,SS,SA,MM, in clockwise direction starting at the upper left image).

We have seen that selective encryption using the hierarchical progressive and spectral selection JPEG modes still leaves perceptually relevant information in the remaining data after encrypting 30% of the original image data. Successive approximation and especially a hybrid variant which additionally protects the DC coefficient deliver much better results in terms of security. Relating these results to the MPEG case, SNR scalability will be most suited to apply selective encryption to scalable video data.

2. Wavelet-based Techniques

Wavelet-based techniques devoted to video encryption have not been discussed in literature so far, although all proposals made for image encryption may be applied to each frame of a video independently of course. The lack of a wavelet-based video coding standard explains this situation. As in the section

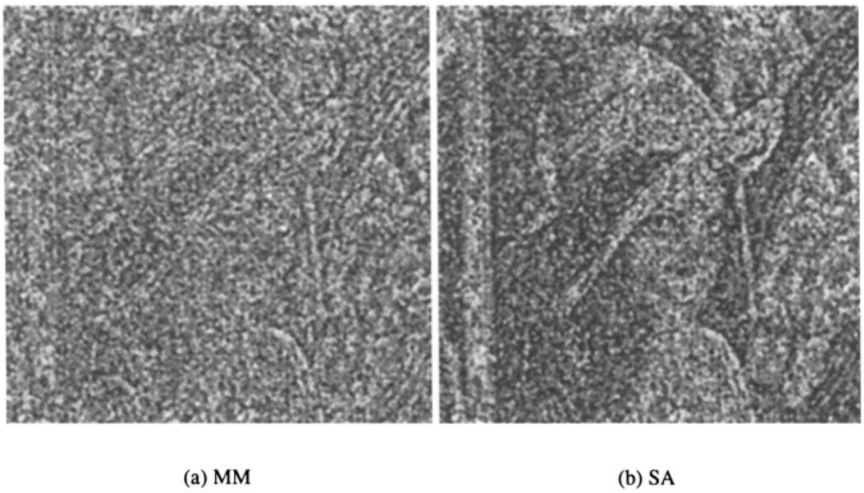


Figure 5.19. Images from Fig. 5.18 median filtered (3x3 kernel) and blurred (5x5 filter).

on DCT-based techniques, we distinguish between schemes operating during the compression stage ("compression oriented") and schemes applicable to a given bitstream ("bitstream oriented'). Note that most wavelet-based compression schemes use arithmetic coding as their entropy coding stage which does not provide a one-to-one correspondence among symbols and codewords like Huffman coding as used in DCT-based systems does. Therefore, techniques manipulating single coefficients cannot be employed in the transform domain.

2.1 Compression Oriented Schemes

2.1.1 Coefficient Selective Bit Encryption

Similar to their proposals for DCT-based systems, Zeng and Lei [185, 186] suggest to encrypt selected parts of the transform coefficients' binary representation. They compare refinement, significance, and sign bits with respect to their entropy and compressibility. Based on this analysis, it is suggested to encrypt bits that are not highly compressible due to their high entropy and low intercorrelation. The corresponding selection limits the influence of the encryption process to rate-distortion performance: sign bits and refinement bits. Of course, refinement bit encryption can be used only as an additional security technique since it does not provide enough confidentiality as a standalone approach. Similar doubts with respect to security of sign bit encryption are valid as in the DCT case. Corresponding experimental results are provided by Zeng and Lei who propose to combine sign bit encryption in combination with other techniques like block permutation or block rotation (see section 2.1.3 (chapter 5)).

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
medium	medium	low	yes	no	no

Table 5.15. Overall assessment of Coefficient Selective Bit Encryption

2.1.2 Coefficient Permutation

In the context of DCT-based compression systems, coefficient permutation has been proposed as a means to provide confidentiality within the compression pipeline (compare the "Zig-zag Permutation Algorithm" in section 1.1.1 (chapter 5)). In the context of wavelet-based compression schemes, random permutation lists have been proposed by Uehara et al. as well to secure wavelet-subbands [159]. One obvious advantage as compared to the DCT scenario is that the distribution of wavelet coefficients is image dependent and therefore the vulnerability against ciphertext only attacks does not occur. Also, it is claimed [159] that contrasting to the DCT case the observed drop in compression performance is about 2% only.

In this section we use random permutation lists to secure wavelet-coded visual data (compare also [105]). We show that a system based on randomly permuting wavelet-subbands incorporated in the JPEG 2000 or the SPIHT coder generally delivers much worse results in terms of compression performance as given in [159]. The comparison of JPEG 2000 and SPIHT in this context provides interesting insights with respect to the correctness of the zerotree hypothesis.

Encryption Using Random Permutation of Wavelet-Subbands. The basic approach is to permute the wavelet coefficients of different wavelet subbands with dedicated permutation keys. A permutation key is defined as a vector of length n, and n wavelet coefficients can be encrypted using this key. We use an algorithm according to Knuth's "Seminumerical Algorithms" to compute uniformly distributed permutation keys.

In case permutation keys have to be transmitted along with the compressed image data (and not generated on the fly as proposed in [174]) the used keys have to be protected and therefore be encrypted with a standard encryption scheme like AES. For example, the key data itself can be inserted conveniently into the JPEG 2000 bitstream taking advantage of the so-called termination markers.

Encryption based on random permutation lists has been shown to be vulnerable to known plaintext attacks. The use of more different keys increases the overall security of the system. This raises the question how many keys should be used to encrypt the data and what key lengths should be used in order to achieve a satisfying level of security. Additionally it needs to be considered that any key information needs to be stored in the final bitstream and decreases the compression performance. We discuss two key management scenarios:

1 full key scenario: A wavelet subband with n pixels is permuted with a "full" key of length n.

2 key for row scenario: A wavelet subband consisting of n pixels $(n = r * c, r = rows, c = columns)$ is permuted with keys on a per row basis. Therefore, a number $x, 1 <= x <= r$ of keys with length equal to one row c is used, and the keys are exchanged in a round robin fashion.

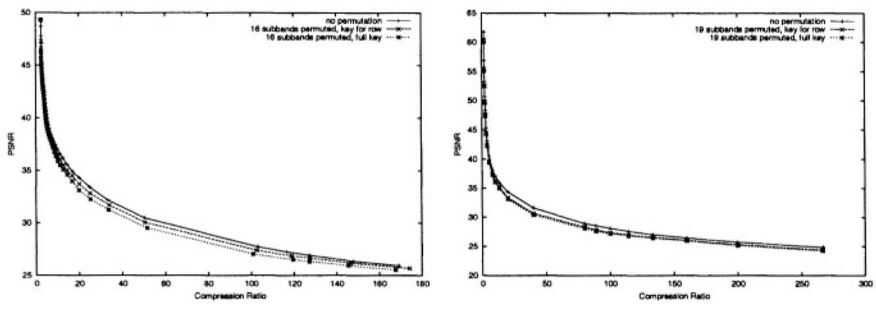
Instead of using randomly chosen permutation keys which need a significant amount of additional storage capacity, a master key together with a key generation algorithm as proposed in [159] can be used to save memory. The usage of a master key with a key generation algorithm can be somewhat weaker in terms of security as compared to using randomly selected keys, however, it turns out that this approach is mandatory to limit the loss in compression efficiency.

Experimental Results. We use the two considered lossy image compression schemes with the default decomposition depth. The testimages are the Lena, lunge, plane, and the graves image each at a resolution of 512 × 512 pixels.

Regarding the "key for row" key management scenario we discuss the worst case in terms of security, where the same permutation key is used for each row of a wavelet subband.

In order to evaluate the compression performance, each testimage is encoded with both considered coding algorithms. Within the coding pipeline, the coefficients of the wavelet subbands are permuted before the quantisation stage. Thereafter, the encrypted and compressed file is decoded, the corresponding wavelet-subbands are inverse-permuted, and the overall rate-distortion performance is computed.

The rate-distortion performance for the lena image is shown in figure 5.20. Note that key material is not included in the bitstream for this comparison. The "no permutation" curve denotes the rate-distortion performance of the original JPEG 2000 and SPIHT algorithm. The curve "key for row" shows the performance when all subbands are permuted and per subband only one key is



(a) JPEG 2000 (b) SPIHT

Figure 5.20. Compression performance, Lena image 512 x 512 pixels

repeatedly used for the rows of the subband, and the curve "full key" shows the rate-distortion performance, when each subband gets a full key for the permutation of the whole subband.

In the case of JPEG 2000 a max. drop of compression performance of 26% using full keys can be observed when coding the lena image (meaning that the encrypted file needs to be increased by 26% in order to achieve the PSNR of the original encoded image; we denote this to be a loss of compression performance of 26%), and only a 13% decrease can be observed when using one key for each row of a subband. Table 5.16 lists the observed maximal drops of compression performance for all tested images. Overall, the "key for row" method degrades compression performance much less, and the zerotree-based

	full key	key for row
JPEG 2000, lena512	26%	13%
JPEG 2000, lunge512	8%	4%
JPEG 2000, plane512	27%	14%
JPEG 2000, graves512	21%	5%
SPIHT, lena512	26%	21%
SPIHT, lunge512	9%	9%
SPIHT, plane512	23%	21%
SPIHT, graves512	22%	15%

Table 5.16. **JPEG 2000/SPIHT: all subbands permuted, max. observed file size increase at a medium compression rate ranging from 25 up to 45**

SPIHT algorithm produces similar results in the "full key" scenario as compared to JPEG 2000. The latter is surprising, since permuting the coefficients of wavelet subbands obviously destroys the zerotree structures which should be essential to the performance of such an algorithm. Considering the zerotree hypothesis, a stronger degradation would have been expected in the SPIHT case which raises doubts about the correctness of this important assumption. However, in the "key for row" scenario the JPEG 2000 compression performance is much less decreased as compared to SPIHT. This is due to the spatial correlation which is preserved among pixels in neighbouring areas (since the same permutation is used for adjacent rows), which allows the context-based arithmetic coding engine of JPEG 2000 to produce better results as compared to the inter-subband zerotree coding of the SPIHT codec.

Compared to the good results shown in [159] both considered schemes produce a significant performance loss (up to 27%), and we do not even include the key data in the final compressed file yet. The compression scheme in the referenced work is not a very sophisticated one (first generation wavelet coding scheme) and SPIHT as well as JPEG 2000 depend much more on pixel context as simple scalar quantisation based schemes.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	medium	yes	no	yes

Table 5.17. Overall assessment of Coefficient Permutation

2.1.3 Coefficient Block Permutation and Rotation

Zeng and Lei [185, 186] also propose a generalisation of the coefficient permutation approach. They suggest to divide each subband into a number of blocks of the same size. The size of these blocks can vary from subband to subband. Within each subband, blocks of coefficients are permuted according to a permutation key which should also differ from one subband to another. Since the local statistics of the wavelet coefficients are preserved, the expected impact on coding performance is smaller as compared to the pure coefficient permutation case (the degradation is the smaller, the larger the block size is selected). On the other hand, using large blocks threatens security due to two reasons:

- The permutation key is small.
- A possible attacker might try exploit edge continuity in the high pass subbands and a smoothness constraint (similar to the attacks against line permutation in the spatial domain, see section 3.1.1 (chapter 5)) in the low pass subband to invert the permutation.

To further increase security without affecting rate-distortion performance it is suggested to additionally use one of eight isometries of each block (which corresponds to rotating and flipping the block). This makes it harder to invert the permutations based on image properties, however, a higher number of rotated versions would be necessary to provide sufficient security.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	low	yes	no	moderately

Table 5.18. Overall assessment of Coefficient Block Permutation and Rotation

2.1.4 Secret Transform Domains

Similar to the idea of using a secret Fourier transform domain (compare section 1.1.1 (chapter 5)) it is also possible to use secret wavelet transforms for an encryption application. The idea of using secret wavelet domains has also been used to increase the security of watermarking schemes. In this context, filter parameterisations [34] and wavelet packet subband structures [33] have been used to conceal the embedding domain.

Contrasting to the Fourier case, all proposals concealing the wavelet transform domain for encryption are integrated into a compression pipeline. Vorwerk et al. [166] propose to encrypt the filter choice used for wavelet decomposition, however, this suggestion remains vague and is not supported by any experiments. In the following we discuss two ways of generating a large variety of wavelet filters out of which a secret one may be chosen for actual decomposition. All these techniques have a significant advantage: the amount of data subject to encryption is minimal since only information about the transform in use needs to be encrypted. Therefore, these methods may be seen as a special variant of header encryption. However, two questions remain unanswered so far:

1 Since a vast share of the data remains unencrypted, is it possible to reconstruct the visual data (or at least a good approximation to it) using the unencrypted material ?

2 Filter choice is important with respect to image quality in wavelet compression schemes. Can the compression quality be maintained when using any of these approaches?

Secret Wavelet Filters: Codebook Approach. Generalised wavelet decompositions (where different filters are used at different decomposition levels) may be employed and the structure of these decompositions may be kept as

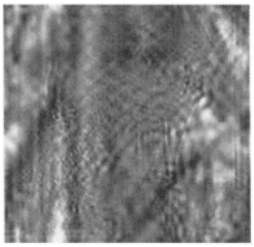


Figure 5.21. Reconstruction using random filters

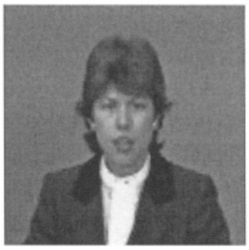


Figure 5.22. Reconstructed image where the heuristic failed at the finest level of decomposition

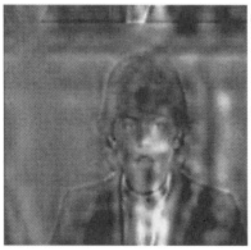


Figure 5.23. Reconstructed image where the heuristic failed at 3 out of 5 levels

key: here non-stationary multiresolution analysis NSMRA [31, 88, 160] or subband variant decompositions (which use the same idea applied for wavelet packet decompositions at the subband level) [161] are possible candidates. We use a library consisting of predefined filters. The size of the keyspace then depends on the size of the filter library (l), the decomposition depth (k), and the type of generalised decomposition (i.e. l^k keys for NSMRA and 4^{l^k} keys for subband variant decomposition).

In the current implementation we chose the NSMRA approach [113], the index of the filter in the library can be chosen by different algorithms.

Figure 5.21 shows the case when random (i.e. wrong) filters are used to reconstruct the image.

Concerning a possible attack against the scheme, we assume the attacker has all knowledge except the indices of the actual filters in use. A brute-force attack is not feasible but by using the following heuristics we get a sufficient result: when the correct filter is applied in the reconstruction process, a rather smooth image results, but when the filter is incorrect, artifacts appear and the resulting image is quite noisy. We measure the difference between neighbouring pixels (both vertical and horizontal) and then we calculate the entropy of these differences. If the filter is correct then the entropy will be low. This heuristic reduces the attack complexity from l^k for brute force to $l \times k$. The heuristics work in most cases, figures 5.22 and 5.23 show reconstructed images where the heuristics fail. These images show that even an partially incorrectly guessing heuristic is better than a pure random attack.

