Post-Compression Multimedia Security

by
Dipl.-Ing. (FH) Dipl.-Ing. Andreas Unterweger

Cumulative dissertation submitted to the
Faculty of Natural Science, University of Salzburg
in partial fulfillment of the requirements
for the Doctoral Degree *Dr. techn.*

Thesis supervisor
Univ.-Prof. Mag. rer. nat. Dr. rer. nat. Andreas Uhl

Department of Computer Sciences
University of Salzburg
Jakob-Haringer-Straße 2
5020 Salzburg, Austria

Salzburg, November 2014

Abstract

Image and video processing after (lat. *post*) compression, i.e., direct data modification on compressed multimedia bit streams, is a complex, but less time-consuming alternative to classical re-compression-based multimedia signal processing. Despite its advantages in terms of speed, very few post-compression approaches have been proposed in the literature. This includes the subject area of multimedia security which imposes additional use-case-specific processing constraints that make post-compression approaches a better fit than classical methods.

This cumulative thesis unites a number of book chapters, journal articles and conference papers on post-compression multimedia security, particularly from the areas of region-of-interest encryption, watermarking and transparent encryption. These publications contribute new post-compression encryption approaches for JPEG, H.264 and H.265 as well as a post-compression watermarking approach for H.264. In addition, they address the facilitation of post-compression region-of-interest encryption in scalable H.264-based video coding.

In addition, this thesis covers encryption-related topics like region-of-interest signalling, face detection and surveillance systems. The corresponding publications contribute initial work on region-of-interest signalling for JPEG, a modification of the object detection approach proposed by Viola and Jones for faster processing, and a framework for fully integrated face encryption for existing surveillance systems with minimal system modification requirements.

Kurzfassung

Bild- und Videoverarbeitung nach (lat. *post*) der Kompression, also die direkte Datenmodifikation auf komprimierten Multimediabitströmen, ist eine komplexe, aber weniger zeitaufwändige Alternative zu klassischer rekompressionsbasierter Multimediasignalverarbeitung. Trotz seiner Vorteile in Bezug auf Geschwindigkeit sind nur wenige solcher Ansätze in der Literatur vorgeschlagen worden. Das schließt das Teilgebiet der Multimediasicherheit ein, das zusätzliche, anwendungsfallspezifische Verarbeitungseinschränkungen auferlegt, für die Postkompressionsverfahren besser geeignet sind als klassische Methoden.

Diese kumulative Dissertation vereint mehrere Buchkapitel, Zeitschriftenartikel und Tagungsbeiträge zum Thema Postkompressionsmultimediasicherheit, im Speziellen aus den Gebieten der Bildbereichsverschlüsselung, der Wasserzeichen und der transparenten Verschlüsselung. Diese Publikationen steuern neue Postkompressionsverschlüsselungsverfahren für JPEG, H.264 und H.265 sowie Postkompressionswasserzeichenverfahren für H.264 bei. Zusätzlich behandeln sie Hilfsmaßnahmen zur einfacheren Postkompressionsbildbereichsverschlüsselung in skalierbarer H.264-basierter Videokodierung.

Außerdem deckt diese Dissertation verschlüsselungsverwandte Themen wie Bildbereichssignalisierung, Gesichtserkennung und Überwachungssysteme ab. Die entsprechenden Publikationen steuern erste Bildbereichssignalisierungsarbeiten für JPEG, eine Modifikation des Objekterkennungsansatzes nach Viola und Jones zur schnelleren Verarbeitung und ein Software-Framework zur voll integrierten Gesichtsverschlüsselung für bestehende Überwachungssysteme mit minimalen Systemmodifikationsanforderungen bei.

Acknowledgments

First and foremost, I want to thank my thesis advisor, Andreas Uhl, for his exceptional supervision. His constant support and availability for questions, comments and research directions make this thesis what it is in terms of both quality and depth.

Second, I want to thank my colleagues from the *Wavelab* group – both former and current ones. Without them, many discussions would not have taken place, many ideas would not have come up and many great suggestions would not have been tried out. I want to especially thank Heinz Hofbauer, Luca Debiasi, Rudi Schraml and Michael Gschwandtner for their ideas which contributed to this thesis as well as for their patience with me and my verbose nature.

Third, I want to thank Dominik Engel for his constant support during our joint research project. Our discussions greatly improved the papers that make up this thesis.

Finally, I want to thank my family and friends for their patience during my Ph.D. studies. With my priorities being shifted and my availability (for activities with them) being limited, they were always sympathetic and encouraged me to pursue my studies. Thank you.

If you are still actually reading this and feel that the witty quote that used to be present in the Acknowledgments section of my previous theses is missing, please make sure to read on.

This thesis has been funded in part by FFG Bridge project 832082.

Contents

1. Introduction

"On the page it looked nothing. The beginning simple, almost comic. [..] This was no composition by a performing monkey!" – Antonio Salieri, *Amadeus*

This thesis covers three main subjects, all of which are part of the broader subject area of post-compression multimedia security. Since even the latter is only a sub-field of post-compression multimedia signal processing, itself a highly specialized research field, the aims and challenges of both are briefly introduced in separate sections before the three main subjects are described.

1.1. Post-compression multimedia signal processing

Multimedia signal processing deals with reading and/or manipulating multimedia data like audio, image and video signals [Gonzalez and Woods, 2007]. Among the most intensely researched types of signals at the time of writing are images and videos, which are the main focus of this thesis. Typically, these signals are compressed in order to reduce storage space requirements [Chrysafis et al., 1999, Wiegand et al., 2003, Sullivan et al., 2012]. For processing, it is therefore necessary to either decompress the data beforehand or to operate on the compressed data directly.

While working on decompressed data is trivial, the decompression process itself may be time-consuming [Schwarz et al., 2007, Sullivan et al., 2012, Sullivan et al., 2013]. Furthermore, it induces the inherent need to re-compress the processed data to keep it in the same format that it was in before processing, forcing additional time and potentially space overhead [Vetro et al., 2003, Xin et al., 2005]. In the case of lossy compression, this further reduces the data quality by adding additional compression artifacts [De Cock et al., 2010].

In order to avoid this, operating on compressed data, i.e., post-compression signal processing, is an alternative. While it does not necessarily require any decompression or re-compression operations, it is in general more difficult to design and implement. This is mainly due to the complexity of multimedia data formats and the data dependencies which enable state-of-the art compression performance in the first place [Woods et al., 2005, Schwarz et al., 2007, Sullivan et al., 2013]. Furthermore, post-compression signal processing algorithms are typically inherently format-dependent.

Nonetheless, the benefit of faster processing often outweighs the aforementioned limitations and makes it worthwhile to invest time in addressing the challenges of post-compression multimedia signal processing. This affects the subject area of post-compression multimedia security in particular, where processing has often to be done in real time [Schulzrinne et al., 1998, Westwater and Furth, 1997]. In such cases, adding time-consuming decompression and re-compression steps is nearly impossible, making the direct manipulation of the compressed data a much more viable, if not the only possible, way of signal processing.

1.2. Post-compression multimedia security

The field of post-compression multimedia signal processing is highly branched, with post-compression multimedia security being one of its sub-fields. This thesis covers multiple subjects of this subject area, all of which share one dilemma: Although security-related processing of

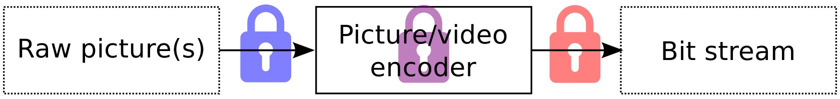


Figure 1.1.: Domains of operation for multimedia-security-related signal processing: Before (left: blue), during (middle: purple) and after (right: red) compression.

multimedia content is nearly always time-critical and therefore predestined to be performed directly on compressed data, there are no or only very few format-compliant post-compression approaches described in the literature, depending on the subject.

What adds to this is the fact that multimedia-security-related operations cannot only be performed on compressed or uncompressed data, but also on transformed or partially compressed data during compression. Multimedia-security-related signal processing approaches can therefore be categorized by the domain in which they operate, as illustrated in Fig. 1.1 (based on the taxonomy used in [Massoudi et al., 2008]):

- **Pre-compression** approaches operate on uncompressed data. They are equivalent to uncompressed signal processing as described above, sharing its advantages and disadvantages. In addition, they can be employed on completely uncompressed sources without requiring decompression or re-compression. However, such sources are typically very rare in the context of multimedia data due to their size.

- **In-compression** approaches have full control over the compression process and therefore have access to both, the compressed and the uncompressed data. Although this allows omitting decompression and re-compression operations, it requires modifications to the compression software. Not only is this often impractical or financially infeasible due to the implementation complexity, but it is potentially impossible when closed-source software is used. It is even more difficult for compression hardware.

- **Post-compression** approaches operate on compressed data. While it is possible that they decompress and re-compress the data in order to be able to operate on uncompressed data, the term is typically used to mean direct manipulation of the compressed data. It is therefore a special form of post-compression signal processing.

Although pre- and in-compression multimedia security approaches are by far more popular than post-compression approaches by the amount of relevant literature, they are not suitable for a whole group of applications: When uncompressed data is not available, like for third-party content such as movies, or images pre-compressed for transmission, pre-compression approaches cannot be used at all unless time-consuming decompression and re-compression is employed. Similarly, in-compression approaches require an infeasible amount of time for decompression and re-compression.

Conversely, post-compression approaches do require these operations at the cost of higher implementation complexity. Hence, they are suitable in particular for the following three subjects covered in this thesis:

- **Region of Interest (RoI) encryption** (obfuscation of parts of a picture or video): In one of the most common use cases – encrypting faces in surveillance videos for privacy – images are almost always pre-compressed by the camera for transmission to reduce bandwidth requirements.

- **Watermarking** (addition of identification information to a picture or video): When third-party content, e.g., a movie in form of a Blu-ray image, has to be watermarked, re-compression might either not be possible at all or too complex due to the data format.
- **Transparent encryption** (limited quality reduction of a picture or video): When copyrighted third-party content, e.g., a Television (TV) show to be broadcast, is transparently encrypted, re-compression (as an alteration) might not be allowed due to transmission channel constraints.

The aforementioned limitations can be broken down further into three conservation properties, which have to be considered for content that is pre-compressed (in the form of one input file or multiple input files). Borrowing from taxonomies of encryption [Massoudi et al., 2008] and watermarking [Stütz et al., 2013] approaches, the following three conservation properties can be applied to approaches from all three subjects:

- **Format compliance**: The output file or files can be decoded by a standard-compliant decoder, i.e., the processed content complies with the standard-defined format in terms of syntax and semantics.
- **Length preservation**: The size of the output file or files is exactly the same as the size of the input file size after processing, i.e., the length of the bit stream is preserved in total.
- **Structure preservation**: The composition of the compressed frame or frames (e.g., block partitioning and as Group of Pictures (GOP) structure), remains the same. This includes the length preservation of all bit-stream units and therefore entails the length preservation of the whole bit stream.

All approaches presented in this thesis are format-compliant. This is mainly due to the practical relevance of this property. Non-compliance could lead to unwatchable content, i.e., the inability of a standard-compliant decoder to decode the files, which is highly undesirable in most use cases. Furthermore, pre-, in- and post-compression approaches can be easily designed to preserve format compliance since the formats are well known and changes to any particular format can be checked for compliance if necessary.

For pre- and in-compression signal processing approaches, it would be possible to preserve structure without the length preservation requirement when using re-compression-based techniques, e.g., through special transcoding schemes [Pranata et al., 2004, De Cock et al., 2010]. Length preservation, however, is very hard to achieve without significant quality losses, since rate-control algorithms are typically not able to operate with one-byte accuracy [Pranata et al., 2004, Wang and Kwong, 2008, Li et al., 2014]. These problems are easier to solve with post-compression signal processing. Since the bit stream units are manipulated directly, any potential changes can be checked against length and structure restrictions before being made during processing.

However, post-compression signal processing approaches also have downsides. First and foremost, their design and implementation is more difficult than that of pre- and in-compression approaches. Second, small changes to the bit stream may cause large changes in the (eventually) decoded picture or pictures due to compression-induces dependencies. These are discussed in detail in Section 1.3. Finally, subject-specific issues arise, which are discussed in the following sections – one for each of the three subjects – including the context and a description of the corresponding subjects.

Figure 1.2.: RoI encryption example: The face as RoI in the left image is encrypted in the right image. The remaining image areas remain unchanged.

1.3. RoI encryption

Encryption serves the purpose of limiting access to data so that only authorized parties can retrieve, i.e., decrypt, the encrypted information [Schneier, 2000]. Selective encryption aims to do so by encrypting not all, but only some bits of the input data [Senior et al., 2005, Massoudi et al., 2008, Engel et al., 2009]. RoI encryption can be seen as a special case of selective encryption where one or more spatial RoI are encrypted while the rest of the picture stays intact, as depicted by example in Fig. 1.2.

One practical application of RoI encryption is privacy preservation in surveillance systems [Dufaux et al., 2006, Martin et al., 2008, Luo, 2010]. Faces of people captured by surveillance video cameras are encrypted so that an unauthorized user viewing the encrypted video footage is not able to identify them. Yet, it is still possible to view the unencrypted parts of the videos, which would not be possible with traditional selective encryption.

This allows, for example, a camera operator to see what is happening in real time without violating the privacy of the captured people. If a crime is committed, the camera operator can notify a security guard or the authorities. The latter can access the footage thereafter, i.e., decrypt the encrypted faces, and identify the responsible people. This information can then be used as evidence, if necessary, without violating the privacy of the people who have been captured before or after the crime has been committed.

RoI encryption in this thesis is always used in the video surveillance context described above. Related work on post-compression RoI encryption is relatively sparse (e.g., [Wu and Wu, 1997, Dufaux et al., 2004, Dufaux and Ebrahimi, 2008a]). In contrast, a high number of pre-compression (e.g., [Boult, 2005, Carrillo et al., 2009, Rahman et al., 2010]) and in-compression approaches (e.g., [Dufaux and Ebrahimi, 2008b, Ouaret et al., 2008, Tong et al., 2010b]) have been proposed. A short overview of existing approaches for the three formats which are most relevant for surveillance systems – Joint Photographic Experts Group (JPEG) [ITU-T T.81, 1992], H.264 [ISO/IEC 14496-10, 2005] and Scalable Video Coding (SVC) [ITU-T H.264, 2007] – can be found in [Auer et al., 2013] (Section 3.5), [Unterweger and Uhl, 2014a] (Section 3.9) and [Unterweger et al., 2015b] (Section 3.7).

One challenge of post-compression RoI encryption, in particular, is drift, i.e., when pixels outside the RoI (which are not intended to be encrypted) change their values as a side effect of

Figure 1.3.: Drift examples: In the original H.264-compressed frame (left), the face is encrypted (middle). The spatial drift on the collar and the concrete blocks in the background is due to inter-block dependencies between encrypted and unencrypted image regions. Future frames (right) exhibit additional temporal drift (e.g., the block beside the person's left eye) due to motion compensation in unencrypted image regions which use encrypted image regions as reference.

the RoI encryption approach. This is highly undesirable and has to be considered in the design of post-compression RoI encryption algorithms. Drift is due to data interdependencies caused by redundancy elimination during compression.

For example, when a block A and its neighboring block B have a similar texture, the (pixel or transform coefficient) values of B are most likely predicted from A to minimize redundancy during compression. When A is encrypted by a post-compression RoI encryption approach, the values of B will be changed due their dependency to the values of A during decoding. This is depicted by example in Fig. 1.3.

The three relevant formats for video surveillance – JPEG, H.264 and SVC – use multiple prediction techniques, causing different kinds of dependencies. These yield different types of drift which can be categorized as follows:

- **Spatial drift** affects neighboring blocks in the same frame. It is caused by dependencies due to intra(-frame) prediction.
- **Temporal drift** affects blocks in neighboring frames. It is caused by motion-compensation-related dependencies due to inter(-frame) prediction.
- **Inter-layer drift** (in scalable formats only) affects blocks in neighboring layers, i.e., layers which use data from their respective base layer by any form of inter-layer prediction.

Neither pre- nor in-compression RoI encryption approaches typically suffer from any kind of drift. Pre-compression approaches manipulate the uncompressed image directly, leaving the areas outside the RoI unmodified (e.g., [Boult, 2005, Carrillo et al., 2009]). In-compression approaches have full control over the encoder, i.e., they can actively prevent the encoder from using any data dependencies which would result in drift (e.g., [Dufaux and Ebrahimi, 2006, Tong et al., 2010a]).

Hence, the avoidance or minimization (depending on the use case) of drift is one of the primary goals for post-compression RoI encryption algorithms. As large amounts of drift make the unencrypted parts a video unwatchable, it is crucial for the video surveillance use case to contain the drift around the RoI. A complete absence of drift would, of course, be desired, although it is hard or impossible to achieve for some formats.

1.4. Watermarking

Adding a watermark to an image or video means embedding data about the creator(s), source(s) and/or other meta data of the image or video into the latter [Acken, 1998, Hartung and Kutter, 1999]. While a watermark does not prevent an attacker from illegitimately copying content, the existence of a (certain) watermark within the copied file can be used to prove ownership, intended use and other properties. In effect, it is a means of evidence in cases of copyright violation. The retrieval of the watermark data is referred to as extraction.

Typically, watermarks are categorized by their properties (adopted from [Hartung and Kutter, 1999] and [Cox et al., 2007]):

- **Capacity**: The average number of bits that can be embedded per image or frame.
- **Robustness**: The degree to which changes to the watermarked file do not impact the extraction. Fragile watermarks cannot be extracted after any minor change, while robust watermarks survive certain transformation types, like re-compression, scaling or cropping.
- **Perceptibility**: The visibility of the watermark to a human observer.
- **Extraction type**: The amount of information required to extract a watermark. Blind extraction can be performed without any additional knowledge, while non-blind extraction requires information about the input file and/or the embedding locations of the watermarks.

The only use case considered in this thesis is watermarking Blu-ray disks during production. Before Blu-ray disks are released, their content may leak in one of the production stages. In order to locate the stage (and potentially the person responsible) in which the content has leaked, a watermark is added to the disk before each stage. If a leak occurs, the stage it occurred in can be inferred from the existence of watermarks in the file. The watermarks of the affected stage as well as those of its predecessors can be extracted successfully, while no watermarks of the following stages can be found.

As explained above, these watermarks cannot prevent a leak, but they can be used as evidence once a leak occurred. Future actions may then be taken to assure that there are no leaks in the future. To make it as difficult as possible for an attacker (i.e., a person who is trying to leak the content unnoticed) to circumvent this form of leak detection, the watermark should be robust against a broad range of transformations like re-compression, scaling, cropping and changes in aspect ratio. When content leaks, conversions to different video formats and/or sizes must not eradicate the watermark. In addition, the watermark needs to be imperceptible for a human observer, obviously.

Imperceptibility depends on the amount and magnitude of changes required to embed the watermark, which affects capacity as a side effect. In post-compression watermarking, these changes may cause drift as described in Section 1.2. Although pre-compression (e.g., [O'Ruanaidh and Pun, 1998, Barni et al., 1998, Lin et al., 2011]) and in-compression approaches (e.g., [Su and Kuo, 2001, Meerwald and Uhl, 2012, Lin and Li, 2011]) can avoid drift by design, they are unsuitable for this use case. On the one hand, repeated re-compression would diminish the video quality with each production stage. On the other hand, Blu-ray watermarking requires structure preservation to avoid reediting menus, chapter marks and other meta data which relies on the position of certain elements in the video data [Blu-ray Disc Association, 2011].

This makes post-compression watermarking approaches the only practically relevant option for Blu-ray disk watermarking in the described used case. Unlike pre- and in-compression approaches, however, post-compression approaches need to address the issue of drift by design in order to avoid visible artifacts and thereby involuntarily perceptible watermarks.

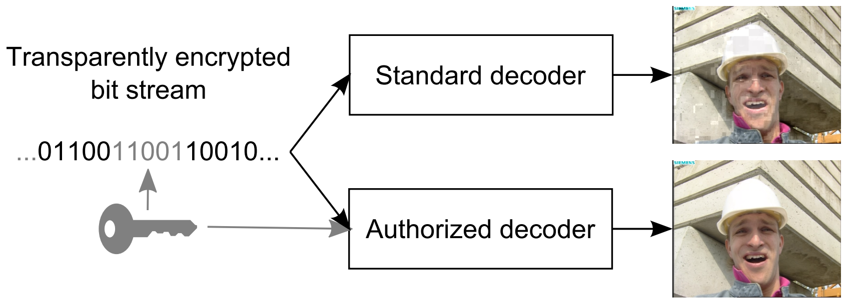


Figure 1.4.: Transparent encryption: A standard decoder creates a slightly degraded version of the partially encrypted original video, while an authorized decoder can use a key to fully decode it.

1.5. Transparent encryption

Transparent encryption aims to partially encrypt an image or video with the goal to simultaneously provide two versions of it [Macq and Quisquater, 1995, Stütz et al., 2010]: The unencrypted original for those who are authorized to decrypt it and a slightly degraded version for those who are not authorized. In contrast to full or RoI encryption (see Section 1.3), transparently encrypted images and videos can still be interpreted by humans, albeit with additional effort due to the quality degradation.

A typical use case for transparent encryption is Pay TV as illustrated in Fig. 1.4. A Pay TV provider sends out a transparently encrypted bit stream which can be received by both, paying and non-paying customers. The paying customers have an authorized decoder (bottom) which decrypts the encrypted bit stream parts and thereby reconstructs the original video. In contrast, non-paying customers with standard decoders (top) cannot decrypt the encrypted bit stream parts and receive a degraded version of the original video.

However, since the degradation is not severe, they can see the content of the original video, albeit with more difficulty due to the degradations. On the one hand, this prevents non-paying customers from enjoying the video, i.e., watching significant amounts of content without paying. On the other hand, the remaining content is sufficient to potentially persuade them to purchase access credentials and become paying customers, if the content is appealing to them.

In order to achieve these goals, two requirements have to bet fulfilled (adopted from [Engel et al., 2009] and [Hofbauer, 2013]):

- **Cryptographic security**: Despite the fact that only parts of the bit stream are encrypted, it must be very hard or impossible for an attacker to reconstruct the full original video without access credentials, i.e., without paying.
- **Quality assurance**: The degradation of the video has to be limited in order for it to be still watchable, i.e., to be of sufficient quality to promote the original content.

Both requirements are more difficult to fulfill when performing transparent encryption with post-compression encryption approaches. The main reason for this is drift, which has been introduced in Section 1.3. While drift is beneficial for transparent encryption in a sense that fewer bits need to be encrypted to achieve the desired degradations, the amount of degradation is harder to control and depends on the prediction mechanisms used in the input bit stream. This makes quality assurance harder.

In terms of cryptographic security, drift is not directly an issue. However, the more redundancy is eliminated during compression using prediction mechanisms, the fewer bits remain for encryption. This may lead to a situation where there are not enough bits left to assure cryptographic security while simultaneously keeping the amount of degradation at an acceptable level for non-paying customers.

Again, as for RoI encryption (see Section 1.3), this issue is practically nonexistent for pre- and in-compression approaches due to the absence of (unintended) drift. However, broadcasting limitations like length preservation (and often structure preservation due to copyright issues) make post-compression transparent encryption the only viable options. Thus, it is worth addressing the problems of cryptographic security and quality assurance in the presence of drift.

2. Contributions

"I'm just a simple man, trying to make my way in the universe." – Jango Fett, *Star Wars: Episode II - Attack of the Clones*

This thesis is cumulative, i.e., it bundles multiple papers. While these papers can be found in Chapter 3, an overview of their contributions is given in this chapter. The contributions are categorized by the three main subjects described in Chapter 1 and are supplemented by additional related contributions in a separate section.

2.1. RoI encryption

The papers included in this thesis contribute RoI encryption approaches for JPEG and H.264 as well as auxiliary measures which simplify the encryption of SVC bit streams. The following sections describe the contributions for each image and video format.

2.1.1. JPEG encryption

Three encryption approaches for Baseline JPEG are proposed which build on top of one another. All approaches have in common that they outperform related work on post-compression JPEG encryption by one or both of the following two aspects: First, the proposed approaches are format-compliant, and, second, two of them are length-preserving as described below.

The basic approach on which the other two build is described in [Unterweger and Uhl, 2012] (Section 3.4). It encrypts Alternating Current (AC) coefficient values by making a series of swapping and scrambling operations at bit stream level. This operation can be done fast and without any decoding operations apart from determining the beginning and end positions of Huffman code words and blocks.

While the approach is not explicitly designed for RoI encryption, it can be trivially extended by limiting the encryption operations to those blocks which are part of a RoI. Since the security analysis in [Unterweger and Uhl, 2012] (Section 3.4) is performed for single blocks, its results apply to RoI encryption as well. This is explained in more detail in [Unterweger et al., 2015b] (Section 3.7) which adopts this approach explicitly for RoI encryption.

The basic approach [Unterweger and Uhl, 2012] (Section 3.4) is length-preserving since swapping and scrambling operations do not change the length of the bit stream. However, the JPEG format requires `FF` bytes at whole-byte positions to be escaped. Thus, when encryption yields an `FF` byte where there was another bit sequence before, the bit stream length increases by one byte due to the required escaping. Conversely, when encryption changes a formerly escaped `FF` byte, the bit stream length decreases by one byte. Thus, on average, the bit stream length is not changed by the proposed encryption approach.

The second approach, which builds on the basic one mentioned above, is described in [Auer et al., 2013] (Section 3.5). It adds additional Direct Current (DC) coefficient difference encryption based on the encryption approach proposed by [Niu et al., 2008]. It shares the properties of the basic encryption approach described above, with one notable exception.

Since JPEG stores DC coefficient differences, the trivial extension of the DC difference encryption approach to a RoI instead of the full picture results in spatial drift outside the RoI. This is

Approach	Format	Length-preserving	Drift-free
[Unterweger and Uhl, 2012]	JPEG	✓*	✓
[Auer et al., 2013]	JPEG	✓*	–
[Unterweger et al., 2015b]	JPEG	–	✓
[Unterweger et al., 2015a]	H.264	–	Partially
[Unterweger and Uhl, 2014a]	H.264	–	Partially (spatial)
[Unterweger and Uhl, 2014b]**	SVC	–	Partially (not temporal)

* On average (depending on escaping) ** Also proposed in [Unterweger and Uhl, 2014a]

Table 2.1.: Overview of proposed RoI encryption approaches and auxiliary techniques as well as their relevant properties.

due to the discrepancy between the encrypted and the unencrypted difference values. A more detailed explanation and solution for this can be found in [Unterweger et al., 2015b] (Section 3.7) which extends this encryption approach to support RoI without drift.

This constitutes the third encryption approach. It supports RoI encryption without drift, but does so at the cost of losing its length-preserving property. The DC coefficient discrepancy is bypassed by omitting encryption for certain DC coefficient difference bits inside the RoI and adding correction values at the RoI borders. Since this operation requires modifying the length of some Huffman code words, length-preservation is no longer likely to be achieved.

Table 2.1 lists all three approaches and their properties, i.e., length preservation and drift-freeness. The corresponding publications are:

[Unterweger and Uhl, 2012] Unterweger, A. and Uhl, A. (2012). Length-preserving Bit-stream-based JPEG Encryption. In *MM&Sec'12: Proceedings of the 14th ACM Multimedia and Security Workshop*, pages 85–89. ACM

[Auer et al., 2013] Auer, S., Bliem, A., Engel, D., Uhl, A., and Unterweger, A. (2013). Bitstream-Based JPEG Encryption in Real-time. *International Journal of Digital Crime and Forensics*, 5(3):1–14

[Unterweger et al., 2015b] Unterweger, A., Van Ryckegem, K., Engel, D., and Uhl, A. (2015b). Building a Post-Compression Region-of-Interest Encryption Framework for Existing Video Surveillance Systems – Challenges, obstacles and practical concerns. *Multimedia Systems*. submitted

2.1.2. H.264 encryption

For H.264, an RoI encryption approach is proposed and described in [Unterweger et al., 2015a] (Section 3.8). In contrast to related work, it refrains from transcoding whenever possible by performing most operations at bit stream level. This keeps re-compression operations at a minimum. In total, it reduces the processing time significantly compared to full re-compression approaches.

However, drift caused by some of the operations at bit stream level cannot be compensated entirely. This is due to the fact that, without full decoding, drift induced by bit-stream-based changes can only be approximated . Thus, compensation through partial re-compression only works with approximated values and therefore achieves no full drift compensation. In most practical cases, however, the remaining drift is limited spatially and temporally and thus rarely perceptible.

In addition, an auxiliary technique for H.264 RoI encryption is proposed and described in [Unterweger and Uhl, 2014a] (Section 3.9). Not being an encryption approach itself, it assesses how the presence of slice group borders around RoI helps reducing spatial drift. It is therefore applicable to all format-compliant H.264 RoI encryption approaches. Although slice groups induce space overhead, it is shown that this overhead is small enough in most practical configurations and a trade-off for the elimination of spatial drift.

Combining the two approaches from [Unterweger et al., 2015a] (Section 3.8) and [Unterweger and Uhl, 2014a] (Section 3.9) would allow for designing an drift-minimized H.264 RoI encryption approach. Since [Unterweger and Uhl, 2014a] (Section 3.9) eliminates spatial drift and can be applied to any format-compliant H.264 RoI encryption approach, in particular the one proposed in [Unterweger et al., 2015a] (Section 3.8), which produces little spatial and temporal drift, a combination of the two would only leave a small amount of temporal drift.

Table 2.1 includes the two approaches mentioned above and their properties, e.g., absence of drift. The corresponding publications are:

[Unterweger et al., 2015a] Unterweger, A., De Cock, J., and Uhl, A. (2015a). Bit-Stream-Based Encryption for Regions of Interest in H.264/AVC Videos With Drift Minimization. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. submitted

[Unterweger and Uhl, 2014a] Unterweger, A. and Uhl, A. (2014a). Slice groups for post-compression region of interest encryption in H.264/AVC and its scalable extension. *Signal Processing: Image Communication*, 29(10):1158–1170

2.1.3. SVC encryption

An auxiliary technique for SVC RoI encryption is proposed and described in [Unterweger and Uhl, 2014b] (Section 3.10). It is based on the same principle as [Unterweger and Uhl, 2014a] (Section 3.9), i.e., slice group borders are placed around RoI to eliminate drift. It is shown that in the case of SVC, even inter-layer drift can be eliminated by this technique.

Again, this auxiliary technique can be applied to any format-compliant RoI encryption approach. The trade-off is similar as for the H.264-based technique, but the space overhead is additionally dependent on the number of scalability layers. Due to SVC limitations, however, the elimination of spatial and inter-layer-drift comes at the price of effectively removing the base layer which imposes restrictions on the use of slice groups.

Nonetheless, it is shown that this influence does not impact rate-distortion performance significantly. Moreover, [Unterweger and Uhl, 2014a] (Section 3.9) proposes alternative methods to deal with base layer restrictions. It also breaks down the results further and analyzes constant and non-constant space overhead portions, allowing for a more detailed analysis of the sources of overhead induced by the drift-eliminating slice groups.

Similar to the auxiliary technique proposed for H.264 RoI encryption, the slice-group based techniques for SVC described in [Unterweger and Uhl, 2014b] (Section 3.10) and [Unterweger and Uhl, 2014a] (Section 3.9) could be combined with the encryption approach from [Unterweger et al., 2015a] (Section 3.8). This would allow for a post-compression SVC encryption approach which leaves only a small amount of temporal drift, but eliminates all other types thereof.

Table 2.1 includes the SVC-based technique mentioned above and its properties, i.e., drift-freeness in particular. The corresponding publications are:

[Unterweger and Uhl, 2014b] Unterweger, A. and Uhl, A. (2014b). Slice Groups for Post-Compression Region of Interest Encryption in SVC. In *IH&MMSec'14: Proceedings of the 2014 ACM Information Hiding and Multimedia Security Workshop*, pages 15–22, Salzburg, Austria. ACM

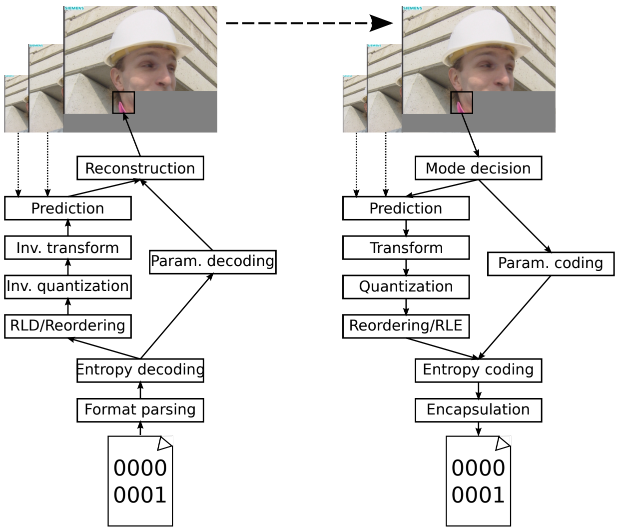


Figure 2.1.: Classical transcoding (left) vs. bit stream transcoding (right): The proposed post-compression watermarking approach operates at entropy coding level and therefore requires no complex decoding or encoding operations.

[Unterweger and Uhl, 2014a] Unterweger, A. and Uhl, A. (2014a). Slice groups for post-compression region of interest encryption in H.264/AVC and its scalable extension. *Signal Processing: Image Communication*, 29(10):1158–1170

2.2. Watermarking

For Blu-ray disks with H.264 video streams using Context-Adaptive Binary Arithmetic Coding (CABAC), a structure-preserving watermark is proposed and described in [De Cock et al., 2014] (Section 3.11). It assures length preservation of all bit stream entities, thereby enabling in-place watermarking without the need to re-edit any other parts of the Blu-ray disk. For extraction, the watermark locations are required, i.e., the extraction process is non-blind.

The watermark is robust against all use-case-typical distortions, like re-compression, anisotropic scaling and cropping. Watermark imperceptibility is guaranteed by only embedding in non-reference frames and an additional quality assurance loop which analyzes drift and removes watermark candidates where necessary. The capacity is sufficient to embed several different watermarks, even on Blu-ray disks with atypical GOP structures (e.g., without B frames).

One of the main contributions of the proposed watermarking approach is the post-compression embedding process. As opposed to re-compression-based approaches which require full re-encoding, e.g., by classical transcoding, after modifying parts of the input video, the proposed approach performs all changes at entropy coding level and can therefore limit all further operations to entropy transcoding. This is illustrated in Fig. 2.1.

Clearly, length and structure preservation can be guaranteed by pure entropy-coding-level transcoding. Whenever a potential watermark-induced change alters the length of the processed bit stream entity, it can be detected and discarded, i.e., the entropy-coded bit stream can be restored for this bit stream entity.

Furthermore, limiting all transcoding processes to the bit stream level omits all further de-

Approach	Format	Structure-preserving	Robust	Extraction type
[De Cock et al., 2014]	H.264	✓	✓	Non-blind

Table 2.2.: Overview of proposed watermarking approaches and their relevant properties.

Approach	Format	Structure-preserving	Cryptographically secure
[Hofbauer et al., 2014]	H.265	✓	✓

Table 2.3.: Overview of proposed transparent encryption approaches and their relevant properties.

coding (inverse quantization, transform, reconstruction etc.) and re-encoding (mode decision, transform, quantization etc.) operations which are required for classical transcoding. Hence, the complexity of the proposed approach is significantly lower than that of re-compression-based approaches.

Table 2.2 lists the proposed watermark approach and its properties mentioned above, e.g., structure-preservation and robustness. The corresponding publication is:

[De Cock et al., 2014] De Cock, J., Hofbauer, H., Stütz, T., Uhl, A., and Unterweger, A. (2014). An Industry-Level Blu-ray Watermarking Framework. *Multimedia Tools and Applications*, pages 1–23

2.3. Transparent encryption

For H.265 [ITU-T H.265, 2013], a transparent encryption approach is proposed and described in [Hofbauer et al., 2014] (Section 3.12). By only encrypting bits which are not entropy coded, length preservation is assured and the implementation complexity is reduced to that of bit stream parsing and bit replacement, i.e., no transcoding whatsoever is required. In addition, by limiting the encryption process itself to pseudo-random bit flipping, structure preservation is assured. This allows in-place transparent encryption for broadcasting applications.

The proposed transparent encryption approach is shown to be cryptographically secure due to its large key space and the properties of the bit stream parts chosen for encryption. Furthermore, the level of quality degradation can be chosen relatively freely depending on the percentage of encrypted bits as well as on the GOP structure and quantization parameters of the input video sequence. A number of configurations are evaluated and practically relevant values are recommended.

With the adoption of H.265 in broadcasting standards such as Digital Video Broadcasting (DVB)-S2 [Digital Video Broadcasting (DVB), 2014], the proposed encryption approach allows for transparent encryption of future (and current prototypical) Pay TV broadcasts. Due to its structure preserving properties and low complexity, it can be implemented in the form of a black box which modifies the bit stream right before the actual broadcasting step. Thus, it can be conveniently enabled and disabled as required or combined with other approaches as necessary.

Table 2.3 lists the proposed transparent encryption approach and its properties mentioned above, i.e., structure-preservation and cryptographic security. The corresponding publication is:

[Hofbauer et al., 2014] Hofbauer, H., Uhl, A., and Unterweger, A. (2014). Transparent Encryption for HEVC Using Bit-Stream-Based Selective Coefficient Sign Encryption. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1986–1990, Florence, Italy. IEEE

2.4. Other contributions

In the following sections, contributions related to RoI encryption are listed. They cover RoI signalling for encryption, face detection speed-up and the design and implementation of surveillance system software.

2.4.1. RoI signalling

One important ability of an RoI encryption implementation is to correctly decrypt the encrypted RoI on request. Apart from a correct decryption algorithm, this requires information on the locations and sizes of the RoI within each image. It is therefore necessary to signal these locations and sizes – either within the image itself or on a side channel. Since the requirement for a separate channel is typically undesired, in-image signalling is preferred.

Various RoI signalling algorithms are proposed and described in [Engel et al., 2013] (Section 3.6). All algorithms compactly represent the RoI locations (coordinates) and sizes by eliminating different types of redundancies. The algorithms are evaluated and compared, and the one with the highest compression efficiency is recommended for practical use.

Although RoI signalling is mentioned and even implemented in some approaches described in the literature (see [Engel et al., 2013] (Section 3.6) and [Unterweger et al., 2015b] (Section 3.7) for an overview of related work), no detailed descriptions of signalling algorithms have been given so far. Similarly, no evaluations in terms of efficiency and signalling overhead have been performed. Thus, [Engel et al., 2013] (Section 3.6) contributes first results in this area.

Furthermore, it describes and evaluates different methods to hide the signalled information in JPEG files. A wide range of common as well as new, more sophisticated techniques are explored in this regard. For both, lossy and lossless signalling, recommendations based on the evaluations are given. Since lossless signalling is typically preferred (or even required), the lossless signalling approach described in this paper contributes an integral part of the surveillance system implementation which is described in Section 2.4.3.

The publication for the RoI signalling approach mentioned above is:

[Engel et al., 2013] Engel, D., Uhl, A., and Unterweger, A. (2013). Region of Interest Signalling for Encrypted JPEG Images. In *IH&MMSec'13: Proceedings of the 1st ACM Workshop on Information Hiding and Multimedia Security*, pages 165–174. ACM

2.4.2. Face detection

A preliminary step for RoI encryption is RoI detection. Since RoI in the context of this thesis are always faces, face detection algorithms are required for RoI detection. A commonly used state-of-the-art face detector is the algorithm by Viola and Jones [Viola and Jones, 2001]. Despite its speed due to numerous optimizations in its design and implementation, there is still room for improvement, i.e., a decrease in run time.

The algorithm by Viola and Jones heavily relies on the use of integral images which have to be recalculated for each scale (potential face size). A simplification to avoid these recalculations is proposed and described in [Gschwandtner et al., 2014] (Section 3.13). A practically feasible approximation for rescaling in the integral image domain instead of the image domain is described, which avoids the requirement for recalculating the integral images on each scale. This approximation can be used for all approaches relying on such recalculations, not just the approach by Viola and Jones.

The achieved speed-up depends on the scaling factor and the number of threads used for parallel processing. In all cases, the run time can be reduced. This allows for faster face detection and therefore reduces the run time of RoI encryption implementations with integrated face detection significantly. For example, the approach can be used to speed up face detection of the RoI encryption framework which is described in Section 2.4.3.

The corresponding publication for the face detection speed-up method mentioned above is:

[Gschwandtner et al., 2014] Gschwandtner, M., Uhl, A., and Unterweger, A. (2014). Speeding Up Object Detection – Fast Resizing in the Integral Image Domain. In *VISAPP 2014 – Proceedings of the 9th International Conference on Computer Vision Theory and Applications*, volume 1, pages 64–72, Lisbon, Portugal. SciTePress

2.4.3. Surveillance system software

Post-compression RoI encryption implementations are only feasible if they can be integrated into existing image communication systems. In the case of face encryption for video surveillance systems, it is crucial that the latter can be extended without major modifications, or, preferably, without any modifications at all.

A full-featured RoI encryption framework is proposed and described in [Unterweger et al., 2015b] (Section 3.7). It can be integrated effortlessly into existing video surveillance systems and makes use of one of the RoI signalling methods mentioned in Section 2.4.1. Design and implementation facets like parallelization, modularity and different RoI detection methods are discussed, highlighting practical aspects of surveillance system software.

Apart from detailed objective evaluations in terms of run time, space overhead and comparisons to other approaches and implementations from the literature, a subjective evaluation of different post-compression encryption approaches is contributed. This way, the human component is considered and quantified. This is crucial since encryption in surveillance systems serves the privacy needs of humans. Understanding the capabilities and limitations of those who are using and potentially attacking the system helps improving surveillance systems as a whole and the proposed framework in particular.

The corresponding publication for the RoI encryption framework mentioned above is:

[Unterweger et al., 2015b] Unterweger, A., Van Ryckegem, K., Engel, D., and Uhl, A. (2015b). Building a Post-Compression Region-of-Interest Encryption Framework for Existing Video Surveillance Systems – Challenges, obstacles and practical concerns. *Multimedia Systems*. submitted

3. Publications

"You can scream now if you want." – Marv, *Sin City*

This chapter presents the papers as originally published. Errata can be found in Chapter 4. Some of the papers as well as the implementations required to produce the reported results have not been written entirely by myself, but by a number of co-authors. Appendix A lists the contributions per person per paper in detail.

The following section preceding the actual papers suggests an order (paper numbering) in which the papers can be read so that the amount of text being reread due to partial text overlaps is minimal. Papers which build on other papers included in this thesis are always ordered accordingly to enable linear reading, if this is desired.

3.1. Suggested order of reading

The papers are grouped by topic. The following sections list the papers for each topic in the suggested order of reading. They also shortly describe which papers build on one another and how much text overlap, if any, there is between them. In this section only, papers are additionally referred to by their paper numbers when dependencies are described. All papers from a given section (subject) can be read without reading any papers from the other sections (subjects).

3.1.1. Preliminaries

The following paper contains information which is relevant for understanding certain aspects of some of the papers from the other (following) subjects.

1. *Compression Artifacts in Modern Video Coding and State-of-the-Art Means of Compensation* ([Unterweger, 2013], Section 3.3): This paper is targeted at readers who are not already familiar with lossy video coding. It gives an overview of compression artifacts which occur, among others, in JPEG, H.264, SVC and H.265 bit streams, which are used in the papers from the other subjects. It is important to differentiate between these artifacts and drift which is explained and dealt with in most of the RoI encryption and watermarking papers.

3.1.2. RoI encryption

The following groups of papers describe RoI-encryption-related topics for JPEG, H.264 and SVC. Each group lists papers for one of these formats.

JPEG encryption

The following papers describe RoI encryption approaches and associated auxiliary methods for JPEG:

2. *Length-preserving Bit-stream-based JPEG Encryption* ([Unterweger and Uhl, 2012], Section 3.4): This paper can be skipped since it is a subset of paper *3* ([Auer et al., 2013], Section 3.5). It is included only for the sake of completeness. This paper does not explicitly mention RoI

encryption, but can be trivially extended to support it, as shown in paper *5* ([Unterweger et al., 2015b], Section 3.7). There are errata for this paper in Section 4.1.

3. *Bitstream-based JPEG Encryption in Real-time* ([Auer et al., 2013], Section 3.5): This paper extends paper *2* ([Unterweger and Uhl, 2012], Section 3.4) by an implementation capable of real-time encryption and decryption. Like paper *2* ([Unterweger and Uhl, 2012], Section 3.4), it does not explicitly mention RoI encryption, but can be extended to support it, as shown in paper *5* ([Unterweger et al., 2015b], Section 3.7). There are errata for this paper in Section 4.2.

4. *Region of Interest Signalling for Encrypted JPEG Images* ([Engel et al., 2013], Section 3.6): This paper establishes methods and results which are used in paper *5* ([Unterweger et al., 2015b], Section 3.7), with the main use case being RoI encryption in JPEG images. It is therefore recommended to read this paper after reading papers *2* ([Unterweger and Uhl, 2012], Section 3.4) and *3* ([Auer et al., 2013], Section 3.5) and before reading paper *5* ([Unterweger et al., 2015b], Section 3.7).

5. *Building a Post-Compression Region-of-Interest Encryption Framework for Existing Video Surveillance Systems – Challenges, obstacles and practical concerns* ([Unterweger et al., 2015b], Section 3.7): This paper builds on the work of papers *2* ([Unterweger and Uhl, 2012], Section 3.4) and *3* ([Auer et al., 2013], Section 3.5) and combines them for use in practical surveillance systems. Although it contains short descriptions of the algorithms it adapted, it is recommended to read papers *2* ([Unterweger and Uhl, 2012], Section 3.4) and *3* ([Auer et al., 2013], Section 3.5) first.

H.264 encryption

The following paper describes a RoI encryption approach for H.264:

6. *Bit-Stream-Based Encryption for Regions of Interest in H.264/AVC Videos With Drift Minimization* ([Unterweger et al., 2015a], Section 3.8): This paper can be read before or after any of the other papers. It is the only paper on (non-scalable) H.264 encryption and does not depend on paper *7* ([Unterweger and Uhl, 2014a], Section 3.9).

SVC encryption

The following papers describe RoI encryption facilitations for SVC:

7. *Slice groups for post-compression region of interest encryption in H.264/AVC and its scalable extension* ([Unterweger and Uhl, 2014a], Section 3.9): This paper extends paper *8* ([Unterweger and Uhl, 2014b], Section 3.10) by results on (non-scalable) H.264 bit streams and provides a number of additional analyses. However, it is recommended to read this paper first since its theoretical parts (sections) 1-3 have been thoroughly revised compared to paper *8* ([Unterweger and Uhl, 2014b], Section 3.10) and the latter only contains a small number of additional results, which can be supplemented after reading this paper.

8. *Slice Groups for Post-Compression Region of Interest Encryption in SVC* ([Unterweger and Uhl, 2014b], Section 3.10): This paper shares some text and results with paper *7* ([Unterweger and Uhl, 2014a], Section 3.9), but it is not a whole subset of the latter. It is recommended to read paper *7* ([Unterweger and Uhl, 2014a], Section 3.9) first and to then read only sections 4.1 and 4.2 of this paper, which give more implementation details and additional results for smaller video resolutions.

3.1.3. Watermarking

The following paper describes a watermarking approach:

9. *An Industry-Level Blu-ray Watermarking Framework* ([De Cock et al., 2014], Section 3.11): This paper can be read before or after any of the other papers. It is the only paper on watermarking and focuses on H.264 video streams on Blu-rays.

3.1.4. Transparent encryption

The following paper describes transparent encryption approaches:

10. *Transparent Encryption for HEVC Using Bit-Stream-Based Selective Coefficient Sign Encryption* ([Hofbauer et al., 2014], Section 3.12): This paper can be read before or after any of the other papers. It is the only paper on transparent encryption and focuses on H.265.

3.1.5. Face detection

The following paper describes an optimization technique which can be used in the context of RoI encryption.

11. *Speeding Up Object Detection – Fast Resizing in the Integral Image Domain* ([Gschwandtner et al., 2014], Section 3.13): This paper can be read before or after any of the other papers. It describes a modification for algorithms like the Viola-Jones object detection approach [Viola and Jones, 2001] to reduce the total running time of the algorithm, which is shown to be a major factor in RoI encryption systems by paper *5* ([Unterweger et al., 2015b], Section 3.7). There are errata for this paper in Section 4.3

3.2. Copyright notices

You are currently reading the online version of this thesis. Due to copyright regulations, only pre-print versions of the papers are included herein. Please refer to the print version to see the papers in the form in which they were originally published.

Note: This is a pre-print version subject to changes in formatting

Compression artifacts in modern video coding and state-of-the-art means of compensation

Andreas Unterweger
University of Salzburg, Austria

ABSTRACT
This chapter describes and explains common as well as less common distortions in modern video coding, ranging from artifacts appearing in MPEG-2 Video, MPEG-4 Part 2, H.264 and VC-1 to scalable and multi-view video coding based distortions, including the proposals for next generation video coding (NVC). In addition to a discussion about avoiding these artifacts through encoder-side measures, a state-of-the-art overview of their compensation at the decoder side is given. Finally, artifacts emerging from new sophisticated coding tools in current and upcoming video coding standards are discussed.

INTRODUCTION
As the coding tools used in modern video coding advanced in the last decades, new compression artifacts emerged, creating the need for sophisticated means of compensation. As the human eye is eventually the final recipient of the coded video, including distortions, artifact compensation based on human visual perception is an important research field, which is faced with new challenges due to new coding tools and the respective new artifacts induced by them.

It is important to be aware of these new artifacts and to analyze their sources in order to be able to compensate for them. As new coding tools are developed, most prominently represented by the current contributions to NVC, a basic understanding of the effects of the artifacts caused by these coding tools as well as their effect on the overall video quality is crucial. Although most of the current research is focused on the compensation of blocking, blurring and ringing artifacts and the development of new coding tools, this book chapter gives an overview of the artifacts caused by existing and new coding tools, focusing on mainstream block-based video coding represented by MPEG-2 Video, MPEG-4 Part 2, H.264, VC-1 and the amendments to H.264 for scalable and multi-view video coding. The interested reader may additionally find an overview of Wavelet-based compression artifacts appearing in Motion JPEG 2000 and others in Watson (1997) and Ramos (2001). Literature on non-mainstream video coding formats like Ogg Theora (Xiph.Org Foundation, 2011) is sparse (Crop, 2010) and therefore out of the scope of this book chapter.

The description of artifacts herein includes a discussion on the impact of new coding tools on artifacts in general and suggestions on how to minimize the appearance of these artifacts, thus eliminating the requirement for compensating them at the decoder side. After summarizing the properties and causes of commonly appearing artifacts such as blocking, blurring and ringing, including a number of artifacts originating from new coding tools, a short outlook on the perception of new artifacts and their connection to quality metrics concludes this chapter.

BACKGROUND
The origins of artifacts in block based transform video coding are, in most cases, directly or indirectly related to quantization errors in the transform domain, which are inevitable when lossily compressing images or sequences thereof. Since the first coding standards of this kind, e.g. JPEG for still image coding and H.261 for video coding, various related visual artifacts have been discussed throughout the literature.

Note: This is a pre-print version subject to changes in formatting

Blocking artifacts

Perhaps *the* most "famous" and most widely studied artifacts in today's block based video coding are blocking artifacts which occur due to the division of frames into macroblocks of rectangular shape. All blocks are coded separately from one another despite a possibly existing spatial correlation between them, yielding visible edges at macroblock borders. Due to the equidistant distribution of macroblock borders in JPEG, MPEG-2 Video and MPEG-4 Part 2 which is caused by the constant transform size of 8x8 samples, blocking artifacts are, in most cases, easily spotted by the Human Visual System (HVS) as a regular structure which does not belong to the image (Wu, 2006).

Due to the intense research concerning blocking artifacts, a number of possibilities for their compensation is available, e.g. (Oosa, 1998) and (Triantafyllidis, 2002). As both MPEG-2 Video and MPEG-4 Part 2 do not have an integrated deblocking filter, the artifact compensation has to be performed at the decoder side. In order not to cause a drift between encoder and decoder, deblocking has to be performed as a form of post processing on the decoded pictures which are displayed, but must not be applied to reference pictures which are used for motion compensated prediction.

Simple forms of deblocking involve low pass filtering at or around all macroblock borders, which causes blurring artifacts at borders which do not expose blocking artifacts (see below), whereas advanced approaches use edge detection algorithms to identify visually prominent edges or adaptively adjust the filter strength and/or area of influence, i.e. the number of samples around the macroblock border, based on image properties, quantizers, coding modes etc. The latter approach is incorporated in both H.264 and VC-1 in the form of an in-loop deblocking filter which is applied to all coded pictures before storing them in the reference buffers, yielding filtered references which are used for motion compensation. As experiments have shown that an image or video with blurring artifacts arising from strong deblocking appears more pleasant to a typical viewer than the corresponding unfiltered image or video (Wiegand, 2003), this supports the decision to incorporate in-loop deblocking filters into both video coding standards to improve the perceived quality of the decoded pictures.

Blurring artifacts

As mentioned above, strong deblocking can expose blurring artifacts due to the loss of high frequencies caused by low pass filtering during the attempt to flatten block edges. However, blurring may also be a result of quantization, if all high-frequency components in the transform domain are quantized to zero, yielding a low-pass-like behavior of the transform and quantization process. Using coarse quantization, i.e. selecting a high quantization parameter, favors blurring as it increases the probability of high-frequency components to be quantized to zero. As the HVS notices the loss of high frequency components to a lower degree than the loss of low-frequency components, the quantization matrices defined by MPEG-2 Video, MPEG-4 Part 2, H.264 and VC-1 cause a coarser quantization of high-frequency components, yielding blurring artifacts for high quantization parameters (Wu, 2006).

All standards mentioned above have no built-in filter to compensate for blurring artifacts and therefore require decoder-side deblurring algorithms, if desired. As the high-frequency components have been quantized to zero at the encoder side, they cannot be restored at the decoder side. Therefore, it is necessary to introduce high frequency components similar to noise, based on the image properties and the number of coefficients quantized to zero. Although sharpening might be an option in some cases, a number of approaches rely on boundary conditions (Ng, 1999) or inverse filtering (Biemond, 2005), yielding oversharpening artifacts or introducing noise. It is important to note that motion blur causes similar effects in the transform domain but requires different forms of compensation, involving – amongst others – deconvolution (Ben-Ezra, 2004).

In chroma-subsampled images (Kerr, 2009), blurring is often also referred to as color bleeding, as one chroma sample may stretch across multiple luma samples. Thus, the blurred chroma sample(s) spread(s) across a wider area, i.e. multiple luma samples, around an edge or other areas of high frequencies in the chroma planes (Wu, 2006). Although chroma subsampling increases the perceived strength of color bleeding due to the wider area affected, color bleeding can also occur in pictures where there is no chroma subsampling.

Note: This is a pre-print version subject to changes in formatting

Mosquito noise

At the borders of moving objects, an artifact called mosquito noise or mosquito effect appears, when a block is coded using inter frame prediction, but only a part of the predicted block contains the predicted moving object. The (static) rest of the block, therefore, differs strongly from the prediction, thus accounting for a major part of the total prediction error. As all video coding standards mentioned above use a form of prediction which operates in the transform domain, the quantization error together with the prediction error may yield a high concentration of error energy in the high frequency components due to the attempt to reduce the ringing from the prediction at the object border, thus yielding high frequency noise in the picture domain. The latter is referred to as mosquito noise if it is visible over a number of frames and the conditions described above apply.

Mosquito noise is visually prominent as the prediction error changes from frame to frame, yielding different high frequency noise patterns. It has to be noted that different coding of the same picture region across multiple pictures may also expose mosquito-noise-like artifacts. Another form of mosquito-like noise can be caused by encoder/decoder drift in MPEG-2 Video and MPEG-4 Part 2, which is due to the finite precision of the floating point operations involved in the transform and inverse transform process, yielding an imperfect reconstruction, differing between encoder and decoder, thus causing a drift between the two which propagates through prediction (Wiegand, 2003).

Ringing artifacts

Another common form of artifacts which manifest as “halos” around sharp edges is known as ringing (Wu, 2006). As steep edges in general contain a larger range of frequencies, the quantization of blocks with steep edges yields an insufficient reconstruction through the sum of basis functions, forming a less steep slope at the position of the original edge and both over- and undershooting at the samples around the original edge. In one-dimensional Fourier analysis, this is also known as Gibbs Phenomenon. Through the smoother slope of the edge it may appear blurry due to the loss of high frequency components whereas the over- and undershooting typically introduces the “halo”-like effect initially mentioned, creating a silhouette-like shade parallel to the original edge.

Note that the “halo” effect also affects low quantization parameters, i.e. small quantization step sizes, as a higher number of non-zero high frequency basis functions does not necessarily improve the approximation of an edge, thus not always decreasing the amount of ringing around sharp edges. Therefore, ringing may also be present in videos coded with high bit rates.

Although there are multiple approaches available describing how to measure ringing effectively, like (Shen, 1998) and (Liu, 2010), there are currently only two approaches available for compensating ringing artifacts, disregarding approaches optimized for JPEG 2000 and the like (Chang, 2005). Despite a sophisticated approach based on projections onto convex sets (POCS) (Zakhor, 2002) which is also used in the context of compensating blocking artifacts, there is an approach based on edge detection and adaptive filtering (Park, 1999) optimized for MPEG-4 Part 2, but applicable to all DCT and quantization based video coding standards which use a transform size of 8x8 samples, including H.264 with its High profile up to a certain extent.

Stair case and basis pattern artifacts

Another visual artifact closely related to ringing is the so-called stair case artifact which refers to the incapability of horizontal and vertical basis functions (as building blocks of the DCT and its variations) to accurately represent diagonal edges (similar to steep edges), thus resulting in the visually prominent presence of horizontal or vertical basis functions (Wu, 2006). Across multiple coded macroblocks, the appearance of a diagonal edge may be similar to the pattern of a stair case rather than that of a smooth diagonal connection between two points. Through the influence of blocking, stair case artifacts become visually more prominent as the “stair case step size” equals the size of a macroblock.

High quantization may reduce the number of non-zero coefficients in a transformed block to one, yielding so-called basis pattern artifacts which are similar to stair case artifacts, but exhibit a single basis function

Note: This is a pre-print version subject to changes in formatting

with its prominent picture domain representation. Note that this artifact is not limited to the scenarios described for stair case artifacts, but applies in general, when high quantization parameters are used which increase the probability of reducing the number of non-zero coefficients to one. If only the DC coefficient remains non-zero after quantization, a smooth “non-texture” block is coded which exhibits strong blocking and blurring and together with adjacent blocks of equal appearance forms a visual distortion referred to as mosaïking.

Summary

As apparent from the commonly described artifacts above, the causes for most of them are related to distortions through quantization. Some of them share manifestation patterns in terms of the quantization step size range and/or the number of non-zero quantized coefficients, increasing the probability under certain circumstances for artifacts to appear together. Therefore, Table 1 summarizes the quantization-dependent characteristics of these artifacts, together with possible accompanying artifacts. Figure 1 visualizes some of the artifacts described in Table 1. Note that this overview only covers the most relevant artifacts described in the literature. Further artifacts which mostly resemble the artifacts described herein may be found in the subsequent sections and (Wu, 2006).

	Causes of appearance			**Coexisting artifacts**
Artifact	**Reason for appearance**	**Typical quantization step size (range)**	**Number of non-zero quantized coefficients**	**Possibly appearing together with**
Blocking	Independent coding of spatially correlated adjacent blocks	High, but also depending on quantization step size (difference) of adjacent blocks	-	Mosaïking if neighboring blocks are also affected
Blurring	Loss of high-frequency components	High	Low (or zero) for high-frequency coefficients	Ringing at sharp edges, color bleeding (chroma)
Ringing	Insufficient approximation of steep edges	-	-	Blurring
Stair cases	Insufficient approximation of diagonal edges	-	-	Basis patterns for low quantization step sizes
Basis patterns	Loss of all but one transform coefficients	Very high	1	Stair cases
Mosquito noise	Quantization of high-frequency components and prediction errors	-	High for high-frequency components with a notable amount of total error energy	-

Table 1: Common artifacts in video coding and their causes (hyphens denote that the given artifact does not depend directly or necessarily on a certain value or value range of the respective cause of appearance)

4

Note: This is a pre-print version subject to changes in formatting

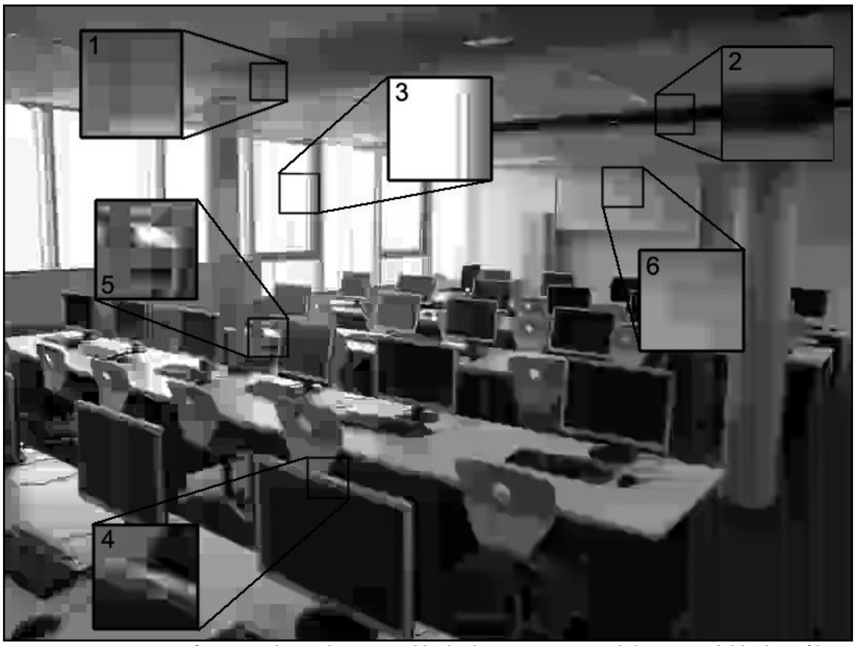


Figure 1: Common artifacts in video coding caused by high quantization and the H.264 deblocking filter (right image part only): blocking at the ceiling (example marked with 1), blurring through deblocking in the right image part (2), ringing at the window borders on the left (3), stair cases at the screens in the front (4), basis patterns between the screens in the middle (5), mosaïking on the white board in the back (6)

NEW CODING TOOLS, NEW ARTIFACTS: ISSUES, CONTROVERSIES, PROBLEMS

New transforms and transform sizes

Although most of the artifacts described above have been depicted and explained when MPEG-2 Video was state of the art or prior, they still appear when coding videos with MPEG-4 Part 2, H.264 and VC-1, albeit sometimes with different causes of appearance and probability as explained below. Concerning blocking artifacts, MPEG-4 Part 2 coded videos expose a similar behavior due to its transform size and function which is equal to the transform used in MPEG-2 Video, i.e. an 8x8 DCT (Richardson, 2003). By contrast, both H.264 and VC-1 support a smaller transform size of 4x4 besides 8x8 (VC-1 also supports 8x4 and 4x8, H.264 allows to switch between 4x4 and 8x8), reducing ringing due to the limited space for over- and undershooting within one transformed block (Wiegand, 2003). Smaller transform sizes also increase the probability of blocking as the number of block borders increases, although the transform size may be chosen adaptively. More information on the type of transform used in H.264 and VC-1 as opposed to the DCT may be found below.

Concerning the increase of the probability of blocking, both H.264 and VC-1 apply an in-loop deblocking filter which adaptively smoothens block borders in order to avoid blocking. The strength and area of

Note: This is a pre-print version subject to changes in formatting

application is determined by various parameters, like the type and quantization parameter of the involved blocks. As can be seen in the right half of Figure 1, high quantization parameters with in-loop deblocking in H.264 lead to blurring, effectively eliminating blocking artifacts. Although the filter can be turned off, this is not encouraged due to the increased presence of blocking artifacts for high quantization parameters as shown in the left half of Figure 1.

As opposed to MPEG-2 Video and MPEG-4 Part 2, which use a DCT of 8x8 size, both H.264 and VC-1 use an approximation thereof, allowing for implementations with additions, subtractions and logical (barrel) shifts only, thus improving performance on modern CPUs. Although H.264 and VC-1 use different approximations of the DCT, the approximations themselves are similar to one another as they are based on the same transform matrix and were derived through similar operations (Lee, 2008). Therefore, the subsequent paragraphs describe the integer transform used in H.264 for the sake of illustration.

Although H.264 supports both a 4x4 and an 8x8 integer transform since the amendment of the fidelity range extensions, allowing to switch between the two when using the High profile (International Telecommunication Union, 2010), actually two approximations of the DCT are used – one of size 4x4 and one of size 8x8. For the sake of simplicity and illustration, the subsequent paragraph will focus on the 4x4 integer transform whose basis functions are illustrated in Figure 2 and compared to the corresponding DCT basis functions. As can be seen, the integer transform clearly is an approximation of the DCT, with similar basis functions arising from this relationship. The detailed derivation and approximation process may be found in Malvar (2003). It has to be noted that the encoder/decoder drift described above is avoided due to the use of integer operations and the resulting absence of rounding errors.

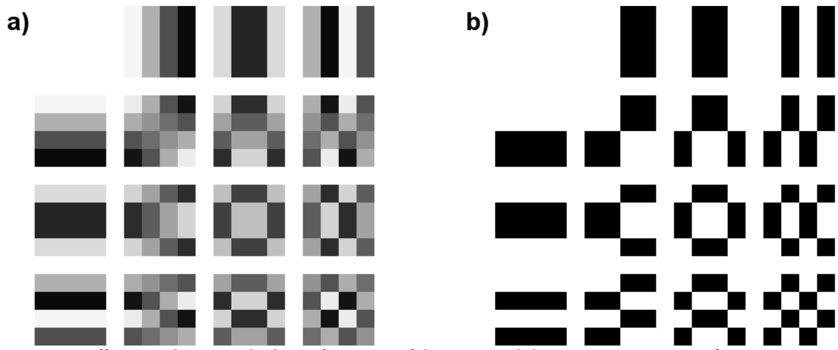


Figure 2: Differences between the basis functions of the DCT and the H.264 integer transform: a) 4x4 DCT basis functions, b) H.264 4x4 integer transform basis functions; both derived through inverse transform of single transform coefficients. Black denotes minimum values, white denotes maximum values; picture domain values are within [-128;127]

Even though the basis functions of the 4x4 DCT and the integer transform are similar, they are not the same, thus yielding different transform coefficients and transform coefficient distributions for a number of input signals (disregarding simple cases like DC only blocks whose DC transform coefficient only differs in magnitude due to scaling). Figure 3 illustrates this using a simple input signal which yields eight transform coefficients when using the DCT, but six when using the integer transform used in H.264. Although the two additional coefficients do not contribute much to the total signal energy and are likely to be quantized to zero, inverse transform of the quantized coefficients will yield different reconstructed signals for both transforms, considering the small loss of signal energy described.

