ENCRYPTION OF WAVELET-CODED IMAGERY USING RANDOM PERMUTATIONS*

Roland Norcen and Andreas Uhl

Department of Scientific Computing, Salzburg University, AUSTRIA

ABSTRACT

In this paper we investigate the impact of using random coefficient permutation to provide confidentiality in wavelet-based still image compression pipelines. We compare the changes in compression performance of JPEG 2000 and SPIHT-based schemes and discuss key management scenarios. The results also have interesting implications with respect to the significance of the zerotree hypothesis.

1. INTRODUCTION

Multimedia data security is a very important topic nowadays. Over the last years a number of different encryption schemes for these data types have been proposed, since methods to provide confidentiality need to be specifically designed to protect multimedia content and fulfil the security requirements for a particular multimedia application.

The so called naive method - the most secure one - is to take the multimedia bitstream and encrypt this stream with the aid of a cryptographically strong cipher like AES. Here, the encryption is performed after the compression stage and due to the complexity of the involved encryption algorithm, such a scheme inherently adds significant latency which often conflicts with real-time constraints. Since runtime performance is often very critical in video encoding and decoding, more efficient methods have been proposed. Such systems - often denoted as "selective" or "soft" encryption systems - usually trade off runtime performance for security, and are therefore - in terms of security - somewhat weaker than the naive method. Whereas selective encryption approaches exploit application specific data structures to create more efficient encryption systems (e.g. encryption of I-encoded blocks in MPEG, packet data of selected layers in JPEG 2000) using secure but slow "classical" ciphers, soft encryption systems employ weaker encryption systems to accelerate the processing speed. Among other approaches, permutations are an interesting class of algorithms to provide confidentiality to multimedia data in the sense of soft encryption. Permutations are known to be extremely fast and turn out to be resistant to bit errors during transmission [1] since errors do not propagate to a large amount of data as it is the case in stronger encryption systems. However, permutations are also known to be vulnerable against known plaintext attacks in case a permutation key is used more than once. In recent work [2] a technique has been proposed in the context of MPEG-4 IPMP which generates permutation keys on the fly by using parts of the video data not involved in the encryption process. This approach resolves the key management problems if the permutation keys have to be exchanged frequently to withstand the abovementioned known-plaintext attacks at the cost of higher computational demand.

In the context of DCT-based compression systems, coefficient permutation has been proposed as a means to provide confidentiality within the compression pipeline [3]. In particular, permutation lists replace the zig-zag order used within the DCT block processing in a random manner. Since such a permutation mapping has the same computational complexity as the original zig-zag scan, the encryption and decryption add very little time to the video coding process. An additional desirable property is that this encryption technique automatically ensures bitstream compliance of the encrypted data.

Two major problems have been identified associated with this approach (besides the above mentioned vulnerability against known-plaintext attacks): since the distribution of the DCT coefficents with respect to their size is invariant among many typical frames to a large extent, the original ordering of the larger coefficents (corresponding to low-frequency information) can be recovered without large effort [4, 5, 6]. Therefore, the scheme is vulnerable to a ciphertext-only attack (which is the weakest attack of all). Additionally, the overall compression rate is decreased significantly by up to 46% [5, 7], which is due to the fact that the MPEG and JPEG VLC tables are optimized with respect to the original zig-zag scan order.

In the context of wavelet-based compression schemes, random permutation lists have been proposed as well to secure wavelet-subbands [8, 9]. One obvious advantage as compared to the DCT scenario is that the distribution of wavelet coefficients is image dependent and therefore the vulnerability against ciphertext-only attacks does not occur. Also, it is claimed [8] that contrasting to the DCT case the observed drop in compression performance is about 2%

* This work has been partially supported by the Austrian Science Fund (project FWF-15170).

only.

In this paper we follow the ideas to use random permutation lists to secure wavelet-coded visual data. We show that a system based on randomly permuting wavelet-subbands incorporated in the JPEG2000 or the SPIHT coder generally delivers much worse results in terms of compression performance as given in [8]. The comparison of JPEG2000 and SPIHT in this context provides intersting insights with respect to the correctness of the zerotree hypothesis. Additionally, we discuss two scenarios for the key management in some detail and we investigate the effect of including key material into the bitstream on the overall compression performance.

The remaining part of this paper is organized as follows: First, the JPEG2000 and SPIHT coding systems are briefely reviewed, thereafter, the encryption scheme based on random permutation lists is explained. Then, we list and analyze the obtained performance results. Finally, concluding remarks are given.

2. JPEG2000 VERSUS SPIHT

Both the JPEG2000 image coding standard and the SPIHT codec use the wavelet transform for energy compaction.

JPEG2000 operates on independent, non-overlapping blocks of wavelet coefficients whose bit-planes are coded in several passes to create an embedded, scalable bitstream. The scheme only exploits intra-subband correlation via context-adaptive arithmetic coding since the strictly independent block coding strategy precludes structures across subbands or even code-blocks.

