https://doi.org/10.2352/ISSN.2470-1173.2018.07.MWSF-371

Hybrid Image Encryption

Martin Steinebach, Huajian Liu, Richard Stein, Felix Mayer

Abstract

Partial image encryption has several advantages compared to standard encryption in the multimedia domain. It preserves structural information of the media files and allows viewing the files in their encrypted state. The drawbacks of partial encryption are the lower security and the file size increase. We propose hybrid encryption, where the most significant frequency bands are strongly encrypted and the other bands are encrypted in a coding-friendly manner to achieve the best compromise between security and file size. We apply this concept to a local encryption approach where facial parts of photographs are recognized and encrypted to achieve a privacy-preserving state of a photo which can efficiently can be decrypted if required.

Context

Local encryption of images is a special form of partial image encryption. Here only a region of an image is encrypted while the rest remains unaltered. This can be necessary when persons on photos are anonymized. This can also be done by blurring or by putting a black box over the face of the person, but the advantage of partial encryption is its reversibility: With the help of a secret key the area can be decrypted and the original photo is available again. In contrast to standard encryption, the image can still be displayed by standard applications as the encryption is executed solely on the visual data stored in the image file, not on the file structure or meta-data.

The most important application for this is privacy-preserving forensics, where persons on a photo are anonymized and the actual faces can only be accessed when it is likely that the persons have been involved in a crime. Partial encryption is an improvement over keeping different copies of the same image with and without faces, as is requires less organizational overhead. Partial encryption is a family of algorithms for various applications, also known as selective encryption. “Partial” can mean two different things in this context:

Data partial only a subset of all elements of an image is encrypted (see the work of Goel et al. [1] for some recent approaches), for example the most significant bits of a raw image or low frequency coefficients of an JPEG file [2]. As a result, the whole image looks encrypted while only a fraction of the data representing the image is actually encrypted.

Spatial partial only a subsection of the image is encrypted, usually a selected region. This approach is less common but has been proposed e.g. for selective quality reduction of images for pre-sale copyright protection. In [3] only areas showing human skin are encrypted. A typical partial encryption algorithm combines a selector and an encryption algorithm. The selector decides what to encrypt, the actual encryption is then executed in a standard manner.

Related Work

Multimedia partial encryption is known for images, video and audio. Applications range form light weight security to media-preview concepts. In the following we list three algorithms as examples of the state of the art in partial image encryption. There are also other media-specific encryption methods not to be confused with partial encryption: visual encryption is basically a visual one-time-pad where image data is obfuscated be distributing it on two slides and de-obfuscated by overlaying one slide with the other. Robust image encryption aims at obfuscating an image in such a way that the obfuscation is reversible even after lossy compression of the obfuscated image.

In ”Partial Encryption of Compressed Images and Videos” [6], a method for partially encrypting JPEG images is presented. For this, the coding method is replaced by ”quadtree” or quaternary tree compression. A quaternary tree is a tree structure in which each node has four leaves. In each leave, all blocks have the same values. So the picture is divided into four equal parts. Each of these parts is split again until the values in that part are the same. In the presented method, only the tree structure is encrypted andthe values of the blocks remain unencrypted. Without the structure it is not possible to determine the position of the blocks and thus a reconstruction of the image is not possible. However, color and brightness of the blocks are known. Due the the approach, the encrypted images are not compatible with standard JPEG image viewers.

”Techniques for a selective encryption of uncompressed and compressed images” [2] describes a method for partially encrypting compressed JPEG images. For this, the Huffman-coded AC coefficients are encrypted. These can only be encrypted if they are nonzero, because all zero coefficients are combined during JPEG compression. For all other coefficients, the code contains a code marker containing the memory needed for the coded coefficient. In the method, the additional code bits are encrypted and the code marker is preserved. As a result, the original coefficient for encrypted coefficients can be limited to one area. This method is compatible with the JPEG format. However, no image sections can be encrypted.