The described attack shows that this scheme is only secure enough for a low-security entertainment application in case the codebook is not very large. The number of different wavelet filters discussed in literature is too small to provide codebooks with sufficient size. In the next section, we will discuss techniques to generate entire families of wavelet filters for that purpose.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	low	yes	no	moderately

Table 5.19. Overall assessment of Secret Wavelet Filters: Codebook Approach

Secret Wavelet Filters: Parametrisation Approach. For the construction of compactly supported orthonormal wavelets, solutions for the dilation equation have to be derived, satisfying two conditions on the coefficients c_k ($\phi(t) = \sum_{k \in \mathbb{Z}} c_k \phi(2t - k)$, with $c_k \in \mathbb{R}$). Here we use parameterised filters generated according to an algorithm proposed by Schneid and Pittner [130]:

Given N parameter values $-\pi \leq \alpha_i < \pi$, $0 \leq i < N$, the following recursion

$$\begin{aligned} c_0^0 &= \tfrac{1}{\sqrt{2}} \text{ and } c_1^0 = \tfrac{1}{\sqrt{2}} \\ c_k^n &= \tfrac{1}{2}\Big((c_{k-2}^{n-1} + c_k^{n-1}) \cdot (1 + \cos \alpha_{n-1}) + \\ &\quad (c_{2(n+1)-k-1}^{n-1} - c_{2(n+1)-k-3}^{n-1})(-1)^k \sin \alpha_{n-1}\Big) \end{aligned}$$

can be used to determine the filter coefficients c_k^N, $0 \leq k < 2N + 2$. We set $c_k = 0$ for $k < 0$ and $k \geq 2N + 2$. Example filters which can be generated using this formula are the Daubechies-6 filter, which can be constructed using the parameters (0.6830127, –0.1830127), or the Haar filter which is generated with the parameter 0.

The number N of parameter values α_i controls the length of the resulting wavelet filter, i.e. $2N + 2$.

Note that similar parameterisations are available for biorthogonal filterbanks [59] and for the lifting scheme in the context of JPEG 2000 [134].

In this section we investigate the properties of a header encryption variant where we keep the parameter to generate the filters for the wavelet transform secret (compare also [73]). This can be easily achieved in the context of JPEG 2000 Part II by simply encrypting the corresponding field containing the custom filters in the header using a cryptographically strong cipher. As a consequence, the amount of data subject to encryption is minimal, since no actual image data but only filter coefficients are protected.

In the following, we investigate the compression quality and the security of the resulting scheme, experiments are performed using the JPEG 2000 Jasper C reference implementation with different decomposition filters.

Whereas the traditional filters used for wavelet compression are tuned for optimal concentration of the energy contained in the image and the separation of high- and low-frequency parts, parameterised filters provide a wide quality range. The advantage as well as the disadvantage of parameterised filters is their variety, not all filters within such a family are equally suited for a specific purpose, in this case, image compression. Fig. 5.24 shows the resulting quality (PSNE. in dB) when compressing the 8 bpp 512 × 512 pixels Lena image using different parameter values with compression ratios 10 and 20, respectively.

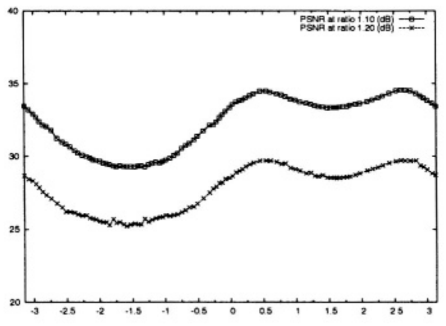


Figure 5.24. Quality of JPEG 2000 compression, using a 1-dimensional parameter space

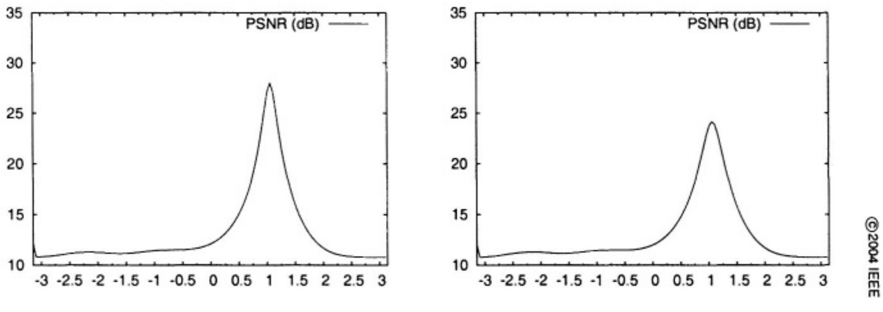
It is clearly displayed that the compression quality of the filters resulting from the parameterisation algorithm varies in the interval [29.5*dB*, 35*dB*] for ratio 10 and [25*dB*, 30*dB*] for ratio 20, respectively. Among other variations, obviously the left half of the parameter range leads to poor filter quality. As a consequence of these findings, a strategy is required to limit the possible compression quality loss introduced by randomly chosen parameters. The most desirable approach would be a heuristic which – given either the parameters to generate the filters or the actual filter coefficients themselves – could determine an approximation of the compression quality to be expected in advance (i.e. without performing the compression). A heuristic of this type would allow a parameter generation and evaluation on the fly, i.e. during the compression stage without significant increase of computational demand. Besides restricting the parameter to positive values no such heuristic could be found.

To avoid low-quality filters, two other approaches might be possibly used:

- Generate the parameters and the corresponding filter coefficients and perform the compression stage. The parameter is used only in case the quality turns out to be sufficient. As this technique is time consuming, it contradicts our goals we want to achieve with the entire system. Only one failure in parameter choice (i.e. one bad quality filter) makes the scheme significantly more expensive than full AES encryption of a JPEG 2000 Part I bitstream.

- Determine parameter values of good quality in advance and restrict the admissible parameters to regions close to that values. Fortunately, the quality of parameters is very much image independent, which makes this approach a feasible and efficient one. However, the decrease of the amount of admissible parameter values is known in advance (also to a potential attacker) and needs to be considered. This fact reduces the overall security of the system since it corresponds to a smaller keyspace.

In the following we focus on the security of the system. The restriction in terms of filter quality values reduces the amount of admissible parameter values as seen before to 20 – 50 % of the entire range, depending on the quality requirements of the target application. At first sight, there seem to be enough parameter values left since the data type of the parameters for this kind of filter is $\mathbb{R}$ (in theory), in practice it is $\mathbb{Q}$. However, close parameters lead to similar filters which in turn lead to similar wavelet transform coefficients. Of course, this might be a threat to the security of the system since an attacker does not need to know the compression parameter exactly to get a "decrypted" image with sufficient quality. In Fig. 5.25 we illustrate this problem. The Lena image is compressed with filters generated by the parameter 1.05841, and subsequently decompressed with a large number of different filters derived from parameters covering the entire possible range. We plot the PSNR of the resulting images against the parameter used for decompression. The desired result would be an isolated single quality peak at the position of the "correct" parameter (that one used for compression) and low values everywhere else.

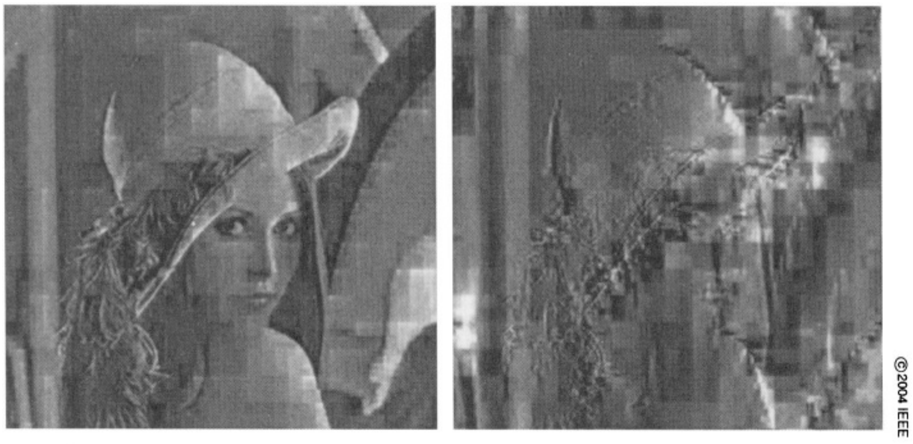


(a) 0.8 bpp (ratio 1:10)

(b) 0.4 bpp (ratio 1:20)

Figure 5.25. Attack against a 1-D parameter scheme, JPEG 2000 compression at ratios 1:10 and 1:20

The result of this experiment is not an isolated PSNR peak but an entire region centred around the correct parameter where the PSNR values are decreasing with increasing distance from the correct value. For example, the parameter range [0.75, 1.25] (which covers about 8 % of the entire parameter range) provides image quality above 20 dB. Fig. 5.26.a visualises an image decompressed with a parameter displaced from the correct one by a distance of ≈ 0.2 in terms of parameter value. Obviously, the quality of this (attacked) image is too high to provide any kind of confidentiality.



(a) 21.86 dB, parameter 1.20841 (b) 10.78 dB, parameter 2.85841

Figure 5.26. Quality of attacked images, JPEG 2000 compression at 0.4 bpp

As a consequence, the number of admissible parameter values needs to be restricted to a rather sparse grid. Taken this fact together with the beforementioned restrictions due to low quality filters the keyspace is too small for a reasonable application in case of the 1-D parameter scheme. However, when taking these restrictions into account the quality of encrypted and attacked images is low enough for applications where the size of the keyspace is not an issue (see Fig. 5.26.b).

In order to increase the available keyspace parameterisations with more parameters (leading to longer filters) can be used. This increases the number of high quality filters significantly if the percentage of good filters remains approximately constant in the entire set of filters, which turns out to be true. To compare the 1-D parameterisation to the 2-D case, the Lena image is compressed with the filters generated by the parameters -1.69159 and -1.84159, and subsequently decompressed with a large number of different filters derived from parameters covering the entire possible range. In Fig. 5.27 we again plot the PSNR of the resulting images against the parameter used for decompression.

The result shows that still a sparse grid needs to be applied to this much larger parameter space. In the 2-D parameter scheme we do not result in the single isolated PSNR peak as well but we still face an entire region where the quality of the encrypted and attacked images is too high. Therefore, the resulting number of admissible parameters still remains rather small in the 2-D case, but the strategy to move to higher dimensional parameterisation schemes turns

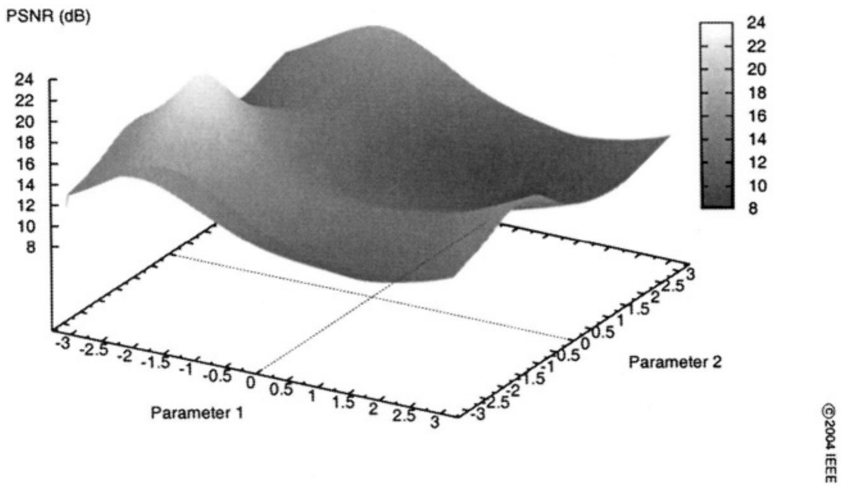
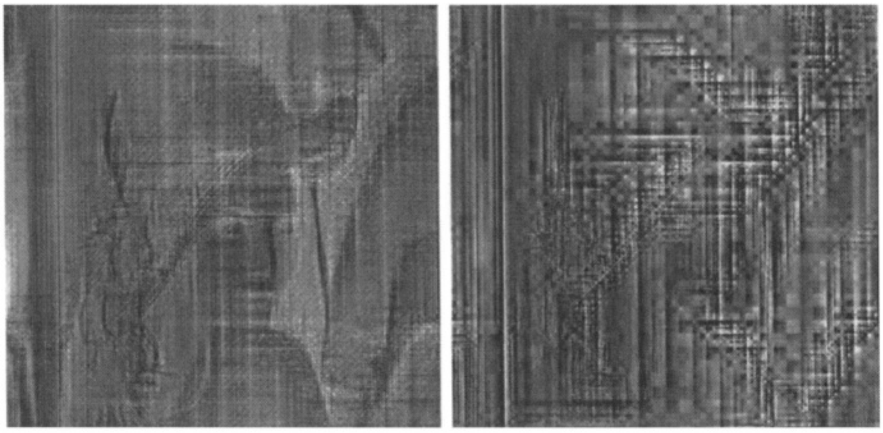


Figure 5.27. Attack against a 2-D parameter scheme, JPEG 2000 compression at ratio 1:10

out to be fruitful in principle and leads to reasonable keyspace sizes at least for low security applications. Fig. 5.28 shows two encrypted and attacked images using the same 2 parameters for compression as given before. The result suggests that a sufficient degree of confidentiality may be achieved with the proposed scheme, provided the limitations as discussed before are addressed properly.

Theoretical and practical work in the field of image and video compression usually prefers biorthogonal filters. In most compression applications the well-known "Biorthogonal 7,9" filter is used. Therefore our hope was that this filter is not an exception but an indication about the superiority of the overall class of biorthogonal filters in this context [164].

For the creation of biorthogonal filters we relied mainly on a paper by Hartenstein [59] because a method was presented which allowed an "easy" implementation: no symbolic computations with programs like *Mathematica* or *Mathlab* were required. Such a prerequisite would have negated the requirement that many tests with different parameters should be performed, and that the program should be able to perform a very quick filter exchange. Additionally, this application should fit into the context of the C++ based compression library and framework developed at our department (called `libganesh++`) which includes the SMAWZ-codec used in the following experiments (compare also [118]). Besides Hartenstein some other authors have proposed addi-


(a) 9.86 dB, parameters 1.00841 and 1.60841

(b) 11.76 dB, parameters -3.04159 and 0.70841

Figure 5.28. Quality of attacked images, JPEG 2000 compression at 0.4 bpp

tional methods for parameterising biorthogonal wavelet filters [122, 108, 111, 93, 92, 69, 64, 107].

Even length filters require that the difference between high- and low-pass filter is a multiple of 4, i.e. $4K$. The general formula to generate these filters is

$$\begin{bmatrix} H(z) \\ G(z) \end{bmatrix} = H_p(z^2)\begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} \tag{5.1}$$

$$\text{with } H_p(z) = A\Lambda(z)S_{L-1}\Lambda(z)\ldots\Lambda(z)S_0 \tag{5.2}$$

$$\text{and } \Lambda(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \tag{5.3}$$

$$\text{and } S_i = \frac{1}{\cos^2\theta_i - \sin^2\theta_i}\begin{bmatrix} \cos\theta_i & \sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}. \tag{5.4}$$

It is obvious that the denominator above must not be 0, therefore the θ_i are limited to $\theta_i \neq (2k+1)\frac{\pi}{4}, k \in \mathbb{Z}$. Hartenstein had two errors in his paper, one in his equation (2) where he had an excess matrix S_L, and the other one in his equation (3) where the restriction for θ_i was too strict. Additionally, he didn't care about the energy-preserving property of his matrices: the value of the determinant must be 1. So Hartenstein didn't give the fraction part for the matrices S_i. The above limitation for θ_i has to be extended in practise, so that it can be formulated like "θ_i should not lie within a neighbourhood of ϵ, centred at odd multiples of π". The result is undefined right at these multiples, but within the neighbourhoods numerical instabilities occur which make it difficult to calculate reasonable results. We discovered that ϵ must be increased for increasing absolute values of K.