The SPIHT coder uses zerotrees for the coding of the wavelet coefficients. The wavelet coefficients are arranged in trees spaning all wavelet subbands, and all pixels within one tree are likely to have similar properties. Due to the statistical properties of wavelet-transformed image data, it is commonly believed that many zerotrees (i.e. trees containing insignificant coefficients) can be found in the wavelet domain. Since each zerotree is encoded with a single symbol, zerotree coding can usually efficiently encode the coefficients of a wavelet transform.

However, the so-called zerotree-hypothesis (i.e. that insignificant coefficents tend to have insignificant children coefficients) being the theoretical basis for the efficiency of zerotree coders, is widely disputed [10]. It is particulary interesting that a codec like JPEG2000 which does not make use of zerotrees can achieve performance results often exceeding those of zerotree-based systems. Applying random permutations to the wavelet subbands independently obviously destroys the zerotree structures (if present). Therefore, compression performance of SPIHT should suffer more from these permutations as compared to JPEG2000 provided the zerotree hypothesis is true.

3. ENCRYPTION USING RANDOM PERMUTATION OF WAVELET-SUBBANDS

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

In case permutation keys have to be transmitted along with the compressed image data (and not generated on the fly as propoposed in [2]) the used keys have to be protected and therefore be encrypted with a standard encryption scheme like AES. For example, the key data itself can be inserted conveniently into the JPEG2000 bitstream taking advantage of the so-called termination markers.

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

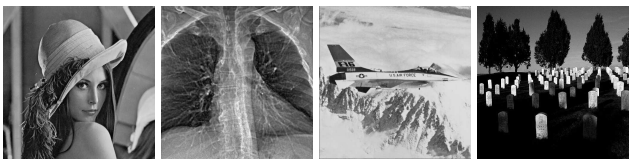
1. full key scenario: A wavelet subband with n pixels is permuted with a "full" key of length n.

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

Instead of using randomly chosen permutation keys which need a significant amount of additional storage capacity, a master key together with a key generation algorithm as proposed in [8] can be used to save memory. The usage of a master key with a key generation algorithm can be somewhat weaker in terms of security as compared to using randomly selected keys, however, it turns out that this approach is mandatory to limit the loss in compression efficiency.

4. RESULTS

We use the two considered lossy image compression schemes with the default decomposition depth. The testimages are the lena, lunge, plane, and the graves image each at a resolution of 512×512 pixels (see figure 1).



(a) lena (b) lunge (c) plane (d) graves

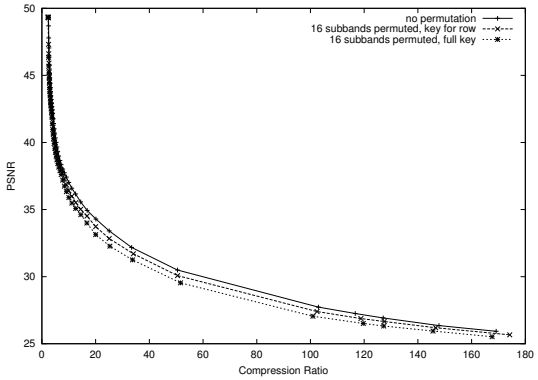
Fig. 1. testimages

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

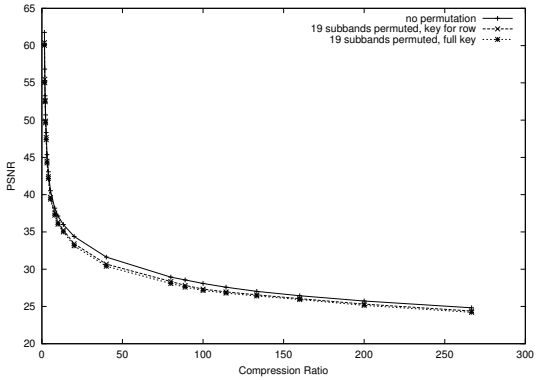
In order to evaluate the compression performance, each testimage is encoded with both considered coding algorithms. Within the coding pipeline, the coefficients of the wavelet subbands are permuted before the quantization stage. Thereafter, the encrypted and compressed file is decoded, the corresponding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

The rate-distortion performance for the lena image is shown in figure 2. Note that key material is not included in the bitstream for this comparison. The "no permutation" curve denotes the rate-distortion performance of the original JPEG2000 and SPIHT algorithm. The curve "key for row" shows the performance when all subbands are permuted and per subband only one key is repeatedly used for the rows of the subband, and the curve "full key" shows the rate-distortion performance, when each subband gets a full key for the permutation of the whole subband.

In the case of JPEG2000 a max. drop of compression performance of 26% using full keys can be observed when coding the lena image (meaning that the encrypted file needs to be increased by 26% in order to achieve the PSNR of the original encoded image; we denote this to be a loss of compression performance of 26%), and only a 13% decrease can be observed when using one key for each row of a subband. Table 1 lists the observed maximal drops of compression performance for all tested images. Overall, the "key for row" method degrades compression performance much less, and the zerotree-based SPIHT algorithm produces similar results in the "full key" scenario as compared to JPEG2000. The latter is surprising, since permuting the coefficients of wavelet subbands obviously destroys the zerotree structures which should be essential to the performance of such an algorithm. Considering the zerotree hypothesis, a stronger degradation would have been expected in the SPIHT case which raises doubts about the correctness of this important assumption. However, in the "key for row" scenario the JPEG2000 compression performance is much less decreased as compared to SPIHT. This is due to the spatial



(a) JPEG2000



(b) SPIHT

Fig. 2. compression performance, lena512

correlation which is preserved among pixels in neighboring areas (since the same permutation is used for adjacent rows), which allows the context-based arithmetic coding engine of JPEG2000 to produce better results as compared to the inter-subband zerotree coding of the SPIHT codec.