”Selective encryption of human skin in JPEG images” [3] describes a method to partially encode human skin in JPEG images. It is based on the previous method, but limits the encryption to blocks where human skin can be seen. For this, the blocks are converted into their chrominance values. By means of these chrominance values, it is possible to decide whether the block contains the color of human skin. If so, the block’s AC coefficients are encrypted. This method is compatible with the JPEG format and enables the encryption of image sections.

Local Partial Encryption

We propose a novel partial encryption approach in this section to encrypt faces locally, which is compatible with JPEG compression standard [5]. The faces in JPEG images are assumed to be given by a surrounding rectangle which can be determined by face detection technology. Face detection is beyond the scope of this paper, so it will focus on the partial encryption of the local regions containing faces in the following.

In contrast to [2], the proposed approach encrypts the quantized DCT coefficients before entropy encoding instead of the encoded DCT coefficients. Encryption before encoding achieves a full encryption of each coefficient, which is therefore of security advantages, while the operation on the encoded coefficients reduces the encryption space of each coefficient, because it has to keep the same encoding size. However full encryption also results in bigger JPEG files, because the encrypted coefficients are pseudo-randomized bits and hence they are compressed less efficiently in the subsequent entropy encoding. Therefore, in addition to full encryption, an encoding-friendly variant is also introduced to avoid the increase of JPEG file size, which behaves as if the encryption would be executed on the quantized and encoded coefficients. Furthermore, for good trade-off between security and compression, a hybrid encryption approach is introduced, which encrypts significant coefficients fully while encrypting other coefficients encoding-friendly.

To prepare the encryption, the coefficients are extracted from JPEG blocks, which are inside the rectangles surrounding faces. Afterwards the coefficients are transformed into a bit-stream. The way of this transformation defines the encryption schemes which are introduced in the following subsections. The bit-stream is encrypted with *Ke* using a symmetric encryption method, e.g. `AES`, and the encrypted bit-stream is transformed into coefficients again. These encrypted coefficients are written at the place of the extracted coefficients, such that the faces become irrecognizable.

Full Encryption In the full encryption scheme, all DCT coefficients are encrypted in full length p, i.e. the upper bound of the quantized DCT coefficient precision which is determined by the sample precision. In baseline JPEG, for instance, the sample precision is 8 bits and the corresponding maximal precision of quantized DCT coefficients is 11 bits, i.e. $p = 11$[5]. Therefore, each coefficient is first represented with a p-bit stream. Then the coefficient bit stream is aligned in a byte array, which is subsequently encrypted using a symmetric encryption method. From the encrypted byte array every p bits are extracted and assigned to each coefficient sequentially.

Table 1 lists an encryption example of 3 coefficients in baseline JPEG. The bit-stream contains the 11 bits of the original coefficients. The encrypted bit-stream is interpreted as the two's compliment of each encrypted coefficient. The additional required bits corresponds to the increase of the length of the code word needed to encode the coefficients after encryption. For example, 4 bits are originally needed to encode the coefficient "4". After encryption, the coefficient value is changed to "-170", which requires 8 bits for encoding. Hence, 5 more bits are required for encoding. The worst case occurs in case of zero coefficients, which need no encoding space before encryption. After encryption, in addition to the encode word for the encrypted coefficient, an 8-bit code marker has to be added for each new non-zero coefficient. In ta-