We can distinguish between three cases, for each one a different Matrix A must be constructed:

K = 0: this is the simplest case

$$A = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{5.5}$$

K > 0: the high-pass filter is longer than the low-pass filter

$$A = \begin{bmatrix} 1 & 1 \\ P_{0K}(z) & P_{1K}(z) \end{bmatrix} \tag{5.6}$$

(this was equation (5) in the Hartenstein paper)

$$P_{0K} = \begin{cases} 1 & K = 0 \\ 1 + \tan\beta_1 z^{-1} - z^{-2} & K = 1 \\ P_{01}(z^k) + \sum_{j=2}^{K} z^{j-K-1} q_j(z) & K > 1 \end{cases}$$

$$P_{1K} = \begin{cases} -1 & K = 0 \\ 1 - \tan\beta_1 z^{-1} - z^{-2} & K = 1 \\ P_{11}(z^k) + \sum_{j=2}^{K} z^{j-K-1} q_j(z) & K > 1 \end{cases}$$

with $q_j(z) = \tan\beta_j - \tan\beta_j z^{2(1-j)}$.

K < 0: the low-pass filter is longer, the formula is almost identical (and we set $K = -K$ to be positive):

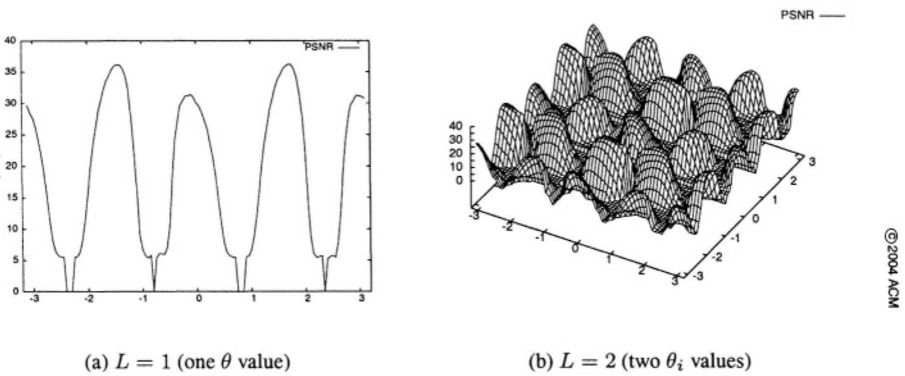
$$A = \begin{bmatrix} Q_{0K}(z) & Q_{1K}(z) \\ 1 & -1 \end{bmatrix} \tag{5.7}$$

with $Q_{0K}(z) = P_{0K}(z)$ and $Q_{1K}(z) = -P_{1K}(z)$

Of course the matrix A must be normalised again, otherwise the subsequently generated filter will not preserve the energy of the signals.

In the following, we investigate the quality of compression conducted by even-length filters.

The quality obtained by compression using even-length parameterised biorthogonal filters according to the construction method of Hartenstein varies by a very large amount, even more as compared to the orthogonal case. As



(a) $L = 1$ (one θ value) (b) $L = 2$ (two θ_i values)

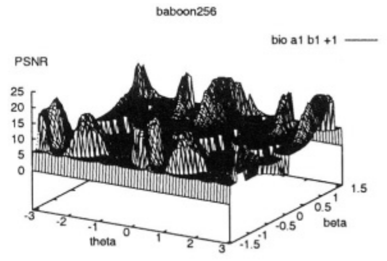
Figure 5.29. Quality values for $K = 0$.

can be seen on figures 5.29(a) and 5.29(b) the maximum value is near 35dB, but values go down to 5 dB as well. Note that at the instances where the filter could not be generated because of numerical instabilities a quality value of 0 was assumed. Some of the figures shown in this paper were generated from the results with tests with the Lena image, some with the baboon image — there are no significant differences between these two result sets. In all experiments the images were compressed with a target bitrate of 80000 bits, this leads to a compression rate of about 6.5 for 8-bit gray-level images with 256*256 pixels.

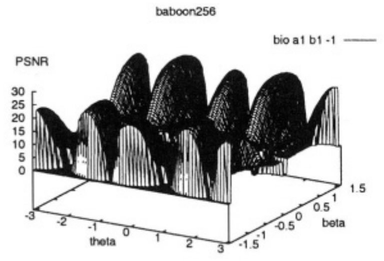
First we look at the most simple case where both filters have the same length ($K = 0$). We examine the results with $L = 1$ and $L = 2$. Figure 5.29(a) shows the first case, we observe a very high variance of the PSNR values. Figure 5.29(b) shows the results obtained by setting $L = 2$, the results look very similar to the previous figure. One can also observe some regular pattern with high-quality areas which could be used for later encryption tests.

When we compare the figures 5.30.a and 5.30.b we see a difference in the maximum PSNR of about 10dB: 24.7 versus 34.6 on the other hand. This shows that it is important to make the right choice between a long high-pass filter together with a short low-pass filter or the short high-pass with the long low-pass filter: the variant with the shorter high-pass filter for decomposition is the better choice. Figure 5.31 shows the frequency response for both filters.

Another interesting point is the comparison of the aforementioned filters with the well-known 7/9 filter: the PSNR in the same experiment lies at 37.7dB. Figure 5.31(c) shows the frequency response of the 7/9 filter. In comparison to figures 5.31(a) and 5.31(b) this looks much better: higher degree of symmetry, and the frequency separation into two bands is much higher. On the other hand the symmetry is lower when we compare it with the frequency response

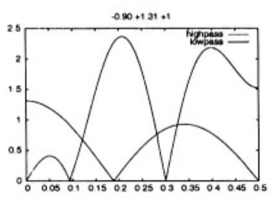


(a) $K = +1, L = 1$

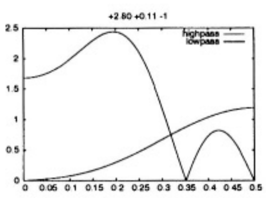


(b) $K = -1, L = 1$

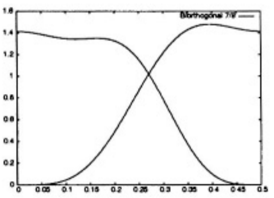
Figure 5.30. Parameterised biorthogonal 4/8 filters.



(a) Frequency response of the filter giving the best result in figure 5.30.a.



(b) Frequency response of the filter giving the best result in figure 5.30.b.



(c) Frequency response of the Biorthogonal 7/9 filter.

Figure 5.31. Frequency response

of orthogonal parameterised filters [118], so this attribute can be a hint towards high-quality filters. But it is widely known that there are other parameters important for compression performance, too.

So we see that when one wants to implement parameterised biorthogonal filters for selective encryption one will be faced with a decreased quality when using the same compression rate, at least when applying even-length filters derived from Hartensteins parameterisation.

We see that even length biorthogonal wavelet filters derived from a parameterisation proposed by Hartenstein have turned out to give extremely varying (and also generally poor) compression results thus making them inappropriate for a selective encryption approach which only protects the filters in use during wavelet decomposition and compression.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	low	yes	no	moderately

Table 5.20. **Overall assessment of Secret Wavelet Filters: Parametrisation Approach**

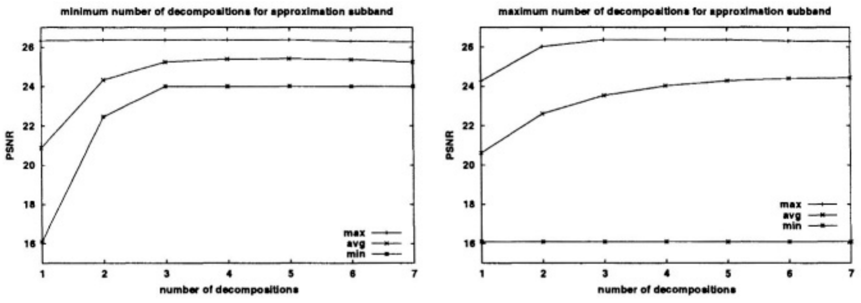
Secret Subband Structures. In this section we propose to use wavelet packet based compression instead of pyramidal compression schemes in order to provide confidentiality (compare also [113, 115–117]). Header information of a wavelet packet (WP) image coding scheme based either on a uniform quantiser or on zerotrees is protected, in particular we use AES to encrypt the subband structure used by the encoder only. In our approach the encoder uses different decomposition schemes with respect to the wavelet packet subband structure for each image (in fact, the subband tree is chosen randomly or determined by some pseudo-random algorithm). These decomposition trees are encrypted and have to be present at the decoder to be able to reconstruct the image data properly.

In our WP based selective encryption approach we do not use a classical best basis selection or a similar method to determine a useful wavelet packet basis, but we use a more-or-less random decomposition tree. This tree can be generated completely random, or using a PRNG (pseudo random number generator) algorithm to decide the decompositions, it is also possible to use a best-basis algorithm as a first step and make random or pseudo-random alterations to it. Using a decomposition tree generated by the best-basis algorithm without further alterations is not reasonable since such trees share common features for many images which would consequently facilitate an attack.

We use the PRNG approach in this work. In order to decide if a certain subband should be decomposed further, first a number is obtained from the PRNG which generates equally distributed float numbers in the interval [0,2[. A weight is computed at every decomposition level: weight at level $i =$ **base value** $+ i \cdot$ **change factor.**

Then the PRNG number is divided by the weight. If the result is smaller than 1, no further decomposition is computed. The reason for introducing the weight is that in case of using PRNG numbers only "shallow" decompositions are more probable than many "deep" decomposition trees. The default values for "base value" and "change factor" are 1 and 0, respectively. The probability that decomposition level i is reached for a given subband is $(\frac{1}{2})^i$ when the default values are used. This means that the probability that the decomposition tree has a depth of exactly one level is $\frac{1}{2}$.

The subband tree carrying the subband structure information of the data is secured (by using AES encryption) for transmission. The amount of data is



(a) approximation subband minimum decomposition level

(b) approximation subband maximum decomposition level

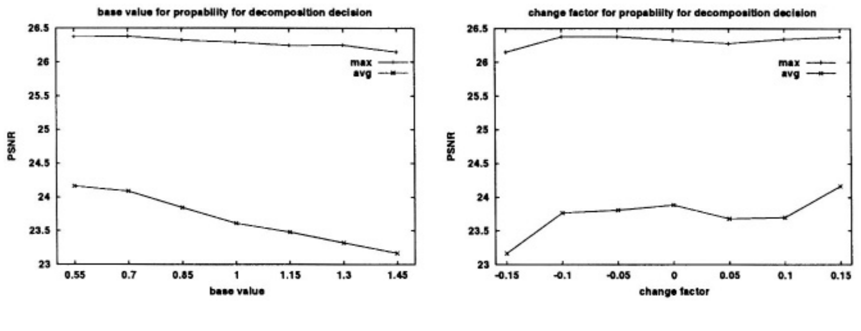
Figure 5.32. minimum and maximum level of decomposition influencing the quality

very low compared to e.g. an approach which encrypts the tree structure of spatially quadtree decomposed images [22] where up to 50% of the overall data are encrypted. In our approach just the PRNG seed value and the two weight factors have to be secured, otherwise the hierarchy information has to be encrypted, 1 bit per subband, which leads to a typical amount of 50 to 200 bits since the decomposition depth has to be limited.

In contrast to classical WP coding schemes we do not employ subband structures specifically tailored for compression but random ones. Therefore, we need to examine the typical rate-distortion performance of the entire system, i.e. how much quality we possibly lose by using this approach. From these results we derive parameter settings to limit this potential quality loss.

We performed our tests with the image “Lena” and cross-checked the results with a subset of the tests using the image “Barbara”, again using the SMAWZ codec operating at 80000 bits. In our tests we varied 6 parameters, as partially shown in figures 5.32 and 5.33. All figures show two or three lines, the maximum PSNR value achieved with a certain parameter combination, the average value and (sometimes) the minimum. Values near 16dB correspond to decompositions where just one decomposition step was performed for the approximation subband (see figure 5.32(a)).

The parameter which has the most influence on the compression quality is the parameter determining the minimum number of decompositions of the approximation subband (see figure 5.32(a)). Setting the minimum number to 2 or lower the probability is high that a decomposition results which gives a bad compression result. Setting the value too high (e.g. 6) can give slightly decreased results as well. In order to limit quality loss, we propose a value of 4 or 5 for this parameter.



(a) base value (b) base scale change factor

Figure 5.33. Various weight factors for the decomposition decision influence on the quality

The parameter which comes next in terms of determining compression quality is the number of maximum decompositions allowed for the approximation subband as shown in figure 5.32(b). We suggest a value of at least 5 but preferably higher since a high value for this parameter significantly increases the number of admissible subband trees which is important for the security of the system. The latter two parameters are intercorrelated since **min** $\leq$ **max**.

The degree of influence of the remaining parameters is low as long as they are set to reasonable values. In particular, the maximum number of decompositions allowed for any subband does not seem to have any impact on compression performance for values ≤ 7. Therefore, this parameter should also be set to the highest reasonable value (i.e. 7) to guarantee a potentially high number of subband trees.

In the following, the two factors determining the weight are investigated in more detail. The "base value" gives the initial probability determining the decomposition decision. At a value of 1 the two possibilities are equal, when the number is lower than 1.0 the decision favours further decompositions. As can be seen in figure 5.33(a) in this case the quality is slightly better. We suggest to set this value to 1.0 or below. The "change factor" changes the weight in a decomposition depth dependent way. If the change factor is 0, the "base value" stays the same on all decomposition levels. Otherwise it is added to the "base value" at every level of decomposition thereby increasing the weight at each decomposition level.

Based on intercorrelation results between figures 5.33(a) and 5.33(b) (not shown), we suggest that the "change factor" value is set to 0.0 (in the case when the above "base scale factor" was set to 1.0) or 0.15 (if the "base value" was set to a value smaller than 1.0).

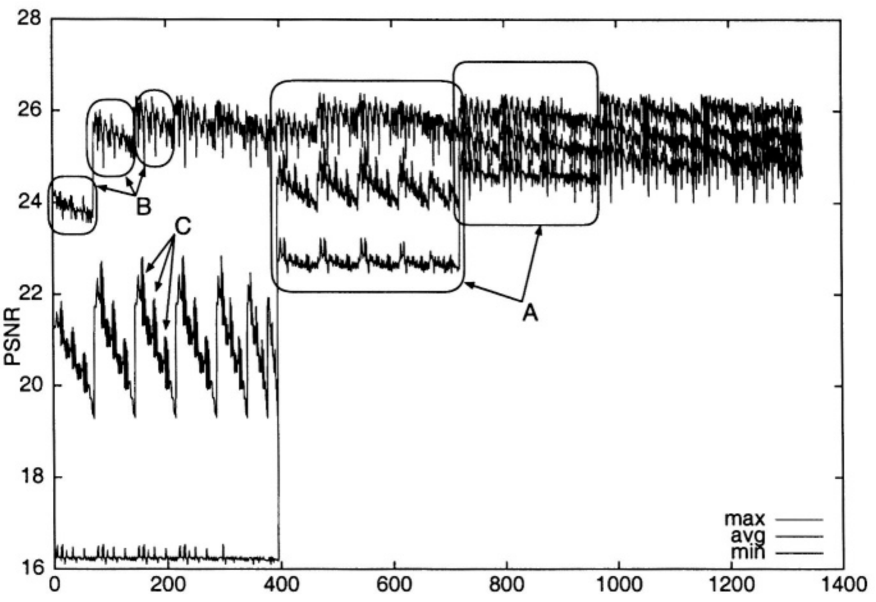


Figure 5.34. All parameters of figures 5.32(a), 5.32(b), 5.33(a), 5.33(b) in one plot

Fig. 5.34 gives an overall impression; it shows a combination of the previous parameters, where every value on the x-axis denotes a fixed parameter set. The variation using the seed values for the PRNG gives 50 values, and from these one maximum, average and minimum are computed.