6

Note: This is a pre-print version subject to changes in formatting

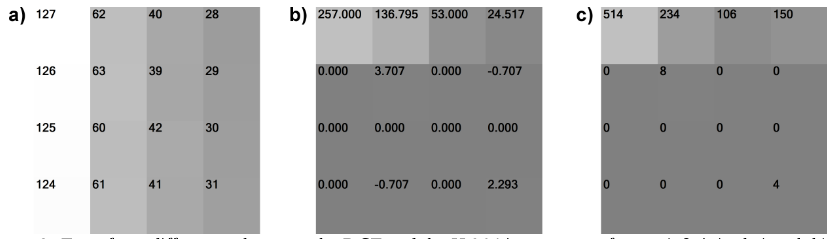


Figure 3: Transform differences between the DCT and the H.264 integer transform: a) Original signal, b) 4x4 DCT transform, c) H.264 4x4 integer transform. Black denotes minimum values, white denotes maximum values; picture domain values are within [-128;127]

Although experiments have been carried out to determine new quantization matrices for H.264 due to the new transform (Richardson, 2003), no research is currently focusing on the impact that the new transform may have on type and quantity of distortions. Due to its historical connection to the DCT, the integer transform used in H.264 shares many of its properties, but also yields different transform coefficient distributions due to its different basis functions, as shown above. Although the speedup through design approximations favoring the capabilities of state-of-the-art CPUs may be convenient, it is necessary to investigate the modified behavior regarding the appearance of artifacts as well as the possible creation of new artifacts which have not been described yet.

The effect of macroblock partitioning

Despite the change in transform size and type, various other new algorithms have been developed for MPEG-4 Part 2, H.264 and VC-1 in order to make coding more efficient or to improve visual quality. Such algorithms are usually referred to as coding tools. One of them is macroblock partitioning in inter-predictive coding, enabling to perform a separate motion search for each part of a macroblock, allowing finding better matches for each part. While all of the aforementioned standards allow to split a 16x16 inter-predicted block into 4 block partitions of size 8x8 each, H.264 additionally supports 16x8 and 8x16 partitions as well as sub-partitions for 8x8 partitions of sizes 8x4, 4x8 and 4x4, respectively (International Telecommunication Union, 2010).

Such partitioning possibilities not only increase the probability of blocking (without the in-loop deblocking filter) but also favor the appearance of an artifact referred to as motion compensation (MC) mismatch (Wu, 2006). MC mismatch describes an effect during motion compensation where the match found during motion estimation does not belong to the object currently being coded, thus appearing misplaced. When using coarse quantization, the difference to the currently coded macroblock cannot be appropriately compensated for, thus yielding additional blocking and blurring which makes the HVS sensible for the “misplaced” object block. While MC mismatch may be a result of the lack of chroma motion estimation (thus trying to find a matching luma block only), it is also possible that it is due to a purely mathematical error measurement like the sum of squared differences (SSD) or the mean squared error (MSE), yielding a motion estimation match with the minimal mathematical difference, but with a distinct perceptual difference as the match does not belong to the object currently being coded.

As described above, an increasing number of partitions and sub-partitions increases the probability for MC mismatches, which become visually more prominent when surrounded by perceptually adequate matches (Wu, 2006). When using low quantization parameters, this problem quasi disappears as the difference through the MC mismatch can be compensated for by predictive coding. Nonetheless, MC mismatches also favor the appearance of mosquito noise due to the borders of mismatching objects found during motion estimation. This noise may also be visible at higher bit rates, i.e. lower quantization parameters.

7

Note: This is a pre-print version subject to changes in formatting

Besides the number and shape of partitions which both increase the probability of certain artifacts as described above, the number of available coding modes to choose from also yield new artifacts. Most prominently, an artifact named flickering or pumping, also known as stationary area fluctuations, appears when the chosen coding modes of one picture area changes over time, i.e. over subsequent frames. As the predicted residuals from intra and inter prediction differ strongly, the form of the coded residual after quantization is different, yielding different errors and thus flickering due to the change of error over time (Chun, 2006).

Although this artifact is often described as having similarities to mosquito noise, its origins are different. As applying temporal smoothing yields side effects when trying to compensate for this artifact during post processing (Wu, 2006), pumping artifacts can be avoided during the coding process by selecting similar modes for co-located regions in subsequent frames as described in Chun (2006). It has to be noted that a similar selection of partitions and sub-partitions is also helpful in order to achieve this, although not all inter predicted partitions have an equivalent intra predicted partition in terms of size. Furthermore, the prediction signals of inter and intra prediction vary strongly, as do the different intra prediction modes, thus requiring careful adaption of quantization parameters in addition to the coding mode selection in order to reduce pumping artifacts effectively.

Multi-view video coding

Another current field of research is multi-view video coding (MVC), i.e. the coding of multiple views of a scene in order to either produce a three dimensional rendering of said scene or a part of it, albeit often limited to the number of existing views and the interpolated views between them. The most prominent configuration is stereoscopic coding, i.e. the coding of two views – one for the left eye and one for the right – which enables a three dimensional effect when each view is exposed to the corresponding eye. There are currently multiple technologies (like polarized glasses or active shutter glasses) in order to achieve this (May, 2005). In terms of video coding, there are currently three basic approaches for multi-view video coding, which will be shortly described in the subsequent paragraphs, each together with the artifacts it induces or favors.

Depth map quantization artifacts

The first approach constitutes the coding of a two dimensional image or texture and a so-called depth map indicating the distance from the camera for each pixel. This depth map can either be provided in special cases or is otherwise estimated by the encoder when given one or multiple views (Smolic, 2007). Depth estimation is a research topic of great current interest due to the emerging three dimensional TV sets and the associated technologies (Ohm, 2010). The coding of depth maps is explicitly specified in MPEG-4 part 2. Using transform, quantization and residual coding, depth maps are compressed like textures, thus yielding similar artifacts (Richardson, 2003).

Assuming quasi-lossless compression of textures, the quantization of depth maps yields a number of different artifacts which are related to their counterparts in regular image and video coding, although their appearance to the viewer may be different. One example is so-called depth ringing where ringing artifacts emerge from depth map compression, yielding distortions of the depth map and therefore the perceived depth (Boev, 2008). Figure 4 a) depicts the effects of depth ringing, also referred to as depth bleeding. As its image distortion counterpart, depth ringing is most prominent at steep edges (of the depth map), i.e. the region between the ball and the checkerboard background in Figure 4 a). In general, fluctuations in depth may be perceived easily in some scenes, making MSE, PSNR and similar metrics unsuitable for the quality estimation of multi-view videos which rely on depth map quantization.

Note: This is a pre-print version subject to changes in formatting

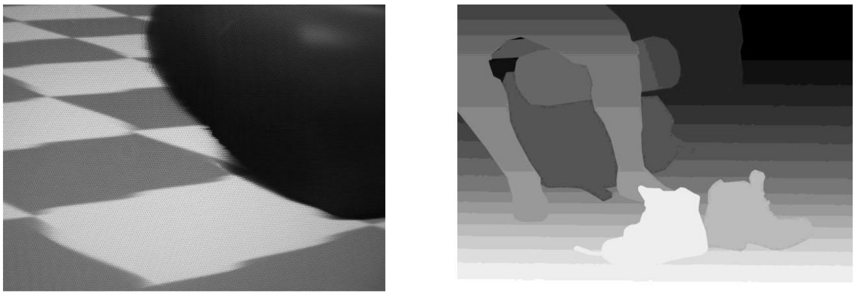


Figure 4: Depth map compression artifacts: a) depth ringing. © 2008, Mobile3DTV (Boev, 2008). Used with permission. b) card board effect. Lighter colors in the depth map indicate greater depth.

Although depth blocking, blurring and similar artifacts may appear when coding depth maps, they have not been described in the literature. Most likely, this is due to the fact that current research efforts are focused on depth estimation and other coding techniques for multi-view video coding. Nonetheless, both depth estimation and harsh quantization may yield an artifact which has been described as card board or puppet theater effect, depicted in Figure 4 b). This refers to a layer-like depth map, similar to the layers of objects in a puppet theater, creating the perception of a number of two dimensional layers instead of smooth depth transitions.
Combining the artifacts of depth map and texture coding, a superposition of them may appear. Depending on the severity of artifacts, they may mask each other, making one so visually prominent that the other one is not visible any more (Wu, 2006). This is also true for all other artifacts described herein, although there is currently no research focused on the human visual perception of jointly appearing artifacts. This may change with the number of emerging coding technologies, currently represented by multi-view and scalable video coding, increasing the number of artifacts and therefore the probability of their joint appearance.

Frame packing artifacts
The second approach allowing for stereoscopic video coding only is an extension specifically available for H.264, called frame packing (Vetro, 2011). It uses a supplemental enhancement information (SEI) message to signal the frame packing of the pictures in the coded video. Frame packing refers to the coding of both views – left and right – in one single view, with both core encoding and decoding algorithms being possibly unaware of the existence of two separate views, thus maintaining compatibility to the H.264 standard as the core coding tools do not need to be changed. Combining and separating the views before and after coding, respectively, must be performed by the encoder (or a preprocessor) and the decoder (or a postprocessor), respectively as the combination of the two views for coding, the insertion of the SEI message and the separation of the two views for display must be performed.
The latest revision of the H.264 standard (International Telecommunication Union, 2010) specifies a number of frame packing arrangement types depicted in Figure 5. Assuming two views of a size of eight times eight macroblocks each, both views are either horizontally or vertically subsampled, depending on the arrangement type, and then rearranged in order to form a picture of eight times eight macroblocks containing both views. However, one arrangement type – frame alternation, depicted in Figure 5 f) – does not require spatial subsampling as each view is represented by all even and odd pictures, respectively, i.e. it is temporally subsampled. If subsampling is used, upsampling after decoding is necessary to restore the original picture of each view, thus introducing similar artifacts as upsampling in scalable video coding described below.

Note: This is a pre-print version subject to changes in formatting

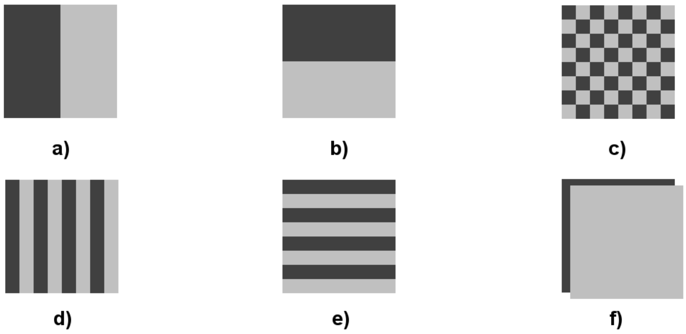


Figure 5: H.264 frame packing arrangement types: a) side-by-side (horizontal), b) top-bottom (vertical), c) checkerboard, d) column alternation, e) row alternation, f) frame alternation. Light and dark gray depict macroblocks of the left and the right view, respectively

In addition to the artifacts caused by subsampling and upsampling – reduced by quincunx sampling of the original samples of both views – crosstalk of artifacts is possible due to the interleaved coding of the two views. Although side-by-side and top-bottom frame packing are only likely to expose these artifacts at the borders between the two views, column and row alternation as well as checkerboard arrangements are expected to introduce crosstalk. As described in Vetro (2011), color bleeding is very likely to propagate across views, mostly when using checkerboards arrangements.

The probability of appearance of mosquito noise, pumping and MC mismatch is also increased due to the interleaving of views, albeit of limited influence to the overall coding performance in terms of PSNR. MC mismatch is favored due to the macroblock size spaced interleaving in most arrangements, causing the motion estimation algorithm to find a match in a macroblock of the other view as the motion estimation search range in most current H.264 encoders is around 16 samples (Jurkiewicz, 2011; Lee, 2008; Richardson, 2003) which equals the size of one macroblock for progressive input.

Using frame alternation arrangements, MC mismatches are even more likely to occur as motion compensation can only be performed on frames which have already been coded, thus disallowing motion compensation in the frame currently being encoded (International Telecommunication Union, 2010). In addition, due to the limited size of the reference lists and practical considerations which limit the number of references used for motion estimation, the number of frames from the other view searched during the motion estimation phase is likely to be greater than the number of frames from the same view that the picture currently being coded belongs to.

Although the artifacts appearing in pictures using frame packing are similar to the artifacts previously described, both subsampling and upsampling have to be considered in terms of the range of artifacts as well as in terms of artifact superposition. Currently, there is only very little research performed in this area, although frame packing is already used intensively by broadcasting and other companies due to its compatibility to H.264 and the frame packing capabilities of the video signal transmission standard HDMI 1.4a.

Due to its increasing popularity in the years 2010 and 2011, more and more H.264 encoders begin to implement support for frame packing (Jurkiewicz, 2011), thus requiring research focused on further coding artifacts due to frame packed coding, including the consideration of the human visual perception of these artifacts when the stereoscopic video is decoded and displayed on a device capable of three-dimensional display using different technologies. As crosstalk and uneven visual quality of the two views

10

are known to influence the perception of depth (Boev, 2008), frame packing requires further investigation regarding these issues.

Artifacts in H.264 MVC

The third approach for coding multiple views is the recent amendment of H.264 named multi-view video coding or MVC for short (Vetro, 2011), extending H.264 in a backwards compatible way. While the first view is coded like a regular H.264 bit stream, all other views use special signaling so that they are ignored by decoders which do not support MVC. In order to improve coding efficiency, all other views may be predicted from the first one or another view of the same access unit, i.e. a view of the same point in time as the view currently being coded (International Telecommunication Union, 2010).

Due to the current widespread use of stereoscopic coding, H.264 MVC defines new Stereo profiles for two views, enabling coding performance similar to the frame packing approach described above, albeit without subsampling and the restrictions imposed thereby (Vetro, 2011). MVC makes use of the similarity of multiple views at any given time instant, referred to as inter-view prediction. As can be seen in Figure 6, the similarity of the left and the right view eases prediction, although predictions from other views may yield a higher amount of MC mismatch artifacts as described above.

Figure 6: Similarity of the left and the right view of a stereoscopic view. Differences are clearly visible in the form of illumination differences in the left part of both images as well as in differences of perspective at the borders on the left and the right

Despite the similarity of both views, there are differences due to the different perspective of the depicted objects, mostly notable at the left and right borders. In addition, the illumination of the two views may be slightly different as shown in the left part of both views in Figure 6. Although illumination compensation, i.e. an algorithm to compensate for luminance differences between the coded view and the view used for prediction, had been part of the working drafts of H.264 MVC, it was not included in the final version of the standard as it would require changes of the low level syntax and the corresponding coding tools. However, notable efforts regarding illumination compensation in terms of the improvement of coding efficiency have been made (Park, 2008) which may be incorporated into future video coding standards (Vetro, 2011).

As the artifacts arising from illumination compensation have not been studied in depth, this leaves room for future research. Other artifacts like crosstalk and different forms of MC mismatches depending on the difference in perspective between the depicted objects in the views used for motion compensation also need a more detailed inspection. Although MPEG-2 Video already included means to encode stereoscopic videos and entailed perceptual considerations of its coding mechanisms (Tseng, 1995), the approach used in MPEG-2 Video and, therefore, its artifacts differ from the approach used in H.264 MVC as the former makes use of the scalability features of MPEG-2, using one view as the base layer and the second one as an enhancement layer, yielding a special form of inter-view prediction based on inter-layer prediction.

Note: This is a pre-print version subject to changes in formatting

Scalable video coding
Scalable video coding (SVC) is available in MPEG-2 Video and MPEG-4 Part 2 and has been introduced as an amendment of H.264 in 2007 (Schwarz, 2007). It generally refers to the ability to decode specified parts of the bit stream, yielding a smaller frame rate, spatial resolution or quality than when decoding the whole bit stream. Scalability relies on layers defining temporal scalability in terms of frame rate, spatial scalability in terms of spatial resolution and quality or SNR scalability in terms of fidelity as well as combinations thereof. In general, the overhead of having multiple layers is small, thus allowing for coding performance similar to a stream containing only the highest frame rate, resolution and quality.

Temporal-scalability-related artifacts
H.264 SVC implements temporal scalability similar to prior standards by using backwards compatible B pictures and special signaling. It makes use of hierarchical prediction structures with B pictures using previously coded B pictures within a group of pictures (GOP) for prediction, each B picture hierarchy being equal to one temporal layer. As only the existing concept of B pictures and special signaling is used, no new artifacts originate besides the ones introduced by B pictures themselves, such as mosquito noise, MC mismatch and others (Wu, 2006). In order to avoid pumping artifacts, (Schwarz, 2007) recommends increasing the quantization parameter in higher temporal layers in a predefined pattern when using hierarchical B pictures to achieve temporal scalability, although this results in PSNR fluctuations inside a GOP.

Spatial-scalability-related artifacts
The implementation of spatial scalability differs between MPEG-2 Video, MPEG-4 Part 2 and H.264 SVC, although the basic concept is similar. Higher spatial layers use the coded information of the spatial layer below by upsampling it and using it for prediction, whereas the lowest spatial layer, also called base layer, is coded regularly, i.e. without the use of scalable video coding tools. The upsampled signal from the lower layer can be based on a reconstructed, i.e. decoded, signal (MPEG-2 Video and MPEG-4 Part 2 as well as inter-layer intra prediction in H.264 SVC) or on transform coefficients of a residual signal (H.264 SVC only, referred to as inter-layer inter prediction).
In H.264 SVC, upsampling also requires the block partitions of the lower layer to be upsampled accordingly, i.e. by a factor of two in each direction when using dyadic differences in resolution between two spatial layers, thereby possibly upsampling blocking artifacts and favoring MC mismatches as motion vectors are scaled accordingly and reference list indices are reused, yielding the same prediction area for motion compensation. In addition, mosquito noise may be upsampled, making it more visible when using higher quantization parameters in the enhancement layer.
The upsampling of transform coefficients in H.264 SVC is performed using a bilinear filter in order to avoid additional signal components due to the in-loop deblocking filter. Inter-layer intra prediction uses a 4-tap FIR filter for the luma samples and a bilinear filter for the chroma samples, albeit based on reconstructed samples instead of transform coefficients. Similar to the required upsampling process in MVC, inter-layer intra prediction upsampling introduces blurring to the prediction signal due to bilinear filtering (Krylov, 2008), thus favoring the appearance of mosquito noise due to the lack of high frequency signal components.

Quality-scalability-related artifacts
Quality scalability is available in a number of different forms throughout the standards mentioned herein. Coarse grain quality scalability in H.264 SVC relies on the same mechanisms as are used for inter-layer inter prediction described above apart from the upsampling operations (the two layers have the same spatial resolution), thus yielding similar artifacts. The difference to be coded is based on the difference between the coarser quantized coefficients in the base layer and the original signal, resulting in finer quantized coefficients in the enhancement layer (Schwarz, 2007). Medium grain scalability uses the same basic concept as coarse grain scalability, whereas fine grain scalability which has already been supported

Note: This is a pre-print version subject to changes in formatting

in MPEG-2 Video and MPEG-4 Part 2 allows for more sophisticated prediction structures between base and enhancement layers.
Although enhancement layers in MPEG-4 Part 2 can only be predicted from the corresponding base layers, MPEG-2 Video allows the prediction of base layers from previously coded enhancement layers, thus introducing a drift between encoder and decoder if enhancement layers are discarded. This has been avoided by a special key picture concept in H.264 SVC which is described in detail in International Telecommunication Union (2010) and Schwarz (2007). Although the aforementioned drift differs from the drift of MPEG-2 Video encoders and decoders described above, no research has been conducted so far in order to explore the visual effects of drift-based distortions.

SOLUTIONS AND RECOMMENDATIONS

Encoder-side artifact awareness

As can be seen from the descriptions in the previous section, new coding tools tend to introduce new forms of artifacts or to modify the presence or number of existing artifacts. Therefore, it is important to know the sources of these new artifacts and to develop strategies to avoid or compensate for them. Taking MC mismatch as a prominent example from the previous section, avoiding the resulting artifacts might be a better idea than trying to compensate for them as both detection and compensation may be difficult. This may be the reason why currently neither is described in the literature.
Modifying the encoder may therefore be a better option in order to avoid or at least reduce MC mismatch significantly. Chroma motion estimation, for example, may assist regular luma motion estimation by supplying additional information to consider when calculation the SSD or MSE in order to avoid MC mismatches, albeit more time consuming than luma-only motion estimation. Another option would be to replace the mathematical functions for difference and error measurement by functions which reflect the properties of the HVS better. This would allow avoiding MC mismatches as well as other artifacts up to a certain extent, incorporating properties of the HVS during coding.
Most current video encoders make use of rate distortion optimization (RDO), i.e. testing multiple possible modes and selecting the mode with the smallest cost. The smallest cost is defined as the mode with the best rate-distortion tradeoff, subject to a predefined relation between rate and distortion in order to calculate these costs for all modes. As distortion is mostly measured in a mathematical sense in these calculations as described below, it may be favorable to replace it with distortion measures which are aware of some important properties of the HVS, considering the most common artifacts and their effect on the human visual perception.

Development and application of new quality metrics

Although a number of multiple similar distortion measurements have been discussed in the literature, each with its own strengths and weaknesses, the structural similarity (SSIM) index developed by Wang (2004) has proven to be a distortion measure that correlates well with the human visual perception subject to certain restrictions (Sheikh, 2006). By measuring the structural similarity in terms of variance and covariance of two images on a per-block (8x8) basis, the structural similarity index is capable of considering blurring, ringing, basis patterns and stair cases as well as other forms of image distortions, albeit unable to detect blocking artifacts with blocks of the size of the SSIM block size (8x8).
A first approach to incorporate SSIM indices as distortion measures has been proposed in Mai (2005). Therein, a subset of an H.264 encoder has been modified to use SSIM indices as a distortion measure for RDO during intra mode decision. As may be anticipated from a classical (peak signal to noise ratio, PSNR) point of view, the PSNR performance of the encoded pictures decreases compared to an unmodified encoder, while the perceptual quality increases. If this approach was extended to all RDO-amenable decisions involving new coding tools, the impact of new artifacts or the (re)appearance of classical artifacts might be reduced significantly, favoring coding modes with a reduced presence of artifacts. For motion estimation and similar coding tools, SSIM block matching may not only help to

13

avoid MC mismatches, but also to find structurally more similar blocks than existing approaches, making the arising differences perceptually easier to encode.

The awareness of the perceptual influence of decisions during coding (mode decision, motion estimation etc.) is crucial. Therefore, it is necessary to include facilities into encoders which are aware of the perceptual impact of these decisions, helping to improve the perceived quality by design. As PSNR and other purely mathematical measures of difference give a general hint of the degree of quality degradation, they have proven insufficient when masking effects of the HVS and small or imperceptible differences come into play (Wang, 2004).

Although the computational complexity and ease of comparability of PSNR and the like is convenient for the purposes of state-of-the-art video coding, it is not in terms of the correlation between this metric and a typical human viewer rating video quality. Instead of sacrificing computational power (and therefore time) for new coding tools which improve the PSNR of a given configuration by a small amount, it is conceivable to sacrifice this time to design an HVS-aware quality metric for use within the encoder (perhaps even in form of a new coding tool) in order to improve the overall quality of the encoded pictures, thus also enabling perceptually aware coding control units which can distribute more bit rate to perceptually critical areas of a picture, thereby reducing the number and strength of perceived artifacts at the same bit rate.

Approaches to switch to a different metric for the measurement of differences and errors have also already been proposed by others, e.g. Ohm (2010), although there is currently no concrete direction observable in terms of a concrete metric to choose. SSIM may be an intermediate approach on the way to a new metric, albeit imperfect as it does not cover all important aspects of the human visual perception and does not correlate well with the HVS at low bit rates (Hofbauer, 2010). Despite its incapability to detect certain types of artifacts as discussed above, its high correlation throughout a wide range of bit rates with the HVS would make it a good choice to replace PSNR in the short or medium term, leaving potential for design optimizations in form of a new or different metric in the long term.

FUTURE RESEARCH DIRECTIONS

Next generation video coding

As the future of video coding and its arising artifacts is closely related to the new coding tools designed, the current development of next generation video coding (NVC) based on H.264's coding tools, also referred to as high efficiency video coding (HEVC), gives an insight into the new coding tools and artifacts that will have to be dealt with in the future. At the time of this writing, a preliminary version (1.0) of the future reference software "HM" (HEVC testing model) has been made available to the public (https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/), implementing most of the new coding tools selected for detailed evaluation after their approval in the call for proposals for NVC.

As the number of new coding tools compared to the latest revision of the H.264 standard (International Telecommunication Union, 2010) increased significantly and the release date of the preliminary reference software did not allow for thorough testing at the time of writing of this book chapter, those of the new coding tools which will probably have the strongest impact on artifacts will be discussed, considering that the current version of the reference software is not the final one and the continuing evaluation process may exclude coding tools described herein as well as include new ones.

A major change in terms of video coding is the macroblock size which is now 64x64 luma samples and referred to as a coding unit (CU) with accompanying concepts for prediction units (PU) and transform units (TU), allowing partitioning and sub-partitioning over four hierarchy levels (down to 4x4) as opposed to the two hierarchy levels in H.264 inter prediction (McCann, 2010). Although the number of partitions does not necessarily change the probability of the appearance of artifacts (the smallest size is still the same as in H.264), the introduction of a 16x16 integer transform might lead to a more significant appearance of ringing artifacts compared to the 4x4 and 8x8 transform sizes in H.264 due to the increased number of coefficients and samples available for over- and undershooting as described in the previous

Note: This is a pre-print version subject to changes in formatting

section. Transform sizes of 32x32 and 64x64, which are also being evaluated, yet increase the probability of ringing artifacts.

Besides the change in transform size, which also requires thorough inspection as described above for integer transforms in general, the interpolation filter for subsamples in the motion estimation and compensation process may be changed, too. The proposed improvements describe the use of a 6-tap directional filter or a 12-tap DCT based interpolation filter, replacing the Wiener and bilinear filter used in H.264 for subsample interpolation. As both approaches change the signal characteristics of the interpolated subsamples and therefore the likeliness to expose artifacts, future research will have to show how this affects the perceptual quality and artifact propagation.

In addition to the coding tools described, an extension of the number of available intra prediction modes has been proposed and modified (Min, 2010), introducing angular intra prediction in contrast to the nine 4x4 and four 16x16 prediction modes in H.264 making use of a limited number of horizontal, vertical and diagonal prediction. With a total of 17 modes for 4x4 PUs, 34 for 8x8, 16x16 and 32x32 PUs and 5 for 64x64 PUs, requiring the interpolation of predicted samples up to an accuracy of 1/32 of a sample, the new intra prediction modes will increase the probability of pumping artifacts further, apart from increasing the number of modes for RDO and therefore computational complexity significantly.

Analysis of existing artifacts

Despite the fact that future research related to HEVC which will have to wait until the reference software and the HEVC specification are finalized, the evaluation of artifacts arising from the emerging SVC and MVC standards will offer a number of opportunities for artifact research. As described above, there are multiple coding tools whose effects on existing and new artifacts have not yet been examined in depth, thus requiring further inspection and analysis.

Furthermore, the superposition of different artifacts and their effect on the HVS becomes more relevant as the number of known artifacts is already high and yet keeps increasing through the introduction of new video coding standards and amendments thereof. Studying which artifacts are visually prominent when appearing in certain constellations with other artifacts might not only provide a clearer perspective on the perception of artifact superpositions, but also on masking effects originating from the HVS in general.

Artifact-aware encoder design

Overall, the consideration of the human visual perception in video encoder design is an important issue to take into account. If the artifact-related properties of the HVS are already considered in the design process of new coding tools and in the encoder, future research can focus on artifact avoidance instead of compensation. Furthermore, the encoder-side awareness of the presence of artifacts could be used to apply artifact compensation algorithms on both, the encoder and the decoder side, more selectively, reducing the post-processing complexity on the latter side through encoder-generated artifact signaling.

In addition, new metrics can be developed which represent the human visual perception better than existing ones, allowing for improved encoder decisions. Such metrics can also be used for the difference and distortion measurements in general, making artifact detection easier. If, in addition, artifact propagation is analyzed in the encoder, video quality can be estimated more precisely, eventually enabling artifact-aware video coding.

CONCLUSION

Despite the increasing rate of improvement in terms of compression efficiency in modern video coding, the avoidance and compensation of coding artifacts are currently not getting the attention they deserves. The development of new coding tools decreases bit rates compared to previous standards at the cost of increased computational complexity and a lack of awareness of the impact of these new coding tools on the appearance of known or new artifacts. It is important to be aware of the artifacts arising from improvements in video coding algorithms, enabling a broader understanding of the human visual perception besides the classical artifacts, such as blocking, ringing, blurring and the like.

Note: This is a pre-print version subject to changes in formatting

Although it might not be the final solution, the consideration of different quality metrics like SSIM for difference and error measurement as well as RDO can be a step towards the awareness of certain artifacts during the encoding process. Be it in form of a new coding tool as an integral part or as an addition to the core coding tools of a video encoder, the consideration of the human visual perception during the coding process can help to improve the perceptual quality of encoded videos in current and future video coding standards. It may also help to gain a better understanding of the influence of new coding tools regarding their vulnerability to induce artifacts.

In doing so, the need for decoder side artifact detection and compensation would also diminish, thus requiring less attention than currently, allowing future research to concentrate on the development of new metrics for quality measurement on the encoder side rather than sophisticated artifact compensation algorithms on the decoder side. Therefore, it is indispensable to focus future research efforts on artifact avoidance at the encoder side or (even before) in the design process of new coding tools. In the end, it is the casual user, unaware of the mere existence of the most sophisticated coding tools, who judges the visual quality and the visibility of coding artifacts.

REFERENCES

Ben-Ezra, M. & Nayar, S. K. (2004) Motion-Based Motion Deblurring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6), 689–698.

Biemond, J., Lagendijk, R. L. & Mersereau, R. M. (1990) Iterative methods for image deblurring. *Proceedings of the IEEE*, 78(5), 856–883.

Boev, A., Hollosi, D. & Gotchev, A. (2008) *Classification of stereoscopic artifacts*. Retrieved February 1, 2011 from http://sp.cs.tut.fi/mobile3dtv/results/tech/D5.1_Mobile3DTV_v1.0.pdf.

Chang, Y.-W. & Chen, Y.-Y. (2005) *Alleviating-Ringing-Artifact Filter Using Voting Scheme*. Paper presented at the ICGST International Journal on Graphics, Vision and Image Processing, Cairo, Egypt.

Chun, S. S., Kim, J. R. & Sull, S. (2006) Intra prediction mode selection for flicker reduction in H.264/AVC. *IEEE Transactions on Consumer Electronics*, 52(4), 1303–1310.

Crop, J., Erwig, A. & Selvaraj, V. (2010) *Ogg Video Coding*. Retrieved September 21, 2011 from http://people.oregonstate.edu/~cropj/uploads/Classes/577finalreport.pdf.

Hofbauer, H. & Uhl, A. (2010) *Visual Quality Indices and Low Quality Images*. Paper presented at the *IEEE 2nd European Workshop on Visual Information Processing*, Paris, France.

International Telecommunication Union (2010) *Recommendation ITU-T H.264 – Advanced video coding for generic audiovisual services (03/2010)*. Geneva, Switzerland: International Telecommunication Union.

Jurkiewicz, A. et al. (2011) *X264 Settings*. Retrieved February 1, 2011 from http://mewiki.project357.com/wiki/X264_Settings.

Kerr, D. A. (2009) *Chrominance Subsampling in Digital Images*. Retrieved February 1, 2011 from http://dougkerr.net/pumpkin/articles/Subsampling.pdf.

Krylov, A. & Nasonov, A. (2008) Adaptive total variation deringing method for image interpolation. *Proceedings of the 15th IEEE International Conference on Image Processing 2008,* 2608–2611.

Note: This is a pre-print version subject to changes in formatting

Lee, J.-B. & Kalva, H. (2008) *The VC-1 and H.264 Video Compression Standards for Broadband Video Services*. New York, New York: Springer Science+Business Media LLC.

Liu, H., Klomp, N. & Heynderickx, I. (2010) *A no-reference metric for perceived ringing*. Paper presented at the Fourth International Workshop on Video Processing and Quality Metrics for Consumer Electronics, Scottsdale, Arizona.

Mai, Z.-Y., Yang, C. L. & Xie, S. L. (2005) Improved best prediction mode(s) selection methods based on structural similarity in H.264 I-frame encoder. *IEEE International Conference on Systems, Man and Cybernetics*, 3, 2673–2678.

Malvar, H. S., Hallapuro, A., Karczewicz, M. & Louis Kerofsky (2003) Low-Complexity Transform and Quantization in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7), 598–603.

May, P. (2005) *A Survey of 3D Display Technologies*. Retrieved February 1, 2011 from http://www.ocuity.co.uk/Ocuity_white_paper_Survey_of_3D_display_technologies.pdf.

McCann, K., Han, W.-J. & Kim, I. K. (2010) *Samsung's Response to the Call for Proposals on Video Compression Technology (JCTVC-A124)*. Retrieved February 1, 2011 from http://wftp3.itu.int/av-arch/jctvc-site/2010_04_A_Dresden/JCTVC-A124.zip.

Min, J.-H., Lee, S., Kim, I.-K., Han, W.-J., Lainema, J. & Ugur, K. (2010) *Unification of the Directional Intra Prediction Methods in TMuC (JCTVC-B100)*. Retrieved February 1, 2011 from http://wftp3.itu.int/av-arch/jctvc-site/2010_07_B_Geneva/JCTVC-B100.zip.

Ng, M. K., Chan, R. H. & Tang, W.-C. (1999) A Fast Algorithm For Deblurring Models With Neumann Boundary Conditions. *Siam J. Sci. Comput.*, 21(3), 851–866.

Ohm, J.-R. (2008) *Recent, Current and Future Developments in Video Coding*. Retrieved February 1, 2011 from http://wiamis2008.itec.uni-klu.ac.at/keynotes/ohm.pdf.

Oosa, K. (1998) *A New Deblocking Filter for Digital Image Compression*. Retrieved February 1, 2011 from http://www.nsc.co.jp/en/tech/report/pdf/7716.pdf.

Park, H. W. & Lee, Y. L. (1999) A Postprocessing Method for Reducing Quantization Effects in Low Bit-Rate Moving Picture Coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1), 161–171.

Park, G. H., Park, M. W., Lim, S. C., Shim, W. S & Lee, Y. L. (2008) Deblocking Filtering for Illumination Compensation in Multiview Video Coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(10), 1457–1461.

Ramos, M. G. & Hemami, S. S. (2001) Suprathreshold wavelet coefficient quantization in complex stimuli: psychophysical evaluation and analysis. *Journal of the Optical Society of America A, Optics, Image Science and Vision*, 18(10), 2385–2397.

Richardson, I.E.G. (2003) *H.264 and MPEG-4 Video Compression*. Chichester, England: John Wiley & Sons Ltd.

Note: This is a pre-print version subject to changes in formatting

Schwarz, H., Marpe, D. & Wiegand, T. (2007) Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. IEEE Transactions on Circuits and Systems for Video Technology, 17(9), 1103–1120.

Sheikh, H. R., Sabir, M.F. & Bovik, A. C. (2006) A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms. *IEEE Transactions on Image Processing*, 15(11), 3440–3451.

Shen, M. Y. & Kuo, C.-C. J. (1998) Review of Postprocessing Techniques for Compression Artifact Removal. Journal of Visual Communication and Image Representation, 9(1), 2–14.

Smolic, A., Müller, K., Stefanoski, N., Ostermann, J., Gotchev, A., Akar, G. B., Triantafyllidis, G. & Koz, A. (2007) Coding Algorithms for 3DTV—A Survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(11), 1606–1621.

Triantafyllidis, G. A., Tzovaras, D. & Strintzis, M. G. (2002) Blocking Artifact Detection and Reduction in Compressed Data. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(10), 877–890.

Tseng, B. L. & Anastassiou, D. (1995) *Perceptual Adaptive Quantization Of Stereoscopic Video Coding Using MPEG-2's Temporal Scalability Structure*. Paper presented at the International Workshop on Stereoscopic and Three Dimensional Imaging 1995, Santorini, Greece.

Vetro, A., Wiegand, T. & Sullivan, G. J. (2011) Overview of the Stereo and Multiview Video Coding Extensions of the H.264/AVC Standard. *Proceedings of IEEE, Special Issue on "3D Media and Displays"*, 99(4), 626–642.

Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. (2004) Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4), 600–612.

Watson, A., Yang, G. Y., Solomon, J. & Villasenor, J. (1997) Visibility of Wavelet Quantization Noise. *IEEE Transactions on Image Processing*, 6(8), 1164–1175.

Wiegand, T., Sullivan, G. J., Bjøntegaard, G. & Luthra, A. (2003) Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7), 560–576.

Wu, H. R. & Rao, K. R. (Eds.) (2006) *Digital Video Image Quality and Perceptual Coding*. Boca Raton, FL: CRC/Taylor & Francis.

Xiph.Org Foundation (2011) *Theora Specification*. Retrieved September 21, 2011 from http://theora.org/doc/Theora.pdf.

Zakhor, A. (2002) Iterative procedures for reduction of blocking effects in transform image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 2(1), 91–95.

ADDITIONAL READING SECTION

Boujut, H., Benois-Pineau, J., Hadar, O., Ahmed, T. & Bonnet, P. (2011) Weighted-MSE based on saliency map for assessing video quality of H.264 video streams. *Proceedings of the SPIE – "Image Quality and System Performance VIII"*, 7867, 78670X–78670X-8.

Brooks, A. C., Zhao, X. & Pappas, T. N. (2008) Structural similarity quality metrics in a coding

Note: This is a pre-print version subject to changes in formatting

context: exploring the space of realistic distortions. *IEEE Transactions on Image Processing*, 17(8), 1261–1273.

Carballeira, P., Tech, G., Cabrera, J., Müller, K., Jaureguizar, F., Wiegand, T. & García, N. (2010) *Block based Rate-Distortion analysis for quality improvement of synthesized views*. Paper presented at the 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video 2010, Tampere, Finland.

Farrugia, R. A. & Debono, C. J. (2010) A Hybrid Error Control and Artifact Detection Mechanism for Robust Decoding of H.264/AVC Video Sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(5), 756–762.

Goldmann, L., De Simone, F., Dufaux, F., Ebrahimi, T., Tanner, R. & Lattuada, M. (2010) *Impact of video transcoding artifacts on the subjective quality*. Paper presented at the Second International Workshop on Quality of Multimedia Experience 2010, Trondheim, Norway.

Hewage, C. T. E. R., Worrall, S. T., Dogan, S. & Kondoz, A. M. (2008) Prediction of stereoscopic video quality using objective quality models of 2-D video. *Electronics Letters*, 44(16), 963–965.

Hwang, Y. & Park, J. (2003) *Reduction of compression Artifacts (Blocking Artifacts, Ringing Artifacts) Using POSC*. Retrieved February 1, 2011 from http://homepages.cae.wisc.edu/~ece533/project/f04/hwang_park.pdf.

Hoffmann, H., Itagaki, T., Wood, D., Hinz, T. & Wiegand, T. (2008) A Novel Method for Subjective Picture Quality Assessment and Further Studies of HDTV Formats. *IEEE Transactions on Broadcasting*, 54(1), 1–13.

Kim, W. S., Ortega, A., Lai, P. L., Tian, D. & Gomila, C. (2009) *Depth map distortion analysis for view rendering and depth coding*. Paper presented at the 16th IEEE International Conference on Image Processing 2009, Cairo, Egypt.

Le Meura, O., Ninassib, A., Le Calletc, P. & Barbac, D. (2010) Do video coding impairments disturb the visual attention deployment? *Signal Processing: Image Communication*, 25(8), 597–609.

Lee, J. & Park, H. W. (2011) A New Distortion Measure for Motion Estimation in Motion-Compensated Hybrid Video Coding. *Signal Processing: Image Communication*, 26(2), 75–84.

Liu, T., Wang, Y., Boyce, J. M., Yang, H. &Wu, Z. (2009) A Novel Video Quality Metric for Low Bit-Rate Video Considering Both Coding and Packet-Loss Artifacts. *IEEE Journal of Selected Topics in Signal Processing*, 3(2), 280–293.

Merkle, P., Morvan, Y., Smolic, A., Farin, D., Müller, K., de With P. H. N. & Wiegand, T. (2008) The effects of multiview depth video compression on multiview rendering. *Signal Processing: Image Communication*, 24(1), 73–88.

Shao, L. & Kirenko, I. (2007) Coding Artifact Reduction Based on Local Entropy Analysis. *IEEE Transactions on Consumer Electronics*, 53(2), 691–696.

Shao, L., Zhang, H. & Liu, Y. (2011) A Generalized Coding Artifacts and Noise Removal Algorithm for Digitally Compressed Video Signals. *Lecture Notes in Computer Science*, 6523/2011, 1–9.

Note: This is a pre-print version subject to changes in formatting

Sugimoto, O., Naito, S., Sakazawa, S. & Koike, A. (2009) *Objective perceptual video quality measurement method based on hybrid no reference framework*. Paper presented at the 16th IEEE International Conference on Image Processing 2009, Cairo, Egypt.

Suk, J.-Y., Lee, G. W. & Lee, K.-I. (2005) Effective Blocking Artifact Reduction Using Classification of Block Boundary Area. *Lecture Notes in Computer Science*, 3768/2005, 606–616.

Unterweger, A. & Thoma, H. (2007) *The Influence of Bit Rate Allocation to Scalability Layers on Video Quality in H.264 SVC*. Paper presented at the Picture Coding Symposium 2007, Lisboa, Portugal.

van den Branden Lambrecht, C. J. & Verscheure, O. (1996) Perceptual Quality Measure using a Spatio-Temporal Model of the Human Visual System. *Digital Video Compression: Algorithms and Technologies*, 2668, 450–461.

Vo, D. T. & Nguyen, T. Q. (2009) *Optimal motion compensated spatio-temporal filter for quality enhancement of H.264/AVC compressed video sequences*. Paper presented at the 16th IEEE International Conference on Image Processing 2009, Cairo, Egypt.

Wang, Z., Bovik, A. C. & Lu, L. (2002) Why is Image Quality Assessment So Difficult? Paper presented at the *IEEE International Conference on Acoustics, Speech, & Signal Processing 2002,* Orlando, Florida.

Winkler, S. & Mohandas, P. (2008) The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics. *IEEE Transactions on Broadcasting*, 54(3), 660–668.

Wu, H. R. & Yuen, M. (1997) A generalized block-edge impairment metric for video coding. *IEEE Signal Processing Letters*, 4(11), 317–320.

Yang, J. & Wu, H. (2010) Robust Filtering Technique for Reduction of Temporal Fluctuation in H.264 Video Sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(3), 458–462.

Zhang, Z. L., Shi, H. & Wan, S. (2009) *A Novel Blind Measurement of Blocking Artifacts for H.264/AVC Video*. Paper presented at the Fifth International Conference on Image and Graphics 2009, Xi'an, China.

KEY TERMS & DEFINITIONS

Artifact	Image distortion induced by side effects of a coding tool and/or quantization
Block	Rectangular unit of image samples grouped for coding
Blocking	Artifact caused by independent coding of neighboring blocks
Blurring	Artifact caused by loss of high frequency components, making the block appear fuzzy
Coding tool	Distinct set of algorithms within a video encoder to improve compression or picture quality
Macroblock	Synonym for a block or a group thereof (depending on the context)
Ringing	Artifact related to Gibbs Phenomenon in Fourier analysis, creating a “halo” consisting of over- and undershooting samples as well as blurring parallel to steep edges due to the insufficient approximation of the original edge by the quantized coefficients in the transform domain

BIOGRAPHY

Andreas Unterweger received his Master's-equivalent Diploma degree in Information Technology and Systems Management (with distinction) from the Salzburg University of Applied Sciences in 2008 and his Master's degree in Computer Science (with distinction) from the University of Salzburg in 2011. He is

Note: This is a pre-print version subject to changes in formatting

currently pursuing his Ph.D. degree in Computer Science at the University of Salzburg where he specializes on selective video encryption. In addition, he is an external lecturer at the Salzburg University of Applied Sciences, teaching Microcontroller Programming and Applied Mathematics in the Bachelor's degree program. His current research interests include real-time video coding and selective video encryption.

Length-preserving Bit-stream-based JPEG Encryption

Andreas Unterweger
University of Salzburg
Department of Computer Sciences
Jakob-Haringer-Straße 2
Salzburg, Austria
andreas.unterweger@stud.sbg.ac.at

Andreas Uhl
University of Salzburg
Department of Computer Sciences
Jakob-Haringer-Straße 2
Salzburg, Austria
uhl@cosy.sbg.ac.at

ABSTRACT

We propose a new method to encrypt baseline JPEG bit streams by selective Huffman code word swapping and coefficient value scrambling based on AES encryption. Furthermore, we show that our approach preserves the length of the bit stream while being completely format-compliant. In contrast to most existing approaches, no recompression is necessary as the encryption is applied directly to the bit stream. In addition, we assess the effort required for brute-force and known-plaintext attacks on pictures encrypted with our approach, showing that both are practically infeasible.

Categories and Subject Descriptors

I.4.2 [**Image Processing and Computer Vision**]: Compression (Coding)—*JPEG, Huffman code*; E.3 [**Data**]: Data Encryption—*AES*

General Terms

Algorithms, Security, Experimentation

Keywords

JPEG, AES, encryption, length-preserving, Huffman code

1. INTRODUCTION

The encryption of compressed images to ensure privacy is an active research topic for a variety of different compressed image and video formats. For Joint Picture Experts Group (JPEG)-compressed images [4] in particular, several approaches exist due to the widespread use of this image format. While most of them require recompressing the original data to some extent, the method proposed in this paper operates on bit-stream level, using only swap and scramble operations, thus being very fast.

A number of approaches have been proposed which do either not preserve the length of the original file or break format compliance. This includes techniques such as zig zag permutation [5, 16] (which significantly increases the file size) and the use of permuted Huffman tables [18]. Similarly, DC bit plane scrambling as proposed e.g. in [7] increases the file size and is thus not length preserving as opposed to our approach.

In terms of simple length-preserving encryption algorithms on Discrete Cosine Transform (DCT)-based images and videos, pseudo-randomly toggling DC and/or AC coefficient signs as proposed e.g. in [12, 1] is frequently used. However, the attack complexity for breaking such themes is significantly lower than in our approach as we encrypt multiple bits per coefficient instead of only one (the sign bit).

Similarly, encrypting a limited number of bits on bit-stream level starting from the DC coefficient [13] or the high-frequency AC coefficients [14], respectively, is of lower security as compared to the proposed approach which encrypts all coefficients and additionally increases the complexity by reordering blocks. The technique of reordering all blocks within a picture, which is used as part of our approach in a spatially limited fashion, has already been proposed in [20], [8] and [10] (and analysed in [17] and others). Although it increases the total attack complexity depending on the picture size, it does not allow for Region of Interest (RoI) encryption without a significant decrease in attack complexity, as opposed to our approach.

In terms of code-word-based techniques such as the one we propose, only a small number approaches have been published. Besides swapping code words of equal length between blocks for AC value histogram spreading as proposed in [19], a method to shuffle code words with the same in-block position between blocks exists for MPEG-4 [17] which could also be applied to JPEG pictures. However, the latter approach may yield non-format-compliant bit streams and both methods are not intended to be used for RoI encryption as opposed to our approach.

Another method described in [17] encrypts multiple concatenated Variable-Length Code (VLC) symbols and maps them to another string of valid VLC symbols so that the total length is preserved. Note that this approach, which has been applied to MPEG-4 bit streams, cannot be used for JPEG as, in the latter, each Huffman code word is followed by a signed coefficient residual represented by a number of bits which is encoded in the Huffman code word. Changing the code words in a length-preserving way changes the number of coefficient bits encoded in the Huffman code word as opposed to the actual subsequent bits in the bit stream, making the bit stream parser get out of sync and thus breaking format compliance.

Note that our bit-stream-based approach is designed for encryption without the need for recompression, which is useful when there is no possibility to intercept the encoding process. One practical use case is the encryption of pictures from surveillance cameras, most of which send streams of JPEG pictures (which are already encoded). Although real-time recompression is possible with state-of-the-art hardware, omitting this step may save equipment and costs.

This paper is structured as follows: In section, 2 we describe our approach. In section 3, we give an estimation of the effort necessary for a successful attack on a picture encrypted with our approach. Finally, we provide an outlook in section 4 before concluding the paper.

2. BIT STREAM ENCRYPTION

In baseline JPEG, $8 \cdot 8$ blocks of cosine-transformed quantized AC coefficients are zig-zag scanned and run-length coded before being Huffman coded. Hereby, the Huffman code words only encode run-length pairs as symbols, whereas the actual coefficient values are written directly to the bit stream as signed residue from 0 or $-2^s + 1$, respectively, where s denotes the length part of the run-length symbol.

Our approach consists of three different operations. Firstly, the order of the run-length coded symbols together with their corresponding coefficient values (referred to as code-word-value pairs henceforth) is permuted. Secondly, the coefficient value bits are scrambled and thirdly, the order of all blocks within an Interleaved Minimum Coded Unit (iMCU) which use the same Huffman table is permuted. Both, the second and third operation, are described in detail at the end of this section, while the subsequent paragraphs describe the first operation, i.e. the permutation of the order of code-word-value pairs.

Permutations of this order lead to a change of the order of the zero runs in each block, thereby altering the positions of the coefficient values within the $8 \cdot 8$ block. Figure 1 depicts this by example.

On the top left, an examplary block with four non-zero coefficient values is shown. The dots denote that the rest of the coefficients are zero. The DC coefficient is neither changed nor considered. On the top right, the zig-zag scanned values of the examplary block are depicted and grouped with their preceding zeros. Each of these groups is coded as a Huffman code word (black) of the run-length symbol (depicted on top of each Huffman code word) and the coefficient value (grey). E.g., the first coefficient (5), which is preceded by no (i.e. 0) zeros, requires 3 bits (101) to be repesented, thus leading to a run-length of 0/3. As the rest of the blocks' coefficients apart from the four ones depicted are zero, an End of Block (EOB) is signalled.

By swapping the groups of Huffman code words and coefficient values (if there is more than one group), the zero runs and therefore the position of the coefficient values within the block change, as depicted at the bottom of figure 1. However, the bit stream remains format-compliant as the exchange of code words does neither change the Huffman codes themselves nor does it change the total number of coefficients. In addition, it does not change the length of the JPEG file, thus being length-preserving.

The code-word-value pair order permutation through element-wise reordering is derived as follows. Before processing a JPEG file, an Advanced Encryption Standard (AES) [9] implementation in Output Feedback (OFB) mode is initialized with a given initialization vector and key, which can be

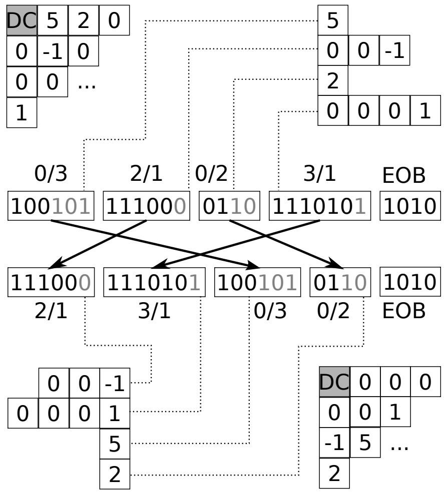


Figure 1: Example of run-length permutation: The order of Huffman coded run-length symbols and their corresponding coefficient values is permuted, thereby changing the position of the values in the coefficient matrix

file- or user-dependent. It then serves as a Pseudo-Random Number Generator (PRNG) by using n AES-encrypted output bits, where 2^n is the desired range of the PRNG. Note that any cryptographic PRNG could be used here.

Code-word-value pair order permutation is then performed by swapping the current code word and its corresponding coefficient value at position i with the code-word-value pair at position $rand(n)$ where $rand$ denotes a call to the AES-based PRNG with an upper value bound of n and n is equal to the number of total code-word-value pairs. For the example in figure 1, $n = 4$, yielding the consumption of 2 encrypted output bits of the AES encoder per possible swap operation.

In addition to code-word-value pair order permutation, our approach changes the coefficients' values in the bit stream (grey bits in figure 1). This is done by toggling each of the n value bits depending on whether or not the AES-based PRNG described above returns a binary zero or one when using one bit. Similar to the run-length order permutation, this does not change the length of the JPEG file as the value bits actually represent a signed residual of fixed size per code word (see above).

Furthermore, the order of all blocks using the same Huffman table within an iMCU is permuted. Figure 2 shows an example with 4:2:0 subsampling [6] where the U and the V block use the same Huffman table (marked grey). After the permutation, the order of U and V is switched, with the bit stream still being format-compliant. The permutation itself is derived as described for run-length permutations above. Note that no code-word-value pairs are exchanged

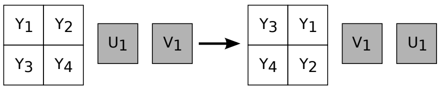


Figure 2: Example of block order permutation: The order of blocks using the same Huffman code words within an iMCU is permuted. In this example, the Y blocks use one set of Huffman code words (white), while both, the U and the V block, use another (grey)

Figure 3: Example of an encrypted picture region with the proposed method (right) and the corresponding original region (left) from a JPEG-compressed version of the picture "woman" from the LIVE data base [15]

among the blocks as this would break format compliance – only whole blocks with all their code-word-value pairs are exchanged. Again, this does not change the length of the JPEG file.

Figure 3 shows an example of a picture region encrypted with our approach with an original JPEG quality of 75%. Note that the number of possible order permutations in blocks with few code-word-value pairs and/or small coefficient values (e.g. in the area above the woman's head) is small, making those blocks appear nearly undistorted, i.e. unencrypted. Conversely, the other blocks exhibit significant distortion due to the reordering and scrambling, revealing that the local encryption strength of our approach depends on the amount of information contained in a block as explained in more detail in the next section.

Note that although the woman's silhouette is recognizable in the encrypted picture, no more details (like facial characteristics) can be extracted from it. This partial encryption is due to the fact that the DC coefficients are not encrypted as described in the next section. Although this allows creating a picture with $\frac{1}{64}^{th}$ of the orginal size out of DC coefficients, all information contained high frequency coefficients is lost this way without proper decryption and therefore does not compromise the security of our approach.

3. SECURITY ANALYSIS

In order to assess the cryptographic security of our approach, its three main components are analyzed in terms of attack complexity, i.e. the number of possible combinations per iMCU and the probability of success for key extraction in a known-plaintext attack. The key space depends on the AES key size and can be up to $2^{256} \approx 10^{77}$ for AES with 256 bit keys [9].

The approach described in the previous section relies on three independent scrambling mechanisms: permuting the order of code-word-value pairs, toggling value bits and permuting the order of blocks within an iMCU. Due to their independence, the number of possible combinations can be analyzed separately and eventually multiplied to yield the overall number of possible combinations.

Let m denote the total number of blocks in an iMCU and let n_i denote the total number of code words in the i^{th} block of an iMCU. Let $l_{i,j}$ denote the length of the j^{th} code-word-value pair of the i^{th} block of an iMCU in bits. The number of possible values (i.e. bit combinations) $c_v(i,j)$ for each value is $2^{l_{i,j}}$, thus being

$$N_v = \prod_{i=1}^{m} \prod_{j=1}^{n_i} c_v(i,j) = \prod_{i=1}^{m} \prod_{j=1}^{n_j} 2^{l_{i,j}} \tag{1}$$

for all blocks within an iMCU.

The number of permutations $p_{rlv}(i)$ of code-word-value pairs of each block is $n_i!$, where $x!$ denotes the factorial of x. Thus, the total number of code-word-value pair permutations is

$$N_{rlv} = \prod_{i=1}^{m} p_{rlv}(i) = \prod_{i=1}^{m} n_i! \tag{2}$$

for all blocks within an iMCU. Similarly, the number of block permutations $p_b(x)$ for x blocks which use the same Huffman code words is $x!$, thus being

$$N_b = \prod_{k=1}^{h} p_b(n_h(k)) = \prod_{k=1}^{h} n_h(k)! \tag{3}$$

for all blocks within an iMCU, where h denotes the total number of different AC Huffman tables and $n_h(k)$ is the number of blocks using the k^{th} Huffman table so that $\sum_{k=1}^{h} n_h(k) = m$.

In total, this yields an overall number N of possible combinations per iMCU of

$$N = N_v \cdot N_{rlv} \cdot N_b = \prod_{i=1}^{m} \prod_{j=1}^{n_i} 2^{l_{i,j}} \cdot \prod_{i=1}^{m} n_i! \cdot \prod_{k=1}^{h} n_h(k)! \tag{4}$$

In order to estimate the values for $l_{i,j}$ and n_i for typical natural JPEG pictures, the reference pictures of the LIVE data base presented in [15] have been encoded with different quality settings (between 0 and 100% with 5% step size) using the JPEG reference encoder. The encoded files have then been analysed in terms of the average number of runs per block and the average length of coefficient values.

We consider the average values to be an appropriate measure for the following reason: Our algorithm's attack complexity depends on the number of code-word-value pairs within a block, i.e. it varies with its number of non-zero coefficients. The distribution of the latter (not depicted) reveals that the self-information of a block with a high number of code-word-value pairs is greater than that of a block with a small number thereof.

We assume that the semantic self-information of a block rougly correlates with its self-information in terms of the number of code-word-value pairs. This assumption is supported by the fact that blocks with a small number of code-

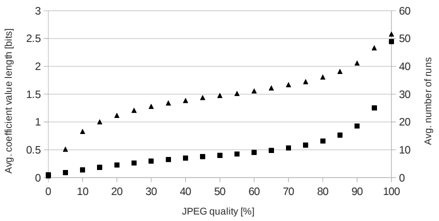


Figure 4: Average number of runs per block (squares) and average coefficient value bit length (triangles) over JPEG quality for the JPEG-compressed LIVE reference picture set [15]

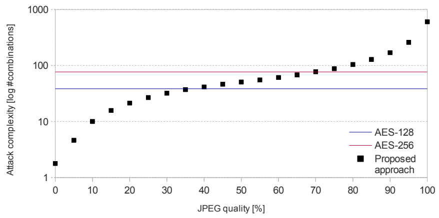


Figure 5: Average attack complexity over JPEG quality for the JPEG-compressed LIVE reference picture set [15] for the proposed approach and AES for comparison

word-value pairs (e.g. 1 or 2) are very unlikely to compromise the content of the whole picture, thus having a low amount of semantic self-information. Note that semantic self-information is hard to measure and therefore requires a justifiable approximation. Thus, we use the average number of code-word-value pairs to represent a block with a medium to high amount of self-information as a practical approximation of a possibly critical block of the picture.

Figure 4 depicts the average number of runs per block and the average length of coefficient values as functions of JPEG quality for the reference pictures of the LIVE data base. Both functions increase monotonically with quality, showing that pictures with finer quantization contain a higher number of runs per block and longer coefficient values. This allows for a higher number of combinations, making an attack on an iMCU harder.

Using the average values $n(q)$ and $l(q)$ at quality q instead of all n_i and $l_{i,j}$, respectively, yields a simplified equation for the overall number $N(q)$ of combinations dependent on the JPEG quality q:

$$N(q) = 2^{l(q)\cdot m\cdot n(q)} \cdot (n(q)!)^m \cdot \prod_{k=1}^{h} n_h(k)! \qquad (5)$$

Using the Gamma function as an extension of the factorial function which is only defined for natural numbered arguments, $N(q)$ can be expressed as

$$N(q) = 2^{l(q)\cdot m\cdot n(q)} \cdot (\Gamma(n(q)+1))^m \cdot \prod_{k=1}^{h} n_h(k)! \qquad (6)$$

Thus, an attack on an iMCU composed of 4:2:0 subsampled (i.e. $m = 6$ as entailed by the JPEG standard [4]) average blocks compressed with JPEG quality q requires trying

$$N(q) = 2^{6l(q)\cdot n(q)} \cdot (\Gamma(n(q)+1))^6 \cdot 4! \cdot 2! \qquad (7)$$

combinations, if both chroma components' AC coefficients use the same Huffman table. For a JPEG quality of 75% (which is the default value of the JPEG reference encoder), this yields $N(75) \approx 10^{87}$, which is greater than the number of possible 256 bit keys, thus making a brute-force attack on the AES key more efficient than trying to reorder and descramble the iMCU. Figure 5 illustrates this and a comparison to AES's attack complexity for different JPEG quality values.

Note that each iMCU can be attacked separately, thereby possibly revealing enough information about the picture that the rest of the picture's iMCUs do not need to be decrypted. This way, the total number of combinations for a full picture does not reprent a valid metric for the number of combinations to try for an attack.

Note that an attacker may eliminate some orderings of code-word-value pairs as high values of high frequency AC coefficients (most of all chroma) are very unlikely to appear in natural images [11]. This reduces the effective values of n_i and $n(q)$, respectively. However, it is hard to quantify the actual reduction as it depends on the picture's content, potentially known signal characteristics and the coefficient distribution of the attacked iMCU's blocks.

Regarding known-plaintext attacks, AES is considered to be not vulnerable [3]. If an attacker has both, the original and the encrypted JPEG picture, deriving the key from the permutations and scrambled bits is nearly as hard as a brute-force attack on the AES key itself [2] which is considered infeasible for 256 bit keys by today's standards.

4. FUTURE WORK

Multiple extensions of our approach are possible and remain future work: firstly, the DC coefficient differences of each block could be scrambled similar to the AC coefficient values, increasing the total number of possible combinations to try for decryption. Secondly, blocks between different iMCUs could be swapped as long as the corresponding blocks in the iMCUs use a the same Huffman tables, yet increasing the total number of possible combinations. Note that both extensions are easy to implement and preserve the length of the bit stream.

Finally, it is possible to use our proposed approach for RoI encryption. iMCUs containing the RoIs can be encrypted while the rest of the picture stays intact. Although limiting the encryption to a set of iMCUs is trivial, signalling them is not, if the length is to be preserved. However, if this limitation is lifted, embedding the RoI information can be done by inserting a comment segment into the bit stream which contains a bitmap of all iMCUs where a one denotes that the iMCU is encrypted, while a zero denotes that it is not. Such a segment increases the file size by the marker size (2 bytes) plus its length field (2 bytes) plus the size of the bitmap, i.e. $\lceil \frac{n_{iMCU}}{8} \rceil$ bytes where n_{iMCU} denotes the number of iMCUs in the picture.

5. CONCLUSION

We proposed a new approach to encrypt JPEG-compressed pictures by performing swap and scramble operations on their bit streams in a format-compliant and length-preserving way. Furthermore, we showed the practical infeasibility of both, brute-force and known-plaintext attacks. Given the fact that our approach operates on bit stream level and does not require any recompression, it can be considered faster than existing non-length-preserving approaches with a comparable level of security. Additionally, our approach allows for RoI encryption, which makes it usable for surveillance applications.

6. ACKNOWLEDGMENTS

This work is supported by FFG Bridge project 832082.

7. REFERENCES

[1] B. Bhargava, C. Shi, and Y. Wang. MPEG video encryption algorithms. *Multimedia Tools and Applications*, 24(1):57–79, 2004.

[2] A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique Cryptanalysis of the Full AES. In D. Lee and X. Wang, editors, *Advances in Cryptology (ASIACRYPT 2011)*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer Berlin / Heidelberg, 2011.

[3] J. Daemen and V. Rijmen. *The Design of Rijndael: AES — The Advanced Encryption Standard*. Springer Verlag, 2002.

[4] ITU-T T.81. Digital compression and coding of continuous-tone still images — requirements and guidelines, Sept. 1992. Also published as ISO/IEC IS 10918-1.

[5] C. Kailasanathan. Compression performance of JPEG encryption scheme. In *Proceedings of the 14th International IEEE Conference on Digital Signal Processing, DSP '02*, July 2002.

[6] D. A. Kerr. Chrominance Subsampling in Digital Images. `http://dougkerr.net/pumpkin/articles/Subsampling.pdf`, Jan. 2012.

[7] M. Khan, V. Jeoti, and M. Khan. Perceptual encryption of JPEG compressed images using DCT coefficients and splitting of DC coefficients into bitplanes. In *2010 International Conference on Intelligent and Advanced Systems (ICIAS)*, pages 1–6, June 2010.

[8] S. Lian, J. Sun, and Z. Wang. A novel image encryption scheme based-on jpeg encoding. In *Proceedings of the Eighth International Conference on Information Visualisation 2004 (IV 2004)*, pages 217–220, July 2004.

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

[10] X. Niu, C. Zhou, J. Ding, and B. Yang. JPEG Encryption with File Size Preservation. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing 2008 (IIHMSP '08)*, pages 308–311, Aug. 2008.

[11] W. Pennebaker and J. Mitchell. *JPEG – Still image compression standard*. Van Nostrand Reinhold, New York, 1993.

[12] U. Potdar, K. T. Talele, and S. T. Gandhe. Comparison of MPEG video encryption algorithms. In *Proceedings of the International Conference on Advances in Computing, Communication and Control*, pages 289–294, New York, NY, USA, 2009. ACM.

[13] W. Puech and J. M. Rodrigues. Crypto-Compression of Medical Images by Selective Encryption of DCT. In *European Signal Processing Conference 2005 (EUSIPCO'05)*, Sept. 2005.

[14] W. Puech and J. M. Rodrigues. Analysis and cryptanalysis of a selective encryption method for JPEG images. In *WIAMIS '07: Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services*, Washington, DC, USA, 2007. IEEE Computer Society.

[15] K. Seshadrinathan, R. Soundararajan, A. Bovik, and L. Cormack. Study of Subjective and Objective Quality Assessment of Video. *IEEE Transactions on Image Processing*, 19(6):1427–1441, June 2010.

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

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

[18] C.-P. Wu and C.-C. J. Kuo. Fast encryption methods for audiovisual data confidentiality. In *SPIE Photonics East - Symposium on Voice, Video, and Data Communications*, volume 4209, pages 284–295, Boston, MA, USA, Nov. 2000.

[19] B. Yang, C.-Q. Zhou, C. Busch, and X.-M. Niu. Transparent and perceptually enhanced JPEG image encryption. In *16th International Conference on Digital Signal Processing*, pages 1–6, July 2009.

[20] Y. Ye, X. Zhengquan, and L. Wei. A Compressed Video Encryption Approach Based on Spatial Shuffling. In *8th International Conference on Signal Processing*, volume 4, pages 16–20, Nov. 2006.

Note: This is a pre-print version subject to changes in formatting

Bitstream-based JPEG Encryption in Real-time

Stefan Auer
Salzburg University of Applied Sciences, Austria
Alexander Bliem
Salzburg University of Applied Sciences, Austria
Dominik Engel
Salzburg University of Applied Sciences, Austria
Andreas Uhl
University of Salzburg, Austria
Andreas Unterweger*
University of Salzburg, Austria

ABSTRACT

We propose a framework to encrypt Baseline JPEG files directly at bitstream level, i.e., without the need to recompress them. Our approach enables encrypting more than 25 pictures per second in VGA resolution, allowing real-time operation in typical video surveillance applications. In addition, our approach preserves the length of the bitstream while being completely format-compliant. Furthermore, we show that an attack on the encryption process, which partly relies on AES, is practically infeasible.