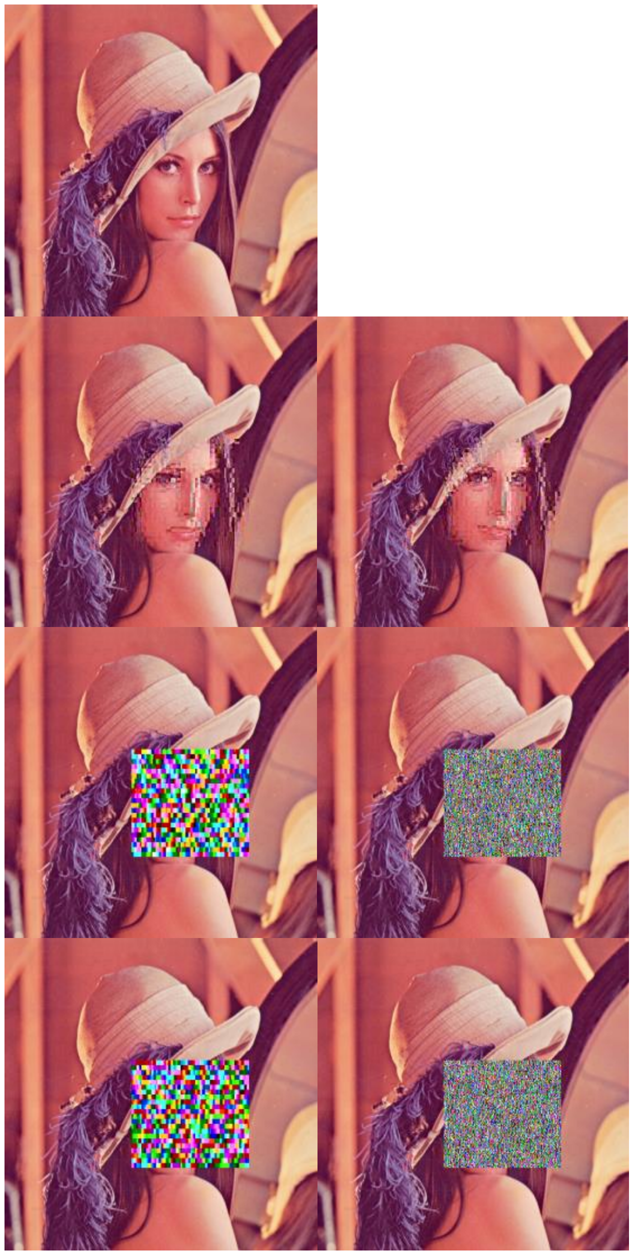

Figure 1. *Visual impact. First line: Original. Second line: Coding-friedly encryption with 8 (left) and 64 (right) coefficients. Third line: Full encryption with 2 and 32 coefficients. Fourth line: Hybrid encryption with 2 and 32 coefficients.*

ble 1 , the coefficient “0” becomes “-228” after encryption, which requires 8 bits for encoding and a 8-bit marker has to be added. Thus, 16 additional bits are required due to the encryption.

Table 1: Example full encryption

Origin coefficients	105	4	0
Bit-stream	00001101001	00000000100	00000000000
Encrypted bit-stream	10000111101	11101010110	11100011100
Encrypted coefficients	−963	−170	−228
Add. required bits	3	5	16

Encoding-Friendly Encryption The goal of encoding-friendly encryption is to encrypt the faces without increasing the image size. This is done by limiting the encryption result to coefficients with the same encoding length, which leads to a similar result as encrypting the coefficients after the entropy encoding step [2],[3]. Since the DC coefficients are encoded in a different way and their coding rely on previous DC coefficients, the following only applies to AC coefficients.

For each coefficient the encoding process is emulated and the result is aligned in a byte array and encrypted. From the encrypted array the same number of bits are extracted for each coefficient as in encoding and a decoding is simulated to retain the coefficients. Thus, each coefficient are encrypted into a coefficient value which requires the same code size, as shown in table 2. Subsequently, the encrypted JPEG image size will remain the same as the original one.

Table 2: Example code-friendly encryption

Origin coefficients	105	4	0
Bit-stream	1101001	100	—
Encrypted bit-stream	1000011	111	—
Encrypted coefficients	67	7	0
Add. required bits	0	0	0

Hybrid Encryption The hybrid encryption scheme combines the full encryption and the encoding-friendly encryption. The DC and the most significant AC coefficients, which contain the great majority of the visual content, are encrypted with full encryption, while all other coefficients are encrypted encoding-friendly.

For each coefficient, the bits are extracted according to the applied encryption scheme. These bits are then aligned in a byte array and encrypted. The same number of bits are extracted from the encrypted array for each coefficient and then converted into the encrypted coefficient value. Table 3 gives a hybrid encryption example, where two small coefficients “4” and “0” are encrypted using encoding-friendly scheme, while the big coefficient “105” is fully encrypted.

