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)

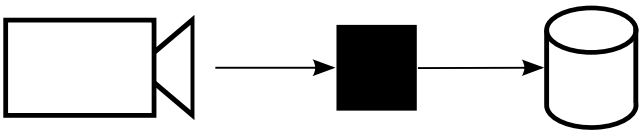

Fig. 2 Black-box encryption: The compressed video stream captured by a camera (left) is encrypted by the black box (middle) before being stored in a database (right)

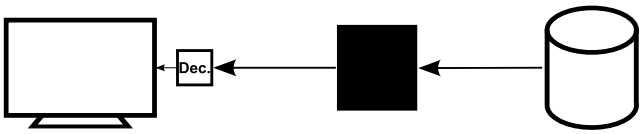

Fig. 3 Black-box decryption: The compressed video stream retrieved from a database (right) is decrypted by the black box (middle) before being decoded and displayed on a screen (left)

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

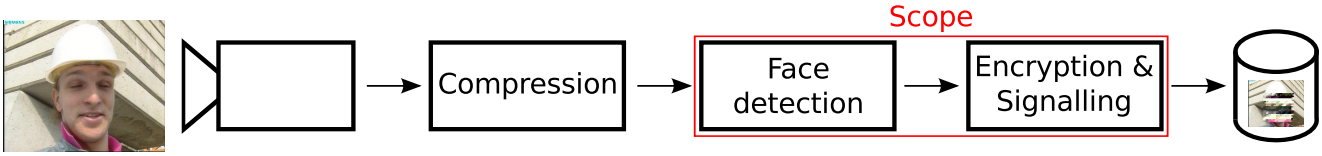

Fig. 1 Video surveillance system with post-compression RoI encryption functionality: Captured images are compressed, faces are detected, encrypted and signalled, and the encrypted images are stored in a data base. The focus of this paper are those components which are not present in regular, i.e., non-encrypting,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

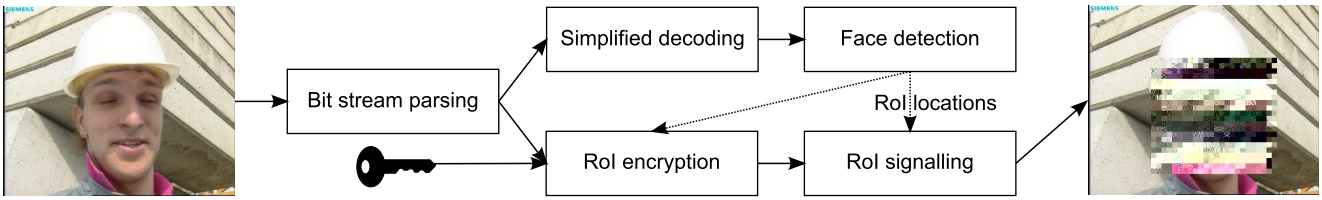

Fig. 4 Components of the encryption black box: An unencrypted input picture is parsed and decoded for face detection. The obtained face coordinates are used for limiting the encryption to said regions and written to the encrypted output file as they are required later for decryption

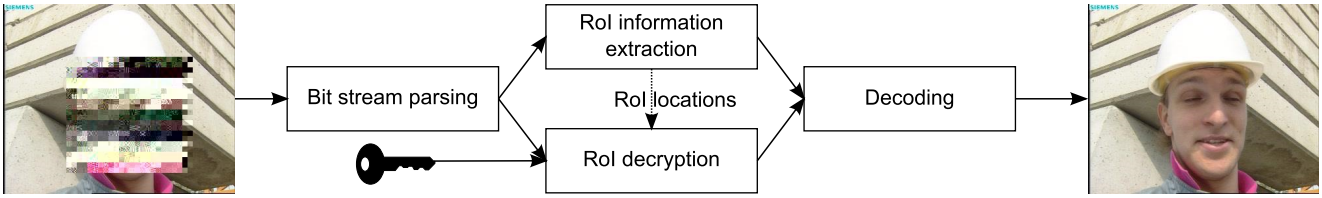

Fig. 5 Components of the decryption black box: An encrypted input picture is parsed, but not decoded. The RoI coordinates are extracted from the input file and used for limiting the decryption to said regions, resulting in an unencrypted output picture