Keywords: JPEG, Encryption, Real-time, Length-Preserving, Framework

INTRODUCTION

The encryption of compressed images to ensure privacy is an active research topic for a variety of different compressed image and video formats. For JPEG-compressed images (International Telecommunication Union, 1992) in particular, several approaches exist due to the widespread use of this image format. Most of them require recompressing the original data to some extent. The method proposed in this paper operates on bitstream level and uses only swap and scramble operations, which makes it very fast.

A number of approaches have been proposed which do either not preserve the length of the original file or break format compliance. These include techniques such as zig zag permutation (Kailasanathan, 2002; Tang, 1996), which significantly increases the file size, and the use of permuted Huffman tables (Wu & Kuo, 2000). Similarly, DC bit plane scrambling as proposed for example by Khan, Jeoti, & Kahn (2010) increases the file size and is therefore not length preserving as opposed to our approach.

In terms of simple length-preserving encryption algorithms on DCT-based images and videos, pseudo-randomly toggling DC and/or AC coefficient signs as proposed for example by Potdar, Talele, & Gandhe (2009), or Bhargava, Shi, & Wang (2003) is frequently used.

Note: This is a pre-print version subject to changes in formatting

However, the attack complexity for breaking such themes is significantly lower than in our approach as we encrypt multiple bits per coefficient instead of only one (the sign bit).

Similarly, encrypting a limited number of bits on bitstream level starting from the DC coefficient (Puech & Rodrigues, 2005) or the high-frequency AC coefficients (Puech & Rodrigues, 2007), respectively, is of lower security as compared to the proposed approach which encrypts all coefficients and additionally increases the complexity by reordering blocks. The technique of reordering all blocks within a picture, which is used as part of our approach in a spatially limited fashion, has already been proposed in Ye, Zhengquan, and Wei (2006), Lian, Sun, and Wang (2004) and Niu, Zhou, Ding, and Yang (2008) and analyzed in Wen, Severa, Zeng, Luttrell, and Jin (2002) and others. Although it increases the total attack complexity depending on the picture size, it does not allow for RoI encryption without a significant decrease in attack complexity, as opposed to our approach.

In terms of code-word-based techniques such as the one we propose, only a small number approaches have been published. Besides swapping code words of equal length between blocks for AC value histogram spreading as proposed in Yang, Zhou, Busch, and Niu (2009), a method to shuffle code words with the same in-block position between blocks exists for MPEG-4 (Wen et al., 2002) which could also be applied to JPEG pictures. However, the latter approach may yield non-format-compliant bitstreams and both methods are not intended to be used for RoI encryption as opposed to our approach.

Another method described in Wen et al. (2002) encrypts multiple concatenated VLC (Variable Length Code) symbols and maps them to another string of valid VLC symbols so that the total length is preserved. Note that this approach, which has been applied to MPEG-4 bitstreams, cannot be used for JPEG as, in the latter, each Huffman code word is followed by a signed coefficient residual represented by a number of bits which is encoded in the Huffman code word. Changing the code words in a length-preserving way changes the number of coefficient bits encoded in the Huffman code word as opposed to the actual subsequent bits in the bitstream, making the bitstream parser get out of sync and thus breaking format compliance.

Note that our bitstream-based approach is designed for encryption without the need for recompression, which is useful when there is no possibility to intercept the encoding process. One practical use case is the encryption of pictures from surveillance cameras, most of which send streams of JPEG pictures which are already encoded. Although real-time recompression is possible with state-of-the-art hardware, omitting this step may save equipment and costs.

This paper is structured as follows: In the “Bitstream Encryption” section we describe our encryption approach. Subsequently, in the “Security Analysis” section, we give an estimation of the effort required for a successful attack on a picture encrypted with our approach. Next, we present our framework which implements our encryption approach in the “Real-time Encryption Framework” section. Finally, we evaluate the performance of the proposed encryption framework in the “Performance evaluation” section before concluding the paper.

This paper is an extension of our previous work (Unterweger, 2012) which introduced our length-preserving JPEG encryption approach. In this paper, we add an additional DC coefficient encryption step and introduce a practical implementation which is capable of encrypting JPEG images using our approach. Furthermore, we evaluate the performance of our implementation thoroughly, showing that it is suitable for encoding typical JPEG-compressed surveillance camera output in real-time, which makes additional storage facilities unnecessary.

Note: This is a pre-print version subject to changes in formatting

Bitstream Encryption

In baseline JPEG, 8x8 blocks of cosine-transformed quantized AC coefficients are zig-zag scanned and run-length coded before being Huffman coded. Hereby, the Huffman code words only encode run-length pairs as symbols, whereas the actual coefficient values are written directly to the bitstream as signed residue from 0 or -2^s+1, respectively, where *s* denotes the length part of the run-length symbol.

Our approach consists of four different operations. First, the order of the run-length coded symbols together with their corresponding coefficient values (referred to as code-word-value pairs henceforth) is permuted. Second, the coefficient value bits are scrambled and third, the order of all blocks within an iMCU (interleaved Minimum Coding Unit) which use the same Huffman table is permuted. Finally, the DC coefficients are scrambled using the approach proposed by Niu et al. (2008).

As DC coefficient scrambling is known to be easy to break, it is performed as a separate step after the first three using a different key. The purpose of it is purely to make the image appear "more" encrypted to a human viewer. Note that the remaining three steps are already relatively secure, as shown in the "Security Analysis" section.

The fourth step, Niu et al.'s (2008) approach, is not described in detail herein, noting only that it is format-compliant and length-preserving. Both, the second and third step, are described in detail at the end of this section, while the subsequent paragraphs describe the first operation, i.e., the permutation of the order of code-word-value pairs.

Permutations of this order lead to a change of the order of the zero runs in each block, thereby altering the positions of the coefficient values within the 8x8 block. Figure 1 depicts this by example.

Note: This is a pre-print version subject to changes in formatting

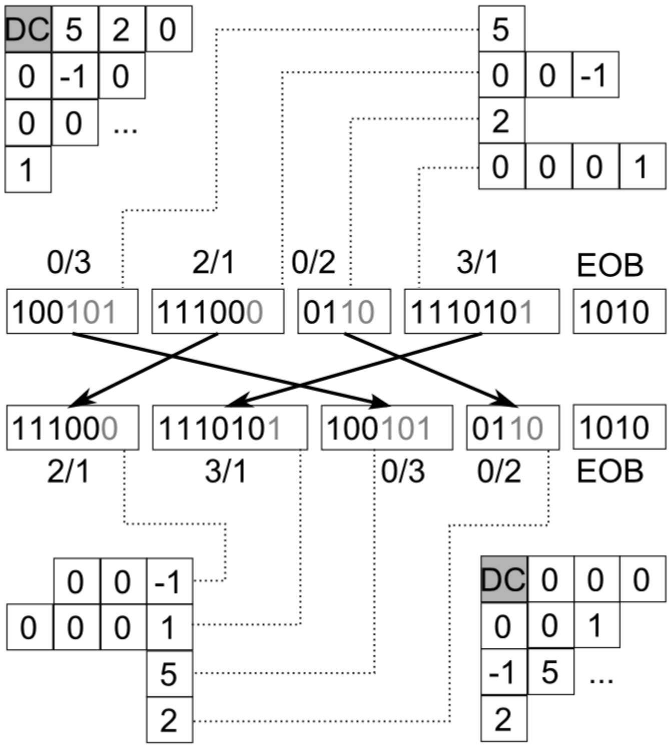


Figure 1. Example of run-length permutation: The order of Huffman coded run-length symbols and their corresponding coefficient values is permuted, thereby changing the position of the values in the coefficient matrix.

On the top left, an exemplary block with four non-zero coefficient values is shown. The dots denote that the rest of the coefficients are zero. The DC coefficient is neither changed nor considered in this step. On the top right, the zig-zag scanned values of the exemplary block are depicted and grouped with their preceding zeros. Each of these groups is coded as a Huffman code word (black) of the run-length symbol (depicted on top of each Huffman code word) and the coefficient value (gray). For example, the first coefficient (5), which is preceded by no (i.e., 0) zeros, requires three bits (101) to be represented, thus leading to a run-length of 0/3. As the rest of the blocks' coefficients apart from the four ones depicted are zero, an EOB (End Of Block) is signaled.

By swapping the groups of Huffman code words and coefficient values (if there is more than one group), the zero runs and therefore the position of the coefficient values within the block change, as depicted at the bottom of Figure 1. However, the bitstream remains format-

Note: This is a pre-print version subject to changes in formatting

compliant as the exchange of code words does neither change the Huffman codes themselves nor does it change the total number of coefficients. In addition, it preserves the length of the JPEG file.

The code-word-value pair order permutation through element-wise reordering is derived as follows. Before processing a JPEG file, an AES (National Institute of Standards and Technology, 2001) implementation in OFB (Output Feedback) mode is initialized with a given initialization vector and key, which can be file- or user-dependent. It then serves as a PRNG (Pseudo-Random Number Generator) by using n AES-encrypted output bits, where 2^n is the desired range of the PRNG. Note that any cryptographic PRNG could be used here.

Code-word-value pair order permutation is then performed by swapping the current code word and its corresponding coefficient value at position i with the code-word-value pair at position $rand(n)$ where *rand* denotes a call to the AES-based PRNG with an upper value bound of n and n is equal to the number of total code-word-value pairs. For the example in Figure 1, $n = 4$, yielding the consumption of two encrypted output bits of the AES encoder per possible swap operation.

In addition to code-word-value pair order permutation, our approach changes the coefficients' values in the bitstream (gray bits in Figure 1). This is done by toggling each of the n value bits depending on whether or not the AES-based PRNG described above returns a binary zero or one when using one bit. Similar to the run-length order permutation, this does not change the length of the JPEG file as the value bits actually represent a signed residual of fixed size per code word (see above).

Furthermore, the order of all blocks using the same Huffman table within an iMCU is permuted. Figure 2 shows an example with 4:2:0 sub-sampling (Kerr, 2013) where the U and the V block use the same Huffman table (marked gray). After the permutation, the order of U and V is switched, with the bitstream still being format-compliant. The permutation itself is derived as described for run-length permutations above. Note that no code-word-value pairs are exchanged among the blocks as this would break format compliance – only whole blocks with all their code-word-value pairs are exchanged. Again, this does not change the length of the JPEG file.

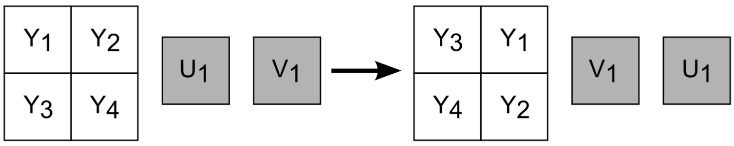


Figure 2. Example of block order permutation: The order of blocks using the same Huffman code words within an iMCU is permuted. In this example, the Y blocks use one set of Huffman code words (white), while both, the U and the V block, use another (gray).

Figure 3 (c) shows an example of a picture with a JPEG quality of 75% that is encrypted with our approach. Note that the number of possible permutations in blocks with few code-word-value pairs and/or small coefficient values (e.g., in the area above the woman's head) is small, making those blocks appear less noisy due to the lack of high frequency components. Conversely, the other blocks exhibit significant distortion due to the reordering and scrambling,

Note: This is a pre-print version subject to changes in formatting

revealing that the local encryption strength of our approach depends on the amount of information contained in a block as explained in more detail in the next section.

a) **b)** **c)**

Figure 3. Example of a picture from the LIVE reference picture set (Seshadrinathan, Soundararajan, & Bovik, 2010) before (a) and after (c) encryption with the proposed method. (b) depicts a variant of our approach where DC coefficient scrambling is omitted.

Note that although the woman's silhouette is recognizable in the encrypted picture when DC coefficient scrambling is omitted (Figure 3 (b)), no more details (like facial characteristics) can be extracted from it. All information contained in high frequency coefficients is lost without proper decryption and therefore does not compromise the security of our approach on small scales, even if there is a successful attack on the scrambled DC coefficients.

Security Analysis

In order to assess the cryptographic security of our approach, its three main components are analyzed in terms of attack complexity, i.e., the number of possible combinations per iMCU and the probability of success for key extraction in a known-plaintext attack. The key space depends on the AES key size and can be up to $2^{256} > 10^{77}$ for AES with 256 bit keys (National Institute of Standards and Technology, 2001).

The approach described in the previous section relies on three independent scrambling mechanisms, disregarding the DC coefficient scrambling which can easily be circumvented: permuting the order of code-word-value pairs, toggling value bits and permuting the order of blocks within an iMCU. Due to their independence, the number of possible combinations can be analyzed separately and eventually multiplied to yield the overall number of possible combinations.

Let m denote the total number of blocks in an iMCU and let n_i denote the total number of code words in the i^{th} block of an iMCU. Let $l_{i,j}$ denote the length of the j^{th} code-word-value pair of the i^{th} block of an iMCU in bits. The number of possible values (i.e., bit combinations) $c_v(i, j)$ for each value is $2^{l_{i,j}}$, thus being

Note: This is a pre-print version subject to changes in formatting

$$N_v = \prod_{i=1}^{m} \prod_{j=1}^{n_i} c_v(i,j) = \prod_{i=1}^{m} \prod_{j=1}^{n_i} 2^{l_{i,j}} \quad (1)$$

for all blocks within an iMCU.

The number of permutations $p_{rlv}(i)$ of code-word-value pairs of each block is $n_i!$, where $x!$ denotes the factorial of x. Thus, the total number of code-word-value pair permutations is

$$N_{rlv} = \prod_{i=1}^{m} p_{rlv}(i) = \prod_{i=1}^{m} n_i! \quad (2)$$

for all blocks within an iMCU. Similarly, the number of block permutations $p_b(x)$ for x blocks which use the same Huffman code words is $x!$, thus being

$$N_b = \prod_{k=1}^{h} p_b(n_h(k)) = \prod_{k=1}^{h} n_h(k)! \quad (3)$$

for all blocks within an iMCU, where h denotes the total number of different AC Huffman tables and $n_h(k)$ is the number of blocks using the k^{th} Huffman table so that

$$\sum_{k=1}^{h} n_h(k) = m \quad (4)$$

In total, this yields an overall number N of possible combinations per iMCU of

$$N = N_v \cdot N_{rlv} \cdot N_b = \prod_{i=1}^{m} \prod_{j=1}^{n_i} 2^{l_{i,j}} \cdot \prod_{i=1}^{m} n_i! \cdot \prod_{k=1}^{h} n_h(k)! \quad (5)$$

In order to estimate the values for $l_{i,j}$ and n_i for typical natural JPEG pictures, the reference pictures of the LIVE data base presented in Seshadrinathan et al. (2010) are encoded with different quality settings (between 0 and 100% with 5% step size) using the JPEG reference encoder. The encoded files are analyzed in terms of the average number of runs per block and the average length of coefficient values.

We consider the average values to be an appropriate measure for the following reason: The attack complexity of our algorithm depends on the number of code-word-value pairs within a block, i.e., it varies with its number of non-zero coefficients. The distribution of the latter (not depicted) reveals that the information content of a block with a high number of code-word-value pairs is greater than that of a block with a small number thereof.

We assume that the semantic information content, i.e., the amount of information a human can extract, of a block roughly correlates with its information content in terms of the number of code-word-value pairs. This assumption is supported by the fact that blocks with a small number of code-word-value pairs (e.g., one or two) are very unlikely to compromise the content on a small scale, thus having a low amount of semantic information content.

Note that semantic information content is hard to measure and therefore requires a justifiable approximation. Thus, we use the average number of code-word-value pairs to represent a block with a medium to high amount of information content as a practical approximation of a possibly critical block of the picture.

Figure 4 depicts the average number of runs per block and the average length of coefficient values as functions of JPEG quality for the reference pictures of the LIVE data base. Both functions increase monotonically with quality, showing that pictures with finer quantization contain a higher number of runs per block and longer coefficient values. This allows for a higher number of combinations, making an attack on an iMCU harder.

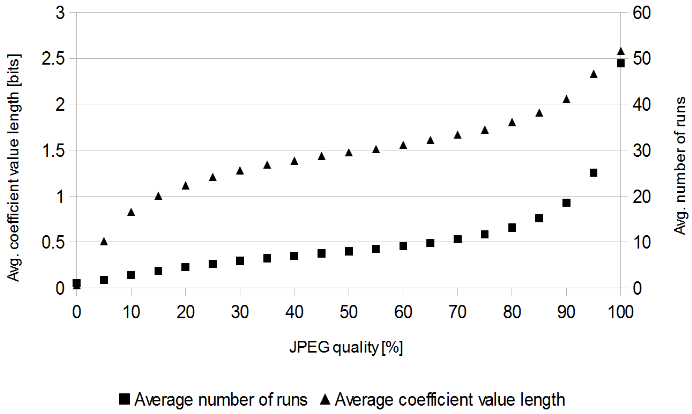


Figure 4. Average number of runs per block (squares) and average coefficient value bit length (triangles) over JPEG quality for the JPEG-compressed LIVE reference picture set (Seshadrinathan et al., 2010).

Using the average values $n(q)$ and $l(q)$ at quality q instead of all n_i and $l_{i,j}$, respectively, yields a simplified equation for the overall number $N(q)$ of combinations dependent on the JPEG quality q:

$$N(q)=2^{l(q)\cdot m\cdot n(q)}\cdot(n(q)!)^m\cdot\prod_{k=1}^{h} n_h(k)! \qquad (6)$$

Using the Gamma function as an extension of the factorial function which is only defined for natural numbered arguments, $N(q)$ can be expressed as

$$N(q)=2^{l(q)\cdot m\cdot n(q)}\cdot(\Gamma(n(q)+1))^m\cdot\prod_{k=1}^{h} n_h(k)! \qquad (7)$$

Thus, an attack on an iMCU composed of 4:2:0 sub-sampled (i.e., m = 6 as entailed by the JPEG standard (International Telecommunication Union, 1992)) average blocks compressed with JPEG quality q requires trying

$$N(q)=2^{6l(q)\cdot n(q)}\cdot(\Gamma(n(q)+1))^6\cdot 4!\cdot 2!=48\cdot 2^{6l(q)\cdot n(q)}\cdot(\Gamma(n(q)+1))^6 \qquad (8)$$

combinations, if the AC coefficients of both chroma components use the same Huffman table. For a JPEG quality of 75% (which is the default value of the JPEG reference encoder), this yields $N(75) > 10^{87}$, which is greater than the number of possible 256 bit keys, thus making a

Note: This is a pre-print version subject to changes in formatting

brute-force attack on the AES key more efficient than trying to reorder and descramble the iMCU. Figure 5 illustrates this and a comparison to the attack complexity for AES for different JPEG quality values.

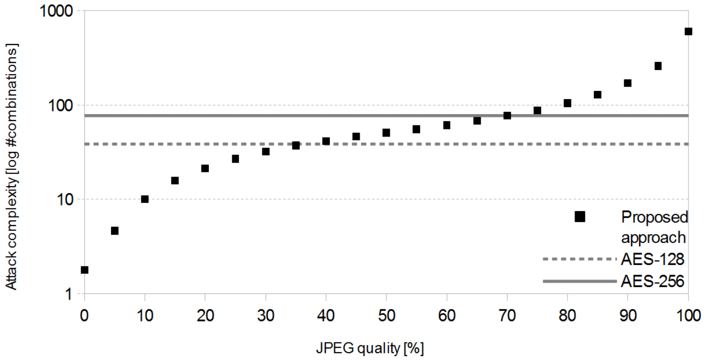


Figure 5. Average attack complexity over JPEG quality for the JPEG-compressed LIVE reference picture set (Seshadrinathan et al., 2010) for the proposed approach and AES for comparison.

Note that each iMCU can be attacked separately, thereby possibly revealing enough information about the picture that the rest of the picture's iMCUs do not need to be decrypted. This way, the total number of combinations for a full picture does not represent a valid metric for the number of combinations to try for an attack.

Note that an attacker may eliminate some orderings of code-word-value pairs as high values of high frequency AC coefficients (most of all chrominance) are very unlikely to appear in natural images (Pennebaker, & Mitchell, 1993). This reduces the effective values of n_i and $n(q)$, respectively. However, it is hard to quantify the actual reduction as it depends on the picture content, potentially known signal characteristics and the coefficient distribution of the blocks in the attacked iMCU.

Regarding known-plaintext attacks, AES is considered to be not vulnerable (Daemen & Rijmen, 2002). If an attacker has both, the original and the encrypted JPEG picture, deriving the key from the permutations and scrambled bits is nearly as hard as a brute-force attack on the AES key itself (Bogdanov, Khovratovich, & Rechberger, 2011), which is considered infeasible for 256 bit keys by today's standards.

Real-time Encryption Framework

To evaluate our encryption approach, we created a framework which implements it. Our framework consists of two parts: The first part is a DLL (Dynamic-Link Library) written in C, which is theoretically platform-independent and performs the actual encryption and decryption. The second part is a .NET Windows Forms GUI (Graphical User Interface) implemented in C#

Note: This is a pre-print version subject to changes in formatting

which enables easy selection of JPEG live streams or stored JPEG pictures. The GUI calls the DLL using PInvoke on Windows (Microsoft, 2013a).

The C DLL is based on the open source project NanoJPEG (Fielder, 2013) and uses the entropy coding implementation of Barrett (2013). Additionally, the AES implementation of Sevilla (2013) is used for the AES-based PRNG as described in the “Bitstream Encryption” section.

The encryption is performed as follows after the bitstream to be encoded is provided to the DLL: First, a conservative approximation of the required amount of memory to store both, the original and the encrypted bitstreams, is made based on the size of the original bitstream. Second, the original bitstream in memory is processed and encrypted on-the-fly to yield the encrypted bitstream. Finally, the encrypted bitstream is written to a memory location provided by the DLL's caller.

The on-the-fly encryption step distinguishes between data which needs to be encrypted and the remainder which can be copied unmodified. Processing the input bitstream from start to end, markers are parsed and evaluated. SOF (Start of Frame), DHT (Define Huffman Table) and SOS (Start Of Scan) markers are evaluated and simultaneously written to the memory location of the output bitstream.

For the actual scan data, decoding is performed at Huffman-code level in order to distinguish the Huffman code words from one another to subsequently apply the encryption algorithm described in the “Bitstream Encryption” section. Once the code-word-value pairs of one iMCU are decoded and stored in memory, the encryption algorithm is applied. Subsequently, encrypted image data is written to the corresponding memory location and the next iMCU is processed, until the end of the scan is reached.

Performance Evaluation

In this section, we assess the performance of our implementation. We consider real-time encoding of JPEG-compressed input streams from surveillance cameras to be one of the main applications of our implementation. Thus, our evaluation focuses solely on execution time and real-time constraints.

As this use case does usually not require a GUI, but exhibits heavily use-case-dependent I/O (Input/Output) performance, we limit our measurements to the net execution time of our DLL's encryption routine which is described in the “Real-time Encryption Framework” section, thus disregarding execution time spent for I/O tasks.

All measurements were performed on an Intel Core 2 Duo T9600 CPU running at 2.8 GHz. In order to minimize measurement errors due to background processes, we booted Windows 7 32-bit in “Safe Mode with Command Prompt” and executed the encryption process which invoked our C DLL so that it was bound to one fixed CPU core with the highest possible process priority. All execution times were measured using QueryPerformanceCounter (Microsoft, 2013b) calls as recommended by Microsoft (2013c) for accurate measurements on multi-core systems.

We distinguish between measurements of on-line and off-line encryption. In order to compensate for caching effects and fluctuations, each picture to be encrypted off-line was in fact

Note: This is a pre-print version subject to changes in formatting

encrypted five times for cache warming, i.e., without considering execution time, and subsequently encrypted 20 times, yielding a final average execution time. In contrast, all pictures to be encrypted on-line were encrypted only once, i.e., without cache warming and without averaging execution times. This effectively simulates practical execution conditions, in which pictures are encrypted one after another without any potential benefits from caching.

To simulate different working conditions, we use different sets of input pictures: For a practical on-line field test with actual surveillance camera pictures, we use the BEHAVEDATA (Laghaee, 2013) picture sets, courtesy of EPSRC project GR/S98146, consisting of 11200 and 62366 JPEG images corresponding to 95 and 100% quality, respectively, each with a spatial resolution of 640x480 pixels (VGA).

To evaluate the effect of JPEG quality on off-line execution time, we use the 29 images of the LIVE reference picture set (Seshadrinathan et al., 2010), ranging between spatial resolutions of 634x438 and 768x512 pixels. We encode them using the standard JPEG encoder with default settings and quality values between 0 and 100% with a step size of 5%.

To evaluate the effect of spatial resolution on off-line execution time, we use the 24 Kodak high-resolution images from SCIEN (2013) with a spatial resolution of 3072x2048 pixels each. Starting at this resolution, we derive smaller versions of the images by symmetrically cropping them in steps of 96/64 pixels in width/height using ImageMagick 6.6.0-4 (ImageMagick Studio, 2013), maintaining both, the original aspect ratio of 3:2 and the images' centers without the need to perform interpolation of any kind. The cropped images are then encoded using the standard JPEG encoder with default settings, i.e., a quality of 75%.

The results of the practical on-line field test in terms of maximum encryption time per picture are 13.86 ms for the 95% quality picture set and 29.80 ms for the 100% quality picture set, respectively. This demonstrates the capability of our implementation to perform hard real-time encoding at a frame rate of at least 25 frames per second, which would allow for a maximum execution time of 40 ms per frame.

Note that the average execution times are 12.11 ms with a standard deviation of 0.87 and 26.34 ms with a standard deviation of 1.16, respectively, which allows for even higher performance under soft real-time conditions, i.e., when buffering makes fluctuations in execution acceptable. In conclusion, real-time processing at 25 frames per second at VGA (640x480) resolution with up to 100% JPEG quality is possible using our implementation.

The remainder of this section is dedicated to the evaluation of off-line encoding performance. Figure 6 shows the effect of JPEG quality on execution time. Except for a quality value of 100%, all pictures can be encrypted in soft real-time at 25 frames per second. Note that this does not contradict the on-line measurements as the picture sizes used in this experiment (see above) are mostly larger than VGA and exhibit stronger fluctuations.

Note: This is a pre-print version subject to changes in formatting

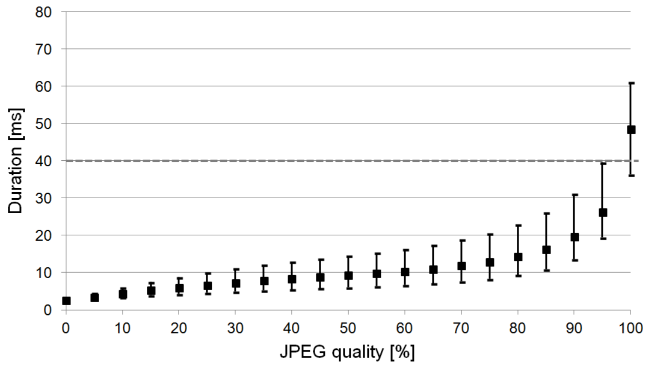


Figure 6. The effect of JPEG quality on net execution time. Squares depict the average execution time of all measurements per quality value. The dashed gray line indicates the threshold for soft real-time encryption at 25 frames per second.

Although execution time increases monotonically with JPEG quality, the increase is larger for higher quality values. This is due to the exponential increase in the number of non-zero coefficients at high quality values, which results in a large number of run-length-value pairs to be swapped and scrambled. Conversely, the number of non-zero coefficients is relatively low and changes sub-linearly for most low quality values.

Figure 7 shows the effect of spatial resolution on execution time. All pictures of the data set show a nearly linear dependency between the pictures' spatial resolution and the net execution time required for their encryption. The main difference between the pictures is the number of their non-zero coefficients induced by their content, which reflects in the slope of the quasi-linear increase in execution time. Thus, pictures with similar image characteristics exhibit similar slopes.

Note: This is a pre-print version subject to changes in formatting

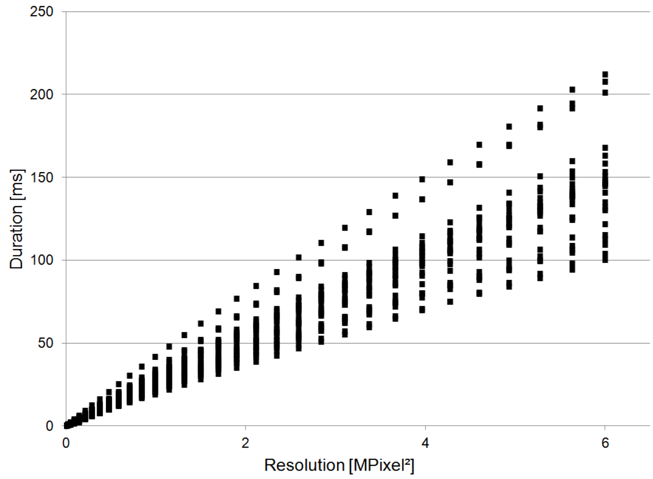


Figure 7. The effect of spatial resolution on net execution time. The different curves correspond to different images, revealing that the image content influences the slope of the quasi-linear increase in execution time as the spatial resolution increases.

As the number of non-zero coefficients directly affects the file size of the corresponding JPEG images, Figure 8 illustrates the results of Figure 7 in terms of file size instead of spatial resolution. It is not surprising that the execution time is, again, nearly linearly dependent on the file size. As our encryption approach is bitstream based, this is a desirable property, as the picture size and content can be used to estimate the required encryption time.

Note: This is a pre-print version subject to changes in formatting

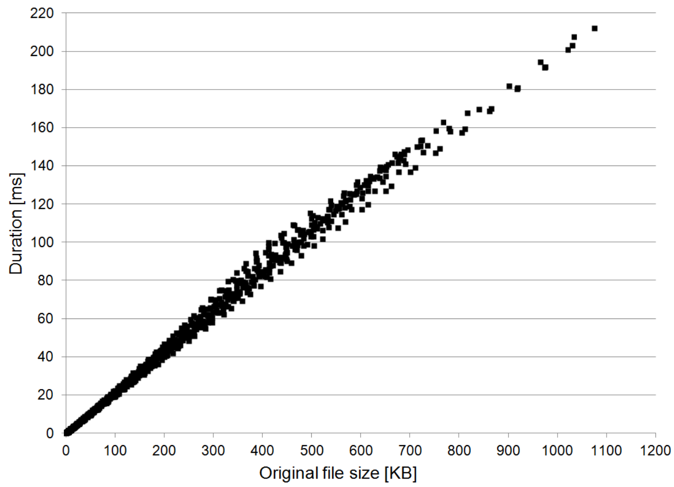


Figure 8. The effect of JPEG file size on net execution time. The different curves correspond to the different images of Figure 7, showing that there is a high correlation between the spatial dimension of a picture and its JPEG file size.

Future Work

Multiple extensions of our approach are possible and remain future work: First, blocks between different iMCUs could be swapped as long as the corresponding blocks in the iMCUs use the same Huffman tables, yet increasing the total number of possible combinations. This extension is easy to implement and preserves the length of the bitstream.

Second, it is possible to use our proposed approach for RoI encryption. iMCUs containing the RoIs can be encrypted while the rest of the picture stays intact. If DC coefficient encryption is not used at all, this form of RoI encryption is length-preserving. However, if DC coefficient encryption is applied, the DC differences around each RoI need to be corrected in order to keep the non-encrypted picture areas undistorted. As the corrected values may differ in length, this approach would not be length-preserving anymore.

Although limiting the encryption to a set of iMCUs is trivial, signaling them is not, if the length is to be preserved. However, if this limitation is lifted, embedding the RoI information can be done by inserting a comment segment into the bitstream which contains for example a bitmap of all iMCUs where a one denotes that the iMCU is encrypted, while a zero denotes that it is not. Such a comment segment increases the file size by the marker size (two bytes) plus its length field (two bytes) plus the size of the bitmap or any other desired form of RoI position/size encoding.

Note: This is a pre-print version subject to changes in formatting

CONCLUSION

We proposed a framework which encrypts JPEG files by performing swap and scramble operations on their respective bitstreams in a format-compliant and length-preserving way. As our encryption approach operates at bitstream level, it does not require recompression, thus enabling real-time encryption for at least 25 pictures per second in VGA resolution. Furthermore, we showed the practical infeasibility of both, brute-force and known-plaintext attacks as well as the extensibility of our apprpach to support RoI encryption, which makes it suitable for surveillance applications.

ACKNOWLEDGEMENTS

This work is supported by FFG Bridge project 832082.

REFERENCES

Barrett, S. (2013). *stbi-1.33 - public domain JPEG/PNG reader*. Retrieved February 6, 2013, from http://nothings.org/stb_image.c.

Bhargava, B., Shi, C., & Wang, Y. (2004). MPEG video encryption algorithms. *Multimedia Tools and Applications, 24*(1), 57-79.

Bogdanov, A., Khovratovich, D., & Rechberger, C. (2011). Biclique Cryptanalysis of the Full AES. In D. Lee, & X. Wang (Eds.), *Advances in Cryptology (ASIACRYPT 2011): Vol. 7073 of Lecture Notes in Computer Science* (pp. 344-371). Berlin / Heidelberg: Springer.

Daemen, J., & Rijmen, V. (2002). *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer Verlag.

Fiedler, M. (2013). *NanoJPEG: a compact JPEG decoder*. Retrieved February 6, 2013, from http://keyj.emphy.de/nanojpeg/.

ImageMagick Studio (2013). *Convert, Edit, and Compose Images*. Retrieved February 9, 2013, from http://www.imagemagick.org/script/index.php.

International Telecommunication Union (1992). *ITU-T T.81. Digital compression and coding of continuous-tone still images | Requirements and guidelines.* Geneva, Switzerland: International Telecommunication Union.

Kailasanathan, C. (2002). Compression performance of JPEG encryption scheme. In A. N. Skodrus, & A. G. Constuntinides (Eds.), *Proceedings of the 14th International IEEE Conference on Digital Signal Processing (DSP '02)*. IEEE.

Kerr, D. A. (2013). *Chrominance Subsampling in Digital Images*. Retrieved February 2, 2013, from http://dougkerr.net/pumpkin/articles/Subsampling.pdf.

Khan, M., Jeoti, V., & Khan, M. (2010). Perceptual encryption of JPEG compressed images using DCT coefficients and splitting of DC coefficients into bitplanes. In *2010 International Conference on Intelligent and Advanced Systems (ICIAS)* (pp. 1-6). IEEE.

Note: This is a pre-print version subject to changes in formatting

Laghaee, A. (2013). *BEHAVE Interactions Test Case Scenarios*. Retrieved February 9, 2013, from http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/INTERACTIONS/.

Lian, S., Sun, J., & Wang, Z. (2004). A novel image encryption scheme based-on JPEG encoding. In *Proceedings of the Eighth International Conference on Information Visualisation 2004 (IV 2004)* (pp. 217-220). IEEE.

Microsoft (2013a). *Calling Native Functions from Managed Code*. Retrieved February 6, 2013, from http://msdn.microsoft.com/en-us/library/ms235282.aspx.

Microsoft (2013b). *QueryPerformanceCounter function (Windows)*. Retrieved February 9, 2013, from http://msdn.microsoft.com/en-us/library/windows/desktop/ms644904%28v=vs.85%29.aspx.

Microsoft (2013c). *Game Timing and Multicore Processors (Windows)*. Retrieved February 9, 2013, from http://msdn.microsoft.com/en-us/library/windows/desktop/ee417693%28v=vs.85%29.aspx.

National Institute of Standards and Technology (2001). *FIPS-197 – Advanced Encryption Standard (AES)*. Gaithersburg, MD: National Institute of Standards and Technology.

Niu, X., Zhou, C. Ding, J., & Yang, B. (2008). JPEG Encryption with File Size Preservation. In J.-S. Pan, X. M. Niu, H.-C. Huang, & L. C. Jain (Eds.), *International Conference on Intelligent Information Hiding and Multimedia Signal Processing 2008 (IIHMSP '08)* (pp. 308-311). IEEE.

Pennebaker, W., & Mitchell, J. (1993). *JPEG – Still image compression standard*. New York, New York: Van Nostrand Reinhold.

Potdar, U., Talele, K. T., & Gandhe, S. T. (2009). Comparison of MPEG video encryption algorithms. In *ICAC3 '09: Proceedings of the International Conference on Advances in Computing, Communication and Control* (pp. 289-294). New York, New York: ACM.

Puech, W., & Rodrigues, J. M. (2005). Crypto-Compression of Medical Images by Selective Encryption of DCT. In *European Signal Processing Conference 2005 (EUSIPCO'05)*. EURASIP.

Puech, W., & Rodrigues, J. M. (2007). Analysis and cryptanalysis of a selective encryption method for JPEG images. In L. O'Conner (Ed.) *WIAMIS '07: Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services*. Los Alamitos, California: IEEE Computer Society.

SCIEN (2013). *Test Images and Videos*. Retrieved February 9, 2013, from http://scien.stanford.edu/index.php/test-images-and-videos/.

Seshadrinathan, K., Soundararajan, R., Bovik, A., & Cormack, L. (2010). Study of Subjective and Objective Quality Assessment of Video. *IEEE Transactions on Image Processing, 19*(6), 1427-1441.

Sevilla, R. R. (2013). *rijndael - An implementation of the Rijndael cipher*. Retrieved February 6, 2013, from http://www.opensource.apple.com/source/CPANInternal/CPANInternal-62/Crypt-Rijndael/rijndael.h.

Note: This is a pre-print version subject to changes in formatting

Tang, L. (1996). Methods for encrypting and decrypting MPEG video data efficiently. In P. Aigrain, W. Hall, T. D. C. Little, & V. M. Bove Jr. (Eds.) *Proceedings of the ACM Multimedia 1996* (pp. 219-229). ACM Press.

Unterweger, A., & Uhl, A. (2012) Length-preserving Bit-stream-based JPEG Encryption. In *MM&Sec'12: Proceedings of the 14th ACM Multimedia and Security Workshop* (pp. 85-89). New York, New York: ACM.

Wen, J., Severa, M., Zeng, W., Luttrell, M., & Jin, W. (2002). A format-compliant configurable encryption framework for access control of video. *IEEE Transactions on Circuits and Systems for Video Technology, 12*(6), 545-557.

Wu, C.-P., & Kuo, C.-C. J. (2000). Fast encryption methods for audiovisual data confidentiality. In *SPIE International Symposium on Voice, Video, and Data Communications, Vol. 4209* (pp. 284-295). SPIE.

Yang, B., Zhou, C.-Q., Busch, C., & and Niu, X.-M. (2009). Transparent and perceptually enhanced JPEG image encryption. In *16th International Conference on Digital Signal Processing* (pp. 1-6). IEEE.

Ye, Y., Zhengquan, X., & Wei, L. (2006). A Compressed Video Encryption Approach Based on Spatial Shuffling. In *8th International Conference on Signal Processing, Vol. 4* (pp. 16-20). IEEE.

Region of Interest Signalling for Encrypted JPEG Images

Dominik Engel
Salzburg University of Applied Sciences
Urstein Süd 1
Puch/Salzburg, Austria
dengel@en-trust.at

Andreas Uhl
University of Salzburg
Dept. of Computer Sciences
Jakob-Haringer-Str. 2
Salzburg, Austria
uhl@cosy.sbg.ac.at

Andreas Unterweger
University of Salzburg
Dept. of Computer Sciences
Jakob-Haringer-Str. 2
Salzburg, Austria
aunterweg@cosy.sbg.ac.at

ABSTRACT

We propose and evaluate different methods to signal position and size of encrypted RoIs (Regions of Interest) in JPEG images. After discussing various design choices regarding the encoding of RoI coordinates with a minimal amount of bits, we discuss both, existing and newly proposed approaches to signal the encoded coordinates inside JPEG images. By evaluating the different signalling methods on various data sets, we show that several of our proposed encoding methods outperform JBIG in this special use case. Furthermore, we show that one of our proposed signalling methods allows length-preserving lossless signalling, i.e., storing RoI coordinates in a format-compliant way inside the JPEG images without quality loss or change of file size.

Categories and Subject Descriptors

E.2 [**Data**]: Data Storage Representations—*Object representation*; E.4 [**Data**]: Coding and Information Theory—*Data compaction and compression*; I.4.2 [**Image Processing and Computer Vision**]: Compression (Coding)—*JPEG*

General Terms

Algorithms, Theory, Measurement

Keywords

JPEG, Region of Interest, Coordinates, Encoding, Signalling

1. INTRODUCTION

In the last decade, a large number of region of interest encryption approaches have been proposed, especially for image and video formats using DCT-domain-based compression, like JPEG [12]. Although the human eye is capable of detecting encrypted picture regions easily, state-of-the-art software is not. There have been attempts to detect encrypted picture regions automatically [3], i.e., without the need to signal them explicitly. However, despite their reported near-100% accuracy, it is clear that perfect detection is not possible, albeit necessary for correct decryption in most cases.

Therefore, there is an immanent need to store the encrypted RoIs' coordinates inside the JPEG file in order to have them available during decryption. As the JPEG file format has no means of signalling encrypted regions, unlike JPEG2000 [1], different methods of encoding these coordinates have to be evaluated and a detailled analysis of possible signalling methods is required.

All RoI encryption approaches for JPEG proposed so far handle RoI signalling in one of three ways. The first method involves using JPEG comment segments with an unspecified coordinate encoding [2], which is straight-forward, but does not take into account that some applications do not tolerate size changes of the JPEG file. Thus, we explore different options and present solutions for a variety of typical practical contraints.

The second and most common signalling method relies on an external signalling channel [4]. As signalling RoIs in the JPEG image itself has significant advantages as compared to using a separate channel, this paper proposes and evaluates possibilities to store coordinates of encrypted RoIs inside the JPEG images themselves.

Finally, the third signalling method is to omit signalling details altogether [13], which can make decryption impossible or dependent on human RoI identification. As this is not acceptable in most cases, an analysis of encoding and signalling methods for encrypted RoIs in JPEG images is required.

This paper is structured as follows: In Section 2, we discuss design choices for RoI coordinate encodings and select a subset thereof for further evaluation. In Section 3, we propose different methods to signal the encoded coordinates inside a JPEG file. Subsequently, in Section 4, we evaluate all encoding and signalling methods in order to find appropriate combinations for different use cases before we conclude the paper in Section 5.

IH&MMSec'13, June 17–19, 2013, Montpellier, France.

2. ROI COORDINATE ENCODING

Before the RoI coordinates can be signalled, they have to be encoded appropriately. As the number of signalled bits may be limited or even influence the picture quality, depending on the signalling method used, a compact encoding is desired. In this Section, we discuss several design choices for encodings, aiming at listing a set of practically useable encodings to be evaluated in Section 4.

As most state-of-the-art encryption approaches for JPEG

Name	Explicit	Value encoding	Differential	Bits per RoI	Overhead bits per file
Bitmap	–	Fixed length	N/A	0	n_{iMCU}
List	✓	Fixed length	–	$2 \cdot \lceil log_2(n_{iMCU}+1) \rceil$	$2 \cdot \lceil log_2(n_{iMCU}+1) \rceil$
VList	✓	Exp. Golomb	–	Variable	2
DList	✓	Signed Exp. Golomb	✓	Variable	2
ACBitmap	–	Fixed + ABAC	N/A	0	Variable
ACList	✓	Fixed + ABAC	–	Variable	Variable
ACVList	✓	Exp. G. + ABAC	–	Variable	Variable
ACDList	✓	S. Exp. G. + ABAC	✓	Variable	Variable
JBitmap	–	JBIG	–	Variable	12 (header only)

Table 1: List of coordinate encodings to be evaluated and their respective storage requirements

operate on a block [15] or iMCU (interleaved Minimum Coding Unit, multiple luminance and the corresponding chrominance blocks) level [22], coordinates are limited to iMCU granularity. Note that this limitation is also imposed – often self-imposed [3] – on format-independent encryption approaches which operate in the image domain.
Furthermore, chrominance subsampling is assumed to be 4:2:0 [14] as it is the default setting in the JPEG reference software and widely used [20]. This enforces a fixed iMCU size of six blocks, four of which are luminance blocks [12], limiting the coordinate granularity to rectangular image blocks of $16 \cdot 16$ pixels size.
Subsequently, the following variables are used: w and h denote a picture's width and height in pixels, respectively. Furthermore, the width and height in iMCUs are defined as $w_{iMCU} = \lceil \frac{w}{16} \rceil$ and $h_{iMCU} = \lceil \frac{h}{16} \rceil$, respectively. In addition, $n_{iMCU} = w_{iMCU} \cdot h_{iMCU}$ denotes the total number of iMCUs in a picture. Finally, n_{RoI} specifies the number of RoIs to be encoded. All coordinate encodings described in the subsequent subsections are summarized in Table 1 using the aforementioned variables.

2.1 Implicit vs. explicit encoding

Coordinates can be encoded either implicitly or explicitly. While implicit encoding entails deriving the actual coordinates locally, e.g., from the bit patterns in a bitmap, explicit encoding stores the actual coordinates globally so that they can be read directly. Hence, the simplest form of implicit encoding, i.e., a bitmap for all iMCUs where a zero bit means "not encrypted" and a one bit means "encrypted", requires n_{iMCU} bits to be stored (see Table 1: "Bitmap").
In contrast, explicit coordinate encoding requires storing a list of coordinates, specifying the location and size of each RoI. Both, location and size, are described by a horizontal and vertical component, referred to as X and Y coordinate, respectively, yielding four coordinates in total.
In addition, it is necessary to specify a special coordinate signalling the end of the coordinate list. For the sake of simplicity and practicality, we subsequently use a RoI with a size of zero to signal the end of the list. This is reflected in the per-file overhead of all explicit encodings listed in Table 1 accounting for the additional end-of-list entry as storing n_{RoI} RoIs requires $n_{RoI}+1$ list entries in total. Each list entry consists of 4 coordinates, 2 of which are X and Y coordinates, respectively.

2.2 Component vs. index encoding

Although separate X and Y coordinates allow locating the encrypted RoIs easily, two components (X and Y) need to be stored to specify one location. When using a fixed bit length per component, the X and Y coordinate require $\lceil log_2(w_{iMCU}+1) \rceil$ and $\lceil log_2(h_{iMCU}+1) \rceil$ bits of space, respectively.
Alternatively, an index can be assigned to each iMCU, starting with zero for the top-left-most iMCU and increasing in the left-to-right and top-to-bottom direction. This way, a location identified by two components (X and Y) can be specified by a single index which requires $\lceil log_2(n_{iMCU}+1) \rceil$ bits when using a fixed bit length per index. Note that this is always shorter than or in the worst case as long as signalling two separate components since $\lceil log_2(n_{iMCU}+1) \rceil = \lceil log_2(w_{iMCU} \cdot h_{iMCU}+1) \rceil \leq \lceil log_2(w_{iMCU}) + log_2(h_{iMCU}) + 1 \rceil \leq \lceil log_2(w_{iMCU}+1) \rceil + \lceil log_2(h_{iMCU}+1) \rceil$, which is the number of bits required for two separately stored X and Y coordinates. Thus, index encoding is to be preferred over component encoding and all explicit encodings listed in Table 1 encode iMCU indices instead of X and Y coordinates.

2.3 Fixed-length vs. variable-length encoding

As the picture width and height are known, the maximum number of bits required to encode one iMCU index can be determined easily. If this fixed bit length is used for all indices, encoding one RoI requires $2 \cdot \lceil log_2(n_{iMCU}) \rceil$ bits in total (see Section 2.2), the factor of two being required to account for both, the location and size of the RoI (see Section 2.1). This way, each encoded RoI requires the same number of bits, regardless of its own size and location (see Table 1: "List").
As RoIs usually do not span the whole picture, using a constant number of bits which allows specifying the whole picture size can be disadvantageous. Similarly, RoI locations on the top-left require a high number of bits, although their corresponding iMCU start indices are small. Hence, the use of variable-length encoding for both iMCU indices, specifying the encrypted RoI's location and size, is to be evaluated. One method for variable-length coding are Exponential-Golomb codes as used e.g., for encoding a subset of H.264 syntax element values [19]. As a RoI's position and size (represented as iMCU indices) are always positive, a zeroth order (i.e., $k = 0$) unsigned Exponential-Golomb code ("ue(v)" following the notation of the H.264 standard [11]) can be used to encode them. Table 2 shows examples of values and their respective encoded bit representation.
As can be seen, a value of zero can be signalled using one bit. Hence, an end-of-list entry (with position and size being zero) can be signalled using two bits (see Table 1). Gener-

Value	ue(v) code word	se(v) code word
...	–	...
-4	–	0001001
-3	–	00111
-2	–	00101
-1	–	011
0	1	1
1	010	010
2	011	00100
3	00100	00110
4	00101	0001000
...	...	...

Table 2: List of exemplary values and their respective zeroth order Exponential-Golomb code words. Hyphens denote invalid value ranges

ally, any positive integer value x requires $2 \cdot \lceil log_2(x+2) \rceil - 1$ bits. Thus, one iMCU index requires a maximum of $2 \cdot \lceil log_2(n_{iMCU} + 2) \rceil - 1$ bits. As the actual number of bits can be smaller, depending on the actual iMCU indices to be encoded, the storage requirements per RoI are variable when using Exponential-Golomb encoded list entries (see Table 1: "VList" for variable-length coded list), possibly reducing the number of stored bits compared to fixed-length encoding.

2.4 Differential encoding

Although variable-length coding reduces the storage requirements when encoding small indices, the converse is true for large indices, i.e., indices identifying iMCUs at the bottom-right of a picture. In order to overcome this drawback, each index can be stored relative to its predecessor, replacing the actual value to be encoded by a differential value which is very likely to be smaller. For example, a location/size pair (l_2, s_2) can be encoded as $(l_2 - l_1, s_2 - s_1)$ relative to its preceding location/size pair (l_1, s_1). As all RoIs are known, their order in the RoI list can be chosen so that the differential values to be encoded are minimal in terms of size.
However, it is not guaranteed that there is an order of entries in the RoI list so that all differences are positive, thus requiring the ability to encode negative differences as well. Signed Exponential-Golomb codes which support both, positive and negative values, are described in the H.264 standard [11]. Following the latter's notation, such zeroth order codes are referred to as "se(v)". Table 2 shows examples of values and their respective encoded bit representation.
In general, any integer value x requires $2 \cdot \lceil log_2(2 \cdot |x| + 2) \rceil - 1$ bits as signed Exponential-Golomb code word, which is more than the amount required for the respective unsigned Exponential-Golomb code word. Nonetheless, we include this encoding approach as its storage requirements depend on the RoI's coordinates' differences (see Table 1: "DList" for differentially encoded list) which depend on the values and ordering of the RoIs, unlike all other encodings.

2.5 Entropy coding

Each of the encodings described above makes use of different representations and/or properties of the list of RoI coordinates. However, none of them aims at effectively eliminating redundancy. Thus, a modified version of each encoding is included in Table 1 which essentially adds an entropy coding step after the original encoding process, indicated by a "C" (for compressed) prefix in the encoding's name.
Arithmetic coding [24] (prefixed with an additional "A") is chosen for the entropy coding step as it theoretically allows for quasi optimal, i.e., close-to-entropy, performance. As the number of different values to be encoded is equal to n_{RoI} for n_{ROI} RoI location/size pairs and smaller than or equal to $2 \cdot n_{RoI}$ for separately encoded location and size values, binary arithmetic coding (abbreviated BAC in Table 1) calculated in fixed-precision integer arithmetic as described in the JPEG standard [12] is evaluated.
As signalling the symbols' probabilities (or the corresponding subintervals) would require additional bits, adaptive coding, i.e., the dynamic adjustment of the symbol probabilities, is used to optimize coding efficiency [19]. Starting with equal probabilities for both symbols, zero and one, the subinterval ranges are adjusted according to the changing symbol frequencies during encoding. Note that end-of-stream markers can be omitted as the decoding process can stop the arithmetic decoding process as soon as the end-of-list marker (a RoI with location and size zero) is found.

2.6 Bi-level image compression

As the implicitly encoded bitmap described in Section 2.1 is in fact a bi-level image, the use of a compressor which is optimized for this type of images has to be evaluated for comparison. Due to its widespread use, we choose the JBIG compression standard [9] in combination with one of its application profiles [10] for this task (see Table 1: "JBitmap" for JBIG-compressed bitmap).
In order to compensate for its relatively large file header with a total size of 20 bytes, we shorten the former by the eight bytes which signal the image's width and height as they can also be derived otherwise, e.g., from the JPEG picture. This reduces the total per-file overhead to twelve bytes, thus allowing for a fairer comparison.

2.7 Summary

A number of choices have to be made when designing an encoding for a list of RoIs, few of which are clear without prior evaluation. As outlined in Section 2, encoding iMCU indices always requires less than or as many bits as encoding separate X and Y coordinates, Thus, all encodings to be evaluated encode iMCU indices. As most other design criteria of possible encodings depend on either the number and/or size of the RoIs and/or the picture, a selected subset of possible encodings (see Table 1) covering all of the aforementioned criteria has to be evaluated in Section 4.

3. ROI SIGNALLING

The encoded RoIs' coordinates need to be signalled in some form in order to identify the RoIs at a later point in time, e.g., during the decryption process. Thus, in this Section, we propose a number of different ways to store the encoded RoI coordinates directly inside the JPEG file. In order to account for the different needs of conceivable use cases, the proposed signalling methods are chosen to cover a number of different combinations of the following aspects:

1. **Format compliance**: The strict fulfillment of all syntactical and semantical requirements imposed by the JPEG standard [12]

2. **Losslessness**: The exact preservation of all (visible) picture data
3. **Availability**: The guarantee that the proposed method will work on every JPEG picture
4. **Length-preservation**: The guarantee that the picture's file size does not change (suitable for length-preserving encryption methods like [22])

Furthermore, the capacity, i.e., the amount of storable bits, of each signalling method is given. Note the capacity of some of the proposed methods depends on the picture and/or its metadata. As the number of RoIs is usually not known in advance for all pictures, all methods need to be evaluated in terms of usability for storing encoded RoI coordinates as proposed in Section 2, which is done in Section 4.
All proposed methods are described with regards to the aforementioned aspects and summarized in Table 3 for convenience. For reasons of practicality, we assume that all JPEG pictures are Baseline JPEG pictures [12] with three color components – Y, Cb and Cr, i.e., one luminance and two chrominance components. Note that most methods will, however, work with differently coded JPEG pictures (e.g., arithmetically coded ones) as well.

3.1 Use of COM and APP segments

The first method, the insertion of a COM (Comment) segment into the JPEG file according to Annex B of the JPEG standard [12], has already been proposed by others (e.g., [2]). One COM segment may contain up to 65533 payload bytes, plus its marker (2 bytes) and length field (2 bytes), totalling to 65537 stored bytes. As the number of COM segments is theoretically unlimited, so is the total capacity of this signalling method. Signalling n bits requires $n_{COM} = \left\lceil \frac{\lceil \frac{n}{8} \rceil}{65533} \right\rceil$ COM segments with a total of $(n_{COM} - 1 + \epsilon) \cdot 65537 + 4 + \lceil \lceil \frac{n}{8} \rceil \; mod\, 65533 \rceil$ bytes, where mod denotes the integer modulus operator and ϵ is a correction factor of 1, if there is no remainder (of the modulus operation), and 0 otherwise. The rounding to full bytes is due to the fact that a COM segment's length field is expressed in bytes, not bits.
As an alternative to the COM segment, an Application Data (APP) segment can be used, which is equivalent in terms of structure. As there are 16 different APP markers, it is theoretically possible to encode four more bits into an APP segment than into a comment segment of equal total size. As the capacity is otherwise the same, signalling n bits in APP markers requires $n_{APP} = \left\lceil \frac{\lceil \frac{n}{4} \rceil}{2 \cdot 65533.5} \right\rceil$ APP segments with a total of $(n_{APP} - 1 + \epsilon) \cdot 65537 + 4 + \left\lceil \frac{\lceil \lceil \frac{n}{4} \rceil \; mod\, (2 \cdot 65533.5) \rceil}{2} \right\rceil$ bytes. Due to the additional 4 bits per segment, APP segment signalling is to be preferred over COM segment signalling in terms of capacity. However, there may already be APP segments in the JPEG file, in which case the gain in capacity may be reduced. Moreover, there may be a border case in which all different APP segment types are already present in the file, making it impossible to store any data in this way.
One commonly used APP segment type is APP_1, typically storing data in the Exchangeable Image File Format (EXIF) [6]. If such data is present, but not crucial for further processing, it can be replaced by encoded RoI coordinates. However, this method of stripping EXIF data depends on the presence of the latter and is usally very limited in terms of capacity. A more detailed description of this method and its capacity is provided in [5], which is why it is not evaluated separately herein.

3.2 Use of dummy tables

Although JPEG Baseline pictures with three components use the maximum number of Huffman tables per file, it is possible to add an arbitrary amount of dummy Tables at the end of the file by inserting Huffman table (DHT) segments containing encoded bits. One such Table can be identified easily during the decoding process. As the "defined" code words are not actually used, they do not necessarily need to be valid. Hence, it is possible to define up to 16 sets of 255 theroretically contradictory maximum length Huffman code words defining one 8-bit value each. In total, this allows storing $16 \cdot 255 \cdot 8 = 32640$ bits at the expense of $4099 \cdot 8 = 32792$ stored bits (see [12, p. 45]). Note that an additional four bytes are required for the marker (two bytes) and the length field (two bytes) per segment. As an arbitrary amount of dummy Tables with the same destination identifier can be inserted, the capacity of this approach is theoretically unlimited.
Similar to dummy Huffman tables, dummy quantization tables can be defined by inserting Quantization Table (DQT) segments. One such segment can store 8 bits for each of the 64 quantization table positions. This allows for storing $64 \cdot 8 = 512$ bits at the expense of $65 \cdot 8 = 520$ stored bits (see [12, p. 44]). Again, the four bytes of overhead for the marker (two bytes) and additional length field (two bytes) have to be accounted for once per segment. The capacity is, again, unlimited due to the theoretically unlimited amount of dummy Tables when using the same destination identifier.

3.3 Information hiding

As an alternative to bit-stream-based changes to signal the RoIs, classic information hiding approaches, especially steganographic ones, can be used. An overview of state-of-the art methods, of which we consider the widely used coefficient-based approaches, i.e., those which alter bits in the DCT domain, is given in [5]. As encrypted RoIs can typically be identified by the human eye, the main aim of using information hiding for signalling is not hiding the bits, but storing them within the image itself. Thus, hiding schemes like F5 [23] which are known to be vulnerable to attacks [7] are considered as well.
The approaches' capacities is not evaluated herein as it has been evaluated in the literature, e.g., [8] for coefficient-based information hiding. For JPEG images with one channel, i.e., grey-scale images, a capacity of 0.02 bits per non-zero AC coefficient has been reported. As we assume having three channels per image, it is safe to use the aforementioned capacity as a lower bound, requiring only to determine the average number of non-zero AC coefficients of the test data. Note that information hiding is not necessarily lossy as reversible approaches have been proposed (e.g., [16, 18]).

3.4 Length-preserving signalling

A method without overhead is the use of bits occupied by unused code words in the Huffman tables, i.e., code words

Method	Compliant	Lossless	Available	Length-preserving	Capacity (bits)
COM segment	✓	✓	✓	–	∞
APP segment	✓	✓	–	–	∞
EXIF data stripping	✓	✓*	–	✓	Variable
Dummy DHT	✓	✓	✓	–	∞
Dummy DQT	✓	✓	✓	–	∞
Steganographic (coefficients)	✓	–**	✓	Depends	Variable
Reuse of unused DHT entries	✓	✓	–	✓	Variable
DQT bit stealing	✓	Depends	–	✓	Variable
Data before first marker	–	✓	✓	–	∞
Data after last byte	–	✓	✓	–	∞

* Original EXIF data will be lost, ** Reversible techniques proposed (e.g., [16, 18])

Table 3: List of proposed methods for RoI signalling in JPEG pictures broken down by the aspects listed in Section 3

which are not used throughout the file. Although it is simple to find unused code words, even when e.g., decrypting, the number of unused code words may be very low or even zero, if the Huffman table only contains used code words or if there is no Huffman table to begin with. Even if there are unused code words, each of them only allows for storing 8 bits. Furthermore, as the number of unused code words varies from file to file, this method's capacity highly depends on the encoder which created the file and therefore has to be evaluated.
Another method of storing encoded RoI coordinates is by stealing bits from the quantization table(s), i.e., by modifying the bits of some quantization table entries, if there is a quantization table in the first place. There are two possibilities of doing so: One way is to change one bit at a time, starting at the high frequency entries of the chrominance quantization tables. After each modification, the JPEG file is decoded and compared to the version with unchanged quantization tables. Although this is computationally very expensive, it can also be done during the decoding process to find out which bits of the quantization tables were used. However, the capacity is highly dependent on the picture and possibly zero. Alternatively, if distortions are acceptable up to a certain degree, a fixed number of bits can be used, omitting the trial-and-error process described before. Although this allows for a higher capacity, it does so at the expense of picture quality, which has to be assessed.

3.5 Non-format-compliant signalling

A way to losslessly signal encoded RoI coordinates is to insert them at either the very beginning of the file, i.e., before the first marker, or at its end, i.e., after the last data byte. Adding data in this way is, however, not format compliant as the standard only allows for 0xFF fill bytes preceding each marker. In addition, in both cases, special care has to be taken in order to escape 0xFF payload bytes which would otherwise be interpreted as markers. Depending on how escaping is done, this may lead to additional overhead. As this method of signalling encoded RoI coordinates is not format compliant, most image viewers and editors will not be able to open files edited by it anymore.

4. EVALUATION

In order to evaluate the RoI signalling methods presented in Section 3 in combination with the coordinate encoding methods proposed in Section 2, we first evaluate each aspect separately and subsequently combine them. As a practical JPEG RoI encryption application we choose the encryption of people in pictures of surveillance cameras.
In total, eleven test sets are used – three indoors and eight outdoors sets. The three indoors data sets[1] with a total of 3271 pictures with a spatial resolution of $360 \cdot 288$ pixels each are courtesy of the EPSRC funded MOTINAS project (EP/D033772/1). The eight outdoors data sets[2] with a total of 67616 pictures with a spatial resolution of $640 \cdot 480$ pixels each are courtesy of EPSRC project GR/S98146. All data sets include ground truth for people's coordinates within each picture, which is subsequently used as set of RoIs to be encoded and signalled. RoIs which exceed one or more of the pictures' borders are omitted.

4.1 Encoded RoI bit length assessment

In order to perform coordinate encoding of the data sets' RoIs, we implemented the different encoding methods presented in Section 2 in Python, except for arithmetic encoding and JBIG compression, for which we used the Python implementation of David MacKay[3] and JBIG-KIT[4], respectively. As we restrict the coordinates' accuracy to iMCUs of $16 \cdot 16$ pixels size (see Section Section 2), we rounded the data sets' RoI coordinates so that all blocks containing an RoI were considered to be encrypted as a whole. Before actually encoding the rounded coordinates, they were translated into iMCU indices as explained in Section 2.2.
Tables 4 and 5 show the average number of bits per picture required to encode the RoIs of the indoors and the outdoors data sets, respectively. As the RoI count of the pictures has a significant impact on the number of bits required, the results are grouped by RoI count, considering only pictures from the data set with the stated number of RoIs. Note that pictures without, i.e., zero, RoIs are omitted as they are discussed separately in the second part of this Section.
It is clearly visible from the results of both data sets that entropy coding (in the right half of each Table) always im-

[1] `ftp://motinas.elec.qmul.ac.uk/pub/av_people/`
[2] `http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/INTERACTIONS/`
[3] `http://shedskin.googlecode.com/svn/trunk/examples/ac_encode.py`
[4] `http://www.cl.cam.ac.uk/~mgk25/jbigkit/`

RoIs	Pictures	Bitmap	List	VList	DList	ACBitmap	ACList	ACVList	ACDList	JBitmap
1	1907	414.00	36.00	*30.94*	34.94	196.00	31.27	31.04	33.70	206.96
2	959	414.00	54.00	61.53	56.53	196.00	*52.07*	59.01	54.20	229.30
Non-0	2866	414.00	42.01	41.15	42.15	150.99	38.22	*40.38*	40.55	214.42

Table 4: Average number of bits required to encode the RoIs of each picture of the indoors data set with a given number of RoIs. The best, i.e., minimal, number of bits for each distinct picture subset is italicized

RoIs	Pictures	Bitmap	List	VList	DList	ACBitmap	ACList	ACVList	ACDList	JBitmap
1	1444	1200.00	44.00	*33.58*	37.58	138.32	35.64	34.04	36.05	212.34
2	3449	1200.00	66.00	64.72	55.80	241.83	58.78	60.92	*52.26*	231.46
3	2820	1200.00	88.00	92.81	81.97	255.98	80.66	85.31	*74.41*	238.52
4	1877	1200.00	110.00	135.89	106.16	333.66	105.88	121.25	*96.51*	245.66
5	10822	1200.00	132.00	166.46	135.98	393.94	128.51	150.00	*122.08*	260.68
6	814	1200.00	154.00	204.50	163.71	433.94	149.21	180.02	*144.71*	267.50
7	3	1200.00	176.00	232.67	168.67	430.67	177.00	208.00	*153.00*	266.67
8	2	1200.00	198.00	270.00	176.00	430.00	197.50	247.50	*158.50*	256.00
Non-0	21234	1200.00	108.38	129.92	107.54	329.75	103.34	117.70	*97.18*	248.64

Table 5: Average number of bits required to encode the RoIs of each picture of the outdoors data set with a given number of RoIs. The best, i.e., minimal, number of bits for each distinct picture subset is italicized

proves encoding efficiency, i.e., it reduces the number of bits, except in the case of only one RoI when using a list of variable-length coded indices ("VList"). Thus, implementing an entropy coding step following the actual coordinate encoding step should always be considered when encoding more than one RoI. In the case of a single RoI, variable-length coded indices ("VList") give the best results on average over all eleven test sets.

For a higher number of RoIs, the outdoors data sets (Table 5) allow for a more thorough analysis due to the data sets' widespread range of RoI counts. They clearly show that a differentially coded list of values which is entropy coded ("ACDList") always gives the best results. The higher the number of RoIs is, the higher the bit savings of this method are compared to all of the others besides "JBitmap". Note that there are only very few pictures with seven and eight RoIs, respectively, making the results only reliable for up to six RoIs. Nonetheless, averaging the number of bits spent over all pictures with RoIs (last line of Table 5) reveals that the "ACDList" encoding is optimal for data sets which contain a high number of pictures with more than one RoI. The maximum number of bits required for one list of RoIs over all data sets (not listed in the Table) is 219 bits.

Additionally considering the 959 pictures of the indoors data set (Table 4) containing two RoIs shows that an entropy coded list of indices ("ACList") yields a good performance as well, albeit only smaller by about two bits in this special case as compared to the "ACDList" encoding. Interestingly, the "ACVList" encoding shows the best overall performance over the complete indoors data sets, being 0.17 bits shorter than the "ACDList" encoding on average. This is due to the fact that the number of pictures in the indoors data set with one RoI is higher than the number of pictures with two RoIs and that the "ACVList" encoding requires the smallest number of bits for encoding one RoI as compared to all other entropy-coding-based encodings in the indoors data sets.