Implementation

Our local partial encryption approach encodes the quantized JPEG coefficients. To process the image and manipulate the coefficients, the libjpeg library is used. It allows accessing the coefficients in the blocks of the components (brightness, luminance and chrominance).

Table 3: Example hybrid encryption

Origin coefficients	105	4	0
Bit-stream	00001101001	100	—
Encrypted bit-stream	10000111101	111	—
Encrypted coefficients	−963	7	0
Add. required bits	3	0	0

From an image, all blocks of the region to be encrypted are selected. From these blocks, depending on the safety parameters, the first k coefficients are extracted from each block. These are symmetrically encrypted with OpenSSL and the result is stored as coefficients in the JPEG image. OpenSSL offers several symmetric methods, so one of them can be chosen. However, the coefficients must first be transformed into a byte array to be able to encrypt them because this format is expected by OpenSSL. By encrypting an equal sized array is created, except for up to eight padding bytes.

In full encryption, the coefficients should be completely encrypted. Therefore, all required bits must be included in the encryption. The coefficients occupy 16 bits as data type ’short’, but require fewer bits in storage. Let x be the number of actual storage bits needed, which is 11 in the JPEG standard implementation. To not wast storage space, it is necessary that not to use fill bits expanding the 11 bit to 16 bit for more convenience during encryption. Instead, they are written consecutively in memory and form a bitstream. After encryption, again x bits per coefficient are read from the byte array and the sign is extended. This allows the encrypted coefficients to be substituted for the original coefficients. Additional data, such as the padding bytes required by block ciphers, is stored in the metadata.

Encoding-friendly encryption encodes the coefficients into code bits specified by the JPEG format which are utilized during preparation of the entropy encoding. These bits are then combined into a byte array and this is encrypted. From the resulting array, code bits of the same length are extracted for each coefficient. These are mapped to the corresponding coefficient replacing the existing one. This is possible because the mapping of the coefficients to their code bits and the possible lengths is surjective.

Hybrid encryption combines the two methods by first retrieving all coefficients. Subsequently, the first k coefficients are fully encrypted and all remaining coefficients encoded coding friendly. Again, a byte array is created by flushing the 11-bit coefficients to be fully encoded and the variably long additional code bits of the remaining coefficients.

Visual Security

In this section we show the impact of the different encryption schemes and their parameters on the visual quality (or the level of distortion) of the protected images.

Figure 1 shows examples for Lena. One can see that the encoding-friendly encryption only has a small impact on the image. Visual recognition by a human is hardly impaired. Full encryption has a huge visual impact even with only 2 coefficients encrypted. Hybrid encryption looks similar to the human observer.

But when attacked by simply replacing the encrypted coefficients with zeros as suggested in [4], full encryption of few coefficients shows its weakness as shown in figure 2 (a). The borders

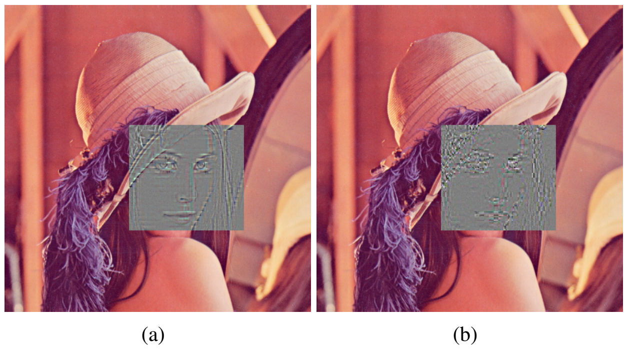

Figure 2. *(a) Example of coefficient removal with full encryption of 2 coefficients. The borders of the face are easily recognizable, (b) With hybrid encryption, the replacement attack has a smaller impact.*

of the face are made visible again by this attack, helping to identify the encrypted face. Here the advantage of hybrid encryption helps to protect the privacy of the encrypted face. As one can see in figure 2 (b) the borders are much more distorted, making a recognition hard.