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

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

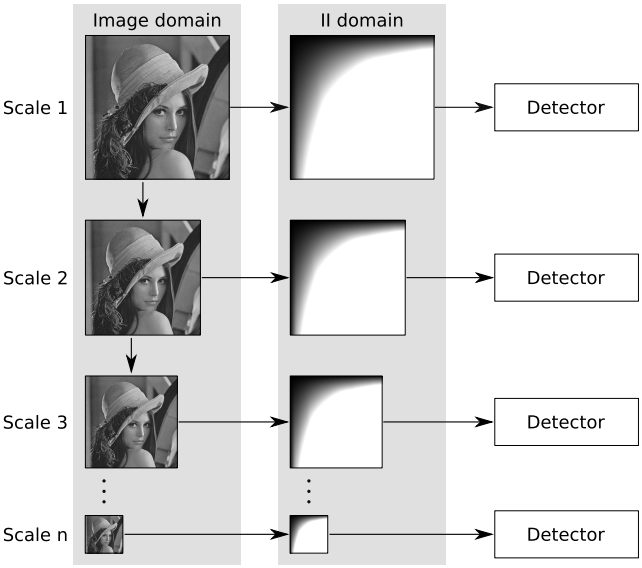

Fig. 6 Multi-scale face detection: The input image is scaled in the image domain. On each scale, an integral image (II) is calculated, followed by face detection using a detector cascade

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

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.

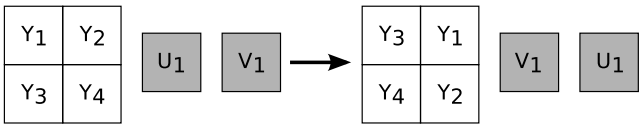

Fig. 7 Block order permutation by Auer et al. [1]: Blocks of the same color use the same Huffman code words. Their order is changed pseudo-randomly in the first AC coefficient encryption step

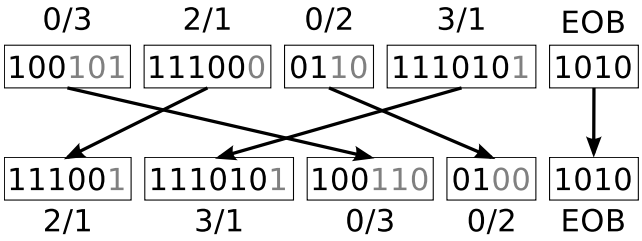

Fig. 8 Combined code-word-value order permutation and value scrambling adopted from Auer et al. [1]: The order of Huffman code words (black) representing run-length/value information and their associated values (grey) is changed pseudo-randomly in the second AC coefficient encryption step; the value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

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.

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]

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

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]

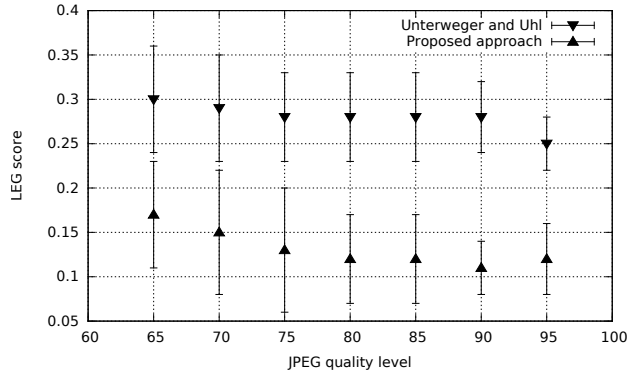

Fig. 12 LEG scores for the *LIVE* reference picture set [35] after a replacement attack on our proposed encryption approach vs. the approach by Unterweger and Uhl [40]. Error bars denote standard deviation

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.