Surprisingly, JBIG compression performs significantly worse than most of our proposed approaches, which is mainly due to the large overhead caused by the JBIG file header. However, it is clearly visible from the outdoors data set in Table 5 that the JBIG based encoding requires fewer bits per additional RoI compared to all other approaches. While our "ACDList" approach requires on average 108.66 bits more for encoding six RoIs than it does for one RoI, "JBitmap" only requires 55.16 bits more. Thus, it is expected that JBIG compression outperforms our approaches for large numbers of RoIs.

As encoding zero RoIs, i.e., the fact that no RoIs are present, is independent of the data set used, both, Table 4 and 5, do not include pictures without RoIs. In order to assess the encoding methods' RoI encoding performance of pictures of this type in general, we used artificial images with different spatial dimensions, all of which had an aspect ratio, i.e., a width-to-height ratio, of 4:3 as is common in surveillance applications. Moreover, all image sizes were rounded to the next integer multiple of 16.

Figure 1 shows the number of bits required to encode zero RoIs for all proposed encodings with picture sizes ranging from $16 \cdot 16$ to $1920 \cdot 1440$ pixels in steps of 16 pixels in width. Although the aspect ratio is fixed (despite the small errors due to rounding), the X axis shows the square root of the image area, making the results applicable to arbitrary aspect ratios. The Y axis shows the required number of bits using a logarithmic scale.

The bit requirements for "VList" and "DList" as well as for their entropy coded variants, "ACVList" and "ACDList", are always constant, regardless of the picture's spatial dimensions, thus forming a combined line at two bits. This property makes the four encodings ideal for quasi all picture sizes, except for a size of $16 \cdot 16$, which we consider of having no practical use. Thus, each of the four encodings is recommended for encoding zero RoIs at all spatial picture dimensions used in practice.

Conversely, the "Bitmap" encoding's requirements in terms of bits increase quadratically with picture width (linearly with increasing picture area), making it inconvenient for

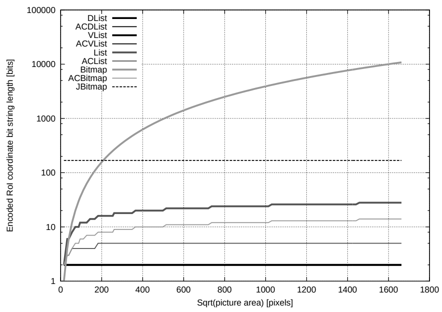


Figure 1: Number of bits required to encode zero RoIs for different spatial picture dimensions

practical use for pictures without RoIs. Note that entropy coding (“ACBitmap”) reduces the required number of bits significantly, albeit still dependent on the picture’s dimensions. The same is true for the “List” and “ACList” encodings, which are similarly inconvenient for encoding zero RoIs in practice.

Although the “JBitmap” encoding has a constant overhead, it is significantly larger (168 bits) than the overhead required by our Exponential-Golomb-based approaches described above (2 bits). Thus, it is not recommended to be used to encode zero RoIs.

As both, the indoors and the outdoors data sets, only cover a limited range of picture dimensions and RoI counts, we additionally assessed the bit length requirements of the proposed encodings using artificial test data sets. In order to artificially create RoIs that resemble real-world characteristics we use the following approach to create n_{RoI} RoIs:

- A random quadtree decomposition up to a maximum level l is created for a test image of dimensions $w{\cdot}h$, following the approach for creating uniformly distributed quadtree decompositions described in [17].
- n_{RoI} leaves are selected randomly from the set of all leaves (in case the number of leaves is less than n_{RoI}, a new quadtree is generated).
- For each selected leaf in the quadtree, a RoI is created with dimensions $q_w \cdot q_h$ where $q_w = \lceil f_w \cdot w \rceil$ with f_w chosen randomly such that $m \leq f_w < 1$. m denotes the minimum relative width and can be specified in $0 < m \leq 1$. q_h is chosen in an analogous manner.
- The position of the RoI is chosen randomly, ensuring that the RoI fits into the area covered by the leaf: Let (l_x, l_y) be the coordinates of the upper left-hand corner of the selected leaf with dimensions $l_w \cdot l_h$. The horizontal position of the RoI is determined as $\lfloor l_x + f_x \cdot (l_w - q_w) \rfloor$, with f_x chosen randomly and $0 \leq f_x < 1$. The vertical position of the RoI is determined in an analogous manner.

The parameters l and m can be used to tune the average size of the generated RoIs. In our test setup, we use $l = 3$

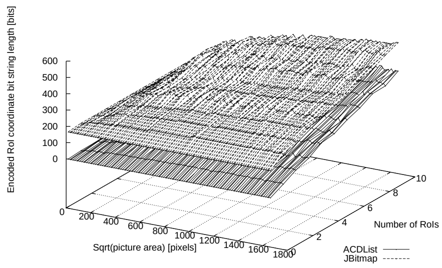


Figure 2: Number of bits required to encode a number of randomly generated, artificial RoIs for different spatial picture dimensions

and $m = 0.2$ to simulate real-world RoIs.

Figure 2 depicts the number of bits required to encode $0 \leq n_{RoI} \leq 10$ randomly generated RoIs for various picture sizes (see above for details). Note that only “ACDList” and “JBitmap” are depicted for comparison for the sake of better graphical representation. All other encodings (not depicted) perform worse except for very small picture sizes which we do not consider being practical, as discussed above, with one exception described below. It is clearly visible that “JBitmap” does not outperform “ACDList” at any picture size or RoI count depicted. This is due to the former’s large overhead, confirming that JBIG may only be suitable for a very large number of RoIs. As “ACVList” performed better than “ACDList” for a small number of RoIs, the two encodings are compared for $0 \leq n_{RoI} \leq 10$ randomly generated RoIs for various picture sizes. Figure 3 depicts the number of bits which are required by the “ACVList” encoding in addition to “ACDList”’s for any given configuration. Note that the X axis starts at 16 as a picture size of 0 is not practically useful. As in the real-world data sets, “ACDList” outperforms “ACVList” for a large number of RoIs. However, in some configurations with one or two RoIs, “ACDList” outperforms “ACVList” by a few bits. Nonetheless, the “ACDList” encoding is clearly the encoding of choice, when the number of RoIs is not known in advance and potentially large.

4.2 JPEG picture capacity assessment

As some of the signalling approaches described in Section 3 have an unknown embedding capacity, we evaluate the latter for our test data sets. The approaches considered herein are the reuse of unused DHT entries, steganographic approaches and DQT bit stealing as described in Section 3. To evaluate the embedding capacity when reusing unused DHT entries, we count the latter in the outdoors data sets’ JPEG files, whose JPEG quality varies between approximately 95 and 100%. Those pictures which have a quality of approximately 100% do not have unused DHT entries at all. Conversely, the pictures which have a quality of approximately 95% allow using 214 entries on average with one byte of capacity each, i.e., 1712 bits in total. The minimum and maximum number of unused entries is 191 (corresponding to 1518 bits) and 343 (2744 bits), respectively, which, in our opinion, is surprisingly high.

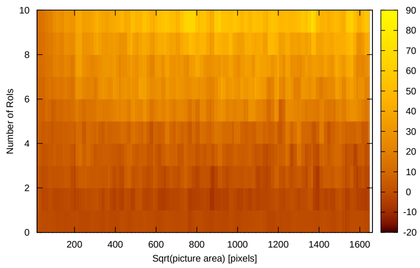


Figure 3: Additional number of bits required for the "ACVList" encoding to encode a number of randomly generated, artificial RoIs for different spatial picture dimensions compared to the "ACDList" encoding

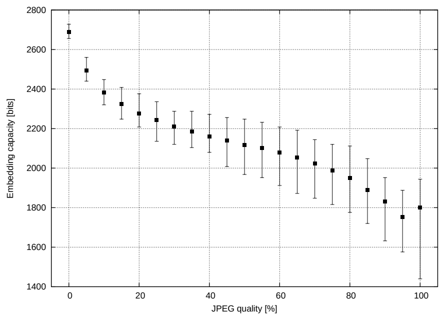


Figure 4: Average embedding capacity of the unused DHT entry reuse approach for different JPEG quality values for the pictures of the LIVE data base [21]. The bars indicate the minimum and maximum capacity for each quality value, respectively

As the outdoors data sets' approximate JPEG quality is within a very limited range, an evaluation spanning a wider range of JPEG quality values is necessary. As the indoors data sets are already compressed using a different compressor, applying JPEG compression would also recompress the existing artifacts, yielding results which are not representative. Thus, we use images from the LIVE data base [21] and compress them with the standard JPEG encoder with quality values ranging from 0 to 100% in 5% steps.
The results are depicted in Figure 4 and show that low JPEG quality values allow for a higher embedding capacity than high quality values. This is due to the standard JPEG encoder using a fixed DHT, making unused entries more likely for lower quality due to the stronger quantization and therefore longer runs with lower absolute coefficient values. Note that the embedding capacity for a quality value of 95% is approximately 1800 bits on average with a minimum value of about 1600 bits, which matches the outdoors data sets' capacity within a small bound.
To evaluate the embedding capacity of steganographic approaches, we use the approximation described in Section 3 estimating the embedding capacity as 0.02 bits per non-zero AC coefficient. Similar to the embedding method explained above, we count the non-zero AC coefficients in all JPEG images of the outdoors data sets. After omitting one image which is all black, we find an average number of about 195117 non-zero AC coefficients per file with a minimum and maximum of 49440 and 246679, respectively. This corresponds to an embedding capacity of about 3902 bits on average with a minimum of 988 bits.
Again, we also count the number of non-zero AC coefficients of the images from the LIVE data base with different JPEG quality values to cover a wider range of the latter. As can be seen in Figure 5, the number of non-zero AC coefficients and therefore the embedding capacity increases quasi linearly for increasing low JPEG quality values and exponentially with increasing high JPEG quality value (note the logarithmic Y scale). For a JPEG quality value of 95%, the embedding capacity is approximately 3000 bits on average with a minimum of about 2000 bits, which differs from the outdoors data sets' capacity, but is within the same order of magnitude.

Finally, we evaluate the DQT bit stealing approach. In order to simulate a worst case bit-stealing scenario, we flip n bits of each 8-bit DQT entry from indices i_1 to i_2 in zigzag order, i.e., in bit stream order of both, luminance and chrominance DQT.
To find suitable values for n, i_1 and i_2, we take a picture data set, decode each picture and then compare it with a decoded version with flipped QT Table entries for all possible values of n, i_1 and i_2. In order to assess the difference between the original and the modified picture, we measure the PSNR value between the two.
As the number of possible combinations of n, i_1 and i_2 is large, we evaluated them exhaustively on a smaller test set – the LIVE data base [21]. Using the JPEG reference encoder, we created Baseline JPEG images with default settings and 50, 75, 95 and 100% quality from the original, i.e., uncompressed, images.
Figure 6 shows the embedding capacity (in terms of total stolen bits, Y axis) over all images of a given JPEG quality so that the distortion of no image exceeds the depicted PSNR value (X axis). Note that the JPEG quality influences the embedding capacity significantly, as does the desired maximum distortion.
Surprisingly, the total capacity is very high, considering that changes of the DQT potentially influence all blocks of a picture. Depending on the desired target distortion, it is possible to embed several hundred bits.
Note that 100% quality does not allow embedding one bit so that no picture exceeds a distortion of 50dB. The same is true for all JPEG quality values when no distortion (∞ dB) is desired. Thus, this approach cannot be used for lossless embedding.
Attempting to verify these results for the outdoors data sets, we split the data sets into pictures with approximately 95% and 100% JPEG quality, respectively, using the obtained settings for a target quality of 35dB. Surprisingly, every picture in both sets exceeds a quality of 50dB compared to its unmodified version, indicating that the embedding capacity for a given target quality is highly dependent on the pic-

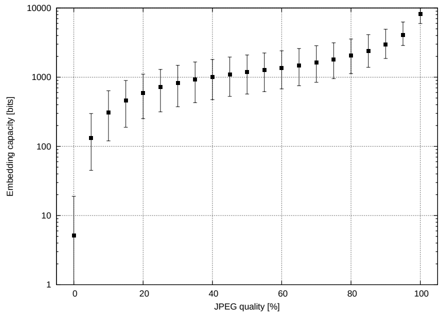


Figure 5: Average embedding capacity of steganographic approaches for different JPEG quality values for the pictures of the LIVE data base [21]. The bars indicate the minimum and maximum capacity for each quality value, respectively

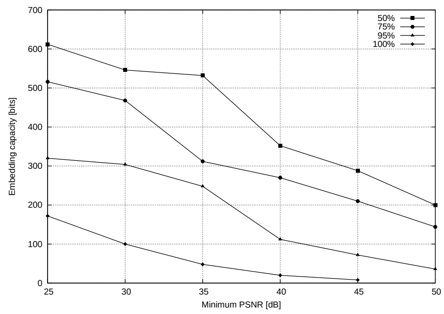


Figure 6: Embedding capacity of the DQT bit stealing approach for different JPEG quality values and maximum target distortions for the pictures of the LIVE data base [21]

tures themselves. Due to the lack of freely available and practically relevant data sets, a thorough examination of this method with more pictures of different characteristics remains future work.

4.3 Combined encoding and signalling

Combining the results of the previous Sections, we subsequently evaluate the feasibility of the combined use of the proposed encoding and signalling methods in order to simulate the actual storage of RoI coordinates in the corresponding JPEG files. Due to the lack of freely available JPEG-encoded data sets with RoI ground truth, only the outdoors data sets can be assessed in this Section. Although this allows no general conclusions regarding the usefulness of the proposed approaches, it is possible to determine possible combinations of signalling and encoding methods suitable for the outdoors data sets, which cover a significant portion of practically relevant pictures and RoI counts for surveillance and encryption applications.

As the average number of bits for encoding RoI coordinates in the outdoors data sets is smallest when using our proposed "ACDList" approach (see Section 4.1), we consider the latter to be appropriate for encoding all RoIs. This choice is further supported by the fact that our "ACDList" approach is one of the few approaches which allows signalling the absence of RoIs by just two bits. Note that for data sets where zero or one RoI(s) are dominant, our "ACVList" approach allows using fewer bits.

Subsequently, we determine which of the signalling approaches described in Section 3 are able to provide enough embedding capacity to store the "ACDList"-encoded RoI coordinates for the outdoors data sets. Trivially, all approaches which offer infinite capacity can be used to signal the RoIs which require a maximum number of 219 bits with our proposed "ACDList" encoding. As format-compliant approaches are in general preferred over non-format-compliant ones, we suggest using COM segments as their overhead is smallest as compared to all other methods which offer infinite capacity (see Section 3).

The reuse of unused DHT entries allows for a largely sufficient minimum capacity of 1518 bits in all pictures with approximately 95% JPEG quality, allowing for lossless and length-preserving RoI signalling. To our knowledge, this is the first time that such an approach has been proposed. However, the pictures with approximately 100% JPEG quality do not allow storing a single bit using this method, making it unusable in this scenario. Consequently, we suggest using this method instead of others whenever possible as its capacity is sufficient to store large numbers of RoIs without quality loss and change of file size.

If quality loss is acceptable, classical steganographic methods allow for a high capacity when using a generalized estimation of the embedding capacity. It is notable that the minimum estimated capacity of these approaches is lower (988 bits) than the minimum capacity provided by the reuse of unused DHT entries, if the latter is available (1518 bits). However, steganographic approaches can be used on practically any picture, making it the method of choice when quality loss is acceptable. Note that the actual capacity highly depends on the method used as well as on the desired distortion, which is outside of the scope of this paper. We refer to the available literature [5] for further details.

Finally, our proposed approach which steals bits from the DQT also allows signalling RoIs, although its capacity is limited and highly dependent on the desired quality in terms of PSNR as well as on the pictures' characteristics. As described in Section 4.2, further evaluations are required in order to determine the usefulness of this method. In general, it can be noted that its capacity is surprisingly high in all tested cases, but limited for large numbers of RoIs as well as for pictures with large spatial dimensions. As changing the DQTs typically influences all blocks of a picture, as opposed to most steganographic approaches, which operate on single blocks, steganographic approaches are recommended at this point, with our approach being an option to be considered as soon as it has been evaluated more thoroughly.

5. CONCLUSION

We proposed and evaluated several methods to encode and signal RoIs in JPEG images. Using a number of data sets, we determined that our proposed arithmetically coded differential lists of iMCU indices are superior to all other evaluated RoI coordinate encoding methods for a large range of RoI counts, outperforming JBIG in this special use case. Furthermore, we showed that using JPEG comment segments to store the encoded RoI coordinates causes the lowest overhead, if the file size is allowed to change. For scenarios which require length-preservation, we proposed a new method which reuses unused Huffman table entries. Although it is not always available, it allows for lossless and length-preserving signalling, if it is available. Finally, we showed that using quantization table bits allows for RoI signalling as well, although further tests are required in order to determine the general restrictions of this method.

6. ACKNOWLEDGMENTS

The authors would like to thank Matthew Sorell for his valuable ideas on which several of the RoI signalling methods are based. This work is supported by FFG Bridge project 832082.

7. REFERENCES

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

[2] T. E. Boult. PICO: Privacy through invertible cryptographic obscuration. In *IEEE/NFS Workshop on Computer Vision for Interactive and Intelligent Environments*, pages 27–38, Lexington, KY, USA, Nov. 2005.

[3] P. Carrillo, H. Kalva, and S. Magliveras. Compression Independent Reversible Encryption for Privacy in Video Surveillance. *EURASIP Journal on Information Security*, 2009:1–13, Jan. 2009.

[4] T. Chattopadhyay and A. Pal. Watermarking H.264 video, Nov. 2007. Available online: `http://www.dspdesignline.com/202805492`.

[5] A. Cheddad, J. Condell, K. Curran, and P. M. Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal Processing*, 90(3):727–752, 2010.

[6] CIPA. DC-008-2010. Exchangeable image file format for digital still cameras: Exif Version 2.3, Apr. 2010. Also published as JIETA: CP-3451B.

[7] J. Fridrich, M. Goljan, and D. Hogea. Steganalysis of JPEG Images: Breaking the F5 Algorithm. In F. A. Petitcolas, editor, *Information Hiding*, volume 2578 of *Lecture Notes in Computer Science*, pages 310–323. Springer Berlin Heidelberg, 2003.

[8] J. Fridrich, T. Pevný, and J. Kodovský. Statistically Undetectable JPEG Steganography: Dead Ends, Challenges, and Opportunities. In *Proceedings of the 9th Workshop on Multimedia & Security*, MM&Sec '07, pages 3–14, New York, NY, USA, 2007. ACM.

[9] ITU. T.82: Information technology – Coded representation of picture and audio information – Progressive bi-level image compression, Mar. 1993.

[10] ITU. T.85: Application profile for Recommendation T.82 – Progressive bi-level image compression (JBIG coding scheme) for facsimile apparatus, Aug. 1995.

[11] ITU-T H.264. Advanced video coding for generic audiovisual services, Nov. 2007.

[12] ITU-T T.81. Digital compression and coding of continuous-tone still images — requirements and guidelines, Sept. 1992. Also published as ISO/IEC IS 10918-1.

[13] C. Kailasanathan. Compression performance of JPEG encryption scheme. In *Proceedings of the 14th International IEEE Conference on Digital Signal Processing, DSP '02*, July 2002.

[14] D. A. Kerr. Chrominance Subsampling in Digital Images. `http://dougkerr.net/pumpkin/articles/Subsampling.pdf`, Jan. 2012.

[15] M. I. Khan, V. Jeoti, and A. S. Malik. On perceptual encryption: Variants of DCT block scrambling scheme for JPEG compressed images. In T.-H. Kim, S. K. Pal, W. I. Grosky, N. Pissinou, T. K. Shih, and D. Slezak, editors, *FGIT-SIP/MulGraB*, volume 123 of *Communications in Computer and Information Science*, pages 212–223. Springer, 2010.

[16] W.-C. Kuo, S.-H. Kuo, and L.-C. Wuu. High Embedding Reversible Data Hiding Scheme for JPEG. In *Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 74–77, oct. 2010.

[17] R. Kutil and D. Engel. Methods for the anisotropic wavelet packet transform. *Applied and Computational Harmonic Analysis*, 25(3):295–314, 2008.

[18] C.-C. Lin and P.-F. Shiu. DCT-based reversible data hiding scheme. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, ICUIMC '09, pages 327–335, New York, NY, USA, 2009. ACM.

[19] D. Marpe, H. Schwarz, and T. Wiegand. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, July 2003.

[20] W. Pennebaker and J. Mitchell. *JPEG – Still image compression standard*. Van Nostrand Reinhold, New York, 1993.

[21] H. R. Sheikh, M. F. Sabir, and A. C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, Nov. 2006.

[22] A. Unterweger and A. Uhl. Length-preserving Bit-stream-based JPEG Encryption. In *MM&Sec'12: Proceedings of the 14th ACM Multimedia and Security Workshop*, pages 85–89. ACM, Sept. 2012.

[23] A. Westfeld. High capacity despite better steganalysis, F5 - a steganographic algorithm. In *Proceedings of the 4th Information Hiding Workshop '01*, Portland, OR, USA, Apr. 2001.

[24] I. Witten, R. Neal, and J. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Multimedia Systems manuscript No.
(will be inserted by the editor)

Building a Post-Compression Region-of-Interest Encryption Framework for Existing Video Surveillance Systems

Challenges, obstacles and practical concerns

Andreas Unterweger · Kevin Van Ryckegem · Dominik Engel · Andreas Uhl

Received: November 13, 2014 / Accepted: (will be entered by the editor)

Abstract We propose an encryption framework design and implementation which add region of interest encryption functionality to existing video surveillance systems with minimal integration and deployment effort. Apart from region of interest detection, all operations take place at bit stream level and require no re-compression whatsoever. This allows for very fast encryption and decryption speed at negligible space overhead. Furthermore, we provide both, objective and subjective security evaluations of our proposed encryption framework. Furthermore, we address design- and implementation-related challenges and practical concerns. These include modularity, parallelization and, most notably, the performance of state-of-the-art face detectors. We find that their performance, despite their frequent use in surveillance systems, is not insufficient for practical purposes, both, in terms of speed and detection accuracy.

Keywords Face Detection · Encryption · JPEG · Region of interest · Parallelization · Subjective evaluation

Mathematics Subject Classification (2000) 68W10 · 68W27 · 68W40 · 94A08 · 94A60 · 94A62

Andreas Unterweger and Andreas Uhl
University of Salzburg
Department of Computer Sciences
Jakob-Haringer-Straße 2
5020 Salzburg, Austria
Tel.: +43-662-8044-6334
E-mail: aunterweg@cosy.sbg.ac.at

Kevin Van Ryckegem and Dominik Engel
Josef Ressel Center for User-Centric Smart Grid Privacy, Security, and Control
Salzburg University of Applied Sciences
Urstein Süd 1
5412 Puch/Hallein, Austria

1 Introduction

The purpose of most video surveillance systems is to capture people and their actions as well as to store the captured footage with the ability to view it, either on-line or off-line. This is done to achieve two main goals: First, abnormal behavior (e.g., a fight between two people) can be detected on-line by a camera operator. Second, after an incident (e.g., a robbery), the authorities can use the captured footage off-line to identify some or all of the people involved.
In either case, the privacy of those people who are not involved is invaded. This does not only include people who are bystanders in the scenarios described above, but to a greater extent uninvolved people who are captured at all other points in time, i.e., when no incidents occur.
One solution to protect the privacy of these people is to encrypt the video footage. However, most existing approaches have either one or both of the following two disadvantages:

- **Full-picture encryption**: Approaches which encrypt the whole picture (e.g., [18, 21, 27] for the Joint Photographic Experts Group (JPEG) format and [7, 2, 20] for H.264, the two most commonly used compression standards in video surveillance) require full decryption for on-line viewing, i.e., the camera operator sees everyone involved, reviving the privacy issue for bystanders.
- **Surveillance system modification**: Most encryption approaches in the literature operate either before (e.g., [4, 3, 32]) or during (e.g., [39, 8, 19]) video compression, i.e., they are pre- or in-compression encryption algorithms which require one of the following two modifications to the surveillance infrastructure when the latter has to be extended for privacy protection: a) The video encoder within the cameras is modified so that it performs the encryption. However, modifying camera hardware or firmware is often impossible due to manufacturer-imposed limitations or very expensive; b)

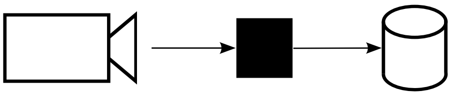


Fig. 2 Black-box encryption: The compressed video stream captured by a camera (left) is encrypted by the black box (middle) before being stored in a database (right)

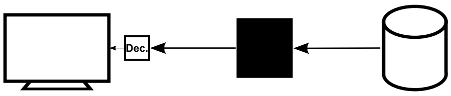


Fig. 3 Black-box decryption: The compressed video stream retrieved from a database (right) is decrypted by the black box (middle) before being decoded and displayed on a screen (left)

The compressed video streams from the cameras are transcoded by an additional hardware or software component. The encryption takes place within this component since it allows for full control over the encoder and its input data. However, transcoding is computationally very expensive and often increases bit rate and/or decreases quality, which is unacceptable.

Both of these disadvantages can be avoided by using post-compression Region of Interest (RoI) encryption. In RoI encryption, typically, the faces of all captured people are encrypted, while the rest of the picture stays intact (which requires format-compliant encryption). While this protects the privacy of those people, (encrypted) on-line viewing is still possible, since all actions can be followed and analyzed by the camera operator while not revealing people's faces. For off-line viewing, only authorized decryption-key owners (e.g., the authorities) have access to the unencrypted footage for the purpose of legal investigations.

Fig. 1 depicts a schematic of a typical surveillance system which is capable of post-compression RoI encryption. For simplicity, only the encryption side, i.e., without any decryption options, is shown. The main components of such a system are face (RoI) detection as well as encryption with integrated signalling so that the encrypted regions can be located during decryption (not depicted).

A regular surveillance system without encryption capabilities lacks these two components (face detection and encryption plus signalling). The framework described in this paper provides them as well as the corresponding decryption counterparts (not depicted). It operates on compressed video data from a non-encrypting surveillance system. As described above, the compression step is typically performed within the surveillance camera.

Post-compression RoI encryption allows extending existing surveillance systems without having to interfere with the camera hardware or firmware. At the same time, no transcoding whatsoever is required. Since all encryption operations are performed after video compression, the encryption can be added to the surveillance system in form of a black box, which is put in place between the unencrypted camera output and the storage system, as illustrated in Fig. 2.

An (authorized) operator is still able to look at the partially encrypted content with the surveillance system's unmodified on-line viewing software, as long as the encryption is format-compliant. This additionally saves costs by not requiring any changes to the on-line viewing software's video decoder.

For authorized off-line viewing, a second black box has to be put in place, which decrypts the faces before they are displayed, as illustrated in Fig. 3. Again, no changes to the viewing software are required, since all involved bit streams, i.e., both, encrypted and unencrypted, are format-compliant.

In this paper, we present a post-compression RoI encryption framework which can be used to effortlessly extend existing surveillance systems as described above. We describe its algorithms and evaluate its performance in detail, thereby providing insights into the practical challenges of extending existing surveillance systems. Before doing so, we describe related work as well as the paper's structure and contributions in the following two sections.

1.1 Related work

Related work on encryption frameworks for video surveillance can be divided into three categories: encryption systems, post-compression RoI encryption approaches (without a surrounding encryption system) and other related work. We discuss the relevant literature for each category below.

1.1.1 Encryption systems

Encryption systems typically perform RoI detection, encryption and RoI signalling, i.e., storing the RoI coordinates in order to have them available during decryption. We discuss relevant encryption systems which perform at least two of these three steps and explicitly exclude those systems which do not encrypt reversibly, but only obscure the captured images (e.g., through black boxes or blurring) to achieve privacy, since they do not allow to recover the original, unobfuscated images.

Chattopadhyay and Boult [5] propose an in-compression encryption system, which requires changing the JPEG encoder within the cameras, as opposed to our system, which does not require modifications of any surveillance equipment. For face detection, they use a background subtraction algorithm with thresholding, which requires capturing the background (a picture without any RoI) for each camera. In contrast, our system uses a dedicated face detector, which does not require capturing additional background images, thus being faster and cheaper to deploy. The RoI coordinates in Chattopadhyay's and Boult's approach are stored as comments in the

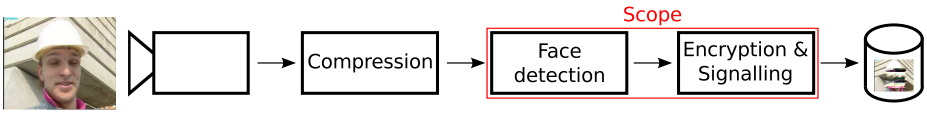


Fig. 1 Video surveillance system with post-compression RoI encryption functionality: Captured images are compressed, faces are detected, encrypted and signalled, and the encrypted images are stored in a data base. The focus of this paper are those components which are not present in regular, i.e., non-encrypting, video surveillance systems (components outside the illustrated scope).

JPEG file for decryption, similar to our approach. However, they do not specify how the RoI coordinates are represented and encoded, while we provide a detailed description of our RoI coordinate encoding.

Sohn et al. [36] describe an in-compression encryption system for Scalable Video Coding (SVC), where the camera images are transcoded using an additional transcoding server. As described in Section 1, this is undesirable due to the high computational complexity of the transcoding process, which is why our approach does not perform any transcoding operations. For face detection, Sohn et al. use the algorithm by Viola and Jones [42], which we use as well. Their approach supports only one RoI, which is impractical for real-world surveillance camera footage. In contrast, our approach supports an arbitrary number of RoI.

Dufaux et al. [11] as well as Martínez-Ponte et al. [24] and others propose encryption systems for JPEG 2000, which are implemented as in-compression encryption approaches, but could be implemented as a post-compression encryption approach like ours. While cameras delivering JPEG 2000 bit streams exist, JPEG is much more common, which is why our encryption system is based on JPEG. For face detection, Dufaux et al. as well as Martínez-Ponte et al. use the algorithm by Viola and Jones [42] which we use as well. While their discussions on signalling RoI remain theoretical with some JPEG-2000-specific suggestions, but without concrete implementations, we provide a detailed description and implementation of our RoI signalling algorithm.

Boult [3] describes a pre-compression encryption system for JPEG which requires modifications to the cameras or a separate encoding or transcoding server. Similar to the approaches described above, this is impractical. In addition, as any pre-compression approach, Boult's requires additional communication with the encoder in order to avoid compressing the encrypted RoI so that they remain decryptable. This yields a significant bit rate overhead as opposed to our approach, which has a negligibly small overhead. For face detection, Boult uses an unspecified face detection software, as opposed to the common algorithm by Viola and Jones [42] that we use. Similar to Chattopadhyay's and Boult's approach, RoI coordinates in Boult's approach are stored as comments in the JPEG file, although there is no description on the coordinate encoding, as opposed to our approach.

Iqbal et al. [14] propose an encryption system for H.264, which requires transcoding after RoI detection and additionally performs post-compression encryption when necessary. Besides the computational complexity of the required transcoding step, the format compliance of their encryption approach is questionable, likely requiring changes to the video decoder of the surveillance system. In contrast, our encryption approach is completely format-compliant and therefore works with off-the-shelf decoders. For face detection, we use the common algorithm by Viola and Jones [42], while Iqbal et al. have no explicit RoI detection mechanism, but assume that all relevant RoI information is provided, which is impractical. They use Motion Picture Experts Group (MPEG)-21 to store meta data from which the RoI coordinates can be derived. As MPEG-21 is Extensible Markup Language (XML)-based, it introduces a significant bit rate overhead, as opposed to our RoI signalling approach, which has a negligibly small overhead.

Senior et al. [34] give an abstract description of their privacy-preserving video surveillance system. Since they do not provide any implementation details or results whatsoever, a comparison with our encryption system is not possible.

1.1.2 Post-compression RoI encryption approaches

Although some post-compression RoI encryption approaches have been proposed, literature on the topic is sparse. Since JPEG and H.264 are the most common compression standards used in surveillance cameras, we limit this overview to these two. Furthermore, we limit approaches to those which are format-compliant, apart from a few notable exceptions.

As JPEG does not use intra prediction apart from Direct Current (DC) coefficients, the blocks a JPEG image consists of are basically independent from an encryption point of view. This means that almost every full encryption approach for JPEG can be trivially extended to be a RoI encryption approach.

Tang [38] proposes permuting the zig-zag scan order of Discrete Cosine Transform (DCT) coefficients, which requires bit-stream-level entropy- and run-length-transcoding. While this approach is of relatively low computational complexity, it decreases the compression performance and therefore increases the file size significantly. In contrast, our approach has a negligibly small overhead, i.e., virtually no impact on the compression performance.

Wu and Kuo [44] describe an encryption method which generates multiple Huffman tables and switches between them pseudo-randomly for each code word. Although this method is fast, it is not format-compliant, as opposed to ours (or MPEG-4 Intellectual Property Management and Protection (IPMP) by Wen et al. [43], since the generated bit stream cannot be parsed with an off-the-shelf JPEG encoder due to the incompatible Huffman code words. Note that Wu's and Kuo's method is technically no RoI encryption approach, but could be extended accordingly. However, this would not solve the problem of non-format-compliance.

DC and/or Alternating Current (AC) coefficient sign scrambling is a common encryption technique used for many DCT-based compression formats, including JPEG, e.g., the works of Zeng and Lei [47] as well as Lian et al. [22]. However, when only encrypting few and/or small RoI, the key space can be small enough to allow for practical attacks. Since our approach encrypts multiple bits per coefficient instead of only the sign bit, the key space and therefore the attack complexity are significantly higher.

Puech and Rodrigues [30,31] describe two methods where a number of bits at bit-stream level are encrypted starting from the DC coefficient [30] or the AC coefficient with the highest spatial frequency of each block [31], respectively. Similar to the coefficient sign encryption approaches described above, these methods have a lower attack complexity than our approach, since we encrypt all coefficients at bit-stream level.

Ye et al. [46], Lian et al. [22] as well as Niu et al. [27] propose encryption through block shuffling. Although we use a similar method in one step of our approach, our method is spatially limited and therefore allows for RoI encryption. Although the approaches by Ye et al., Lian et al. as well as Niu et al. could be extended to support RoI encryption, this would significantly reduce their key space and their attack complexity, as opposed to our approach, which uses additional encryption steps to assure a larger key space.

Yang et al. [45] describe an encryption method which swaps code words of equal length between blocks. Again, the key space of this method is very small when used for RoI encryption, as the number of code words with equal length within a RoI is typically much lower than the number of code words within a block, which our approach uses for swapping. Thus, the attack complexity of our approach is significantly higher than the complexity of the approach proposed by Yang et al.

For H.264, no post-compression RoI encryption algorithms have been proposed so far. However, Dufaux and Ebrahimi [9] describe an approach for MPEG-4 Part 2, which can be adapted for H.264. Although their encryption step is performed at bit-stream level, their method of avoiding prediction-related artifacts outside the RoI requires selective re-encoding of the video. This is impractical due to the necessary transcoding step and its associated computation time and bit rate overhead. In contrast, our approach requires no transcoding and has a negligibly small overhead.

1.1.3 Other related work

Privacy protection in video surveillance systems is an active research topic. Apart from the literature described above, we discuss other notable related papers. For the sake of conciseness, we do not aim at providing an exhaustive list, but focus on closely related work.

Cheung et al. [6] propose an alternative to RoI encryption by removing RoI from the video (by background pixel replacement) and embedding them back into the video using reversible data hiding. Although their approach could theoretically be implemented at bit-stream level, its large bit rate overhead and quality degradation when using off-the-shelf decoders are unacceptable for practical applications. In contrast, our approach has a negligibly small overhead and no quality degradation outside the RoI.

Dufaux and Ebrahimi [10], Newton et al. [26] and Melle and Dugelay [25] describe frameworks for assessing privacy protection solutions by applying face recognition algorithms to the obfuscated and/or encrypted images. The recognition rates are used as objective measures for determining the degree of RoI obfuscation. In our paper, we use the same (or comparable in the case of Melle's and Dugelay's assessment [25]) face databases for testing as they do, but provide both, objective and subjective evaluation. This way, the detection and recognition rates of human viewers can be compared and added to other objective measures.

1.2 Structure and contributions

This paper is structured as follows: In Section 2, we describe our encryption framework. In Section 3, we evaluate its performance and practical usefulness, before concluding the paper in Section 4.

This paper contributes an implementation of an encryption framework for surveillance systems which combines and extends existing algorithms for face detection, RoI encryption and RoI signalling. As opposed to previous frameworks, ours can be easily integrated into existing surveillance systems without the need to modify any surveillance equipment, making it very easy to deploy. Furthermore, this paper contributes an evaluation of the implemented framework in terms of both, objective and subjective measurements, pointing out challenges when extending existing surveillance systems.

2 Encryption framework

As described in Section 1, our encryption framework can be thought of as two black boxes – one for encryption and one

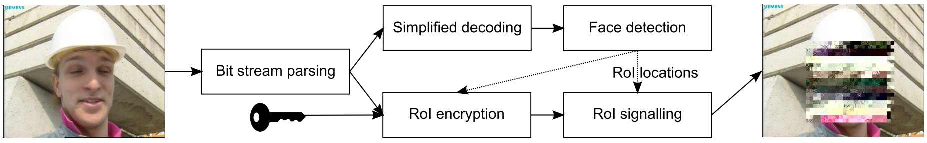


Fig. 4 Components of the encryption black box: An unencrypted input picture is parsed and decoded for face detection. The obtained face coordinates are used for limiting the encryption to said regions and written to the encrypted output file as they are required later for decryption

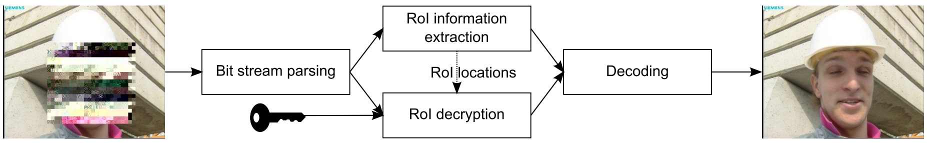


Fig. 5 Components of the decryption black box: An encrypted input picture is parsed, but not decoded. The RoI coordinates are extracted from the input file and used for limiting the decryption to said regions, resulting in an unencrypted output picture

for decryption. An abstract view of the components of both is depicted in Figures 4 and 5, respectively. Each of the main components and their interconnections are described in detail below.

2.1 Face detection

The first main component of our encryption framework depicted in Fig. 4 is face detection which is used to find the RoI locations and pass them to the RoI encryption algorithm. Since face detection is carried out in the image domain, it is necessary to decode the image for this step, which we do by using *OpenCV*[1]. As all other components operate at bit-stream level, i.e., without the necessity to decode image data, the decoding process for face detection can be simplified to processing gray-scale (luminance) data only. Since the chrominance channels are ignored for the actual face detection, a gray-scale version of the input image yields the same results at lower decoding complexity.
If the surveillance system itself provides face detection functionality, e.g., through the camera firmware or other existing components, this information can be used directly, e.g. as shown in the approach by Unterweger and Uhl [41]. This allows replacing the face detection step by communication with the existing face detection component. Our implementation supports this, but assumes no such component is available by default.
We use the *OpenCV* implementation of the common face detection approach by Viola and Jones [42] with the extended feature set from [23]. This is one of the best face detection approaches and implementations in terms of detection performance that is available for free at the time of writing [16, 33]. It uses a cascade of detectors in a sliding window on

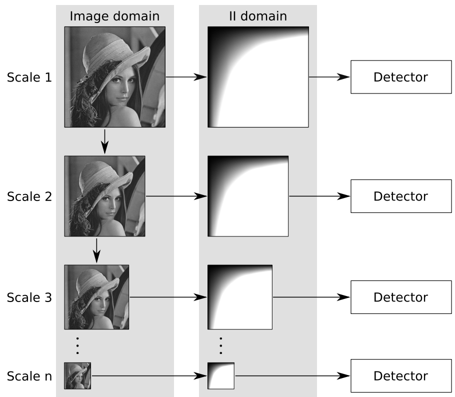


Fig. 6 Multi-scale face detection: The input image is scaled in the image domain. On each scale, an integral image (II) is calculated, followed by face detection using a detector cascade

the image. Each detector rejects non-face regions with high probability by thresholding sums of Haar-like features which are calculated efficiently by using integral images. This approach is used on multiple scales of the original input image, as illustrated in Fig. 6 to find faces of different size.
The speed and detection rate of the described implementation depends on two parameters: On the one hand, the *scale factor* parameter specifies the ratio r of image widths/heights between two adjacent scales, where $r > 1$. Higher values yield higher speed at a lower detection rate due to the smaller number of scales, while lower values yield lower speed at a higher detection rate.
On the other hand, the *min. neighbors* parameter defines the number of neighboring windows n with respect to the sliding

[1] `http://opencv.org/`

window which have to report detections as well so that one actual detection is returned. Higher values of n yield fewer detections with higher confidence, while lower values of n yield more detections with lower confidence.

For on-line processing in video surveillance, high speed and high detection rates would be desirable. However, since there is no straight-forward way to get both at the same time, a trade-off has to be found. Thus, we define three different parameter configurations for evaluation so that they cover about an order of magnitude in execution time around the *OpenCV* defaults:

– **Good**: $r = 1.05, n = 1$
– **Default**: $r = 1.1, n = 3$ (*OpenCV* defaults)
– **Fast**: $r = 1.25, n = 1$

We set the *min. neighbors* parameter to 1 (default value 3) in order to increase the number of detected faces. Although the confidence for each detected face is lower this way, the algorithm is less likely to miss faces, which would be undesirable in our use case. For all other parameters, we use default values.

Since the approach by Viola and Jones is not suitable to detect faces in a high number of different poses, we use two separate cascades – one for frontal and for profile face detection – and combine the results. As our encryption approach processes units of $16 \cdot 16$-pixel-sized blocks, all face coordinates are additionally rounded to the nearest block borders.

2.2 Encryption

For RoI encryption, we modify the full encryption approach proposed by Auer et al. [1]. Thus, in the following sections, we describe the original approach and our modifications, respectively.

2.2.1 Encryption approach by Auer et al.

The encryption approach proposed in [1] processes four $8 \cdot 8$-pixel-sized blocks (for typical 4:2:0 YCbCr subsampling [17]) at a time, i.e., it operates on $16 \cdot 16$-pixel-sized units. It encrypts AC and DC coefficients separately, i.e., a different key is used for each coefficient type. Both keys are initialization vectors for Advanced Encryption Standard (AES) encoders operating in Cipher Feedback (CFB) mode which generate pseudo-random bit sequences for encryption. DC coefficient encryption is applied to DC coefficient differences (relative to the DC coefficient of the preceding block) in the bit stream, since the JPEG format stores them instead of the actual full coefficient values. The actual encryption approach is based on the work of Niu et al. [27] and scrambles all DC value difference bits (not the preceding Huffman code words with length information) by xor-ing them with the pseudo-random bit sequence described above.

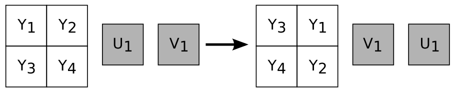


Fig. 7 Block order permutation by Auer et al. [1]: Blocks of the same color use the same Huffman code words. Their order is changed pseudo-randomly in the first AC coefficient encryption step

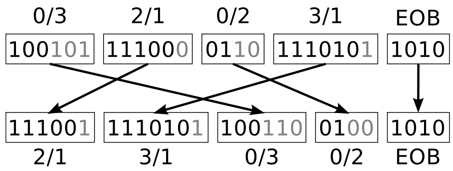


Fig. 8 Combined code-word-value order permutation and value scrambling adopted from Auer et al. [1]: The order of Huffman code words (black) representing run-length/value information and their associated values (grey) is changed pseudo-randomly in the second AC coefficient encryption step; the value bits (grey) are scrambled in the third AC coefficient encryption step.

AC coefficient encryption uses a different key and consists of three steps:

– **Block order permutation**: Those $8 \cdot 8$-pixel-sized blocks within a $16 \cdot 16$-pixel-sized unit which use the same Huffman code words are swapped pseudo-randomly so that their order is changed as depicted in Fig. 7.
– **Code-word-value order permutation**: For each block, all Huffman code words and their associated coefficient values are swapped pseudo-randomly so that their order is changed as depicted by the arrows and framed code-word-value pairs in Fig. 8.
– **Value scrambling**: The value bits associated with each Huffman code word are scrambled as depicted in Fig. 8.

In the second and third steps, the End Of Block (EOB) marker remains unchanged. A more detailed description of all encryption steps as well as a thorough security analysis can be found in [40, 1]. Note that, although Auer et al. [1] describe a full-picture encryption approach, their security analysis is limited to single blocks, so it can be used for our modified approach as well.

All described operations perform format-compliant changes at bit-stream level. They don't increase the bit stream length with the notable exception of encryption-induced `FF` bytes at whole byte positions, which require escaping. Conversely, previously escaped `FF` bytes may be changed through encryption, decreasing the bit stream length. This way, on average, the file size is expected to remain unchanged.

2.2.2 Proposed encryption approach

We extend the approach of Auer et al. described in the previous section so that it supports RoI encryption. Both, AC

Fig. 9 DC coefficient errors through difference value discrepancies: Limiting the approach of Auer et al. [1] to a RoI without modifications yields incorrect DC coefficient values outside the RoI. Original image from the *LIVE* reference picture set [35]

Fig. 10 AC (left) vs. AC and DC encryption (right): When DC encryption is used, the RoI is extended visually on the right in some cases to compensate for the encryption-induced DC difference discrepancy. Original (uncropped) image from the *LIVE* reference picture set [35]

and DC coefficient encryption can be enabled or disabled for each $16 \cdot 16$-pixel-sized block based on the coordinates returned by the face detection algorithm. Although this works trivially for AC coefficients without any modifications, DC coefficient encryption requires two modifications in order to function properly.

First, since DC coefficient differences are encrypted, limiting the DC coefficient difference encryption to a RoI modifies the sum of all differences within the RoI. This yields a discrepancy between the unencrypted and the encrypted DC coefficients outside the RoI. In the online viewing case, this would result in significant distortions of the image, as depicted in Fig. 9.

To solve this, the discrepancy between the encrypted and unencrypted coefficients is summed up during encryption within the RoI and added to the first DC coefficient difference outside the RoI to restore the original DC coefficient value. In case the DC coefficient difference exceeds the minimum (-2047) or maximum (2047) limit, it is distributed among as many blocks as necessary, visually extending the RoI as depicted in Fig. 10 (rightmost encrypted blocks in the right image). Practically, one or two extra blocks outside the actual RoI suffice for this compensation in most cases.

Second, if more than one block outside the RoI is required for the DC coefficient difference compensation described above, visual degradation may occur after decryption. In case n blocks are required for compensation, the DC coefficient difference of the n^{th} block is restored successfully. However, the other $n-1$ blocks still exhibit a discrepancy between their original DC coefficient differences and those which have been adjusted for partial compensation. Thus, the DC coefficient values of these blocks could not be restored to their original values.

To solve this, the maximum (accumulated) compensation value within the RoI during encryption is limited to the range between -1023 and 1023, respectively. If these limits are exceeded, none of the DC coefficients in the same block row are encrypted (but other RoI or rows are not affected by this). The decoder can detect this analogously and skip the decryption of the remaining coefficients.

In addition, the encryption per DC coefficient difference is limited to the 7 Least Significant Bits (LSB) (representing the value range of -127 to 127). In total, this limits the maximum (summed) compensation value to $\pm 1023 \pm 254 = \pm 1277$, i.e., even in the worst case, more than one block is only required for compensation if the first DC coefficient difference outside the RoI is outside the range $\pm 2047 \mp 1277 = \pm 770$.

As shown in Table 1, this only occurs in very few blocks at very high quality levels over all pictures from the *LIVE* reference picture set when using the JPEG reference software. Even in those affected blocks, it is highly unlikely that the (summed) compensation value is actually large enough to require further blocks for compensation. One exception occurs at 100% quality levels: One block in one of the compressed pictures contains a high DC coefficient difference (1847). For such rare cases, the limits can be reduced further if necessary.

Quality [%]	Critical blocks per picture	Max. DC diff.
0	0	2
5	0	11
10	0	23
15	0	35
20	0	47
25	0	58
30	0	68
35	0	81
40	0	92
45	0	103
50	0	115
55	0	132
60	0	142
65	0	168
70	0	184
75	0	231
80	0	308
85	0	370
90	0	615
95	0.2	924
100	30.7	1847

Table 1 DC coefficient difference statistics when compressing the *LIVE* reference picture set [35]: The average number of blocks for which the difference exceeds 770 is zero for all quality levels but 100%, except for very few blocks at 95% quality levels; the maximum difference in all pictures only exceeds 770 for 95 and 100% quality levels

Quality [%]	Unencrypted DC coeff. bits per picture [%]
0	0
5	0
10	0
15	0
20	0
25	0
30	0
35	0
40	0
45	0
50	0
55	0
60	0.01
65	0.03
70	0.04
75	0.12
80	0.34
85	0.54
90	1.56
95	2.93
100	6.31

Table 2 DC coefficient bit statistics when encrypting the *LIVE* reference picture set [35]: The average number of DC coefficient difference bits which remain unencrypted increases with quality levels

However, for almost all other cases (quality levels of 95% and below), no modifications are required.
Our implementation is based on the one from Auer et al. [1] which itself is based on *NanoJPEG*[2] that is written in C. Our modifications to the original implementation are as described above. Due to these modifications, an updated security analysis as presented in the next section is required.

2.2.3 Security analysis

Although a detailed security analysis of the original encryption approach of Auer et al. [1] is given in their paper, our modified approach described in Section 2.2.2 requires additional analysis. Since the AC coefficient encryption is the same as in [1], their results, which are per block and thus apply to RoI encryption as well, do not need to be re-analyzed. Since the DC coefficient encryption is independent from the AC coefficient encryption, attacks to the DC coefficient encryption do not affect the AC coefficient encryption. As our modified encryption algorithm does not encrypt some DC coefficient difference Most Significant Bits (MSB), an analysis of the number of unencrypted bits and its potential consequences is necessary.
Table 2 shows the average number of unencrypted DC coefficient difference bits in percent over all pictures from the *LIVE* reference picture set when using the JPEG reference software. All bits are encrypted up to quality levels of 55%. The number of unencrypted bits increases with quality levels of 60% and above, reaching more than 5% at 100% quality levels. Since encryption is critical at these quality levels, as explained in Section 2.2.2, it is not advisable to use our approach at 100% quality levels. For quality levels between 60% and 95%, replacement attacks for the encrypted bits have to be taken into consideration.

[2] `http://keyj.emphy.de/nanojpeg/`

2.2.4 Effect of replacement attacks

A straight-forward replacement attack for any selective encryption approach is to set all encrypted values to zero while keeping the unencrypted values. In our case, since AC encryption is performed separately, all AC coefficients are set to zero. DC coefficient differences which are encrypted entirely are also set to zero. In contrast, for those DC coefficient differences of which only the k LSB out of n are encrypted, the k LSB are set to zero, while the $n-k$ unencrypted MSB are kept as they are.
We use this replacement attack on our proposed approach and the one by Unterweger and Uhl [40]. The latter performs AC encryption in the same way as the approach by Auer et al. [1], but does not encrypt DC coefficient differences, i.e., they remain intact after the attack. Fig. 11 shows the results of the attack on a worst-case example image for JPEG quality levels of 90 (*a)* and *b)*) and 95% (*c)* and *d)*), respectively for our (*a)* and *c)*) and their approach (*b)* and *d)*), respectively. It is clear that the DC coefficients carry enough information to reconstruct a rough version of the image when using the encryption approach by Unterweger and Uhl, i.e., their approach can be attacked by setting the AC coefficients to

a) b) c) d)

Fig. 11 Effects of a replacement attack on our proposed encryption approach (*a)* and *c)*) vs. the encryption approach by Unterweger and Uhl [40] (*b)* and *d)*). *a)* and *b)* use JPEG quality levels of 90%, *c)* and *d)* quality levels of 95%. Original image from the *LIVE* reference picture set [35]

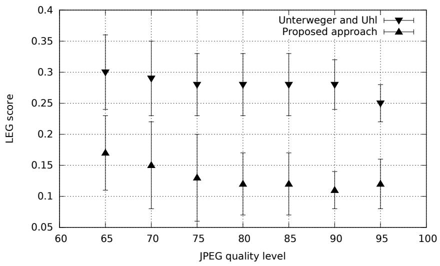


Fig. 12 LEG scores for the *LIVE* reference picture set [35] after a replacement attack on our proposed encryption approach vs. the approach by Unterweger and Uhl [40]. Error bars denote standard deviation

zero. In contrast, our approach reveals significantly less information of the original image, even when all encrypted DC coefficient difference bits are set to zero.

For quality levels of 95%, one could argue that the silhouette of the woman can still be reconstructed partially. However, at 90% and lower (not depicted), basically no relevant visual information can be extracted through a replacement attack in this worst-case example.

Fig. 12 visualizes the results of the attack for all relevant quality levels on all images from the *LIVE* data base. We use LEG [13] as a similarity metric to compare the unencrypted (original) images and their attacked counterparts since Peak Signal-to-Noise Ratio (PSNR) does not reflect subjective quality well. An LEG score of 1 means perfect similarity, while a score of 0 means total dissimilarity.

It can be seen that the attack on our approach yields significantly lower scores than the attack on the approach by Unterweger and Uhl. Although the LEG scores are relatively low in both cases, the scores for attacked images encrypted with our approach is close to zero, i.e., very dissimilar to the unencrypted images. This means that the two images have little in common and that our proposed encryption approach is not vulnerable to replacement attacks in practice.

Summarizing all security- and attack-related results, our approach is secure due to the additional DC coefficient encryption which is not present in the approach by Unterweger and Uhl (and therefore in the AC encryption portion of the approach by Auer et al.). However, it is recommended to use our approach only for JPEG quality levels of 90% and below to avoid the potential reconstruction of small amounts of image information due to unencrypted bits required for format-compliant decryption.

2.3 Signalling

In order for the decryption process to only decrypt RoI, the locations of the latter have to be signalled. This is done after encryption and consists of two basic steps: First, the location information for all RoI is encoded; second, the encoded information is embedded into the JPEG file. The following sections describe the encoding and the embedding process, respectively.

2.3.1 Coordinate encoding

We slightly modify the RoI encoding approach proposed by Engel et al. [12]. Due to the high similarity of the two approaches, we give a basic overview of the coordinate encoding steps and highlight the differences.

The coordinate encoding process consists of the following steps:

- **Indexing**: Each $16 \cdot 16$-pixel-sized block is assigned an index as depicted in Fig. 13. The top-left-most block is assigned index zero; the blocks to its right are assigned ascending indices in increments of one. This is continued for all blocks in the remaining rows up until the bottom-right-most block.
- **Initial representation**: Each RoI is represented by a tuple containing its first and its last block index. The RoI in Fig. 13 yield the representation $(8,16),(11,27),(30,30)$.
- **Size calculation**: Since the last block index is always larger than the first, it can be represented relative to the first in order to save bits. This is nearly equivalent to calculation the size (length) of the RoI, but takes into consideration that the size of a RoI cannot be zero. The RoI in Fig. 13 yield $(8,8),(11,16),(30,0)$. Note that the approach by Engel et al. uses the actual size of the RoI and reserves the value zero for signalling the end of the list of RoI.
- **Differential representation**: Each RoI but the first is represented relative to its predecessor, i.e., the two elements of the tuple it is represented by are subtracted from the two corresponding tuple elements its predecessor is represented by. The RoI in Fig. 13 yield the differential representation $(8,8),(3,8),(19,-8)$.
- **Variable length coding**: Each tuple element is encoded by a zeroth order signed Exponential Golomb code word [12, 15]. The encoded tuple elements are concatenated to a bit string. No additional delimiters are required since the code words can be separated from one another by design.

The approach by Engel et al. consists of two additional steps that we omitted. The first additional step is entropy coding using adaptive arithmetic coding, which we found to yield relatively little reduction in bit string length given the required computation time. The second additional step is optimizing the bit string size by changing the order of the RoI. Since this requires an exhaustive search which has a higher than exponential time complexity, it is practically infeasible when encoding ten or more RoI. Thus, we omit this step.

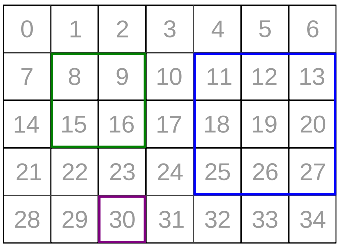


Fig. 13 Block indices for RoI signalling with three exemplary RoI: Block indices increase from left to right and top to bottom

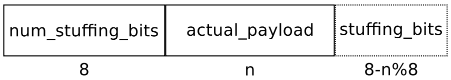


Fig. 14 `COM` segment payload extension: If the length n of the actual payload is not a multiple of a byte, stuffing bits are added. Their number is signalled before the start of the actual payload

2.3.2 Coordinate embedding

Engel et al. [12] evaluate multiple methods to embed the encoded RoI locations into a JPEG file. While their lossy approach provides sufficient embedding capacity, it cannot be used in the context of video surveillance since the inability to restore the original images after decryption is undesirable and may violate legal regulations.

For lossless embedding with unlimited capacity, they suggest inserting `COM` segments into the JPEG file. A decoder may skip these segments, if they are placed correctly, for example before the `SOI` marker. Thus, an off-the-shelf decoder ignores the `COM` segments, but our decryption system can use them to extract the RoI locations for decryption.

A `COM` segment consist of a marker, a length field and the payload. In our use case, the payload is basically the encoded bit string. However, since a `COM` segment has no means of signalling lengths with bit granularity, we extend our payload as depicted in Fig. 14.

Consider a payload (denoted as `actual_payload` in Fig. 14) of size n (bits). If the length of the actual payload is not a multiple of a byte, adding $8 - n\%8$ bits (`stuffing_bits`), where % denotes the modulo operator, extends the payload so that its length becomes a multiple of a byte. Thus, the total payload length can be specified by the length field of the `COM` segment (not depicted).

In addition, the number of the inserted `stuffing_bits` has to be signalled so that the decoder knows which bits belong to the actual payload. This information, denoted as `num_stuffing_bits`, is added before the actual payload and is a binary representation of the value $8 - n\%8$ in case the length of the actual payload is not a multiple of a byte, or zero otherwise. Although three bits would suffice to signal this information, it is easier, i.e., computationally less complex, for the decoder to parse if it is one byte, i.e., 8 bits, long.

In summary, the RoI location embedding induces an overhead by inserting the encoded bit string into the JPEG file, together with additional components required for decoding. These components yield a small overhead as follows: First, the `COM` marker adds two bytes. Second, the length field of the `COM` segment adds another two bytes. Third, `num_stuffing_bits` adds one more byte. Finally, between 0 and 7 `stuffing_bits` are required.

Our implementation is based on the one from Engel et al. [12]. Our modifications to the original implementation are as described above.

2.4 Decryption system

As depicted in Fig. 5, the decryption systems works analogously to the encryption system, but all operations are reversed. First, the RoI locations are extracted from the JPEG file. Second, the extracted locations are used by the RoI decryption algorithm to decide which blocks to decrypt. Finally, the decrypted JPEG file can be processed further, e.g., by decoding and displaying it. A face detection step is no longer required since all RoI locations are extracted from the signalling information.

2.5 Common auxiliary components

Fig.s 4 and 5 give an abstract view of the main components of the encryption and decryption black box, respectively. However, a number of common auxiliary components are not depicted therein to keep the figures clear. Due to their practical relevance, we describe these auxiliary components below.

2.5.1 Input/output modules

Since different surveillance systems deliver compressed images in different ways, our system uses a modular architecture for the input and output interfaces for both, the encryption and the decryption black box. This way, input and output modules can be combined as needed, supporting nearly arbitrary ways of compressed-image delivery.

The most common way for compressed-image delivery in state-of-the-art surveillance systems as of the time of writing is network-based file storage. Cameras are connected to a central file server over a network and upload the compressed images as files onto the server, e.g., using File Transfer Protocol (FTP). From there, they can be viewed or moved for

the purpose of archiving.
Our default input module reads files from an input folder and passes them one by one for further processing. Similarly, our default output module collects the processed files one by one and writes them an output folder. This is one of the simplest ways to encrypt and decrypt images as needed.
More complex and transparent ways of deployment, like direct network traffic interception, where the images are encrypted or decrypted on the fly, are possible as well. Due to the modular architecture of our framework, corresponding input and output modules can be written without changing any other parts of our software. Furthermore, input and output modules can be combined arbitrarily, depending on the needs for deployment in the surveillance system at hand.

2.5.2 Parallel processing

Depending on the hardware that our encryption framework is running on, likely not all the available computational power is used when processing images one by one. Thus, we use the Python *multiprocessing* module to parallelize encryption and decryption. Note that we do not use the similar *thread* and *threading* modules due to their inferior performance.
During initialization, *n* encryption or decryption processes (bold rectangles in the middle of Fig. 15) are created by the main process (bold rectangle on the left). The latter reads the data of the images to be encrypted or decrypted from the input module and puts them into a synchronized input queue. Each encryption or decryption process takes the data for one image from the input queue, processes it and puts the data for the corresponding processed image into a synchronized output queue. The main process takes the data of the processed images from the output queue and passes them to the output module. A special marker inserted by the main process into the input queue indicates that no more files should be processed, causing the encryption or decryption processes to terminate.

3 Evaluation

We evaluate our framework in terms of runtime, space overhead and face detection rate. In addition, we perform a subjective evaluation of images encrypted with our proposed encryption method.

3.1 Runtime

To evaluate the runtime of our framework, we use a total of six test sequences: *akiyo*, *foreman* and *crew* (all in Common Intermediate Format (CIF) resolution) serve as standard sequences with known characteristics and a varying amount of faces; *hall* (CIF) and *ice* (4CIF) serve as surveillance footage with easy and hard to detect faces, respectively; *vidyo1* (720p) serves as surveillance-like footage with high resolution.
In the following sections, we explain our test methodology and give the results of encoding (face detection plus encryption and signalling) and decoding times (signal extraction plus decryption), respectively. In addition, we provide an analysis of the time spent in each component of our framework.

3.1.1 Test methodology

To minimize measurement errors, each sequence is encoded and decoded separately for a total of eight times each – three times for cache warming and five times for actual processing. These five times are averaged and the standard deviation is calculated to determine the degree of fluctuation of the averaged results.
To simulate surveillance camera output, the input sequences are pre-encoded as JPEG files with *avconv*[3] with a quality parameter of 1, which roughly corresponds to a JPEG quality level of 90%. The files are placed on a Random Access Memory (RAM) drive (*ramfs* mount) and processed directly thereon in order to minimize input/output-related variations. We use a virtual server with 8 cores (on two physical 6-core Intel Xeon E5-2620 Central Processing Units (CPU)) and 8 GB of RAM running *CentOS* 6.4. The reason for using such powerful hardware to evaluate our framework is explained in Section 3.1.4.

3.1.2 Encoding time

The encoding time strongly depends on the face detection parameters. Fig. 16 shows the encoding times for the three parameter configurations described in Section 2.1. Note that the good configuration (bottom) uses a different y axis offset as it is about a power of ten slower than the fast configuration (top).
The total runtimes are clearly dependent on image resolution, with only small variations between sequences of the same resolution. For example, the *foreman* sequence (filled rectangles) with only one face requires only slightly lower processing time than the *crew* sequence (filled upside triangles) with about a dozen faces. The standard deviation is very small, indicating negligible measurement errors.
The *vidyo1* (720p) sequence (empty circles) requires about a factor of ten more processing power than the CIF sequences (filled geometric forms) in all configurations. The *ice* sequence (empty squares) are between the two in terms of runtime.
Parallelization decreases the runtimes of all sequences in all configurations when using up to the number of physically available cores (8 in our setup). Using more threads than

[3] `https://libav.org/avconv.html`

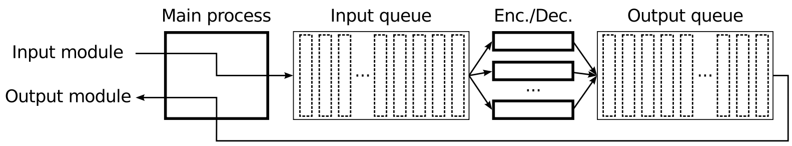


Fig. 15 Parallelization: The main process puts the data of images to be processed (dashed) into an input queue, from which the worker processes take one at the time. The processed data of each encoded or decoded image is put into an output queue, from which it is retrieved by the main process

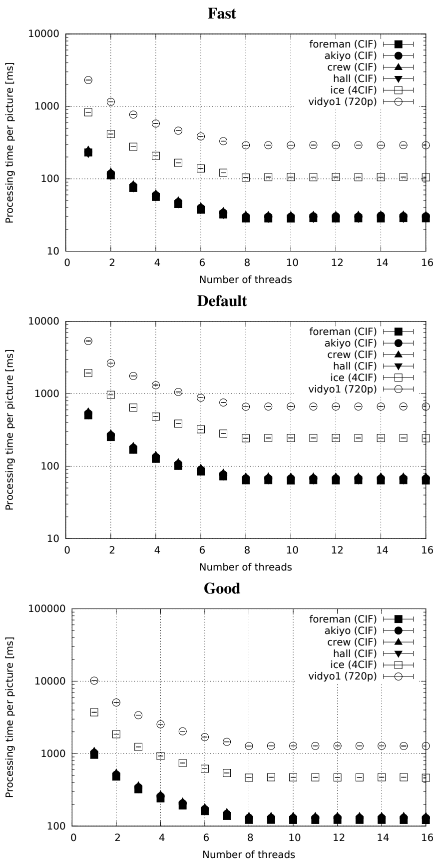


Fig. 16 Encoding times with different face detection configurations: Fast (top), default (middle), good (bottom). The y axis of the good configuration uses a different offset than the other two, illustrating that it is approximately a power of ten slower than the fast configuration

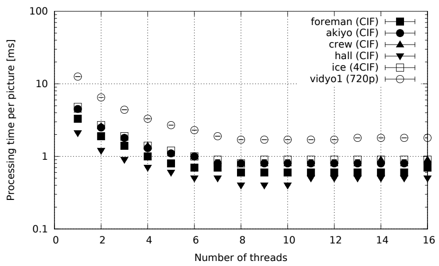


Fig. 17 Decoding times of different encoded sequences, i.e., encrypted files with signalling information: All sequences can be decoded in real-time

physical cores does not decrease the runtimes further. It is thus recommended to use an amount of threads which is equal to the number of physical cores to minimize runtimes. Real-time processing is only possible when using 8 threads and the fast face detection configuration for CIF sequences (filled geometric forms). With their 30 frames per second, the average execution time of about 30 ms is barely low enough for real-time encoding. However, as described in Section 3.1.4, this is mainly due to face detection and can be significantly improved when using different face detection algorithms.