One noticeable aspect of the impact of encoding-friendly encryption of JPEG blocks is the image resolution. The already comparatively small impact to the visual quality is even smaller when a high-resolution image is encrypted in this way. It does not add significant noise to the image, but rather looks like a chaotic blurring. When the number of displayed pixels is smaller than that of the stored pixels, the display image will be the result of a downscaling algorithm. This will further reduce the impact of the encryption.

File Size

File size and the strength of visual encryption relate to each other. The more coefficients are encrypted, the worse the entropy encoding of JPEG works, resulting in increased file size. In Figure 3 we provide a number of examples. All three encryption strategies are shown. As to be expected, full encryption (FE) and hybrid encryption (HE) result in the biggest files. When encrypting all coefficients in this way, file sizes increase up to a factor of 2.5 if the face regions take up a significant part of the image as with Lena. Code-friendly does not influence the file size. Full and hybrid encryption share the same size as in both cases the stated number of encrypted coefficients are fully encrypted. Only the remainder is code-friendly encrypted with hybrid encryption without any size increase.

Figure 4 shows a comparison of the file size increase for full encryption or hybrid encryption. For this purpose, faces (as examples for local sections) were recognized and encrypted on 100 images. The photos are of people who perform activities, and are mostly no portrait photos. Not all faces were detected and also other parts of the pictures were encrypted due to face detection failures. However, for this evaluation the reliability of face detection did not matter.

Here, not absolute, but relative ratios are used after the encryption. On the x-axis is the number of coded coefficients and on the y-axis the total increment is listed as a multiplicative factor. So if there's a dot at $(8,2)$, there's a picture whose encryption,

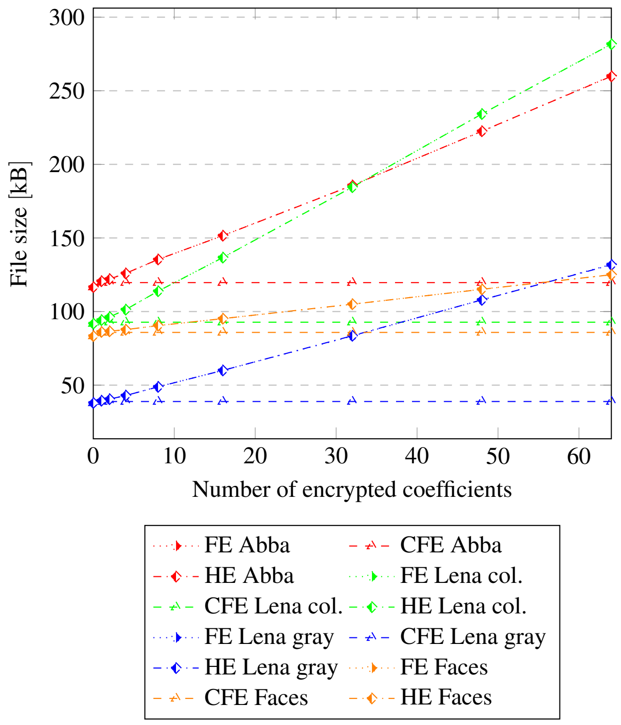

Figure 3. *File size depends on encrypted coefficients. FE: Full Encryption, CFE: Code-friendly encryption, HE: hybrid encryption. HE and FE share the same size*

with 8 coefficients per block, requires twice the size. In addition, a straight line is drawn, which represents the total increase of all files. It can be seen that only in a small part of the images, a large increase in size arises.

In the figure, one image has a particularly large growth. This image is a low-quality moderately-resolved portrait and originally required very little space (45 kB). For other images, the file size is initially reduced. These images have additional data beyond the end-of-file marker, which is removed during encryption and re-encoding.

An encryption of 32 coefficients per block is relatively secure against attacks. Here, the file size is increased more than fourfold and less than twice with only eight images.

Thus, the detected faces in these images can be securely encrypted when the total size is doubled. It can be seen that most pictures have a moderate increase in size.