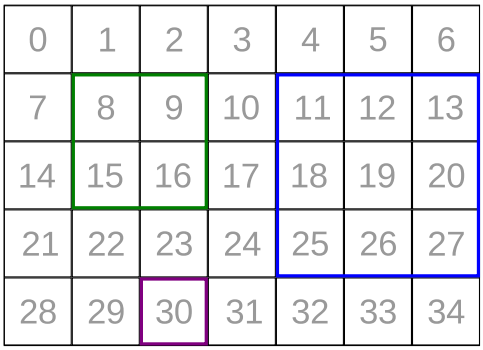

Fig. 13 Block indices for RoI signalling with three exemplary RoI: Block indices increase from left to right and top to bottom

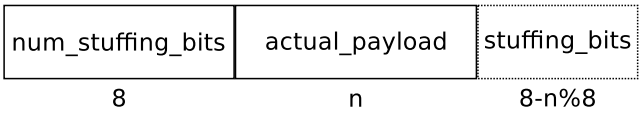

Fig. 14 `COM` segment payload extension: If the length n of the actual payload is not a multiple of a byte, stuffing bits are added. Their number is signalled before the start of the actual payload

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`

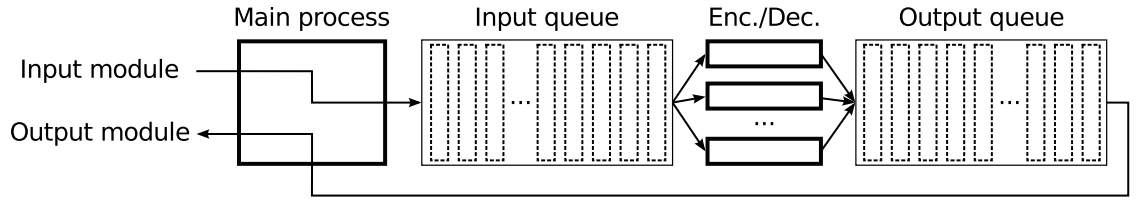

Fig. 15 Parallelization: The main process puts the data of images to be processed (dashed) into an input queue, from which the worker processes take one at the time. The processed data of each encoded or decoded image is put into an output queue, from which it is retrieved by the main process

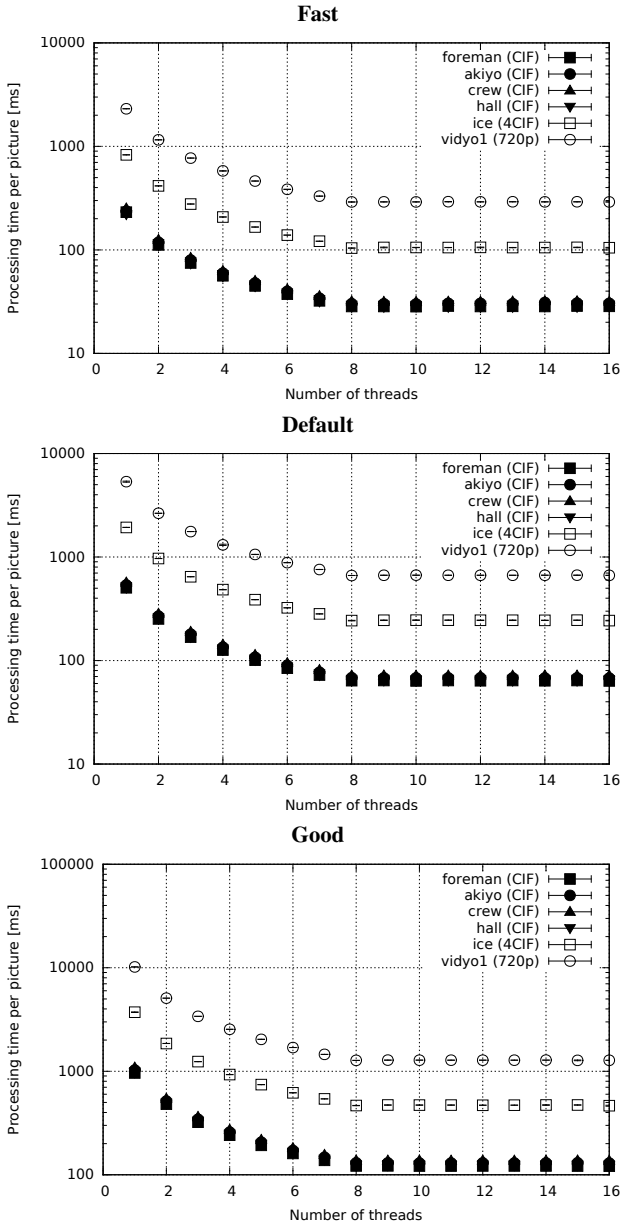

Fig. 16 Encoding times with different face detection configurations: Fast (top), default (middle), good (bottom). The y axis of the good configuration uses a different offset than the other two, illustrating that it is approximately a power of ten slower than the fast configuration