Clusters of PSNR-values with different magnitudes are evident, and they result from the order used for the loops over the arguments to collect and aggregate the PSNR values. Clusters of the three biggest sizes were labelled "A", "B" and "C" in the figure: Areas marked as "A" denote the "outer loop", each bubble (and the neighbours which are not explicitly marked) contains samples generated with the same value for the parameter "minimum number of decompositions of the approximation subband". Areas like "B" contain samples with a fixed value for "maximum number of decompositions of the approximation subband". Since the maximum number must not be smaller than the minimum and both parameters have a maximum of 7 it is clear that the number of samples with an identical minimum decomposition depth for the approximation becomes smaller with increasing values. This also means that the width of the "A" bubbles decreases. Entries marked with "C" show identical values of the next parameter, in this case our "base value". The remaining factors are almost invisible at this resolution.

As a consequence of these results, we set the minimum number of approximation subband decompositions to 4 and the corresponding maximum value to 7. The maximum number of allowed decompositions for any other subband is set to 7. The "base value" and the "change factor" are set to 0.55 and 0.15, respectively. These settings improve the compression performance of admissible wavelet packet trees significantly: the maximum is 26.19 dB, the minimum is 24.98 dB, and the average of all subband trees is 25.60 dB.

In the following, we focus on possible attacks and their complexity. We assume that the attacker has all knowledge about the coder except the key for the symmetric cipher (AES) which protects the subband tree structure information. To be able to reconstruct the tree the attacker has four possibilities:

1 Break the cipher,

2 Reconstruct the tree with the help of unencrypted data.

3 Exhaustive search for the correct subband structure,

4 a different, yet unknown method

Using AES as symmetric cipher the first option is computationally infeasible. In the following we provide some details concerning the options 2 and 3.

First we discuss the reconstruction of the subband structures if a uniform quantiser is used for coding. We assume that the coefficients are stored in a subband oriented manner, i.e. coefficients of one subband correspond to a contiguous part of the bitstream. Within every single subband the coefficients are stored in scanline order, going from left to right, and each line from top to bottom. The subbands are traversed in depth-first order. If a different scan order for the coefficients would have been used (e.g. scanning the tree structured image data in one run for all values, line by line), then some details in the following procedure had to be changed but the overall method stays the same. In any case, in order to reconstruct the image, the attacker has to superimpose a subband structure.

We demonstrate that this can be done by exploiting the distinct statistical properties of WP subbands in two basic steps: first, reconstruction of the size of the approximation subband and second, reconstruction of the remaining tree

Of course, the most important step in this attack is to reconstruct the size of the approximation subband since here the most important information is concentrated. The statistics for the approximation subband differ significantly from the detail subbands: In the approximation there is roughly an equal distribution of all values, in the detail subbands many values are close to 0, and only few coefficients have high values (positive and negative as well). Because of this difference the separation is achieved fairly easily. We use the following formula to estimate the distribution of the coefficients, a slightly modified variance

$$v = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - m)^2$$

– m is the previously computed average of all coefficients and N denotes the number of coefficients currently examined. This means that we compute the distance of the first N coefficients from the average of all coefficients. v is evaluated for an increasing N following the scan order.

Figure 5.35 shows three curves of v related to different subband structures where the peak of the curves just before the steady descent gives the size of the approximation subband. In the current scenario the attacker can also assume that the size of the approximation subband is a power of 2, if the maximum value of the modified variance is not exactly a power of 2, it has to be close to it at least. The fact that this modified variance is capable of detecting the approximation size has been verified in experiments and so a better (i.e. faster) attacking method can be constructed to detect at first the descending slope by taking a few sample values going "left" until a power of 2 is reached which gives the maximum value.

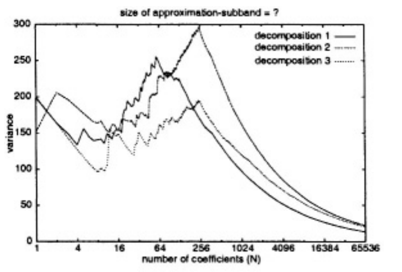


Figure 5.35. Variance for increasing number of coefficients

Once the size of the approximation subband is known, an attacker can get a rough impression of the image but lacks details (compare figure 5.36).

The next step is to reconstruct the remaining decomposition tree which contains the detail subbands. This can be done by exploiting the fact that adjacent lines of coefficients within a subband are correlated since they represent pixels which are situated close to each other. For the subbands in the neighbourhood of the approximation subband there is an upper bound of the size of the subband: the size of the approximation subband. A lower bound exists as well: a subband of size 1×1 coefficients (in theory), but in practice we may assume the minimum side length to be 2, 4 or even 8 coefficients. Lengths less than 8 are more difficult to cover since in general the amount of data for computing correlations is not big enough. So automatic guesses are not very reliable for small subband sizes, very small subbands usually decrease rate-distortion performance and should be avoided anyway.

In order to determine the size of a subband, we perform a loop ranging from the minimum side length to the maximum side length for the subband in question using subsequent powers of 2. At each step a correlation factor (i.e.

a distance) between adjacent lines of coefficients is computed. Each line is assumed to be a vector of pixels and then the distance between these vectors is calculated. Within this process, we face three situations: We assume a subband size which is smaller than the correct size, we guess the correct size or we assume a size which is larger than the actual size.

In the case we assumed a size which is smaller than the correct size the actual line of coefficients is split into two lines in our subband in testing. When we test the distance a large value is detected. If we assumed the correct size the difference between the lines is smallest.

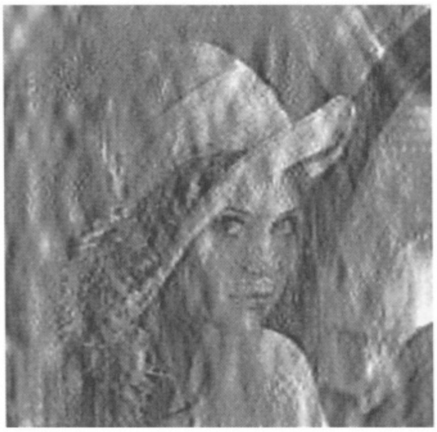


Figure 5.36. Reconstruction using a wrong decomposition tree but the correct approximation subband size

In the case we assumed a size which is too large we put two lines of the correct subband into one line of our subband. Here we have a correlation as well since consecutive lines in our subband denote alternating lines in the original subband. Therefore, a correlation exists but the distance is higher than in the exact match.

By comparing the distances from several assumed side lengths we can determine the actual size. The iteration step which gives the smallest distance is assumed that it corresponds to the correct match. This iteration is applied recursively to all subbands, the maximum side length is calculated according to the subbands in the neighbourhood. The same process is applied to columns of coefficients as well in order to achieve a higher stability and reliability of the results. Concerning the type of distance measure used, the most promising results were obtained when two L^i-**norms** were used and one of them was the Euclidean norm.

As a consequence, it turns out that our approach is not secure enough when a uniform quantiser is used to encode the coefficients. In most modern wavelet compression methods however more sophisticated coders (as the zerotree coder) are used. These coders require that the encoder and decoder have to be exactly in sync because of the high level of context. In this case the difficulty for attacks is much higher. First investigations show that the complexity is moved near a brute force search for the key used for symmetric encryption. Using a zerotree coder instead of a uniform quantiser has another advantage in general:

the compression rates for a given image quality are better. What can be done in this case to reconstruct the image follows next.

As we have seen, the second option can not be successful in the case of zerotree-based encoding. The stored bits contain more information than actual values, information about significance is stored as well. Additionally, coefficient data is not stored subband oriented but significance oriented. To reconstruct a wavelet packet transformed image using zerotrees the encoder and decoder have to be in perfect synchronisation. This synchronisation is possible just in the case when both parts have the same knowledge about the tree structure. Otherwise the encoder and decoder are out of sync and the bits are interpreted in the wrong way on the decoding end, e.g. bits denoting the significance can be interpreted as sign bits. For an analysis of partial encryption of zerotree encoded imagery see [22].

In the following we investigate option three (exhaustive search for the correct subband structure) in some detail. Equation 5.8 gives the number of possible decomposition trees $f(n)$ reaching up to level $n + 1$.

$$f(n) = \sum_{i=0}^{4} \binom{4}{i} \cdot (f(n-1))^i \tag{5.8}$$

with $f(0) = 1$ and $a(0) = 1$

For $n = 4$ (5 decomposition levels) this number is in the order of 10^{78} or 2^{261} which is higher than the complexity of a brute-force attack against encryption using a 256-bit-key AES cipher. But not all subband trees are admissible if a certain compression quality must be guaranteed. Equation 5.9 shows the number of possible decomposition trees $a(n)$ if a minimum and maximum decomposition depth of the approximation subband has been specified. Case (a) has to be applied if the number of decompositions for the approximation subband is below the minimum, (b) is the standard case, and (c) applies when the number of decomposition for the approximation subband reached the maximum.

$$a(n) = \begin{cases} a(n-1)\sum_{i=0}^{3} \binom{3}{i} \cdot (f(n-1))^i & \text{(a)} \\ \sum_{i=0}^{3} \binom{3}{i} \cdot (1 + a(n-1))(f(n-1))^i & \text{(b)} \\ \sum_{i=0}^{3} \binom{3}{i} \cdot (f(n-1))^i & \text{(c)} \end{cases} \tag{5.9}$$

When restricting the parameters as suggested previously in section we still result in approximately 2^{4185} decomposition trees (since the difference to the non-restricted setting is 2^{4120} "only"). If the information about restricted parameter ranges is published and further parameters are known to the attacker (like the parameters determining the weight - which should be encrypted anyway), the more probable trees could be tested first. Regarding the vast number of different decomposition trees, attack complexity can not become lower as an attack against 256-bit AES as long as the maximum allowed number of decompositions for all subband is set high enough.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	high	yes	no	moderately

Table 5.21. **Overall assessment of Secret Subband Structures (zerotree variant)**

2.2 Bitstream Oriented Schemes

2.2.1 SPIHT Encryption

Cheng and Li [22] discuss a partial encryption scheme for SPIHT which can be applied to any zerotree-based wavelet coding scheme. The basic observation is as follows: the compression algorithm produces many different types of bits – sign bits, refinement bits, and significance bits of pixels and sets. The decompression algorithm has to interpret each bit under the correct context. Incorrect significance bits may lead to an incorrect interpretation of subsequent bits, this is not the case when sign bits or refinement bits are decoded incorrectly. As a consequence it is suggested to encrypt the significance information of sets and pixels of the two lowest resolution pyramid levels. The reason for not encrypting all significance information is as follows: the significance information of the low resolution levels is used to initialise the different lists used by the algorithm. If the states of these lists are incorrect right from the start of the decoding, it is hardly possible for the algorithm to recover from the error. The information left unencrypted is of low value for an attacker since without the significance information the type of bits can not be distinguished from another. Basically the argumentation is quite similar to the case of encrypting the wavelet packet subband structure only (see last section).

The amount of data encrypted is very small in this proposal – less that 7% in case of 256 × 256 pixels images and less than 2% in case of 512 × 512 pixels images. The question of bitstream compliance is not discussed, but this property can be easily obtained. Although the methods seems to be very secure, no experimental attacks have been mounted against the scheme proving its robustness.

2.2.2 JPEG 2000 Encryption

For selectively encrypting the JPEG 2000 bitstream we have two general options. First, we do not care about the structure of the bitstream and sim-

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	low	high	yes	yes	no

Table 5.22. Overall assessment of SPIHT Encryption

ply encrypt a part, e.g. the first 10% of the bitstream. In this case, the main header and a couple of packets including packet header and packet data are encrypted. Since basic information necessary for reconstruction usually located in the main header is not available at the decoder, encrypted data of this type can not be reconstructed using a JPEG 2000 decoder. Although this seems to be desirable at first sight, an attacker could reconstruct the missing header data using the unencrypted parts, and, additionally, no control over the quality of the remaining unencrypted data is possible. Therefore, the second option is to design a JPEG 2000 bitstream format compliant encryption scheme which does not encrypt main and packet header but only packet data. This option is investigated further.

Grosbois et al. [57] propose the first partial encryption scheme for JPEG 2000 bitstreams. A pseudo random inversion of the bits in certain layers is suggested, but no further details with respect to amount and position of the encrypted data are given. Also, no attacks are demonstrated. Wee and Apostopoulos [172] integrate Motion JPEG 2000 into their secure scalable streaming concept (SSS, compare also section 1.4 (chapter 5)) by exploiting the different ways of achieving scalability in JPEG 2000. Of course, JPEG 2000 suits much better in the SSS context as compared to other codecs since scalability is an inherent property, different mixtures of quality and resolution levels are experimentally evaluated. Triple-DES and AES in CBC mode are used for encryption which means that data has to be padded to suit the block-size specification of these algorithms. A very interesting issue is discussed by Kiya et al. [72] in the context of encrypting packet data of JPEG 2000 streams. Straightforward encryption of this data may lead to the emulation of marker codes which cause the resulting bitstream to be non-compliant and would cause a decoder to crash. They suggest to perform the encryption process based on half bytes in a specific marker aware mode which uses the hexadecimal notation of the markers. Wu and Deng [180] also address the problem of compliant encryption and suggest to check the compliance during the encryption process and to change the process accordingly in case of marker generation. The same authors also discuss an access control scheme using a key generation scheme for parts of the codescheme [181] relying on their former work on hash trees for authenticating JPEG 2000 streams [109].

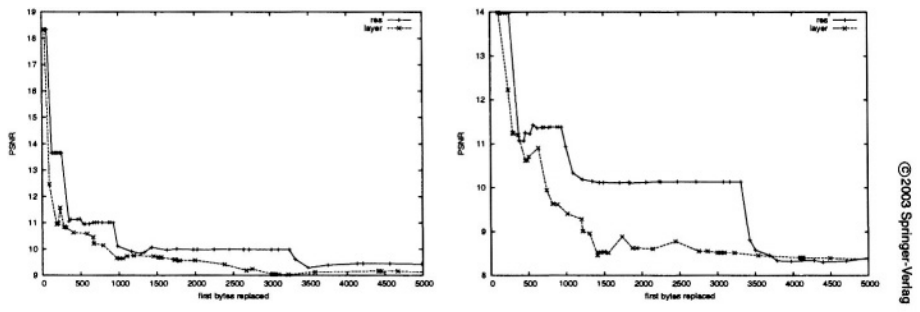
In the following, we will investigate how much packet data needs to be protected to provide reasonable confidentiality and we will attack a corresponding encryption scheme (compare also [106]). These experiments may be performed online at `http://www.ganesh.org/book/`. We have decided to discuss the encryption efficiency of a JPEG 2000 partial encryption scheme with respect to two different classes of visual image data. The first class of visual data discussed is typical still image data and the testimage representing this class is the Lena image at different resolutions including 256 × 256 and 512 × 512 pixels. Since this special type of visual data is usually encoded in lossy mode, the lena image is lossy coded in our experiments (at a fixed rate of 2 bpp). The second type of digital visual data should represent an application class where lossless coding is important. We have therefore decided to use an angiogram as testimage in this case (see the corresponding appendix), since angiograms represent an important class of medical image data where lossless coding is usually a mandatory requirement.