3.1.3 Decoding time

Decoding the encoded files, i.e., extracting the signalling information and decrypting the RoI, is theoretically not dependent on face detection parameter configurations since no face detection is taking place – the locations of the faces are extracted from the signalling information. However, if more faces have been detected during encoding, more extraction and decryption operations are required during decoding. Thus, we use the images encoded with the good parameter configuration as a worst-case scenario. The decoding runtimes of the encoded files is shown in Fig. 17.
Again, the runtimes depend on the image resolution, but not as much as during encoding. Conversely, the variation between sequences of the same resolution is larger since the operations carried out during decoding depend on the

number and size of the RoI. Hence, a sequence like *akiyo* (filled squares) with one small face is decoded faster than a sequence like *foreman* (filled rectangles) with one larger face. Similarly, the *crew* sequence (filled upside triangles) with about a dozen of small faces requires more processing time than the *foreman* and *akiyo* sequences.
As during encoding, decoding benefits from parallelization up to the number of physical cores (8). Using a higher number of threads decreases performance slightly. Real-time processing is possible for all sequences of all resolutions, so on-line viewing is possible even with much less powerful hardware. For example, CIF sequences require less than one millisecond per frame for decoding when using 8 threads.

3.1.4 Breakdown of encoding time per component

Due to the significant differences in runtime between encoding and decoding, it is necessary to analyze the runtimes of each component of our framework individually. Since encryption and decryption as well as signalling and signal extraction are symmetric operations, it is sufficient to break down only the components of the encoding process for this – the components of the decoding process are practically identical, apart from the face detection component, which is only required for encoding.
Table 3 breaks down the runtime per component. The results have been obtained from the first of five runs (as described above) per sequence using one thread. The results when using multiple threads do not differ significantly. All values are rounded to two decimal places.
It is clear that face detection, in all configurations, requires by far the biggest portion of the total runtime. It includes the time for decoding the image since it is the only operation which requires the decoded picture data. The face detection time percentage increases with resolution since the number of scales at which the cascades have to be evaluated increases exponentially with the input resolution. Small variations between sequences of the same resolution can be observed depending on the image content due to the earlier (or later) rejection of false positives in the cascades. In nearly all scenarios, face detection requires more than 99% of the total runtime.
The remaining time is spent on encryption and signalling as well as on the intermediate time measurements themselves (denoted as miscellaneous). Encryption generally requires more time than signalling, which is not surprising given that signalling is a relatively simple operation. It accounts for about the same percentage of runtime as the measurement overhead (miscellaneous), which contributes only an insignificantly small amount to the total runtime.
In the good face detection parameter configuration, encryption, signalling and the measurement overhead combined account for between 4.8 ($0.37+0.05+0.06 = 0.48\%$ of 1000 ms for the *foreman* sequence from 16) and 18 ms ($0.12+0.03+0.03 = 0.18\%$ of 10000 ms for the *vidyo1* sequence), respectively, which is consistent (apart from rounding errors and slight timing variations) with the symmetric decoding operations whose runtime is depicted in Fig. 17. This shows that the face detection component of our system is the bottle neck and that all other components are sufficiently fast for real-time processing, even on less powerful hardware.

3.2 Space overhead and face detection rate

The rate of detected faces and the space overhead induced by RoI signalling are inherently coupled. Since a higher number of detected faces increases the number of RoI which need to be encrypted and signalled, face detection rates and signalling-induced space overhead are analyzed together. The same sequences as in Section 3.1 are used.

3.2.1 Test methodology

A ground truth for the faces in the used sequences is obtained through manual segmentation. The segmentation shape is rectangular since the automatic face detection implementation we use also returns rectangular regions. The coordinates of the segmented faces are rounded to the nearest $16 \cdot 16$ block border to minimize the influence of small segmentation variations on the one hand and, on the other hand, to ensure fairness as encryption and signalling of the automatically detected faces in our system only work with $16 \cdot 16$ pixel granularity as explained in Section 2.3.1.
We compare the automatically detected faces against the ground truth by processing each frame f_i of a sequence separately. Let $\mathbb{G}$ be the set of pixels in frame f_i which belong to at least one face from the ground truth, i.e., a pixel is an element of $\mathbb{G}$ if and only if it is contained in a face rectangle. Analogously, let $\mathbb{D}$ be the set of pixels in frame f_i which belong to at least one automatically detected face.
We evaluate the detection rate of the automatic face detection algorithm by calculating precision and recall. Precision specifies the percentage of actual face pixels returned by the automatic detector (relative to all pixels returned by the detector), while recall specifies the percentage of returned ground truth face pixels (relative to all available ground truth face pixels). Precision p_i and recall r_i are calculated as follows:

$$p_i = \frac{|\mathbb{G} \cap \mathbb{D}|}{\mathbb{D}} \tag{1}$$

$$r_i = \frac{|\mathbb{G} \cap \mathbb{D}|}{\mathbb{G}}, \tag{2}$$

where $||$ denotes the operator which returns the number of elements in a set, i.e., the area (number of pixels) in our case.

	Sequence	Face detection time [%]	Encryption time [%]	Signalling time [%]	Miscellaneous time [%]
Fast	*foreman* (CIF)	97.36	2.21	0.12	0.31
	akiyo (CIF)	97.58	1.90	0.29	0.23
	crew (CIF)	98.24	1.27	0.23	0.26
	hall (CIF)	99.31	0.36	0.09	0.24
	ice (4CIF)	99.64	0.25	0.03	0.08
	vidyo1 (720p)	99.42	0.46	0.07	0.05
Default	**Sequence**	**Face detection time [%]**	**Encryption time [%]**	**Signalling time [%]**	**Miscellaneous time [%]**
	foreman (CIF)	99.38	0.45	0.05	0.12
	akiyo (CIF)	98.88	0.87	0.13	0.12
	crew (CIF)	99.13	0.65	0.10	0.12
	hall (CIF)	99.80	0.07	0.01	0.12
	ice (4CIF)	99.83	0.11	0.01	0.05
	vidyo1 (720p)	99.70	0.24	0.04	0.02
Good	**Sequence**	**Face detection time [%]**	**Encryption time [%]**	**Signalling time [%]**	**Miscellaneous time [%]**
	foreman (CIF)	99.52	0.37	0.05	0.06
	akiyo (CIF)	99.38	0.48	0.07	0.07
	crew (CIF)	99.37	0.44	0.11	0.08
	hall (CIF)	99.70	0.20	0.03	0.07
	ice (4CIF)	99.77	0.18	0.01	0.04
	vidyo1 (720p)	99.82	0.12	0.03	0.03

Table 3 Encoding time breakdown for different face detection configurations: Fast (top), default (middle), good (bottom). In all configurations, face detection is the component which requires the biggest portion of the runtime

For the special case that both, $\mathbb{G}$ and $\mathbb{D}$, are empty, i.e., when there are no faces in frame f_i, we define p_i and r_i to be 1. The final precision and recall values p and r are calculated by averaging the values p_i and r_i, respectively.
For the space overhead calculations, each JPEG file (representing f_i) of the unencrypted input sequence, with size s_{u_i} is compared to the corresponding encrypted output file, with size s_{e_i}. The latter contains all required signalling information for RoI decryption as described in Section 2.3. The space overhead o_i for the JPEG file representing f_i is calculated as:

$$o_i = \frac{s_{e_i} - s_{u_i}}{s_{u_i}} \tag{3}$$

The final overhead value o is calculated by averaging the values o_i.

3.2.2 Overhead values and detection rates for OpenCV

The overhead values and detection rates for our proposed implementation described in Section 2 are listed in Table 4. Since the numbers are based on the face detector results, which depend on the latter's configuration, the results for the three parameter configurations described in Section 2.1 are listed separately.
It is clear that the space overhead is negligibly small (below 0.3% for all but the *crew* sequence). In general, sequences with more faces (e.g. *crew*) yield higher overhead values than sequences with fewer faces (e.g., *hall*) due to the higher signalling overhead. For sequences with a similar amount of faces (e.g., *foreman* and *akiyo*), the face size is an important factor influencing the overhead, since larger RoI dimensions inherently require more signalling bits.
Somewhat surprisingly, the face detection rates do not differ by a large amounts. In particular, the fast and good configurations, which differ by about one order of magnitude in terms of execution time (compare Section 3.1.2), yield nearly identical precision and recall values. This allows for two conclusions: First, the fast configuration is typically to be preferred over the good configuration due to its speed; second, the default configuration, which has lower recall values than both, the good and the fast configuration, is not recommended for this use case. From this and the configuration

Sequence	Overhead [%]	Precision [%]	Recall [%]
	Fast		
foreman (CIF)	0.136	51.9	82.1
akiyo (CIF)	0.243	53.2	99.5
crew (CIF)	0.355	33.2	53.3
hall (CIF)	0.102	62.7	28.2
ice (4CIF)	0.083	36.6	20.6
vidyo1 (720p)	0.161	18.1	70.8
	Default		
Sequence	**Overhead [%]**	**Precision [%]**	**Recall [%]**
foreman (CIF)	0.106	76.1	77.0
akiyo (CIF)	0.220	55.8	99.3
crew (CIF)	0.195	65.5	32.5
hall (CIF)	0.069	96.7	24.2
ice (4CIF)	0.045	82.9	8.8
vidyo1 (720p)	0.079	46.1	58.6
	Good		
Sequence	**Overhead [%]**	**Precision [%]**	**Recall [%]**
foreman (CIF)	0.137	50.3	82.4
akiyo (CIF)	0.242	53.2	99.5
crew (CIF)	0.353	33.2	53.3
hall (CIF)	0.102	62.3	28.3
ice (4CIF)	0.083	36.1	20.7
vidyo1 (720p)	0.163	18.0	70.6

Table 4 Overheads and detection rates for different face detection configurations: Fast (top), default (middle), good (bottom). All values are in percent.

parameters it is clear that a smaller value of the *min. neighbors* parameter has a much more significant impact on the detection rate than the *scale factor* parameter.

However, the detection rates for some sequences are not optimal in either of the three configurations. For the video surveillance use case it is essential to encrypt as many face pixels as possible, i.e., to achieve a recall of 1 (100%). The precision is secondary since encrypting non-faces (false positives) returned by the face detector is much less of an issue than "forgetting" to encrypt an actual face. Such "forgotten" faces are typically very small in size or they are in a pose between frontal and profile, which is nearly impossible to detect with the used cascades [42].

This phenomenon reflects in the recall values of all configurations. For example, the recall for the *akiyo* sequence which contains one frontal face is near-perfect (above 99%), while the recall for the *ice* sequence with up to 8 small faces in different poses barely exceeds 20% in the good and fast configurations. The recall values for the remaining sequences range from slightly below 30% (*hall*) to slightly above 80% (*foreman*).

This is clearly unacceptable for privacy preservation in a video surveillance system. It is also quite surprising, given the wide-spread usage of the *OpenCV* face detector in general, and in face encryption literature in particular. Due to these results, we additionally assessed the performance of two other face detectors – one free and one commercial – for comparison in the following sections.

3.2.3 Detection rates for Sun et al.'s approach

Sun et al. [37] propose a face detector based on deep convolutional network cascades. It is supposed to outperform the state of the apart as of 2013 and therefore the approach by Viola and Jones. We apply the same test methodology as described in Section 3.2.1, but with only one configuration, since there are no face detection parameters that can be configured.

Overhead measurements are omitted as explained below. Runtimes cannot be measured accurately since the only available implementation is a closed-source Windows executable. This additional evaluation therefore purely focuses on detection rates, which are shown in Table 5.

Although the precision values are significantly higher than those of *OpenCV*, the recall values are lower. The differences are significant for almost all sequences. Most notably, the *crew* and *ice* sequences have recall values of below 2.5 and 1.5%, respectively, which is not only surprising, but also clearly infeasible for face detection in surveillance systems. Since the *OpenCV* face detector outperforms the one by Sun et al. in terms of recall, which is the main relevant metric for the video surveillance use case, we omit the overhead measurements. For the sake of comparison, we assess the performance of another face detector in comparison to the one from *OpenCV* in the following section in order to draw more general conclusions.

Sequence	Precision [%]	Recall [%]
foreman (CIF)	75.3	80.7
akiyo (CIF)	100.0	60.2
crew (CIF)	70.2	2.3
hall (CIF)	100.0	77.3
ice (4CIF)	93.9	1.3
vidyo1 (720p)	88.7	58.4

Table 5 Detection rates for the face detector by Sun et al.: All recall values are lower than those of *OpenCV*

Sequence	Precision [%]	Recall [%]
foreman (CIF)	88.9	50.4
akiyo (CIF)	100.0	59.1
crew (CIF)	97.7	21.3
hall (CIF)	95.5	77.3
ice (4CIF)	88.1	1.5
vidyo1 (720p)	93.5	49.3

Table 6 Detection rates for the *KeyLemon* face detector: With the exception of the *hall* sequence, the recall values are significantly lower than those of *OpenCV*

3.2.4 Detection rates for KeyLemon

KeyLemon[4] is a Web service for face detection and recognition, which can be used for free with limitations on the amount of data processed per time unit. We use *KeyLemon*'s Python Application Programming Interface (API) to send input sequences image by image and receive coordinates of detected faces from the Web service. We apply the same test methodology as described in Section 3.2.1, but with only one configuration, since there are no face detection parameters that can be configured.

Overhead measurements are omitted as for the approach by Sun et al. Runtimes cannot be measured accurately due to network-induced API call delay and are therefore omitted as well. This additional evaluation therefore purely focuses on detection rates, which are shown in Table 6.

While the precision values are significantly higher (between 80 and 100%) than those of *OpenCV*, the recall values are significantly lower for all but one sequence (which is why we omit additional overhead measurements). The *hall* sequence yields a recall value of 77%, while the latter is below 30% when using *OpenCV*. Conversely, for all other sequences, the recall values are between approximately 40 and 90% lower (relative to the *OpenCV* results) than the *OpenCV* results.

This allows to draw three conclusions: First, commercial face detection solutions do not necessarily outperform freely available state-of-the-art face detectors. Second, the state-of-the-art face detection algorithm proposed by Viola and Jones [42] and its implementation in *OpenCV* (as well the one by

[4] `https://www.keylemon.com/`

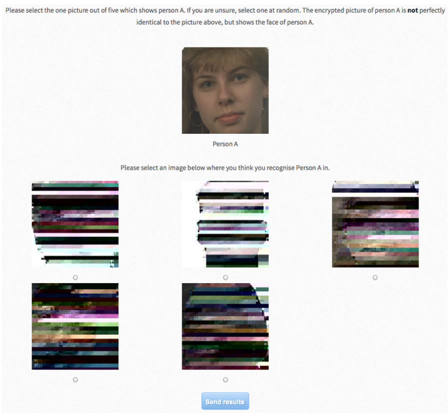


Fig. 18 A (cropped) screenshot of our survey system: The user has to select the encrypted picture which depicts the same person whose face is displayed unencrypted. This example shows DC-only encryption

Sun et al. [37]) are not suitable for use in surveillance systems due to their low recall values. Third, the face detection algorithm in our encryption framework should be replaced. This can be done easily due to the modular design of our framework. It is planned to update our software as soon as a more suitable approach is published.

3.3 Subjective evaluation

In order to assess the security of our proposed encryption approach from a practical point of view, we perform a subjective evaluation. We use the *Color FERET* database [29, 28] for this.

3.3.1 Test methodology

We implemented a survey platform in PHP. It displays a page with an image of a face from the *Color FERET* database, together with five encrypted faces, as depicted in Fig. 18. One of these five faces is of the same person whose unencrypted face is shown. The user has to decide which one it is and is forced to choose randomly when unsure. The encrypted and unencrypted image are not identical, as described in detail below, since this would not be realistic in a surveillance scenario.
We limit the amount of faces by only using the first frontal image (filename postfix *fa*) of each person photographed on April 22, 1994 in the *Color FERET* database as reference (unencrypted) image and the last alternative frontal image (filename postfix *fb*) of the same person as the basis for an encrypted version. If there are not both versions of frontal face images for a person from the database, this person is excluded from our image set. In total, images of 187 different people are used.
We create JPEG files from all images as described in Section 3.1.1 and use an automatic face detector to extract the face areas, which are shown in the survey. As the dimensions of the detected faces vary slightly, we resize them to $200 \cdot 200$ pixels for display in the Web browser so that the image size cannot be used as a clue to identify a person. Although this changes the aspect ratio of some images, the difference is very small (a few pixels in one dimension) and therefore negligible.
We assess the security of three encryption methods:

– AC-only encryption by Unterweger and Uhl [40] (which is identical to the AC encryption part of Auer et al. [1] and our proposed approach)
– Our encryption approach as described in Section 2.2.2
– DC-only encryption as a variant of our encryption approach which omits the AC encryption part from Auer et al. [1]

Users of our survey system see images encrypted with alternating encryption approaches following the pattern ABCABC..., where A stands for AC-only, B for DC-only and C for our proposed encryption approach. Each user sees a total of 30 pages as depicted in Fig. 18, i.e., 10 per encryption method. Examples for AC-only encryption and our proposed encryption method can be found in Fig. 10; an example of DC-only encryption is depicted in Fig.18.
Let c_i be the number of correctly recognized faces by a user when encryption method i is used. Correct in this context means that the user selected one and only one image and that the selected image actually shows the same person that is displayed in the unencrypted image. Analogously, let n_i be the number of incorrectly recognized faces. Incorrect recognition is defined as the opposite of correct recognition, i.e., partially correct selections (where multiple faces are selected and one of them is the correct one) are interpreted as incorrect.
We determine the recognition rate rr_i per encryption method, which is calculated for each encryption method i as:

$$rr_i = \frac{c_i}{d_i} \quad (4)$$

3.3.2 Recognition rates

The recognition rates for the different encryption methods are listed in Table 7, based on 460 recognition tasks (46 participants) per encryption method in total. For AC-only encryption, nearly all faces have been successfully recognized. This demonstrates that the theoretical attack described in Section 2.2.4 is of practical relevance. It also shows that the encryption approach by Unterweger and Uhl by itself is not sufficient to protect faces in a surveillance use case.
DC-only encryption exhibits lower recognition rates than

Encryption method	Recognition rate [%]
AC-only (Unterweger and Uhl [40])	93.92
DC-only	58.47
Proposed method (AC plus DC)	40.08

Table 7 Recognition rates for different encryption methods: AC-only and DC-only encryption are relatively insecure, while the proposed encryption approach makes recognition significantly harder

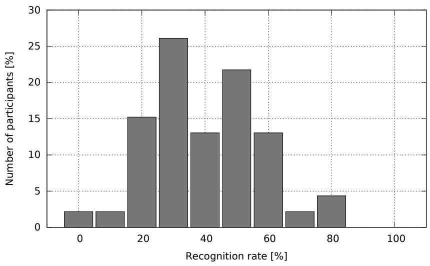


Fig. 19 Distribution of recognition rates for the proposed encryption approach: Most participants achieve recognition rates between 30 and 50%

AC-only encryption, but must still be considered relatively insecure due to the fact that more than half of the faces can still be recognized. Conversely, the proposed encryption approach, which combines AC and DC encryption, has significantly lower recognition rates.

However, the rates are still higher than the probability of guessing, which is 20% (one out of five). It is therefore necessary to analyze the results for this method in detail. Fig. 19 breaks down the participants by recognition rate. It is clear that a large amount of participants score around 30%, which is slightly above the probability of guessing. In addition, a few participants are able to score around 50%.

Although it would seem like there are two groups of participants – experts and non-experts –, we refrain from this particular distinction because our data does not support this assumption. Rather, a number of participants which we would consider to be non-experts scored relatively high, while a number of experts scored relatively low.

To inquire potential reasons for above-average recognition rates, we asked some of the participants who scored 50% or higher how they were able to recognize faces despite the encryption (as illustrated in Fig. 10). From this inquiry, the following results have been obtained:

- **Face borders**: The background of all images is white, leaving nearly no AC or DC differences to encrypt outside the faces, as opposed to inside the faces (which can also be seen for DC-only encryption in Fig. 18). Thus, the face borders are visible with block granularity due to the sudden change between weakly and strongly encrypted image regions. This information suffices to exclude at least some face candidates in the recognition process.
- **Head form**: The face detector returns rectangular regions which sometimes include hair, ears or both. Depending on whether these parts of the head are included, the form of the encrypted region is influenced. In combination with the face borders mentioned above, it is possible to exclude face candidates based on the head form, e.g., due to short vs. long hair.

This allows to draw two conclusions: First, a face detector returning more consistent face regions would be preferred in order to avoid clues about the head form. Second, solid-colored backgrounds make perceptually consistent encryption hard. In practice, however, this is not a problem since backgrounds vary and potential attackers typically cannot rely on the simple case of a uniform test set like in our assessment. Thus, our encryption approach is expected to yield lower recognition rates for non-uniform sets of faces.

Still, faces encrypted with our proposed encryption approach are already significantly harder to recognize than faces encrypted with similar encryption approaches, as shown above. While the recognition rates for our approach are, on average, higher than the probability of guessing, there is still a 6-in-10 chance that the wrong face is chosen. This may or may not extend to the variety of possible practical video surveillance scenarios.

Regardless, it is hard to judge whether the performance of our approach is sufficient in practice despite its theoretical security, mostly due to the lack of subjective evaluations of other encryption approaches. To our knowledge, no comparable evaluation of encryption approaches has been performed in the literature. It is possible that there are no other encryption approaches which achieve probability-of-guessing performance in subjective evaluations. The evaluation of more encryption methods for comparable results therefore remains future work.

3.3.3 Notes on evaluation design

Before concluding this paper, we would like to point out traps and pitfalls in evaluation design that lead us to perform the subjective evaluation a total of three times. We hope that the following suggestions help other researchers who evaluate encryption approaches in the future to avoid these traps and pitfalls, some of which are quite surprising and show unexpected creativity:

- **Image sizes**: If the encrypted images that the user can choose from differ in size, e.g., due to different face sizes, the size of the unencrypted images may suffice to guess the correct encrypted image with high probability. We therefore recommend to resize the images so that they all have the same size.

- **Face variations**: When using the exact same image of a person to create an encrypted and an unencrypted version, the average luminance as well as high-contrast borders in both versions may give enough clues to recognize the encrypted face with high accuracy. We therefore recommend to use slightly different pictures of the same person for the encrypted and unencrypted version, respectively.
- **File names**: The naming convention of the files displayed in the Web browser allows associating encrypted files with their unencrypted counterparts when examining the image properties in the browser. We therefore recommend using either random file names or at least a naming scheme from which no association between the encrypted and unencrypted images is possible.
- **File sizes**: Similar to file names, the size of a file allows excluding some encrypted images of significantly different sizes. This makes a correct guess more likely and increases recognition accuracy. We therefore recommend converting the files to be displayed to a lossless format first (like Portable Network Graphics (PNG)) to make recognition by exclusion of file sizes harder.

In summary, beware of meta data and the clues that they give. Otherwise, repeating evaluations may be a time consuming endeavor for both, the evaluation designers and the users.

4 Conclusion

We presented a full-featured post-compression encryption framework for video surveillance systems. It detects, encrypts and signals faces with negligibly low space overhead. Due to its modular design and parallelization efforts, our encryption framework is able to operate in real time and with minimal integration effort, allowing for easy deployment in existing surveillance systems. Our evaluations show that the performance of state-of-the-art face detectors are the main limitation of our proposed framework. Despite requiring about 99% of the total runtime, the tested face detectors only find a fraction of the faces in our evaluation sequences. This shows that these detectors are not suitable for video surveillance use cases. Moreover, we performed a subjective evaluation of our proposed encryption approach which shows that it makes face recognition harder than comparable approaches.

Acknowledgements The authors would like to thank Stefan Auer and Alexander Bliem for their initial involvement in the region of interest encryption implementation and their ideas for DC correction. In addition, the authors would like to thank Heinz Hofbauer for his valuable suggestions regarding the face detection performance assessment and the image metrics used for security evaluation.
Furthermore, the authors thank all the volunteering participants of their face encryption survey. Moreover, the authors thank *KeyLemon* for providing higher data limits per time unit for batch face detection.
Portions of the research in this paper use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office. This work is supported by FFG Bridge project 832082.

References

1. Auer, S., Bliem, A., Engel, D., Uhl, A., Unterweger, A.: Bitstream-Based JPEG Encryption in Real-time. International Journal of Digital Crime and Forensics **5**(3), 1–14 (2013)
2. Bergeron, C., Lamy-Bergor, C.: Compliant selective encryption for H.264/AVC video streams. In: Proceedings of the IEEE Workshop on Multimedia Signal Processing, MMSP'05, pp. 1–4 (2005). DOI 10.1109/MMSP.2005.248641
3. Boult, T.E.: PICO: Privacy through invertible cryptographic obscuration. In: IEEE/NFS Workshop on Computer Vision for Interactive and Intelligent Environments, pp. 27–38. Lexington, KY, USA (2005)
4. Carrillo, P., Kalva, H., Magliveras, S.: Compression Independent Reversible Encryption for Privacy in Video Surveillance. EURASIP Journal on Information Security **2009**, 1–13 (2009)
5. Chattopadhyay, A., Boult, T.: PrivacyCam: a privacy preserving camera using uclinux on the blackfin DSP. In: IEEE Conference on Computer Vision and Pattern Recognition 2007 (CVPR'07), pp. 1–8. Minneapolis, MN, USA (2007)
6. Cheung, S.S., Paruchuri, J.K., Nguyen, T.P.: Managing privacy data in pervasive camera networks. In: IEEE International Conference on Image Processing 2008 (ICIP'08), pp. 1676–1679. San Diego, CA, USA (2008)
7. Choi, S., Han, J.W., Cho, H.: Privacy-Preserving H.264 Video Encryption Scheme. ETRI Journal **33**(6), 935–944 (2011)
8. Dufaux, F., Ebrahimi, T.: Scrambling for Anonymous Visual Communications. In: Proceedings of SPIE, Applications of Digital Image Processing XXVIII, vol. 5909. SPIE (2005)
9. Dufaux, F., Ebrahimi, T.: Scrambling for privacy protection in video surveillance systems. IEEE Transactions on Circuits and Systems for Video Technology **18**(8), 1168–1174 (2008). DOI 10.1109/TCSVT.2008.928225
10. Dufaux, F., Ebrahimi, T.: A framework for the validation of privacy protection solutions in video surveillance. In: Proceedings of the IEEE International Conference on Multimedia & Expo, ICME '10, pp. 66–71. IEEE, Singapore (2010)
11. Dufaux, F., Ouaret, M., Abdeljaoued, Y., Navarro, A., Vergnenegre, F., Ebrahimi, T.: Privacy Enabling Technology for Video Surveillance. In: SPIE Mobile Multimedia/Image Processing for Military and Security Applications, Lecture Notes in Computer Science. IEEE (2006)
12. Engel, D., Uhl, A., Unterweger, A.: Region of Interest Signalling for Encrypted JPEG Images. In: IH&MMSec'13: Proceedings of the 1st ACM Workshop on Information Hiding and Multimedia Security, pp. 165–174. ACM (2013)
13. Hofbauer, H., Uhl, A.: An effective and efficient visual quality index based on local edge gradients. In: IEEE 3rd European Workshop on Visual Information Processing, p. 6pp. Paris, France (2011)
14. Iqbal, R., Shahabuddin, S., Shirmohammadi, S.: Compressed-domain spatial adaptation resilient perceptual encryption of live H.264 video. In: 2010 10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA), pp. 472–475 (2010)
15. ITU-T H.264: Advanced video coding for generic audiovisual services (2007). `http://www.itu.int/rec/T-REC-H.264-200711-I/en`
16. Jain, V., Learned-Miller, E.: Fddb: A benchmark for face detection in unconstrained settings. Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst (2010). `http://people.cs.umass.edu/~elm/papers/fddb.pdf`

17. Kerr, D.A.: Chrominance Subsampling in Digital Images. `http://dougkerr.net/pumpkin/articles/Subsampling.pdf` (2012)
18. Khan, M., Jeoti, V., Malik, A.: Perceptual encryption of JPEG compressed images using DCT coefficients and splitting of DC coefficients into bitplanes. In: 2010 International Conference on Intelligent and Advanced Systems (ICIAS), pp. 1–6 (2010)
19. Kim, Y., Yin, S., Bae, T., Ro, Y.: A selective video encryption for the region of interest in scalable video coding. In: Proceedings of the TENCON 2007 - IEEE Region 10 Conference, pp. 1–4. Taipei, Taiwan (2007)
20. Kwon, S.G., Choi, W.I., Jeon, B.: Digital video scrambling using motion vector and slice relocation. In: Proceedings of Second International Conference of Image Analysis and Recognition, ICIAR'05, *Lecture Notes in Computer Science*, vol. 3656, pp. 207–214. Springer-Verlag, Toronto, Canada (2005)
21. Lian, S., Sun, J., Wang, Z.: A novel image encryption scheme based-on jpeg encoding. In: Proceedings of the Eighth International Conference on Information Visualisation 2004 (IV 2004), pp. 217–220 (2004)
22. Lian, S., Sun, J., Zhang, D., Wang, Z.: A selective image encryption scheme based on JPEG2000 codec. In: Y.N. K. Aizawa, S. Satoh (eds.) Proceedings of the 5th Pacific Rim Conference on Multimedia, *Lecture Notes in Computer Science*, vol. 3332, pp. 65–72. Springer-Verlag (2004)
23. Lienhart, R., Maydt, J.: An extended set of Haar-like features for rapid object detection. In: 2002 International Conference on Image Processing, pp. I–900–I–903 vol.1 (2002)
24. Martinez-Ponte, I., Desurmont, X., Meessen, J., Delaigle, J.F.: Robust human face hiding ensuring privacy. In: Proceedings of the 6th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'05) (2005)
25. Melle, A., Dugelay, J.L.: Scrambling Faces for Privacy Protection Using Background Self-Similarities. In: 21st IEEE International Conference on Image Processing (ICIP 2014). IEEE, Paris, France (2014)
26. Newton, E., Sweeney, L., Malin, B.: Preserving privacy by de-identifying face images. IEEE Transactions on Knowledge and Data Engineering **17**(2), 232–243 (2005)
27. Niu, X., Zhou, C., Ding, J., Yang, B.: JPEG Encryption with File Size Preservation. In: International Conference on Intelligent Information Hiding and Multimedia Signal Processing 2008 (IIHMSP '08), pp. 308–311 (2008)
28. Phillips, P., Moon, H., Rizvi, S., Rauss, P.: The FERET evaluation methodology for face-recognition algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(10), 1090–1104 (2000)
29. Phillips, P., Wechsler, H., Huang, J., Rauss, P.J.: The FERET database and evaluation procedure for face-recognition algorithms. Image and Vision Computing **16**(5), 295–306 (1998)
30. Puech, W., Rodrigues, J.M.: Crypto-Compression of Medical Images by Selective Encryption of DCT. In: European Signal Processing Conference 2005 (EUSIPCO'05) (2005)
31. Puech, W., Rodrigues, J.M.: Analysis and cryptanalysis of a selective encryption method for JPEG images. In: WIAMIS '07: Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services. IEEE Computer Society, Washington, DC, USA (2007). DOI http://dx.doi.org/10.1109/WIAMIS.2007.21
32. Rahman, S.M.M., Hossain, M.A., Mouftah, H., Saddik, A.E., Okamoto, E.: A real-time privacy-sensitive data hiding approach based on chaos cryptography. In: Proc. of IEEE International Conference on Multimedia & Expo, pp. 72–77. Suntec City, Singapore (2010)
33. Santana, M.C., Déniz-Suárez, O., Hernández-Sosa, D., Lorenzo, J.: A comparison of face and facial feature detectors based on the viola-jones general object detection framework. Machine Vision and Applications **22**(3), 481–494 (2011)
34. Senior, A., Pankanti, S., Hampapur, A., Brown, L., tian, Y.L., Ekin, A., Connell, J., Shu, C.F., Lu, M.: Enabling Video Privacy through Computer Vision. IEEE Security and Privacy **3**(3), 50–57 (2005)
35. Seshadrinathan, K., Soundararajan, R., Bovik, A., Cormack, L.: Study of Subjective and Objective Quality Assessment of Video. IEEE Transactions on Image Processing **19**(6), 1427–1441 (2010)
36. Sohn, H., Anzaku, E., Neve, W.D., Ro, Y.M., Plataniotis, K.: Privacy protection in video surveillance systems using scalable video coding. In: Proceedings of the Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, pp. 424–429. Genova, Italy (2009)
37. Sun, Y., Wang, X., Tang, X.: Deep Convolutional Network Cascade for Facial Point Detection. In: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13, pp. 3476–3483. IEEE Computer Society, Washington, DC, USA (2013)
38. Tang, L.: Methods for encrypting and decrypting MPEG video data efficiently. In: Proceedings of the ACM Multimedia 1996, pp. 219–229. Boston, USA (1996)
39. Tong, L., Dai, F., Zhang, Y., Li, J.: Restricted H.264/AVC video coding for privacy region scrambling. In: 2010 17th IEEE International Conference on Image Processing (ICIP), pp. 2089–2092 (2010)
40. Unterweger, A., Uhl, A.: Length-preserving Bit-stream-based JPEG Encryption. In: MM&Sec'12: Proceedings of the 14th ACM Multimedia and Security Workshop, pp. 85–89. ACM (2012)
41. Unterweger, A., Uhl, A.: Slice groups for post-compression region of interest encryption in H.264/AVC and its scalable extension. Signal Processing: Image Communication (2014). Accepted
42. Viola, P., Jones, M.: Robust Real-time Object detection. In: International Journal of Computer Vision, vol. 57, pp. 137–154 (2001)
43. Wen, J., Severa, M., Zeng, W., Luttrell, M., Jin, W.: A format-compliant configurable encryption framework for access control of video. IEEE Transactions on Circuits and Systems for Video Technology **12**(6), 545–557 (2002)
44. Wu, C.P., Kuo, C.C.J.: Fast encryption methods for audiovisual data confidentiality. In: SPIE Photonics East - Symposium on Voice, Video, and Data Communications, vol. 4209, pp. 284–295. Boston, MA, USA (2000)
45. Yang, B., Zhou, C.Q., Busch, C., Niu, X.M.: Transparent and perceptually enhanced JPEG image encryption. In: 16th International Conference on Digital Signal Processing, pp. 1–6 (2009)
46. Ye, Y., Zhengquan, X., Wei, L.: A Compressed Video Encryption Approach Based on Spatial Shuffling. In: 8th International Conference on Signal Processing, vol. 4, pp. 16–20 (2006)
47. Zeng, W., Lei, S.: Efficient frequency domain selective scrambling of digital video. IEEE Transactions on Multimedia **5**(1), 118–129 (2003)

BIT-STREAM-BASED ENCRYPTION FOR REGIONS OF INTEREST IN H.264/AVC VIDEOS WITH DRIFT MINIMIZATION

Andreas Unterweger⋆*, Jan De Cock*†*, Andreas Uhl*⋆

⋆ University of Salzburg, Department of Computer Sciences, Salzburg, Austria
† Ghent University – iMinds, ELIS – Multimedia Lab, Ledeberg-Ghent, Belgium

ABSTRACT

We propose a new encryption approach for regions of interest in H.264/AVC bit streams. By encrypting at bit stream level and applying drift minimization techniques, we reduce the processing time by up to 45% compared to full re-encoding. Depending on the input video quality, our approach induces an overhead between -0.5 and 1.5% (high resolution sequences) and -0.5 and 3% (low resolution sequences), respectively, to minimize the drift outside the regions of interest. The quality degradation in these regions remains small in most cases, and moderate in a worst-case scenario with a high number of small regions of interest.

Index Terms— H.264/AVC, RoI, Encryption, Drift, Bit Stream

1. INTRODUCTION

In surveillance videos, regions of interest (RoI) like people's faces are often encrypted in order to preserve their privacy. The images are not encrypted entirely so that people's actions can still be seen unencrypted by surveillance personnel to determine whether intervention is required. In addition, decryption allows law enforcement to recover the identities later if necessary. This is not possible with other forms of de-identification like pixelation or blurring (e.g. [1, 2, 3]). An example of a surveillance system with RoI encryption is depicted in Fig. 1.

In this paper, we assume a video surveillance scenario where a typical surveillance camera (as of 2015) outputs compressed videos in the form of Motion JPEG [4] or H.264/AVC bit streams [5] and provides information on the location of the RoI through face detection to the encryption system as in [6].

Many approaches for RoI encryption have been proposed. Most of them either perform encryption format-independently in the image domain (e.g., [7, 8, 2]) or format-family-dependently in the transform domain (e.g., [9, 10, 11]). However, this is not practical since neither the captured images nor the encoder within the camera can be modified in typical surveillance equipment. The alternative of applying these methods by decoding the video stream received from the camera and re-encoding it is considered too time consuming and therefore impractical.

Approaches for bit-stream-based RoI encryption are very sparse. Although solutions for Motion JPEG exist [12, 13, 14], all of the H.264/AVC-focused approaches have severe disadvantages. Note that we only consider RoI encryption approaches, i.e., those which aim at maintaining the visual information outside of the RoI.

Dufaux et al. [15] describe an approach for MPEG-4 Part 2 which can be extended to H.264/AVC (and other DCT-based formats). Although their encryption algorithm can be performed at bit stream level, their method of preventing drift, i.e., the propagation of encrypted pixels into non-RoI areas, requires selectively re-encoding the video from the first encrypted frame onwards. As explained above, this is impractical due to its high computational complexity.

The approaches of Iqbal et al. [16] and Unterweger et al. [6] require bit streams in which all RoIs are in separate slices or slice groups to prevent spatial drift through the imposed prediction borders. This is a serious restriction since it requires the camera to reliably detect RoI and to create according slices. Furthermore, the authors do not discuss temporal drift, which is an important issue addressed in this paper.

Our work aims at minimizing the amount of drift after encryption while significantly reducing the computational complexity required for processing. This way, we contribute an alternative to the infeasible full re-encoding techniques at the cost of a small amount of drift outside the RoI.

This paper is structured as follows: In Section 2, we present our encryption approach. In Section 3, we describe how we minimize drift. Finally, we present a practical evaluation of our method in Section 4 before concluding the paper.

2. ENCRYPTION APPROACH

For all blocks inside a RoI, we perform a three-step bit-stream-level coefficient encryption as follows: First, we change the signs of all AC coefficients by xor-ing the original sign bits with the output of a one-time pad. Second, we encrypt the DC coefficient signs in the same way if the coefficients are stored directly in the bit stream, i.e., if the processed block does not use $16 \cdot 16$ intra prediction, where

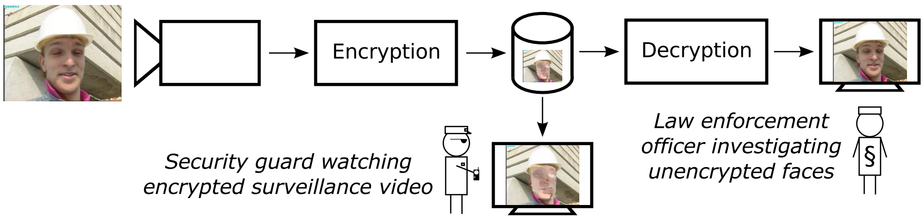


Fig. 1. Surveillance system use case: Images captured by a surveillance camera are encrypted and stored in a database. From there, the encrypted images can either be viewed directly or decrypted and used for legal investigations.

an additional Hadamard transform is used (which makes it impossible to change the DC coefficient signs at bit stream level). Third, we change the least significant bit of all AC coefficients whose absolute value is greater than one. This can be done by entropy code word replacement at bit stream level.

We also perform encryption on all blocks which are not fully, but only in parts inside of a RoI. This assures that no information at the borders of RoI leaks at the cost of encrypting a small number of non-RoI pixels as well. We limit the encryption to the luminance channel since hardly any identification-related information is contained in the chrominance channels and extending our approach would be trivial.

When applying encryption, spatial and temporal drift occurs due to the difference between the encrypted and the unencrypted pixel values which are used for prediction. As illustrated in Fig. 2 (middle-left), this affects regions around the RoI (e.g., on the top-right part of the helmet in the bottom picture), disfiguring them to an extent that they cannot be used any more for video surveillance and similar applications.

Thus, it is necessary to minimize drift. One way to do so is full re-encoding, offering perfect compensation (see Fig. 2, middle-right) at infeasible computational complexity. Therefore, we subsequently propose an approach which aims at minimizing drift at moderate computational complexity.

3. DRIFT MINIMIZATION

When encrypting macroblocks in the RoI, we have to take into account the dependencies that arise due to spatial and temporal prediction. Simply modifying the residual coefficients in a macroblock will affect the surrounding macroblocks due to (spatial) intra prediction, and due to (temporal) motion-compensated prediction.

When fully re-encoding the sequence, the original sequence is first decoded, and subsequently re-encoded using the same mode and motion information as in the input bitstream. In this case, the RoI encryption is embedded after the second (encoding) loop. The complexity of such an approach, however, will be high given the two prediction loops. We use this approach as a benchmark.

In the past, reduced-complexity transcoding approaches have been presented that were based on single-loop or open-loop architectures, mainly in the context of transrating [17]. Open-loop techniques can be used to perform modifications to coefficients with minimal complexity, but are unable to compensate for potential error drift. A *single-loop* approach, however, can compensate for the drift by storing the differences between the reconstructed coefficients before and after encryption. These differences can afterwards be used for intra prediction (IP) and/or motion-compensated prediction (MCP) in dependent blocks.

In this paper, we introduce such a compensation loop in the encryption framework, resulting in a significant reduction of the drift in the macroblocks which are dependent on the RoI area. In our framework, we make a distinction between three types of macroblocks:

- Macroblocks that are not inside the RoI or dependent on the RoI can be processed open-loop (i.e., by performing an entropy decoding and encoding step only), without further calculations.

- For the RoI macroblocks, the encryption approach introduced in Section 2 is performed. Additionally, the differences between the (inverse quantized and transformed) coefficients before and after encryption are calculated and accumulated, as illustrated in Fig. 3. The accumulation is performed along the direction of the intra prediction, or along the motion-compensated prediction direction. The resulting accumulated values are only stored within the RoI, but are not yet used for compensation. Hence, the coefficients for the RoI blocks are only modified by the encryption process, with no impact on the bit rate.

- For the macroblocks that are directly dependent on the RoI (either through intra prediction or motion-compensated prediction), single-loop compensation is applied and the differences that were accumulated in the RoI are used to compensate for the error drift. This

Fig. 2. First (top) and eleventh (bottom) frame of the *foreman* sequence (left-most) with QP 27 and $IP*$ GOP structure. The face as RoI is encrypted without drift compensation (middle-left), full compensation (middle-right) and our approach (right-most), respectively. Our approach shows only very little drift outside the RoI and is comparable to full compensation (middle-right).

will lead to a small increase in the bits required to code these blocks, as discussed in Section 4.3. The single-loop compensation loop is illustrated in Fig. 4. By applying compensation, we notice that a significant portion of the drift is eliminated for the non-RoI area.

An overview of the error accumulation and compensation process for the second and third types of macroblocks is given in Fig. 5, for the case of spatial error propagation (intra prediction). The non-colored macroblocks correspond to the first type of macroblocks and can be processed open-loop.

It has to be noted that the drift compensation will not be perfect, due to non-linear operations in the H.264/AVC/AVC encoder and decoder loops. For example, the division/shift operations during intra prediction (modes 3-8) and motion compensation, along with clipping at the boundaries of the pixel value range (for 8 bits, between 0 and 255) can lead to rounding errors in the compensation loop. As such, we can still encounter drift errors (albeit to a smaller extent) outside the encrypted RoI.

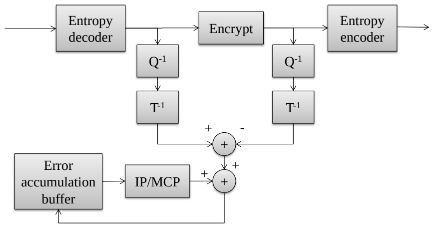


Fig. 3. Schematic transcoder architecture for RoI macroblocks (error accumulation): Differences between the encrypted and unencrypted values are accumulated in an error buffer. $Q^{(-1)}$: (inv.) quantization, $T^{(-1)}$: (inv.) transform, IP: intra prediction, MCP: motion-compensated prediction.

4. EVALUATION

We implemented our proposed approach as described in sections 2-3, and a full re-encoding variant, i.e., a method which fully decodes the bit stream to the pixel domain, fully compensates the drift therein and re-encodes the pictures with the same parameters, for comparison. In this section, we evaluate our approach in terms of execution time, drift and overhead.

For the evaluation, we use the *foreman* and *akiyo* sequences with one RoI each as simple test cases as well as the *crew* sequence with up to eleven RoIs as an advanced test case. All three sequences have CIF resolution $(352 \cdot 288)$. In addition, we use the *Vidyo1* sequence $(1280 \cdot 720)$ with three RoIs and the *Kimono* sequence $(1920 \cdot 1080)$ with one RoI to show the impact of higher resolutions. The RoI are the faces of the depicted actors which were segmented manually. As explained above, this information can be easily augmented to be provided by the surveillance system, together with the bit streams.

We used the H.264/AVC reference software (JM) to create Baseline profile bit streams of the sequences listed above. We used default settings and an $IPPP$ GOP structure, i.e., one I frame followed by 3 P frames, repeatedly, to simulate a typical video surveillance camera configuration.

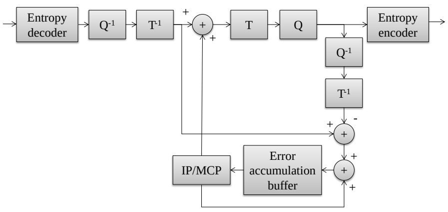


Fig. 4. Schematic transcoder architecture for macroblocks depending on the RoI (error compensation): The accumulated error is corrected approximatively by being considered in the prediction process. The abbreviations are the same as for Fig. 3.

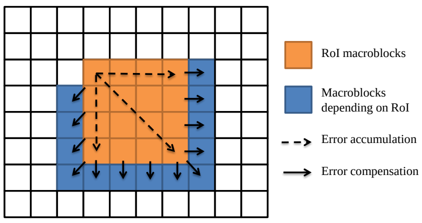


Fig. 5. Overview of error accumulation and compensation for intra prediction. Light (orange) blocks are processed as shown in Fig. 3, dark (blue) blocks as shown in Fig. 4.

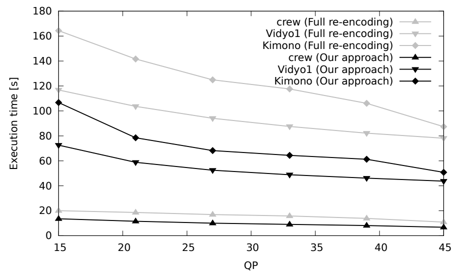


Fig. 6. Execution time of our approach (black) compared to full re-encoding (grey) for different QP: Our approach is between 30 and 45% faster.

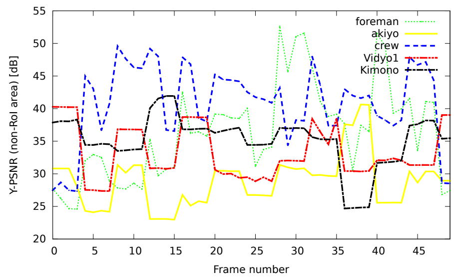


Fig. 7. Per-frame Y-PSNR outside the RoI for different sequences with QP 27 (first 50 frames): Spatial resolution has little to no effect on the quality degradation.

4.1. Execution Time

To measure the transcoding time, i.e., the time for encryption and drift minimization, we executed our software implementation three times for cache warming and five times for measuring the time between the entry and exit of the `main` function. The five measurements were averaged; fluctuations were insignificant at around 1%.

Fig. 6 depicts the transcoding time for one sequence per tested resolution and various QP. Our approach (black) is significantly faster than the implemented full re-encoding implementation (grey), saving between 30 and 45% of the execution time, the upper bound being achieved at high resolutions and QP. Transcoding time decreases with increasing QP in both implementations evenly.

4.2. Drift

To quantify the quality decrease due to drift, we measured the Y-PSNR of all pixels outside the RoI with the respective unencrypted (compressed) sequence as reference. We only depict the value of the first 50 frames for the sake of visualization. Fig. 7 illustrates the Y-PSNR values for all frames of the tested sequences with QP 27.

There are quasi no differences between the CIF and high resolution sequences. The amount of drift largely depends on the amount of movement and changes at GOP borders, most notably in the first few GOPs of the *Vidyo1* sequence (dash-dot-dotted line).

Used as a worst-case scenario, the *crew* sequence (dashed) with up to eleven RoI performs better than most of the other sequences in the first 50 frames, but shows some drops in quality around frame 109 (not depicted). Note, however, that the perceived quality is still relatively very high, i.e., the amount of drift is low and spatially limited at the minimum PSNR value at frame 109, as illustrated in Fig. 8 (bottom right). For the average case like in frame 45 (Fig. 8, top right), there is quasi no drift at all.

Fig. 8. Left: Frames 46 (top) and 109 (bottom) of the *crew* sequence with QP 27; right: Encoded with our proposed approach with examples of average drift (28 dB, top) and worst-case drift (14 dB, bottom), respectively, for a high number of encrypted RoIs.

4.3. Overhead

Finally, we determine the increase in file size caused by our approach due to the drift minimization. Fig. 9 illustrates this increase with respect to the original, i.e., unencrypted compressed, file size.

The bit rate increase depends on the QP. High and low QP induce little increase or even decrease, while medium QP increase the bit rate by about 1-3% for the CIF resolution sequences and significantly less for the high resolution sequences. The exact increase depends on the sequence and the number of RoIs. In general, increasing the resolution decreases the overhead.

The number of additional bits required for drift compensation decreases with increasing QP since high QP induce large quality degradations regardless of the presence of drift. Similarly, low QP yield a high number of non-zero coefficients in the transformed intra and inter prediction residuals, making the amount of bits required for drift minimization relatively small in comparison. For medium QP, the number of additional bits required for drift minimization is about the same, but the number of non-zero coefficients is smaller, therefore leading to a relative increase in bit rate.

Note that the bit rate increase of the *akiyo* sequence can be considered a worst-case scenario since there is practically no movement in the sequence outside the RoI. This allows coding this area with a small amount of bits, making every change due to drift minimization relatively large in comparison.

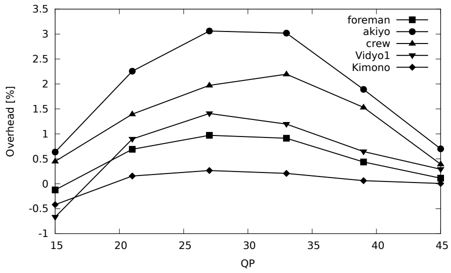


Fig. 9. Bit rate increase for different sequences and QPs: The overhead increases with the number of RoIs and peaks at medium-quality QPs.

5. FUTURE WORK

Two main aspects remain future work. First, the approach proposed in this paper can be combined with the approach of Unterweger et al. [6] which eliminates spatial, but not temporal drift. Since our approach minimizes temporal drift, a combination of the two approaches would allow for nearly drift-free bit-stream-based RoI encryption.

Second, decoding the sequences encoded with our proposed approach restores the RoI, but introduces additional drift outside the RoI due to the mismatch between the original prediction values and our drift-compensated ones. Although it is possible to copy the non-RoI areas (which are unencrypted) from the encrypted video, this does not allow for perfect reconstruction due to the remaining small amount of drift. Thus, a method to signal the RoI (as proposed, e.g., in [18]) as well as to compress and signal the difference signal between the original non-RoI areas and their counterparts with drift has to be devised so that the original video can be fully restored. Note that this may not be necessary for many use cases since, typically, only the RoI need to be fully restored, which is already the case with our approach.

6. CONCLUSION

We proposed a region of interest encryption approach for H.264/AVC bit streams. Despite being significantly faster than full re-encoding, it keeps the amount of drift outside the regions of interest at acceptable levels. The remaining amount of drift in all of the tested sequences is relatively small, apart from the *crew* sequence which exhibits some spatially limited drift in a small number of frames due to the high number of small regions of interest and intra-prediction-related dependencies. The bit rate overhead of our proposed approach is small (1.5% for high resolution sequences and 3% for low resolution sequences, tops) and depends on the quality and motion characteristics of the sequence to be encrypted. Our proposed approach is therefore a viable alternative to full re-encoding without the drawback of high computational complexity.

Acknowledgments

This work is supported by FFG Bridge project 832082.

7. REFERENCES

[1] E.M. Newton, L. Sweeney, and B. Malin, "Preserving privacy by de-identifying face images," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 232–243, Feb 2005.

[2] Frederic Dufaux and Touradj Ebrahimi, "A framework for the validation of privacy protection solutions in video surveillance," in *Proceedings of the IEEE International Conference on Multimedia & Expo, ICME '10*, Singapore, July 2010, pp. 66–71, IEEE.

[3] Pavel Korshunov and Touradj Ebrahimi, "Towards Optimal Distortion-Based Visual Privacy Filters," in *21st IEEE International Conference on Image Processing (ICIP 2014)*, Paris, France, Oct. 2014, IEEE.

[4] ITU-T T.81, "Digital compression and coding of continuous-tone still images — requirements and guidelines," Sept. 1992, Also published as ISO/IEC IS 10918-1.

[5] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[6] Andreas Unterweger and Andreas Uhl, "Slice groups for post-compression region of interest encryption in H.264/AVC and its scalable extension," *Signal Processing: Image Communication*, 2014, accepted.

[7] T. E. Boult, "PICO: Privacy through invertible cryptographic obscuration," in *IEEE/NFS Workshop on Computer Vision for Interactive and Intelligent Environments*, Lexington, KY, USA, Nov. 2005, pp. 27–38.

[8] Paula Carrillo, Hari Kalva, and Spyros Magliveras, "Compression Independent Reversible Encryption for Privacy in Video Surveillance," *EURASIP Journal on Information Security*, vol. 2009, pp. 1–13, Jan. 2009.

[9] Qi Meibing, Chen Xiaorui, Jiang Jianguo, and Zhan Shu, "Face Protection of H.264 Video Based on Detecting and Tracking," in *8th International Conference on Electronic Measurement and Instruments 2007 (ICEMI'07)*, Xi'an, China, Aug. 2007, pp. 2–172–2–177.

[10] Yeongyun Kim, Sung Jin, and Yong Ro, "Scalable Security and Conditional Access Control for Multiple Regions of Interest in Scalable Video Coding," in *International Workshop on Digital Watermarking 2007 (IWDW 2007)*, Yun Shi, Hyoung-Joong Kim, and Stefan Katzenbeisser, Eds., vol. 5041, pp. 71–86. Springer Berlin / Heidelberg, 2008.

[11] Lingling Tong, Feng Dai, Yongdong Zhang, and Jintao Li, "Restricted H.264/AVC video coding for privacy region scrambling," in *2010 17th IEEE International Conference on Image Processing (ICIP)*, Sept. 2010, pp. 2089–2092.

[12] Xiam Niu, Chongqing Zhou, Jianghua Ding, and Bian Yang, "JPEG Encryption with File Size Preservation," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing 2008 (IIHMSP '08)*, Aug. 2008, pp. 308–311.

[13] Bian Yang, Chong-Qing Zhou, C. Busch, and Xia-Mu Niu, "Transparent and perceptually enhanced JPEG image encryption," in *16th International Conference on Digital Signal Processing*, July 2009, pp. 1–6.

[14] Stefan Auer, Alexander Bliem, Dominik Engel, Andreas Uhl, and Andreas Unterweger, "Bitstream-Based JPEG Encryption in Real-time," *International Journal of Digital Crime and Forensics*, vol. 5, no. 3, pp. 1–14, 2013.

[15] Frederic Dufaux and Touradj Ebrahimi, "Scrambling for privacy protection in video surveillance systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1168–1174, 2008.

[16] R. Iqbal, S. Shahabuddin, and S. Shirmohammadi, "Compressed-domain spatial adaptation resilient perceptual encryption of live H.264 video," in *10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA 2010)*, 2010, pp. 472–475.

[17] Jan De Cock, Stijn Notebaert, Peter Lambert, and Rik Van de Walle, "Requantization transcoding for H.264/AVC video coding," *Signal Processing: Image Communication*, vol. 25, no. 4, pp. 235–254, 2010.

[18] Dominik Engel, Andreas Uhl, and Andreas Unterweger, "Region of Interest Signalling for Encrypted JPEG Images," in *IH&MMSec'13: Proceedings of the 1st ACM Workshop on Information Hiding and Multimedia Security*. June 2013, pp. 165–174, ACM.

Slice Groups for Post-Compression Region of Interest Encryption in H.264/AVC and Its Scalable Extension

Andreas Unterweger, Andreas Uhl

University of Salzburg
Department of Computer Sciences
Jakob-Haringer-Straße 2
Salzburg, Austria

Abstract

Encrypting regions of interest in H.264/AVC and SVC bit streams after compression is a challenging task due to drift. In this paper, we assess whether the use of slice groups makes this task easier and what its expense in terms of bit rate overhead is. We introduce the concept of all-grey base layers for SVC which simplify the encryption of regions of interest in surveillance camera applications while obeying all standard-imposed base layer restrictions. Furthermore, we show that the use of slice groups is possible with relatively low overhead for medium and high bit rates (below 5% in most of the tested configurations). This applies to H.264/AVC as well as SVC bit streams with two and three spatial layers, including those with the newly introduced all-grey base layers. Although we are able to contain spatial and inter-layer drift with our proposed encryption setup, temporal drift still remains an issue that cannot be solved by sole usage of slice groups.

Keywords: H.264/AVC, SVC, Slice groups, Selective encryption, Region of Interest, Overhead

1. Introduction

In video surveillance and other applications, there is often the need to disguise people's identities in order to protect their privacy. A common approach to achieve this is the selective encryption of people's faces (also called region-based selective [1] or Region of Interest (RoI) encryption), i.e. encrypting all picture areas which contain a face, while leaving all other picture areas untouched.
This allows for reversible de-identification, i.e., the disguise of identities with the possibility to restore them by undoing the encryption. Restoring is typically only possible with a correct key which is possessed, e.g., by law enforcement authorities in case suspects of a crime need to be identified. Although several techniques for reversible de-identification exist, RoI encryption is one of the most common ones in video surveillance.
While RoI encryption can be applied before (e.g., [2, 3, 4]), during (e.g., [5, 6, 7]) or after compression (e.g., [8, 9, 10]), each with its own advantages and disadvantages [11], most approaches proposed so far focus either on encryption before or during compression. Although this makes drift, i.e., the propagation of parts of encrypted picture areas into non-encrypted ones through spatial and temporal prediction, easier to manage, it does not allow using existing surveillance infrastructure whose input images and/or encoder cannot be modified.
Typically, surveillance cameras have compression hardware built in (as of 2014, Motion JPEG and H.264/AVC are very common) which reduces the bandwidth of the captured and transmitted video footage. Although this saves time and computational resources by not requiring additional encoding hardware, it makes modifications (like additional encryption) to the built-in compression hardware nearly impossible due to the often hard-wired encoder.
In order to be able to reuse this infrastructure notwithstanding, applying RoI encryption after compression has to be considered, reviving the drift issue. Therefore, in this paper, we try to assess the fitness of the slice group coding tool of H.264/AVC [12] and its scalable extension [13], also known as Scalable Video Coding (SVC), to allow selective encryption of picture areas and to contain drift.
For the sake of applicability, we consider a state-of-the-art video surveillance system which delivers H.264/AVC-compressed output. We assume that the surveillance system detects faces (or other regions of interest) using a built-in face detector. This is common in most state-of-the-art surveillance systems. Since the coordinates of the detected faces are available this way, we further assume that the surveillance camera places the detected faces in slice groups. The definition of slice group borders based on the detected RoI does not require any special additional coding standards to be implemented since it is supported

Email addresses: aunterweg@cosy.sbg.ac.at (Andreas Unterweger), uhl@cosy.sbg.ac.at (Andreas Uhl)

Preprint submitted to Signal Processing: Image Communication *September 17, 2014*

by both, H.264/AVC and SVC. Even if face detection functionality is not yet in place in a surveillance system, it can simply be added without requiring major system modifications, e.g., by using an additional black box implementation of a face detector, which does not affect the rest of system.
The main reason for using slice groups is their ability to contain drift to a certain extent, thereby simplifying RoI encryption. Note that slice groups have other uses as well, thereby extending the results of our investigations to scenarios which are not encryption-specific.
By evaluating the limitations and possibilites of slice group coding, we aim at determining whether or not the aforementioned setup simplifies the encryption process in terms of drift. Furthermore, we evaluate the overhead induced by this approach in order to determine whether or not it is of practical use, i.e., for example to be included into existing and/or future surveillance systems to simplify RoI encryption after compression.
So far, no practical post-compression RoI encryption approaches have been proposed for H.264/AVC. [14], which uses one RoI per slice to contain spatial drift, encrypts at bit stream level, but does not evaluate the slice-induced overhead. In addition, only "regular" slice shapes (without Flexible Macroblock Ordering (FMO), i.e., in macroblock scan order from top-left to bottom-right) have been evaluated. This is not adequate for the use case considered in our paper, which requires rectangular RoI.
Although [10] describes a simple encryption approach for MPEG-4 Part 2 which could be extended to H.264/AVC, it is of no practical use since it reencodes the bit stream using intra blocks to avoid drift, which makes it actually an in-compression encryption approach. Although this may be suitable in terms of (transcoding) complexity for the video surveillance use case, the overhead is too large. As reported by [5] who applied the scheme described in [10] to H.264/AVC, overheads exceed 100% is some cases, depending on the complexity of the transcoding operation. Full transcoding with prediction restriction decreases the overhead to about 1-6% [15, 5], but is undesirable due to its high complexity.
Related work on RoI encryption in SVC is sparse. Two approaches are proposed in [16] and [17], albeit without considering or compensating for the effects of drift, which is an important matter. [7] deals with drift by imposing restrictions on the encoding process in terms of a limited motion estimation range as well as interpolation and upsampling constraints. Besides the reported significant increase in bit rate, this method cannot be applied on a bit stream level without recompression. Similarly, [18] proposes separate RoI coding by restricting motion estimation and inter-layer prediction, albeit without the explicit intention to do so for the sake of encryption. However, all of these approaches are in-compression encryption methods and cannot be applied at bit stream level.
Apart from RoI-related experiments and analyses of SVC, the encryption of certain Network Abstraction Layer (NAL) units has been proposed in [19]. However, their proposed encryption approach yields bit streams which are no longer format compliant and can hence not be decoded anymore by a regular decoder. This is not desirable in surveillance applications as the background without the encrypted RoI should be visible and therefore decodable.
Furthermore, the extraction and quality optimization of RoI across multiple layers to lower the total bit rate has been analyzed in [20]. However, the paper mainly focusses on cropping RoI through slice data removal and modification. It is not at all encryption-related and does therefore not take drift into account.
Although slice groups have been used to deal with drift in a number of encryption approaches (e.g., [5, 15]), a detailed examination of its actual usefulness to contain different causes of drift has not been done so far. The overhead induced by some of the aforementioned encryption approaches has been analyzed, but this is not true for the general overhead introduced by slices groups which change from frame to frame to cover RoI. This is especially true for SVC.
A number of analyses on slice groups for H.264/AVC, including overhead measurements for moving RoI, have been performed in [21]. However, they do not actually encode the RoI completely independently, as opposed to our implementation. Thus, their implementation provides an approximation, but no exact slice-group-related results, which are presented in this paper.
This paper is structured as follows: In section 2, the key concepts of video coding with slice groups in H.264/AVC and SVC are described, followed by an analysis of their limitations in section 3. After evaluating several scenarios in terms of feasibility for video surveillance with encrypted RoI in section 4, we conclude our paper.
This paper extends our previous work [22] by slice group overhead results for (non-scalable) H.264/AVC bit streams as well as a dissection of the overhead components. Furthermore, a detailled analysis of drift for both, H.264/AVC and SVC is provided and an additional post-compression approach is proposed and evaluated to circumvent standard-imposed restrictions. In addition, more sequences of actual surveillance footage are used.

2. H.264/AVC and SVC

The H.264 video coding standard allows for efficient compression of moving pictures by exploiting spatial and temporal redundancy. As a detailed description of H.264/AVC's features (as presented in [23]) is not within the scope of this paper, only the coding tools required herein are explained briefly.
In H.264/AVC-compliant bit streams, each coded picture is split into one or more slices, each of which consists of macroblocks of $16 \cdot 16$ luma samples and the corresponding chroma samples. Slices can be summarized to slice groups of specific forms (this is also known as FMO), depending on the so-called slice group map type. As RoI encryption

requires a background left-over, i.e., a region of the picture which does not belong to any encrypted region of interest, only slice group map types 2 (foreground slice groups with left-over background) and 6 (explicit slice group specification) will be considered, as only they allow this. Since slice group map type 6 is practically identical to slice group map type 2 in this use case, we will only consider slice group map type 2 henceforth.

To exploit spatial and temporal redundancy, H.264/AVC allows predicting samples of macroblocks from blocks around the one to be predicted in the same picture as well as from arbitrary blocks in previously coded pictures. In the former case, predictions over slice borders are forbidden, thereby allowing all slices to be decoded independently.

The scalable extension of H.264/AVC specified in its Annex G, also referred to as SVC, allows for multiple so-called layers within one bit stream, which can be accessed or extracted depending on the capabilities of the device decoding the stream. Each layer differs from the others either by frame rate (temporal scalability), resolution (spatial scalability) or quality (Signal-to-Noise Ratio (SNR) scalability). The bottom-most layer is referred to as base layer and coded in a way that is compatible with (non-scalable) H.264/AVC.

All layers but the base layer can exploit inter-layer redundancies by using coded information of lower layers for prediction. The basis of this prediction for spatial and SNR scalability can either be filtered intra-coded samples (inter-layer intra prediction), motion vectors (inter-layer motion prediction) or inter-coded difference signal samples (inter-layer residual prediction), with details for each prediction type to be found in [24]. In contrast, temporal scalability is achieved through hierarchical inter prediction as explained in detail in [13].

Figure 1 shows an example of a scalable bit stream with multiple layers. The base layer (temporal layer 0 (T0), spatial layer 0 (S0) and SNR layer 0 (Q0)) has the lowest possible frame rate, resolution and quality and is used to predict the first spatial enhancement layer (T0, S1, Q0; not labeled) which doubles both, picture width and height. This enhancement layer is further used to predict an enhancement layer of the same resolution, but a doubled frame rate (T1, S1, Q0) as well as an enhancement layer with higher quality (T0, S1, Q1; not labeled) and subsequently a doubled frame rate (T1, S1, Q1).

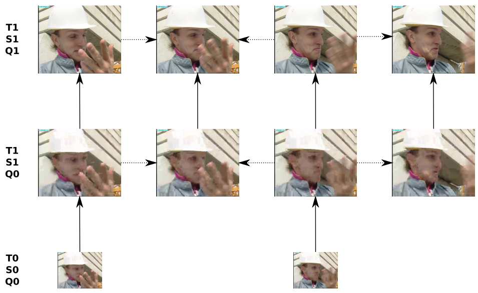


Figure 1: SVC with multiple layers: The base layer with half the frame rate and a quarter of the picture size can be used to predict the first spatial enhancement layer, which itself can be used to predict a second temporal and subsequently a third SNR enhancement layer. Adopted from [13]

3. Standard-imposed limitations

The H.264/AVC standard imposes restrictions on coding tools and parameter values by specifying profiles. As this paper discusses slice groups, we only consider profiles which allow the use of multiple slice groups in the first place. In this section, we investigate other relevant limitations imposed by those profiles.