Processing Time

As in the previous section, the various encryption methods are compared by applying them to different images. The same pictures are used for this. The runtime measurement is based on an Intel Core i7 Q740 processor with 1.74 GHz and 16 gigabytes memory . Figure 5 shows that hybrid encryption requires the most time and coding-friendly encryption the least. The runtime of each encryption type increases as more coefficients are encrypted. However, the time required for full encryption approaches that of hybrid encryption as the number of encrypted coefficients increases. In addition, the duration of the face recognition has a large impact on the total duration. In the examined

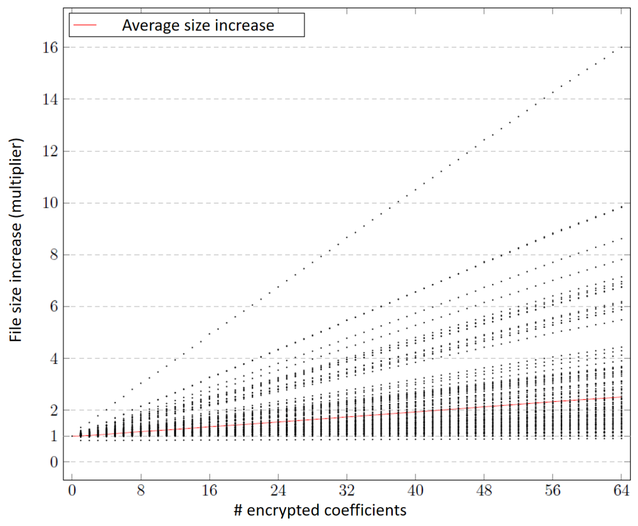

Figure 4. *Comparison of size growth*

images it requires between 0.8 and 8 seconds while the encryption takes between 0.4 and 25 seconds. Overall, face recognition and encryption in the examined images take between 1 and 40 seconds. At this time for the encryption of an image, the method can be practically used with all types of encryption.

Discussion

The evaluation shows that as expected the file size does not increase with code-friendly encryption. Therefore combining full encryption for the content-relevant low frequencies with code-friendly encryption for higher frequencies does not influence the increase of the file size due to partial encryption. Still, the increase by full encryption of the low frequency bands enlarges the files significantly. This will only be acceptable if keeping the structure of the JPEG file intact is of significance. Otherwise a post-JPEG full encryption is much more efficient.

Local encryption is used in some scenarios like privacy-protection or in previews using scarring strategies. Privacy protection may be the most likely use case for local hybrid encryption. Here using an approach providing the best compromise between secure and cost-efficient encryption can be of importance. It allows to work with image files featuring encrypted face regions which can be decrypted if necessary.

We did evaluate the processing time, but it must be mentioned that currently we only have a proof-of-concept implementation. Integrating the partial encrypted directly into an JPEG encoder usually will speed up the process to the point where full and partial encryption applied on many frequencies will not be distinguishable. A comparable experiment is shown in our previous work on mp3 partial encryption [7].

Of course there are alternatives to this approach on local image encryption. One simple way to produce a similar effect would be to cut out relevant regions, encrypt a JPEG copy of the region with a standard algorithm and place the resulting data at the end of the JPEG file after the end-of-JPEG code or store it in the JPEG meta-information. The region would be replaced by a flat (black) area. Most image viewers would ignore all data after the end of the JPEG data and therefore would show the image with the removed area without any problem. If necessary, the encrypted areas could be decrypted and placed on the original position again. We did not evaluate this strategy. It could lead to problems due to lost encrypted areas due to file corrections during copying data with image software. This would be a local standard encryption.