In order to make the explanations and experiments of the proposed techniques simpler, we assume the testimages to be given in 8bit/pixel (bpp) precision and in a squared format. Extensions to images of different acquisition types, higher bitdepth or non-squared format are straightforward.

In order to achieve format compliance, we need to access and encrypt data of single packets. Since the aim is to operate directly on the bitstream without any decoding we need to discriminate packet data from packet headers in the bitstream. This can be achieved by using two special JPEG 2000 optional markers which were originally defined to achieve transcoding capability, i.e. manipulation of the bitstream to a certain extent without the need to decode data. Additionally, these markers of course increase error resilience of the bitstream. These markers are "start of packet marker" (SOP - 0xFF91) and "end of packet marker" (EPH - 0xFF92). The packet header is located between SOP and EPH, packet data finally may be found between EPH and the subsequent SOP. For example, using the official JAVA JPEG 2000 reference implementation (JJ2000 - available at `http://jj2000.epfl.ch`) the usage of these markers may be easily invoked by the options `-Peph on -Psop on`.

Having identified the bitstream segments which should be subjected to encryption we note that packet data is of variable size and does not at all adhere to multiples of a block ciphers block-size. We have to employ AES in CFB mode for encryption, since in this mode, an arbitrary number of data bits can be encrypted, which is not offered by the ECB and CBC encryption modes. Information about the exact specification of the cryptographic techniques used (e.g. key exchange) may be inserted into the JPEG 2000 bitstream taking advantage of so-called termination markers. Parts of the bitstream bounded by termination markers are automatically ignored during bitstream processing and do not interfere with the decoding process. Note that a JPEG 2000 bitstream which is selectively encrypted in the described way is fully compliant to the standard and can therefore be decoded by any codec which adheres to the JPEG 2000 specification.

We want to investigate whether resolution progressive order or layer progressive order is more appropriate for selective JPEG 2000 bitstream encryption. We therefore arrange the packet data in either of the two progression orders, encrypt an increasing number of packet data bytes, reconstruct the images and measure the corresponding quality.



(a) Lena 256 × 256 pixels (b) Lena 512 × 512 pixels

Figure 5.37. Comparison of selective encryption (PSNR of reconstructed images) using resolution or layer progressive encoding - part 2.

Resolution progression is more suited for selectively encrypting the angiogram image at higher rates of encrypted data (see Fig. 5.38, where a lower PSNR means that it is more suited for selective encryption). In order to relate the obtained numerical values to visual appearance, two reconstructed versions of the angiogram, corresponding to the two progression orders, are displayed in Fig. 5.39. In both cases, 1% of the entire packet data has been encrypted.

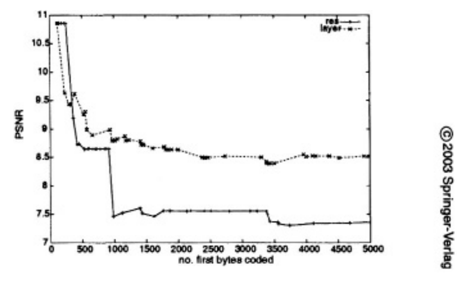
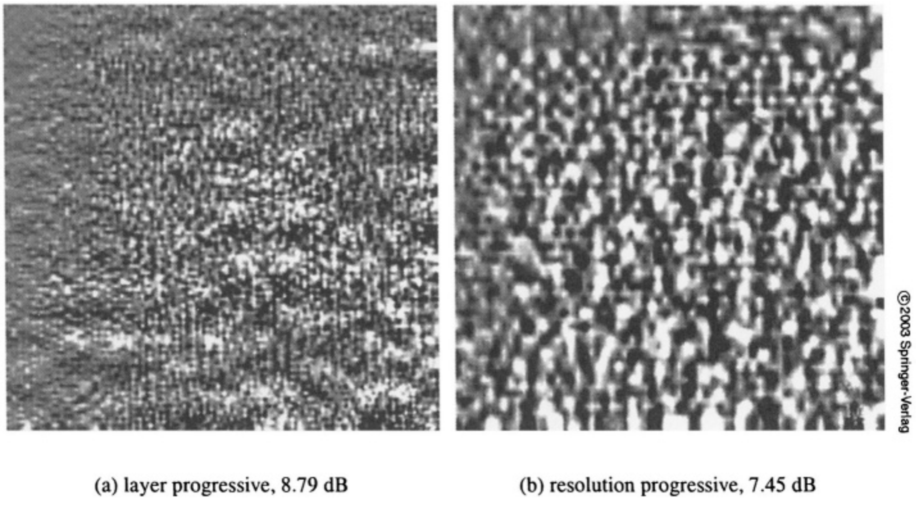


Figure 5.38. Angiogram: Comparison of selective encryption (PSNR of reconstructed images) using resolution or layer progressive encoding - part 1.

Whereas no details are visible using layer progression (Fig. 5.39.a at 8.79 dB), only very high frequency visual information (right lower corner) is visible using resolution progression (Fig. 5.39.b at 7.45 dB).

When considering the Lena image in Fig. 5.37, we observe that resolution progression shows superior PSNR results for both tested image dimensions as compared to layer progression. Two reconstructed versions of the Lena image with 512 × 512 pixels, corresponding to the two progression orders, are displayed in Fig. 5.40. In each case, 1% of the entire packet data has been encrypted. Whereas only very high frequency information is visible in the reconstructed image using layer progression (Fig. 5.40.a at 8.51 dB), important visual features are visible using resolution progression (Fig. 5.40.b at 10.11 dB). In this case, the visible high frequency information is enough to reveal sensible data. At 2 % encrypted packet data, this information is destroyed fully in the resolution progressive case.

The Lena image at lower resolution (256 × 256 pixels) performs equally, and the results are therefore only given for the 512^2 pixels version. Please note also the difference in coarseness of the noise pattern resulting from encryption between resolution and layer progression. Since in resolution progression data corresponding to the higher levels of the wavelet transform is encrypted, the noise introduced by the cipher is propagated by the repeated inverse transform and thereby magnified resulting in a much coarser pattern as compared to layer progression. When summarising the obtained numerical and visual results, it seems that encrypting 1-2% of the packet data in layer progressive mode is sufficient to provide confidentiality for the JPEG 2000 bitstream. This is a very surprising result of course.



(a) layer progressive, 8.79 dB **(b) resolution progressive, 7.45 dB**

Figure 5.39. Comparison of selective encryption (visual quality of reconstructed Angiogram where 1% of the bitstream data have been encrypted) using resolution or layer progressive encoding.

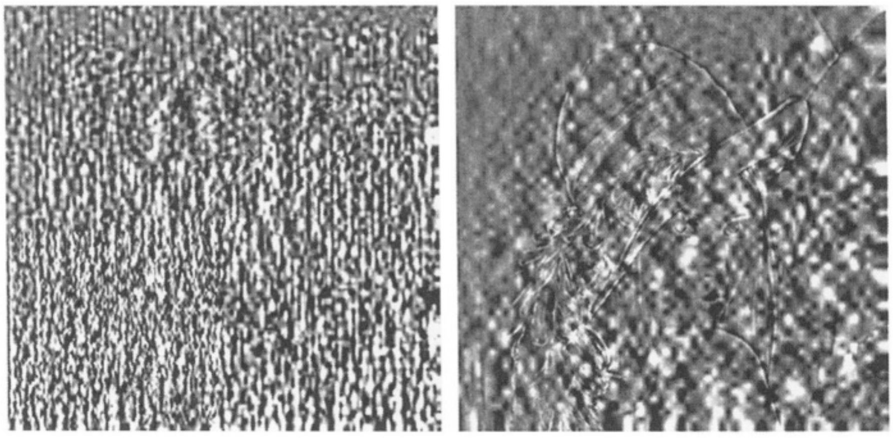

(a) layer progressive, 8.51 dB
(b) resolution progressive, 10.11 dB

Figure 5.40. Comparison of selective encryption (visual quality of reconstructed Lena (512 pixels) where 1% of the bitstream data have been encrypted) using resolution or layer progressive encoding.

We want to assess the security of the presented selective encryption scheme by conducting a simple ciphertext only attack. Therefore, an attacker would replace the encrypted parts of the bitstream by artificial data mimicking typical images ("replacement attack", see also [112]). This attack is usually performed by replacing encrypted data by some constant bits (i.e. in selective bitplane encryption). In encrypting the JPEG 2000 bitstream, this attack does not have the desired effect, since bitstream values are arithmetically decoded and the corresponding model depends on earlier results and corrupts the subsequently required states. Therefore, the reconstruction result is a noise-like pattern similar as obtained by directly reconstructing the encrypted bitstream. We exploit a built-in error resilience functionality in JJ2000 to simulate a bitstream-based replacement attack. An error resilience segmentation symbol in the codewords at the end of each bit-plane can be inserted. Decoders can use this information to detect and conceal errors. This method is invoked in JJ2000 encoding using the option `-Cseg_symbol on`.

If an error is detected during decoding (which is of course the case if data is encrypted) it means that the bit stream contains some erroneous bits that have led to the decoding of incorrect data. This data affects the whole last decoded bit-plane. Subsequently, the affected data is concealed and no more passes should be decoded for this code-block's bit stream. The concealment resets the state of the decoded data to what it was before the decoding of the affected bit-plane started. Therefore, the encrypted packets are simply ignored during decoding.

Using this technique, we again compare selective JPEG 2000 encryption using resolution and layer progressive mode layer progressive by reconstructing images with a different amount of encrypted packets. Decoding is done using error concealment. In Fig. 5.41 and 5.42 we immediately recognise that the PSNR values are significantly higher as compared to directly reconstructed images (see Fig. 5.38 and 5.37). Layer progression is more suited for selectively encrypting the angiogram image. For the lena test images, the situation differs slightly: When encrypting only minor parts of the overall bitstream, layer progression is superior, at higher rates of encryption, the resolution progression scheme shows superior results.

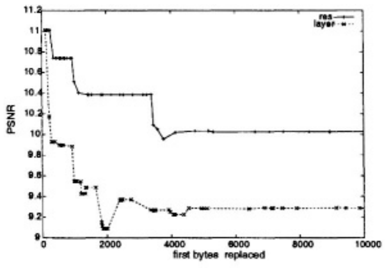
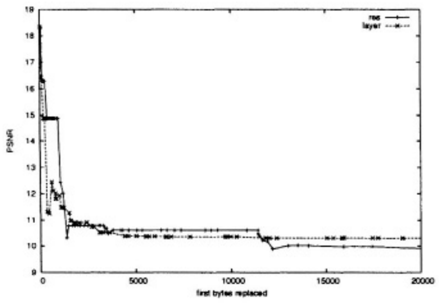
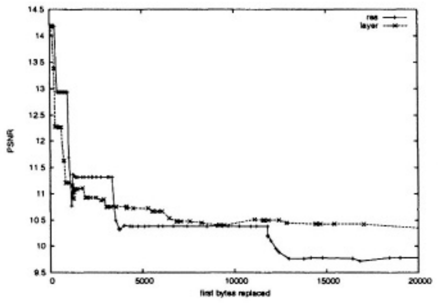


Figure 5.41. Angiogram: PSNR of reconstructed images after replacement attack using resolution or layer progressive encoding - part 1.



(a) Lena 256 × 256 pixels



(b) Lena 512 × 512 pixels

Figure 5.42. PSNR of reconstructed images after replacement attack using resolution or layer progressive encoding - part 2.

Again, the numerical values have to be related to visual inspection. Fig. 5.43.a shows a reconstruction of the selectively compressed angiogram image, where the first 1% of the packets in resolution progressive mode have been encrypted and the reconstruction is done using the error concealment technique. In this case, this leads to a PSNR value of 10.51 dB, whereas the directly reconstructed image has a value of 7.45 dB (see Fig. 5.39.b). The text in the right corner is clearly readable and even the structure of the blood vessels is exhibited. The Lena image performs similarly (see Fig. 5.44.a), all important visual features are reconstructed at 1% encrypted. Here, we have a resulting PSNR of about 11.31 db, whereas the directly reconstructed image has a value of 10.11 dB (see Fig. 5.40.b).

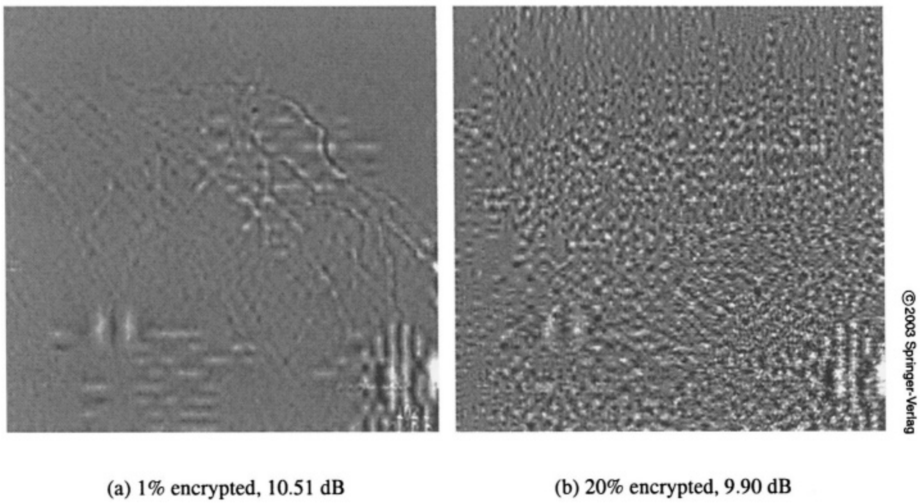
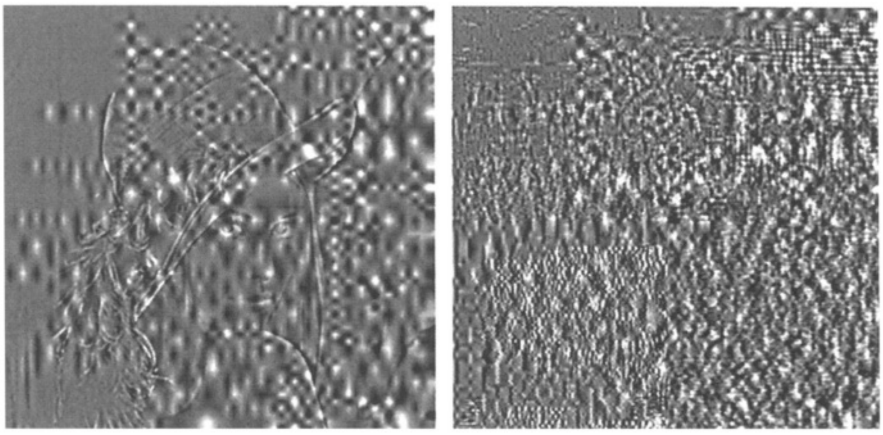


Figure 5.43. Visual quality of reconstructed Angiogram after replacement attack using resolution encoding.