For regular, i.e., non-scalable, H.264/AVC bit streams, only the Baseline and the Extended profile support slice groups. Although both allow using up to eight slice groups in total, one slice group is considered to be the background, i.e., the remainder of what the other seven slice groups encode.

Both, the Baseline and the Extended profile limit the available coding tools, most notably in that they only allow CAVLC entropy coding instead of CABAC. Moreover, the Baseline profile does not allow the use of B slices, i.e., only I and P slices can be used, as opposed to the Extended profile. Note that the lack of B slices is not a problem in surveillance scenarios where real-time transmission is expected, which would be delayed by the use of B frames [25]. The remaining profile constraints do not limit the use case described in this paper significantly and are therefore not described in detail.

For scalable bit streams, only the Scalable Baseline profile supports slice groups. Similar to the H.264/AVC Baseline profile, entropy coding is limited to CAVLC, the number of slice groups cannot exceed seven (plus background) and B slices are not allowed. Furthermore, the base layer may not contain more than one slice group.

This is a severe limitation in an encryption scenario because this means that the regions of interest cannot be in separate slice groups in the base layer. Thus, either a different drift compensation approach for the base layer is required or an alternative to slice groups in the base layer has to be found. As the former is hard to achieve, we consider three additional alternatives to slice groups in the base layer as depicted in Figure 2.

One possibility is to use extended spatial scalability, depicted on the left and in the middle of Figure 2, where the base layer only contains the region of interest and the enhancement layer adds the rest of the video frame. Due to the limitations of the Scalable Baseline profile, the width and height ratios between the base layer and the corresponding region of interest in the enhancement layer have to be either 1 (Figure 2, left), 1.5 (not depicted) or 2 (Fig-

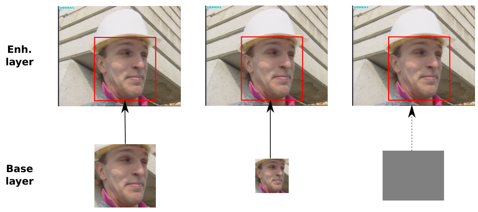


Figure 2: Alternatives to slice groups in the base layer: Left and middle: Extended spatial scalability; right: all-grey base layer

ure 2, middle).
However, this setup is only useful if there is exactly one region of interest. Since this would impose a severe practical limitation, it is not considered in the remainder of this paper. Alternatively, we propose adding a base layer which is all-grey ($Y = C_b = C_r = 128$) as shown in Figure 2, right. Since intra DC prediction and skip modes allow encoding such an artificial layer very compactly, its overhead is relatively small when using the maximum possible width and height ratios of 2, i.e., a base layer with half the width and height of the enhancement layer.
However, it effectively reduces the number of usable spatial layers, which is limited to three in the Scalable Baseline profile, by one. This allows for a maximum of two non-grey spatial layers for actual video content. Depending on the use case, these two remaining layers may be sufficient to provide spatial scalability.
Since the standard-imposed restrictions prevent encryption methods from easily encrypting the base layer (there are no slice groups allowed in order to contain the drift), the base layer would have to be treated separately for encryption in all practical scenarios, entailing different restrictions and drawbacks. There are two possibilities to put the grey base layer in place: a true post-compression approach and a constrained post-compression approach.
In the true post-compression approach, the input bit stream has a regular base layer. During encryption, it is replaced by a grey base layer at bit stream level, as shown in figure 3. If no inter-layer prediction is used (this reduces rate-distortion performance by about 1-2 dB [26]), no reencoding is necessary. The original base layer is irrecoverably lost in this case, which in-turn is expected to increase the rate-distortion performance. This allows for post-compression encryption at the cost of losing the original base layer.
Conversely, in the constrained post-compression approach, the grey base layer has to be put in place by the encoder. This can simply be done by using an all-grey image instead of a downscaled version of the corresponding high resolution image. It constrains the supported bit streams since a grey base layer is already required to be in the input file. This only allows for post-compression encryption if the encoder hardware can be configured to support a

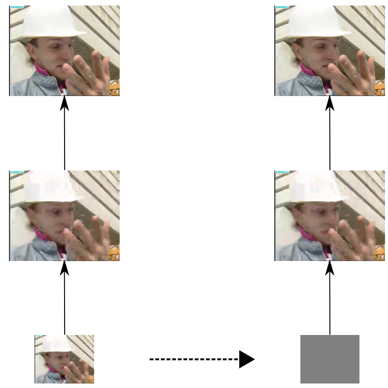


Figure 3: True post-compression encryption: An existing base layer is replaced by an all-grey base layer to circumvent standard-imposed restrictions for slice groups in the base layer

grey base layer. We consider both, the constrained and the true post-compression approaches in this paper and analyze their differences in detail in section 4.
Although there have been multiple proposals for region-of-interest support through slice groups in all layers [27, 28], the final version of the standard does not allow this. Similarly, the technique proposed in [29] to alternatively support regions of interests as enhancement layers is not supported. This paper limits the available options to the ones supported by the standard, i.e., the all-grey base layer introduced above as well a regular (i.e., full-content) base layer for comparison.
When the base layer is encrypted completely, for example, it is not usable by a decoder which only extracts and displays the base layer, yet a standard decoder would not be aware of this when receiving the bit stream. When using a grey base layer, as proposed, the situation is similar: A standard decoder only shows a grey picture, which is still format compliant. However, the overhead of using a grey base layer is expected to be significantly lower as compared to a completely encrypted base layer, which requires a separate encryption approach and additional drift prevention mechanisms in order to avoid inter-layer drift.
Despite the loss of one usable spatial layer, the grey base layer simplifies encryption by containing drift. Although the unavailability of slice groups in the base layer (see above) would normally make encryption harder (without the possibility of using slice groups to contain drift), the fact that the base layer is all grey does not require any encryption and does therefore not induce any drift.
Regarding further limitations imposed by the standard, we will focus on the combination of constrained intra prediction and constrained inter-layer prediction, which ensure single-loop decoding [30]. Since these two limitations severely limit the number of possibilities for prediction and

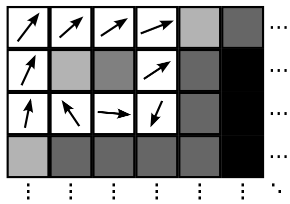


Figure 4: Constrained intra prediction: In a P slice, intra blocks may not use inter blocks for prediction. The grey level of the depicted intra blocks denotes the number of allowed intra modes

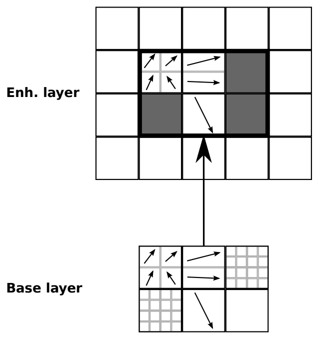


Figure 5: Constrained inter-layer prediction: Upsampled intra blocks (grey) must be reconstructed from base layer intra samples

therefore drift, they are crucial for the RoI encryption use case.
Constrained intra prediction limits the blocks which can be used for intra prediction. Figure 4 illustrates this in a P slice which contains inter (depicted by motion vectors) and intra (depicted by grey levels) macroblocks. Although the black intra blocks may use all possible intra prediction modes, the dark- and light-grey ones may not. For example, the light-grey macroblock at the top left may only use DC prediction since all other prediction directions would require predicting from one of the surrounding inter macroblocks. Note that constrained intra prediction reduces coding efficiency, especially for isolated intra macroblocks, i.e., intra macroblocks surrounded by inter macroblocks.
SVC enforces constrained intra prediction in all layers which are used for inter-layer prediction so that inter-layer predicted samples do not require additional motion compensation in the base layer. Additionally, constrained inter-layer prediction ensures that inter-layer-predicted intra samples are not used for intra prediction themselves, as illustrated in Figure 5.
Inter-layer prediction allows using information from the base layer in the enhancement layer. If blocks are upsampled through inter-layer intra prediction (grey blocks in Figure 5), the corresponding reference block in the base

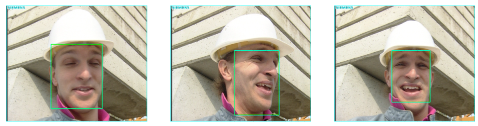


Figure 6: Moving slice groups: Frames 1, 11 and 21 of the *foreman* sequence with one moving foreground slice group around the face (green) and one background slice group (remainder, turquois)

Figure 7: Encrypted RoI: Frames 1, 11 and 21 of the *foreman* sequence. The RoI in this example is the actor's face. Note that the noise is symbolic to illustrate the combination of moving RoI and an arbitrary form of encryption

layer has to be an intra block as well. Constrained intra prediction in the base layer ensures that no additional motion compensation loop is required. Furthermore, if the enhancement layer is used for further inter layer prediction, the upsampled blocks may not be used for further intra prediction due to the constrained intra prediction requirement to avoid multi-loop decoding.
Note that constrained inter prediction [31] has also been proposed, but not incorporated into the final video coding standards. With constrained inter prediction, inter macroblocks must not depend on intra macroblocks from the same slice. This allows minimizing the dependencies between intra and inter data partitions when data partitioning is used. This is useful when the intra data (partition) is lost – the inter data can still be used.

4. Experimental evaluation

In this section, we describe our experimental setup and results. We refer to the term of "moving slice groups" for RoI herein since the position of RoI may change from frame to frame, thereby changing the slice group positions accordingly, as illustrated in Figure 6.
Recall that our use case is encryption, i.e., we assume that the moving slice groups will be encrypted at some point, as illustrated by example in figure 7. Note, however, that we do not propose a specific encryption algorithm – our results are independent of the employed encryption approach as long as the latter is format compliant. The noise in figure 7 is therefore only symbolic.

4.1. Setup

In order to evaluate the effect of slice-group-based RoI for encryption, we added support for moving slice groups to both, the H.264/AVC (*JM*) and SVC (*JSVM*) reference software, since they do not support this by

themselves.
Although the *JM* supports slice group coding in principle, it only does so with one set of coordinates for all frames. Therefore, in our modification, before encoding each frame, the corresponding RoI coordinates are loaded and all data structures containing the slice group information are adapted accordingly. Since the slice groups' position and size are signaled by a Picture Parameter Set (PPS) preceding the corresponding picture, the *ResendPPS* parameter is enabled so that one PPS is inserted before each frame. Note that the PPS data structure needs to be modified as well, albeit before the PPS is written to the output.
In the *JSVM*, slice group coding is implemented partially, but not used. Therefore, it is enabled separately for all spatial layers but the base layer which does not support slice group coding (see section 3). In addition, in each layer, the RoI coordinates are calculated depending on the picture size and the corresponding slice group settings are adapted accordingly. In order to signal the slice groups, one additional PPS per frame and enhancement layer is required. In contrast to the *JM* with its *ResendPPS* parameter, this requires inserting one PPS per frame per enhancement layer by modifying the source code accordingly.
We use a total of six test sequences depicted in figure 8: three common test video sequences (*akiyo*, *foreman* and *crew*, each 300 frames long and in Common Intermediate Format (CIF) resolution) as well as three surveillance video sequences where the camera that captured them is static and people move by (*hall* with 300 frames in CIF resolution, *ice* with 240 frames in 4CIF resolution and *visor_1246522137645_new_4_camera2* (abbreviated *visor* henceforth) from the VISOR data set (`http://www.openvisor.org/video_details.asp?idvideo=323`) with 1019 frames in Quarter Video Graphics Array (QVGA) resolution). All video sequences have 30 frames per second, except the *visor* sequence, which has only 10 frames per second. In addition, the *visor* sequence was converted from the Red Green Blue (RGB) to the YCbCr color space with 4:2:0 subsampling using *ffmpeg*.
The three common video sequences differ in terms of face count and motion, representing both, typical and extreme cases for evaluation. *akiyo* has one RoI and very little motion, while *foreman* has a significant amount of motion. Both have only one RoI. Conversely, the *crew* sequence has a significant amount of motion and a changing number (between 2 and 11) of RoI.
The three surveillance video sequences have no global motion, as mentioned above. *hall* has little local motion and between no and 2 RoI. Conversely, *ice* has a significant amount of motion and between 2 and 7 RoI. In contrast, *visor* has jerky motion due to the low frame rate and no RoI most of the time. The short time intervals in which there are RoI visible, there are between 1 and 7 RoI.
All faces were segmented manually by enclosing them in rectangles. The corresponding coordinates were

Figure 8: Video sequences used for testing (from top-left to bottom-right): Frame 100 of *foreman*, *akiyo*, *crew*, *hall*, *ice* and *visor*

rounded to the nearest macroblock border. Since a maximum of seven slice groups (RoI) are supported in both, H.264/AVC and SVC (see section 3), only the first top-left-most faces are considered, i.e., placed in a separate slice group. This only affects the *crew* sequence, which has more than seven RoI, but does not impact our results. Since we do not actually encrypt the RoI, but only assess the overhead induced by slice groups, the smaller RoI will give an upper bound of the overhead for actual implementations which will likely combine some of the RoI to reduce the number of slice groups to seven.

4.2. Overhead (H.264/AVC)

In the case of H.264/AVC, we distinguish various typical Group Of Pictures (GOP) structures: $I*$ (i.e., only I frames), $I(PPP)*$ (i.e., one I frame, followed by groups of three P frames), $I(bP)*$ (i.e., one I frame, followed by groups with one non-reference B and one P frame each) and $I(BBBBBBBP)*$ (i.e., one I frame, followed by groups of seven B frames and one P frame each, where the B frames are coded hierarchically). Note that GOP structures with B frames require the use of the Extended profile (see section 3).
We encode the test sequences with a constant Quantization

6

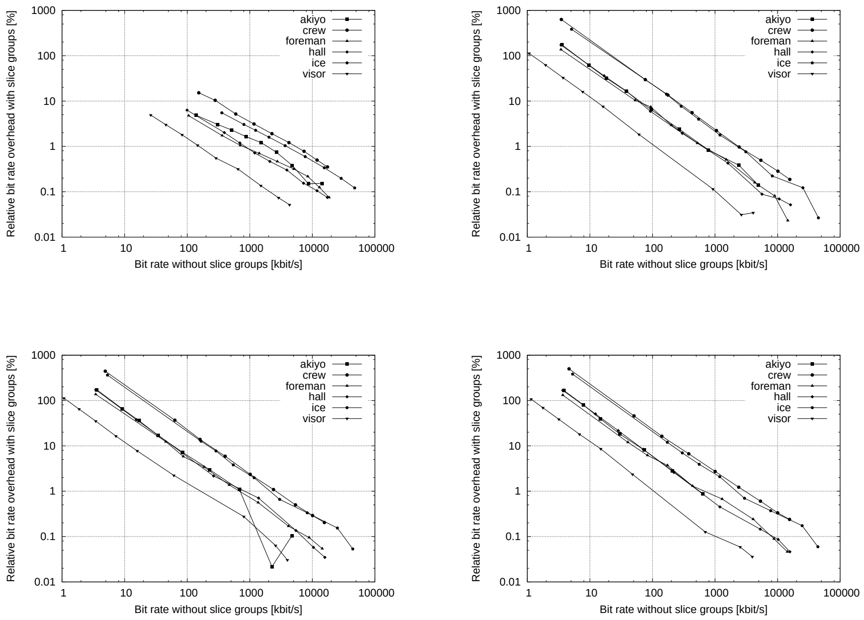


Figure 9: Overhead with slice group coding for different GOP structures: $I*$ (top-left), $I(PPP)*$ (top-right), $I(bP)*$ (bottom-left), $I(BBBBBBBP)*$ (bottom-right)

Parameter (QP) for all frame types and default settings. Using QPs between 3 and 51 with a step size of 6 to double the quantizer step size with each run allows covering the whole QP range (since the results of this paper may be useful for other applications as well, we did deliberately not restrict the QP range to typical surveillance video settings). Each QP-sequence combination is encoded with and without slice groups. Since the difference in terms of distortion between the encoded sequences with and without slice groups is very small (< 0.1 dB), we approximate the overhead introduced by slice group coding by comparing the corresponding bit rates directly.

Figure 9 shows the overhead for the different GOP structures and sequences. In order to make comparisons between the overheads of different GOP structures easier, figure 10 depicts the overhead of the *crew* sequence in detail for all tested GOP structures.

It is obvious that the *crew* and the *ice* sequence (depicted by circles and pentagons in figure 9, respectively) exhibit the highest overhead in nearly all scenarios, since they require the highest number of slice groups. Conversely, the *visor* sequence exhibits the lowest overhead, since it only

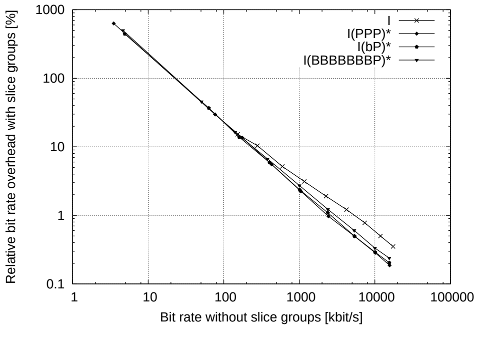


Figure 10: Slice group overhead comparison for the *crew* sequence with different GOP structures

7

requires slice groups for some short parts of the sequence and has a lower frame rate. The *akiyo*, *foreman* and *hall* sequences require about the same number of slice groups in total, so their overhead lies between the two corner cases with nearly none and nearly always the maximum number of slice groups.

Unsurprisingly, the $I*$ GOP structure (depicted by crosses in figure 10) exhibits the lowest overhead percentages. As it uses no inter prediction at all, the lowest possible bit rate is relatively high (see figure 9, top-left). Although this overhead difference compared to other GOP structures is notable for very high bit rates, it becomes insignificantly small for bit rates above 1000 kbit/s (see figure 10). The other GOP structures behave very similarly in terms of relative overhead, thus making the GOP structure choice practically irrelevant for low to medium bit rates. For high bit rates, it is irrelevant as long as the GOP does not consist of I frames only.

In general, the overhead for all GOP structures decreases with the bit rate, i.e., it increases with the QP. For low bit rates, slice group coding adds an unacceptable overhead of up to several hundred per cent. Conversely, for bit rates which are higher than 1000 kbit/s, all sequences but *crew* and *ice* exhibit a small overhead of approximately 1% or less.

The overhead of the *crew* sequence is approximately five times higher than the overhead of the other sequences over a large QP range, i.e., for nearly all bit rates. This is due to the use of the maximum number of slice groups in nearly all frames throughout the sequence and shows that the number of slice groups significantly influences the bit rate overhead.

For very high bit rates, i.e., very low QP, the overhead of the *akiyo* sequence with slice groups fluctuates, resulting in non-depicted data points for some bit rates due to the corresponding very small negative values which cannot be depicted using logarithmic axes. The fluctuations are due to the fact that *akiyo* requires a relatively low bit rate compared to the other sequences. Thus, in the high bit rate range, the slice group borders which prevent intra prediction only affect the number of quantized non-zero coefficients minimally, so that the overhead becomes very low. Depending on the actual coefficients, this impacts further intra prediction, making the very small overhead a nearly random value due to the high impact of the very small changes in the coefficients.

Note that this prediction-border-related overhead is only one part of the total overhead. The overheads depicted above can be split into two components: Firstly, there is a constant overhead for the additional PPS which are required to signal the position and size of the slice groups for each frame. Secondly, the additional prediction borders induced by the slice groups decrease coding efficiency, resulting in an overhead when using a constant QP.

Table 1 shows the first component and the absolute total overhead for the *crew* sequence for the $I(BBBBBBBP)*$ GOP structure (since the GOP structure does not impact

QP	File size diff.	Relative PPS size diff.
3	46249	16.53%
9	41751	18.31%
15	40170	19.03%
21	36613	20.88%
27	34221	22.34%
33	31716	24.11%
39	28773	26.58%
45	28995	26.37%
51	28980	26.39%

Table 1: Absolute overhead in bytes for the *crew* sequence with $I(BBBBBBBP)*$ GOP structure. The rightmost column denotes the relative amount of PPS bytes of the corresponding total absolute overhead

the overhead significantly, as shown above, this can be considered to be representative for this sequence). Without slice groups, there is only one PPS of 9 bytes required. Conversely, when slice groups are used, 7657 bytes are required for all 300 PPS – one per frame, with different sizes each, depending on the number of RoI.

Although the number of PPS bytes required for signalling remains constant ($7657 - 9 = 7648$ bytes), their relative amount increases with increasing QP. Most notably, for very low QP, i.e., very high quality, it only accounts for less than a fifth of the total absolute overhead. The remainder of the overhead is, as described above, due the second overhead component, i.e., the slice-group induced prediction borders. It can be seen that the PPS-related constant overhead does not exceed 27% of the total absolute overhead.

4.3. Overhead (SVC)

In the case of SVC, we use the GOP size of the default JSVM configuration, i.e., four. Since GOP structures with B frames are not allowed in combination with slice groups (see section 3), we use P frames instead. Thus, an $(IPPP)*$ GOP structure, i.e., a repeated sequence of one I frame and followed by three P frames, is used.

We encode the test sequences with a constant QP for both frame types and default settings with two and three dyadic spatial layers. The base layer is all grey (see section 3), although we test ”classical” base layers (with the actual down-sized input video) as well for comparison. Inter-layer prediction is set to adaptive to allow for optimal coding efficiency.

In this section, we consider the constrained post-compression approach, in which the grey base layer is already put into place by the encoder, as described in section 3. An analysis of the differences between this approach and the true post-compression approach is provided in section 4.4.

Note that we use 4CIF versions of *crew* and *foreman* for these measurements since CIF sequences with three spatial layers would yield impractically small base layers. Since we were unable to obtain a 4CIF version of *akiyo*, we omit-

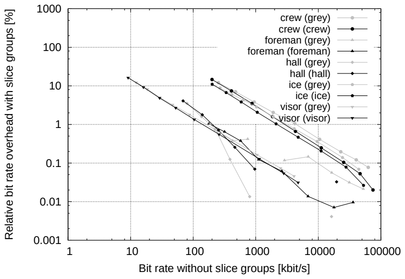


Figure 11: Overhead with slice group coding for different sequences when using two dyadic spatial layers. The names in parentheses denote the used base layer sequence.

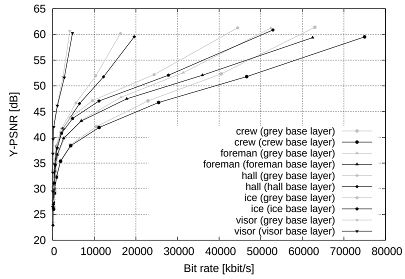


Figure 12: Rate-distortion plot for SVC with two dyadic spatial layers and slice groups. Different base layers (depicted in grey and black) result in significantly different enhancement layer Y-PSNR.

ted it from this test set. Note, however, that we kept the *visor* sequence in the test set due to its relevance as the only low-frame-rate surveillance video.

As in the H.264/AVC experiments (see section 4.2), each QP-sequence combination is encoded with and without slice groups. Again, the difference in terms of distortion between the encoded sequences with and without slice groups is very small (< 0.15 dB), so we approximate the overhead introduced by slice group coding by comparing the corresponding bit rates directly.

As depicted in Figure 11, in the case of two spatial layers, the overhead shows a similar dependency on the bit rate as in the H.264/AVC case (see section 4.2). While very low bit rates result in infeasibly large overhead, medium and high bit rates exhibit moderate to low overhead.

The *crew* and *ice* sequences exhibit the highest overhead when using slice groups due to the large number of RoI, as in the H.264/AVC case (see section 4.2). The *foreman* and *hall* sequences profit from scalability more than the other sequences, resulting in very small negative overhead values ($< 0.1\%$ absolute). Note that these values cannot be depicted properly due to the logarithmic Y axis.

Using an all-grey base layer does not affect the overhead significantly due to the use of slice groups. Compared to the classical base layer configuration, however, an all-grey base layer allows using slice-group-based encryption for SVC in the first place, since slice groups cannot be used in the base layer (see section 3).

Figure 12 shows a rate-distortion plot for the two-layer case with slice groups, where the Y-PSNR values are those of the enhancement layer. The plot allows comparing the all-grey base layer with a classical base layer. It is obvious that the all-grey base layer results in significantly better rate-distortion performance (up to 5 dB) for medium and high bit rates.

Since an all-grey base layer greatly improves rate-distortion performance avoiding the need for additional drift compensation due to encryption in the base layer, it can be considered a better solution than a classical base layer for this use case. As the overhead due to slice groups is similar in both, the all-grey and the classical base layer scenario (see above), this is also true for other potential use cases in which the base layer does not have to be the downsampled input sequence.

Note that an all-grey base layer in a scenario with two spatial layers defies the purpose of scalable video coding, since one of the two layers becomes unusable for content. However, it allows establishing a baseline for comparison in terms of overhead and allows assessing the usefulness of the concept. In order for all-grey base layers to be practically useful, a scenario with three spatial layers has to be considered so that two spatial layers remain for actual content.

When increasing the number of spatial layers to the maximum of three (see section 3), the overhead due to slice groups increases, as depicted in Figure 13. The overall overhead is significantly higher than in the two-layer case (see Figure 11) for low to medium bit rates. This is due to the fact that slice groups introduce prediction borders which reduce coding efficiency and the three-layer case (with two enhancement layers with slice groups) uses double the amount of slice groups than the two-layer case (with one enhancement layer with slice groups). However, for high bit rates, the overhead is still relatively small and therefore practically negligible for most use cases.

Compared to the two-layer case, the all-grey base layer configuration in the three-layer case allows for an overhead which is approximately as low as the overhead in the classical base layer configuration. Although the all-grey base layer configuration exhibits a higher overhead for medium-to-high bit rates, the actual overhead is only insignificantly higher.

However, in the three-layer case the rate-distortion performance improvement of the all-grey base layer is only very

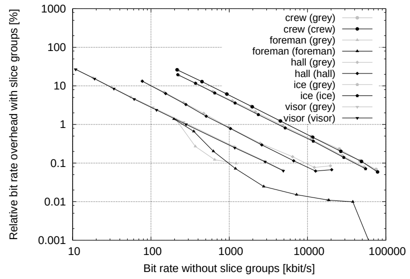


Figure 13: Overhead with slice group coding for different sequences when using three dyadic spatial layers. The names in parentheses denote the used base layer sequence.

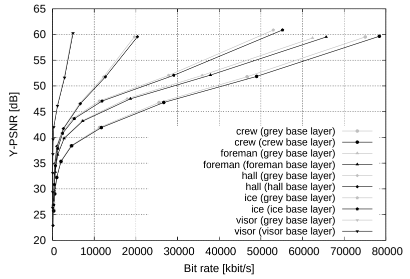


Figure 14: Rate-distortion plot for SVC with three dyadic spatial layers and slice groups. Different base layers (depicted in grey and black) result in similar enhancement layer Y-PSNR.

small, as depicted in Figure 14. Although there are still differences of up to 1 dB between an all-grey and a classical base layer in terms of enhancement layer Y-PSNR, the performance improvement is nowhere near the improvements of the two-layer case (see above).

This is mainly due to the fact that there are two enhancement layers, which use most of the bit rate and the fact that the first enhancement layer can be used to predict parts of the second one through inter-layer prediction. This makes the three-layer case with an all-grey base layer similar to a two-layer case with an additional all-grey bit stream, which is very likely not used at all for inter-layer prediction. However, an all-grey base layer still has advantages compared to a classical base layer for the use case in this paper, since base layer encryption cannot rely on slice groups due to base layer limitations (see above). Thus, an all-grey base layer is still to be preferred over a classical base layer in the three-layer case.

4.4. True post-compression approach performance

Since the previous section dealt with the performance of the constrained post-compression approach, this section aims at highlighting the performance differences of the true post-compression approach, in which a regular base layer is replaced by a grey base layer during encryption.

As described in detail in section 3, the true post-compression approach requires a "regular" base layer in an SVC bit stream which does not use inter-layer prediction. During encryption, the original base layer is removed (which can be done safely since no inter-layer-prediction-related dependencies can yield drift) and replaced by an all-grey base layer.

Both, the constraint of not allowing inter-layer prediction and the replacement of the base layer, change the rate-distortion performance significantly. Although the base layer constraint is known to result in a decrease of about 1-2 dB [26], the use of an all-grey base layer has been shown to increase rate-distortion performance significantly when using two spatial layers with inter-layer prediction in section 4.3.

Thus, it is necessary to evaluate the overall change in rate-distortion performance in this section. We do this by evaluating several differently coded versions of the *crew* sequence in 4CIF resolution with the same basic encoding parameters as in section 4.3.

Figure 15 shows the results for two dyadic spatial layers. The imposed constraint (no inter-layer prediction) on the base layer (dotted black line) decreases rate-distortion performance by about 1 dB, as expected, compared to SVC with inter-layer prediction (solid black line). However, the replacement of the base layer by an all-grey base layer (grey line) increases the performance significantly, yielding even higher Y-PSNR values than SVC with inter-layer prediction. The difference is small for low bit rates, but reaches up to 5 dB for very high bit rates.

Conversely, figure 16 shows the results for three dyadic spatial layers, where the differences between the different configurations practically vanish for most bit rates. Even though SVC with inter-layer prediction is slightly superior to the grey base layer without inter-layer prediction for the true post-compression approach, the difference is only about 0.5 dB.

Note that in both, figure 15 and 16, the performance of the true post-compression approach (grey line) is equal to the performance of the constrained post-compression approach described in section 4.3. In summary, both approaches outperform SVC with inter-layer prediction in terms of rate-distortion performance when using two spatial layers and are only marginally inferior when using three spatial layers. This makes them adequate alternatives which simplify encryption at the expense of one lost, i.e., grey, spatial layer. It also justifies the restriction to disallow inter-layer prediction in the base layer for the true post-compression approach.

Figure 18: Example for temporal drift: The first, second, third, fifth and tenth frame (from left to right) of the *foreman* sequence where one block in the first frame (left-most) has been modified (as in figure 17). The top row shows the original frames, whereas the bottom row shows the frames with temporal drift (second from the left to right-most).

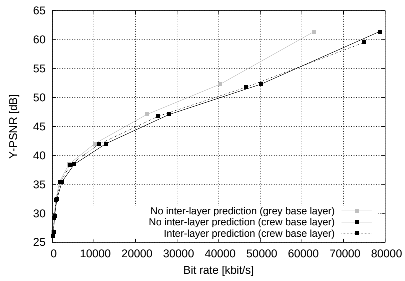


Figure 15: Rate-distortion plot for SVC with two dyadic spatial layers and slice groups with different coding configurations to illustrate the performance differences of the true post-compression approach.

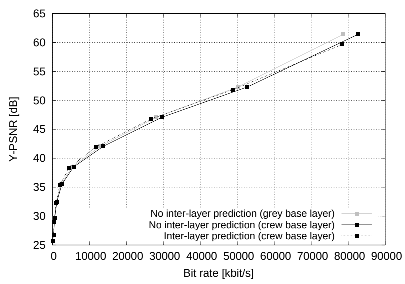


Figure 16: Rate-distortion plot for SVC with three dyadic spatial layers and slice groups with different coding configurations to illustrate the performance differences of the true post-compression approach.

Figure 17: Example for spatial drift due to one changed macroblock in the first frame of the *foreman* sequence: original frame (left) versus modified frame with drift (right).

4.5. Drift

In H.264/AVC, two types of drift can occur – spatial and temporal drift due to intra and inter prediction, respectively. Due to the interdependency of macroblocks through the respective forms of prediction, changes to one macroblock influence one or more other macroblocks. Figures 17 and 18 illustrate this by example. In SVC, inter-layer drift, i.e., the propagation of errors between a base and an enhancement layer, may occur in addition.

Slice groups are able to contain spatial drift as they form a prediction border for intra prediction. This simplifies the encryption of slice groups in an I frame since the blocks outside the RoI, i.e., those which are contained in the slice group which forms the background, cannot use encrypted data for prediction.

For SVC, this applies to the co-located I frames in the higher layers as well, since each block in them is either coded using intra prediction or inter-layer intra prediction, which must use co-located base layer intra samples (see section 3). In addition, the slice group coordinates and size scale along with the spatial layer in our use case, which keeps encoded data inside the RoI due to the co-location property. Thus, encoding each RoI as one slice group prevents spatial drift and inter-layer drift for all I

frames.
However, slice groups are not able to contain temporal drift, since motion vectors may cross slice group boundaries. In H.264/AVC, this means that P and B frames are very likely to exhibit drift. Within each inter frame itself, however, slice groups contain spatial drift due to imposed intra prediction border as well as the limitation of motion vector predictors.
This is also true for P frames in SVC. However, in higher layers, inter-layer drift may occur. Although constrained intra prediction limits the spatial propagation of inter-layer-predicted samples, inter-layer motion and residual prediction may upscale drift-induced errors from the base layer which have been created through temporal drift. This means that inter-layer drift in this use case is only a consequence of temporal drift and can be prevented, if temporal drift can be eliminated.
In summary, when using slice groups in the use case described herein, spatial drift is contained in I, P and B frames for H.264/AVC, and in I and P frames for SVC. However, temporal drift cannot be contained in P and B frames for H.264/AVC, and P frames for SVC. Inter-layer drift in SVC can be contained in I frames in the described use case due to the co-location of slice groups in all layers, but cannot be contained in P frames as it is a consequence of temporal drift.

5. Future work

Although this paper shows that slice groups help containing drift in H.264/AVC and SVC, post-compression encryption approaches which make use of this have yet to be developed. Since the problem of temporal drift remains, this is a challenging task and remains future work.
In addition, the detailed effects of SNR scalability have to be studied. Although SNR scalability can be considered as a special case of spatial scalability where width and height remain the same, the overhead of slice groups in SNR layers may be significantly lower due to the more restricted inter-layer prediction mechanisms. This would make SVC encryption yet more feasible, since SNR layers are identical to spatial layers in terms of drift as analyzed in this paper.

6. Conclusion

We showed the impact of slice group coding on post-compression encryption for a typical surveillance use case. We analyzed the slice-group-induced bit rate overhead as well as the usefulness of slice groups for the containment of drift. For medium and high bit rates, H.264/AVC as well as SVC configurations with two and three layers can be used to reduce drift with slice groups with relatively low overhead. In contrast, for low bit rates, the overhead is too large for practical use. Furthermore, we introduced the concept of all-grey base layers which simplifies encryption significantly in the two- and three-layer case of SVC, albeit at the cost of losing one spatial scalability layer. Finally, we showed that the containment of drift in SVC can be reduced to the containment of temporal drift in H.264/AVC for this surveillance use case.

7. Acknowledgments

This work is supported by FFG Bridge project 832082.

References

[1] Y. Ou, C. Sur, K. H. Rhee, Region-based selective encryption for medical imaging, in: Proceedings of the International Conference on Frontiers in Algorithmics (FAW'07), Lecture Notes in Computer Science, Springer-Verlag, Lanzhou, China, 2007, pp. 62–73.
[2] T. E. Boult, PICO: Privacy through invertible cryptographic obscuration, in: IEEE/NFS Workshop on Computer Vision for Interactive and Intelligent Environments, Lexington, KY, USA, 2005, pp. 27–38.
[3] P. Carrillo, H. Kalva, S. Magliveras, Compression Independent Reversible Encryption for Privacy in Video Surveillance, EURASIP Journal on Information Security 2009 (2009) 1–13.
[4] F. Dufaux, T. Ebrahimi, A framework for the validation of privacy protection solutions in video surveillance, in: Proceedings of the IEEE International Conference on Multimedia & Expo, ICME '10, IEEE, Singapore, 2010, pp. 66–71.
[5] L. Tong, F. Dai, Y. Zhang, J. Li, Prediction restricted H.264/AVC video scrambling for privacy protection, Electronic Letters 46 (1) (2010) 47–49. doi:10.1049/el.2010.2068.
[6] Z. Shahid, M. Chaumont, W. Puech, Selective and scalable encryption of enhancement layers for dyadic scalable H.264/AVC by scrambling of scan patterns, in: 16th IEEE International Conference on Image Processing, Cairo, Egypt, 2009, pp. 1273–1276.
[7] Y. Kim, S. Yin, T. Bae, Y. Ro, A selective video encryption for the region of interest in scalable video coding, in: Proceedings of the TENCON 2007 - IEEE Region 10 Conference, Taipei, Taiwan, 2007, pp. 1–4.
[8] T.-L. Wu, S. F. Wu, Selective encryption and watermarking of MPEG video (extended abstract), in: H. R. Arabnia (Ed.), Proceedings of the International Conference on Image Science, Systems, and Technology, CISST '97, Las Vegas, USA, 1997.
[9] F. Dufaux, T. Ebrahimi, Video surveillance using JPEG 2000, in: Proceedings of the SPIE Applications of Digital Image Processing XXVII, Vol. 5588, 2004, pp. 268–275.
[10] F. Dufaux, T. Ebrahimi, Scrambling for privacy protection in video surveillance systems, IEEE Transactions on Circuits and Systems for Video Technology 18 (8) (2008) 1168–1174. doi:10.1109/TCSVT.2008.928225.
[11] A. Massoudi, F. Lefèbvre, C. D. Vleeschouwer, B. Macq, J.-J. Quisquater, Overview on selective encryption of image and video, challenges and perspectives, EURASIP Journal on Information Security 2008 (Article ID 179290) (2008) doi:10.1155/2008/179290, 18 pages.
[12] ITU-T H.264, Advanced video coding for generic audiovisual services, http://www.itu.int/rec/T-REC-H.264-200711-I/en (Nov. 2007).
[13] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable H.264/MPEG4-AVC extension, in: Proceedings of the IEEE International Conference on Image Processing, ICIP '06, IEEE, Atlanta, GA, USA, 2006, pp. 161–164.
[14] R. Iqbal, S. Shirmohammadi, A. E. Saddik, J. Zhao, Compressed-domain video processing for adaptation, encryption, and authentication, IEEE Multimedia 15 (2) (2008) 38–50.
[15] F. Dufaux, T. Ebrahimi, H.264/AVC video scrambling for privacy protection, in: Proceedings of the IEEE International Conference on Image Processing, ICIP '08, IEEE, San Diego, CA, USA, 2008, pp. 47–49.

[16] H. Sohn, E. Anzaku, W. D. Neve, Y. M. Ro, K. Plataniotis, Privacy protection in video surveillance systems using scalable video coding, in: Proceedings of the Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, Genova, Italy, 2009, pp. 424–429.

[17] Y. Kim, S. Jin, Y. Ro, Scalable Security and Conditional Access Control for Multiple Regions of Interest in Scalable Video Coding, in: Y. Shi, H.-J. Kim, S. Katzenbeisser (Eds.), International Workshop on Digital Watermarking 2007 (IWDW 2007), Vol. 5041, Springer Berlin / Heidelberg, 2008, pp. 71–86.

[18] S. S. F. Shah, E. A. Edirisinghe, Evolving Roi Coding in H.264 SVC, in: VISAPP 2008: Proceedings of the Third International Conference on Computer Vision Theory and Applications – Volume 1, 2008, pp. 13–19.

[19] C. Li, X. Zhou, Y. Zhong, NAL level encryption for scalable video coding, in: Advances in Multimedia Information Processing, PCM'08, Springer-Verlag, 2008, pp. 496–505. doi:10.1007/978-3-540-89796-5.

[20] D. Grois, E. Kaminsky, O. Hadar, Roi adaptive scalable video coding for limited bandwidth wireless networks, in: 2010 IFIP Wireless Days (WD), 2010, pp. 1–5.

[21] Y. Dhondt, S. Mys, K. Vermeirsch, R. Van de Walle, Constrained Inter Prediction: Removing Dependencies Between Different Data Partitions, in: J. Blanc-Talon, W. Philips, D. Popescu, P. Scheunders (Eds.), Advanced Concepts for Intelligent Vision Systems, Vol. 4678 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 720–731.

[22] A. Unterweger, A. Uhl, Slice Groups for Post-Compression Region of Interest Encryption in SVC, in: IH&MMSec'14: Proceedings of the 2014 ACM Information Hiding and Multimedia Security Workshop, ACM, Salzburg, Austria, 2014, pp. 15–22.

[23] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 560–576.

[24] C. A. Segall, G. J. Sullivan, Spatial scalability within the H.264/AVC scalable video coding extension, IEEE Transactions on Circuits and Systems for Video Technology 17 (9) (2007) 1121–1135. doi:10.1109/TCSVT.2007.906824.

[25] A. Leontaris, P. Cosman, Compression Efficiency and Delay Tradeoffs for Hierarchical B-Pictures and Pulsed-Quality Frames, IEEE Transactions on Image Processing 16 (7) (2007) 1726–1740.

[26] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable video coding extension of the H.264/AVC standard, IEEE Transactions on Circuits and Systems for Video Technology 17 (9) (2007) 1103–1120. doi:10.1109/TCSVT.2007.905532.

[27] T. M. Bae, T. C. Thang, D. Y. Kim, J. W. K. Yong Man Ro, J. G. Kim, Multiple Region-of-Interest Support in Scalable Video Coding, ETRI Journal 28 (2) (2006) 239–242.

[28] T. C. Thang, T. M. Bae, Y. J. Jung, Y. M. Ro, J.-G. Kim, H. Choi, J.-W. Hong, Spatial Scalability of Multiple ROIs in Surveillance Video, http://wftp3.itu.int/av-arch/jvt-site/2005_04_Busan/JVT-O037.doc (Jan. 2005).

[29] J.-H. Lee, C. Yoo, Scalable ROI algorithm for H.264/SVC-based video streaming, in: 2011 IEEE International Conference on Consumer Electronics (ICCE), 2011, pp. 201–202.

[30] H. Schwarz, T. Hinz, D. Marpe, T. Wiegand, Constrained inter-layer prediction for single-loop decoding in spatial scalability, in: IEEE International Conference on Image Processing (ICIP) 2005, Vol. 2, 2005, pp. II–870–873.

[31] P. Lambert, W. D. Neve, Y. Dhondt, R. V. de Walle, Flexible macroblock ordering in H.264/AVC, Journal of Visual Communication and Image Representation 17 (2) (2006) 358–375.

Slice Groups for Post-Compression Region of Interest Encryption in SVC

Andreas Unterweger
University of Salzburg
Department of Computer Sciences
Jakob-Haringer-Straße 2
Salzburg, Austria
aunterweg@cosy.sbg.ac.at

Andreas Uhl
University of Salzburg
Department of Computer Sciences
Jakob-Haringer-Straße 2
Salzburg, Austria
uhl@cosy.sbg.ac.at

ABSTRACT

In this paper, we assess the adequacy of slice groups for the reduction of drift which occurs in bit-stream-based region of interest encryption in SVC. For practical surveillance camera applications, we introduce the concept of all-grey base layers which simplify the encryption of regions of interest while obeying all standard-imposed base layer restrictions. Furthermore, we show that the use of slice groups is possible with relatively low overhead for most practical configurations with two or three spatial layers. In addition, we analyze the effect of spatial resolution on overhead, showing that an increase in resolution decreases the relative overhead.

Categories and Subject Descriptors

I.4.2 [**Image Processing and Computer Vision**]: Compression (Coding)—*SVC, Slice groups*; E.3 [**Data**]: Data Encryption—*Selective encryption*

General Terms

Experimentation, Measurement, Verification

Keywords

SVC, Slice groups, Selective encryption, Region of Interest, Overhead

1. INTRODUCTION

In video surveillance and other applications, there is often the need to disguise people's identities in order to protect their privacy. A common approach to achieve this is the selective encryption of people's faces (also called region-based selective [15] or Region of Interest (RoI) encryption), i.e. encrypting all picture areas which contain a face, while leaving all other picture areas untouched.

This allows for reversible de-identification, i.e., the disguise of identities with the possibility to restore them by undoing the encryption. Restoring is typically only possible with a correct key which is possessed, e.g., by law enforcement authorities in case suspects of a crime need to be identified. Although several techniques for reversible de-identification exist, RoI encryption is one of the most common ones in video surveillance.

While RoI encryption can be applied before (e.g., [2, 3, 7]), during (e.g., [23, 20, 11]) or after compression (e.g., [24, 4, 6]), each with its own advantages and disadvantages [14], most approaches proposed so far focus either on encryption before or during compression. Although this makes drift, i.e., the propagation of parts of encrypted picture areas into non-encrypted ones through spatial and temporal prediction as depicted in figure 1, easier to manage, it does not allow using existing surveillance infrastructure whose input images and/or encoder cannot be modified.

Typically, surveillance cameras have compression hardware built in (as of 2013, Motion JPEG and H.264 are very common) which is used to reduce the bandwith of the captured and transmitted video footage. Although this saves time and computational resources by not requiring additional encoding hardware, it makes modifications (like an additional encryption step) to the built-in compression hardware quasi impossible due to the often hard-wired encoder.

In order to be able to reuse this infrastructure notwithstanding, applying RoI encryption after compression has to be considered, reviving the drift issue. Therefore, in this pa-

Figure 1: Examples for spatial and temporal drift: The first, second and tenth frame (from left to right) of the *foreman* sequence (top: original, bottom: encrypted) where one macroblock block row around the eyes in the first frame (left) has been encrypted.

IH&MMSec'14, June 11–13, 2014, Salzburg, Austria.
Copyright 2014 ACM 978-1-4503-2647-6/14/06 ...$15.00.
http://dx.doi.org/10.1145/2600918.2600940.

per, we try to assess the fitness of the slice group coding tool of Scalable Video Coding (SVC) [17] for allowing to selectively encrypt picture areas and containing drift.
For the sake of applicability, we consider a state-of-the-art video surveillance system which delivers SVC bit streams. We assume that the surveillance system detects faces (or other regions of interest) and places them in slice groups which are to be encrypted after compression. The main reason for using slice groups is their ability to contain drift to a certain extent, thereby simplifying RoI encryption. Note that slice groups have other uses as well, thereby extending the results of our investigations to scenarios which are not encryption-specific.
By evaluating the limitations and possibilities of slice group coding, we aim at determining whether or not the aforementioned setup simplifies the encryption process in terms of drift. Furthermore, we evaluate the overhead induced by this approach in order to determine whether or not it is of practical use, i.e., for example to be included into existing and/or future surveillance systems to simplify RoI encryption after compression.
Related work on RoI encryption in SVC is sparse. Two approaches are proposed in [21] and [10], albeit without considering or compensating for the effects of drift, which is an important matter. [11] deals with drift by imposing restrictions on the encoding process in terms of a limited motion estimation range as well as interpolation and upsampling constraints. Besides the reported significant increase in bit rate, this method cannot be applied on a bit stream level without recompression. Similarly, [19] proposes separate RoI coding by restricting motion estimation and inter-layer prediction, albeit without the explicit intention to do so for the sake of encryption. However, all of these approaches are in-compression encryption methods and cannot be applied at bit stream level.
Apart from RoI-related experiments and analyses of SVC, the encryption of certain Network Abstraction Layer (NAL) units has been proposed in [13]. However, their proposed encryption approach yields bit streams which are no longer format compliant and can hence not be decoded anymore by a regular decoder. This is not desirable in surveillance applications as the background without the encrypted RoI should be visible and therefore decodable. Furthermore, the optimization of RoI across multiple layers to lower the total bit rate has been analyzed in [8].
Although slice groups have been used to deal with drift in a number of encryption approaches (e.g., [23, 5]), a detailed examination of its actual usefulness to contain different causes of drift has not been done so far. The overhead induced by some of the aforementioned encryption approaches has been analyzed, but this is not true for the general overhead introduced by slices groups which change from frame to frame to cover RoI. This is especially true for SVC.
This paper is structured as follows: In section 2, the key concepts of video coding with slice groups in SVC are described, followed by an analysis of their limitations in section 3. After evaluating several scenarios in terms of feasibility for video surveillance with encrypted RoI in section 4, we conclude our paper.

2. Scalable Video Coding

SVC is specified as the scalable extension of H.264, specified in its Annex G [9]. It allows for multiple so-called layers within one bit stream, which can be accessed or extracted depending on the capabilities of the device decoding the stream. Each layer differs from the others either by frame rate (temporal scalability), resolution (spatial scalability) or quality (Signal-to-Noise Ratio (SNR) scalability). The bottom-most layer is referred to as base layer and coded in a way that is compatible with (non-scalable) H.264.
All layers but the base layer can exploit inter-layer redundancies by using coded information of lower layers for prediction. The basis of this prediction for spatial and SNR scalability can either be filtered intra-coded samples (inter-layer intra prediction), motion vectors (inter-layer motion prediction) or inter-coded difference signal samples (inter-layer residual prediction), with details for each prediction type to be found in [18]. In contrast, temporal scalability is achieved through hierarchical inter prediction as explained in detail in [17].

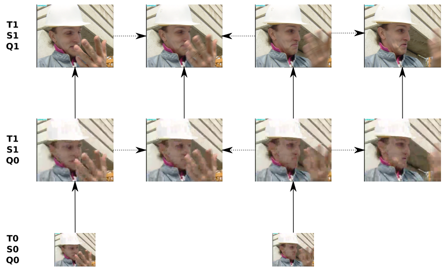


Figure 2: SVC with multiple layers: The base layer with half the frame rate and a quarter of the picture size can be used to predict the first spatial enhancement layer, which itself can be used to predict a second temporal and subsequently a third SNR enhancement layer. Adopted from [17]

Figure 2 shows an example of a scalable bit stream with multiple layers. The base layer (temporal layer 0 (T0), spatial layer 0 (S0) and SNR layer 0 (Q0)) has the lowest possible frame rate, resolution and quality and is used to predict the first spatial enhancement layer (T0, S1, Q0; not labeled) which doubles both, picture width and height. This enhancement layer is further used to predict an enhancement layer of the same resolution, but a doubled frame rate (T1, S1, Q0) as well as an enhancement layer with higher quality (T0, S1, Q1; not labeled) and subsequently a doubled frame rate (T1, S1, Q1).
In each layer, a coded picture is split into slices which can be summarized to slice groups of specific forms, depending on the so-called slice group map type. As RoI encryption requires a background left-over, i.e., a region of the picture which does not belong to any encrypted region of interest, only slice group map types 2 (foreground slice groups with left-over background) and 6 (explicit slice group specification) will be considered, as only they allow this. Since slice group map type 6 is practically identical to slice group map type 2 in this use case, we will only consider slice group map type 2 henceforth.
To exploit spatial and temporal redundancy, samples can be

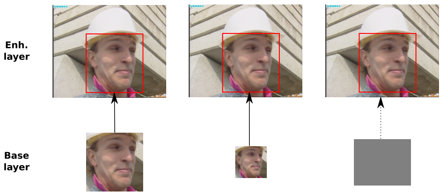


Figure 3: Alternatives to slice groups in the base layer: Left and middle: Extended spatial scalability; right: all-grey base layer

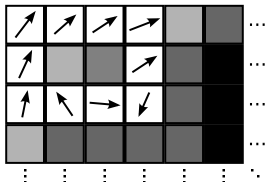


Figure 4: Constrained intra prediction: In a P slice, intra blocks may not use inter blocks for prediction. The grey level of the depicted intra blocks denotes the number of allowed intra modes

predicted from neighboring samples around the block to be predicted (in the same picture) as well as from samples of arbitrary blocks in previously coded pictures. In the former case, predictions over slice borders are forbidden, thereby allowing all slices to be decoded independently and preventing spatial drift.

3. STANDARD-IMPOSED LIMITATIONS

The H.264 standard imposes restrictions on coding tools and parameter values by specifying profiles. As this paper discusses slice groups, we only consider profiles which allow the use of multiple slice groups in the first place. In this section, we investigate other relevant limitations imposed by those profiles.

For scalable bit streams, only the Scalable Baseline profile supports slice groups. Although it allows using up to eight slice groups in total, one slice group is considered to be the background, i.e., the remainder of what the other seven slice groups encode. In addition, entropy coding is limited to CAVLC, the number of slice groups cannot exceed seven (plus background) and B slices are not allowed. Furthermore, the base layer may not contain more than one slice group.

This is a severe limitation in an encryption scenario because this means that the regions of interest cannot be in separate slice groups in the base layer. Thus, either a different drift compensation approach for the base layer is required or an alternative to slice groups in the base layer has to be found. As the former is hard to achieve, we consider three additional alternatives to slice groups in the base layer as depicted in Figure 3.

One possibility is to use extended spatial scalability, depicted on the left and in the middle of Figure 3, where the base layer only contains the region of interest and the enhancement layer adds the rest of the video frame. Due to the limitations of the Scalable Baseline profile, the width and height ratios between the base layer and the corresponding region of interest in the enhancement layer have to be either 1 (Figure 3, left), 1.5 (not depicted) or 2 (Figure 3, middle). However, this setup is only useful if there is exactly one region of interest. Since this would impose a severe practical limitation, it is not considered in the remainder of this paper. Alternatively, we propose adding a base layer which is all-grey ($Y = C_b = C_r = 128$) as shown in Figure 3, right. Since intra DC prediction and skip modes allow encoding such an artificial layer very compactly, its overhead is relatively small when using the maximum possible width and height ratios of 2, i.e., a base layer with half the width and height of the enhancement layer.

However, it effectively reduces the number of usable spatial layers, which is limited to three in the Scalable Baseline profile, by one. This allows for a maximum of two non-grey spatial layers for actual video content. Depending on the use case, these two remaining layers may be sufficient to provide spatial scalability.

Despite the loss of one usable spatial layer, the grey base layer simplifies encryption by containing drift. Although the unavailability of slice groups in the base layer (see above) would normally make encryption harder (without the possibility of using slice groups to contain drift), the fact that the base layer is all grey does not require any encryption and does therefore not induce any drift.

Although there have been multiple proposals for region-of-interest support through slice groups in all layers [1, 22], the final version of the standard does not allow this. Similarly, the technique proposed in [12] to alternatively support regions of interests as enhancement layers is not supported. This paper limits the available options to the ones supported by the standard, i.e., the all-grey base layer introduced above as well a regular (i.e., full-content) base layer for comparison.

Regarding further limitations imposed by the standard, we will focus on the combination of constrained intra prediction and constrained inter-layer prediction, which ensure single-loop decoding [16]. Since these two limitations severely limit the number of possibilities for prediction and therefore drift, they are crucial for the RoI encryption use case.

Constrained intra prediction limits the blocks which can be used for intra prediction. Figure 4 illustrates this in a P slice which contains inter (depicted by motion vectors) and intra (depicted by grey levels) macroblocks. Although the black intra blocks may use all possible intra prediction modes, the dark- and light-grey ones may not. For example, the light-grey macroblock at the top left may only use DC prediction since all other prediction directions would require predicting from one of the surrounding inter macroblocks.

SVC enforces constrained intra prediction in all layers which are used for inter-layer prediction so that inter-layer predicted samples do not require additional motion compensation in the base layer. Additionally, constrained inter-layer prediction ensures that inter-layer-predicted intra samples are not used for intra prediction themselves, as illustrated in Figure 5.

Inter-layer prediction allows using information from the base

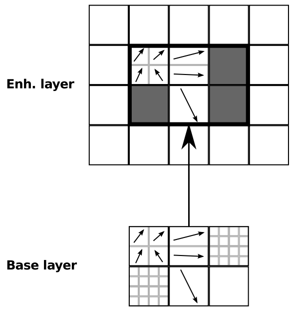


Figure 5: Constrained inter-layer prediction: Upsampled intra blocks (grey) must be reconstructed from base layer intra samples

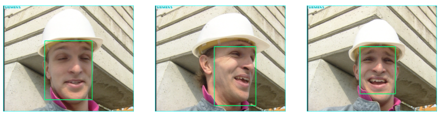


Figure 6: Moving slice groups: Frames 1, 11 and 21 of the *foreman* sequence with one moving foreground slice group around the face (green) and one background slice group (remainder, turquoise)

layer in the enhancement layer. If blocks are upsampled through inter-layer intra prediction (grey blocks in Figure 5), the corresponding reference block in the base layer has to be an intra block as well. Constrained inter prediction in the base layer ensures that no additional motion compensation loop is required. Furthermore, if the enhancement layer is used for further inter layer prediction, the upsampled blocks may not be used for further intra prediction due to the constrained intra prediction requirement to avoid multi-loop decoding.

4. EXPERIMENTAL EVALUATION

In this section, we describe our experimental setup and results. We refer to the term of "moving slice groups" for RoI herein since the position of RoI may change from frame to frame, thereby changing the slice group positions accordingly, as illustrated in Figure 6. Recall that our use case is encryption, i.e., we assume that the moving slice groups will be encrypted at some point, as illustrated by example in figure 7.

4.1 Setup

In order to evaluate the effect of slice-group-based RoI for encryption, we added support for moving slice groups to the SVC reference software (*JSVM*) since it does not support this by itself.
In the *JSVM*, slice group coding is implemented partially, but not used. Therefore, it is enabled separately for all spatial layers but the base layer which does not support slice

Figure 7: Encrypted RoI: Frames 1, 11 and 21 of the *foreman* sequence. The RoI in this example is the actor's face

group coding (see section 3). This is done by setting the slice group map type to 2 using the current layer's Picture Parameter Set (PPS) in *LayerEncoder::process*.
In each layer, the RoI coordinates are calculated depending on the picture size and the corresponding slice group settings (number of slice groups, top-left and bottom-right coordinates) in the PPS are adapted accordingly. In order to determine the absolute frame number in the layer being processed, a helper variable is introduced which counts the number of processed Group Of Pictures (GOP). Together with the frame index of the current GOP (which is provided by the encoder), an absolute frame number can be calculated so that the corresponding RoI coordinates can determined.
In order to signal the slice groups, one additional PPS per frame and enhancement layer is needed. Although the PPS changes described above take effect in the encoder immediately, the decoder needs to take notice of them by a PPS update. This requires inserting one PPS per frame per enhancement layer into the bit stream using the corresponding functions provided by the *NalUnitEncoder* class. Note that it is essential to use the *LayerEncoder::xAppendNewExtBinDataAccessor* and *LayerEncoder::addParameterSetBits* functions to assign the PPS NAL unit and its corresponding overhead to the correct layer.
We use three test sequences with 300 frames each to simulate typical surveillance senarios. *akiyo* has one RoI and very little motion, while *foreman* has a significant amount of motion and also one RoI. Conversely, the *crew* sequence has a changing number of RoI. Since a maximum of seven slice groups (RoI) is supported in SVC (see section 3), only the first top-left-most faces are considered, i.e., placed in a separate slice group. All faces were segmented manually by enclosing them in rectangles. The corresponding coordinates were rounded to the nearest macroblock border.
We use both, Common Intermediate Format (CIF) and 4CIF resolution, in order to determine the impact of spatial resolution on the measurements. While the following section gives a detailed description of the results for CIF resolution, section 4.3 describes the differences when using 4CIF resolution.

4.2 Overhead (CIF)

We use the GOP size of the default JSVM configuration, i.e., four. Since GOP structures with B frames are not allowed in combination with slice groups (see section 3), we use P frames instead. Thus, an $(IPPP)*$ GOP structure, i.e., a repeated sequence of one I frame and followed by three P frames, is used.
We encode the test sequences with a constant Quantization Parameter (QP) for both frame types and default settings with two and three dyadic spatial layers. The base layer is all grey (see section 3), although we test "classical" base

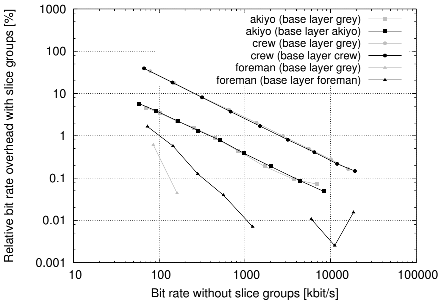


Figure 8: Overhead with slice group coding for different CIF sequences when using two dyadic spatial layers

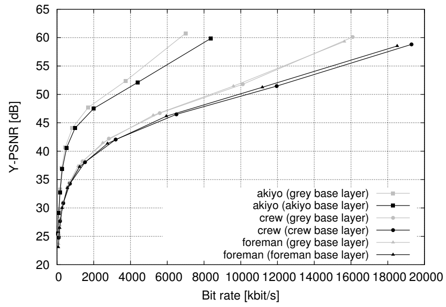


Figure 9: Rate-distortion plot for SVC with two dyadic spatial layers and slice groups. Different base layers (depicted in grey and black) result in significantly different enhancement layer Y-PSNR.

layers (with the actual down-sized input video) as well for comparison. Inter-layer prediction is set to adaptive to allow for optimal coding efficiency.

We encode the test sequences with a constant QP for all frame types and default settings. Using QPs between 3 and 51 with a step size of 6 to double the quantizer step size with each run allows covering the whole QP range. Each QP-sequence combination is encoded with and without slice groups. Since the difference in terms of distortion between the encoded sequences with and without slice groups is very small (< 0.15 dB), we approximate the overhead introduced by slice group coding by comparing the corresponding bit rates directly.

As depicted in Figure 8, it is obvious that the *crew* sequence (depicted by circles in figure) exhibits the highest overhead in quasi all scenarios, since it requires the highest number of slice groups. Conversely, the *foreman* sequence exhibits the lowest overhead, since it requires only one additional slice group (apart from the background) for the first half of the sequence. It profits from scalability more than the other sequences, resulting in some very small negative overhead values ($< 0.1\%$ absolute). Note that these values cannot be depicted properly due to the logarithmic Y axis.

In general, the overhead decreases with the bit rate, i.e., it increases with the QP. For low bit rates, slice group coding adds an unacceptable overhead of up to nearly one hundred per cent. Conversely, for bit rates which are higher than 500 kbit/s, all sequences but *crew* exhibit a small overhead of approximately 1% or less.

Using an all-grey base layer does not affect the overhead significantly due to the use of slice groups, except for very low bit rates, which are impractical. Compared to the classical base layer configuration, however, an all-grey base layer allows using slice-group-based encryption for SVC in the first place, since slice groups cannot be used in the base layer (see section 3).

Figure 9 shows a rate-distortion plot for the two-layer case with slice groups, where the Y-PSNR values are those of the enhancement layer. The plot allows comparing the all-grey base layer with a classical base layer. It is obvious that the all-grey base layer results in significantly better rate-distortion performance (up to 5 dB) for medium and high bit rates.

Since an all-grey base layer greatly improves rate-distortion performance avoiding the need for additional drift compensation due to encryption in the base layer, it can be considered a better solution than a classical base layer for this use case. As the overhead due to slice groups is similar in both, the all-grey and the classical base layer scenario (see above), this is also true for other potential use cases in which the base layer does not have to be the downsampled input sequence.

Note that an all-grey base layer in a scenario with two spatial layers defies the purpose of scalable video coding, since one of the two layers becomes unusable for content. However, it allows establishing a baseline for comparison in terms of overhead and allows assessing the usefulness of the concept. In order for all-grey base layers to be practically useful, a scenario with three spatial layers has to be considered so that two spatial layers remain for actual content.

When increasing the number of spatial layers to the maximum of three (see section 3), the overhead due to slice groups increases, as depicted in Figure 10. The overall overhead is significantly higher than in the two-layer case (see Figure 8) for low to medium bit rates. This is due to the fact that slice groups introduce prediction borders which reduce coding efficiency and the three-layer case (with two enhancement layers with slice groups) uses double the amount of slice groups than the two-layer case (with one enhancement layer with slice groups). However, for high bit rates, the overhead is still relatively small and therefore practically negligible for most use cases.

Compared to the two-layer case, the all-grey base layer configuration in the three-layer case allows for an overhead which is approximately as low as the overhead in the classical base layer configuration. Although the all-grey base layer configuration exhibits a higher overhead for medium-to-high bit rates, the actual overhead is only insignificantly higher.

However, in the three-layer case the rate-distortion performance improvement of the all-grey base layer is only very small, as depicted in Figure 11. Although there are still

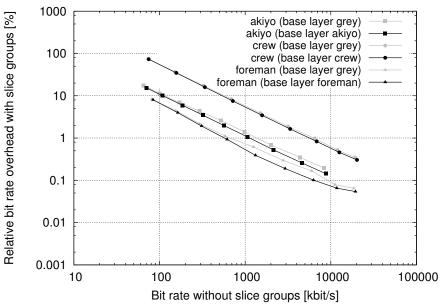


Figure 10: Overhead with slice group coding for different CIF sequences when using three dyadic spatial layers

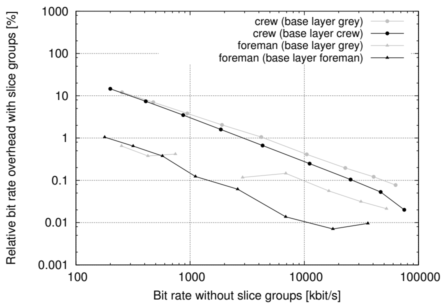


Figure 12: Overhead with slice group coding for different 4CIF sequences when using two dyadic spatial layers

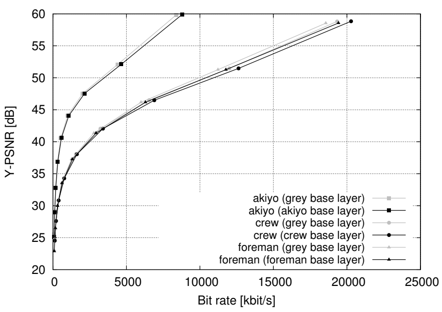


Figure 11: Rate-distortion plot for SVC with three dyadic spatial layers and slice groups. Different base layers (depicted in grey and black) result in similar enhancement layer Y-PSNR.

differences of up to 1 dB between an all-grey and a classical base layer in terms of enhancement layer Y-PSNR, but the performance improvement is nowhere near the improvements of the two-layer case (see above).

This is mainly due to the fact that there are two enhancement layers, which use most of the bit rate and the fact that the first enhancement layer can be used to predict parts of the second one through inter-layer prediction. This makes the three-layer case with an all-grey base layer similar to a two-layer case with an additional all-grey bit stream, which is very likely not used at all for inter-layer prediction. However, an all-grey base layer still has advantages compared to a classical base layer for the use case in this paper, since base layer encryption cannot rely on slice groups due to base layer limitations (see above). Thus, an all-grey base layer is still to be preferred over a classical base layer in the three-layer case.

4.3 Overhead (4CIF)

In order to analyze the influence of spatial resolution on overhead, we repeat the experiments of the previous section with sequences in 4CIF resolution. Note that a 4CIF version of *akiyo* could not be obtained, which is why the following paragraphs only describe results for the *foreman* and the *crew* sequences.

Figure 12 shows the overhead induced by moving slice groups with two spatial layers, like in the previous section. As expected, the decrease of the relative overhead with increasing bit rate is quasi identical, while the overhead values are mostly smaller. Due to the higher spatial resolution, the percentage of macroblocks which are affected by the slice-group-induced prediction borders is smaller, thereby increasing coding efficiency compared to the CIF case depicted in figure 8.

The overhead for the *foreman* sequence (triangles) is 1% or lower for all QP. Although the use of an all-grey base layer introduces a larger overhead than in the CIF case depicted in figure 8, it can still be considered insignificantly small for most QP.

The overhead for the *crew* sequence (circles) is about two to three times lower than in the CIF case depicted in figure 8 when comparing equal QP, and quasi identical when comparing equal bit rates. This is due to the high number of slice groups in the *crew* sequence which induce a significant number of prediction borders. At 4CIF resolution, these have a smaller effect than at CIF resolution, as described for the *foreman* sequence above.