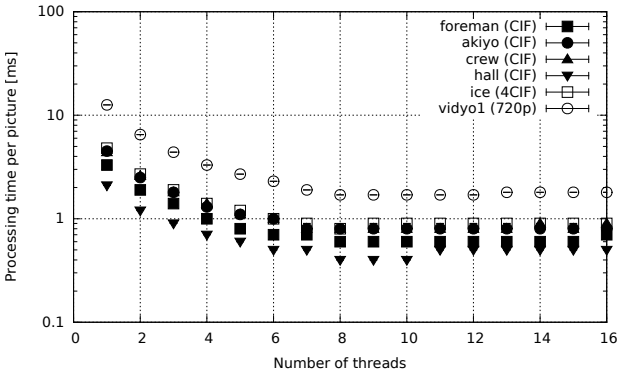

Fig. 17 Decoding times of different encoded sequences, i.e., encrypted files with signalling information: All sequences can be decoded in real-time

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.

Fast	**Sequence**	**Face detection time [%]**	**Encryption time [%]**	**Signalling time [%]**	**Miscellaneous time [%]**
	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

Fast			
Sequence	**Overhead [%]**	**Precision [%]**	**Recall [%]**
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/`

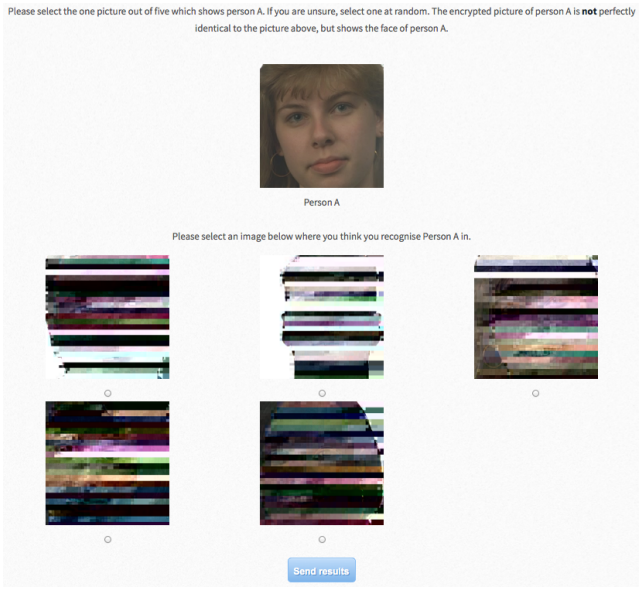

Fig. 18 A (cropped) screenshot of our survey system: The user has to select the encrypted picture which depicts the same person whose face is displayed unencrypted. This example shows DC-only encryption

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:

$$rr_i = \frac{c_i}{d_i} \tag{4}$$

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

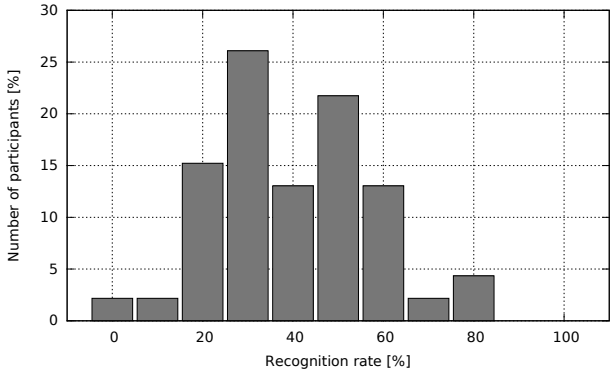

Fig. 19 Distribution of recognition rates for the proposed encryption approach: Most participants achieve recognition rates between 30 and 50%

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)