When increasing the percentage of encrypted packet data steadily, we finally result in 20% percent of the packet data encrypted where neither useful visual nor textual information remains in the image (see Fig. 5.43.b and 5.44.b). This result is confirmed also with other images including other angiograms and other still images and can be used as a rule of thumb for a secure use of selective encryption of the JPEG 2000 bitstream. It has to be noted that the amount of data required to be encrypted is significantly lower in the case of zerotree-based coding schemes. This is due to the higher level (i.e. inter subband) of context information in those coding schemes.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
medium+scalable	low	high	yes	no	no

Table 5.23. Overall assessment of JPEG 2000 Encryption



(a) 1% encrypted, 11.31 dB (b) 20% encrypted, 9.77 dB

Figure 5.44. Visual quality of reconstructed Lena (512 pixels) after replacement attack using resolution encoding.

3. Further Techniques

3.1 Raw Image Data

3.1.1 Permutations

Applying permutations to the raw image data is the simplest and fastest way to apply encryption technology to visual data. Hybrid Pay-TY systems (i.e. analog signal transmission but encryption and decryption is done in the digital domain) have extensively made use of this technology. The Nagravision/Syster system for example applies line permutations within blocks of 32 lines, VideoCrypt (as formerly used by the SKY network) uses specific permutations within each line of the video frame by cutting each line at a secret position and interchanging the two resulting sub-lines. Although the key material is changed frequently, both systems are vulnerable to a ciphertext only attack by using smoothness constraints (guessing a correct permutation results in a smoother image than guessing incorrectly) and known facts about the generation of the permutation keys. Macq and Quisquater [89] propose to use line permutations in the context of a multiresolution image decomposition which facilitates a good control over the amount of degradation. However, this scheme is not more secure than "pure" permutation in the image domain.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	low	yes	no	yes

Table 5.24. Overall assessment of Permutations applied to raw image data

3.1.2 Chaos-based Systems

Chaos-based encryption of visual data uses the principle of applying chaotic maps with strong mixing properties to the raw image data. The basic idea is that (continuous) chaotic maps exhibit similar properties as (discrete) cryptographic systems. Usually, these systems are hybrids between permutation and substitution ciphers with specific properties. Therefore, they are very fast. Scharinger [128] was the first to apply a class of such maps called Kolmogorov flows for this purpose, Fridrich refined and systematised this approach [50, 51]. These algorithms have also been used to conceal logo-type images for watermarking generation [129, 167].

The most famous example of a chaotic map is the "Baker Map" B (defined on $[0, 1]^2$) as follows:

$B(x, y) = (2x, y/2)$ for $0 \leq x < 1/2$ and $B(x, y) = (2x - 1, y/2 + 1/2)$ for $1/2 \leq x \leq 1$.

The left vertical half of the domain [0, 1/2) × [0,1) is stretched horizontally and contracted vertically to be mapped to the domain [0,1) × [0,1/2). In the same way the right half [1/2,1) × [0,1) is mapped to [0,1) × [1/2, 1). Fig. 5.45 illustrates this principle.

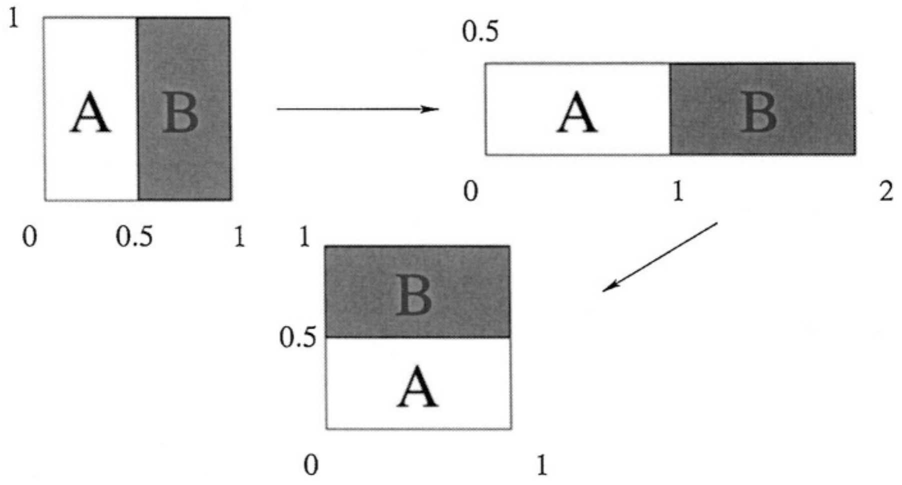


Figure 5.45. Baker Map (1/2,1/2).

The Baker map may be generalised as follows. Instead of processing two halves of the unit interval one considers k vertical rectangles $[F_{i-1}, F_i) \times [0, 1)$ for $i = 1, \ldots, k$ and $F_i = p_1+p_2+\cdots+p_i$, $F_0 = 0$, such that $p_1+\cdots+p_k = 1$. F_i is the left lower corner of rectangle i. This generalisation stretches each rectangle horizontally by a factor $1/p_i$ and contracts it vertically by the factor p_i. Subsequently, the resulting rectangles are stapled above each other.

In order to be able to apply this map to pixel values, it need to be discretised to a bijection between pixel values. We define a sequence of k positive integers $n_1, \ldots, n_k$ where each n_i divides the width N of a squared image without remainder and $n_1 + \cdots + n_k = N$, $N_i = n_1 + \cdots + n_i$, and $N_0 = 0$. The pixel (r, s) with $N_{i-1} \leq r < N_i$ and $0 \leq s < N$ is mapped to the following pixel ($q_i = N/n_i$):

$$(q_i(r - N_i) + s \pmod{q})_i) , (s - s \pmod{q})_i)/q_i + N_i) .$$

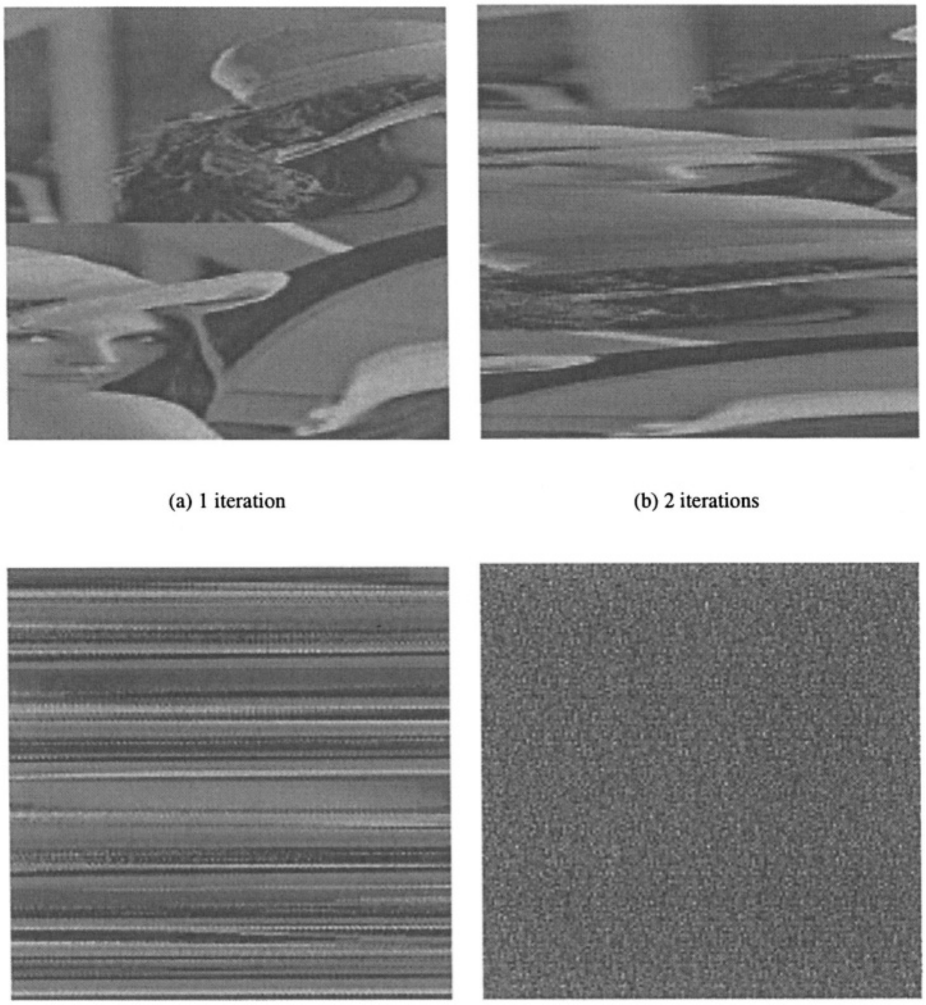
So far, the technique is a pure pixel permutation, defined by the discretised Baker map. The permutation key is the choice of the values for the n_i. In order to obtain stronger mixing properties, the map is further generalised to a three dimensional map. A pixel (r, s) with gray value g_{rs} is mapped to $B(r, s)$ with gray value $h(r, s, g_{rs})$ which means that the new gray value depends on pixel position and former gray value. In order to guarantee reversibility, the function h has to be a bijection in the third variable, e.g. $h(r, s, g_{rs}) = g_{rs} + r * s \pmod{L}$ where L is the number of available gray values.

After a low number of iterations (compare Fig. 5.46 which has been generated using J. Scharingers demo page[1]) this technique results in an image with equalised histogram. In order to add diffusion properties, a non-linear feedback shiftregister generator is applied to each column ($g^*_{rs} = g_{rs} + G(g^*_{rs-1}) \pmod{L}$ with arbitrary seed). The key material of the entire system consists of the parameters of the chaotic map, the number of iterations, the parameters of the gray value transform h, and the values of the shiftregister generator.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	0	medium-high	yes	no	yes

Table 5.25. Overall assessment of Chaotic Encryption

For more information and a detailed description of various flavours of this techniques please refer to Chapter 4 of the Multimedia Security Handbook [54].



(a) 1 iteration

(b) 2 iterations

(c) 4 iterations

(d) 10 iterations

Figure 5.46. Baker Map (1/2, 1/2) applied to Lena.

3.1.3 Bitplane Encryption

For the results in this section see also [104, 112]. For simplicity, we assume an 512×512 pixels image to be given in 8bit/pixel (bpp) precision. We consider the 8bpp data in the form of 8 bitplanes, each bitplane associated with a position in the binary representation of the pixels. The encryption approach is to e.g. AES encrypt a subset of the bitplanes only, starting with the bitplane containing the most significant bit (MSB) of the pixels. Each possible subset of bitplanes may be chosen for encryption, however, the minimal percentage of data to be encrypted is 12.5 % (when encrypting the MSB bitplane only), increasing in steps of 12.5 % for each additional bitplane encrypted. We use an AES implementation with blocksize 128 bit and a 128 bit key. The 128 bit block is filled with a quarter of a bitplane line (512/4 = 128 bits). The encrypted bitplanes are transmitted together with the remaining bitplanes in plain text.

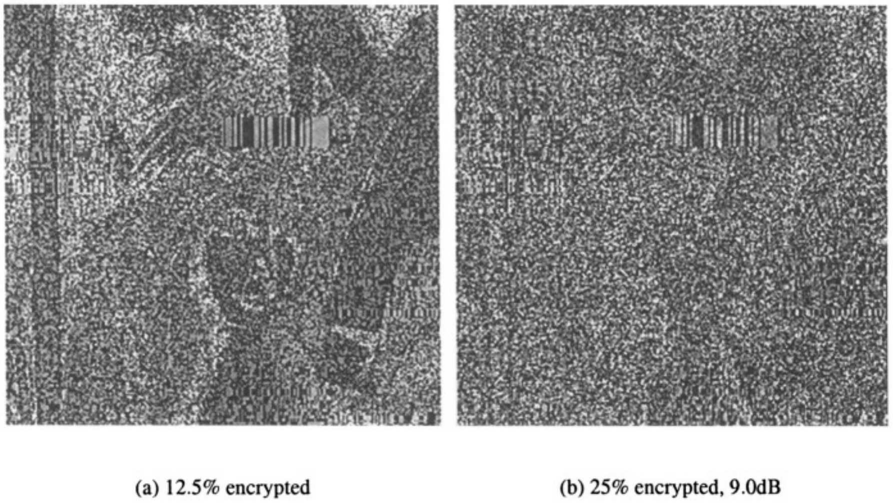
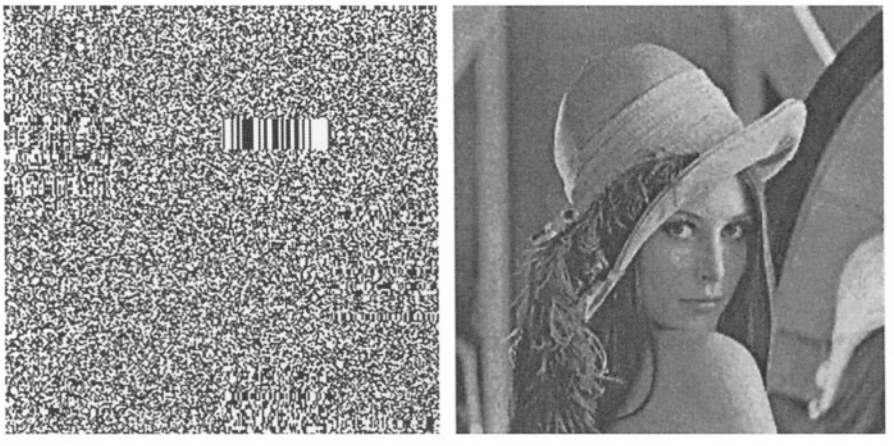


Figure 5.47. Visual examples for selective bitplane encryption, direct reconstruction.

Fig. 5.47 shows two examples of directly reconstructed images after selectively encrypting 1 and 2 bitplane(s). Whereas in the case of encrypting the MSB only structural information is still visible, encrypting two bitplanes leaves no useful information in the reconstruction, at least when directly reconstructing the image data.

Note the pattern reminiscent of a bar code in the upper right quarter of the image. Fig. 5.48.a shows the encrypted MSB of the Lena image where this pattern is exhibited even clearer. This phenomenon is due to the fact that AES encryption is used with identical key for all blocks in the image. Consequently, if there are identical plain text quarter-lines directly situated above each other which also adhere to the AES block-border (i.e. starting at pixel positions 0, 128, 256, or 384), these data produce identical ciphertext blocks. Identical blocks of ciphertext are again arranged as identical quarter-lines thereby generating the barcode effect. For the corresponding region with identical quarter-lines starting at pixel position 128 in the MSB of the Lena image refer to Fig. 5.51.a.



(a) encrypted MSB (b) 50% encrypted, 31.8dB

Figure 5.48. Further visual examples for selective bitplane encryption.