When the *crew* sequence is encoded with an all-grey base layer (grey circles), the overhead is slightly higher than in the regular base layer case (black circles). Although this deviates from the behavior at CIF resolution, where both curves overlap quasi completely, the difference can still be considered to be insignificantly small.

Figure 13 shows the overhead induced by moving slice groups with three spatial layers. As in the two-layer case, the overhead for the *foreman* sequence (triangles) is about 1% or lower. The overhead for the *crew* sequence (circles) is about two to four times lower than in the CIF case depicted in figure 10 when comparing equal QP, and quasi identical when

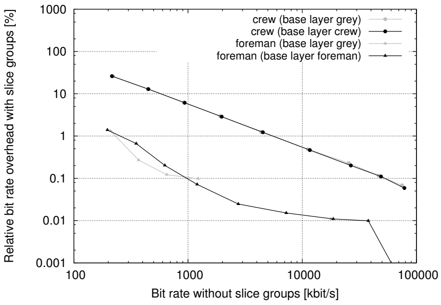


Figure 13: Overhead with slice group coding for different 4CIF sequences when using three dyadic spatial layers

comparing equal bit rates.
When using an all-grey base layer (grey circles), the overhead is quasi indistinguishable from the regular base layer case (black circles). Consequently, using all-grey base layers at higher resolutions is recommended for two and three layers, as in the CIF resolution case.
Regarding the overhead results for higher resolutions in general, it can be concluded that equal bit rates yield quasi equal overhead values. Since the overhead decreases with bit rate, the relative overhead decreases with increasing resolution. Thus, at higher resolutions than 4CIF, it is to be expected that the overhead with moving slice groups becomes so small that it can, for most use cases, be ignored.

5. FUTURE WORK

This paper shows that slice groups help containing drift in SVC. Although this result can be extended to (non-scalable) H.264 for the most part (since SVC is built upon H.264), a detailed analysis of the overhead induced by slice groups in (non-scalable) H.264 bit streams is desired. Furthermore, the use of B frames and other GOP structures on both, the overhead and the ability to contain drift has to be investigated.
In addition, the detailed effects of SNR scalability have to be studied. Although SNR scalability can be considered as a special case of spatial scalability where width and height remain the same, the overhead of slice groups in SNR layers may be significantly lower due to the more restricted inter-layer prediction mechanisms. This would make SVC encryption yet more feasible, since SNR layers are identical to spatial layers in terms of drift as analyzed in this paper.

6. CONCLUSION

We showed the impact of slice group coding on post-compression encryption for a typical surveillance use case. We analyzed the slice-group-induced bit rate overhead as well as the usefulness of slice groups for the containment of drift. For medium and high bit rates, configurations with two and three spatial layers can be used to reduce drift with slice groups with relatively low overhead. For low bit rates, the overhead is too large for practical use at CIF resolution, but moderate at 4CIF and higher resolutions since the relative overhead decreases with increasing resolution. Furthermore, we introduced the concept of all-grey base layers which simplifies encryption significantly in the two- and three-layer case, albeit at the cost of losing one spatial scalability layer.

7. ACKNOWLEDGMENTS

This work is supported by FFG Bridge project 832082.

8. REFERENCES

[1] T. M. Bae, T. C. Thang, D. Y. Kim, J. W. K. Yong Man Ro, and J. G. Kim. Multiple Region-of-Interest Support in Scalable Video Coding. *ETRI Journal*, 28(2):239–242, Apr. 2006.
[2] T. E. Boult. PICO: Privacy through invertible cryptographic obscuration. In *IEEE/NFS Workshop on Computer Vision for Interactive and Intelligent Environments*, pages 27–38, Lexington, KY, USA, Nov. 2005.
[3] P. Carrillo, H. Kalva, and S. Magliveras. Compression Independent Reversible Encryption for Privacy in Video Surveillance. *EURASIP Journal on Information Security*, 2009:1–13, Jan. 2009.
[4] F. Dufaux and T. Ebrahimi. Video surveillance using JPEG 2000. In *Proceedings of the SPIE Applications of Digital Image Processing XXVII*, volume 5588, pages 268–275, Aug. 2004.
[5] F. Dufaux and T. Ebrahimi. H.264/AVC video scrambling for privacy protection. In *Proceedings of the IEEE International Conference on Image Processing, ICIP '08*, pages 47–49, San Diego, CA, USA, Oct. 2008. IEEE.
[6] F. Dufaux and T. Ebrahimi. Scrambling for privacy protection in video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1168–1174, 2008.
[7] F. Dufaux and T. Ebrahimi. A framework for the validation of privacy protection solutions in video surveillance. In *Proceedings of the IEEE International Conference on Multimedia & Expo, ICME '10*, pages 66–71, Singapore, July 2010. IEEE.
[8] D. Grois, E. Kaminsky, and O. Hadar. Roi adaptive scalable video coding for limited bandwidth wireless networks. In *2010 IFIP Wireless Days (WD)*, pages 1–5, Oct. 2010.
[9] ITU-T H.264. Advanced video coding for generic audiovisual services, Nov. 2007. `http://www.itu.int/rec/T-REC-H.264-200711-I/en`.
[10] Y. Kim, S. Jin, and Y. Ro. Scalable Security and Conditional Access Control for Multiple Regions of Interest in Scalable Video Coding. In Y. Shi, H.-J. Kim, and S. Katzenbeisser, editors, *International Workshop on Digital Watermarking 2007 (IWDW 2007)*, volume 5041, pages 71–86. Springer Berlin / Heidelberg, 2008.
[11] Y. Kim, S. Yin, T. Bae, and Y. Ro. A selective video encryption for the region of interest in scalable video coding. In *Proceedings of the TENCON 2007 - IEEE Region 10 Conference*, pages 1–4, Taipei, Taiwan, Oct. 2007.
[12] J.-H. Lee and C. Yoo. Scalable ROI algorithm for H.264/SVC-based video streaming. In *2011 IEEE*

International Conference on Consumer Electronics (ICCE), pages 201–202, Jan. 2011.

[13] C. Li, X. Zhou, and Y. Zhong. NAL level encryption for scalable video coding. In *Advances in Multimedia Information Processing, PCM'08*, pages 496–505. Springer-Verlag, Dec. 2008.

[14] A. Massoudi, F. Lefèbvre, C. D. Vleeschouwer, B. Macq, and J.-J. Quisquater. Overview on selective encryption of image and video, challenges and perspectives. *EURASIP Journal on Information Security*, 2008(Article ID 179290):doi:10.1155/2008/179290, 18 pages, 2008.

[15] Y. Ou, C. Sur, and K. H. Rhee. Region-based selective encryption for medical imaging. In *Proceedings of the International Conference on Frontiers in Algorithmics (FAW'07)*, Lecture Notes in Computer Science, pages 62–73, Lanzhou, China, Aug. 2007. Springer-Verlag.

[16] H. Schwarz, T. Hinz, D. Marpe, and T. Wiegand. Constrained inter-layer prediction for single-loop decoding in spatial scalability. In *IEEE International Conference on Image Processing (ICIP) 2005*, volume 2, pages II–870–873, Sept. 2005.

[17] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable H.264/MPEG4-AVC extension. In *Proceedings of the IEEE International Conference on Image Processing, ICIP '06*, pages 161–164, Atlanta, GA, USA, Oct. 2006. IEEE.

[18] C. A. Segall and G. J. Sullivan. Spatial scalability within the H.264/AVC scalable video coding extension. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1121–1135, Sept. 2007.

[19] S. S. F. Shah and E. A. Edirisinghe. Evolving Roi Coding in H.264 SVC. In *VISAPP 2008: Proceedings of the Third International Conference on Computer Vision Theory and Applications – Volume 1*, pages 13–19, 2008.

[20] Z. Shahid, M. Chaumont, and W. Puech. Selective and scalable encryption of enhancement layers for dyadic scalable H.264/AVC by scrambling of scan patterns. In *16th IEEE International Conference on Image Processing*, pages 1273–1276, Cairo, Egypt, Nov. 2009.

[21] H. Sohn, E. Anzaku, W. D. Neve, Y. M. Ro, and K. Plataniotis. Privacy protection in video surveillance systems using scalable video coding. In *Proceedings of the Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 424–429, Genova, Italy, Sept. 2009.

[22] T. C. Thang, T. M. Bae, Y. J. Jung, Y. M. Ro, J.-G. Kim, H. Choi, and J.-W. Hong. Spatial Scalability of Multiple ROIs in Surveillance Video. http://wftp3.itu.int/av-arch/jvt-site/2005_04_Busan/JVT-O037.doc, Jan. 2005.

[23] L. Tong, F. Dai, Y. Zhang, and J. Li. Prediction restricted H.264/AVC video scrambling for privacy protection. *Electronic Letters*, 46(1):47–49, Jan. 2010.

[24] T.-L. Wu and S. F. Wu. Selective encryption and watermarking of MPEG video (extended abstract). In H. R. Arabnia, editor, *Proceedings of the International Conference on Image Science, Systems, and Technology, CISST '97*, Las Vegas, USA, Feb. 1997.

Noname manuscript No.
(will be inserted by the editor)

An Industry-Level Blu-ray Watermarking Framework

Jan De Cock · Heinz Hofbauer · Thomas Stütz · Andreas Uhl · Andreas Unterweger

the date of receipt and acceptance should be inserted later

Abstract In this paper, we present our H.264 Blu-ray watermarking framework which operates at bit stream level and preserves the length of the underlying bit stream. Apart from a description of our watermark embedding and detection (and synchronisation) approaches, we discuss the embedding capacity for different exemplary Blu-ray disks based on their bit stream characteristics as well as the robustness of our watermark to H.264 transcoding and resizing. Furthermore, we assess the parallelizability of our embedding approach and the impact of different hard drive configurations on the overall embedding speed, showing that low access times are as relevant as high transfer rates when maximum speedup through parallelization is desired. Lastly, this paper provides a discussion on a variety of design choices and practical issues which arise when designing an industry-level watermarking framework.

Keywords framework, watermarking, H.264, length-preserving, parallelization

Acknowledgements Special thanks to SONY DADC Austria AG, in particular Reinhard Blaukovitsch, for the cooperation in the project and the insights into industry requirements.
This work has been supported by the FFG bridge project 834165.

Jan De Cock
Ghent University – iMinds, Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
E-mail: jan.decock@ugent.be

Heinz Hofbauer · Andreas Uhl · Andreas Unterweger
University of Salzburg, Jakob Haringer Str. 2, 5020 Salzburg, Austria
E-mail: {hhofbaue, uhl, aunterweg}@cosy.sbg.ac.at

Thomas Stütz
FH Salzburg, Urstein Süd 1, 5412 Puch bei Hallein, Austria
E-mail: thomas.stuetz@fh-salzburg.ac.at

1 Introduction

As more and more movies are released on Blu-ray disk, the number of illegitimate copies which make it onto a variety of platforms throughout the Internet **before** the official release date increases, resulting in significant financial losses. Usually, sales[1] for the second week are about $60 - 80\%$ lower than the first week. This makes the first week the most influential for financial success of a release. A leak prior to the release can thus reduce the revenue of the financially most rewarding sales period. While DRM following release can also be an issue, it is not usually solved by means of a watermark but rather by copy protection mechanisms [10]. It should also be noted that the goal of the watermarking system presented in this paper is not to prevent but to reveal leaks. The security on-site, i.e., in the production plants, and the employed process security is responsible for preventing leaks.

We explicitly only deal with the leakage of content prior to the release date. The goal of the watermarking

[1] Sales information is taken from `http://www.the-numbers.com`

scheme is twofold. On the one hand, it allows to ascertain whether the content was leaked from a specific site, or conversely to plausibly deny that a leak has occurred. And, on the other hand, when a leak has happened, it allows to identify the source of the leak and aid in improving the local security arrangements to prevent further leaks. Content can be leaked in different production stages of a Blu-ray disk, making it necessary to identify the stage in which the leak occurred in order to eliminate it. One way to do so is by adding a watermark after the completion of each production stage. If content leaks, the existence of the watermarks from previous production steps identifies the production step in which the leak occurred.

A number of constraints are imposed on such a watermarking system intended for industrial application. In conjunction with our industrial partner, SONY DADC Austria AG, we identified the following list of constraints for both, practical and economical reasons.

Firstly, the watermark has to be robust against transcoding. The leaked video could be altered in terms of format, bitrate or aspect ratio, e.g., by reencoding to another format. In order to identify the source of the leak, the watermark has to be robust against such changes in order to be detected reliably after a leak.
Secondly, the watermark has to be invisible to the human eye. Any change in quality is a problem for a content provider since it would displease consumers and content creators alike. This in turn can impact sales and the reputation of the content provider.
Thirdly, the size of the watermarked content has to be equal to the size of the original content, i.e., the watermarking process has to be length-preserving. This is a practical restriction originating from a concurrent workflow. On the one side the video content is handled and on the other side the accompanying content, e.g., menus and chapter lists, are handled. On the menu side, the jump-in points to the video content are offset based. As such, they would have to be adjusted whenever the length of the video content changes. This would introduce a higher cost in the production process since the concurrency in the workflow would be inhibited.
Finally, Blu-ray watermarking has to be fast. While "fast" does not necessarily mean real-time processing, it means that undue delays in the production should not occur. This, again, would influence the production cost and is not acceptable. This implies that bitstream-based watermarking is more feasible than other watermarking techniques which require format-compliant reencoding and subsequent compliancy checks.

All other constraints which are usually assumed when dealing with a modern watermarking system, e.g., the requirement of blind watermarking or further robustness issues, are second to these primary concerns.

In this paper, we present a watermarking framework which fulfills all of the aforementioned criteria by watermarking a user-defined selection of the Blu-ray disk's video tracks. As nearly two thirds of the Blu-ray disks released to date contain video streams which are H.264-compliant[2] [7], most of which use context adaptive binary arithmetic coding (CABAC) [9] entropy coding, our approach is targeted at H.264 with CABAC.

Although full watermarking frameworks like ours have not been described in the literature, bit-stream-based and length-preserving watermarking approaches for H.264 have been proposed before. Our watermarking framework uses a variation of the approaches proposed in [11, 12] and [13], which both embed watermarks by changing motion vector differences in the bitstream. Although we do so as well, our modification allows for a significantly higher embedding capacity than the approach described in [13]. This is due to the greater set of modifications allowed by our approach as described in detail in Section 2. Although the capacity of our approach is slightly smaller than the one described in [11], the latter is limited to context adaptive variable length coding (CAVLC) entropy coding, which is rarely used on H.264-compliant Blu-rays.

CAVLC and CABAC are the two ways in which H.264 bit streams are entropy-coded. We apply the watermarking approach of Stütz et al. [11], which performs CAVLC watermarking, to CABAC entropy coded bit streams. Since entropy coding is inherently lossless, the actual changes we make to the visual data are entirely identical to the changes of the approach by Stütz et al. Therefore, both our approaches share the same properties with respect to rate distortion performance, subjective quality degradation and robustness, and security which are therefore not discussed in detail herein (they are described in detail in [11, 12]).

CABAC approaches come at the expense of an additional entropy reencoding step, which is not required by [11, 12] as they aim at finding substitutable code word parts which do not require entropy reencoding. While our CABAC approach employs only one entropy reencoding step for the entire bitstream, numerous fine grain entropy reencodings step are applied in the approach of [13]. The advantage of the approach of [13] is that actual watermark embedding can be implemented by simple bit substitutions. However, in our targeted application scenario this feature (substitution watermarking) is not required.

[2] `http://www.blu-raystats.com/Stats/TechStats.php` as of February 18, 2013

Since our watermarking framework is similar to the CAVLC framework proposed by Stütz et al. [11] as described above, we do not aim at reinvestigating their results, but instead focus on the industry-level implementation of our framework as well as on practical considerations thereby complementing results in [11, 12]. Thus, the contributions of this paper are as follows: First, we detail the technical approach to conduct H.264-CABAC bitstream-based embedding of the CAVLC technique in [11, 12] and explain the corresponding differences to [13]. Second, we discuss questions of detection and (re-)synchronisation in manipulated (i.e. scaled, cropped, transcoded) video. Finally, highly practical questions like computational embedding issues (runtime and storage aspects) as well as embedding capacity are covered.

This paper is structured as follows: In Section 2, we describe our watermarking framework, including the details of our H.264-CABAC-based watermarking algorithm w.r.t. embedding and detection. Subsequently, in Section 3, we outline practical considerations that evolved during the development of our framework (quality control, synchronisation and actual transcoding). Finally, in Section 4, we evaluate our watermarking approach as well as our framework in terms of speed and embedding capacity before concluding this paper in Section 5.

2 Framework Overview

Our watermarking framework consists of two major parts – one for watermark embedding and one for watermark detection. Figure 1 shows the components of the watermark embedding process as well as their interdependencies. The dotted line indicates the interfaces between our framework (on the right) and pre- or post-processing steps which are out of scope.

The watermarking process involves the following steps and components: Firstly, the demuxed H.264 stream is split into the smallest possible groups of pictures (GOPs) to allow parallelized watermarking. Secondly, each GOP is analyzed for possible watermark locations using a modified version of the H.264 reference software (JM). Thirdly, a quality control loop eliminates watermark locations which cause spatial drift as described in detail in Section 2.1.

Finally, the remaining watermarks are embedded using a transcoder as described in Section 2.2 before the watermarked GOPs are merged back together to form the watermarked output stream. Note that the watermark embedding framework additionally outputs detection information for the watermarks, i.e., the precise locations of the watermarks so that they can be found

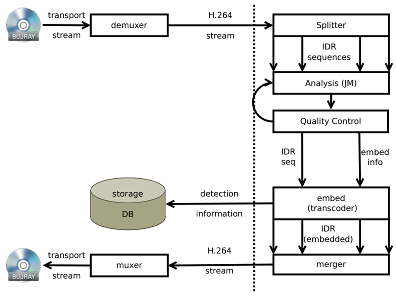


Fig. 1 Watermark Embedding Overview

again during detection, the process of which is described in Section 2.3.

2.1 Watermarking Approach

The basic principle for robust watermarking is to embed the watermark in coefficients of known robustness [4]. For a real world application this requires a feature which is robust against transcoding as well as spatial transformation, i.e., scaling and cropping. Since current video coding standards, e.g., H.264 [7], H.265 [3] and MPEG-4 Part 2 [6], rely on DCT based encoding the DC coefficient, i.e., the average luminance over a macroblock, is a good choice. Furthermore, if the spatial transformation applied to a video can be inverted, the same average luminance can be regained for a given macroblock. The utilization of the average luminance in the DC coefficient for watermarking is further affirmed by literature, see Hartung and Kuttner [5] for an overview or Chen et al. [1] for the use of DC coefficients for H.264. Overall, the known robustness characteristics regarding transcoding and spatial transformations render the DC coefficients the optimal choice for our application scenario.

It is possible to change the luminance of a macroblock by changing the motion vector differences in order to predict from another macroblock. If the new macroblock is brighter or darker, then the macroblock originally used for prediction, the predicted macroblock in the frame will also be brighter or darker. In this way we can adjust the average luminance with a minimal change in the bitstream. To find suitable blocks for watermark embedding we modify the MVD of a macroblock (i.e., we scan every macroblock in the reference frame in a given search radius for brighter and darker macroblocks which do not introduce a too large distortion). A macroblock can be watermarked if we find a brighter (embedding a 1-Bit) and a darker (embedding a 0-Bit) macroblock to predict from.

Only a subset of the candidate MVD changes preserves the length of the bitstream. In H.264/CABAC MVDs are binarized (MVDs larger than 9 are encoded using exponential Golomb codes). Exponential Golomb codes consist of a prefix and a suffix. The bits of the suffix are encoded in bypass mode, i.e., all bits are assumed to have equal probability. In a perfect arithmetic encoder equal probabilities would result in a same length bitstream, in the case of the H.264 arithmetic encoder length-preservation is at least very likely.

While our approach employs all these candidate MVD (with same prefix, but different suffix) the approach of Zou and Bloom [13] further reduces the candidate MVD changes dramatically. Zou and Bloom consider only MVD changes that preserve the exact arithmetic encoder / decoder state. No probability states are updated in bypass mode and the range variable R (codIRange in [7, see clause 9.3.1.2]) is also preserved [13]. However, it has to be checked whether the encoding of the different suffix results in the same offset L (codIOffset in [7, see clause 9.3.1.2]). Therefore the suffix bits need be to arithmetically encoded (for all candidate changes) and checked against the offset from encoding the original suffix. The variable codIOffset is in 16 bit register precision and requires a minimum precision of 10 bits [7, see clause 9.3.1.2]. Thus only one of 1024 candidate changes will not be rejected (using the conservative assumption of a uniform distribution on the values of codIOffset). The significant reduction of candidate changes reduces the capacity and / or requires to analyze more candidate changes. Furthermore, while the approach of Zou and Bloom requires a significant amount of entropy encoding in the analysis step, our approach completely avoids any entropy encoding in the analysis stage and performs only one entropy encoding pass in the embedding stage.

A change in a macroblock can introduce further bit errors through the prediction modes utilized by the H.264. In order to prevent inter-frame propagation of errors we watermark only non-reference frames, i.e., we utilize non-reference B-frames or if the GOP structure is of the form IP* we only change macroblocks from the trailing P-frame in the GOP. There is still the problem of intra frame predictions which can lead to spatial drift in the same frame. In order to deal with this we employ a quality assurance (QA) loop, described in Section 3.1, which detects drift in the decoded frame and reverts the macroblock changes which introduce the drift.

The drift is only removed if a given error is exceeded in non-watermarked macroblocks (for the used threshold see Section 3.1). In order to prevent the drift we remove the embedding from possible prediction sources. Since the intra prediction predicts from macroblocks to the left and above of the current macroblock, only embeddings in this region are removed. By removing all possible prediction ancestors, the QA-loop does not impact the performance of the system unduly, but the embedding capacity is reduced more than strictly necessary. However, since the capacity is still high enough, see Section 4.2, this faster way of removing drift sources is preferable to a slower but more precise method.

2.2 Embedding Approach

When changing the MVDs of a CABAC bit stream by changing the corresponding CABAC code words, the state of the arithmetic coder is very likely to change, resulting in invalid bit streams if the code words are only replaced. Hence, a bit stream transcoder is required which performs the CABAC reencoding so that the rest of the bit stream remains valid. Note that no actual pixel-level decoding or reencoding is necessary as all required changes only involve the entropy coding layer.

As regular transcoders are not capable of performing entropy-only-reencoding with the additional ability to change MVDs, we used a special bitstream transcoder developed at Ghent University which is capable of performing the required changes [2]. The transcoder provides an interface which allows locating and changing the desired MVDs for each frame and outputs the modified, i.e., watermarked, bit stream.

In the transcoder, a cascade of a decoder and an encoder, which is typically used in video stream adaptation, is avoided. Not only will such a cascaded approach lead to a higher complexity (since it combines a decoder and encoder loop), it will also introduce a quality loss, even at identical quantization settings (caused by rounding). To avoid these drawbacks, an open-loop mechanism is used in our transcoder [2]. First, the bit stream is entropy decoded, resulting in the syntax elements listed in the H.264 specification (such as macroblock types, MVDs, and residual coefficients). Then, the MVDs are modified where needed, while all other elements remain identical, hereby avoiding changes which are not related to the watermarking process. Subsequently, the syntax elements are again entropy coded with the updated state of the arithmetic coder.

2.3 Synchronization and Detection

Watermark detection is non-blind and relies on a detection info file containing temporal and spatial watermark location as well as the embedded bit along with the original feature value. In order to extract the watermark from the video under test we have to synchronize the video under test and the original video by reconstructing the original spatial and temporal dimensions. In the spatial domain, eventual scaling and cropping needs to be reversed. For the temporal dimension we only deal with cut or added frames at the beginning of the video.

In essence we only need to determine the crop (left, right, top and bottom) of the original video to the video under test. Given the crop (c_l, c_r, c_t and c_b) we can calculate the inverse aspect ratio and scale since the original video size, $o_w \times o_h$ and the size of the video under test, $t_w \times t_h$, is known from the detection file and actual bitstream respectively. To invert the scaling and cropping by linearly transforming the video from $t_w \times t_h \mapsto (o_w - c_l - c_r) \times (o_h - c_t - c_b)$ and pad with black border according to c_l, c_r, c_t and c_b. See Section 3.2 for strategies how to actually determine crop parameters.

Since the spatial dimensions are aligned we can now utilize the watermark information from the detection file to do a scan for temporal alignment. Utilizing N watermark bits we can scan the first F frames of the video under test and calculate the correlation C, as given below. Under the assumption that the video is watermarked the scan should yield a unique frame offset where the correlation reaches maximum. If the highest correlation is not unique a rescan of the prospective offsets with an increased N should reduce the number of equal correlations until only one remains. This offset is taken as temporal shift and used in the actual watermark detection with the whole watermark sequence. This approach differs from traditional temporal synchronization approaches which utilize redundancy in the watermark, e.g. [8]. However, since we utilize a non-blind watermarking scheme we do not require redundancy since the whole watermark information is available during detection.

Given a synchronized video under test we then have two binary sequences, one is the original watermark sequence, $\mathrm{wm}, \forall i : \mathrm{wm}_i \in \{0, 1\}$, from the detection file which consists of the bits embedded in the original video. The other sequence is the extracted watermark sequence, $\mathrm{ex}, \forall i : \mathrm{ex}_i \in \{0, 1\}$ which is extracted from the synchronized video under test. The extracted watermark sequence is calculated by extracting the relevant feature from the given location and comparing it with the original feature as given in the detection info, this process is illustrated in figure 2.

The detection is based on the probability of false positive, i.e., the probability that a watermark is detected in a non-watermarked video. The watermark bits wm_i are drawn from a uniform random distribution in $\{0, 1\}$ and we assume that the extracted bits ex_i are also uniformly distributed in $\{0, 1\}$. We calculate the correlation between wm and ex in the following manner

$$C = \frac{1}{n} \sum_{i=1}^{n} (2\,\mathrm{wm}_i - 1)(2\,\mathrm{ex}_i - 1),$$

where n is the number of bits of wm and ex. The probability of false positive is then the probability that two random sequences have at least correlation C. We can easily see that each member of the sum, $(2\,\mathrm{wm}_i - 1) \cdot (2\,\mathrm{ex}_i - 1)$, is a Bernoulli trial with $p = q = \frac{1}{2}$. Thus C has a binomial distribution $B(n, p)$ and the probability of false positive is consequently

$$\mathrm{pfp}(C) = \sum_{k=k_C}^{n} \binom{n}{k} p^k q^{(n-k)} = \\ = \sum_{k=k_C}^{n} \binom{n}{k} \left(\frac{1}{2}\right)^n = \frac{1}{2^n} \sum_{k=k_C}^{n} \binom{n}{k},$$

where $k_C = \frac{(C+1)n}{2}$.

We assume video under test is a leaked video if the probability of false positive is lower than a threshold, i.e., $\mathrm{pfp}(C) < T_C$, T_C defaults to 10^{-12} but can be freely chosen.

Figure 3 gives an overview over the probability of false positives (pfp) under different scaling and quantization parameters. An original video (a sample of Band MF), which is encoded in H.264 with HD1080 resolution, was watermarked, 1839 bits were embedded in 1644 frames. The pfp is given in logarithmic scale and capped at 10^{-100}, the default threshold (10^{-10}) for watermark detection is also given. As can be seen the wa-

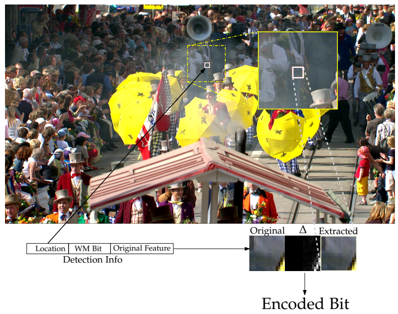


Fig. 2 Extraction of a single watermark bit from a video under test.

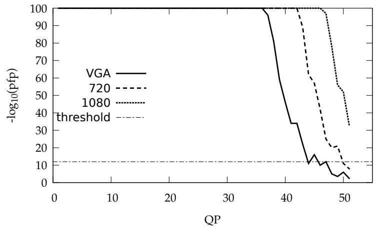


Fig. 3 Probability of false positive for different quantization parameters and resolutions. Plot is capped at pfp $= 10^{-100}$ and the default threshold is also given at 10^{-12}.

termark detection is robust against scaling, bit rate reduction and the transformation to a different aspect ratio, i.e., $16 : 9 \mapsto 4 : 3$. Figure 4 shows the sample part of a frame from the original and for the rescaled versions. The samples from the rescaled version were taken from the sequence with a QP for which the pfp $< T_C$, which is QP 44 and 50 for VGA and HD720 respectively, compare fig. 3. For more information about watermark correlation under different embedding strength and quality parameters see [11].

3 Practical Considerations

In this section we provide information about effects and circumstances which in practice impacted the design and decision making regarding the framework. The topics presented here were selected because they have a huge impact on either the design of the framework, like the decoder, or are important to consider for practical application, i.e., quality assurance and length preservation.

We look at the quality assurance and show how, and why, the current embedding strength was chosen. We explain the practical considerations behind the process of dealing with the situation when a GOP changes length and finally, we explain why the use of a transcoder is necessary and what problems can arise from using a transcoder.

3.1 Quality Assurance

For quality assurance the need to utilize a fast and reliable metric on a basic level lead to the use of the MSE for watermark embedding and quality assurance. In [11, 12] a subjective experiment is presented, which suggests that an embedding strength of 100 in terms of MSE is sufficiently low to be imperceivable. Since our approach and the one from [11, 12] share the same properties as explained in Section 1, we use an embedding strength of 100 as well. On the one hand we found that using MSE 100 as a limit for the macroblock change allows for a sufficient number of watermark bits. Statistics about the possible number of embedded watermark bits, depending on source material, are given in Section 4.2. This high embedding strength results in a good detection response and low probability of false positives, even for highly impaired images, as detailed in Section 2.3, fig. 3. However, a error of higher than 100 MSE can still occur through prediction from a modified macroblock and drift of the error.

We can preclude temporal drift by systematically avoiding embedding in frames which are a source of temporal prediction. This leaves non-reference B-frames or, in the case of GOPs with IP* structure, trailing P-frames for embedding. However, spatial drift of the error can still occur for such frames.

As the targeted application scenario requires reliably high quality, we introduce a quality assurance stage to eliminate spatial drift. In order to prevent a higher than allowed distortion the quality assurance loop checks the whole frame for errors that surpass our MSE 100 limit. If such errors are found the QA loop traces the source of the predictions which introduces these errors and reverts any changes to the responsible macroblocks. A given macroblock is used for prediction only by macroblocks to the right and downwards of the current block. Conversely, the source of an error for a given macroblock is located left or upwards of the current macroblock. The QA loop searches for potential sources of error drift and removes the embedding from them. While this lowers the embedding capacity, the resulting capacity is still high enough for all practical purposes, see Section 4.2.

3.2 Synchronization Method

In the final framework we chose a semiautomatic method for watermark synchronization to improve detection. The main reason was to increase the stability of the detection. The drawback of the semiautomatic method is that human intervention is needed to measure crop, if present. While this is more costly, in terms of personnel cost and time, it also increases the detection rate by providing exact crop detection. However, the time consumed by exact crop measures is refunded by the fast scanning for synchronization which can be done when crop is known.

The other option would be a fully automated synchronization by detecting both crop and synchronization algorithmically. The problem with a fully auto-

(a) HD1080 (b) HD720, QP 50 (c) VGA, QP 44

Fig. 4 Side by side comparison of a sample of frame 111 from the test sequence. Shown are the original and scaled version (for failed watermark detection).

mated approach is that the fast scanning for synchronization requires a known crop while the automatic detection of the crop requires two known matching frames. Thus, we have to switch to a different synchronization method.

3.2.1 Automated Detection of Temporal Displacement

For synchronization without known crop we utilize a scale-invariant feature based synchronization, by extracting scale-invariant features from the original video and the video under test and searching for matching frames. The advantage of this approach is the fact that the potential scale and crop of the video under test do not have to be known in order find temporal synchronization. While this works well it also has certain drawbacks. The scale-invariant feature extraction, and the subsequent matching, is slow and computationally expensive. Furthermore, it requires the original video, as opposed to the current approach, since only the original feature values are stored in the detection info file (see fig. 2). This further increases the computational demand for this method since the original video also has to be decoded. Additionally, care has to be taken to not wrongly synchronize with repeating sequences. An example of this would be a transition sequence which appears multiple times in the video under test and can match with the same transition sequence at another point in time in the video. While these problems can be handled, the resulting synchronization attempt is more complicated and time consuming than the one currently employed.

The method used for the detection of the temporal offset is based on outlier detection. Using scale-invariant-features we can calculate the difference between matching feature points. While this difference hardly ever becomes zero, due to changes in quality during re-compression of the video under test, we expect matching frames to produce a significantly lower feature distance than non-matching frames. In order to find the offset, a frame from the video under test is compared to frames in a search window of the original video. On this search window we perform outlier detection and find the best matching frame. If no outliers are found, the search windows is advanced in the original video and the process is repeated. If an outlier is detected we apply another detection with the next frame of the video under test. This has to be done in order to ascertain whether the outlier was a set of matching frames as opposed to a random statistic outlier. We assume a true match if the following N consecutive frames from the video under test also match the following N consecutive frames from the original video. This process is illustrated in fig 5.

Since not all scale-invariant feature detection approaches exhibit the same performance, a number of tests were conducted to find the most likely candidate. In order to find the best feature detector and feature point extractor pair we conducted an experiment using the SURF, ORB, FAST, STAR, HARRIS and MSER detectors and SURF, ORB and BRIEF extractors provided by the OpenCV. A test set was generated based on four short sequences to be used as original video as well as a number of expected changes, i.e. temporal crop, combined with scaling and quality reductions. The videos under test exhibit offsets of 10 or 25 frames, down-sampling to HD720 and VGA resolution (from an HD1080 original video) combined with a bit rate cap of 1024kbps and 200kbps. The experiments using the above algorithm (with a search window of 21 and

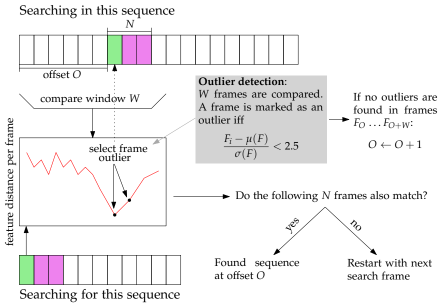


Fig. 5 Overview over the temporal shift detection using scale-invariant-based features.

Table 1 Temporal offset detection rates for various combinations of feature detectors and extractors.

[%]	Extractor		
Detector	SURF	ORB	BRIEF
SURF	100.000	84.375	84.375
ORB	75.000	68.750	59.375
FAST	84.375	87.500	90.625
STAR	56.250	56.250	50.000
HARRIS	78.125	81.250	78.125
MSER	81.250	81.250	81.250

5 required consecutive matches) produces the detection rates as shown in table 1. SURF is clearly best choice among those tested. However, for all of the detectors under test the introduction of crop, especially under low quality conditions produces faulty synchronization.

In addition to finding the correct offset in low quality, scaled and cropped videos under test there is also a systematic error which is introduced by repeating or similar sequences which can lead to a faulty offset detection. Typical examples of similar sequences are cross fades, fades to black and scene change sequences. There is no clear way to exclude these sequence except by increasing the number of necessary consecutive matches (N in the above algorithm). However, increasing the number of consecutive matches also leads to an overall lower performance when detecting temporal shift in low quality sequences.

3.2.2 Automated Detection of Spatial Displacement

The automatic detection of spatial displacement assumes a temporal alignment and tries to find the crop and scale which leads to the spatial displacement. The only other influencing factor, besides spatial changes, is the quality of the video under test.

While there is the option of using the feature points extracted for temporal synchronization to find the projection of one video into the other, experimental results showed that this is unreliable. There are instance where the number of feature points are insufficient to find a projection. Another problem is avoidance of features points which can not be matched, while this is required for some sequences it will introduce errors into others. Overall the use of extracted feature points for spatial synchronization did not consistently perform well enough.

Thus, in order to detect crop an approach based on template matching is the obvious solution. The template matching approach uses the video under test as a template and tries to find it in the original video. A direct search however is bound to produce a mismatch if scaling also affects the video under test. In order to compensate for scaling we have to do a template match with different scale factors. A list of possible scale factors, with visual examples, are given in fig. 6.

This exemplifies that we have to consider different scales when performing template matching. The scale space is hardly limited besides very one-sided scaling options, like stretching along one axis and shortening along the other. What further complicates the matter is the fact that template matching, under these transforms with the template error as distance measure does not create a convex space. This is illustrated in fig. 7 where for each scaling factor the value of the best match is given as a heat map. If the space were convex, we could perform a gradient descent search for the optimal match. However, since the space is not convex, we have to do a more complex, and consequently computationally more expensive search.

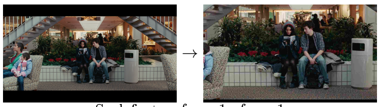
Assuming, based on the examples from fig. 6, no more than half of a picture is cut and upscaled and at most a down-sampling to VGA from HD1080 the scale space is in the range $\mathcal{S} = [0.5, 3] \times [0.5, 3]$. Assuming we utilize a search step of δ_s we can calculate the maximum number of pixels by which we will miss the correct resolution. This can be done by down-sampling with the maximum scale which is also at the largest distance from the chosen search step. The pixel difference δ_p will then be

$$\delta_p = \frac{1920}{3 - \frac{\delta_s}{2}} 3 - 1920.$$

Conversely, we can calculate δ_s for a given δ_p by

$$\delta_s = 6 - 6 \frac{1920}{\delta_p + 1920}.$$

For a negligible pixel difference, i.e. $\delta_p^N < 0.5$ such that rounding to integer produces the correct resolution, the

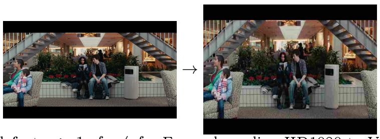


Scalefactor $fx = 1$, $fy = 1$

(a) Crop

Scalefactor $fx = 1$, $fy < 1$ or $fx < 1$, $fy = 1$

(b) Crop and upscale



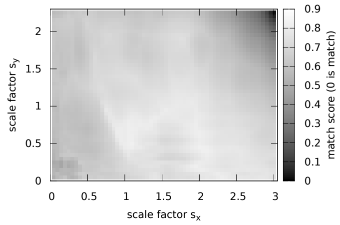
Scalefactor > 1, $fx \neq fy$; Example scaling HD1080 to VGA
$fx = 1920/640 = 3$, $fy = 1080/480 = 2.25$

(c) Scale

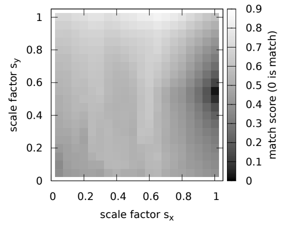
Scalefactor > 1, $fx = fy$

(d) Scale, keep aspect ratio

Fig. 6 Examples of different scale factors based on various possible spatial distortions.



(a) Scale



(b) Crop and upscale

Fig. 7 The heat map shows the matching score (lower is better) for different scales, separate for the x- and y-axis. The heat maps are for the examples Scale (fig. 6c) and Crop and upscale (fig. 6b).

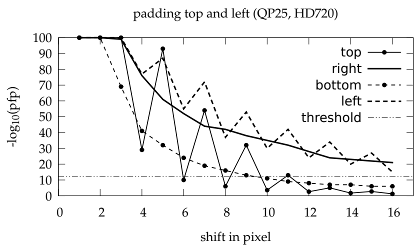


Fig. 8 Detection rate when the spatial de-synchronization in the video under test could not be correctly compensated.

resulting search step size is $\delta_s^N = 0.00156$. Searching in $\mathcal{S}$ with δ_s^N would result in over $2.5 \cdot 10^6$ template matches. For a rough comparison $2.5 \cdot 10^3$ matches take about 10 minutes. Thus, clearly this approach is not feasible.

The question is then what influence δ_p has on the detection rate, since an increase in δ_p significantly increases δ_s. We did a test with a medium quality sequence and simply shifted the video under test in the range from 1 to 16 pixels.

Figure 8 shows the result for the detection. Note that the y-axis is capped at 10^{-100}. A 6 pixel shift is the first offset where detection fails, as such the pixel error has to be significantly lower. Note that the ffmpeg library used for rescaling treats even picture sizes differently due to alignment-related optimizations, exhibiting the depicted fluctuations in the detection rate for the top and left curves.

Let us assume $\delta_p^2 = 2$, since for each of the cases the detection rate for a two pixel shift is well above the threshold. This would result in $\delta_s^2 = 0.006$ with $160 \cdot 10^3$ required matching steps, which would take almost 11 hours.

Overall, using a semiautomatic method is faster and more accurate than the fully automatic method.

3.3 Transcoding

As described in Section 2.2, the change of the state of the arithmetic coder requires reencoding. Due to these introduced changes, the positions in the bit stream where the arithmetic coder performs its renormalization may change, thus potentially changing the length of the bit stream. As the arithmetic coder is reinitialized at slice boundaries, these length changes cannot influence subsequent slices, unless they are watermarked as well.

As changes in length are not allowed, watermarked GOPs are replaced by their original, i.e., unwatermarked, versions during the merging process at the end. This way, all watermarked GOPs whose length remains unchanged are kept and the GOPs whose length changed are not watermarked. Note that this is easy to do, but lowers the embedding capacity, influencing detection later. We discuss this in detail in Section 4.2. It is also possible to preserve length at NALU level using a similar process which replaces all watermarked NALUs whose lengths differ with their original versions.

Another practical issue that has to be considered during the watermarking process involves open GOPs. An open GOP references pictures which are not contained in that GOP, as opposed to a closed GOP in which each picture can be decoded independently of pictures from other GOPs. Although open GOPs can be easily detected, they cannot be watermarked unless they are grouped together with preceding and/or subsequent closed GOPs. For the sake of simplicity, we detect and omit open GOPs from the watermarking process. Note that this potentially reduces the embedding capacity depending on the number of open GOPs. We analyze and discuss this in detail in Section 4.2.

4 Statistics and Evaluation

In this section we will evaluate two important properties discussed in previous sections.

First, the framework was designed with separate splitting and merging steps in order to utilize the context separate GOP structure for parallelization. We will show how parallelization influences the embedding process and illustrate where the bottlenecks for parallelization are.

Second, in previous sections we argued that the chosen embedding strength is sufficient to embed a high number of watermark bits even with the possible loss of potential watermarking locations due to length changes. We will give statistics about the actual occurrence of length changes and open GOPs as well as occurrence and distribution of watermark bits in an embedded stream.

4.1 Parallelization and Runtime

The QA loop performs a number of decodings of the original bit stream in order to find suitable watermarkable macroblocks. Consequently, the QA loop has high computational requirements and is slow. An example of this is given in table 2 where watermarking a 30 minute sequence takes a total of almost 12 hours. This is unsuitable for a practical application and the time requirement has to be reduced. If parallelization is possible the watermarking time can be split among a number of cores or machines and consequently reduce the overall watermarking time greatly (at the cost of computational power).

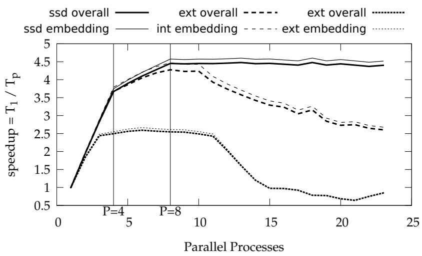
Our framework splits the H.264 bitstream into separate GOPs, performs analysis and embedding per GOP and, after sanity checks, merges the GOPs together to create the watermarked bitstream. The important part is that GOPs do not share a context, i.e., we can handle GOPs separately without interdependence on the bitstream side. Since we embed a random sequence based on a key the same concept of independent context holds for the embedded bits. Thus we can parallelize the analysis and embedding steps, which accounts for the major part of the watermarking time.

For the figures and tables in this section we used a 30 minute full HD (HD1080) subsequence of the Hancock movie. Parallelization was done on a machine with an INTEL core i7-3770 with four physical cores and eight logical cores via hyper-threading, all cores share a common L3 cache and a separate L2 and L1 cache is available per core. In order to distinguish between cache effects and tertiary storage effects on parallelization we ran the experiments twice on the same PC but with different hard disks.

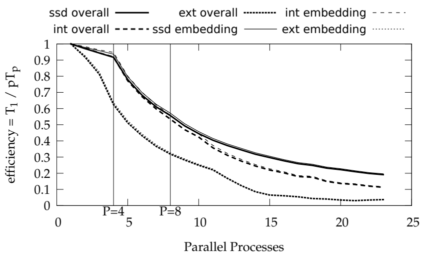
For the cache test run we used a SSD disk (denoted *ssd* where applicable), a Liteon solid state disk (LCT-256M3S) with 256 GB capacity, an average transfer rate of 324.9 MB/sec and 0.1ms average access time. Another test is done with a regular internal disk (denoted *int*), a Western Digital Caviar Blue (WD10EALX) with 1 TB capacity, an average transfer rate of 101.7 MB/sec and 16.8ms average access time. In order to show the impact of a slower disk we used a external hard disk (denoted *ext* where applicable), a Western Digital Caviar Green (WD20EARX) with 2 TB capacity, an average transfer rate of 37.2 MB/sec and 14 ms average access time. The limiting factor for the transfer rate of the external disk was the transfer speed over the USB port rather than the actual hard disk transfer rate. Throughput and access time measurements were performed with HD Tune 2.55.

Table 2 Time distribution for watermarking a 30 minute full HD sequence.

task	*ssd*			*int*			*ext*		
	time [s]	% of total	time	time [s]	% of total	time	time [s]	% of total	time
splitting	262	0.61	4:22	325	0.75	5:25	454	0.87	7:34
embedding	42849	99.30	11:54:09	43009	99.16	11:56:49	51365	99.03	14:16:05
merging	39	0.09	39	40	0.09	40	46	0.09	46
total	43151	100.00	11:59:11	43375	100.00	12:02:55	51866	100.00	14:24:26
parallel 4x	11762	27.26	3:16:02	11798	27.20	3:16:38	20765	40.03	5:46:05
parallel 8x	9691	22.46	2:41:31	10141	23.38	2:49:01	20354	39.24	5:39:14



(a) Parallelization Speedup



(b) Parallelization Efficiency

Fig. 9 Speedup and efficiency plot for parallelization with p processes on an Intel i7-3770 CPU with 4 cores and 8 logical cores (through hyper-threading).

Table 2 shows the time required for a full watermarking run and how the required time is distributed among splitting, embedding and merging. The table also shows the total time required for embedding under 4× and 8× parallelization, i.e., four or eight analysis/embedding steps are started simultaneously, the overall splitting and mering time for the parallelization processes is the same.

A more detailed overview is given in fig. 9 were the speedup and efficiency are given for a different number of parallel processes. Given are the overall time, i.e., splitting, embedding and merging combined, as well as embedding only.

If we disregard HDD limitations, i.e., the *ssd* case, it can clearly be seen that parallelization up to the number of physical cores is almost linear (efficiency > 0.9). Further parallelization up to the number of logical cores still improves overall speedup but at a lower rate, this is due to cache conflicts in the shared L2 and L1 cache between two logical cores. Parallelizing with a number of processes higher than the number of logical cores does not improve speedup.

From the *int* and *ext* cases we can see that access time is not a limiting factor for initial speedup. Even though the *int* HDD has the slowest access time it shows the same basic speedup pattern as the *ssd* case while the *ext* HDD has a tremendous impact on speedup. There is still a speedup and the overall process benefits from parallelization but when the HDD transfer limit is reached, at $P = 3$ in the figure, further parallelization does not improve overall computation speed.

Furthermore the parallelization should never use more cores than are actually present, counting logical cores. While we see in fig. 9 that utilizing more threads is not detrimental as long as the HDD is able to handle the seek time, which is easily the case for SSD disks. However, when looking at the speedup for the *ext* case it is clear that a high number of threads, and associated reads and writes, can cause a slowdown due to seek time. In the figure at $P = 11$ for the *ext* case and $P = 10$ for the *int* case showcase this stalls. Since the *int* case shows a slowdown earlier than the *ext* case this behaviour cannot be due to transfer rate. However, when looking at the average access time of the *int* and *ext* case, 16.8ms and 14ms respectively, it is clear that this slowdown is due to seek stalls during reads and writes. These seek stalls prevent the required data from reaching the worker threads leading to an overall drop in speedup, in extreme cases, e.g., $P = 21$, the speedup can drop below 1. This is a hard limit of the HDD, meaning the tertiary storage transfer rate as well as average access time limits the parallelization.

Overall, it is clear that the parallelization works well, almost a linear speedup with the number of processes used, but is limited by sharing primary memory as well as the access time and throughput of tertiary memory.

4.2 Embedding Capacity

To evaluate the embedding capacity of our watermarking approach, we used the main movies of nine different Blu-ray disks. All movies were watermarked completely, i.e., from beginning to end. The results are summarized in table 3.

We distinguish two different capacities: On the one hand, applications which require length-preservation at NALU level enforce that NALUs whose length changed during the watermarking process are replaced by their unmodified versions, i.e., the unwatermarked NALUs. This replacement reduces the number of embedded bits, leaving a total capacity denoted as "Capacity (N)". On the other hand, applications which require length-preservation at GOP level tolerate NALU-level length changes as long as the GOP length remains the same. Similar to the NALU-level length preservation, GOP-level length preservation enforces GOPs whose length changed during the watermarking process to be replaced by their unmodified versions. This replacement reduces the number of embedded bits on a GOP level, leaving a total capacity denoted as "Capacity (G)".

As NALU-level length preservation only required replacing single NALUs whose length changed during the watermarking process, it generally allows for a higher capacity than the GOP-level length preservation. The latter has to discard all bits in a GOP when its length changed, reducing the capacity significantly if the number of GOPs is low, i.e., the number of frames and therefore NALUs per GOP is high. In the examples listed in table 3 the capacity of the NALU-length preservation watermarking approach is between about 1.5 and 3 times higher than the capacity of the GOP-length preserving approach.

It is clear that the overall embedding capacity varies strongly, although several conclusions can be drawn: Firstly, movies with lots of motion, e.g., Resident Evil: Extinction, tend to have a higher capacity, whereas the opposite is true for movies with little motion, e.g., Enemy at the Gates. Secondly, movies which are longer, e.g., Gandhi with more than 270,000 frames, tend to have a higher capacity, whereas the opposite is true for short movies, e.g., Maya with less than 132,000 frames (which was to be expected). Thirdly, movies with a high percentage of non-reference B frames (denoted as b frames), e.g., 1492, tend to have a higher capacity than movies with a low percentage of b frames, e.g., Maya.

Furthermore, the distribution of watermark bits as given by the capacity in table 3 is not uniform but also depends on the structure of the video. Figure 10 illustrates this on a high capacity video (1492) and a low capacity video (Enemy at the Gates). The figure gives the average number of bits per frame calculated on a GOP basis and is plotted over the frame number, which represents the location of the capacity in the video.

However, there is another important factor which influences the embedding capacity: the existence of open GOPs. As open GOPs cannot be watermarked (see section 3.3), the potential watermarking capacity is reduced by each open GOP, therefore being lower when there is a high percentage of open GOPs. Although movies with little motion and a significant number of b frames, e.g., Enemy at the Gates, have a significantly lower capacity compared to the other movies in table 3, the number of embedded bits is still very high and allows for easy detection.

Note that the relative number of open GOPs seems to be very low, although a larger test set would be necessary in order to evaluate this in more detail. In our small test set, most movies have either no or only one open GOP, which is located at either the very beginning or the very end of the corresponding movie. Note that open GOPs at the end of a movie do not necessarily reduce the capacity as linearly scrolling credits lead to MVDs which are mostly zero and can therefore not be watermarked using our approach.

5 Conclusion

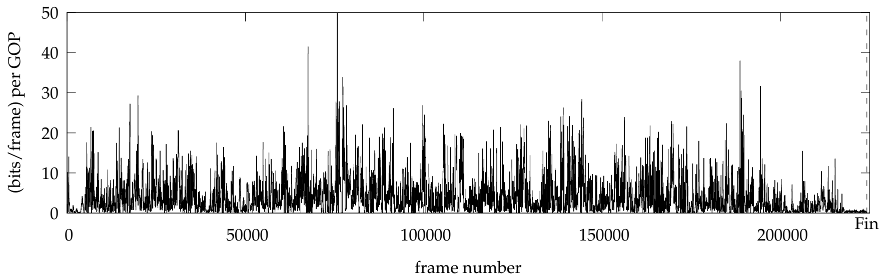
We presented a Blu-ray watermark embedding and detection framework which offers robustness to transcoding and scaling. In addition, we showed how different videos and bit stream characteristics influence the embedding capacity and run time. Furthermore, we showed that our approach is highly parallelizable subject to hard disk limitations, revealing that the hard disk's access time is as crucial for achieving maximum speedup as the hard disk's transfer rate.

From a practical point of view, we discussed that splitting the bit stream enables parallelized embedding in the first place. Furthermore, the design choice to only mark non-reference frames helps avoiding temporal drift, thereby making the quality control loop in the embedder less complex. In conclusion, we showed that the robustness and run time of our framework suffice to meet industry-level requirements.

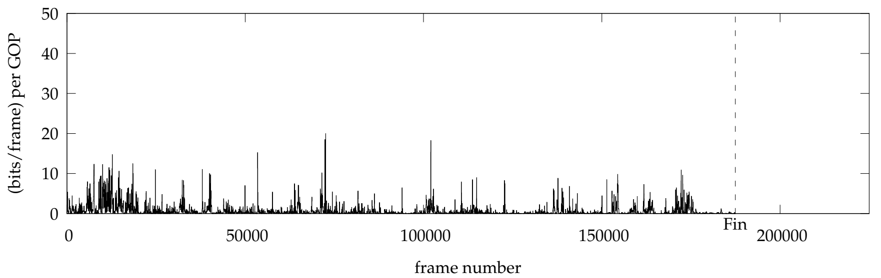
From a theoretical point of view, we gained knowledge about the requirements for an industry-level wa-

Table 3 Embedding capacity with frame and GOP statistics for a set of exemplary Blu-ray main movies

Movie name	Capacity (N)	Capacity (G)	# Frames	% b frames	#GOPs	#Open GOPs
1492	1,253,766	680,313	224,208	56.44	10,169	0
American Beauty	361,339	251,432	174,874	34.66	1,016	1
Cazzia	599,521	237,560	130,056	57.22	7,890	0
Enemy at the Gates	146,445	46,974	187,447	45.82	5,723	297
Gandhi	650,533	285,677	274,486	45.30	7,965	197
Independence Day	255,108	138,870	208,272	32.89	2,839	1
Maya	285,539	131,738	132,673	34.94	1,633	0
Resident Evil: Extinction	1,078,844	363,975	135,246	52.35	8480	0
Thor	402,095	229,693	164,920	32.36	1814	1



(a) 1492 (680,313 bits in 224,208 frames)



(b) Enemy at the Gates (46,974 bits in 130,056 frames)

Fig. 10 Location of the embedding capacity in the given videos. Note that the y and x axes are to the same scale, the dashed vertical line denotes the end of the corresponding video.

termark application. Namely, properties which are often thought of as irrelevant in science, like length changes, are important in practice since they entail complicated changes in the rest of the Blu-ray image. Other considerations which are usually treated with higher priority in science, e.g., blind watermarking, are of less or no concern. In conclusion, the design of watermarking methods should further improve the preservation of source properties, i.e., more than just format compliance, to boost applicability.

References

1. Chen, T., Liu, S., Yao, H., and Gao, W. (2006). Spatial Video Watermarking Based on Stability of DC Coefficients. In Yeung, D., Liu, Z.-Q., Wang, X.-Z., and Yan, H., editors, *Advances in Machine Learning and Cybernetics*, volume 3930 of *Lecture Notes in*

Computer Science, pages 1033–1042. Springer Berlin Heidelberg.
2. Cock, J. D., Notebaert, S., Lambert, P., and de Walle, R. V. (2010). Requantization transcoding for H.264/AVC video coding. *Signal Processing: Image Communication*, 25(4):235–254.
3. Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3, J. and ISO/IEC JTC1/SC29/WG11 (2012). High efficiency video coding (HEVC) text specification draft 8. http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=6465.
4. Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., and Kalker, T. (2007). *Digital Watermarking and Steganography*. Morgan Kaufmann.
5. Hartung, F. and Kutter, M. (1999). Multimedia watermarking techniques. In *Proceedings of the IEEE, Special Issue on Protection of Multimedia Content*, volume 87, pages 1079–1107.
6. ISO/IEC 14496-2 (2004). Information technology – coding of audio-visual objects, Part 2: Visual.
7. ITU-T H.264 (2007). Advanced video coding for generic audiovisual services. http://www.itu.int/rec/T-REC-H.264-200711-I/en.
8. Lin, E. T. and Delp, E. J. (2004). Temporal synchronization in video watermarking. *IEEE Transactions on Signal Processing*, 52(10):3007–3022.
9. Marpe, D., Schwarz, H., and Wiegand, T. (2003). Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636.
10. Sinha, R. K., Machado, F. S., and Sellman, C. (2010). Don't Think Twice, It's All Right: Music Piracy and Pricing in a DRM-Free Environment. *Journal of Marketing*, 74(2):40–54.
11. Stütz, T., Autrusseau, F., and Uhl, A. (2013). Inter-frame H.264/CAVLC structure-preserving substitution watermarking. Technical Report 2013–02, Deptartment of Computer Sciences, University of Salzburg, Salzburg, Austria. Available at http://www.cosy.sbg.ac.at/research/tr.html.
12. Stütz, T., Autrusseau, F., and Uhl, A. (2014). Non-blind structure-preserving substitution watermarking of H.264/CAVLC inter-frames. *IEEE Transactions on Multimedia*. to appear.
13. Zou, D. and Bloom, J. (2010). H.264 stream replacement watermarking with CABAC encoding. In *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME '10*, pages 117–121, Singapore.

TRANSPARENT ENCRYPTION FOR HEVC USING BIT-STREAM-BASED SELECTIVE COEFFICIENT SIGN ENCRYPTION

Heinz Hofbauer *Andreas Uhl* *Andreas Unterweger*

University of Salzburg, Jakob Haringer Str. 2,
5020 Salzburg, Austria

ABSTRACT

We propose a selective encryption scheme for HEVC which allows for transparent encryption in a wide range of quantization parameters. Our approach focusses on the AC coefficient signs, since they can be altered directly in the bit stream without entropy reencoding. This allows for fast encryption and decryption while retaining full format-compliance and length-preservation. Furthermore, we show our approach's applicability for a number of use cases by evaluating the quality degradation and robustness against attacks.

Index Terms— HEVC, transparent, encryption, bit stream, coefficient, key space

1. INTRODUCTION

We introduce an encryption scheme based on selective sign encryption of quantized transform parameters, aimed at format-compliant transparent encryption. The principal points of this encryption scheme is format compliance, i.e., the encrypted stream is decodable by a standard-compliant decoder. Digital rights management (DRM), more specifically transparent encryption, is its main field of application.

Perceptual or transparent encryption means that consumers to are able to view a preview version of the video, but in a lower quality, e.g. [1]. While preventing unauthorized consumers from accessing the full version, it is available to authorized consumers. This can be used in a pay-per-view scheme where a lower quality preview version is available from the outset to attract the viewers' interest.

Sufficient encryption aims at preventing a pleasant viewing experience, e.g. [2]. In practice, this means a reduction in quality to a point where the video is heavily distorted, but may still be recognizable. Since the content of a video is still recognizable, sufficient encryption is the middle ground between transparent encryption and *content security*, where no content should be discernible, e.g. [3].

Our proposed encryption scheme is HEVC-specific. Although numerous approaches for DCT-based video coding standards like MPEG-2 Video, MPEG-4 Part 2 and H.264 have been proposed [4, 5, 6, 7, 8, 9, 10], the latter flip all AC coefficient signs to encrypt the content, aiming at full encryption. In contrast, our approach selectively flips AC coefficient signs of the luminance channel, reducing the quality slightly, but noticeably, allowing for transparent encryption.

In addition, our approach is bit-stream based, i.e., it can be applied directly at a bit-stream level without the need to fully decode the video. Van Wallendael et al. [11] as well as Shadid and Puech [12] have investigated HEVC bit stream elements which are suitable for format-compliant bit-stream-based encryption without changes in length, one of which are AC coefficient signs. Our approach selectively encrypts the latter, allowing for transparent encryption, while retaining full format compliance and length preservation.

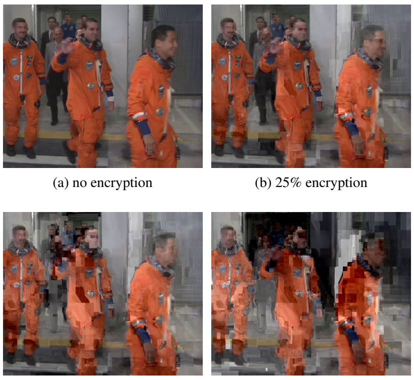
This paper contributes a new approach for transparent encryption which modifies a fixed percentage of coefficient signs in the bit stream rather than a fixed percentage of the total number of coefficients per block. Quantization parameters (QP) as well as GOP structure heavily affect the number of coefficients in the bit stream. This in turn affects the resulting quality and key space size, thus we will provide a thorough analysis of the visual quality impact and the key space size depending on encoding structure and QP.

This paper is structured as followed: In section 2, we describe our encryption approach. In section 3, we evaluate it with respect to quality and security before concluding the paper in section 4.

2. ENCRYPTION METHOD

Full sign encryption [12] is clearly in the region of sufficient encryption, but even partial sign encryption can introduce strong distortions. Therefore, we encrypt only a part of the coefficients of each block while keeping the parsing overhead minimal. Furthermore, with our approach we only encrypt sign bits in the luminance channel since the distortion introduced by encrypting chroma channels results in chromatic aberration which are more noticeable by the human visual system.

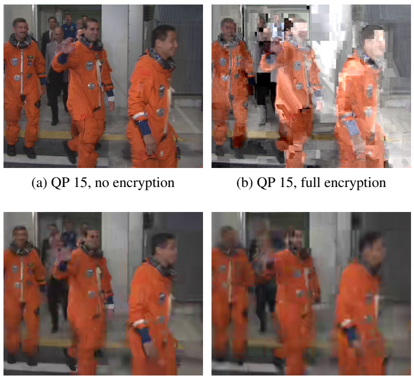
HEVC stores the coefficient signs for each block raw in the bit stream, i.e., without entropy coding. This makes it easy to manipulate them directly without impacting format compliance, while keeping the parsing overhead low. We



(a) no encryption (b) 25% encryption

(c) 50% encryption (d) 75% encryption

Fig. 1. Visual example of the sign encryption from the *crew* sequence with *randomaccess* structure and QP 21.



(a) QP 15, no encryption (b) QP 15, full encryption

(c) QP 39, no encryption (d) QP 45, no encryption

Fig. 2. Visual example of original and full sign encryption and different QP parameters.

pseudo-randomly flip a specified percentage p of signs in the bit stream. Since the scan order in HEVC is inverted, i.e., the high-frequency coefficients come first, we only encrypt the first p percent of the sign bits in the bit stream, excluding the DC coefficient sign to avoid extreme drift.

Note that coefficient signs are only stored for non-zero coefficients. Thus, encrypting p percent of the sign bits does not yield identical results to encrypting p percent of all transform coefficients (including zero coefficients). However, our proposed approach is faster as it minimizes parsing overhead and does not require any decoding.

Figure 1 shows examples of partial sign encryption. From the figure it can be seen that for transparent encryption between 25% and 50% of the signs have to be encrypted. Encrypting more than 50% of signs introduces strong distortions and results in sufficient encryption. However, even full sign encryption is not acceptable for content security, as illustrated in fig. 2 (a)–(b).

3. EVALUATION

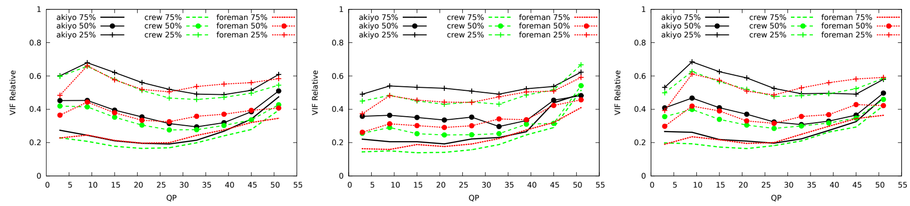
The quality analysis utilizes the visual image fidelity (VIF) image metric by Sheikh and Bovik [13]. The VIF significantly outperforms other image metrics when it comes to block based artifacts, especially in lower quality ranges, as shown by Hofbauer and Uhl [14]. In order to properly evaluate the encryption scheme, different traits of the bitstream need to be taken into account. The prediction structure has a huge influence on the propagation of the error introduced by the encryption. As such, three GOP types are used in this evaluation which reflect a variety of possible application scenarios. The GOP structures chosen are the default reference software configurations: *intra* (I frames only), *lowdelay* (one I frame, followed by groups of four P frames) and *randomaccess* (groups of one I frame followed by 32 B frames).

As a test set we chose the well known high, medium and low motion sequences, *crew*, *foreman* and *akiyo*, respectively. The different motions types were chosen because they influence prediction and the amount of coefficients for encryption.

Furthermore, we have to take into account replacement attacks [15, 16, 17, 18], which can reduce the visual distortion by replacing encrypted bitstream elements by elements which are statistically more likely to not introduce an error. Since the sign distribution is uniform, the easiest attack is to set the coefficients for which the signs are encrypted to zero. Given the differential coding nature, this will introduce less distortion than sign-flipped coefficients. In the subsequent figures, *orig* will refer to the original, i.e., unencrypted, bitstream, while *enc* will refer to the encrypted version.

The replacement attack on average increases the VIF quality by $\bar{Q}_{\text{VIF}} = 0.0306$, with a median of $\tilde{Q}_{\text{VIF}} = 0.0328$ and $\sigma_{Q_{\text{VIF}}} = 0.0199$, which results in an average quality increase by a factor of $\bar{F}_{\text{VIF}} = 1.172$, with a median of $\tilde{F}_{\text{VIF}} = 1.146$ and $\sigma_{F_{\text{VIF}}} = 0.114$. This shows that the quality increase of a replacement attack is not a security risk for this type of encryption.

Figure 3 shows the relative quality reduction of the encrypted sequence compared to the unencrypted sequence over different quantization parameters. Three behaviours, in relation to QP, can be discerned. For lower QP, there is a severe drop in quality with the same encryption type, which is more closely examined in sec. 3.2. For middle-range QP, between



(a) *intra* GOP structure (b) *lowdelay* GOP structure (c) *randomaccess* GOP structure

Fig. 3. Relative quality (VIF) of the attacked and original sequence for the given QP for 75%, 50% and 25% encryption.

15 and 40, there is a relatively stable reduction in quality, within a range of about 0.1. For higher QP, the quality drop becomes less severe.