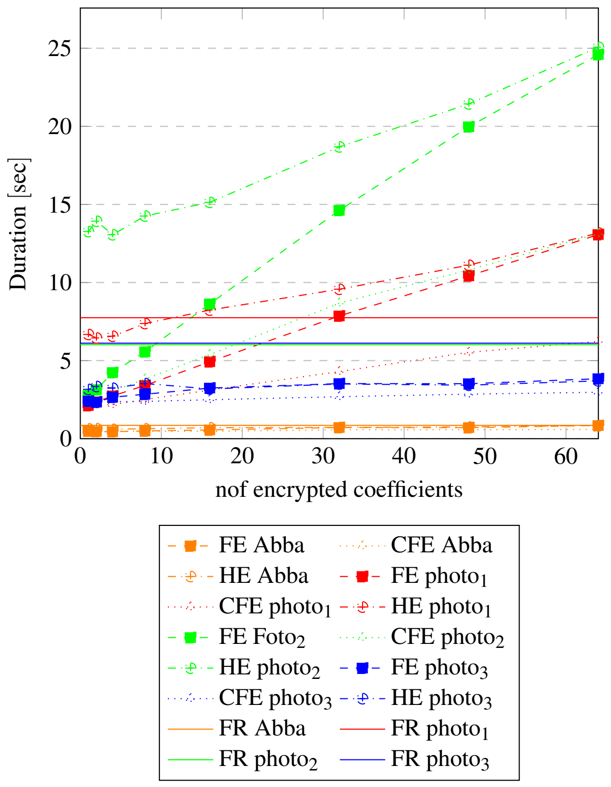

Figure 5. *Processing time in relation to number of encrypted coefficients. FE: Full Encryption, CFE: Code-friendly encryption, HE: hybrid encryption, FR: face recognition*

Finally we want to come back to the initial motivation of our hybrid encryption approach, the privacy protection of detected face regions. Here the full encryption part of the hybrid encryption will obviously have the greatest visual impact. For a scenario without attacks, local partial full encryption will suffice. But against automated attack removing the fully encrypted coefficients and then matching the remainder with face detection, the blurring of the code-friendly encryption is another line of defense which should increase error rates.

Summary and Conclusion

We introduce hybrid encryption, an approach in the area of local partial encryption, combining full and code-friendly encryption. While full encryption is memory-expensive, code-friendly encryption is rather weak security wise. Here the full encryption is used to protect relevant coefficients while code-friendly encryption is used in addition to increase the security of full encryption against replacement attacks. As code-friendly encryption is transparent with respect to storage space, this can be seen as a free increase of security .

Acknowledgments

This work was supported by the German Federal Ministry of Education and Research (BMBF) as well as by the Hessen State Ministry for Higher Education, Research and the Arts (HMWK) within CRISP.

References

[1] A. Goel and K. Chaudhari. 2016. Median based pixel selection for partial image encryption. In 2016 Sixth International Conference on Image Processing .eory, Tools and Applications (IPTA). 15. DOI:h.p://dx.doi.org/10.1109/IPTA.2016. 7820931

[2] Marc Van Droogenbroeck and Raphael Benede.. 2002. Techniques For ASelective Encryption Of Uncompressed And Compressed images. h.p://citeseerx.ist.psu edu/viewdoc/citations; jsessionid=521EAD584D0E0042E4FE39A712E518E7?doi=10.1.1.19.4661

[3] J. M. Rodrigues, W. Puech, and A. G. Bors. 2006. Selective Encryption of Human Skin in JPEG Images. In 2006 International Conference on Image Processing. 19811984. DOI:h.p://dx.doi.org/10.1109/ICIP.2006.312886

[4] Martina Podesser, Hans peter Schmidt, and Andreas Uhl. 2002. Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments. In in Mobile Environments, 5th Nordic Signal Processing Symposium, 1037.

[5] William B. Pennebaker and Joan L. Mitchell. 1992. JPEG Still Image Data Compression Standard (1st ed.). Kluwer Academic Publishers, Norwell, MA, USA.

[6] H. Cheng and X. Li. Partial encryption of compressed images and videos. Trans. Sig. Proc., 48(8):24392451, Aug. 2000. ISSN 1053-587X.

[7] Steinebach, Martin and Berchtold, Waldemar. (2017). MP3 Partial Encryption for DRM. Electronic Imaging. 2017. 28-35. 10.2352/ISSN.2470-1173.2017.7.MWSF-322.

Author Biography

Dr. Martin Steinebach is the manager of the Media Security and IT Forensics division at Fraunhofer SIT. From 2003 to 2007 he was the manager of the Media Security in IT division at Fraunhofer IPSI. In 2003 he received his PhD at the Technische Universität Darmstadt for this work on digital audio watermarking. Since 2016 he is honorary professor at Technische Universität Darmstadt.