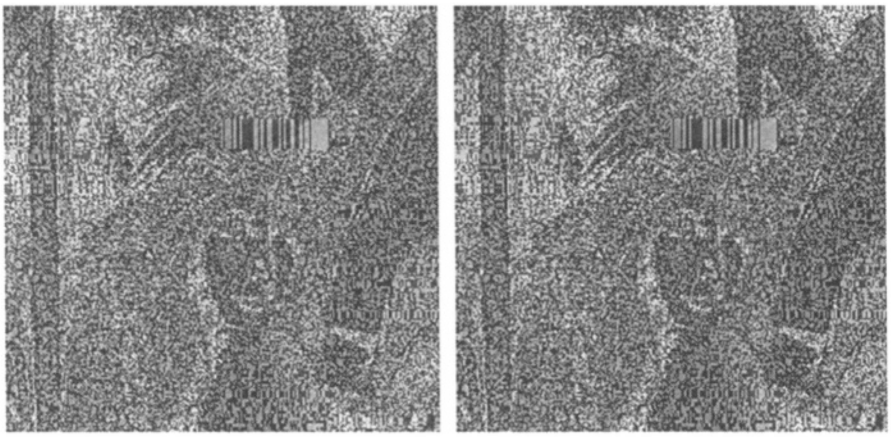
Note that it is of course important to encrypt the MSB first and continue with the bitplanes corresponding to the next bits in the binary representation. Fig. 5.48.b shows the case where the image is directly reconstructed after 4 bitplanes have been encrypted starting from the least significant bit (LSB). Almost no degradation is visible here – consequently it hardly makes any sense at all to encrypt these data. Table 5.26 gives the PSNR values of images subjected to the SE approach. Whereas the PSNR is constant 9 dB when encrypting the MSB first, PSNR decreases steadily from 51 dB to 14 dB for each additional bitplane encrypted and reaches 9 dB when encrypting all bitplanes after all in the case when the LSB bitplane is encrypted first.

# Bitplanes	1	2	3	4	5	6	7	8
First: LSB	51	44	38	32	26	20	14	9
First: MSB	9	9	9	9	9	9	9	9

Table 5.26. PSNR of images after direct reconstruction related to the number of encrypted bitplanes and to the ordering of the bitplanes.

A technique to eventually increase the security could be not to disclose which bitplanes have been subjected to encryption besides the MSB. Fig. 5.49

shows directly reconstructed images where the MSB and n-th most significant bitplanes have been encrypted. Clearly, the visual quality is comparable to encrypting the MSB alone (compare Fig. 5.47.a).



(a) MSB + 4th (b) MSB + 5th

Figure 5.49. Visual examples for encryption of MSB and one additional bitplane.

Additionally, the statistical properties of bitplanes of natural images and encrypted bitplanes are fairly different. Table 5.27 compares the number of runs consisting of 5 identical bits contained in bitplanes (plaintext and ciphertext). All but the three less significant bitplanes show a much higher value of runs in the plaintext version. Therefore, the "secret" which bitplanes have been encrypted can be immediately solved using simple statistics.

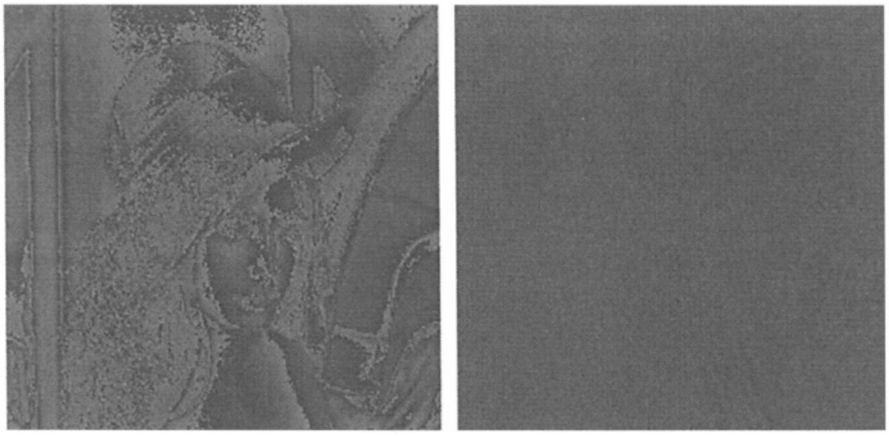
Bitplane	MSB	2	3	4	5	6	7	LSB
Plain	45	39	32	20	11	5	4	4
Encrypted	4	4	4	4	4	4	4	4

Table 5.27. Number of runs consisting of 5 identical bits (rounded to thousand, Lena image).

As a consequence, the most secure way to perform selective bitplane encryption is to encrypt the MSB bitplane and subsequently additional bitplanes in the order of decreasing significance with respect to their position in the binary representation.

In the following we assess the security of selective bitplane encryption by conducting two types of simple ciphertext only attacks. A shortcoming of many investigations of visual data encryption is the lack of quantifying the quality of the visual data that can be obtained by attacks against encryption. Mostly visual examples are provided only. The reason is the poor correlation of PSNR and other simple quality measures and perceived quality especially for low-quality images [143]. Note for example that the PSNR computed between the image Lena and its entirely AES encrypted version is 9.2 dB whereas PSNR between Lena and an image with constant grayvalue 128 is 14.5 dB ! Both images do not carry any structural information related to Lena, however, the PSNR values differ more than 5 dB. For the most simple attack we may even relate the visual examples to meaningful numerical values.

Assuming the cipher in use is unbreakable we conduct the first attack by directly reconstructing the selectively encrypted images. The encrypted parts introduce noise-type distortions (see Fig. 5.47). Therefore, we replace the encrypted parts by artificial data mimicking typical images. The encrypted bitplane is replaced by a constant 0 bitplane and the resulting decrease in average luminance is compensated by adding 64 to each pixel if only the MSB bitplane was encrypted, 96 if the MSB and next bitplane have been encrypted, and so on. Subsequently, reconstruction is performed as usual, treating the encrypted and replaced parts as being non-encrypted.



(a) 25% encrypted, 13.2dB (b) 50% encrypted

Figure 5.50. Visual examples for the efficiency of the Replacement Attack.

Fig. 5.50 shows two visual examples of image reconstructions as obtained by the Replacement Attack (2 and 4 bitplanes are encrypted). Whereas a direct reconstruction of an image with 2 bitplanes encrypted suggests this setting to be “safe” (with 9.0dB quality, see Fig. 5.47.b), the Replacement Attack reveals that structural information is still present in the reconstructed image (with 13.2dB quality, see Fig. 5.50.a). However, the visual information is severely alienated. Obviously, not only the visual appearance but also the numerical PSNR values have been significantly improved by the Replacement Attack. In any case, even if a Replacement Attack is mounted, encrypting 4 bitplanes (i.e. 50% of the original data) leads to perfectly satisfying results (Fig. 5.50.b).

For the simplest case of this encryption technique, we assume the MSB bitplane to be encrypted only. The idea of the *Reconstruction Attack* is to reconstruct the MSB data with the aid of the unencrypted remaining data. We exploit the well known property, that most regions of natural images are covered by areas with smoothly changing gray values (except edges, of course). In areas of this type, the MSBs of all pixels tend to be identical (except for the case of medium luminance). In order to automatically detect such areas we define a 2 × 2 pixels search window in which all 16 possible combinations of MSB configurations are tested. In this test, a certain set of differences among the 4 pixel values is computed for each of the 16 MSB configurations. Out of the set of differences, the smallest difference is selected and the corresponding configuration of the MSB bits in the search window is defined to be the reconstruction. Fig. 5.51.a shows the MSB of the Lena image and Fig. 5.51.b a reconstructed bitplane obtained as described above.

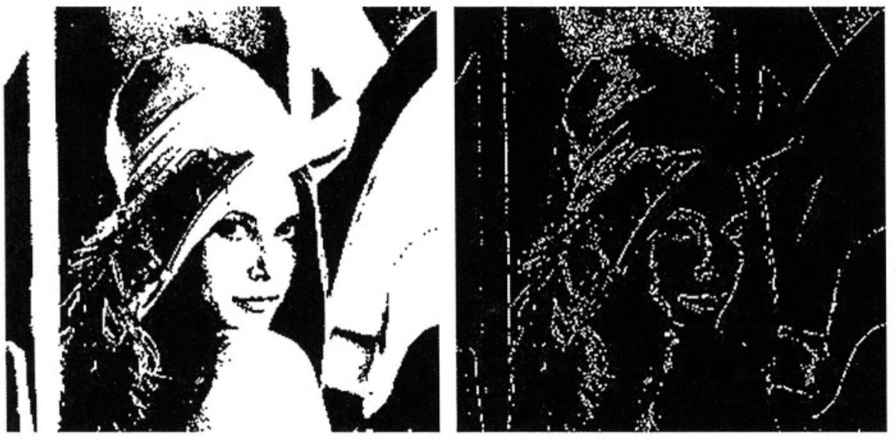


Figure 5.51. MSB of the Lena image and reconstructed Bitplane.

It is clearly visible that smooth areas are satisfactorily recovered (black=0) whereas edges are represented by white lines. This "edge-detection capability" is due to the fact that when the search window hits an edge, the difference operation leads to an attempt to compensate thereby setting the MSB to different values at both sides of the edge. Fig. 5.52 shows an image (index=1) resulting from the Reconstruction Attack where about 50% of the smooth areas are recovered correctly. A second difference exists with equally low value which is obtained as well by setting all MSB values constant (white=1) in smooth areas. Using this as additional information, a second reconstruction is obtained where the remaining 50% of the smooth areas are recovered correctly (see Fig. 5.52 – index=2).

When combining these two reconstructed "half-images" the original may be obtained easily by choosing the correct areas from the respective half-images (see Fig. 5.52).

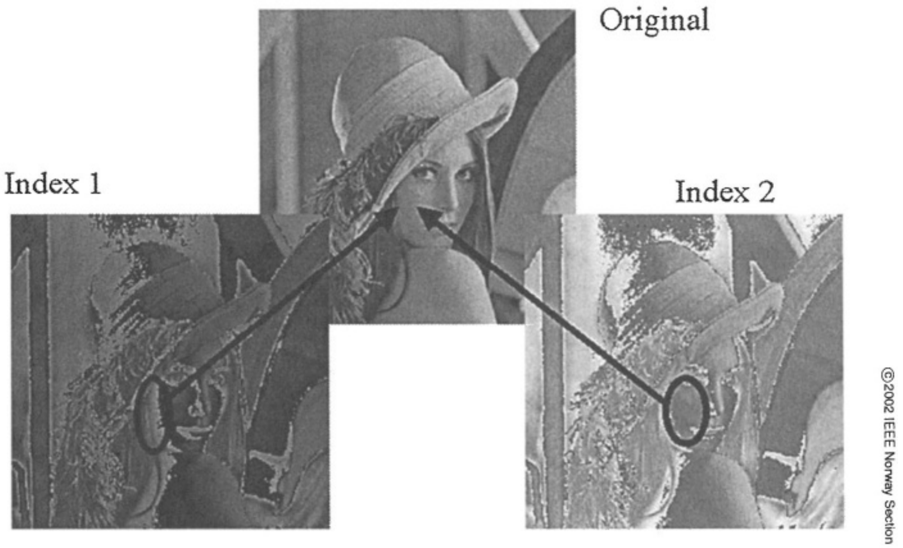


Figure 5.52. Combination of two half-images after Reconstruction Attack.

However, the complexity of this attack increases significantly if more bitplanes are encrypted and also the reliability of the results is drastically reduced. Summarising it seems that a relatively high amount of data needs to be encrypted to achieve reasonable security.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
high	low	low-medium	yes	no	yes

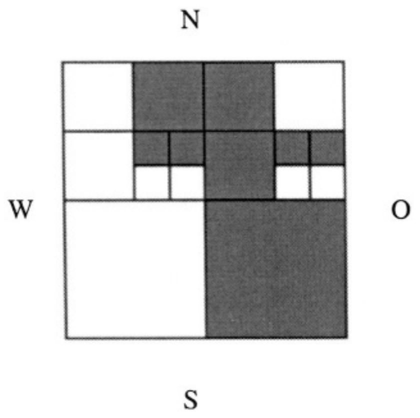
Table 5.28. Overall assessment of Bitplane Encryption

3.2 Quadtrees

Quadtree compression partitions the visual data into a structural part (the quadtree structure) and colour information (the leave values). Cheng et al. [21, 22, 82] suggest to encrypt the quadtree structure only and to keep the leave values in plaintext. As it is the case with wavelet packets (see section 2.1.4 (chapter 5)) a brute-force attack is not feasible due to the exceedingly high number of possible quadtrees. The only way to attack such a scheme is to try to deduce the quadtree structure from the non-encrypted leave values. In this context, the authors discuss two variants how the leave values may be organised in the compressed file (i.e. scan order, Fig. 5.53 shows an example for each of the orderings):

1 Leaf ordering I: is a depth first scan which starts with the NW quadrant, counterclockwise.

2 Leaf ordering II: is a line oriented scan within each level of the quadtree, starting with the smallest leaves.



Leaf Ordering I: 0010011011110010

Leaf Ordering II: 1111000001100101

Figure 5.53. Example for leaf ordering I and II.

It turns out there exists an attack against leaf ordering I whereas ordering II seems to be secure. The problem with leaf ordering I is that leaves which are neighbours in the quadtree are also neighbours in the bitstream. Based on this observation one notices that runs (i.e. a sequence of identical leave values) provide information about the local quadtree structure, since four identical adjacent leave values can never be situated at the same quadtree level (in this case no quadtree partitioning would have taken place). These facts may be exploited to significantly reduce the admissible quadtrees during an attack.

In case of lossy quadtree compression, the quadtree structure covers about 15 – 25 % of the entire data. Therefore, a significant amount of data needs

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
high	high	high	-	yes	no

Table 5.29. Overall assessment of Quadtree Encryption

to be encrypted. If the data is not given in quadtree compressed form (which is highly probable), the complexity of compression needs to be considered as well.

The authors also suggest to encode and subsequently encrypt motion vector fields using quadtrees. This can be done in identical manner, however, it has to be taken into account that size of the corresponding motion vector frames is rather small and the size of the keyspace (i.e. number of admissible quadtrees) may be rather small as well.

3.3 Fractal-based system

Roche et al. [125] propose an access control system for fractal encoded visual data which may be operated ranging from a full encryption mode to a variant where the images are only slightly distorted ("transparent encryption", see section 4 (chapter 5)). The main idea is to partially encrypt the binary representation of the luminance scale parameter. Whereas reconstruction or filtering of the corrupted image data is extremely difficult due to the highly non-linear distortions induced by fractal interpolation, the amount of parsing to extract the data subjected to encryption is significant. Further, due to the robustness of the decoding process, this technique is hardly useful to provide real confidentiality. Transparent encryption may be a better application field.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
low	high	low	yes	yes	no

Table 5.30. Overall assessment of Encrypting Fractal Encoded Data

3.4 Vector quantisation based system

Chen et al. [20] propose a very interesting image encryption system which is actually a hybrid between image encryption and image steganography. It is called a "virtual image cryptosystem". Using terminology from information hiding [68] a cover image is used to embed the image to be encrypted. The embedding stage is performed using a vector quantisation codebook which is derived from both the cover image and the image to be protected. The security relies on the fact that two random vectors are encrypted in secure manner, concatenated repeatedly to generate a keystream which is then XORed with the list of indices from vector quantisation. This simple scheme is of course a threat for the security of the system. Secure encryption of the entire list of indices would be the other choice, however, computational amount for this operation is significant. We rate the second approach only.

time(E)	time(P)	Security	BS compl.	BS proc.	affects R/D
medium	high	high	yes	no	no

Table 5.31. **Overall assessment of the Virtual Image Cryptosystem**

3.5 Base-switching based system

Chang and Lin [24] use a recent lossless image codec for their encryption scheme. After partitioning the image into 3×3 pixels blocks these blocks are transformed involving the "base value" which is the difference between maximum and minimum of the pixel values in the blocks. The authors assume that this value stays in the range [1, 128] and the security of the entire system resides in encrypting this base value. Without having access to the base value the remaining data for each block cannot be decoded and the field containing the base value for the next block cannot be identified. Although it is claimed that there would be 128! mappings from the plain base value to the encrypted one, a brute-force attack only requires 128 guesses per block to test all possible base values. Taking into account several plausibility criteria, an attack could probably be mounted with much less effort. Regarding this observation, taken together with the fact that the compression scheme is not very effective and hardly ever used, the system does not seem to be very practical. Additionally this method is susceptible to known plaintext attacks because then the attacker knows the encrypted und the unencrypted version of the base value, so that the following bits can be interpreted without a chance of error.