Compared to the good results shown in [8] both considered schemes produce a significant performance loss (up to 27%), and we do not even include the key data in the final compressed file yet. The compression scheme in the referenced work is not a very sophisticated one and SPIHT as well as JPEG2000 depend much more on pixel context as simple scalar quantization based schemes.

In the following we focus on the amount of key data to be included in the bitstream additionally. In the worst case, a permutation key of length $n, n = 2^x$ needs $n \times x$ bits to be stored. Suppose a 512×512 pixels image at 8 bits per pixel encoded at 5 decomposition levels resulting in 16 subbands to be permuted with randomly selected keys. The uncompressed size of the raw data of the considered image is about 256 kbytes. The full keys for the 16 wavelet subbands need

	full key	key for row
JPEG2000, lena512	26%	13%
JPEG2000, lunge512	8%	4%
JPEG2000, plane512	27%	14%
JPEG2000, graves512	21%	5%
SPIHT, lena512	26%	21%
SPIHT, lunge512	9%	9%
SPIHT, plane512	23%	21%
SPIHT, graves512	22%	15%

Table 1. JPEG2000/SPIHT: all subbands permuted, max. observed file size increase at a medium compression rate ranging from 25 up to 45

48 bytes (subband consisting of 8×8 pixels) up to 131072 bytes (subband with 256×256 pixels) in the worst case. Approximately 490 kbytes of memory are needed in this case. This amount of memory has to be added to the compressed file size which results in a file size increase of up to 290%. Additionally, these key data need to be encrypted which reduces the entire approach to complete absurdity.

The "key for row" method can save key data memory with the drawback of losing security since more rows of a subband are permuted with the same key. Considering the example used in our experiments (only one key is used for the rows of each subband, and all subbands are permuted), approximately 1.3 kbyte are needed to store the randomly chosen keys (for all subbands) in the worst case. In this situation we still result an increase of the file of 12% if the image is coded at a ratio of 25 which hardly is acceptable.

In order to increase the overall performance, a master key with a key generation algorithm has to be used. Then, only the master key needs to be stored and encrypted in the compressed file, all used keys are generated with a well selected key generation algorithm. The additional space needed to store the master key is minimal, but the usage of a key generation algorithm might be somewhat weaker as already mentioned. The performance loss shown in table 1 has to be taken into account anyway.

5. CONCLUSION

We have observed that the random permutation of coefficients of wavelet subbands degrades the compression performance significantly when used within the JPEG2000 or the SPIHT coding pipelines. In order to limit the compression loss to the observed extent, the key material signalled in the respective bitstreams has to be restricted to a minimum (e.g. by using a master key and a key generation procedure). Interestingly both considered compression schemes behave similarly when permutations are employed which implies that the zerotree hypothesis is of minor importance for the coding efficiency than generally believed.

6. REFERENCES

[1] Ali Saman Tosun and Wu chi Feng, "On error preserving encryption algorithms for wireless video transmission," in *Proceedings of the ninth ACM Multimedia Conference 2001*, Ottawa, Canada, Oct. 2001, pp. 302–307.

[2] Jiangtao Wen, Mike Severa, Wenjun Zeng, Max Luttrell, and Weiyin Jin, "A format-compliant configurable encryption framework for access control of video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 545–557, June 2002.

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

[4] Klara Nahrstedt and Lintian Qiao, "Is MPEG Encryption by Using Random List Instead of ZigZag Order Secure ?," in *Proceedings of Int. Sym. Consumer Electronics (ISCE97)*, 1997, pp. 226–229.

[5] Lintian Qiao and Klara Nahrstedt, "Comparison of MPEG encryption algorithms," *International Journal on Computers and Graphics (Special Issue on Data Security in Image Communication and Networks)*, vol. 22, no. 3, pp. 437–444, 1998.

[6] T. Uehara and R. Safavi-Naini, "Chosen DCT coefficients attack on MPEG encryption schemes," in *Proceedings of the 2000 IEEE Pacific Rim Conference on Multimedia*, Sydney, Dec. 2000, pp. 316–319, IEEE Signal Processing Society.

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

[8] T. Uehara, R. Safavi-Naini, and P. Ogunbona, "Securing wavelet compression with random permutations," in *Proceedings of the 2000 IEEE Pacific Rim Conference on Multimedia*, Sydney, Dec. 2000, pp. 332–335, IEEE Signal Processing Society.

[9] Wenjun Zeng and Shawmin Lei, "Efficient frequency domain video scrambling for content access control," in *Proceedings of the seventh ACM International Multimedia Conference 1999*, Orlando, FL, USA, Nov. 1999, pp. 285–293.

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