The drop in quality reduction for high QP is influenced by the lower number of non-zero coefficients and the fact that the quality is already so low that any further impairment is not registered as strongly by the image metric, see the high QP cases without encryption in fig. 2 (c)–(d). Furthermore, given the already low quality of sequences with a QP higher than 40, further sign encryption would lead to a quality so low that it could no longer be used as a preview.

Therefore, we suggest using our encryption approach for bit streams with mid-range QP, between 15 and 40. This is also the range which is considered useful for most applications. Using on our method in this QP range lowers the quality in a way which is suitable for the described low-quality-preview scenario.

Note that the GOP structure has quasi no effect on the results at all. This can be seen from fig. 3 (a)-(c), which show very similar courses in terms of relative quality. Therefore, our approach can be used for all tested GOP structures.

3.1. Key Space Size

Figure 4 shows the average number of encrypted bits per frame. It can be seen that the key space is heavily influenced by the prediction structure of the sequence. Most key bits are compacted into I frames, since the B and P frames have a higher number of zero coefficients which are not used in sign encryption. In fig. 4 the *intra* structure has the highest number of key bits, while *lowdelay* (with only a single I frame) exhibits the lowest number of key bits. These two cases can be used as an upper and lower bound for the actual number of key bits when a different GOP structure is used, as seen in the case of *randomaccess*.

Furthermore, the number of non-zero coefficients decreases with a QP increase, consequently decreasing the number of encrypted signs and therefore the key space. This is only an apparent detrimental phenomenon since the quality in this higher QP range is already so low that any further impairment would results in sufficient rather than transparent encryption.

However, using the lower limit of the *lowdelay* GOP structure for the border case of 25% encrypted signs at QP 40, we only have about 3 bits per frame, which would result in about 720 bits for 10 second sequence at 24fps. This is clearly borderline for a security application. However, a slight increase of I frames as is the case of the *randomaccess* GOP structure, would increase this to about 25 bits per frame and consequently to 6000 bits for the same sequence.

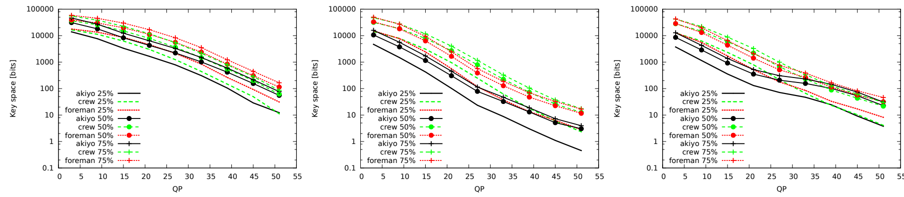
3.2. The Curious Case of High Quality Encryption

As can be seen in fig. 3, the relative quality of the encrypted sequences drops significantly at QP 3, which is against the general trend for lower QP. Hence, we analyzed the low QP (high quality) range in more detail. Figure 6 depicts the relative quality of the *foreman*, *akiyo* and *crew* sequence with the *randomaccess* GOP structure between QP 1 and 15 in steps of 1. We subsequently analyze the cause of the depicted behaviour.

For lower QP, the transform coefficient magnitudes get larger due to the smaller quantization step size. Flipping the signs of these coefficients due to encryption results in very high or very low pixel values in the picture domain, respectively. This may cause clipping to 0 or 255 for some pixels of a block. Since these clipped pixels are used for prediction, further clipping in predicted blocks is more likely to occur.

For high QP, this rarely happens and is thus negligible. The lower the QP gets, the higher the transform coefficient magnitudes get (see above) and the more likely clipping occurs, lowering the overall quality. The most extreme quality drop is at QP 4 which corresponds to a quantization step size of 1 [19]. Figure 5 illustrates this on for the foreman sequence and QP 1 to 8 for 25% encryption.

For lower QP, the encoder is more likely to bypass the transform for some blocks, i.e., it quantizes the residual pixel values directly. Since the residual values are bounded between -255 and 255, as opposed to the transform coefficients (whose magnitude may be larger), the probability of clipping in the image domain significantly decreases when flipping signs, yielding a better visual quality. Since the number of



(a) *intra* GOP structure (b) *lowdelay* GOP structure (c) *randomaccess* GOP structure

Fig. 4. Average per-frame key space of the proposed encryption methods for the given QP.

(a) QP 1 (b) QP 2 (c) QP 3 (d) QP 4 (e) QP 5 (f) QP 6 (g) QP 7 (h) QP 8

Fig. 5. Quality samples for lower QP of frame 25 of the *foreman* sequence with *randomaccess* GOP structure and 25% encryption.

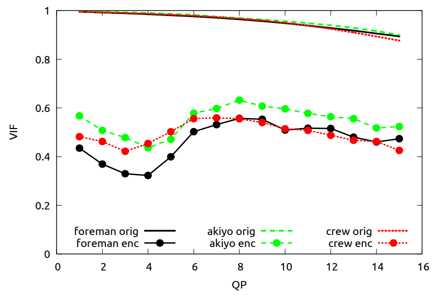


Fig. 6. Details of the low QP for the 25% encryption for the test sequences with *randomaccess* GOP structure.

blocks for which the transform is bypassed increases for lower QP, the overall quality rises again for very low QP below 4.

Due to this effect and the clipping described above, we do not recommend using our approach for very low QP, i.e., very high quality.

4. CONCLUSION

We proposed a bit-stream-based encryption approach for HEVC which pseudo-randomly flips a fixed percentage of sign bits in the bit stream. Due to its design, our approach is easy to implement and suitable for transparent and sufficient encryption, e.g., in a pay-per-view scenario for mid-range QP. We showed that the key space is sufficiently large for this application, allowing for security against attacks.

5. ACKNOWLEDGEMENTS

This work is supported by FFG Bridge project 832082.

6. REFERENCES

[1] Q. Li and I. J. Cox, "Using perceptual models to improve fidelity and provide resistance to valumetric scaling for quantization index modulation watermarking," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 2, pp. 127–139, June 2007.

[2] Thomas Stütz and Andreas Uhl, "Efficient format-compliant encryption of regular languages: Block-based cycle-walking," in *Proceedings of the 11th Joint IFIP TC6 and TC11 Conference on Communications and Multimedia Security, CMS '10*, B. De Decker and I. Schaumüller-Bichl, Eds., Linz, Austria, May 2010, vol. 6109 of *IFIP Advances in Information and Communication Technology*, pp. 81–92, Springer.

[3] Thomas Stütz and Andreas Uhl, "A survey of H.264 AVC/SVC encryption," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 3, pp. 325–339, 2012.

[4] F. Dufaux and T. Ebrahimi, "Scrambling for Anonymous Visual Communications," in *Proceedings of SPIE, Applications of Digital Image Processing XXVIII*. 2005, vol. 5909, SPIE.

[5] F. Dufaux and T. Ebrahimi, "Region-Based Transform-Domain Video Scrambling," in *Proceedings of Visual Communications and Image Processing, VCIP'06*. 2006, SPIE.

[6] F. Dufaux, M. Ouaret, Y. Abdeljaoued, A. Navarro, F. Vergnenegre, and T. Ebrahimi, "Privacy Enabling Technology for Video Surveillance," in *Proceedings of SPIE, Mobile Multimedia/Image Processing for Military and Security Applications*. 2006, vol. 6250, SPIE.

[7] Frederic Dufaux and Touradj Ebrahimi, "H.264/AVC video scrambling for privacy protection," in *Proceedings of the IEEE International Conference on Image Processing, ICIP '08*, San Diego, CA, USA, Oct. 2008, pp. 47–49, IEEE.

[8] Frederic Dufaux and Touradj Ebrahimi, "Scrambling for privacy protection in video surveillance systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1168–1174, 2008.

[9] Mourad Ouaret, Frederic Dufaux, and Touradj Ebrahimi, "Enabling Privacy For Distributed Video Coding by Transform Domain Scrambling," *SPIE Visual Communications and Image Processing*, vol. 6822, pp. E8222, Jan. 2008.

[10] Lingling Tong, Feng Dai, Yongdong Zhang, and Jintao Li, "Visual security evaluation for video encryption," in *Proceedings of the International Conference on Multimedia*, New York, NY, USA, 2010, MM '10, pp. 835–838, ACM.

[11] G. Van Wallendael, A. Boho, J. De Cock, A. Munteanu, and R. Van de Walle, "Encryption for high efficiency video coding with video adaptation capabilities," in *2013 IEEE International Conference on Consumer Electronics (ICCE)*, 2013, pp. 31–32.

[12] Zafar Shahid and William Puech, "Investigating the structure preserving encryption of high efficiency video coding (HEVC)," in *Real-Time Image and Video Processing 2013*, 2013, number 8656 in Proceedings of SPIE, pp. 86560N–86560N–10.

[13] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, May 2006.

[14] Heinz Hofbauer and Andreas Uhl, "Visual quality indices and low quality images," in *IEEE 2nd European Workshop on Visual Information Processing*, Paris, France, July 2010, pp. 171–176.

[15] M. Podesser, H.-P. Schmidt, and A. Uhl, "Selective bitplane encryption for secure transmission of image data in mobile environments," in *CD-ROM Proceedings of the 5th IEEE Nordic Signal Processing Symposium (NORSIG 2002)*, Tromso-Trondheim, Norway, Oct. 2002, IEEE Norway Section, file cr1037.pdf.

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

[17] Thomas Stütz and Andreas Uhl, "On JPEG2000 error concealment attacks," in *Advantages in Image and Video Technology: Proceedings of the 3rd Pacific-Rim Symposium on Image and Video Technology, PSIVT '09*, Tokyo, Japan, Jan. 2009, Lecture Notes in Computer Science, pp. 851–861, Springer.

[18] Heinz Hofbauer and Andreas Uhl, "Selective encryption of the MC-EZBC bitstream and residual information," in *18th European Signal Processing Conference, 2010 (EUSIPCO-2010)*, Aalborg, Denmark, Aug. 2010, pp. 2101–2105.

[19] M. Budagavi, A. Fuldseth, G. Bjontegaard, V. Sze, and M. Sadafale, "Core Transform Design for the High Efficiency Video Coding (HEVC) Standard," *IEEE Journal of Selected Topics in Signal Processing*, 2013, to appear.

Speeding Up Object Detection
Fast Resizing in the Integral Image Domain

Michael Gschwandtner, Andreas Uhl and Andreas Unterweger
Department of Computer Sciences, University of Salzburg, Jakob-Haringer-Straße 2, Salzburg, Austria
aunterweg@cosy.sbg.ac.at

Keywords:
Integral Image, Resizing, Object Detection, Performance.

Abstract:
In this paper, we present an approach to resize integral images directly in the integral image domain. For the special case of resizing by a power of two, we propose a highly parallelizable variant of our approach, which is identical to bilinear resizing in the image domain in terms of results, but requires fewer operations per pixel. Furthermore, we modify a parallelized state-of-the-art object detection algorithm which makes use of integral images on multiple scales so that it uses our approach and compare it to the unmodified implementation. We demonstrate that our modification allows for an average speedup of 6.38% on a dual-core processor with hyper-threading and 12.6% on a 64-core multi-processor system, respectively, without impacting the overall detection performance. Moreover, we show that these results can be extended to a whole class of object detection algorithms.

1 INTRODUCTION

Integral images, initially developed under the name "summed-area tables" (Crow, 1984), have regained a lot of attention since Viola and Jones proposed an object detection framework (Viola and Jones, 2001) which makes heavy use of them. Popular implementations of this framework, such as the OpenCV library (Willow Garage, 2012), perform object detection on multiple scales, i.e., they resize the original image multiple times and run the detection algorithm on each resized image, also referred to as scale. Although the object detection is relatively fast due to the use of integral images, the need to recompute the integral image for each scale impacts the performance significantly. Therefore, in this paper, we propose a new algorithm which allows resizing the integral images themselves, i.e., in the integral image domain instead of the image domain, omitting the need to recompute the integral image for each scale.
Despite efforts to speed up the computation of integral images in general (Hensley et al., 2005) as well as on different architectures like GPUs (Bilgic et al., 2010), literature on integral images is sparse. While Crow (Crow, 1984) initially described how to perform simple operations, like, e.g., blurring, in the integral image domain, Heckbert (Heckbert, 1986) generalized the underlying theory, thereby extending its scope to arbitrary filters with polynomial kernels. Hussein (Hussein et al., 2008) improved and extended Heckbert's work by enabling non-uniform filtering. Although both frameworks, Heckbert's and Hussein's, allow to perform resizing operations, they take input from the integral image domain and produce output in the image domain. In contrast, our approach performs all operations directly in the integral image domain, hereby omitting the need to recompute the integral image after resizing.
Although algorithms performing operations directly in the integral image domain have been proposed (e.g., (Yu et al., 2010) for histogram thresholding), none of them changes the size of the integral image itself, as opposed to the algorithm we propose. As performing operations on multiple scales in the integral image domain is part of several state-of-the-art algorithms, such as SURF (Bay et al., 2008), local binary patterns (LBP) (Ahonen et al., 2004) and Viola's and Jones' framework for object detection as ex-

plained above, our main contribution of resizing in the integral image domain inherently allows speeding up algorithms relying on the computation of integral images on multiple scales. Note that, although some algorithms allow scaling up the features instead of scaling down the integral images, a significant number of implementations (Willow Garage, 2012) recompute the integral images and thus profit from our contribution.
This paper is structured as follows: In section 2, we propose an algorithm for resizing in the integral image domain without distortions by imposing certain restrictions on the resizing factor. In section 3, we extend this algorithm to support arbitrary resizing factors, albeit at the cost of negligible distortions. After evaluating our algorithm in section 4 in terms of performance, quality and parallelizability, we conclude our paper in section 5.

2 EXACT RESIZING

In the following sections we describe how a given integral image can be resized. We distinguish between exact and approximate resizing, where exact means that each pixel of the resized integral image is identical to the corresponding pixel of an integral image which is calculated from a bilinearly resized version of the original image, the resizing process of which has been performed in the image domain.

2.1 Integral Images

An integral image II of a given image I represents the sum of all its pixels from the top-left corner to every pixel, excluding the column and row of the pixel (note that some definitions include the pixel's column and row, requiring corresponding changes in the subsequent formulas). Hence, it is calculated as (Willow Garage, 2012)

$$II(x,y) = \sum_{x'=0}^{x-1} \sum_{y'=0}^{y-1} I(x',y') \tag{1}$$

This allows calculating the sum S of all pixels within a rectangular area R in constant time (Crow, 1984) as

$$S_R = II(x_r,y_b) - II(x_l,y_b) - II(x_r,y_t) + II(x_l,y_t) \tag{2}$$

where x_l, x_r, y_t and y_b are R's left, right, top and bottom coordinates, respectively, as depicted in figure 1. Note that, in order to reconstruct single

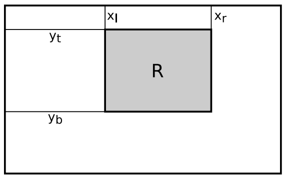


Figure 1: Use of an integral image for summing all pixels within a rectangular area R constrained by its coordinates x_l, x_r, y_t and y_b. Adopted from (Crow, 1984)

pixels from the integral image (see next section for details), the integral image's dimensions are $(w+1)\cdot(h+1)$ if the original image's dimensions are $w \cdot h$.

2.2 Naïve Resizing

Resizing algorithms usally perform operations in the image domain, i.e., on the image's pixels. As becomes clear from equation (2), it is possible to extract every single pixel as a rectangle of width and height one of the original image I from its integral image II:

$$\begin{aligned} I(x,y) =& II(x,y) + II(x+1,y+1) - \\ & II(x,y+1) - II(x+1,y) \end{aligned} \tag{3}$$

Therefore, it is theoretically possible to implement any image-domain-based resizing filter in the integral image domain by filtering using on-the-fly extraction of the original image's pixels and subsequent calculation of the resized image's integral image.
However, this is computationally more expensive as accessing each pixel requires four operations in the integral image domain as opposed to one in the image domain. Furthermore, it is necessary to access locations in the integral image which are one row apart in order to derive a single pixel of the original image. This may cause a higher number of the CPU's cache lines to be occupied, if the integral image is stored sequentially in memory.

2.3 Resizing By A Power Of Two

In the following section we propose a resizing algorithm for integral images which eliminates the need to extract the original image's pixels from the integral image in a computationally expensive way. However, for the algorithm to work exactly, the resizing factor needs to be a power of two.

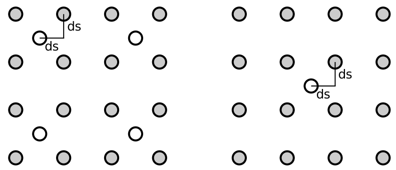


Figure 2: Resizing by a power of two using a special case of bilinear interpolation where the interpolated samples (white) have the same distance ds to all surrounding original samples (gray).

Note that we discuss ways to circumvent this restriction in section 3.
Consider the following, simplified resizing scenario: A given image I with width w and height h, where both, w and h, are even, is to be resized by a factor of two in each dimension, yielding the image I_h with width $\frac{w}{2}$ and height $\frac{h}{2}$. Using bilinear interpolation as depicted in figure 2 (left), the samples of I (gray) are used to determine the samples of I_h (white) as:

$$I_h(x,y) = \frac{1}{4} \cdot (I(2x,2y) + I(2x+1,2y) + I(2x,2y+1) + I(2x+1,2y+1)) \quad (4)$$

The integral image II_h at position $(0 \leq x \leq \frac{w}{2}, 0 \leq y \leq \frac{h}{2})$ of I_h can then be calculated by

$$II_h(x,y) = \sum_{x'=0}^{x-1} \sum_{y'=0}^{y-1} I_h(x',y') \quad (5)$$

which can be expanded to

$$\frac{1}{4} \cdot \sum_{x'=0}^{x-1} \sum_{y'=0}^{y-1} I(2x',2y') + I(2x'+1,2y') + I(2x',2y'+1) + I(2x'+1,2y'+1) \quad (6)$$

This can be rewritten as

$$II_h(x,y) = \frac{1}{4} \cdot \sum_{x'=0}^{2x-1} \sum_{y'=0}^{2y-1} I(x',y') \quad (7)$$

where the summand can subsequently be expressed as a sample of the integral image II of the original image I:

$$II_h(x,y) = \frac{1}{4} \cdot II(2x,2y) \quad (8)$$

Note that this equation only depends on the original image's integral image II and has no dependency to the original image I. Furthermore, it trivially allows repeated application (e.g., twice for a resizing factor of four as illustrated in figure 2 (right)), thereby enabling resizing by arbitrary powers of two. As can be easily shown, a given integral image II can be resized by a factor of 2^n in each dimension to an integral image II_n as

$$II_n(x,y) = \frac{1}{2^{2n}} \cdot II(2^n x, 2^n y) \quad (9)$$

Based on this observation, we formulate our approach to resize integral images as follows: An integral image can be resized by a power of two with bilinear interpolation using only one single integral image sample per calculated sample using equation (9). Note that the latter assumes both, the corresponding image's width and height, to be integer multiples of 2^n. For all other cases, resizing cannot be performed exactly at the integral image's borders. However, the handling of these borders in approximate form is described in section 3.2.

3 APPROXIMATE RESIZING

In order to overcome the limitations of the resizing approach proposed in the previous section in terms of image dimensions and resizing factors, we present an extension which can deal with arbitrary resizing factors and image borders. Nonetheless, this extended approach is largely based on the limited approach presented in the previous section.

3.1 Resizing Arbitrarily

In this section we explain how to extend equation (8) in order to support arbitrary resizing factors. We do so by splitting the formula into two parts – the factor in front of the sum and the sum itself. By modifying each of them separately, we derive an equation which can be used to resize arbitrarily in the integral image domain.
When resizing by a factor two in each dimension, we observe that the factor of $\frac{1}{4}$ in front of the sum in equation (8) corresponds to the inverse of the combined (i.e., multiplied) resizing factor. Simply put, each pixel of the resized image covers an area of 4 pixels in the original image, as depicted in figure 3 (left). This is equivalently true for the corresponding integral image pixels.
Extending this observation to arbitrary resizing factors, henceforth denoted as $2a$, it is obvious that each pixel of the resized image now covers an

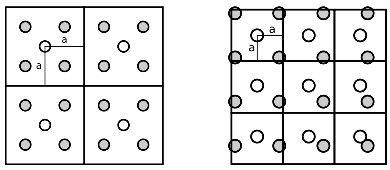


Figure 3: Resized (integral) image samples (white) covering a constant area of $4a^2$ (black rectangle) in the original (integral) image (gray samples) when resizing by a factor of $2a = 2$ (left) and $2a = 1.16$ (right), respectively. For both, the resizing filter offset $b = 0.5$ samples (see equation (10)).

area of $2a \cdot 2a = 4a^2$ square pixels in the original image. Figure 3 (right) depicts this for a resizing factor of $2a = 1.16$, where the covered area is $4a^2 = 1.3456$ square pixels. Note that each pixel of the resized image is located at the center of the area it covers in the original image, i.e., its distance to each side of the rectangle enclosing this area is a.

Although changing the factor in equation (8) to $\frac{1}{4a^2}$ does not introduce an error, the required corresponding change of each summand does. Replacing $II(2x+1, 2y+1)$ by $II(2ax+b, 2ay+b)$ (where $-a < b < a$ denotes a offset corresponding to the desired resizing filter phase, which is typically constant for all samples) is not possible in general, as a is not necessarily an integer.

Therefore, we suggest performing bilinear interpolation in the integral image domain in order to get an approximation of the virtual pixel at position $II(2ax+b, 2ay+b)$ based on the values of the surrounding integral image pixels. Note that this approximation introduces a small error compared to the bilinear interpolation in the image domain. This error is estimated empirically in section 4.2.

Summarizing the above modifications to equation (8), an integral image II can be resized by a factor of $2a$ in both dimensions in the integral image domain in the same (mathematical) way as in the image domain, i.e., by bilinear interpolation. Doing so yields a resized integral image II_r which can be calculated by

$$\begin{aligned} II_r(x,y) &\approx \frac{1}{4a^2} \cdot bilinear(II, (2ax+b, 2ay+b)) \\ &= \frac{1}{4a^2} \cdot \left(\begin{bmatrix} 1-dx & dx \end{bmatrix} \begin{bmatrix} i_{tl} & i_{bl} \\ i_{tr} & i_{br} \end{bmatrix} \begin{bmatrix} 1-dy \\ dy \end{bmatrix} \right) \end{aligned} \tag{10}$$

for all values of x and y except the borders (see section 3.2 for details), where

$$\begin{aligned} & i_{tl} = II(x', y'), i_{tr} = II(x'+1, y') \\ & i_{bl} = II(x', y'+1), i_{br} = II(x'+1, y'+1) \\ & x' = \lfloor 2ax + b \rfloor, y' = \lfloor 2ay + b \rfloor \\ & dx = 2ax + b - x', dy = 2ay + b - y' \end{aligned} \tag{11}$$

The value of b has to be chosen according to the desired filter phase as explained above. Note that equation (10) uses the same formula for bilinear interpolation as any comparable algorithm in the image domain would. The only difference is that the latter operates on the image's pixels, while the former operates on the integral image's.

3.2 Handling Of Borders

For positive b, the rightmost column and the bottommost row can, in most cases, not be calculated by equation (10) as non-existing samples of the original integral image, i.e., samples whose coordinates are larger than the image's width and/or height, respectively, would have to be accessed. For negative b, the same applies to the leftmost column and the topmost row.

The latter case ($b < 0$) is trivial to handle: All samples can be set to zero as the first row and column of an integral image is by definition (see equation 1) zero. The former case ($b > 0$) can be handled in a way similar to the approach described in section 3.1. While the area covered by the integral image pixels at the border is as large as the area covered by the other integral image pixels, the unavailability of pixels beyond the border requires linear interpolation of the border pixels instead of full bilinear interpolation. Resizing at the right border $x = x_r$ can be performed by calculating

$$\begin{aligned} II_r(x_r, y) &\approx \frac{1}{4a^2} \cdot linear(II, (2ax_r + b, 2ay + b)) \\ &= \frac{1}{4a^2} \cdot ((1-dy) \cdot II(\lfloor 2ax_r + b \rfloor, y') + \\ &\qquad dy \cdot II(\lfloor 2ax_r + b \rfloor, y')) \end{aligned} \tag{12}$$

where $y' = \lfloor 2ay + b \rfloor$ and $dy = 2ay + b - y'$. Resizing at the bottom border is equivalent for constant $y = y_b$ and variable x. In case the bottom-rightmost pixel cannot be calculated by one of these formulas, it can be approximated without interpolation by

$$II_r(x_r, y_b) \approx \frac{1}{4a^2} \cdot II(\lfloor 2ax_r + b \rfloor, \lfloor 2ay_b + b \rfloor). \tag{13}$$

4 EVALUATION

In order to assess the speed, quality and parallelizability of our approach, we created three different implementations in three different languages. Firstly, we created a CUDA program for power-of-two resizing to show the achievable degree of parallelism resulting from the reduced number of memory accesses in this special case (for details see section 4.1). Secondly, we implemented arbitrary resizing in Python including OpenCV's resizing capabilities for comparison to show the quality difference, i.e., the error induced by our approximation. Finally, we modified OpenCV's LBP based (Ahonen et al., 2004) object detection algorithm to use our resizing approach to show the latter's performance and practical use.
All tests were carried out on an Intel Core 2 Duo E6700 desktop system with an NVIDIA GeForce 8500 GT graphics card running Ubuntu 11.10 64-bit, unless noted otherwise. We used version 2.4.3 of OpenCV with support for the Intel Thread Building Blocks (TBB) library (version 4.1 Update 1).

4.1 Parallelizability

For the special case of resizing by a power of two in each dimension, our algorithm for exact bilinear interpolation (see equation (8)) requires fewer memory accesses per sample to be calculated (one) than classical bilinar interpolation in the image domain does (four, see equation (4)). Hence, our approach is not slower than classical bilinear interpolation. Additionally, each sample requires a different source integral image sample to be calculated from. Therefore, each sample can be calculated completely independently, allowing for massive parallelization.
Furthermore, if the desired output of the resizing operation is an integral image, classical bilinear interpolation has to be followed by an integral image calculation which is hard to parallelize efficiently, while this is not the case with our approach as its output is another integral image. Thus, a resizing operation with an integral image as final result can be parallelized more easily when using our approach.
To show the latter's parallelizability, we created a straight-forward, unoptimized GPU implementation for resizing an integral image by a factor of two in each dimension in which each image sample is resized by a separate thread calculating

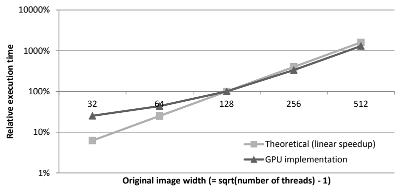


Figure 4: Speedup over number of threads when resizing integral images by a factor of two in each dimension: theoretical linear speedup (light gray rectangles) vs. actual implementation's speedup (dark gray triangles).

equation (9). Given a 32 bits per sample integral image with dimensions $(w+1)\cdot(w+1)$, where w is a power of two, our implementation spawns $(\frac{w}{2}+1)\cdot(\frac{w}{2}+1)$ threads on the GPU for calculating the resized integral image.
Note that we did not use the GPU's built-in bilinear resizer in order to keep the implementation as simple as possible. Since the main aim of our implementation is to demonstrate parallelizability, this does not affect the results. Since using the built-in bilinear resizer would only speed up the filtering operation in terms of consumed clock cycles per processed group of pixels, it only differs from the straight-forward implementation by a constant multiplicative factor, which vanishes when considering relative speedup values.
Our implementation's net execution time, i.e., the actual computation time on the GPU, is measured using CUDA events (using CUDA version 4.0 bundled with driver version 304.43). In order to avoid the influence of caching effects, before each actual measurement, the GPU kernel is executed three times for cache warming. Subsequently, the actual kernel is executed five times. The average time of these five executions is used to represent the actual net execution time.
Figure 4 depicts the relative net execution time of our resizing approach and a theoretical linear speedup representing the performance of an ideal algorithm with one constant time memory access per sample for comparison. The x axis denotes the values of w, while the y axis denotes the speedup relative to the execution time of our GPU implementation's performance for $w = 128$ which is the medium measurement point.
As can be seen, the speedup is nearly ideal for larger image dimensions. Although a small overhead remains compared to the theoretically achievable speedup, this is to be expected due to

the GPU's internal thread scheduling overhead. For smaller image dimensions, the measurements fluctuate significantly due the small number of threads to be executed. Benefitting from the GPU's ability to let multiple threads access the memory at the same time under certain conditions, the achievable speedup for a small number of threads is higher than the simplified theoretical limit and has therefore to be rated with care. However, for a large number of threads this effect becomes relatively small and can therefore be disregarded.

4.2 Quality

For arbitrary resizing as described in section 3.1, the quality degradation, i.e., the error introduced by our approach as compared to resizing in the image domain, needs to be assessed. To do so, we use the LIVE (Seshadrinathan et al., 2010) reference picture set and process each picture I in the following way. Firstly, I is resized bilinearly to I_{ref} with OpenCV to serve as a reference. Secondly, OpenCV is used to compute the integral image of I, followed by applying our algorithm for resizing in the integral image domain and subsequently reconstructing the resized image I_{new} using equation (3). Finally, both, I_{new} and I_{ref}, are upsampled with nearest neighbour interpolation using OpenCV to fit I's dimensions and compared to the original image I to determine the respective differences.

Table 1 summarizes the minimum, maximum and average PSNR differences for each resizing factor. Hereby, positive values mean that our approach's PSNR is higher than OpenCV's, while the converse is true for negative values. Although the absolute gap between the minimum and maximum PSNR difference for each factor is not very high in general (around 2dB on average), a factor-dependent trend regarding the average difference can be seen.

While our approach achieves a higher PSNR for large resizing factors (greater than 6.2), the converse is true for small resizing factors (less than 3.1). If little actual interpolation is required (e.g., for factors like 1.0, 2.0 or 3.0), the PSNR differences are smallest on average. Conversely, they amount to up to 4 dB for quasi-pathological cases like resizing factors of 1.9.

Although this may seem relatively high, thorough investigation shows that high differences are mainly caused by sub-sample shifts of the image introduced by our algorithm. Interpolating in the integral image domain partly "moves" the area associated with each interpolated column and row to their corresponding neighbours, thereby introducing a sub-pixel shift when reconstructing the image. As this augments the error signal, the PSNR increases. Assuming that most practical applications are not affected by shifts of this magnitude, the quality difference between our bilinear resizer and OpenCV's can be considered acceptably small.

4.3 Performance

As state-of-the art object detection algorithms make heavy use of integral images on multiple scales as explained in section 1, we modified one of them – OpenCV's LBP based object detection algorithm – as an example. Note that this can be done for other multi-scale integral-image-based object detection algorithms in a similar fashion, making the subsequent results applicable to them as well.

While OpenCV's original LBP detector implementation resizes the input image in the image domain and computes its integral image on each scale (see figure 5 left), our modification uses the integral image of the original image and resizes it in the integral image (II) domain (see figure 5 right). The actual detection operations on the resized integral images remain unchanged. However, our modification does not require the integral images to be computed at each scale. Note that this theoretically allows discarding the input picture as soon as the first integral image is calculated. This can save a significant amount of memory, e.g., on embedded systems, when the input image is not needed otherwise. As the default resizing factor used by OpenCV is 1.1 per scale in each dimension, our approximate resizing approach described in section 3.1 is used.

In order to assess the influence of our resizing approach on object detection performance, we trained the LBP detector with OpenCV's face detection training data set and a negative data set from `http://note.sonots.com/SciSoftware/haartraining.html`. Subsequently, we assessed its detection performance using the four CMU/MIT frontal face test sets from `http://vasc.ri.cmu.edu/idb/images/face/frontal_images`. The test data set includes eye, nose and mouth coordinates for each face in each of the 180 pictures. A face is considered detected if and only if all of the aforementioned coordinates are within one of the rect-

Table 1: Minimum, maximum and average PSNR differences between our approximate resizing approach and OpenCV's bilinear resizer over all pictures of the LIVE data base (Seshadrinathan et al., 2010) for different resizing factors F. All PSNR difference values are in dB.

F	MIN	MAX	AVG	F	MIN	MAX	AVG	F	MIN	MAX	AVG
1.00	-0.00	-0.00	-0.00	**4.10**	-1.78	0.92	-0.46	**7.20**	-0.70	1.83	0.74
1.10	-3.14	-0.24	-1.93	**4.20**	-2.32	0.57	-0.61	**7.30**	-0.78	1.94	0.96
1.20	-3.14	0.26	-1.57	**4.30**	-1.15	1.10	-0.01	**7.40**	-0.44	1.58	0.68
1.30	-3.77	-1.29	-2.78	**4.40**	-1.13	0.75	-0.03	**7.50**	0.52	2.00	1.28
1.40	-3.18	-1.42	-2.62	**4.50**	-1.68	0.83	-0.28	**7.60**	-0.50	1.76	1.01
1.50	-3.10	0.16	-1.10	**4.60**	-1.49	1.06	-0.07	**7.70**	-0.77	1.37	0.62
1.60	-2.71	0.53	-0.04	**4.70**	-1.60	1.29	-0.07	**7.80**	-0.10	1.46	0.75
1.70	-2.95	-0.84	-1.82	**4.80**	-0.75	1.65	0.50	**7.90**	-0.50	1.57	0.63
1.80	-2.47	-1.29	-1.94	**4.90**	-2.14	0.95	-0.25	**8.00**	-0.47	1.86	1.48
1.90	-4.06	-0.89	-2.02	**5.00**	-0.52	2.35	0.83	**8.10**	-0.84	1.75	0.75
2.00	-1.82	-0.00	-0.25	**5.10**	-0.84	1.39	0.49	**8.20**	-0.81	1.54	0.66
2.10	-3.45	-1.73	-2.58	**5.20**	-0.86	1.11	0.34	**8.30**	-1.04	1.25	0.37
2.20	-3.32	0.21	-1.59	**5.30**	-1.46	1.35	-0.13	**8.40**	-1.20	1.65	0.47
2.30	-3.17	-0.57	-2.03	**5.40**	-0.97	1.25	0.07	**8.50**	-0.53	1.77	0.98
2.40	-2.17	0.58	-0.53	**5.50**	-1.36	1.42	0.40	**8.60**	-1.08	1.39	0.56
2.50	-2.87	0.62	-1.11	**5.60**	-0.74	1.34	0.55	**8.70**	-0.53	1.67	0.63
2.60	-3.35	-0.07	-1.93	**5.70**	-1.29	1.32	0.13	**8.80**	-0.86	1.90	0.99
2.70	-3.06	0.53	-1.46	**5.80**	-0.10	1.59	0.80	**8.90**	-1.13	1.49	0.65
2.80	-2.48	-0.23	-1.33	**5.90**	-0.62	1.56	0.44	**9.00**	-0.83	2.10	0.81
2.90	-2.41	0.24	-1.07	**6.00**	-0.26	1.48	0.96	**9.10**	-0.10	1.70	1.08
3.00	-1.57	1.82	-0.08	**6.10**	-1.57	1.40	-0.15	**9.20**	-0.33	1.77	0.90
3.10	-1.85	0.32	-0.68	**6.20**	-0.98	1.39	0.35	**9.30**	-0.04	1.84	1.04
3.20	-1.94	1.20	0.52	**6.30**	-0.72	1.48	0.52	**9.40**	-0.87	1.67	0.83
3.30	-1.98	0.94	-0.32	**6.40**	-0.17	1.83	1.17	**9.50**	-0.93	1.48	0.42
3.40	-2.21	-0.00	-0.96	**6.50**	-1.24	1.10	0.21	**9.60**	-0.22	2.09	1.42
3.50	-1.51	0.65	-0.19	**6.60**	-1.10	1.82	0.52	**9.70**	-1.07	1.91	0.84
3.60	-1.41	0.81	0.13	**6.70**	-0.31	1.32	0.50	**9.80**	-0.54	1.90	1.02
3.70	-1.92	0.43	-0.55	**6.80**	-1.26	1.57	0.33	**9.90**	-0.69	1.81	0.65
3.80	-1.77	0.32	-0.39	**6.90**	-0.22	1.83	0.84	**10.00**	-0.64	1.90	1.13
3.90	-1.77	0.91	-0.27	**7.00**	-1.21	1.60	0.56				
4.00	-1.07	1.08	0.72	**7.10**	-0.73	1.98	1.10				

angles returned by the LBP based detector. The detection rate is determined as the ratio of the number of detected faces to the total number of faces.

In total, the detection rate does not change, i.e., both, OpenCV's and our modification's, detection rates are exactly the same, namely 46.19%. It should be noted that not all detected faces coincide completely, i.e., the detected rectangles differ slightly due to the sub-pixel shift introduced by our approach on smaller scales as explained in section 4.2. In addition, 15% of all pictures exhibit differences in the number of detected faces, which is mainly due to fact that the detector's training was performed using regularly resized training data. We conjecture that, when using our resizing approach during training as well, the aforementioned differences will possibly vanish.

In order to assess the performance gain of our modification in terms of execution time in a fair way, we do not perform execution time measurements for our unoptimized modification and the highly optimized original OpenCV code. Instead, we deduce the performance gain as follows: As our resizing approach in the integral image domain is identical to bilinear interpolation in the image domain in terms of operations (see section 3.1), our modification does not impact the resizing speed. Conversely, as our approach does not require integral image calculations at any scale but the first (see above), the remaining integral image calculations do not need to be performed. Therefore, the execution time of these integral image calculations relative to the detector's to-

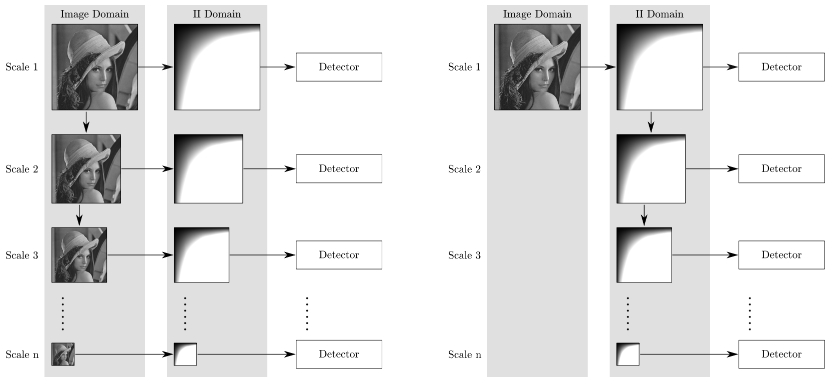


Figure 5: Illustration and comparison of OpenCV's multi-scale LBP detector (left) and our modification of it (right). By using the proposed integral image resizing approach, our modification does not require the recalculation of the integral images on each scale.

tal execution time is equivalent to the potential speedup of our approach compared to the current OpenCV implementation.

For the accurate measurement of single functions' execution times, `rdtsc` (Intel, 2012) commands are placed before and after the corresponding function calls inside the OpenCV code. We use the aforementioned CMU/MIT test set and execute the detector a total of 110 times for each image – ten times for cache warming and 100 times for the actual time measurement as described in section 4.1. To address the question of scalability, two additional test systems with comparable software configurations are used for this evaluation: a mobile system (henceforth referred to as system B) with an Intel Core i5 540M CPU with two physical cores capable of hyper-threading, i.e., a total of four virtual CPU cores, and a server system (henceforth referred to as system C) with 4 AMD Opteron 6274 CPUs with 16 cores each, i.e., a total of 64 physical CPU cores.

The results vary strongly depending on three parameters: the image size, the image content and the number of available CPU cores. The former two parameters influence the number of actual resizing operations being performed, yielding different speedups for different picture sizes and content types. As the default resizing factor per scale is 1.1 in each dimension, larger images with big objects to be detected exhibit a larger speedup than smaller images do. Similarly, when large image areas without detectable objects are present, the speedup is greater as more execution time is spent in the integral image calculation routines than in the actual detector due to the LBP cascades on each scale terminating quickly.

The influence of the third and most influential parameter, i.e., the number of available CPU cores, is summarized in table 2. It shows that the default test system with two CPU cores (referred to as system A) spends on average 4.64% of the detector's execution time on the described integral image calculations, which is equivalent to an average speedup of 4.64% of our proposed modification compared to the existing OpenCV implementation (see above). Using the four virtual cores of test system B, the speedup increases to an average of 6.38%. This is due to the fact that the integral image calculations cannot be parallelized efficiently, while the converse is true for most of the detector's other code parts. Thus, our proposed modification using our integral image based resizing approach provides better scalability. This becomes even clearer when considering test system C with 64 total CPU cores with an average and maximum speedup of 12.6% and 37.25%, respectively.

Note that our modification also allows speeding up the overall detection process when TBB support in OpenCV is disabled, i.e., when only one CPU core is actually used. This can be seen in the corresponding row in table 2 which reveals an average speedup of 2.9% in this case. However, it is not recommended to only use one CPU core

Table 2: Dependency of integral image calculation time at lower scales of the LBP detector on the number of available CPU cores n. All values are relative to the detector's total execution time for the respective hardware and software configuration.

n	SYS	AVG	SDEV	MIN	MAX
1	A*	2.9%	0.71%	1.57%	6.78%
2	A	4.66%	0.66%	2.92%	6.89%
4	B**	6.38%	0.78%	4.38%	9.87%
64	C	12.6%	4.86%	4.21%	37.25%

* TBB support disabled
** 2 cores with hyper-threading

when more are available as a higher number of CPU cores increases the speedup significantly as shown above.

5 CONCLUSION

We proposed a new approach for image resizing which works entirely in the integral image domain. For the special case of power-of-two resizing, we presented a highly parallelizable version of our approach which requires only a quarter of the operations compared to regular bilinear interpolation in the image domain, but provides the same exact results. Furthermore, we evaluated the practicality of our general approach by modifying one of multiple state-of-the-art multi-scale integral-image-based object detection algorithms in OpenCV without degrading its detection performance. In total, a speed-up of an average of 6.38% and 12.6% could be achieved on a dual-core mobile computer and a multi-processor server, respectively. Moreover, we showed that similar results can be achieved for all multi-scale integral-image-based object detection algorithms.

ACKNOWLEDGEMENTS

This work is supported by FFG Bridge project 832082.

REFERENCES

Ahonen, T., Hadid, A., and Pietikäinen, M. (2004). Face Recognition with Local Binary Patterns. In Pajdla, T. and Matas, J., editors, *Computer Vision – ECCV 2004*, volume 3021 of *Lecture Notes in Computer Science*, pages 469–481. Springer Berlin Heidelberg.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359.

Bilgic, B., Horn, B. K., and Masaki, I. (2010). Efficient integral image computation on the GPU. In *2010 IEEE Intelligent Vehicles Symposium (IV)*, pages 528–533, San Diego, CA, USA. IEEE.

Crow, F. C. (1984). Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 207–212, New York, NY, USA. ACM.

Heckbert, P. S. (1986). Filtering by repeated integration. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 315–321, New York, NY, USA. ACM.

Hensley, J., Scheuermann, T., Coombe, G., Singh, M., and Lastra, A. (2005). Fast Summed-Area Table Generation and its Applications. *Computer Graphics Forum*, 24(3):547–555.

Hussein, M., Porikli, F., and Davis, L. (2008). Kernel integral images: A framework for fast non-uniform filtering. In *IEEE Conference on Computer Vision and Pattern Recognition 2008 (CVPR 2008)*, pages 1–8, Anchorage, AK, USA. IEEE.

Intel (2012). Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z. `http://www.intel.com/Assets/PDF/manual/253667.pdf`.

Seshadrinathan, K., Soundararajan, R., Bovik, A., and Cormack, L. (2010). Study of Subjective and Objective Quality Assessment of Video. *IEEE Transactions on Image Processing*, 19(6):1427–1441.

Viola, P. and Jones, M. (2001). Robust Real-time Object detection. In *International Journal of Computer Vision*, volume 57, pages 137–154.

Willow Garage (2012). OpenCV. `http://opencv.willowgarage.com/`.

Yu, C., Dian-ren, C., Xu, Y., and Yang, L. (2010). Fast Two-Dimensional Otsu's Thresholding Method Based on Integral Image. In *2010 International Conference on Multimedia Technology (ICMT)*, pages 1–4, Ningbo, China. IEEE.

4. Errata

"I always tell the truth. Even when I lie." – Tony Montana, *Scarface*

Since the papers in Chapter 3 are printed as (originally) published, this chapter lists errata for those papers for which there are any.

4.1. Errata for *Length-preserving Bit-stream-based JPEG Encryption*

The following corrections are required in the paper printed in Section 3.4:

- The residue range is not between 0 and $-2^s + 1$, but between $-2^s + 1$ and $2^s - 1$ (first paragraph of section 2).

4.2. Errata for *Bitstream-based JPEG Encryption in Real-time*

The following corrections are required in the paper printed in Section 3.5:

- The residue range is not between 0 and $-2^s + 1$, but between $-2^s + 1$ and $2^s - 1$ (first paragraph of the *Bitstream Encryption* section).

4.3. Errata for *Speeding Up Object Detection – Fast Resizing in the Integral Image Domain*

The following corrections are required in the paper printed in Section 3.13:

- The formula in the x axis label of the execution time graph should not be *sqrt(number of threads) - 1*, but *2*(sqrt(number of threads) - 1)* (Figure 4).
- The argument *As can be seen, the speedup is nearly ideal for larger image dimensions. Although a small overhead remains compared to the theoretically achievable speedup, this is to be expected due to the GPU's internal thread scheduling overhead.* is incorrect and should be *As can be seen, the speedup is ideal for larger image dimensions. Benefitting from the GPU's ability to let multiple threads access the memory at the same time under certain conditions, the achievable speedup for a large number of threads is even higher than the simplified theoretical limit.* (second-to-last paragraph of section 4.1).
- The argument *Benefitting from the GPU's ability to let multiple threads access the memory at the same time under certain conditions, the achievable speedup for a small number of threads is higher than the simplified theoretical limit and has therefore to be rated with care.* is incorrect and should be *This is to be expected due to the GPU's internal thread scheduling overhead.* (last paragraph of section 4.1).

5. Conclusion

"Everything that has a beginning has an end." – The Oracle, *The Matrix Revolutions*

Post-compression multimedia security is a narrow subject area with a fragile use-case specific balance between advantages and disadvantages. When designed and implemented carefully and correctly, post-compression multimedia security approaches outperform pre- and in-compression approaches in many regards, such as processing time and space overhead. Furthermore, post-compression approaches often allow achieving properties such as length or structure preservation that would be very hard or even impossible to accomplish otherwise. However, this comes at the price of higher design and implementation complexity.

The papers forming this thesis covered the subjects of region of interest encryption, watermarking and transparent encryption. In each subject, the state of the art was extended with respect to post-compression processing and its restrictions. Several advances could be achieved and a number of problems solved.

Two length-preserving region of interest encryption approaches for JPEG have been proposed. For H.264 and its scalable extension, auxiliary techniques which help facilitating region of interest encryption have been described. Together with a region of interest encryption approach for H.264, the construction of a nearly drift-free post-compression region of interest encryption approach for H.264 and its scalable extension is possible.

A structure-preserving watermarking approach for H.264 bit streams has been proposed which allows tracing leaks in the Blu-ray production process without the need to re-encode or alter any other part of the Blu-ray disk or its data. For H.265, a structure-preserving transparent encryption approach has been described which can be used for current and future Pay TV broadcasts. While offering the possibility to transmit a partially encrypted version of the broadcast video for non-paying customers, it can be transmitted without re-compression or other changes in broadcasting equipment.

Moreover, advances in areas related to the three main subjects have been reported. The face detection approach by Viola and Jones has been sped up by an algorithmic simplification involving the properties of integral images to enable faster face detection for region of interest encryption. A time- and space-effective encoding and signalling approach for region of interest coordinates has been described for the same use case, being the first fully documented and evaluated algorithm of its kind. Finally, a framework for video surveillance systems that combines signalling and encryption at bit stream level has been developed and evaluated.

In summary, significant progress has been made in the three main subjects – region of interest encryption, watermarking and transparent encryption. However, even though some very challenging problems, such as drift minimization for post-compression region of interest encryption in H.264 and its scalable extension, have been solved in the papers contained in this thesis, many more challenges and open research questions remain.

In particular, the question of how to eliminate all temporal drift without re-compression during region of interest encryption still remains unanswered. With the increasing complexity of compression standards, finding answers also becomes more difficult. But as long as there are use cases which rely on or at least greatly benefit from post-compression multimedia security approaches, this subject area will remain actively researched and answers to even more difficult questions will eventually be found.

Bibliography

[Acken, 1998] Acken, J. M. (1998). How watermarking adds value to digital content. *Communications of the ACM*, 41(7):75–77.

[Auer et al., 2013] Auer, S., Bliem, A., Engel, D., Uhl, A., and Unterweger, A. (2013). Bitstream-Based JPEG Encryption in Real-time. *International Journal of Digital Crime and Forensics*, 5(3):1–14.

[Barni et al., 1998] Barni, M., Bartolini, F., Cappellini, V., and Piva, A. (1998). A DCT-domain system for robust image watermarking. *Signal Processing, Special Issue on Copyright Protection and Control*, 66(3):357–372.

[Blu-ray Disc Association, 2011] Blu-ray Disc Association (2011). White paper Blu-ray Disc Read-Only Format. 2.B Audio Visual Application Format Specifications for BD-ROM Version 2.5. `http://blu-raydisc.com/assets/Downloadablefile/BD-ROM-AV-WhitePaper_110712.pdf`.

[Boult, 2005] Boult, T. E. (2005). PICO: Privacy through invertible cryptographic obscuration. In *IEEE/NFS Workshop on Computer Vision for Interactive and Intelligent Environments*, pages 27–38, Lexington, KY, USA.

[Carrillo et al., 2009] Carrillo, P., Kalva, H., and Magliveras, S. (2009). Compression Independent Reversible Encryption for Privacy in Video Surveillance. *EURASIP Journal on Information Security*, 2009:1–13.

[Chrysafis et al., 1999] Chrysafis, C., Taubman, D., and Drukarev, A. (1999). Overview of JPEG2000. In *The IS&T Image Processing, Image Quality, Image Capture, Systems Conference, PICS '99*, pages 333–338, Savannah, GA, USA.

[Cox et al., 2007] Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., and Kalker, T. (2007). *Digital Watermarking and Steganography*. Morgan Kaufmann.

[De Cock et al., 2014] De Cock, J., Hofbauer, H., Stütz, T., Uhl, A., and Unterweger, A. (2014). An Industry-Level Blu-ray Watermarking Framework. *Multimedia Tools and Applications*, pages 1–23.

[De Cock et al., 2010] De Cock, J., Notebaert, S., Lambert, P., and de Walle, R. V. (2010). Requantization transcoding for H.264/AVC video coding. *Signal Processing: Image Communication*, 25(4):235–254.

[Digital Video Broadcasting (DVB), 2014] Digital Video Broadcasting (DVB) (2014). Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications. Part II: S2-Extensions (DVB-S2X) - (Optional). `https://www.dvb.org/resources/public/standards/a83-2_dvb-s2x_den302307-2.pdf`. Also published as ETSI EN 302 307.

[Dufaux and Ebrahimi, 2006] Dufaux, F. and Ebrahimi, T. (2006). Region-Based Transform-Domain Video Scrambling. In *Proceedings of Visual Communications and Image Processing, VCIP'06*. SPIE.

[Dufaux and Ebrahimi, 2008a] Dufaux, F. and Ebrahimi, T. (2008a). H.264/AVC video scrambling for privacy protection. In *Proceedings of the IEEE International Conference on Image Processing, ICIP '08*, pages 47–49, San Diego, CA, USA. IEEE.

[Dufaux and Ebrahimi, 2008b] Dufaux, F. and Ebrahimi, T. (2008b). Scrambling for privacy protection in video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1168–1174.

[Dufaux et al., 2006] Dufaux, F., Ouaret, M., Abdeljaoued, Y., Navarro, A., Vergnenegre, F., and Ebrahimi, T. (2006). Privacy Enabling Technology for Video Surveillance. In *SPIE Mobile Multimedia/Image Processing for Military and Security Applications*, Lecture Notes in Computer Science. IEEE.

[Dufaux et al., 2004] Dufaux, F., Wee, S., Apostolopoulos, J., and Ebrahimi, T. (2004). JPSEC for secure imaging in JPEG2000. In Tescher, A. G., editor, *Applications of Digital Image Processing XXVII*, volume 5558, pages 319–330. SPIE.

[Engel et al., 2009] Engel, D., Stütz, T., and Uhl, A. (2009). A survey on JPEG2000 encryption. *Multimedia Systems*, 15(4):243–270.

[Engel et al., 2013] Engel, D., Uhl, A., and Unterweger, A. (2013). Region of Interest Signalling for Encrypted JPEG Images. In *IH&MMSec'13: Proceedings of the 1st ACM Workshop on Information Hiding and Multimedia Security*, pages 165–174. ACM.

[Gonzalez and Woods, 2007] Gonzalez, R. C. and Woods, R. E. (2007). *Digital Image Processing*. Prentice Hall, 3rd edition.

[Gschwandtner et al., 2014] Gschwandtner, M., Uhl, A., and Unterweger, A. (2014). Speeding Up Object Detection – Fast Resizing in the Integral Image Domain. In *VISAPP 2014 – Proceedings of the 9th International Conference on Computer Vision Theory and Applications*, volume 1, pages 64–72, Lisbon, Portugal. SciTePress.

[Hartung and Kutter, 1999] Hartung, F. and Kutter, M. (1999). Multimedia Watermarking Techniques. *Proceedings of the IEEE*, 87(7):1079–1107.

[Hofbauer, 2013] Hofbauer, H. (2013). *Visual Evaluation, Scaling and Transport of Secure Videos*. PhD thesis, University of Salzburg, Department of Computer Sciences.

[Hofbauer et al., 2014] Hofbauer, H., Uhl, A., and Unterweger, A. (2014). Transparent Encryption for HEVC Using Bit-Stream-Based Selective Coefficient Sign Encryption. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1986–1990, Florence, Italy. IEEE.

[ISO/IEC 14496-10, 2005] ISO/IEC 14496-10 (2005). Information technology – coding of audio-visual objects – part 10: Advanced video coding.

[ITU-T H.264, 2007] ITU-T H.264 (2007). Advanced video coding for generic audiovisual services. `http://www.itu.int/rec/T-REC-H.264-200711-I/en`.

[ITU-T H.265, 2013] ITU-T H.265 (2013). High efficiency video coding. `http://www.itu.int/rec/T-REC-H.265-201304-I`.

[ITU-T T.81, 1992] ITU-T T.81 (1992). Digital compression and coding of continuous-tone still images — requirements and guidelines. Also published as ISO/IEC IS 10918-1.

[Li et al., 2014] Li, B., Li, H., Li, L., and Zhang, J. (2014). λ Domain Rate Control Algorithm for High Efficiency Video Coding. *IEEE Transactions on Image Processing*, 23(9):3841–3854.

[Lin et al., 2011] Lin, H., Yang, S., and Xu, L. (2011). Watermark algorithm for color image authentication and restoration. In *2011 International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT)*, volume 6, pages 2773–2776.

[Lin and Li, 2011] Lin, Y.-C. and Li, J.-H. (2011). Reversible watermarking for H.264/AVC videos. *World Academy of Science, Engineering and Technology*, 78:828–831.

[Luo, 2010] Luo, M. (2010). *Robust and Blind 3D Watermarking*. PhD thesis, Department of Computer Science, University of York, York, UK.

[Macq and Quisquater, 1995] Macq, B. M. and Quisquater, J.-J. (1995). Cryptology for digital TV broadcasting. *Proceedings of the IEEE*, 83(6):944–957.

[Martin et al., 2008] Martin, V., Chabert, M., and Lacaze, B. (2008). An interpolation-based watermarking scheme. *Signal Processing*, 88(3):539–557.

[Massoudi et al., 2008] Massoudi, A., Lefèbvre, F., Vleeschouwer, C. D., Macq, B., and Quisquater, J.-J. (2008). Overview on selective encryption of image and video, challenges and perspectives. *EURASIP Journal on Information Security*, 2008(Article ID 179290):doi:10.1155/2008/179290, 18 pages.

[Meerwald and Uhl, 2012] Meerwald, P. and Uhl, A. (2012). An efficient robust watermarking method integrated in H.264/SVC. *Transactions on Data Hiding and Multimedia Security VII*, 7110:1–14.

[Niu et al., 2008] Niu, X., Zhou, C., Ding, J., and Yang, B. (2008). JPEG Encryption with File Size Preservation. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing 2008 (IIHMSP '08)*, pages 308–311.

[O'Ruanaidh and Pun, 1998] O'Ruanaidh, J. J. K. and Pun, T. (1998). Rotation, scale and translation invariant spread spectrum digital image watermarking. *Signal Processing*, 66(3):303–317.

[Ouaret et al., 2008] Ouaret, M., Dufaux, F., and Ebrahimi, T. (2008). Enabling Privacy For Distributed Video Coding by Transform Domain Scrambling. *SPIE Visual Communications and Image Processing*, 6822:E8222.

[Pranata et al., 2004] Pranata, S., Guan, Y. L., and Chua, H. C. (2004). Improved bit rate control for real-time MPEG watermarking. In *Proceedings of the IEEE International Conference on Image Processing, ICIP '04*, pages 2619–2623, Singapore. IEEE.

[Rahman et al., 2010] Rahman, S. M. M., Hossain, M. A., Mouftah, H., Saddik, A. E., and Okamoto, E. (2010). A real-time privacy-sensitive data hiding approach based on chaos cryptography. In *Proc. of IEEE International Conference on Multimedia & Expo*, pages 72–77, Suntec City, Singapore.

[Schneier, 2000] Schneier, B. (2000). *Secret and Lies*. John Wiley & Sons, first edition.

[Schulzrinne et al., 1998] Schulzrinne, H., Rao, A., and Lanphier, R. (1998). Real Time Streaming Protocol (RTSP). RFC 2326. `http://www.ietf.org/rfc/rfc2326.txt`.

[Schwarz et al., 2007] Schwarz, H., Marpe, D., and Wiegand, T. (2007). Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120.

[Senior et al., 2005] Senior, A., Pankanti, S., Hampapur, A., Brown, L., tian, Y.-L., Ekin, A., Connell, J., Shu, C. F., and Lu, M. (2005). Enabling Video Privacy through Computer Vision. *IEEE Security and Privacy*, 3(3):50–57.

[Stütz et al., 2013] Stütz, T., Autrusseau, F., and Uhl, A. (2013). Inter-frame H.264/CAVLC structure-preserving substitution watermarking. Technical Report 2013–02, Deptartment of Computer Sciences, University of Salzburg, Salzburg, Austria. Available at `http://www.cosy.sbg.ac.at/research/tr.html`.

[Stütz et al., 2010] Stütz, T., Pankajakshan, V., Autrusseau, F., Uhl, A., and Hofbauer, H. (2010). Subjective and objective quality assessment of transparently encrypted JPEG2000 images. In *Proceedings of the ACM Multimedia and Security Workshop (MMSEC '10)*, pages 247–252, Rome, Italy. ACM.

[Su and Kuo, 2001] Su, P.-C. and Kuo, C.-C. J. (2001). An integrated approach to image watermarking and JPEG-2000 compression. *Journal of VLSI Signal Processing*, 27(1):35–53.

[Sullivan et al., 2013] Sullivan, G., Boyce, J., Chen, Y., Ohm, J.-R., Segall, C., and Vetro, A. (2013). Standardized Extensions of High Efficiency Video Coding (HEVC). *IEEE Journal of Selected Topics in Signal Processing*, 7(6):1001–1016.

[Sullivan et al., 2012] Sullivan, G., Ohm, J., Han, W.-J., and Wiegand, T. (2012). Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668.

[Tong et al., 2010a] Tong, L., Dai, F., Zhang, Y., and Li, J. (2010a). Prediction restricted H.264/AVC video scrambling for privacy protection. *Electronic Letters*, 46(1):47–49.

[Tong et al., 2010b] Tong, L., Dai, F., Zhang, Y., and Li, J. (2010b). Visual security evaluation for video encryption. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 835–838, New York, NY, USA. ACM.

[Unterweger, 2013] Unterweger, A. (2013). Compression Artifacts in Modern Video Coding and State-of-the-Art Means of Compensation. In Farrugia, R. A. and Debono, C. J., editors, *Multimedia Networking and Coding*, chapter 2, pages 28–49. IGI Global, Hershey.

[Unterweger et al., 2015a] Unterweger, A., De Cock, J., and Uhl, A. (2015a). Bit-Stream-Based Encryption for Regions of Interest in H.264/AVC Videos With Drift Minimization. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. submitted.

[Unterweger and Uhl, 2012] Unterweger, A. and Uhl, A. (2012). Length-preserving Bit-stream-based JPEG Encryption. In *MM&Sec'12: Proceedings of the 14th ACM Multimedia and Security Workshop*, pages 85–89. ACM.

[Unterweger and Uhl, 2014a] Unterweger, A. and Uhl, A. (2014a). Slice groups for post-compression region of interest encryption in H.264/AVC and its scalable extension. *Signal Processing: Image Communication*, 29(10):1158–1170.

[Unterweger and Uhl, 2014b] Unterweger, A. and Uhl, A. (2014b). Slice Groups for Post-Compression Region of Interest Encryption in SVC. In *IH&MMSec'14: Proceedings of the 2014 ACM Information Hiding and Multimedia Security Workshop*, pages 15–22, Salzburg, Austria. ACM.

[Unterweger et al., 2015b] Unterweger, A., Van Ryckegem, K., Engel, D., and Uhl, A. (2015b). Building a Post-Compression Region-of-Interest Encryption Framework for Existing Video Surveillance Systems – Challenges, obstacles and practical concerns. *Multimedia Systems*. submitted.

[Vetro et al., 2003] Vetro, A., Christopoulos, C., and Sun, H. (2003). Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, 20(2):18–29.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Robust Real-time Object detection. In *International Journal of Computer Vision*, volume 57, pages 137–154.

[Wang and Kwong, 2008] Wang, H. and Kwong, S. (2008). Rate-Distortion Optimization of Rate Control for H.264 With Adaptive Initial Quantization Parameter Determination. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(1):140–144.

[Westwater and Furth, 1997] Westwater, R. and Furth, B. (1997). *Real-time Video Compression*. Kluwer Academic Publishers.

[Wiegand et al., 2003] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576.

[Woods et al., 2005] Woods, J. W., Cheng, P., Wu, Y., and Hsiang, S.-T. (2005). Interframe subband/wavelet scalable video coding. In Bovik, A., editor, *Handbook of Image and Video Processing*, Communications, Networking, and Multimedia, chapter 6.2, pages 799–817. Academic Press, 2nd edition.

[Wu and Wu, 1997] Wu, T.-L. and Wu, S. F. (1997). Selective encryption and watermarking of MPEG video (extended abstract). In Arabnia, H. R., editor, *Proceedings of the International Conference on Image Science, Systems, and Technology, CISST '97*, Las Vegas, USA.

[Xin et al., 2005] Xin, J., Lin, C.-W., and Sun, M.-T. (2005). Digital Video Transcoding. *Proceedings of the IEEE*, 93(1):84–97.

A. Breakdown of Authors' Contributions

"Am I to be reading for all eternity?" – Loki, *Thor: The Dark World*

This chapter lists all authors of the papers included in this thesis and provides a breakdown of the respective contributions per person per paper in percent. Empty positions indicate zero (0), i.e., no contributions. Since the explicit contribution of my advisor, Andreas Uhl, cannot be stated for a single paper, it is omitted in the following breakdown.

The order of the papers is identical to the order of the papers in Chapter 3. Note that author names on most papers are **not** listed in order of their relative contributions.

	Contribution (in %)								
Publication	Stefan Auer	Alexander Bliem	Jan De Cock	Dominik Engel	Michael Gschwandtner	Heinz Hofbauer	Thomas Stütz	**Andreas Unterweger**	Kevin Van Ryckegem
Unterweger, A. (2013). Compression Artifacts in Modern Video Coding and State-of-the-Art Means of Compensation. In Farrugia, R. A. and Debono, C. J., editors, *Multimedia Networking and Coding*, chapter 2, pages 28–49. IGI Global, Hershey.								**100**	
Unterweger, A. and Uhl, A. (2012). Length-preserving Bit-stream-based JPEG Encryption. In *MM&Sec'12: Proceedings of the 14th ACM Multimedia and Security Workshop*, pages 85–89. ACM.								**100**	
Auer, S., Bliem, A., Engel, D., Uhl, A., and Unterweger, A. (2013). Bitstream-Based JPEG Encryption in Real-time. *International Journal of Digital Crime and Forensics*, 5(3):1–14.	10	10		5				**75**	
Engel, D., Uhl, A., and Unterweger, A. (2013). Region of Interest Signalling for Encrypted JPEG Images. In *IH&MMSec'13: Proceedings of the 1st ACM Workshop on Information Hiding and Multimedia Security*, pages 165–174. ACM.				10				**90**	

Publication	Contribution (in %)								
	Stefan Auer	Alexander Bliem	Jan De Cock	Dominik Engel	Michael Gschwandtner	Heinz Hofbauer	Thomas Stütz	**Andreas Unterweger**	Kevin Van Ryckegem
Unterweger, A., Van Ryckegem, K., Engel, D., and Uhl, A. (2015b). Building a Post-Compression Region-of-Interest Encryption Framework for Existing Video Surveillance Systems – Challenges, obstacles and practical concerns. *Multimedia Systems*. submitted.				5				**90**	5
Unterweger, A., De Cock, J., and Uhl, A. (2015a). Bit-Stream-Based Encryption for Regions of Interest in H.264/AVC Videos With Drift Minimization. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. submitted.			10					**90**	
Unterweger, A. and Uhl, A. (2014a). Slice groups for post-compression region of interest encryption in H.264/AVC and its scalable extension. *Signal Processing: Image Communication*, 29(10):1158–1170.								**100**	
Unterweger, A. and Uhl, A. (2014b). Slice Groups for Post-Compression Region of Interest Encryption in SVC. In *IH&MMSec'14: Proceedings of the 2014 ACM Information Hiding and Multimedia Security Workshop*, pages 15–22, Salzburg, Austria. ACM.								**100**	
De Cock, J., Hofbauer, H., Stütz, T., Uhl, A., and Unterweger, A. (2014). An Industry-Level Blu-ray Watermarking Framework. *Multimedia Tools and Applications*, pages 1–23.			10			30	30	**30**	
Hofbauer, H., Uhl, A., and Unterweger, A. (2014). Transparent Encryption for HEVC Using Bit-Stream-Based Selective Coefficient Sign Encryption. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1986–1990, Florence, Italy. IEEE.						30		**70**	

Publication	Contribution (in %)								
	Stefan Auer	Alexander Bliem	Jan De Cock	Dominik Engel	Michael Gschwandtner	Heinz Hofbauer	Thomas Stütz	**Andreas Unterweger**	Kevin Van Ryckegem
Gschwandtner, M., Uhl, A., and Unterweger, A. (2014). Speeding Up Object Detection – Fast Resizing in the Integral Image Domain. In *VISAPP 2014 – Proceedings of the 9th International Conference on Computer Vision Theory and Applications*, volume 1, pages 64–72, Lisbon, Portugal. SciTePress.					10			**90**	