4. Transparent Encryption

Macq and Quisquater [89, 90] introduce the term transparent encryption mainly in the context of digital TV broadcasting: a broadcaster of pay TV does not always intend to prevent unauthorised viewers from receiving and watching his program, but rather intends to promote a contract with nonpaying watchers. This can be facilitated by providing a low quality version of the broadcasted program for everyone, only legitimate (paying) users get access to the full quality visual data. This is meant also by the term "try and buy" scenario. Therefore, privacy is not the primary concern in such an environment.

Transparent encryption aims at protecting the details of the data which enable a pleasant viewing experience in an efficient manner. If this data are missing, the user is (hopefully) motivated to pay for the rest of the data. Another application area are preview images in image and video databases.

Transparent encryption can be implemented in various ways. The simplest approach is to use any arbitrary selective encryption scheme and to restrict the amount of data encrypted in a way that the visual information content is still perceivable. However, in many cases the quality of the data will be too bad to be useful in the context of transparent encryption. Consequently, techniques specifically tailored to the transparent encryption scenario have been developed.

As already briefly mentioned in section 3.1.1 (chapter 5), Macq and Quisquater [89] propose to use line permutations in the transform domain of a lossless multiresolution transform. The permutations are only applied in the region of the transform domain corresponding to fine grained details of the data. Droogenbroeck and Benedett [39] propose to encrypt bitplanes of the binary representation of raw image data, contrasting to the privacy focused approach discussed in section 3.1.3 they suggest to start with the LSB bitplane. With respect to JPEG encoded images, the authors suggest to encrypt sign and magnitude bits of medium and high frequency DCT coefficients (note that this is exactly just the other way round as compared to the Scalable Coefficient Encryption algorithm in section 1.1.1 where low frequency coefficients are encrypted only). Droogenbroeck [38] extends this latter idea to "multiple encryption" where different sets of DCT coefficients are encrypted by different content owners, and "over encryption" where these sets do not have an empty intersection (i.e. coefficients are encrypted twice or even more often). Also spatial selective encryption is discussed. The standalone encryption of motion vectors has found to be too weak for privacy focused encryption (see section 1.2.1). Bodo et al. [17] propose a technique called "waterscrambling" where they embed a watermark into the motion vectors of an MPEG stream. In particular, the suggest to DCT transform the motion vector field and to add a watermark using a robust (secret) spread spectrum technique. The high robustness of the scheme leads to the desired side effect that the video is distorted, only a legitimate user has access to the key and may descramble the motion vectors.

Transparent encryption may be implemented in the most efficient way in the context of scalable or embedded bitstreams. As already mentioned in section 1.4, transparent encryption is achieved in this environment by simply encrypting the enhancement layer(s). This has been proposed by Kunkelmann and Horn using a scalable video codec based on a spatial resolution pyramid [76,75] and by Dittmann and Steinmetz [35,36] using a SNR scalable MPEG-2 encoder/decoder (compare section 1.4). Yuan et al. [182] finally propose to use MPEG-4 FGS for transparent encryption, where contrasting to the previous approaches several enhancement layers are suggested to be used.

5. Commercial Applications and Standards

5.1 JPSEC — secure JPEG 2000

JPSEC will be a standard also known as ISO/IEC 15444-8, it is an extension to the JPEG-2000 standard. According to the current timeline [133, resolution 42] we can expect the FDIS (Final Draft International Standard) in December 2004, and the final standard in February 2005. Since JPSEC is not finalised at the time of this writing the following information might change.

JPSEC allows the content creators and providers to protect parts (called "zone of influence" ZOI) of a JPSEC file. It distinguishes between image data and non-image data (e.g. headers), and it allows manifold protection schemes: fragile integrity verification (using cryptographic hashes), semi-fragile verification (usually with the help of watermarks), source authentication, conditional access, secure scalable streaming and transcoding, registered content identification and of course confidentiality (by encryption or selective encryption). JPSEC allows multiple applications of the above protection methods on the same data, for example it will be possible during creation to first authenticate the image, watermark it, and then to encrypt a part of it. On the side of the recipient the process is then reversed. Some protection methods are predefined (such as encryption using AES), but others can be attached. The standardisation committee decided to set up a registry for such additional protection methods, such a registry allows the unique identification of the protection methods in any JPSEC bitstream.

5.2 IPMP — Intellectual Property Management and Protection

IPMP is a standard within the MPEG family which has been developed at first for MPEG-2 and MPEG-4. During the time some problems with this version of the standard have been found (there were interoperability conflicts with security and with flexibility). A second attempt, now part of MPEG-21 ("Multimedia Framework") tries to address the wishes of consumers and manufacturers, this new version was "back-ported" to MPEG-2 and -4, and can be found there as IPMP-X.

IPMP tries to create a way of interoperability for the deployment of content and applications, it distinguishes between 5 different communities: end-users or consumers, content providers, device manufacturers, service providers, and content authors. IPMP tries to meet the goals of all these groups by the creation of an extensive framework. One important part of this framework is the concept of the “IPMP tools”: they are modules that perform one or more functions like authentication, decryption or watermarking on an IPMP terminal, such modules are identified by an ID, they can be embedded in a bitstream, downloaded or acquired by other means. When a user requests a specific content, then the following steps are executed: the IPMP tools description is accessed, the relevant IPMP tools are retrieved, instantiated, initialised and updated during the content consumption [71, 98].

For working documents and information about the availability of finished standards please go to the official MPEG home page[2], for some open-source code which implements IPMP you might want to visit sourceforge[3] or use your favourite search engine.

5.3 MPEG, DVB & CSA

ETSI (European Telecommunications Standards Institute) says in their DVB cookbook [45, section 4]:

> In many cases DVB-based services will either be of the pay type or will at least include some elements which are not supposed to be freely available to the public at large. The term Conditional Access is frequently used to describe systems that enable the control over the access to programmes, services etc.
>
> Conditional Access (CA) systems consist of several blocks; among others, the mechanism to scramble the programme or service, the Subscriber Management System (SMS), in which all customer data are stored and the Subscriber Authorisation System (SAS), that encrypts and delivers those code words which enable the descrambler to make the programme legible.

In this book we focus on the actual image or video compression and encryption methods, so we leave out most of the surrounding framework and infrastructure. Several approaches exist to encrypt MPEG data, some comply to existing standards, others do not. Betacrypt is an example for the latter, it was based on a first version of Irdeto, but Betacrypt seems to be gone (due to various reasons, e.g. provider bankruptcy, or successful piracy). Other encryption schemes are Simulcrypt and Multicrypt:

Simulcrypt allows the use of multiple set-top boxes, each with its own CA system. The CA system (sometimes also called CAM = conditional access module) that received codes that this module recognises then performs the decryption. Multicrypt allows several CA systems to coexist in the same set-top box, usually using PCMCIA slots (CI = common interface) to plug them into the box, the MPEG data is sent to all (that is usually: both) modules in sequence. Some literature sometimes also mentions Equicrypt with references to [58] or [80], but it seems that this method did not evolve beyond project status[4].

The MPEG standards include an encryption mechanism which allows vendor-specific plugins, it is called "Common Scrambling Algorithm" CSA. CSA is built as a combination of a block cipher and a stream cipher. The block cipher uses a 64-bit key to generate 56 different 64-bit keys for the individual rounds to encrypt a 64-bit block. The stream cipher uses several LFSRs in parallel, each 10 bits long, the output is fed back via an S-box permutation. During decryption the data is first decrypted using the stream cipher and then by the block cipher.

The algorithm was secret for several years, it was build just in hardware. Some information about it can be obtained from two patents [13, 12]. Eventually some software leaked into the internet, the binary (FreeDec) has been reverse-engineered and so the code became public. Some sites on the internet contain information about it, an example is `http://csa.irde.to/` .

The idea is that every provider uses the same algorithm to encrypt the transmitted MPEG stream. Each provider can use its own algorithm to calculate the seed value for CSA. On the customer side the MPEG-receiver needs a "Conditional Access Module" from the respective provider which enables the decryption. Modern receivers contain an interface following the "Common Interface Standard", such an interface is basically a PCMCIA slot, and the access module (also called "CI module") is a smartcard within a PCMCIA adapter. The chip on the smartcard is responsible for the correct generation of the CSA seed values. This seed is also called "Control Word" or "Common Key".

In the following we list some commercial systems, some of them provide conditional access to video within the framework specified above, others do not, but they rarely publish detailed information about the inner workings of their products. However, most claim that their systems comply to the DVB standard.

Conax: a Telenor-offspring, see `http://www.conax.com/`

Cryptoworks: provided by Philips, see `http://www.software.philips.com/↩ InformationCenter/Global/FHomepage-NoXCache.asp?lNodeId=866` .

Irdeto: Irdeto Access, originally based in the Netherlands, is a subsidiary of the international subscriber platform group MIH Limited, which is a subsidiary of Naspers. See `http://www.irdetoaccess.com/`

Mediaguard: sometimes also referred to as "SECA" (Societe Europeene de Controle d'Acces), developed by Nagra France, a part of the Kudelski Group. see `http://www.nagra.fr/`

Nagravision: another descendant of the Kudelski Group, see `http://www.nagravision.com/`

Viaccess: a member of the France Telecom group, `http://www.viaccess.fr/index_solutions.html`

Videoguard: provided by NDS, see `http://www.nds.com/`

Using this two-stage encryption schemes gives two points of possible attacks: CSA, and the individual CI modules. Currently there exists no attacks on CSA, at least no attacks that are known in the public. Such an attack would be fatal: due to its nature it would allow to circumvent all the individual encryption schemes. This lack of an attack can be explained by two different reasons: first, the cipher is good enough to withstand all the attacks from cryptographers around the world. And second, nobody cares to attack the cipher. Since the cipher is used to protect multimedia content and since it is the commercial basis of some content providers one can assume that there are other forces which might have opposite interests. Therefore option two is unlikely.

The second class of attacks is directed against individual providers and their CI modules. The smartcards are handed out to the customers and therefore they must be considered to be in enemy territory. The providers take precautions against dissection of such smartcards, but in some cases pirates were successful.

5.4 DVD

DVDs can be protected using an encryption method called CSS, which was developed in 1996 by Matsushita. The sectors of such a DVD are encrypted using a chain of keys:

Title keys: these keys are used to protect the actual contents on the DVD.

Disc keys: these keys are used to encrypt/decrypt the title keys on the DVD.

Player keys: these keys are used to protect the access to the disc keys. The disc key is stored 400-fold in encrypted form on the DVD, each time encrypted with a different player key. Each DVD player manufacturer gets its own player key, the manufacturer must take care because it must not be compromised.

To access a title on a DVD the player has to use its own key to decrypt the disc key. This disc key is used to obtain the title key for a specific title on the DVD. Prior to this encryption sequence the DVD drive and the unit performing the CSS decryption have to authenticate to each other, to verify that the partner module complies with the DVD standards. DVD copy protection mechanisms are described in more detail in [16].

Suddenly in 1999 a software tool called "DeCSS" appeared on the net and spread like a virus. It was a tool which enabled to view, read and copy encrypted DVDs on computers. This feature was desperately needed because the

DVD player manufactures focused on mainstream operating systems but neglected others, like the emerging masses of Linux systems. Without official support the users had to build their own DVD players.

Soon a person was identified who should be responsible for this tool: Jon Johansen. In 2002 Johansen was accused of distributing a copyright circumvention technique, but the appellate court in Oslo, Norway, confirmed the ruling of the first instance court that he is not guilty. Some, for instance the US film industry, claim that Johansen is the author of DeCSS, others say that he was just a front man. Details can be found in a document floating around in the internet titled "The Truth about DVD CSS cracking by MoRE and [dEZZY/DoD]" [5].

After the spreading of DeCSS the DVD CCA (Copy Control Association) was founded, this group is now responsible for the licensing of CSS, details can be obtained from their web-pages, e.g. `http://www.dvdcca.org/css/`.

Besides the use of DeCSS to crack CSS, CSS also contains some weaknesses which allow a brute-force attack to get access within a reasonable time. The first weakness is its short key, 40 bits, this is way too short for current cryptosystems. See as an example the web site `http://www.distributed.net/`: The RC5 56-bit challenge was completed within 250 days, 40 bits are a $\frac{1}{65536}$ of work. Also take into account that the computing hardware advanced tremendously: in one formulation Moores law predicts that the processing power doubles every 18 months, this also means that the expected time to crack a cipher by brute-force is reduced by a factor of 2. RC5-56 was completed in 1997, this means that the processing power is 16-fold in 2003 and the expectations are that in 2006 the number will be 64.

Another weakness is the use of LFSRs, depending on the actual attack the knowledge of 5 or 6 bytes of output of the LFSRs is sufficient to obtain the key. Stevenson shows another attack, this time the target is the hash of the disk key, with an attack complexity of 2^{25} [147], recall from above that a stupid brute-force attack has complexity 2^{40}. An overview and a collection of these weaknesses was written by Greg Kesden[6].

5.5 Other commercial products

This section includes companies and products which do not suit into the categories above, e.g. because such a product scrambles *analog* video data, most products are for CCTV purposes. The information provided here cannot be complete since companies and products come and go. The information is provided as-is and should not be considered a recommendation.

- Mel Secure Systems Ltd provides a product "Imagelock" for video encryption/scrambling.
  See `http://www.melsecuresystems.co.uk/imagelock.html`

- SLD Security & Communications sells TeleGuard for video transmission via radio.
  See http://teleguard.biz/products/videotx/digitalradio.htm

- Verint seems to sell video systems from different producers, including CCTV systems with encrypted wireless transmission.
  See http://www.verint.com/video_solutions/

- Ovation Systems produce various video encryption or scrambling systems, online at http://www.ovation.co.uk/Products/products.html

- Tango Systems, Inc. produces systems for video transmission.
  See http://www.trangosys.com/products/Overview.cfm

- and many more ...

Notes

1 http://www.cast.uni-linz.ac.at/Research/DIP/ImageEncryption/bernoulli.html
2 currently at http://www.chiariglione.org/mpeg/
3 http://sourceforge.net/projects/openipmp/
4 http://www.cordis.lu/infowin/acts/rus/projects/ac051.htm
5 from e.g. http://www-2.cs.cmu.edu/%7Edst/DeCSS/MoRE+DoD.txt
6 available at http://www-2.cs.cmu.edu/%7Edst/DeCSS/Kesden/